UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Ingeniería en Informática
Proyecto Fin de Carrera

"Manual para la creación de videojuegos mediante el motor Unreal Development Kit"

Autor: Julián Marchante de la Dueña
Tutor: Juan Peralta Donate

A todos los míos

AGRADECIMIENTOS

Con la realización de este proyecto se pone punto y final a todo un largo camino que ha sido la carrera de Ingeniería Informática.

Mediante estas líneas querría agradecer todo el apoyo que he tenido para conseguir finalizar.

Primeramente quiero dar las gracias a mis padres, Julián y Juani, por todo su apoyo incondicional en todas y cada una de las decisiones tomadas, tanto buenas como malas. Siempre los he tenido a mi lado para aconsejarme y ayudarme.

A mis hernamos, Jessica y Sergio, por aguantarme siempre, y sobre todo al jugón de mi hermano, por su tiempo dedicado para probar todas las versiones del juego; y sus ideas y sugerencias para mejorarlo.

A Juan Peralta, mi tutor del proyecto, por todos sus consejos y tiempo invertido en ayudarme a terminar con éxito este proyecto fin de carrera.

Y muy especialmente, a Iman, mi novia, mi compañera en este camino. Agradecerle el haberme soportado en las buenas y en las malas situaciones. Por su ánimo en los peores momentos, su paciencia conmigo y su ayuda en el proyecto, en éste y en todos los realizados durante la carrera. Sin ella el camino habría sido imposible.

CONTENIDO

1. INTRODUCCIÓN. 22
1.1. Descripción de los capítulos del documento. 22
1.2. Objetivos del proyecto. 23
2. ESTADO DE LA CUESTIÓN. 24
2.1. *"Game Engine"* – Motor de videojuegos. 24
2.1.1. ¿Qué son? 24
2.1.2. Historia de los motores de videojuegos. 25
2.1.3. Tipos de motores. 27
2.1.4. Lista de motores. 28
2.2. "*Unreal Engine*" – Motor Unreal. 30
2.2.1. Componentes. 31
2.2.2. Historia del motor Unreal Engine. 34
3. UNREAL EDITOR. 42
3.1. Menú de la aplicación. 42
3.2. Barra de herramientas de Unreal Editor. 42
3.3. Vistas del editor. 44
3.3.1. Funcionalidades en las pantallas del editor. 44
3.3.2. Mover la vista en las pantallas. 45
3.4. Comienzo de un nivel. 46
3.4.1. Creando formas. 48
3.4.2. Modificar la figura de la forma/modelo. 55
3.4.3. Añadir la forma/modelo al nivel. 61
3.5. Crear terreno. 66
4. CONTENT BROWSER 69
4.1. Importar elemento. 74
4.2. Crear nuevo elemento. 75
4.3. Información accesible. 77
5. UNREAL KISMET. 84
5.1. Realizar movimientos en la plantilla de la herramienta. 84
5.2. Agregar elementos. 85
5.2.1. Eventos. 85
5.2.2. Acciones. 90
5.2.3. Variables. 107
5.2.4. Condiciones. 108
5.2.5. Elemento Matinee. 113

5.2.6. Etiquetas. 113
5.2.7. Agrupación de elementos. 114
5.2.8. Crear evento con objeto seleccionado. 114
5.3. Enlace entre elementos. 115
6. UNREAL MATINEE. 117
6.1. Creación de grupos. 118
6.2. Secuencia de tiempo. 121
6.3. Elemento Matinee en la herramienta Unreal Kismet. 123
7. ELEMENTOS BÁSICOS DE UN NIVEL. 125
7.1. Aparición del personaje principal. 125
7.2. Iluminación. 127
7.3. Acoplar objetos al nivel desde Content Browser. 136
7.4. Sonido y mensajes de información. 143
7.5. Modificar inventario de armas del personaje principal. 147
7.6. Modificar el tipo de *"colisión"* de los objetos en el nivel. 153
7.7. Mostrar información en pantalla – UIScene. 155
7.8. Volúmenes. 161
7.9. Lanzar el nivel/mapa creado. 173
8. TUTORIALES. 180
8.1. Tutorial 1: Crear puerta. 180
8.2. Tutorial 2: Apagar/encender luces con pulsador. 192
8.3. Tutorial 3: Acoplar linterna al personaje principal. 198
8.4. Tutorial 4: Poner la cámara en 3ª persona. 203
8.5. Tutorial 5: Crear movimiento de cámara. 211
8.6. Tutorial 6: Desaparición de objeto al tocarlo. 218
8.7. Tutorial 7: Teletransportar al personaje principal. 222
8.8. Tutorial 8: Mostrar animación de "Skeletal Mesh". 228
8.9. Tutorial 9: Agregar personajes distintos al principal. 234
8.10. Tutorial 10: Controlar cuándo un personaje muere. 238
8.11. Tutorial 11: Personaje se traslade de un punto a otro. 242
8.12. Tutorial 12: Personaje dispare al jugador cuando lo "ve". 248
8.13. Tutorial 13: Mostrar valor de variable en pantalla. 254
8.14. Tutorial 14: Poner viento. 256
8.15. Tutorial 15: Acoplar "lanzador" de personaje. 258
8.16. Tutorial 16: Poner cielo en el nivel. 261
8.17. Tutorial 17: Cambiar material de un "Static Mesh". 262
8.18. Tutorial 18: Crear material "espejo". 271

8.19. Tutorial 19: Crear material “lluvia”. 281
8.20. Tutorial 20: Acoplar vídeo como material. 288
8.21. Tutorial 21: Cambiar tamaño del personaje. 296
8.22. Tutorial 22: Unir dos niveles – Streaming. 297
8.23. Tutorial 23: Finalizar juego. 301
8.24. Tutorial 24: Cambiar/modificar la pantalla de “Bienvenida” 303
8.25. Tutorial 25: Crear instalador del juego completo (.EXE). 304
9. CREAR NUEVO PERSONAJE. 307
9.1. Creación del “Skeletal Mesh”. 307
9.2. Exportar “Skeletal Mesh” de 3D Studio Max a UDK. 313
9.3. Importar “Skeletal Mesh” a UDK. 317
9.4. Creación de elementos básicos a partir del “Skeletal Mesh”. 319
9.5. Código a insertar/modificar para utilizar el nuevo personaje. 331
9.6. Compilación del código 334
10. CASO PRÁCTICO: JUEGO “CrossPlane”. 336
10.1. Nivel interior. 336
10.2. Nivel exterior. 345
11. GESTIÓN DEL PROYECTO. 355
11.1. Diagramas de Gantt. 355
11.1.1. Planificación estimada. 355
11.1.2. Planificación real. 357
11.1.3. Análisis de planificación. 359
11.2. Presupuesto. 360
11.3. Comercialización del proyecto. 363
12. CONCLUSIONES. 364
13. LÍNEAS FUTURAS. 365
14. REFERENCIAS. 366
15. ANEXOS. 371
15.1. Anexo I: Glosario de términos. 371
15.2. Anexo II: Guía del juego “CrossPlane”. 373

ÍNDICE DE TABLAS

Tabla 1. Motores con licencia Open Source..... 29
Tabla 2. Motores con licencia Freeware. 29
Tabla 3. Motores con licencia Propietario..... 30
Tabla 4. Presupuesto detallado de personal..... 360
Tabla 5. Presupuesto detallado de equipos..... 361
Tabla 6. Presupuesto detallado de costes directos..... 362
Tabla 7. Presupuesto total del proyecto..... 363
Tabla 8. Glosario de términos. 372

ÍNDICE DE IMÁGENES

Imagen 1: Entorno Unreal Engine 30
Imagen 2: Componentes “Unreal Engine”. 31
Imagen 3: Motor gráfico “Unreal Engine” 32
Imagen 4: Unreal Script 34
Imagen 5: Carátula juego “Unreal” 35
Imagen 6: Carátula juego “Unreal Tournament” 35
Imagen 7: Página web “The Unreal Developer Network” 36
Imagen 8: Carátula juego “Unreal Tournament 2003” 37
Imagen 9: Carátula juego “Unreal Championship” 37
Imagen 10: Carátula juego “Unreal Tournament 2004” 38
Imagen 11: Carátula juego “Unreal Championship 2” 39
Imagen 12: Carátula juego “Gears of War” 40
Imagen 13: Interfaz “Unreal Editor”. 42
Imagen 14: Menú de la aplicación 42
Imagen 15: Barra de herramientas “Unreal Editor” 42
Imagen 16: Vistas del editor 44
Imagen 17: Herramientas de vistas del editor 44
Imagen 18: Menú de modos de creación de un nivel. 47
Imagen 19: Ejemplo “additive”. 47
Imagen 20: Ejemplo “Substractive”. 47
Imagen 21: Forma/modelo. 48
Imagen 22: Traslation Mode. 49
Imagen 23: Localización de traslado objeto en 2 ejes. 50
Imagen 24: Traslation Mode insertando valores. 50
Imagen 25: Scaling Mode uniforme. 51
Imagen 26: Scaling Mode no uniforme. 52
Imagen 27: Localización Scaling Mode en 2 ejes. 53
Imagen 28: Scaling Mode insertando valores. 53
Imagen 29: Rotation Mode 54
Imagen 30: Rotation Mode insertando valores. 54
Imagen 31: Formas posibles del modelo. 55
Imagen 32: Propiedades forma hexágono. 56
Imagen 33: Propiedades forma cono. 56
Imagen 34: Propiedades forma cilindro. 57
Imagen 35: Propiedades forma plano. 58

Imagen 36: Propiedades forma esfera. 58
Imagen 37: Propiedades forma escaleras lineales. 59
Imagen 38: Propiedades forma escaleras curvadas. 60
Imagen 39: Propiedades forma escaleras en espiral. 60
Imagen 40: Propiedades forma Volumetric. 61
Imagen 41: Herramientas para añadir la forma al nivel. 62
Imagen 42: Ejemplo material por defecto. 63
Imagen 43: Menú para aplicar material. 64
Imagen 44: Surface Properties. 65
Imagen 45: Opciones “Alignment” del material en objeto. 66
Imagen 46: Interfaz Unreal TerrainEdit. 67
Imagen 47: Interfaz “Content Browser”. 69
Imagen 48: Menú de la opción “View”. 70
Imagen 49: Menú desplegado de “Browser Windows”. 71
Imagen 50: Jerar quía paquetes en “Content Browser”. 72
Imagen 51: Elementos mostrados en “Content Browser”. 72
Imagen 52: Menú paquete. 73
Imagen 53: Ejemplo submenú elemento Texture2D. 74
Imagen 54: Opciones creación/importación elementos. 75
Imagen 55: Opciones de creación/importación elementos al “Content Browser”. 76
Imagen 56: Menú creación nuevo elemento. 76
Imagen 57: Pantalla creación nuevo elemento. 77
Imagen 58: Actor Classes. 78
Imagen 59: Opción de agregar elemento al nivel. 79
Imagen 60: Levels 79
Imagen 61: Scene. 80
Imagen 62: Primitive Stats. 81
Imagen 63: Building Stats. 81
Imagen 64: Texture Stats. 82
Imagen 65: Log. 82
Imagen 66: Start Page. 83
Imagen 67: Interfaz Kismet. 84
Imagen 68: Menú añadir elemento. 85
Imagen 69: Menú añadir evento. 86
Imagen 70: Ejemplo evento. 86
Imagen 71: Elemento “Take Damage”. 87
Imagen 72: Elemento “Destroyed”. 87

Imagen 73: Elemento “Level Loaded”. 88
Imagen 74: Elemento “Level Startup”. 88
Imagen 75: Elemento “Death”. 89
Imagen 76: Elemento “Get Inventory”. 89
Imagen 77: Elemento “Player Spawned”. 89
Imagen 78: Elemento “Remote Event”. 90
Imagen 79: Menú insertar acción. 91
Imagen 80: Elemento “Actor Factory”. 92
Imagen 81: Elemento “Destroy”. 92
Imagen 82: Elemento “Get Distance”. 92
Imagen 83: Elemento “Modify Health”. 93
Imagen 84: Elemento “Teleport”. 93
Imagen 85: Elemento “Start Firing At”. 94
Imagen 86: Elemento “Stop Firing”. 94
Imagen 87: Elemento “Look At”. 94
Imagen 88: Elemento “Set Camera Target”. 95
Imagen 89: Elemento “Activate Remote Event”. 95
Imagen 90: Ejemplo asociación correcta con “Remote Event”. 95
Imagen 91: Elemento “Attach To Event”. 96
Imagen 92: Elemento “Add Float”. 96
Imagen 93: Elemento “Add Int”. 97
Imagen 94: Elemento “Cast To Float”. 97
Imagen 95: Elemento “Cast To Int”. 97
Imagen 96: Elemento “Divide Float”. 98
Imagen 97: Elemento “Divide Int”. 98
Imagen 98: Elemento “Multiply Float”. 98
Imagen 99: Elemento “Multiply Int”. 99
Imagen 100: Elemento “Subtract Float”. 99
Imagen 101: Elemento “Subtract Int”. 99
Imagen 102: Elemento “Console Command”. 100
Imagen 103: Elemento “Delay”. 100
Imagen 104: Elemento “Log”. 101
Imagen 105: Elemento “Timer”. 101
Imagen 106: Elemento “Trace”. 101
Imagen 107: Elemento “Bool”. 102
Imagen 108: Elemento “Float”. 102
Imagen 109: Elemento “Int”. 102

Imagen 110: Elemento “Object”. 103
Imagen 111: Elemento “String”. 103
Imagen 112: Elemento “Vector”. 103
Imagen 113: Elemento “Random”. 104
Imagen 114: Elemento “Toggle”. 104
Imagen 115: Elemento “Toggle Hidden”. 105
Imagen 116: Elemento “Toggle HUD”. 105
Imagen 117: Elemento “Open Scene”. 106
Imagen 118: Elemento “Close Scene”. 106
Imagen 119: Elemento “Play Announcement”. 107
Imagen 120: Menú insertar variable. 107
Imagen 121: Tipos de variables. 108
Imagen 122: Menú insertar condición. 108
Imagen 123: Ejemplo elemento de comparación. 109
Imagen 124: Elemento “Compare Bool”. 109
Imagen 125: Elemento “Compare Float”. 110
Imagen 126: Elemento “Compare Int”. 110
Imagen 127: Elemento “Compare Objects”. 111
Imagen 128: Elemento “Float Counter”. 111
Imagen 129: Elemento “Int Counter”. 112
Imagen 130: Elemento “Has Inventory”. 112
Imagen 131: Elemento “Is Alive”. 112
Imagen 132: Elemento “Matinee”. 113
Imagen 133: Elemento “Etiqueta”. 113
Imagen 134: Elemento “Agrupación de elementos”. 114
Imagen 135: Entradas y salidas de elemento. 115
Imagen 136: Entrada de variables de elemento. 116
Imagen 137: Ejemplo enlace entre elementos. 116
Imagen 138: Interfaz “Unreal Matinee”. 117
Imagen 139: Sección de grupos de interfaz “Matinee”. 117
Imagen 140: Menú creación grupos. 118
Imagen 141: Ejemplo de creación de grupo. 118
Imagen 142: Menú creación track en grupo. 119
Imagen 143: Ejemplo creación tracks en grupo. 120
Imagen 144: Sección “Curve Editor” de “Matinee”. 120
Imagen 145: Sección de secuencia de tiempo en “Matinee”. 121
Imagen 146: Inicio y fin de secuencia de tiempo. 121

Imagen 147: Línea de tiempo. 122
Imagen 148: Ejemplo de creación de punto de control. 123
Imagen 149: Ejemplo elemento “Matinee”. 124
Imagen 150: Player Start. 125
Imagen 151: Menú botón secundario en editor. 126
Imagen 152: Menú “Add Actor”. 126
Imagen 153: Insertar “PlayerStart”. 127
Imagen 154: Menú secundario elemento de iluminación. 128
Imagen 155: Point Light. 130
Imagen 156: Point Light “Toggleable”. 130
Imagen 157: Añadir punto de luz. 131
Imagen 158: Añadir punto de luz “Toggleable”. 132
Imagen 159: Foco. 132
Imagen 160: Foco “Toggleable”. 133
Imagen 161: Añadir foco. 134
Imagen 162: Añadir foco “Toggleable”. 135
Imagen 163: Luz ambiental. 135
Imagen 164: Añadir luz ambiental. 136
Imagen 165: Traslation Mode. 137
Imagen 166: Traslation Mode en 2 ejes. 138
Imagen 167: Traslation Mode insertando valores. 138
Imagen 168: Scaling Mode uniforme. 139
Imagen 169: Scaling Mode no uniforme. 140
Imagen 170: Scaling Mode no uniforme en 2 ejes a la vez. 141
Imagen 171: Scaling Mode insertando valores. 141
Imagen 172: Rotation Mode. 142
Imagen 173: Rotation Mode insertando valores. 142
Imagen 174: Elemento “Play Announcement”. 143
Imagen 175: Ejemplo secuencia de sonidos. 144
Imagen 176: Sección “AmbientSound” en herramienta “Actor Classes”. 144
Imagen 177: “Ambient Sound Movable”. 145
Imagen 178: “Ambient Sound Non Loop”. 145
Imagen 179: “Ambient Sound Simple Toggleable”. 146
Imagen 180: Menú “World Properties”. 147
Imagen 181: Propiedades del mundo. 148
Imagen 182: Código inventario. 149
Imagen 183: Ejemplo factoría de arma en nivel. 149

Imagen 184: Sección “UTWeaponPickupFactory” en herramienta “Actor Classes”. . 150
Imagen 185: Propiedades “Weapon Pickup Factory”. 151
Imagen 186: Tipos de armas. 151
Imagen 187: “Give Inventory”. 152
Imagen 188: Propiedades “Give Inventory”. 152
Imagen 189: Asociación del “Target” de “Give Inventory”. 153
Imagen 190: Propiedades “Static Mesh”. 154
Imagen 191: Tipos de colisión. 154
Imagen 192: Interfaz editor “Unreal UIScene”. 155
Imagen 193: Sección gráfica de “Unreal UIScene Editor”. 156
Imagen 194: Herramientas de elementos a añadir. 156
Imagen 195: Ejemplos elementos añadidos en “UIScene”. 157
Imagen 196: Propiedades “UIScene”. 157
Imagen 197: Menú de selección “UnrealKismet Editor”. 158
Imagen 198: Etiqueta “Pause Game While Active”. 159
Imagen 199: Etiqueta “Display Cursor”. 159
Imagen 200: Etiqueta “Scene Input Mode”. 160
Imagen 201: “Open Scene”. 160
Imagen 202: Propiedades “Open Scene”. 160
Imagen 203: “Close Scene”. 161
Imagen 204: Tipos de volúmenes. 162
Imagen 205: Propiedades “PhysicsVolume”. 163
Imagen 206: Propiedades “Gravity Volume”. 165
Imagen 207: Propiedades “Kill Volume”. 166
Imagen 208: Propiedades “Water Volume”. 167
Imagen 209: Propiedades “Space Volume”. 169
Imagen 210: Propiedades “Ladder Volume”. 170
Imagen 211: Propiedades “Trigger Volume”. 171
Imagen 212: Propiedades “Post Process Volume”. 172
Imagen 213: Menú “Build”. 173
Imagen 214: Menú compilación iluminación. 173
Imagen 215: Selección “AI Paths”. 174
Imagen 216: Selección compilación completa. 174
Imagen 217: Interfaz “Swarm Agent”. 175
Imagen 218: Ejemplo resultados compilación. 175
Imagen 219: Ejemplo resultados compilación de iluminación. 176
Imagen 220: Localización “Play from Here”. 177

Imagen 221: Situación “Play from Here in Viewport”. 178
Imagen 222: Situación “Play on PC”. 179
Imagen 223: Tutorial 1-Suelo..... 180
Imagen 224: Tutorial 1-Pared..... 181
Imagen 225: Tutorial 1-Hueco puerta. 181
Imagen 226: Tutorial 1-Situación objeto-puerta en “Content Browser”..... 182
Imagen 227: Tutorial 1-Menú agregar puerta. 182
Imagen 228: Tutorial 1-Situación puerta en el hueco. 183
Imagen 229: Tutorial 1-Lanzador para dinámica de puerta. 183
Imagen 230: Tutorial 1-Zona interacción lanzador. 184
Imagen 231: Tutorial 1-Situación trigger en puerta. 184
Imagen 232: Tutorial 1-Menú inserción trigger de tipo “Touch”. 185
Imagen 233: Tutorial 1-Trigger añadido en herramienta “Kismet”. 185
Imagen 234: Tutorial 1-Creación elemento Matinee. 186
Imagen 235: Tutorial 1-Herramienta “Matinee”. 186
Imagen 236: Tutorial 1-Creación grupo vacío con nombre “puerta”..... 187
Imagen 237: Tutorial 1-Track de movimiento añadido al grupo. 187
Imagen 238: Tutorial 1-Dinámica de movimiento de puerta guadado..... 188
Imagen 239: Tutorial 1-Enlace elemento Trigger y elemento Matinee. 189
Imagen 240: Tutorial 1-Etiqueta “Max Trigger Count” de elemento Trigger. 189
Imagen 241: Tutorial 1-Menú lanzador de tipo “Used”. 190
Imagen 242: Tutorial 1-Trigger añadido en herramienta “Kismet”. 190
Imagen 243: Tutorial 1-Etiqueta “Aim To Interact” de trigger tipo “Used”..... 191
Imagen 244: Tutorial 1-Apertura y cierre de puerta automáticamente. 191
Imagen 245: Tutorial 1-Movimiento contínuo de apertura y cierre de puerta. 192
Imagen 246: Tutorial 2-Localización “PointLightToggleable” en “Actor Classes”.... 193
Imagen 247: Tutorial 2-Situación luces en nivel..... 193
Imagen 248: Tutorial 2-Situación triggers junto a luces. 194
Imagen 249: Tutorial 2-Zonas de interacción de cada trigger..... 195
Imagen 250: Tutorial 2-Creación eventos en “Kismet” partiendo de los trigger..... 195
Imagen 251: Tutorial 2-Creación elementos “Toggle” de cada trigger. 196
Imagen 252: Tutorial 2-Variables de las luces. 197
Imagen 253: Tutorial 2-Asociación variables luces con “Toggle”. 197
Imagen 254: Tutorial 3-Situación “PlayerStart” en nivel. 198
Imagen 255: Tutorial 3-Localización “SpotLightMoveable” en “Actor Classes”..... 199
Imagen 256: Tutorial 3-Situación punto de luz junto “PlayerStart”. 199
Imagen 257: Tutorial 3-Creación elemento “Player Spanwed” en “Kismet”. 200

Imagen 258: Tutorial 3-Creación elemento “Attach to Actor”. 200
Imagen 259: Tutorial 3-Asociación variable de la luz. 201
Imagen 260: Tutorial 3-Asociación variable del jugador. 202
Imagen 261: Tutorial 3-Etiquetas de posición relativa en “Attach to Actor”. 202
Imagen 262: Tutorial 3-Etiquetas de rotación relativa en “Attach to Actor”. 203
Imagen 263: Tutorial 4-Situación del “PlayerStart”. 204
Imagen 264: Tutorial 4-Localización “Camera Actor” en “Actor Classes”. 204
Imagen 265: Tutorial 4-Situación “CameraActor” junto al “PlayerStart”. 205
Imagen 266: Tutorial 4-Localización “Snap View to Actor” en menú. 206
Imagen 267: Tutorial 4-Creación elemento “Level Loaded” en “Kismet”. 207
Imagen 268: Tutorial 4-Elemento “Attach to Actor”. 208
Imagen 269: Tutorial 4-Elemento “Set Camera Targer”. 208
Imagen 270: Tutorial 4-Variable representante de la cámara acoplada al nivel. 209
Imagen 271: Tutorial 4-Variable jugador. 210
Imagen 272: Tutorial 4-Ejemplo vista en tercera persona. 210
Imagen 273: Tutorial 4-Ejemplo vista aérea. 211
Imagen 274: Tutorial 5-Localización “Camera Actor” en “Actor Classes”. 212
Imagen 275: Tutorial 5-Situación de la cámara en el nivel. 212
Imagen 276: Tutorial 5-Localización “Snap View to Actor” en menú. 213
Imagen 277: Tutorial 5-Creación elemento “Matinee” en “Kismet”. 214
Imagen 278: Tutorial 5-Interfaz herramienta “Matinee”. 214
Imagen 279: Tutorial 5-Creación de grupo vacío llamado “cámara”. 215
Imagen 280: Tutorial 5-Track de movimiento en el grupo “cámara”. 215
Imagen 281: Tutorial 5-Secuencia de movimiento creada con la cámara. 216
Imagen 282: Tutorial 5-Creación del grupo “DirGroup” y su track “Director”. 217
Imagen 283: Tutorial 5-Asociación de la cámara con el “DirGroup”. 217
Imagen 284: Tutorial 5-Asociación “Level Loaded” con elemento “Matinee”. 218
Imagen 285: Tutorial 6-Acople de objeto en el nivel. 219
Imagen 286: Tutorial 6-Creación de evento de contacto con objeto en “Kismet”. 220
Imagen 287: Tutorial 6-Creación elemento “Toggle Hidden” en “Kismet”. 220
Imagen 288: Tutorial 6-Asociación elementos contacto y “Toggle Hidden”. 221
Imagen 289: Tutorial 6-Variable representativa del objeto del nivel. 222
Imagen 290: Tutorial 7-Nivel inicial. 223
Imagen 291: Tutorial 7-Creación eventos en “Kismet”desde los trigger. 224
Imagen 292: Tutorial 7-Creación elementos “Teleport” en cada trigger. 225
Imagen 293: Tutorial 7-Variables del jugador asociadas a “Teleport”. 226
Imagen 294: Tutorial 7-Localización de “Note” en “Actor Classes”. 226

Imagen 295: Tutorial 7-Situación del objeto “Note” en el nivel. 227
Imagen 296: Tutorial 7-Creación variables a partir objetos “Note”. 228
Imagen 297: Tutorial 8-Personaje acoplado al nivel. 229
Imagen 298: Tutorial 8-Acople de objeto arma al personaje. 230
Imagen 299: Tutorial 8-Elemento “Matinee” en “Kismet”. 230
Imagen 300: Tutorial 8-Creación de grupo vacío en “Matinee”. 231
Imagen 301: Tutorial 8-Propiedades grupo. 231
Imagen 302: Tutorial 8-Creación de track tipo “Anim Control” en el grupo. 232
Imagen 303: Tutorial 8-Asociación de animación en el track. 232
Imagen 304: Tutorial 8-Asociación elemento trigger con elemento “Matinee”. 233
Imagen 305: Tutorial 9-Elemento “Actor Factory” en “Kismet”. 234
Imagen 306: Tutorial 9-Tipos de “Actor”. 235
Imagen 307: Tutorial 9-Etiqueta “Factory” del elemento “Actor Factory”. 236
Imagen 308: Tutorial 9-Modificación etiquetas en “Actor Factory”. 236
Imagen 309: Tutorial 9-Situación elemento “PathNode” en nivel. 237
Imagen 310: Tutorial 9-Variable representativa del objeto “PathNode”. 237
Imagen 311: Tutorial 9-Asociación “Level Loaded” con “Actor Factory”. 238
Imagen 312: Tutorial 10-Variable de tipo “Object”asociada a “Actor Factory”. 239
Imagen 313: Tutorial 10-Creación evento “Death”. 240
Imagen 314: Tutorial 10-Acople elemento “Attach to Event” y asociarlo. 240
Imagen 315: Tutorial 10-Asociación “Attach to Event” con “Actor Factory”. 241
Imagen 316: Tutorial 10-Ejemplo de implementación a la muerte del “Actor”. 242
Imagen 317: Tutorial 11-Nivel inicial. 242
Imagen 318: Tutorial 11-Aparición nuevo personaje en nivel. 243
Imagen 319: Tutorial 11-Situación elementos “PathNode” en nivel. 244
Imagen 320: Tutorial 11-Variable “Object” asociada a elemento “Actor Factory”. 244
Imagen 321: Tutorial 11-Creación elemento “Move To Actor”. 245
Imagen 322: Tutorial 11-Asociación “Object” y “PathNode” con “Move to Actor”. . 246
Imagen 323: Tutorial 11-Creación elemento “Move To Actor”. 247
Imagen 324: Tutorial 11-Menú activación retardo. 247
Imagen 325: Tutorial 11-Realización de bucle de ejecución. 248
Imagen 326: Tutorial 12-Nivel inicial. 248
Imagen 327: Tutorial 12-Aparición nuevo personaje en nivel. 249
Imagen 328: Tutorial 12-Variable “Object”. 250
Imagen 329: Tutorial 12-Creación elemento “Trace”. 251
Imagen 330: Tutorial 12-Variable de jugador. 251
Imagen 331: Tutorial 12-Inserción elemento “Start Firing At” y asociación. 252

Imagen 332: Tutorial 12-Inserción elemento “Stop Firing” y asociación..... 253
Imagen 333: Tutorial 12-Bucle de ejecución. 253
Imagen 334: Tutorial 13-Elemento “Log”. 254
Imagen 335: Tutorial 13-Menú elemento “Log”..... 254
Imagen 336: Tutorial 13-Tipos de variables de entrada..... 255
Imagen 337: Tutorial 13-Ejemplo uso elemento “Log”..... 255
Imagen 338: Tutorial 14-Elemento viento. 256
Imagen 339: Tutorial 14-“WindDirectionalSource” en “Actor Classes”..... 256
Imagen 340: Tutorial 14-Propiedades de “WindDirectionalSource”..... 257
Imagen 341: Tutorial 14-Ejemplo de uso de viento..... 258
Imagen 342: Tutorial 15-Elemento lanzador..... 258
Imagen 343: Tutorial 15-Localización de “UTJumpPad” en “Actor Classes”. 259
Imagen 344: Tutorial 15-Elemento “PathNode”..... 259
Imagen 345: Tutorial 15-Propiedades lanzador “UTJumPad”..... 260
Imagen 346: Tutorial 15-Ejemplo de uso de lanzador. 261
Imagen 347: Tutorial 16-Ejemplo de acople de cielo al nivel. 261
Imagen 348: Tutorial 16-Ejemplo de dinámica realizada con el cielo..... 262
Imagen 349: Tutorial 17-Nivel inicial..... 263
Imagen 350: Tutorial 17-Elemento “Material Instance Actor”..... 263
Imagen 351: Tutorial 17-“MaterialInstanceActor” en “Actor Classes”..... 264
Imagen 352: Tutorial 17-Propiedades de “StaticMesh”..... 265
Imagen 353: Tutorial 17-Modificación de material de “Static Mesh”..... 266
Imagen 354: Tutorial 17-Propiedades elemento “Material Instance”..... 266
Imagen 355: Tutorial 17-Elemento “Matinee” en “Kismet”..... 267
Imagen 356: Tutorial 17-Interfaz herramienta “Unreal Matinee”..... 267
Imagen 357: Tutorial 17-Creación de grupo vacío. 268
Imagen 358: Tutorial 17-Creación track “Vector Material Param” en grupo..... 268
Imagen 359: Tutorial 17-Propiedades track “Vector Material Param”..... 268
Imagen 360: Tutorial 17-Primer punto de control en secuencia de tiempo. 269
Imagen 361: Tutorial 17-Segundo punto de control en secuencia de tiempo. 270
Imagen 362: Tutorial 17- Modificación de color en secuencia de tiempo..... 270
Imagen 363: Tutorial 17-Asociación “Level Loaded” y elemento “Matinee”..... 271
Imagen 364: Tutorial 18-Nivel inicial..... 272
Imagen 365: Tutorial 18-“SceneCaptureCubeMapActor” en “Actor Classes”..... 272
Imagen 366: Tutorial 18-Situación de “SceneCaptureCubeMapActor”en nivel. 273
Imagen 367: Tutorial 18-Creación elemento “TextureRenderTargetCube”. 274
Imagen 368: Tutorial 18-Propiedades “SceneCaptureCubeMap”..... 275

Imagen 369: Tutorial 18-Creación “Static Texture”. 276
Imagen 370: Tutorial 18-Creación de material. 276
Imagen 371: Tutorial 18-Interfaz “Unreal Material Editor”. 277
Imagen 372: Tutorial 18-Creación elementos “Multiply”, “Fresnel” y “Constant”. ... 277
Imagen 373: Tutorial 18-Asociación elementos. 278
Imagen 374: Tutorial 18-Creación de elemento de textura. 278
Imagen 375: Tutorial 18-Asociación del conjunto completo. 279
Imagen 376: Tutorial 18-Propiedades “Texture”. 279
Imagen 377: Tutorial 18-Enlace secuencia con propiedades del material. 280
Imagen 378: Tutorial 18-Nivel con material espejo acoplado. 281
Imagen 379: Tutorial 19-Creación de material. 282
Imagen 380: Tutorial 19-Interfaz “Unreal Material Editor”. 282
Imagen 381: Tutorial 19-Textura de lluvia. 283
Imagen 382: Tutorial 19-Elemento “Texture Sample”. 283
Imagen 383: Tutorial 19-Asociación “Texture Sample” con material. 284
Imagen 384: Tutorial 19-Propiedades del material. 284
Imagen 385: Tutorial 19-Asociación “Texture sample” con “Opacity”. 285
Imagen 386: Tutorial 19-Elemento “Panner”. 285
Imagen 387: Tutorial 19-Asociación de “Panner” y “Texture Sample”. 286
Imagen 388: Tutorial 19-Elemento “TextCoord” 286
Imagen 389: Tutorial 19-Asociación de “TexCoord” con “Panner”. 287
Imagen 390: Tutorial 19-Vista implementación en editor. 287
Imagen 391: Tutorial 19-Ejemplo de uso de lluvia. 288
Imagen 392: Tutorial 20-Aplicación “RAD Video Tools”. 289
Imagen 393: Tutorial 20-Pantalla conversión a “.bink”. 290
Imagen 394: Tutorial 20-Creación de material. 291
Imagen 395: Tutorial 20-Interfaz “Unreal Material Editor”. 291
Imagen 396: Tutorial 20-Elemento”Texture Sample” con video acoplado. 292
Imagen 397: Tutorial 20-Asociación del “Texture Sample” con material. 293
Imagen 398: Tutorial 20-Implementación completa en editor. 293
Imagen 399: Tutorial 20-Material acoplado incorrectamente. 294
Imagen 400: Tutorial 20-Propiedades objeto. 295
Imagen 401: Tutorial 20-Material acoplado correctamente. 296
Imagen 402: Tutorial 21-Variables respecto tamaño de personaje 296
Imagen 403: Tutorial 22-Elemento “Prepare Map Change”. 297
Imagen 404: Tutorial 22-Propiedades del elemento “Prepare Map Change”. 297
Imagen 405: Tutorial 22-Elemento “Commit Map Change”. 298

Imagen 406: Tutorial 22-“Prepare Map Change” y “Commit Map Change”. 298
Imagen 407: Tutorial 22-Implementación en mapa de llegada. 299
Imagen 408: Tutorial 22-Tipos de volumen. 300
Imagen 409: Tutorial 22-Interfaz “Levels”. 301
Imagen 410: Tutorial 23-Elemento “Console Command”. 301
Imagen 411: Tutorial 23-Propiedades elemento “Console Command”. 302
Imagen 412: Tutorial 23-Modificación de la etiqueta “Commands”. 302
Imagen 413: Tutorial 23-Asociación jugador con “Console Command”. 302
Imagen 414: Tutorial 23-Implementación de ejecución comando. 303
Imagen 415: Tutorial 24-Pantalla de bienvenida. 303
Imagen 416: Tutorial 25-Interfaz “Unreal Frontend”. 304
Imagen 417: Tutorial 25-Finalización de la compilación. 305
Imagen 418: Tutorial 25-Entrada del nombre del juego. 306
Imagen 419: Tutorial 25-Comprensión de archivos. 306
Imagen 420: Estructura creación de personajes. 307
Imagen 421: Interfaz aplicación “3D Studio Max”. 307
Imagen 422: Esqueleto. 308
Imagen 423: Malla y esqueleto. 308
Imagen 424: Acople de malla y esqueleto. 309
Imagen 425: Menú de asociación de objeto de malla con hueso del esqueleto. 310
Imagen 426: Sección de nombres de huesos. 311
Imagen 427: Edición de secuencia de frames. 312
Imagen 428: Barra de frames. 312
Imagen 429: Botón para crear un punto de control de frames. 312
Imagen 430: ActorX 313
Imagen 431: Utilidades en la aplicación “3D Studio Max”. 314
Imagen 432: Utilidad “ActorX” en aplicación “3D Studio Max”. 315
Imagen 433: “Animation Explorer”. 316
Imagen 434: “Animation Manager” de la utilidad ActorX. 317
Imagen 435: Elemento “SkeletalMesh” en “Content Browser”. 318
Imagen 436: Interfaz “Unreal AnimSet Editor”. 318
Imagen 437: Pestaña “Mesh” en el “Unreal AnimSet Editor”. 319
Imagen 438: Elemento “Physics Asset” en “Content Browser”. 319
Imagen 439: Interfaz del editor de los elementos “Physics Asset”. 320
Imagen 440: Elemento “AnimSet” en el “Content Browser”. 320
Imagen 441: Interfaz del editor de los elementos “AnimSet”. 321
Imagen 442: Animaciones pertenecientes a un “AnimSet”. 321

Imagen 443: Reproducción de animación. 322
Imagen 444: Localización de “Socket Manager”. 323
Imagen 445: Creación de “Socket”. 323
Imagen 446: Nombre del “Socket”. 323
Imagen 447: Interfaz del “Socket Manager”. 324
Imagen 448: Interfaz “Anim Tree Editor”. 325
Imagen 449: Asociación “Skeletal Mesh” con “Anim Tree”. 325
Imagen 450: Elemento “UTAnimBlendByPhysics”. 326
Imagen 451: Elemento “UDKAnimBlendByPosture”. 326
Imagen 452: Elemento “UTAnimBlendByFall”. 327
Imagen 453: Elemento “UDKAnimBlendByIdle”. 327
Imagen 454: Elemento “AnimNodeBlendDirectional”. 328
Imagen 455: Asociación completa de elementos en “Anim Tree”. 328
Imagen 456: Etiqueta “Display Name” del elemento “AnimSet”. 329
Imagen 457: Elemento “UTAnimNodeSeqWeap”. 329
Imagen 458: Ejecución de la segunda salida. 330
Imagen 459: Ejecución de la tercera salida. 330
Imagen 460: Implementación de la clase del nuevo personaje. 332
Imagen 461: Implementación de llamada a clase del personaje a utilizar. 332
Imagen 462: Implementación de la clase “UTPawn”. 333
Imagen 463: Implementación de la nueva clase “UTPawn”. 334
Imagen 464: Ejemplo de avisos en la compilación de los archivos del motor. 334
Imagen 465: Ejemplo de la correcta compilación de los archivos del motor. 335
Imagen 466: Juego “CrossPlane”. 336
Imagen 467: Cámara en 3ª persona del juego. 337
Imagen 468: Implementación dotación de armas del protagonista. 337
Imagen 469: Implementación de consulta de condición. 338
Imagen 470: Implementación de condiciones en apertura puerta. 339
Imagen 471: Ejemplo visualización. 339
Imagen 472: Ejemplo de implementación de visualizaciones con condición. 340
Imagen 473: Ejemplo de implementación de apertura de puerta en “Matinee”. 340
Imagen 474: Reproducción de animación de personaje. 341
Imagen 475: Control de vida del enemigo acoplado. 342
Imagen 476: Apagado y encendido de luces mediante pulsador. 342
Imagen 477: Lista de luces. 343
Imagen 478: Reproducción de música al inicio del nivel. 343
Imagen 479: Modificación de la música. 344

Imagen 480: Reproducción de sonido en “Matinee”..... 344
Imagen 481: Implementación de teletransporte aleatorio del protagonista..... 345
Imagen 482: Fin del nivel..... 345
Imagen 483: Cámara en 1ª persona del juego. 346
Imagen 484: Imagen del cielo. 347
Imagen 485: Implementación de reproducción de truenos. 347
Imagen 486: Elemento de reproducción del sonido del viento. 348
Imagen 487: Elemento de reproducción del sonido del agua..... 348
Imagen 488: Elemento del viento..... 349
Imagen 489: Imagen de lluvia en el juego. 349
Imagen 490: Implementación de visualización..... 350
Imagen 491: Implementación de visualización en “Matinee”..... 350
Imagen 492: Ejemplo de visualización de “llegada de enemigos”. 351
Imagen 493: Comienzo de disparo a una distancia dada..... 351
Imagen 494: Movimiento de patrulla de personaje acoplado..... 352
Imagen 495: Imagen del protagonista nadando en el agua. 352
Imagen 496: Distorsión de la imagen al estar el protagonista dentro del agua..... 353
Imagen 497: Fin del nivel..... 354
Imagen 498: Fundimiento de la imagen a negro. 354
Imagen 499: Diagrama de Gantt – Planificación estimada. 356
Imagen 500: Diagrama de Gantt – Planificación real. 358

1. INTRODUCCIÓN.

1.1. Descripción de los capítulos del documento.

La estructura del documento se presenta a continuación indicando los distintos capítulos en que está compuesto:

- **Introducción**.- Presentación de los objetivos del proyecto.
- **Estado de la cuestión**.- Características e historia de los *"Game Engine"* (motor de videojuegos) desde su aparición. Se refleja a su vez los distintos tipos de motores que existen hasta el momento.
  También se presenta la historia y características del motor *"Unreal"*. Se indican los componentes que forman parte de dicho motor como las distintas versiones que ha tenido hasta el momento.
- **Unreal Editor**.- Descripción de la herramienta *"Unreal Editor"* del motor.
- **Content Browser**.- Descripción de la herramienta *"Content Browser"* del motor.
- **Unreal Kismet**.- Descripción de la herramienta *"Unreal Kismet"* del motor.
- **Unreal Matinee**.- Descripción de la herramienta *"Unreal Matinee"* del motor.
- **Elementos básicos de un nivel**.- Presentación de algunos elementos básicos necesarios para la creación de un nivel.
- **Tutoriales**.- Capítulo donde se exponen diferentes tutoriales para la realización de acciones mediante las herramientas del motor.
- **Nuevo personaje**.- Se indica la forma de poder crear y utilizar un nuevo personaje. Se presenta el desarrollo desde su creación en una herramienta de modelado 3D hasta su importación y uso en el motor.
- **Caso práctico**.- Juego propio implementado a modo de ejemplo mediante *"Unreal Development Kit"*.
- **Conclusiones**.- Consideraciones finales alcanzadas una vez finalizado el proyecto.
- **Líneas futuras**.- Proposiciones de implementación que no han sido tratadas en el caso práctico.
- **Gestión del proyecto**.- Presentación gráfica de las fases, tareas y actividades programadas para el desarrollo indicando las relaciones entre ellas, recursos asignados y estimaciones de tiempo.
- **Referencias**.- Colección de fuentes que han sido consultadas para el desarrollo del proyecto.

1.2. Objetivos del proyecto.

El objetivo principal de este proyecto fin de carrera es la elaboración de un manual donde se explica la forma de crear un videojuego mediante el motor de videojuegos *"Unreal Development Kit"* [1].

Con la aparición de los motores de videojuegos se facilitó el desarrollo de videojuegos, ya que proporcionan gran cantidad de herramientas, sin necesidad de partir de cero. En el proyecto fin de carrera se ha utilizado unos de los motores completos y sobre todo gratuitos a los que se puede tener acceso, tanto profesionales como particulares.

Con el manual desarrollado en el proyecto fin de carrera se ha querido proporcionar el punto de partida para todos aquellos que quieran crear su propio juego mediante este motor.

En la elaboración del manual se han seguido las siguientes fases:

- Estudio de todas las herramientas proporcionadas por el motor UDK [1].
- Forma de creación de un juego mediante dichas herramientas.

Adicionalmente al manual, se ha desarrollado un propio videojuego como ejemplo de creación de un videojuego mediante el motor *"Unreal Development Kit"*. En este videojuego se ha intentado plasmar todo aquello expuesto en el manual.

2. ESTADO DE LA CUESTIÓN.

2.1. *"Game Engine"* – Motor de videojuegos.

En esta sección se va a presentar una breve historia de los *"Game Engine"* ó *"motor de videojuegos"* utilizados para la realización de videojuegos.

Un *"Game Engine"* es un sistema compuesto por varias herramientas para la creación y desarrollo de videojuegos. Dicho sistema incluye funcionalidad para la compilación, renderización de gráficos 2D y 3D, sonido, animación, inteligencia artificial, sistema distribuido, paralelización, asignación de memoria y gráficos.

También se puede considerar como una serie de rutinas de programación que permiten el diseño, la creación y la representación de un videojuego.

Una ventaja de estos motores es el poder reutilizar todas las herramientas para crear diferentes juegos, logrando que el desarrollo de un juego sea mucho más rápido y económico. [2] [3]

2.1.1. ¿Qué son?

Los *"Game Engine"* proporcionan un conjunto de herramientas de desarrollo visual, además de componentes de software reutilizables. Estas herramientas se proporcionan generalmente en un entorno de desarrollo integrado para permitir el desarrollo de juegos simplificado y rápido.

La mayoría de las suites Game Engine proporcionan instalaciones que facilitan el desarrollo, tales como gráficos, sonido, física y AI funciones. Estos Game Engine a veces son llamados "middleware", ya que proporcionan una plataforma de software flexible y reutilizable que proporciona todas las funcionalidades básicas necesarias para desarrollar una aplicación de juego. Es una herramienta creada por programadores y para programadores.

Al igual que otras soluciones middleware, los Game Engine suelen proporcionar abstracción de la plataforma, permitiendo que el mismo juego se pueda ejecutar en varias plataformas, como consolas de videojuegos y ordenadores personales con pocos o ningún cambio realizado en el código fuente del juego. A menudo, los Game Engine están diseñados con una arquitectura basada en componentes que permite a ciertos sistemas específicos ser sustituidos o ampliados por otros más especializados (y, a menudo más caros), como Havok [4] para la física, Miles Sound System [5] para el sonido, o Bink [6] para el vídeo.

Algunos motores, como *RenderWare* [7], están diseñados como una serie de componentes independientes que pueden ser combinados para crear un motor personalizado, en lugar del enfoque más común de extender o personalizar una solución integrada flexible. Sin embargo, la extensibilidad sigue siendo una alta prioridad en los "Game Engine" debido a la gran variedad de usos para los que se aplican. A pesar de la especificidad del nombre, los Game Engine a menudo se utilizan para otros tipos de aplicaciones con gráficos en tiempo real, como demostraciones de marketing, visualizaciones arquitectónicas, simulaciones y entornos de modelado.

Otros sólo proporcionan capacidades de tiempo real en 3D en lugar de la amplia gama de funcionalidad que necesitan los juegos. Estos motores se basan en el

desarrollador para crear el resto de la funcionalidad o conseguirla mediante otros componentes de middleware. Estos tipos de motores son llamados generalmente *"motor de gráficos"*, *"motor de renderizado"*, o *"motor 3D"* en lugar del término más general *"motor de juegos"*. Algunos ejemplos de los motores gráficos son: *Crystal Space*, *Genesis3D*, *Irrlicht*, *motor JMonkey*, *OGRE*, *RealmForge*, *Truevision3D*, y el *Vision Engine*. [8]

2.1.2. Historia de los motores de videojuegos.

Antes de los Game Engine, los juegos eran desarrollados generalmente como entidades singulares: un juego para el Atari 2600 [9], por ejemplo, tuvo que ser diseñado desde muy bajo nivel para hacer un uso óptimo del hardware de los gráficos.

Otras plataformas tenían más libertad de acción, pero incluso cuando los gráficos no eran una preocupación, las limitaciones de la memoria por lo general saboteaban los intentos de crear el diseño que un motor necesitaba. Incluso en plataformas con más capacidad, muy poco se podía reutilizar entre los juegos. Así la mayoría de los diseños de juegos en la década de 1980 fueron diseñados con un fuerte conjunto de reglas con una pequeña cantidad de datos y gráficos.

La primera generación de motores de gráficos o renderizadores (y precursor de lo que hoy conocemos como motores) fue dominada por tres entidades: Brender de Argonaut Software [10], Renderware de Criterion Software Limited y RenderMorphics de Reality Lab [11].

Reality Lab fue el más rápido de los tres y fue el primero en ser adquirido por parte de Microsoft. El equipo de RenderMorphics: Servan Keondjian, Seekings Kate y Rabson Doug se sumaron a los proyectos de Microsoft que hizo cambiar la empresa Reality Lab en Direct3D [12]. Renderware fue comprado finalmente por EA (Electronic Arts) [13], pero fue marginado por el gigante de los juegos.

El término *"Game Engine"* surgió a mediados de la década de 1990, especialmente en relación con los juegos 3D como los "shooters" en primera persona (FPS). Tal fue la popularidad de los juegos *"Doom"* y *"Quake"* que, en lugar de trabajar a partir de cero, muchos desarrolladores compraron la licencia de las partes fundamentales del software y a partir de ellas diseñaron sus propios gráficos, personajes, armas y niveles. La separación de las normas específicas del juego y los datos básicos, como la detección de colisiones y la entidad del juego, produjo que los equipos pudiesen crecer y especializarse. [14]

Más tarde, los juegos, como *"Quake III Arena"* y *"Unreal"* de "Epic Games" de 1998, fueron diseñados con este enfoque en mente, con el motor y el contenido desarrollado de forma separada. La práctica de la concesión de licencias tecnológicas, como se ha demostrado, es una corriente útil de ingresos auxiliares para algunos desarrolladores de videojuegos, ya que una licencia para un motor de juego de alta calidad puede variar desde 10.000 a millones de dólares, y el número de licenciatarios pueden alcanzar varias docenas de empresas, como se ha visto con el Unreal Engine. Los motores permiten el desarrollo de secuelas de juegos de una manera más rápida y fácil, que es una ventaja valiosa en la industria del video juego. [15]

Los motores modernos de juegos son algunas de las aplicaciones más complejas creadas, a menudo con docenas de sistemas finamente sincronizados que interactúan

entre ellos controlados con precisión. La continua evolución de los motores de juegos ha creado una fuerte separación entre la renderización, guión y diseño de los niveles.

"Threading" está adquiriendo más importancia hoy en día debido a los modernos sistemas multi-núcleo (por ejemplo, Cell) y al aumento de la demanda del realismo. Threads normalmente incluyen la representación, transmisión, audio, y la física. Los juegos de carreras típicamente han estado a la vanguardia del threading ya que el motor de físicas se ejecuta en un hilo separado mucho antes que otros subsistemas principales, en parte porque la renderización y las tareas relacionadas necesitan una actualización a sólo 30-60 Hz. Por ejemplo, en PlayStation 3, la física corre en *"Need for Speed"* a 100 Hz en comparación con Forza Motorsport 2 a 360 Hz. [16]

Aunque el término de Game Engine fue utilizado por primera vez en la década de 1990, hay algunos sistemas a principios de la década de 1980 que también son considerados como motores de juegos, tales como Adventure Game Interpreter (AGI) de Sierra, el sistema SCUMM de LucasArts y el motor Freescape de Incentive Software. A diferencia de la mayoría de los motores de juego moderno, estos motores de juego nunca fueron utilizados en los productos de terceros (excepto el sistema SCUMM, que fue autorizada y utilizada por Humongous Entertainment).

Game Middleware.

En el sentido más amplio del término, los motores de juegos se pueden describir como middleware. En el contexto de los videojuegos, sin embargo, a menudo es usado para referirse a los subsistemas de la funcionalidad dentro de un motor de juegos. Algunos middleware realizan sólo una cosa, pero hace que sea más convincente y más eficiente que el middleware sea de propósito general. Por ejemplo, *SpeedTree* se utiliza para representar los árboles y la vegetación en un videojuego. [17]

Los cuatro paquetes de middleware más utilizados que proporcionan la funcionalidad de los subsistemas incluyen: Bink RAD Game Tools, Firelight FMOD, Havok y GFX Scaleform.

RAD Game Tools Bink se desarrolla para la prestación básica de vídeo, junto con el audio Miles y Granny 3D. Firelight FMOD es una herramienta y librería de audio. Havok ofrece un sistema de simulación física robusta, junto con un conjunto de soluciones de animación y comportamiento. Scaleform ofrece GFX de alto rendimiento Flash UI, junto con una solución de vídeo de alta calidad de reproducción.

Algunos middleware contienen el código fuente completo mientras que otros sólo ofrecen una API que hace referencia a una biblioteca. Algunos programas de middleware pueden necesitar un cierto tipo de licencia, generalmente para un acceso más amplio para el código fuente completo.

Tendencias actuales.

La tecnología del Game Engine va madurando volviéndose cada vez más fácil de utilizar. La aplicación de los motores se ha ampliado en su alcance. Ahora están siendo utilizados para visualización, formación, medicina y aplicaciones militares de simulación. Para facilitar esta accesibilidad, las nuevas plataformas de hardware están siendo dirigidas por los motores de juego, incluidos los teléfonos móviles (por ejemplo, los teléfonos Android e iPhone) y navegadores web (por ejemplo, Shockwave, Flash, Webvision Trinigy es, Silverlight, Unity Web Player, O3D y dhtml puro).

Además, están surgiendo motores de juegos que se basan en lenguajes de alto nivel tales como Java y C#/.NET (por ejemplo, TorqueX y Visual3D.NET) o Python (Panda3D). La mayoría de los juegos 3D están ahora limitados por GPU (es decir, limitada por la potencia de la tarjeta gráfica), la ralentización del potencial de lenguajes de alto nivel es insignificante, mientras que el aumento de la productividad que ofrecen estos idiomas trabaja en beneficio de los desarrolladores del motor de juegos. Estas tendencias actuales están siendo impulsadas por empresas como Microsoft para apoyar el desarrollo independiente (Indie) de juegos. Microsoft desarrolló XNA como el SDK de elección para todos los juegos lanzados para Xbox y productos relacionados. [18]

2.1.3. Tipos de motores.

Existen multitud de motores con diferente nivel de conocimientos en programación. Para ver lo diferentes que pueden llegar a ser unos de otros se van a mostrar tres tipos de motores: roll-your-own version, mostly-ready version y el motor point-and-click. [19]

Motores *"roll your own - créatelo tú mismo"* (nivel más bajo)

A pesar del costo, muchas empresas (así como los responsables de juegos Indie) tratan de desarrollar sus propios motores. Esto significa que utilizan aplicaciones disponibles al público, como las API de XNA, DirectX, OpenGL, para crear sus propios motores. Además, recurren a otras bibliotecas, tanto comerciales o de código abierto, para su implementación. Estas bibliotecas pueden incluir funcionalidades para la física como Havok y ODE, funcionalidades para los gráficos como OpenSceneGraph y funcionalidades GUI como AntTweakBar.

En general, estos sistemas proporcionan a los programadores la mayor flexibilidad, permitiéndoles escoger y elegir los componentes que quieren e integrarlos exactamente como ellos quieren. Pero también se pierde mucho tiempo en su construcción. Además, los programadores con frecuencia se tienen que construir todas las herramientas a partir de cero, ya que rara vez se puede confiar en que todas las bibliotecas utilizadas trabajen juntas a la perfección. Esto hace al motor muy poco atractivo para la mayoría de los desarrolladores de juegos.

Motores *"Mostly ready – todo listo"* (nivel intermedio)

Estos motores están listos desde el primer momento para trabajar con ellos, proporcionando renderización, interfaz gráfica de usuario, física, etc. Muchos de ellos incluso poseen un conjunto de herramientas las cuales no hay que configurar para su funcionamiento conjunto (roll-your-own). Ejemplos de esta categoría son OGRE y Genesis3D, que son de código abierto, de bajo precio como Torque, y de precio muy alto, como Unreal, id Tech y Gamebryo.

Todos estos motores requieren todavía un poco de programación para lograr poner en marcha un juego completo. Se puede llegar a necesitar código de bajo nivel para hacer trabajar un juego. Muchos de estos motores son el producto de decenas de personas trabajando a lo largo de cientos de horas, y proporcionarán un mejor rendimiento con menos esfuerzo que la mayoría de los motores del primer tipo, incluso si no llegan a realizar exactamente lo que se quiere.

Motores *"Point-and-click - apuntar y pulsar"* (nivel más alto)

Motores de este tipo son cada vez más comunes hoy en día. Incluyen una herramienta completa que permite la creación de un juego propio. Estos motores, como GameMaker, Torque Game Builder y Unity3D, están diseñados para ser lo más amigable posible, y para utilizar tan poco código como sea posible. Eso no quiere decir que teniendo conocimientos de programación no ayuda, pero no es una necesidad.

El problema con muchos motores de este tipo es que pueden llegar a ser extremadamente limitados. Muchos proporcionan uno o dos tipos de géneros, o uno o dos tipos de modos gráficos. Incluso frente a las restricciones de estas herramientas, es posible hacer juegos muy creativos o incluso encontrar formas creativas en torno a esas restricciones. Lo mejor de estos motores es que permiten trabajar de forma rápida sin necesidad de mucho trabajo.

2.1.4. Lista de motores.

Se van a proporcionar unas listas de los principales motores de juegos divididos según su licencia de uso [20].

En la tabla 1 se muestra el listado de los motores más importantes con licencia gratuita y código abierto con libre distribución del código y desarrollo libre.

Free y Open Source

NOMBRE	DESCRIPCIÓN
Aleph One	Motor desarrollado en el lenguaje C y utilizado para implementar el juego *"Marathon"*. Licencia tipo GPL.
Axiom Engine	Motor desarrollado en lenguaje C# con licencia GPL.
Blender	Motor desarrollado en lenguaje C++ con licencia GPL. Un ejemplo de juego creado es *"Color Cube"*.
Box2D	Motor desarrollado en lenguaje C++ con licencia MIT. Un juego desarrollado con este motor es *"Angry Birds"*.
Build engine	Motor desarrollado en lenguaje C y licencia Custom. Un ejemplo de juego es *"Duke Nuken 3D"*.
Crystal Space	Motor desarrollado en lenguaje C++ con licencia LGPL.
Cube	Motor desarrollado en lenguaje C++ y su licencia es zlib.
Delta3d	Motor desarrollado en lenguaje C++ con licencia LGPL.
Flexible Isometric Free Engine	Motor desarrollado en lenguaje C++ y su licencia es LGPL. Un ejemplo de juego desarrollado con este motor es *"Unknown Horizons"*.
Genesis3D	Motor desarrollado en lenguaje C.
id Tech	Motor desarrollado en lenguaje C con licencia GPL. Ejemplos de juegos creados con este motor son *"Doom"*, *"Doom 2"*, *"HeXen"*, *"Quake II"* y *"Quake III Arena"*.
OpenSceneGraph	Motor desarrollado en lenguaje C++.
Panda3D	Motor desarrollado en lenguaje C++ y licencia BSD. El juego *"Pirates of the Caribbean"* ha sido creado con este motor.
PixelLight	Motor desarrollado en lenguaje C++ con licencia LGPL.

Quake engine	Motor desarrollado en lenguaje C con licencia GPL.
Second Life	Motor desarrollado en lenguaje C++.

Tabla 1.Motores con licencia Open Source.

En la siguiente tabla, tabla 2, se especifican los motores con licencia freeware siendo motores de software no libre que se distribuyen sin costo para su uso y por un tiempo ilimitado una versión de prueba. En el caso que el usuario quiera utilizar una versión completa del motor habilitando toda su funcionalidad debe pagar por él.

Freeware

NOMBRE	DESCRIPCIÓN
Adventure Game Studio	Este motor es uno de los más populares a la hora de desarrollar juegos de aventuras amateur.
DX Studio	Motor freeware con un conjunto muy completo de herramientas para el desarrollo de videojuegos en 3D. Posee una licencia de pago para tener acceso a ciertas funcionalidades.
Unity	Motor muy complete en el que se puede desarrollar para web, Windows y Mac. Obteniendo una licencia se puede desarrollar para iPhone, Android, Nintendo Wii, Playstation 3 y la Xbox 360.
Unreal Engine	Completo motor para desarrollar juegos para PC. Obteniendo una licencia se puede desarrollar juegos para Xbox 360 y PS3.

Tabla 2.Motores con licencia Freeware.

Por úlitmo se especifican en la tabla 3 aquellos motores de videojuegos con licencia en la que el usuario tiene limitaciones para usarlo, modificarlo o redistribuirlo.

Propietario

NOMBRE	DESCRIPCIÓN
Bork3D Game Engine	Motor para desarrollar juegos de plataformas para iPhone e iPad.
BigWorld	Servidor, cliente y herramientas de desarrollo de MMOG para Windows, Xbox 360 y PS.
BRender	Motor gráfico 3D en tiempo real para simuladores y herramientas gráficas.
DX Studio	Motor que permite la creación de juegos en tiempo real y simuladores.
Enigma Engine	Motor de juegos de táctica en tiempo real, utilizado para *"Blitzrieg"*.
Freescape (1986)	Uno de los primeros motores 3D propietarios, utilizado en *"Driller"* y *"3D Construction Kit"*.
Gamestudio	Sencillo motor para principiantes.
id Tech 4 (también conocido "motor Doom 3")	Utilizado para los juegos *"Doom 3*, *Quake 4*, *Prey* y *Quake Wars*". En Septiembre de 2011 llegará a ser Open Source.
Infinity Engine	Permite la creación de videojuegos de rol.
M.U.G.E.N	Creado por *"Elecbyte"* para juegos de lucha e 2D.

Q (game engine)	Completo framework desarrollado por el equipo de Direct3D que permite el desarrollo para PC, Wii, PS2, PS3, Xbox, Xbox 360, PSP, iPhone etc.
RAGE	Creado por *"Rockstar Games"* para mejorar sus juegos para Xbox 360 y PlayStation 3. Utilizado para el juego Grand Theft Auto 4.
Scaleform	Motor gráfico utilizado para visualizar elementos Adobe Flash, HUDs y texturas animadas para juegos en PC, Mac, Linux, Xbox 360, PlayStation 2, PlayStation Portable, PlayStation 3 y Wii.
Source engine	Desarrollado por *"Valve Software"* para el juego *"Half-Life 2"*.
Unity	Motor que permite el desarrollo de videjuegos para web, Windows, Mac OS X, iOS (iPod, iPhone, and iPad), Android, Nintendo Wii, XBox 360 y PS3.
Zillions of Games	Utilizado para desarrollar juegos de tablero, como ajedrez.

Tabla 3.Motores con licencia Propietario.

2.2. *"Unreal Engine"* – Motor Unreal.

"Unreal Engine" es un motor de juegos desarrollado por Epic Games [21]. El primer desarrollo con este motor fue "Unreal" en 1998, shooter en primera persona, que ha sido la base de muchos juegos desde entonces, incluyendo Unreal Tournament, Deus Ex, Turok, Tom Clancy's Rainbow Six 3: Raven Shield, Tom Clancy's Rainbow Six: Vegas, America's Army, Red Steel, Gears of War, Bioshock, Bioshock 2, Star Wars Republic Commando, Operaciones Tácticas: Asalto contra el terrorismo, Borderlands, Destroy All Humans! Path of the Furon, Batman: Arkham Asylum y Sección 8.

Unreal Engine, en pocas palabras, es un sistema que organiza tus propios elementos, como personajes, diseños, armas, efectos de sonido, música, etc, dentro de un gran entorno interactivo visual. En la imagen 1 se muestra todo este estorno mencionado. [22] [23] [24]

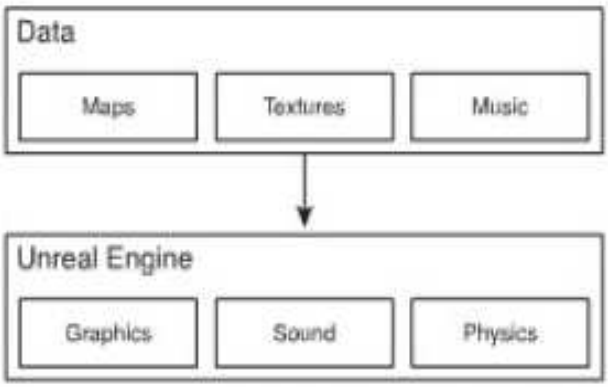


Imagen 1:Entorno Unreal Engine

Aunque primeramente se utilizó para shooters en primera persona, se ha utilizado con éxito en una gran variedad de géneros, incluyendo el sigilo (Tom Clancy's Splinter Cell), MMORPG (Vanguard: Saga of Heroes), así como en juegos de rol como Mass Effect, The Last Remnant, Matar Piso y Harry Potter y la piedra filosofal.

Con su núcleo escrito en C++, Unreal Engine cuenta con un alto grado de portabilidad siendo una herramienta utilizada por los desarrolladores de juegos soportando a una gran multitud de plataformas, incluyendo Microsoft Windows, Linux, IOS, Mac OS y Mac OS X en ordenadores personales y muchas consolas de videojuegos como Dreamcast, Xbox, Xbox 360, PlayStation 2 y PlayStation 3, aunque la última versión del Unreal Engine, UE3, no funciona en plataformas de generación anterior. Una gran parte del código de un juego se puede escribir con UnrealScript, un lenguaje de programación propietario, pero la mayoría del juego se puede modificar sin ahondar profundamente en el motor interno. Además, al igual que los paquetes de middleware, Unreal Engine ofrece diversas herramientas para la creación de contenidos, tanto para los diseñadores como los artistas.

2.2.1. Componentes.

Cada uno de los componentes de este motor pueden trabajar independientemente, pero todo el poder de desarrollo de este motor viene de hacer trabajar a todos conjuntamente en armonía gracias a un núcleo central. En la imagen 2 se muestra la relación entre el núcleo central y los distintos componentes. [25]

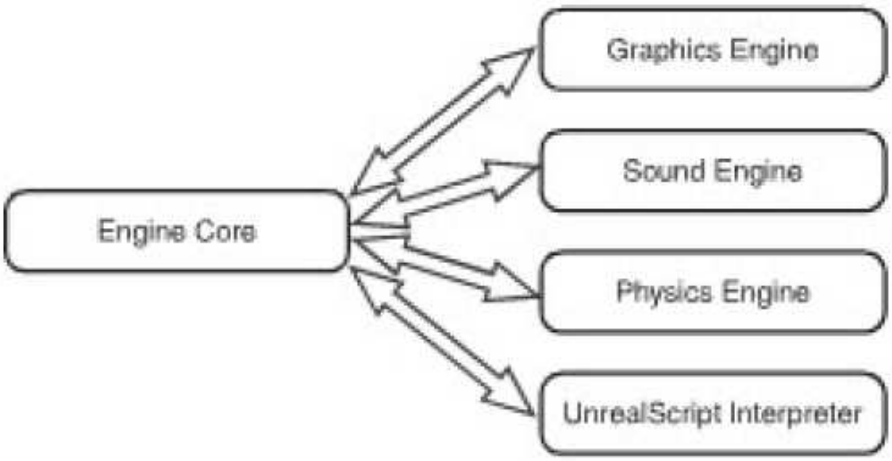


Imagen 2:Componentes "Unreal Engine".

Motor gráfico.

El motor gráfico controla todo lo que se muestra por pantalla. Sin este motor, sólo se vería una gran lista de valores como la localización del jugador, vida, armas que posee el jugador, entre otras. El motor recoge toda esta información y presenta el entorno mostrado durante el juego.

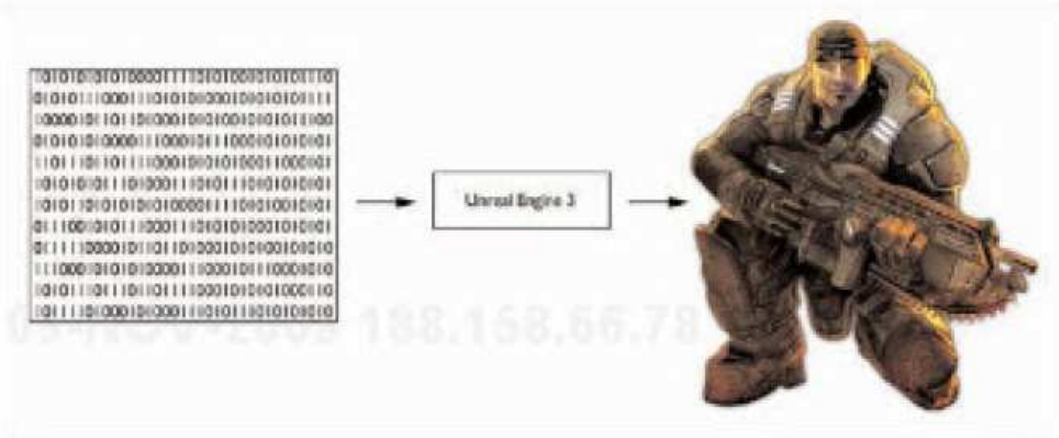


Imagen 3:Motor gráfico "Unreal Engine"

Es responsable de realizar todos los cálculos necesarios denominados *"behind –the-scenes"* (detrás de las escenas), como qué objetos se muestran delante de otros, y cuáles ocultan la visualización de otros.

Cuando algunos objetos son ocultados por otros, éstos no son renderizados por completo. Esta manera de trabajar permite liberar ciclos de trabajo para optimizar la eficiencia.

Una parte muy importante en la renderización es la iluminación. El motor proporciona un sistema robusto de iluminación. Permite una iluminación y sombreado dinámico, con lo que aumenta el realismo.

Motor de sonido.

El 90 por ciento que se percibe de un entorno es por medio del oído. El sonido es una parte importantísima en los juegos, aunque gráficamente sea de lo mejor. El sonido en un juego permite mejorar el realismo: el sonido producido por el jugador al pasar por distintos terrenos, la rotura de un cristal, recarga del arma, etc.

El motor de sonido permite obtener todos estos sonidos en el juego. Los sonidos a utilizar en un juego, el motor los acopla a los eventos que se produzcan, para reproducirlos simultáneamente, utilizando el hardware de sonido que posee el equipo.

Motor de físicas.

Todos los efectos que se producen en el mundo real pueden ser simulados en un juego mediante simulación e implementación. Estos efectos son simulados gracias al motor de físicas PhysX.

PhysX soporta casi la totalidad de simulaciones físicas, como cuerpos estáticos e incluso cuerpos dinámicos. Cada uno de ellos puede ser interactivo con los eventos del juego y pueden ser manipulados por el jugador.

Este motor puede trabajar en conjunto con el motor de sonido para obtener efectos de sonido dinámicos y realistas.

Manejador de entradas.

Sin poder manejar al personaje principal de un juego, no se tiene juego, se tendrá una animación de éste.

Las entradas de un juego pueden ser: botones presionados, movimientos del ratón y la forma más moderna, mandatos vocales. Cada tipo de entrada debe ser reconocido y convertido a mandatos que produzcan una reacción en el juego. Éste es el trabajo a realizar por el manejador de entradas.

Cada vez que se produce una entrada en el juego, el manejador crea una serie de propiedades para realizar los cambios oportunos en dicho juego. Por ejemplo, cuando el jugador presiona la tecla de *"salto"*, el manejador envía la señal al núcleo del motor para reproducir la animación de salto del personaje principal y el envío al motor gráfico la información necesaria para renderizar la acción en pantalla, a la vez que se envía la señal al motor de sonido para reproducir el correspondiente sonido. Claro está, que toda esta secuencia de acciones se produce de una manera muy rápida sin producirse un retardo entre la entrada y la reacción producida.

Infraestructura en red (Network).

Una opción actual a la hora de desarrollar un juego es la de multijugador (multiplayer). Durante un juego multijugador, los equipos de todos los jugadores deben intercambiar información a través de la red.

La forma más eficiente de realizar todo esto es limitar la información que es enviada a través de la red a la más importante del juego: posición de cada jugador y la interacción inmediata con el entorno. Enviando sólo la información más relevante se consigue un juego rápido aún con conexiones lentas.

Intérprete UnrealScript.

UnrealScript es uno de los aspectos más revolucionarios del motor Unreal Engine. Éste es un lenguaje de programación propio que permite realizar cambios en el motor, sin tener que modificar el código fuente.

Este lenguaje es muy similar a los lenguajes Java y C++, pero intentando que fuese algo más sencillo de aprender y utilizar.

Un intérprete de este lenguaje es el componente del motor responsable de transformar el código escrito es UnrealScript en código que el motor puede procesar.

A continuación, en la imagen 4, se muestra la secuencia producida desde un código escrito en este lenguaje hasta que pueda ser interpretado por el motor.

```
local PlayerReplicationInfo Leading, PRI;

Text=ScrollingMessage;

if(InStr(Text,"%If")!=-1||InStr(Text,"%Ip"!=-1)
{
     //find the leading player
     Leading=None;

ForEach AllActors(class'PlayerReplicationInfo',PRI)
   if (!PRI.bIsSpectator && (Leading==None||PRI.Score>
       Leading=PRI;

if(Leading!=None)
{
     Text=Replace(Text,"%Ip",Leading.PlayerName);
     Text=Replace(Text,"%If",string(int(Leading.Score)));
}
else
Text="";
}
```

Imagen 4:Unreal Script

2.2.2. Historia del motor Unreal Engine.

En el verano del año 1998 se produjo el nacimiento del motor Unreal Engine. Epic Games no estaba creando un juego, buscaba un sistema de herramientas y programas para poder crear cualquier juego.

Unreal Engine proporcionó a Epic Games, y a las compañías con licencia, un buen punto de partida de desarrollo de videojuegos. De repente, no era necesario re-crear todo desde el principio y se podía combinar el motor con la propia implementación. Con esto se podían crear videojuegos de una manera mucho más rápida.

Unreal Engine 1.3.

Dicha versión del motor Unreal Engine ya integraba renderización, detección de colisiones, IA, física, distribución del sistema y sistema de gestión de archivos en un motor completo. Era necesario realizar algunas concesiones para mantener los niveles de rendimiento con el hardware que se disponía en ese momento. Por ejemplo, Epic decidió utilizar *"la detección de colisiones cilíndrica"* sobre la *"detección de colisiones IK"* en un esfuerzo por mantener la tasa de fotogramas en equipos que eran comunes en el momento de su lanzamiento. Internamente, Epic desarrolló los juegos Unreal y Unreal Tournament mediante este motor.

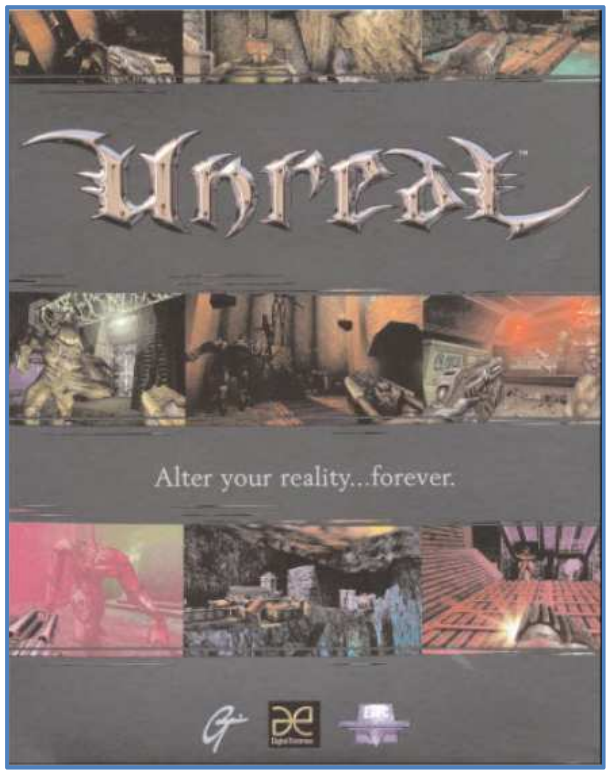


Imagen 5:Carátula juego "Unreal"

El juego Unreal fue importante, pero Unreal Tournament fue memorable. Se muestran las carátulas de ambos juegos en las imágenes 5 y 6.

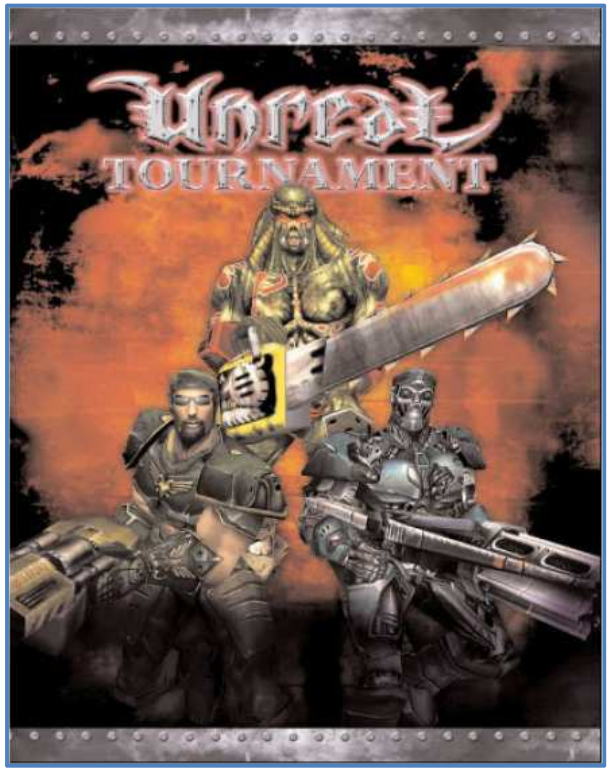


Imagen 6:Carátula juego "Unreal Tournament"

Transformó el concepto de *"death match"* y poseía la Inteligencia Artificial de los personajes más avanzada que en todos los juegos se había visto.

Unreal Engine 1 utilizó la API Glide [26], desarrollada específicamente para 3dfx GPU, y se ha actualizado para OpenGL y Direct3D. Probablemente la razón principal de su popularidad fue que la arquitectura del motor y la inclusión de un lenguaje de scripting lo hizo muy fácil de utilizar. Otra de las mejoras de Unreal en comparación con los otros motores era su tecnología de red, que en gran medida mejoraba la escalabilidad del motor en modo multijugador. Unreal fue también el primero en utilizar un verdadero modelo cliente-servidor en la arquitectura del motor.

Entre Marzo y Julio de 2001, Epic Games lanzó "*The Unreal Developer Network*", también llamado UDN [27]. Mediante esta web, los usuarios de Unreal Engine pueden encontrar documentación y tutoriales actualizados sobre este motor. A día de hoy, "*udn.epicgames.com*" es el sitio de referencia online para el desarrollo con Unreal Engine.

Imagen 7:Página web "The Unreal Developer Network"

Unreal Engine 2.

La segunda versión de Unreal Engine hizo su debut con el ejército de Estados Unidos. En esta versión el núcleo y el motor de renderizado fueron completamente re-escritos y con el nuevo UnrealEd 2 integrado. También integró el Karma SDK para la física, lo que potencia las físicas de los personajes en Unreal Tournament 2003 y Unreal Championship (2003), enfocados totalmente al tipo de juego multiplayer (multijugador).

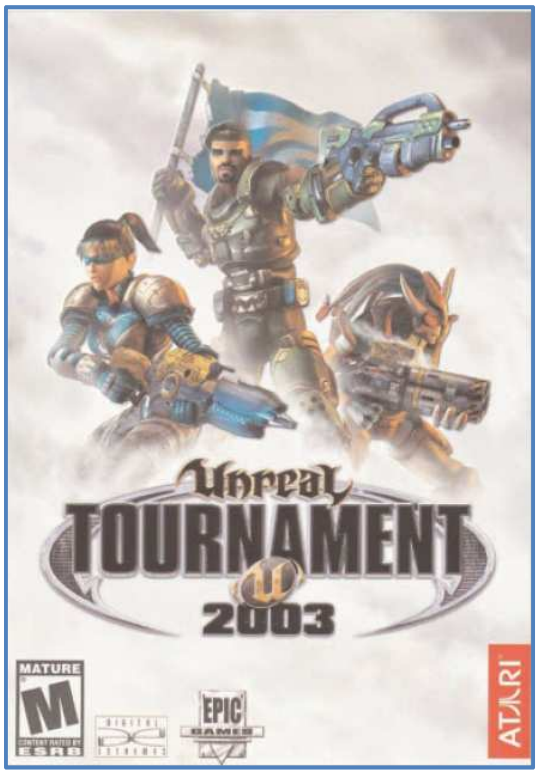


Imagen 8:Carátula juego "Unreal Tournament 2003"

Imagen 9:Carátula juego "Unreal Championship"

Otro elemento que integra esta versión es la herramienta Matinee, un sistema integrado para la creación de animaciones de los objetos de un videojuego.

Muchos elementos del motor también se actualizaron, con mejoras y añadiendo soporte para las consolas PlayStation 2, GameCube y Xbox. Con UE2.5, una

actualización, se mejoró el rendimiento de la representación física y añadió físicas de vehículos y el editor de las partículas para UnrealEd y soporte de 64 bits en Unreal Tournament 2004.

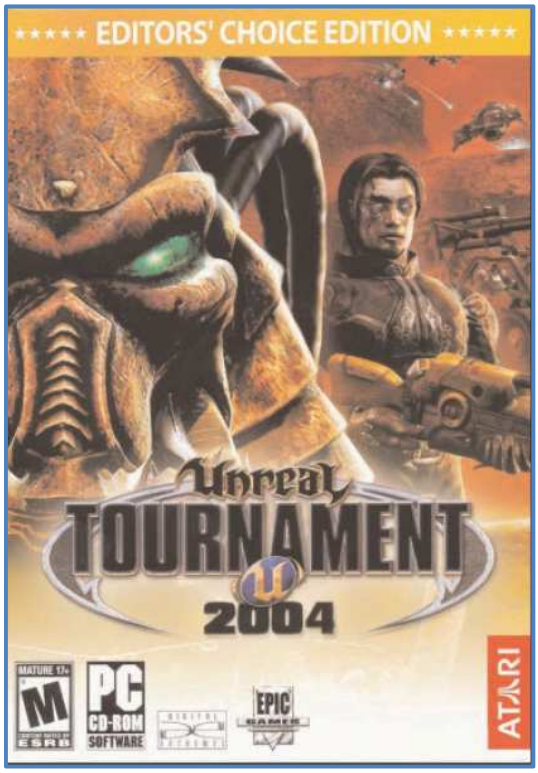


Imagen 10:Carátula juego "Unreal Tournament 2004"

Una versión especializada de UE2.5 llamada UE2X, que fue utilizada para Unreal Championship 2, desarrolla optimizaciones específicas de la primera generación de Xbox. EAX 3.0 y también compatible con sonido.

Imagen 11:Carátula juego "Unreal Championship 2"

El 24 de marzo de 2011, Ubisoft Montreal reveló que habían hecho funcionar con éxito el Unreal Engine 2.5 en la nueva plataforma de Nintendo, Nintendo 3DS.

Unreal Engine 3.

La tercera generación de Unreal Engine (UE3) fue diseñada para DirectX 9/10/11 en PC, Xbox 360 y PlayStation 3. Su renderización soporta muchas técnicas avanzadas, iluminación per-pixel, sombras dinámicas y basada en las herramientas disponibles en las versiones anteriores del motor.

Epic ha utilizado esta versión del motor para sus juegos Unreal Tournament 3, Gears of War, y una versión mejorada de Gears of War 2. Epic Games demostró con el juego Gears of War toda la capacidad que poseía esta versión del Unreal Engine. Un estilo especial de cámara en tercera persona, un rico y extremadamente detallado mundo y uno de los más viscerales estilos de combate. Con todas estas propiedades, *"Gears of War"* llegó a ser uno de los grandes éxitos en los videojuegos.

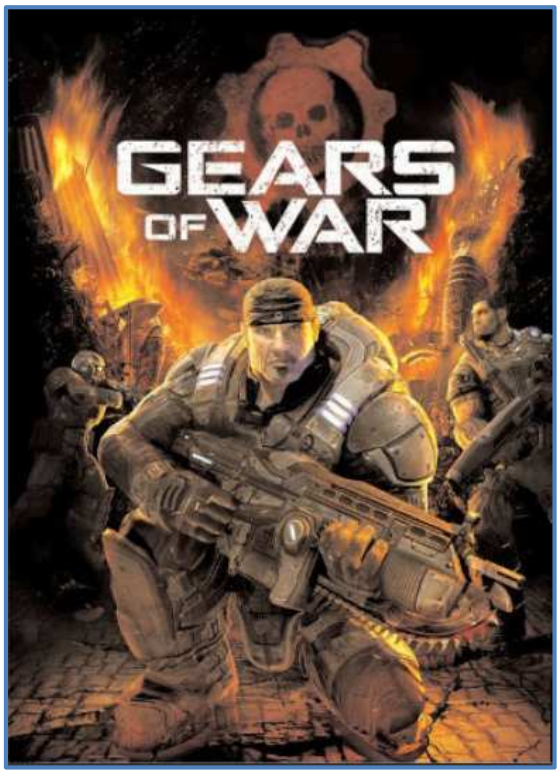


Imagen 12:Carátula juego "Gears of War"

Debido a la concesión agresiva de licencias, esta versión actual se ha ganado un gran apoyo por parte de grandes desarroladores, como Atari, Activision, Capcom, Disney, Konami, Koei, 2K Games, Midway Games, THQ, Ubisoft, Sega, Sony, Electronic Arts, Square Enix, 3D Realms y muchos más.

Epic Games ha anunciado en la GDC 2009 [28] una serie de mejoras al motor de Unreal 3. Entre ellos:

- Iluminación estática de alta calidad con efectos de última generación, como sombras suaves, penumbras y la reflexión especular.
- La posibilidad de agregar efectos de fractura de mallas estáticas para simular entornos destructibles.
- Suavizado de los cuerpos físicos.
- Simulación de grandes multitudes.

Además de la industria del juego, UE3 ha sido adoptado por muchos proyectos de diseño y simulación de construcción, simulación de conducción, centros comerciales virtuales y guiones de cine.

Unreal Development Kit (UDK)

Mientras que el Unreal Engine 3 ha sido bastante abierto para desarrollar proyectos con él, la capacidad de publicar y vender juegos hechos con UE3 está restringida bajo licencia. Sin embargo, en noviembre de 2009, Epic lanzó una versión gratuita de su motor, llamada Unreal Development Kit (UDK), que está disponible para el público en general. De acuerdo con la actual EULA [29], los desarrolladores de juegos pueden vender sus juegos mediante el pago a Epic de 99$ en un principio, y el 25% de todos los ingresos si son superiores a 50.000$. Además, en diciembre de 2010, UDK viene con soporte para la creación de juegos iOS.

Unreal Engine 4.

Mark Rein, vicepresidente de Epic Games, reveló el 18 de agosto de 2005 que el Unreal Engine 4 había estado en desarrollo desde 2003. La única persona que trabajaba en el sistema de Unreal Engine 4 hasta ese momento era Tim Sweeney, director técnico y fundador de Epic Games. Sin embargo, en la 2006 Game Developers Conferencia, Tim Sweeney dijo que el desarrollo no se iniciaría en serio hasta algún momento de 2008.

Predijo que la próxima generación de juegos de consolas llegaría en 2009, en la que los diseñadores deberían trabajar con CPUs con 20 o más núcleos, 80 o más subprocesos de hardware, y más de un teraflop de potencia de cálculo.

Michael Capps, presidente de Epic Games, indicó que el Unreal Engine 4 debería estar listo alrededor de 2012.

3. UNREAL EDITOR.

Al arrancar el motor, ésta es la primera herramienta que se muestra. Es el punto inicial para poder crear un nuevo nivel. A partir de esta herramienta se tiene acceso a todas las restantes que componen el motor y son necesarias para su implementación. [30].

Una vez instalado el motor, al ejecutarlo, se muestra la siguiente pantalla:

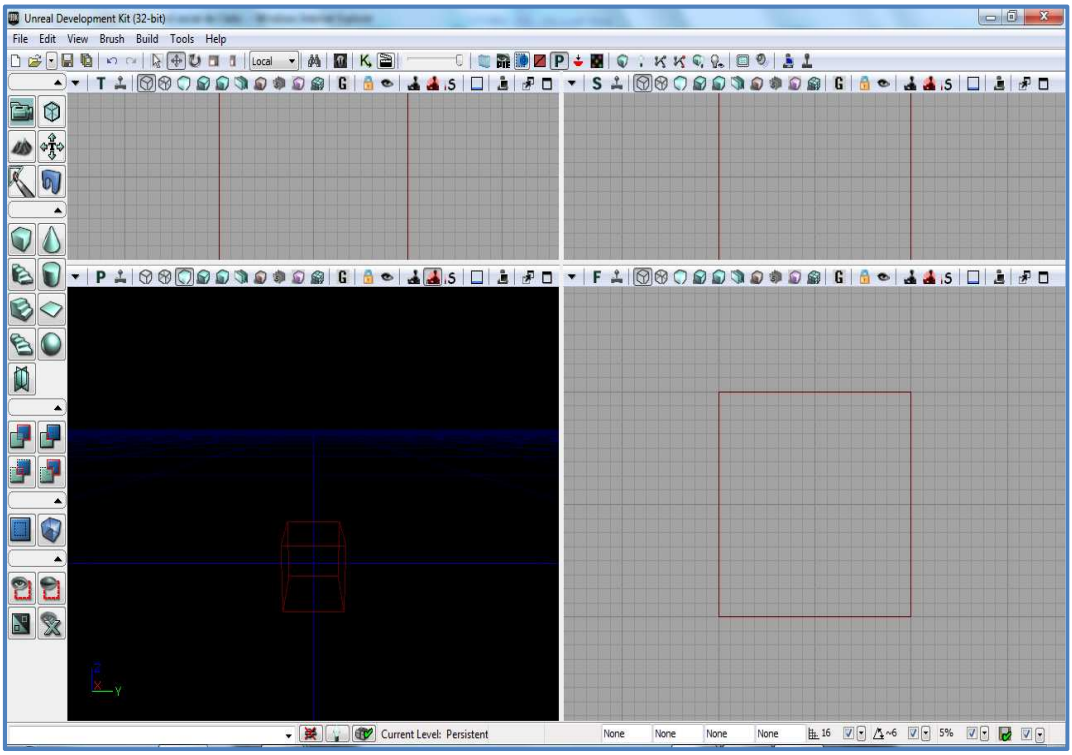


Imagen 13:Interfaz "Unreal Editor".

3.1. Menú de la aplicación.

Como en cualquier aplicación se tiene un menú en la parte superior de la ventana mediante el cual se puede crear un nuevo archivo, acceder a distintas herramientas del motor, compilación parcial o total de todos los elementos implementados, etc.

En la imagen 14 se muestra este menú.

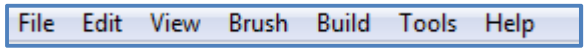


Imagen 14:Menú de la aplicación

A lo largo del manual se irán mostrando las utilidades de cada una de las secciones de este menú.

3.2. Barra de herramientas de Unreal Editor.

Imagen 15:Barra de herramientas "Unreal Editor"

Se van a explicar cada uno de los iconos mostrados en la barra de herramientas para conocer su funcionamiento:

- .- Crear un nuevo nivel.
- .- Abrir el directorio que contiene los niveles ya guardados.
- .- Guardar el nivel actual.
- .- Guardar todos los niveles.
- .- Deshacer lo realizado.
- .- Rehacer la acción deshecha.
- .- Selection mode: Herramienta para seleccionar.
- .- Translation mode: Herramienta para trasladar.
- .- Rotation mode: Herramienta para rotar.
- .- Scaling mode: Herramienta para escalar uniformemente.
- .- Non-uniform Scaling mode: Herramienta para escalar de forma no uniforme.
- .- Funcionalidad para buscar elementos del mapa creado.
- .- Acesso al *"Content Browser"*.
- .- Acceso al *"Kismet"*.
- .- Acceso al *"Matinee"*.
- .- Allow Traslucent Selection: Los objetos seleccionados son mostrados de forma transparente.
- .- Only Load Visible Levels in PIE: Mostrar los niveles de PIE.
- .- Toggle Brush Polygons: Activar polígonos.
- .- Enable/disable sockets snapping: Mostrar los puntos denominados *"sockets"* en los elementos *"Skeletal Mesh"*.
- .- Select to have *"Particle Systems"* use distance LOD in the perspective viewport: Mostrar los sistemas de partículas en la vista perspective.
- .- Build Geometry for visible levels: Compilar los elementos que componen la geometría del nivel.
- .- Build Lighting: Compilar la iluminación del nivel.
- .- Build Paths: Compilar paths del nivel.
- .- Build Node Covers: Compilar Node Covers del nivel.
- .- Build All: Compilar al completo todo el nivel.
- - Ligthing Quality Settings: Seleccionar la calidad que se desea de la iluminación.
  Para ir modificando la calidad de la iluminación sólo hay que ir pinchando con el botón izquierdo del ratón sobre el icono, el cual irá mostrando la calidad seleccionada:

  - .- Calidad media.
  - .- Calidad alta.
  - .- Calidad óptima.
- .- FullScreen Mode: Mostrar la interfaz a pantalla completa.
- .- Toggle Real Time Audio: Permite activar/desactivar el sonido acoplado al nivel en modo edición.
- .- Start this Level on PC: Lanzar el nivel en modo juego completo.
- .- Play this level in an editor windows: Se lanza el nivel permitiendo la jugabilidad, pero sin acoplarlo a toda la estructura de juego completo.

3.3. Vistas del editor.

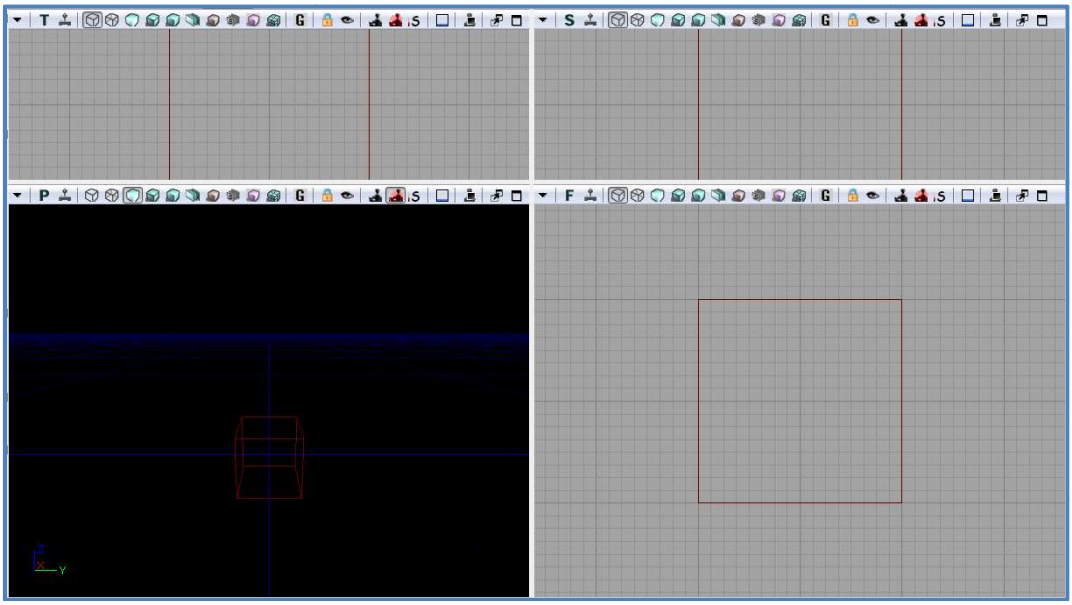


Imagen 16:Vistas del editor

Unreal Editor tiene la misma apariencia que los programas de modelado en 3D. En la imagen 16 se muestran las principales vistas del modelado: planta, alzado, perfil y vista en 3D.

Se podrá ir trabajando en cada una de las vistas según convenga. Para seleccionar una vista basta simplemente con pinchar en cualquier zona libre de ésta con el botón izquierdo del ratón y ya se tendrá seleccionada para poder trabajar en ella. Además la vista activa tendrá un marco de color amarillo indicando que ésa es la vista activa en ese momento.

3.3.1. Funcionalidades en las pantallas del editor.

En dicha sección se van a explicar cada una de las herramientas que poseen las vistas del editor:

Imagen 17:Herramientas de vistas del editor

Las pantallas pueden ser intercambiadas al gusto del usuario mediante los botones de cada una de las ventanas:

F Front
S Side
P Perspective
T Top

Para poder cambiar una por otra sólo hay que pinchar sobre dichos botones e irá modificándose la ventana pasando de una a otra en orden secuencial. Otra forma es pinchar con el botón secundario (normalmente botón derecho) del ratón y aparecerá un submenú para elegir el tipo de ventana que se quiere.

Se puede ver el mundo creado en tiempo real (real time) activando el icono . Esto puede ser útil para ver en el editor elementos como humo, fuego, etc., que no se podría ver hasta ejecutar el nivel.

Otra funcionalidad que se tiene es poder esconder los detalles mostrados en las vista. Para ello se utiliza el icono G. Activándolo/ desactivándolo se mostrarán o no los detalles del nivel creado en dicha vista.

Se puede bloquear el punto de vista mediante el icono . Así la vista no irá modificándose con los cambios que se vayan realizando, dejándola siempre fija. A su vez también se puede bloquear ciertos objetos a la cámara de la vista. Para ello hay que seleccionar los objetos a bloquear y utilizar el icono .

Hay algunos elementos del mapa que sólo se muestran de forma dinámica ó en ejecución. Las vistas permiten poder ver un previo de estos elementos mediante para los elementos de *"Streaming"* y para los elementos de *"Post Process"*.

Las vistas también permiten modificar la velocidad de movimiento de la cámara. Se tienen 3 niveles de velocidad de la cámara: más lento , intermedio y el más rápido . Para pasar de uno a otro simplemente hay que pinchar con el botón izquierdo del ratón y se irá pasando de un nivel a otro secuencialmente.

El nivel creado puede ser lanzado en cada una de las vistas pinchando en el icono y el nivel será lanzado en dicha vista.

Podemos querer tener alguna de las vistas en una ventana independiente, aparte de la ventana principal del Unreal Editor. Para ello sólo hay que pulsar en el icono y la nueva ventana con la vista será mostrada.

También cada una de las ventanas puede ser maximizada, mediante el icono activándolo, y será la única ventana que aparecerá. Para volver a tener las 4 vistas del editor sólo hay que desactivar el mismo botón.

3.3.2. Mover la vista en las pantallas.

Una de las principales ventajas a la hora de modelar es el dominar el movimiento en cada una de las vistas, el cómo moverse dentro de la vista.

En este caso hay que hacer distinción entre la ventana *"Perspective"* y las ventanas *"Side"*, *"Front"* y *"Top"*.

- Vista "Perspective": En dicha vista podemos movernos en 3 dimensiones en el mapa con lo que será muy alto el nivel de movimientos que se podrá hacer.
  - Para avanzar hacia delante ó hacia atrás con la vista se pueden utilizar las teclas direccionales del teclado (teclas ↑ y ↓). También se puede realizar el mismo movimiento presionando el botón izquierdo del ratón y a la vez moverlo hacia delante ó hacia atrás para avanzar ó retroceder en el mapa, respectivamente. En el caso que el ratón utilizado tenga la rueda central se puede utilizar para realizar el mismo movimiento.
  - Mediante las teclas direccionales del teclado de izquierda y derecha (teclas ← y →) se puede mover la vista hacia dichas direcciones pero sin girarla.
  - Si se quiere girar la vista, pero sin realizar un desplazamiento, hay que mantener presionado el botón derecho del ratón y desplazarlo de modo que se realice el movimiento deseado.
  - Otro movimiento que se puede hacer es desplazar la vista en altura. Para ello se mantienen presionados el botón izquierdo y el botón derecho del ratón simultáneamente y desplazándolo hacia delante ó hacia atrás para subir ó bajar la vista, respectivamente.
- Vista *"Side"*, *"Front"* y *"Top"*: En dicho caso el mapa se muestra en 2 dimensiones con lo que los movimientos que se pueden realizar son diferentes al caso anterior comentado.
  - Se puede acercar ó alejar la vista al mapa mostrado manteniendo pulsado el botón izquierdo y el botón derecho del ratón simultáneamente y moverlo hacia delante ó hacia atrás. Si el ratón que se está utilizando posee la rueda central, se puede utilizar para realizar dicho movimiento.
  - Para realizar el desplazamiento sobre el mapa hay que mantener presionado el botón izquierdo ó derecho del ratón y desplazarlo para realizar el movimiento deseado.

3.4. Comienzo de un nivel.

En esta sección se va a explicar la creación de la física ó estructura elemental que compone un nivel, como son las paredes, suelo, etc. En definitiva, todos aquellos objetos que no pueden ser acoplados desde el Content Browser y deben ser creados.

Cuando se quiere crear un nivel ó mapa, primero hay que tener en cuenta si se quiere un nivel exterior ó interior. Esto se preguntará al crear un nuevo archivo en el editor, indicando en qué modo se quiere realizar.

Los modos que se pueden seleccionar son:

- *"Additive mode"*.- Para crear caras externas.
- *"Substractive mode"*.- Para crear caras internas.

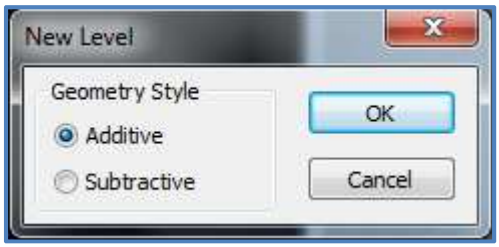


Imagen 18:Menú de modos de creación de un nivel.

Para que queden claros dichos modos se van a mostrar las imágenes del mismo cubo creado en los dos modos. En la imagen 19 se muestra un cubo con las caras exteriores creadas y en la imagen 20 se muestra el mismo cubo pero con las caras interiores.

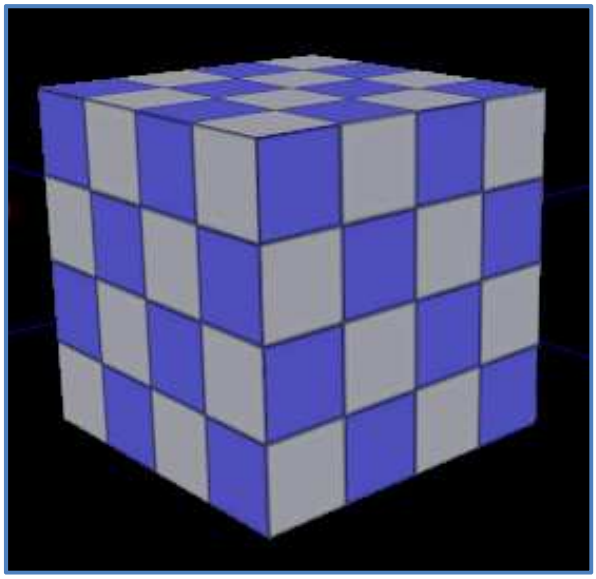


Imagen 19:Ejemplo "additive".

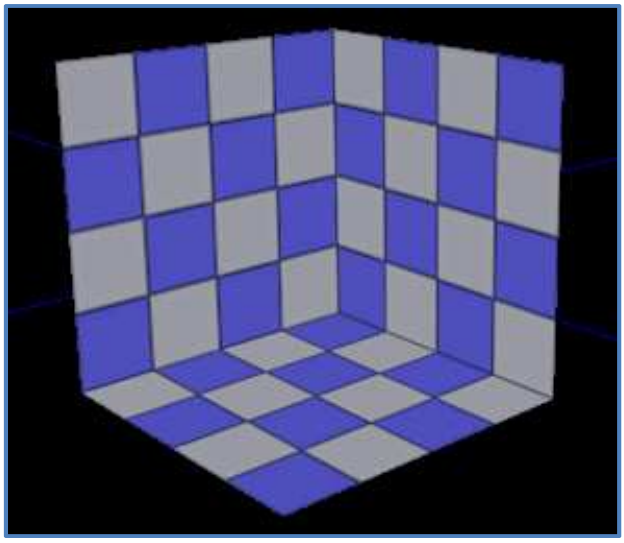


Imagen 20:Ejemplo "Substractive".

Hay que tener en cuenta que estos dos modos de creación de niveles son excluyentes, no pudiendo utilizar las dos formas en el mismo nivel.

3.4.1. Creando formas.

Mediante el editor se pueden ir creando objetos del mapa como las paredes, habitaciones completas, etc. Para ello se tiene un modelo ó forma mediante el cual se irán haciendo todos los objetos que se quiera. Este modelo es la forma de aristas rojas que se puede apreciar en todas las vistas, como se muestra en la imagen 21.

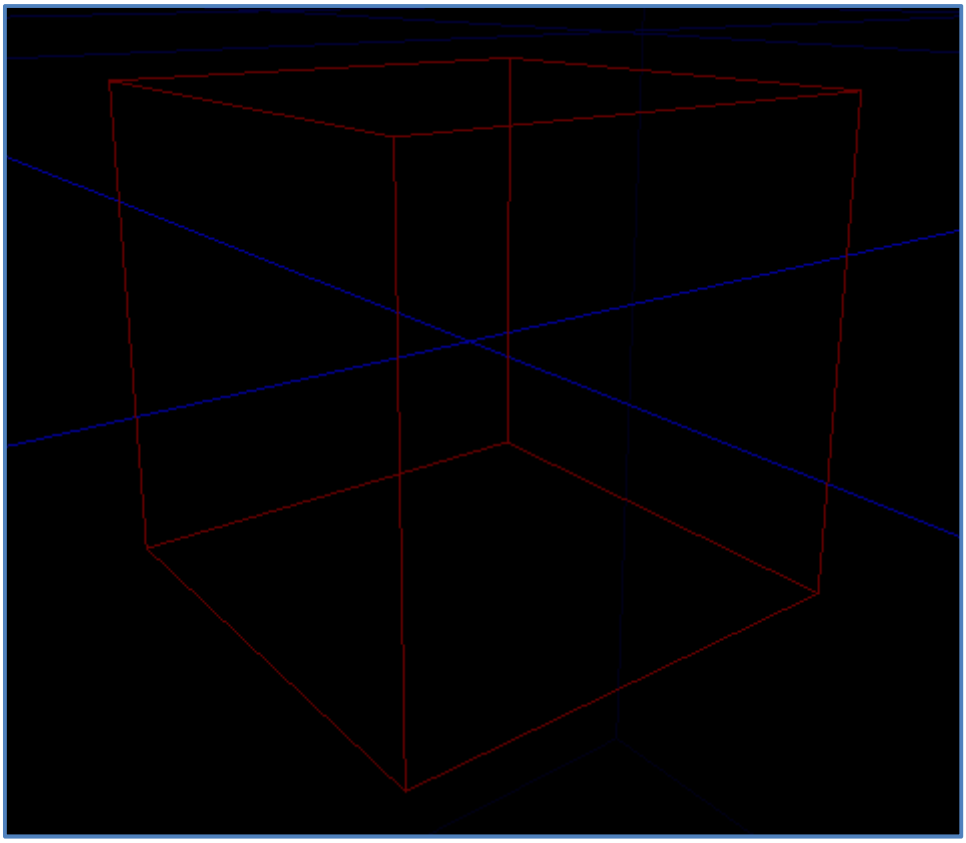


Imagen 21:Forma/modelo.

Dicho modelo podrá ser modificado en sus dimensiones, girado en todos los ejes y desplazado para lograr obtener el objeto deseado.

- <u>Modificar posición</u>
  En la barra de herramientas superior hay que tener activado el modo *"Traslation Mode"* pulsando en el icono .

  Con el modo activado y seleccionando la forma/modelo (pinchando con el botón izquierdo del ratón en cualquier arista de color rojo) aparecerán unas flechas de distinto color en cada uno de los ejes de coordenadas.

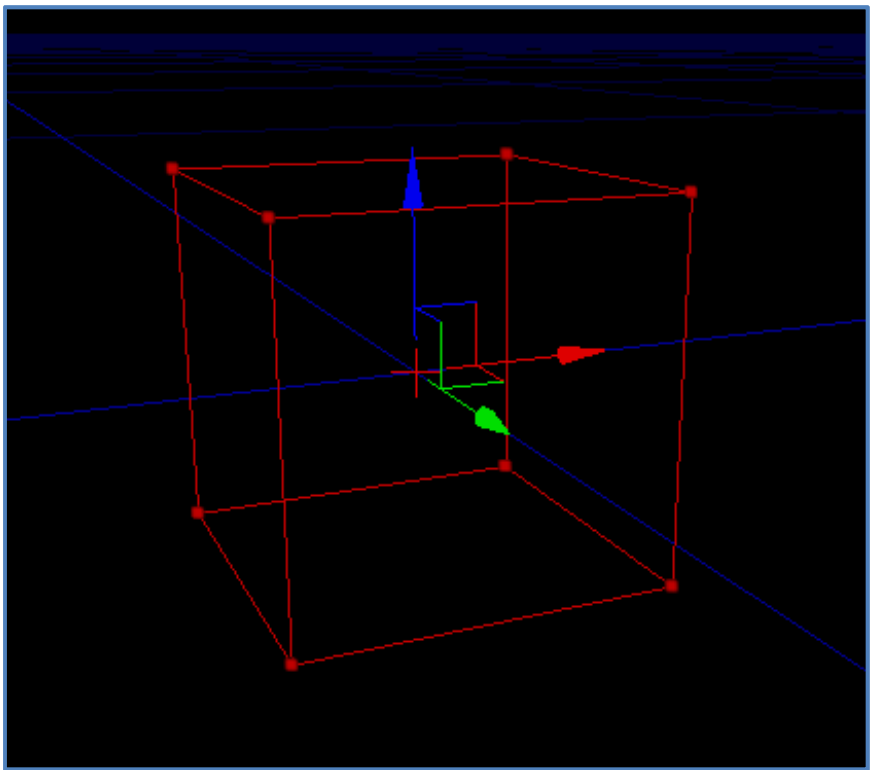


Imagen 22:Traslation Mode.

Mediante estas flechas se podrá modificar la posición de la forma en cada uno de los ejes. Cuando se sitúa el ratón sobre alguna flecha, ésta cambia de color y aparece el símbolo de movimiento.

Para poder realizar el movimiento hay que mantener presionado el botón izquierdo del ratón sobre la flecha del eje en que se quiere mover y desplazar el ratón en el movimiento que se quiere realizar.

De la manera comentada, el movimiento se realiza individualmente en cada uno de los ejes, pero hay otra opción de poder mover el objeto, que es realizarlo en dos ejes a la vez. Para ello se tienen enlaces entre cada par de flechas que situando el ratón sobre ellos y pulsando el botón izquierdo del ratón se puede realizar el movimiento en los dos ejes.

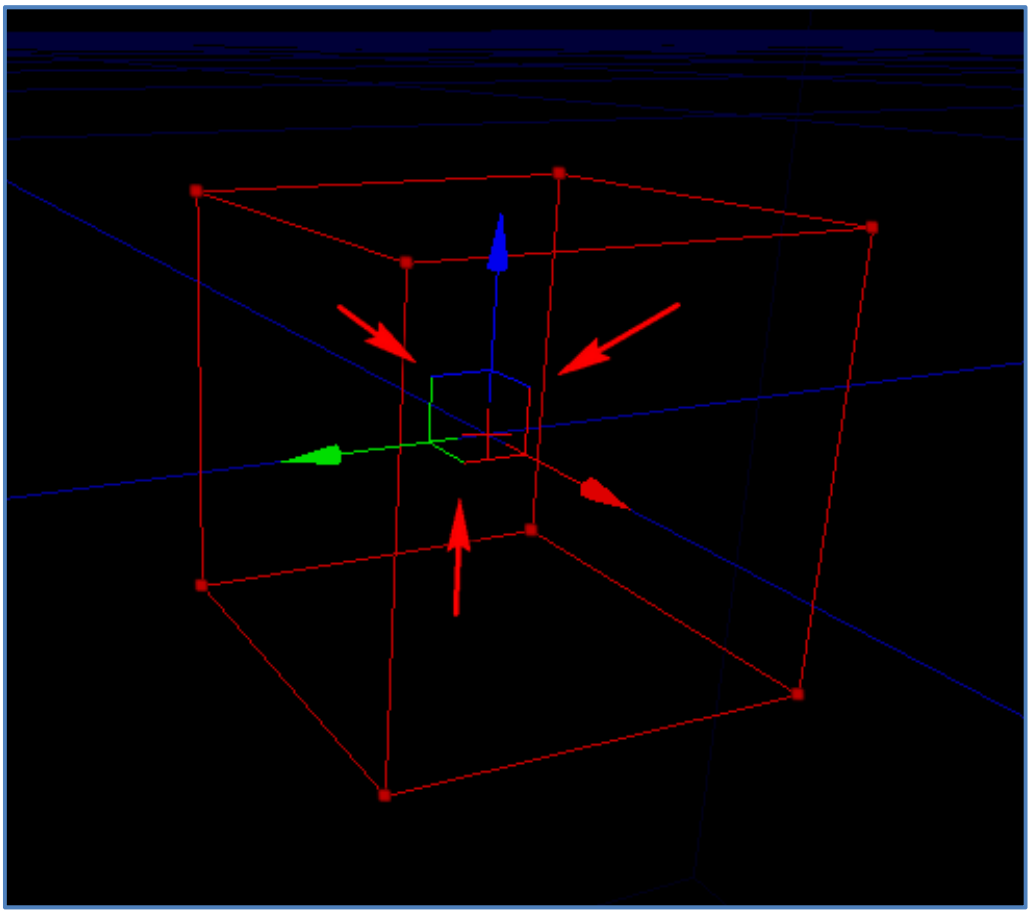


Imagen 23:Localización de traslado objeto en 2 ejes.

Otra opción de transladar el objeto es indicando el valor de sus coordenadas ó el valor de su modificación en cada uno de los ejes.

Pinchando con el botón derecho del ratón sobre el icono, Traslation Mode, se mostrará una nueva ventana, imagen 24, donde se podrá indicar el valor en cada uno de los ejes.

Imagen 24:Traslation Mode insertando valores.

En esta nueva ventana se podrá indicar si los valores dados en los ejes son de manera absoluta o relativa mediante el chekpoint *"Relative"*; y para realizar los cambios sólo hay que presionar el botón *"Apply"*.

- <u>Modificar tamaño</u>
  La modificación del tamaño de la forma puede realizarse de manera uniforme, en todos los ejes de coordenadas, ó modificar el tamaño en cada uno de los ejes independientemente.

  - Uniformemente.- En la barra de herramientas superior hay que tener activado el modo *"Scaling Mode"* pinchando en el icono .

    Con el modo activado y seleccionando la forma (pinchando con el botón izquierdo del ratón en cualquier arista de color rojo) aparecerán los elementos en cada eje que permitirán modificar el tamaño.

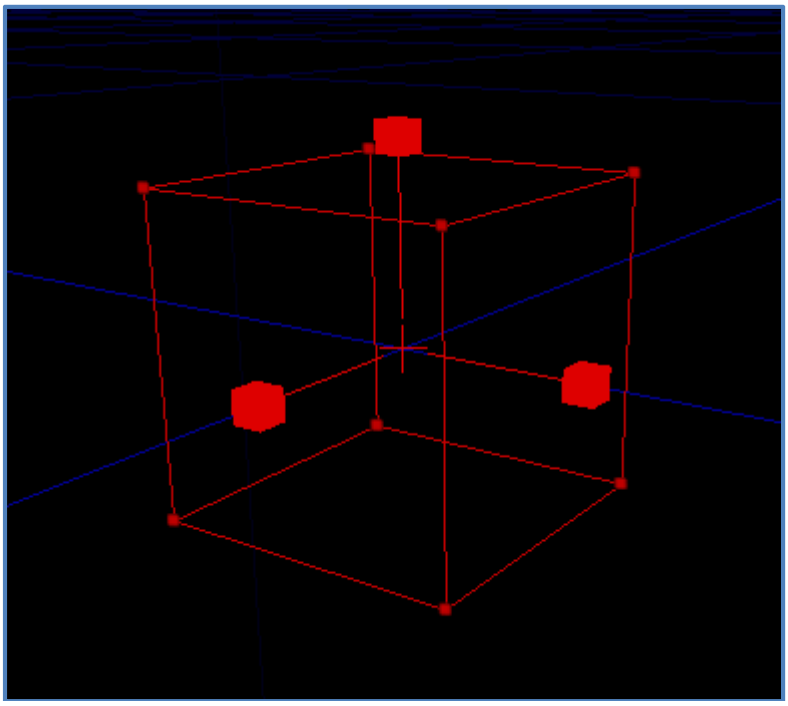


Imagen 25:Scaling Mode uniforme.

    Para modificar el tamaño hay que situar el ratón sobre cualquiera de estos elementos y manteniendo pulsado el botón izquierdo, desplazar el ratón hasta tener el tamaño deseado.

    En este caso el elemento utilizado para modificar el tamaño no es relevante, ya que como la modificación es uniforme se puede elegir cualquiera.

  - No uniformemente.- En la barra de herramientas superior hay que tener activado el modo *"Non-Uniform Scaling Mode"* pinchando en el icono .

    Con el modo activado y seleccionando la forma (pinchando con el botón izquierdo del ratón sobre cualquier arista de color rojo) aparecerán los elementos en cada eje que permitirán modificar el tamaño del objeto independientemente en cada eje.

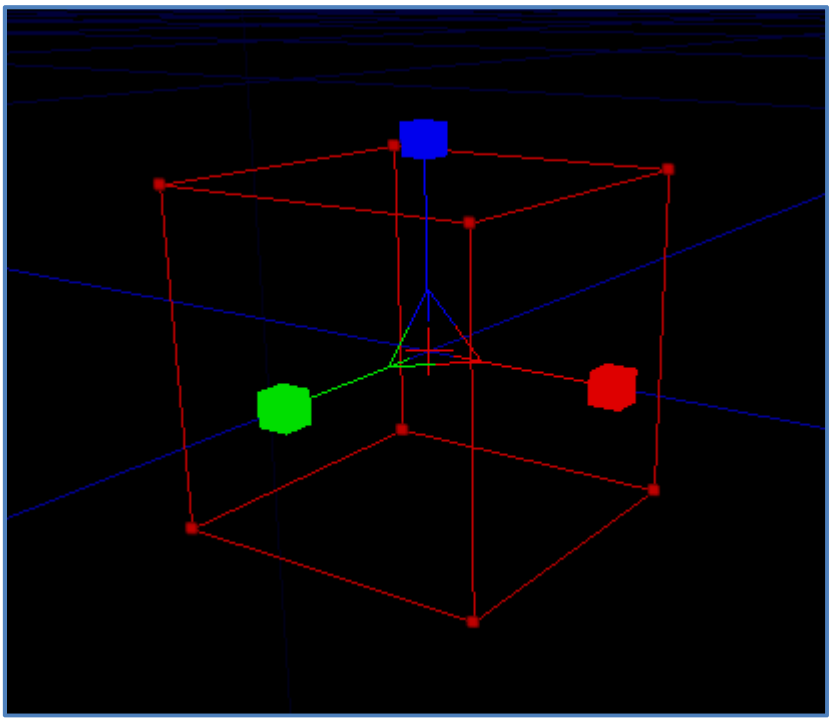


Imagen 26:Scaling Mode no uniforme.

Cada uno de los elementos que aparecen, mostrados en la imagen 26, permiten modificar el tamaño de la forma en cada uno de los ejes. Cuando se sitúa el ratón sobre algún elemento, éste cambia de color y aparece el símbolo de movimiento.

Para modificar el tamaño de la forma hay que mantener presionado el botón izquierdo del ratón sobre el elemento del eje y desplazar el ratón para ir modificando el tamaño hasta obtener el deseado.

De la manera comentada, la modificación del tamaño se hace solamente en cada uno de los ejes, pero se tiene otra opción de modificar el tamaño en dos ejes de coordenadas a la vez. Para ello se tienen enlaces entre cada par de ejes de coordenadas que permiten realizar la modificación en los dos ejes a la vez.

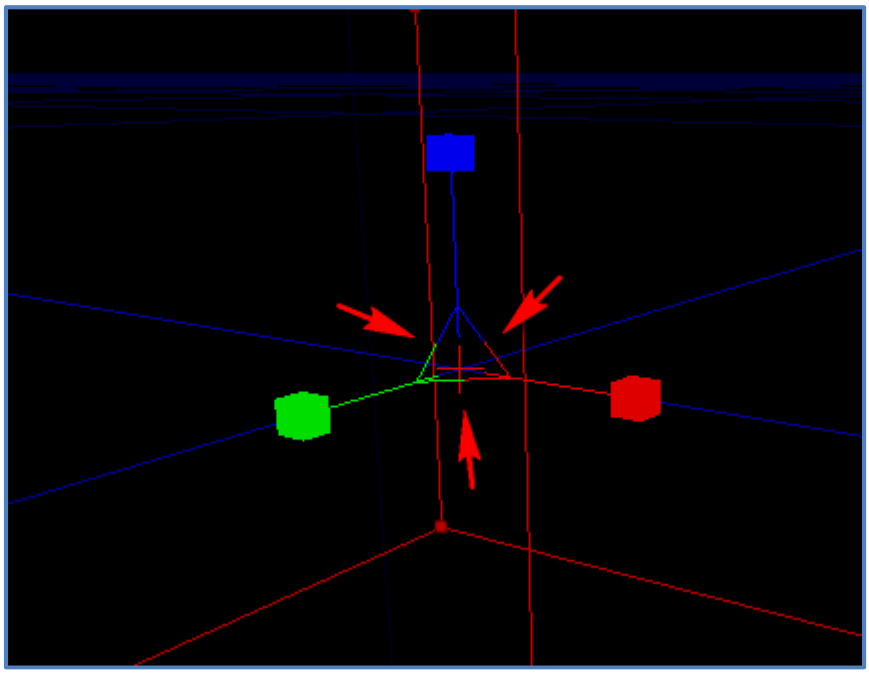


Imagen 27:Localización Scaling Mode en 2 ejes.

El editor permite realizar el escalado de las formas de otra forma diferente, indicando la escala que se quiere dar a cada uno de los ejes de coordenadas.

Pinchando con el botón derecho del ratón sobre cualquiera de los iconos ,*"Non-Uniform Scaling Mode"*, ó , *"Scaling Mode"*, se mostrará una nueva ventana, mostrada en la imagen 28, donde se podrá indicar la escala en cada uno de los ejes.

Imagen 28:Scaling Mode insertando valores.

Mediante el chekpoint que se muestra se puede indicar si la escala dada en cada eje se realiza de forma absoluta o relativa; y para realizar los cambios simplemente hay que pulsar sobre el botón *"Apply"*.

- <u>Girar forma</u>

  Otra opción que se puede realizar en la forma es poder girarla en cada uno de los ejes de coordenadas.

  En la barra de herramientas superior hay que tener activado el modo *"Rotation Mode"* pinchando en el icono . Con el modo activado y seleccionando la forma (pinchando con el botón izquierdo del ratón sobre cualquier arista de

color rojo) aparecerán unas circunferencias de distinto color en cada uno de los ejes de coordenadas, mostradas en la imagen 29.

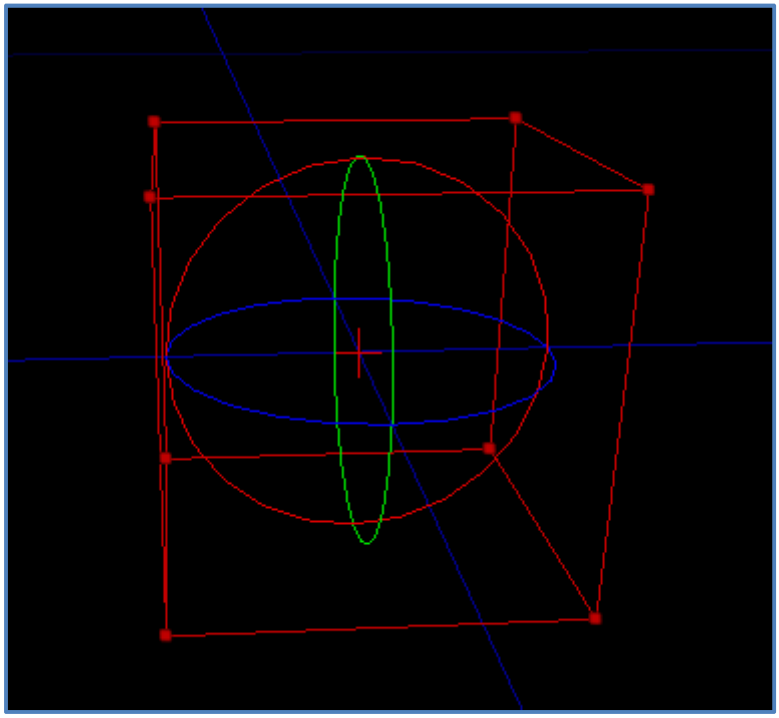


Imagen 29:Rotation Mode

Cada una de las circunferencias que aparecen permite girar la forma en cada uno de los ejes. Cuando se sitúa el ratón sobre alguna circunferencia, ésta cambia de color y aparece el símbolo de movimiento.
Manteniendo presionado el botón izquierdo del ratón y realizando un movimiento del ratón, se puede ir girando sobre el eje de coordenadas elegido hasta obtener el giro deseado.

En esta modificación hay que ir de eje en eje de coordenadas para encontrar la posición que se quiere.

Otra opción que se puede utilizar es indicando el valor del ángulo que se quiere dar a la forma en cada uno de los ejes de coordenadas.

Pinchando con el botón derecho del ratón sobre el icono, *"Rotation Mode"*, se mostrará una nueva ventana, imagen 30, en el editor donde se podrá indicar el valor del ángulo en cada uno de los ejes.

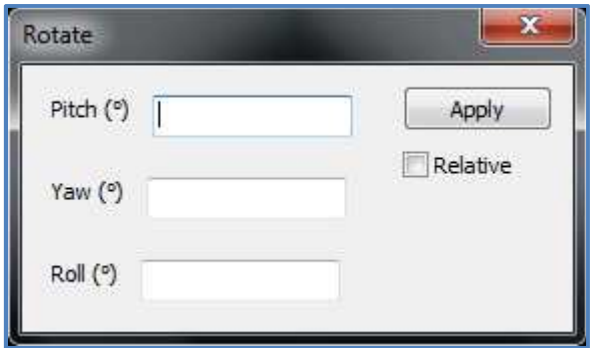


Imagen 30:Rotation Mode insertando valores.

Con el chekpoint se indica si los ángulos dados son de forma relativa o absoluta y para realizar los cambios simplemente hay que presionar en el botón *"Apply"*.

3.4.2. Modificar la figura de la forma/modelo.

No siempre se querrá realizar un objeto con el modelo de un hexágono, por ejemplo se querrá tener la forma de un cilindro, o cualquier otra forma. Para ello se utilizarán los comandos que se encuentran en la parte izquierda de la pantalla, mostrados en la siguiente imagen:

Imagen 31:Formas posibles del modelo.

Mediante dichos comandos se puede modificar la forma del modelo para ir creando distintos objetos. Como se puede apreciar se puede poner la forma de un hexágono , cono , cilindro , plano y esfera .

Otras formas que se pueden realizar son las de crear escaleras. Se pueden crear escaleras normales y escaleras en espiral y . En este caso se indicará la altura que se quiere que tenga la escalera, la anchura, el número de escalones que se quiere y la altura entre ellos, entre otras especificaciones.

La forma Volumetric es utilizada para acoplar varios planos juntos.

Modificar características de la forma/modelo.

Se puede acceder a las características de cada una de las formas pulsando con el botón derecho del ratón sobre el icono de la forma requerida, apareciendo un submenú en el que se mostrarán sus características. En este nuevo submenú se permite modificar el valor de dichas características dando el valor deseado.

- HEXÁGONO:

En este submenú se pueden introducir los valores de los 3 ejes de coordenadas del hexágono que se quiere crear. Además se puede introducir el espesor de la pared (Wall Thickness) e indicar si se quiere el objeto hueco (Hollow) ó teselado (Tessellated).

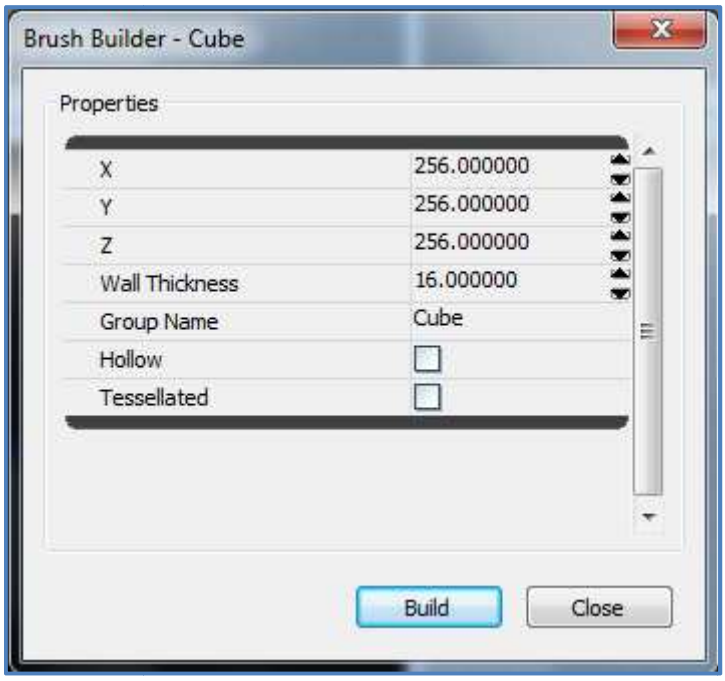


Imagen 32:Propiedades forma hexágono.

- CONO:

En este submenú se puede indicar el valor en el eje Z al igual que el radio interior (Inner Radius) y el radio exterior (Outer Radius). Además se pueden indicar los lados (Sides) que se quieren tener, si se quiere hueco (Hollow) y si se quiere alinear con los ejes (Align To Side).

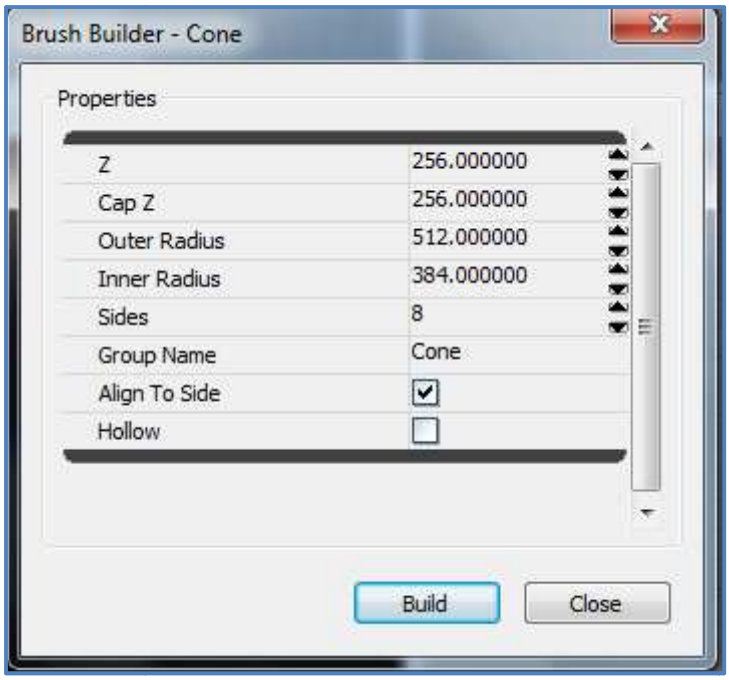


Imagen 33:Propiedades forma cono.

- CILINDRO:

En este caso las características que se pueden introducir son exactamente las mismas que las indicadas en el anterior punto, debido a la similitud de las dos formas.

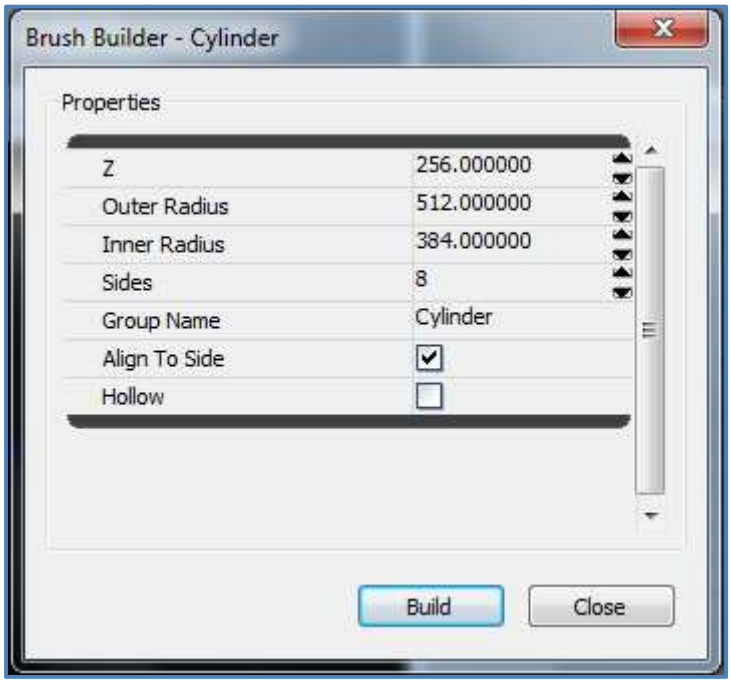


Imagen 34:Propiedades forma cilindro.

- PLANO:

Las características que se pueden introducir en este caso son los valores de los ejes X e Y del plano. A su vez también se pueden introducir el número de segmentos que se quiere tener en cada uno de los lados, tanto en el eje X (XSegments) como en el eje Y (YSegments). Otra característica a elegir es en qué eje se quiere tener el plano. Para ello hay que desplegar el menú de la etiqueta *"Axis"* y seleccionarlo.

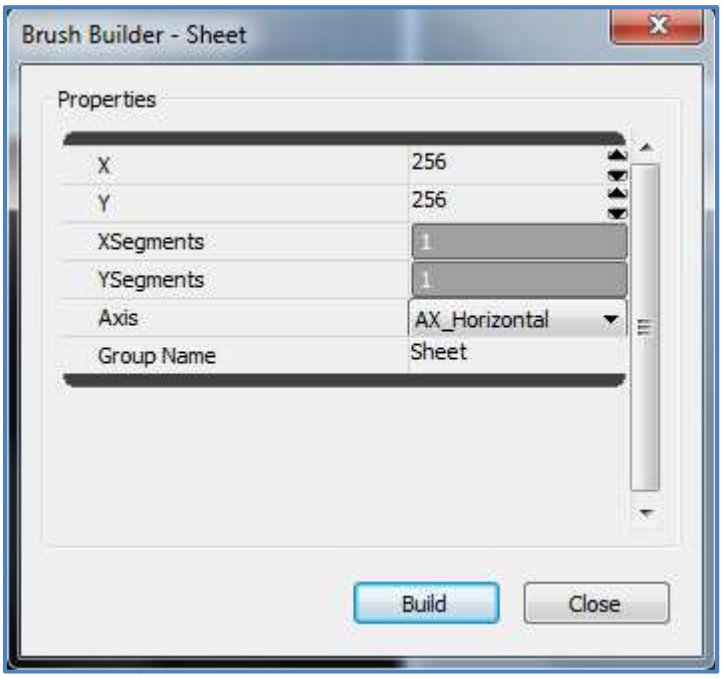


Imagen 35:Propiedades forma plano.

- ESFERA:

Para realizar el modelo de una esfera, se puede introducir el radio y el nivel de extrapolación que quiere tener. Cuanto mayor sea este nivel, mayor número de caras tendrá, siendo el modelo más realista.

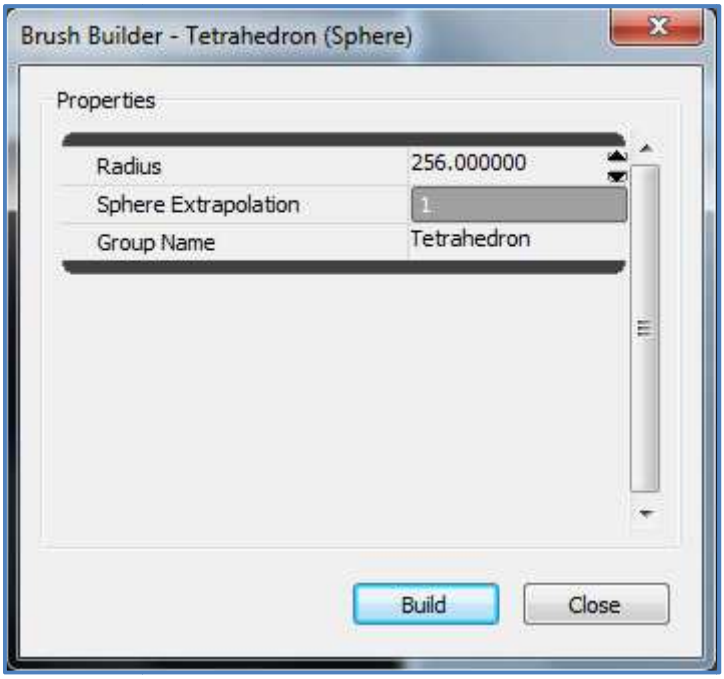


Imagen 36:Propiedades forma esfera.

- ESCALERAS LINEALES:

Mediante este menú se puede introducir el valor de la longitud de cada escalón (Step Length), la altura (Step Height) y la anchura (Step Width). Además se puede indicar el número de escalones que se quiere (Num Steps).

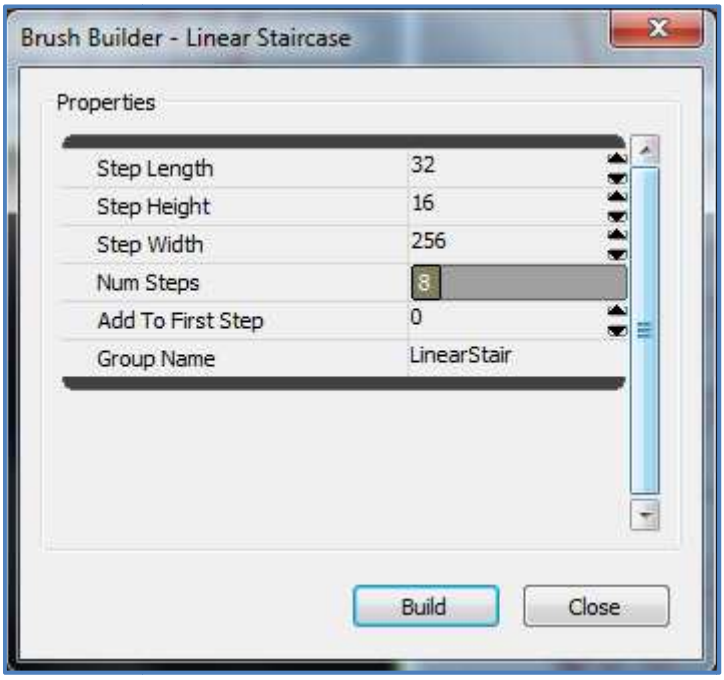


Imagen 37:Propiedades forma escaleras lineales.

- ESCALERAS CURVADAS:

Al ser éste un modelo de escaleras curvadas hay que indicar el radio de curvatura (Inner Radius) y el ángulo (Angle Of Curve). Como en el caso anterior, se puede indicar la altura (Step Height) y la anchura (Step Width) de cada uno de los peldaños, al igual que el número de éstos (Num Steps).

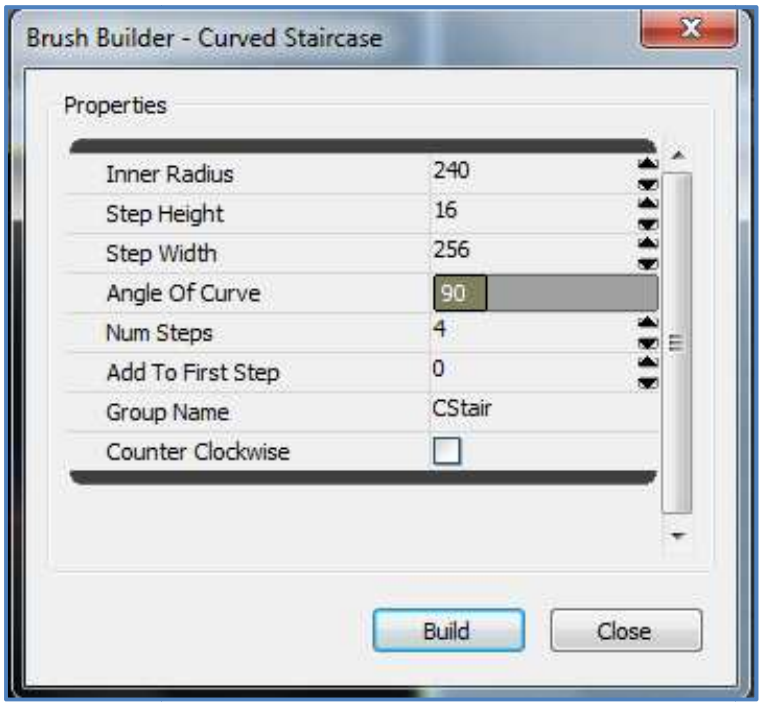


Imagen 38:Propiedades forma escaleras curvadas.

- ESCALERAS EN ESPIRAL:

Al igual que en los dos casos anteriores se puede proporcionar el valor de las características básicas de una escalera. Lo diferente en este caso es el indicar el espesor de peldaño (Step Thickness) y el número de peldaños que se quiere en 360 grados.

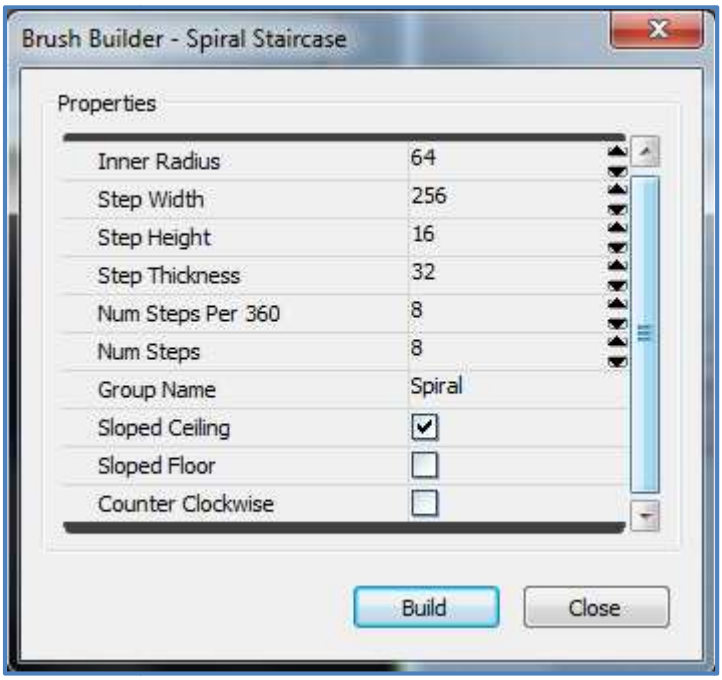


Imagen 39:Propiedades forma escaleras en espiral.

- FORMA VOLUMETRIC:

Dicha forma es utilizada para realizar formas no corrientes mediante acoplamiento de varios planos verticales en el eje Z. Puede ser usado para crear efectos como humo, fuego, etc. Se puede indicar el valor en el eje Z, el radio (Radius) y el número de hojas/planos (Sheets) que debe tener la forma.

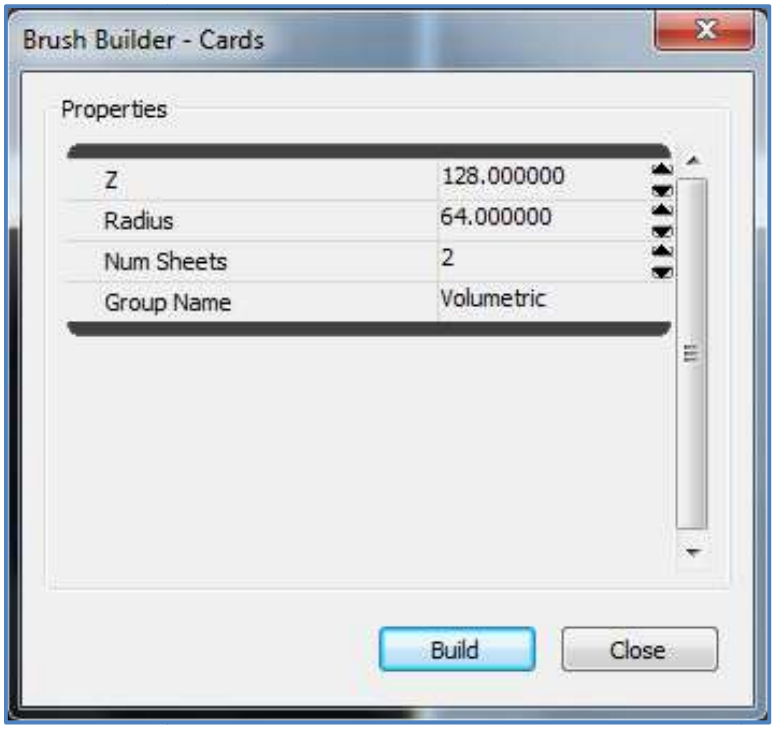


Imagen 40:Propiedades forma Volumetric.

En todos los menús contextuales mostrados en cada una de la formas se puede modificar el valor que se desee y pulsando en el botón *"Build"* se crearía el modelo dado en las vistas del editor.

Cuando la forma/modelo tiene las propiedades deseadas, puede ser añadida al nivel. La manera de realizarlo se muestra en el siguiente punto, *"Añadir la forma/modelo al nivel"*.

3.4.3. Añadir la forma/modelo al nivel.

La forma/modelo puede ser añadida en el nivel para que sea un objeto consistente. Según el modo seleccionado en la creación del nuevo nivel serán mostradas las caras internas o externas.

Hay ocasiones que el objeto que se quiere realizar es la intersección, la adicción ó la sustracción entre varias formas. Un ejemplo de dicha circunstancia puede ser el crear el hueco para unas puertas en una pared ó el hueco para unas ventanas.

Para ello se tienen las siguientes herramientas, mostradas en la imagen 41.

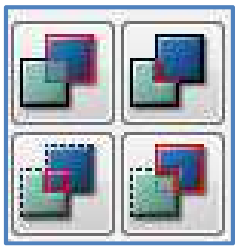


Imagen 41:Herramientas para añadir la forma al nivel.

Se van a explicar cada una de ellas:

- Herramienta *"Add"* .- Esta herramienta creará el objeto definido por el modelo. En el caso de solapamiento con otros objetos, sólo se mostrará la parte del modelo que no se solapa con el otro objeto.

- Herramienta *"Subtract"* .- Este caso es lo contrario a lo comentado en la herramienta "Add". En caso de solapamiento sólo se creará la parte del modelo que está solapada con el otro objeto.

- Herramienta *"Intersect"* .- Esta herramienta es utilizada para obtener la intersección entre dos objetos que se solapan.

- Herramienta *"Deintersect"* .- Mediante esta herramienta se obtiene la parte del modelo que no se solapa con otro objeto. Es una herramienta prácticamente igual que "Add", sólo que en este caso sí se crean las caras que hacen contacto con el otro objeto.

Cuando se añade la forma al nivel, el material que posee es el que tiene por defecto Unreal Editor, de cuadros blancos y azules, como se muestra en la imagen 42.

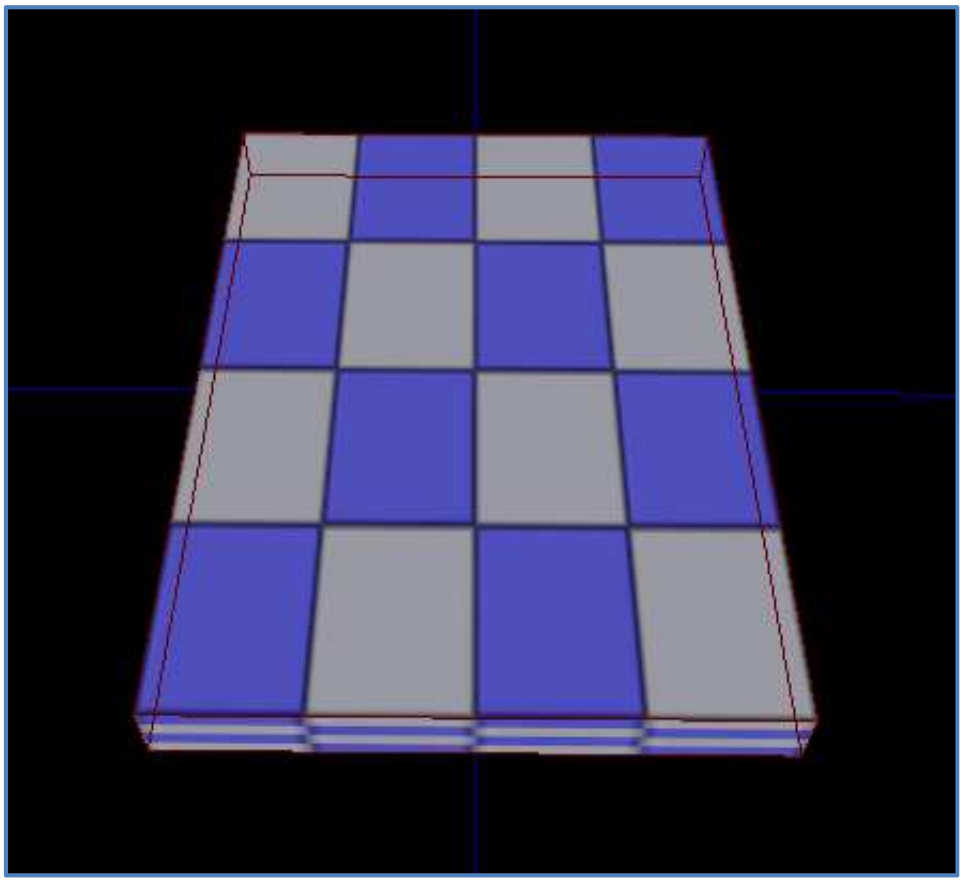


Imagen 42:Ejemplo material por defecto.

A los objetos creados se les puede acoplar cualquier material proporcionado por el Content Browser.

Acoplar material a la forma agregada.

Se tiene acceso a todos los materiales del editor a través de la herramienta Content Browser. Para abrir dicha herramienta hay que pulsar en el icono de la barra de herramientas superior.

Con la herramienta abierta, se puede utilizar el filtro de la parte superior para obtener todos los materiales que se pueden utilizar. Para seleccionar el material que se quiere acoplar sólo hay que pulsar con el botón izquierdo del ratón sobre el material quedando éste seleccionado. El material que ha sido seleccionado quedará enmarcado con un cuadro amarillo alrededor para diferenciarlo.

Con el material seleccionado hay dos maneras de acoplarlo al objeto:

- Arrastrando: El material puede ser acoplado arrastrándolo directamente sobre el objeto en cualquiera de las vistas del editor.

  Para ello hay que pulsar con el botón izquierdo sobre el material y sin dejar de pulsar, desplazar el ratón hacia el editor y dejar de pulsar sobre la forma a la que se quiere acoplar el material.

  Para indicar dónde se puede acoplar el material seleccionado, cuando se pase el puntero del ratón por encima, en la parte inferior de éste aparecerá el símbolo

indicando que se puede acoplar. En caso contrario aparecerá el símbolo de prohibido para indicar que no se puede acoplar.

- Mediante menú contextual: Es necesario ir a las vistas del editor.
  Se puede desplazar la pantalla de la herramienta Content Browser hacia un lado para poder tener acceso a las vistas o cerrarla. En los dos casos el material seguirá seleccionado.

  En la vista *"Perspective"* del editor hay que pulsar con el botón derecho del ratón sobre la forma a la que se quiere acoplar el material, con lo que aparecerá el siguiente menú contextual, mostrado en la imagen 43.

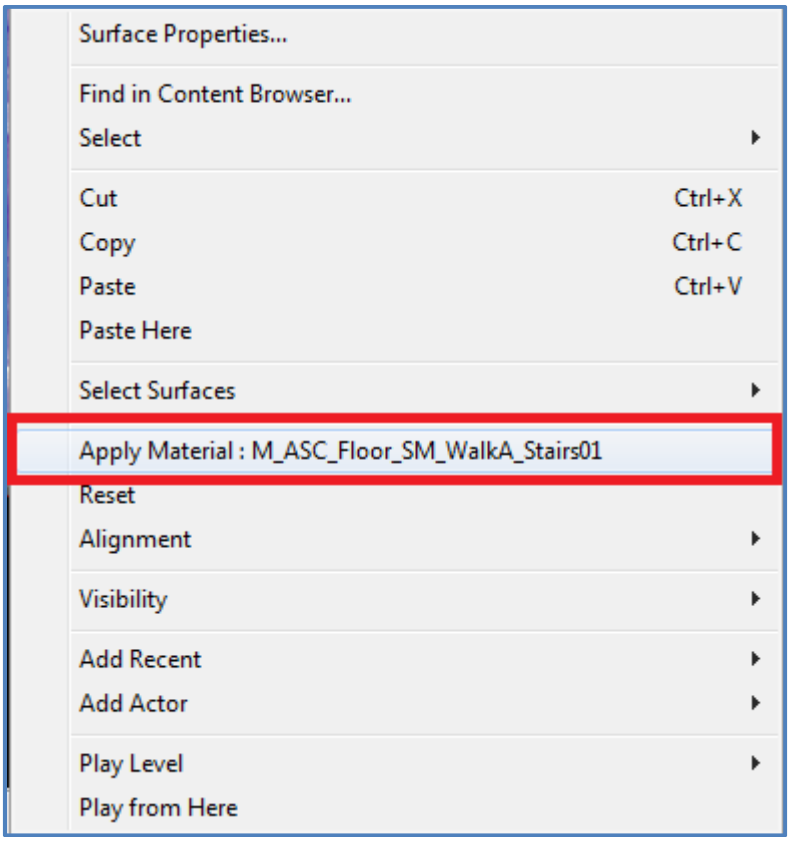


Imagen 43:Menú para aplicar material.

El material será acoplado en toda la cara del objeto seleccionada, teniendo varias opciones de mostrar este material:

- Extender el material a toda la cara seleccionada.
  En este caso, si el tamaño del material es inferior a la forma, el material puede mostrarse excesivamente distorsionado presentando una imagen no real.

- Usar el tamaño propio del material.
  El material será replicado tantas veces como sea necesario hasta rellenar toda la cara.

Para poder seleccionar el tipo de presentación del material en el objeto hay que seguir los siguientes pasos:

- Pulsar con el botón derecho del ratón sobre la forma con el material acoplado a modificar y en el menú contextual mostrado pinchar en *"Surface Properties…"*.

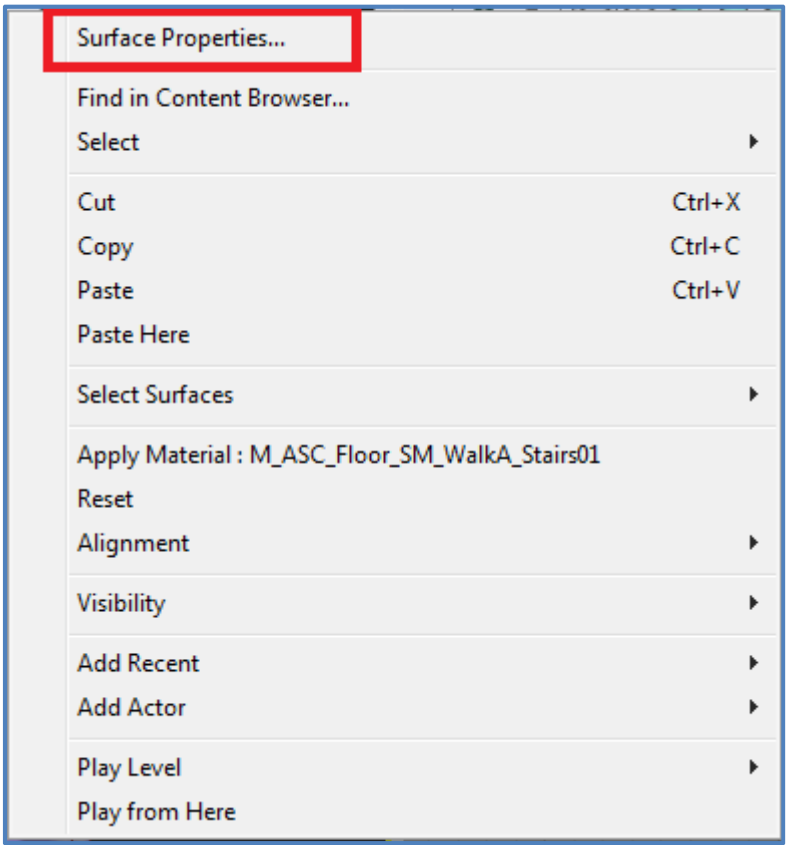


Imagen 44:Surface Properties.

- En la nueva ventana mostrada, en la sección *"Alignment"*, se tienen todas las opciones que se pueden tener a la hora de mostrar el material.

  Seleccionando el deseado y pulsando en el botón *"Apply"*, la presentación del material será modificada según la opción.

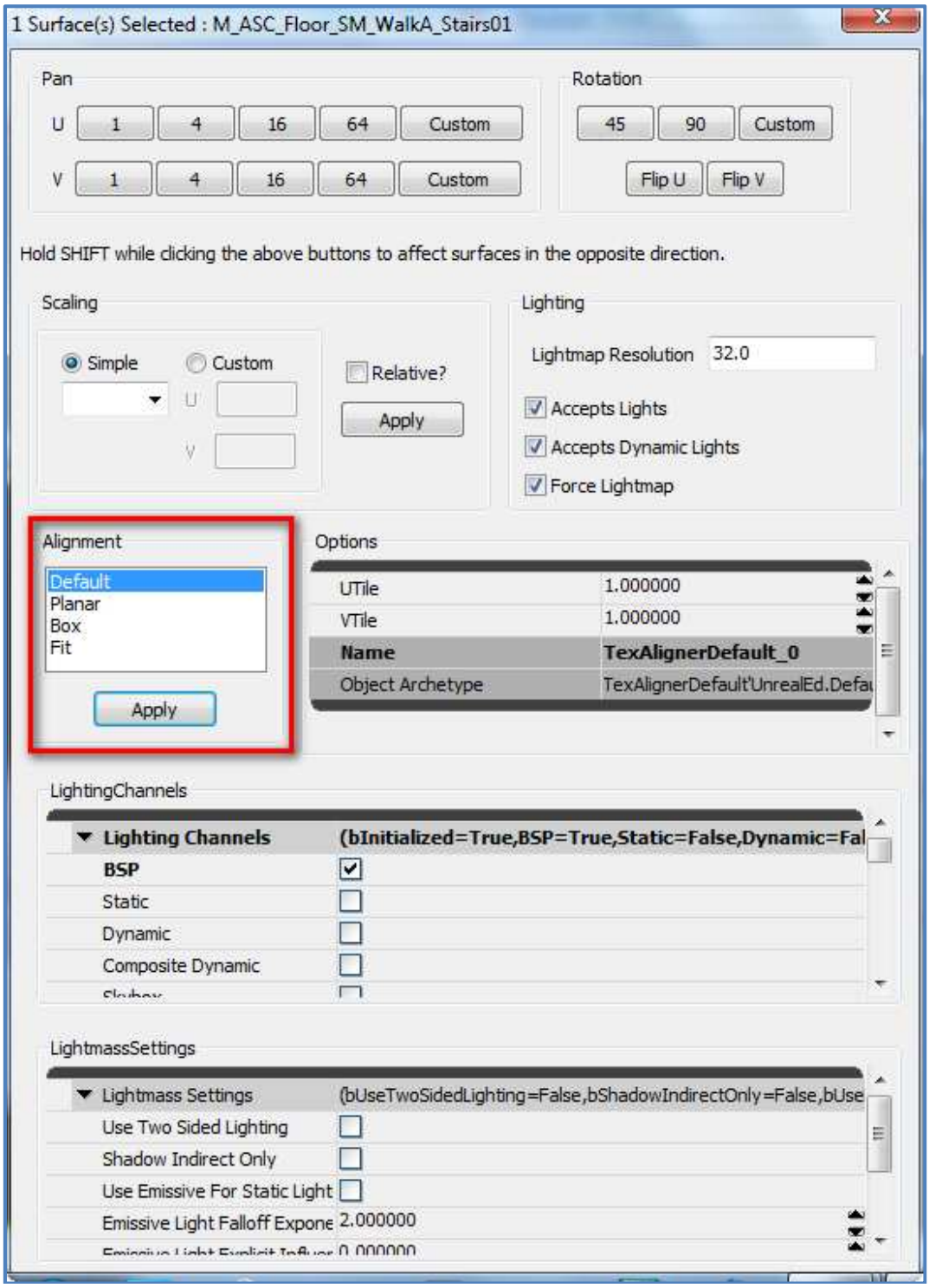


Imagen 45:Opciones "Alignment" del material en objeto.

3.5. Crear terreno.

Los niveles exteriores pueden ser creados mediante una herramienta propia que permite definir terrenos. Lo primero que hay que realizar es obtener la plantilla sobre la cual se van a ir realizando las modificaciones siendo ésta un plano liso:

Tools → new Terrain → Se indica la posición de los ejes X, Y y Z → Se indica la apariencia.

Para poder modificar esta plantilla se pulsa en el icono (Terrain Editing Mode) y se obtendrá la herramienta para poder modificar el terreno (Unreal TerrainEdit). En la imagen 46 se presenta la interfaz de esta herramienta de creación de terrenos.

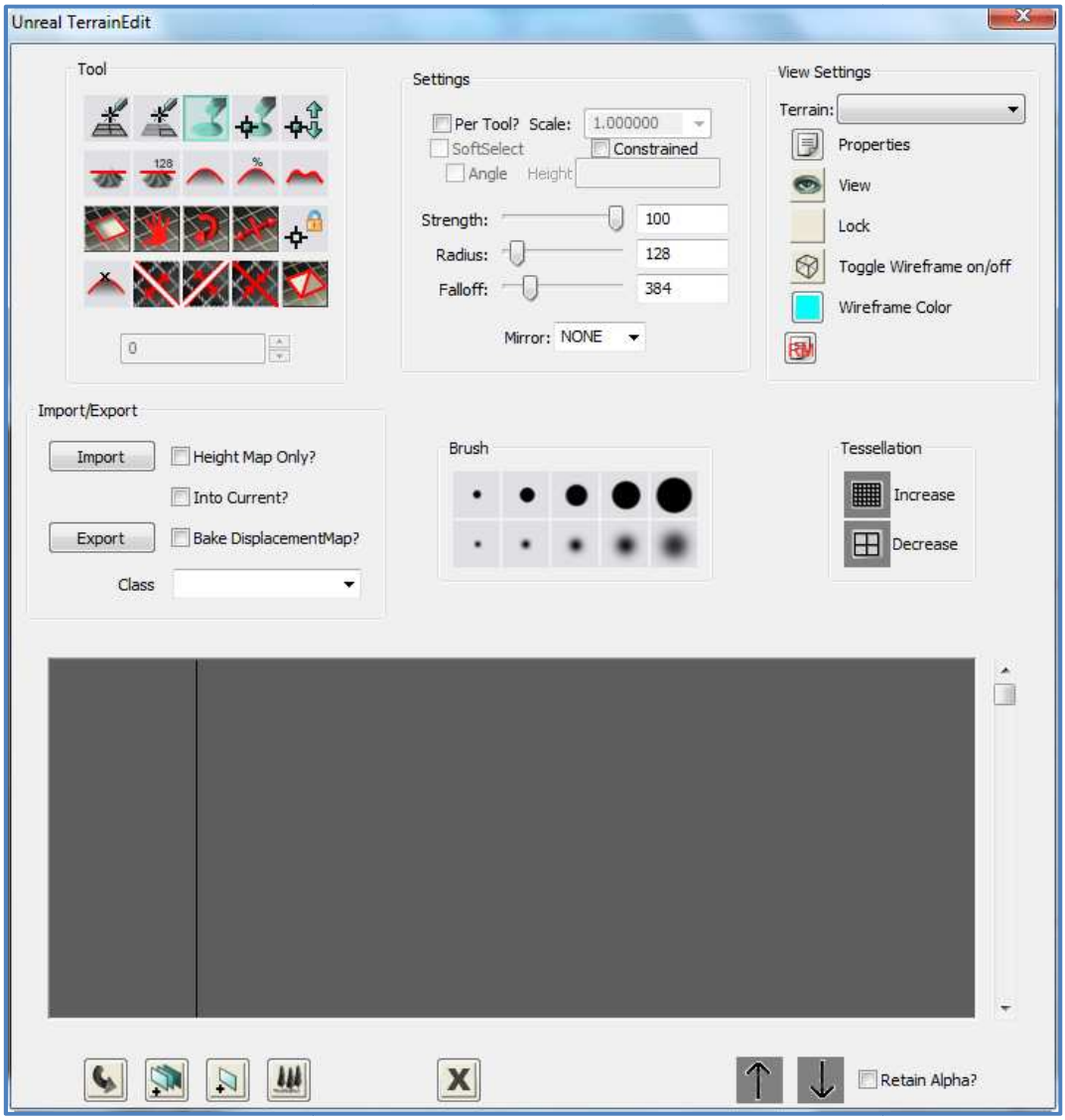


Imagen 46:Interfaz Unreal TerrainEdit.

Hay que tener en cuenta que para poder realizar la acción de edición hay que tener presionada la tecla 'Ctrl' mientras se mantiene pulsado el botón izquierdo del ratón.

Añadir textura/material al terreno.

Con la herramienta *"Unreal Terrain Edit"* abierta observaremos que en la parte inferior aparece el terreno que se ha creado.

Para aplicar un material primero se debe obtener el elemento*"Terrain Layer Setup"*. Para ello, sobre el terreno damos al botón secundario del ratón y se elige *"new Terrain Setup Layer"*. Se inserta el nombre y se pulsa en *"Ok"*.

Se puede observar cómo en el Content Browser, se ha creado un nuevo objeto de tipo Terrain Layer Setup. En las propiedades de este nuevo objeto hay que indicar el material que se quiere aplicar. Para obtener el menú de las propiedades:

Botón secundario ratón sobre el objeto → Properties…

En el submenú que aparece se despliega el árbol de etiquetas hasta encontrar la etiqueta *"Material"*. En esta etiqueta se indica el material que se quiere aplicar pero tiene que ser un objeto propio del terreno, indicando cómo crearlo a continuación:

Para ello hay que crear un nuevo objeto *"Terrain Material"*:

Botón secundario sobre "Terrain Layer Setup" en "Unreal TerrainEdit" → "New Terrain Material" → se le da un nombre → OK

Como se ha comentado antes, se crea un nuevo objeto en el Content Browser. Ahora hay que aplicar el material a este objeto creado. Se abre el submenú de propiedades del objeto Terrain Material (botón secundario ratón → properties…). Se despliegan las etiquetas hasta encontrar *"Material"*. En esta etiqueta se especifica el material que se quiere aplicar. Se busca en el Content Browser el material a aplicar y se selecciona. Con el material seleccionado se vuelve al submenú de propiedades y a la derecha de la etiqueta *"Material"* se pulsa sobre el icono . El contenido de la etiqueta se modifica, mostrando ahora el material que previamente se seleccionó.

Una vez que el objeto *"Terrain Material"* contiene la textura que se quiere, se puede asociar al *"Terrain Layer Setup"*:

Botón secundario ratón sobre "Terrain Layer Setup" → en etiqueta "Material" → pulsar el icono (teniendo seleccionado el objeto "Terrain Material" en el "Content Browser").

Con todo esto el terreno creado contendrá la textura deseada.

4. CONTENT BROWSER

La herramienta *"Content Browser"* es con la cual se puede acceder y/o explorar todos los elementos del motor, como objetos que se pueden añadir al mapa (Static Mesh), materiales a acoplar a los objetos, etc.

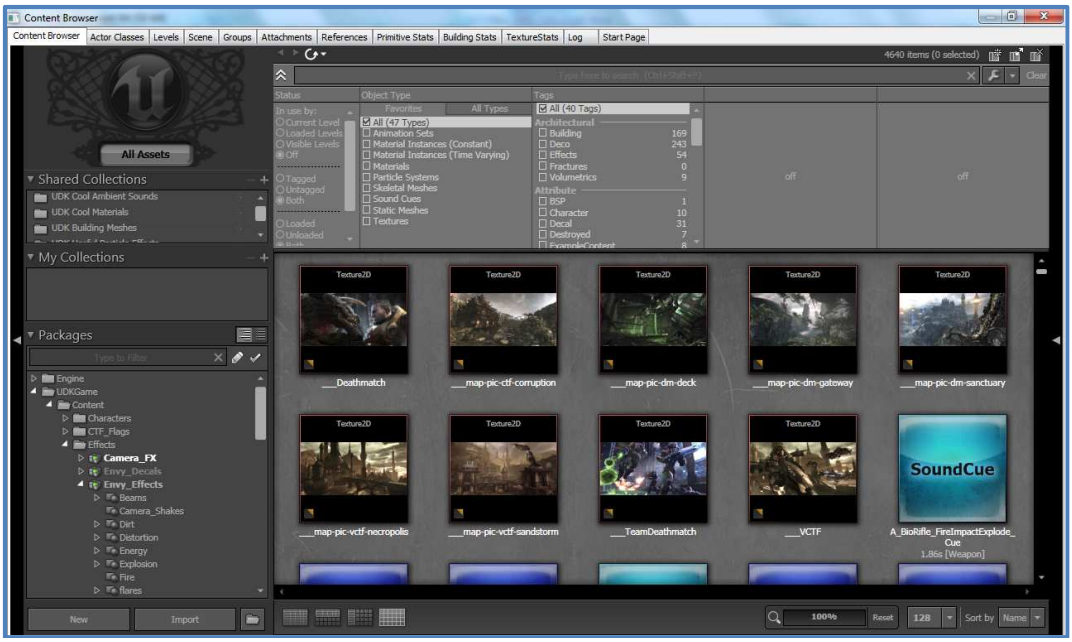


Imagen 47:Interfaz "Content Browser".

Para acceder a dicho explorador hay que pulsar sobre el icono de la barra de herramientas superior del editor abriéndose una nueva ventana mostrando todos los elementos. Esta nueva ventana se muestra en la imagen 47.

Otra forma de poder abrir dicho explorador es mediante la opción *"View"* del menú superior. Al pinchar sobre dicha opción se despliega el submenú:

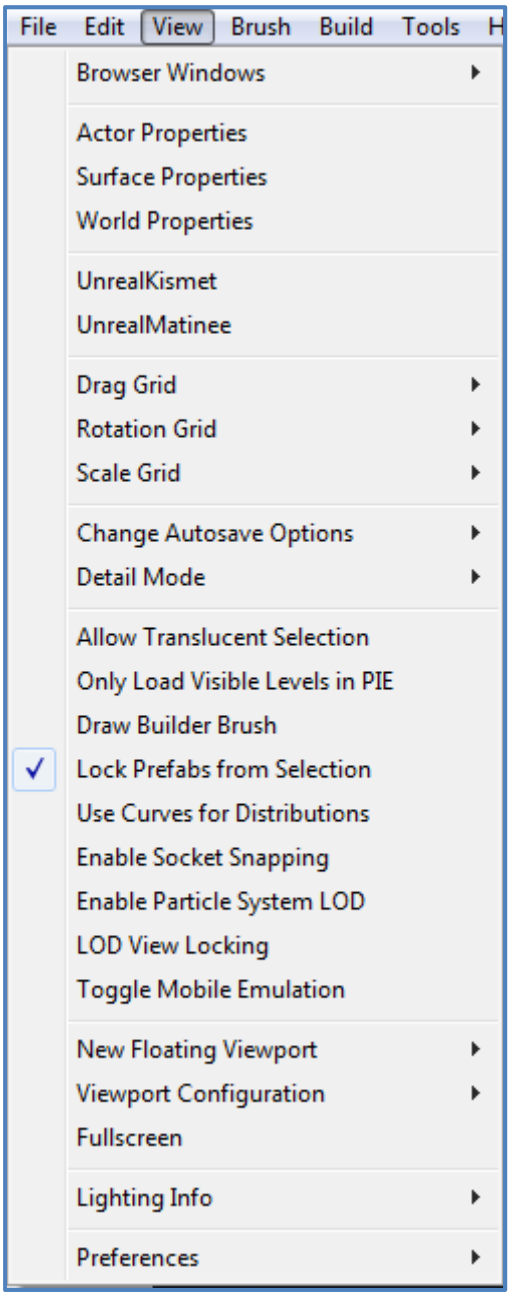


Imagen 48:Menú de la opción "View".

Al desplegar el submenú de *"Browser Windows"* se podrá seleccionar el *"Content Browser"*:

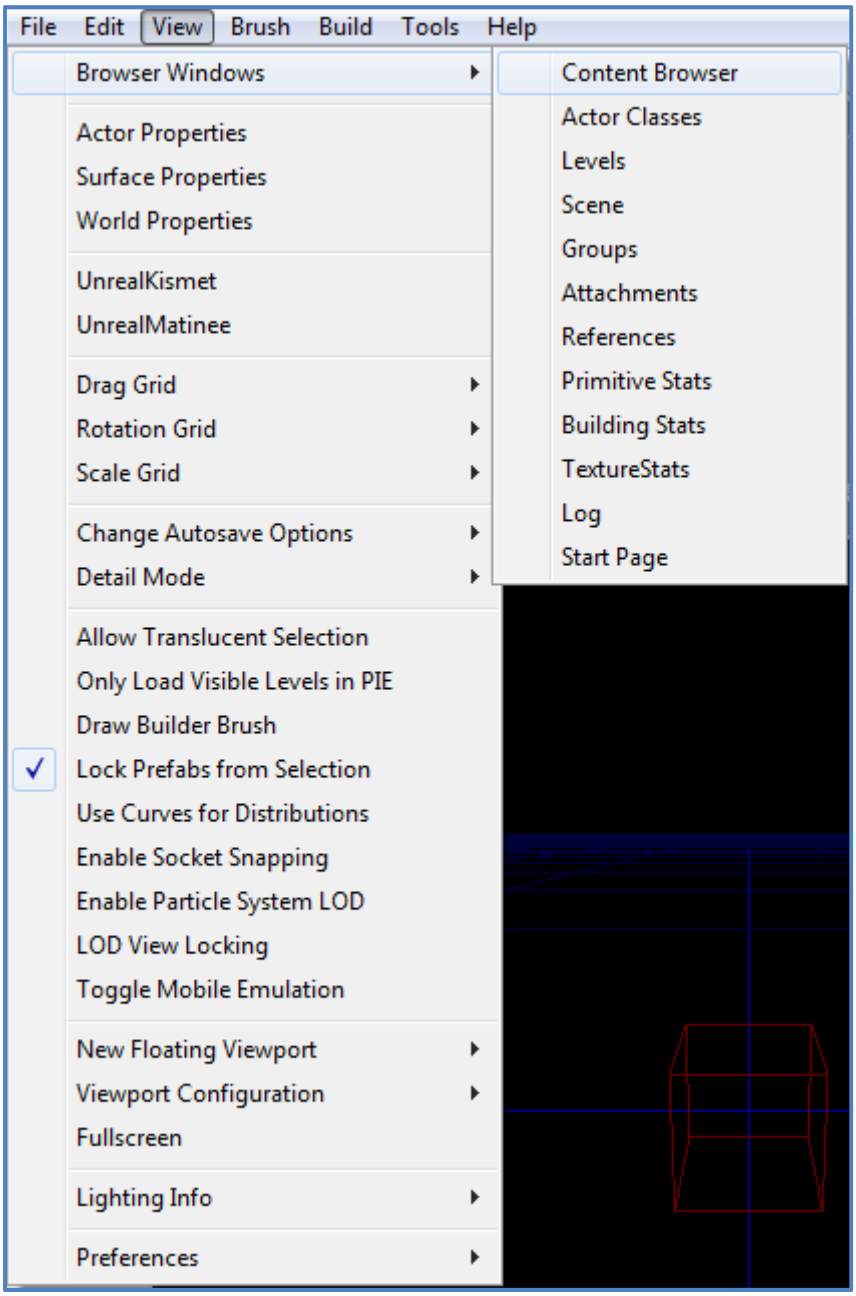


Imagen 49:Menú desplegado de "Browser Windows".

En la parte izquierda se pueden apreciar todos los paquetes que contiene el motor partiendo del principal "*Engine*" ó "*UDKGame*", mostrándose en la imagen 50.

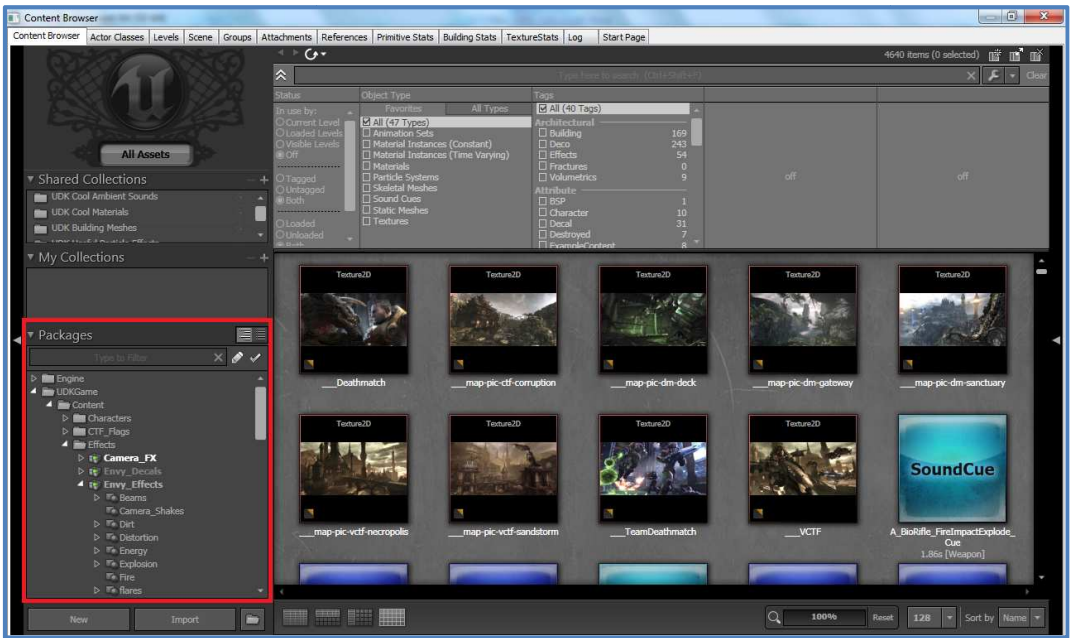


Imagen 50:Jerarquía paquetes en "Content Browser".

En la parte central se muestran todos los elementos que contiene el paquete que esté activo. En la parte superior de esta zona se tiene el filtro para mostrar los elementos que se quieren, al igual que una entrada de texto para poder buscar el elemento que se quiere:

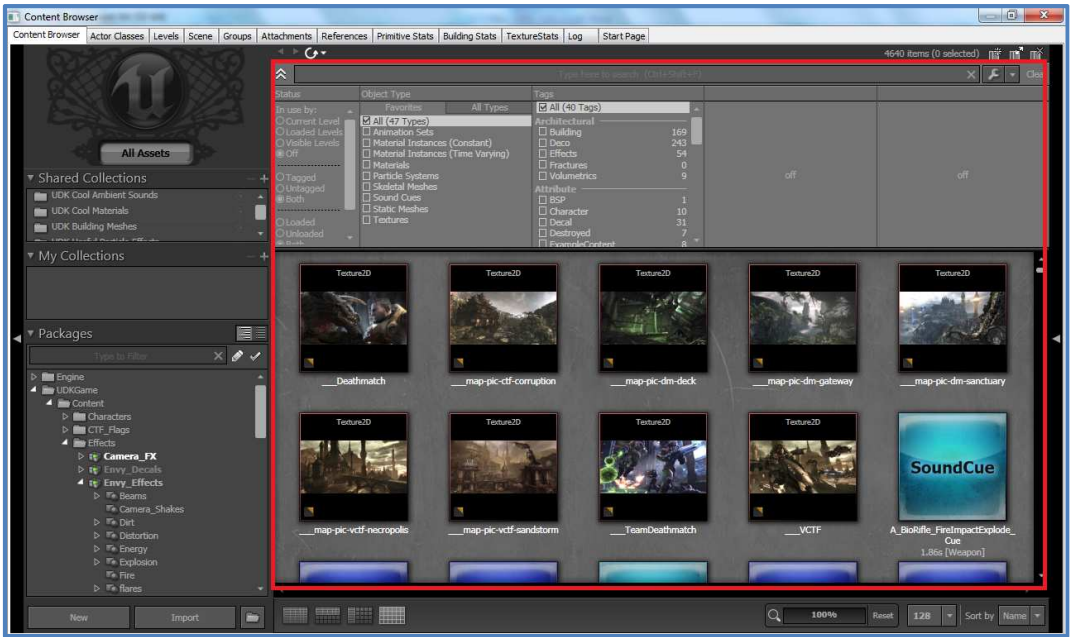


Imagen 51:Elementos mostrados en "Content Browser".

Cuando se realizan cambios en los distintos paquetes, ya sea insertando un nuevo elemento, borrando ó modificando uno existente, hay que volver a guardarlo. Para ello sobre el paquete a guardar se pulsa sobre el botón derecho, desplegándose el siguiente submenú:

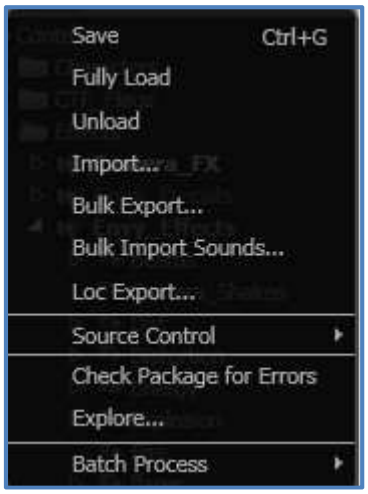


Imagen 52:Menú paquete.

Mediante este submenú se pueden realizar las siguientes acciones sobre el paquete seleccionado:

- Save.- Guardar todos los cambios realizados en el paquete.
- Fully Load.- Guardar completamente todos los archivos que contiene el paquete.
- Unload.- Descargar el paquete seleccionado.
- Import.- Importar algún elemento al paquete.
- Bulk Export.- Exportar todo el contenido del paquete.
- Bulk Import Sound.- Importar todo el contenido referente a sonido.
- Source Control.- Control del código fuente.
- Check Package for Errors.- Comprobación de errors en el paquete.
- Explore.- Explorar el contenido del paquete.
- Batch Process.- Proceso por lotes.

Los distintos elementos de cada paquete mostrados podrán ser seleccionados para su uso. El elemento que ha sido seleccionado quedará enmarcado con un marco amarillo para su localización como se puede observar en la imagen 53.

Mediante la pulsación del botón derecho del ratón sobre el elemento se accederá a su submenú:

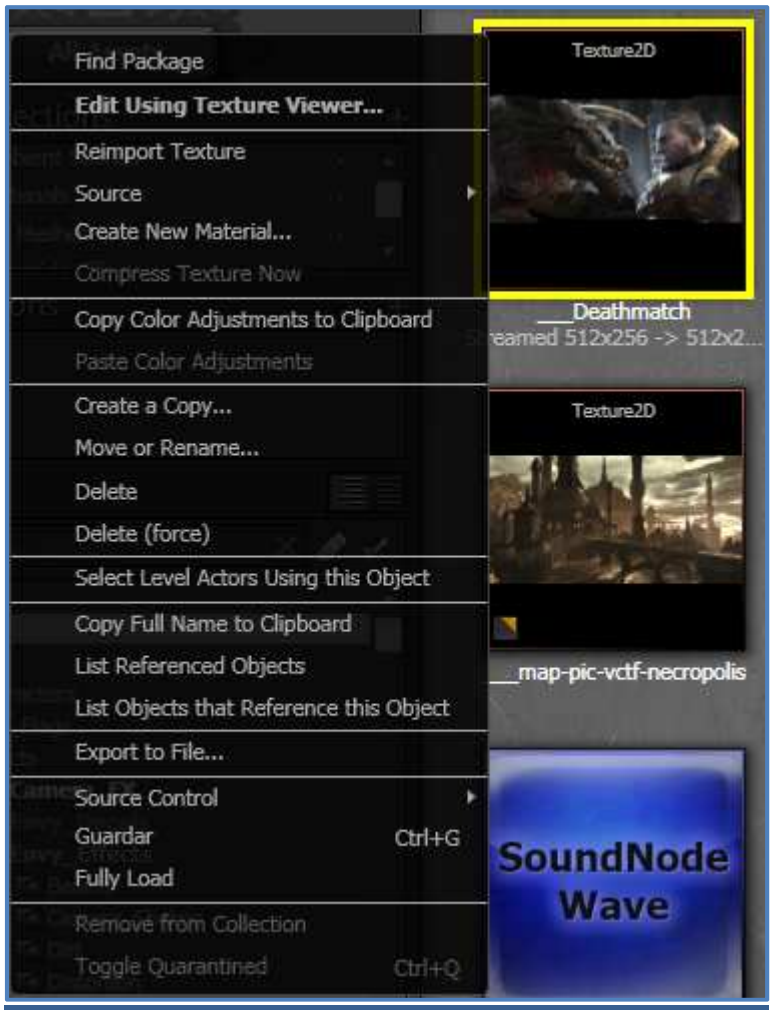


Imagen 53:Ejemplo submenú elemento Texture2D.

Mediante este submenú se podrá guardar el elemento en el caso que haya sido modificado, abrir el editor correspondiente al tipo de elemento, crear una copia, renombrar, mover de carpeta, borrar y demás acciones propias al tipo de elemento que sea.

Cada uno de los elementos seleccionados puede ser editado para modificar sus propiedades. Para poder modificar sus características se debe abrir mediante el editor propio según el tipo que sea. Para abrir el correspondiente editor hay que hacer una pulsación doble con el botón izquierdo del ratón sobre el elemento a editar.

4.1. Importar elemento.

Mediante esta herramienta se pueden crear ó importar todo tipo de elementos nuevos. Para ello, en la parte inferior izquierda se tienen las herramientas necesarias para seleccionar el elemento a importar desde el ordenador o crearlo. En la imagen 54 se muestra la posición de estas herramientas.

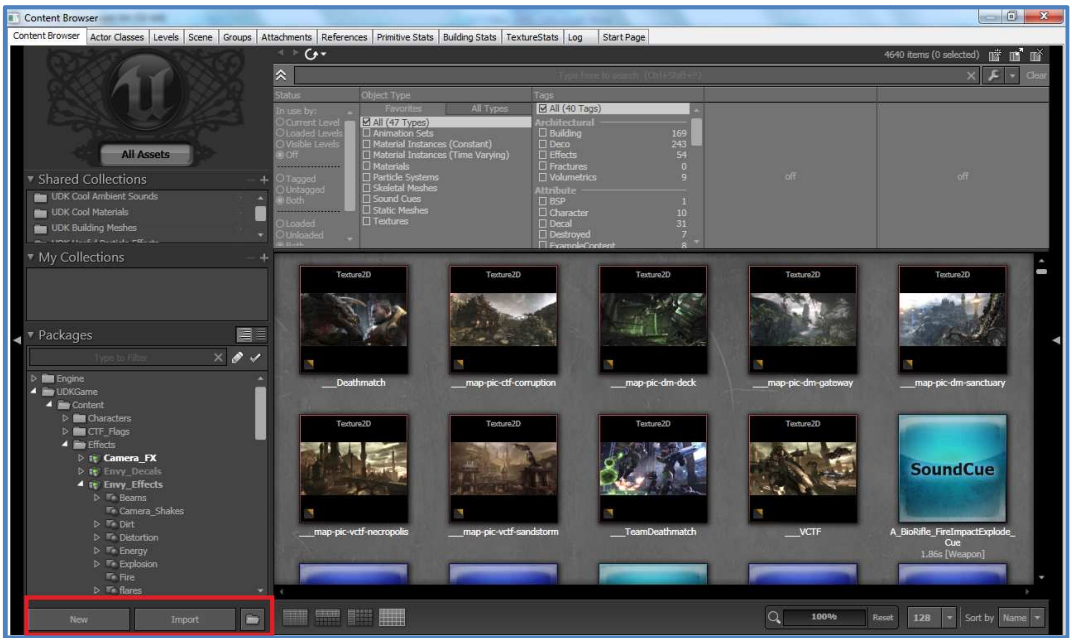


Imagen 54:Opciones creación/importación elementos

Los formatos que se deben respetar para importar los distintos elementos son:

- Apex Clothing Asset: “.aca”.
- Apex Destructuble Asset: “.pda”.
- Apex Generic Asset: “.acml”.
- Collada Skeletal Mesh: “.dae”.
- FBX meshes y Skeletal Mesh: “.fbx”.
- Texture: “.bmp”, “.pcx”, “.tga”, “.float”, “.psd”, “.png”.
- Static Mesh: “.t3d”, “.ase”
- Skeletal Mesh: “.psk”.
- Sound: “.wav”.
- Unreal World: “.t3d”.
- Unreal Anim Object: “.t3d”.
- Material: “.t3d”
- Unreal Package: “.T3DPKG”.
- Unreal Sequence: “.t3d”.
- Multichannel Sound: “.wav”.
- Speed tree: “.srt”.
- Unreal Terrain: “.t3d”.
- Bink Movie: “.bik”.
- SWF Movie: “.swf”, “.gfx”.

4.2. Crear nuevo elemento.

También se pueden crear elementos desde cero. Para crear un elemento se puede pulsar sobre el botón *“New”* de la parte inferior:

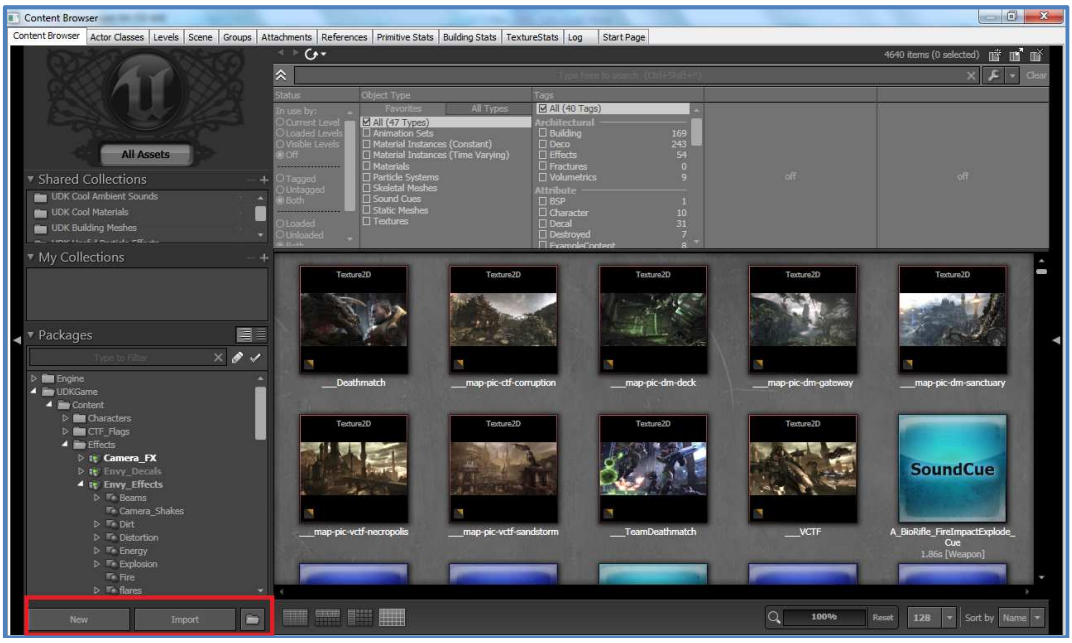


Imagen 55:Opciones de creación/importación elementos al "Content Browser".

Otra opción es pulsar sobre cualquier zona vacía de la parte central con el botón derecho del ratón y se desplegará un submenú donde se podrá elegir el tipo del elemento a crear:

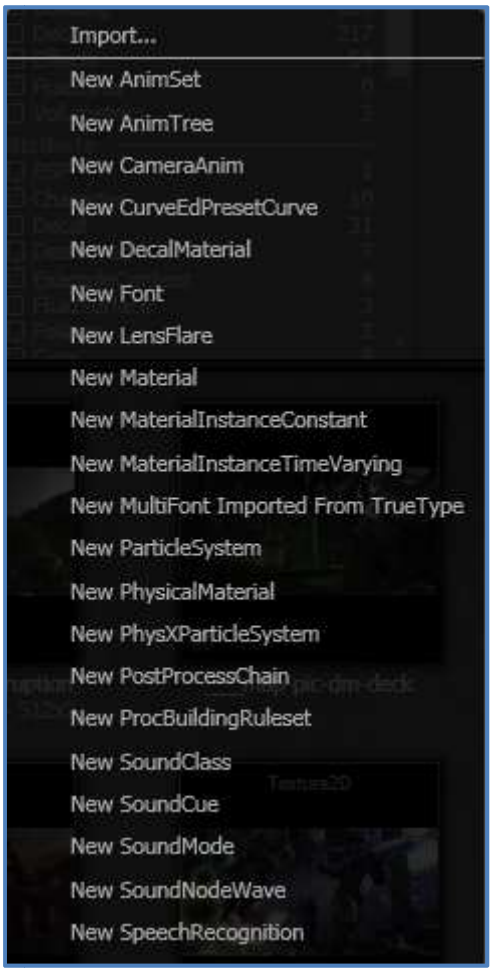


Imagen 56:Menú creación nuevo elemento.

Cuando se haya seleccionado el tipo de elemento a crear se abrirá una nueva ventana, mostrada en la imagen 57, donde se podrá indicar el paquete, el grupo y el nombre a dar al elemento a crear.

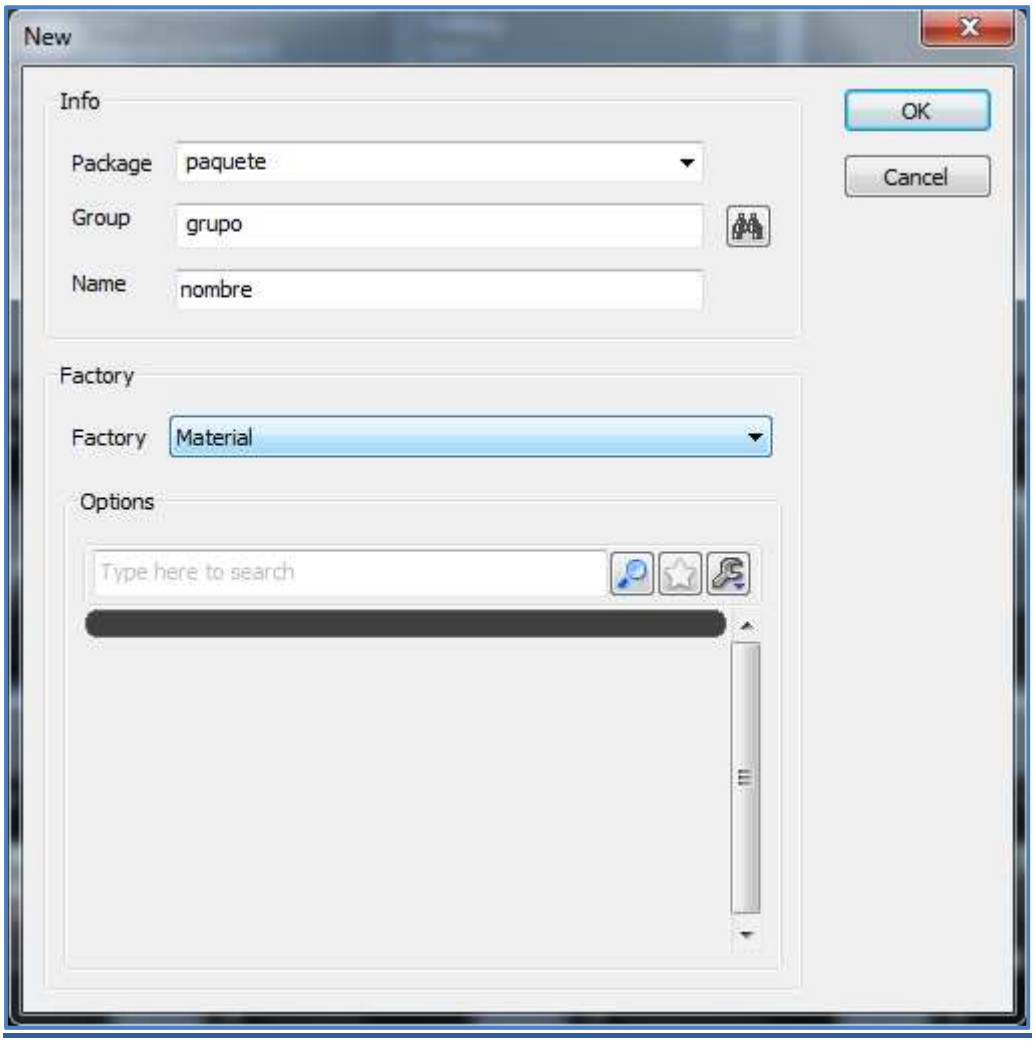


Imagen 57:Pantalla creación nuevo elemento.

Cuando se hayan introducido los datos y pulsado el botón *"OK"* se abrirá el editor correspondiente a ese tipo de elemento. Por cada elemento se tiene un editor propio donde las características pueden ser modificadas.

4.3. Información accesible.

Al abrir la herramienta Content Browser se tiene acceso a diversa información extra del nivel que se está creando.

En la parte superior de la pantalla se pueden apreciar otras pestañas además de la pestaña que contiene al Content Browser. Pulsando en cada una de ellas se podrá acceder a información referente al nivel.

Las pestañas a las que se puede acceder mediante esta herramienta son:

Actor Classes.

Se muestran todos los tipos de actores que pueden ser añadidos al nivel. Todos los elementos están mostrados en forma jerárquica y en estructura de árbol.

En dicho explorador se pueden mostrar también todos los objetos que se pueden acoplar al nivel. Para mostrarlos sólo hay que quitar el tick que se encuentra en la etiqueta*"Use 'Actor' as Parent?"* de la parte superior. Realizando este cambio se muestran todos los objetos que cuelgan de *"Object"* comprobando que *"Actor"* es un objeto del mismo.

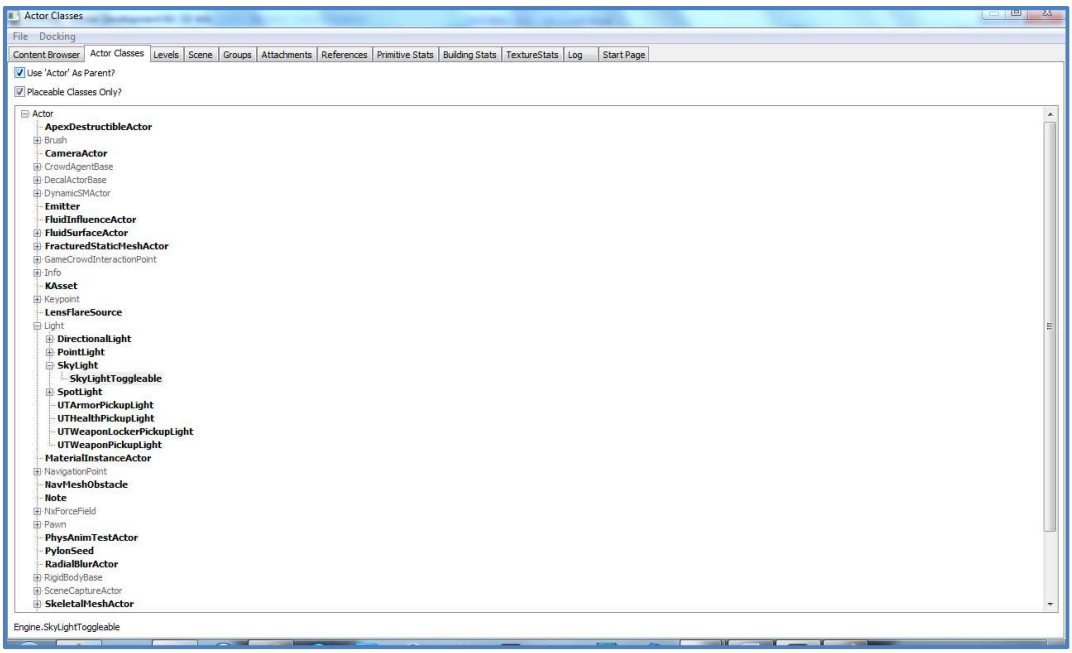


Imagen 58:Actor Classes.

Todos los objetos y actores pueden ser acoplados al nivel. Simplemente hay que seleccionar el objeto y cerrar la ventana. Aunque se cierra la ventana el objeto seguirá seleccioando para poder acoplarlo.

Una vez en el editor, en cualquiera de las ventanas del editor y en cualquier zona despejada de objetos, se pulsa con el botón secundario del ratón. Se mostrará un submenú en el cual aparecerá la opción de agregar el objeto que se tenía marcado en el Actor Classes al comienzo de la cuarta subsección:

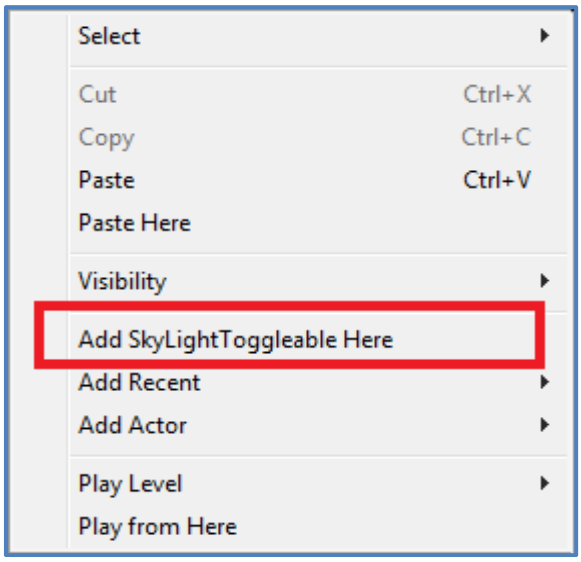


Imagen 59:Opción de agregar elemento al nivel.

Levels.

En esta pestaña se muestra el nivel que se está realizando indicando todos los objetos que contiene y todos los demás niveles que pueden estar referenciados por el que se está creando.

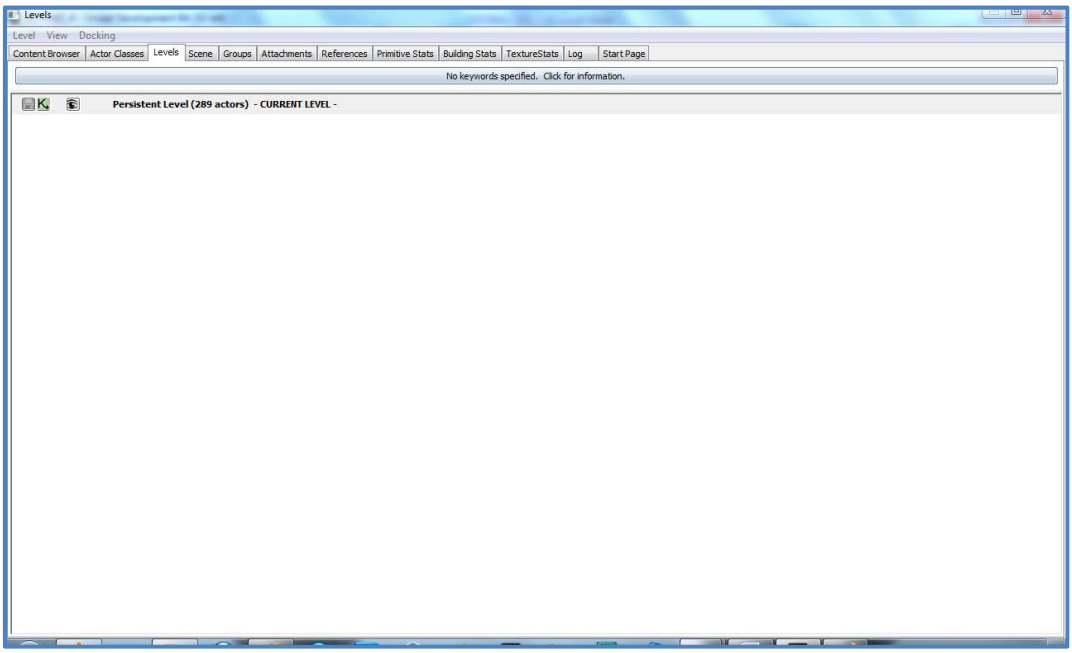


Imagen 60:Levels

Scene.

En dicha pestaña se muestran todos los objetos que se tienen en el nivel indicando ciertas características de cada uno:

- Nombre del actor.
- Tipo del actor.
- El grupo al que pertenece.

- Si está asociado en el Unreal Kismet.
- La localización.- Indicando las coordenadas de los ejes de coordenadas.
- El paquete en el que se encuentra.

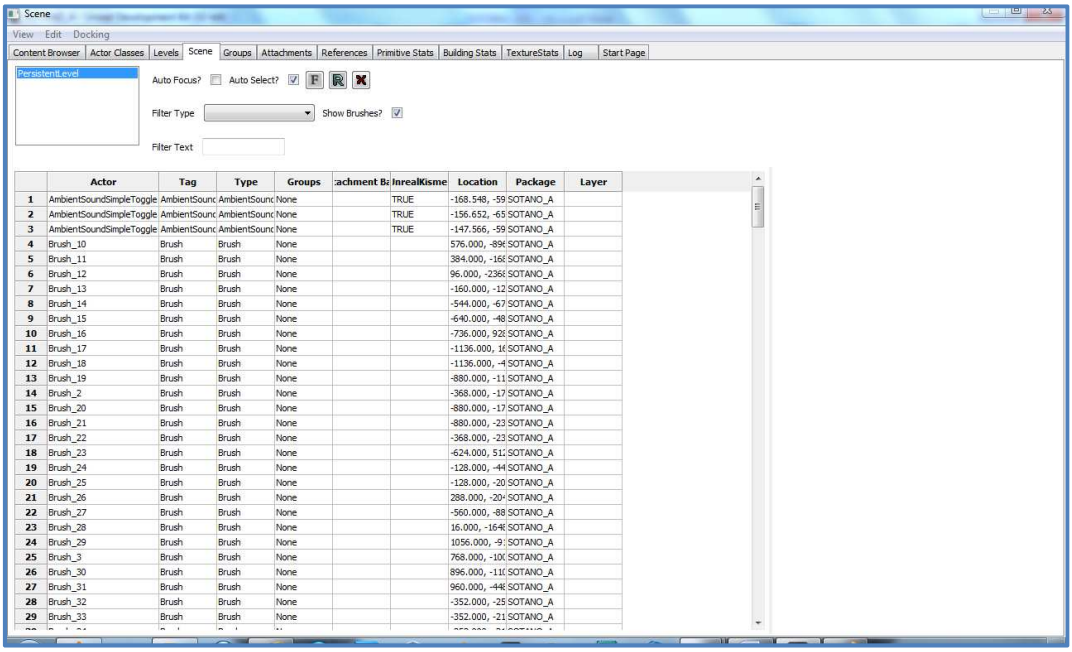


Imagen 61:Scene.

Groups.

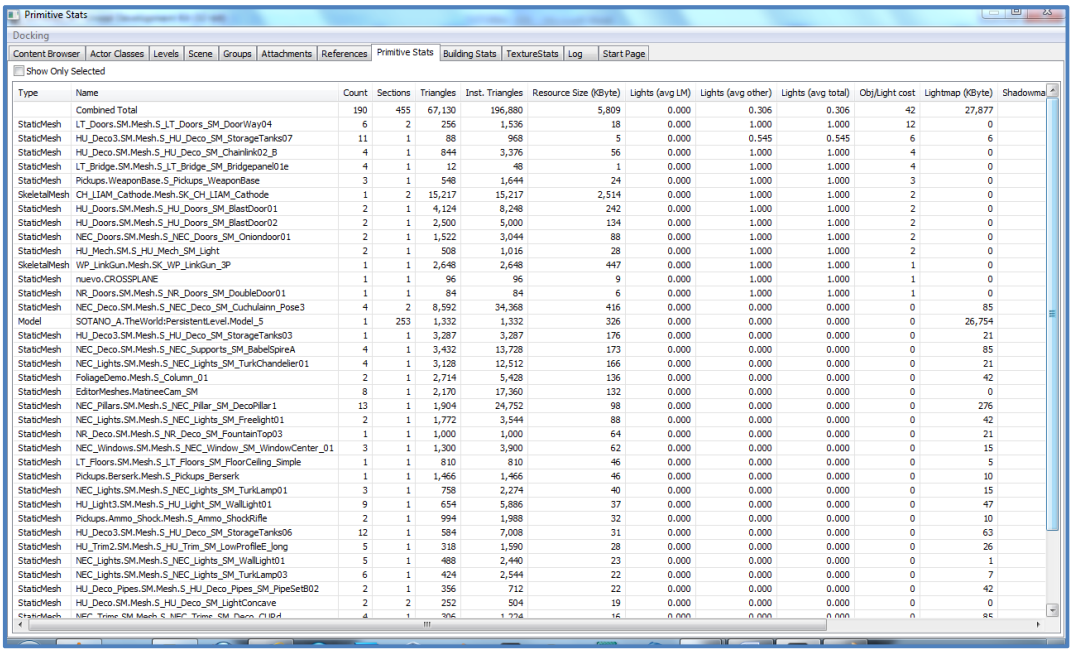
Se muestran los distintos grupos de objetos que se tienen en el nivel creado.

References.

En esta pestaña se muestran todas las referencias que se tienen en el nivel creado.

Primitive Stats.

Es mostrado un resumen de todos los objetos primitivos usados en la creación del nivel. Se indica el tipo de objeto primitivo usado al igual que el número de objetos que se tiene entre otra información.



Primitive Stats

Docking

Content Browser | Actor Classes | Levels | Scene | Groups | Attachments | References | Primitive Stats | Building Stats | TextureStats | Log | Start Page

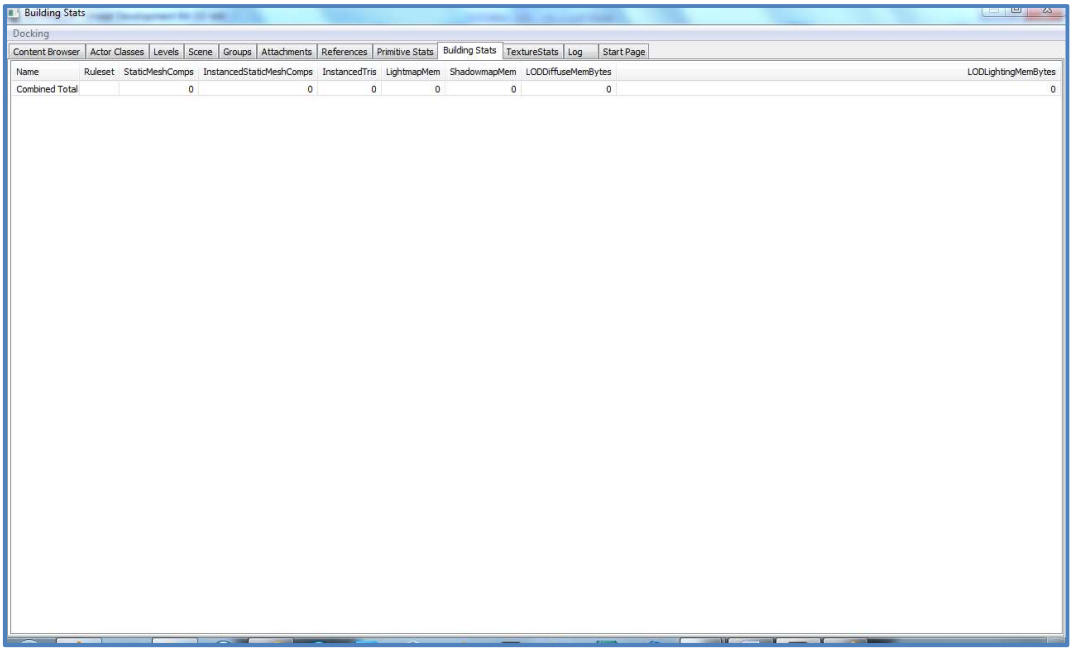
Show Only Selected

Type	Name	Count	Sections	Triangles	Inst. Triangles	Resource Size (KByte)	Lights (avg LM)	Lights (avg other)	Lights (avg total)	Obj/Light cost	Lightmap (KByte)	Shadowma
	Combined Total	190	455	67,130	196,880	5,809	0.000	0.306	0.306	42	27,877	
StaticMesh	LT_Doors.SM.Mesh.S_LT_Doors_SM_DoorWay04	6	2	256	1,536	18	0.000	1.000	1.000	12	0	
StaticMesh	HU_Deco3.SM.Mesh.S_HU_Deco_SM_StorageTanks07	11	1	88	968	5	0.000	0.545	0.545	6	6	
StaticMesh	HU_Deco.SM.Mesh.S_HU_Deco_SM_Chainlink02_B	4	1	844	3,376	56	0.000	1.000	1.000	4	0	
StaticMesh	LT_Bridge.SM.Mesh.S_LT_Bridge_SM_Bridgepanel01e	4	1	12	48	1	0.000	1.000	1.000	4	0	
StaticMesh	Pickups.WeaponBase.S_Pickups_WeaponBase	3	1	548	1,644	24	0.000	1.000	1.000	3	0	
SkeletalMesh	CH_LIAM_Cathode.Mesh.SK_CH_LIAM_Cathode	1	2	15,217	15,217	2,514	0.000	1.000	1.000	2	0	
StaticMesh	HU_Doors.SM.Mesh.S_HU_Doors_SM_BlastDoor01	2	1	4,124	8,248	242	0.000	1.000	1.000	2	0	
StaticMesh	HU_Doors.SM.Mesh.S_HU_Doors_SM_BlastDoor02	2	1	2,500	5,000	134	0.000	1.000	1.000	2	0	
StaticMesh	NEC_Doors.SM.Mesh.S_NEC_Doors_SM_Oniondoor01	2	1	1,522	3,044	88	0.000	1.000	1.000	2	0	
StaticMesh	HU_Mech.SM.S_HU_Mech_SM_Light	2	1	508	1,016	28	0.000	1.000	1.000	2	0	
SkeletalMesh	WP_LinkGun.Mesh.SK_WP_LinkGun_3P	1	1	2,648	2,648	447	0.000	1.000	1.000	1	0	
StaticMesh	nuevo.CROSSPLANE	1	1	96	96	9	0.000	1.000	1.000	1	0	
StaticMesh	NR_Doors.SM.Mesh.S_NR_Doors_SM_DoubleDoor01	1	1	84	84	6	0.000	1.000	1.000	1	0	
StaticMesh	NEC_Deco.SM.Mesh.S_NEC_Deco_SM_Cuchulainn_Pose3	4	2	8,592	34,368	416	0.000	0.000	0.000	0	85	
Model	SOTANO_A.TheWorld:PersistentLevel.Model_5	1	253	1,332	1,332	326	0.000	0.000	0.000	0	26,754	
StaticMesh	HU_Deco3.SM.Mesh.S_HU_Deco_SM_StorageTanks03	1	1	3,287	3,287	176	0.000	0.000	0.000	0	21	
StaticMesh	NEC_Deco.SM.Mesh.S_NEC_Supports_SM_BabelSpireA	4	1	3,432	13,728	173	0.000	0.000	0.000	0	85	
StaticMesh	NEC_Lights.SM.Mesh.S_NEC_Lights_SM_TurkChandelier01	4	1	3,128	12,512	166	0.000	0.000	0.000	0	21	
StaticMesh	FoliageDemo.Mesh.S_Column_01	2	1	2,714	5,428	136	0.000	0.000	0.000	0	42	
StaticMesh	EditorMeshes.MatineeCam_SM	8	1	2,170	17,360	132	0.000	0.000	0.000	0	0	
StaticMesh	NEC_Pillars.SM.Mesh.S_NEC_Pillar_SM_DecoPillar1	13	1	1,904	24,752	98	0.000	0.000	0.000	0	276	
StaticMesh	NEC_Lights.SM.Mesh.S_NEC_Lights_SM_Freelight01	2	1	1,772	3,544	88	0.000	0.000	0.000	0	42	
StaticMesh	NR_Deco.SM.Mesh.S_NR_Deco_SM_FountainTop03	1	1	1,000	1,000	64	0.000	0.000	0.000	0	21	
StaticMesh	NEC_Windows.SM.Mesh.S_NEC_Window_SM_WindowCenter_01	3	1	1,300	3,900	62	0.000	0.000	0.000	0	15	
StaticMesh	LT_Floors.SM.Mesh.S_LT_Floors_SM_FloorCeiling_Simple	1	1	810	810	46	0.000	0.000	0.000	0	5	
StaticMesh	Pickups.Berserk.Mesh.S_Pickups_Berserk	1	1	1,466	1,466	46	0.000	0.000	0.000	0	10	
StaticMesh	NEC_Lights.SM.Mesh.S_NEC_Lights_SM_TurkLamp01	3	1	758	2,274	40	0.000	0.000	0.000	0	15	
StaticMesh	HU_Light3.SM.Mesh.S_HU_Light_SM_WallLight01	9	1	654	5,886	37	0.000	0.000	0.000	0	47	
StaticMesh	Pickups.Ammo_Shock.Mesh.S_Ammo_ShockRifle	2	1	994	1,988	32	0.000	0.000	0.000	0	10	
StaticMesh	HU_Deco3.SM.Mesh.S_HU_Deco_SM_StorageTanks06	12	1	584	7,008	31	0.000	0.000	0.000	0	63	
StaticMesh	HU_Trim2.SM.Mesh.S_HU_Trim_SM_LowProfileE_long	5	1	318	1,590	28	0.000	0.000	0.000	0	26	
StaticMesh	NEC_Lights.SM.Mesh.S_NEC_Lights_SM_WallLight01	5	1	488	2,440	23	0.000	0.000	0.000	0	1	
StaticMesh	NEC_Lights.SM.Mesh.S_NEC_Lights_SM_TurkLamp03	6	1	424	2,544	22	0.000	0.000	0.000	0	7	
StaticMesh	HU_Deco_Pipes.SM.Mesh.S_HU_Deco_Pipes_SM_PipeSetB02	2	1	356	712	22	0.000	0.000	0.000	0	42	
StaticMesh	HU_Deco.SM.Mesh.S_HU_Deco_SM_LightConcave	2	2	252	504	19	0.000	0.000	0.000	0	0	
StaticMesh	NEC_Trims.SM.Mesh.S_NEC_Trims_SM_Deco_CLIPd	4	1	306	1,224	16	0.000	0.000	0.000	0	85	

Imagen 62:Primitive Stats.

Building Stats.

Se muestra un resumen de todas las construcciones usadas en la creación del nivel.



Building Stats

Docking

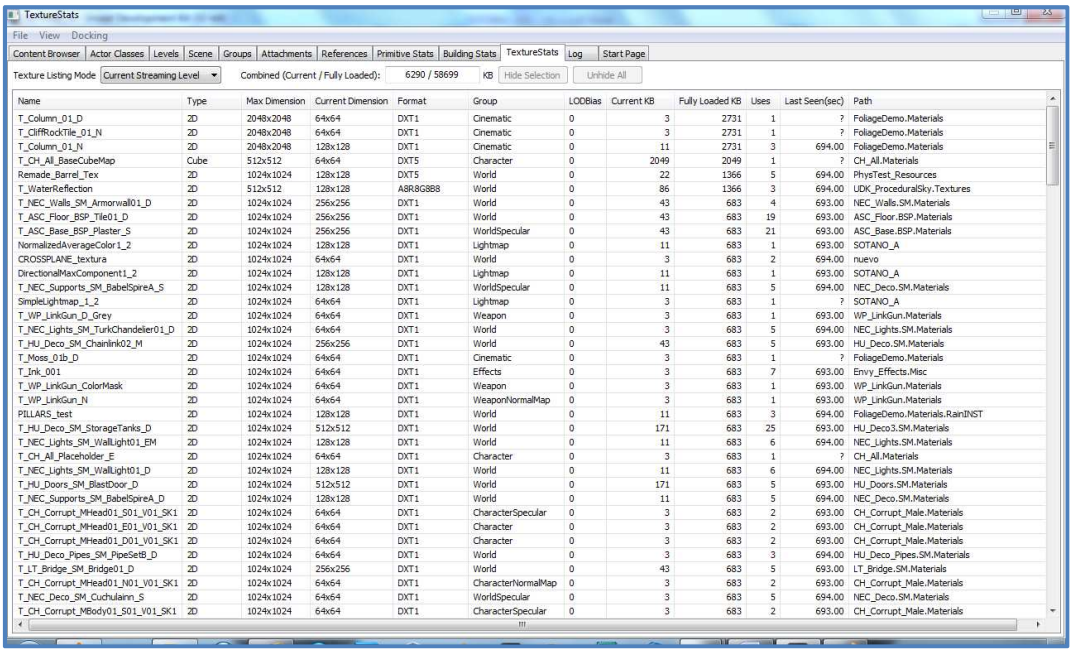
Content Browser | Actor Classes | Levels | Scene | Groups | Attachments | References | Primitive Stats | Building Stats | TextureStats | Log | Start Page

Name	Ruleset	StaticMeshComps	InstancedStaticMeshComps	InstancedTris	LightmapMem	ShadowmapMem	LODDiffuseMemBytes	LODLightingMemBytes
Combined Total		0	0	0	0	0	0	0

Imagen 63:Building Stats.

TextureStats.

Resumen de todas las texturas utilizadas en todos los objetos del nivel. Entre otra información proporcionada, se indica el nombre de cada textura, el tipo que es, las dimensiones de la textura, las dimensiones utilizadas en el objeto, el formato y el path donde se encuentra.



Name	Type	Max Dimension	Current Dimension	Format	Group	LODBias	Current KB	Fully Loaded KB	Uses	Last Seen(sec)	Path
T_Column_01_D	2D	2048x2048	64x64	DXT1	Cinematic	0	3	2731	1	?	FoliageDemo.Materials
T_CliffRockTile_01_N	2D	2048x2048	64x64	DXT1	Cinematic	0	3	2731	1	?	FoliageDemo.Materials
T_Column_01_N	2D	2048x2048	128x128	DXT1	Cinematic	0	11	2731	3	694.00	FoliageDemo.Materials
T_CH_All_BaseCubeMap	Cube	512x512	64x64	DXT5	Character	0	2049	2049	1	?	CH_All.Materials
Remade_Barrel_Tex	2D	1024x1024	128x128	DXT5	World	0	22	1366	5	694.00	PhysTest_Resources
T_WaterReflection	2D	512x512	128x128	A8R8G8B8	World	0	86	1366	3	694.00	UDK_ProceduralSky.Textures
T_NEC_Walls_SM_Armorwall01_D	2D	1024x1024	256x256	DXT1	World	0	43	683	4	693.00	NEC_Walls.SM.Materials
T_ASC_Floor_BSP_Tile01_D	2D	1024x1024	256x256	DXT1	World	0	43	683	19	693.00	ASC_Floor.BSP.Materials
T_ASC_Base_BSP_Plaster_S	2D	1024x1024	256x256	DXT1	WorldSpecular	0	43	683	21	693.00	ASC_Base.BSP.Materials
NormalizedAverageColor1_2	2D	1024x1024	128x128	DXT1	Lightmap	0	11	683	1	693.00	SOTANO_A
CROSSPLANE_textura	2D	1024x1024	64x64	DXT1	World	0	3	683	2	694.00	nuevo
DirectionalMaxComponent1_2	2D	1024x1024	128x128	DXT1	Lightmap	0	11	683	1	693.00	SOTANO_A
T_NEC_Supports_SM_BabelSpireA_S	2D	1024x1024	128x128	DXT1	WorldSpecular	0	11	683	5	694.00	NEC_Deco.SM.Materials
SimpleLightmap_1_2	2D	1024x1024	64x64	DXT1	Lightmap	0	3	683	1	?	SOTANO_A
T_WP_LinkGun_D_Grey	2D	1024x1024	64x64	DXT1	Weapon	0	3	683	1	693.00	WP_LinkGun.Materials
T_NEC_Lights_SM_TurkChandelier01_D	2D	1024x1024	64x64	DXT1	World	0	3	683	5	694.00	NEC_Lights.SM.Materials
T_HU_Deco_SM_Chainlink02_M	2D	1024x1024	256x256	DXT1	World	0	43	683	5	693.00	HU_Deco.SM.Materials
T_Moss_01b_D	2D	1024x1024	64x64	DXT1	Cinematic	0	3	683	1	?	FoliageDemo.Materials
T_Ink_001	2D	1024x1024	64x64	DXT1	Effects	0	3	683	7	693.00	Envy_Effects.Misc
T_WP_LinkGun_ColorMask	2D	1024x1024	64x64	DXT1	Weapon	0	3	683	1	693.00	WP_LinkGun.Materials
T_WP_LinkGun_N	2D	1024x1024	64x64	DXT1	WeaponNormalMap	0	3	683	1	693.00	WP_LinkGun.Materials
PILLARS_test	2D	1024x1024	128x128	DXT1	World	0	11	683	3	694.00	FoliageDemo.Materials.RainINST
T_HU_Deco_SM_StorageTanks_D	2D	1024x1024	512x512	DXT1	World	0	171	683	25	693.00	HU_Deco3.SM.Materials
T_NEC_Lights_SM_WallLight01_EM	2D	1024x1024	128x128	DXT1	World	0	11	683	6	694.00	NEC_Lights.SM.Materials
T_CH_All_Placeholder_E	2D	1024x1024	64x64	DXT1	Character	0	3	683	1	?	CH_All.Materials
T_NEC_Lights_SM_WallLight01_D	2D	1024x1024	128x128	DXT1	World	0	11	683	6	694.00	NEC_Lights.SM.Materials
T_HU_Doors_SM_BlastDoor_D	2D	1024x1024	512x512	DXT1	World	0	171	683	5	693.00	HU_Doors.SM.Materials
T_NEC_Supports_SM_BabelSpireA_D	2D	1024x1024	128x128	DXT1	World	0	11	683	5	694.00	NEC_Deco.SM.Materials
T_CH_Corrupt_MHead01_S01_V01_SK1	2D	1024x1024	64x64	DXT1	CharacterSpecular	0	3	683	2	693.00	CH_Corrupt_Male.Materials
T_CH_Corrupt_MHead01_E01_V01_SK1	2D	1024x1024	64x64	DXT1	Character	0	3	683	2	693.00	CH_Corrupt_Male.Materials
T_CH_Corrupt_MHead01_D01_V01_SK1	2D	1024x1024	64x64	DXT1	Character	0	3	683	2	693.00	CH_Corrupt_Male.Materials
T_HU_Deco_Pipes_SM_PipeSetB_D	2D	1024x1024	64x64	DXT1	World	0	3	683	3	694.00	HU_Deco_Pipes.SM.Materials
T_LT_Bridge_SM_Bridge01_D	2D	1024x1024	256x256	DXT1	World	0	43	683	5	693.00	LT_Bridge.SM.Materials
T_CH_Corrupt_MHead01_N01_V01_SK1	2D	1024x1024	64x64	DXT1	CharacterNormalMap	0	3	683	2	693.00	CH_Corrupt_Male.Materials
T_NEC_Deco_SM_Cuchulainn_S	2D	1024x1024	64x64	DXT1	WorldSpecular	0	3	683	5	694.00	NEC_Deco.SM.Materials
T_CH_Corrupt_MBody01_S01_V01_SK1	2D	1024x1024	64x64	DXT1	CharacterSpecular	0	3	683	2	693.00	CH_Corrupt_Male.Materials

Imagen 64:Texture Stats.

Log.

Son mostrados todos los mensajes de información de eventos en la creación del nivel.

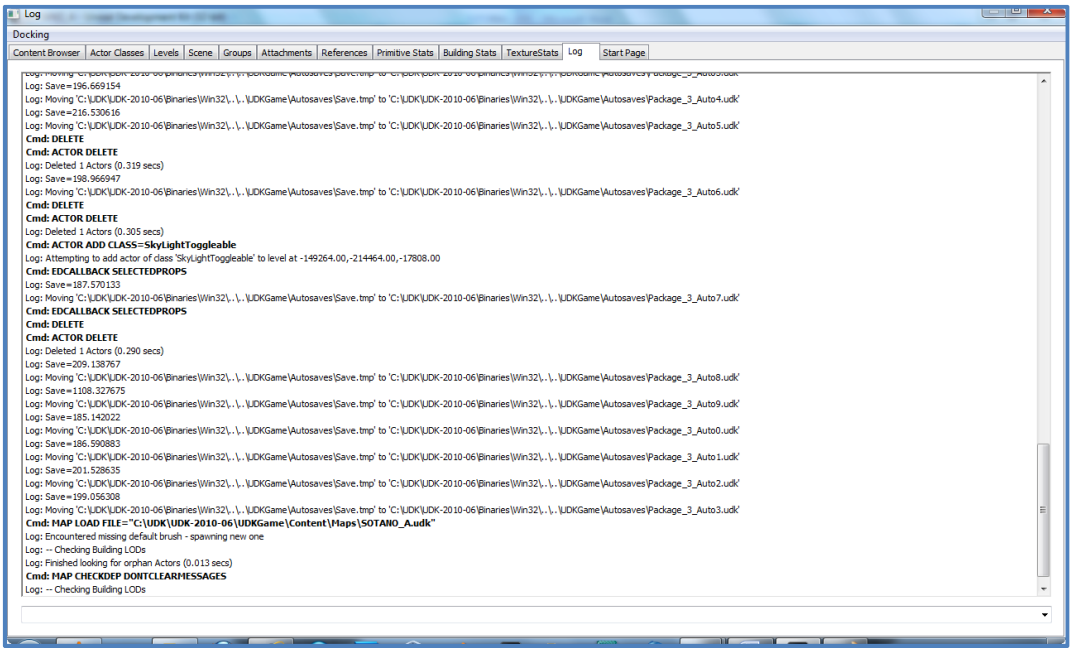


Imagen 65:Log.

Start Page.

Se muestra la página principal de UDK (www.udk.com).

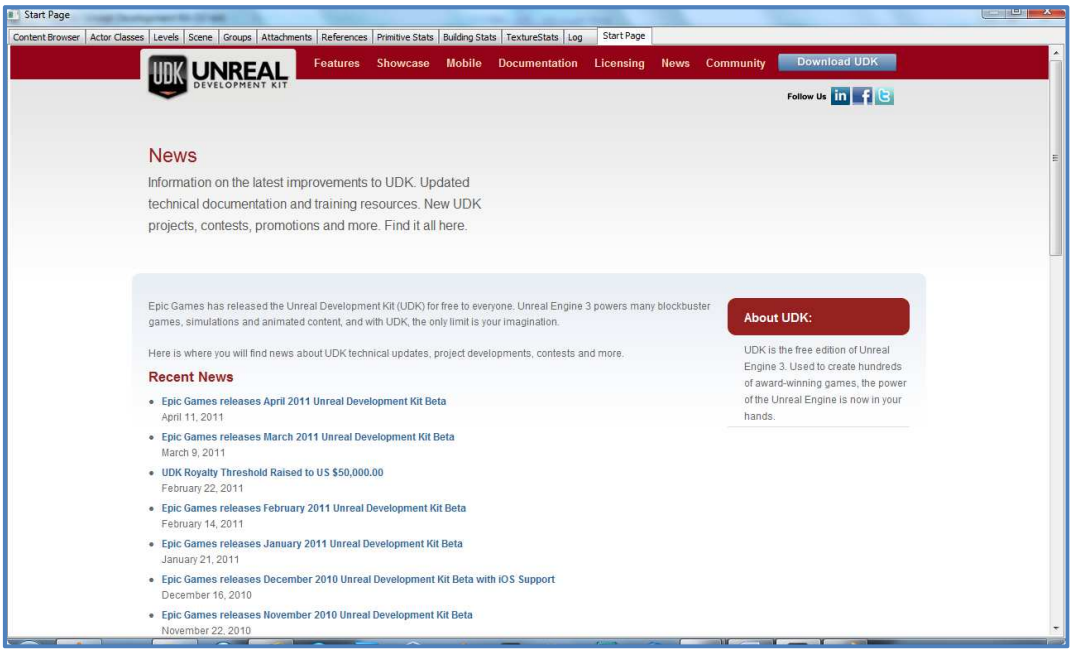


Imagen 66:Start Page.

5. UNREAL KISMET.

Dicha herramienta permite modificar el flujo de ejecución del mapa. La interfaz que se nos presenta es una pantalla vacía en la cual se podrán ir añadiendo distintos elementos para realizar modificaciones en el flujo del nivel, mostrada en la imagen 67.

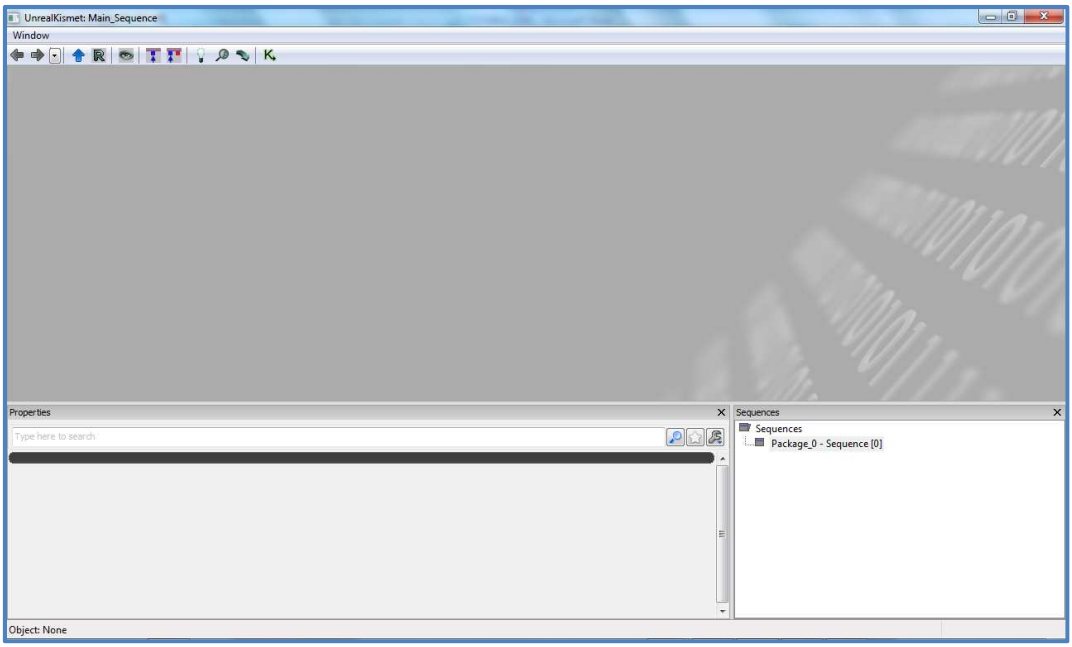


Imagen 67:Interfaz Kismet.

En la zona superior-central se podrán ir añadiendo los distintos elementos para crear el flujo del mapa creado.

En la sección inferior izquierda serán mostradas las propiedades del elemento seleccionado. Todas las características serán mostradas con su respectivo valor pudiendo ser modificadas según conveniencia.

Y por último, en la sección inferior derecha se presentará la jerarquía de paquetes en la secuencia del flujo de ejecución creada.

Cada uno de los niveles que se creen tienen asociado su propio flujo de acciones en la ejecución implementado en dicha herramienta.

5.1. Realizar movimientos en la plantilla de la herramienta.

La plantilla ofrecida en la herramienta es bastante extensa, con lo que se deberá ir desplazando para ir añadiendo nuevos elementos o trasladarlos de sitio.

Para poder desplazar la plantilla y mover la vista sobre ella simplemente hay que mantener pulsado el botón derecho del ratón y realizar el movimiento deseado.

Para trasladar un elemento de sitio primero hay que seleccionarlo pulsando sobre él con el botón izquierdo del ratón. El elemento seleccionado quedará enmarcado en amarillo para diferenciarlo de los demás. Con el elemento seleccionado y sin dejar de pulsar el botón izquierdo del ratón, a la vez que se pulsa la tecla ‘Ctrl’ del teclado, se realiza el moviento del ratón desplazando el elemento. Pueden moverse varios elementos a la vez del mismo modo: para seleccionar varios elementos se mantiene la

tecla 'Ctrl' del teclado pulsada mientras con el botón izquierdo del ratón se van seleccionando los otros elementos.

Hay otra manera de poder seleccionar varios elementos a la vez. Para ello hay que mantener pulsadas las teclas 'Ctrl' y 'Shift' del teclado a la vez que el botón izquierdo del ratón. Se creará un cuadro el cual podrá modificarse su tamaño moviendo el ratón. Los elementos que queden enmarcados por este cuadro o simplemente haga contacto, quedarán seleccionados.

5.2. Agregar elementos.

El modo de uso es muy intuitivo, ya que los distintos elementos a utilizar son *"cajas"* que se van enlazando unas a otras mediante *"flechas"*. Estas *"flechas"* representan los caminos que puede tomar el flujo realizando las distintas acciones que se tengan y tomando uno u otro según las condiciones dadas.

Se tienen cinco tipos de elementos principales en Kismet: acciones, elemento Matinee, condiciones, variables y eventos. Para acceder a los distintos elementos que se pueden añadir hay que desplegar el submenú mostrado en la imagen 68, pulsando con el botón derecho sobre cualquier zona vacía:

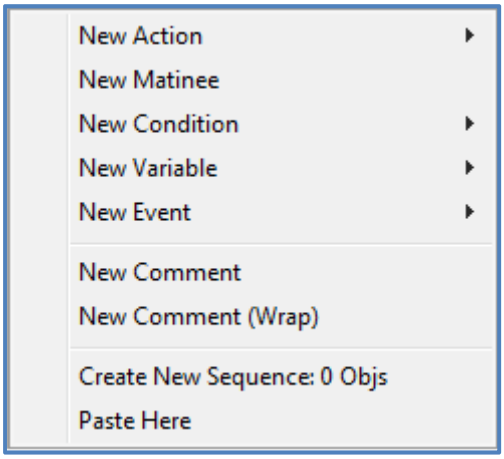


Imagen 68:Menú añadir elemento.

5.2.1. Eventos.

Los eventos son el comienzo de cualquier secuencia de un flujo de ejecución. Se podría decir que es: cuando sucede algo, qué hay que realizar.

El menú de eventos se presenta en la imagen 69.

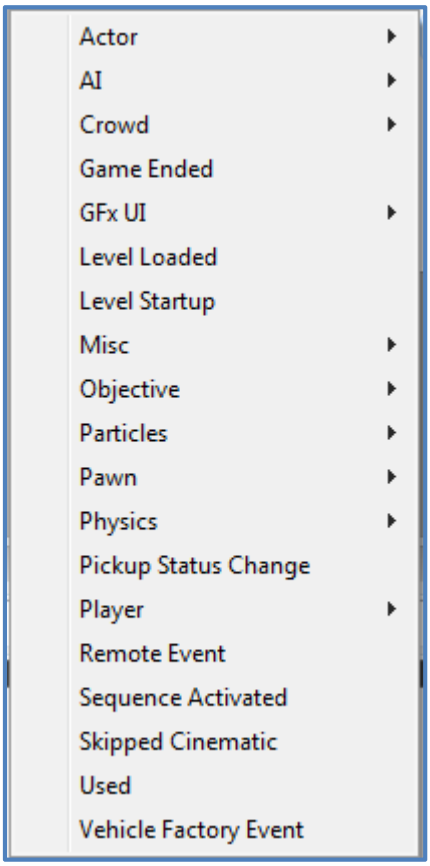


Imagen 69:Menú añadir evento.

Todos los eventos tienen una propiedad llamada *"Max Trigger Count"* en la cual se indica las veces que se quiere que se lance dicho evento. Hay que tener en cuenta que si se quiere poner que se realice siempre hay que poner el valor a 0 en dicha propiedad.

Siempre el punto de partida de un flujo es un evento producido en el nivel. El evento más general que se tiene es cuando el mapa es lanzado ó iniciado:

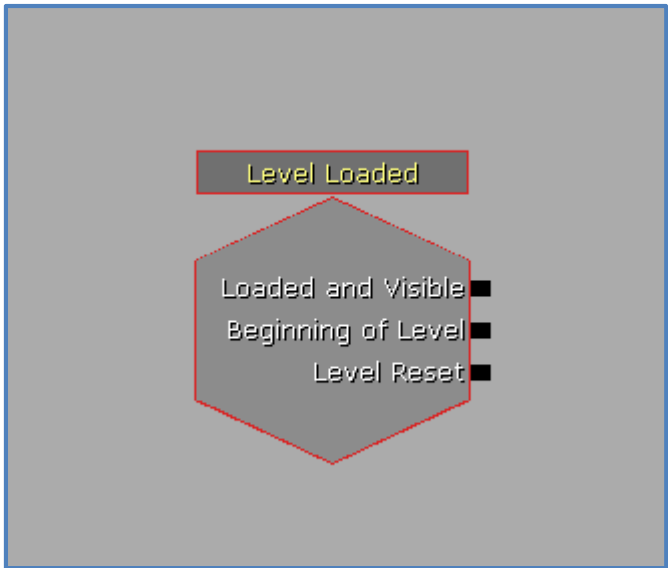


Imagen 70:Ejemplo evento.

Como se puede observar, estos elementos no tienen entradas, sólo tienen puntos de salida, ya que son los que inician cualquier flujo de ejecución.

NOTA: En el caso que no se inserte un evento al comienzo del flujo de ejecución, dicha funcionalidad no será realizada.

Los eventos más importantes que se pueden utilizar son:

- **Actor → Take Damage**.- El personaje principal sufre algún daño. Dentro de las propiedades de este elemento se puede indicar el tipo de daño que se quiere utilizar, el valor mínimo de daño para lanzar el evento, así como el valor del daño a disminuir de la vida del personaje.

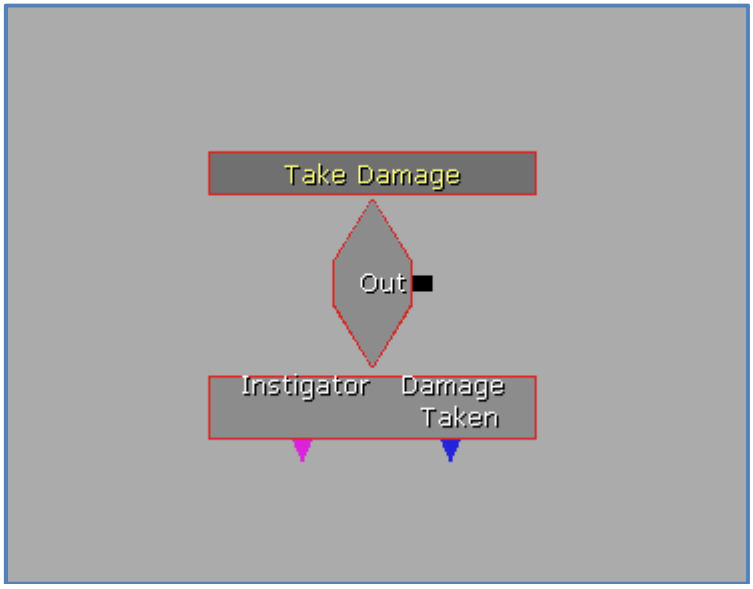


Imagen 71:Elemento "Take Damage".

- **Actor → Destroyed**.- Este evento es lanzado cuando el contador de vida del personaje llega a 0.

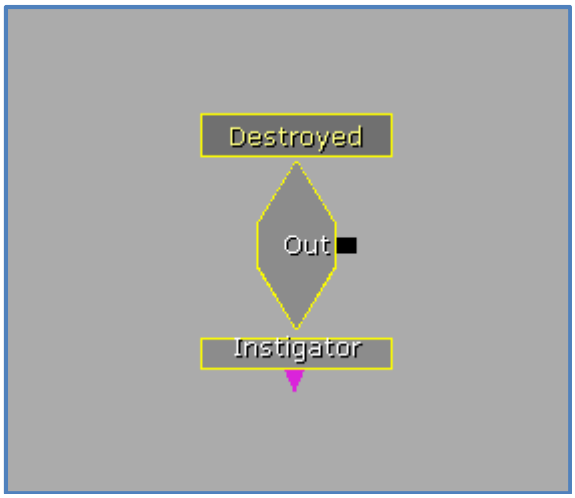


Imagen 72:Elemento "Destroyed".

- **Level Loaded**.- Este evento es uno de los principales. Representa la carga del nivel creado y su comienzo. Por lo tanto, todo aquello que se quiera que se produzca al inicio del nivel deberá ser lanzado por este evento.

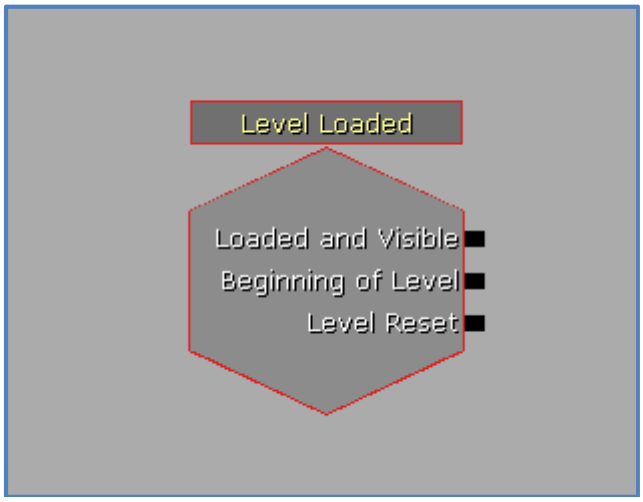


Imagen 73:Elemento "Level Loaded".

- **Level StartUp**.- Evento lanzado cuando el nivel comienza. Es un evento muy parecido al evento comentado anteriormente, pero en este caso no se tiene en cuenta la carga el nivel.

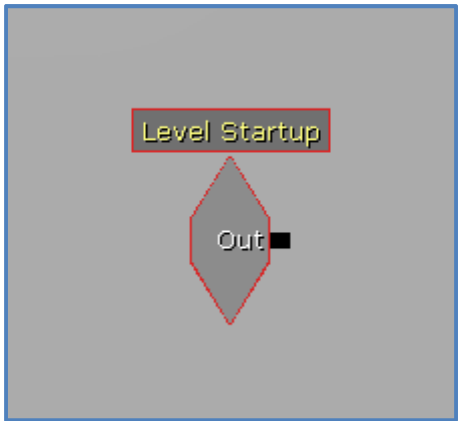


Imagen 74:Elemento "Level Startup".

- **Pawn → Death**.- Este evento se puede asociar a cualquier actor del nivel que tenga la propiedad de *"vida"*. Será lanzado cuando esta propiedad llegue a cero.

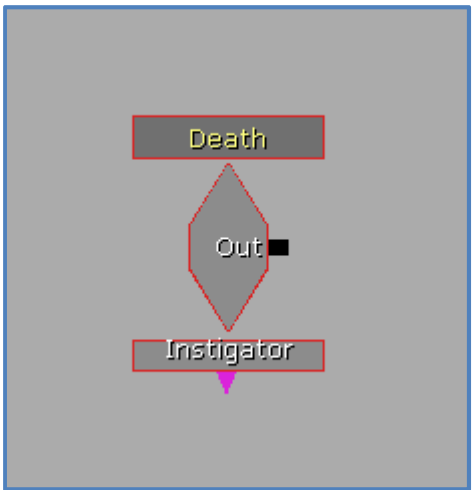


Imagen 75:Elemento "Death".

- **Pawn → GetInventory**.- Evento que será lanzado cuando el personaje asociado obtenga los elementos indicandos en la etiqueta "*Inventory*" en su inventario.

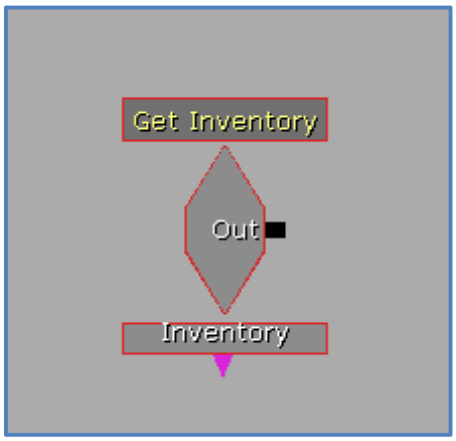


Imagen 76:Elemento "Get Inventory".

- **Player → PlayerSpawned**.- Evento lanzado cuando el personaje principal es cargado en el nivel.

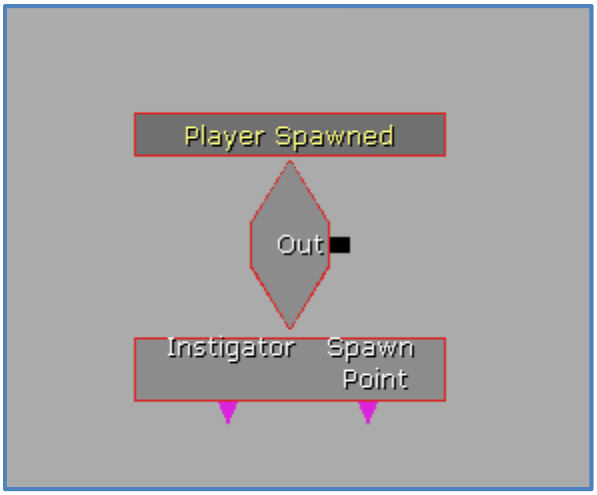


Imagen 77:Elemento "Player Spawned".

- **Remote Event**.- Mediante este evento se puede crear una cierta funcionalidad compuesta por diversas acciones y dotarle de un nombre concreto. Mediante se podrá hacer llamada a este evento desde otra acción y ejecutarla sin necesidad de volver a acoplar todos los elementos de nuevo.
  En la propiedad *"Event Name"* se podrá introducir el nombre que se quiere dar a esta funcionalidad.

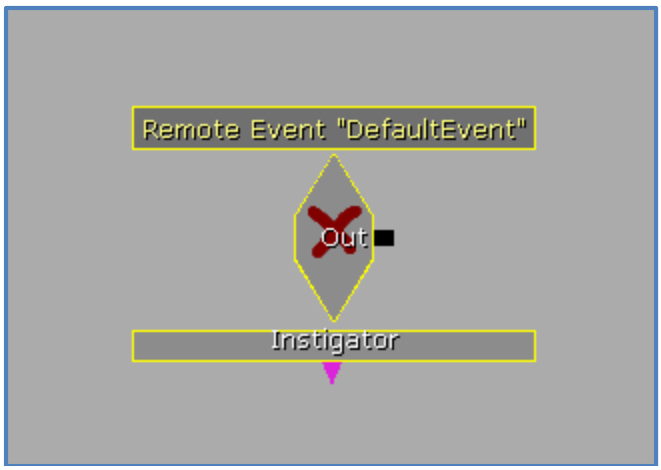


Imagen 78:Elemento "Remote Event".

5.2.2. Acciones.

Son todas aquellas acciones que se realizan cuando se recibe una señal por parte de los eventos. Ejemplos de estas acciones pueden ser realizar un teletransporte, apagar luces, etc.

En la imagen 79 se muestra todos los tipos de acciones que se pueden insertar.

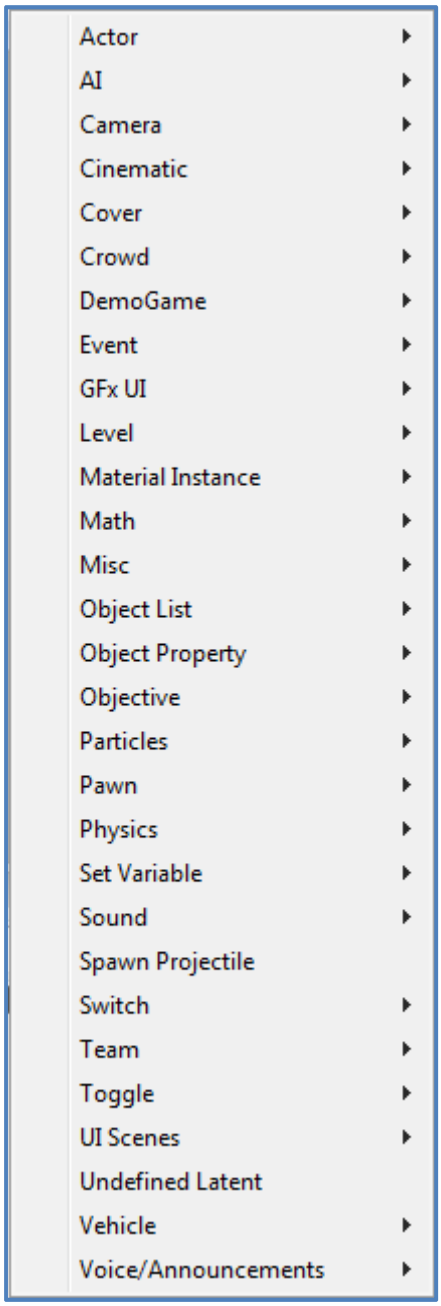


Imagen 79:Menú insertar acción.

Las acciones más importantes que se pueden utilizar son:

- **Actor → Actor Factory**.- Esta acción será utilizada para poder cargar un elemento *"Actor"* en el nivel de modo dinámico, en ejecución.
  En la propiedad "Factory" se indicará el tipo de elemento que se quiere cargar. También hay que indicar el punto donde se quiere que aparezca el *"Actor"* mediante la entrada *"Spawn Point"* de esta acción.

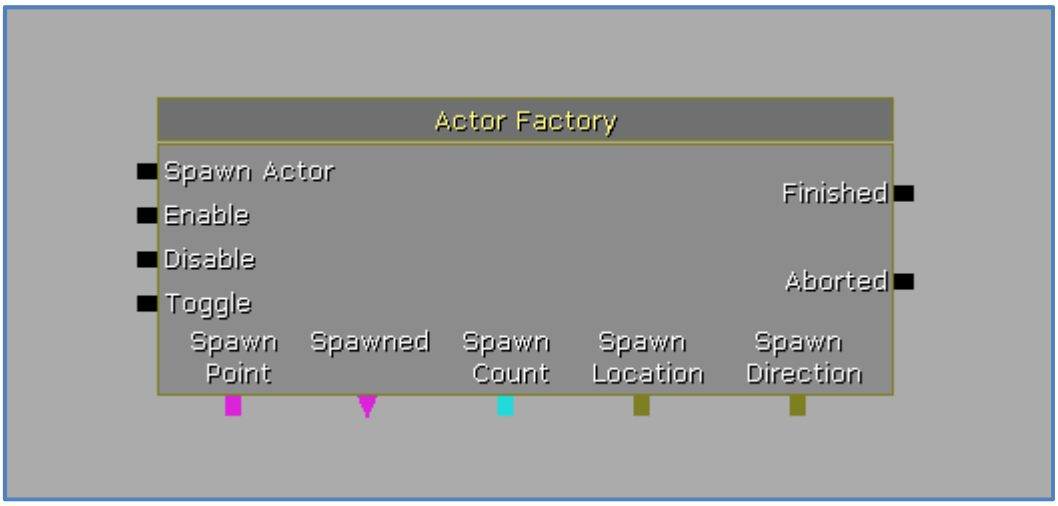


Imagen 80:Elemento "Actor Factory".

- **Actor → Destroy**.- Cuando se quiera eliminar algún elemento del nivel en modo dinámico, en ejecución, se puede utilizar esta acción.
  En la entrada *"Target"* hay que asociar el elemento que se quiere eliminar.

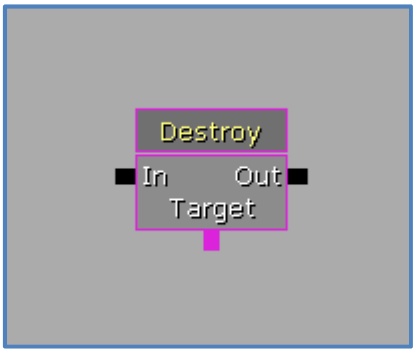


Imagen 81:Elemento "Destroy".

- **Actor → Get Distance**.- Con esta acción se puede saber la distancia que hay dentro del nivel entre dos elementos.
  Los elementos a saber su distancia se acoplarán mediante las entradas *"A"* y *"B"* de la acción y el resultado será guardado en la variable que se acople a la entrada *"Distance"*.

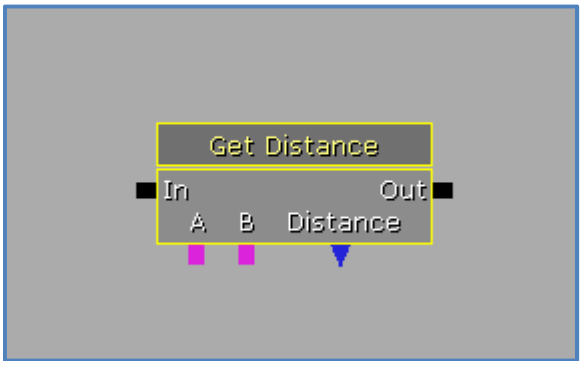


Imagen 82:Elemento "Get Distance".

- **Actor → Modify Health**.- Con esta acción se podrá modificar el contador de *"vida"* del actor asociado a ella.

En la propiedad *"Damage Type"* se indicará el tipo de daño que se quiere modificar su valor.
En la entrada *"Target"* será donde se acople el actor a modificar su *"vida"*.
El valor a incrementar o decrementar en la *"vida"* se indicará en la propiedad *"Amount"* ó se acoplará en la entrada con el mismo nombre mediante una variable con el valor.

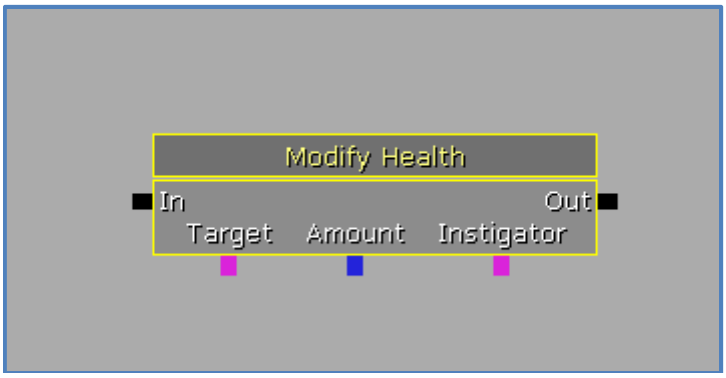


Imagen 83:Elemento "Modify Health".

- **Actor → Teleport**.- Mediante esta acción se puede realizar que el elemento que se asocie a él sea teletransportado a un punto concreto, apareciendo en ese punto.
  El elemento que se quiere teletransportar se acoplará mediante la entrada *"Target"* de la acción y el punto de destino se indicará en la entrada *"Destination"*.

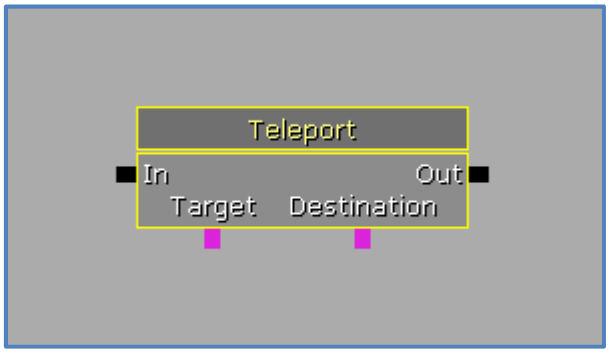


Imagen 84:Elemento "Teleport".

- **AI → Start Firing At**.- Aquellos elementos del nivel que posean la propiedad de *"disparo"*, se les podrá indicar que empiecen a disparar según la condición que se quiera.
  El actor que queremos que empiece a disparar se indicará en la entrada *"Target"* de la acción y a dónde ó quién disparar se indicará en la entrada *"Fire At"*.

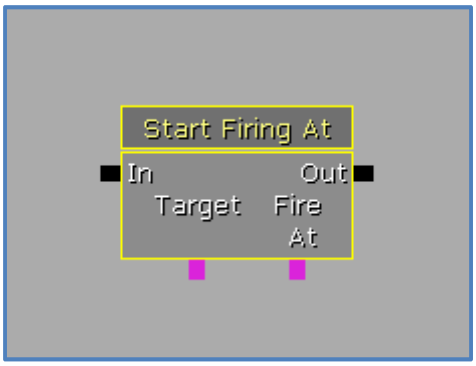


Imagen 85:Elemento "Start Firing At".

- **AI → Stop Firing**.- Al igual que se puede indicar a un actor empiece a disparar, comentado anteriormente, se puede indicar que deje de disparar.
  En este caso sólo hay que indicar el actor asociado a esta acción mediante la entrada *"Target"*.

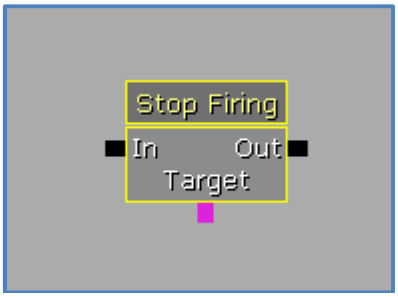


Imagen 86:Elemento "Stop Firing".

- **Camera → Look At**.- Mediante este elemento se puede indicar a cualquier personaje que mire hacia donde se desea. Se debe asociar dicho personaje al elemento mediante la entrada *"Target"* y el dónde mirar en la entrada *"Focus"*.

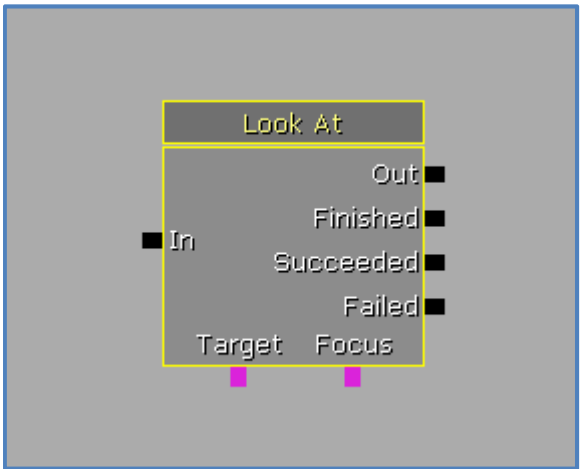


Imagen 87:Elemento "Look At".

- **Camera → Set Camera Target**.- Con este elemento se puede asociar la cámara que se desee al personaje que se tenga asociado.

Para ello se utilizan las entradas *"Target"*, para indicar el personaje al que se quiere asociar una nueva cámara, y *"Cam Target"* para indicar la cámara a asociar.

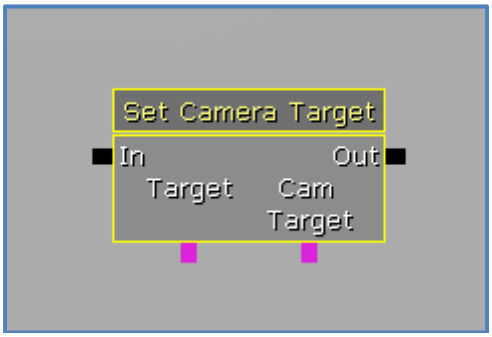


Imagen 88:Elemento "Set Camera Target".

- **Event → Activate Remote Event**.- Cuando se ha declarado un *"Remote Event"* podrá ser llamado cuando se desee. Para ello se utiliza este elemento, indicándole el *"Remote Event"* a realizar mediante el nombre de dicho evento. El nombre del *"Event"* se indica en el atributo *"Event Name"*.

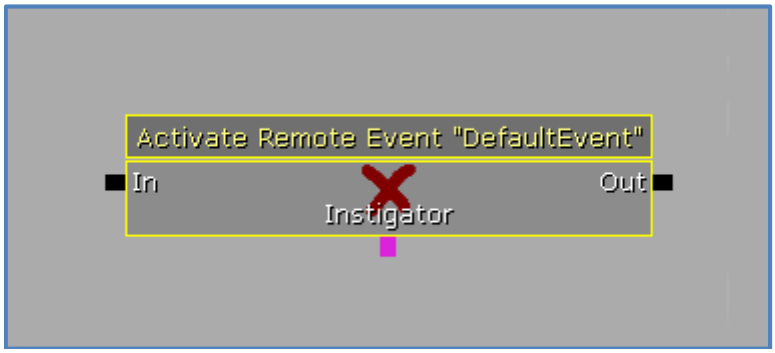


Imagen 89:Elemento "Activate Remote Event".

Cuando se le pasa un nombre de un *"Event"* que no corresponde con ningún *"Remote Event"* creado previamente, se muestra un aspa roja, mostrada en la imagen 89, indicando el hecho de error.
Cuando se proporciona un nombre de *"Remote Event"* correcto, se muestra un tick verde indicando que la llamada se hace correctamente, mostrada en la imagen 90.

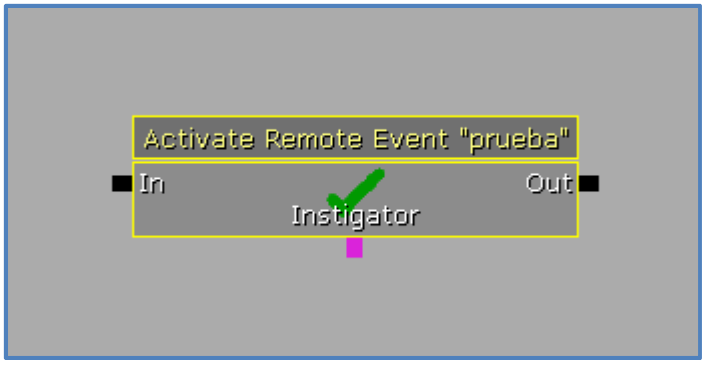


Imagen 90:Ejemplo asociación correcta con "Remote Event".

- **Event → Attach To Event**.- Mediante este elemento se puede asociar cualquier objeto del nivel a un evento. Se utilizarán las entradas correspondientes para poder asociar el objeto del nivel con el evento deseado.

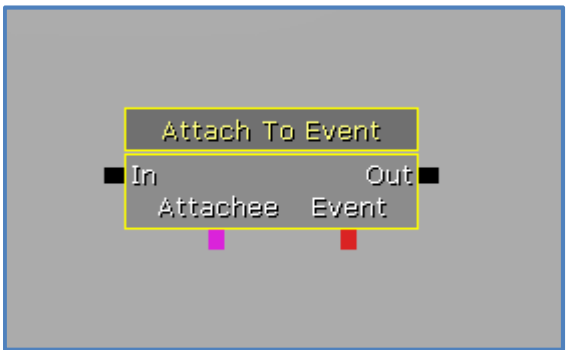


Imagen 91:Elemento "Attach To Event".

- **Math → Add Float**.- Mediante este elemento se pueden sumar dos variables de tipo *"Float"* y devolver el resultado en una variable de tipo *"Float"* ó en otra de tipo *"Int"*. En las distintas entradas se deben asociar las distintas variables para obtener el resultado.

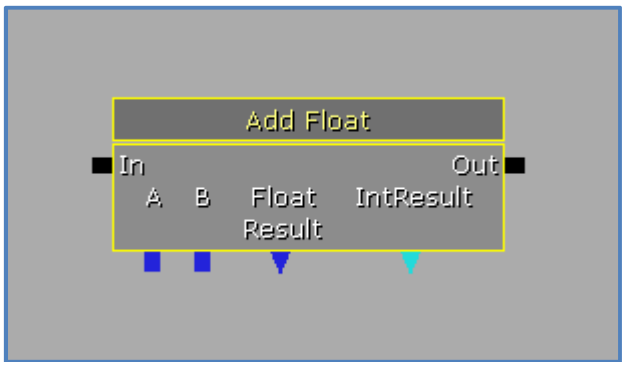


Imagen 92:Elemento "Add Float".

No es necesario tener las variables de entrada como objetos independientes. Si no se quiere tener otro elemento más con la/las variables, los valores a sumar se pueden introducir en las propiedades del elemento *"Value A"* y *"Value B"*. Lo que sí es necesario es acoplar una variable para el resultado de la operación y poder guardarla en ella para poder acceder más tarde a ella.

- **Math → Add Int**.- Este elemento es igual al anterior pero en este caso los elementos a sumar son de tipo *"Int"*. A su vez, el resultado obtenido se puede devolver en una variable de tipo *"Float"*.

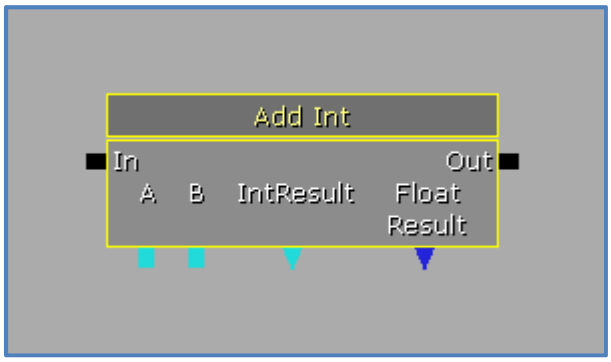


Imagen 93:Elemento "Add Int".

Los datos de entrada se pueden introducir mediante variables o introduciendo el valor deseado en los atributos del elemento.

- **Math → Cast To Float**.- Este elemento permite transformar una variable de tipo *"Int"* en su correspondiente de tipo *"Float"*.

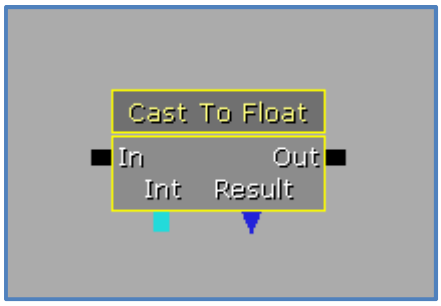


Imagen 94:Elemento "Cast To Float".

- **Math → Cast To Int**.- Este elemento permite transformar una variable de tipo *"Float"* a su correspondiente de tipo *"Int"*.
  Mediante este elemento se obtiene el número entero, despreciando los decimales que se tengan. En el caso que se quiera redondear al entero más próximo hay que poner tick en el atributo *"Truncate"* del elemento.

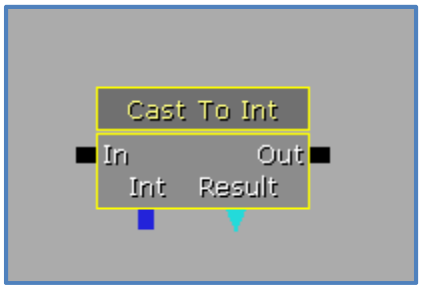


Imagen 95:Elemento "Cast To Int".

- **Math → Divide Float**.- Con este elemento se puede realizar la operación de división entre dos variables de tipo *"Float"*. El resultado se puede obtener también en una variable de tipo *"Int"* acoplándola en la correspondiente salida.

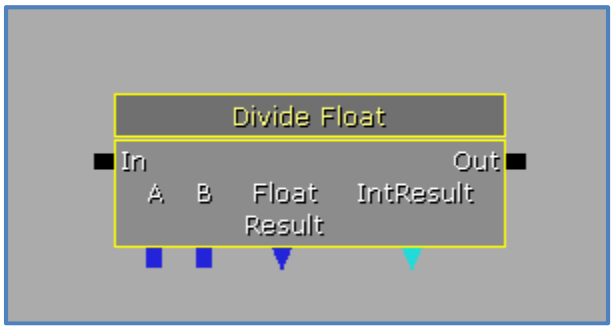


Imagen 96:Elemento "Divide Float".

- **Math → Divide Int**.- Con este elemento se puede realizar la operación de división entre dos variables de tipo *"Int"*. El resultado se puede obtener también en una variable de tipo *"Float"* acoplándola en la correspondiente salida.

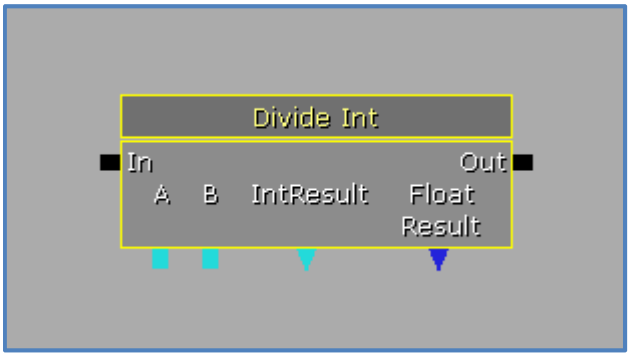


Imagen 97:Elemento "Divide Int".

- **Math → Multiply Float**.- Con este elemento se puede realizar la operación de multiplicación entre dos variables de tipo *"Float"*. El resultado se puede obtener también en una variable de tipo *"Int"* acoplándola en la correspondiente salida.

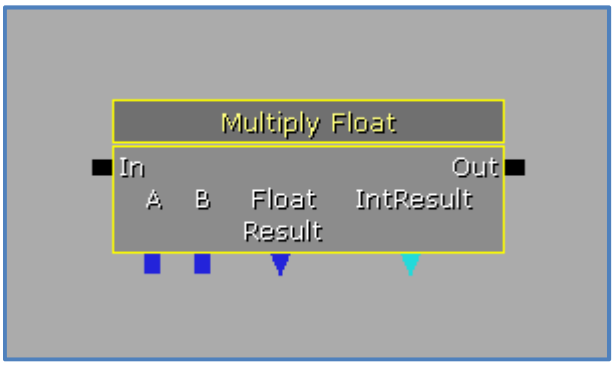


Imagen 98:Elemento "Multiply Float".

- **Math → Multiply Int**.- Con este elemento se puede realizar la operación de multiplicación entre dos variables de tipo *"Int"*. El resultado se puede obtener también en una variable de tipo *"Float"* acoplándola en la correspondiente salida.

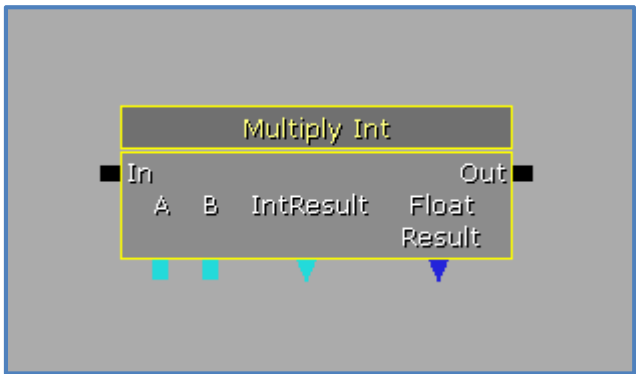


Imagen 99:Elemento "Multiply Int".

- **Math → Subtract Float**.- Con este elemento se puede realizar la operación de resta entre dos variables de tipo *"Float"*. El resultado se puede obtener también en una variable de tipo *"Int"* acoplándola en la correspondiente salida.

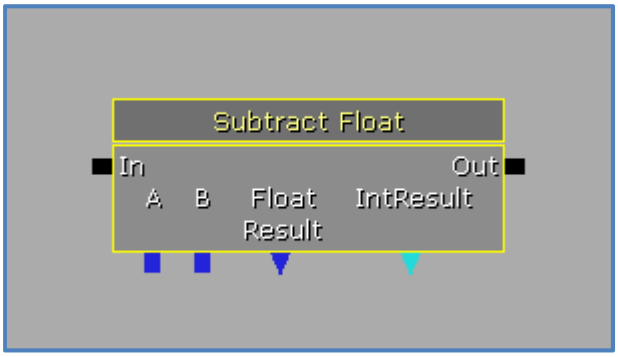


Imagen 100:Elemento "Subtract Float".

- **Math → Subtract Int**.- Con este elemento se puede realizar la operación de resta entre dos variables de tipo *"Int"*. El resultado se puede obtener también en una variable de tipo *"Float"* acoplándola en la correspondiente salida.

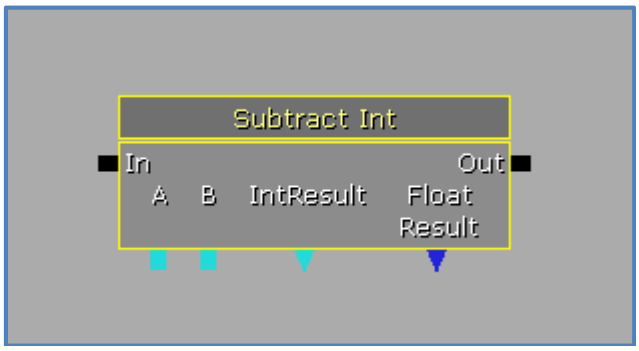


Imagen 101:Elemento "Subtract Int".

- **Misc → Console Command**.- Este elemento permite ejecutar los comandos de consola en ejecución. Los distintos atributos que se quieren ejecutar deben ser escritos en la etiqueta *"Commands"*, cada uno en una posición diferente. Por defecto, esta lista de comandos sólo tiene una posición, la posición 0. Para poder ampliar el número de posiciones hay que pulsar en el icono y aparecerán las nuevas posiciones, [1] [2]…, para utilizar.

En la entrada *"Target"* hay que indicar el objeto invocador de estos comandos, teniendo que indicar al jugador, mediante una variable del tipo *"Player"*.

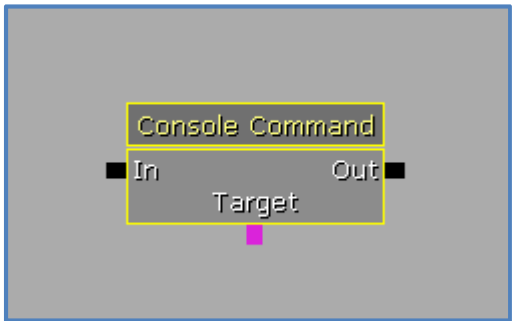


Imagen 102:Elemento "Console Command".

- **Misc → Delay**.- Este elemento permite realizar una pausa de tiempo en la ejecución. Los segundos que se quiere pausar la ejecución de indican en el atributo *"Duration"*. Otra forma de indicar la duración de esta pausa es utilizar una variable con el tiempo deseado y acoplarla a la entrada *"Duration"*.
  Mediante las entradas *"Start"* y *"Stop"* se indica cuándo se quiere que comience ó termine la toma de tiempo, respectivamente.

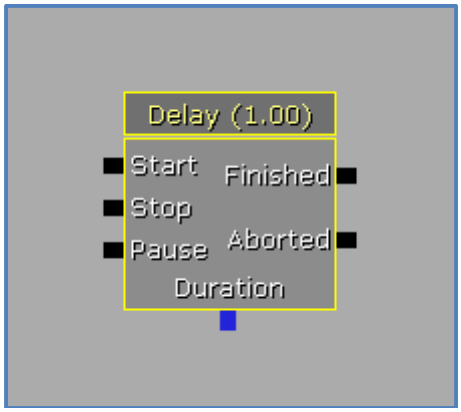


Imagen 103:Elemento "Delay".

- **Misc → Log**.- Hay veces que se quiere mostrar por pantalla el valor de alguna variable, ó el contenido de un objeto, etc. Para ello se puede utilizar este elemento.
  Al insertar dicho elemento sólo se tiene la entrada *"Target"* para poder acoplar algún elemento. En el caso que se quiera mostrar otro tipo de variable hay que:
  Pinchar con el botón derecho de ratón sobre el elemento → Expose variables → se mostrarán todos los tipo que se pueden acoplar.
  Cuando se hay elegido el tipo deseado, en el elemento aparecerá la entrada correspondiente al tipo para poder acoplarlo.

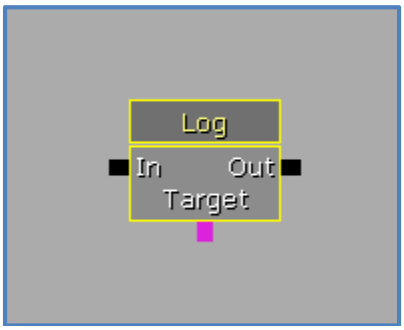


Imagen 104:Elemento "Log".

- **Misc → Timer**.- Gracias a este elemento se puede obtener el tiempo que ha transcurrido. Para ello se inicia la toma de tiempo mediante la entrada *"Start"* y se termina esta toma mediante la entrada *"Stop"* guardando este tiempo en una variable de tipo *"Float"* asociada en la entrada *"Time"*.

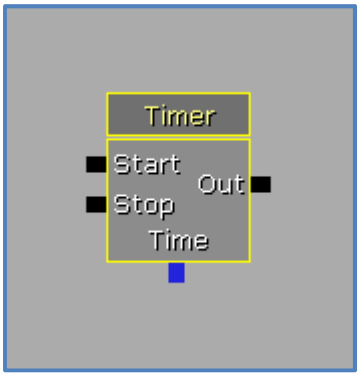


Imagen 105:Elemento "Timer".

- **Misc → Trace**.- Este elemento permite realizar la traza que existe entre dos objetos del nivel. Mediante esta traza se puede saber a la distancia que se encuentran y sobre todo si hay algún obstáculo entre ellos.
  Como se puede observar, las salidas del elemento son las dos circunstancias que se pueden dar: si hay obstáculos entre los objetos o no. Según esta circunstancia se puede realizar una acción u otra.
  En las entradas *"Start"* y *"End"* se indicarían los objetos a realizar la traza entre ellos.

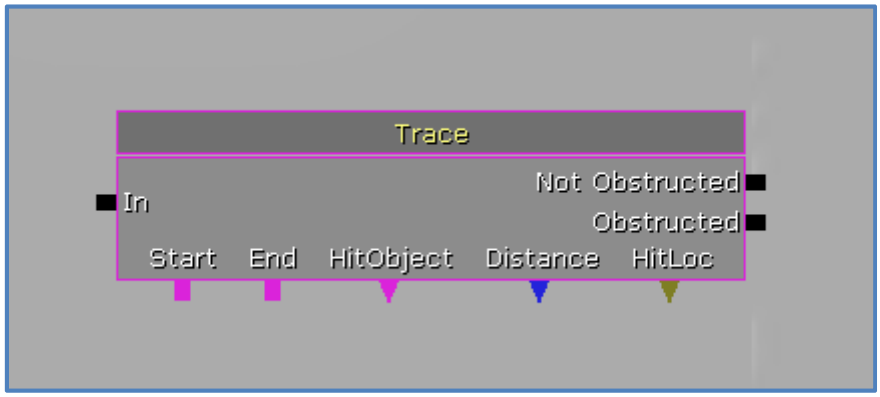


Imagen 106:Elemento "Trace".

- **Set Variable → Bool**.- Para poder modificar el valor de una variable *"Bool"* se utiliza este elemento. Se indica la variable a modificar, entrada *"Target"*, y el nuevo valor que se quiere que tenga, entrada *"Value"*.

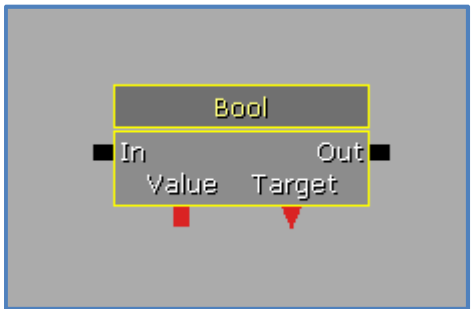


Imagen 107:Elemento "Bool".

- **Set Variable → Float**.- Para poder modificar el valor de una variable *"Float"* se utiliza este elemento. Se indica la variable a modificar, entrada *"Target"*, y el nuevo valor que se quiere que tenga, entrada *"Value"*.

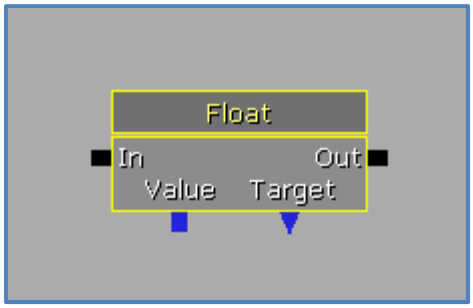


Imagen 108:Elemento "Float".

- **Set Variable → Int**.- Para poder modificar el valor de una variable *"Int"* se utiliza este elemento. Se indica la variable a modificar, entrada *"Target"*, y el nuevo valor que se quiere que tenga, entrada *"Value"*.

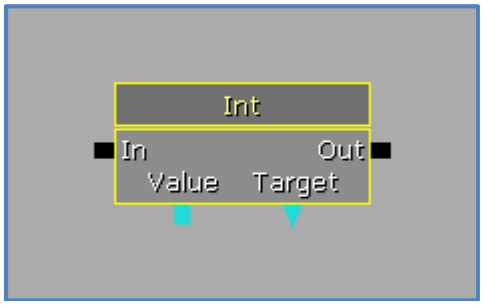


Imagen 109:Elemento "Int".

- **Set Variable → Object**.- Para poder modificar el valor de una variable *"Object"* se utiliza este elemento. Se indica la variable a modificar, entrada *"Target"*, y el nuevo valor que se quiere que tenga, entrada *"Value"*.

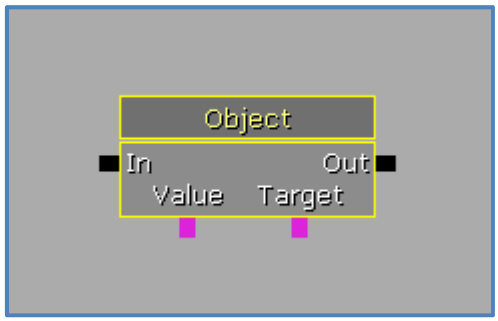


Imagen 110:Elemento "Object".

- **Set Variable → String**.- Para poder modificar el valor de una variable *"String"* se utiliza este elemento. Se indica la variable a modificar, entrada *"Target"*, y el nuevo valor que se quiere que tenga, entrada *"Value"*.

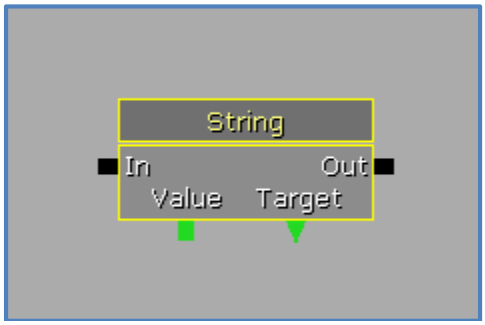


Imagen 111:Elemento "String".

- **Set Variable → Vector**.- Para poder modificar el valor de una variable *"Vector"* se utiliza este elemento. Se indica la variable a modificar, entrada *"Target"*, y el nuevo valor que se quiere que tenga, entrada *"Value"*.

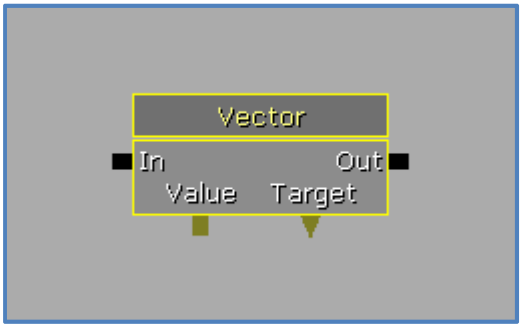


Imagen 112:Elemento "Vector".

- **Switch → Random**.- Este elemento permite obtener aleatoriedad a la hora de realizar un flujo de acciones u otro.
  El número de salidas deseado se indica en la propiedad *"Link Count"*. Según el número indicado aparecerán tantas salidas *"Link"* como se haya indicado.

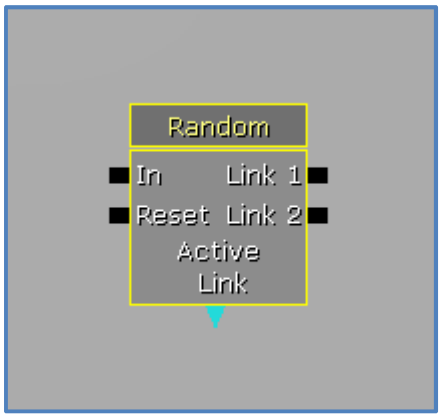


Imagen 113:Elemento "Random".

- **Toggle → Toggle**.- Mediante este elemento se pueden activar/desactivar todos aquellos objetos que tienen esta propiedad. El uso más común de este elemento es el apagar/encender las luces del nivel.
  Con la entrada *"Turn On"* se indica que se active el elemento asociado y mediante la entrada *"Turn Off"* se indica que se desactive dicho elemento. El elemento se debe asociar mediante la entrada *"Target"* utilizando una variable para dicho elemento.

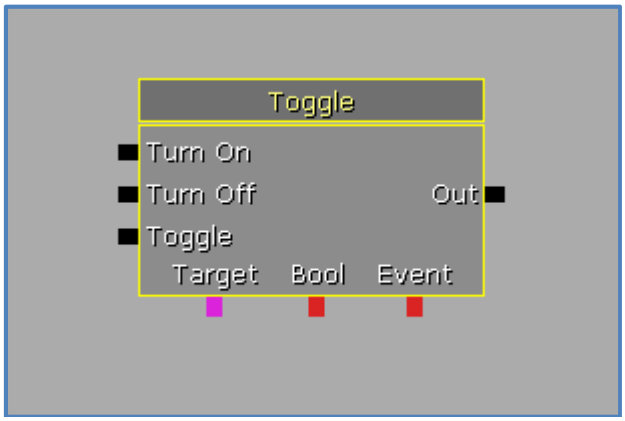


Imagen 114:Elemento "Toggle".

- **Toggle → Toggle Hidden**.- Con este elemento se puede mostrar u ocultar el objeto del nivel que se tenga asociado.
  Mediante la entrada *"Hide"* se indica que se oculte el objeto y mediante la entrada *"UnHide"* se indica que se muestre. Como en todos los elementos de Unreal Kismet, el objeto a asociar se realiza mediante la entrada *"Target"* utilizando una variable que represente al objeto deseado.

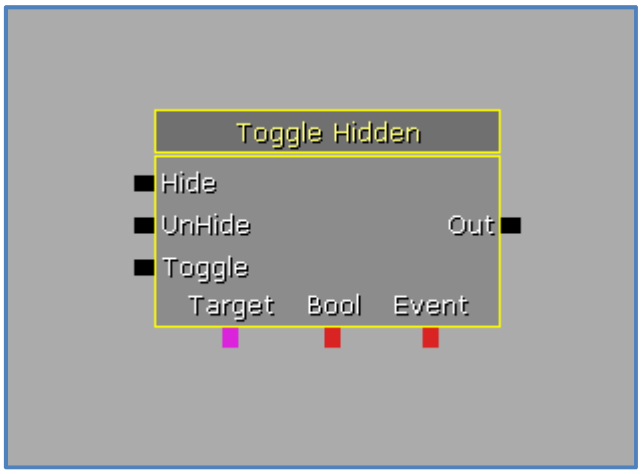


Imagen 115:Elemento "Toggle Hidden".

- **Toggle → Toggle HUD**.- Se denomina *"HUD"* al conjunto de elementos gráficos que son mostrados en la pantalla en la ejecución del nivel, como el número de balas, el nivel de vida, etc.
  Mediante esta acción se puede hacer que estos elementos gráficos se oculten, no ocupando espacio en la pantalla del jugador.
  Con la entrada *"Show"* se indica que los elementos gráficos se muestren y con la entrada *"Hide"* se indica que se oculten.

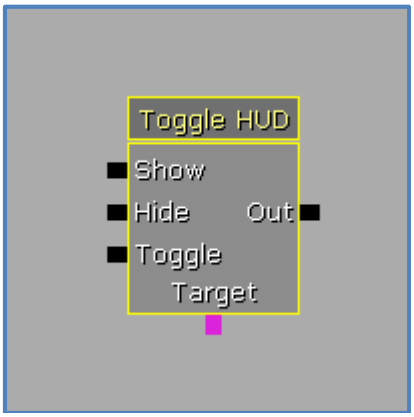


Imagen 116:Elemento "Toggle HUD".

- **UI Scenes → Open Scene**.- Las *"UI Scenes"* son objetos gráficos mostrados en ejecución en la pantalla del jugador. Estos objetos son pantallas con sus propios elementos (botones, texto, etc) que puede hacerse que aparezcan y desaparezcan.
  Para abrir estas pantallas se utiliza el elemento *"Open Scene"* el cual permite mostrar esta nueva pantalla en ejecución del nivel.
  Se debe indicar la pantalla a mostrar en el atributo *"Scene"* debiendo éste ser de tipo *"UI Scene"*.
  Para seleccionar el archivo *"UI Scene"* deseado hay que minimizar Unreal Kismet y abrir la herramienta Content Browser .
  Se selecciona el archivo *"UI Scene"* deseado, y con el archivo marcado como selección se vuelve a la herramienta Kismet, y en el atributo *"Scene"* se pulsa sobre el icono .

El archivo seleccionado en el Content Browser aparecerá en el atributo del elemento.

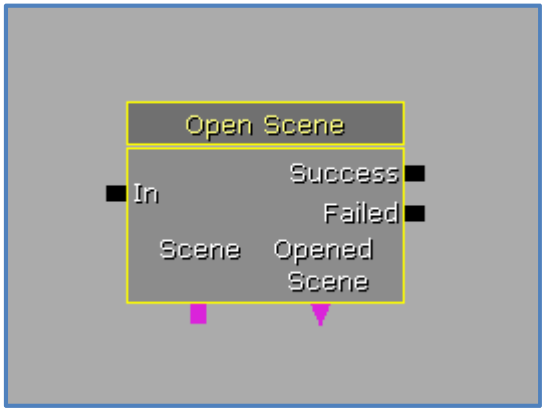


Imagen 117:Elemento "Open Scene".

- **UI Scene → Close Scene**.- Permite ocultar las pantallas mostradas como objetos *"UI Scenes"* en ejecución.
  En el atributo *"Scene"* se indica la *"UI Scene"* a cerrar, teniendo que ser del tipo *"UI Scene"*.
  Para seleccionar el archivo *"UI Scene"* deseado hay que minimizar Kismet y abrir la herramienta Content Browser.
  Se selecciona el archivo *"UI Scene"* deseado, y con el archivo marcado como selección se vuelve a la herramienta Kismet y en el atributo *"Scene"* se pulsa sobre el icono .
  El archivo seleccionado en Content Browser aparecerá en el atributo del elemento.

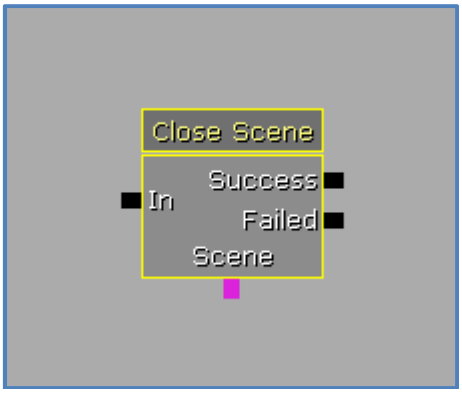


Imagen 118:Elemento "Close Scene".

- **Voice / Announcements → Play Announcement**.- Mediante este elemento se pueden mostrar anuncios sonoros y escritos durante la ejecución del nivel.
  El texto que se quiera que se muestre por pantalla se debe introducir en el atributo *"Announcement Text"* del elemento.
  Al mismo tiempo que el texto, se puede emitir un sonido. Para ello se debe indicar en el atributo *"Announcement Sound"* el archivo *".cue"* que se quiere reproducir.
  Para seleccionar el archivo de sonido hay que minimizar Kismet y abrir la herramienta Content Browser.

Se selecciona el archivo *".cue"* deseado y con el archivo marcado como selección se vuelve a la herramienta Kismet y en el atributo *"Announcement Sound"* se pulsa sobre el icono .
El archivo de sonido seleccionado en el Content Browser aparecerá en el atributo del elemento.

Imagen 119:Elemento "Play Announcement".

5.2.3. Variables.

Las variables permiten almacenar distintos elementos utilizados para realizar las acciones. Los tipos de variables son mostrados en la imagen 120.

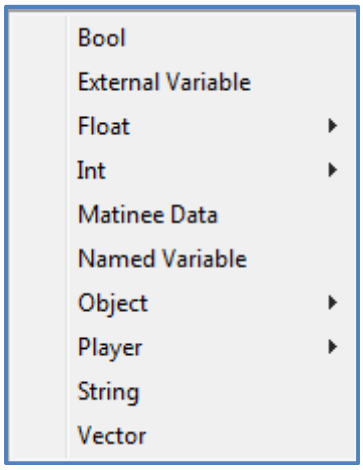


Imagen 120:Menú insertar variable.

En Kismet, cada tipo de variables se representa mediante un color, para su mejor identificación:

- Boolean .- Rojo.
- Integers.- Cian.
- Floats.- Azul.
- Objects.- Rosa.
- Strings.- Verde.
- Vectors.- Amarillo.

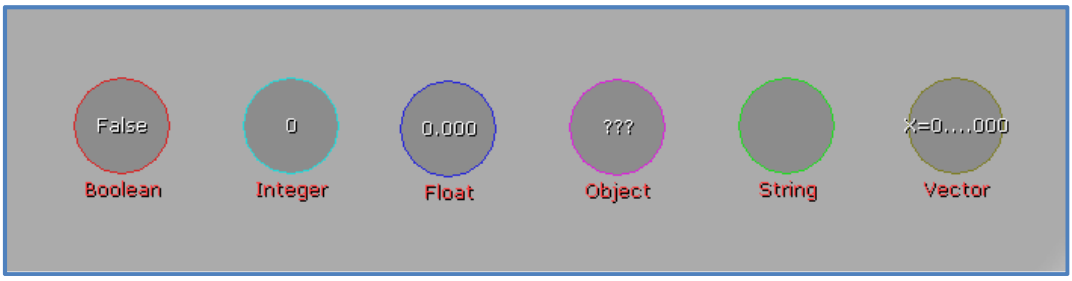


Imagen 121:Tipos de variables.

Como se puede observar, las variables se representan como círculos rodeados de su correspondiente aro de color y pudiendo acompañarlos con el nombre que se le quiere dar a esa variable.

5.2.4. Condiciones.

Las condiciones son utilizadas para comprobar si se ha producido algún hecho en la secuencia del nivel. Según se cumpla o no la condición, se podrán realizar unas acciones u otras.

Las condiciones que se nos permiten son mostradas en la imagen 122.

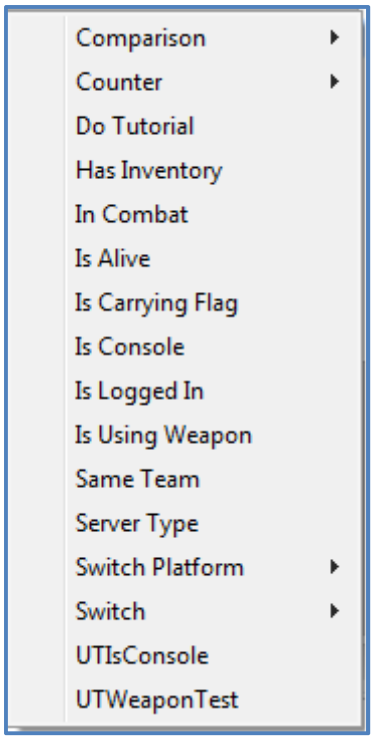


Imagen 122:Menú insertar condición.

Un ejemplo de un elemento de comparación es el comprobar si una variable boolean tiene un valor u otro, como el mostrado en la imagen 123.

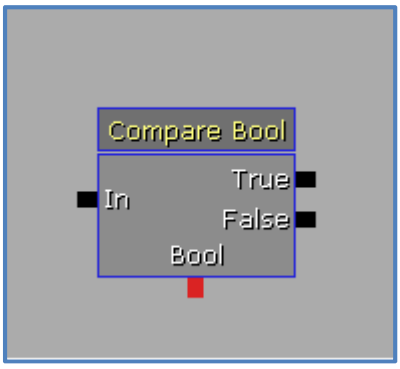


Imagen 123:Ejemplo elemento de comparación.

Las condiciones más importantes que se pueden utilizar son:

- **Comparison → Compare Bool**.- Mediante este elemento puede controlarse el valor que tiene una variable de tipo *"Bool"*.
  La variable será asociada mediante la entrada *"Bool"* y según el valor que tenga se tomará una salida u otra del elemento.

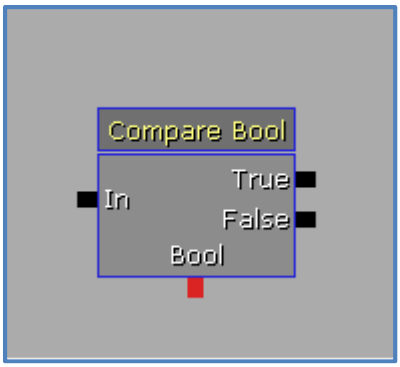


Imagen 124:Elemento "Compare Bool".

- **Comparison → Compare Float**.- Mediante este elemento puede controlarse el valor que tiene una variable de tipo *"Float"*.
  Hay dos opciones de comparar una variable de tipo *"Float"* con un valor dado: asociar al elemento una variable de tipo *"Float"* con el valor que se quiere comparar en alguna de las entradas, *"A"* ó *"B"*, asociando en la otra entrada la variable a comparar; y otra opción es introducir el valor a comparar en alguno de los atributos, *"Value A"* ó *"Value B"*, y asociar la variable a comparar en la entrada *"A"* ó *"B"* distinta a donde se puso el valor en los atributos.
  Según la comparación de los dos valores se activará una salida u otra siguiendo el flujo de ejecución por dicha salida.

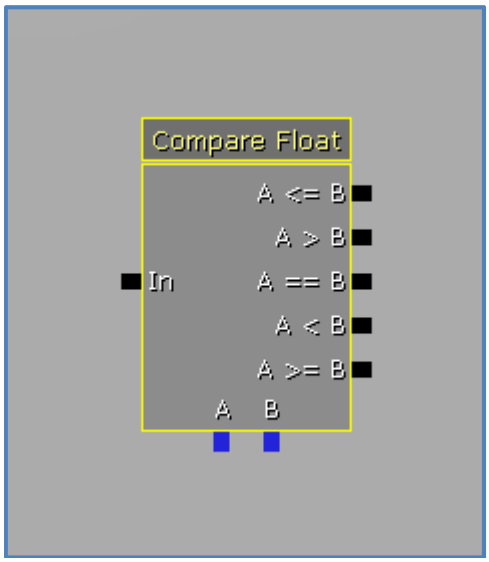


Imagen 125:Elemento "Compare Float".

- **Comparison → Compare Int**.- Mediante este elemento puede controlarse el valor que tiene una variable de tipo *"Int"*.
  Hay dos opciones de comparar una variable de tipo *"Int"* con un valor dado: asociar al elemento una variable de tipo *"Int"* con el valor que se quiere comparar en alguna de las entradas, *"A"* ó *"B"*, asociando en la otra entrada la variable a comparar; y otra opción es poner el valor a comparar en alguno de los atributos, *"Value A"* ó *"Value B"*, y asociar la variable a comparar en la entrada *"A"* ó *"B"* distinta a donde se puso el valor en los atributos.
  Según la comparación de los dos valores se activará una salida u otra siguiendo el flujo de ejecución por esa salida.

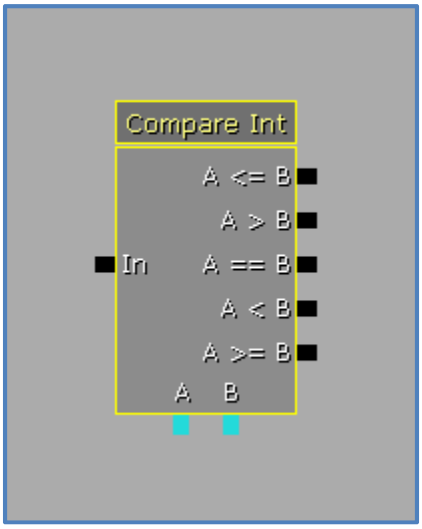


Imagen 126:Elemento "Compare Int".

- **Comparison → Compare Objects**.- Mediante este elemento se pueden comparar dos objetos para comprobar si son iguales o distintos.
  En este caso los dos objetos a comparar deben ser asociados mediante variables por medio de las entradas *"A"* y *"B"*.
  Las salidas que se tienen son sólo si los dos objetos son iguales o diferentes activándose la salida correspondiente.

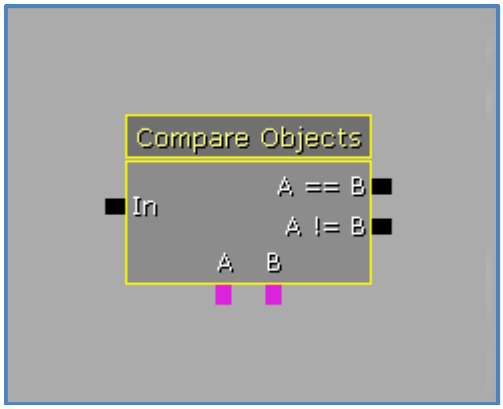


Imagen 127:Elemento "Compare Objects".

- **Counter → Float Counter**.- Permite realizar un contador de tipo *"Float"* y según la comparación de los valores se puede tener una salida u otra en la ejecución.
  Los valores de las variables se pueden asociar al elemento mediante las propiedades *"Value A"* y *"Value B"* ó mediante variables independientes utilizando las entradas *"A"* y *"B"*.

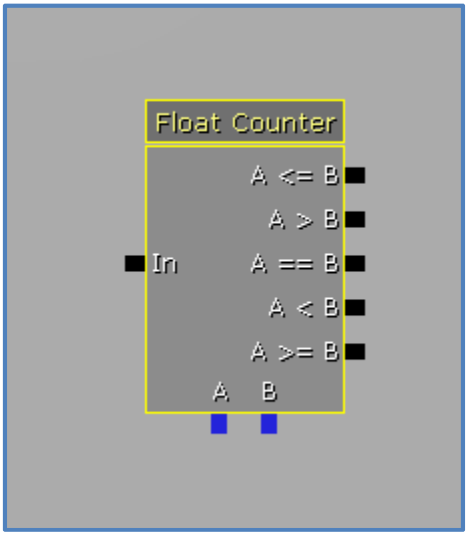


Imagen 128:Elemento "Float Counter".

- **Counter → Int Counter**.- Permite realizar un contador de tipo *"Int"* y según la comparación de los valores se puede tener una salida u otra en la ejecución.
  Los valores de las variables se pueden asociar al elemento mediante las propiedades *"Value A"* y *"Value B"* ó mediante variables independientes utilizando las entradas *"A"* y *"B"*.

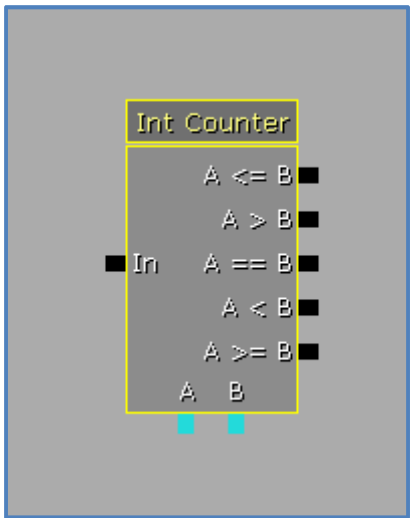


Imagen 129:Elemento "Int Counter".

- **Has Inventory**.- Se puede controlar si alguno de los actores del nivel posee algún objeto concreto en su inventario.
  El objeto que se quiere comprobar si el personaje lo posee en su inventario, se indica en la propiedad *"Requiered Inventory"* eligiendo en el desplegable el objeto a comprobar.
  En la entrada *"Target"* se acoplará la variable que contiene al actor a comprobar su inventario.

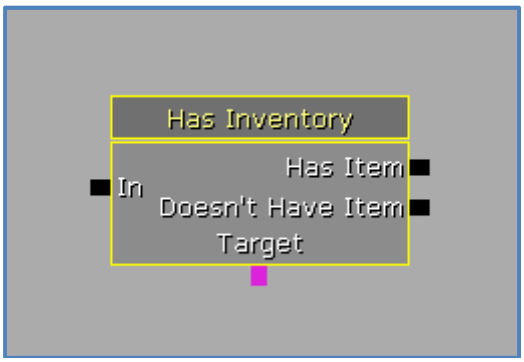


Imagen 130:Elemento "Has Inventory".

- **Is Alive**.- Se puede comprobar si un determinado personaje continúa con vida o no.
  En la entrada *"Players"* se debe acoplar la variable que represente al personaje que se quiere comprobar su vida.

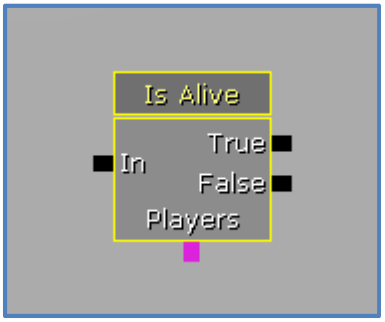


Imagen 131:Elemento "Is Alive".

5.2.5. Elemento Matinee.

Nos permite crear nuevos elementos de la herramienta Unreal Matinee. Dicha herramienta será explicada en detalle en el próximo capítulo.

Este tipo de elemento se muestra en la imagen 132.

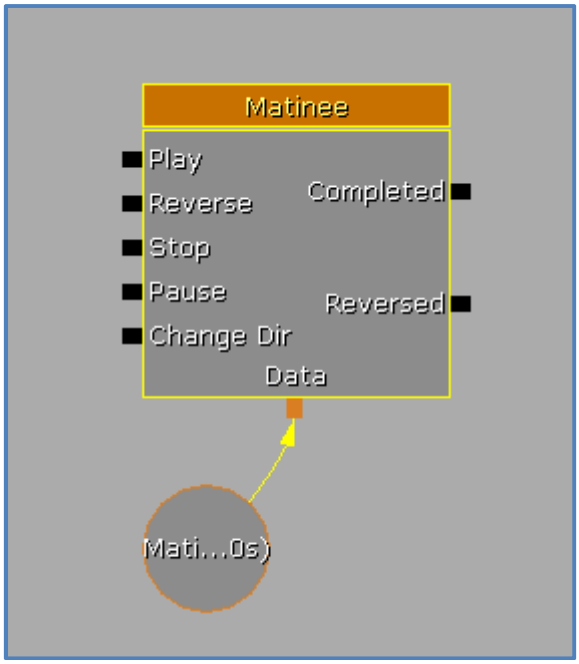


Imagen 132:Elemento "Matinee".

Mediante este elemento se puede:

- Reproducir la secuencia creada en la herramienta Unreal Matienee.
- Reproducir la secuencia creada inversamente.
- Parar la reproducción.
- Pausar la reproducción.

5.2.6. Etiquetas.

Otro elemento que se puede agregar es el que proporciona la forma de insertar etiquetas/comentarios en el diagrama.

Para ello se selecciona *"New Comment"* del menú principal (imagen 68) y se especifica el texto que se quiere dar a esa etiqueta.

Imagen 133:Elemento "Etiqueta".

En la imagen 133 se muestra un ejemplo de etiqueta en el diagrama, con el texto *"Etiqueta"*.

5.2.7. Agrupación de elementos.

Algunas veces se puede querer agrupar varios elementos por funcionalidad para tenerlos más identificados. La forma que se tiene en Unreal Kismet para agrupar varios elementos es creando un marco en el cual se acoplarán dentro todos los elementos que se quiera.

En el menú principal (imagen 68) se selecciona *"New Comment (Wrap)"*. Si se tienen todos los elementos a enmarcar ya seleccionados, aparecerá un marco conteniendo a todos los elementos. En el caso de no tener ningún elemento seleccionado aparecerá un marco simple en el que se podrá modificar su tamaño y posición para contener a los elementos que se quiera. Para modificar el tamaño hay que pinchar en la esquina inferior derecha, donde hay un triángulo negro, con el botón izquierdo del ratón y desplazar esa esquina para darle el tamaño deseado.

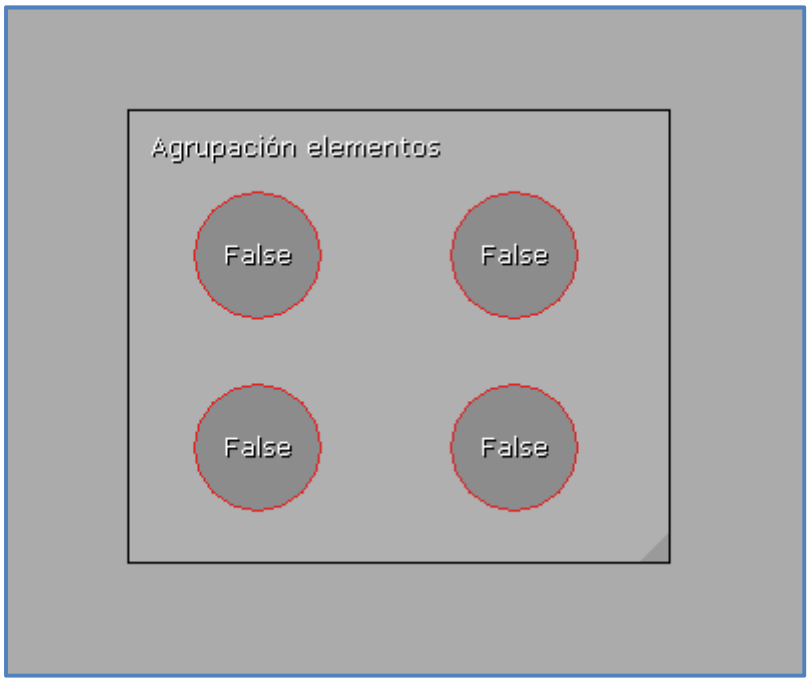


Imagen 134:Elemento "Agrupación de elementos".

5.2.8. Crear evento con objeto seleccionado.

Con cualquier objeto del nivel se puede crear un evento asociado a él. Los pasos a seguir para realizar el evento son:

- Seleccionar el objeto a crear el evento en alguna de las vistas del editor.
- Con este objeto seleccionado abrir la herramienta Kismet pulsando en el icono de la barra de herramientas superior del editor.
- Pinchar en alguna zona en blanco con el botón derecho del ratón.
- Se desplegará un submenú donde se seleccionará *"Create new sequence with selected object"*.

- Seleccionar el tipo de evento que se quiere realizar:
  - De contacto.- Dicho evento se producirá cuando el personaje principal haga contacto con el objeto.
  - De uso.- Dicho evento se produce cuando el personaje principal realice la función *"Usar"* dentro de acción del objeto.
  - De daño.- Cuando el objeto enlazado sufra algún daño.

5.3. Enlace entre elementos.

Todos los elementos de la herramienta Unreal Kismet serán enlazados entre ellos. Cada uno de ellos tiene una serie de entradas, donde se iniciará la correspondiente función del elemento, y unas salidas, valores obtenidos de la ejecución de la funcionalidad. Estas entradas y salidas son los cuadros negros que se pueden observar en los laterales como se muestra en la imagen 135.

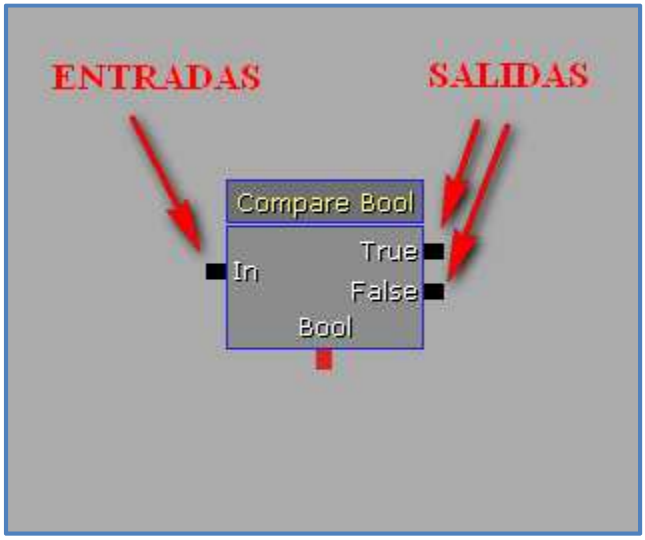


Imagen 135:Entradas y salidas de elemento.

Los restantes puntos que pueden tener los elementos son utilizados poder acoplar otros elementos necesarios para el funcionamiento. Lo más común es tener que acoplar alguna variable necesaria. Como en el caso del ejemplo anterior, la entrada *"Bool"* es para poder enlazar una variable del tipo *"Boolean"* y realizar su comparación.

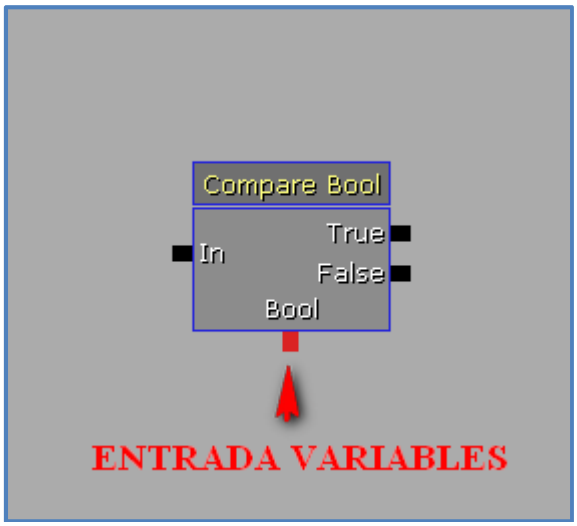


Imagen 136:Entrada de variables de elemento.

En estos puntos es donde se irán enlazando entre ellos mediante *"flechas"*. Estas flechas indican los flujos que se tienen en los cuales se irán realizando las operaciones que se encuentren. En la imagen 137 se muestra un enlace entre dos elementos de la herramienta.

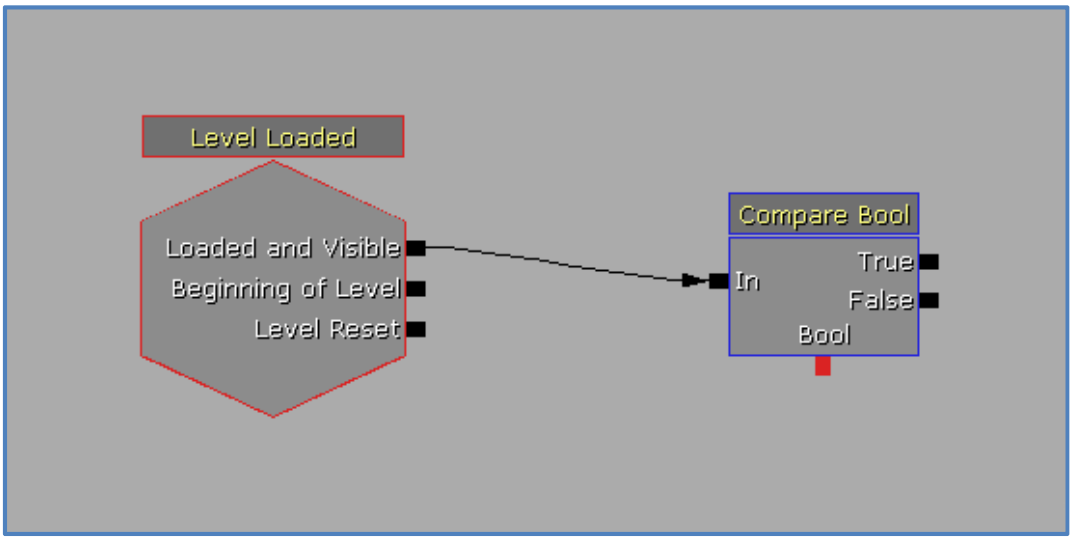


Imagen 137:Ejemplo enlace entre elementos.

Crear enlace.

Para poder crear este enlace hay que pinchar con el botón izquierdo del ratón sobre el primer extremo y sin soltar desplazarse al extremo final, quedando el enlace creado.

Borrar enlace.

Para poder borrar estos enlaces entre los elementos hay que mantener pulsada la tecla 'Alt' del teclado y pinchar con el ratón sobre uno de los extremos del enlace a borrar.

6. UNREAL MATINEE.

"Unreal Matinee" es una herramienta proporcionada por el motor *"Unreal"* que permite modificar las propiedades de los objetos Actor en ejecución, además de permitir realizar animaciones de ellos. [31]

Estas propiedades pueden ser la localización, el tamaño y otras propiedades propias de cada objeto, como color.

La interfaz de esta herramienta es mostrada en la imagen 138.

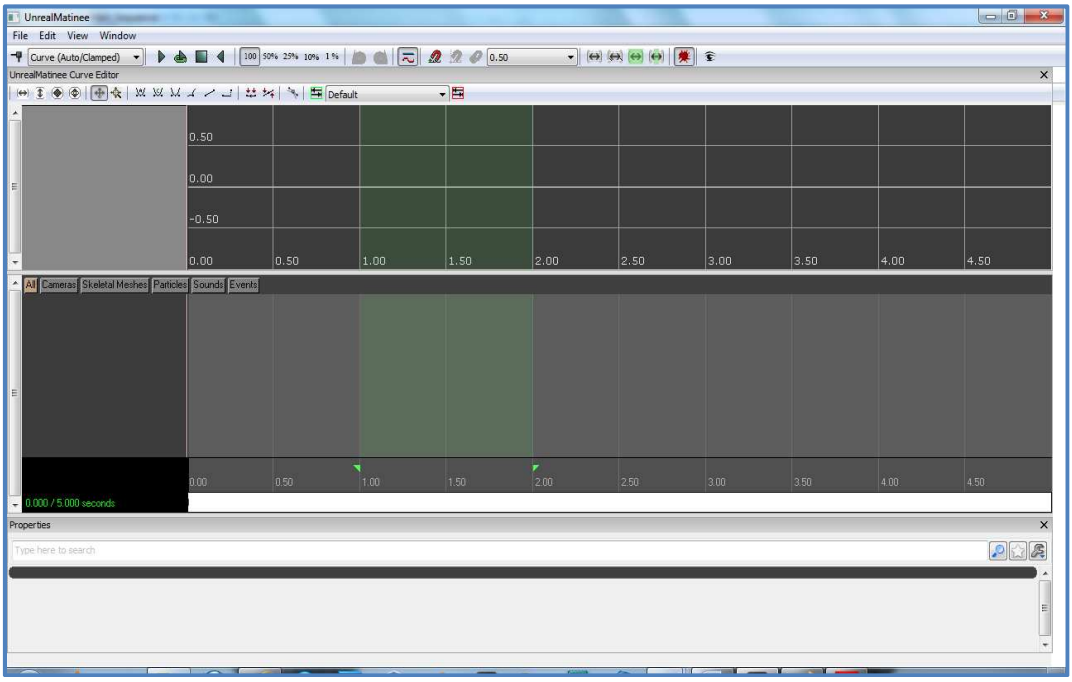


Imagen 138:Interfaz "Unreal Matinee".

Dicha herramienta puede abrirse mediante el icono de la barra de herramientas superior del editor ó al crear un elemento de este tipo en la herramienta Unreal Kismet y hacer doble click sobre él.

Cuando se quiere crear una animación primero hay que crear un *"grupo"*. Los grupos se irán creando en la zona inferior del *"UnrealMatinee Curve Editor"*:

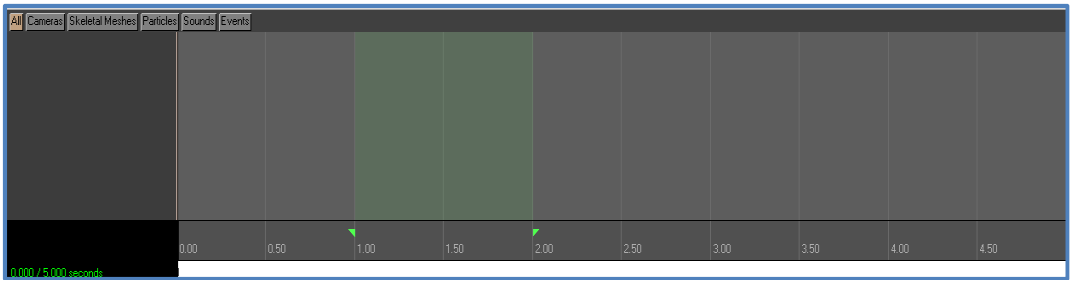


Imagen 139:Sección de grupos de interfaz "Matinee".

6.1. Creación de grupos.

En la parte izquierda de la sección de grupos es donde se podrán crear los distintos grupos. Para ello se debe pulsar el botón derecho del ratón mostrando un menú con todos los tipos de grupos a poder crear, imagen 140.

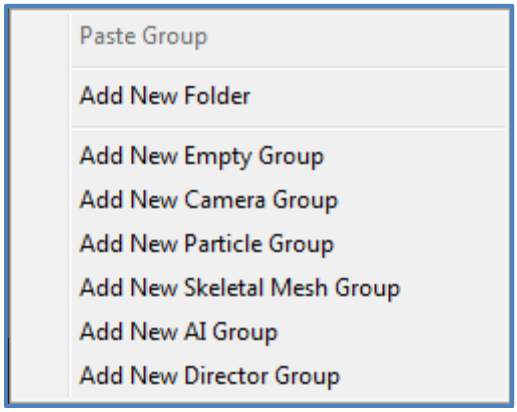


Imagen 140:Menú creación grupos.

Los tipos de grupos mostrados son:

- Empty.- Tipo de grupo general y vacío.
- Camera.- Grupo en que se modificará elementos tipo *"Camera"*.
- Particle.- Modificación de un objeto de tipo *"Particle System"*.
- Skeletal Mesh.- Grupo para poder modificar un objeto de tipo *"Skeletal Mesh"*.
- AI.- Modificación elementos de tipo *"AI"*.
- Director.- Para modificar la cámara principal del juego.

Al seleccionar el tipo de grupo se pedirá proporcionar el nombre a dar creándose con el nombre indicado, tal y como se muestra en la imagen 141:

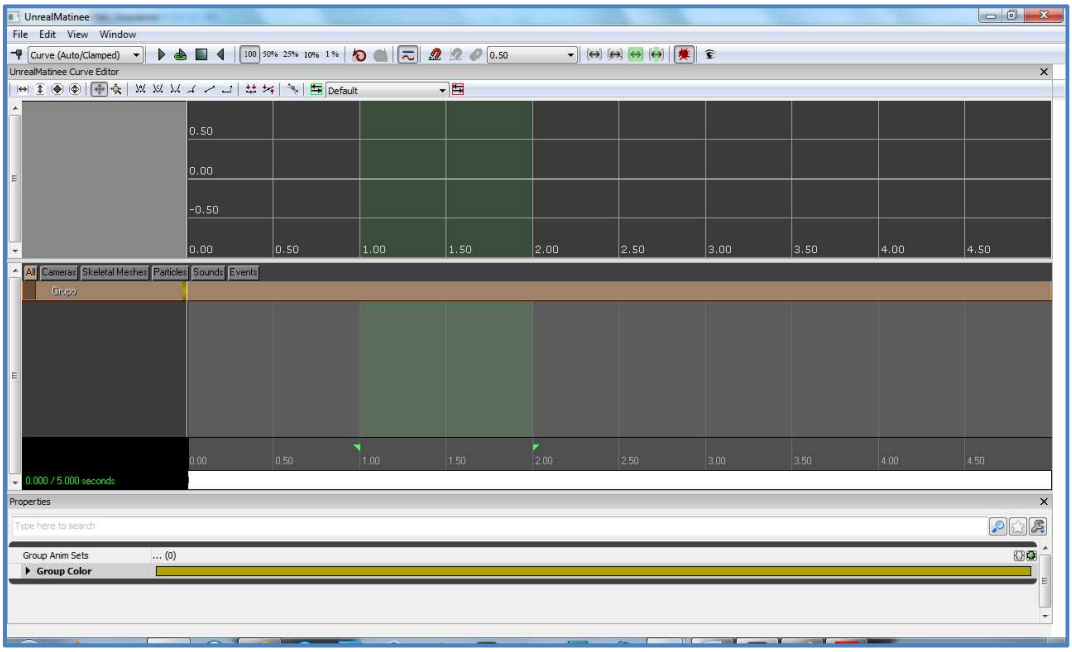


Imagen 141:Ejemplo de creación de grupo.

Creación de tracks.

Dentro de cada uno de los grupos hay que ir añadiendo *"tracks"* que serán los encargados de realizar la animación ó modificación del tipo que se quiera. Mediante estos *"tracks"* se podrá modificar la posición de un objeto del nivel, el color de un material, reproducir sónico, entre otras.

Seleccionando el grupo creado y pulsando el botón derecho del ratón sobre él, se mostrará un menú, imagen 142, con todos los *"tracks"* que se pueden agregar al grupo:

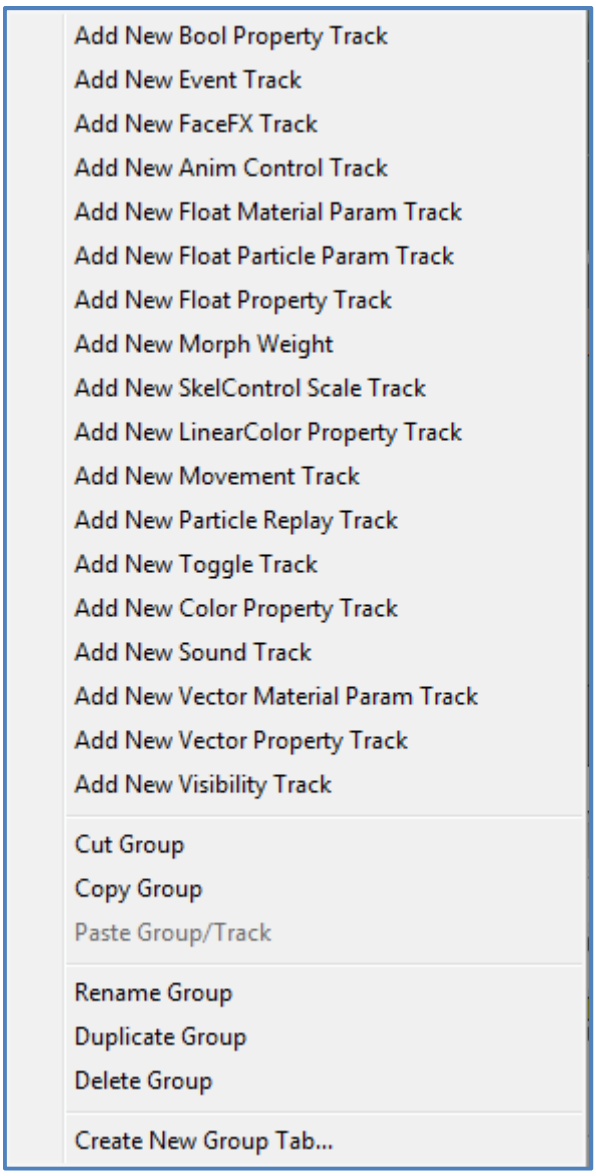


Imagen 142:Menú creación track en grupo.

Los distintos track son:

- Event.- Creación de nuevas salidas dentro del elemento Matinee de la herramienta Unreal Kismet.
- Anim Control.- Poder ejecutar una animación de un *"Anim Set"* perteneciente a un *"Skeletal Mesh"*.

- Movement.- Modificar la posición de un objeto mediante la animación.
- Sound.- Reproducir un sonido. El archivo de sonido para poder reproducirse debe ser de extensión *".cue"*.
- Vector Material Param.- Para modificar el color del material de un objeto del nivel.

Los *"tracks"* creados se mostrarán junto al grupo al que pertenecen, mostrando ejemplos de estos *"tracks"* en la imagen 143.

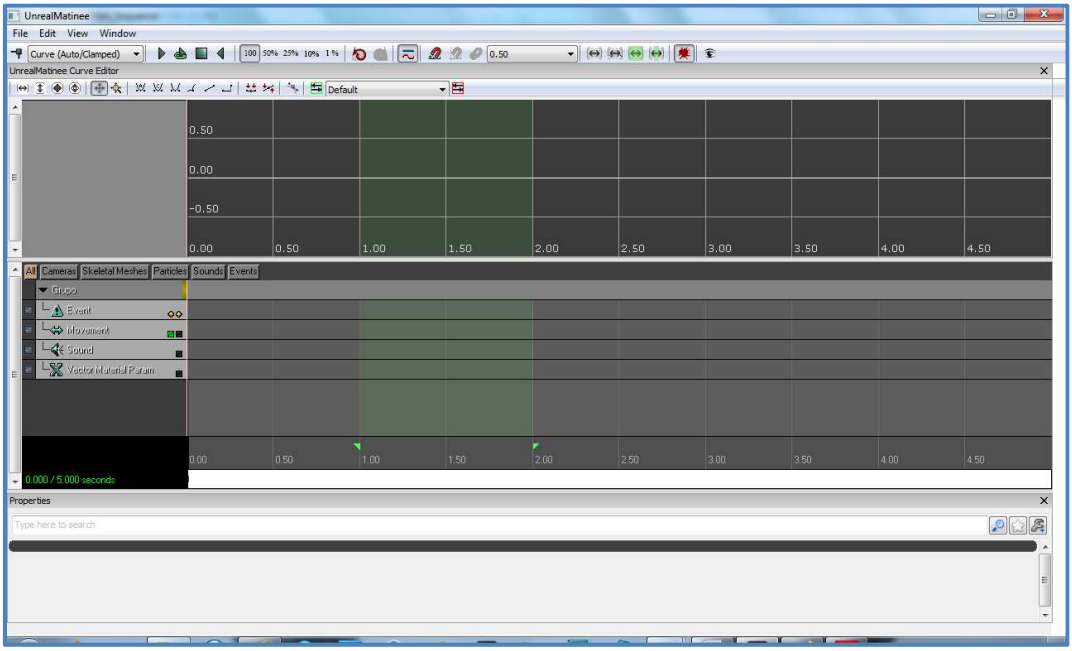


Imagen 143:Ejemplo creación tracks en grupo.

Cada uno de los grupos, en su nombre, posee en la parte derecha un pequeño cuadrado que activándolo se muestran los valores de cada *"track"* en la parte superior del *"UnrealMatinee Curve Editor"*, imagen 144.

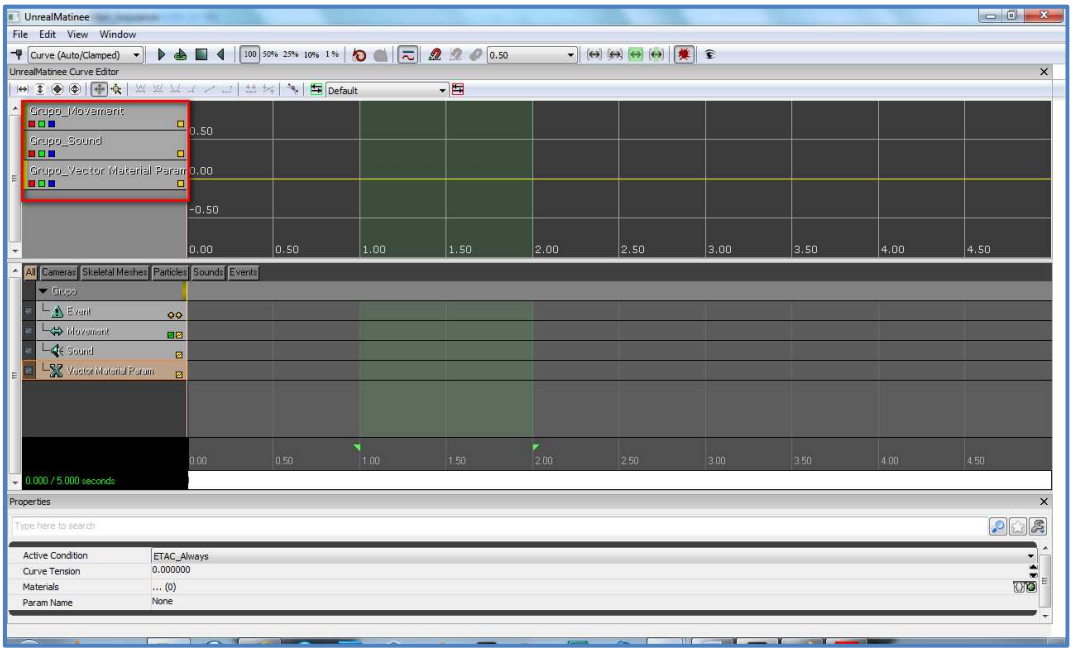


Imagen 144:Sección "Curve Editor" de "Matinee".

6.2. Secuencia de tiempo.

En Unreal Matinee se muestra la secuencia de tiempo en la que se irán modificando los distintos elementos asociados con los *"track"*, según el tipo que sean.

Esta secuencia de tiempo es mostrada en la interfaz del editor:

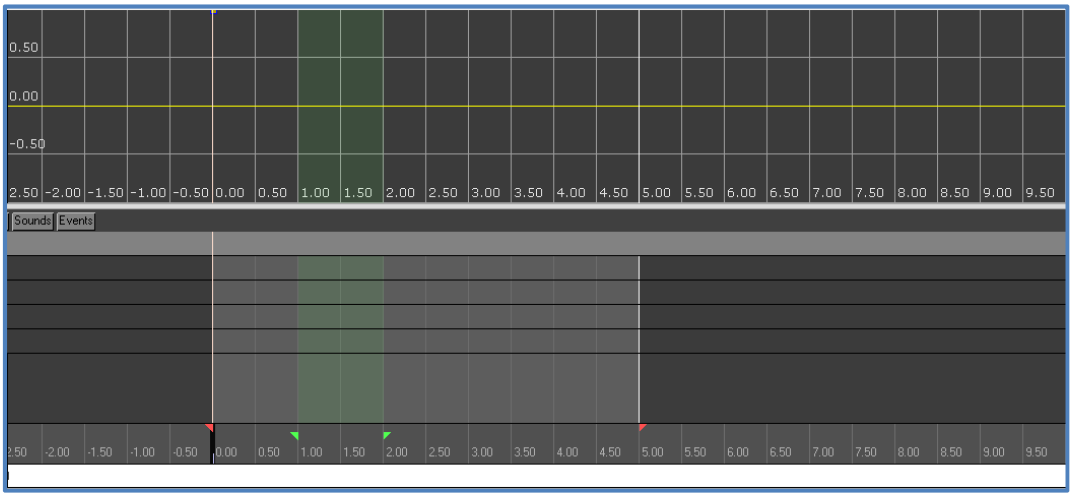


Imagen 145:Sección de secuencia de tiempo en "Matinee".

La secuencia de tiempo se reproducirá entre los dos puntos de tiempo marcados con un triángulo rojo, mostrados el inicio y el fin en la imagen 146. Se puede apreciar también los límites de tiempo porque el color del fondo es de un color distinto que el resto.

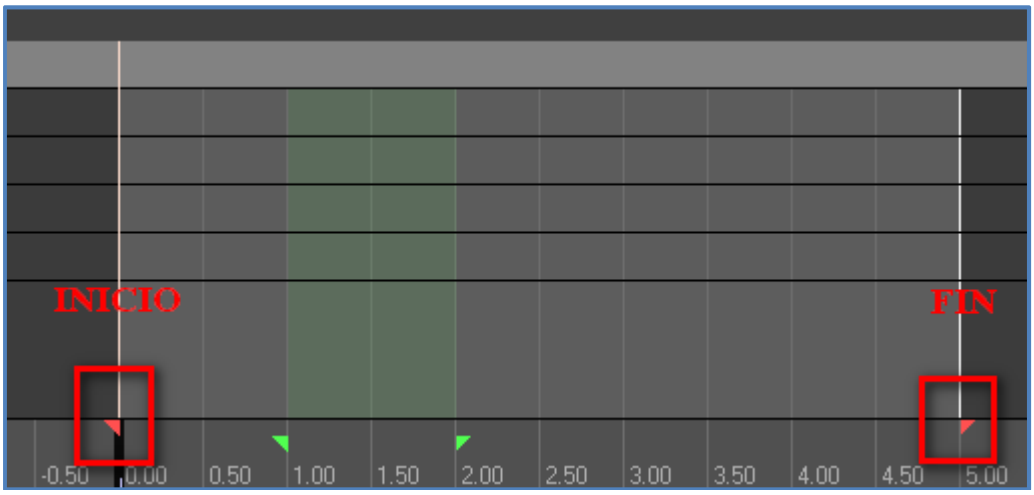


Imagen 146:Inicio y fin de secuencia de tiempo.

Estos puntos pueden ser modificados para ampliar o reducir el tiempo. Para ello se pulsa el botón principal del ratón sobre ellos y sin dejar de pulsar se mueve el ratón en el sentido que se desee para modificar el punto.

A su vez, para concretar mucho más el tiempo, se puede hacer zoom sobre el tiempo para obtener puntos más detallados o menos. Se utiliza la rueda central del ratón en un sentido y en otro para hacer zoom más cercano ó más lejano de los puntos temporales.

También se puede hacer un desplazamiento a lo largo de la línea temporal para ir al punto deseado. Se pulsa el botón principal del ratón sobre el tiempo, y sin dejar de pulsar, se realiza el movimiento del ratón para desplazar la línea temporal hasta encontrar el tiempo deseado.

En cada uno de los *"track"* se pueden modificar los elementos asociados mediante una animación ó modificación.

Estas modificaciones se especifican en un tiempo concreto de tiempo. Para elegir el tiempo en que se tiene que realizar la modificación se tiene una *"línea temporal"*, imagen 147, que permite desplazarla al tiempo deseado e indicar que en ese punto se debe realizar la modificación.

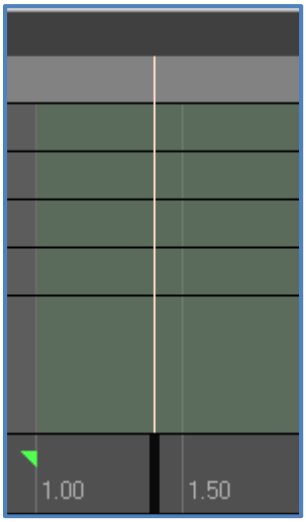


Imagen 147:Línea de tiempo.

Esta *"línea temporal"* puede ser modificada pulsando con el botón izquierdo del ratón sobre la parte de color negro y desplazando en un sentido o en otro para llevarla al tiempo deseado.

Para realizar las modificaciones en cada *"track"* hay que crear *"puntos de control"* en los cuales se indica que se realice la modificación indicada.

Se puede crear un *"punto de control"* seleccionando el *"track"* donde crearlo, y teniendo la *"línea temporal"* en el tiempo deseado, se pulsa la tecla 'Enter' del teclado. En la misma *"línea temporal"* aparecerá un triángulo de color rojo indicando que se ha creado un *"punto de control"* y pudiendo realizar modificaciones de los elementos asociados al *"track"*. En la imagen 148 se muestra un ejemplo de creación de un punto de control en un *"track"* de tipo *"Event"* a los 1:20 segundos.

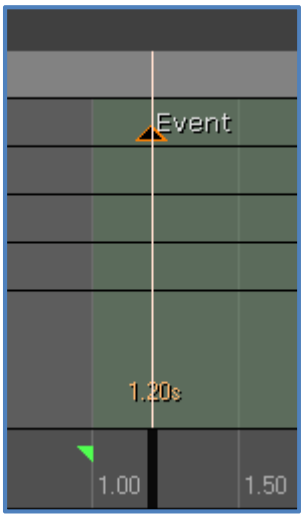


Imagen 148:Ejemplo de creación de punto de control.

Otra forma de crear estos puntos de control es mediante el icono de la barra de herramientas superior del editor, en la parte izquierda.

Al crear un *"punto de control"* se pueden mostrar mensajes de error indicando que el elemento asociado al *"track"* no es el adecuado para realizar las modificaciones del tipo indicado por dicho *"track"*.

Pulsando con el botón derecho del ratón sobre el triángulo que representa el *"punto de control"* se obtienen opciones para poder borrarlo y/o concretar el tiempo en que se quiere posicionar.

Con la animación creada puede comprobarse su funcionamiento reproduciéndola.

En la barra de herramientas superior del editor se tienen los iconos necesarios para reproducirla: .

En aquellas animaciones que lo permitan, se pueden comprobar las modificaciones que se realizan en el editor mientras se está reproduciendo, minimizando la herramienta Unreal Matinee y observando las vistas de Unreal Editor.

6.3. Elemento Matinee en la herramienta Unreal Kismet.

Cuando se ha desarrollado toda la animación y/o animación, el editor puede ser cerrado quedando todos los cambios guardados en el elemento de la otra herramienta Unreal Kismet.

En este elemento de Kismet se tienen las entradas necesarias para poder reproducir toda la animación, mostradas en la imagen 149.

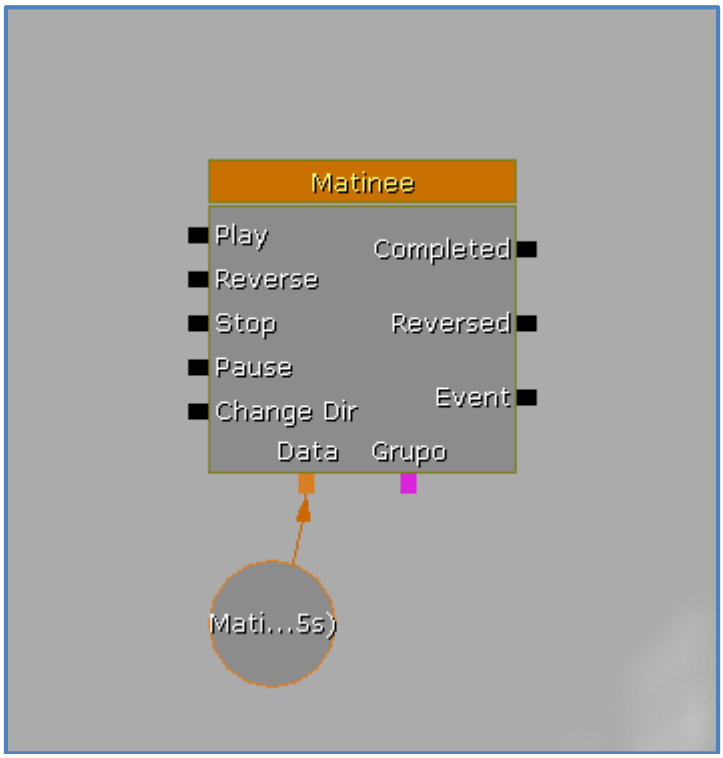


Imagen 149:Ejemplo elemento "Matinee".

Las entradas que se tienen son:

- Play.- Reproduce la animación.
- Reverse.- Reproduce la animación pero en sentido inverso.
- Stop.- Para la reproducción de la animación.
- Pause.- Pausa la reproducción de la animación.

Y las salidas son:

- Completed.-. Se activa cuando la reproducción ha finalizado.
- Reversed.- Se activa cuando se reproduce la animación pero en sentido inverso.

Se pueden crear distintas salidas en este elemento mediante el "*track*" de tipo "*Event*" en el tiempo deseado para poder ejecutar distintas acciones durante la reproducción de la animación.

En la parte inferior se muestran los distintos grupos creados y los objetos del nivel asociados a ellos mediante variables que los representan.

Este elemento puede ser asociado a cualquier otro elemento para realizar distintas acciones a la vez que lanzado por cualquier evento producido en el nivel.

7. ELEMENTOS BÁSICOS DE UN NIVEL.

En este capítulo se van a exponer todos los elementos básicos necesarios para crear un nuevo nivel en el motor Unreal Development Kit. [32] [33]

7.1. Aparición del personaje principal.

En el nivel se debe indicar dónde se quiere que aparezca el personaje principal al lanzarlo.Este punto de aparición del personaje será marcado por un elemento propio en el editor que podrá ser desplazado según conveniencia, mostrado en la imagen 150. [34]

Imagen 150:Player Start.

En este elemento se muestra una flecha la cual representa la orientación con la que aparecerá el personaje principal.

Para acoplar este elemento al nivel se deben seguir los siguientes pasos:

- Pulsar con el botón derecho del ratón sobre el punto donde se quiere que aparezca en cualquier vista del editor mostrándose el menú mostrado en la imagen 151.

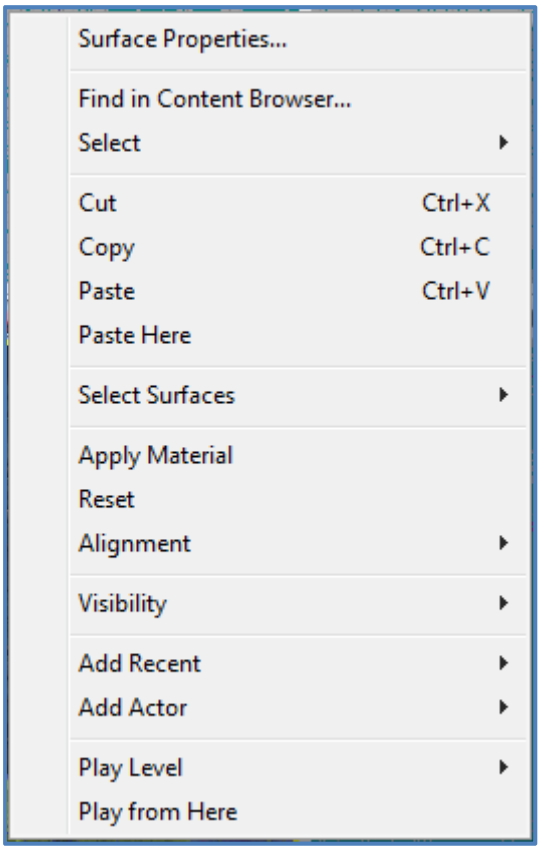


Imagen 151:Menú botón secundario en editor.

- Abrir submenú de la etiqueta *"Add Actor"* del menú contextual mostrado:

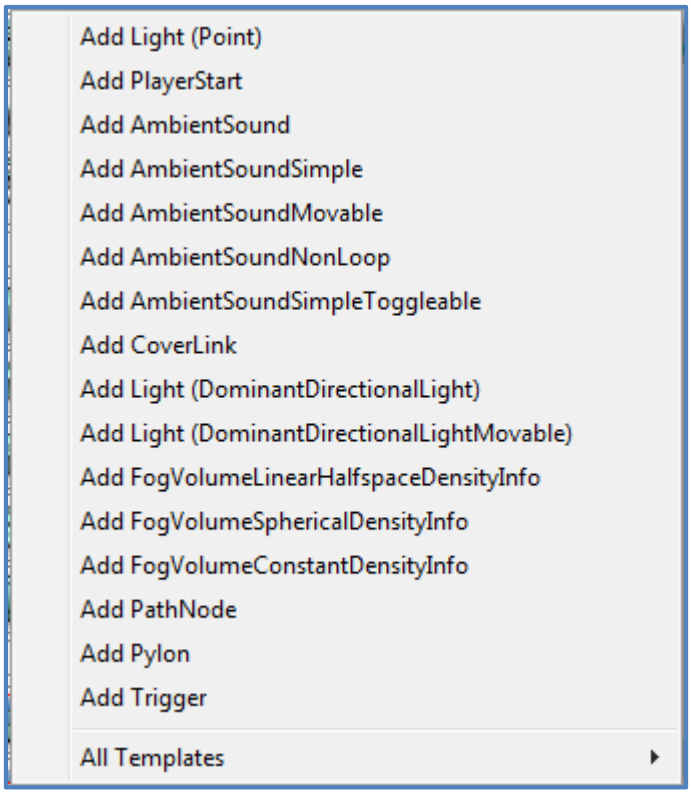


Imagen 152:Menú "Add Actor".

- Pinchar en la etiqueta *"Add PlayerStart"* para que en el editor aparezca el elemento deseado:

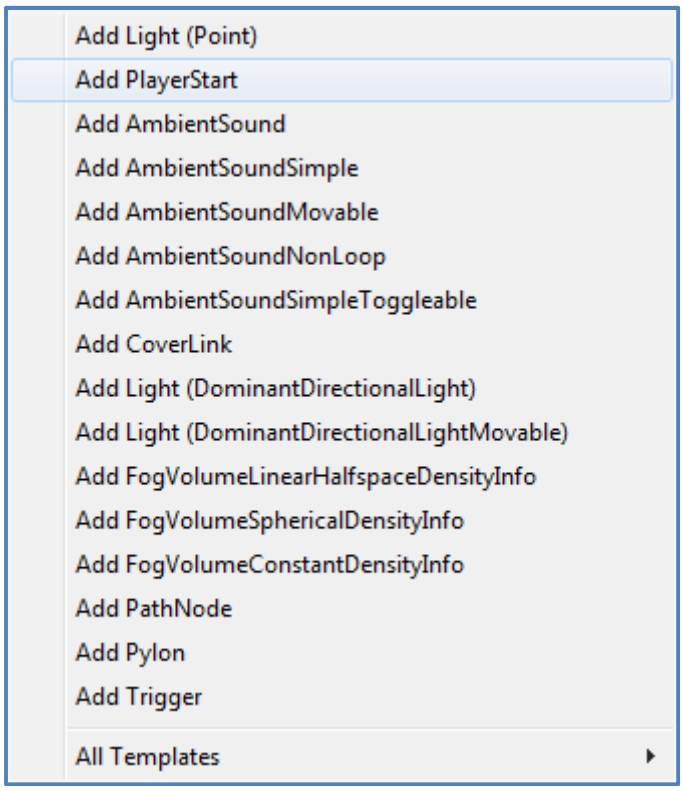


Imagen 153:Insertar "PlayerStart".

- Con el objeto ya acoplado en el nivel, se puede trasladar activando el icono , Traslation Mode, de la barra de herramientas superior para situarlo en el lugar deseado.

7.2. Iluminación.

En todo nivel es necesario acoplar algún tipo de iluminación. Hay distintos tipos de iluminiación que pueden ser acoplados: [35]

- Punto de luz.
  Típico punto de luz con un radio de acción concéntrico.
- Foco.
  Dicho tipo de luz representa un foco de luz, con lo que sólo se iluminará aquello a lo que el foco está apuntando.
- Luz ambiental.
  Este tipo de luz es utilizado normalmente con niveles exteriores, ya que es un tipo de luz con un radio de acción muy amplio.

Una característica que hay que tener en cuenta a la hora de acoplar la iluminación al nivel es si en ejecución se va a querer modificar, como apagar/encender, cambiar de color, etc. Para este caso hay que utilizar una característica especial llamada *"Toggleable"*. Por cada uno de los tipos de luces comentados anteriormente se tiene un nuevo tipo exactamente igual pero con la propiedad de cambio.

Para poder observar y/o modificar las características de la iluminación es necesario acceder a ellas. Para poder acceder a las características se puede realizar de varias maneras, teniendo seleccionado el objeto previamente:

- Pulsando la tecla 'F4' del teclado.
- Presionando el botón derecho del ratón y en el submenú que se despliega seleccionar la primera opción: *"objeto Properties…"* (imagen 154).

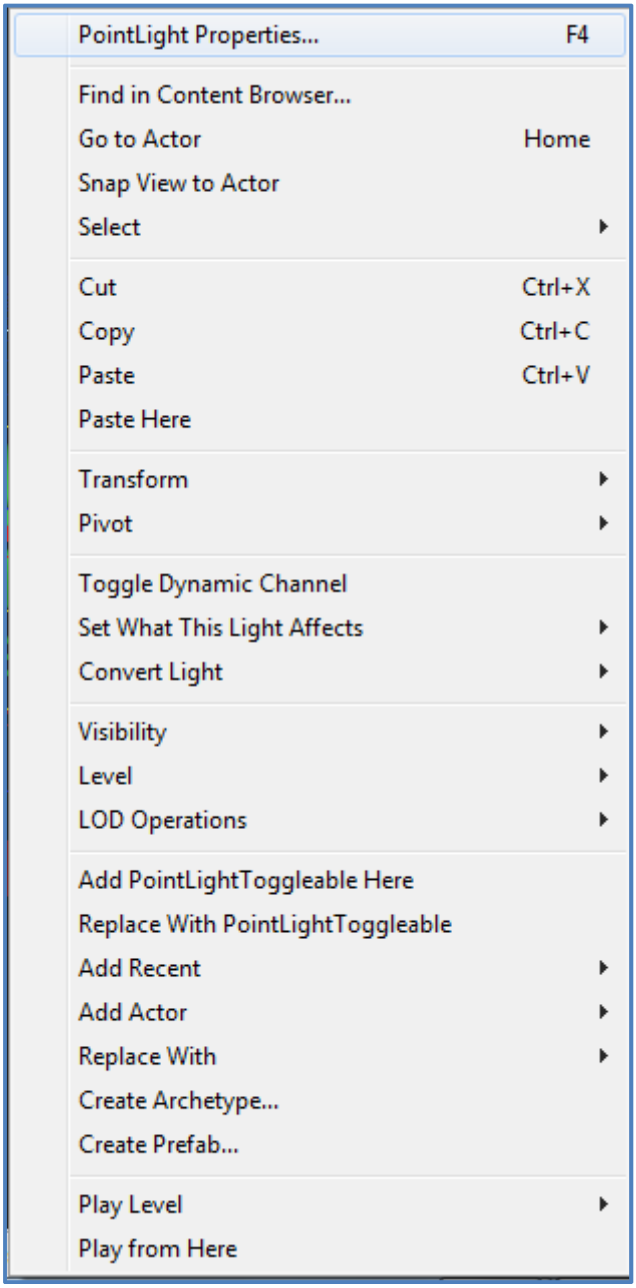


Imagen 154:Menú secundario elemento de iluminación.

En dichas características se pueden modificar diversos aspectos de las luces, y entre otros los más importantes son:

- Color de emisión: Mediante esta característica se puede modificar el color de emisión de la luz.

Para ello hay que ir a *Ligth → Light Component → Light Component → Ligth Color.*

Para elegir el color se tienen dos formas de realizarlo:

- Al pinchar sobre la barra de color blanco, se abrirá un editor de color presentando una paleta muy completa de colores donde se podrá seleccionar el color deseado.
- Al desplegar dicha etiqueta se podrá indicar el color mediante sus componentes RGB (Red – Green – Blue). Hay que tener en cuenta que todos los valores de los colores están entre el negro (0 - 0 - 0) y el blanco (255 – 255 – 255). Otro componente que se puede modificar es la componente del brillo de este color en la etiqueta A.

- Canal de emisión de la luz: El editor *Unreal Editor* permite elegir distintos canales de emisión de la luz.
  Se puede elegir el canal desplegando: *Light → Light Component → Light Component → Lighting Channels.*
  Sólo con poner un tick en el/los canal/es deseados para emitir en ellos.
  Se debe tener en cuenta que para que los objetos creados sean vistos deben estar en el mismo canal en que la luz está emitiendo.

- Brillo: Se puede seleccionar la intensidad del brillo que se quiere que emita la luz.
  Para elegir el brillo hay que: Light → Light Component → Light Component → Brightness.
  En este caso hay que indicar el valor indicado del brillo que se desea.

- Radio de intensidad: Con esta característica se puede modificar el radio de acción de la luz.
  Dicho radio se indica en *Light → Light Component → ObjetoComponent → Radius.*
  Se tiene que indicar el valor del radio que se quiere que se emita la luz.
  Otra manera más cómoda y visual de indicar el radio de influencia de la luz es con el objeto de la luz seleccionado pinchar en el icono del menú superior ó . Mediante los elementos en cada uno de los ejes que aparecen se puede modificar el radio hasta obtener el deseado.

- Posición de la luz: La posición del objeto de la luz puede ser modificada dentro del nivel creado.
  La forma más cómoda de realizar el movimiento es de la forma visual trasladándola en los 3 ejes de coordenadas. Para ello hay que presionar en el icono con el objeto seleccionado. Realizando esto aparecerán 3 flechas en cada uno de los ejes de coordenadas partiendo del objeto desde las cuales se podrá mover el objeto hasta ponerlo en el lugar deseado.
  Otra forma de indicar la posición es dando el valor concreto en cada uno de los ejes de coordenadas, X, Y y Z. En la etiqueta *Movement→ Location* se indicarán estos valores de los ejes.

Punto de luz.

Este tipo de luz representa el típico punto de luz real, con iluminación circular en todos los sentidos y diferente intensidad según la distancia al punto de emisión.

En el editor, el punto de luz se muestra con la siguiente imagen:

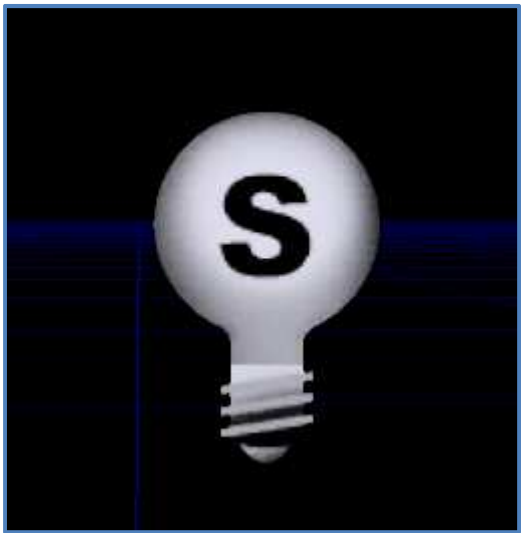


Imagen 155:Point Light.

Y el punto de luz "toggleable" mediante:

Imagen 156:Point Light "Toggleable".

En este caso la intensidad de la luz es por igual en todos los ejes, no existiendo diferencia de intensidad de luz entre un eje u otro. Dicha intensidad es mayor o menor según se esté más o menos próximo al punto emisor.

Para poder acoplar un punto de luz al mapa se puede realizar de la siguiente manera:

- Pulsar con el botón secundario del ratón sobre cualquier zona despejada de cualquier vista del editor.
- En el menú desplegado seleccionar "*Add Actor*".

- En el nuevo submenú desplegado seleccionar *"Add Light (point)"* (imagen 157):

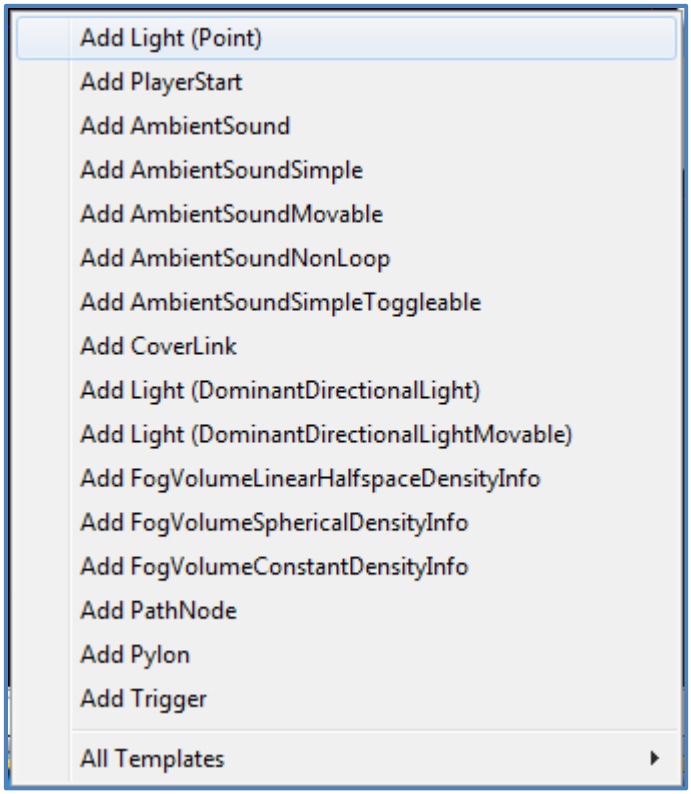


Imagen 157:Añadir punto de luz.

En el caso del punto de luz *"toggleable"* hay que realizar los siguientes pasos:

- Abrir la herramienta Content Browser mediante el icono de la barra de herramientas superior.
- Seleccionar la etiqueta *"Actor Classes"* de las etiquetas superiores.
- Navegar en el árbol de elementos: *Light → PointLight → PointLightToggleable y seleccionarlo.*
- Cerrar la pantalla de *"Actor Classes"* con el elemento seleccionado.
- Pinchar con el botón secundario del ratón sobre cualquier zona despejada de cualquier ventana del editor y seleccionar *"Add PointLightToggleable Here"*:

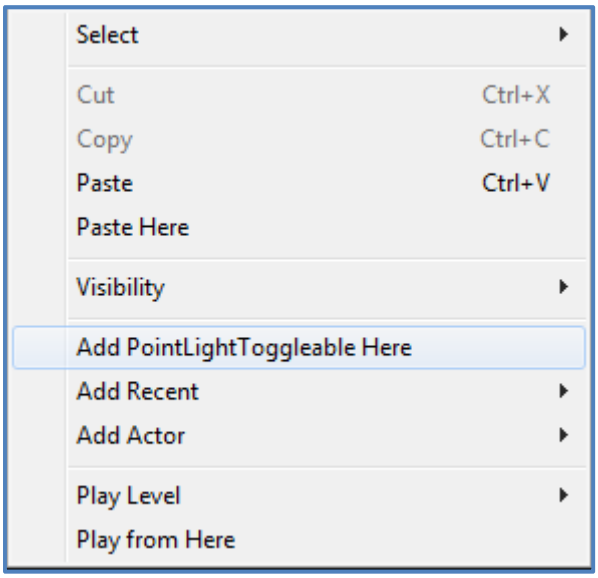


Imagen 158:Añadir punto de luz "Toggleable".

Foco.

Este tipo de luz representa a focos de luz donde la luz sólo ilumina una cierta zona en una sola dirección.

En el editor este tipo de luz es representado mediante el icono:

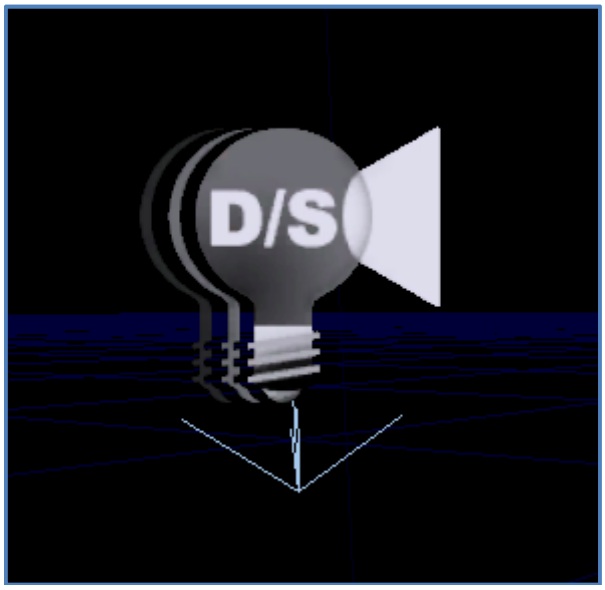


Imagen 159:Foco.

Y el mismo tipo de luz pero de tipo "toggleable" se representa:

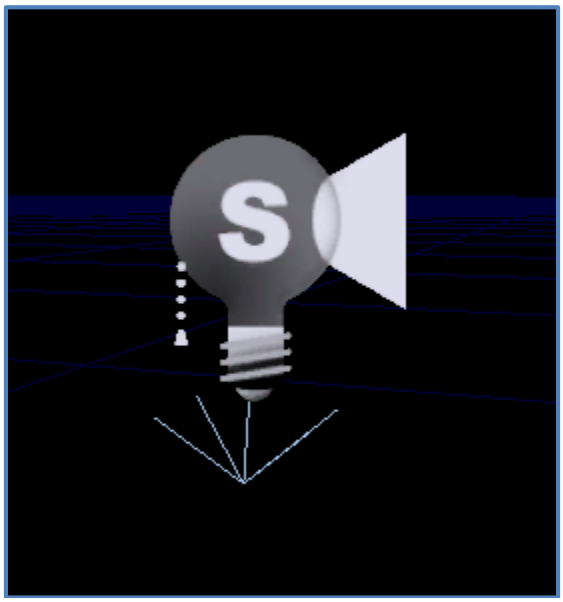


Imagen 160:Foco "Toggleable".

Las características de este tipo de luz son diferentes a un punto de luz normal. En este caso hay que indicar ciertos aspectos propios:

- Dirección de la luz: Como este tipo de luz sólo ilumina en una dirección, se debe indicar esta dirección.

  Hay dos formas de indicar la dirección:

  - En los iconos de la luz se puede observar una flecha, la cual indica la dirección hacia dónde se proyecta la luz. Dicha dirección puede ser modificada rotándola mediante el icono, Rotation Mode, de la barra de herramientas superior en cada uno de los ejes de coordenadas.
    Mediante esta forma la flecha irá girando en el eje especificado pudiendo llegar a tener la dirección deseada.

  - Indicando el ángulo concreto que se quiere tener en cada uno de los ejes. Para ello hay que abrir las características del objeto teniéndolo seleccionado y presionando F4.
    Ir a la etiqueta *Light → Light Component → SpotLightComponent → Rotation* e indicar en cada uno de los ejes el ángulo deseado.

- Radio del foco: Otra de las características que se pueden dar a este tipo de luz es el radio que tendrá el foco.
  Para indicar el radio que se desea hay que ir a la etiqueta *Light → Light Component → SpotLightComponent → Outer Cone Angle e insertar el ángulo.*

- Longitud del foco: Otra característica de este tipo de luz es la longitud a la que se quiere que llegue la intensidad de la luz.

  Para modificar dicha longitud hay que presionar el icono ó de la barra de herramientas superior. Mediante los elementos que se muestran se puede modificar el tamaño del cono representado en el editor que representa la longitud.

Para acoplar un foco al mapa hay que seguir los siguientes pasos:

- Abrir la herramienta Content Browser mediante el icono de la barra de herramientas superior.
- Seleccionar la etiqueta *"Actor Classes"* de las etiquetas superiores.
- Navegar en el árbol de elementos: *Light → SpotLight → SpotLightMovable y seleccionarlo.*
- Cerrar la pantalla de *"Actor Classes"* con el elemento seleccionado.
- Pinchar con el botón secundario del ratón sobre cualquier zona despejada de cualquier ventana del editor y seleccionar *"Add DominantSpotLight Here"*:

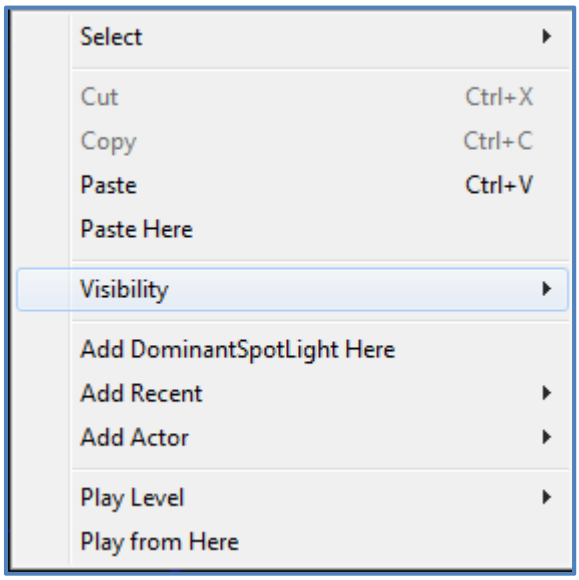


Imagen 161:Añadir foco.

En el caso de querer acoplar un tipo de luz foco pero poder modificarlo dinámicamente en ejecución (con característica *"toggleable"*) se debe realizar:

- Abrir la herramienta Content Browser mediante el icono de la barra de herramientas superior.
- Seleccionar la etiqueta *"Actor Classes"* de las etiquetas superiores.
- Navegar en el árbol de elementos: *Light → SpotLight → SpotLightToggleable y seleccionarlo.*
- Cerrar la pantalla de *"Actor Classes"* con el elemento seleccionado.
- Pinchar con el botón secundario del ratón sobre cualquier zona despejada de cualquier ventana del editor y seleccionar *"Add SpotLightToggleable Here"*:

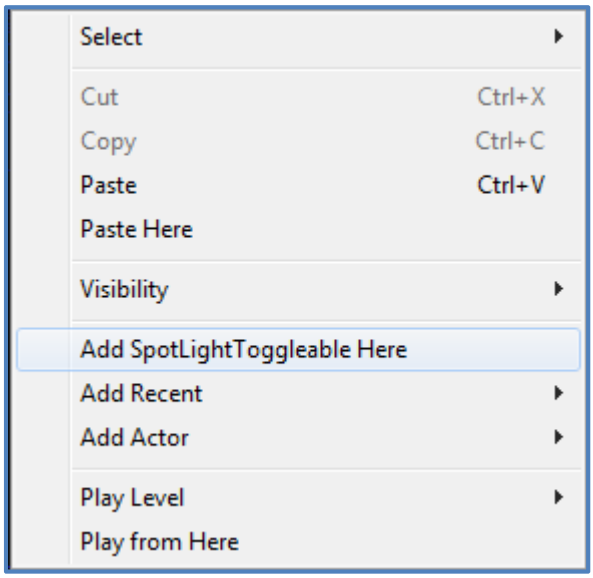


Imagen 162:Añadir foco "Toggleable".

Luz ambiental.

Este tipo de luz tiene un radio de acción muy amplio con lo que es muy utilizado en los niveles exteriores, representando la iluminación del sol.

En el editor este tipo de luz es mostrado de la siguiente manera:

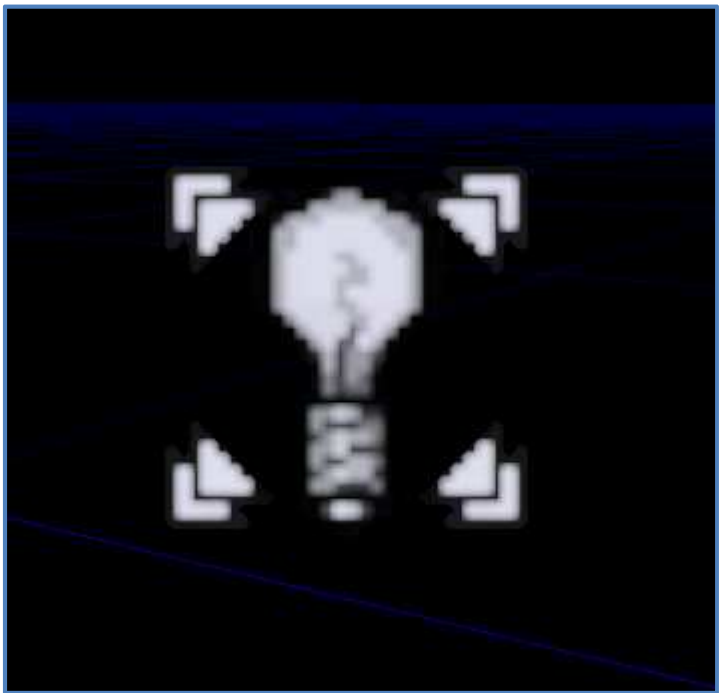


Imagen 163:Luz ambiental.

Para poder añadir este tipo de luz se deben seguir los siguientes pasos:

- Abrir la herramienta Content Browser mediante el icono de la barra de herramientas superior.
- Seleccionar la etiqueta *"Actor Classes"* de las etiquetas superiores.

- Navegar en el árbol de elementos: *Light → SkytLight → SkyLightToggleable y seleccionarlo.*
- Cerrar la pantalla de *"Actor Classes"* con el elemento seleccionado.
- Pinchar con el botón secundario del ratón sobre cualquier zona despejada de cualquier ventana del editor y seleccionar *"Add SkyLightToggleable Here"*:

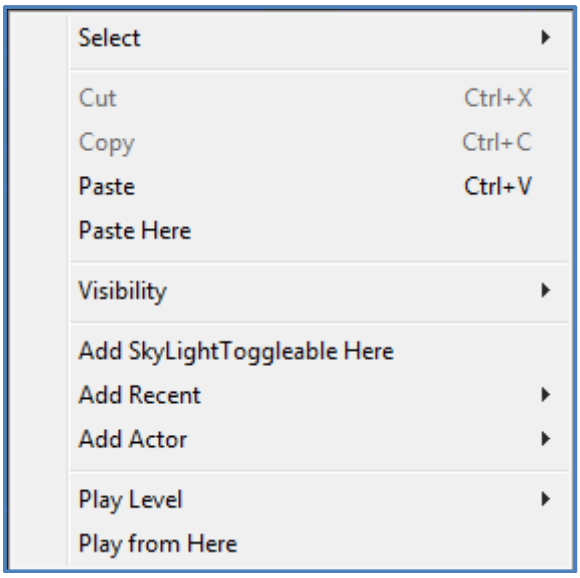


Imagen 164:Añadir luz ambiental.

Este tipo de luz sólo puede ser acoplado una sola vez. En el caso que ya se encuentre un elemento acoplado, no se mostrará la opción de volver acoplarlo aunque se seleccione en el *"Actor Classes"*.

7.3. Acoplar objetos al nivel desde Content Browser.

Para acoplar objetos al nivel es necesario realizarlo a través del Content Browser, herramienta del editor comentada en el capítulo 4. [36]

Los objetos estáticos acoplables, como son columnas, puertas, etc., son llamados en el editor *"Static Mesh"*. Estos objetos ya tienen su propia textura y material referenciados con lo que simplemente con acoplarlos es suficiente para tener el objeto completo.

Para acoplar el objeto hay que seguir los siguientes pasos:

- Abrir la herramienta del Content Browser pulsando el icono de la barra de herramientas superior.
- Buscar el *"Static Mesh"* que se quiere acoplar y seleccionarlo. Para buscar se puede utilizar el filtro por tipos de la parte superior mostrando sólo los elementos que se desean ó si se conoce el nombre del objeto se puede introducir en el campo superior para realizar su búsqueda.
- Con el objeto seleccionado hay dos maneras de poder acoplarlo:

- Arrastrarlo a la vista.- El *"static mesh"* puede ser arrastrado desde el Content Browser al punto de cualquier vista del editor para tenerlo en el nivel.
  Simplemente hay que pulsar con el botón principal del ratón sobre el objeto y sin dejar de pulsar mover el ratón hasta el punto de cualquier vista que se quiere tener. Cuando se deje de pulsar el ratón sobre la vista, el objeto aparecerá en el editor.
- En el editor, en cualquiera de las vistas, se pulsa en cualquier espacio libre con el botón derecho del ratón. En el menú que se despliega elegir la etiqueta *"Add Actor"* desplegándose un nuevo submenú. En este nuevo submenú seleccionar la etiqueta *"Add Static Mesh: Static Mesh <nombre_objeto_seleccionado>"* y el objeto ya se tendrá en el editor.

Una vez que el objeto ya se tiene en el editor, éste puede ser modificado tanto en su posición, tamaño y ángulo.

- Modificar posición.-
  Al estar realizando un modelado 3D, la mejor forma de modificar la posición de un objeto es de forma visual.
  En la barra de herramientas superior hay que tener activado el modo *"Traslation Mode"* pulsando en el icono .
  Con el modo activado y seleccionando el objeto a modificar (pulsando con el botón izquierdo del ratón sobre él) aparecerán unas flechas de distinto color en cada uno de los ejes de coordenadas.

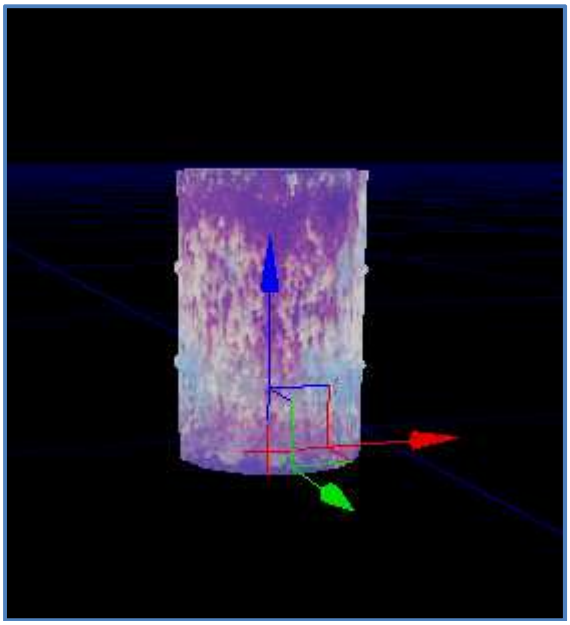


Imagen 165:Traslation Mode.

Mediante estas flechas se podrá modificar la posición del objeto en cada eje. Cuando se sitúa el ratón sobre alguna flecha, ésta cambia de color y aparece el símbolo de movimiento.

Para mover el objeto hay que mantener presionado el botón izquierdo del ratón sobre la flecha del eje en que se quiere mover y desplazar el ratón en el movimiento que se quiere realizar.

De la manera comentada, el movimiento se realiza individualmente en cada uno de los ejes, pero hay otra forma de poder mover el objeto, que es realizarlo en dos ejes a la vez. Para ello se tienen enlaces entre cada par de flechas que situando el ratón sobre ellos y pulsando con el botón izquierdo del ratón se puede realizar el movimiento del objeto en los dos ejes. Estos enlaces son mostrados en la imagen 166.

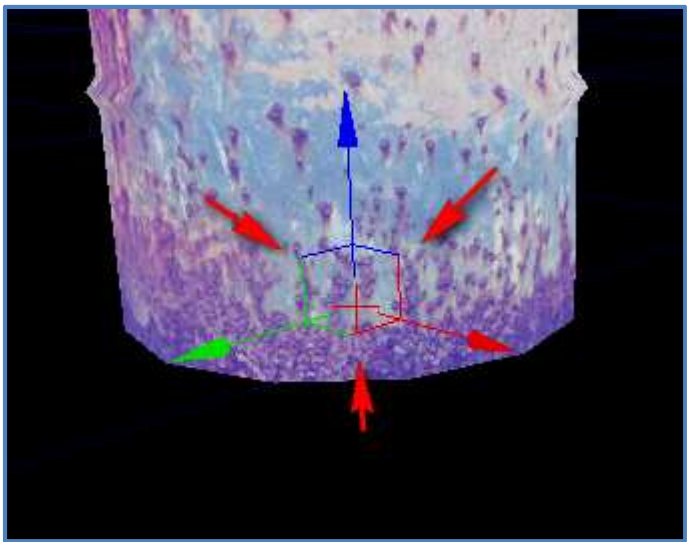


Imagen 166:Traslation Mode en 2 ejes.

Otra opción de transladar el objeto es indicando el valor de sus coordenadas ó el valor de su traslado en cada uno de los ejes.

Pinchando con el botón derecho del ratón sobre el icono, Traslation Mode, se mostrará una nueva ventana, imagen 167, donde se podrá indicar el valor en cada uno de los ejes.

Imagen 167:Traslation Mode insertando valores.

En esta nueva ventana se podrá indicar si los valores dados en los ejes son de manera absoluta o relativa mediante el chekpoint *"Relative"*; y para realizar los cambios sólo hay que presionar el botón *"Apply"*.

- Modificar tamaño.-
  La modificación del tamaño de un objeto se puede realizar de manera uniforme en todos los ejes de coordenadas ó modificar el tamaño en cada uno de los ejes independientemente.

- Uniformemente.- En la barra de herramientas superior hay que tener activado el modo *"Scaling Mode"* pulsando el icono .
  Con el modo activado y seleccionando el objeto a modificar (pinchando con el botón izquierdo del ratón sobre él) aparecerán los elementos en cada eje que permitirán modificar el tamaño, mostrados en la imagen 168.

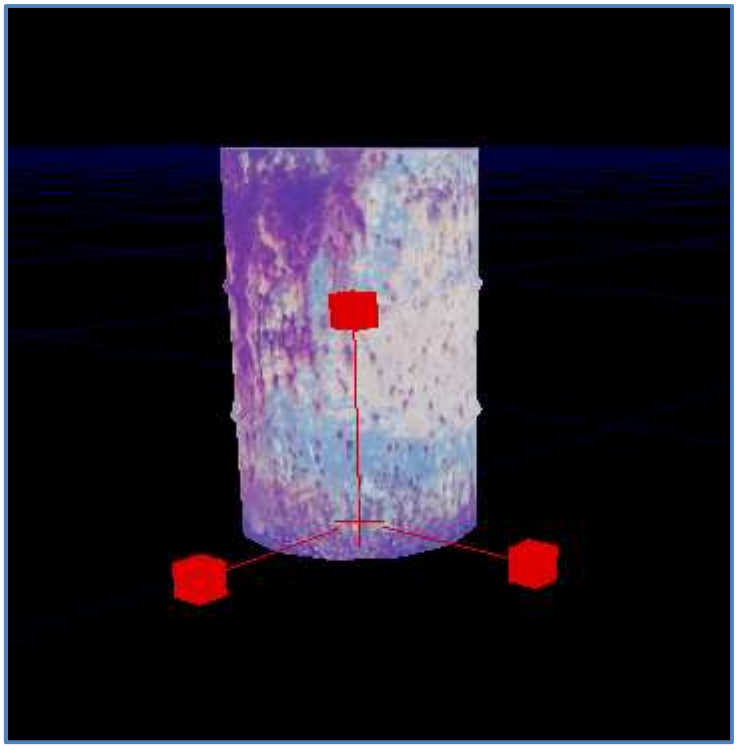


Imagen 168:Scaling Mode uniforme.

Para modificar el tamaño hay que situar el ratón sobre cualquiera de estos elementos y manteniendo pulsado el botón izquierdo, desplazar el ratón hasta tener el tamaño deseado.

En este caso el elemento utilizado para modificar el tamaño no es relevante, ya que como la modificación es uniforme se puede elegir cualquiera.

- No uniformemente.- En la barra de herramientas superior hay que tener activado el modo *"Non-Uniform Scaling Mode"* pulsando el icono .

  Con el modo activado y seleccionando el objeto a modificar (pinchando con el botón izquierdo del ratón sobre él) aparecerán los elementos en cada eje que permitirán modificar el tamaño del objeto independientemente en cada eje, mostrados en la imagen 169.

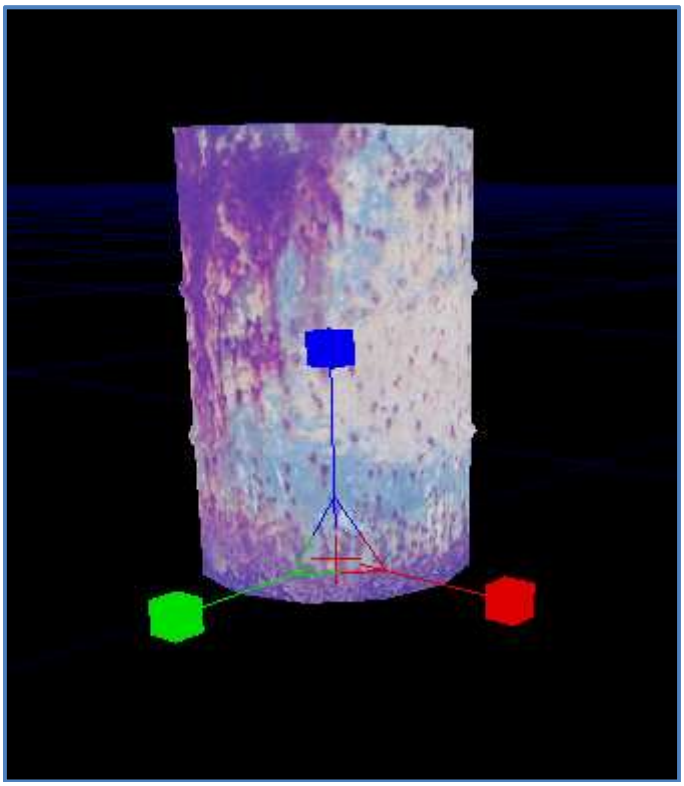


Imagen 169:Scaling Mode no uniforme.

Cada uno de los elementos que aparecen permite modificar el tamaño del objeto en cada uno de los ejes. Cuando se sitúa el ratón sobre algún elemento, éste cambia de color y aparece el símbolo de movimiento.

Para modificar el tamaño del objeto hay que mantener presionado el botón izquierdo del ratón sobre el elemento del eje y desplazar el ratón para ir modificando el tamaño hasta obtener el deseado.

De la manera comentada, la modificación del tamaño se hace solamente en cada uno de los ejes, pero se tiene otra opción de modificar el tamaño en dos ejes de coordenadas a la vez. Para ello se tienen enlaces entre cada par de ejes de coordenadas que permiten realizar la modificación en los dos ejes a la vez, mostrados estos enlaces en la imagen 170.

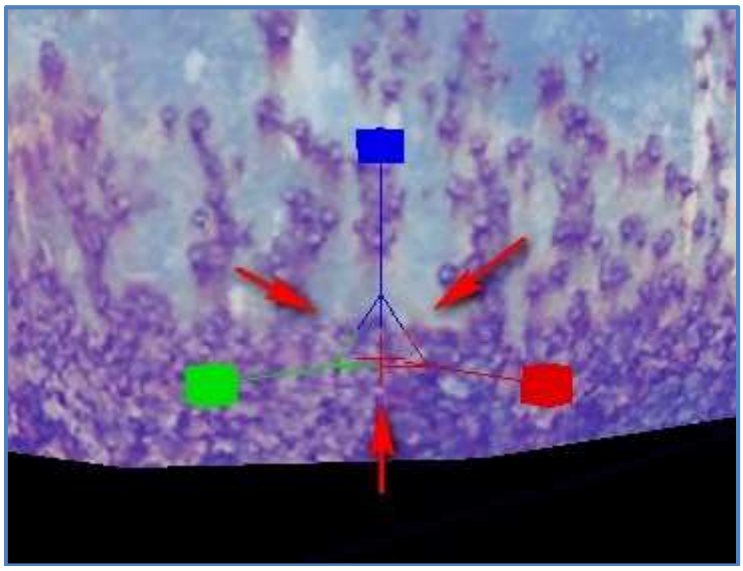


Imagen 170:Scaling Mode no uniforme en 2 ejes a la vez.

El editor permite realizar el escalado de los objetos de otra forma diferente, indicando la escala que se quiere dar en cada uno de los ejes de coordenadas.

Pulsando con el botón derecho del ratón sobre cualquiera de los iconos ,*"Non-Uniform Scaling Mode"*, ó , *"Scaling Mode"*, se mostrará una nueva ventana, imagen 171, donde se podrá indicar la escala en cada uno de los ejes.

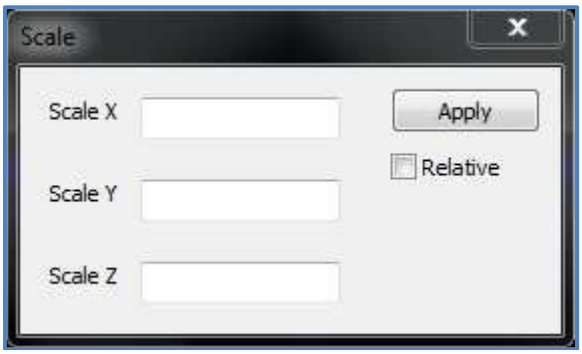


Imagen 171:Scaling Mode insertando valores.

Mediante el chekpoint que se muestra se puede indicar si la escala dada en cada eje se realiza de forma absoluta o relativa; y para realizar los cambios simplemente hay que pulsar sobre el botón *"Apply"*.

- Girar objeto.-
  Otra opción que se puede realizar en los objetos es poder girarlos en cada uno de los ejes de coordenadas.

  En la barra de herramientas superior hay que tener activado el modo *"Rotation Mode"* pulsando el icono . Con el modo activado y seleccionando el objeto a modificar (pulsando con el botón izquierdo del ratón sobre él) aparecerán unas circunferencias de distinto color en cada uno de los ejes de coordenadas.

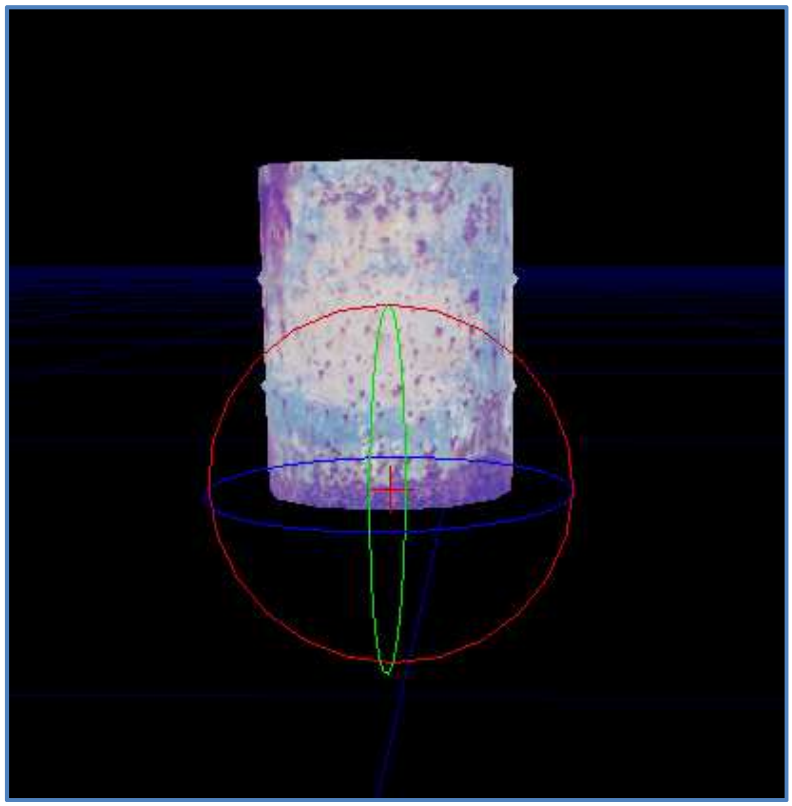


Imagen 172:Rotation Mode.

Cada una de las circunferencias que aparecen permite girar el objeto en cada uno de los ejes. Cuando se sitúa el ratón sobre alguna circunferencia, ésta cambia de color y aparece el símbolo de movimiento.

Manteniendo presionado el botón izquierdo del ratón y realizando un movimiento del ratón, se puede ir girando el objeto sobre el eje de coordenadas elegido hasta obtener el ángulo deseado.

En esta modificación hay que ir de eje de coordenadas en eje de coordenadas para encontrar la posición que se quiere del objeto.

Otra opción que se puede utilizar es indicando el valor del ángulo que se quiere dar al objeto en cada uno de los ejes de coordenadas. Pinchando con el botón derecho del ratón sobre el icono, *"Rotation Mode"*, se mostrará una nueva ventana, imagen 173, en el editor donde se podrá indicar el valor del ángulo en cada uno de los ejes.

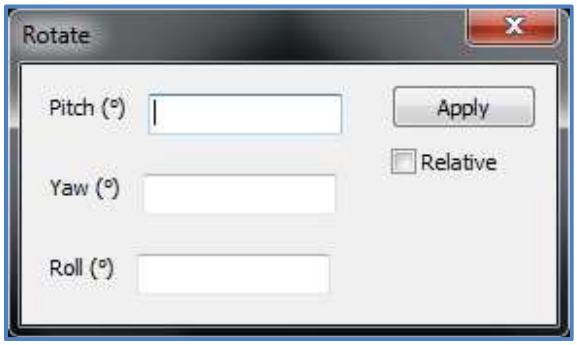


Imagen 173:Rotation Mode insertando valores.

Con el chekpoint se indica si los ángulos dados son de forma relativa o absoluta y para realizar los cambios simplemente hay que presionar en el botón *"Apply"*.

7.4. Sonido y mensajes de información.

A la hora del desarrollo de un juego, muchas veces es necesario mostrar mensajes informativos para guiar al jugador, tanto para indicar lo que tiene que hacer ó mostrar información de cómo continuar el juego. [37]

La forma más común de mostrar estos mensajes informativos al jugador es mediante texto mostrado por pantalla. En el motor Unreal, y más concretamente en la herramienta Unreal Kismet, se tiene un elemento que permite mostrar texto por pantalla. Este elemento es llamado *"Play Announcement"* mostrado de la siguiente manera:

Imagen 174:Elemento "Play Announcement".

En las propiedades de este elemento se tiene la etiqueta *"Announcement Text"* en la cual se podrá insertar el texto que se querrá mostrar por pantalla al jugador.

Asociando este elemento en el flujo de cualquier acción con los demás elementos se podrá mostrar el texto en el momento que se desee.

Este elemento, a la vez que se muestra un texto, permite reproducir un sonido. Este sonido se puede indicar en la etiqueta llamada *"Announcement Sound"* seleccionándolo previamente en el Content Browser.

También este elemento puede ser utilizado para reproducir solamente un sonido, sin mostrar texto. De esta manera se pueden reproducir sonidos en el nivel para hacerlo más realista, como reproducir sonido en la apertura de una puerta, sonido de truenos en una tormenta, etc.

Por ejemplo, en la imagen 175 se muestra una secuencia de truenos.

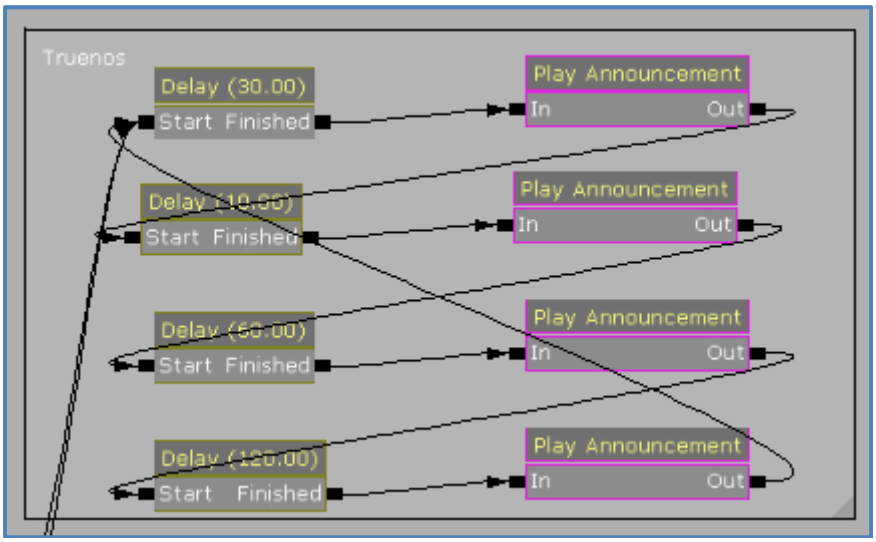


Imagen 175:Ejemplo secuencia de sonidos.

Otra forma para acoplar sonido a un nivel es utilizando los elementos llamados *"AmbientSound"*. Estos sonidos son reproducidos de forma uniforme en todas direcciones a partir de un punto dado. Según se encuentre el jugador más cerca o más lejos del punto de reproducción, el sonido será más o menos intenso. Esta particularidad no se podía dar en el elemento anterior de proximidad al punto de reproducción, ya que el sonido se reproduce siempre igual.

Se puede tener acceso a los elementos *"AmbientSound"* a través de la herramienta *"Actor Classes"*. Para llegar a ella es necesario abrir Content Browser mediante el icono de la barra de herramientas superior y seleccionando la etiqueta con el mismo nombre.

Los elementos se encuentran dentro de la etiqueta *"Keypoint"*:

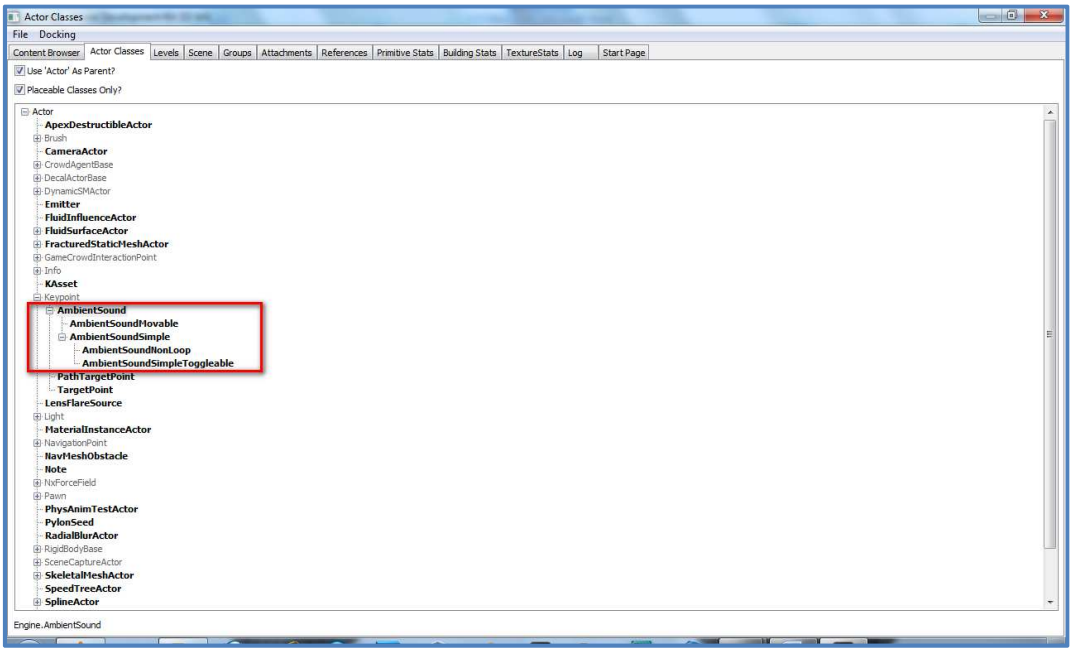


Imagen 176:Sección "AmbientSound" en herramienta "Actor Classes".

Como se puede observar, se tienen diversos elementos dentro de la etiqueta *"AmbientSound"*:

- AmbientSoundMovable: El sonido de este tipo de elemento cubre todo el nivel completo.

Imagen 177: "Ambient Sound Movable".

El sonido a reproducir hay que insertarlo en la etiqueta *"Sound Cue"* (*Audio → Audio Component → Sound Cue*) de las propiedades del objeto (pulsando 'F4' con el objeto seleccionado). El sonido que se tiene que añadir debe ser de extensión *".cue"*, que es una extensión del editor.

El volumen del sonido reproducido por este elemento también puede ser modificado. Para ello hay que modificar la etiqueta con nombre *"Volume Multiplier"* (*Audio → Audio Component → Volume Multiplier*) indicando el valor por el cual se quiere multiplicar el sonido.

- AmbientSoundNonLoop: Este elemento permite indicar el radio del espacio donde se puede escuchar el sonido.

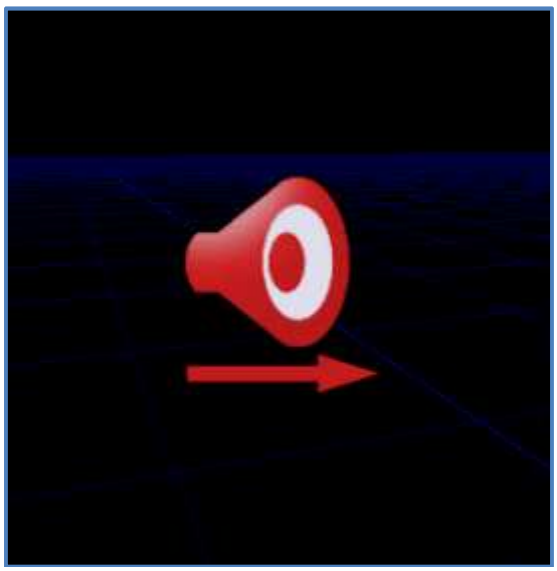


Imagen 178: "Ambient Sound Non Loop".

Los sonidos a reproducir se deben insertar en la etiqueta *"Sound Slots"* (*AmbientSoundSimple → Ambient Properties → Sounds → Sound Slots*) añadiendo una posición por cada sonido a reproducir.

Se permite indicar el radio máximo en que se puede escuchar el sonido y el radio mínimo, realizándose una atenuación del sonido entre estos dos radios. El radio máximo se indica en la etiqueta de nombre *"Radius Max"* (*AmbientSoundSimple → Ambient Properties → Attenuation → Radius Max*) y el radio mínimo se indica en la etiqueta de nombre *"Radius Min"* (*AmbientSoundSimple → Ambient Properties → Attenuation → Radius Min*). El tipo de atenuación que se quiere dar al sonido se puede indicar en la etiqueta *"Distance Model"* (*AmbientSoundSimple → Ambient Properties → Attenuation → Distance Model*) desplegando el menú.

- AmbientSoundSimpleToggleable: Este elemento es similar al anterior pero con la propiedad *"Toggleable"* que permite activar o desactivar de forma dinámica y en ejecución del nivel.

Imagen 179:"Ambient Sound Simple Toggleable".

Los sonidos a reproducir se deben insertar en la etiqueta *"Sound Slots"* (*AmbientSoundSimple → Ambient Properties → Sounds → Sound Slots*) añadiendo una posición por cada sonido a reproducir.

Se permite indicar el radio máximo en que se puede escuchar el sonido y el radio mínimo, realizándose una atenuación del sonido entre estos dos radios. El radio máximo se indica en la etiqueta de nombre *"Radius Max"* (*AmbientSoundSimple → Ambient Properties → Attenuation → Radius Max*) y el radio mínimo se indica en la etiqueta de nombre *"Radius Min"* (*AmbientSoundSimple → Ambient Properties → Attenuation → Radius Min*). El tipo de atenuación que se quiere dar al sonido se puede indicar en la etiqueta *"Distance Model"* (*AmbientSoundSimple → Ambient Properties → Attenuation → Distance Model*) desplegando el menú.

Mediante todos estos elementos se pueden acoplar distintos tipos de sonido en el nivel haciéndolo más realista y atractivo a la hora de jugar con él. Se puede reproducir música en el nivel que irá cambiando según la acción se desarrolle y todos aquellos sonidos necesarios para el mundo creado.

7.5. Modificar inventario de armas del personaje principal.

Una opción importante en un juego es el ir modificando el inventario de armas del jugador principal, añadiendo nuevas armas ó quitando las que se tenían. [38]

Se va a comentar cada una de las opciones que se tienen para modificar el inventario:

Lanzar el nivel con el inventario de armas vacío.

Al lanzar el nivel implementado se puede querer tener el inventario de armas vacío, sin que el personaje principal tenga ninguna arma. El motor Unreal permite realizar esta circunstancia siguiendo los siguientes pasos:

- Pinchar en la etiqueta *"View"* del menú de la aplicación y en el submenú mostrado, imagen 180, seleccionar *"World Properties"*:

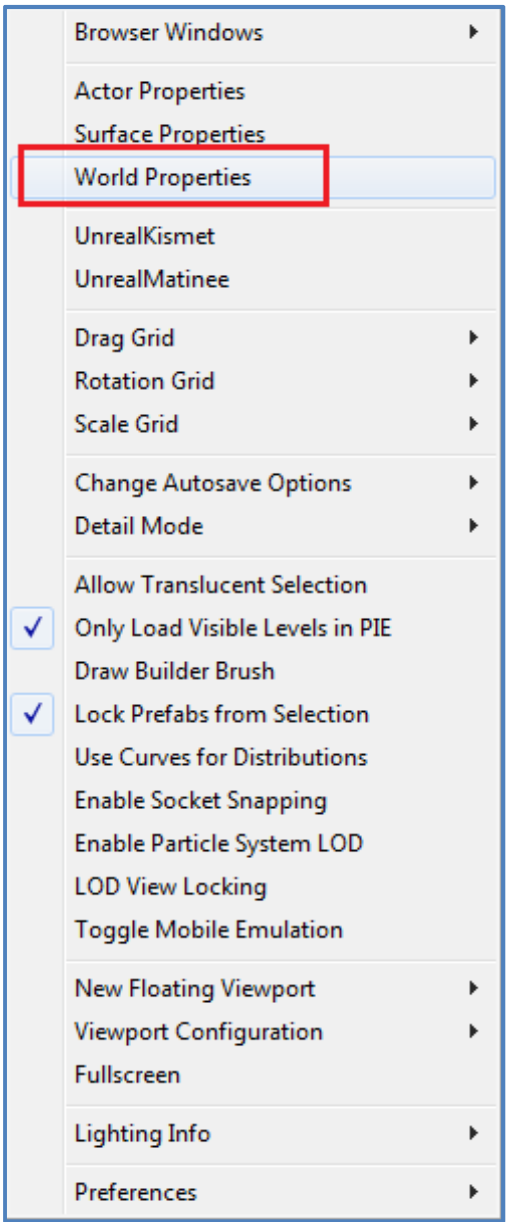


Imagen 180:Menú "World Properties".

- Se abrirá una nueva ventana donde se muestran todas las características del mundo/nivel creado:

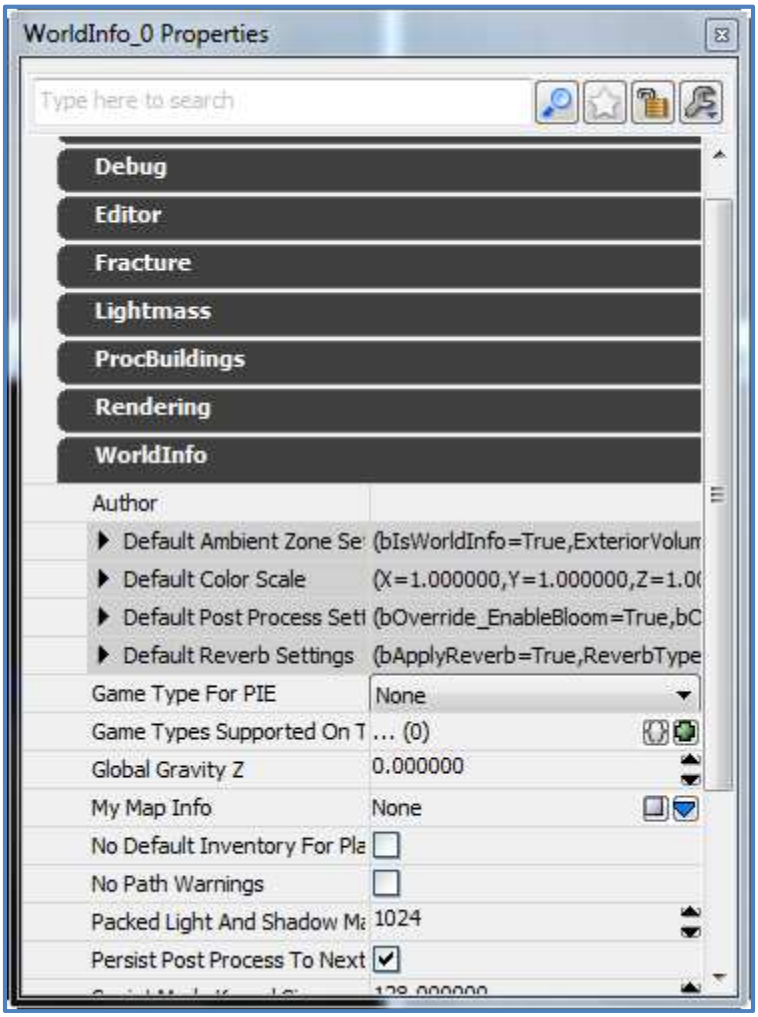


Imagen 181:Propiedades del mundo.

- Dentro de la sección *"WorldInfo"* se tiene una etiqueta que permite no dotar al personaje principal de armas. Esta etiqueta se llama *"No Default Inventory For Player"* la cual activándola realizará la acción buscada.

Otra opción de poder vaciar el inventario de armas al personaje principal es modificando el código fuente de las clases que constituyen todo el motor UDK.

En una de las clases implementadas es donde se indica el inventario que se quiere dotar al personaje al ser lanzado el nivel. Esta clase se encuentra en la siguiente jerarquía de directorios:

UDK / Development / Src / UTGame / Classes / UTGame.uc

En el código de esta clase, en la sección *"default properties {…}"* se tiene la variable donde se indica el arma del inventario. Esta variable el la llamada *"DefaultInventory (0)="* siendo un array de elementos, donde se puede indicar las clases de armas a utilizar. En la imagen 182 se muestra la posición de esta variable:

```
Acronym="???"

DefaultInventory(0)=class'UTWeap_LinkGun'

VictoryMessageClass=class'UTGame.UTVictoryMessage'
```

Imagen 182:Código inventario.

Si se borrase la clase del arma, en este caso *"class'UTWeap_LinkGun'"*, el inventario de armas estará vacío.

Mediante la modificación de este código, también se puede cambiar el tipo de arma que se quiere tener por defecto en el inventario del personaje principal. Simplemente hay que cambiar la clase la cual implementa al arma que se desea utilizar y ponerla en la variable indicada anteriormente.

En el motor UDK se tienen implementadas varias armas que pueden ser utilizadas por el personaje principal. El nombre de los archivos que contienen la implementación de las armas sigue la nomenclatura de comenzar con *"UTWeap_"* posibilitando así su localización.

Añadir armas al inventario durante el juego.

En el nivel creado pueden acoplarse ciertos elementos que representan factorías de armas. En estos objetos se puede recoger el tipo de arma que exponen pasando a formar parte del inventario del personaje.

Imagen 183:Ejemplo factoría de arma en nivel.

En la imagen 183 se muestra un ejemplo de una factoría de armas acoplado en el nivel.

Para poder acoplar estos elementos al nivel hay que seguir los siguientes pasos:

- Estos elementos son llamados *"UTWeaponPickupFactory"*.
  Se pueden encontrar en la herramienta *"Actor Classes"* la cual puede abrirse mediante el icono de la barra de herramientas superior del editor y seleccionando la pestaña con el mismo nombre.
  La jerarquía de directorios, imagen 184, que hay que seguir para encontrarlos es:

  NavigationPoint → PickupFactory → UDKPickupFactory → UTPickupFactory → UTWeaponPickupFactory.

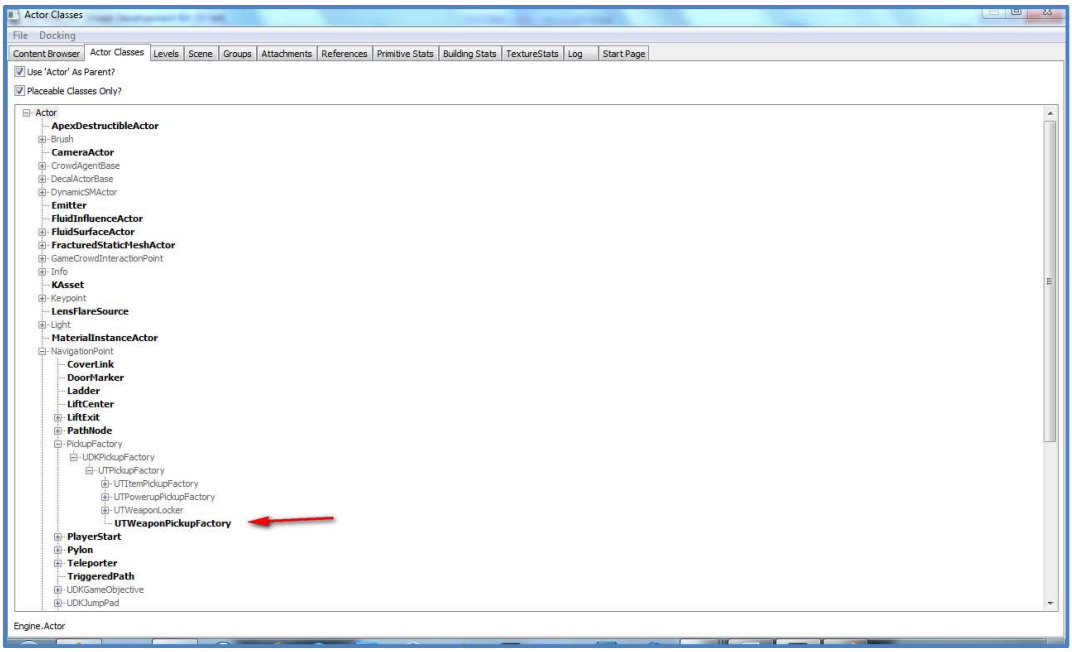


Imagen 184:Sección "UTWeaponPickupFactory" en herramienta "Actor Classes".

- Seleccionar el elemento a acoplar y volviendo al editor, en cualquier vista, se pulsa el botón derecho del ratón seleccionando *"Add UTWeaponPickupFactory Here"* en el submenú mostrado.
  Mediante estas acciones se tiene el objeto acoplado pudiéndolo trasladar activando el icono , *"Traslation Mode"*, para situarlo en la posición deseada.

- En el objeto factoría hay que indicar el tipo de arma que se quiere que exponga. Para ello hay que abrir las características del objeto, seleccionándolo y pulsando la tecla 'F4' del teclado.
  Se abrirá una nueva ventana mostrando todas las características del objeto. En la sección *"UTWeaponPickupFactory"*, en la etiqueta *"Weapon Pickup class"* hay que seleccionar el tipo de arma.

Imagen 185:Propiedades "Weapon Pickup Factory".

En dicha etiqueta se tiene un desplegable donde se mostrarán todas las armas que se pueden utilizar. En el motor vienen implementadas tres armas diferentes, mostradas en la imagen 186. El arma que se seleccione será mostrada por el objeto para que el jugador principal pueda disponer de ella.

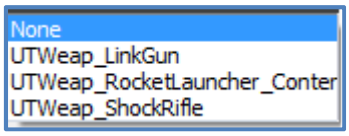


Imagen 186:Tipos de armas.

Existe otra forma, menos visual, de poder añadir armas al inventario del jugador. Esta forma se realiza mediante un elemento de la herramienta Unreal Kismet el cual agrega el arma indicada al inventario.

Este elemento es llamado *"Give Inventory"* y la manera de utilizarlo es:

- Este elemento pertenece a la herramienta Unreal Kismet con lo que habrá que abrirla mediante el icono de la barra de herramientas superior del editor.
- En cualquier zona libre de la herramienta realizar:

  Botón derecho del ratón → New Action →Pawn → Give Inventory

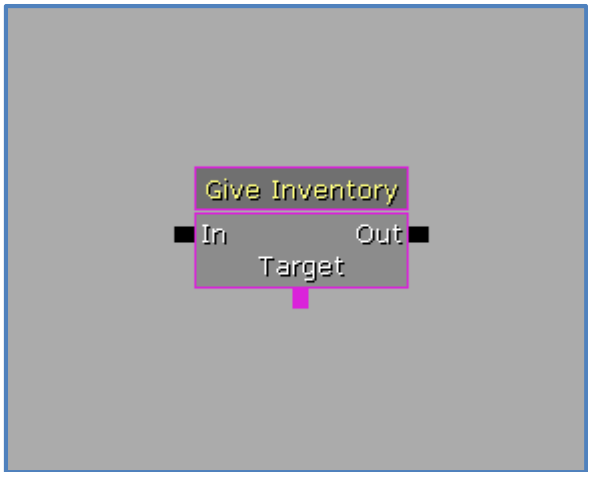


Imagen 187:"Give Inventory".

- En las propiedades de este elemento, imagen 188, hay que indicar el/los objetos que se quieren añadir al inventario.
  En la etiqueta *"Inventory List"* se muestra la lista de objetos a poder añadir. Para crear una nueva posición a la lista hay que pulsar sobre el icono de esta etiqueta. La nueva posición será mostrada junto con un desplegable con todos los objetos que se pueden añadir al inventario.

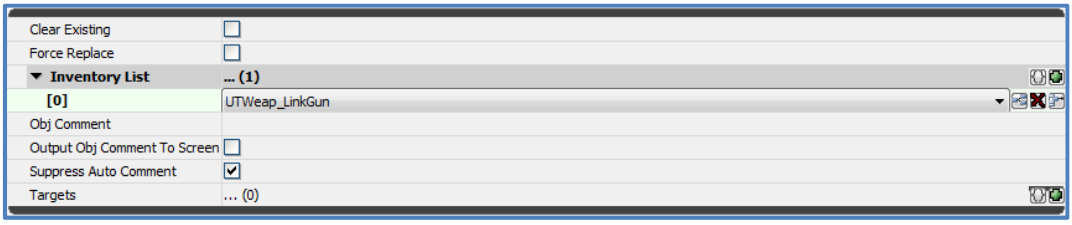


Imagen 188:Propiedades "Give Inventory".

- Al elemento hay que indicar el personaje al cual añadir los objetos a su inventario.
  En este caso hay que indicarle que sea al jugador principal, con lo que es necesaria una variable que representa a este jugador.
  Para obtener dicha variable realizar:

  Botón derecho del ratón → New Variable → Player → Player.

  Y en esta variable hay que desactivar la etiqueta *"All players"* de la propiedades.

  Se debe unir la variable con la entrada *"Target"* ofrecida por el elemento *"Give Inventory"* tal y como se muestra en la imagen 189.

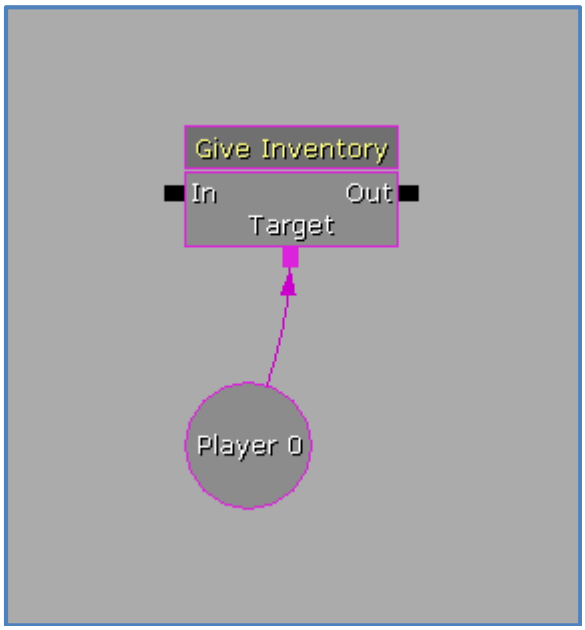


Imagen 189:Asociación del "Target" de "Give Inventory".

- Este elemento puede ser asociado a cualquier otro elemento ó evento ya creados. Hay que tener en cuenta que con el evento *"Load Level"* no funciona, teniéndose que utilizar el evento *"Player Spawed"*.

Se han mostrado todas las opciones de poder modificar el inventario de armas del personaje principal, de gran utilidad a la hora de modificar las condiciones de los personajes de un nivel.

7.6. Modificar el tipo de *"colisión"* de los objetos en el nivel.

En todos los objetos acoplados a un nivel se puede modificar el tipo de colisión que tienen respecto al jugador principal. Se puede realizar la opción que el objeto sea como una ilusión la cual se puede atravesar, que sea un objeto sólido, etc. [39]

El tipo de colisión del objeto se puede modificar en sus propiedades. Las propiedades de un objeto se muestran seleccionado el objeto y pulsando la tecla 'F4' del teclado. Con esto se mostrará una nueva ventana con todas las propiedades del objeto, mostrada en la imagen 190.

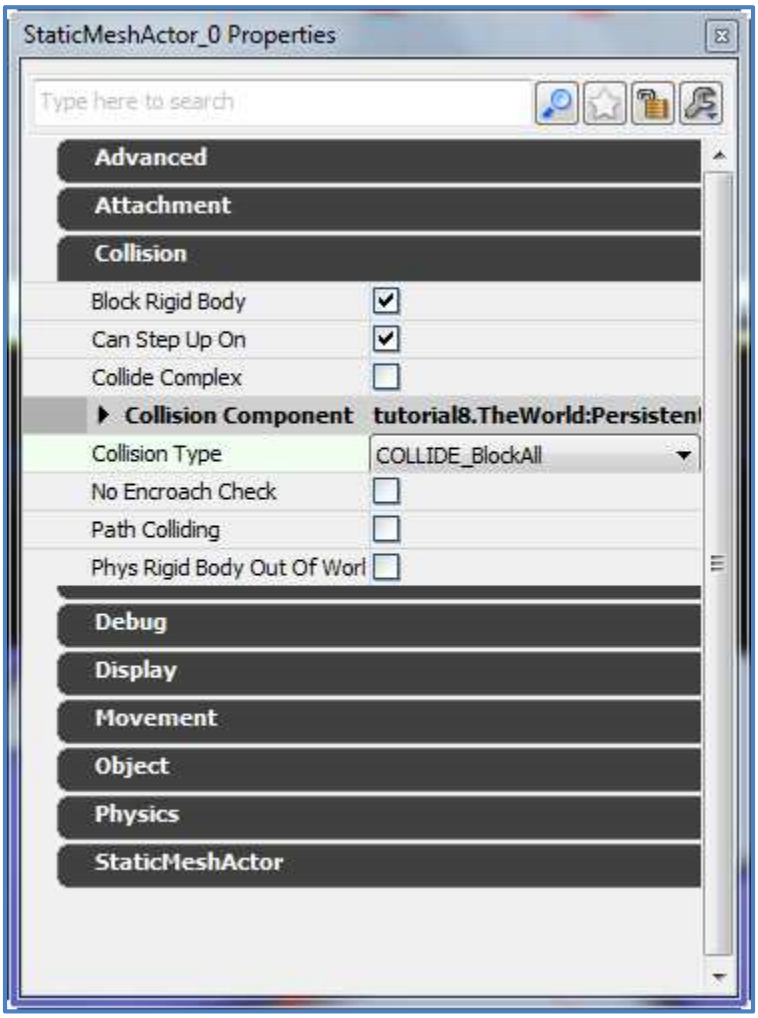


Imagen 190:Propiedades "Static Mesh".

En la sección con nombre *"Collision"* se pueden ver todas las etiquetas referentes a la colisión del personaje principal con este objeto. El tipo de colisión se encuentra en la etiqueta con nombre *"Collision Type"* y los distintos tipos de colisión son:

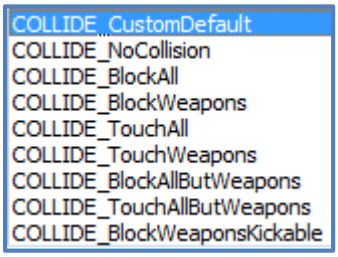


Imagen 191:Tipos de colisión.

- COLLIDE_NoCollision.- En este caso no se produce ninguna colisión con el objeto pudiéndose atravesar.
- COLLIDE_BlockAll.- Eligiendo esta opción el objeto se hace sólido ofreciendo colisión tanto en armas como en jugadores.
- COLLIDE_BlockWeapons.- Esta opción permite bloquear sólo las armas. En este caso se puede atravesar el objeto con el personaje principal pero sí que ofrece colisión en las armas.

- COLLIDE_TouchAll.- Con este tipo de colisión, los objetos pueden ser apilados, ofreciendo una colisión de contacto entre los distintos objetos.
- COLLIDE_BlockAllButWeapons.- Este tipo de colisión permite el disparar con las armas a través de los objetos. Sí ofrecen colisión ante el personaje principal pero no a las armas.
- COLLIDE_BlockWeaponsKickable.- Mediante este tipo de colisión se puede interactuar con los objetos. Se ofrece una colisión ante las armas, pero al disparar al objeto, éste interactuará según el disparo recibido. Como poder mover el objeto, volcarlo, etc.

7.7. Mostrar información en pantalla – UIScene.

La información mostrada por pantalla es llamada HUD en el motor Unreal. Ejemplos de la información mostrada puede ser el mapa del nivel y posición del jugador, nivel de vida que tiene el jugador, inventario de armas y número de proyectiles en cada una de las armas, etc. [40]

Otro tipo de información que puede mostrarse son menús de selección mediante botones e imágenes para pedir la interacción del jugador.

UDK ofrece un tipo especial de elementos para mostrar este tipo de información en pantalla. Este elemento es llamado *"UIScene"* pudiéndose crear mediante su correspondiente editor *"Unreal UIScene Editor"*, mostrado en la imagen 192.

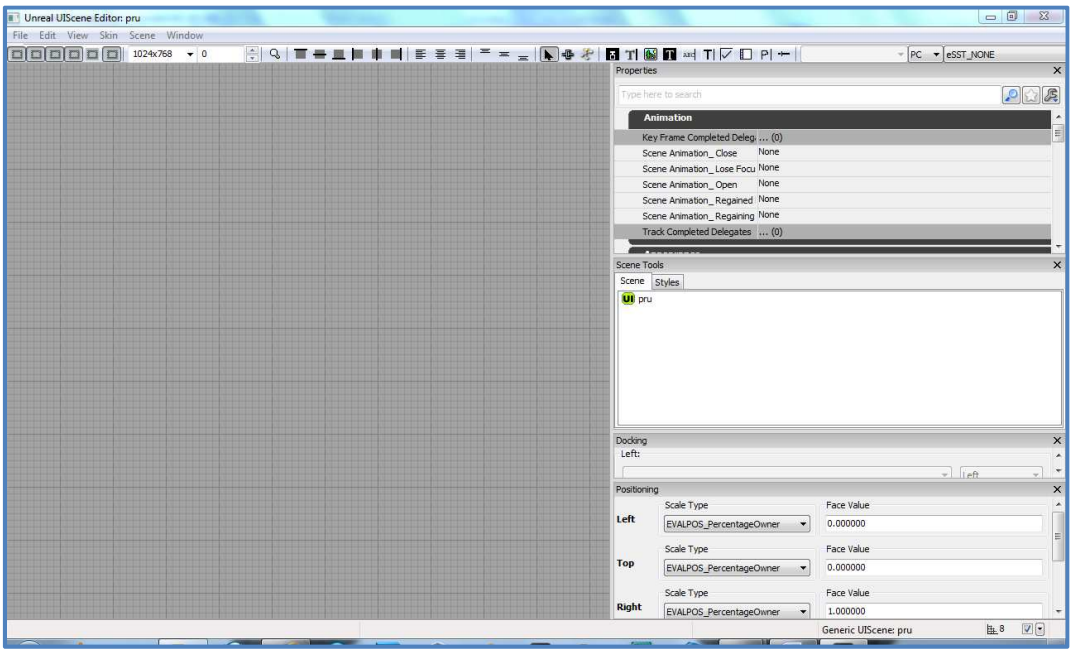


Imagen 192:Interfaz editor "Unreal UIScene".

Un *"UIScene"* es una pantalla que quedará superpuesta a la pantalla principal cuando se quiera mostrar.

En la parte izquierda del editor se muestra el tamaño de la pantalla en la cual se podrá ir añadiendo los distintos elementos. En la imagen 193 se muestra la edición de la pantalla.

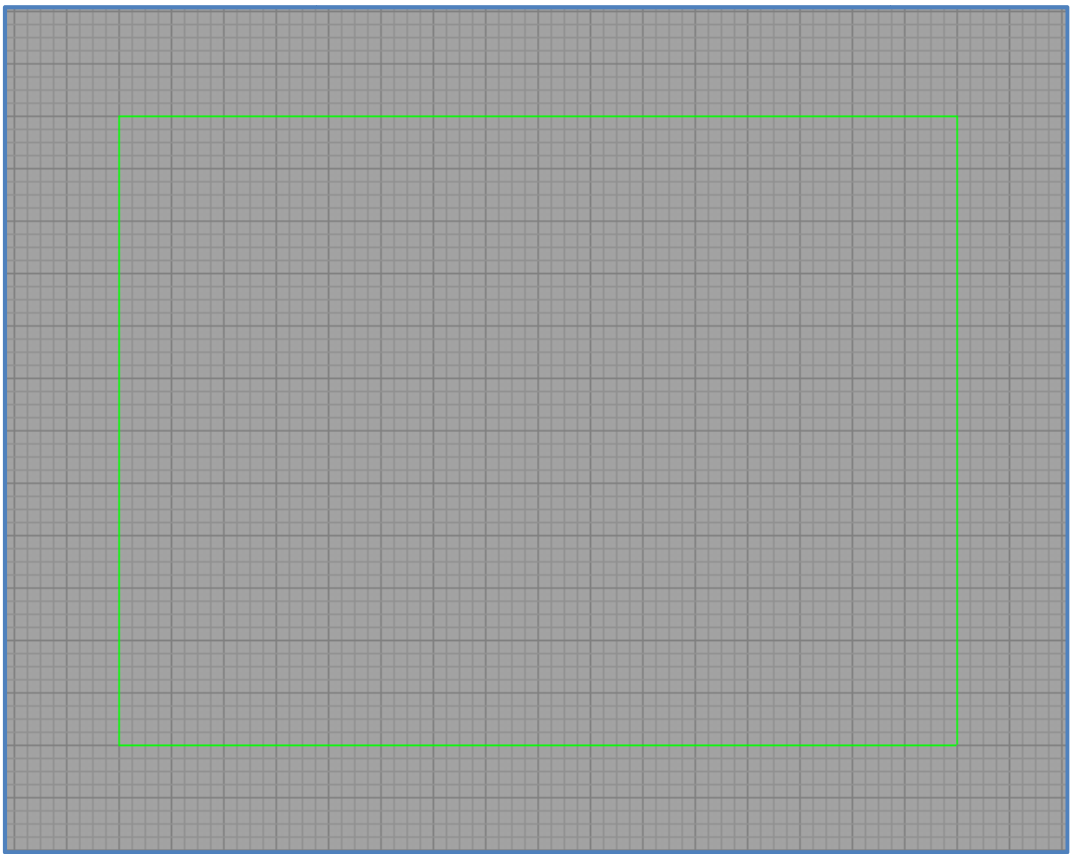


Imagen 193:Sección gráfica de "Unreal UIScene Editor".

Los elementos que se pueden añadir a la pantalla se encuentran en la barra de herramientas superior, presentados en la imagen 194.

Imagen 194:Herramientas de elementos a añadir.

Siendo cada uno de los elementos:

- Button .- Pulsando en este icono se puede añadir un botón.
- Edit Box .- Añadir elemento para agregar texto.
- Image .- Añadir una imagen en la pantalla.
- Label .- Añadir una etiqueta.
- Label Button .- Añadir un botón pudiendo insertar una etiqueta.
- Toggle Button .- Añadir un botón para poder modificar la característica *"Toggleable"* de ciertos objetos.
- Check Box .- Elemento para poder poner o quitar un tick.
- List .- Acoplar una lista de elementos.
- Panel .- Añadir un panel a la pantalla.

- Slider .- Añadir el elemento *"slider"* a la pantalla.

En la imagen 195 se muestran todos los elementos comentados anteriormente insertados en la pantalla:

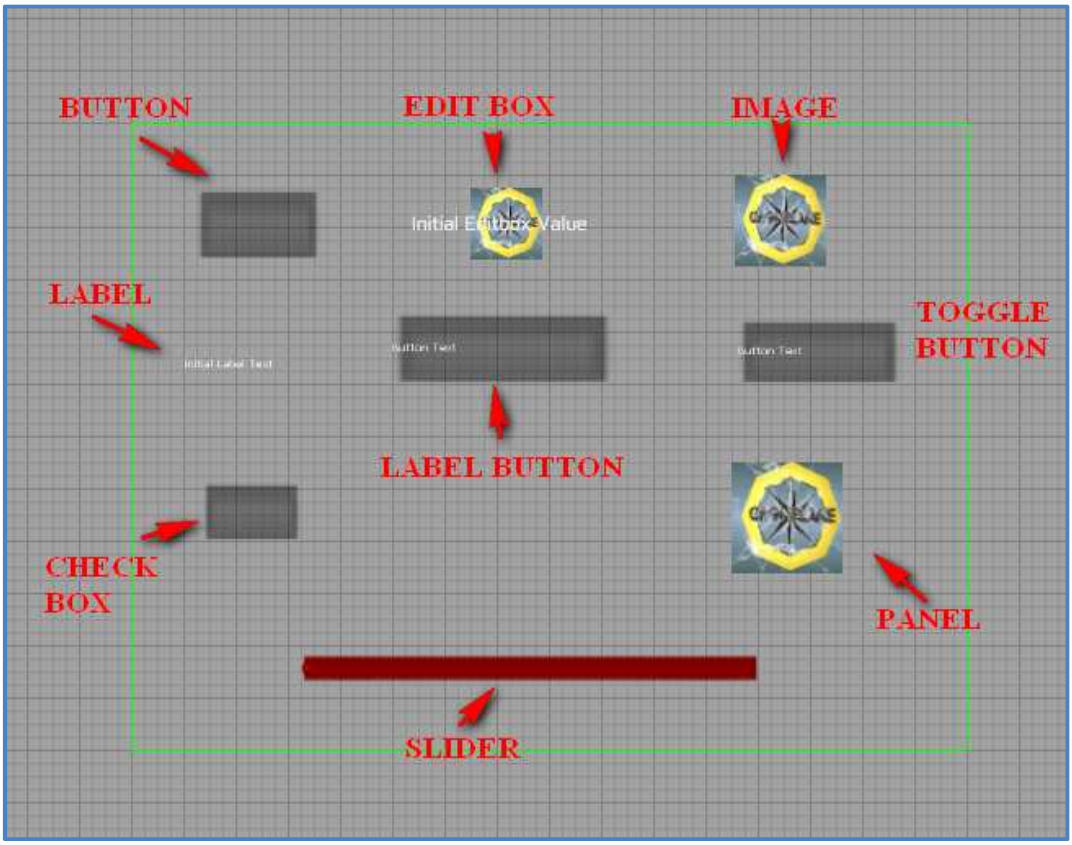


Imagen 195:Ejemplos elementos añadidos en "UIScene".

Las propiedades de cada uno de los elementos, al seleccionarlos, son mostradas en el panel *"Properties"* de la parte derecha del editor, presentadas en la imagen 196.

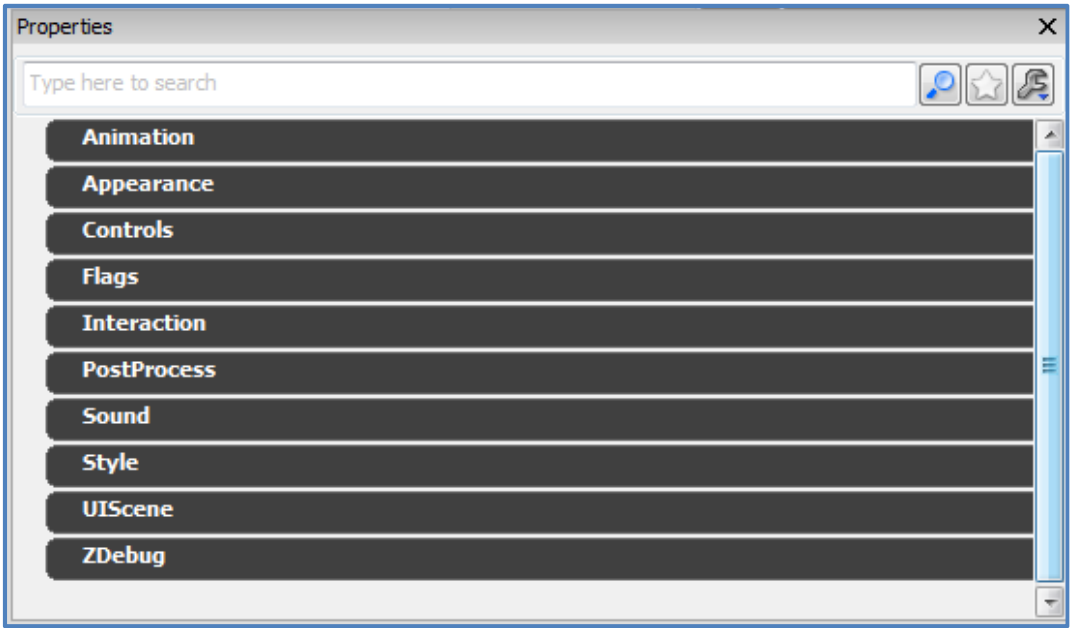


Imagen 196:Propiedades "UIScene".

Un elemento añadido a la pantalla puede tener su propio flujo de ejecución. Para crear el flujo de ejecución se debe seleccionar el elemento y en el menú mostrado al pulsar el botón derecho del ratón sobre él, seleccionar *"UnrealKismet Editor"*:

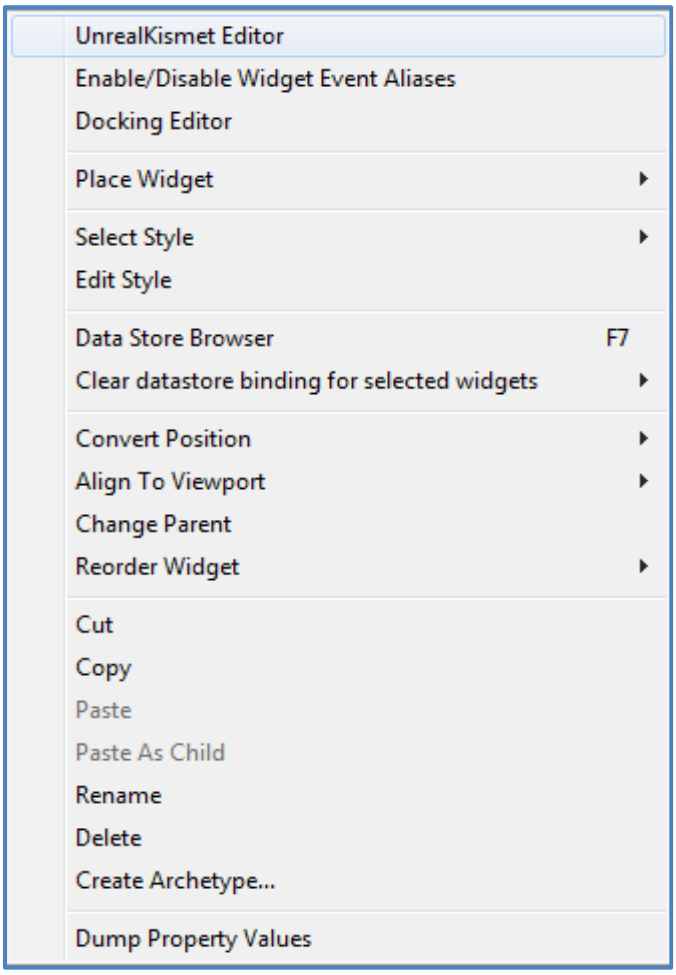


Imagen 197:Menú de selección "UnrealKismet Editor".

Se abrirá la herramienta Unreal Kismet que permitirá ir añadiendo elementos propios de la herramienta para realizar la dinámica deseada.

Cuando se crea una pantalla *"UIScene"* hay que tener en cuenta diversas opciones a la hora de mostrarla en la pantalla principal del juego.

Puede realizarse que al mostrar esta pantalla, el juego quede bloqueado hasta dejar de mostrarla. En las propiedades de la pantalla se tiene una etiqueta dentro de la sección *"Flags"* llamada *"Pause Game while Active"* que activada, al mostrar la pantalla creada, el juego se bloqueará.

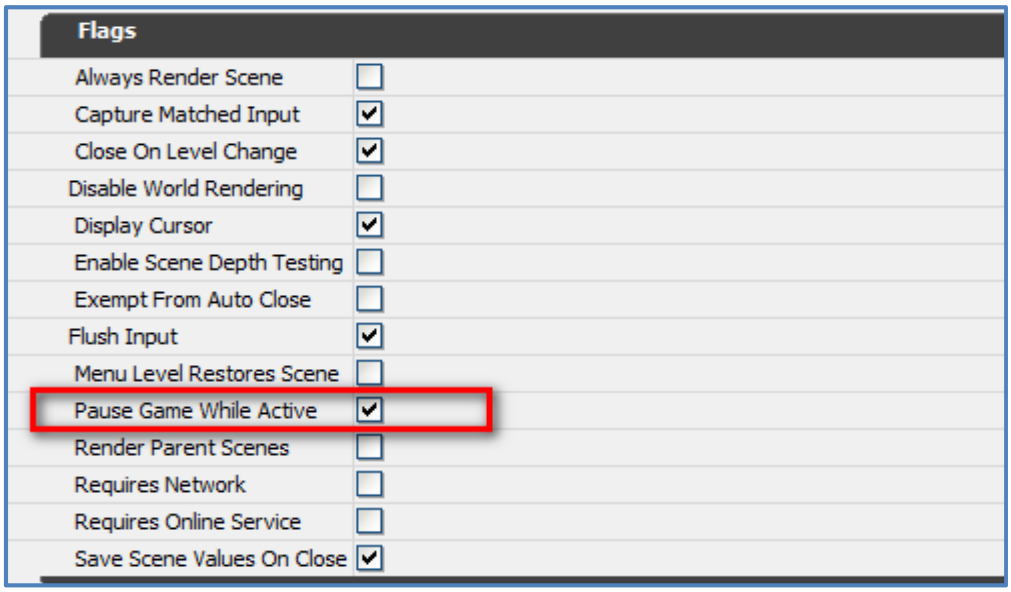


Imagen 198:Etiqueta "Pause Game While Active".

Otra opción que se permite es que el cursor del ratón aparezca o no mientras se está mostrando la pantalla creada. En las propiedades de la pantalla se tiene una etiqueta dentro de la sección *"Flags"* llamada *"Display Cursor"* que activada, el cursor del ratón se mostrará en la pantalla.

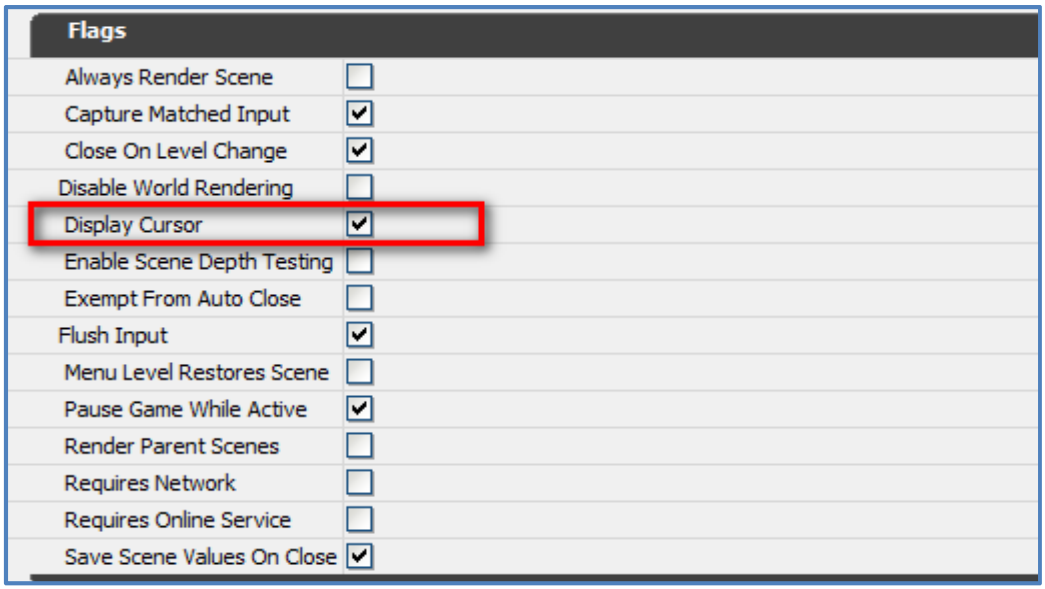


Imagen 199:Etiqueta "Display Cursor".

Con estas características puede crearse una pantalla, HUD, que informe a lo largo del desarrollo del juego. Para ello habría que quitar el tick tanto a la propiedad *"Display Cursor"* como a la propiedad *"Pause Game While Active"* para no bloquear el juego ni mostrar el cursor.

Para crear esta información también hay que modificar la etiqueta *"Scene Input Mode"* de la sección *"Interaction"* seleccionando *"INPUTMODE_None"*.

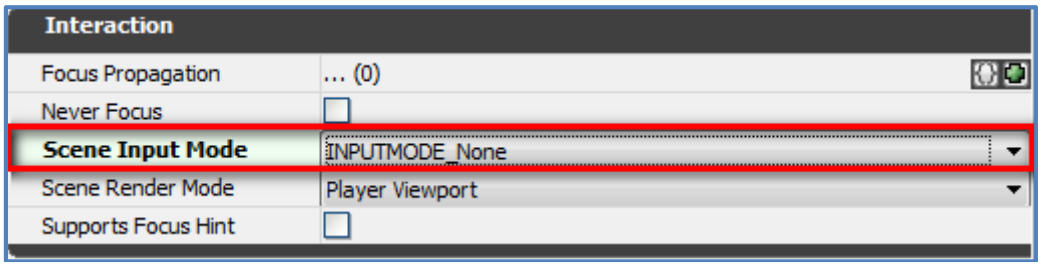


Imagen 200:Etiqueta "Scene Input Mode".

Para mostrar las pantallas *"UIScene"* se debe utilizar la herramienta Unreal Kismet del editor. Dicha herramienta se puede abrir mediante el icono de la barra de herramientas superior del editor. Esta herramienta contiene un elemento para poder abrir una *"UIScene"*. Este elemento se llama *"Open Scene"* y se puede obtener de la siguiente manera:

Botón derecho del ratón → New Action → UI Scenes → Open Scene.

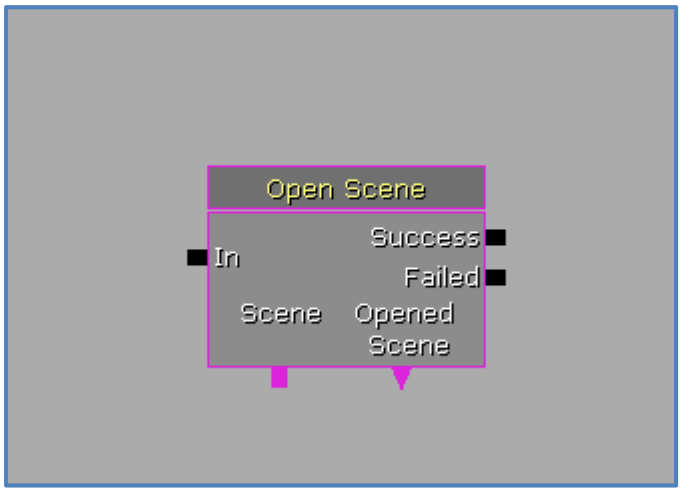


Imagen 201:"Open Scene".

En las propiedades de este elemento se debe indicar la *"UI Scene"* que se quiere mostrar, mostradas en la imagen 202.

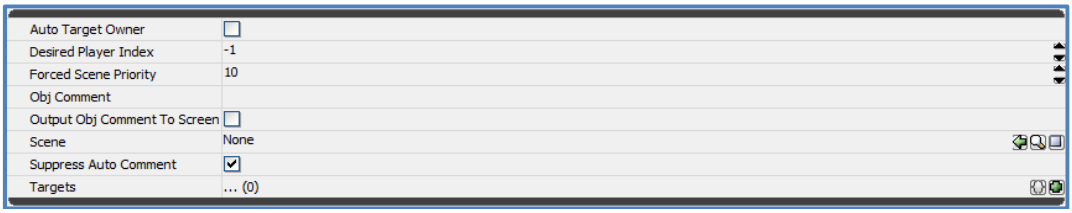


Imagen 202:Propiedades "Open Scene".

En la etiqueta llamada *"Scene"* es donde se indicará la *"UI Scene"*. Para añadirla hay que seleccionar el elemento *"UI Scene"* en el Content Browser y volviendo a las propiedades de *"Open Scene"* se pulsa el icono de la etiqueta.

Al igual que se puede abrir una *"UI Scene"*, también se puede cerrar. El elemento proporcionado para poder cerrarla es llamado *"Close Scene"* obteniéndolo:

Botón derecho del ratón → New Action → UI Scenes → Close Scene

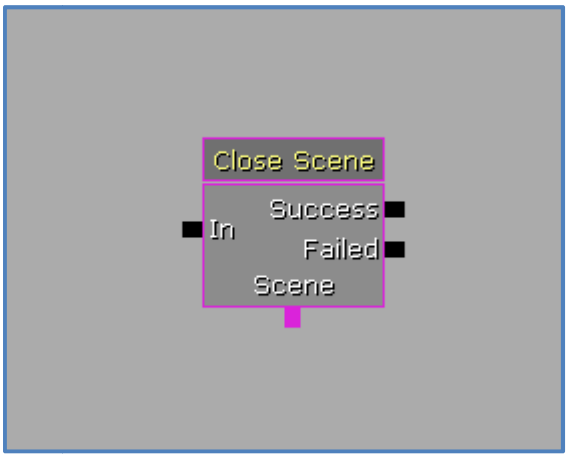


Imagen 203: "Close Scene".

En las propiedades de este elemento se indicará la *"UI Scene"* a cerrar en la etiqueta con nombre *"scene"* y añadiéndola de la misma manera que en el elemento *"Open Scene"*.

Estos elementos de apertura y cierre de los *"UI Scene"* pueden ser acoplados a cualquier otro elemento del flujo de secuencia y a cualquier acción o evento creados en el nivel.

7.8. Volúmenes.

En el nivel creado se pueden insertar volúmenes con los que se puede modificar el flujo del juego, modificar las propiedades del jugador ó simular ciertos efectos de la naturaleza. [41]

Para poder acoplar un volumen al nivel hay que presionar el botón derecho del ratón sobre el icono de la sección izquierda de la pantalla.

Se mostrará una nueva ventana mostrando todos los tipos de volúmenes que se pueden acoplar, mostrada en la imagen 204.

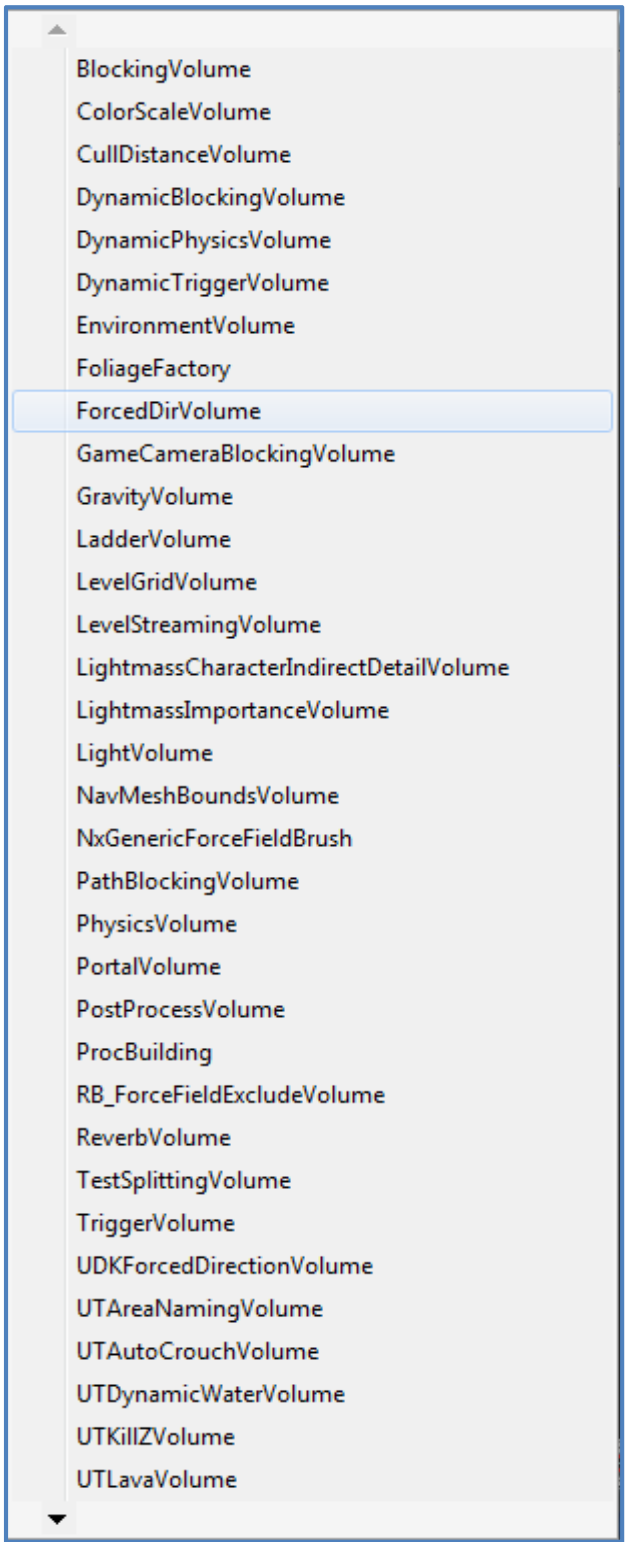


Imagen 204:Tipos de volúmenes.

Physics Volume.

Mediante este volumen se pueden modificar las propiedades del jugador al hacer contacto con él, como modificar el nivel de vida, etc.

Todas las modificaciones se indican en las propiedades del volumen, obteniéndolas pulsando la tecla ‘F4’ del teclado:

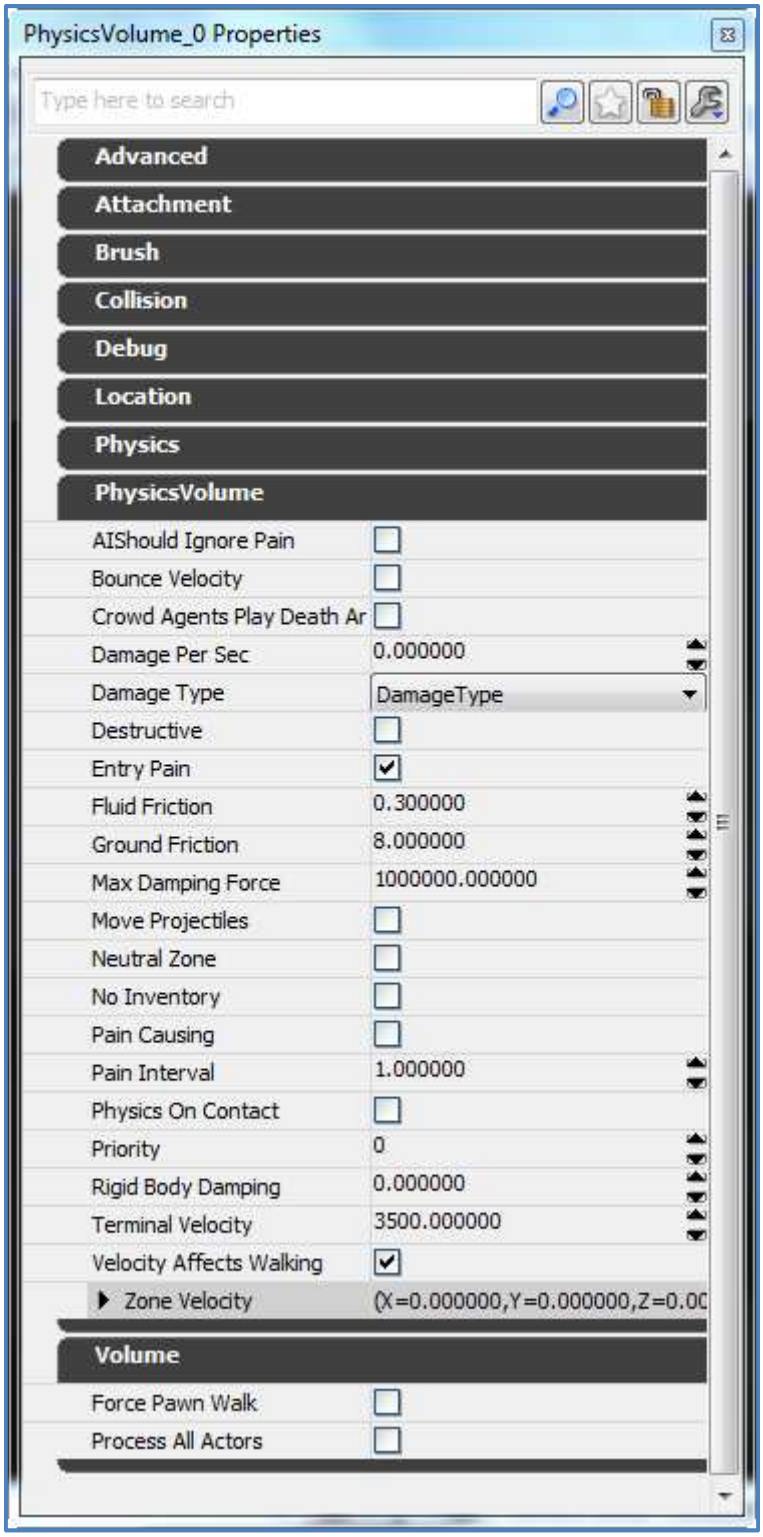


Imagen 205:Propiedades "PhysicsVolume".

- Etiquetas *"Pain Causing"* y *"Damage Per Sec"*.- Activando la primera de las etiquetas se indica que el personaje sufrirá daño, el indicado en la segunda de las etiquetas.

- Etiqueta *"Ground Friction"*.- Se utiliza para controlar la fricción del jugador con el suelo. Mediante la manipulación del valor de la etiqueta se pueden simular zonas de hielo ó aceite.
- Etiqueta *"Zone Velocity"*.- Para aplicar una fuerza en alguna de las direcciones de algún eje de coordenadas.
- Etiqueta *"PhysicsOnContact"*.- Activándola se ejercen las fuerzas en el plano de contacto, y no cuando se entra en el volumen.
- Etiqueta *"Bounce Velocity"*.- Combinada con el valor positivo de Z de la etiqueta *"ZoneVelocity"* se realiza que los objetos que caigan sobre el volumen reboten.
- Etiqueta *"Neutral Zone"*.- Los objetos que se encuentran dentro del volumen no reciben daño.
- Etiqueta *"RigidBodyDamping"*.- Pérdida de energía de los cuerpos rígidos.

Dinamic Physics Volume.

Este volumen es igual al anterior comentado, sólo que la localización y rotación pueden ser animadas mediante la herramienta Unreal Matinee.

Gravity Volume.

Es utilizado para modificar la fuerza de la *"gravedad"* en el nivel. En sus propiedades se puede modificar el valor deseado:

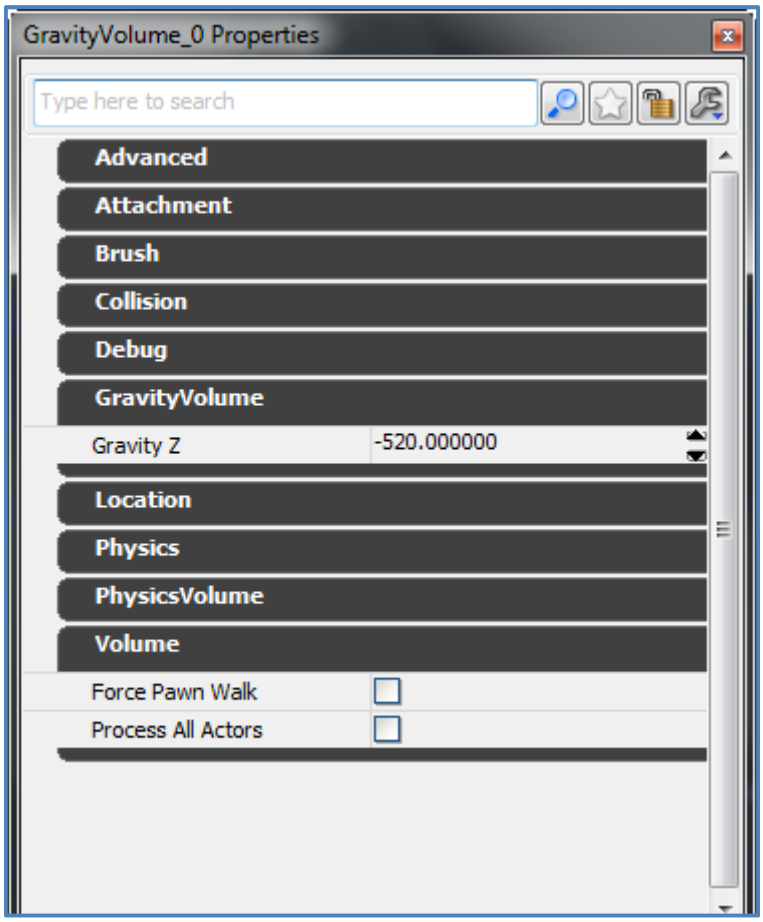


Imagen 206:Propiedades "Gravity Volume".

En la etiqueta *"Gravity Z"* de la sección *"GravityVolume"* es donde se indica el valor. Insertando un valor negativo, la fuerza se ejerce hacia el suelo. Con un valor positivo la fuerza se ejerce en sentido contrario, en contra del suelo.

UT Kill Volume.

Los personajes que entren en este volumen perderán toda su vida, morirán.

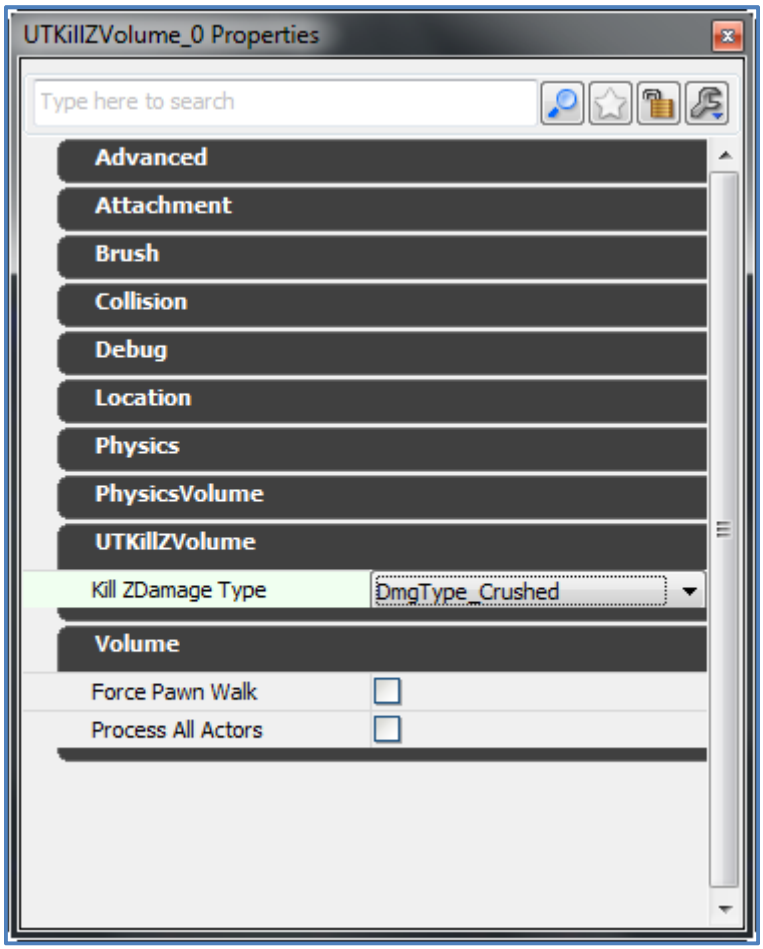


Imagen 207:Propiedades "Kill Volume".

En la etiqueta *"Kill ZDamage Type"* de la sección *"UTKillZVolumne"* se indicará el tipo de daño que recibirá el personaje.

UTWater Volume.

Volumen que simula los efectos que se producen al entrar en agua, como presentando flotación, etc.

Para realizar todos los cambios deseados hay que abrir las propiedades, presionando la tecla 'F4' del teclado:

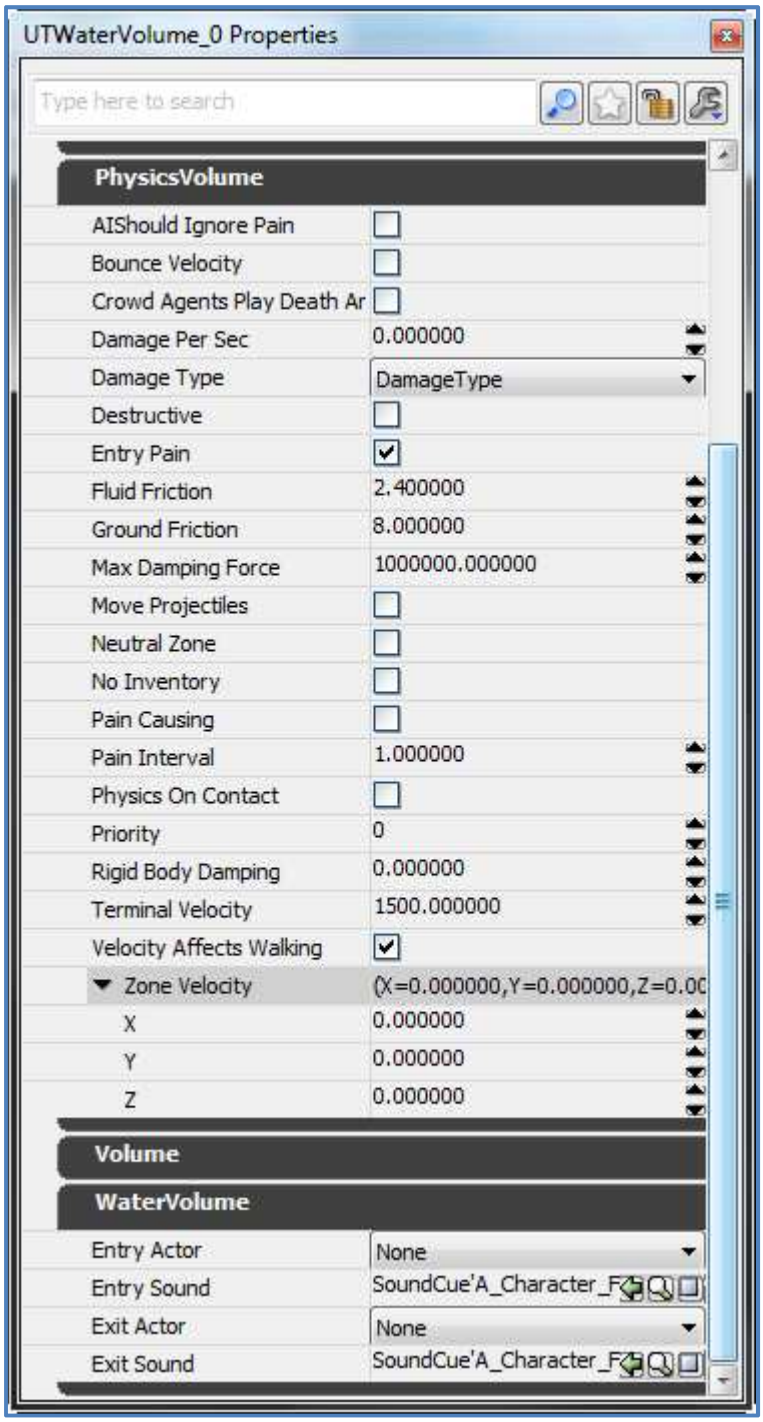


Imagen 208:Propiedades "Water Volume".

- Etiqueta *"PhysicsVolumne → Fluid Friction"*.- Controla la resistencia del "agua" cuando el personaje se encuentra en ella.
- Etiqueta *"WaterVolume → Entry Actor"*.- Utilizado para indicar qué actor se quiere mostrar al entrar en el volumen, como "splash effects".
- Etiqueta *"Water Volume → Exits Actor"*.- Utilizado para indicar qué actor se quiere mostrar al salir del volumen, como *"splash effects"*.

- Etiqueta *"Water Volume → Entry Sound"*.- Sonido a reproducir cuando el personaje entra en el volumen. El sonido seleccionado debe ser de tipo *"Sound Cue"*.
- Etiqueta *"Water Volume → Exit Sound"*.- Sonido a reproducir cuando el personaje sale del volumen. El sonido seleccionado debe ser de tipo *"Sound Cue"*.
- Etiqueta *"Pawn Entry Actor"*.- Para modificar el Actor del personaje principal al entrar en el volumen.
- Etiqueta *"Zone Velocity → Z"*.- Modificando el valor de esta etiqueta se puede hacer que el personaje *"flote"* o se *"hunda"*, según el valor positivo ó negativo del valor, a velocidad más o menos rápida.

UT Space Volume.

Simula las propiedades del espacio exterior.

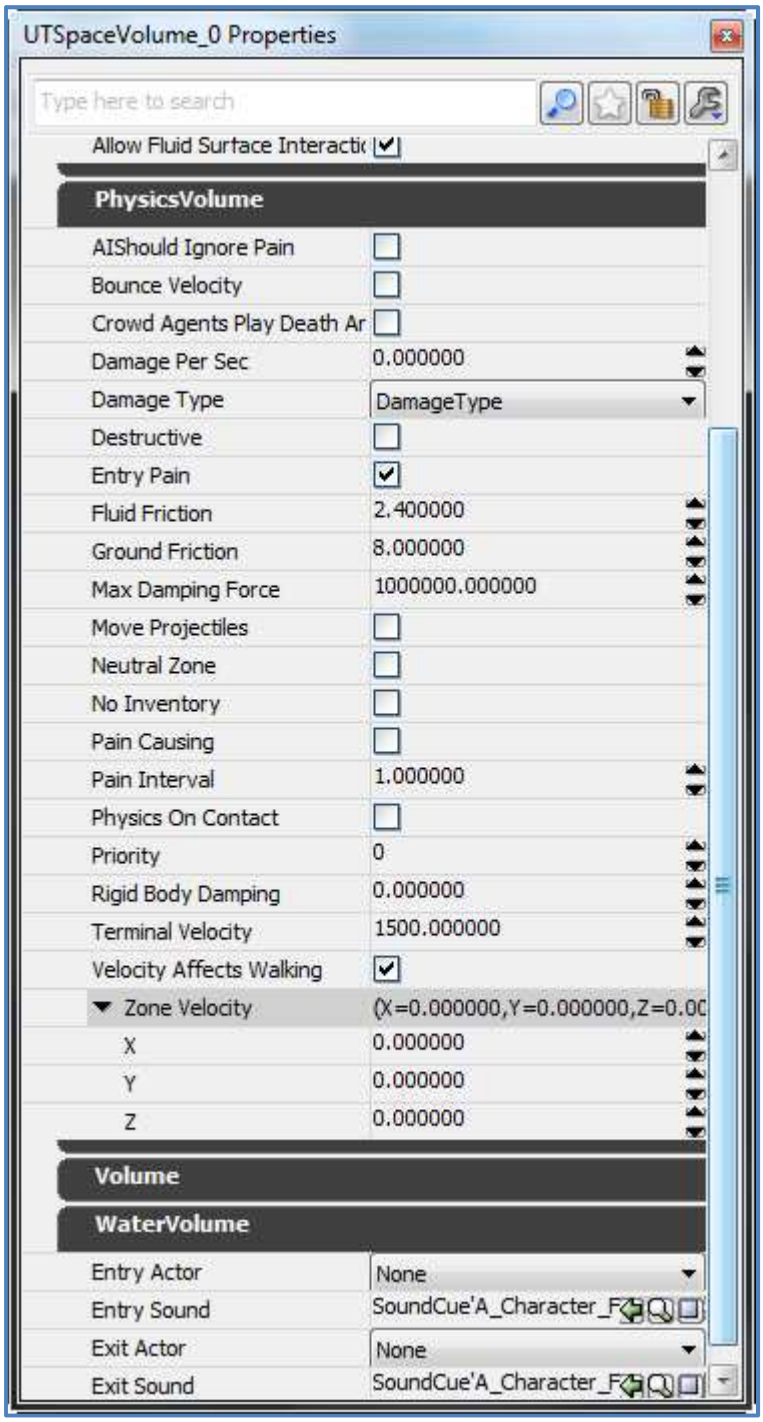


Imagen 209:Propiedades "Space Volume".

Ladder Volume.

Utilizado para simular escaleras, con las que el jugador podrá subir y bajar paredes. La dirección de escalado en el volumen es mostrada mediante una flecha. En las propiedades se pueden modificar diversos aspectos:

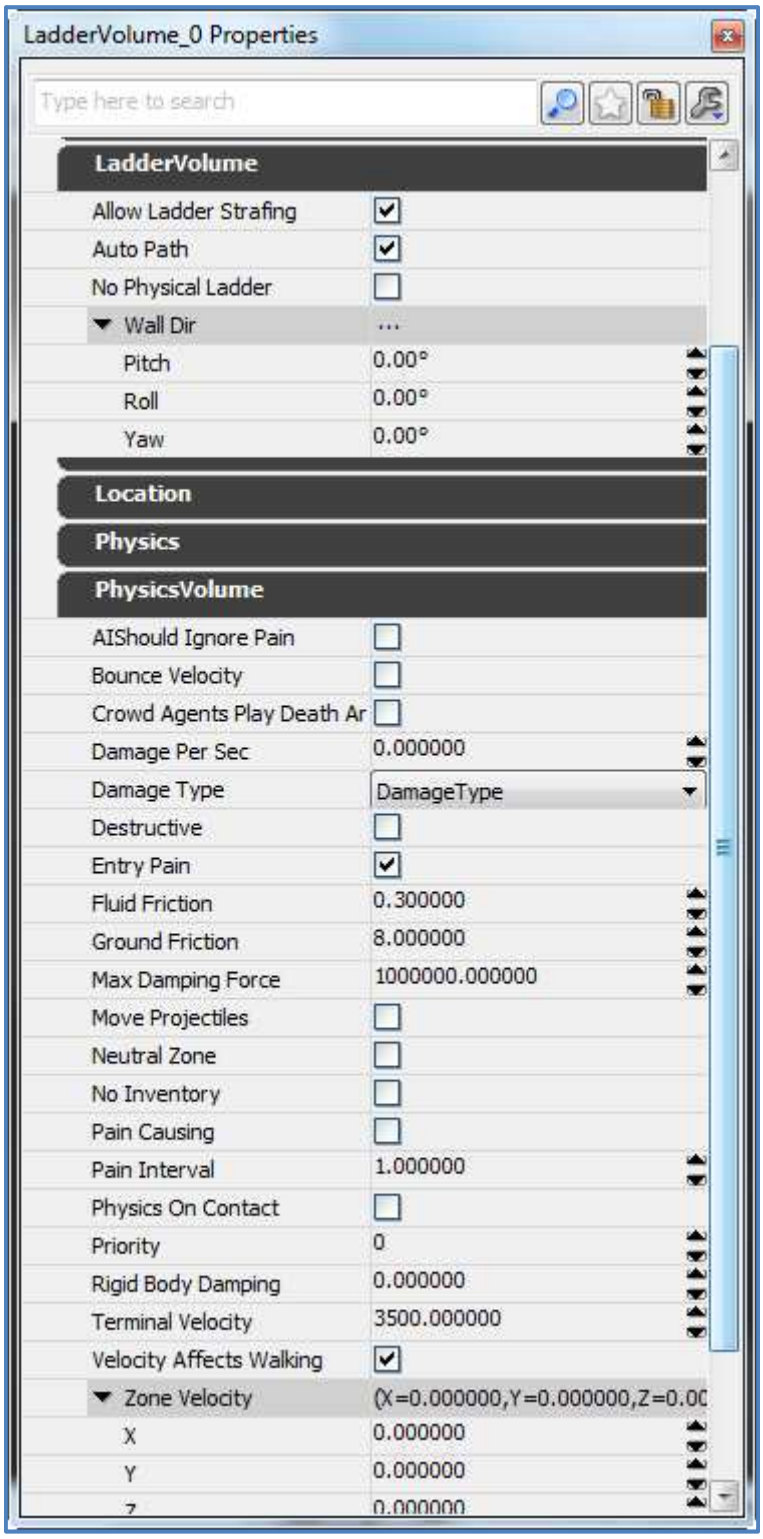


Imagen 210:Propiedades "Ladder Volume".

- Etiqueta *"AutoPath"*.- Teniendo activada esta etiqueta se crean automáticamente los objetos *"Ladder Actors"* en los extremos superior e inferior del volumen. Si no se activa, habrá que situar estos objetos manualmente.

- Etiqueta *"Allow Ladder Strafing"*.- Para permitir el movimiento horizontal del personaje en medio del volumen. Si no se activa, sólo se realizará movimiento vertical.
- Etiqueta *"NoPhysicalLadder"*.- Si se encuenta activa, el jugador comenzará a escalar en cuanto entre en el volumen, independientemente si hay superficie escalable o malla estática.

Cuando el volumen es acoplado, se debe compilar mediante *"Build → AI Paths"* para que aparezcan los objetos *"Ladder Actors"* representados por una escaleras pequeñas. Las flechas que poseen estas escaleras indican hacia dónde debe ser la vista del personaje para poder escalar por el volumen.

Trigger Volume.

Volumen utilizado para lanzar eventos al hacer contacto con él.

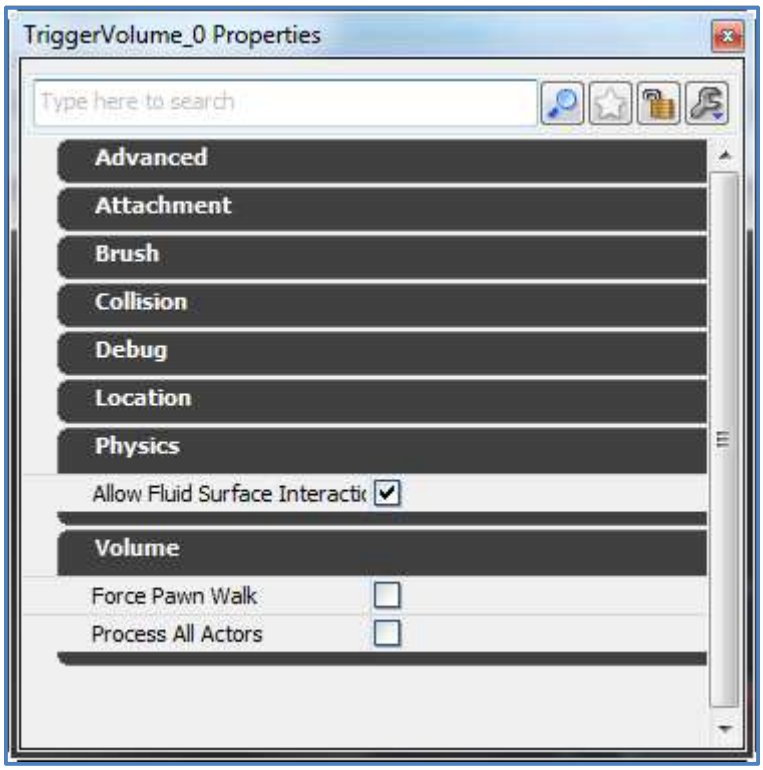


Imagen 211:Propiedades "Trigger Volume".

Dynamic Trigger Volume.

Tipo de volumen igual al anterior pero la localización y rotación pueden ser modificados por la herramienta Unreal Matinee.

Post Process Volume.

Permite modificar el *"post-process"* del nivel, como difuminar la cámara cuando se entra en el volumen, etc.

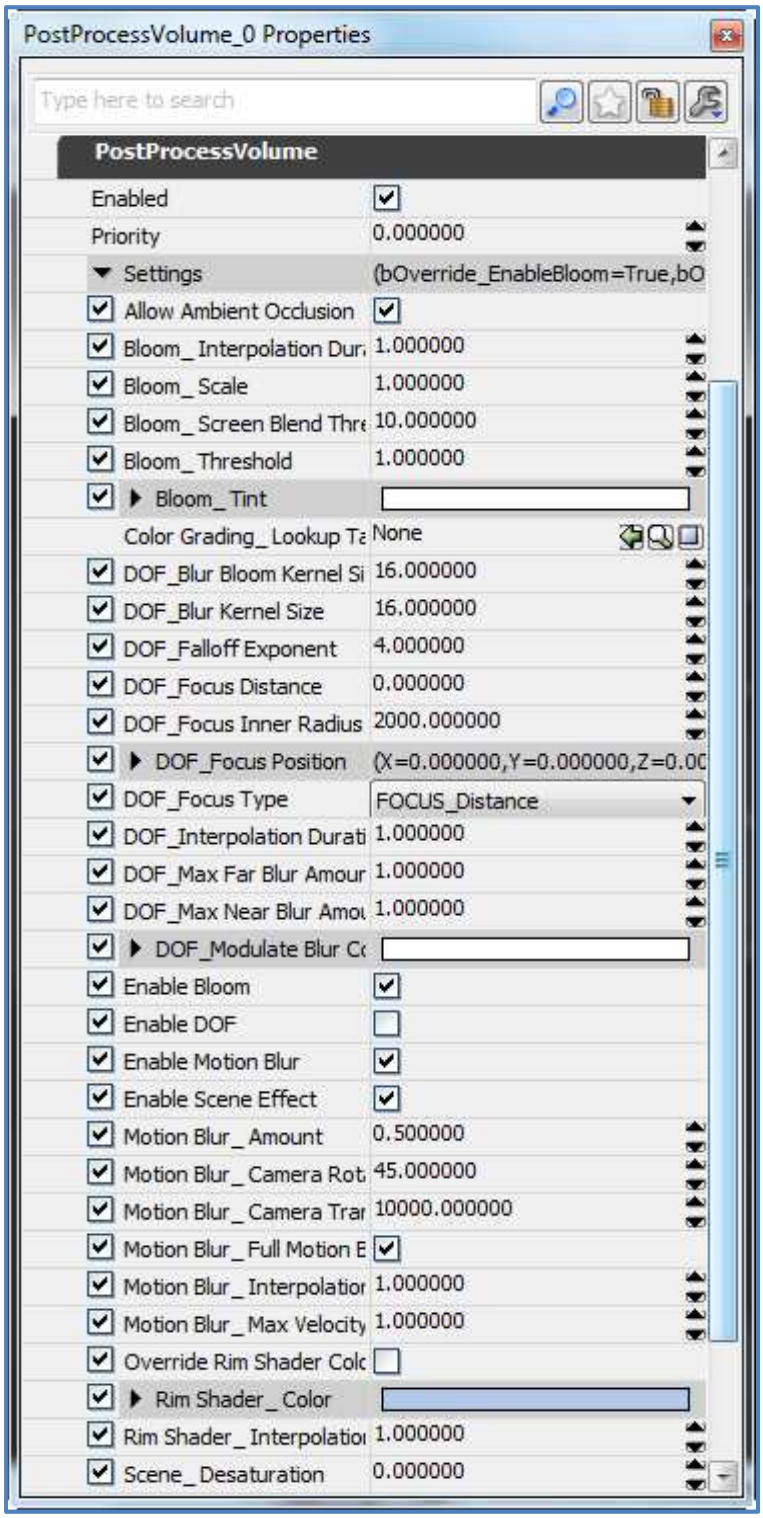


Imagen 212:Propiedades "Post Process Volume".

Mediante todas las etiquetas propias del volumen, se puede modificar los efectos "*post-process*" al entrar en él. Se puede difuminar la cámara, poner niebla, etc.

TRUCO: Si se quiere obtener la forma/modelo de las mismas dimensiones que un volumen ya creado se utiliza:

Seleccionar volumen creado → botón derecho del ratón → Polygons → To Brush.

Y ya se tiene la forma/modelo exactamente igual que el volumen.

7.9. Lanzar el nivel/mapa creado.

Mientras se está creando un nivel, se tiene la opción de poder lanzarlo en el motor para comprobar lo implementado, ver cómo se va construyendo, etc.

Una condición que hay que realizar antes de poder lanzar el nivel es compilarlo. Los elementos básicos que hay que compilar de un nivel son la iluminación y toda la dinámica de ejecución:

- Compilar la iluminación.- La iluminación puede ser compilada mediante el menú principal de la aplicación en la etiqueta *"Build"*.

  Pulsando en esta etiqueta se mostrará un nuevo submenú donde se muestra la opción de compilar sólo la iluminación (Lighting), mostrada en la imagen 213.

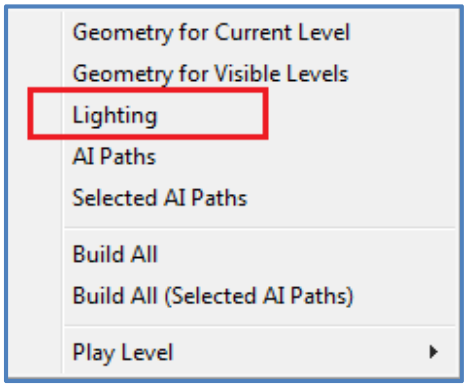


Imagen 213:Menú "Build".

  Seleccionando este tipo de compilación se mostrará un nuevo menú donde se pueden modificar las opciones de compilación:

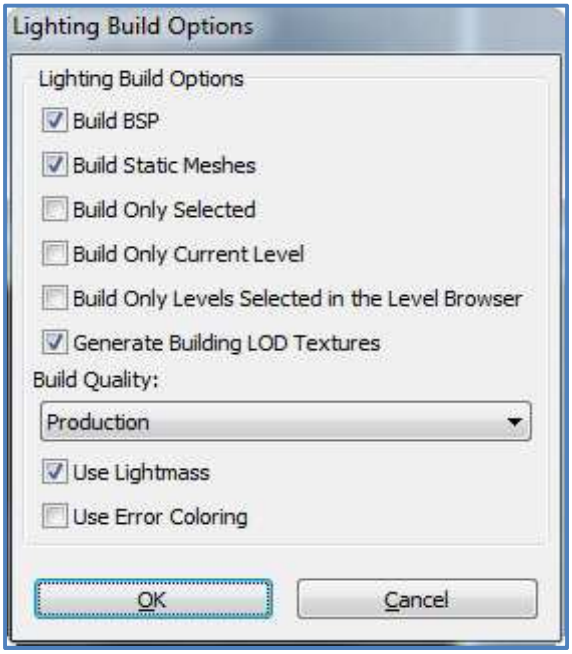


Imagen 214:Menú compilación iluminación.

En este nuevo menú hay que quitar el tick en la etiqueta *"Use Lightmass"* para no realizar la compilación mediante la herramienta gráfica. Sin utilizar la herramienta gráfica la compilación será más rápida.

Pulsando en el botón *"OK"* se realizará toda la compilación de la iluminación que se tiene acoplado al nivel.

El menú de opciones de compilación de la iluminación también puede ser mostrado mediante el icono de la barra de herramientas superior del editor.

- Compilar la dinámica de ejecución.- La dinámica de ejecución puede ser compilada mediante el menú principal de la aplicación en la etiqueta *"Build"*. Pinchando en esta etiqueta se mostrará un nuevo submenú donde se muestra la opción de compilar la dinámica (AI Paths):

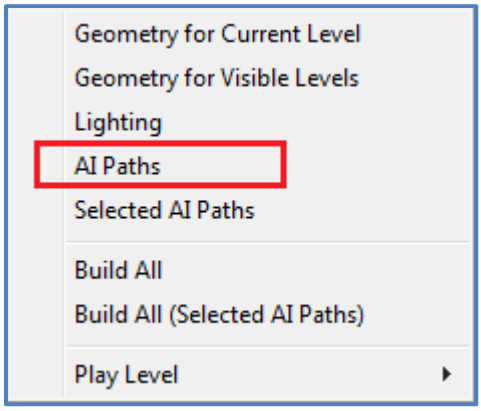


Imagen 215:Selección "AI Paths".

- Esta compilación puede ser realizada también mediante el icono de la barra de herramientas superior del editor.

Realizando esta compilación, el nivel ya puede ser lanzado, pudiéndose jugar en él para comprobar lo implementado.

Al finalizar por completo el nivel, es aconsejable realizar una compilación completa de todos los elementos que forman parte de él. Para realizar esta compilación hay que seleccionar la opción *"Build All"* del menú mostrado por *"Build"*:

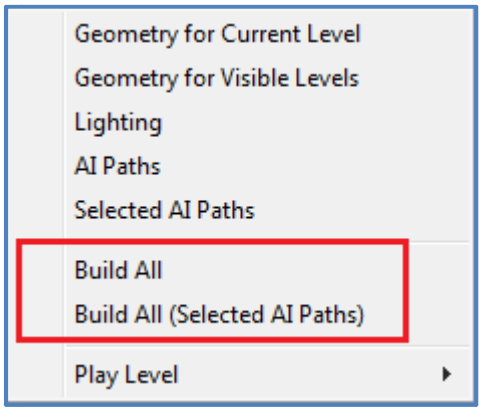


Imagen 216:Selección compilación completa.

Otra forma de realizar la compilación completa del nivel es mediante el icono que se presenta en la barra de herramientas superior del editor.

Cuando se lanza esta compilación, es ejecutada una aplicación que muestra cada uno de los tipos de elementos que se van compilando al igual qué tanto por ciento del nivel ha sido compilado. Esta aplicación es llamada *"Swarm Agent"* y su interfaz es mostrada en la imagen 217.

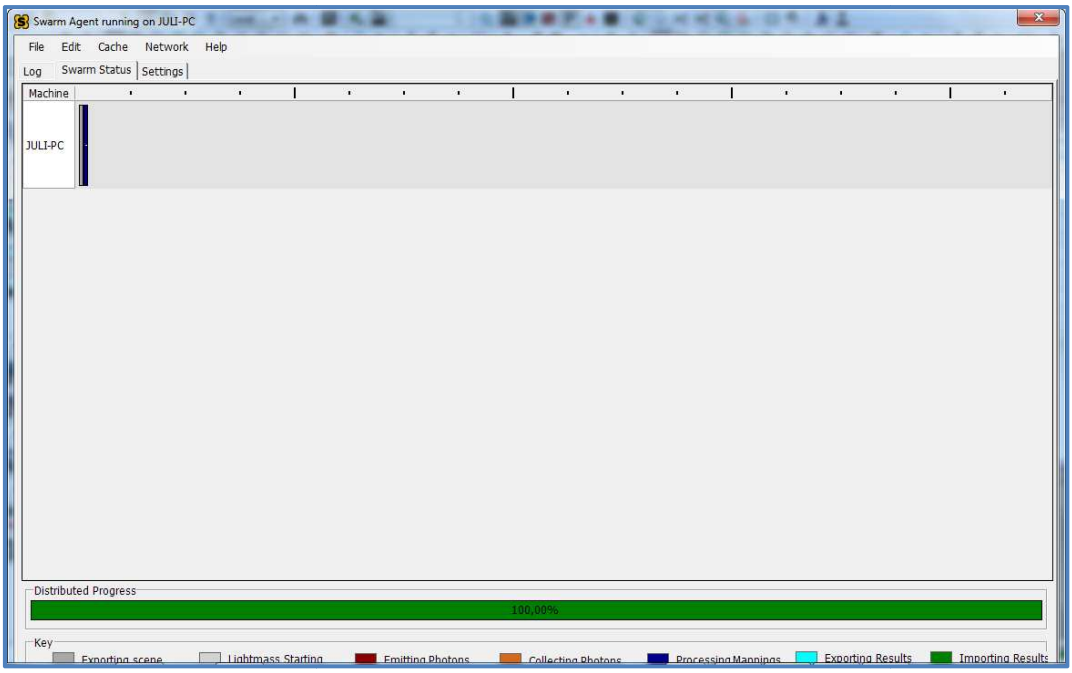


Imagen 217: Interfaz "Swarm Agent".

Al finalizar cualquier compilación del nivel, se muestra información de todos los errores y avisos encontrados, a la vez que todos los elementos que han sido compilados, como se muestra en las imagenes 218 y 219.

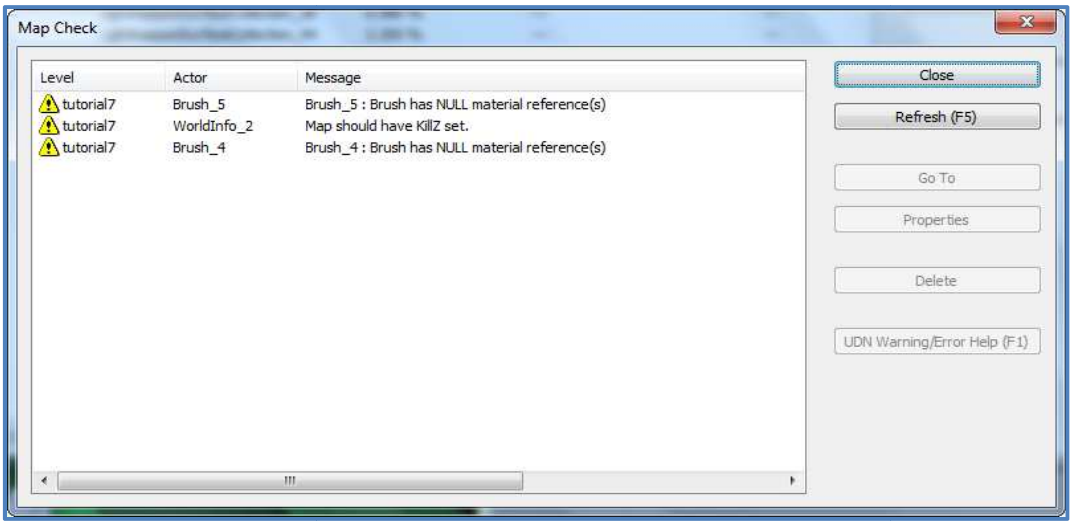


Imagen 218:Ejemplo resultados compilación.

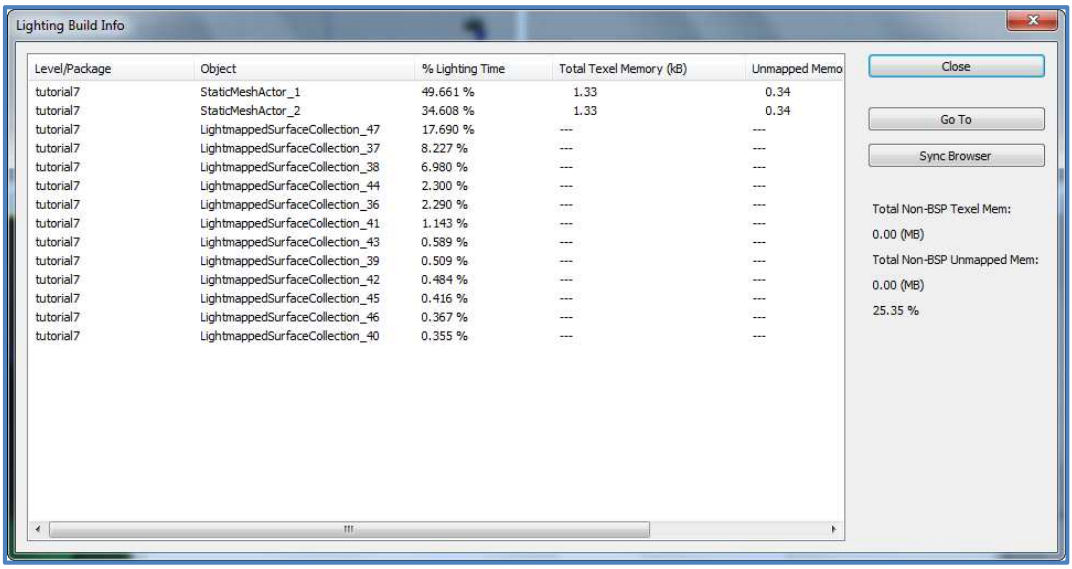


Imagen 219:Ejemplo resultados compilación de iluminación.

Una vez realizada la compilación, el nivel creado puede ser lanzado de dos maneras diferentes:

- De una manera simple.- Se empezará a jugar simplemente en el nivel.
  El lanzar el nivel de esta forma también tiene distintas opciones:
  - Lanzarlo desde un punto indicado por ratón:
    De esta forma el personaje principal (jugador) aparecerá en el punto del nivel indicado por el ratón.

    En el punto que se quiera lanzar el mapa se pulsa el botón derecho del ratón. En el menú contextual, mostrado en la imagen 220, que aparece se selecciona *"Play from Here"*.

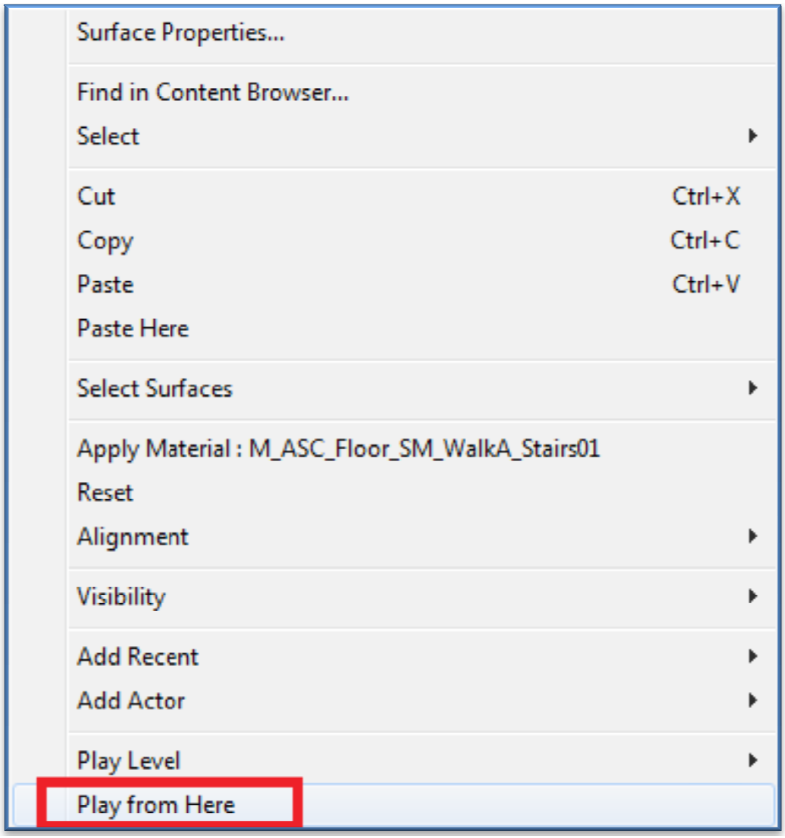


Imagen 220:Localización "Play from Here".

Se abrirá una nueva ventana con el nivel lanzado desde el punto deseado donde se podrá jugar.

Hay otra opción de lanzar el nivel de esta manera sin mostrarlo en una nueva ventana. El juego se lanzaría en la misma vista del editor. En el submenú contextual mostrado anteriormente se elige *"Play Level"* mostrando un nuevo submenú donde se elige *"Play from Here in ViewPort"*.

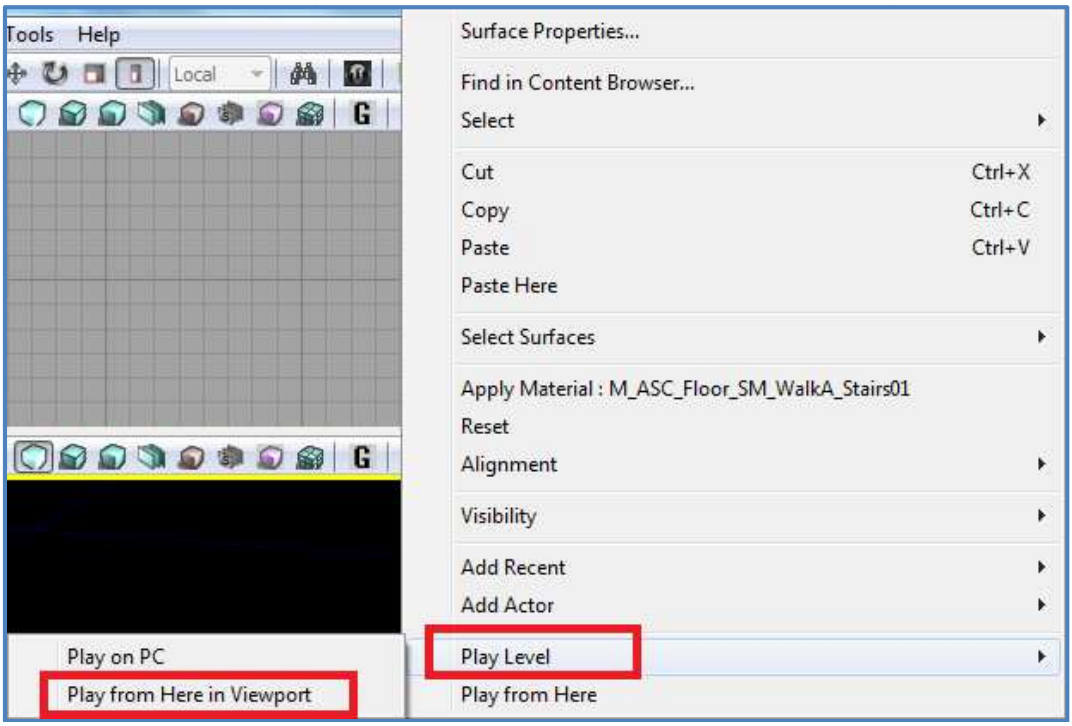


Imagen 221:Situación "Play from Here in Viewport".

- Lanzarlo desde el elemento *"PlayerStart"*.
  Si en el nivel se ha acoplado el elemento *"PlayerStart"* se puede lanzar comenzando el juego desde ese punto.
  En la barra de herramientas superior del editor se debe pulsar el icono y se iniciará el juego en una ventana nueva.

NOTA: En los dos casos anteriores, para terminar la ejecución del juego hay que presionar la tecla 'Esc' del teclado.

- Otra forma de lanzar el nivel es incorporándolo en toda la estructura de juego completo proporcionada por el motor UDK.

  Dicha estructura de juego completo está compuesta por todos los menús de inicio del juego, pausa, fin de ejecución, etc.

  Para iniciar esta ejecución hay que pulsar el icono de la barra de herramientas superior del editor. La ejecución completa del juego con el nivel creado se lanzará en una nueva ventana.

  Otra forma de lanzar esta ejecución es pulsar con el botón derecho del ratón sobre cualquier zona de cualquier vista del editor. En el menú contextual que se despliega seleccionar *"Play Level"* que a su vez se desplegará otro nuevo submenú. En este nuevo submenú elegir *"Play on PC"* y la ejecución se realizará en una nueva ventana.

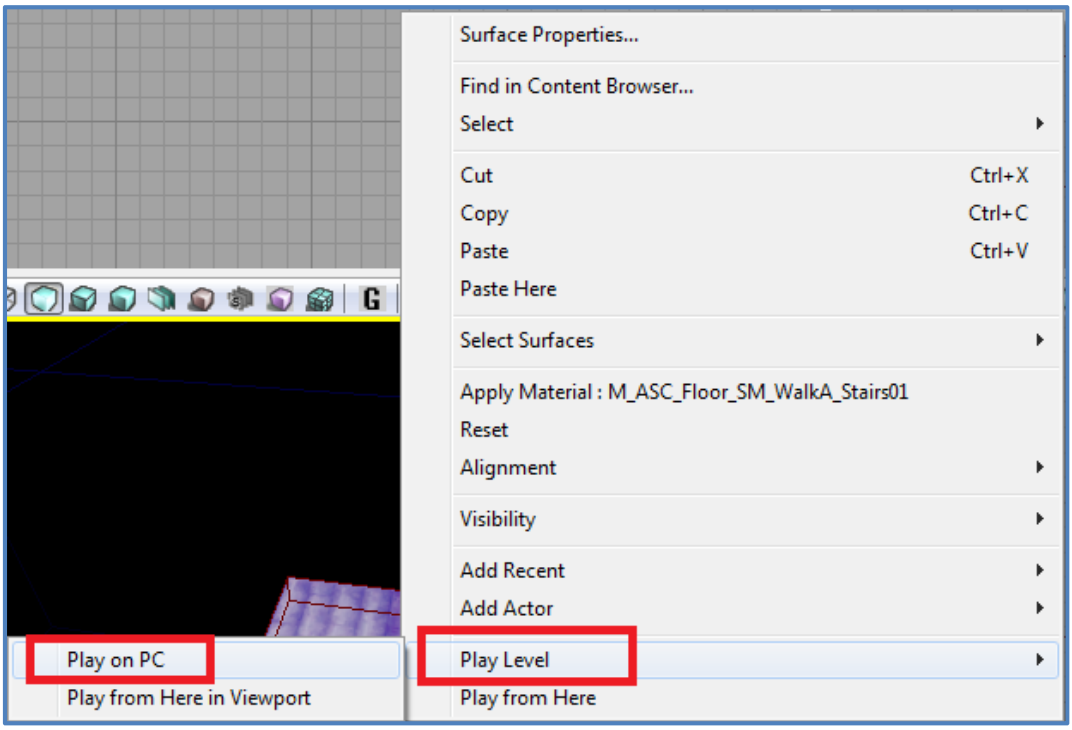


Imagen 222:Situación "Play on PC".

8. TUTORIALES.

En este capítulo se presentan diversos tutoriales en los cuales se muestra cómo realizar distintas acciones y situaciones de un juego mediante el motor UDK. [42] [43]

En dichos tutoriales se especificarán cada uno de los pasos que hay que seguir para implementarlo.

8.1. Tutorial 1: Crear puerta.

Uno de los elementos más usados y necesarios a la hora de crear un nivel es el poder utilizar puertas o similar. [44]

La secuencia del conjunto de la puerta es el realizar un hueco en la pared e interponer un objeto (mesh) en ese hueco que actúe como puerta, que podrá ser trasladado para permitir el paso ó prohibirlo.

Pasos a seguir:

1. Se creará un volumen horizontal que representará el suelo:
   Al crear el nuevo nivel se debe elegir el modo *"Additive"*.
   Se modifica el tamaño de la forma indicando el deseado para el suelo, pulsando en el icono , *"CSG Add"*, para agregarlo al nivel.

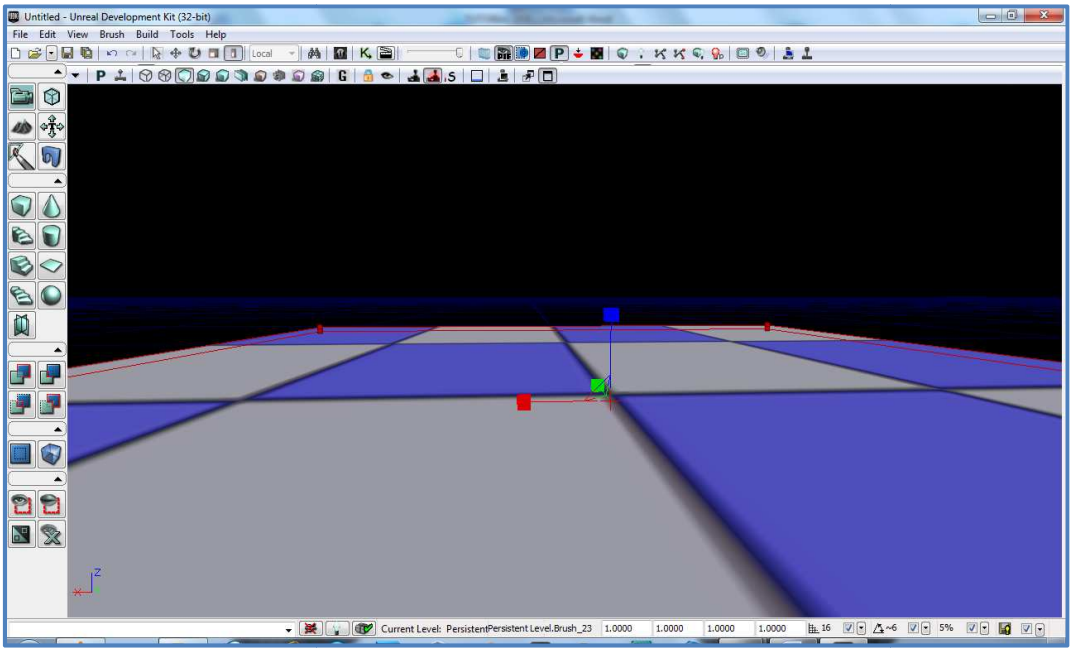


Imagen 223:Tutorial 1-Suelo.

2. Se modificará el tamaño de la forma/modelo para acoplar un volumen vertical que representará una pared. Esta pared se colocará en medio del volumen anterior (suelo) impidiendo el paso.
   Cuando se haya colocado la forma/modelo en el lugar y con el tamaño deseado, se pulsará en el icono , *"CSG Add"*.

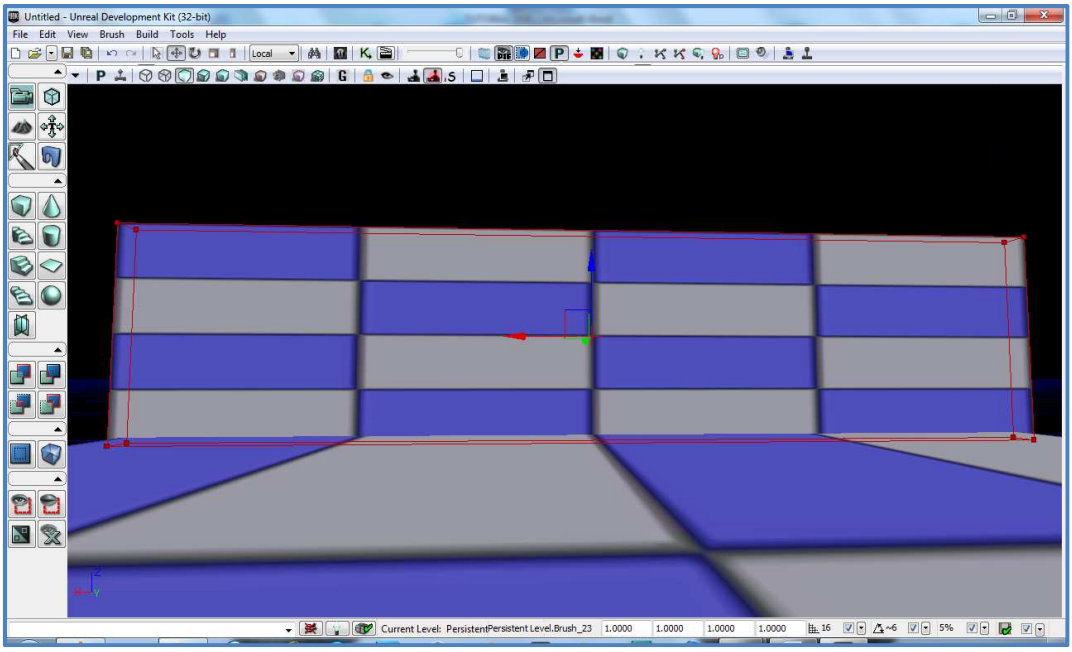


Imagen 224:Tutorial 1-Pared.

3. Ahora se realizará el hueco en la pared para permitir el paso de un lado al otro. Se modificará la forma/modelo con el tamaño del hueco que se quiere realizar.

   Cuando se tenga posicionada en el lugar deseado, se pulsará sobre el icono , *"CSG Subtract"*, y se creará un hueco en la pared.

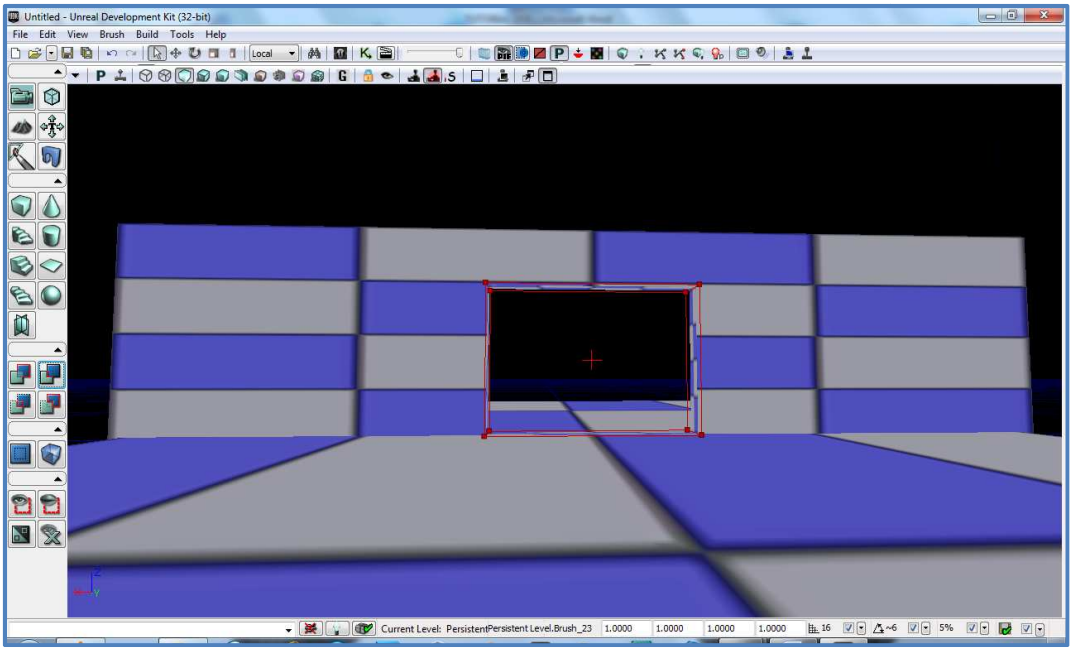


Imagen 225:Tutorial 1-Hueco puerta.

4. Ya se tiene un hueco el cual permite el paso de un lado a otro, pero se va a acoplar un *"mesh"* en dicho hueco que hará las veces de puerta.

Se puede utilizar cualquier *"StaticMesh"* como puerta. Para este tutorial se va a utilizar *"StaticMesh'LT_Doors.SM.Mesh.S_LT_Doors_SM_DoorWay04'"* que viene junto al editor.

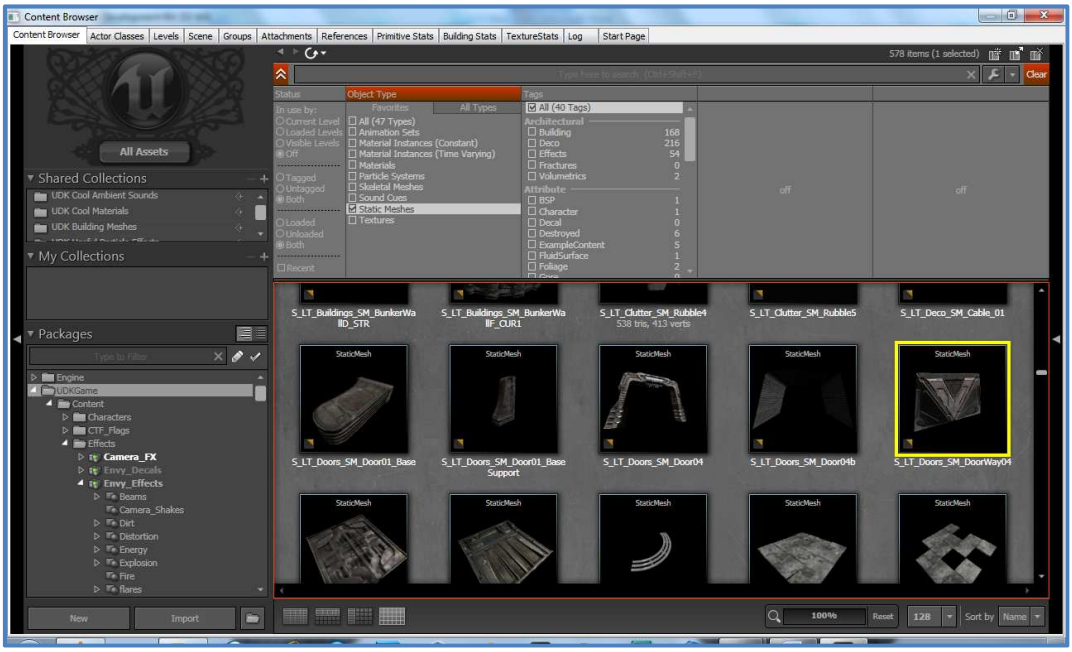


Imagen 226:Tutorial 1-Situación objeto-puerta en "Content Browser".

5. Para poder modificar la posición de este *"mesh"* en ejecución, hay que agregarlo siendo del tipo *"InterpActor"*.

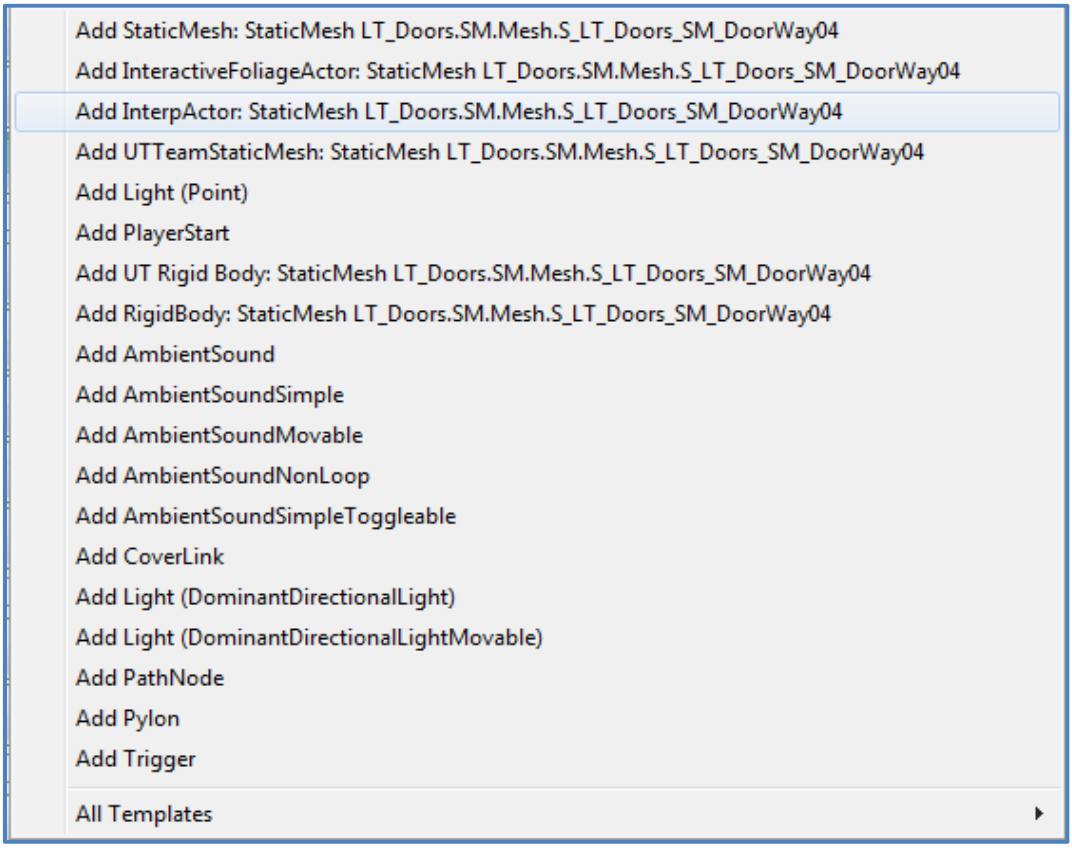


Imagen 227:Tutorial 1-Menú agregar puerta.

El objeto agregado se indicará en color rosa-morado en las vistas para indicar que es del tipo *"InterpActor"*.

6. Se debe modificar el tamaño y posición de este objeto para situarlo justo en el hueco realizado.

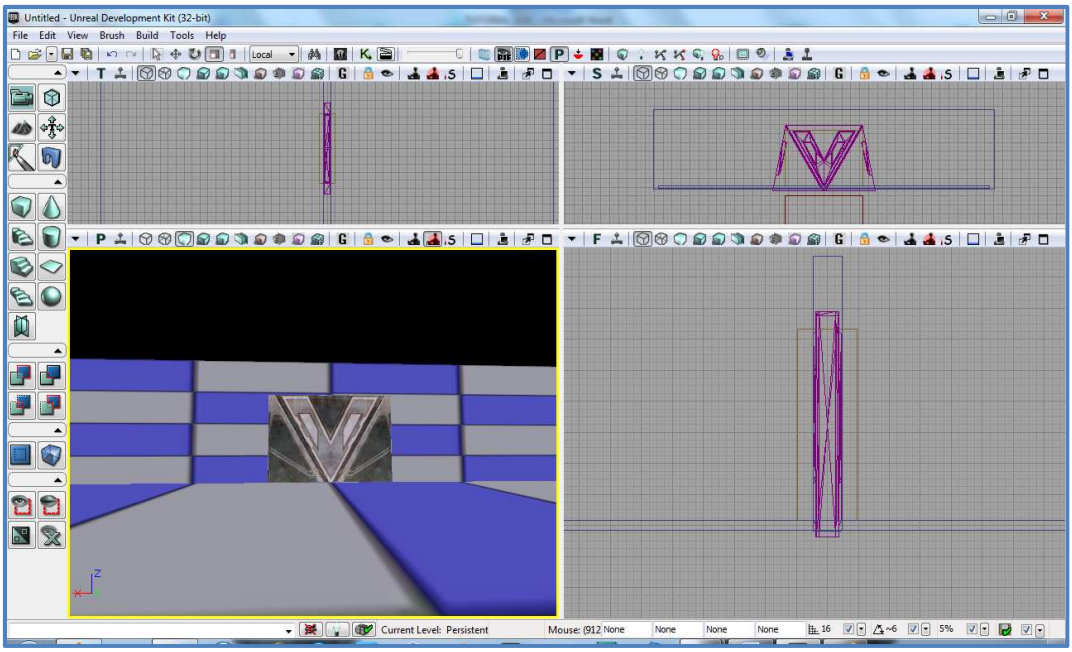


Imagen 228:Tutorial 1-Situación puerta en el hueco.

Por defecto, al agregar un objeto de tipo *"InterpActor"*, éste no es de tipo sólido, pudiendo atravesarlo. Para hacer que este objeto agregado sea sólido hay que acceder a sus propiedades, pulsar la tecla 'F4' del teclado, y en el campo *Collision→Collision Type* poner el tipo *"COLLIDE_BlockAll"*.

7. La dinámica de esta puerta será de apertura por aproximación. El jugador al acercarse a la puerta, ésta se abrirá automáticamente y se cerrará al alejarse.
   El editor proporciona un elemento *"lanzador"* de eventos ó acciones llamado *"Trigger"*:

Imagen 229:Tutorial 1-Lanzador para dinámica de puerta.

Al seleccionar dicho elemento, se mostrará un cilindro de color verde alrededor de él representando la zona en la que se puede interactuar con este pulsador:

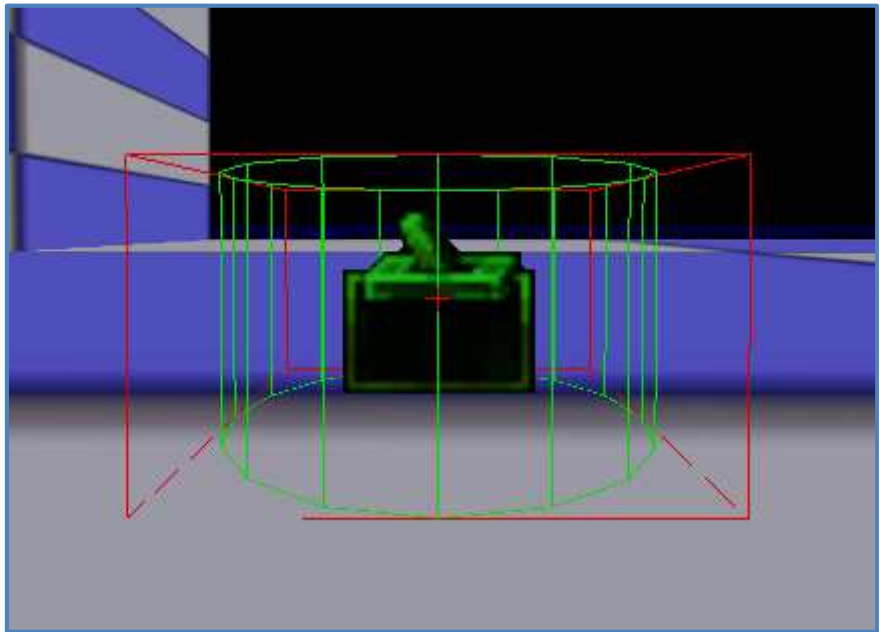


Imagen 230:Tutorial 1-Zona interacción lanzador.

Esta zona puede ser modificada, ampliándola o reduciéndola mediante el modo *"Scaling Mode"* del editor.

8. Se colocará un elemento *"Trigger"* justo en medio de la puerta, modificando la zona para interactuar según se quiera.

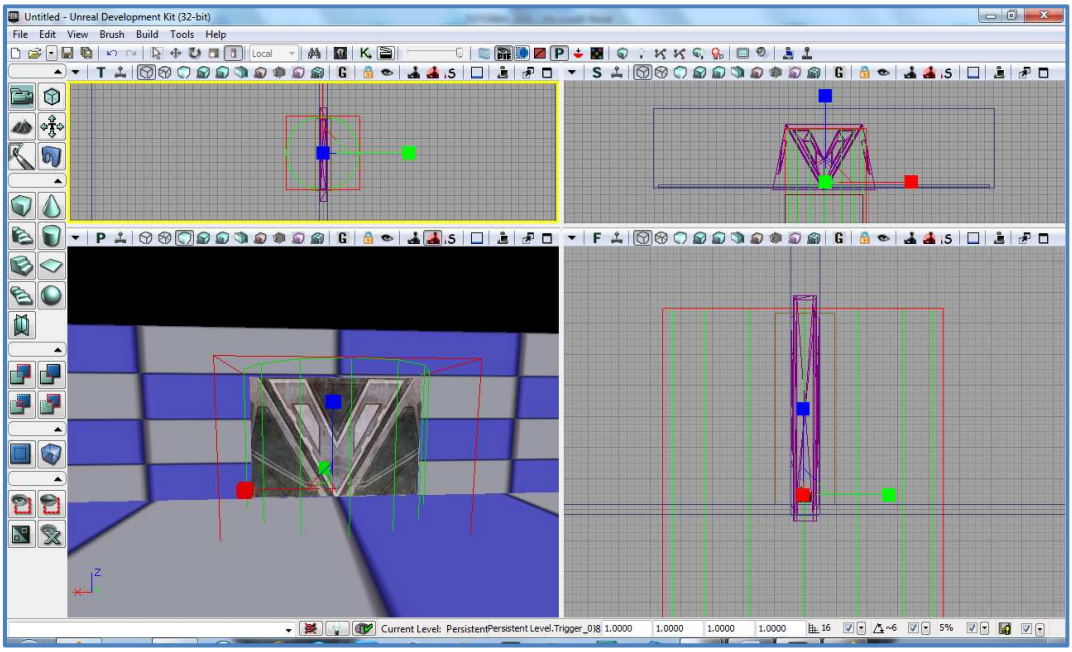


Imagen 231:Tutorial 1-Situación trigger en puerta.

9. Los siguientes pasos se realizarán con la herramienta Unreal Kismet. Se abrirá dicha herramienta mediante el icono de la barra de herramientas superior y con el elemento *"Trigger"* seleccionado.

10. En cualquier zona libre de Unreal Kismet se pulsará con el botón derecho del ratón. En el menú mostrado se elegirá

    New Event using Trigger_x → Touch

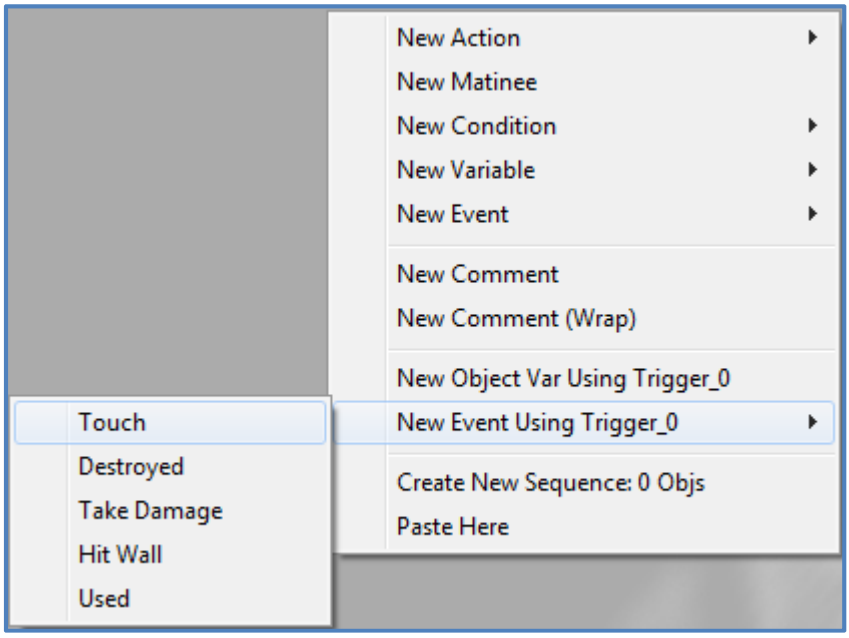


Imagen 232:Tutorial 1-Menú inserción trigger de tipo "Touch".

Mediante esta acción se creará un evento asociado al *"Trigger"* del tipo *"contacto"*, que al hacer contacto el jugador con la zona de interacción, se activará dicho *"lanzador"*, tal y como se muestra en la imagen 233.

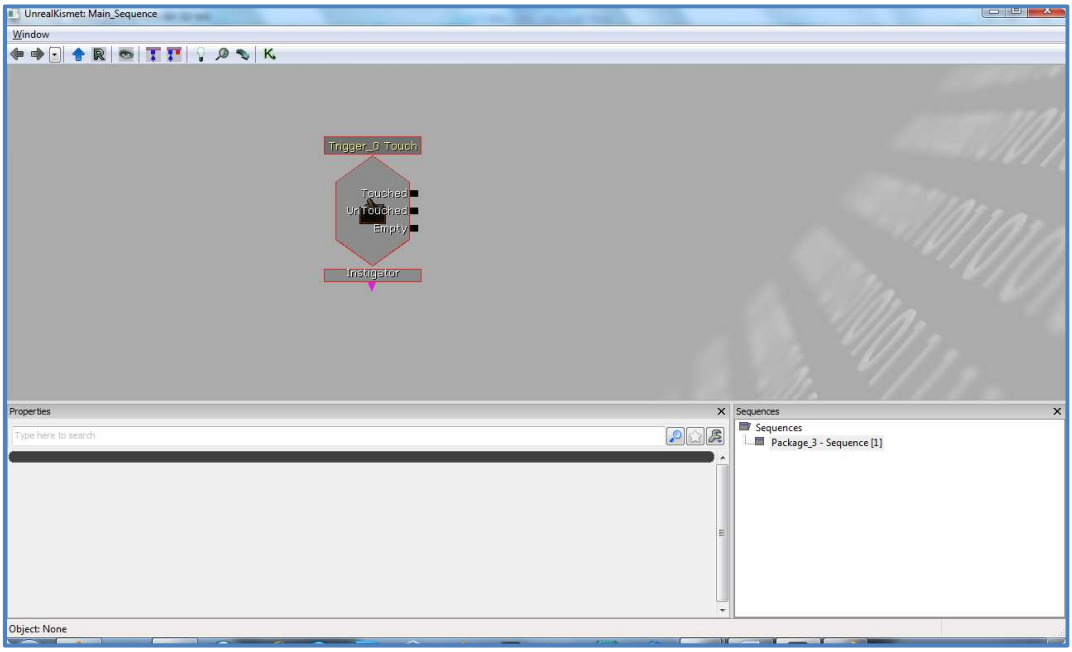


Imagen 233:Tutorial 1-Trigger añadido en herramienta "Kismet".

11. En Unreal Kismet se creará un elemento Matinee que será el encargado de mover el objeto de la puerta.

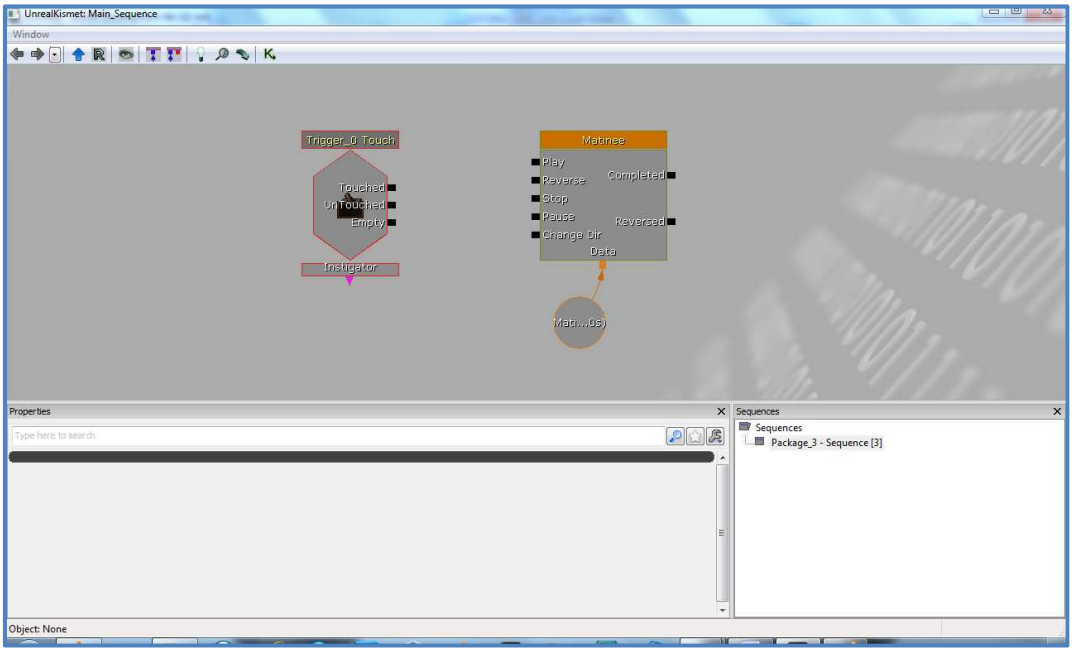


Imagen 234:Tutorial 1-Creación elemento Matinee.

12. Antes de abrir el elemento Matinee, hay que volver al editor y seleccionar el objeto *"InterpActor"* que representa la puerta.

13. Con el objeto de la puerta seleccionado se debe abrir el objeto Matinee haciendo doble pulsación con el botón izquierdo del ratón sobre él.

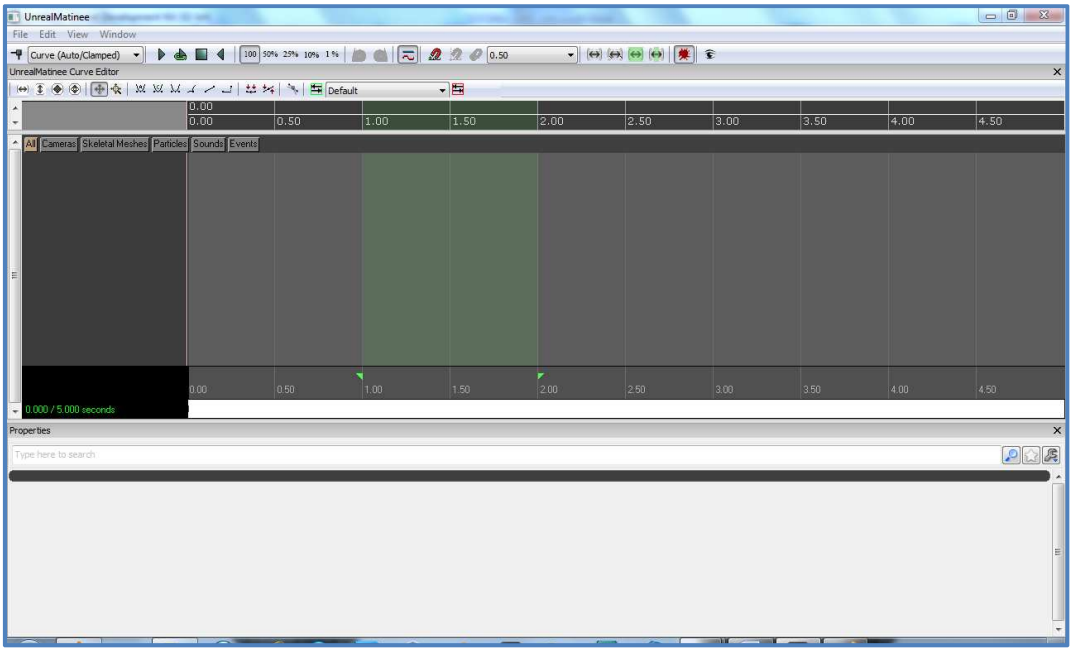


Imagen 235:Tutorial 1-Herramienta "Matinee".

14. Se creará un grupo vacío con el nombre *"Puerta"* mediante el menú contextual mostrado, pulsando con el botón derecho del ratón, y seleccionando *"Add New Empty Group"*.

    En la imagen 236 se muestra la creación del grupo indicado anteriormente.

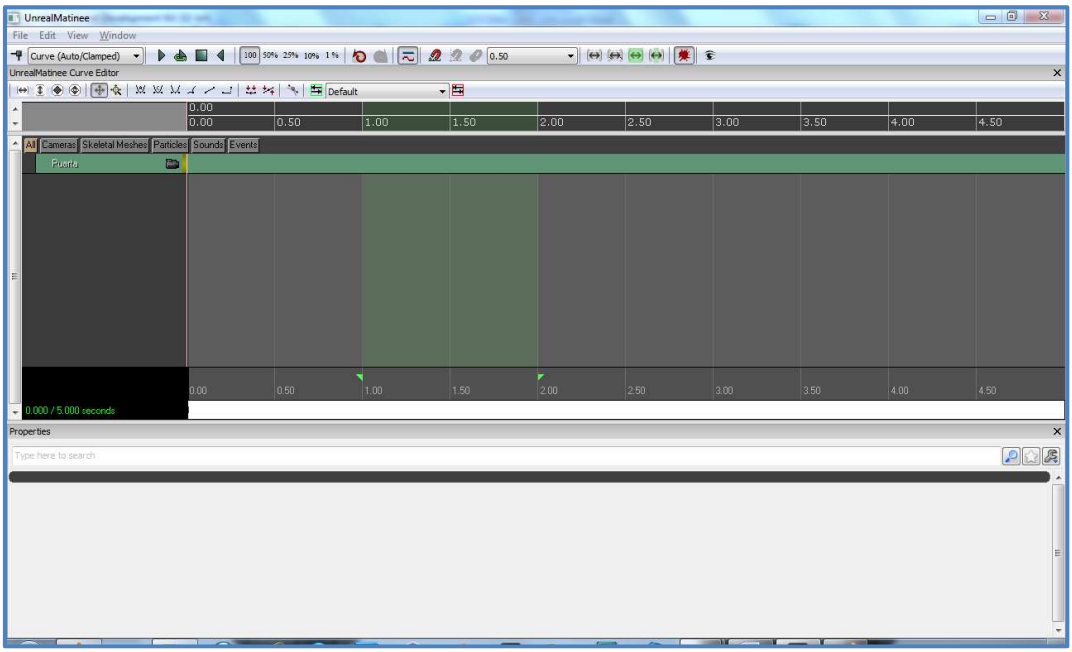


Imagen 236:Tutorial 1-Creación grupo vacío con nombre "puerta".

15. En el nuevo grupo creado se añadirá un *"track"* de movimiento en el cual se grabará el movimiento de la puerta. Para ello se seleccionará el grupo *"Puerta"* y pulsando con el botón derecho del ratón, en el menú contextual mostrado, se seleccionará *"Add New Movement Track"*, imagen 237.

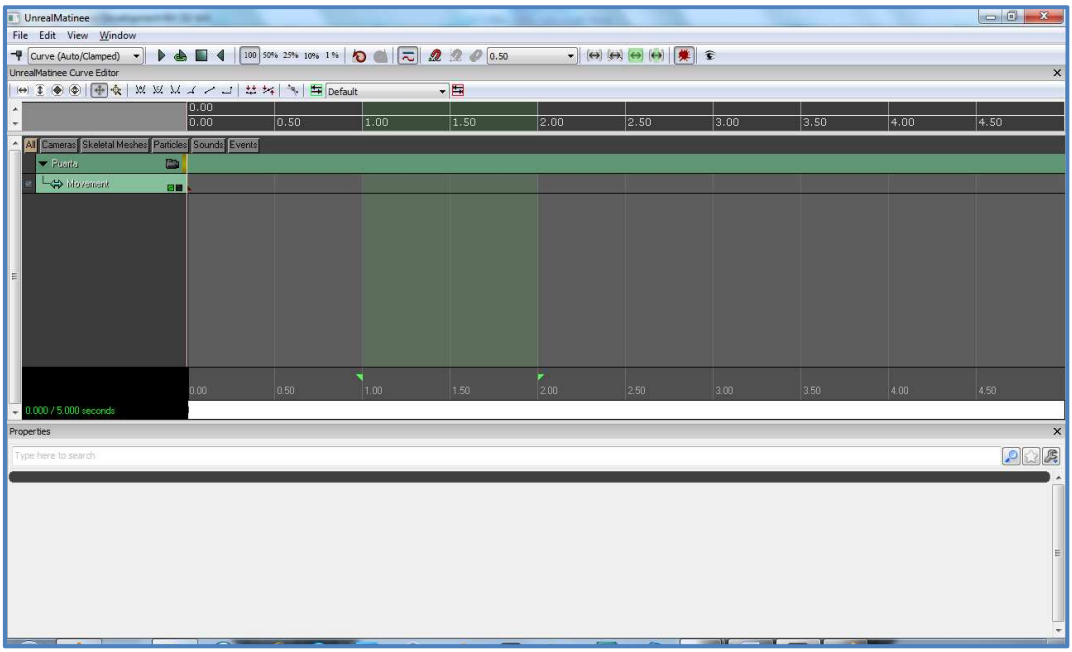


Imagen 237:Tutorial 1-Track de movimiento añadido al grupo.

16. En este track de movimiento es donde se va a guardar el movimiento de la puerta.

Se guardará la posición inicial de la puerta. Con el track seleccionado y el tiempo en 0.0 segundos se pulsará 'Enter'. Si se ha guardado correctamente aparecerá un triángulo con las aristas rojas en el track de movimiento.

Ahora hay que guardar la posición final de la puerta. Se moverá el tiempo a los segundos que se quiere que tarde el movimiento, en este caso a los 5 segundos. Se presionará 'Enter' apareciendo el triángulo de aristas rojas (key) en ese tiempo y sin cerrar Unreal Matinee hay que volver al editor y mover la puerta en la dirección que se quiera.

Una vez guardado el movimiento, se puede comprobar su correcta grabación moviendo la línea de tiempo en el Matinee y comprobando en el editor que la puerta realiza el movimiento deseado.

Cuando todo se ha realizado correctamente, se puede cerrar Unreal Matinee.
En la imagen 238 se pueden observar los puntos de control creados en el track de movimiento.

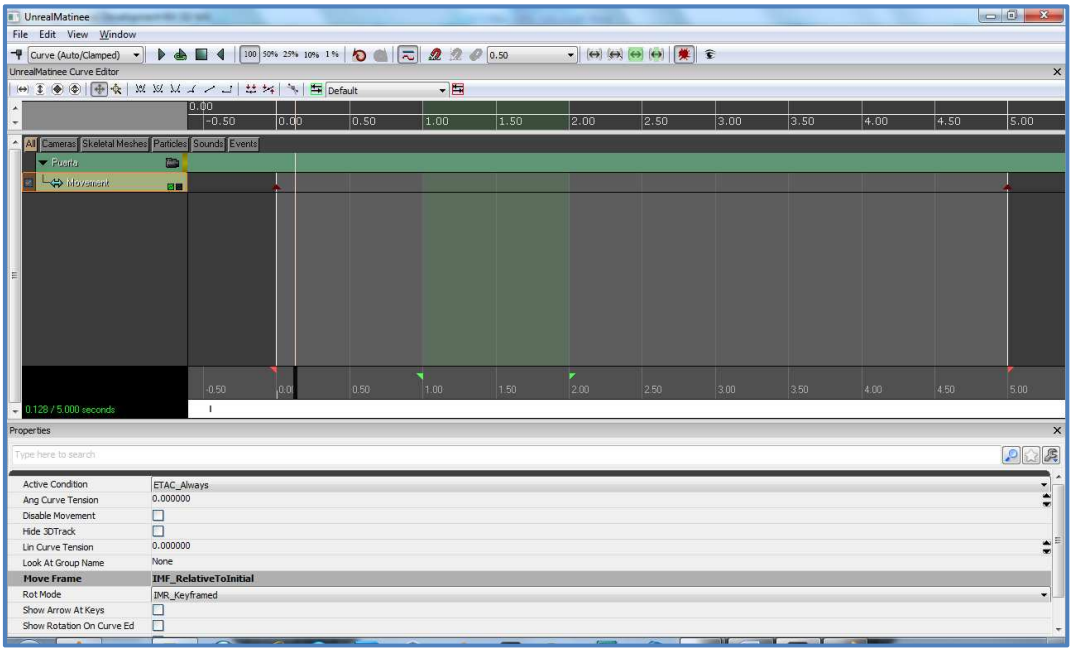


Imagen 238:Tutorial 1-Dinámica de movimiento de puerta guadado.

17. Ahora sólo falta unir el *"Trigger"* con el movimiento que se quiere realizar, guardado en el Matinee, imagen 239.

    - Para que al contactar con el *"Trigger"* se abra la puerta hay que unir la salida *"Touched"* del *"Trigger_x Touch"*, con la entrada *"Play"* del Matinee.
    - Para cuando el jugador deje de hacer contacto, la puerta se cierre, hay que unir la salida *"UnTouched"* del *"Trigger_x Touch"*, con la entrada *"Reverse"* del Matinee.

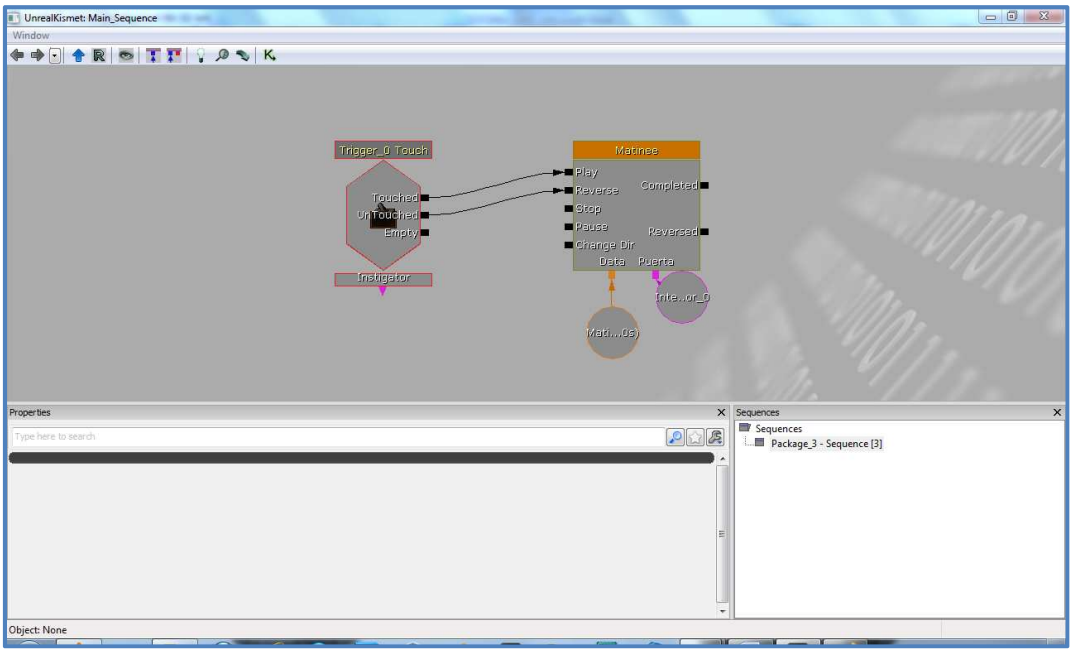


Imagen 239:Tutorial 1-Enlace elemento Trigger y elemento Matinee.

Los elementos *"Trigger"* tienen un campo en las propiedades llamado *"Max Trigger count"* en el cual se indica las veces que se quiere que se active. Por defecto viene el valor de 1, con lo que sólo se activa una sóla vez. Si se quiere que se active indefinidamente hay que poner el valor de 0 en ese atributo:

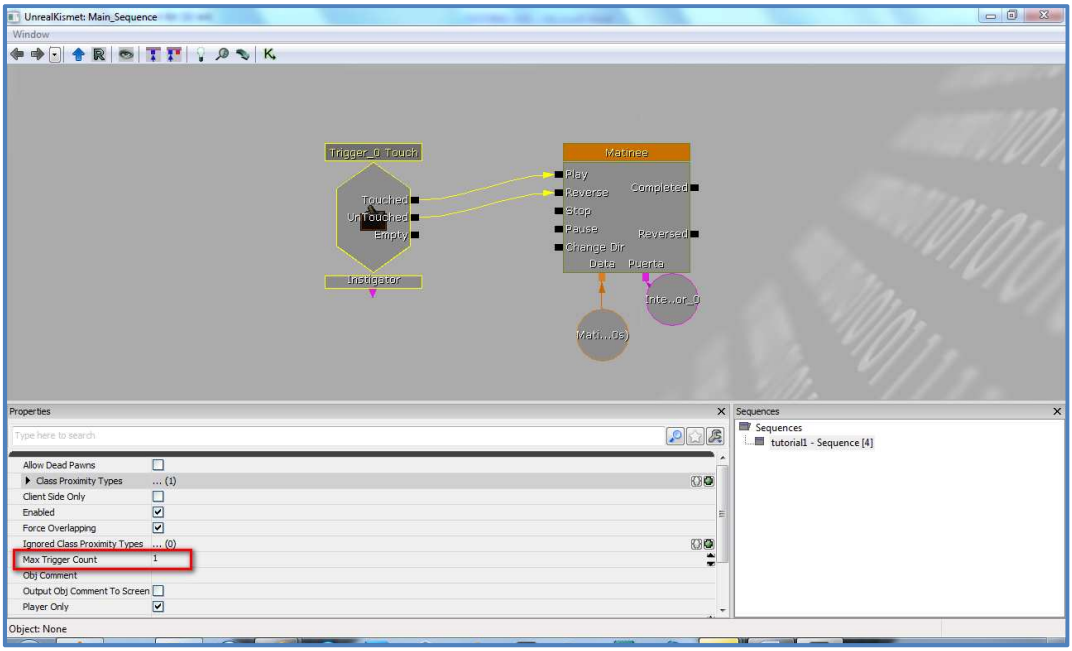


Imagen 240:Tutorial 1-Etiqueta "Max Trigger Count" de elemento Trigger.

Otro tipo de puerta que se puede implementar es realizar la apertura mediante un interruptor. Para este tipo de apertura el único punto que hay que modificar del tutorial expuesto anteriormente es el número 10:

En cualquier zona libre de Unreal Kismet se pulsará con el botón derecho del ratón. En el menú mostrado se elegirá:

New Event using Trigger_x → Used

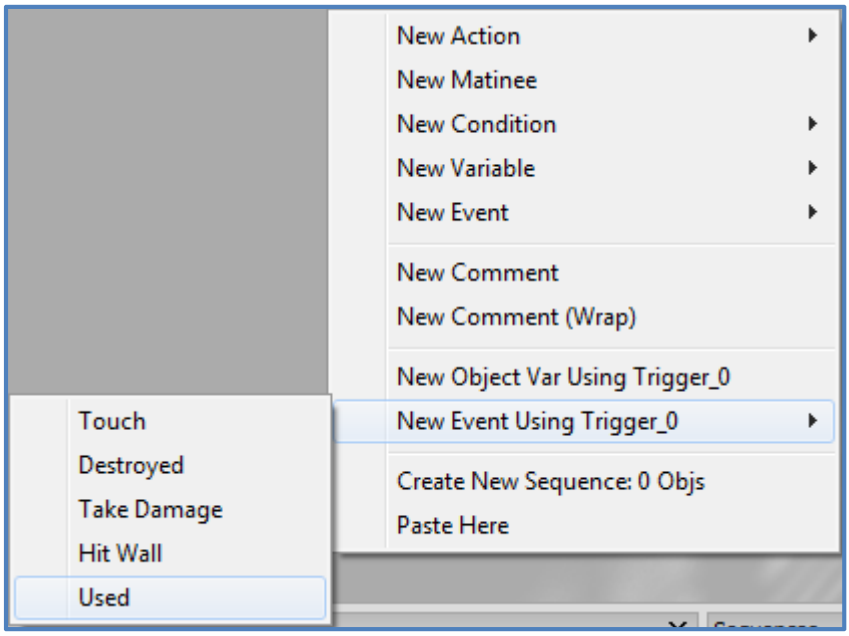


Imagen 241:Tutorial 1-Menú lanzador de tipo "Used".

Mediante esta acción se creará un evento asociado al *"Trigger"* del tipo *"uso"*, que al hacer uso en la zona de interacción, se activará dicho *"lanzador"*:

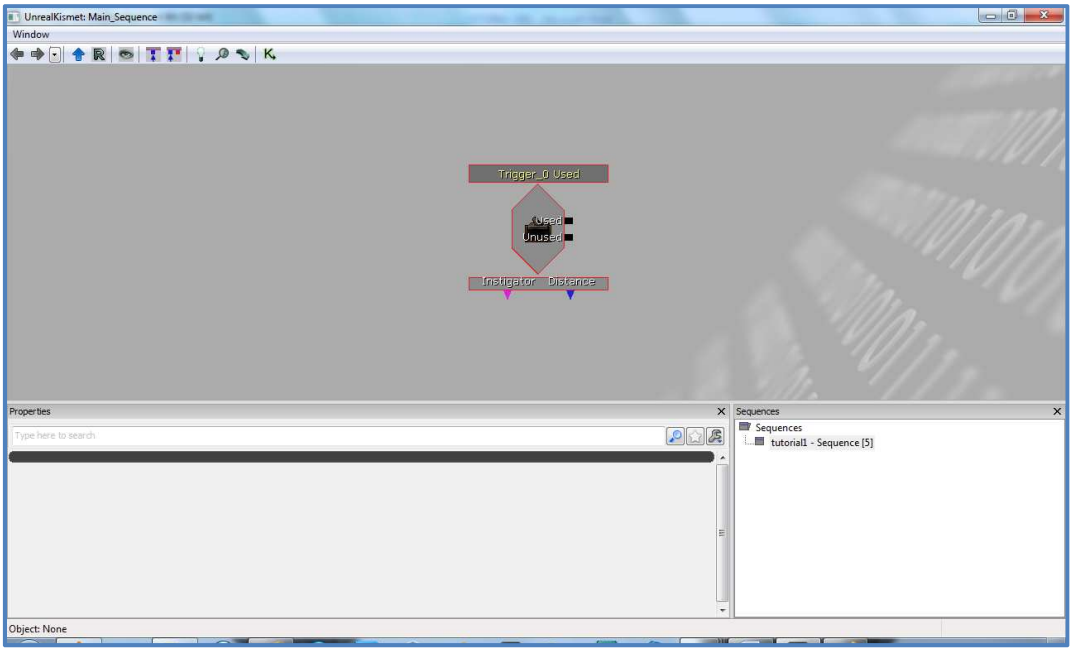


Imagen 242:Tutorial 1-Trigger añadido en herramienta "Kismet".

Todos los restantes puntos del tutorial son iguales sólo que para unir ahora el *"Trigger"* con el Matinee se realiza: la salida *"Used"* del *"Trigger"* se une con la entrada *"Play"* del Matinee.

El *"Trigger"* de uso tiene una particularidad en sus propiedades. Posee una propiedad llamada *"Aim To Interact"* que en el caso que esté activa, es necesario mirar al punto del *"Trigger"* mientras se usa, pulsando tecla 'E' del teclado, para activarlo. Si

se desactiva, sólo con estar en la zona del *"Trigger"* y pulsar la tecla 'E' del teclado (hacer uso), será lanzado. En la imagen 243 se muestra la localización de esta propiedad.

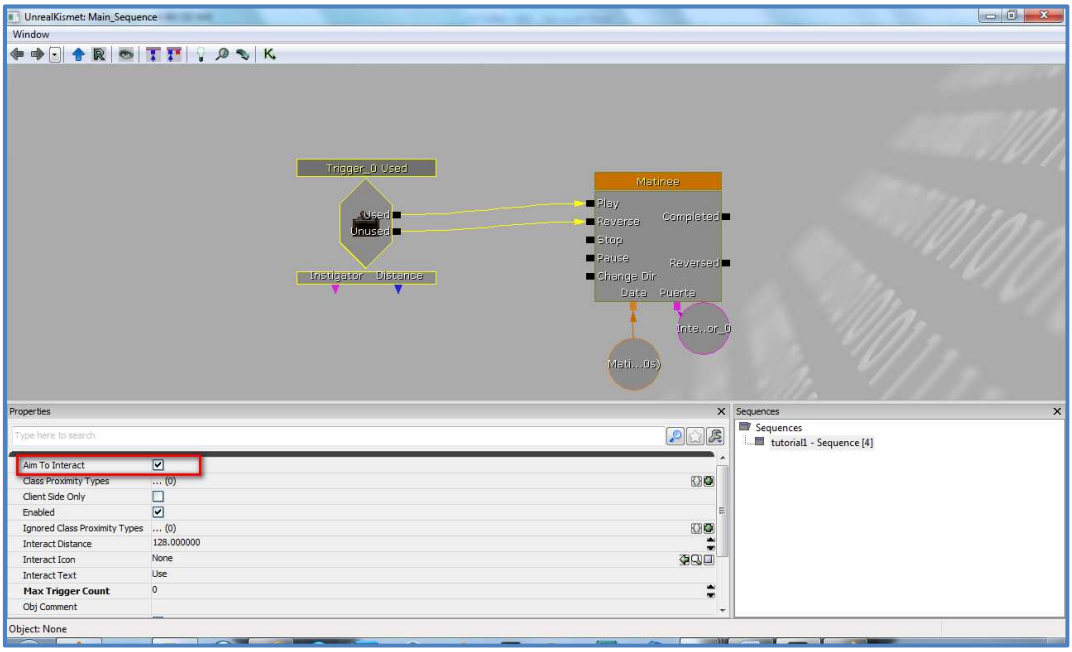


Imagen 243:Tutorial 1-Etiqueta "Aim To Interact" de trigger tipo "Used".

En este tutorial se han mostrado las dos formas más comunes de tener puertas en un nivel. Se ha mostrado la parte básica, pudiendo realizar otras implementaciones del movimiento de la puerta, como:

- Una vez que se realice el movimiento de apertura, la puerta vuelva a cerrarse. Para ello sólo hay que unir la salida *"Completed"* del Matinee con la entrada *"Reverse"* del mismo elemento, tal y como se muestra en la imagen 244.

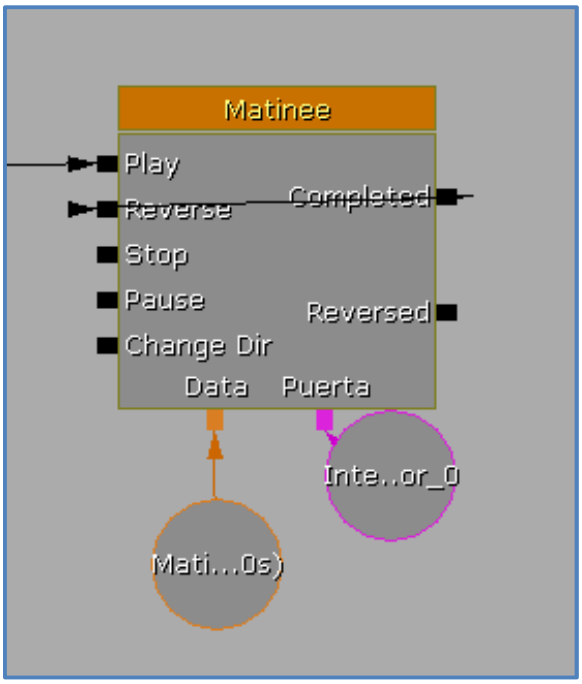


Imagen 244:Tutorial 1-Apertura y cierre de puerta automáticamente.

- Una vez activado el *"Trigger"*, la puerta esté continuamente abriendo y cerrando.

  Hay que unir la salida *"Completed"* del Matinee con la entrada *"Reverse"* del mismo elemento y la salida *"Reversed"* con la entrada *"Play"*, mostrado en la imagen 245.

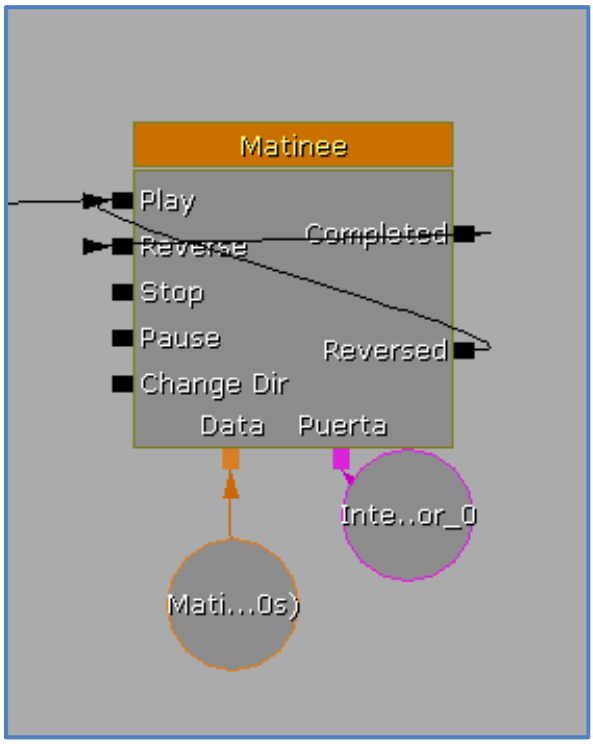


Imagen 245:Tutorial 1-Movimiento contínuo de apertura y cierre de puerta.

8.2. Tutorial 2: Apagar/encender luces con pulsador.

Uno de los elementos indispensables en un nivel es la iluminación. En el capítulo anterior se han expuesto todos los tipos de iluminación que se pueden acoplar al nivel, y en este tutorial se mostrará la forma de poder encender ó apagar las luces que poseen la propiedad de *"Toggleable"*. [45]

Pasos a seguir:

1. Se acoplarán al nivel las luces desde el *"Actor Classes"*, abriéndolo mediante el icono de la barra de herramientas superior del editor y seleccionando la pestaña con el mismo nombre.

   Para ello, dentro del *"Actor Classes"* se navegará en la jerarquía de directorios hasta encontrar las luces que se quieren acoplar:

   Light → Point Light → PointLightToggleable

   En la imagen 246 se muestra la situación de estos elementos.

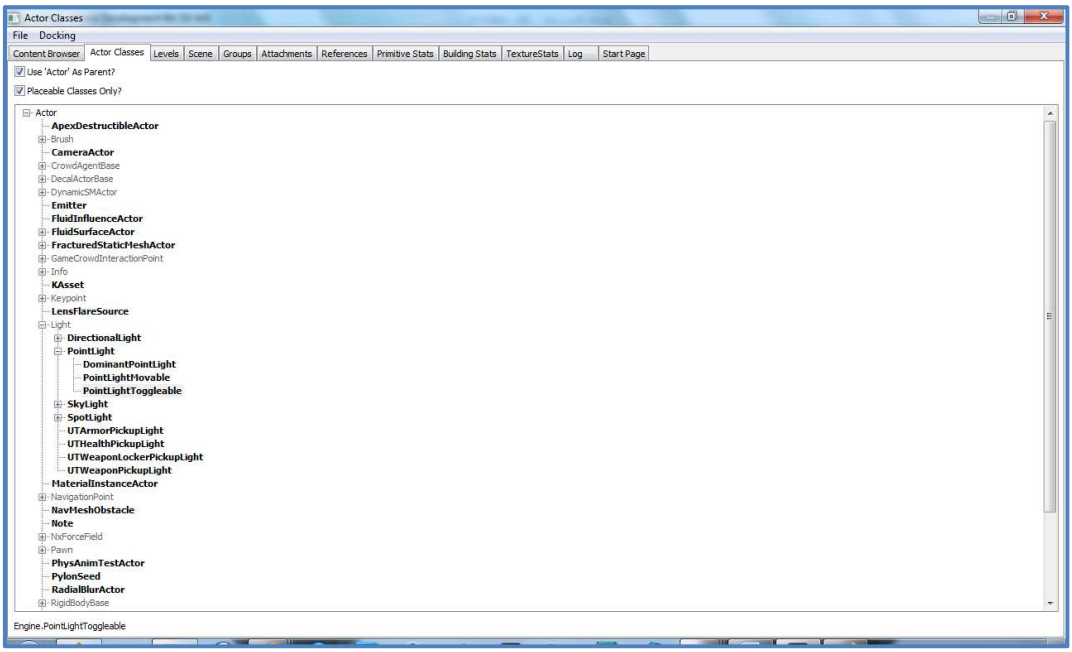


Imagen 246:Tutorial 2-Localización "PointLightToggleable" en "Actor Classes".

En el caso de este tutorial se ha elegido el tipo *"PointLight"* pero se puede seleccionar cualquier tipo de luz, sólo debe cumplir que posea la propiedad *"Toggleable"*.

2. Se colocará una luz en cada una de las esquinas del plano del suelo, más una en el centro.

   Para comprobar luego el funcionamiento más claramente, las luces serán de distinto color, tal y como se muestra en la imagen 247.

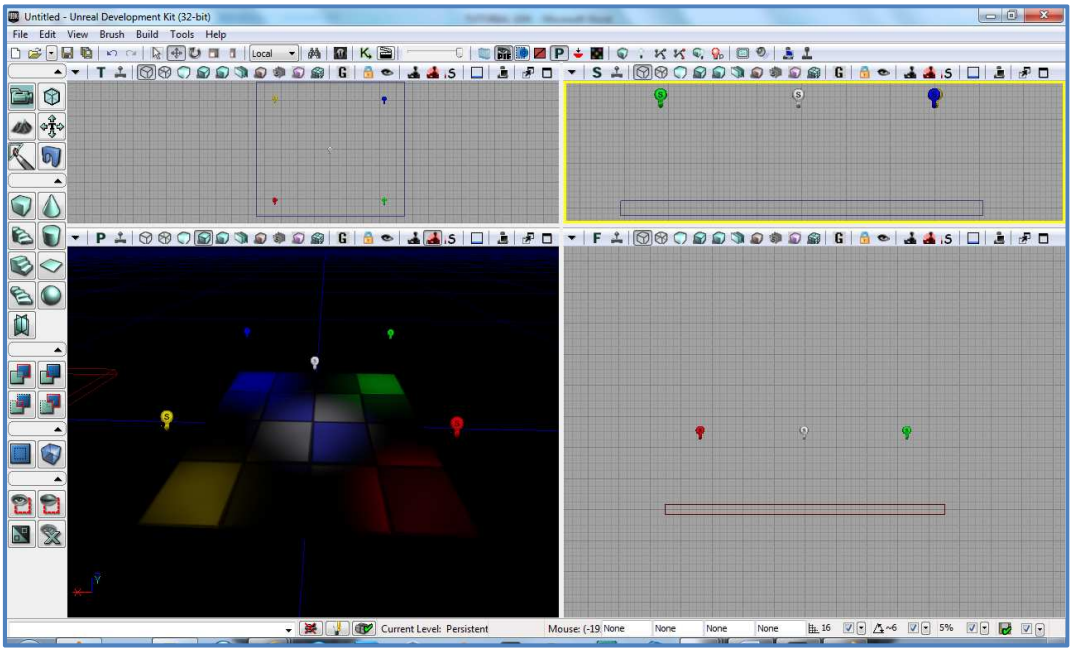


Imagen 247:Tutorial 2-Situación luces en nivel.

NOTA: No es necesario acoplar una a una todas las luces desde el *"Actor Classes"*. Se acopla una de ellas y las demás son copiadas. Para obtener un objeto

copia de otro hay que pulsar la tecla 'Alt' del teclado a la vez que se realiza un traslado del objeto. Mediante esta operación el objeto original permanecerá en su lugar, y se habrá creado una copia.

3. Para lanzar el evento de apagar ó encender las luces es necesario un elemento llamado *"Trigger"* que al interactuar con él lanzará la acción a realizar.
   Cada una de las luces acopladas serán independientes, con lo que habrá que acoplar un elemento *"Trigger"* por cada una de la luces. Además se situarán cada uno en la misma vertical que la luz:

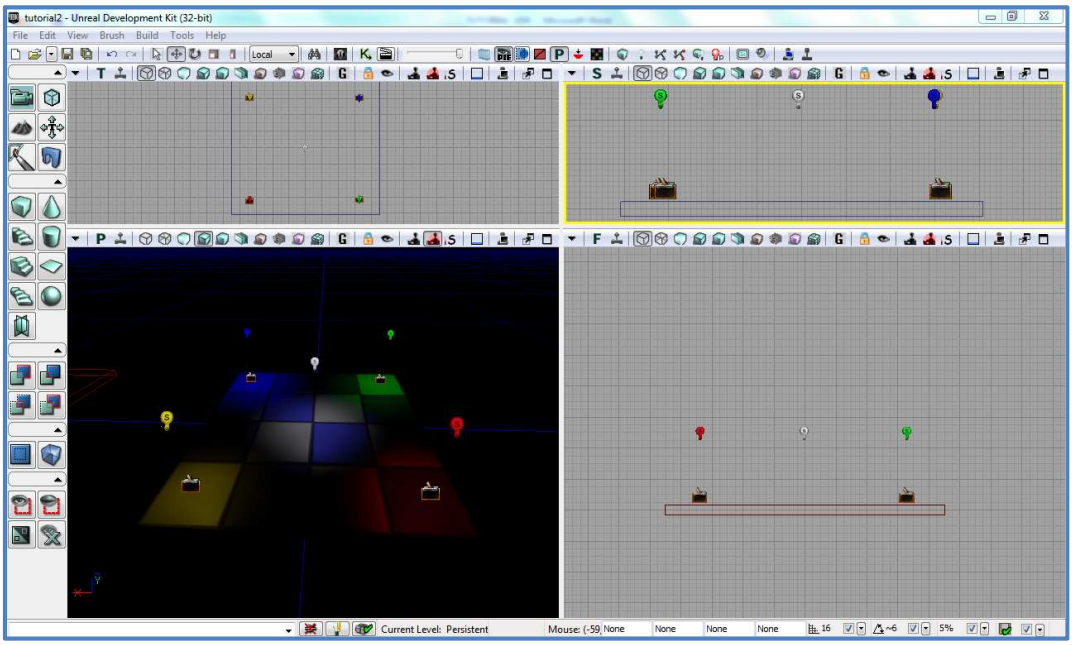


Imagen 248:Tutorial 2-Situación triggers junto a luces.

4. En la herramienta Unreal Kismet se debe crear un evento con cada uno de los *"Trigger"* acoplados.
   Se selecionarán todos los *"Trigger"* a la vez. Para ello se irá pinchando en cada uno de ellos manteniendo pulsada la tecla 'Ctrl' del teclado.

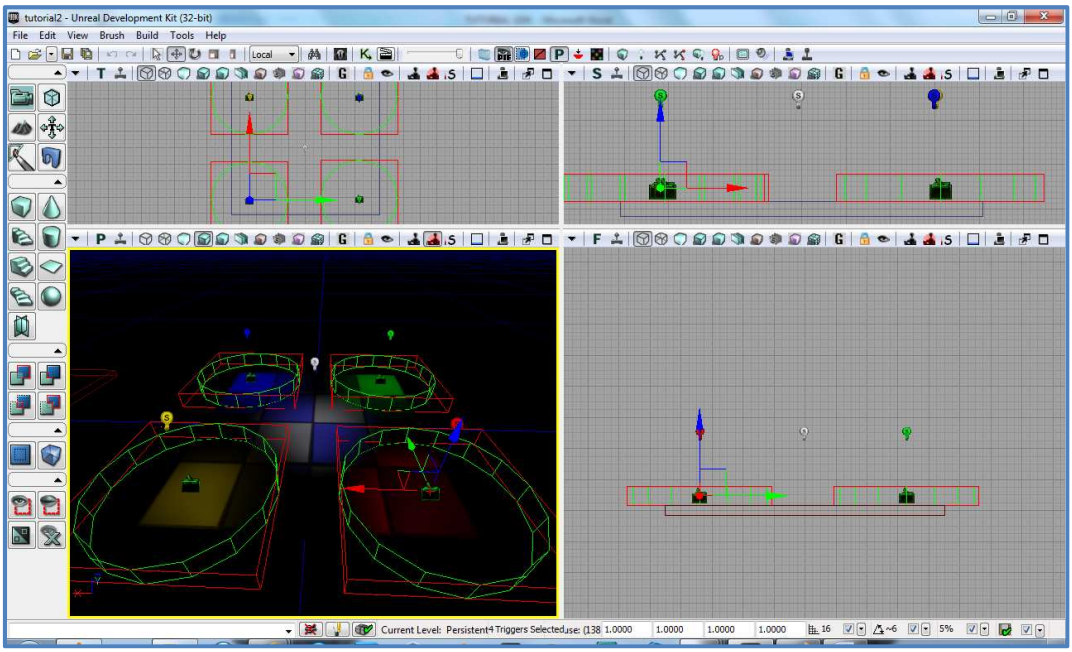


Imagen 249:Tutorial 2-Zonas de interacción de cada trigger.

Se abrirá la herramienta Unreal Kismet mediante el icono de la barra de herramientas superior con todos los *"Trigger"* seleccionados.
En cualquier zona despejada se pulsa con el botón derecho del ratón y en el menú desplegado seleccionar:

New Events using Trigger_x, … → Touch

En este caso se ha seleccionado el tipo *"contacto"* en el *"Trigger"*, que será lanzado cuando el jugador haga contacto con la zona de acción (cilindro).

Los eventos de contacto serán creados con cada uno de los *"Trigger"* tal y como se muestra en la imagen 250.

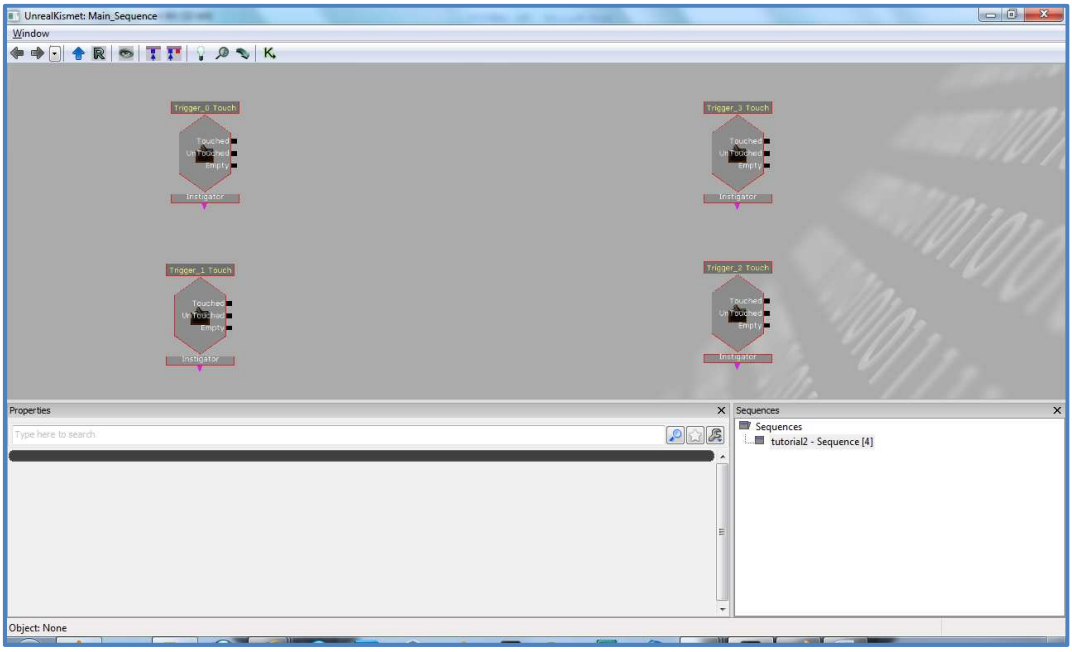


Imagen 250:Tutorial 2-Creación eventos en "Kismet" partiendo de los trigger.

En cada uno de los elementos, hay que modificar la etiqueta *"Max Trigger Count"* para que se pueda utilizar indefinidamente, no sólo una vez, como viene por defecto. El valor que hay que poner es 0 en dicha etiqueta.

5. A cada uno de los eventos creados hay que asociarle el elemento encargado de apagar/encender, el que tiene acceso a la propiedad *"Toggleable"* de los objetos. Este elemento es llamado *"Toggle"*, pudiéndose insertar de la siguiente manera:

   Pinchar con el botón derecho del ratón →New Action → Toggle → Toggle.

   En este elemento *"Toggle"* se tiene la entrada de encerder, *"Turn On"*, y la entrada de apagar, *"Turn Off"*. Por lo tanto, se unirá la salida *"Touched"* del *"Trigger"* con la entrada *"Turn On"* del *"Toggle"* y la salida *"UnTouched"* del *"Trigger"* con la entrada *"Turn Off"* del *"Toggle"*. Así se encenderán las luces cuando estemos cerca de ellas y se apagarán cuando nos alejemos. Ver imagen 251.

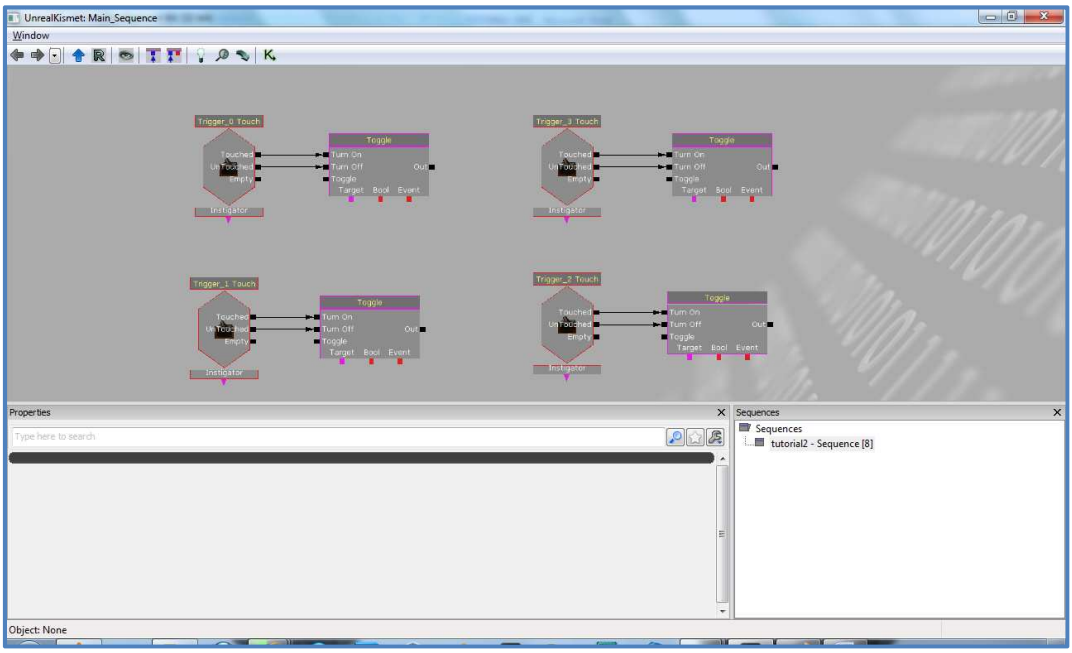


Imagen 251:Tutorial 2-Creación elementos "Toggle" de cada trigger.

6. En la herramienta Unreal Kismet se necesita tener unas variables que representen a las luces del nivel, para poder interactuar con ellas.
   Se seleccionan todas las luces en el editor, manteniendo la tecla 'Ctrl' del teclado para seleccionar más de un objeto.

   Con todas las luces seleccionadas, se vuelve a la herramienta Kismet mediante el icono de la barra de herramientas superior.

   En cualquier zona despejada se pulsa el botón derecho del ratón y en el menú desplegado se selecciona

   New Object vars using PointLightToggleable_x,…

   Aparecerán cinco variables que representan las luces acopladas al nivel, mostradas en la imagen 252.

Imagen 252:Tutorial 2-Variables de las luces.

7. A los elementos *"Toggle"* hay que indicarles las variables las cuales hay que modificar.

   Las variables a modificar hay que acoplarlas en la entrada *"Target"* del elemento *"Toggle"*.

   Cada uno de los elementos *"Trigger"* hay que asociarlo a su correspondiente luz. En el editor, se puede ver el nombre de cada objeto gracias al menú de características, tecla 'F4' del teclado, en la parte superior. Sabiendo el nombre de cada *"Trigger"* y de cada luz, se asocian en la herramienta Kismet de la manera mostrada en la imagen 253.

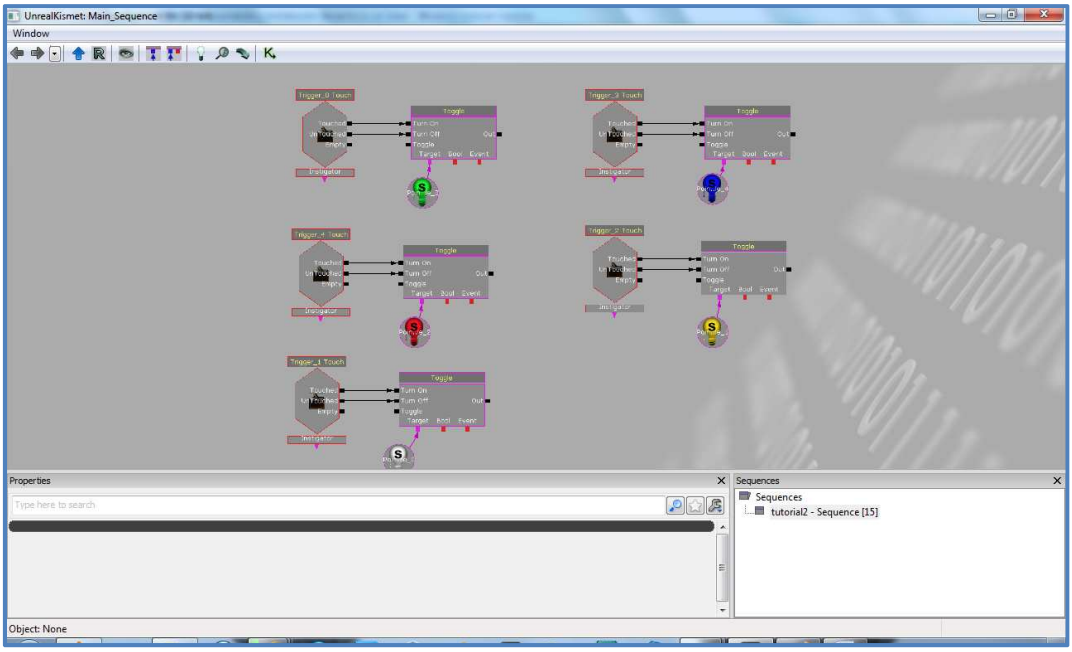


Imagen 253:Tutorial 2-Asociación variables luces con "Toggle".

En el nivel creado, según se vaya avanzando, las luces irán apagándose y encendiéndose según el jugador esté próximo a ellas.

En el tutorial mostrado, se han utilizado *"Triggers"* del tipo contacto, pero se podrían utilizar otros tipos, como de uso. El *"Trigger"* se acoplaría en cualquier objeto usado como interruptor y al crear el evento en la herramienta Unreal Kismet se seleccionaría el tipo *"Used"*.

8.3. Tutorial 3: Acoplar linterna al personaje principal.

Al personaje principal pueden serle acoplados distintos objetos, los cuales acompañarán al personaje en su movimiento. [46]

En este tutorial se va a presentar la forma de acoplar una linterna al personaje principal del juego.

Pasos a seguir:

1. Es necesario tener acoplado en el nivel el elemento *"Player StartUp"*, imagen 254, el cual servirá de referencia para acoplar la luz que servirá de linterna.

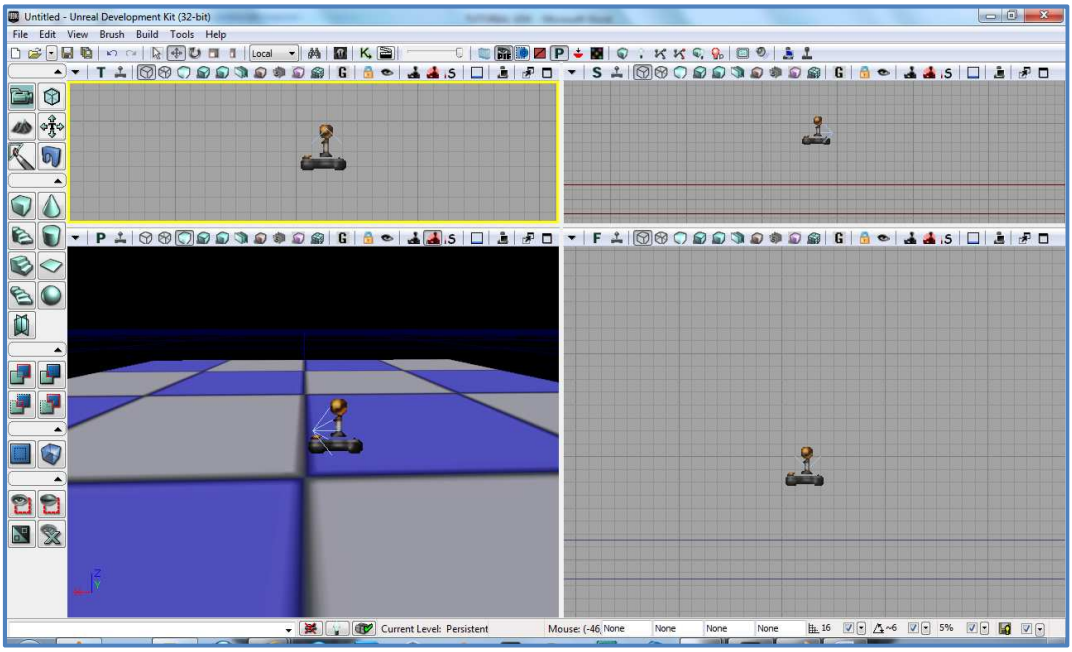


Imagen 254:Tutorial 3-Situación "PlayerStart" en nivel.

2. El tipo de luz que servirá como linterna es el *"SpotLightMoveable"*.
   Este tipo de luz se encuentra en el *"Actor Classes"* pudiéndose abrir pulsando el icono de la barra de herramientas superior, en la correspondiente pestaña:

   Light → SpotLight → SpotLightMoveable

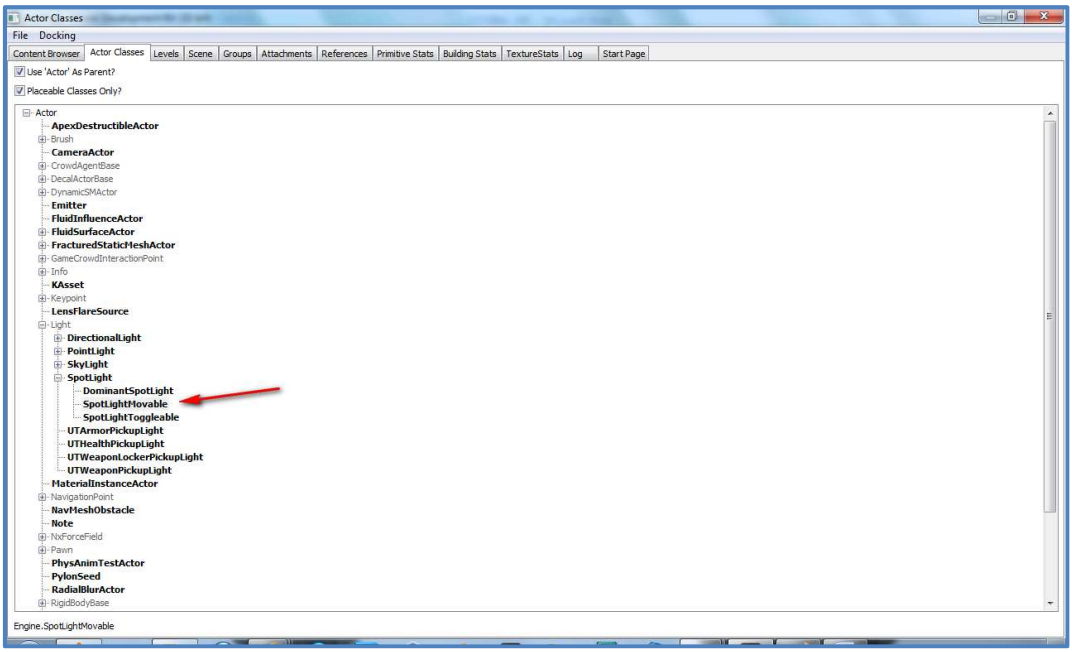


Imagen 255:Tutorial 3-Localización "SpotLightMoveable" en "Actor Classes".

3. El elemento de la iluminación debe acoplarse al nivel cerca del objeto *"PlayerStart"*, en la posición donde se quiera asociar.

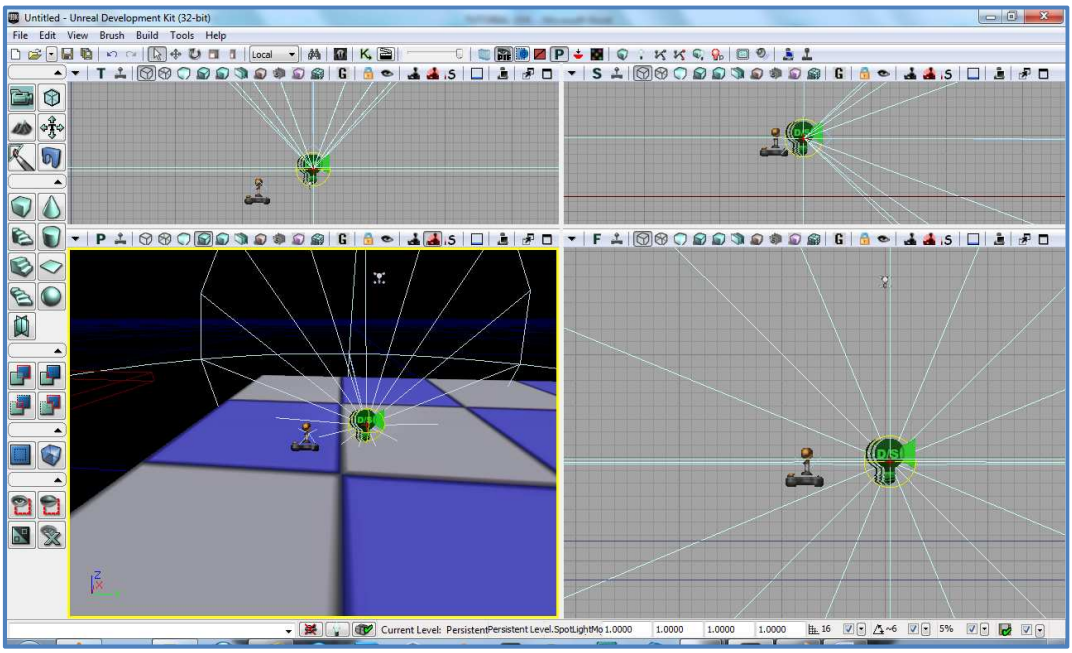


Imagen 256:Tutorial 3-Situación punto de luz junto "PlayerStart".

4. Mediante las características de la luz, pulsando tecla 'F4' del teclado, se puede modificar el radio del foco para iluminar lo deseado, al igual que la longitud de este foco.

   También se podría modificar el ángulo de iluminación del foco para que apunte en la misma dirección que la vista del personaje principal.

5. Abrir la herramienta Unreal Kismet mediante el icono de la barra de herramientas superior.

Se debe insertar el elemento que represente la carga del personaje principal. Este elemento es el llamado *"Player Spanwed"* y se puede insertar mediante:

Botón derecho del ratón → New Event → Player → Player Spanwed.

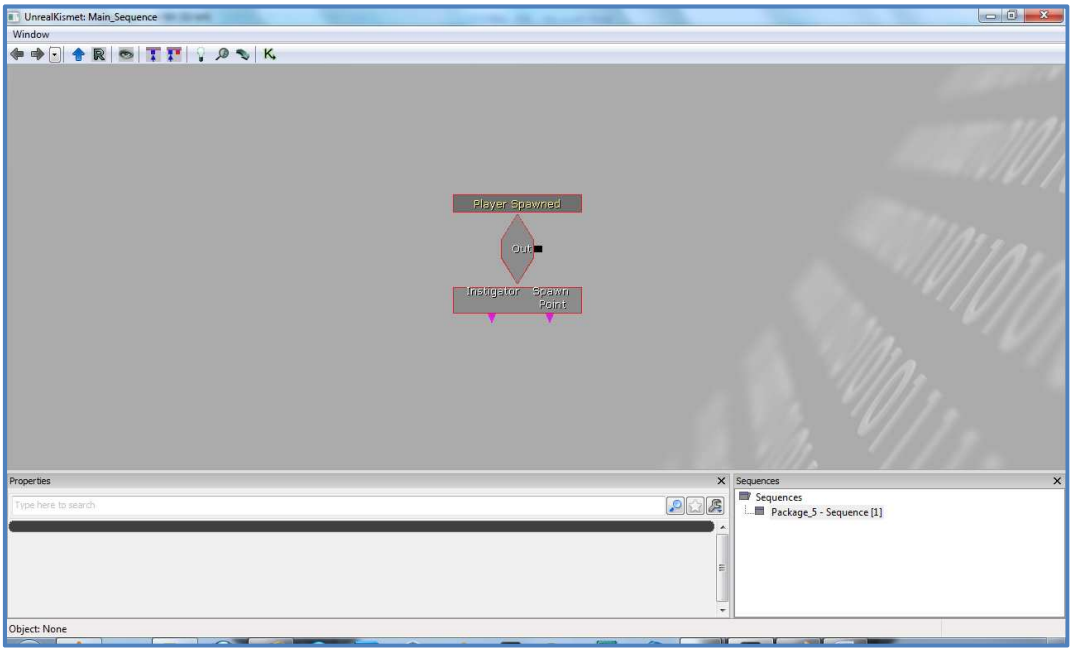


Imagen 257:Tutorial 3-Creación elemento "Player Spanwed" en "Kismet".

6. Es necesario el elemento que acople cualquier objeto al *"Actor"* del personaje principal.
   Este elemento es llamado *"Attach to Actor"* pudiéndose insertar mediante:

   Botón derecho del ratón → New Action → Actor → Attack to Actor.

   Se unirá el evento insertado en el punto anterior con este nuevo elemento, tal y como se muestra en la imagen 258.

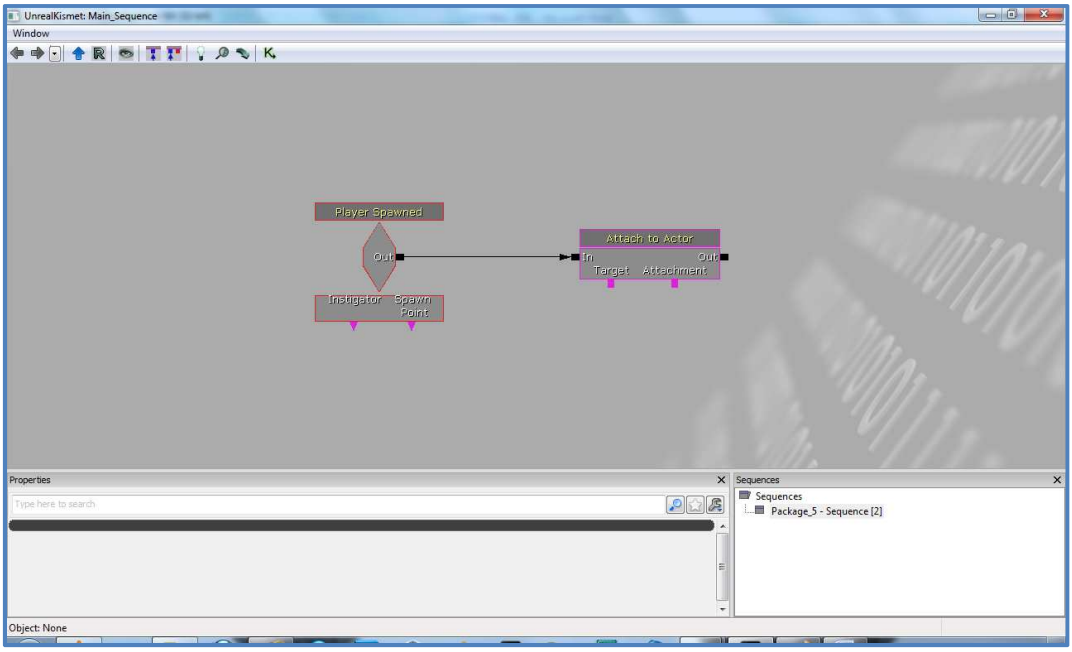


Imagen 258:Tutorial 3-Creación elemento "Attach to Actor".

7. Hay que tener una variable que represente el punto de luz que se quiere acoplar. En el editor hay que seleccionar el objeto de la luz, y con el objeto seleccionado volver a la herramienta Unreal Kismet. Presionar el botón derecho del ratón en cualquier zona despejada y seleccionar *"NewObject var using SpotLightMoveable_x"*. Aparecerá la variable que representa la luz del editor.

   Esta variable hay que acoplarla a la entrada *"Attachment"* del elemento *"Attack to Actor"*, tal y como se muestra en la imagen 259.

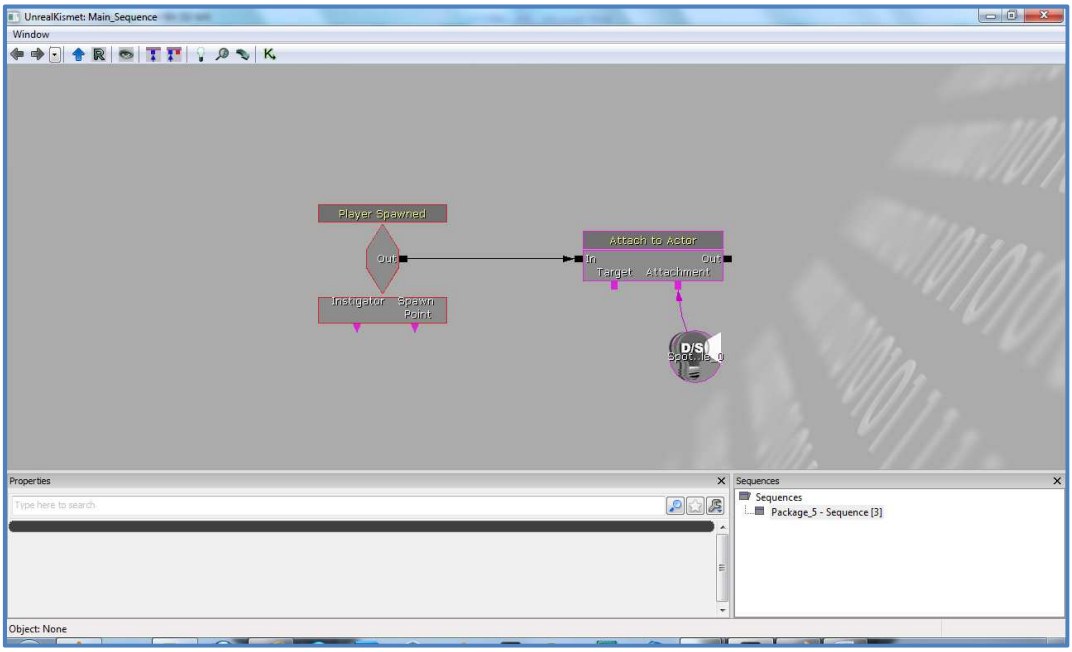


Imagen 259:Tutorial 3-Asociación variable de la luz.

8. En el elemento *"Attack to Actor"*, hay que indicar el Actor al que hay que acoplar el objeto. En el caso del tutorial se va a acoplar al personaje principal con lo que es necesaria una variable que represente a este Actor. Para obtener esta variable:

   Botón derecho del ratón → New Variable → Player → Player

   Esta variable tiene que representar sólo al personaje principal, con lo que no hay que seleccionar la etiqueta *"All Players"* de las propiedades.

   Esta variable hay que unirla con la entrada *"Target"* del elemento *"Attack to Actor"* para indicar al actor del personaje principal, imagen 260.

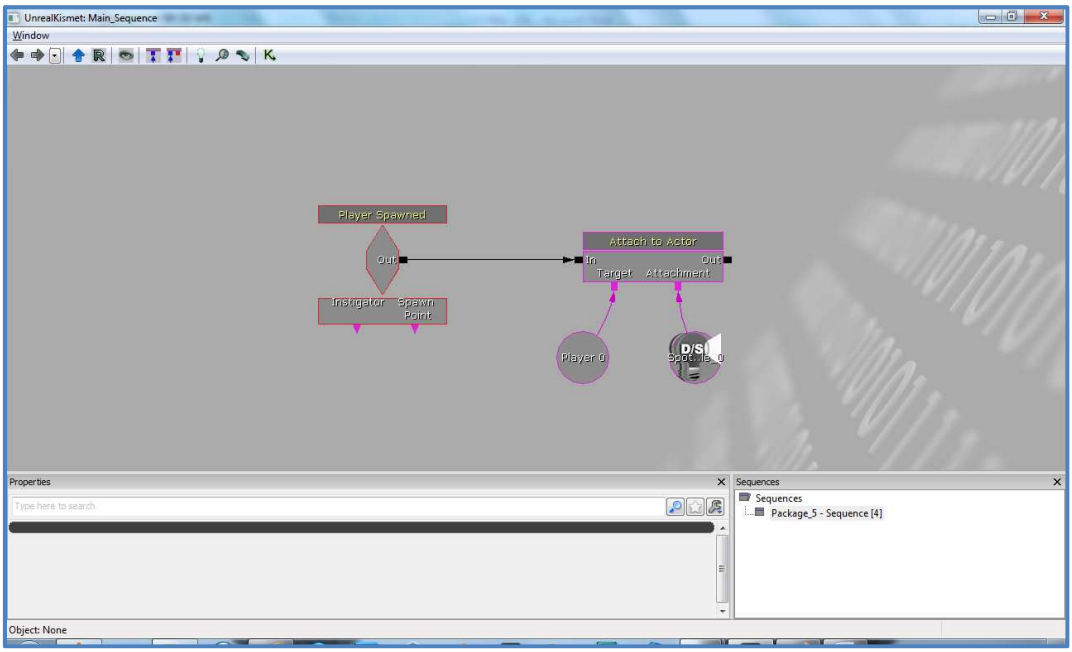


Imagen 260:Tutorial 3-Asociación variable del jugador.

9. En las propiedades del elemento *"Attack to Actor"* se tienen varias etiquetas en las cuales se puede indicar la posición relativa del objeto a acoplar al igual que la rotación:

   - Posición relativa: Se puede indicar la posición relativa que se quiere dar al objeto a acoplar a partir del punto de aparición del personaje principal. Los valores de dicha posición relativa se deben indicar en cada uno de los ejes. La pestaña donde se indican estos valores es llamada *"Relative Offset"*.

     También para utilizar estos valores hay que activar la etiqueta *"Use Relative Offset"*.

     Estas etiquetas son mostradas en la imagen 261.

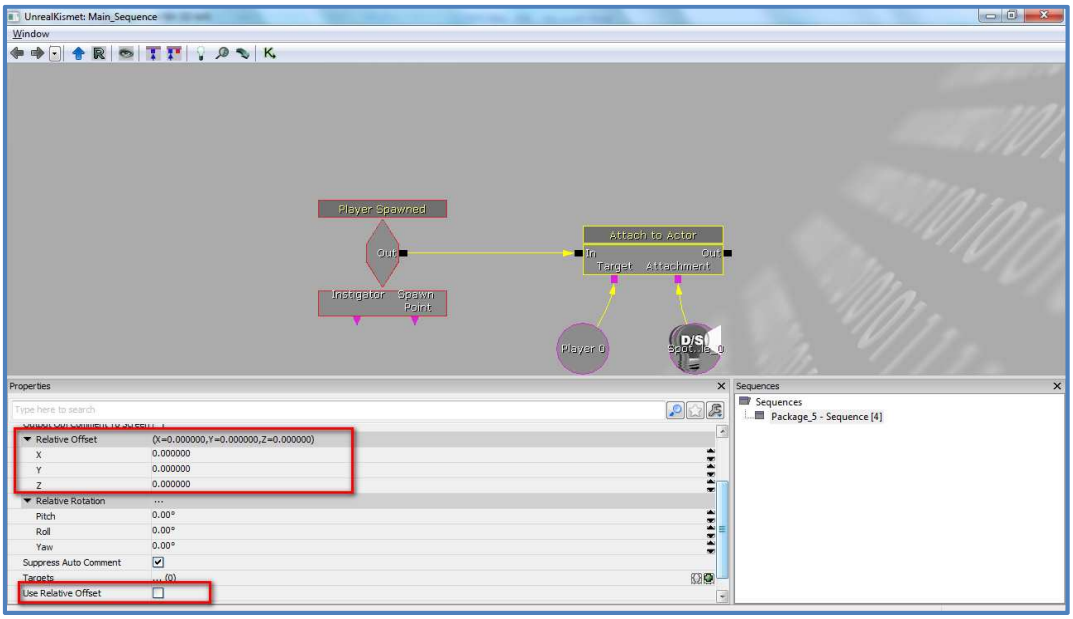


Imagen 261:Tutorial 3-Etiquetas de posición relativa en "Attach to Actor".

- Rotación relativa: Al igual que se puede indicar la posición relativa del objeto, también se puede indicar el ángulo.

  Dicho ángulo relativo se debe indicar en cada uno de los ejes. La pestaña donde se indican estos valores es llamada *"Relative Rotation"*.
  También para utilizar estos valores hay que activar la etiqueta *"Use Relative Rotation"*.

  En la imagen 262 se muestra la localización de las etiquetas en las propiedades.

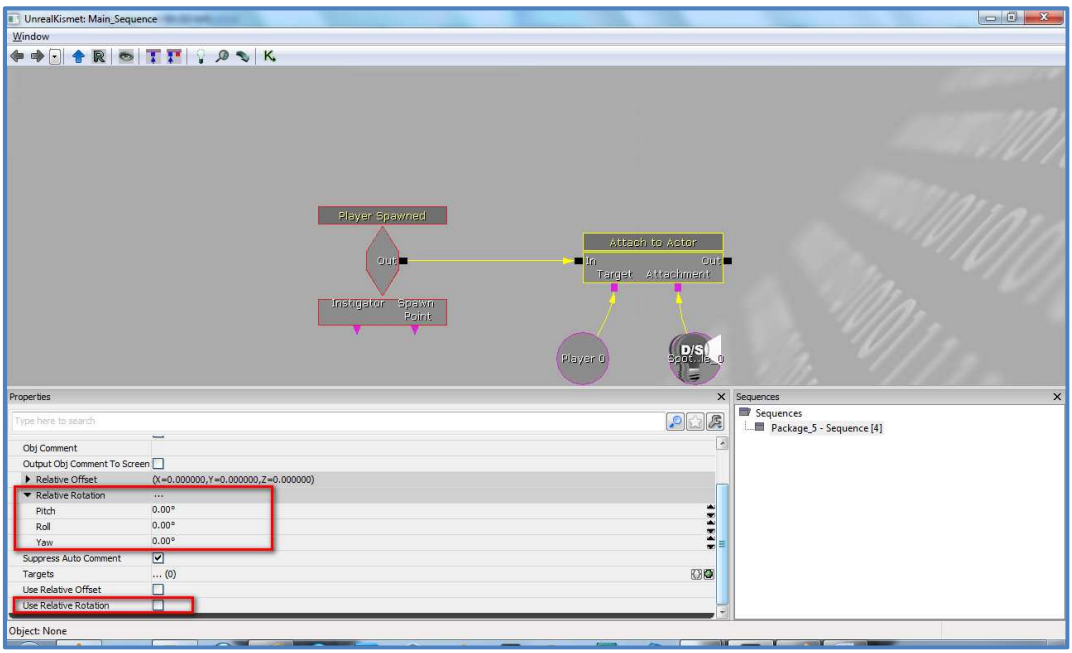


Imagen 262:Tutorial 3-Etiquetas de rotación relativa en "Attach to Actor".

8.4. Tutorial 4: Poner la cámara en 3ª persona.

Una opción importante a la hora de realizar un juego es tomar la decisión de cómo será la perspectiva del jugador. [47]

En el motor Unreal la perspectiva que se tiene por defecto es en 1ª persona. En este tutorial se va a mostrar la forma de modificar la perspectiva de juego, poniéndola en 3ª persona, en la vertical del personaje principal, etc.

Pasos a seguir:

1. Es recomendable tener acoplado el elemento *"PlayerStart"* en el nivel para tener una referencia de la aparición del personaje principal y acoplar la cámara, tal y como se muestra en la imagen 263.

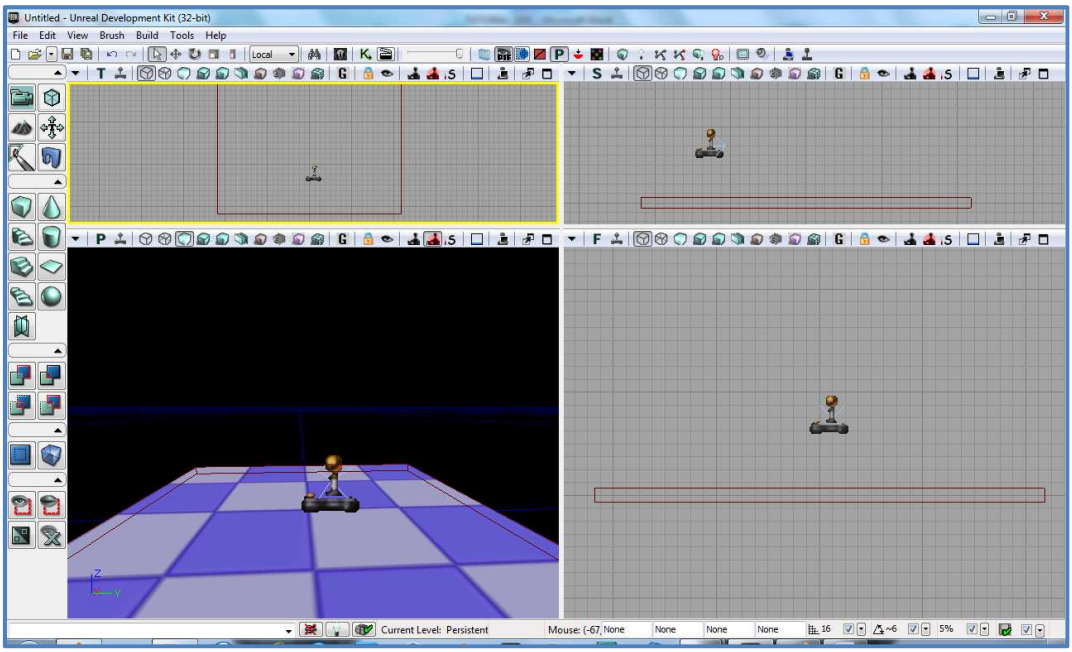


Imagen 263:Tutorial 4-Situación del "PlayerStart".

2. El nuevo punto de vista será el proporcionado por un objeto especial del editor denominado "*Camera*".

   Este objeto puede ser encontrado en la herramienta "*Actor Classes*" accediendo a través del icono de la barra de herramientas superior del editor en la correspondiente pestaña.

   El nombre que posee este elemento es "*CameraActor*" y su localización es mostrada en la imagen 264.

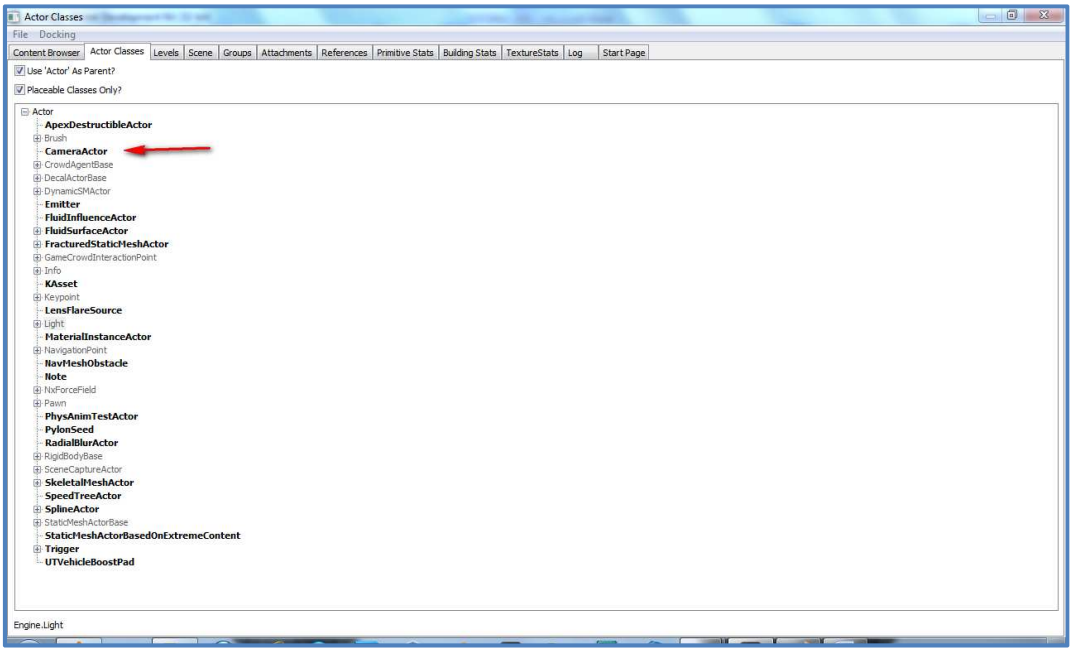


Imagen 264:Tutorial 4-Localización "Camera Actor" en "Actor Classes".

3. El elemento "*CameraActor*" debe ser acoplado en el nivel en la posición que se quiera para tener la perspectiva deseada.

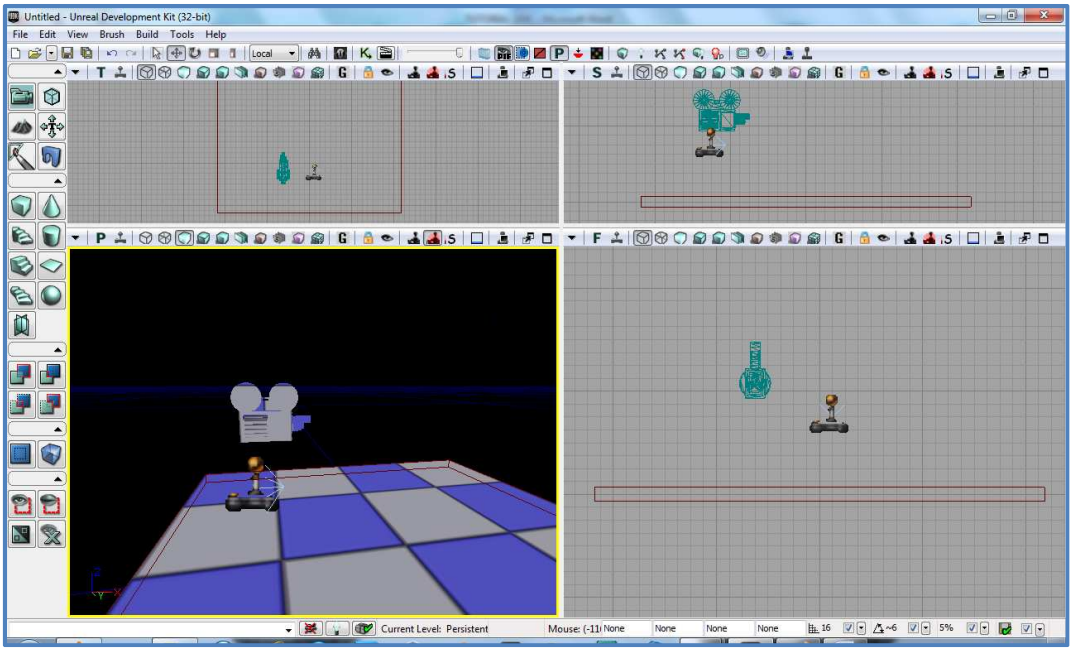


Imagen 265:Tutorial 4-Situación "CameraActor" junto al "PlayerStart".

La mejor forma de colocar la cámara es obtener la visión que se tiene a través de ella.

Para obtener esa visión hay que seguir los siguientes pasos:

- Seleccionar la cámara la cual comprobar su visión.
- Sobre el objeto de la cámara, presionar con el botón derecho del ratón apareciendo un submenú.
- En el menú mostrado seleccionar *"Snap View To Actor"*, imagen 266, y en la pantalla *"Perspective"* del editor se mostrará la visión de la cámara seleccionada.

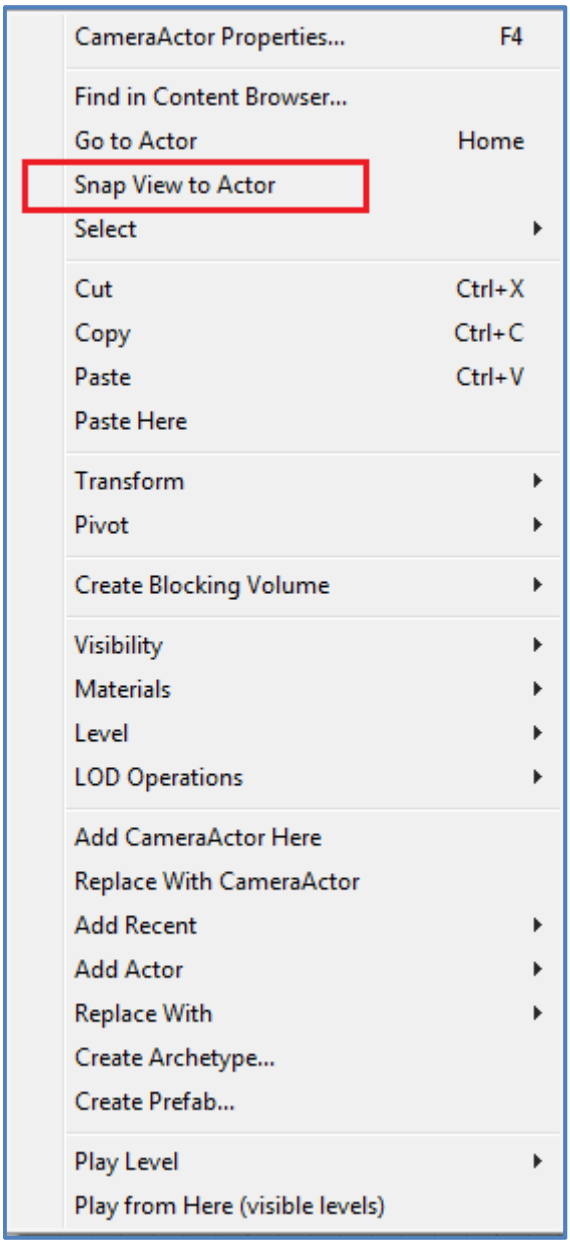


Imagen 266:Tutorial 4-Localización "Snap View to Actor" en menú.

El objeto de la cámara podrá ser desplazado y girado de la misma manera que los otros objetos del editor para colocarlo en la posición deseada.

4. La restante implementación debe realizarse en la herramienta Unreal Kismet para acoplar la cámara al personaje principal.

   Se abrirá la herramienta mediante el icono de la barra de herramientas superior del editor.

   Hay que insertar en el Kismet el evento que representa la carga del nivel. Mediante este evento, las acciones asociadas serán ejecutadas nada más arrancar el nivel. Este evento es llamado *"Level Loaded"*, imagen 267, y se obtiene mediante:

   Botón derecho del ratón → New Event →Level Loaded

En este caso, las acciones que se quieren realizar nada más arrancar el nivel son el acoplar la cámara al personaje principal y que la vista del jugador sea la proporcionada por la cámara seleccionada.

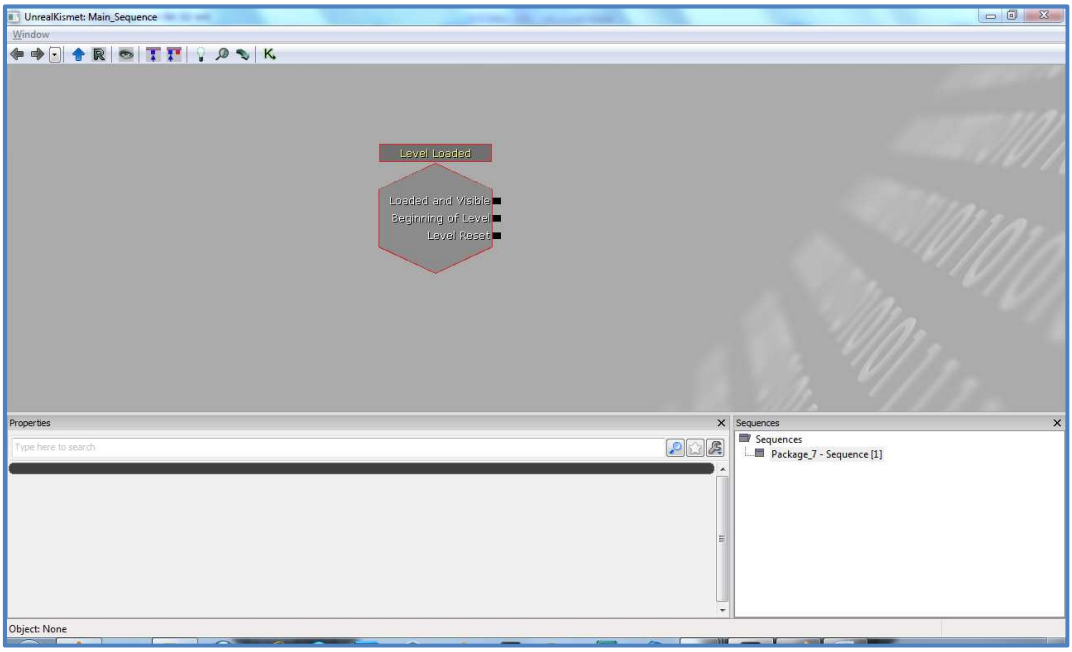


Imagen 267:Tutorial 4-Creación elemento "Level Loaded" en "Kismet".

En las propiedades del *"Level Loaded"* se tiene una etiqueta llamada *"Max Trigger Count"* que representa las veces que se quiere que se lance el evento. Por defecto se tiene el valor 1, con lo que sólo se lanzaría una sola vez. En este caso se quiere tener indefinidamente, lanzándose todas las veces, sin número fijado. Para ello hay que poner el valor de 0 que es el valor de infinito.

5.Otro elemento que hay que insertar es el encargado de acoplar cualquier objeto a un Actor. Este elemento es llamado *"Attach to Actor"* y se obtiene mediante:

Botón derecho del ratón → New Action → Actor → Attach to Actor

Se unirán todas las salidas del *"Level Loaded"* con la entrada de este nuevo elemento, tal y como se muestra en la imagen 268.

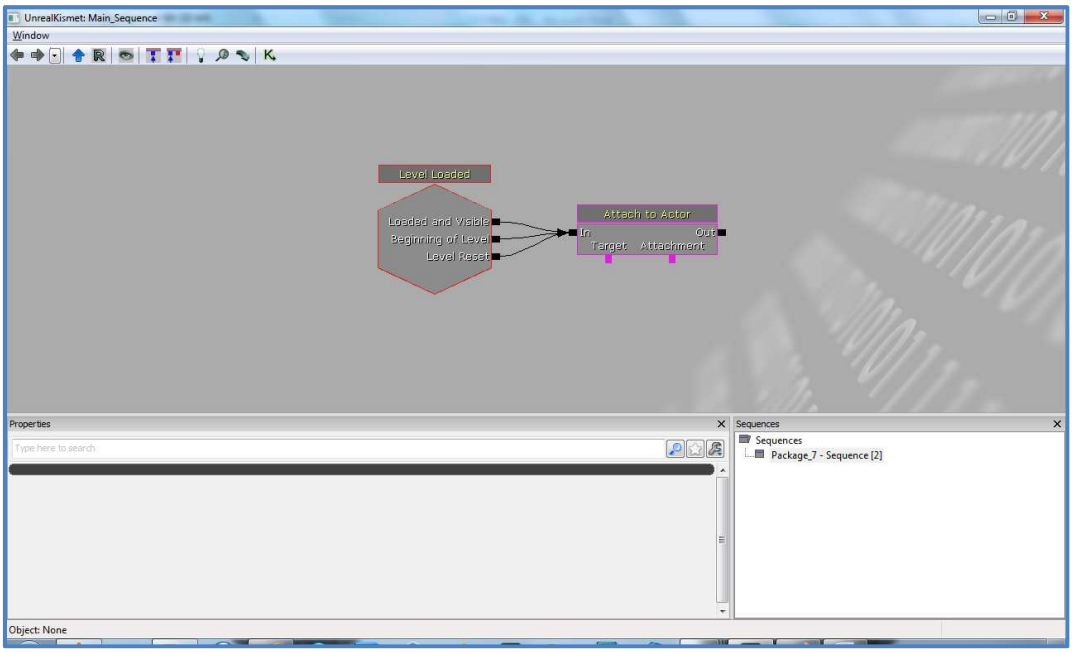


Imagen 268:Tutorial 4-Elemento "Attach to Actor".

6. Es necesario realizar el cambio de perspectiva del juego a la proporcionada por la cámara.

   El elemento que permite realizar esta operación es llamada *"Set Camera Target"* y se puede insertar mediante:

   Botón derecho del ratón → New Action → Camera → Set Camera Target

   Se unirán todas las salidas del *"Level Loaded"* con la entrada de este nuevo elemento, imagen 269.

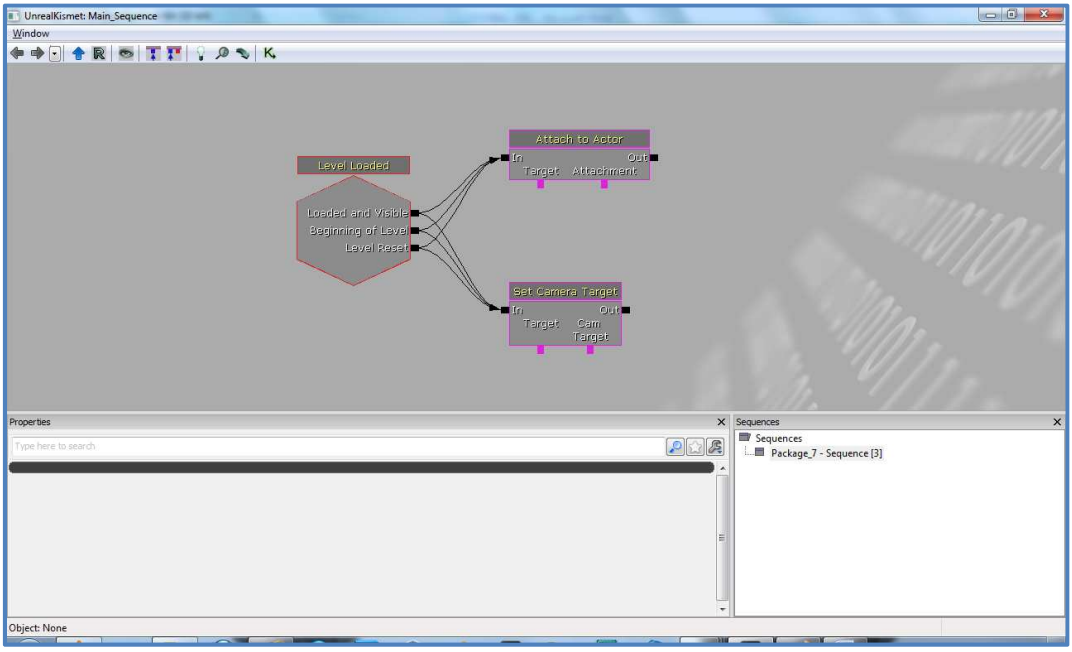


Imagen 269:Tutorial 4-Elemento "Set Camera Targer".

7. En Kismet es necesaria una variable que represente la cámara acoplada en el editor.

En el editor se selecciona el objeto de la cámara y sin dejar de seleccionarlo volver a la herramienta Kismet.

Pulsar con el botón derecho del ratón sobre cualquier zona despejada y en el menú mostrado seleccionar *"Create Object Var Using CameraActor_x"* y aparecerá la variable que representa a la cámara.

Esta variable hay que unirla con la entrada *"Attachment"* del *"Attach to Actor"* y la entrada *"Cam Target"* del *"Set Camera Target"*, imagen 270.

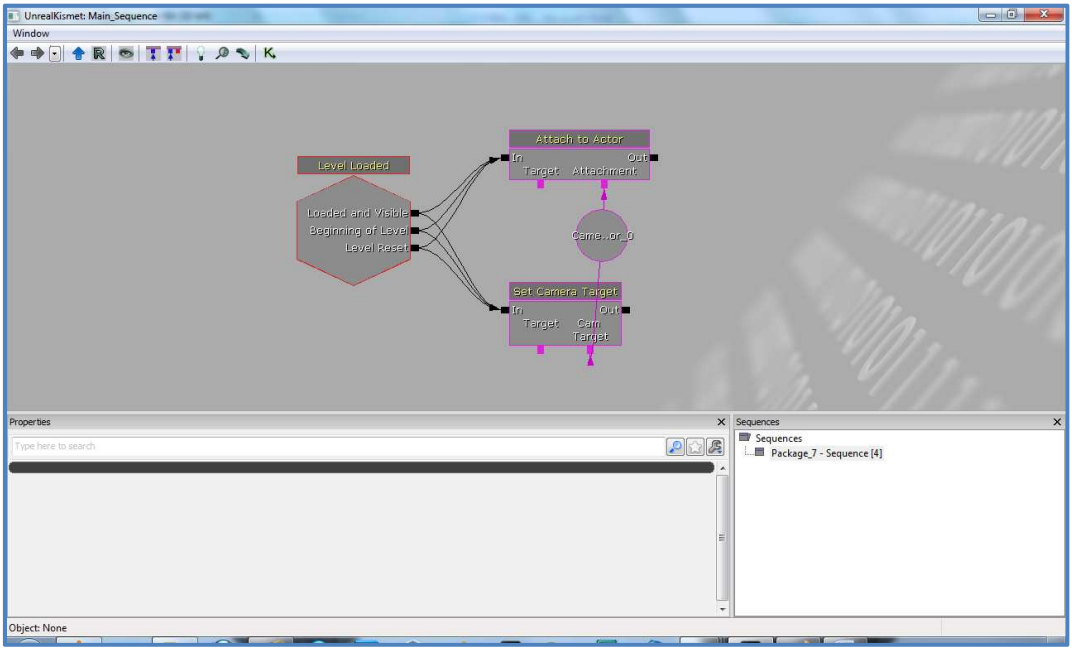


Imagen 270:Tutorial 4-Variable representante de la cámara acoplada al nivel.

8. También es necesaria una variable que represente el Actor al que acoplar la cámara y la perspectiva.

   En el caso del tutorial, este Actor es el jugador, con lo que hay que obtener una variable que lo represente, mediante:

   Botón derecho del ratón → New Variable → Player → Player

   En las propiedades de esta variable no se debe seleccionar la etiqueta *"All Players"*, para que sólo represente al jugador principal.

   La variable se unirá con la entrada *"Target"* de los elementos *"Attach to Actor"* y *"Set Camera Target"*, mostrado en la imagen 271.

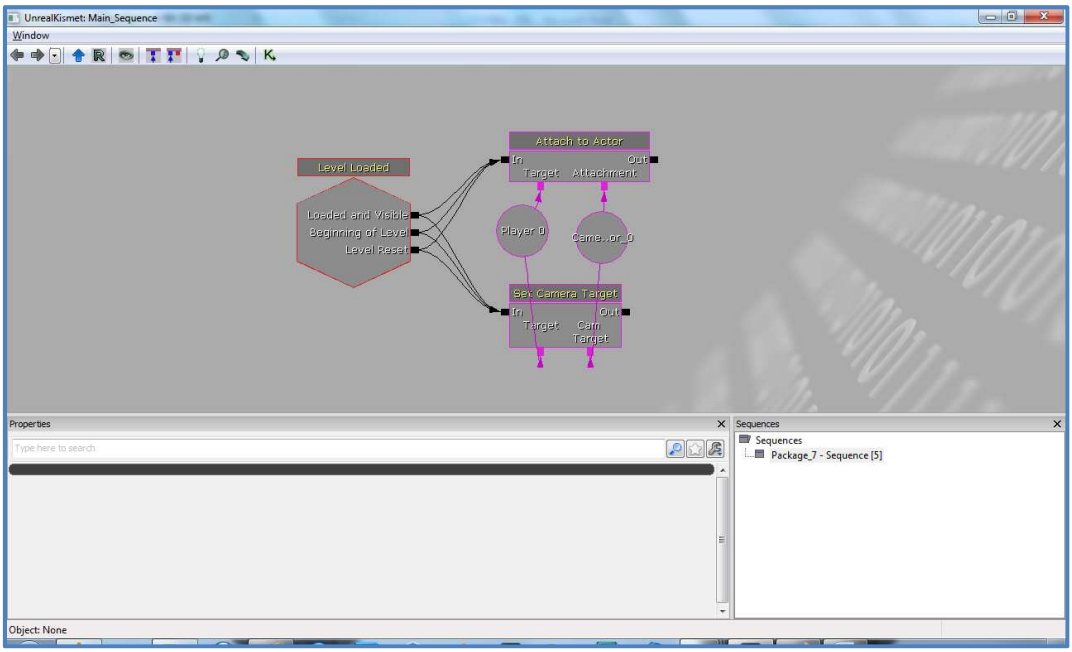


Imagen 271:Tutorial 4-Variable jugador.

Las opciones de perspectiva de la cámara son infinitas. Se puede llegar a tener la perspectiva del famoso juego *"Gears of War"*:

Imagen 272:Tutorial 4-Ejemplo vista en tercera persona.

O una vista aérea como en el juego *"Diablo"*:

Imagen 273:Tutorial 4-Ejemplo vista aérea.

8.5. Tutorial 5: Crear movimiento de cámara.

Durante el desarrollo del juego, puede ser necesario el mostrar alguna acción o lugar del mapa diferente al que se encuentra el jugador. [48]

Como ejemplo de esta situación, puede ser el querer mostrar la apertura/cierre de una puerta al utilizar un interruptor, mostrar el camino a tomar a partir de un punto o realizar una acción, etc.

Pasos a seguir:

1. La vista que se quiere mostrar será la proporcionada por el movimiento de una cámara. El objeto que representa a las cámaras se denomina *"Camera Actor"*.

   Este elemento puede ser encontrado en la herramienta *"Actor Classes"* accediendo a través del icono de la barra de herramientas superior del editor, en la correspondiente pestaña. En la imagen 274 se muestra la localización de este elemento.

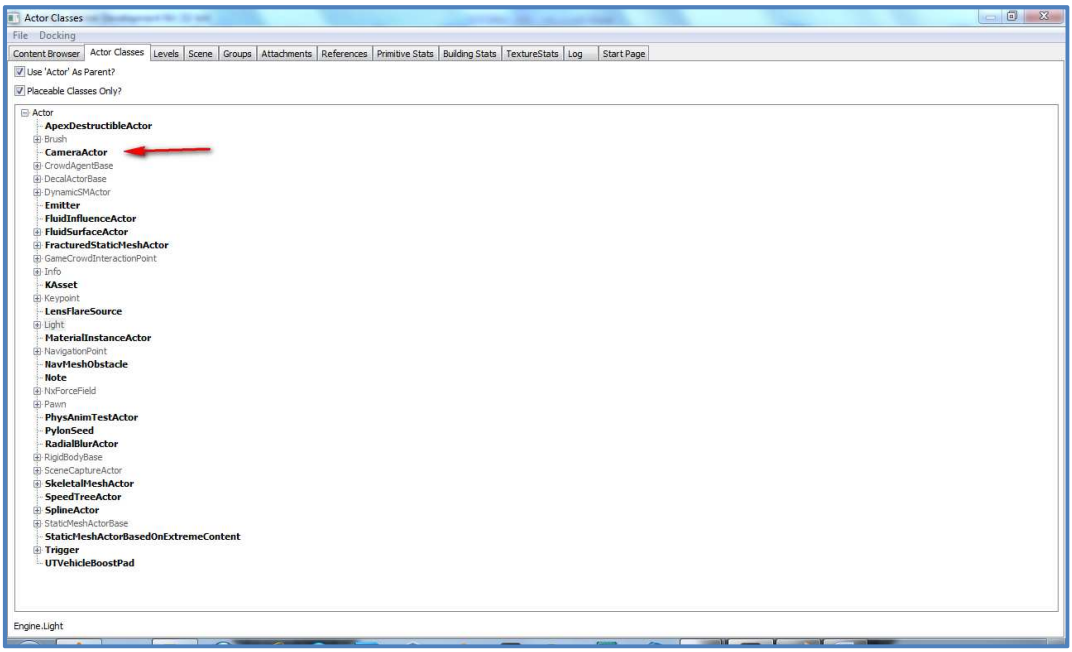


Imagen 274:Tutorial 5-Localización "Camera Actor" en "Actor Classes".

2. El objeto de *"Camera Actor"*, imagen 275, se acoplará en el nivel en la posición que se quiera. Este objeto se podrá trasladar y girar para poder obtener la vista inicial deseada.

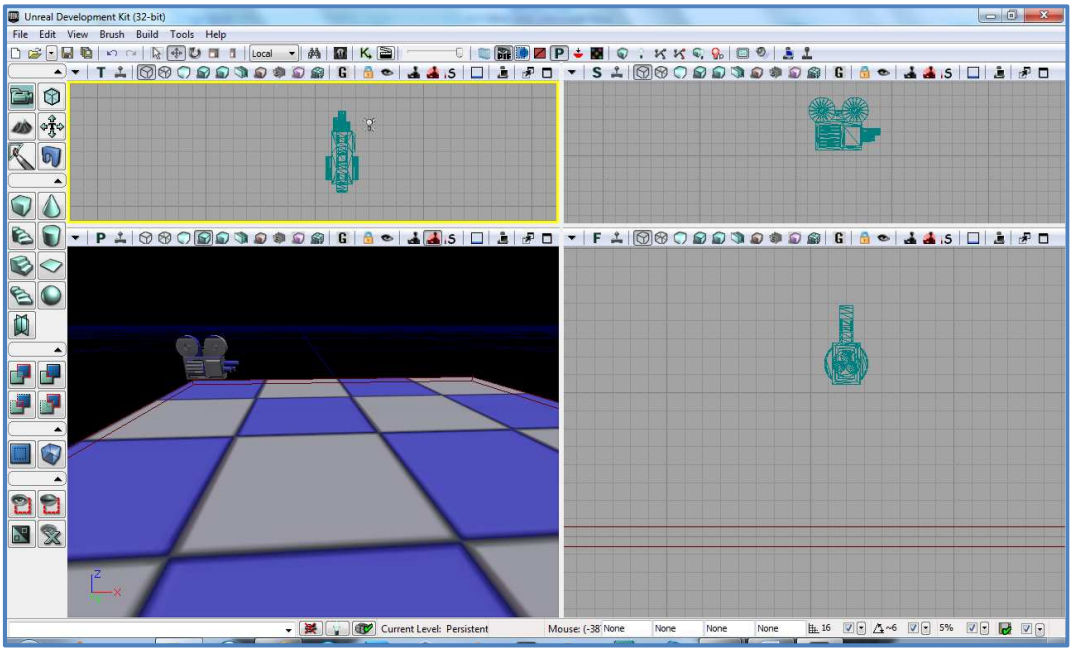


Imagen 275:Tutorial 5-Situación de la cámara en el nivel.

La mejor forma de colocar la cámara es obtener la visión que se tiene a través de esa cámara. Para obtener esa visión hay que seguir los siguientes pasos:

- Seleccionar la cámara la cual comprobar su visión.
- Sobre el objeto de la cámara presionar con el botón derecho del ratón apareciendo un submenú.

- En el menú mostrado seleccionar *"Snap View To Actor"*, imagen 276, y en la pantalla *"Perspective"* del editor se mostrará la visión de la cámara seleccionada.

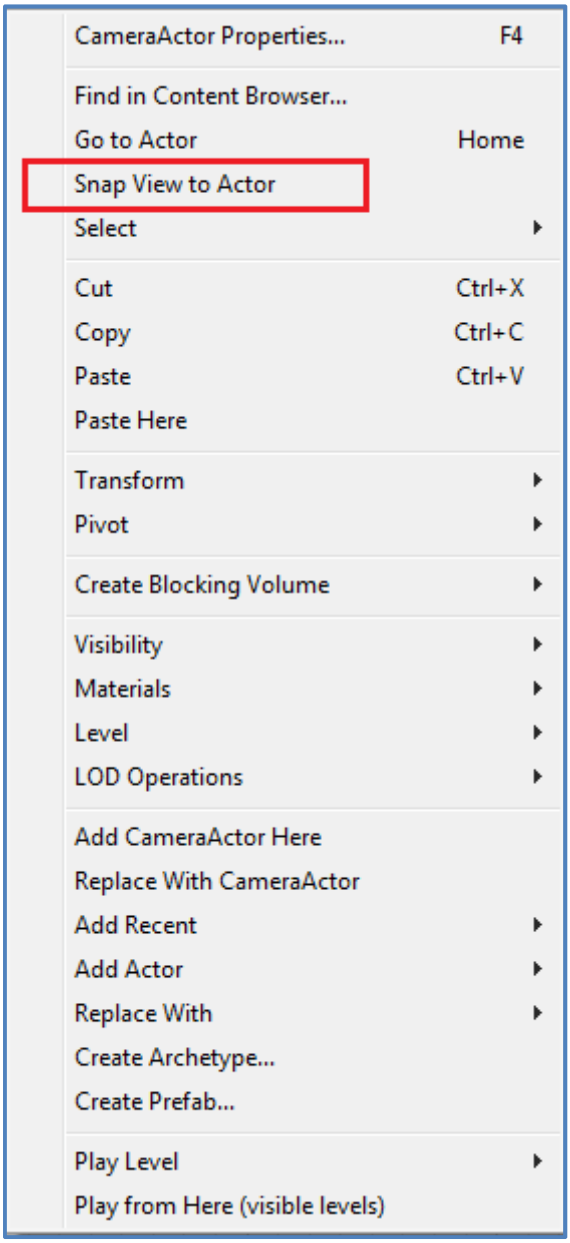


Imagen 276:Tutorial 5-Localización "Snap View to Actor" en menú.

3. Este nuevo objeto acoplado al nivel debe ser seleccionado y sin dejar de seleccionarlo abrir la herramienta Unreal Kismet mediante el icono de la barra de herramientas superior del editor.

   En la herramienta Kismet se crea un nuevo elemento Matinee mediante:

   Botón derecho del ratón → New Matinee

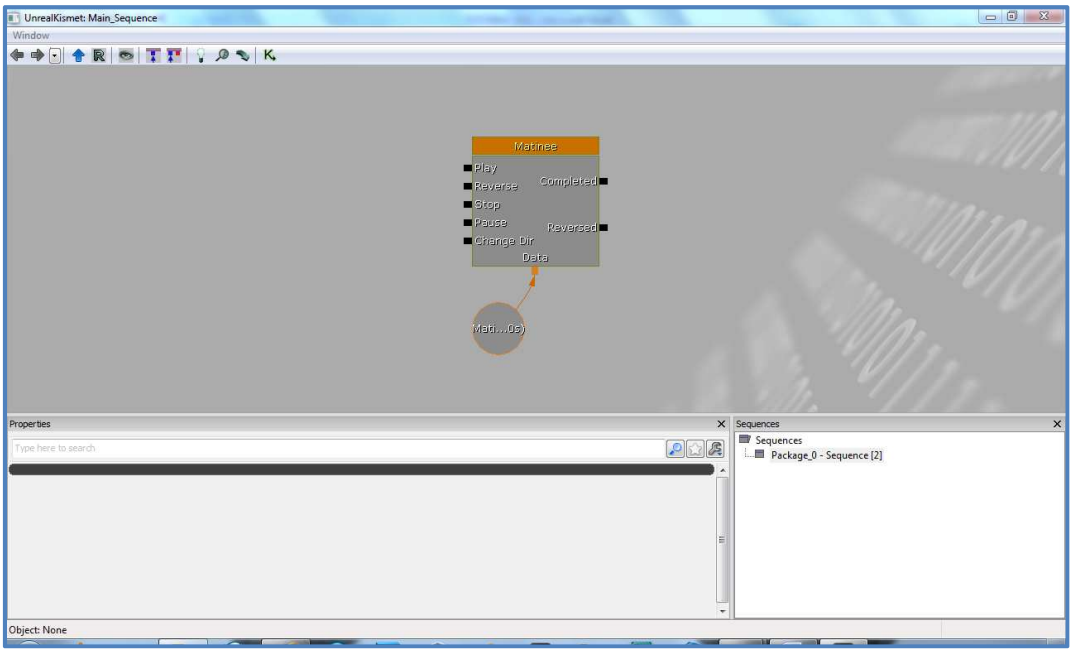


Imagen 277:Tutorial 5-Creación elemento "Matinee" en "Kismet".

4. Se debe abrir este nuevo elemento insertado. Para ello hay que realizar doble click sobre él abriéndose la herramienta Unreal Matinee, tal y como se muestra en la imagen 278.

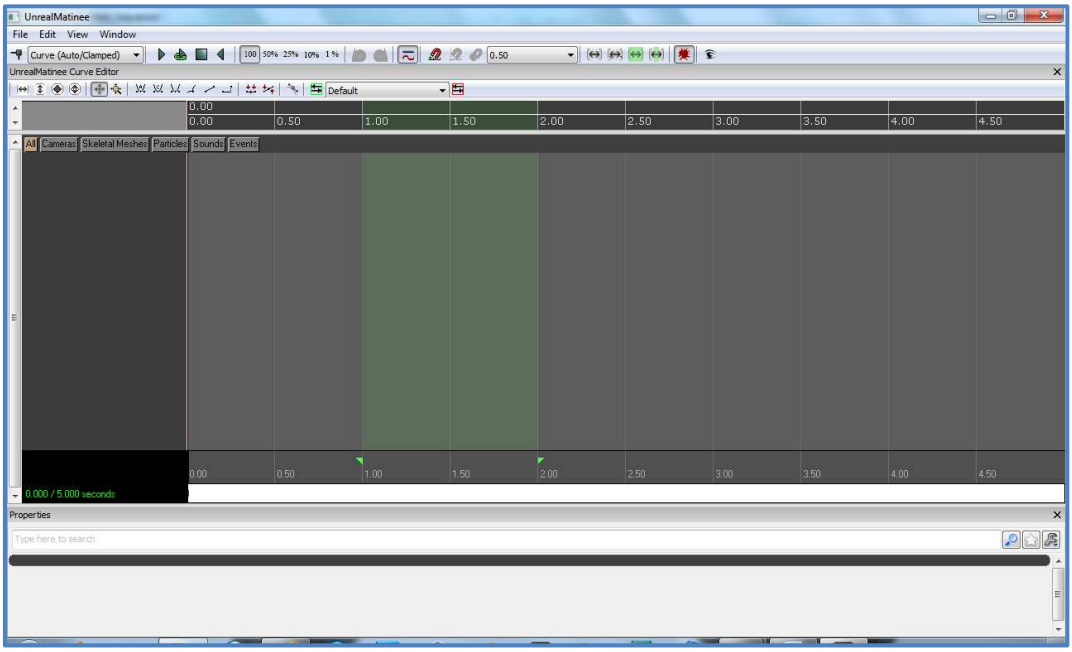


Imagen 278:Tutorial 5-Interfaz herramienta "Matinee".

5. Se debe crear un nuevo grupo vacío llamado *"Camara"*.

   Para crear este nuevo grupo se pulsará con el botón derecho del ratón y en el menú mostrado seleccionar *"Add New Empty Group"*. Aparecerá el nuevo grupo creado, tal y como se muestra en la imagen 279.

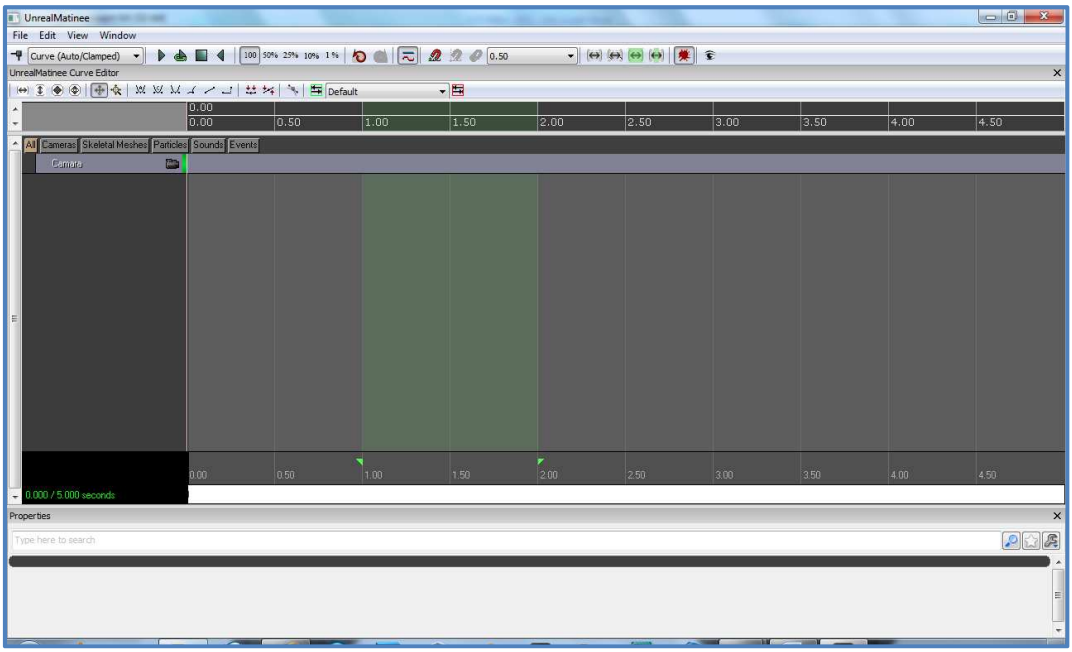


Imagen 279:Tutorial 5-Creación de grupo vacío llamado "cámara".

6. Dentro de este grupo se debe crear un nuevo track de movimiento.

   Seleccionando el grupo, se pulsa el botón derecho del ratón y en el menú mostrado se selecciona *"Add New Movement Track"*. Se muestra el track creado en la imagen 280.

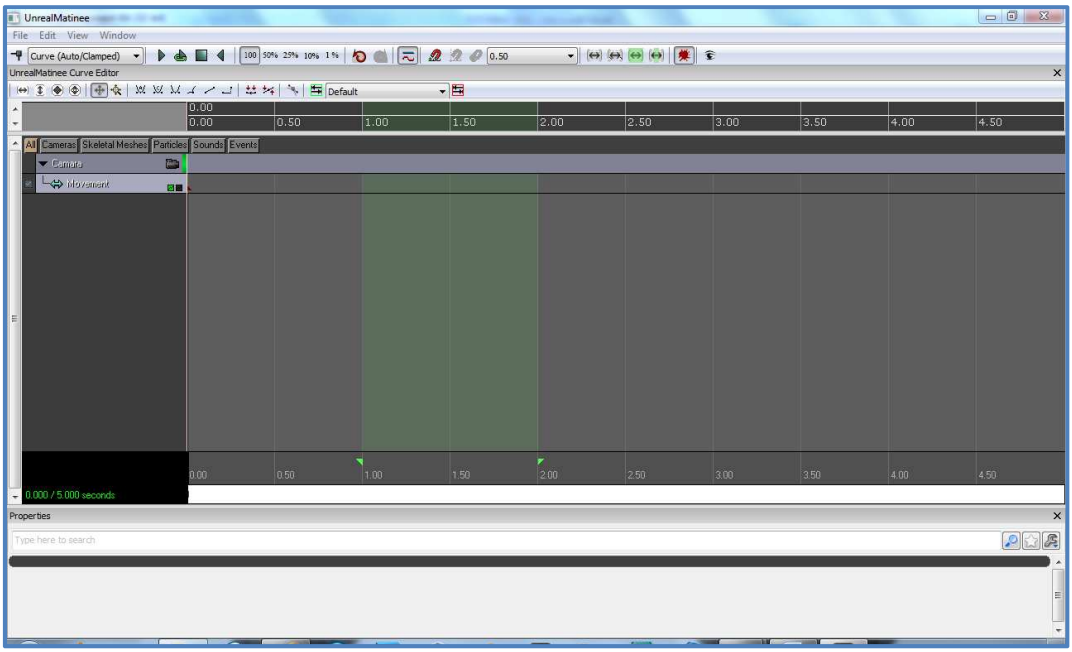


Imagen 280:Tutorial 5-Track de movimiento en el grupo "cámara".

   En este track creado se guardará el movimiento del objeto cámara que se tiene en el editor.

7. En el track de movimiento se van guardando puntos de control con diferente posición de la cámara. Estos puntos de control se muestran con triángulos rojos y entre punto y punto de control se guarda el movimiento del objeto. Los puntos de

control se crean colocando la línea temporal en el tiempo que se quiere y con el track seleccionado presionar la tecla 'Enter' del teclado.

El movimiento de la cámara se puede realizar activando el *"Traslation Mode"* del editor y realizándolo después de crear un punto de control.

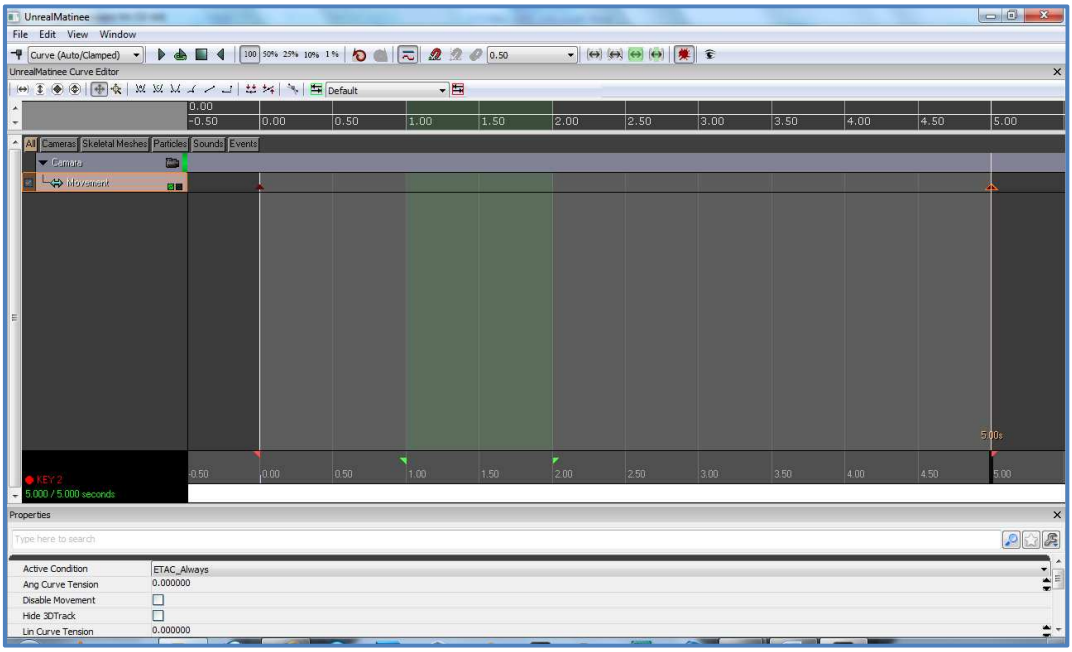


Imagen 281:Tutorial 5-Secuencia de movimiento creada con la cámara.

El movimiento creado puede ser comprobado moviendo la línea temporal entre los puntos de control y la cámara en el editor debe realizar el movimiento guardado.

8. Se debe mostrar lo captado por la cámara en la que se ha realizado el movimiento en el transcurso del juego.

   Se debe crear un nuevo grupo llamado *"Director Group"* en el Matinee, pulsando con el botón derecho del ratón y en el menú mostrado seleccionar *"Add New Director Group"*. Este nuevo grupo se muestra en la imagen 282.

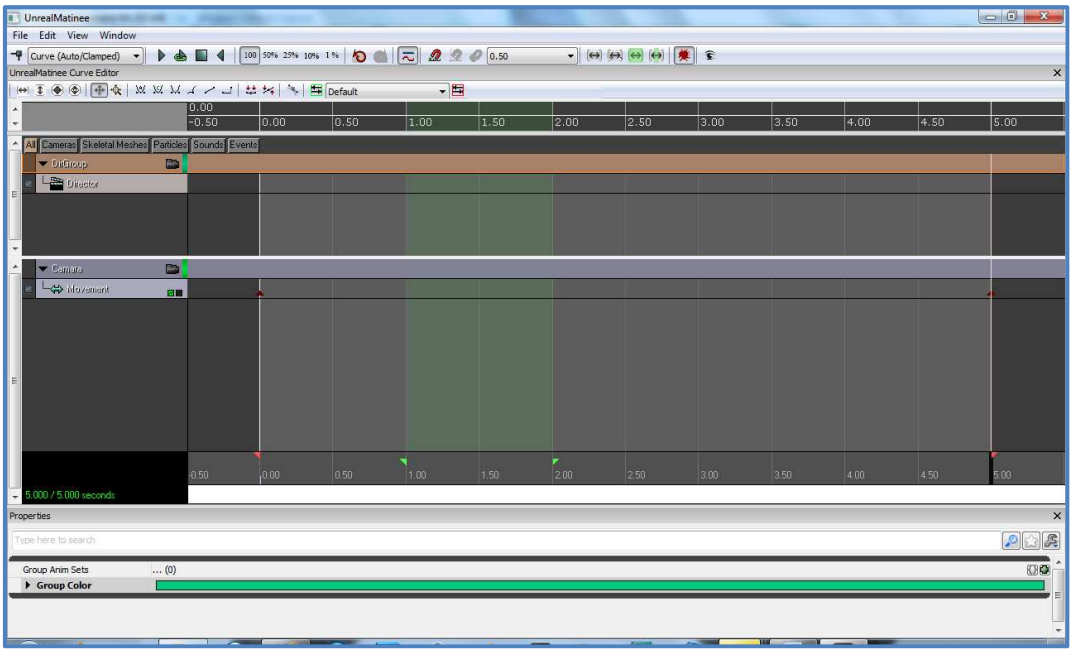


Imagen 282:Tutorial 5-Creación del grupo "DirGroup" y su track "Director".

En el grupo creado con nombre *"DirGroup"* se tiene un track llamado *"Director"* que es donde se indicará cuándo mostrar la imagen de la cámara.

9. En el track *"Director"* se creará un punto de control, en el tiempo a partir del cual se quiere mostrar la imagen de la cámara.

   Para insertar un punto de control se moverá la línea temporal al tiempo deseado y al pulsar la tecla 'Enter' se pedirá que se indique el grupo que se quiere asociar, seleccionando en este caso el grupo creado anteriormente con nombre *"Camara"* que es el que posee el movimiento de la cámara, como se muestra en la imagen 283.

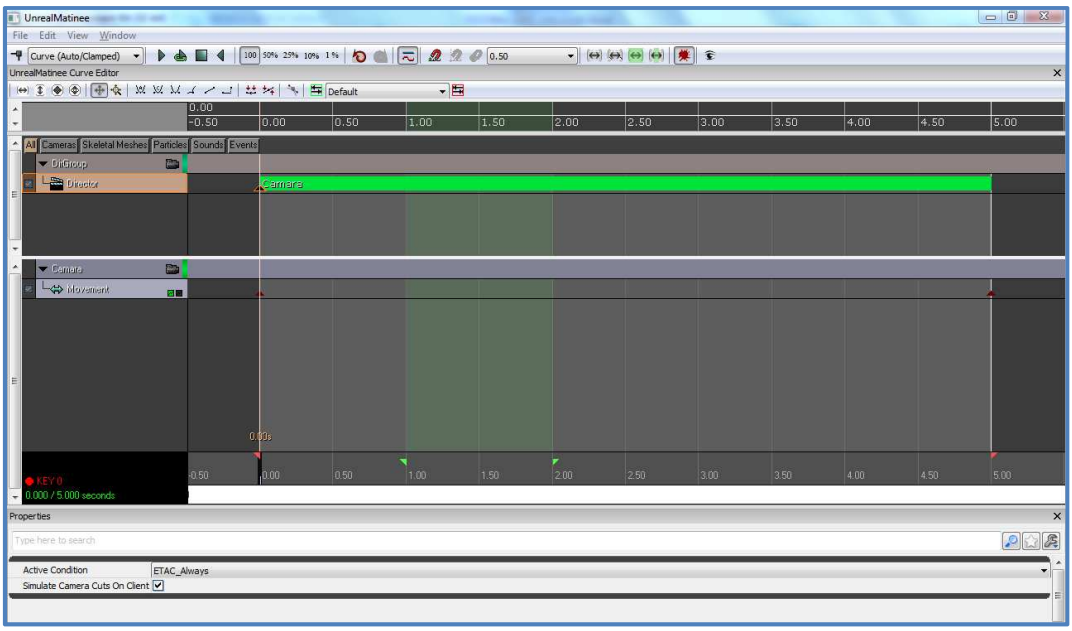


Imagen 283:Tutorial 5-Asociación de la cámara con el "DirGroup".

10. Ya puede cerrarse la herramienta Unreal Matinee quedando guardados los cambios realizados.

    El objeto que se tiene en la herramienta Unreal Kismet posee todos los cambios, mostrando la imagen del movimiento de la cámara pudiéndose asociar a los demás elementos y reproducirlo en el flujo del nivel. Se muestra un ejemplo de asociación del elemento Matinee en la imagen 284.

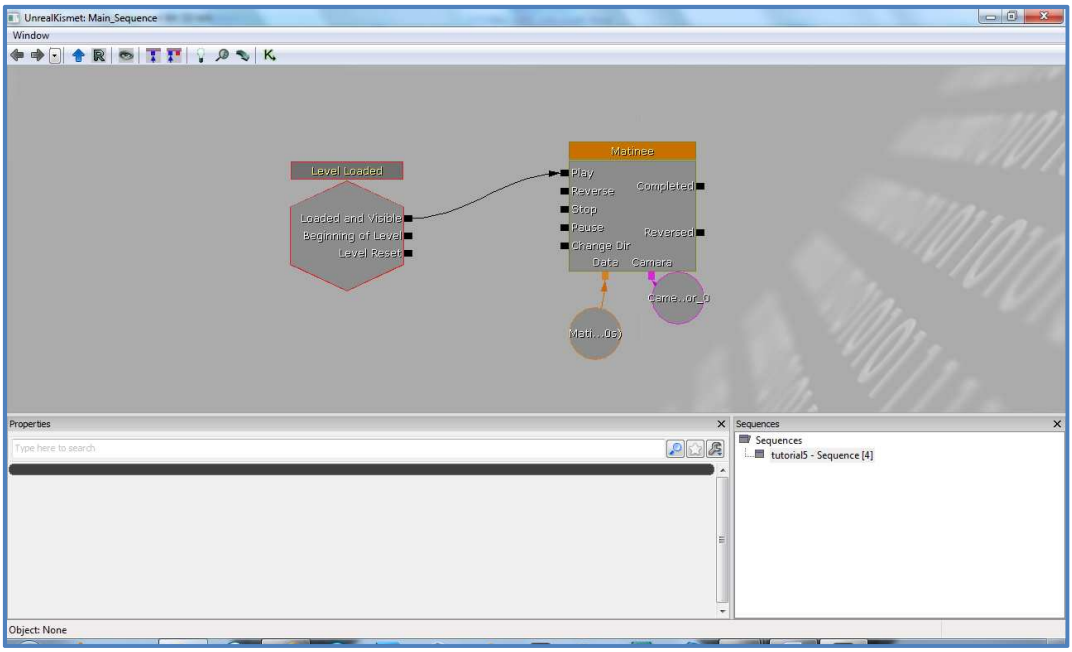


Imagen 284:Tutorial 5-Asociación "Level Loaded" con elemento "Matinee".

8.6. Tutorial 6: Desaparición de objeto al tocarlo.

Una opción que se puede querer hacer en el juego es que los objetos puedan aparecer o desaparecer según distintas acciones realizadas. [49]

Todos los objetos acoplados en el nivel tienen la propiedad que se muestren o no en la ejecución del juego. Esta propiedad puede encontrarse en las propiedades de cada objeto, pulsando la tecla 'F4', en la etiqueta *"Hidden"* dentro de la jerarquía *Display → Hidden.* Activando esta etiqueta, el objeto estará oculto en la ejecución del nivel, y mostrándolo en el caso contrario.

Una cualidad de esta propiedad es que se puede modificar en tiempo de ejecución, mientras el nivel es lanzado, pudiendo hacer que objetos aparezcan o desaparezcan.

En este tutorial se va a mostrar la forma de realizar que un objeto desaparezca estando próximo a él, pero se puede utilizar un interruptor, que el personaje principal se encuentra en una zona deseada, etc., existiendo multitud de opciones. La forma de realizar todas estas opciones sería prácticamente igual.

Pasos a seguir:

1. El objeto al cual modificar su visibilidad debe ser del tipo *"InterpActor"* para poder ser modificado.

Debe ser añadido al nivel desde el Content Browser pudiéndose abrir mediante el icono de la barra de herramientas superior del editor.

Con el objeto seleccionado, mostrado con un marco amarillo alrededor, se vuelve al editor y en cualquier vista realizar:

Botón derecho del ratón → Add Actor → Add InterpActor: Static Mesh <nombre_objeto>

Siendo <nombre_objeto> el nombre del mesh que se quiere importar.

En la imagen 285 se muestra un ejemplo de objeto insertado en el nivel creado.

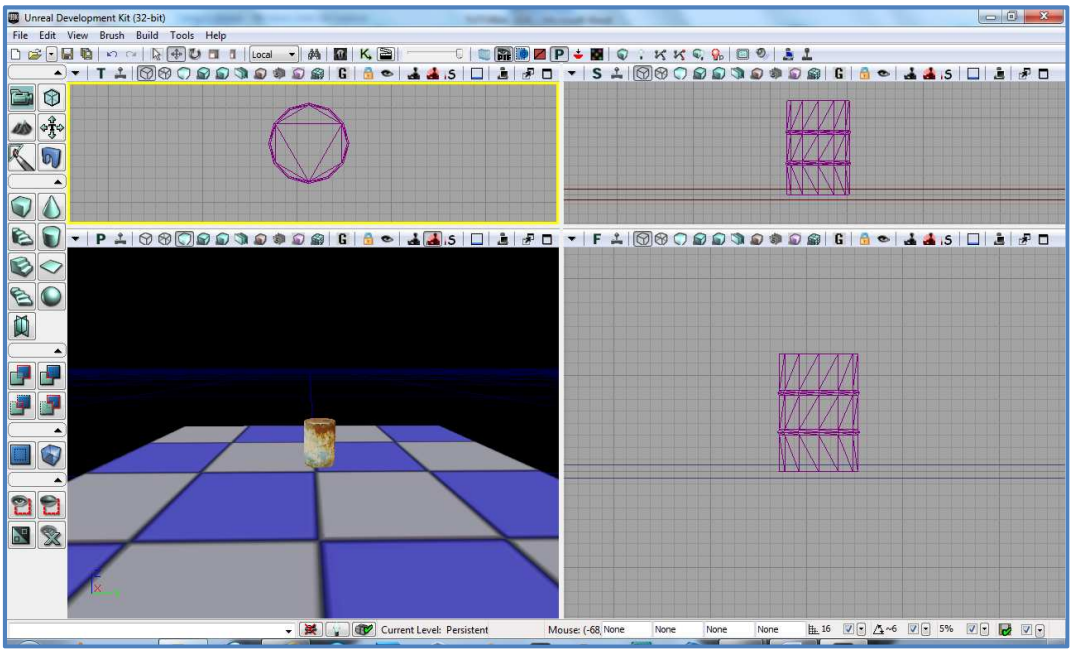


Imagen 285:Tutorial 6-Acople de objeto en el nivel.

En este tutorial, se ha utilizado el objeto *"StaticMesh'Interactive_Meshes.SM.Barrel_01'"*, pudiéndose utilizar cualquier otro para ello.

2. La modificación de la visibilidad en ejecución se hará mediante la herramienta Unreal Kismet.

   Se seleccionará el objeto a modificar y sin dejar de estar seleccionado se abre Kismet mediante el icono de la barra de herramientas superior. En la herramienta abierta hay que crear un evento de contacto con el objeto seleccionado mediante las acciones:

   Botón derecho del ratón →New Event using InterpActor_x →Touch.

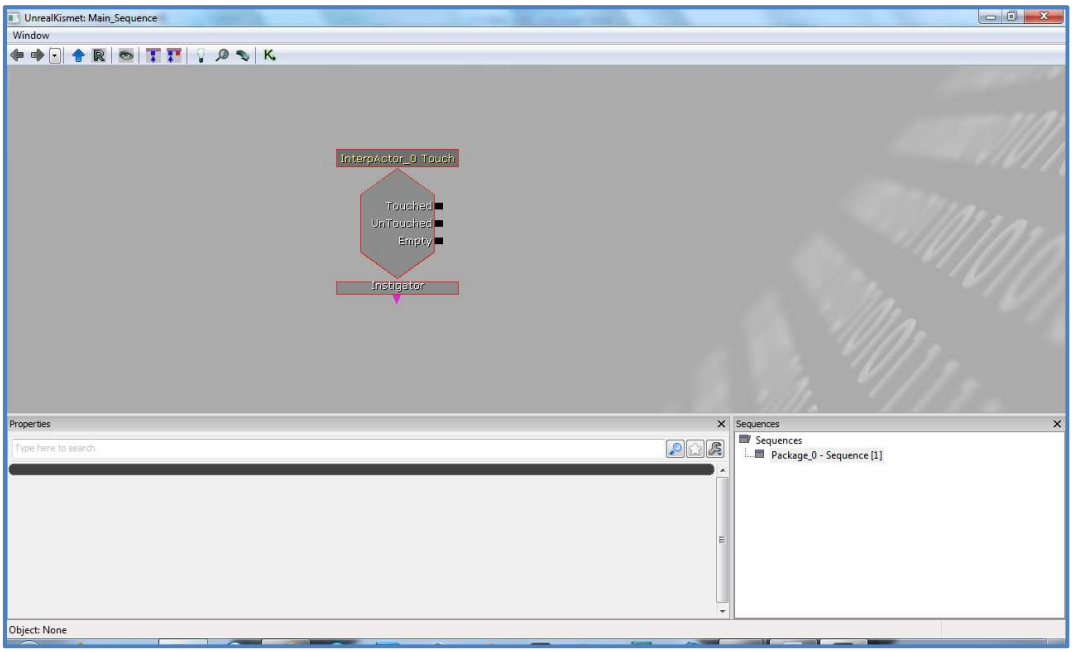


Imagen 286:Tutorial 6-Creación de evento de contacto con objeto en "Kismet".

Hay que tener en cuenta en modificar la etiqueta "*Max Trigger Count*" si se quiere utilizar indefinidamente (valor 0), ya que el valor por defecto es 1, con lo que sólo se puede utilizar una vez.

Este evento, imagen 286, se ha insertado ya que en el tutorial se va a mostrar a modificar la visibilidad del objeto al estar próximo a éste, pero la acción ó evento para modificar la visibilidad puede ser cualquier otro.

3. El elemento que modifica la visibilidad de cualquier objeto acoplado es llamado "*Toggle Hidden*", imagen 287, y puede ser insertado mediante:

   Botón derecho del ratón → New Action → Toggle → Toggle Hidden.

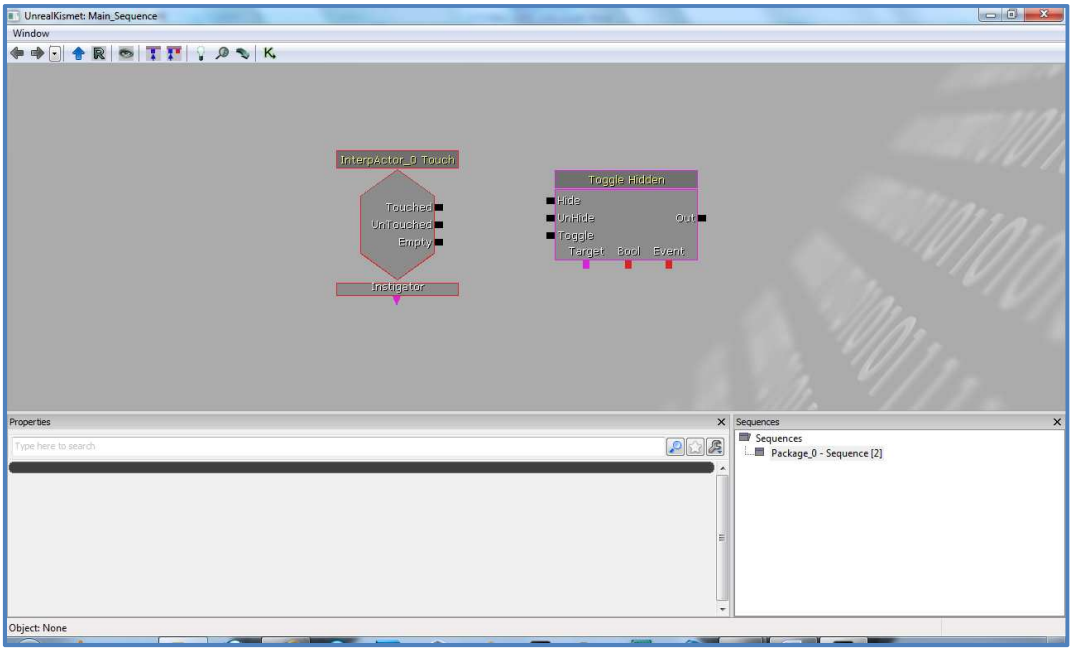


Imagen 287:Tutorial 6-Creación elemento "Toggle Hidden" en "Kismet".

4. Se asocian estos dos elementos (imagen 288).

   En el evento se presentan las salidas de contacto, *"Touched"*, y de no contacto, *"UnTouched"*. En el elemento *"Toggle Hidden"* se tienen las entradas de esconder, *"Hide"*, y de mostrar, *"UnHide"*, el objeto acoplado. Se podrán asociar estas entradas y salidas según se desee. En este tutorial, se quiere que al estar en contacto con el objeto no se muestre, y al contrario cuando no se esté en contacto con él.

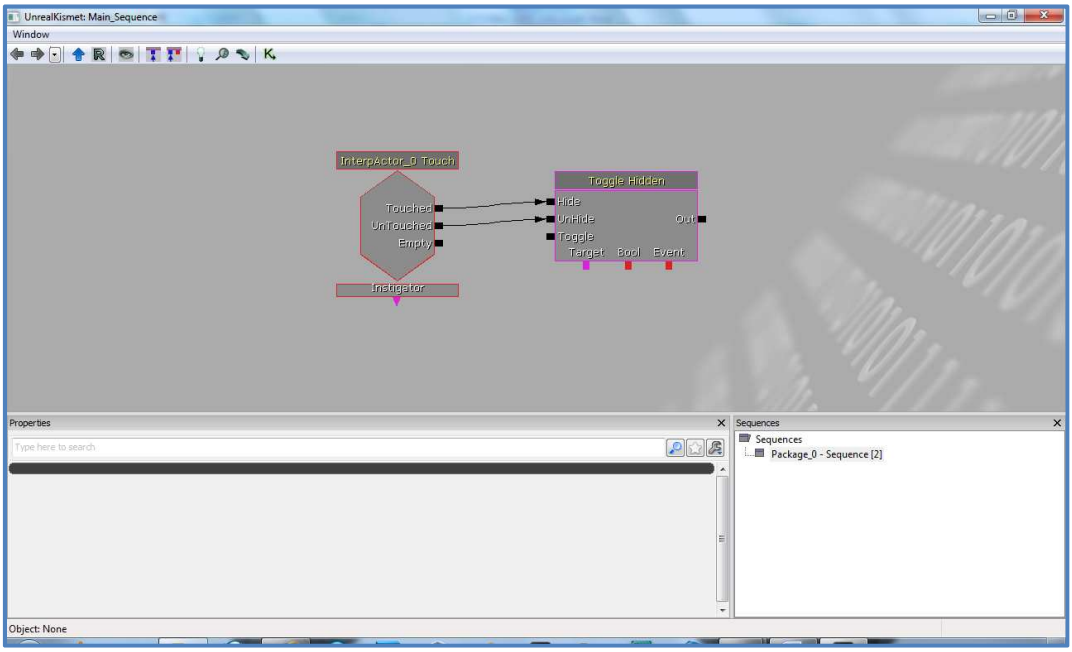


Imagen 288:Tutorial 6-Asociación elementos contacto y "Toggle Hidden".

5. En el elemento *"Toggle Hidden"* hay que indicar el objeto al que se le quiere modificar su visibilidad.

   Para ello es necesaria una variable que represente al objeto que se tiene en el editor. Para conseguir esta variable se debe seleccionar el objeto en el editor, y con el objeto seleccionado volver a la herramienta Unreal Kismet y realizar:

   Botón derecho del ratón → New Object Var using InterpActor_x.

   Esta variable hay que asociarla con la entrada *"Target"* del elemento, para poder así modificar su visibilidad, tal y como se muestra en la imagen 289.

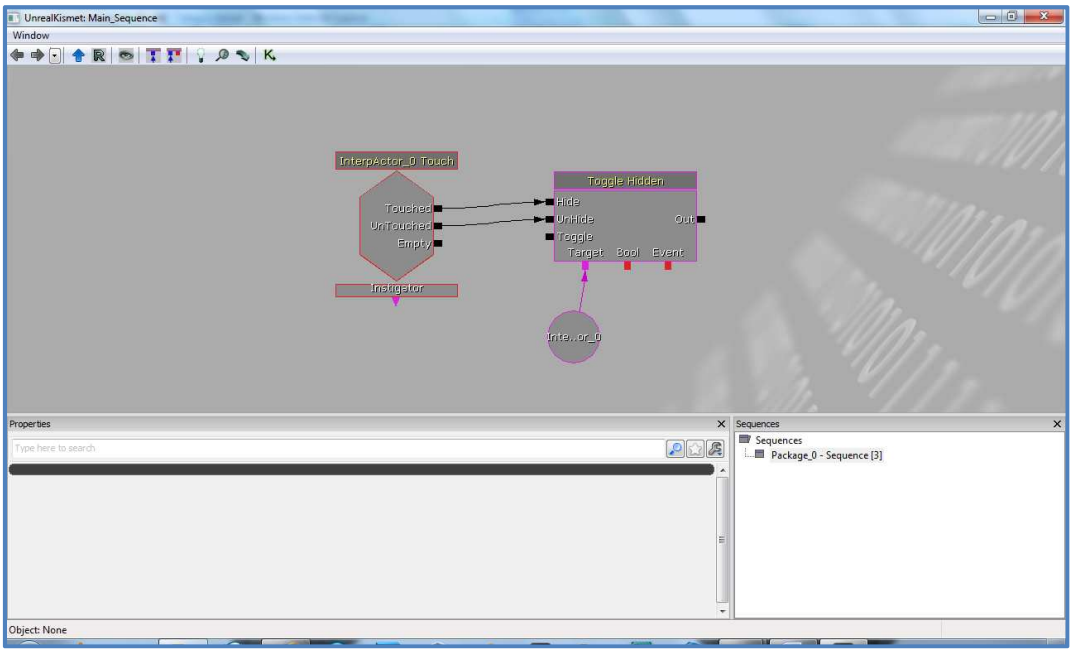


Imagen 289:Tutorial 6-Variable representativa del objeto del nivel.

6. Cuando el personaje principal se acerque al objeto, éste desaparecerá, volviendo a aparecer cuando el personaje se aleje.Puede realizarse la acción del cambio de visibilidad de un objeto por cualquier otro evento ó trigger, según se desee en el flujo del juego.

8.7. Tutorial 7: Teletransportar al personaje principal.

En el transcurso del nivel, puede ser necesario el trasladar al personaje principal a otro punto diferente del nivel de donde se encuentra. [50]

En este tutorial se va a mostrar cómo realizar un teletransporte del personaje principal de un punto a otro del nivel. Se parte de un nivel en el cual se tienen dos plataformas con diferentes colores de luz, mostrado en la imagen 290. En dichas plataformas se tienen añadidos dos *"Static Mesh"* que servirán de referencia de pulsadores, y junto a estos objetos se presentan acoplados dos *"Trigger"* que serán los encargados de lanzar el evento, en este caso de tipo *"uso"*.

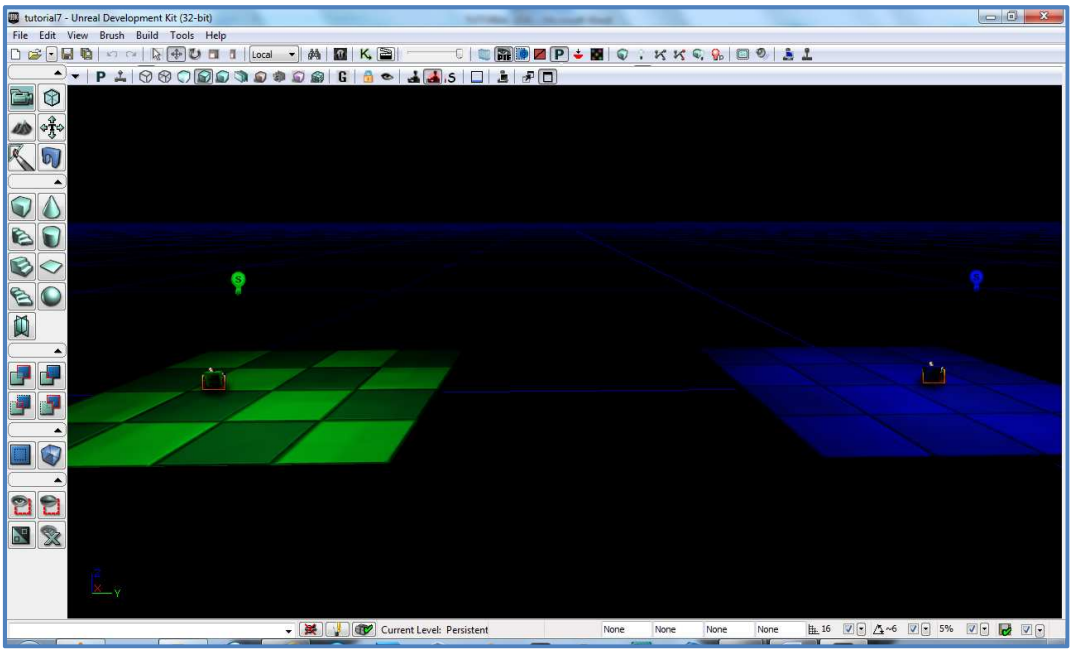


Imagen 290:Tutorial 7-Nivel inicial.

El jugador, al utilizar el pulsador, se trasladará de una plataforma a otra.

Pasos a seguir:

1. De cada uno de los *"trigger"* del nivel hay que obtener el evento de tipo *"uso"* en la herramienta Unreal Kismet.

   Esta herramienta puede abrirse mediante el icono de la barra de herramientas superior del editor. Antes de la apertura de la herramienta hay que seleccionar los dos *"trigger"* en el editor pulsando sobre ellos y manteniendo pulsada la tecla 'Ctrl' para seleccionar el segundo. Con la herramienta abierta se realiza:

   Botón derecho del ratón → New Events using Trigger_x… → Used.

   Mediante esta secuencia se tendrán los dos eventos creados del tipo uso mostrados en la siguiente imagen:

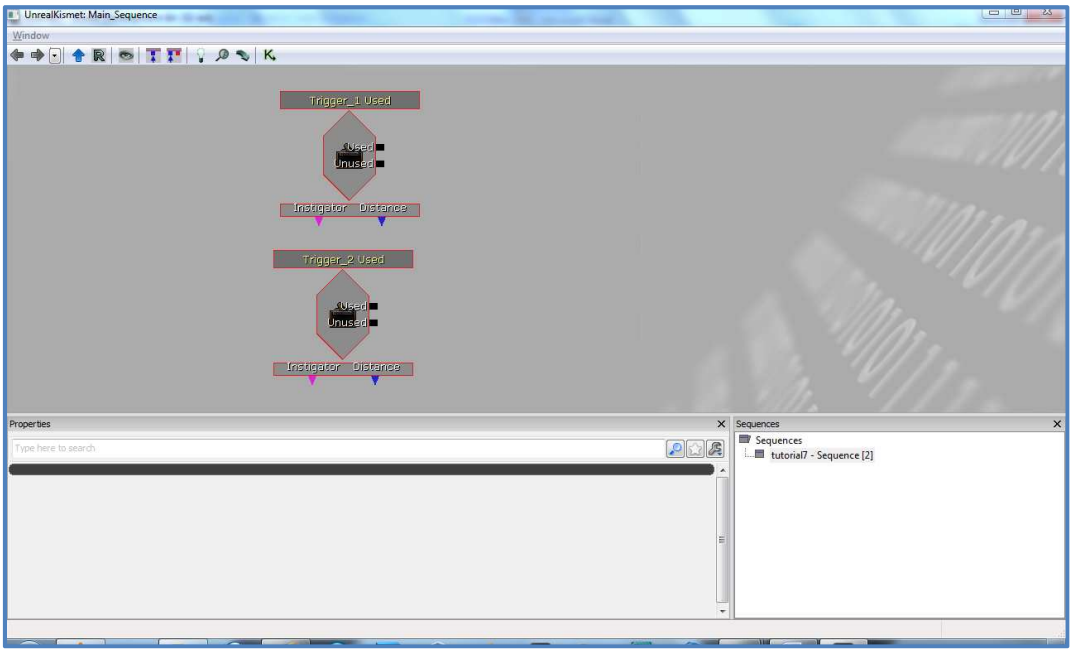


Imagen 291:Tutorial 7-Creación eventos en "Kismet"desde los trigger.

Hay que recordar en modificar la etiqueta *"Max Trigger Count"* si se quiere utilizar de manera infinita añadiendo el valor 0, ya que el valor por defecto es 1 de las propiedades de los eventos.

También para que sólo con estar en la zona de acción del *"trigger"* y presionar la tecla 'E' el teclado se lance el evento, es necesario no seleccionar la etiqueta *"Aim To Interact"* de las propiedades de los eventos.

2.El elemento que permite realizar el teletransporte del personaje es llamado *"Teleport"*. Hay que añadir este elemento *"Teleport"* por cada uno de los eventos añadidos en el punto anterior. Para ello:

Botón derecho del ratón → New Action → Actor → Teleport.

Cada uno de estos nuevos elementos se asociarán con la salida *"Used"* de cada uno de los eventos, mostrado en la imagen 292.

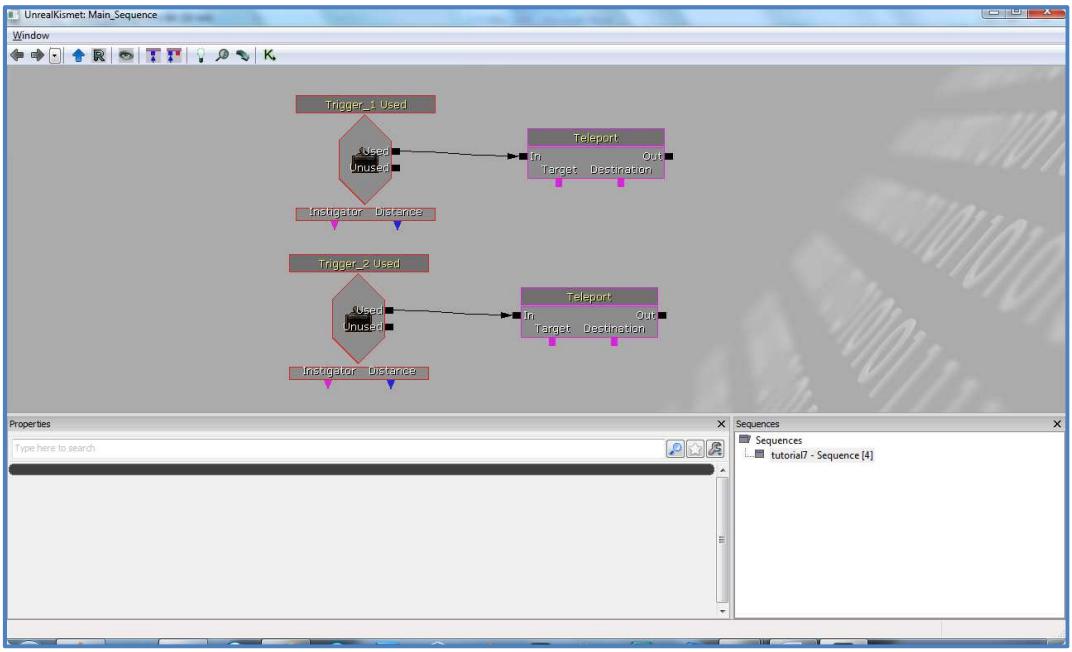


Imagen 292:Tutorial 7-Creación elementos "Teleport" en cada trigger.

3. En los elementos *"Teleport"* hay que asociar ó indicar el objeto que se quiere transportar. En este caso, se quiere teletransportar al jugador, personaje principal. Se necesitará una variable que represente al personaje principal:

   Botón derecho del ratón → New Variable → Player → Player.

   Esta variable debe representar sólo al jugador principal, con lo que hay que no seleccionar la etiqueta *"All Players"* de las propiedades.

   La variable añadida hay que asociarla con el *"Teleport"* mediante la entrada *"Target"* de este elemento, imagen 293.
   Se puede añadir una sola variable y asociarla con las dos entradas *"Target"* de los dos elementos ó crear tantas variables *"Player 0"* por cada elemento *"Teleport"* y asociarlas a las entradas *"Target"*. El resultado de las dos implementaciones es el mismo.

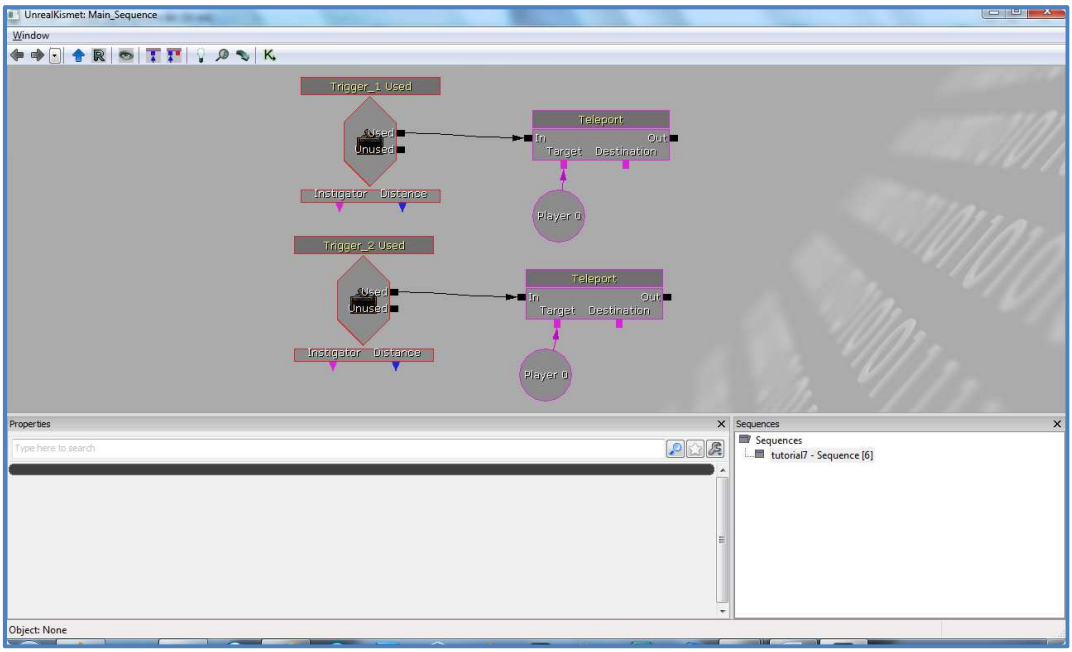


Imagen 293:Tutorial 7-Variables del jugador asociadas a "Teleport".

4. Al realizar el teletransporte hay que indicar los puntos de aparición en el nivel. Estos puntos de aparición deben ser especificados mediante objetos en el editor. Los objetos más comunes de utilizados son los llamados *"Note"*.

   Estos objetos *"Note"* se pueden encontrar en la herramienta *"Actor Classes"* la cual se puede abrir mediante el icono de la barra de herramientas superior del editor y seleccionando la etiqueta con el mismo nombre.

   Dentro del *"Actor Classes"*, este objeto se encuentra dentro de *"Actor"*, mostrada su localización en la imagen 294.

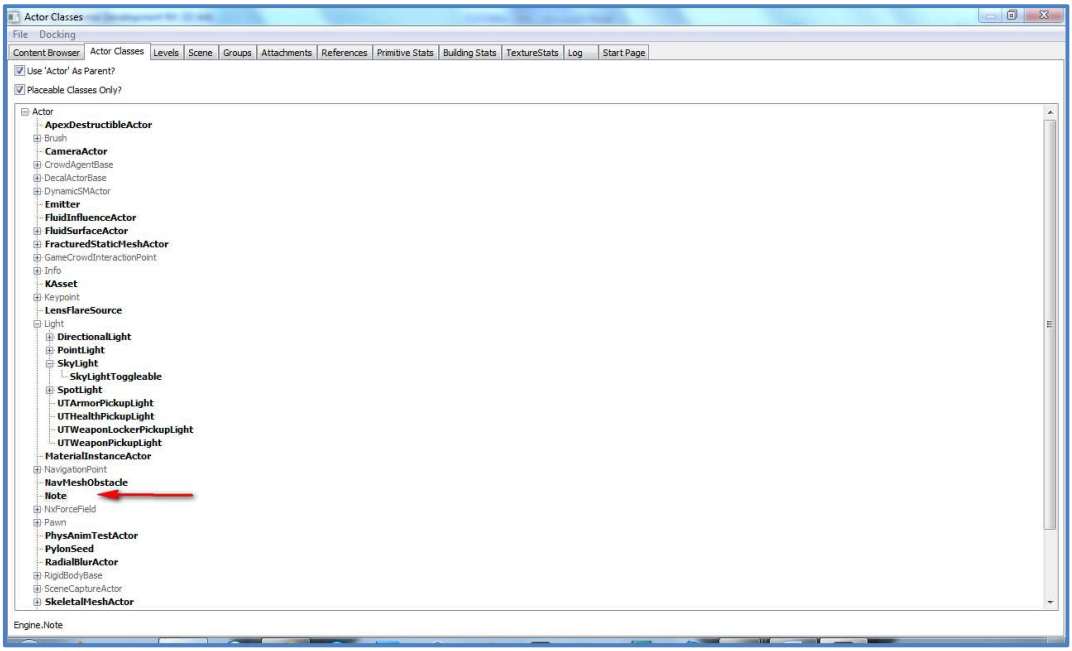


Imagen 294:Tutorial 7-Localización de "Note" en "Actor Classes".

Este objeto se acoplará al nivel seleccionándolo y volviendo al editor se debe pulsar el botón derecho del ratón en cualquier vista seleccionando *"Add Note Here"* en el menú desplegado

Se añadirá un objeto de tipo *"Note"* en cada una de las plataformas, mostrados en la imagen 295, en el lugar donde se quiera que reaparezca el personaje al ser teletransportado.

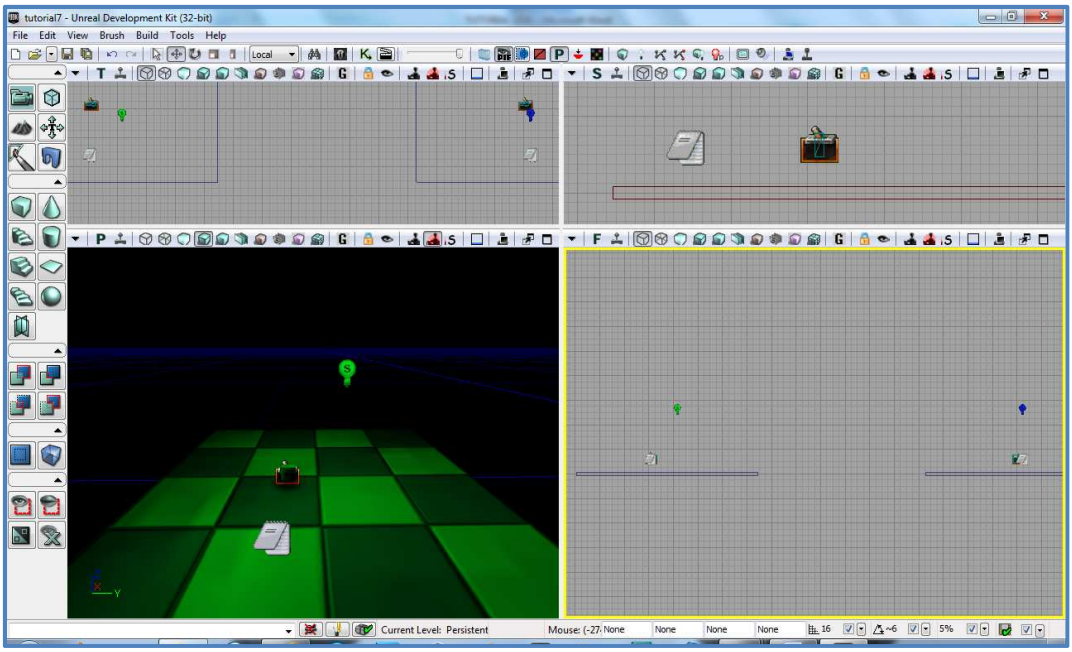


Imagen 295:Tutorial 7-Situación del objeto "Note" en el nivel.

En cada objeto "Note" se puede apreciar una flecha la cual representa el sentido en que aparecerá el personaje principal. Activando el *"Rotation Mode"*, , de la barra de herramientas principal del editor se puede girar esta flecha para obtener el sentido deseado.

5. Estos objetos *"Note"* se necesitan en la herramienta Unreal Kismet para poder asociarlos a los elementos *"Teleport"* e indicar el punto de reaparición del personaje.

   En el editor se deben seleccionar los dos objetos *"Note"* y volviendo a la herramienta Kismet se realiza:

   Botón derecho del ratón → New Object Vars using Note_x…

   Con esto se crearán las variables que representan a los objetos *"Note"* que se tienen en el editor. Cada una de estas variables debe ser asociada a las entradas *"Destination"* de los elementos *"Teleport"*, cada una donde corresponda para realizar el teletransporte correcto.

   Para saber el nombre de cada uno de los objetos que se tienen en el editor se pueden obtener las características de éstos, seleccionando el objeto y pulsar tecla 'F4' del teclado, mostrando en la parte superior el nombre del objeto. Sabiendo el nombre de cada uno de los objetos se puede asociar cada una de las variables *"Note"* con el elemento *"Teleport"* que le corresponde, como en la imagen 296.

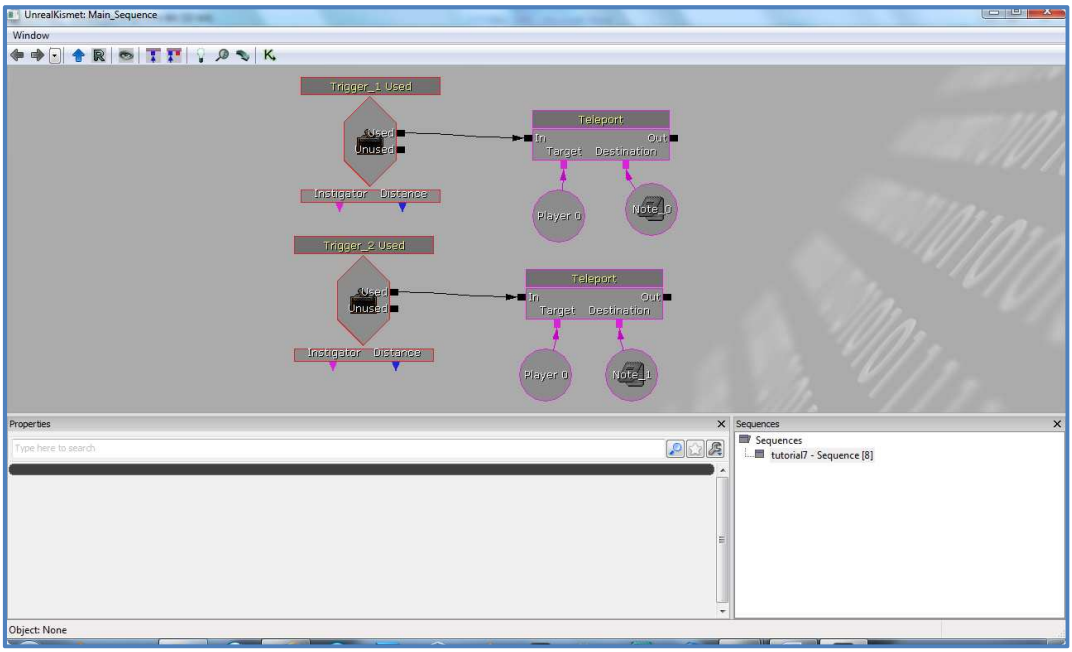


Imagen 296:Tutorial 7-Creación variables a partir objetos "Note".

6. Con la implementación realizada, cuando se haga uso del pulsador en la plataforma azul, el jugador será teletransportado a la plataforma verde, e inversamente en el caso contrario.

Con este tutorial se ha mostrado la manera de realizar un teletransporte del jugador de un punto a otro, pudiéndose realizar con cualquier acción ó evento lanzado en el nivel.

8.8. Tutorial 8: Mostrar animación de "Skeletal Mesh".

Los elementos *"Skeletal Mesh"* son un tipo especial de objetos, los cuales pueden poseer movimientos, ya que en su estructura poseen un *"esqueleto"* que permiten realizar animaciones en el objeto.

Estos *"Skeletal Mesh"* al ser importados en el motor UDK, pueden tener asociados ciertas animaciones, las cuales podrán ser reproducidas en la ejecución del nivel.

En el tutorial se va a mostrar cómo reproducir una animación de un *"Skeletal Mesh"* durante la ejecución del nivel.

Pasos a seguir:

1. En el nivel se tiene que añadir el *"Skeletal Mesh"* que se quiere utilizar y al que pertenece la animación.

   Todos los objetos *"Skeletal Mesh"* se encuentran en la herramienta Content Browser, pudiéndose acceder a ella mediante el icono de la barra de herramientas superior del editor.

   Seleccionando el objeto *"Skeletal Mesh"*, se vuelve al editor, en cualquier vista se pulsa con el botón derecho del ratón y en el menú mostrado hay que seleccionar:

   Add Actor → Add SkeletalMeshCinematic: Skeletal Mesh <nombre_objeto>

donde <nombre_objeto> es el nombre del "Skeletal Mesh" seleccionado.

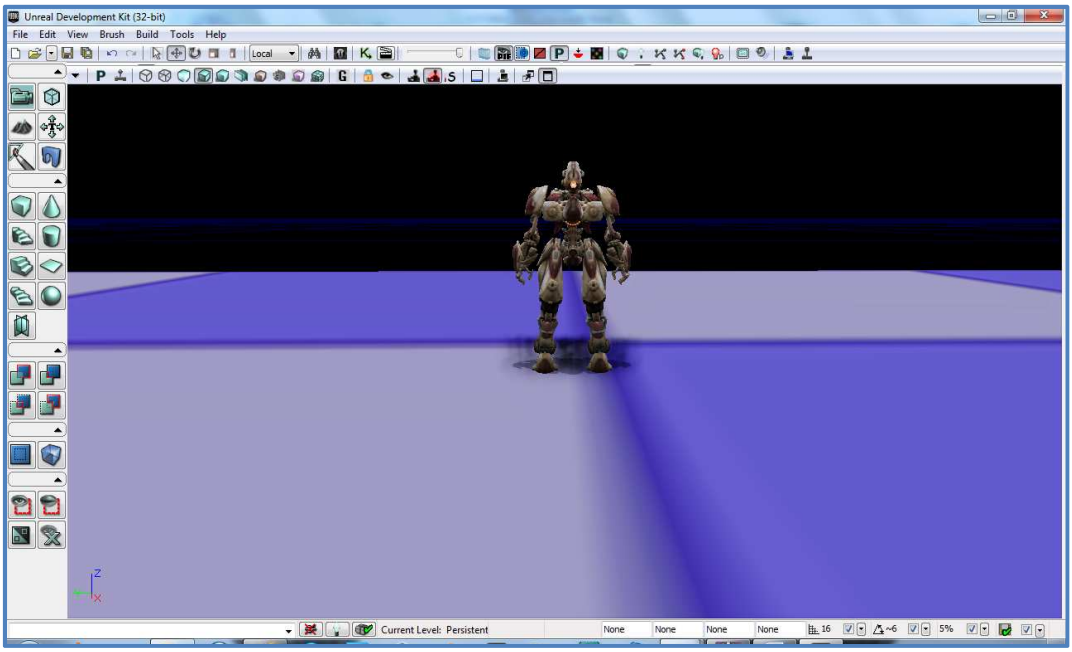


Imagen 297:Tutorial 8-Personaje acoplado al nivel.

Para el tutorial se va a utilizar el *"Skeletal Mesh"* llamado *"SK_CH_LIAM_Cathode"*, mostrado en la imagen 297, pudiéndose utilizar cualquier otro.

2. A un objeto añadido al nivel se le puede asociar otros objetos.
   En este caso, se va a asociar un objeto *"arma"* a una de las manos del *"Skeletal Mesh"* añadido.

   En el Content Browser se debe buscar el mesh *"arma"* que se quiere asociar y añadirlo al nivel. Para poner el objeto *"arma"* en un punto del *"robot"* se realiza lo siguiente:

   - Activar *"Enable/disable socket snapping"* mediante el icono de la barra de herramientas superior del editor.
   - Activando dicho icono aparecen distintos puntos en cada uno de los *"Skeletal Mesh"*, imagen 298.
     Estos puntos son en los cuales se puede asociar otro elemento.
   - Se selecciona el objeto arma y se pulsa con el ratón sobre el punto del *"robot"* donde se quiere asociar. En el caso del tutorial se va a asociar a una de las manos.

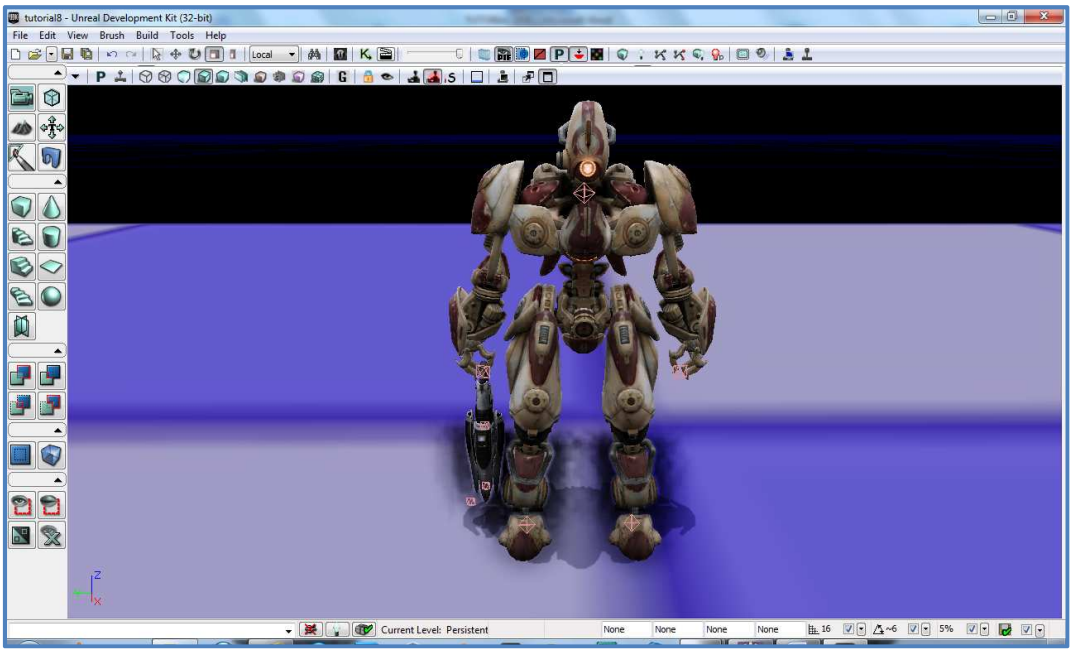


Imagen 298:Tutorial 8-Acople de objeto arma al personaje.

3. La animación del *"Skeletal Mesh"* se reproducirá mediante un elemento de la herramienta Matinee.

   Se debe abrir la herramienta Unreal Kismet mediante el icono de la barra de herramientas superior del editor con el *"robot"* seleccionado.
   Se debe crear un elemento de la herramienta Matinee mediante:

   Botón derecho del ratón → New Matinee.

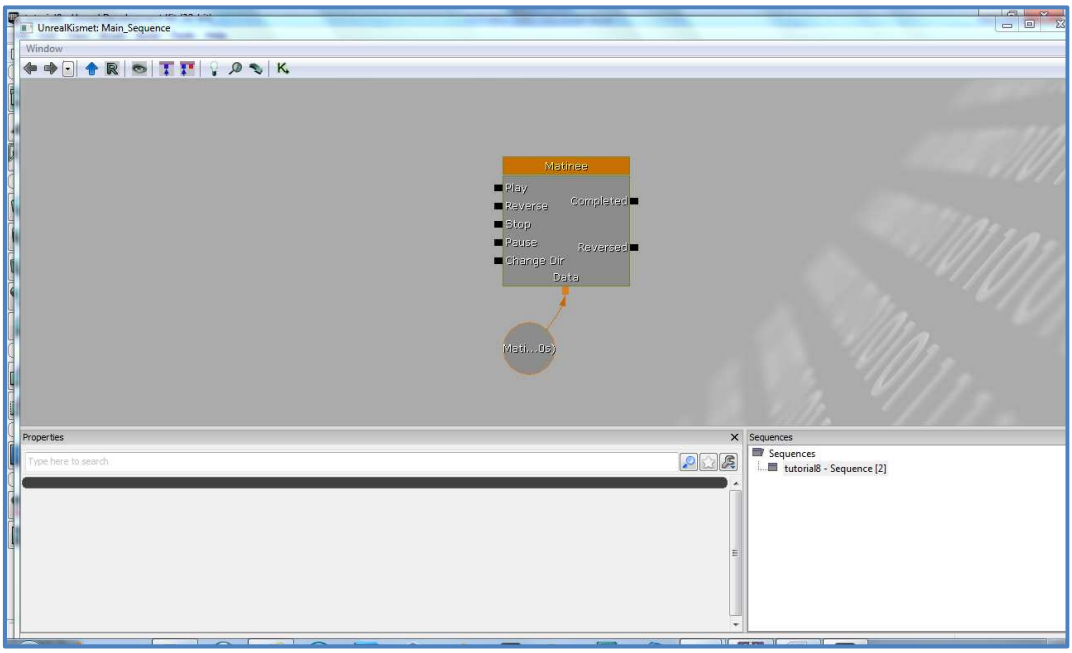


Imagen 299:Tutorial 8-Elemento "Matinee" en "Kismet".

4. Hay que modificar el elemento creado mediante el editor Unreal Matinee.
   Para abrir el editor se hace doble click sobre el elemento creado.

Se debe crear un nuevo grupo vacío en el que se le pondrá el nombre de *"animación"* mediante:

Botón derecho del ratón sobre zona izquierda →Add New Empty Group

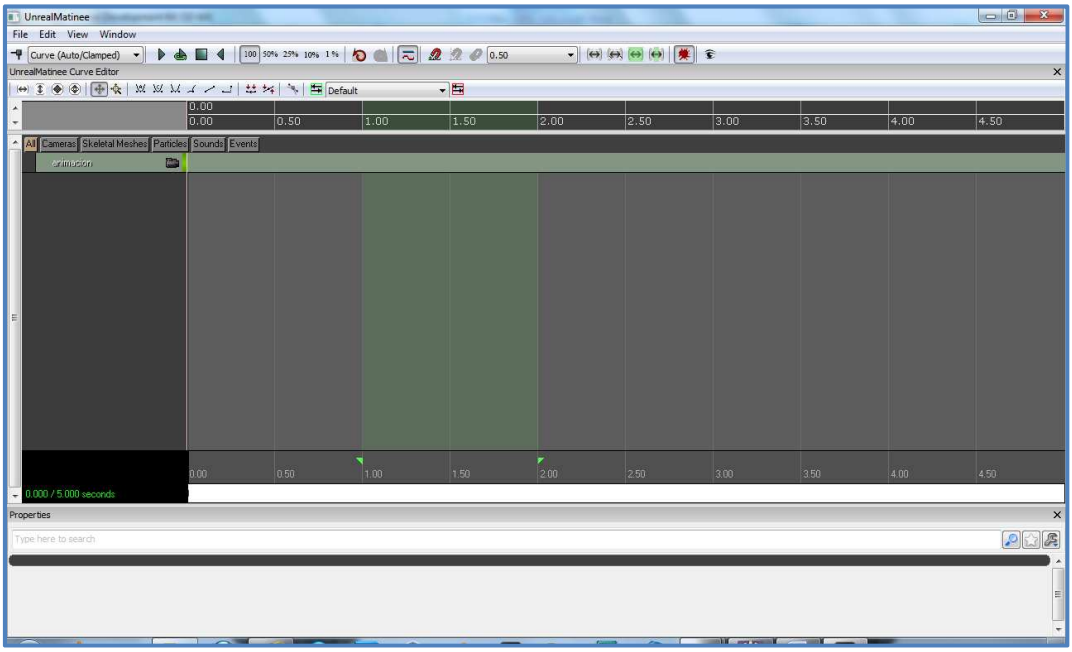


Imagen 300:Tutorial 8-Creación de grupo vacío en "Matinee".

5. En las propiedades del grupo creado hay que agregar el *"Anim Set"* que contiene la animación a realizar.

   Los *"AnimSet"* son objetos del motor que contienen animaciones de un *"Skeletal Mesh"*. Normalmente por cada uno de los *"Skeletal Mesh"* se tiene un objeto *"AnimSet"* que contiene sus animaciones.

   Al seleccionar el grupo creado, en la parte inferior se pueden observar las propiedades de éste, en las que hay una etiqueta llamada *"Group Anim Sets"* (imagen 301). En esta etiqueta hay que añadir un nuevo elemento pulsando en el icono apareciendo la posición [0]. En esta posición hay que añadir un objeto "AnimSet" seleccionándolo en el Content Browser y pulsando en el icono.

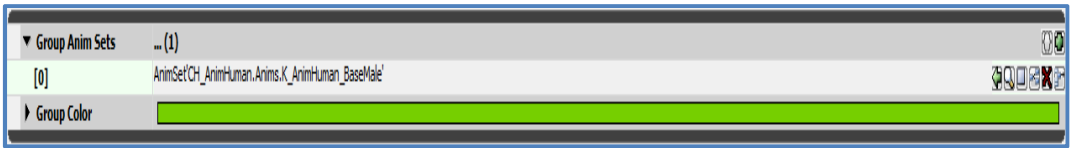


Imagen 301:Tutorial 8-Propiedades grupo.

En el tutorial se ha añadido el *"AnimSet"* llamado *"AnimSet'CH_AnimHuman.Anims.K_AnimHuman_BaseMale'"* que contiene las animaciones del *"robot"*.

6. En el grupo llamado *"animación"* se debe crear un track de tipo *"Anim Control "*. Con el grupo seleccionado se pulsa el botón derecho del ratón y en el menú desplegado se selecciona *"Add New Anim Control Track"*, mostrándolo en la imagen 302.

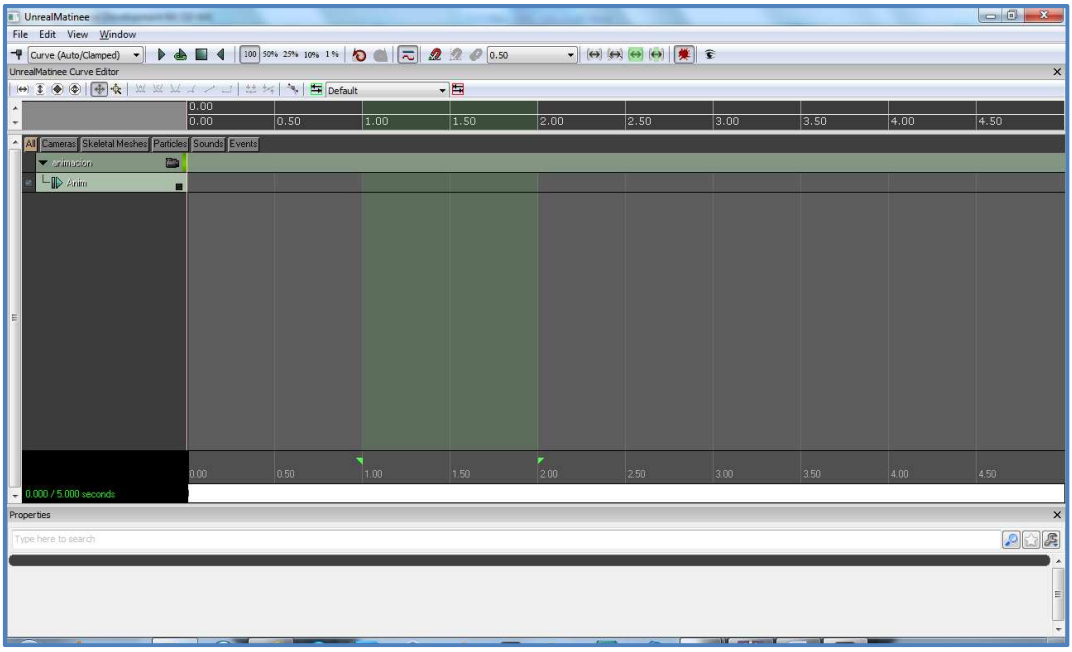


Imagen 302:Tutorial 8-Creación de track tipo "Anim Control" en el grupo.

7. En este track creado, al crear un punto de control, se pedirá que se seleccione la animación a mostrar.

   La línea temporal se situará en el tiempo deseado para crear el punto de control y pulsar la tecla 'Enter' del teclado.

   Se mostrará un menú donde se podrá elegir la animación a reproducir de todas las que contiene el *"AnimSet"*. El punto de control se representará con un triángulo rojo en la secuencia del track.

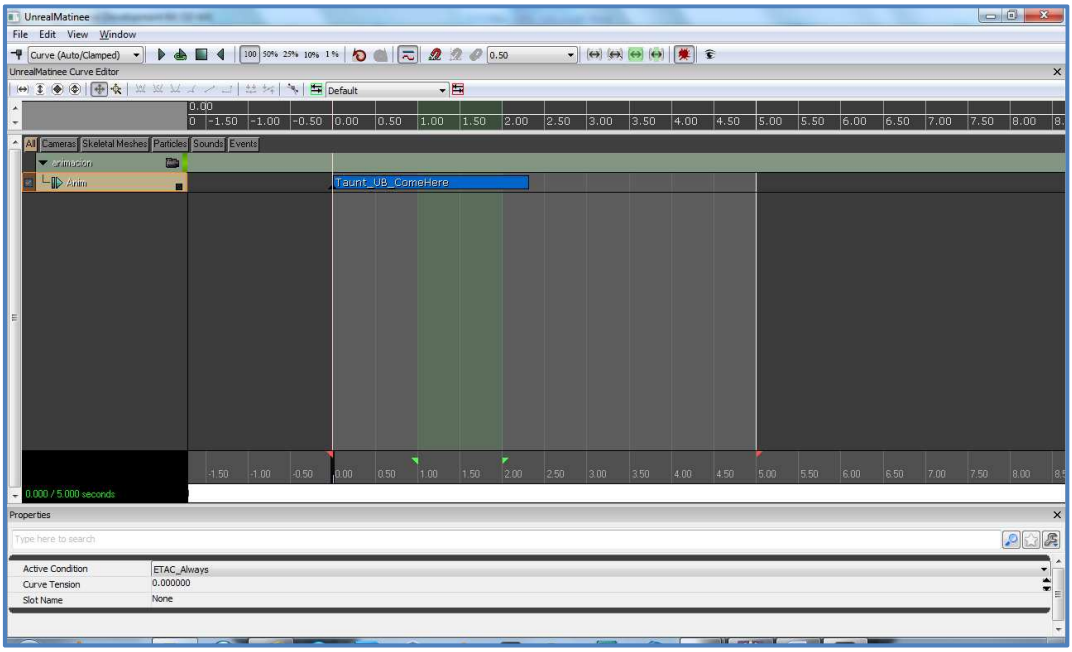


Imagen 303:Tutorial 8-Asociación de animación en el track.

En el tutorial se ha elegido la animación con nombre *"Taunt_UB_ComeHere"*, imagen 303, que se realizará un movimiento del brazo del robot indicando *"acércate"*.

8. La herramienta Unreal Matinee ya puede ser cerrada ya que no se va a realizar ningún cambio más.

   En el elemento de Kismet se tiene guardada toda la ejecución de la animación, la cual puede ser reproducida cuando se quiera. En el tutorial, se va a añadir un *"Trigger"* en el nivel de tipo *"contacto"*, haciendo que al estar próximo a él se reproducirá la animación del *"robot"*.

   Se añade el objeto *"Trigger"* en el nivel, pulsando el botón derecho del ratón sobre cualquier zona vacía:

   Botón derecho del ratón → Add Actor → Add Trigger.

   Este objeto tendrá que ser mostrado en la ejecución del nivel, con lo que se desactivará su etiqueta *Display → Hidden* de las propiedades (seleccionar objeto y pulsar tecla 'F4' del teclado).

   Con el objeto *"Trigger"* seleccionado volver a Kismet y realizar:

   Botón derecho del ratón → New Event using Trigger_x → Touch

   Mediante esta acción se tendrá el evento de contacto asociado al *"Trigger"* del nivel. Modificar la etiqueta de este evento *"Max Trigger Count"* para que se pueda utilizar indifinidamente, insertando el valor 0.

   Unir la salida *"Touched"* del evento *"Touch"* con la entrada *"Play"* del elemento Matinee para que al contactar con el *"Trigger"* la animación se reproduzca, tal y como se muestra en la imagen 304.

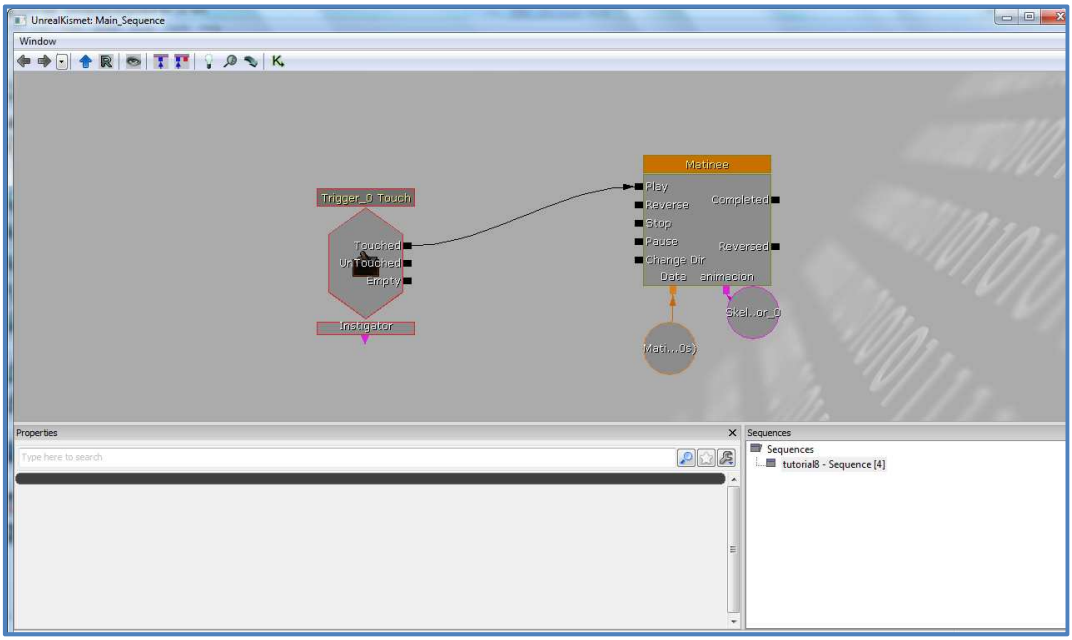


Imagen 304:Tutorial 8-Asociación elemento trigger con elemento "Matinee".

En el tutorial se ha utilizado un objeto *"Trigger"* para lanzar la ejecución de la animación, pero se puede utilizar cualquier evento ó acción del nivel para mostrarla.

También junto a cada animación se puede reproducir sonido, para hacer más impactante y realista su visualización.

8.9. Tutorial 9: Agregar personajes distintos al principal.

En el nivel se pueden agregar otros personajes además del personaje principal (jugador). Estos personajes son llamados *"actor"* y pueden ser agregados al nivel mediante un elemento especial de la herramienta Unreal Kismet llamado *"Actor Factory"*.

Mediante este elemento se pueden agregar *"enemigos"* que ataquen al jugador. En este tutorial se va a exponer la forma de agregar un "enemigo" en el nivel que comience a atacar al jugador.

Pasos a seguir:

1. Como se ha comentado anteriormente, el elemento que permite realizar la aparición de otros *"actor"* en el nivel pertenece a la herramienta Unreal Kismet.

   Se abrirá dicha herramienta mediante el icono de la barra de herramientas superior del editor y en cualquier zona despejada se realiza:

   Botón derecho del ratón → New Action → Actor → Actor Factory.

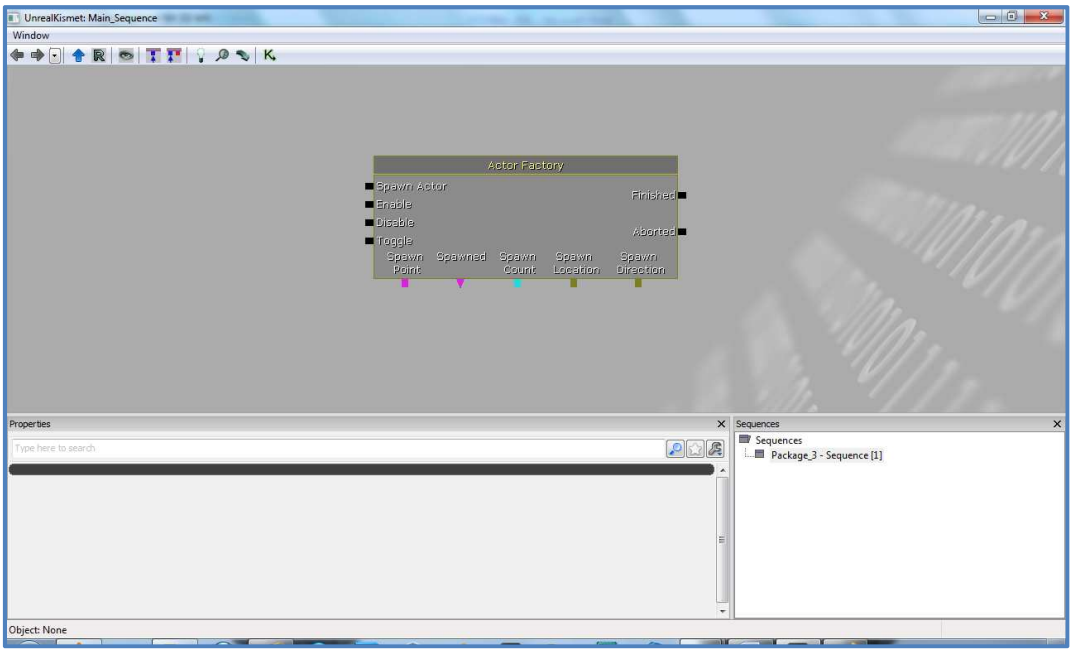


Imagen 305:Tutorial 9-Elemento "Actor Factory" en "Kismet".

2. En las propiedades de este elemento insertado hay que indicar el tipo de *"actor"* que se quiere acoplar. En la etiqueta con nombre *"Factory"* se debe indicar el tipo de *"actor"* a añadir. Para mostrar todos los tipos disponibles hay que pulsar sobre el icono de la etiqueta. Se mostrará una nueva ventana con todos los tipos de *"actor"*, tal y como se muestra en la imagen 306.

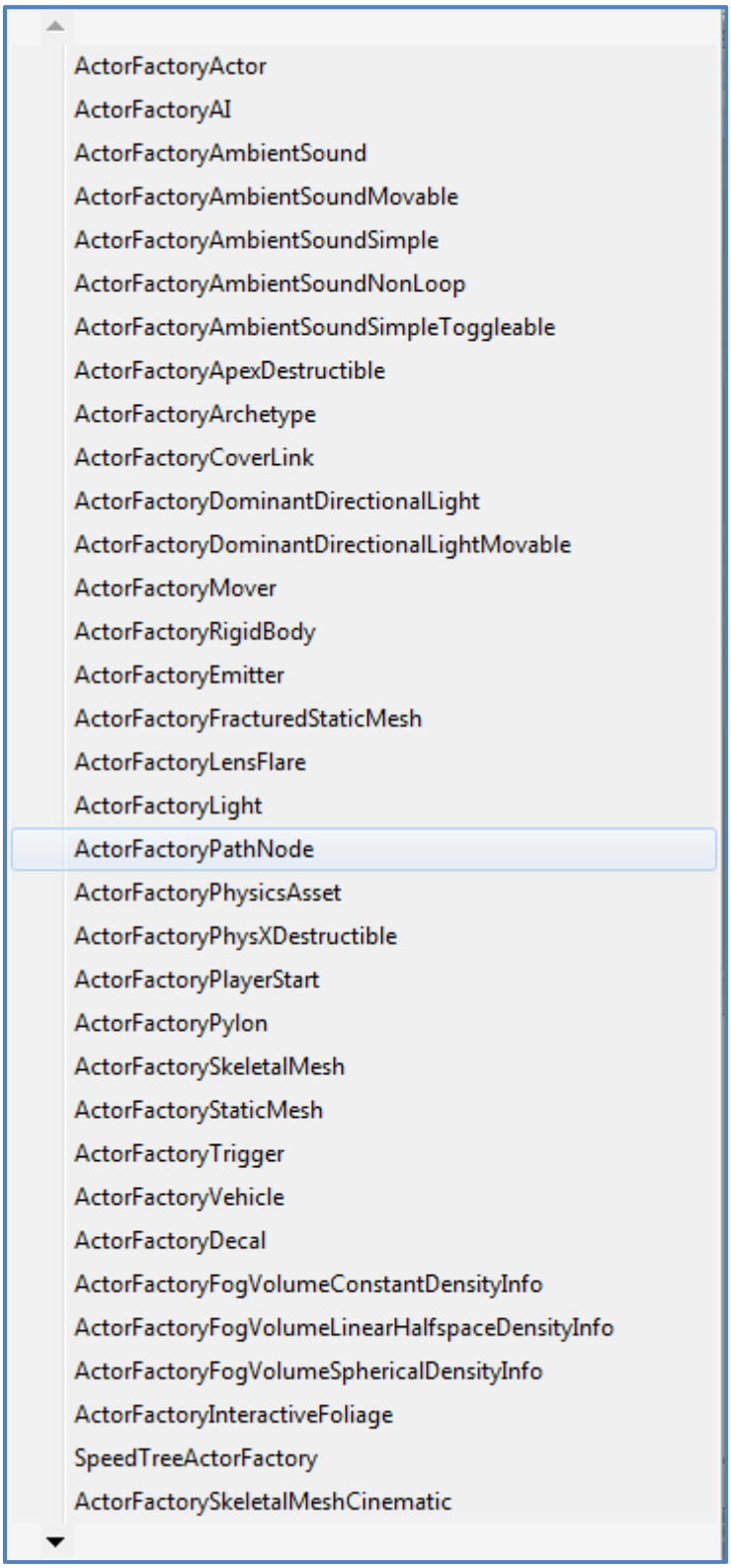


Imagen 306:Tutorial 9-Tipos de "Actor".

3. En este tutorial se está mostrando cómo acoplar un *"enemigo"*, con lo que el tipo que hay que seleccionar es *"UTActorFactoryAI"*.

   Al elegir este tipo se mostrarán nuevas propiedades en la etiqueta *"Factory"* propias del tipo elegido:

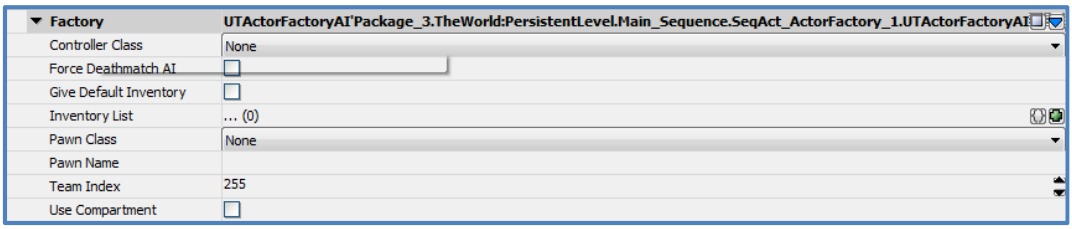


Imagen 307:Tutorial 9-Etiqueta "Factory" del elemento "Actor Factory".

Mediante la etiqueta *"Force Deathmatch AI"* se puede activar que el *"actor"* agregado comience a atacar al jugador, con lo que se activará dicha etiqueta.

En la etiqueta *"Give Default Inventory"* se podrá activar/desactivar la opción que el personaje contenga el inventario de armas por defecto o no. En este caso, como va a ser un *"enemigo"* que va a atacar al jugador se activará esta opción.

En *"Inventory List"* se pueden añadir objetos al inventario del personaje a agregar. Para agregar un elemento a la lista hay que pulsar en el icono añadiendo una posición, y elegir el objeto en la lista desplegable mostrada. En *"Pawn Class"* se debe elegir la clase que contiene toda la dinámica del personaje, como sus movimientos, apariencia, etc. La clase que se va a utilizar es *"UTPawn"* que es la que contiene todas las características de un *"robot"*.

Las restantes características son para añadir el nombre que se le quiere dar al *"actor"* agregado y el equipo al que se quiere que pertenezca.

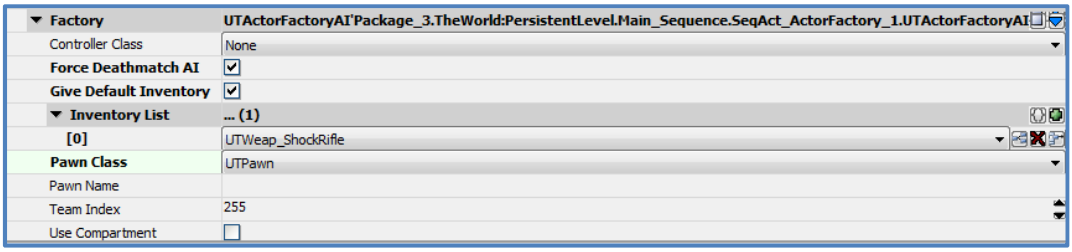


Imagen 308:Tutorial 9-Modificación etiquetas en "Actor Factory".

4. Una propiedad importante también en este elemento es la llamada *"Spawn Count"* en donde se debe indicar el número de *"actor"* que se quiere incorporar al nivel.

5. Una vez que se ha definido el tipo de *"actor"* que se va a incorporar al nivel, hay que indicar el punto donde se quiere que aparezca, imagen 309. Para ello hay que volver al editor y en cualquier vista se realiza:

   Botón derecho del ratón → Add Actor → Add PathNode.

   Realizando esta acción se tendrá un nuevo objeto en el nivel que será el punto donde aparecerá el *"actor"*. Este objeto podrá ser trasladado activando el icono , *"Traslate Mode"*, para colocarlo en el lugar deseado.

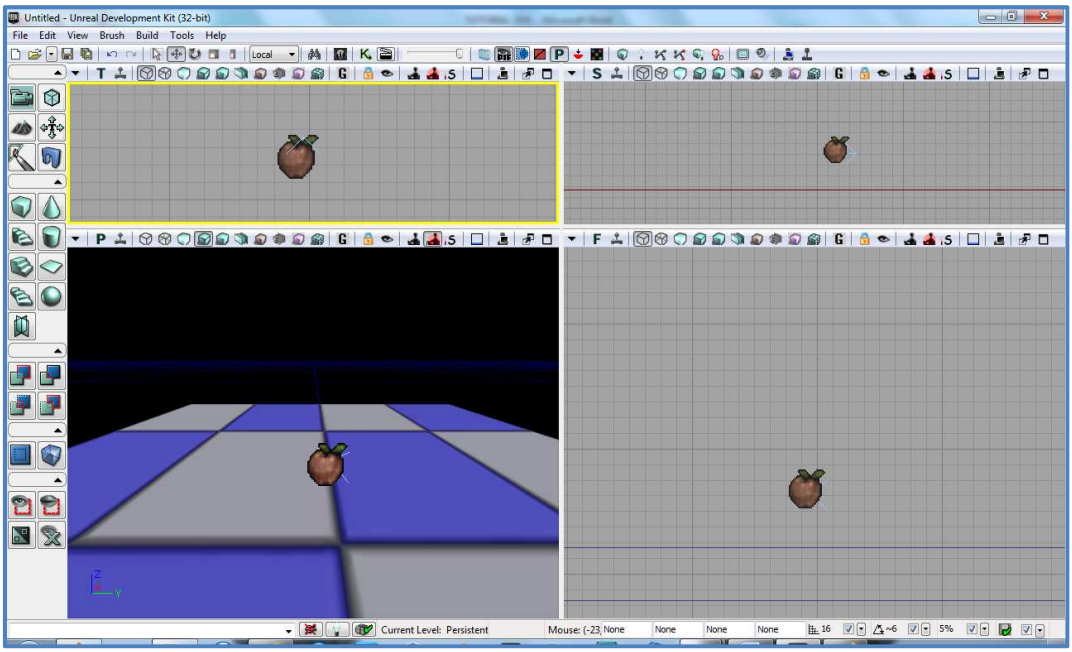


Imagen 309:Tutorial 9-Situación elemento "PathNode" en nivel.

En el objeto nuevo creado se puede apreciar una flecha horizontal, que representa el sentido que tendrá el personaje al aparecer. Dicho sentido puede ser girado mediante el *"Rotation Mode"* de la barra de herramientas superior.

6. De vuelta a la herramienta Unreal Kismet, es necesario tener una variable que represente al objeto *"PathNode"* que indica el lugar de aparición del personaje. Para obtener dicha variable se realiza:

   Botón derecho del ratón → New Object var using pathnode _x.

   Esta variable se asociará al elemento *"Actor Factory"* mediante la entrada *"Spawn Point"* ofrecida por éste, tal y como se muestra en la imagen 310.

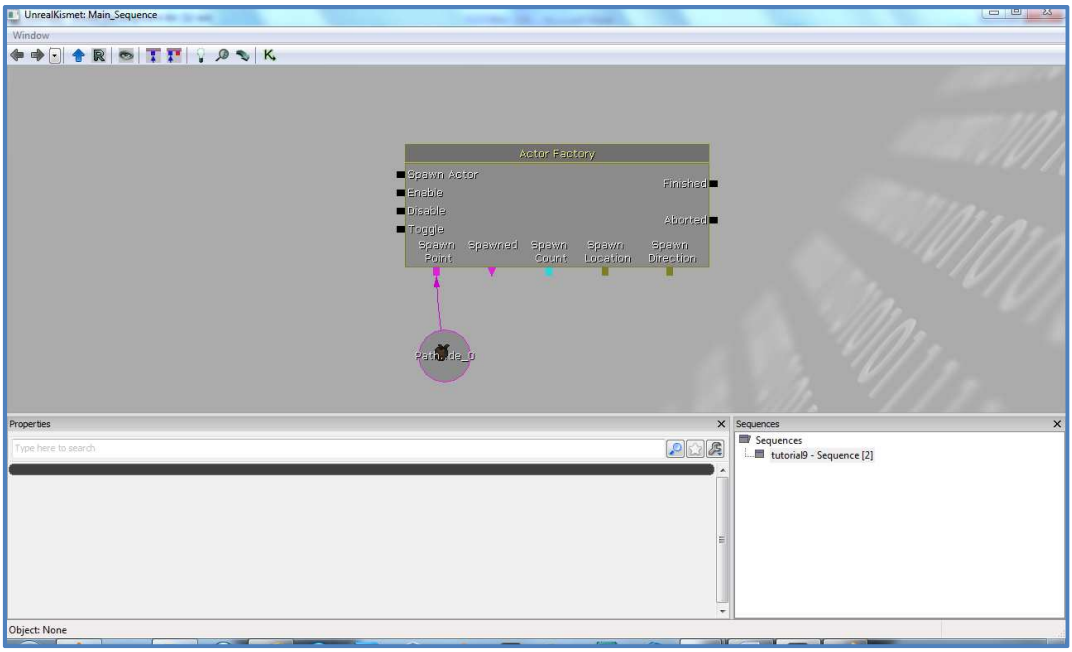


Imagen 310:Tutorial 9-Variable representativa del objeto "PathNode".

7. Mediante lo realizado anteriormente, aparecerá un *"robot"* enemigo en el punto indicado. Este elemento habría que asociarlo con otros elementos al flujo de acciones pudiendo ser cualquier evento ó acción implementada.

   En este tutorial se quiere que nada más lanzar el nivel, el *"enemigo"* aparezca y empiece a atacar al jugador. Para ello se va a utilizar el evento que representa el inicio del nivel, llamado *"Level Loaded"* uniéndose con la entrada *"Spawn Actor"* del elemento *"Actor Factory"*:

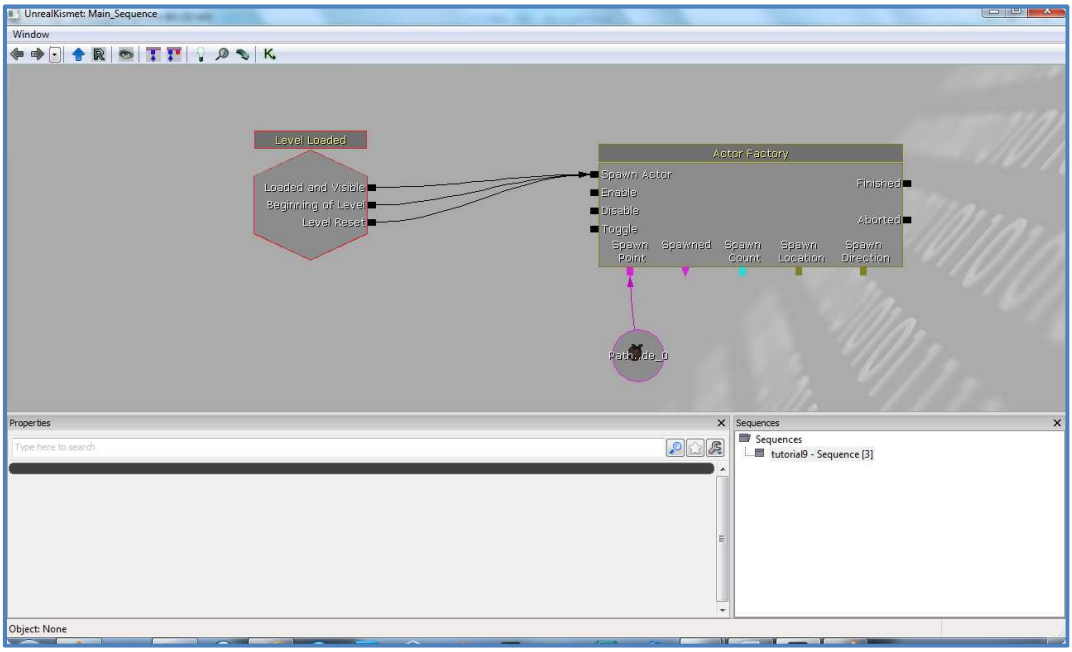


Imagen 311:Tutorial 9-Asociación "Level Loaded" con "Actor Factory".

   Se debe tener en cuenta el modificar la etiqueta *"Max Trigger Count"* al valor de 0 para poder utilizarlo indefinidamente, porque el valor por defecto de esta etiqueta es 1 y sólo se podría utilizar una sola vez.

Mediante este tutorial se ha mostrado el poder agregar un *"enemigo"* al nivel. Pero esto es sólo un ejemplo de las posibilidades que tiene el elemento *"Actor Factory"* ya que se podría agregar al nivel cualquier tipo de *"actor"*.

8.10. Tutorial 10: Controlar cuándo un personaje muere.

Una de las acciones que se pueden llegar a querer controlar es cuándo un *"enemigo"* o cualquier otro personaje *"muere"*, para poder realizar distintas acciones o lanzar ciertos eventos.

En este tutorial se van a mostrar los pasos a seguir para controlar en la implementación del nivel cuándo un personaje *"enemigo"* ha muerto.

Para este tutorial se da por hecho tener conocimientos de cómo añadir un personaje al nivel mediante el elemento *"Actor Factory"* de la herramienta Unreal Kismet. En el caso de no saber cómo realizarlo, ver Tutorial 9.

Se va a partir de lo implementado en el *"Tutorial 9"* para seguir con este tutorial.

Pasos a seguir:

1. Es necesario almacenar en una variable global el personaje que aparece en el nivel. La variable va a ser de tipo *"Object"* ya que debe ser una variable global y lo mostrado puede ser cualquier tipo de *"actor"*. Para crear esta variable global realizar:

   Botón derecho del ratón → New Variable →Object → Object.

   La variable creada debe ser asociada con el elemento *"Actor Factory"* con su salida *"Spawned"*, tal y como se muestra en la imagen 312.

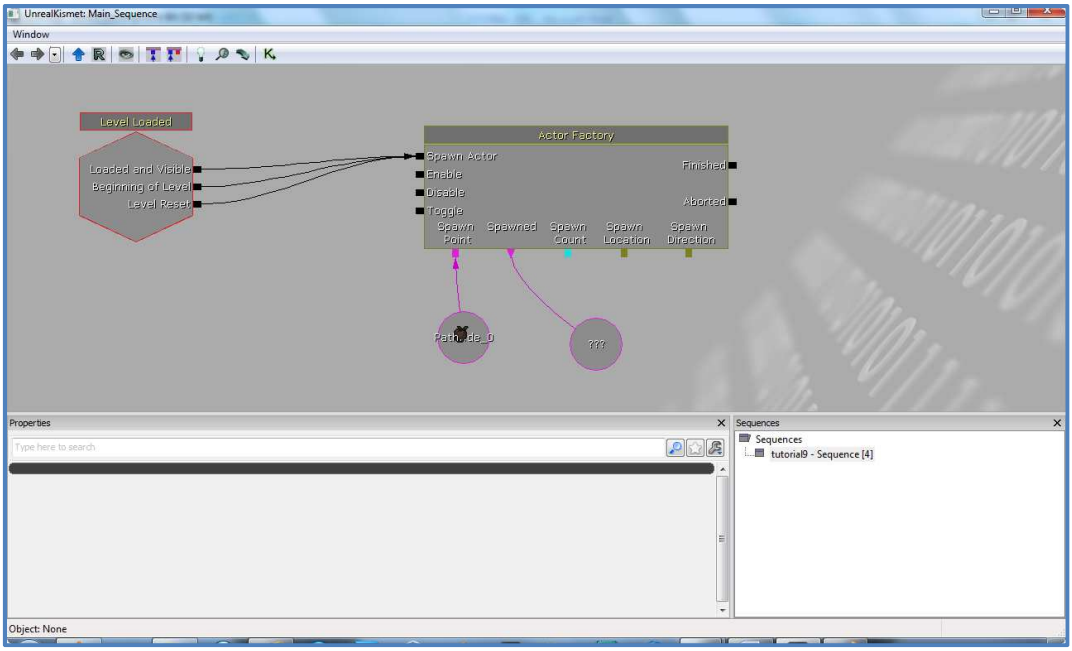


Imagen 312:Tutorial 10-Variable de tipo "Object"asociada a "Actor Factory".

2. El evento que se quiere controlar es cuándo un personaje añadido al nivel muere. El elemento que representa la muerte de un personaje es llamado *"Death"*, y se puede añadir este evento de la manera:

   Botón derecho del ratón → New Event → Pawn → Death.

   La entrada con nombre *"Instigator"* hay que unirla con la variable global creada en el punto anterior, mostrado en la imagen 313.

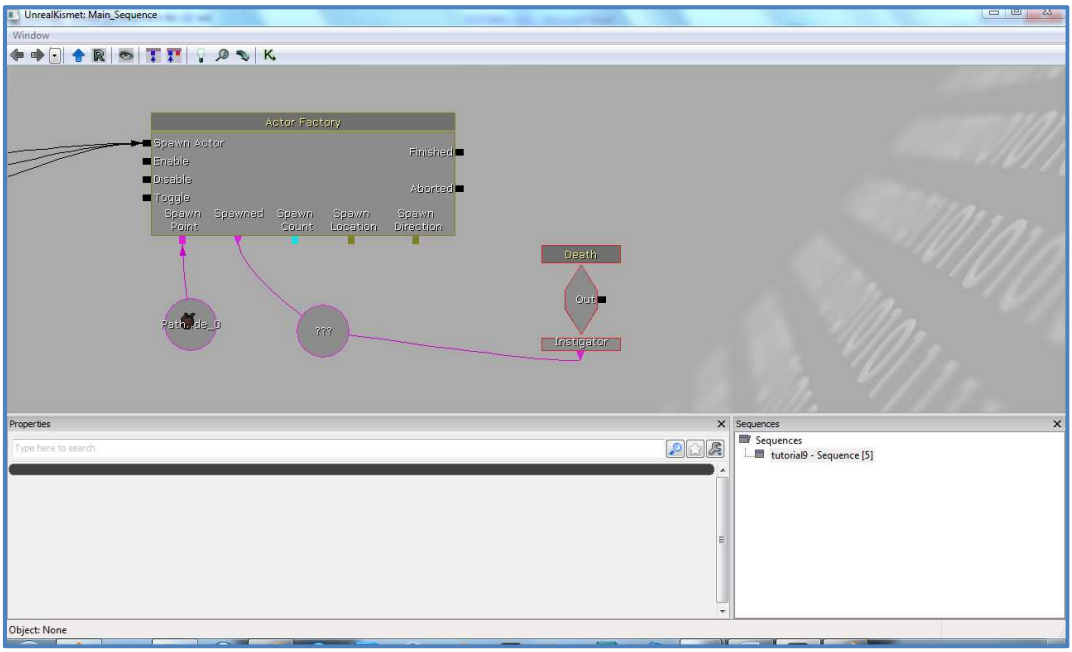


Imagen 313:Tutorial 10-Creación evento "Death".

3. Para terminar de controlar el evento es necesario un elemento que asocie la variable con el evento. Mediante esta acción se asocia el personaje con el evento, haciendo que el evento de muerte haga referencia a la vida del personaje.

   El elemento que permite realizarlo es llamado *"Attach To Event"* y se puede insertar de la manera:

   Botón derecho del ratón → New Action → Event → Attach To Event.

   En las entradas de este nuevo elemento hay que asociar la variable global, en entrada *"Attachee"*, y el evento, en la entrada *"event"*:

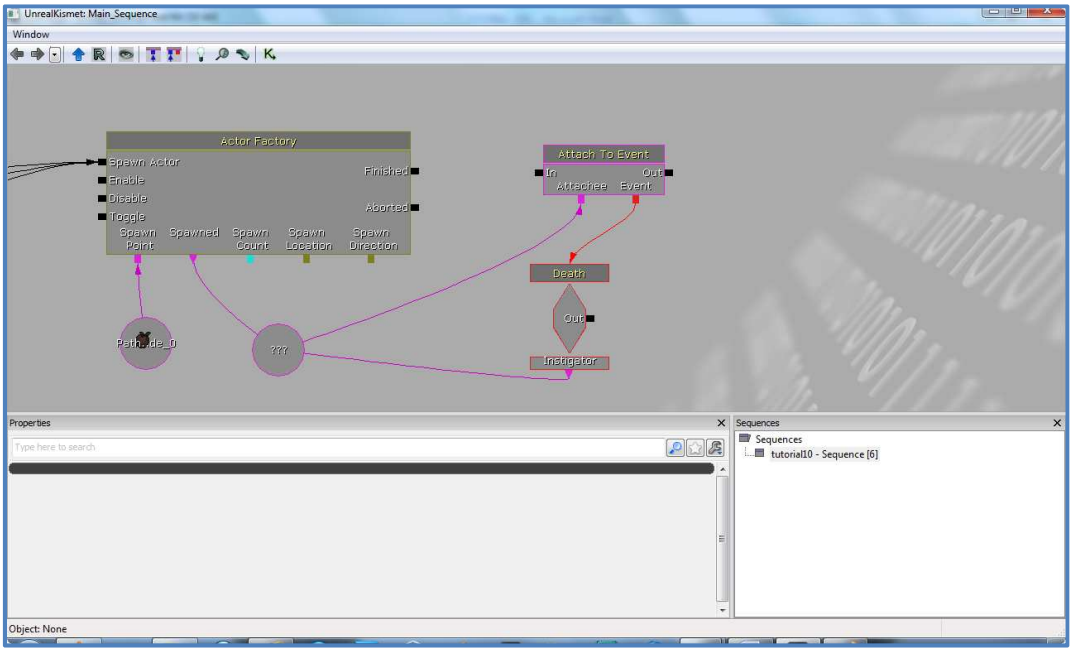


Imagen 314:Tutorial 10-Acople elemento "Attach to Event" y asociarlo.

4. Para que la secuencia sea completa y se realicen todas las acciones, hay que unir la salida del elemento *"Actor Factory"* llamada *"Finished"* con la entrada del elemento *"Attach to Event"* llamada *"In"*.

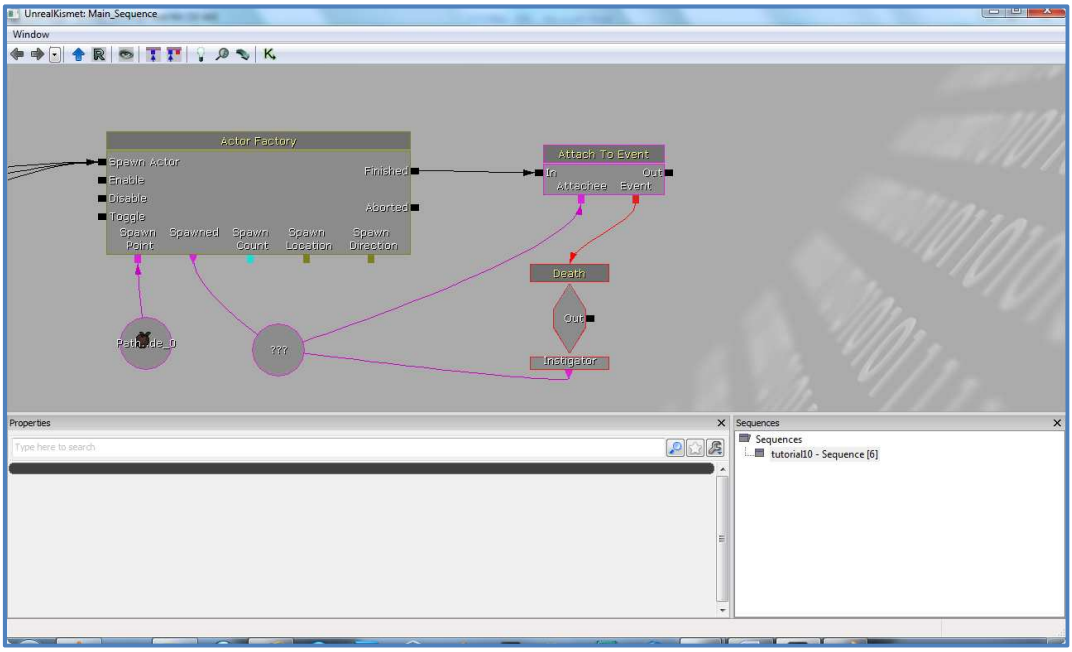


Imagen 315:Tutorial 10-Asociación "Attach to Event" con "Actor Factory".

5. La secuencia de acciones que se añada a partir de la salida *"Out"* del evento *"Death"* será la que se realice cuando el personaje mostrado en el nivel muera.

   En el tutorial simplemente se mostrará un mensaje por pantalla indicando que ha muerto el enemigo. Para ello se va a utilizar un elemento *"Play Announcement"* en el que insertará el mensaje de texto *"EL ENEMIGO HA MUERTO"*. Este tipo de elemento se puede conseguir mediante:

   Botón derecho del ratón → New Action → Voice/Announcement → Play Announcement.

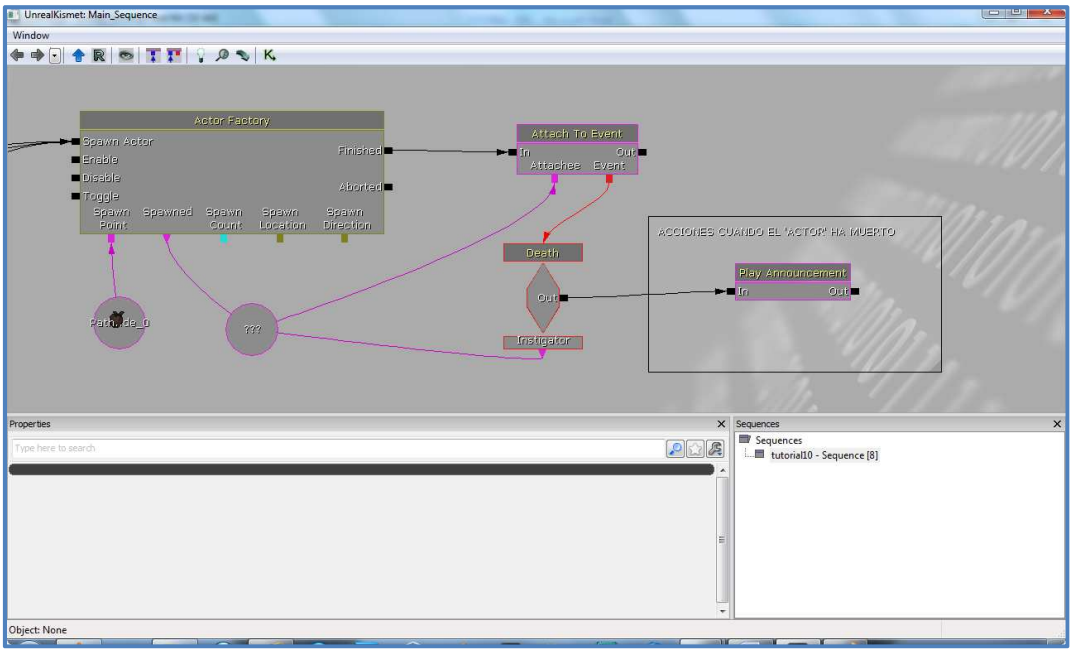


Imagen 316:Tutorial 10-Ejemplo de implementación a la muerte del "Actor".

8.11. Tutorial 11: Personaje se traslade de un punto a otro.

Cuando un personaje (actor) es acoplado al nivel, puede quererse que se mueva de un punto a otro. Unreal Editor permite realizar esta acción de los personajes con una pequeña implementación en la herramienta Unreal Kismet.

Para realizar el tutorial se parte de un escenario básico con una plataforma y un muro en medio de ella, mostrado en la imagen 317.

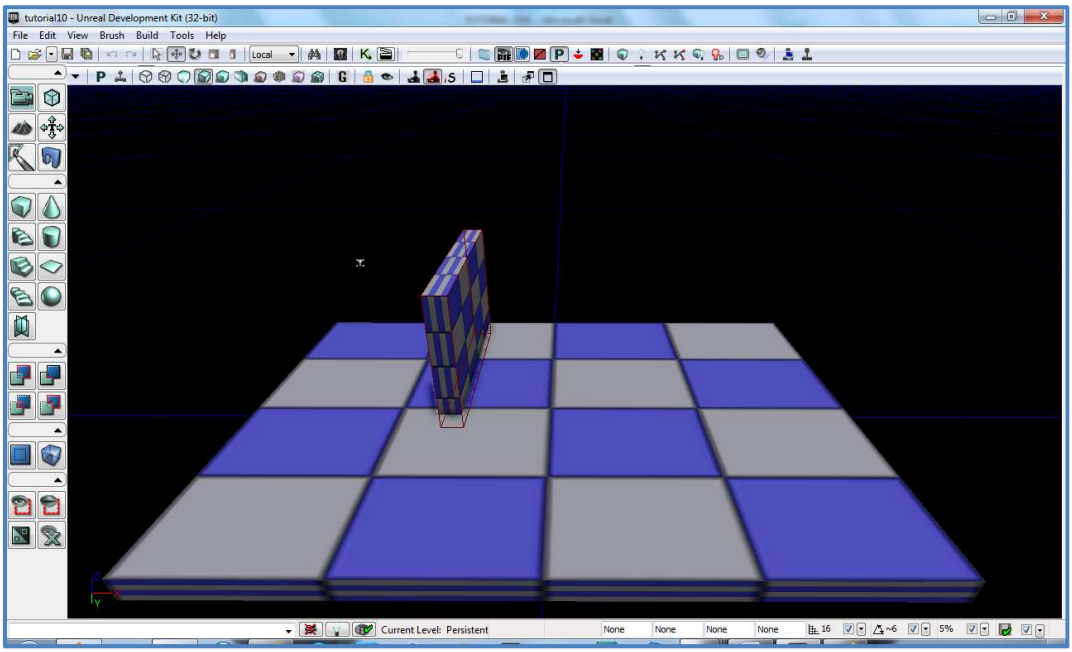


Imagen 317:Tutorial 11-Nivel inicial.

Pasos a seguir:

1. En la herramienta Unreal Kismet se deben añadir los elementos necesarios para que aparezca un nuevo personaje en el nivel, mostrados en la imagen 318.
   En el caso de no saber cómo realizarlo, ver el Tutorial 9.

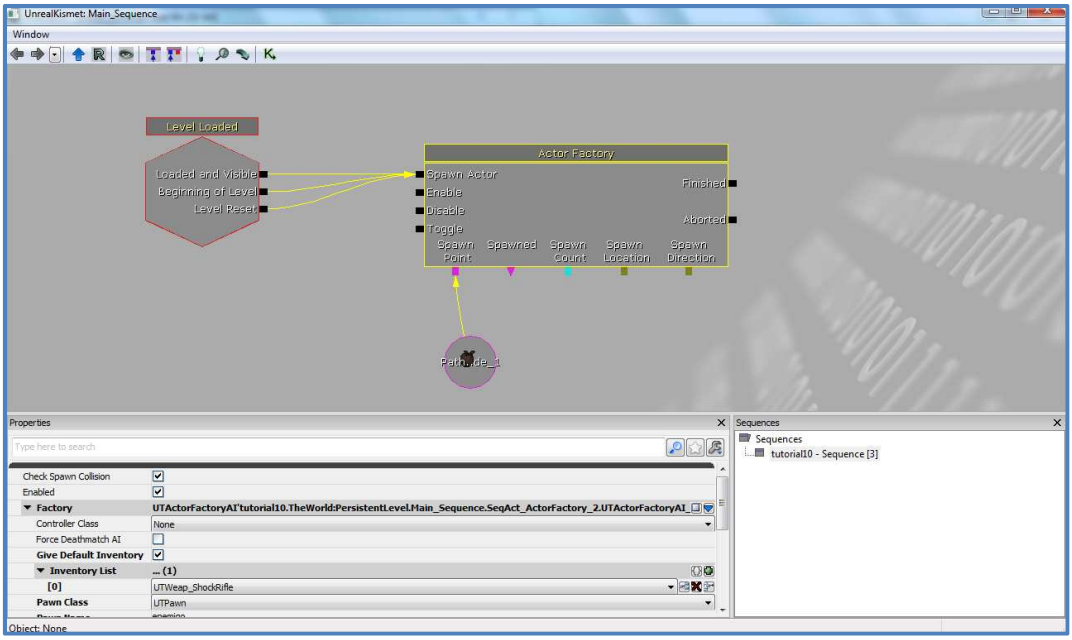


Imagen 318:Tutorial 11-Aparición nuevo personaje en nivel.

   En *"Actor Factory"*, en sus propiedades, se desactivará la etiqueta *"Force Deathmatch AI"* ya que no se quiere que el personaje ataque al jugador, sólo queriéndose comprobar su movimiento.

2. Para que el personaje se traslade de un punto a otro es necesario acoplar al nivel otro objeto *"PathNode"* que representará el otro punto entre los que se trasladará el personaje. En cualquier vista del editor:

   Botón derecho del ratón → Add Actor → Add PathNode.

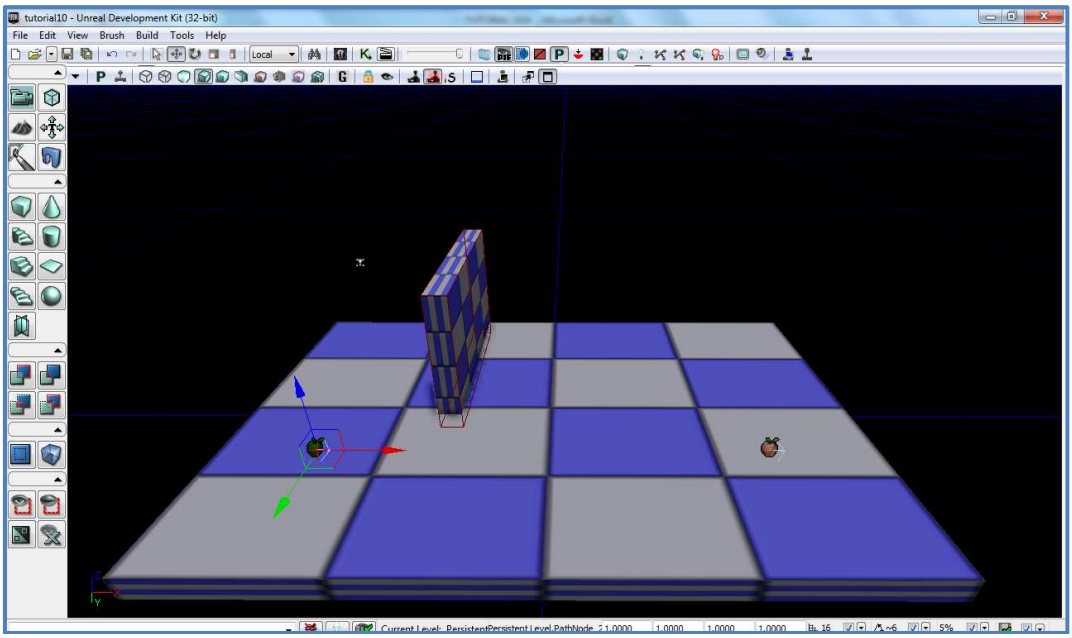


Imagen 319:Tutorial 11-Situación elementos "PathNode" en nivel.

3. En Unreal Kismet es necesario tener una variable global que represente al personaje que se ha añadido al nivel. Para obtener una variable global hay que realizar:

   Botón derecho del ratón → Add Variable → Object → Object.

   Para que el personaje añadido al nivel se almacene en esta variable global, hay que unirla con la salida *"Spawed"* del elemento *"Actor Factory"*, tal y como se muestra en la imagen 320.

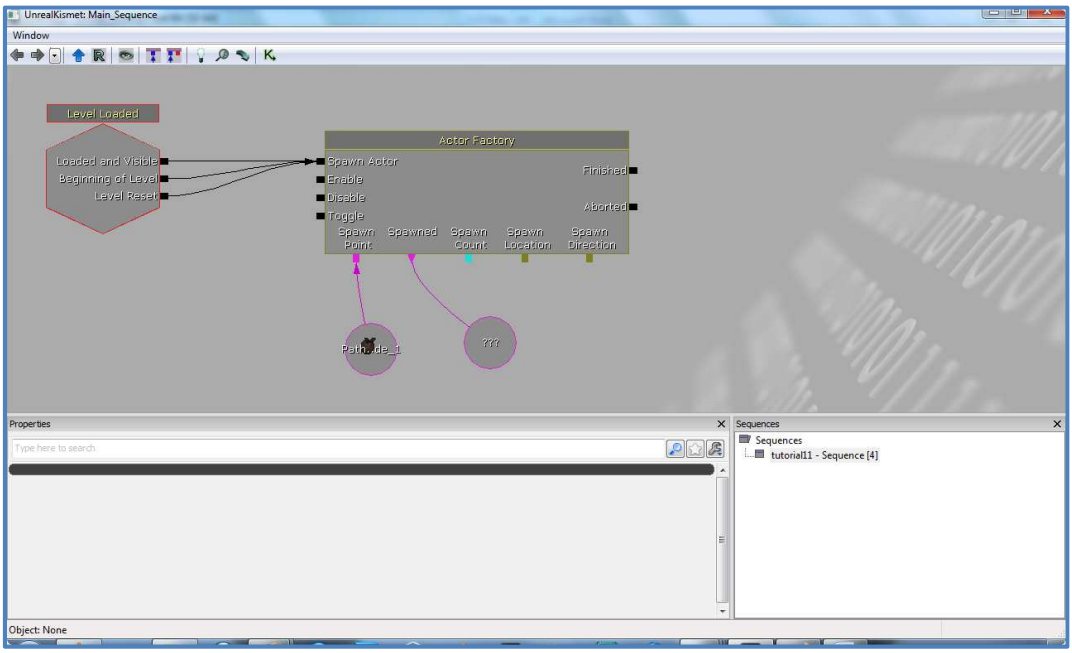


Imagen 320:Tutorial 11-Variable "Object" asociada a elemento "Actor Factory".

4. Ahora hay que realizar la implementación para que el personaje se traslade al otro punto. El elemento que permite realizar esta acción es llamado *"Move To Actor"* y puede añadirse mediante:

Botón derecho del ratón → New Action → AI → Move To Actor.

Se asociará con el elemento *"Actor Factory"* para seguir con la secuencia de acciones.

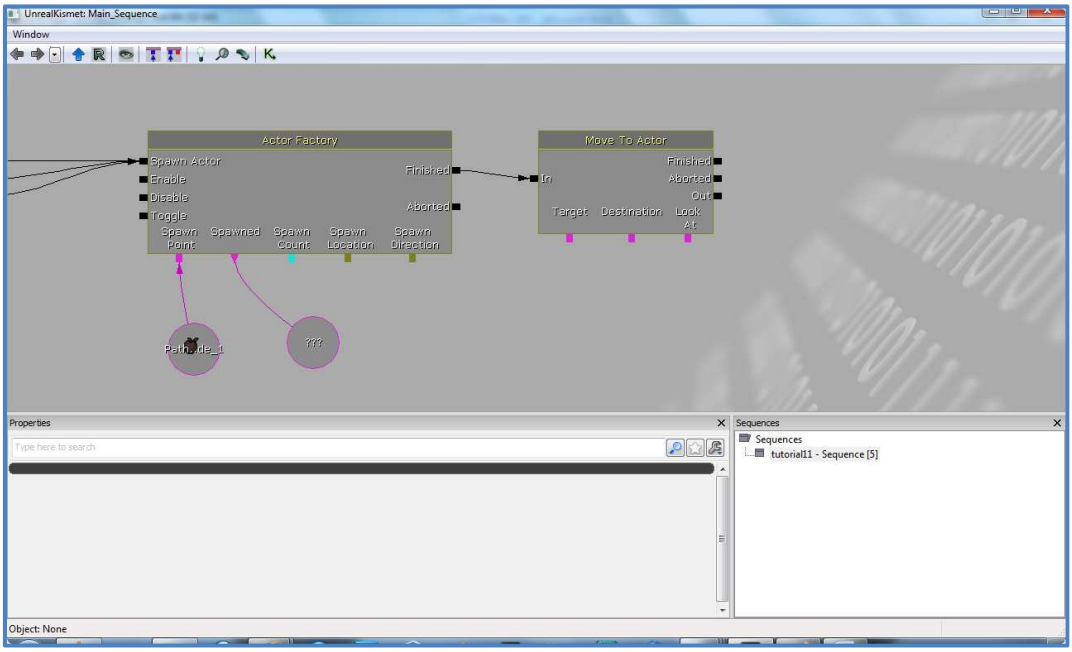


Imagen 321:Tutorial 11-Creación elemento "Move To Actor".

5. En el nuevo elemento añadido se indica el objeto el cual mover, en este caso la variable global que contiene la referencia al personaje añadido. La variable global hay que asociarla con la entrada *"Target"* del elemento *"Move To Actor"*.

   A su vez, hay que indicar el punto al que se quiere que se traslade el personaje. Para ello se necesita una variable que represente a este punto. En el editor, con el segundo *"PathNode"* seleccionado, volver al Kismet y pulsar el botón derecho del ratón seleccionando *"New Object Var using PathNode_x"*.

   Esta variable creada se asocia con la entrada *"Destination"* del elemento *"Move To Actor"*.

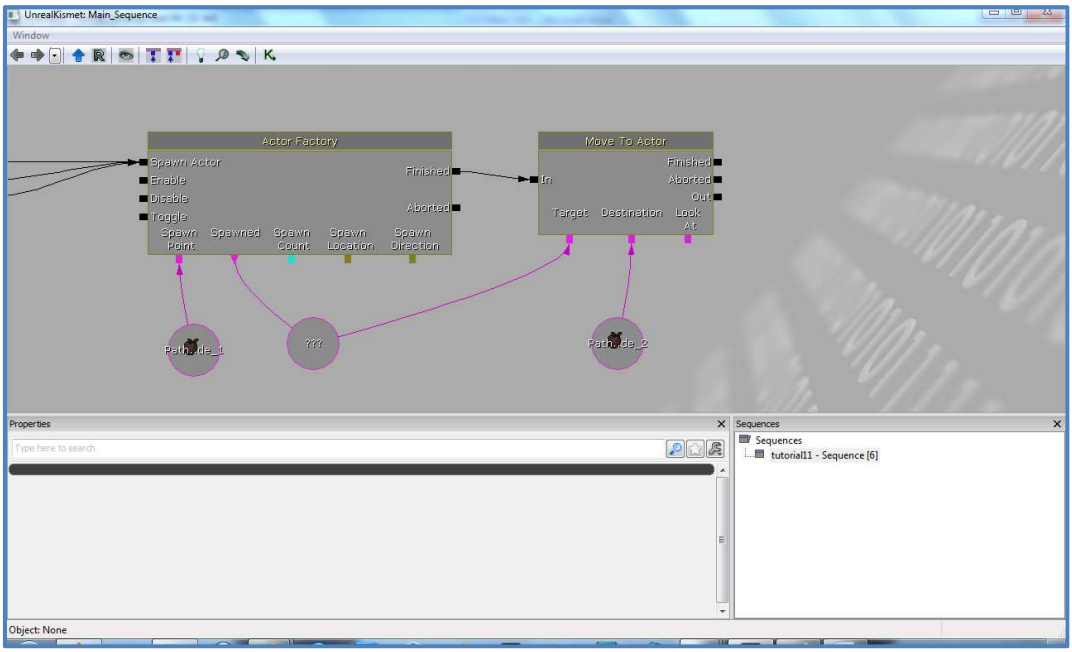


Imagen 322:Tutorial 11-Asociación "Object" y "PathNode" con "Move to Actor".

Con esta implementación el personaje al ser añadido se moverá desde el punto donde aparece hasta el segundo punto dado.

Puede ampliarse toda esta acción para que al llegar el personaje al segundo punto, éste vuelva al primer punto. Dicha implementación se indica en el siguiente punto.

6. Hay que añadir un nuevo elemento *"Move To Actor"* para hacer que el personaje vuelva al punto inicial.

   En este nuevo elemento se indica el punto al que se quiere que se mueva el personaje, siendo el mismo en el cual apareció. Se puede hacer una copia de la variable que representa a este punto mediante teclas 'Ctrl+C' para copiar, y 'Ctrl+V' para copiar, ó unir la misma variable a la entrada "Destination".

   En la entrada *"Target"* hay que asociar la variable que representa al personaje.

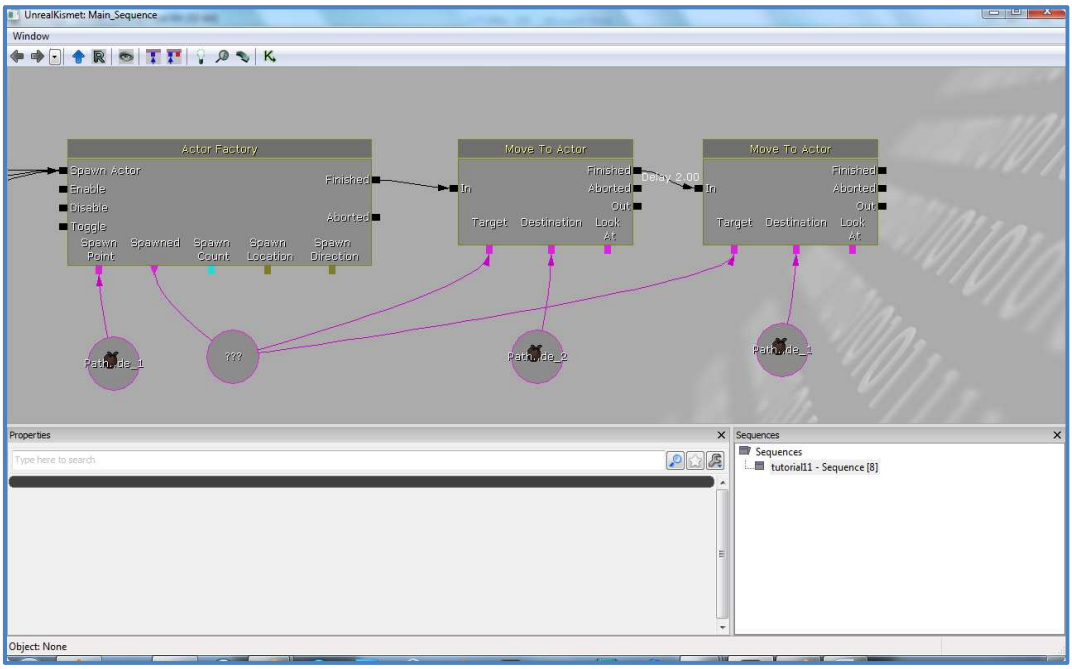


Imagen 323:Tutorial 11-Creación elemento "Move To Actor".

Al asociar los dos elementos *"Move To Actor"* hay que activar un retardo en la salida del primero para que no existan problemas de ejecución. Para activar el retardo de tiempo se pulsa con el botón derecho del ratón sobre la salida *"Finished"* mostrando un menú de acciones, mostrado en la siguiente imagen:

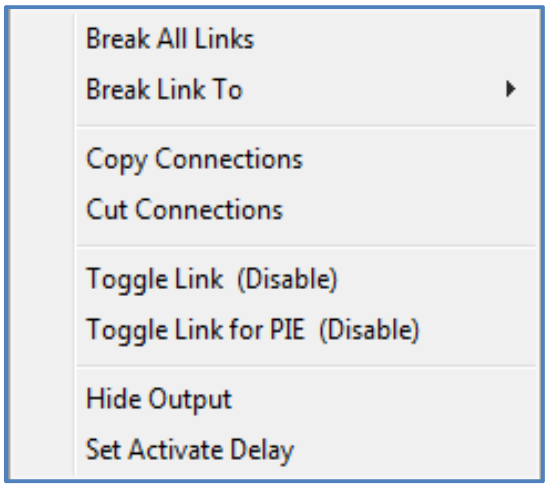


Imagen 324:Tutorial 11-Menú activación retardo.

En este menú mostrado, mediante la opción *"Set Activate Delay"* se podrá realizar este retardo indicando los segundos antes de continuar con el flujo de secuencia.

7. Otra opción puede ser que el personaje se traslade continuamente entre los dos puntos:

   Para que el flujo de secuencia entre en un ciclo de ejecución, simplemente hay que unir la salida de la última acción con la entrada de la acción primera del bucle. En este caso se unirá la salida del segundo elemento *"Move To Actor"* con la entrada del primer elemento *"Move To Actor"*, mostrado en la imagen 325.

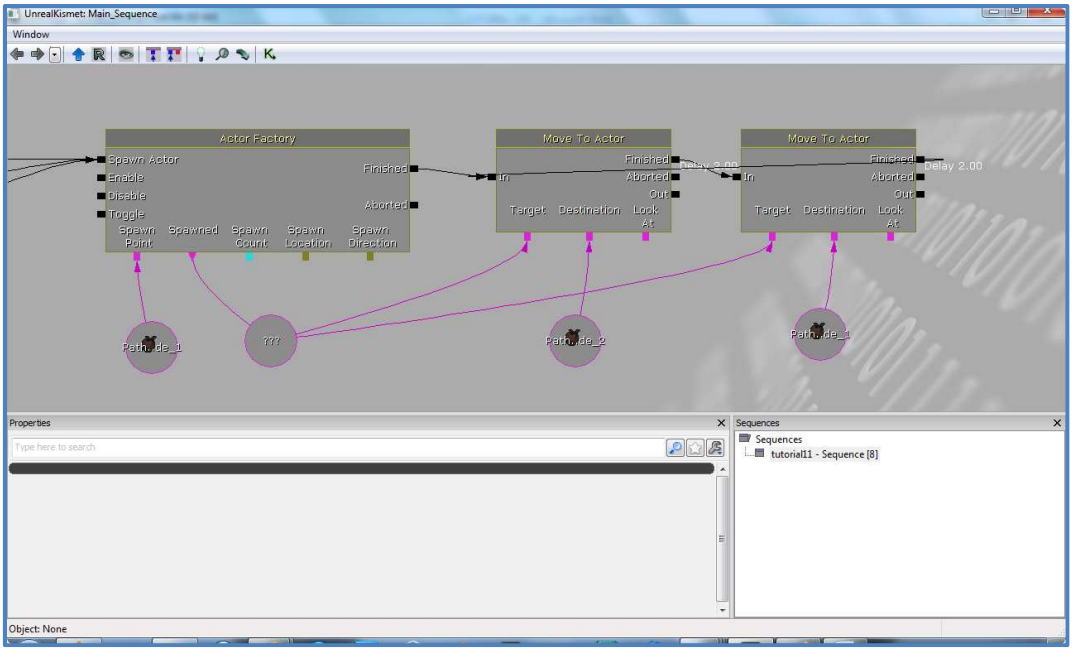


Imagen 325:Tutorial 11-Realización de bucle de ejecución.

Hay que recordar en activar un retardo de tiempo en la salida del segundo elemento *"Move To Actor"* para que no existan conflictos de ejecución.

8.12. Tutorial 12: Personaje dispare al jugador cuando lo "ve".

En este tutorial se va a mostrar otra forma de realizar la acción que el personaje añadido al nivel empiece a disparar al jugador. Para realizar el tutorial se parte de un escenario básico con una plataforma y un muro en medio de ella, mostrado en la imagen 326.

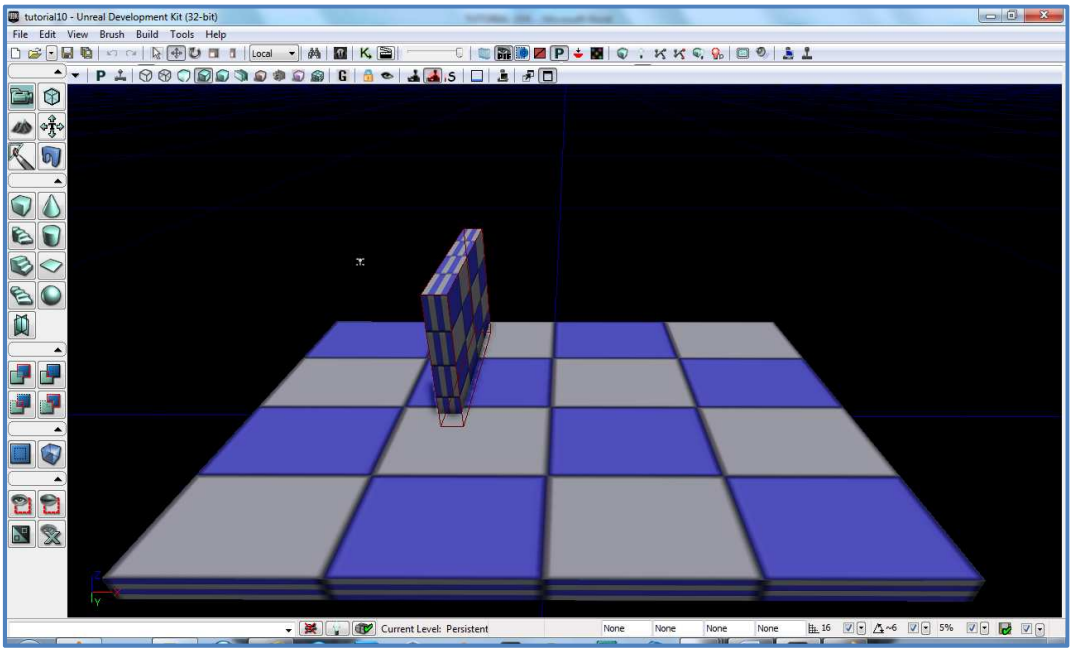


Imagen 326:Tutorial 12-Nivel inicial.

Pasos a seguir:

1. En la herramienta Unreal Kismet se deben añadir los elementos necesarios para que aparezca un nuevo personaje en el nivel.

   En caso de no saber cómo realizarlo, ver el Tutorial 9.

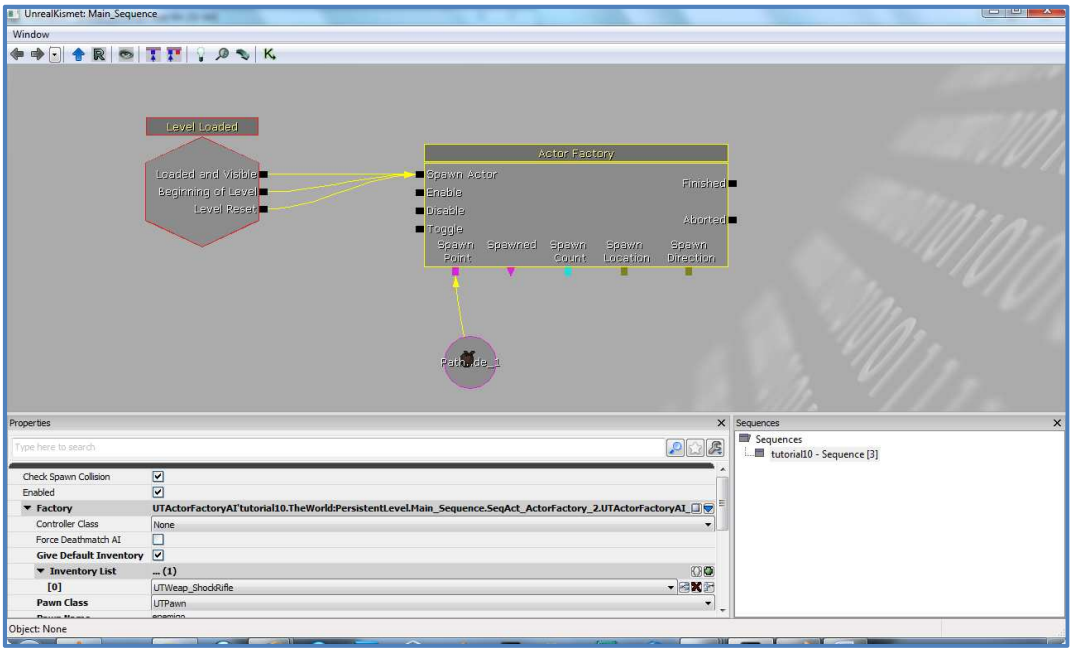


Imagen 327: Tutorial 12-Aparición nuevo personaje en nivel.

   En el elemento *"Actor Factory"*, se desactivará la etiqueta *"Force Deathmatch AI"* de las propiedades.

2. En Kismet es necesario tener una variable global que represente al personaje que se ha acoplado al nivel. Para obtener una variable global se realiza:

   Botón derecho del ratón → Add Variable → Object → Object.

   Para que el personaje añadido al nivel se almacene en esta variable global, hay que unirla con la salida *"Spawed"* del elemento *"Actor Factory"*:

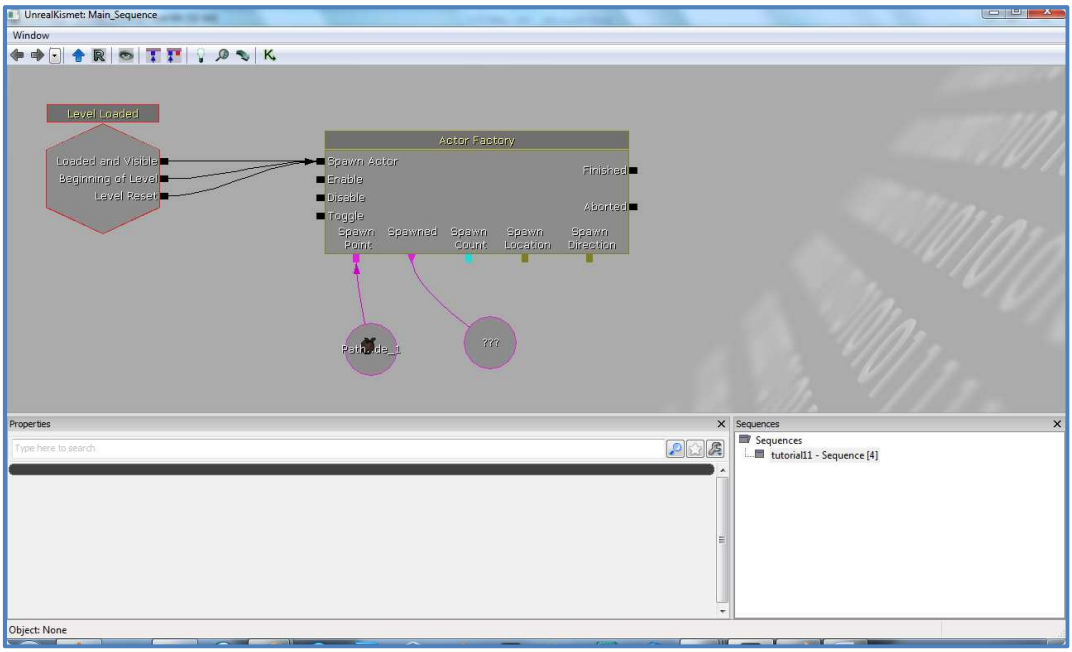


Imagen 328:Tutorial 12-Variable "Object".

3. En la herramienta se permite comprobar si existen obstáculos en la trayectoria entre dos puntos dados.

   Esta comprobación se realizar gracias al elemento llamado *"Trace"* que devuelve si hay o no obstáculos en la línea que une los dos puntos dados. Para obtener un elemento de este tipo realizar:

   Botón derecho del ratón → New Action → Misc → Trace.

   En este elemento se tienen dos salida: una indicando que no hay obstáculos, y otra indicando que sí los hay.

   Otra opción que se puede realizar en este elemento es indicar la distancia a comprobar, a partir del punto *"Start"*. Para ello se tiene la etiqueta *"Distance"* en las propiedades donde se puede indicar el valor deseado.

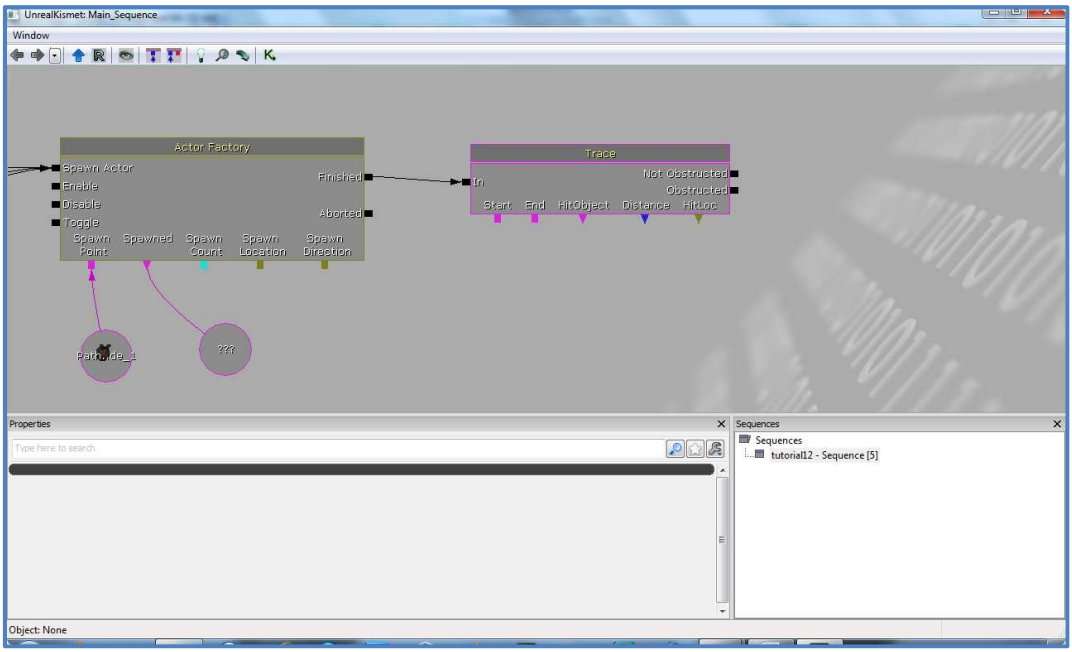


Imagen 329:Tutorial 12-Creación elemento "Trace".

4. Es necesaria una variable que represente al jugador. Esta variable será utilizada como uno de los puntos entre los cuales comprobar si existen obstáculos o no. Para obtener esta variable:

   Botón derecho del ratón → New Variable → Player → Player

   En las propiedades de esta variable se desactiva la etiqueta *"All Players"* para que represente sólo al jugador principal.

   Esta variable se unirá con la entrada *"End"* del elemento *"Trace"*.

   El otro punto entre los que comprobar si hay obstáculos es el personaje añadido al nivel, con lo que habrá que unir la variable global que lo representa con la entrada *"Start"* del elemento *"Trace"*.

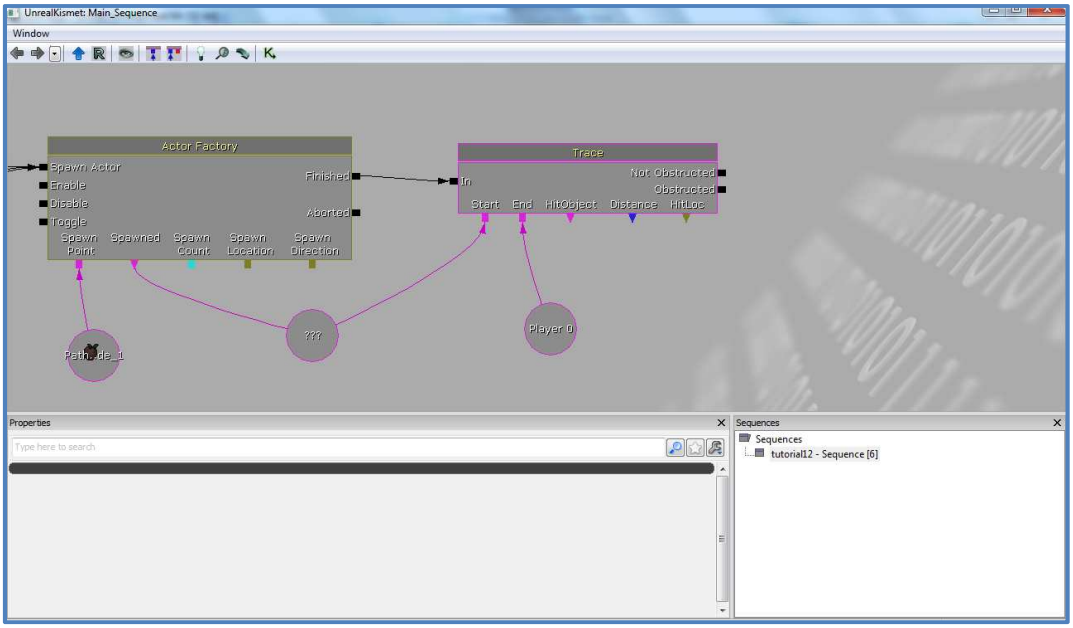


Imagen 330:Tutorial 12-Variable de jugador.

5. Cuando no existan obstáculos entre los dos personajes, se quiere que el personaje acoplado empiece a disparar al personaje principal. En Kismet existe un elemento que permite realizar la acción de comienzo de disparo llamado *"Start Firing At"*.

   Botón derecho del ratón → New Action →AI → Start Firing At.

   Este elemento nuevo se unirá con la salida *"Not Obstructed"* del elemento *"Trace"*. A este elemento se le indicará qué *"actor"* debe empezar a disparar. Para ello hay que unir la variable global del personaje acoplado con la entrada *"Target"* del elemento nuevo.

   También hay que indicar a qué se quiere disparar. En este caso se quiere disparar al jugador, con lo que se debe asociar la variable *"Player 0"* en la entrada *"Fire At"* del elemento.

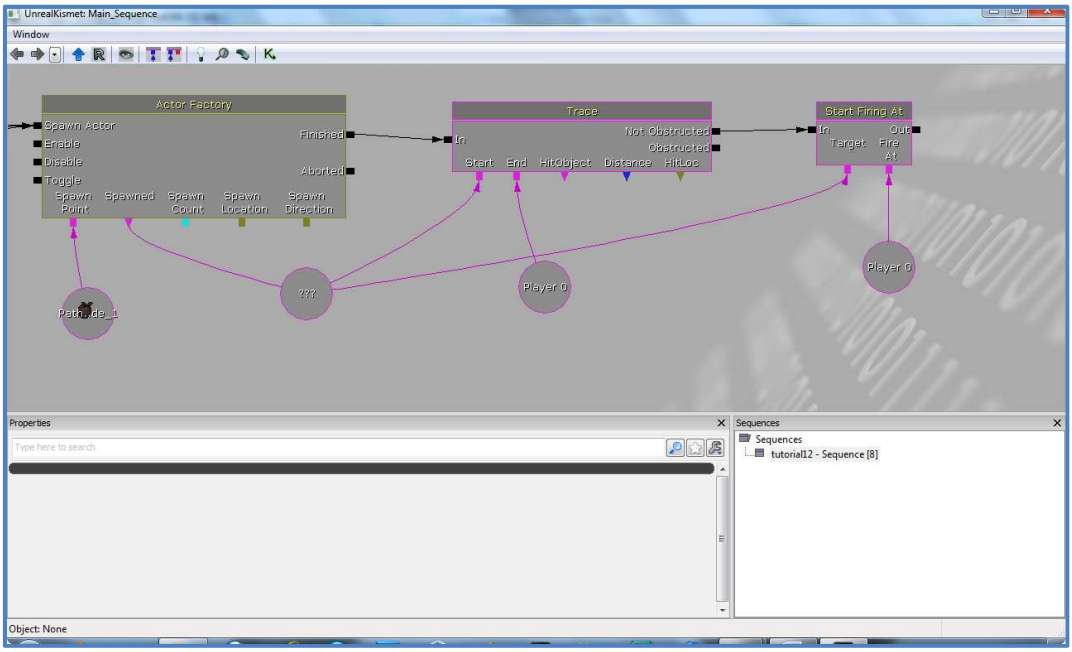


Imagen 331:Tutorial 12-Inserción elemento "Start Firing At" y asociación.

6. La otra opción que se quiere tener es cuando existan obstáculos entre los dos personajes, se deje de atacar al jugador. El elemento llamado *"Stop Firing"* permite realizar lo indicado, obteniéndose:

   Botón derecho del ratón → New Action → AI → Stop Firing.

   Se unirá con la salida *"Obstructed"* del elemento *"Trace"* y en la entrada *"Target"* se asocia el personaje que no tiene que atacar, el personaje acoplado al nivel.

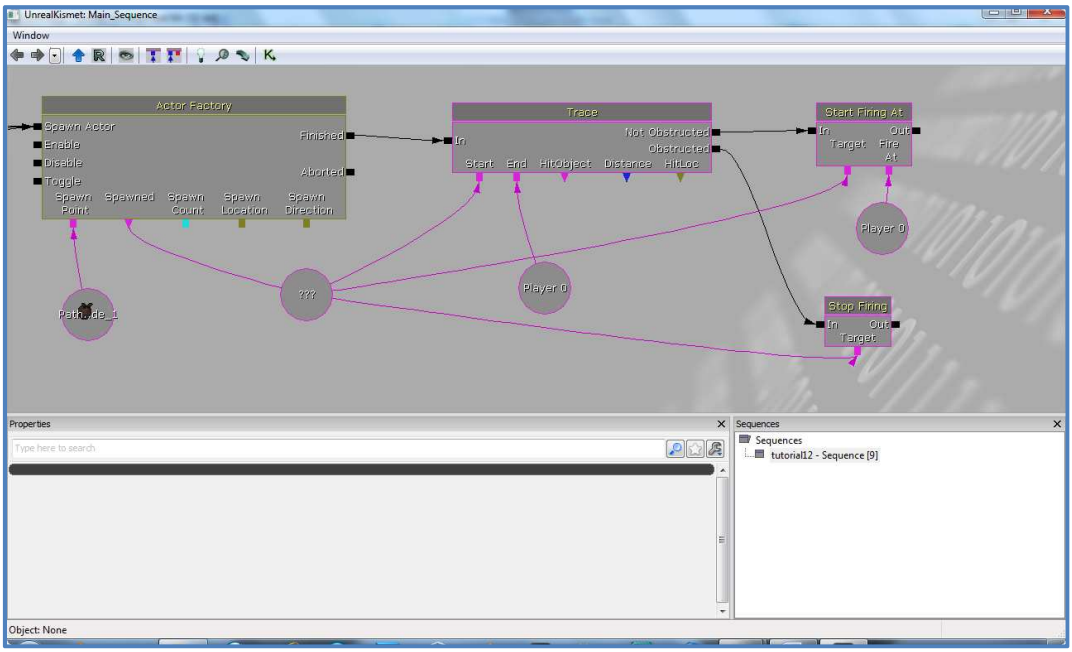


Imagen 332: Tutorial 12-Inserción elemento "Stop Firing" y asociación.

7. Se quiere que las acciones de disparar y dejar de disparar se esté realizando constantemente, indicando si hay obstáculos o no.

   Para ello unir las salidas de los dos elementos *"Start Firing At"* y *"Stop Firing"* con la entrada del elemento *"Trace"* para que se cree un bucle de ejecución.

   Para que no existan problemas en la ejecución de la acción hay que activar un cierto retardo en las salidas de los elementos *"Start Firing At"* y *"Stop Firing"*, aunque sea muy pequeño el tiempo dado.

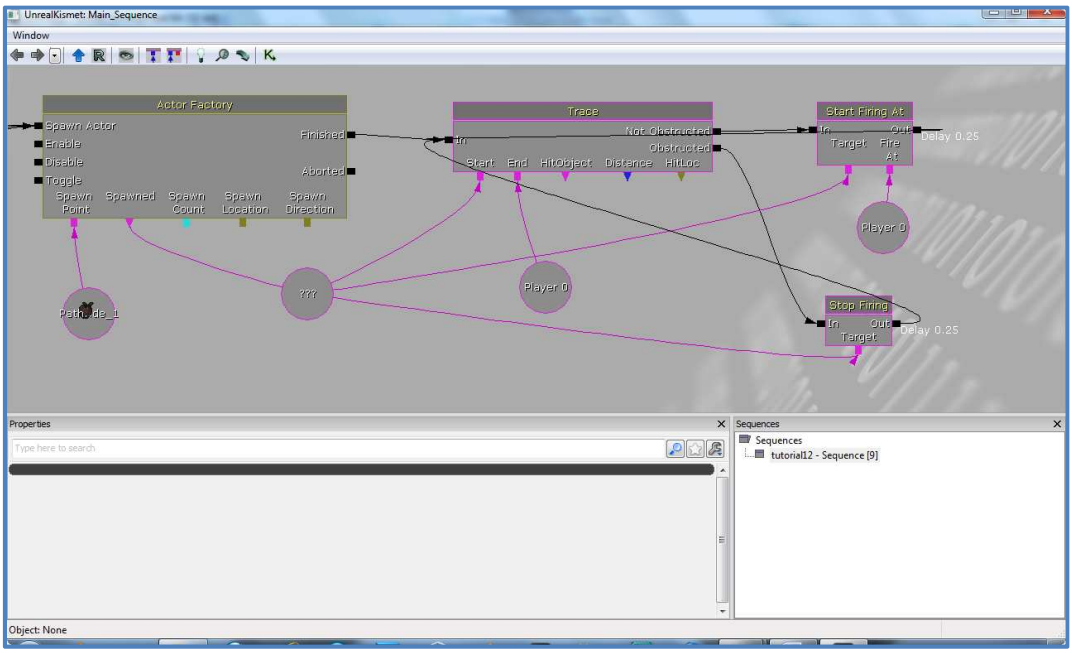


Imagen 333:Tutorial 12-Bucle de ejecución.

Para activar el retardo de tiempo se debe pulsar con el botón derecho del ratón sobre la salida y en el menú mostrado seleccionar *"Set Activate Delay"* indicando el tiempo deseado.

8.13. Tutorial 13: Mostrar valor de variable en pantalla.

Puede ser necesario el poder mostrar por pantalla el valor de una variable. Esto puede ser útil para visualizar un contador, controlar una acción, entre otras. En la herramienta Unreal Kismet se proporciona un elemento que permite visualizar el valor de una variable en pantalla llamada *"Log"* obteniéndose mediante:

Botón derecho del ratón → New Action → Misc → Log

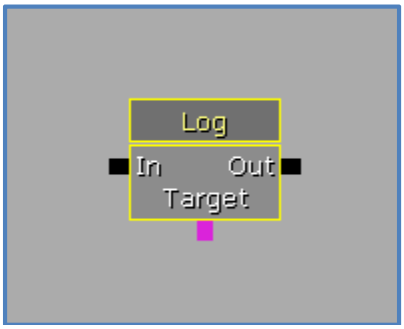


Imagen 334:Tutorial 13-Elemento "Log".

En la entrada *"Target"* se deberá asociar la variable la cual se quiere visualizar su valor. Por defecto se tiene para asociar elementos de tipo *"Object"* y para el caso de asociar algún otro tipo de variable, hay que obtener su correspondiente tipo de entrada.

Para obtener las distintas entradas que se pueden tener simplemente:

- Pulsar con el botón derecho del ratón sobre el elemento.
- En el menú mostrado, imagen 335, seleccionar *"Expose Variable"*.

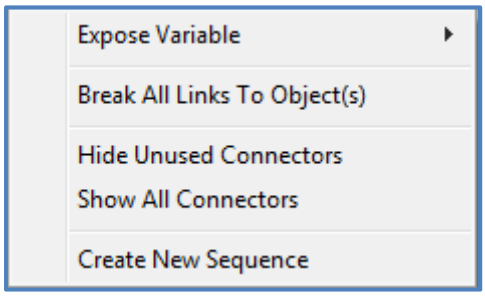


Imagen 335:Tutorial 13-Menú elemento "Log".

- Al seleccionar esta opción, se mostrará un nuevo menú, imagen 336, donde se mostrarán todos los tipos de variables que se puede mostrar la entrada:

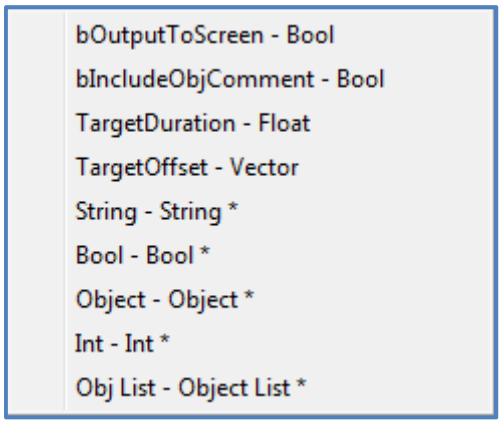


Imagen 336:Tutorial 13-Tipos de variables de entrada.

En las propiedades de este elemento, se muestra una propiedad donde se puede indicar el tiempo que se desea que se muestre el valor en pantalla. Esta propiedad es llamada *"Target Duration"* donde se debe indicar el valor.

Por defecto, el valor de la variable será mostrado en la parte media inferior izquierda de la pantalla, pero puede ser mostrado en cualquier posición indicándolo en la propiedad *"Target Offset"*, insertando las coordenadas en los tres ejes.

A modo de ejemplo para el tutorial, se va a mostrar por pantalla el valor de una variable de tipo *"Int"* la cual va a ser modificada cada segundo. Esta variable se mostrará indefinidamente creando un bucle para ello. La implementación es de la siguiente manera:

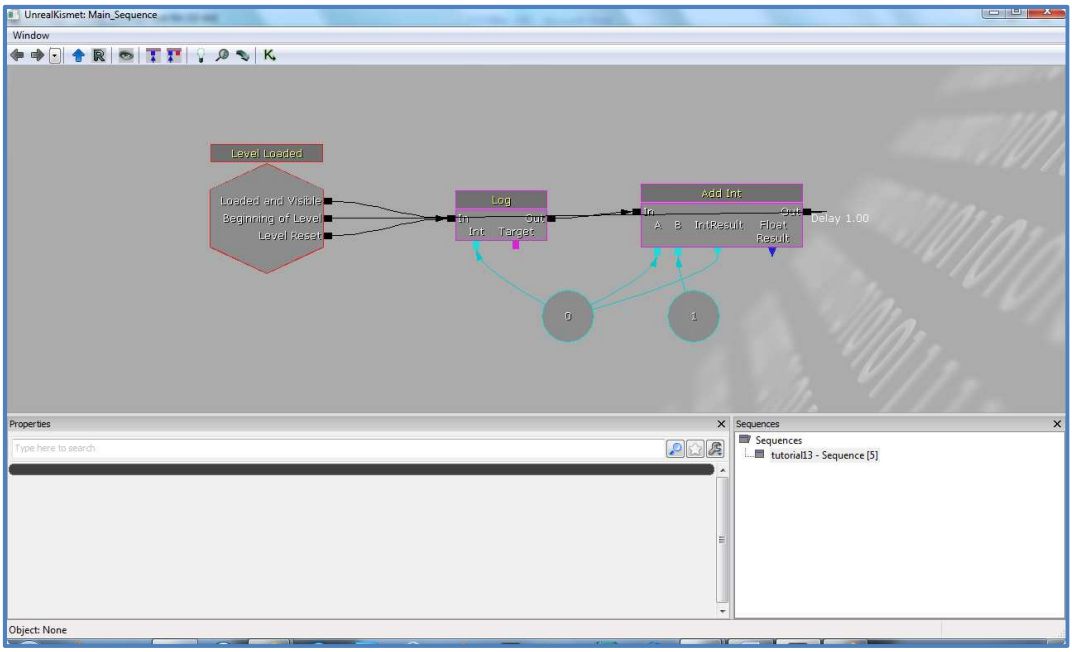


Imagen 337:Tutorial 13-Ejemplo uso elemento "Log".

En este ejemplo, se ha utilizado el elemento *"Add Int"* que sumará el valor de dos variables *"Int"* guardando el resultado obtenido en otra variable, la cual es la mostrada por el elemento *"Log"*.

Nada más lanzar el nivel, el valor de la variable será mostrado en la pantalla, en la parte inferior izquierda, indefinidamente. Esta implementación podría ser utilizada como contador de tiempo, para ir mostrando el tiempo que se lleva transcurrido.

8.14. Tutorial 14: Poner viento.

Uno de los elementos más interesantes que se pueden acoplar a un nivel es el viento. El motor UDK posee un elemento que permite simular la acción del viento. Los restantes objetos del nivel interactuarán con este elemento, realizando un movimiento simulando dicha acción del viento.

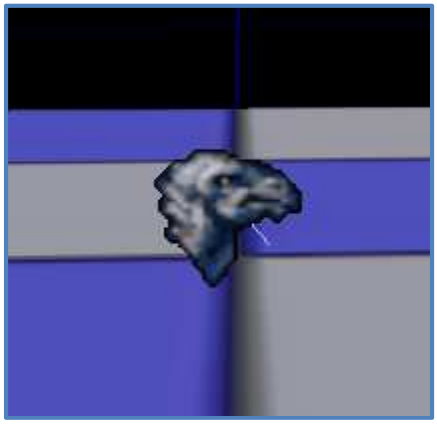


Imagen 338:Tutorial 14-Elemento viento.

Este elemento se puede encontrar en la herramienta *"Actor Classes"* pudiéndola abrir mediante el icono de la barra de herramientas superior del editor y seleccionando la etiqueta con dicho nombre.

En la jerarquía de secciones, imagen 339, se puede obtener dicho elemento mediante:

Info → WindDirectionalSource

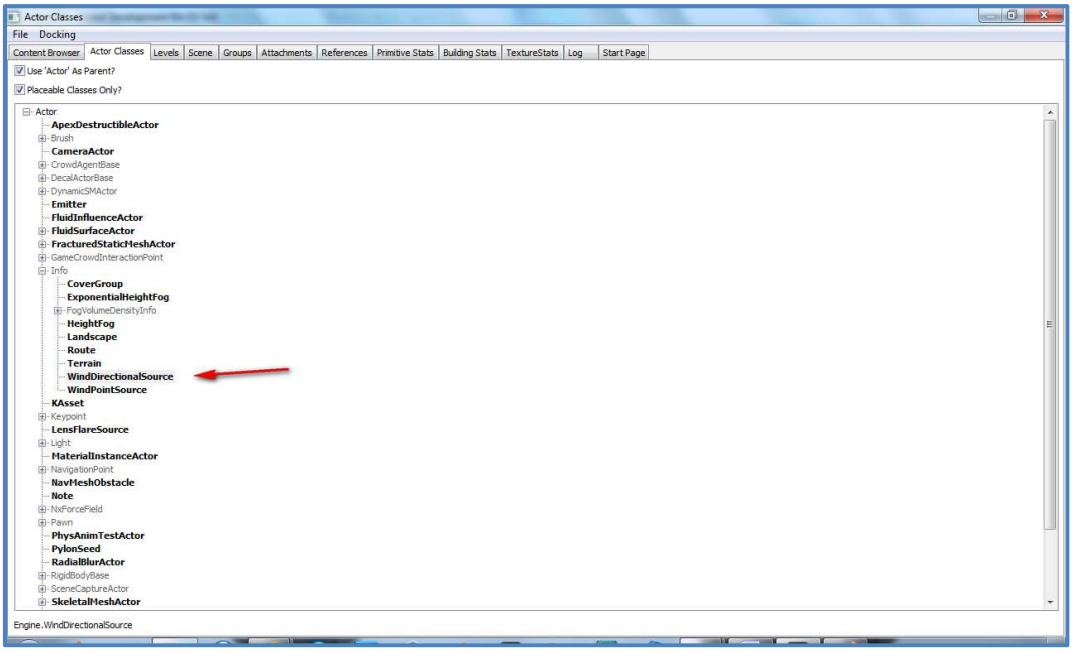


Imagen 339:Tutorial 14-"WindDirectionalSource" en "Actor Classes".

Al añadir este objeto al nivel puede observarse que se muestra una flecha. Esta flecha representa la dirección del viento que se simulará. Esta dirección puede ser modificada activando el icono , *"Rotation Mode"*, para indicar la dirección del viento deseada.

En las propiedades de este elemento se pueden modificar ciertas características. Para obtener las propiedades, se selecciona el objeto y presionar la tecla 'F4' del teclado:

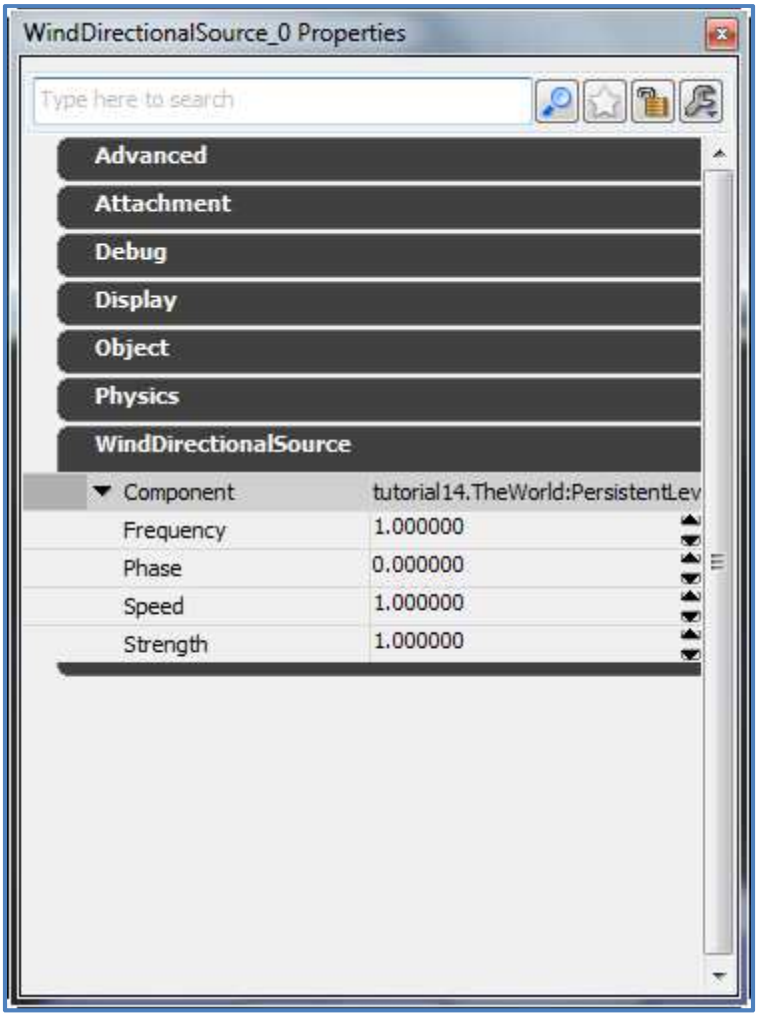


Imagen 340:Tutorial 14-Propiedades de "WindDirectionalSource".

Como se puede observar, se puede modificar la velocidad del viento en la etiqueta *"Speed"*, la frecuencia mediante la etiqueta *"Frequency"*, entre otras.

En el tutorial, se han añadido al nivel varios *"Static Mesh"* con forma de árbol para comprobar el funcionamiento del elemento viento acoplado, tal y como se muestra en la imagen 341.

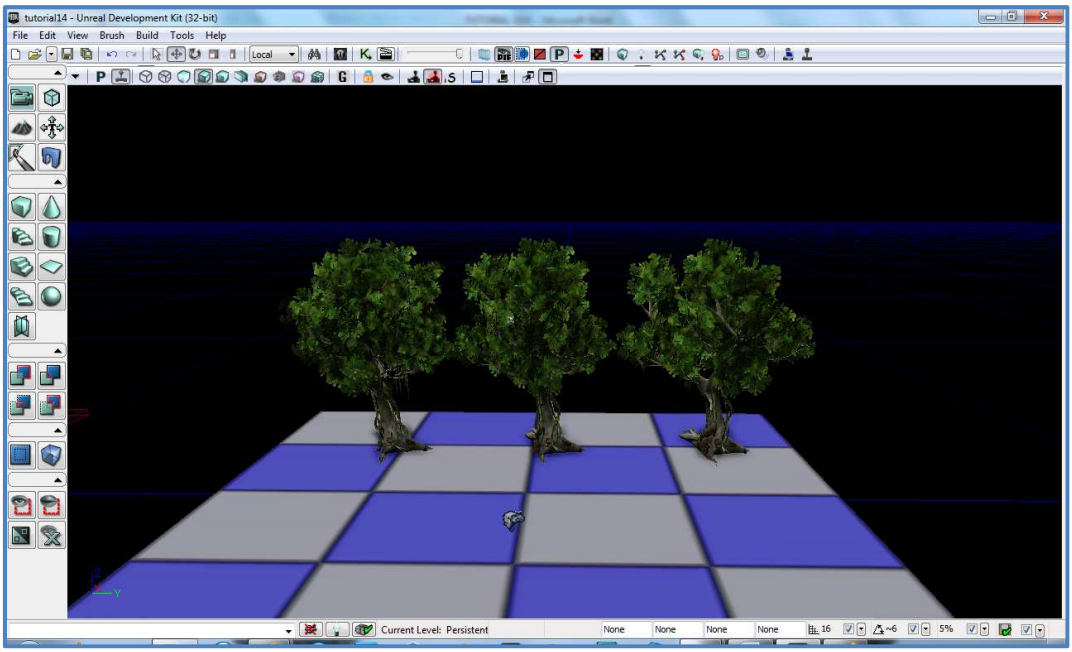


Imagen 341:Tutorial 14-Ejemplo de uso de viento.

8.15. Tutorial 15: Acoplar "lanzador" de personaje.

Para trasladar al personaje de un punto a otro, se tiene la posibilidad de poder *"lanzarlo"*, que el personaje salga volando hasta el punto de aterrizaje indicado. Esta forma de traslado del personaje puede ser utilizado para poder subir a distintos niveles de altura, salvar obstáculos encontrados, etc.

El motor UDK posee un elemento que realiza esta acción con el personaje principal:

Imagen 342:Tutorial 15-Elemento lanzador.

Este elemento se puede encontrar en la herramienta *"Actor Classes"* pudiéndola abrir mediante el icono de la barra de herramientas superior del editor y seleccionando la etiqueta con dicho nombre.

En la jerarquía de secciones se puede obtener dicho elemento mediante:

Navigation Point → UDKJumpPad → UTJumpPad.

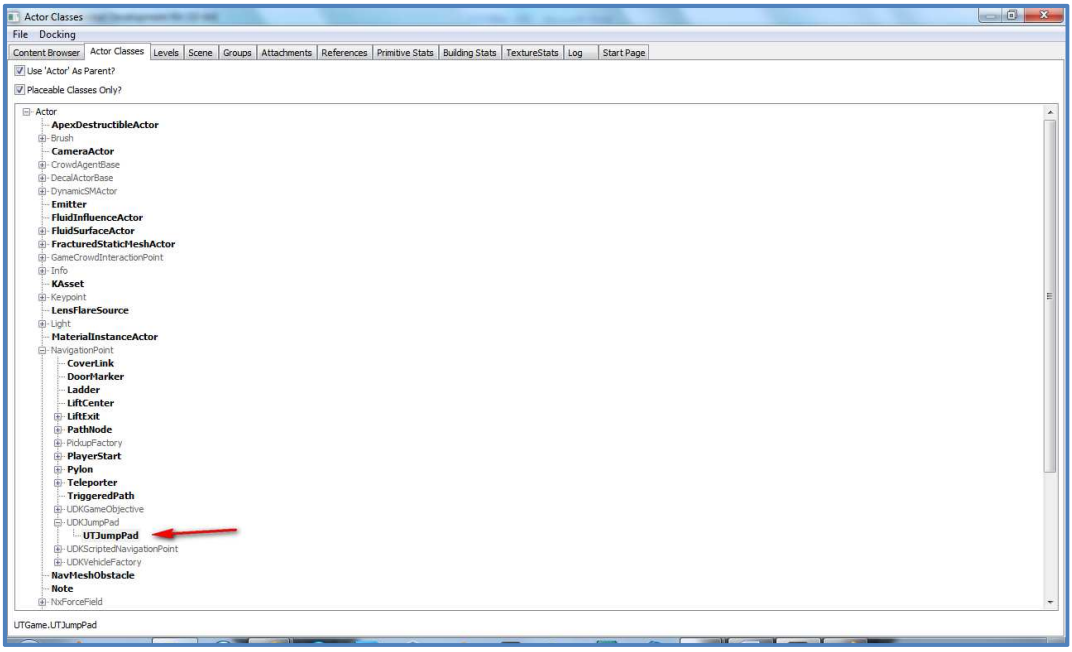


Imagen 343:Tutorial 15-Localización de "UTJumpPad" en "Actor Classes".

Al añadir este objeto al nivel, puede observarse que se muestra una flecha. Esta flecha representa la dirección hacia donde el personaje será lanzado. Esta dirección puede ser modificada activando el icono , *"Rotation Mode"*, para indicar la dirección deseada.

También hay que indicar el punto donde se quiere aterrizar el personaje lanzado. Este elemento debe ser de tipo *"PathNode"* teniendo que acoplar uno al nivel. Para ello:

Botón derecho del ratón → Add Actor → Add PathNode

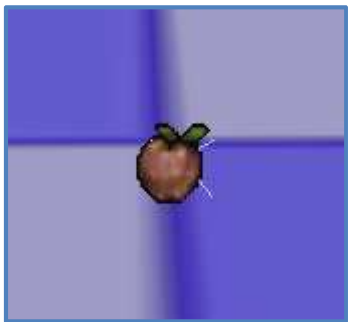


Imagen 344:Tutorial 15-Elemento "PathNode".

Con el objeto del punto de "aterrizaje" acoplado al nivel hay que indicar en el elemento *"UTJumpPad"* el punto donde debe llegar el personaje. Para realizar todo ello hay que abrir las propiedades del elemento *"UTJumpPad"* mediante la tecla 'F4' del teclado:

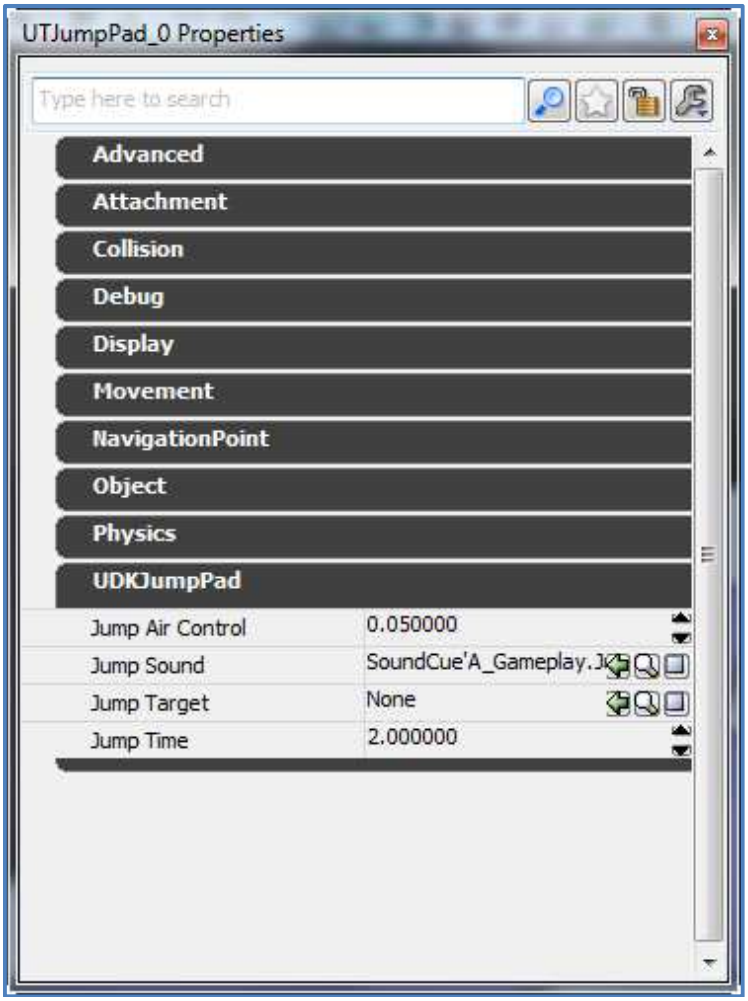


Imagen 345:Tutorial 15-Propiedades lanzador "UTJumPad".

En la etiqueta *"Jump Target"* indicar el objeto *"PathNode"* acoplado anteriormente. Para poder realizar la asociación, primero hay que anclar la ventana de las propiedades del *"UTJumpPad"* mediante el icono de la parte superior derecha de la ventana. Mediante esto se hará que al seleccionar cualquier otro objeto en el editor, esta ventana no se modifique, mostrando siempre las características del *"UTJumpPad"*.

Se seleccionará el objeto *"PathNode"* en el editor, asociándolo en la etiqueta *"Jump Target"* del elemento *"UTJumpPad"* pulsando el icono de la misma etiqueta.

En las propiedades se puede indicar también el tiempo que tarda el personaje en ir de un punto a otro. Este tiempo se indica en la etiqueta *"Jump Time"* insertando los segundos que se quiere tardar.

Para que el *"lanzador"* funcione perfectamente, antes de lanzar el nivel, hay que compilar al menos los elementos AI mediante:

Etiqueta "Build" del menú de la aplicación → AI Paths.

En el tutorial, imagen 346, se ha acoplado este tipo de *"lanzador"* del personaje indicando el punto de *"aterrizaje"*:

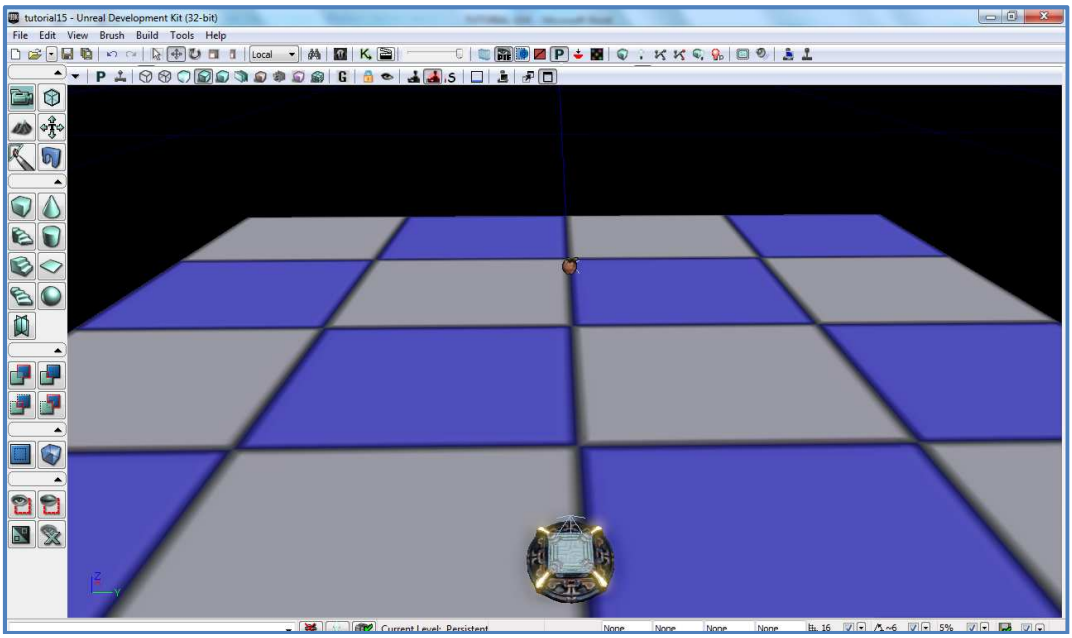


Imagen 346:Tutorial 15-Ejemplo de uso de lanzador.

8.16. Tutorial 16: Poner cielo en el nivel.

En los niveles de un mundo exterior, para hacerlos más realistas, se puede añadir *"cielo"*. La forma más sencilla de poner *"cielo"* en el nivel es añadir una esfera, ó media, muy extensa, donde en su interior se encuentra el nivel. A esta esfera se le asociará una textura asemejando a un cielo real.

Se muestra un ejemplo de todo ello en la imagen 347.

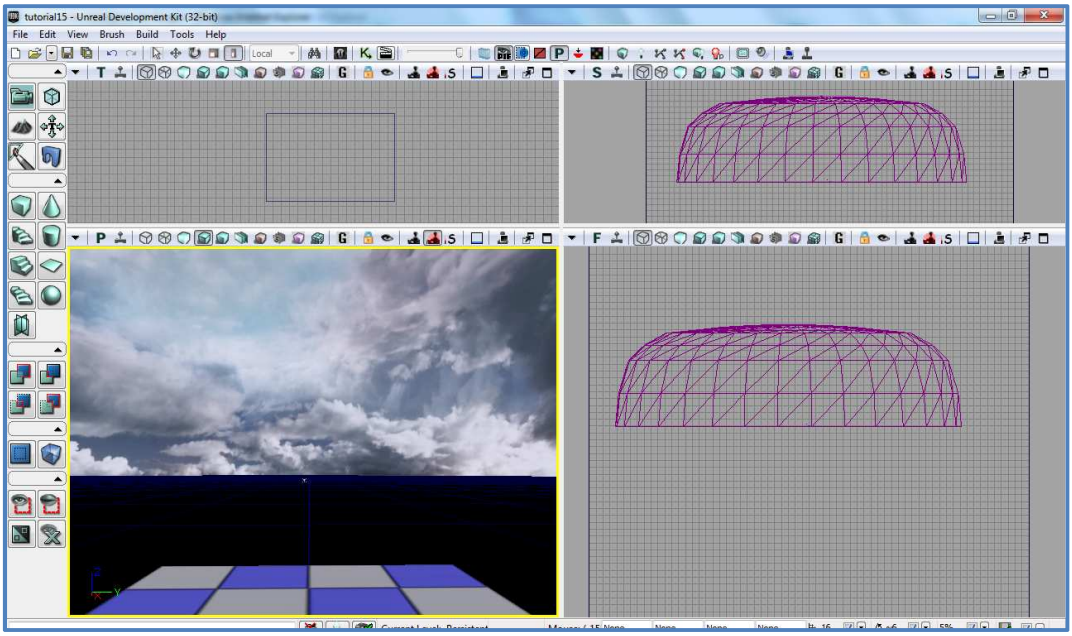


Imagen 347:Tutorial 16-Ejemplo de acople de cielo al nivel.

El motor UDK proporciona elementos *"Static Mesh"* que se pueden utilizar como *"cielo"* en el nivel. Estos elementos se encuentran en la herramienta Content Browser, pudiéndose abrir mediante el icono de la barra de herramientas superior del editor,

conteniendo en su nombre la palabra *"Sky"*. Este objeto *"cielo"* puede dotarse de movimiento para aumentar el grado de realismo.

En el tutorial se ha implementado el movimiento del cielo mediante un elemento Matinee en la herramienta Unreal Kismet. Se ha dotado este movimiento de un bucle para que esté continuamente realizándose. Este movimiento se iniciará nada más lanzar el nivel, mediante un elemento *"Load Level"*:

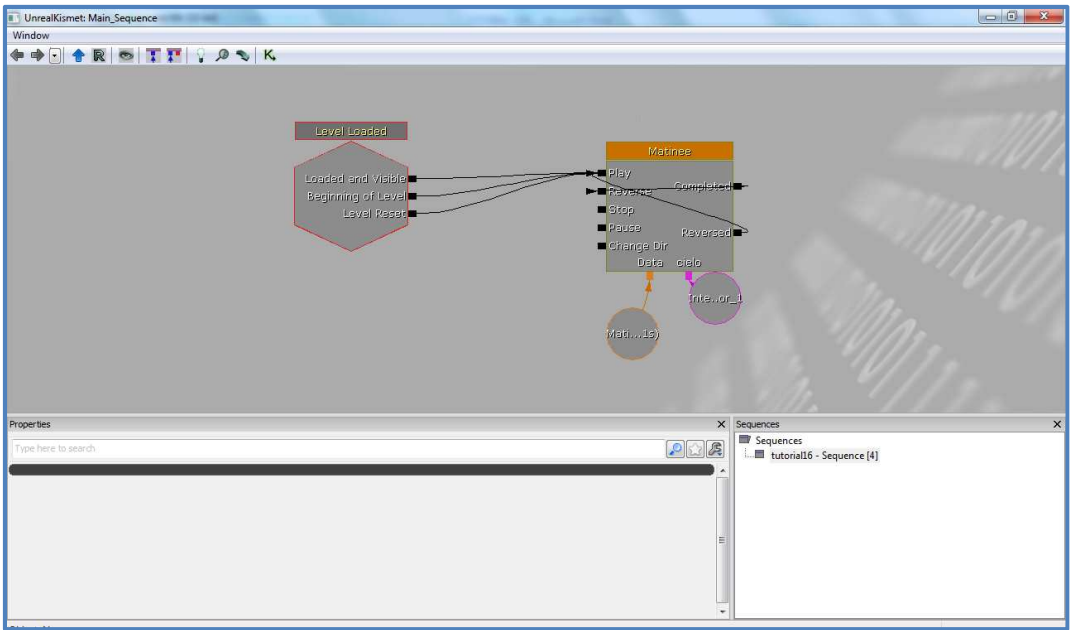


Imagen 348:Tutorial 16-Ejemplo de dinámica realizada con el cielo.

Al arrancar el nivel, se podrá comprobar cómo el cielo se mueve en un sentido y en el otro, indefinidamente.

8.17. Tutorial 17: Cambiar material de un "Static Mesh".

Durante el juego, el material acoplado a un *"mesh"* puede ser modificado, cambiando, por ejemplo, su color. En el tutorial se va a partir de un escenario donde se han acoplado dos *"mesh"*, objetos, a los cuales se le cambiará el color del material, tal y como se muestra en la imagen 349.

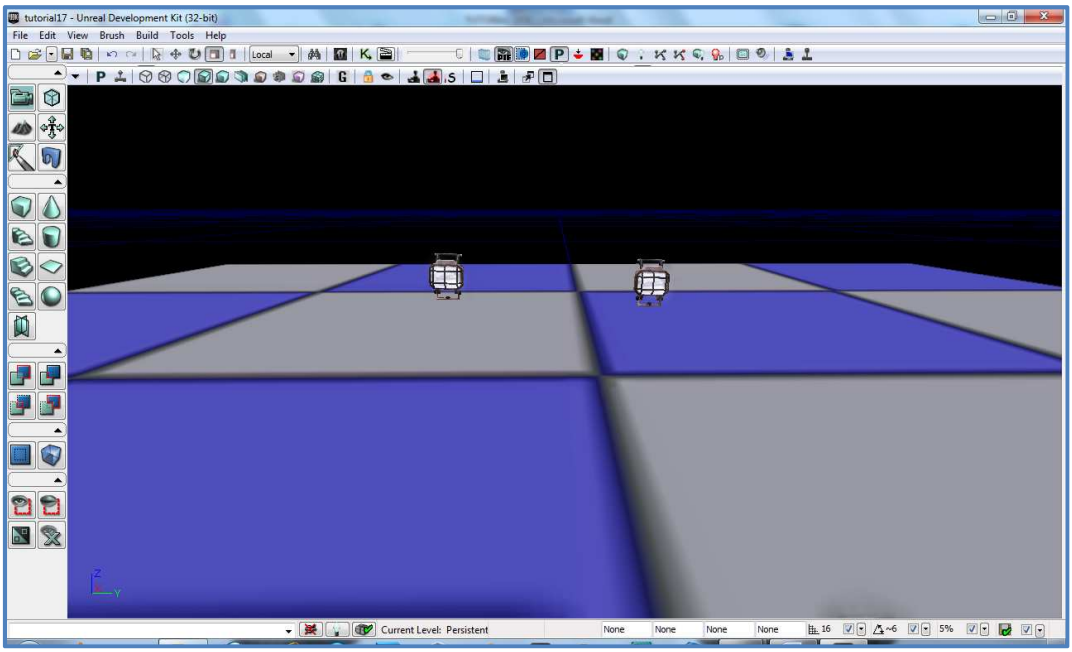


Imagen 349:Tutorial 17-Nivel inicial.

Pasos a seguir:

1. Para modificar el material de un objeto es necesario acoplar al nivel un elemento especial del editor llamado *"Material Instance Actor"*:

Imagen 350:Tutorial 17-Elemento "Material Instance Actor".

Este elemento puede ser encontrado en la herramienta *"Actor Classes"*, pudiéndola abrir mediante el icono de la barra de herramientas superior del editor y seleccionando la pestaña con ese nombre, y en la localización indicada en la imagen 351.

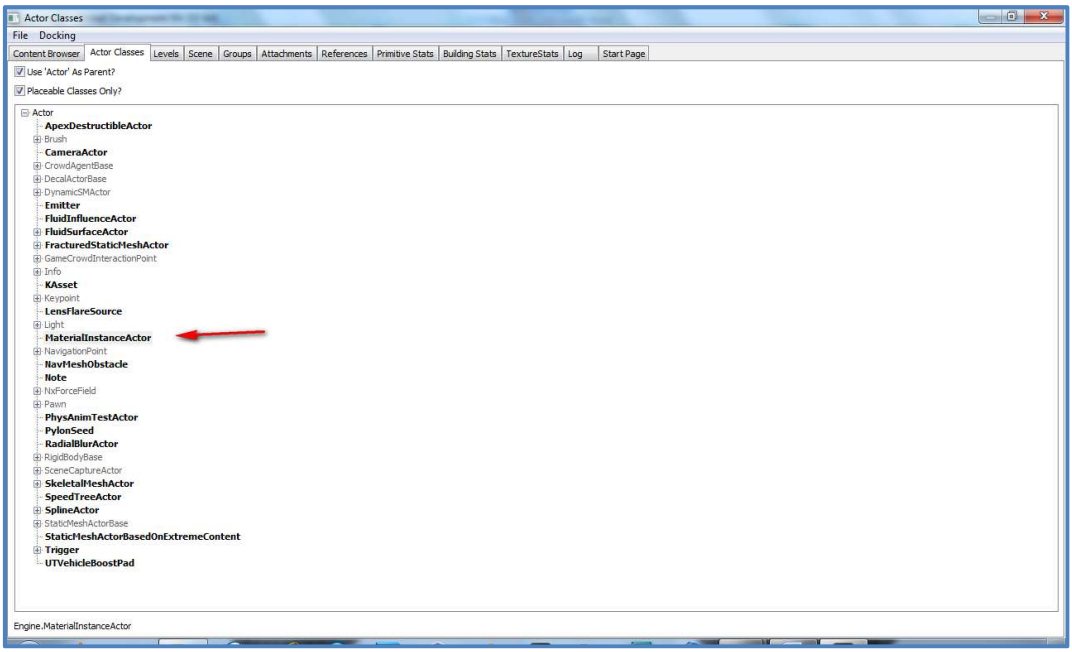


Imagen 351:Tutorial 17-"MaterialInstanceActor" en "Actor Classes".

Para acoplar este elemento debe ser seleccionado y volviendo al editor, en cualquier vista, pulsar el botón derecho del ratón y en el menú mostrado elegir *"Add MaterialInstanceActor Here"*.

2. En este nuevo elemento, en sus propiedades, hay que asociar el Material que se quiere modificar. El material el cual se va a partir es uno exactamente igual pero con el color rojo, mostrado en la imagen 353. Para modificarlo simplemente se abre el material con su editor y se indica el color deseado.

   Para asociar un material nuevo a un *"Static Mesh"* se realiza en sus propiedades (imagen 352), presionando la tecla 'F4' del teclado.

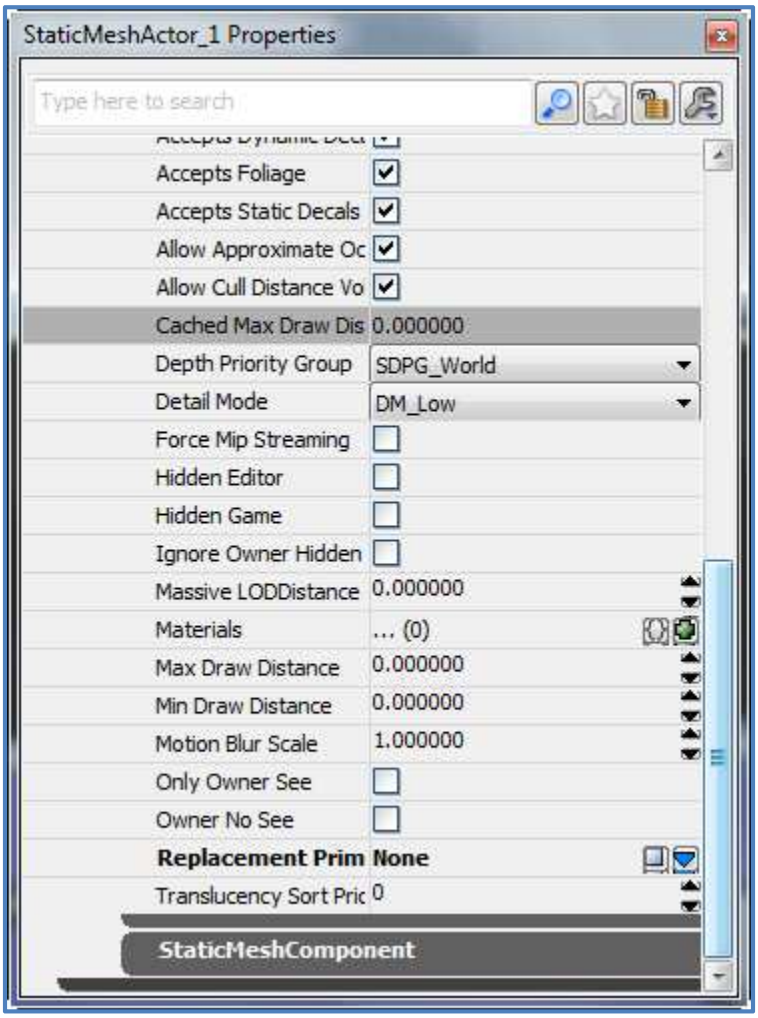


Imagen 352:Tutorial 17-Propiedades de "StaticMesh".

En la etiqueta *"StaticMeshActor → StaticMeshComponent → Rendering → Materials"*, se añade un nuevo elemento pulsando en el icono y el material a utilizar se insertará en la posición creada mediante el icono de la misma etiqueta. El material que se debe acoplar para luego modificarlo en ejecución debe ser del tipo *"MaterialInstanceConstant"*.

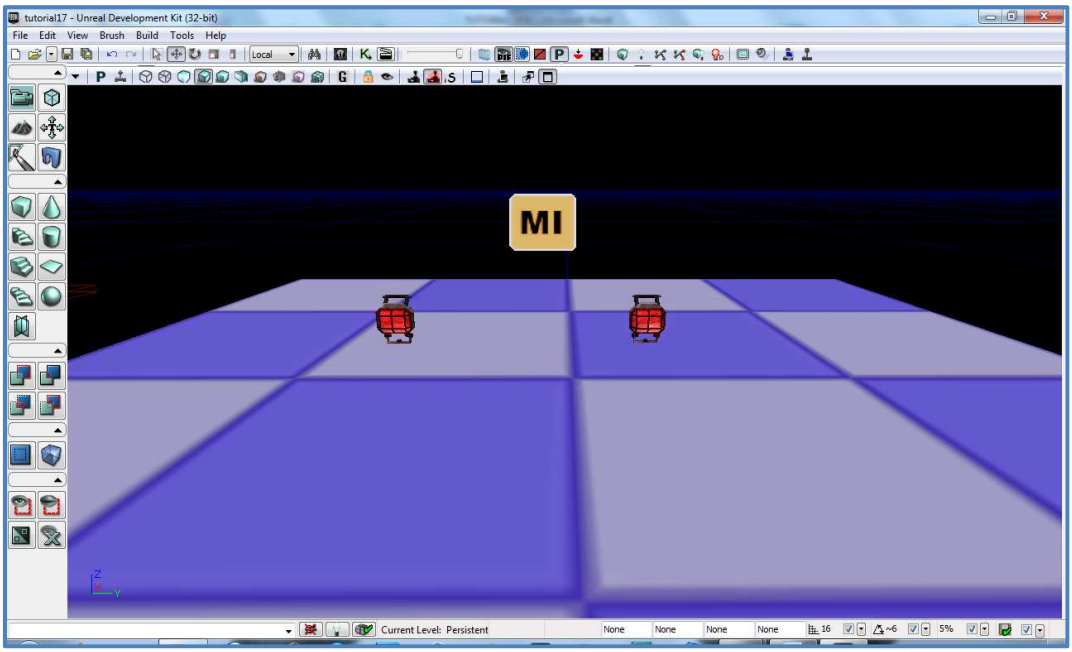


Imagen 353:Tutorial 17-Modificación de material de "Static Mesh".

El tipo de material *"MaterialInstanceConstant"* se debe indicar en las características del elemento *"MaterialInstanceActor"*, en etiqueta *"MaterialInstanceActor → Mat Inst"*:

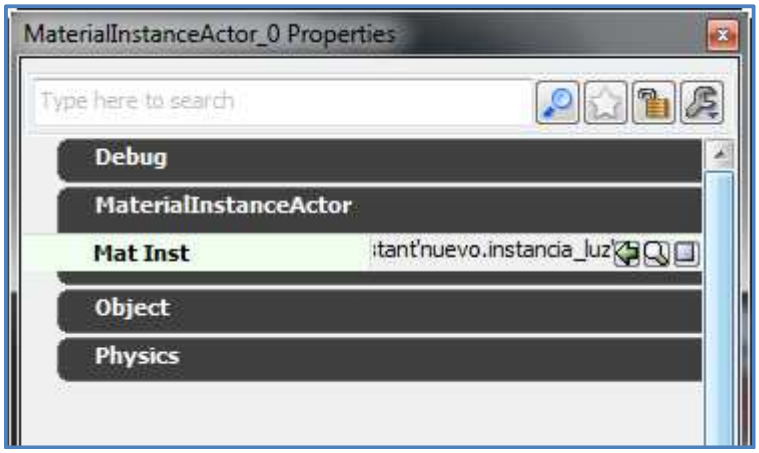


Imagen 354:Tutorial 17-Propiedades elemento "Material Instance".

3. La modificación del material en ejecución se realizará mediante el elemento Matinee de la herramienta Unreal Kismet. Se puede abrir Kismet mediante el icono de la barra de herramientas superior del editor. Se debe añadir un elemento de tipo Matinee mediante:

   Botón derecho del ratón → New Matinee.

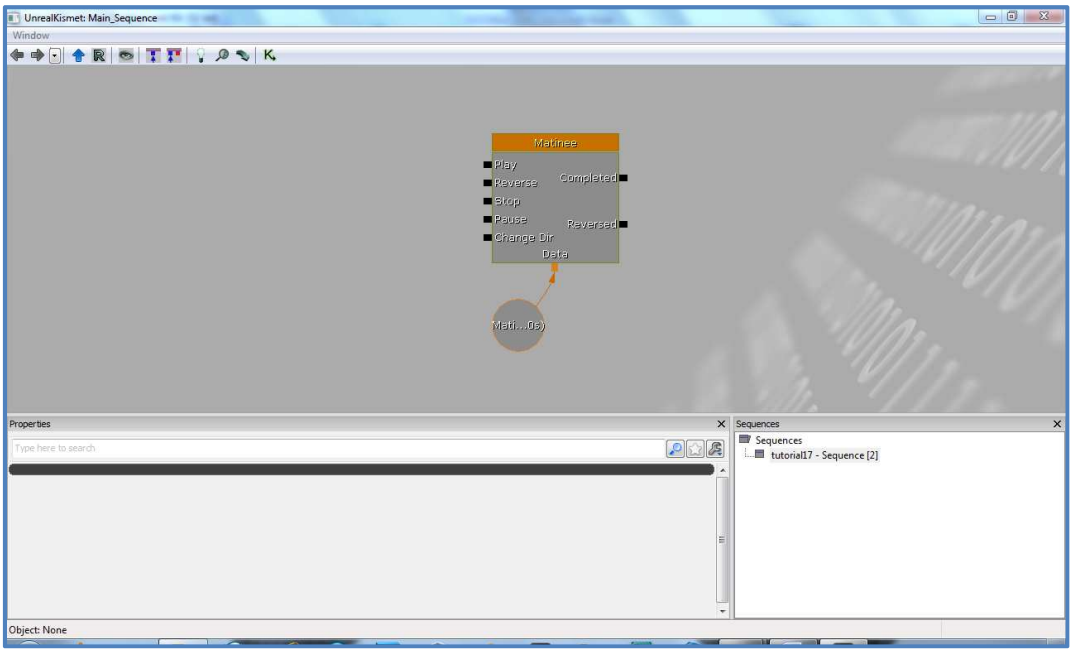


Imagen 355:Tutorial 17-Elemento "Matinee" en "Kismet".

4. En este elemento se implementará la acción de modificar el color de un material. La implementación, se realiza mediante la herramienta Unreal Matinee. Simplemente con hacer doble click sobre el elemento se abrirá la herramienta para poder modificarlo.

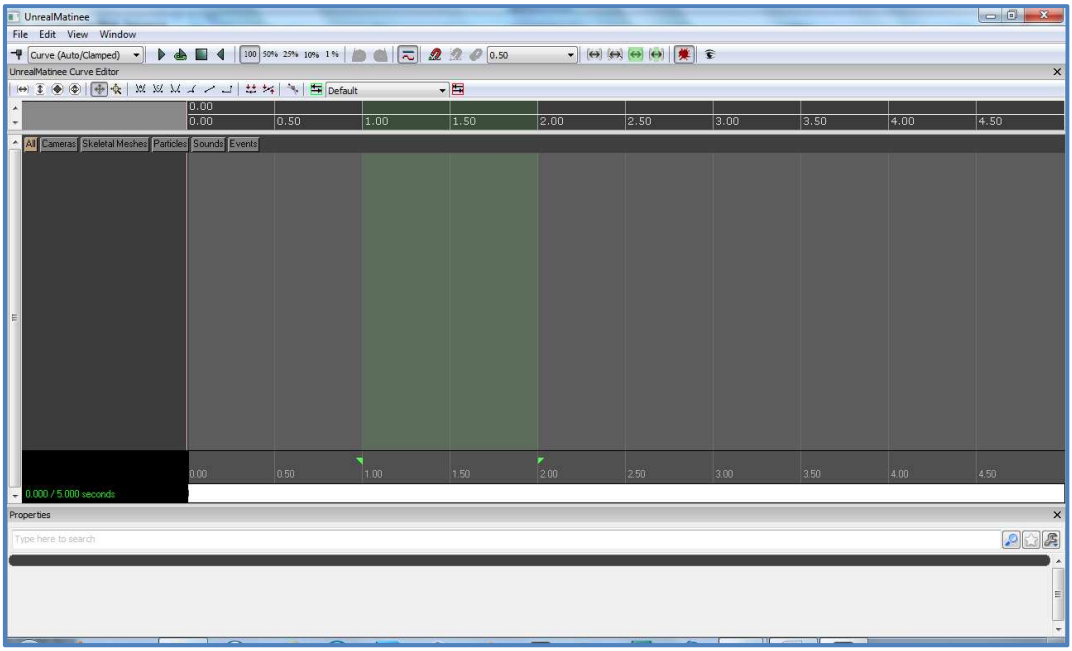


Imagen 356:Tutorial 17-Interfaz herramienta "Unreal Matinee".

5. En la herramienta abierta hay que crear un nuevo grupo vacío pulsando con el botón derecho del ratón sobre la sección izquierda. En el menú mostrado seleccionar *"Add New Empty Group"* dándole el nombre deseado.

   Antes de crear este grupo es recomendable tener seleccionado el objeto *"MaterialInstanceActor (MI)"* en el editor.

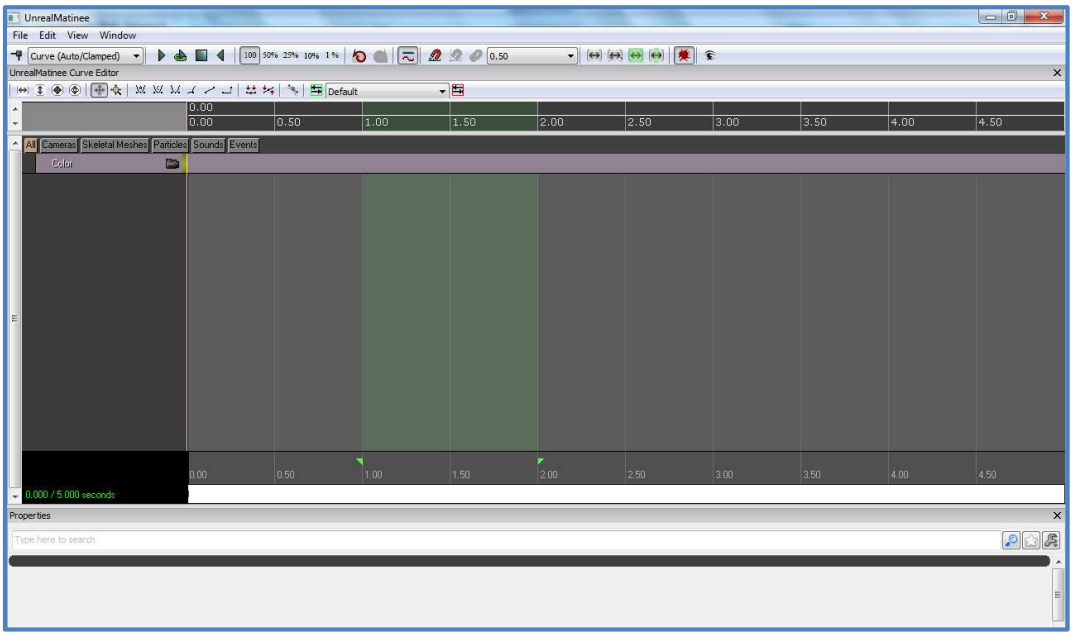


Imagen 357:Tutorial 17-Creación de grupo vacío.

6. A este grupo creado hay que añadir un track del tipo *"Vector Material Param"*. Seleccionado el grupo creado se pulsa el botón derecho del ratón y en el menú mostrado seleccionar *"Add New Vector Material Param track"*, creando el track para modificar el color del material.

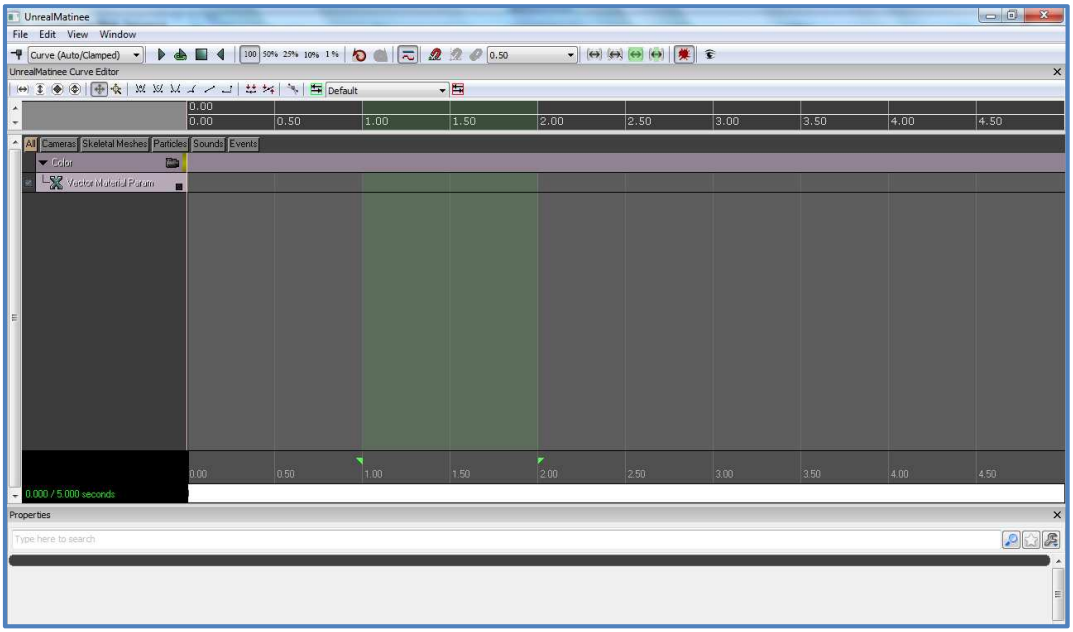


Imagen 358:Tutorial 17-Creación track "Vector Material Param" en grupo.

7. Dentro del material, el color puede estar representado por una variable. En el caso que exista esta variable del color, el nombre de esta variable debe ser indicado en las propiedades del track *"Vector Material Param"* creado en el anterior punto.

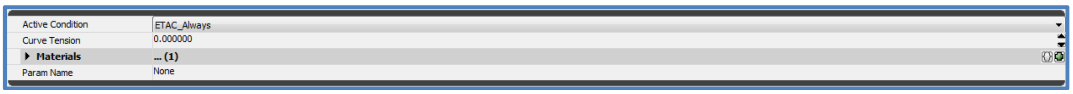


Imagen 359:Tutorial 17-Propiedades track "Vector Material Param".

En la etiqueta *"Param Name"* se indicará el nombre de la variable del color del material.

8. En el track creado será donde se modificará el color del material. En el instante inicial se creará un punto de control, presionando la tecla 'Enter' del teclado, almacenando el color inicial. Este punto de control será mostrado mediante un triángulo rojo en el track.

   En la parte derecha del nombre del track se puede apreciar un pequeño cuadrado. Si se pulsa sobre él, activándolo, en la parte superior del editor aparecerá un nuevo elemento llamado *"Color_Vector Material Param"*:

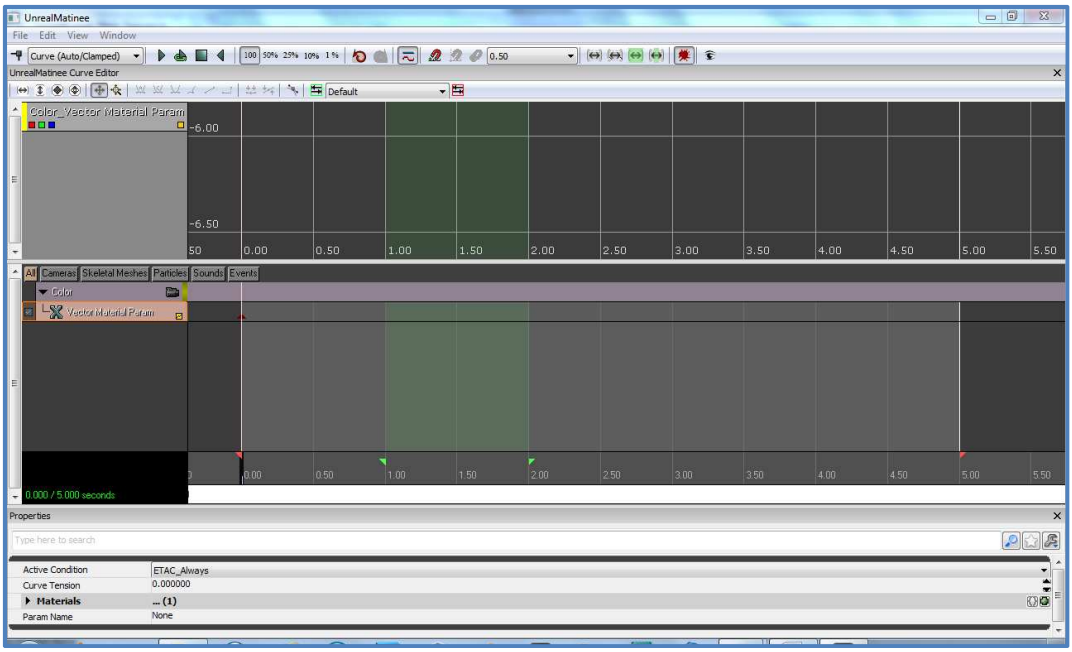


Imagen 360:Tutorial 17-Primer punto de control en secuencia de tiempo.

   Se muestran también las tres componentes del color RGB (Red – Green –Blue) las cuales pulsando sobre ellas serán activadas y mostrados su valores.

9. La línea temporal se desplaza al tiempo de finalización de la duración del cambio del color del material.

   En este caso se va a seleccionar cinco segundos creándose un nuevo punto de control. En cada uno de los puntos de control, en la sección *"Color_Vector Material Param"*, se aprecian ciertos puntos blancos, mostrados en la imagen 361.

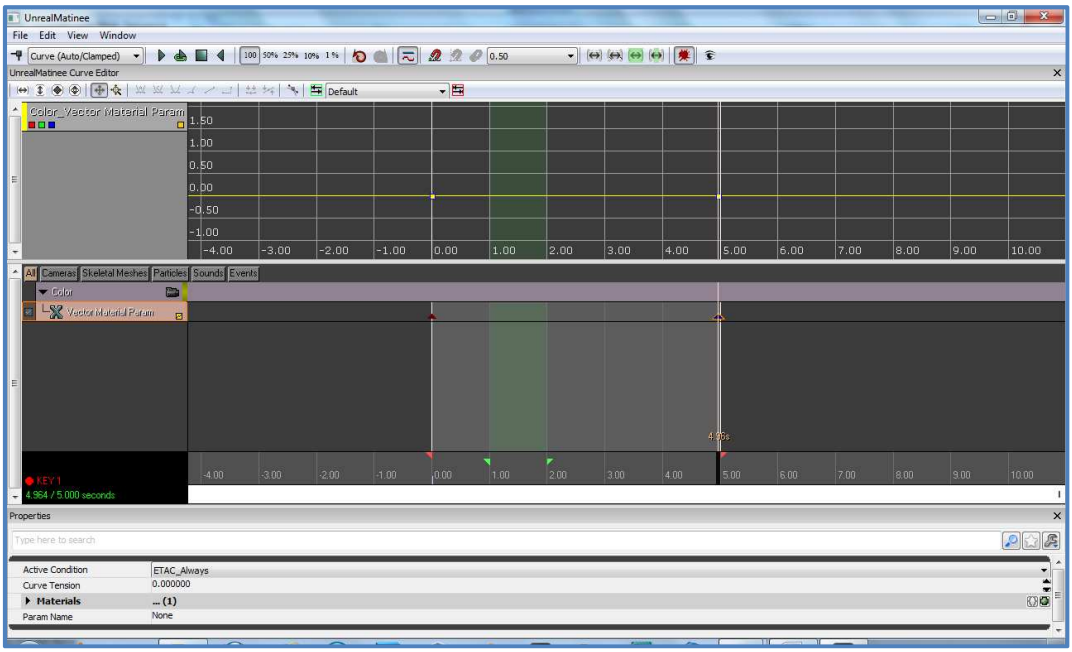


Imagen 361:Tutorial 17-Segundo punto de control en secuencia de tiempo.

Mediante estos puntos se podrá ir modificando el valor de las tres componentes del color (RGB). Para ello se deja seleccionada la componente a modificar en la parte izquierda pulsando con el botón derecho del ratón. En el menú mostrado se selecciona *"Set Value"* insertando el valor a dar.

10. Se van a modificar los valores de la componentes de la siguiente manera en el segundo punto de control creado:

- R = 0
- G = 1
- B = 0

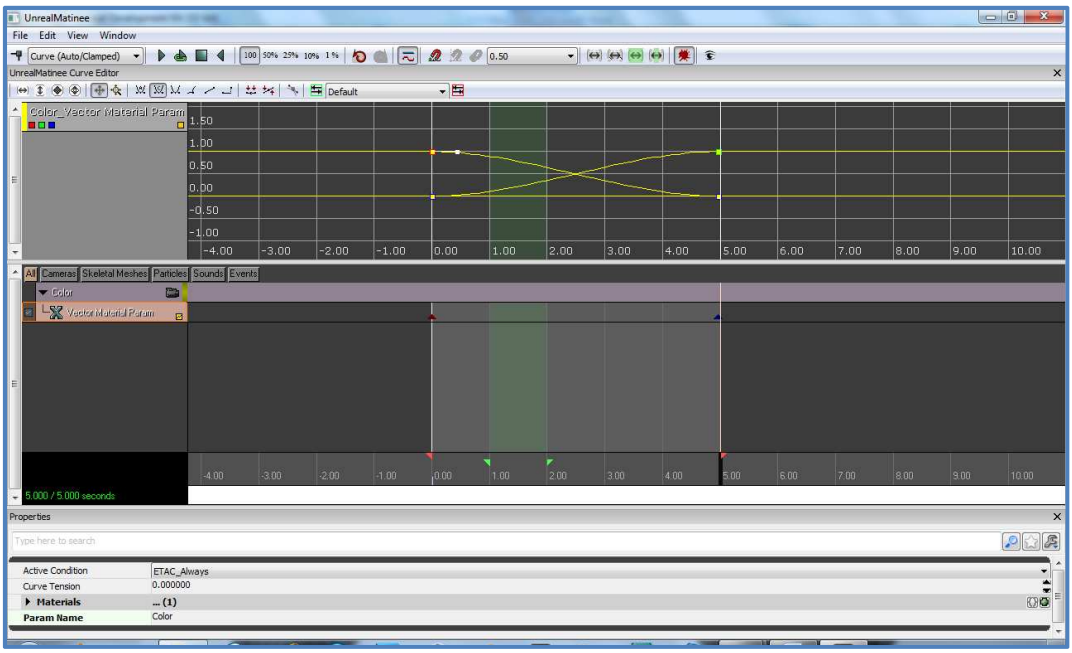


Imagen 362:Tutorial 17- Modificación de color en secuencia de tiempo.

Trasladando la línea de tiempo se puede apreciar que el color del *"mesh"* va modificándose según se va desplazando.

11. Ya se tiene implementada la forma de modificar el color del material en el elemento Matienee, pudiéndose cerrar el editor.

    En Kismet, este elemento con el cambio de color almacenado, puede ser asociado con cualquier otra acción ó evento para realizarlo.

    En el caso del tutorial se va a proceder a realizar el cambio nada más cargar el nivel, realizándose continuamente el cambio. La implementación se muestra en la imagen 363.

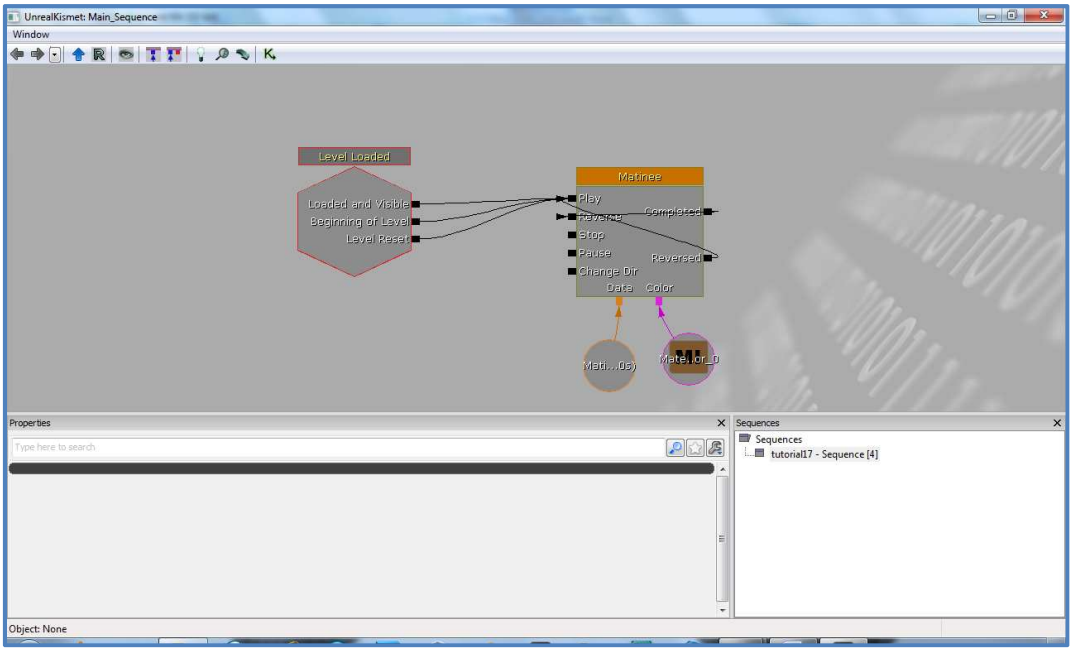


Imagen 363:Tutorial 17-Asociación "Level Loaded" y elemento "Matinee".

12. Al lanzar el nivel, el color de los *"Static Mesh"* irán cambiando continuamente entre el rojo y el verde.

Con este tutorial se ha mostrado la manera de modificar un material en ejecución. En este caso, se ha modificado sólo el color, pero mediante esta eimplementación puede ser modificado cualquier elemento del material.

8.18. Tutorial 18: Crear material "espejo".

Un material especial que se puede crear es aquel que simula a un *"espejo"*. En este tutorial se va a mostrar la manera de poder crear este tipo de material, que podrá ser acoplado a cualquier objeto.

Se va a partir de un escenario básico donde se encuentra una pared en el centro de la plataforma, tal y como se muestra en la imagen 364.

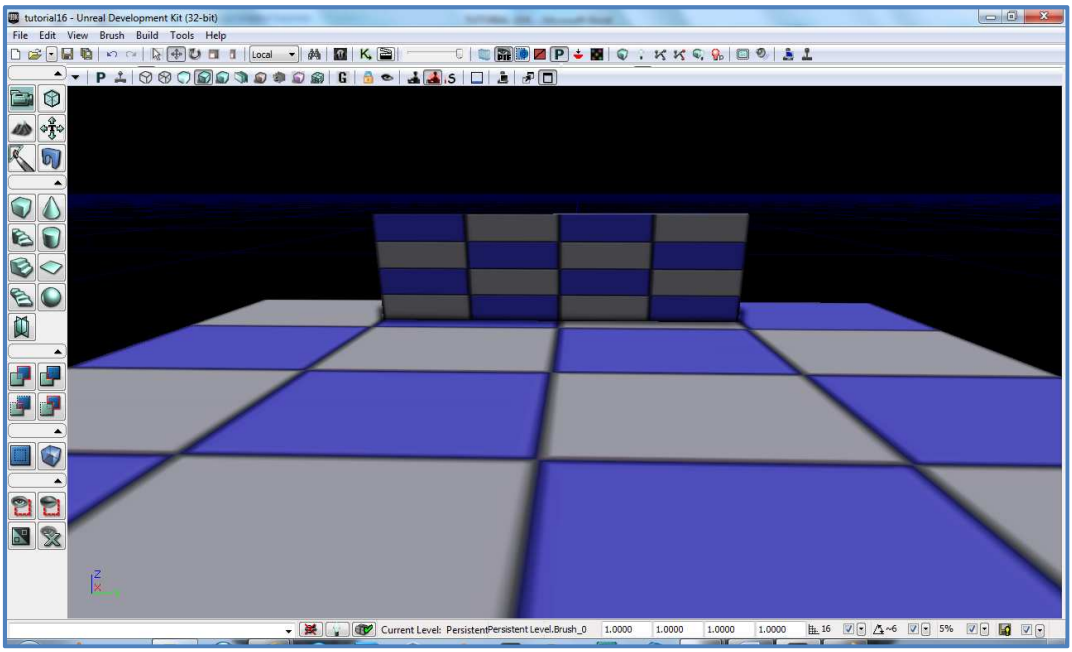


Imagen 364:Tutorial 18-Nivel inicial.

Pasos a seguir:

1. El *"espejo"* será la imagen captada por una cámara y mostrada mediante el material, simulando así un espejo. El elemento que captura la imagen es llamado *"SceneCaptureCubeMapActor"* encontrándose en la herramienta *"Actor Classes"*. Esta herramienta puede abrirse mediante el icono de la barra de herramientas superior del editor. El elemento buscado se encuentra en:

   Scene Capture Actor –> SceneCaptureCubeMapActor.

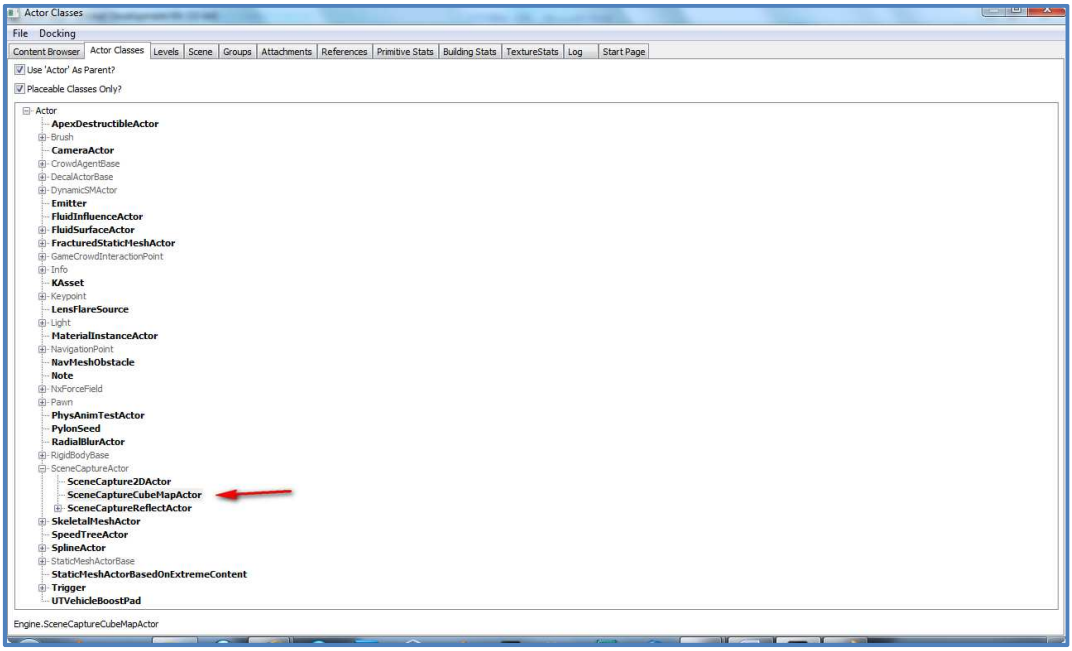


Imagen 365:Tutorial 18-"SceneCaptureCubeMapActor" en "Actor Classes".

2. Se debe añadir este elemento de captura de imagen en el editor. Colocarlo próximo de la imagen a captar. En este caso, al simular un espejo, hay que colocarlo cerca del objeto donde se acoplará el material.

   Para añadir este elemento pulsar con el botón derecho del ratón sobre cualquier vista del editor y en el menú desplegado seleccionar *"Add SceneCaptureCubeMapActor Here"*.

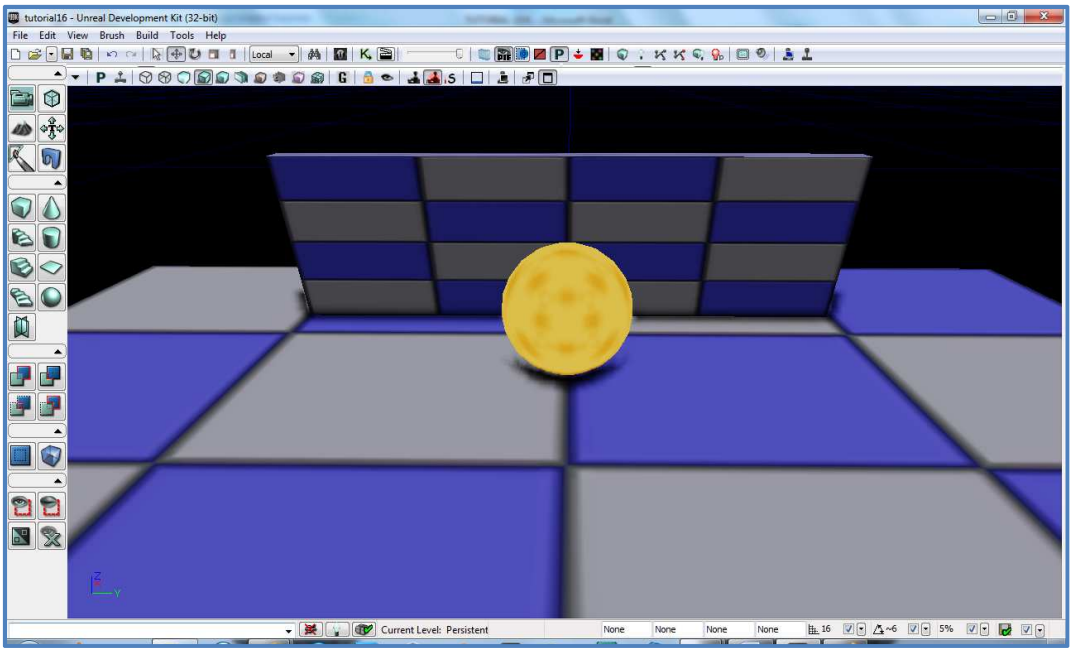


Imagen 366:Tutorial 18-Situación de "SceneCaptureCubeMapActor"en nivel.

3. Se debe volver a la herramienta Content Browser pulsando en el icono de la barra de herramientas superior del editor.

   En el directorio deseado hay que crear un elemento de tipo *"TextureRenderTargetCube"*. Para crearlo pulsar el botón derecho del ratón sobre cualquier zona libre de la zona central y en el menú mostrado seleccionar *"New TextureRenderTargetCube"* (imagen 367).

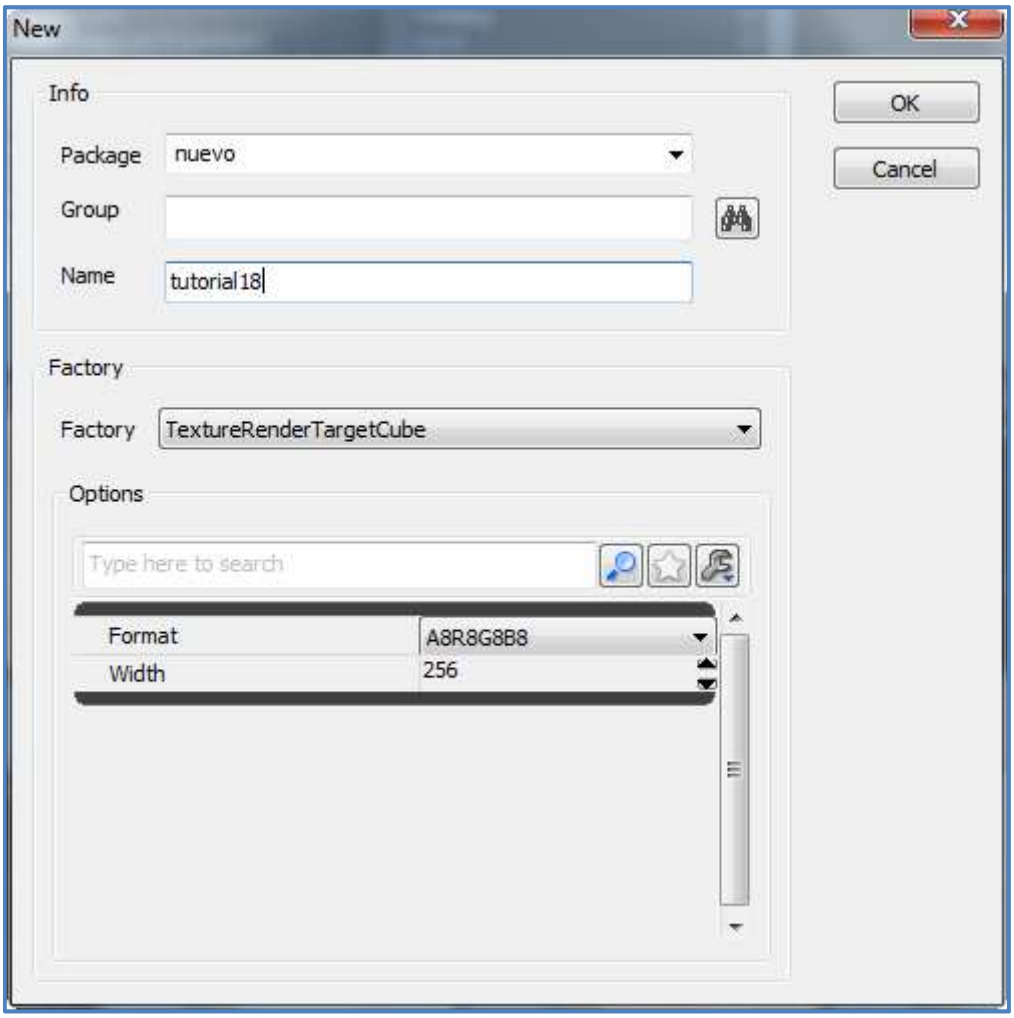


Imagen 367:Tutorial 18-Creación elemento "TextureRenderTargetCube".

4. Modificar las propiedades del elemento *"SceneCaptureCubeActor"*. Para obtener dichas propiedades, mostradas en la imagen 368, hay que seleccionar el objeto en el editor y pulsar la tecla 'F4' del teclado:

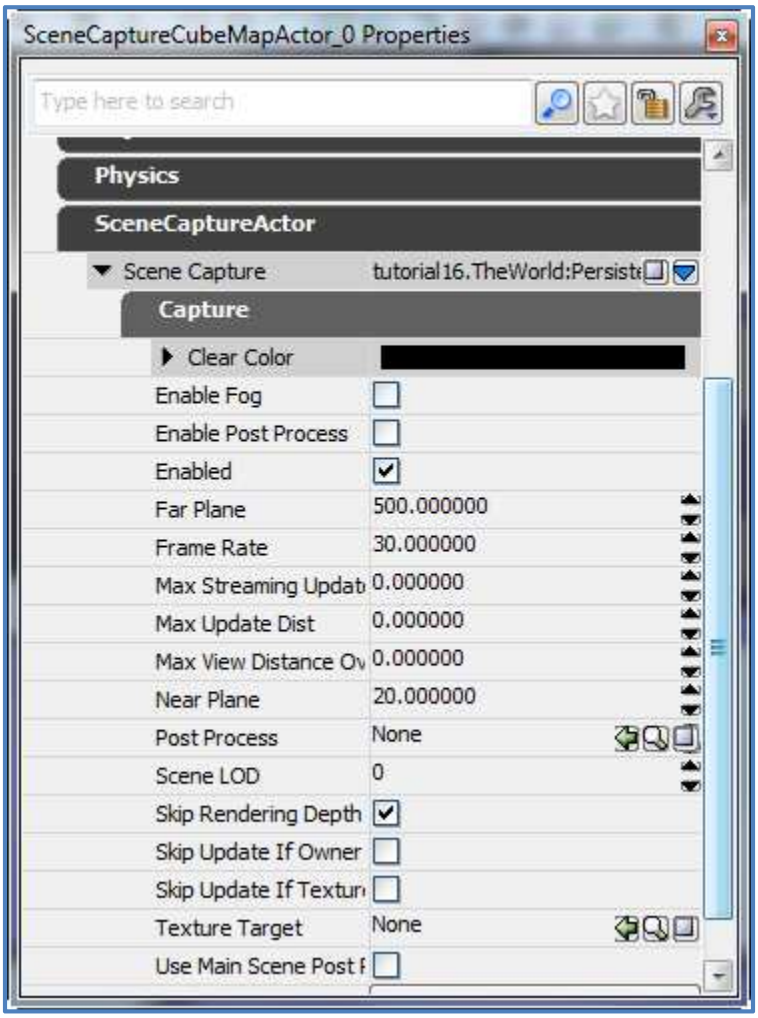


Imagen 368:Tutorial 18-Propiedades "SceneCaptureCubeMap".

En la etiqueta con nombre *"Far Plane"* se debe introducir un valor muy alto y en la etiqueta *"Texture Target"* asociar el elemento *"TextureRenderTargetCube"* creado en el punto anterior. Para poder asociarlo hay que abrir Content Browser y seleccionarlo, y sin dejar de seleccionarlo, volver a la etiqueta y presionar el icono .

5. A partir del elemento *"TextureRenderTargetCube"* crear un nuevo elemento del tipo *"Static Texture"*.

   En "Content Browser" seleccionar el elemento *"TextureRenderTargetCube"* y pulsar el botón derecho del ratón. En el menú mostrado seleccionar *"Create Static texture…"*, introducir el nombre deseado a esta nueva textura y pulsar el botón *"OK"*:

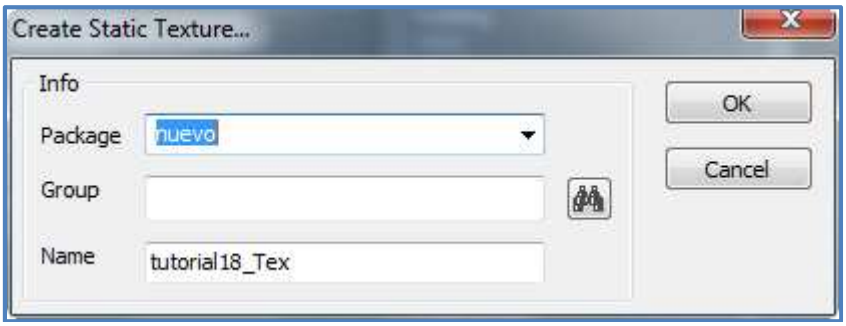


Imagen 369:Tutorial 18-Creación "Static Texture".

6. Con los elementos creados anteriormente puede crearse el material que representará el espejo. Para crear un elemento de tipo material, en el Content Browser pulsar el botón derecho del ratón y en el menú mostrado hay que seleccionar "*New Material*":

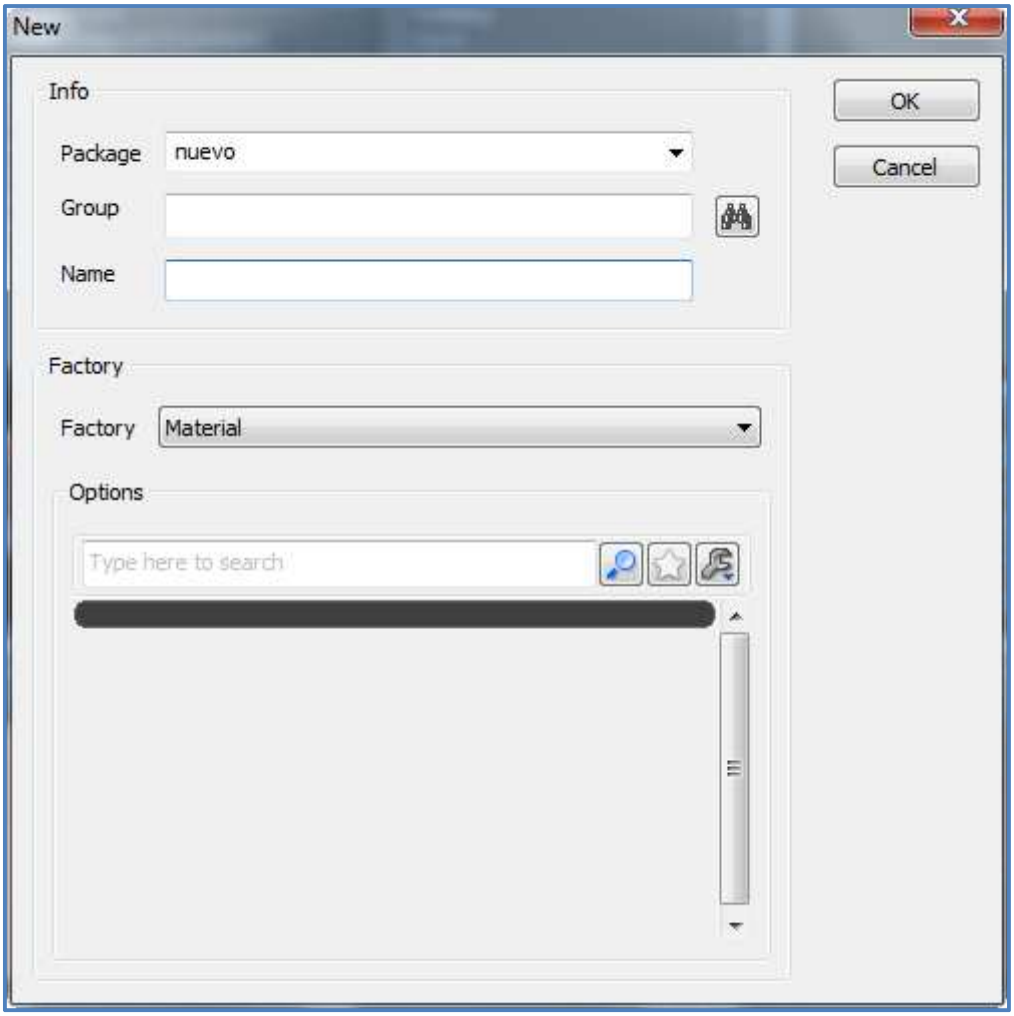


Imagen 370:Tutorial 18-Creación de material.

7. Se abrirá la herramienta "*Unreal Material Editor*", utilizada para crear un nuevo material:

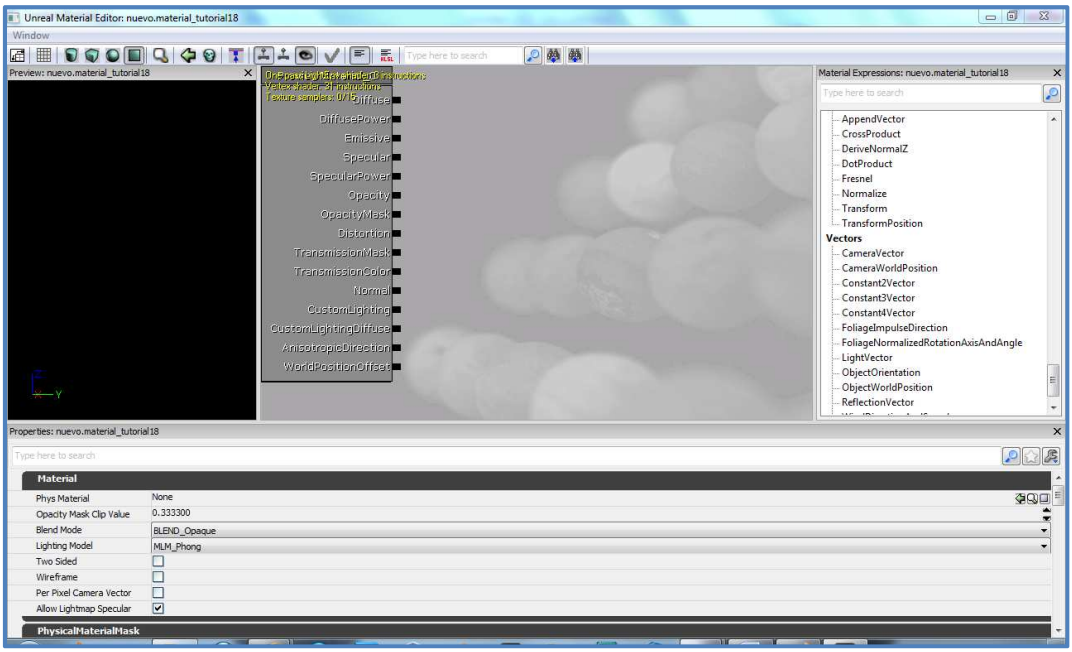


Imagen 371:Tutorial 18-Interfaz "Unreal Material Editor".

Mediante el directorio de elementos mostrados en la sección derecha del editor se deben añadir los siguientes elementos:

- Dos elementos "*Multiply*".
- "*Fresnel*".
- "*Constant*".- Con valor 1.

Los elementos añadidos son mostrados en la siguiente imagen:

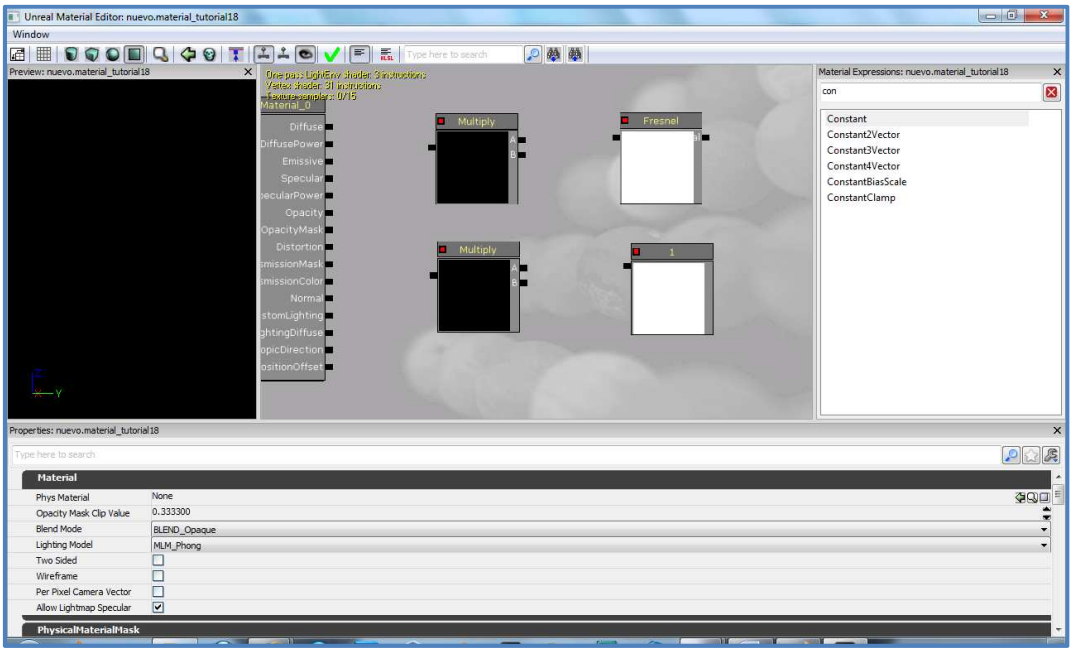


Imagen 372:Tutorial 18-Creación elementos "Multiply", "Fresnel" y "Constant".

Para obtener los distintos elementos se puede utilizar el buscador que se encuentra en la parte superior de la sección de elementos. Los elementos son añadidos a la parte central seleccionándolos y arrastrándolos a la parte central.

8. Los elementos añadidos en el material se deben asociar de la siguiente manera:
   Las salidas de un elemento *"Multiply"* se unen con las entradas del elemento *"Fresnel"* y del elemento *"Constant"*.

   La entrada de este elemento *"Multiply"* hay que unirla a una de las salidas del otro elemento *"Multiply"*. Todo ello es mostrado en la imagen 373:

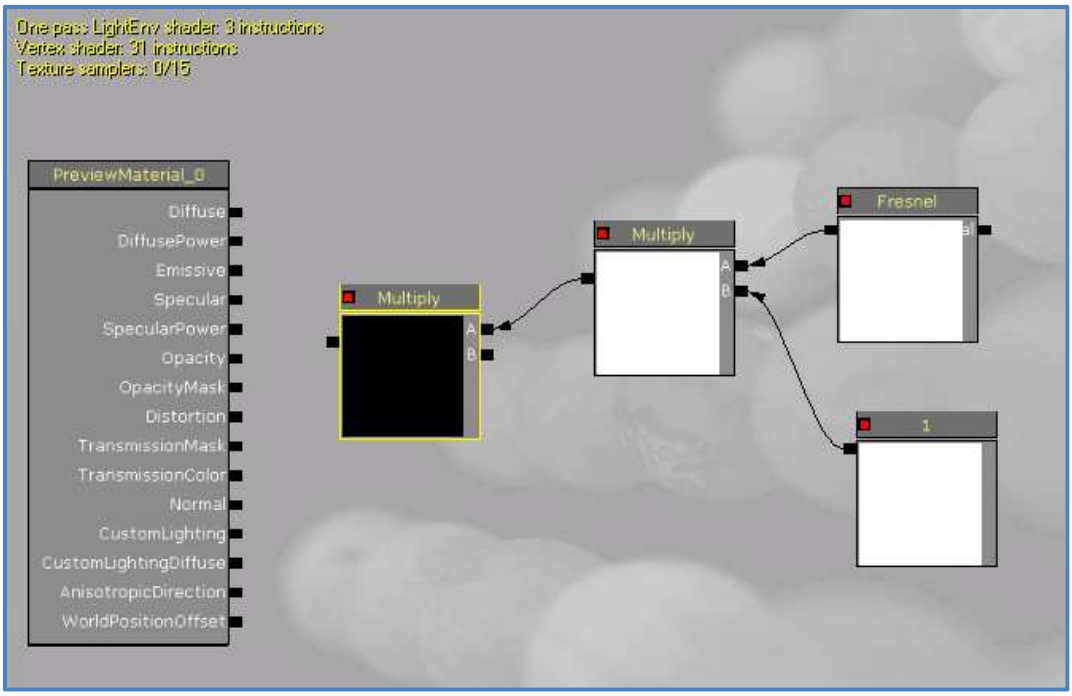


Imagen 373:Tutorial 18-Asociación elementos.

9. Crear otra secuencia de elementos. Se añade un elemento llamado *"Texture Sample"* en el que a la salida se asociará el elemento *"Transform"* y a la salida de éste se asociará el elemento *"Reflection Vector"*:

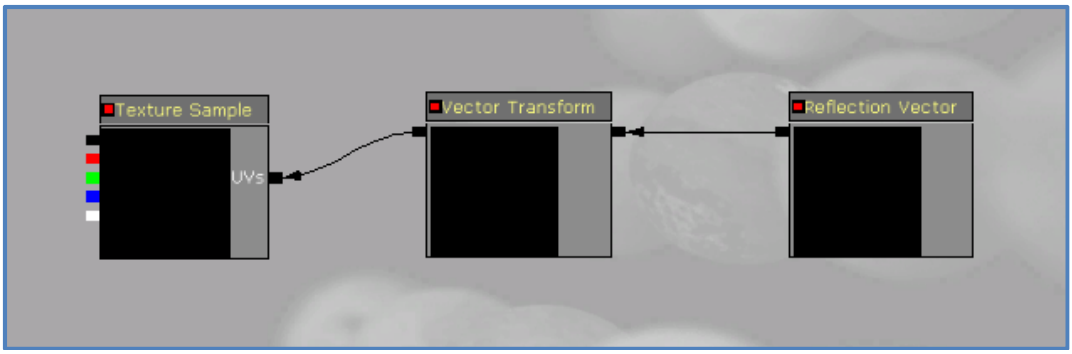


Imagen 374:Tutorial 18-Creación de elemento de textura.

10. La secuencia creada en el punto anterior hay que unir su primera entrada, de color negra, con la salida del *"Multiply"* libre, mostrado en la siguiente imagen:

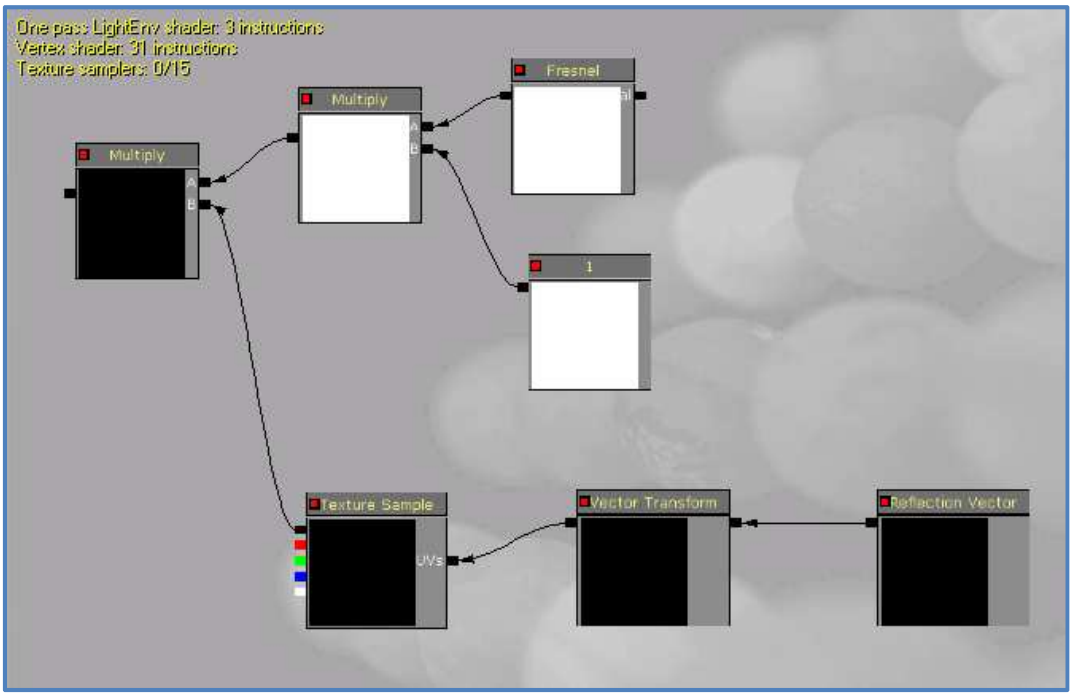


Imagen 375:Tutorial 18-Asociación del conjunto completo.

11. El elemento *"Texture Sample"* debe ser modificado para asociarlo con el elemento *"TextureRenderTargetCube"* creado en puntos anteriores.

    Seleccionando el elemento *"Texture Sample"* se mostrarán sus propiedades en la zona inferior del editor. La etiqueta a modificar es la llamada *"Texture"*, imagen 376. Para asociar el elemento *"TextureRenderTargetCube"* hay que volver al Content Browser y seleccionarlo, y con el elemento seleccionado, pulsar el icono de la misma etiqueta.

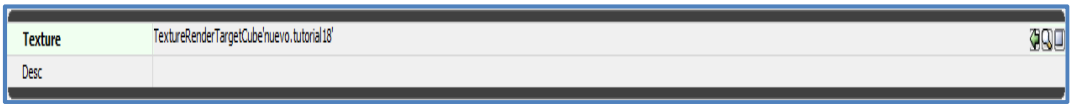


Imagen 376:Tutorial 18-Propiedades "Texture".

12. Para finalizar con la edición del material unir toda la secuencia de elementos con las características principales del material.

    La entrada del elemento *"Multiply"* hay que unirla con la salida *"Diffuse"* ó con la salida *"Emissive"*:

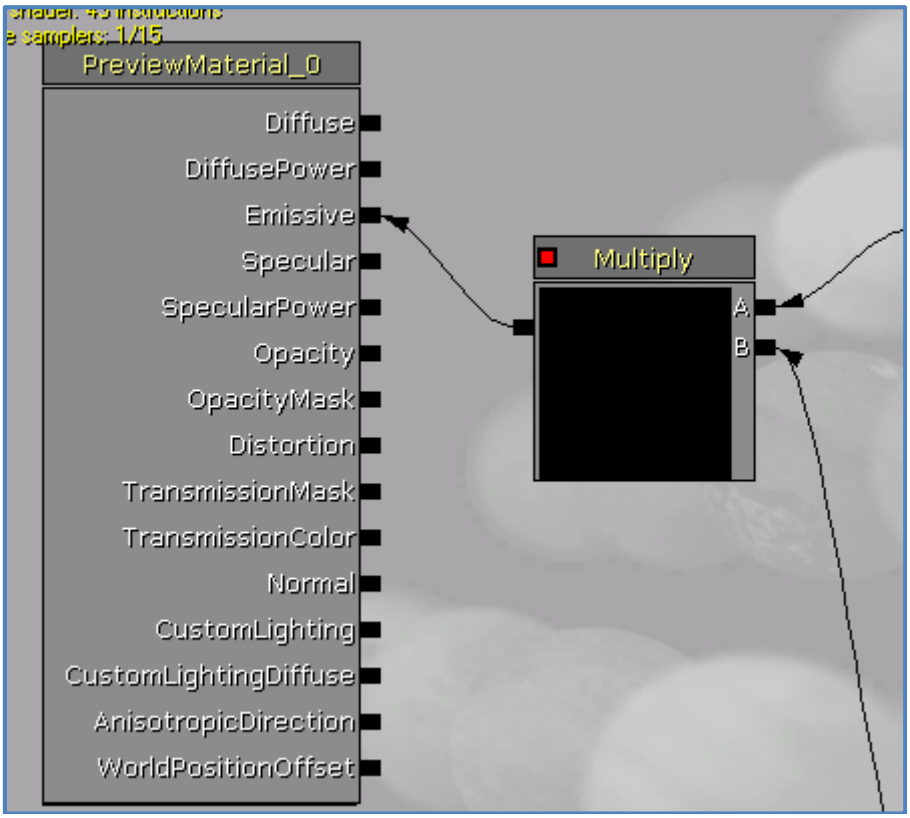


Imagen 377:Tutorial 18-Enlace secuencia con propiedades del material.

13. Para almacenar el material se utiliza el icono situado en la barra de herramientas superior. Mediante esta acción el material queda guardado y creado pudiéndose cerrar el editor.

14. En el Content Browser se dispone del material *"espejo"* creado para poder acoplarlo a cualquier objeto del nivel.

    Para acoplar el material a un objeto, éste se debe seleccionar, y volviendo al editor, sobre el objeto, pulsar el botón derecho del ratón y seleccionar *"Apply Material: material_tutorial18"*.

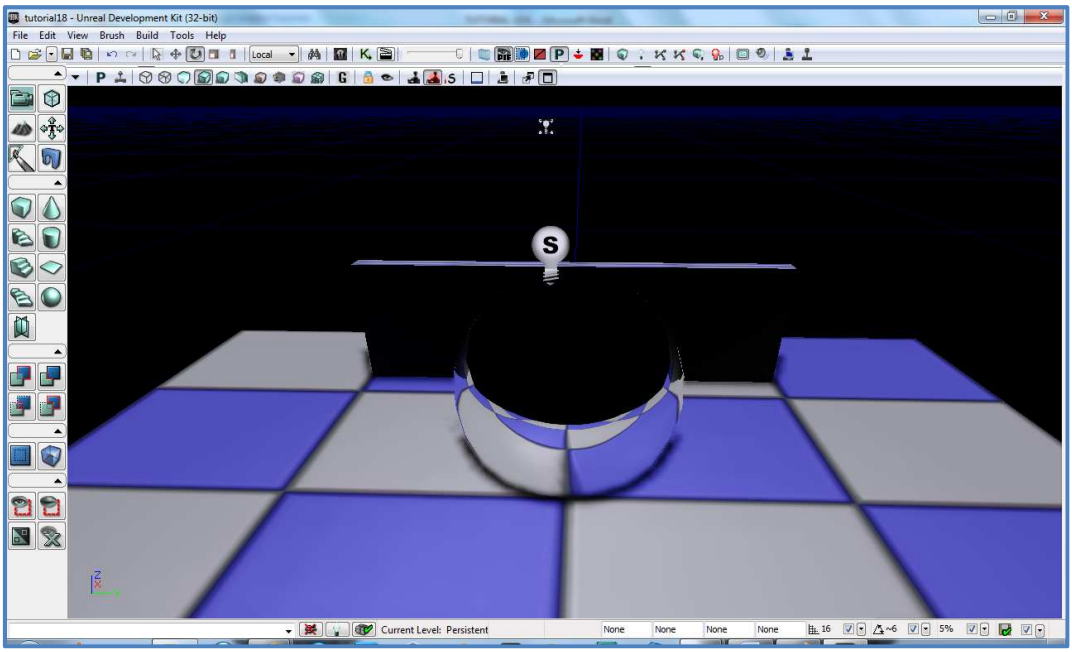


Imagen 378:Tutorial 18-Nivel con material espejo acoplado.

Con el material acoplado a la pared vertical, el jugador será mostrado en ella al acercarse, simulando un espejo.

8.19. Tutorial 19: Crear material "lluvia".

El fenómeno atmosférico *"lluvia"* puede simularse en los niveles para dotarlos de mayor realismo. Para poder simular *"lluvia"* es necesario crear un nuevo material que represente a esta "lluvia", el cual será acoplado a planos verticales.

Pasos a seguir:

1. Crear un nuevo elemento de tipo *"Material"* en la herramienta Content Browser accesible mediante el icono de la barra de herramientas superior del editor.

   Para crear un elemento de tipo *"Material"*, en el Content Browser pulsar el botón derecho del ratón y en el menú mostrado seleccionar *"New Material"*:

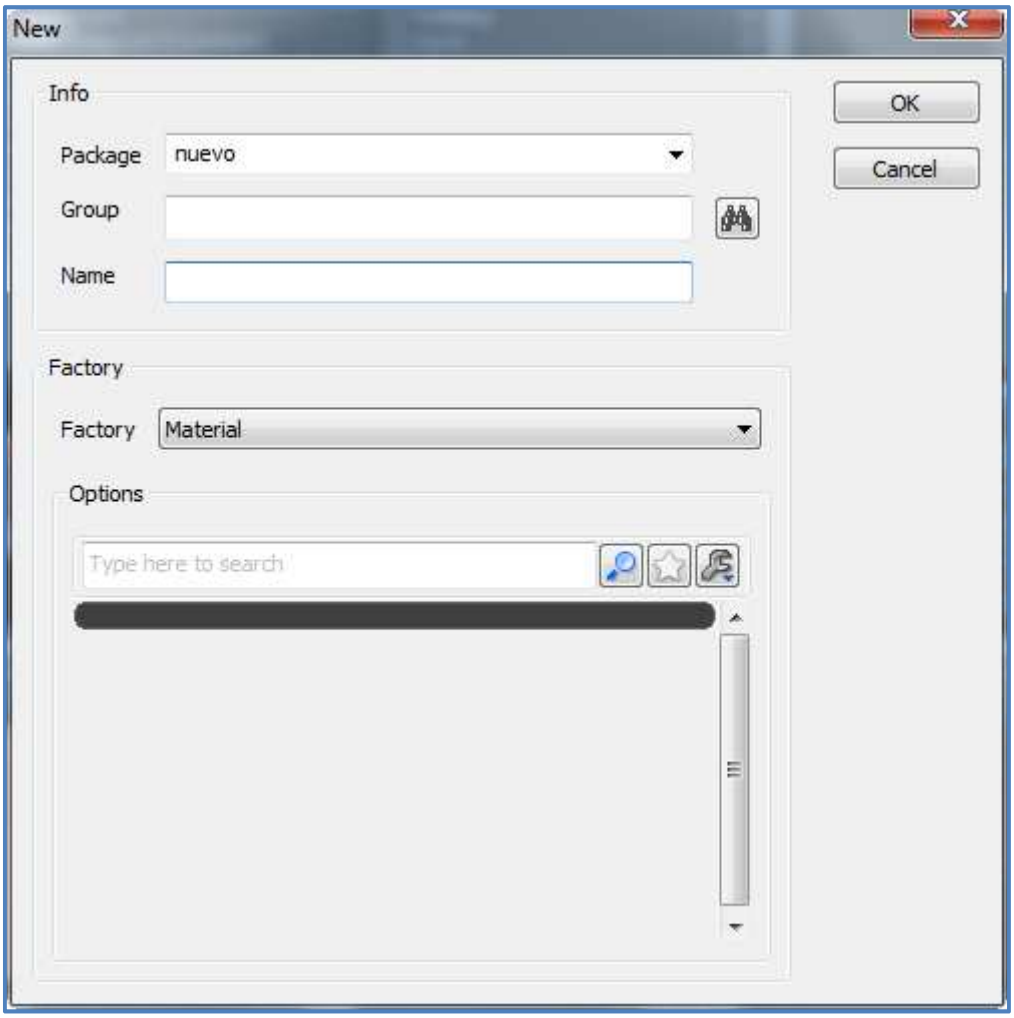


Imagen 379:Tutorial 19-Creación de material.

2. Se abrirá la herramienta *"Unreal Material Editor"*, utilizada para crear un nuevo material:

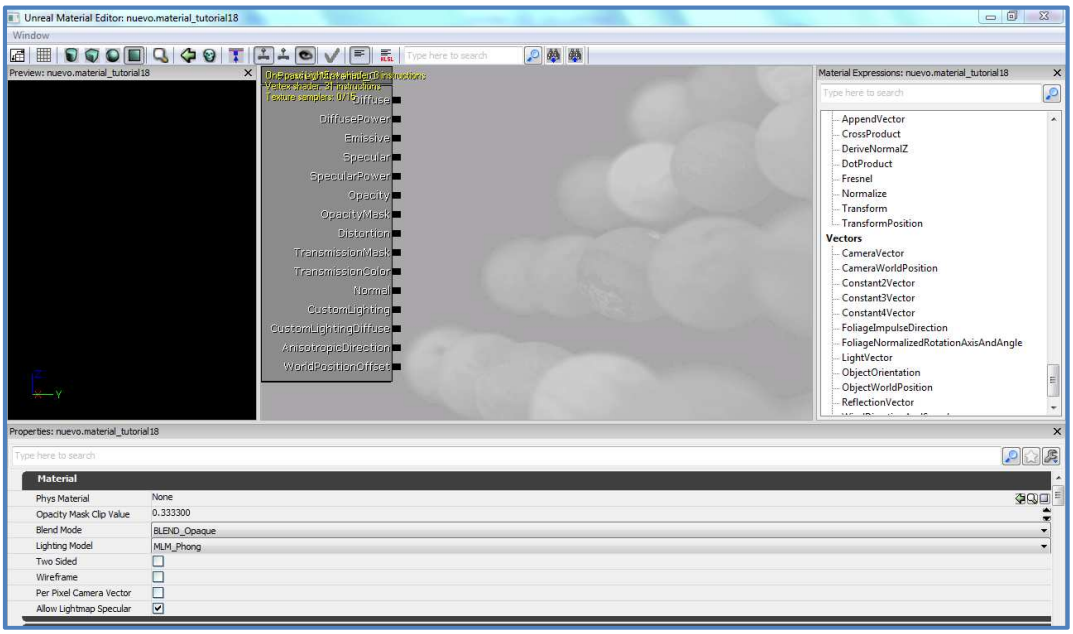


Imagen 380:Tutorial 19-Interfaz "Unreal Material Editor".

3. Para realizar el material de lluvia es necesaria una textura que asemeje la lluvia de la naturaleza. Esta textura debe ser una imagen de fondo negro con los elementos de agua en color blanco. Un ejemplo de esta textura sería:

Imagen 381:Tutorial 19-Textura de lluvia.

Esta imagen debe ser importada en el editor para poder usarla. Para importar una textura se pueden utilizar las herramientas ofrecidas por Content Browser. Hay que tener en cuenta que al importar una textura, ésta debe tener dimensiones cuadradas y con dimensiones múltiplo de 32, por ejemplo: 256 x 256, 512 x 512, etc.

4. Añadir el elemento que permite utilizar la textura importada en el punto anterior. Este elemento es llamado "*Texture Sample*" pudiéndose localizar gracias al buscador de la zona superior derecha.

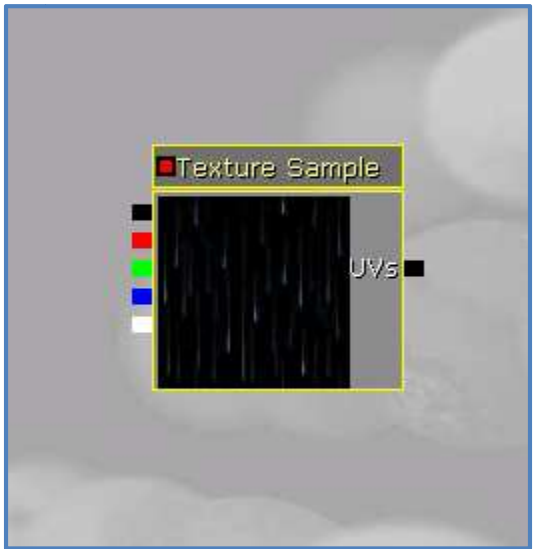


Imagen 382:Tutorial 19-Elemento "Texture Sample".

Al añadir el elemento en el editor del material, si se tiene seleccionada la textura en el Content Browser, dicho elemento tendrá asociada dicha textura.

En el caso que no se asocie automáticamente, la textura hay que indicarla en la etiqueta *"Texture"* de las propiedades del elemento. En el Content Browser se selecciona la textura y pulsando en el icono de la etiqueta, la textura quedará añadida.

5. Asociar este elemento con las propiedades generales del material. Se debe unir la primera de las entradas del elemento *"Texture Sample"* con la salida *"Emisive"* de las propiedades generales del material:

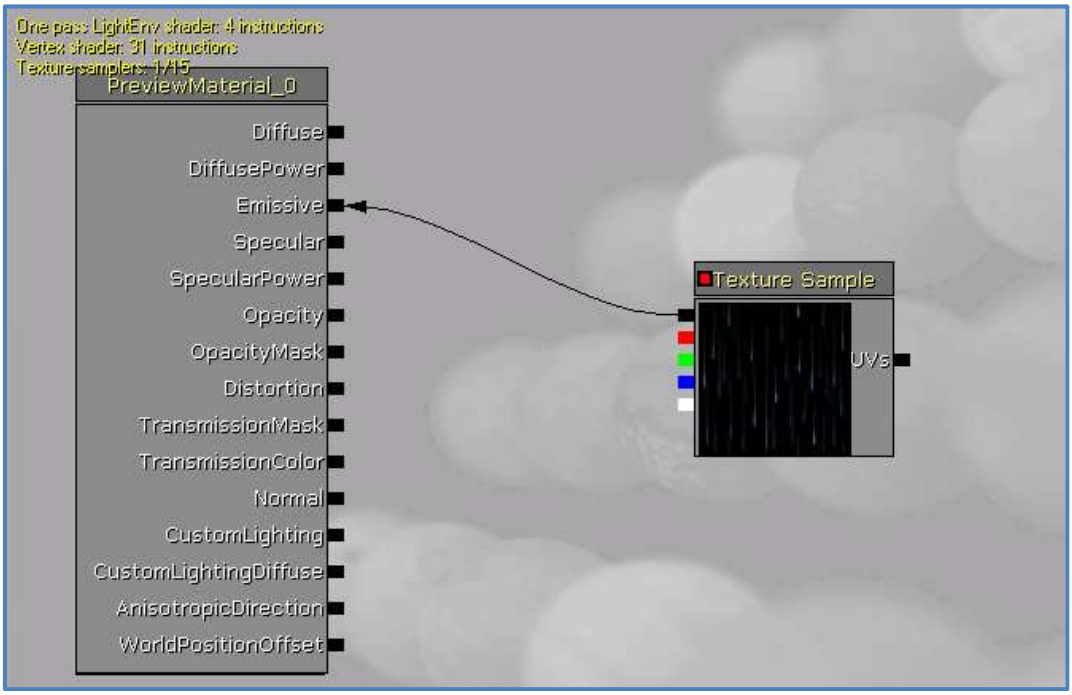


Imagen 383:Tutorial 19-Asociación "Texture Sample" con material.

6. Las propiedades generales del material deben ser modificadas para cumplir las condiciones deseadas.

   En la etiqueta *"Blend_Mode"* seleccionar *"BLEND_Translucent"*.

   En la etiqueta *"Lighting Model"* seleccionar *"MLM_Unlit"*.

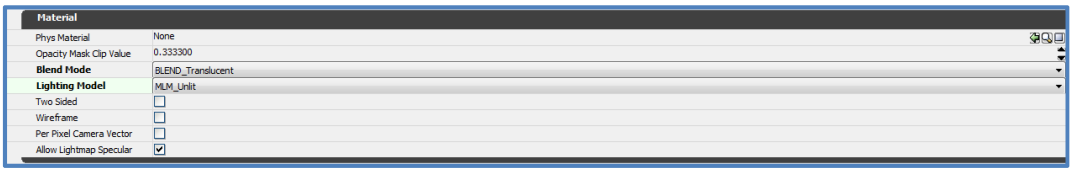


Imagen 384:Tutorial 19-Propiedades del material.

7. La entrada de color rojo del elemento *"Texture Sample"* debe ser asociada con la salida *"Opacity"* de las características generales del material:

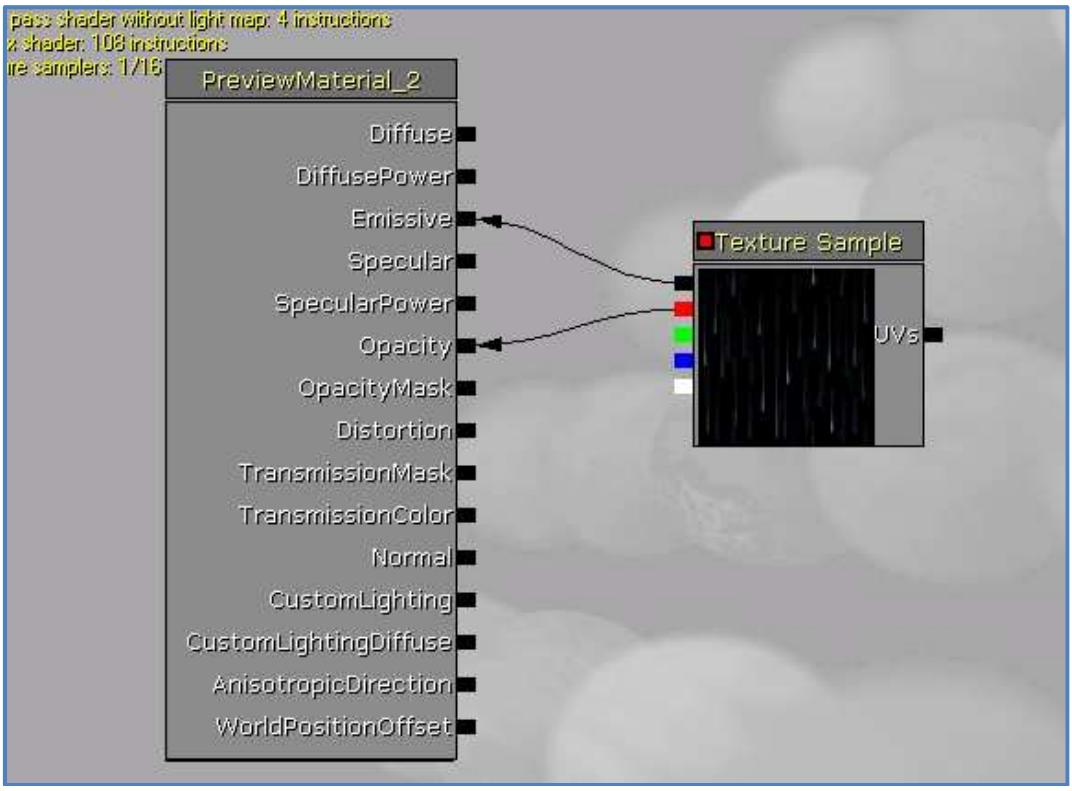


Imagen 385:Tutorial 19-Asociación "Texture sample" con "Opacity".

8. Añadir el elemento de nombre *"Panner"* para poder mover la textura en los ejes.

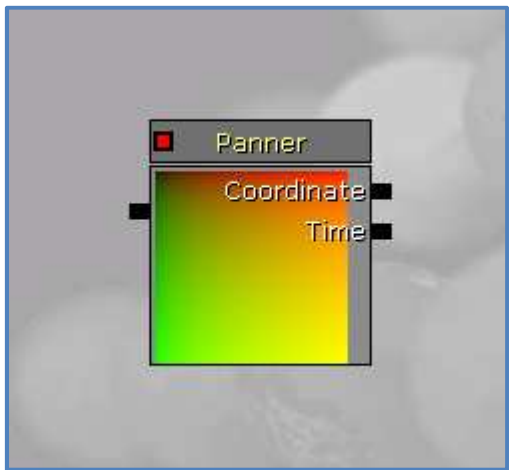


Imagen 386:Tutorial 19-Elemento "Panner".

Este elemento permite mover la textura en los ejes X e Y con una velocidad dada. En las propiedades se proporcionan las etiquetas *"Speed X"* y *"Speed Y"* donde se puede indicar la velocidad en que se desplace la textura en cada uno de los ejes.

En el caso de la textura de lluvia sólo se quiere desplazar en el eje Y, insertando el valor de la velocidad deseada. El elemento *"Panner"* se asociará a *"Texture Sample"* para realizar el movimiento dado:

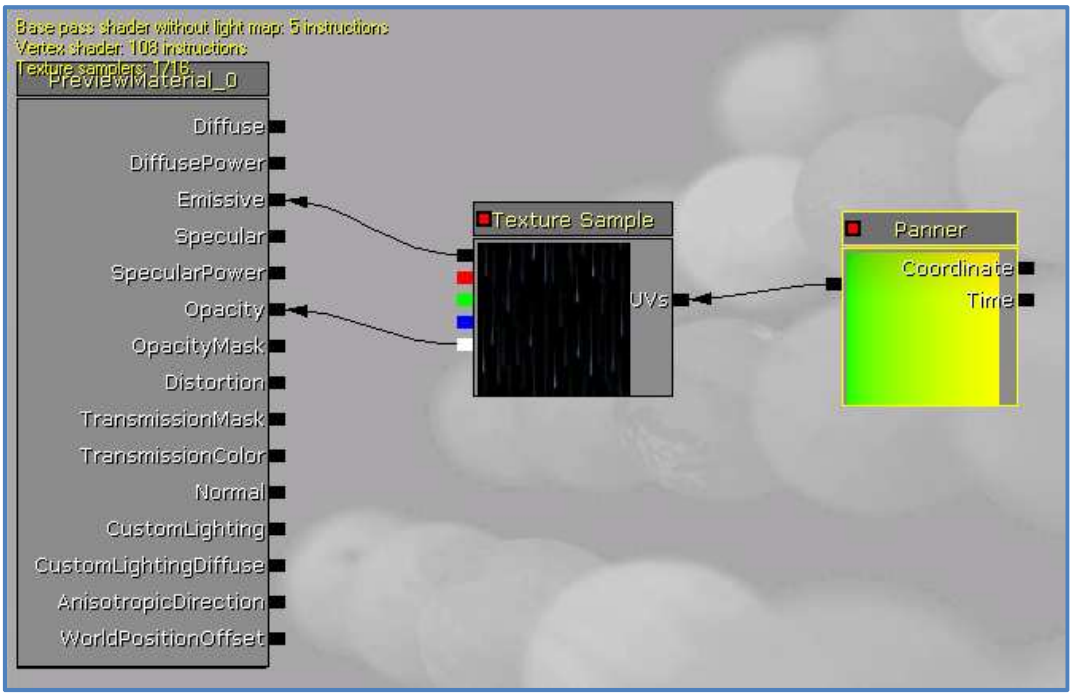


Imagen 387:Tutorial 19-Asociación de "Panner" y "Texture Sample".

9. Otro elemento que se puede añadir al material de lluvia es el llamado *"Texture Coordinate"*. Este elemento permite elegir las veces a replicar la textura en los dos ejes de coordenadas.

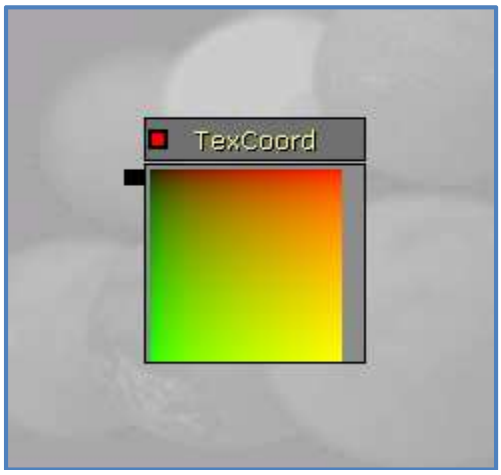


Imagen 388:Tutorial 19-Elemento "TextCoord"

En las etiquetas *"UTiling"* y *"VTiling"* de las propiedades de este elemento se indican las veces que se debe replicar la textura en los ejes de coordenadas X eY.

La asociación se este elemento se realiza de la manera indicada en la siguiente imagen:

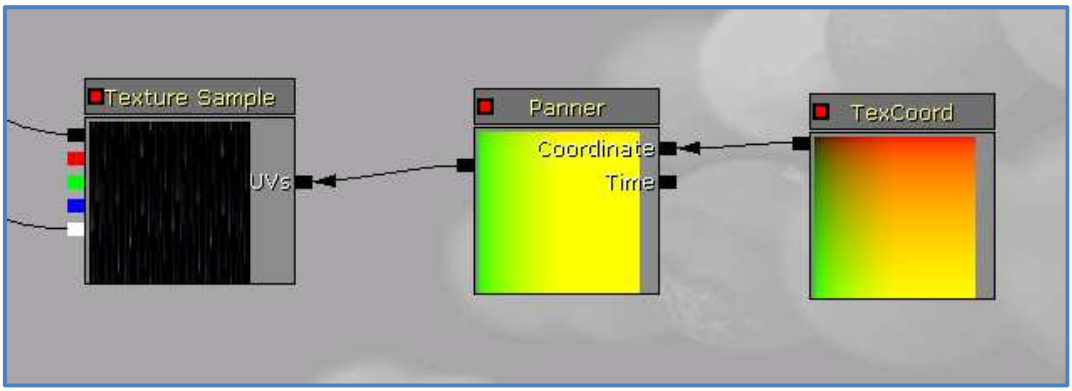


Imagen 389:Tutorial 19-Asociación de "TexCoord" con "Panner".

10.Con todos estos elementos añadidos ya se tendría el material de lluvia creado para poder utilizarlo. Su implementación completa sería de la manera:

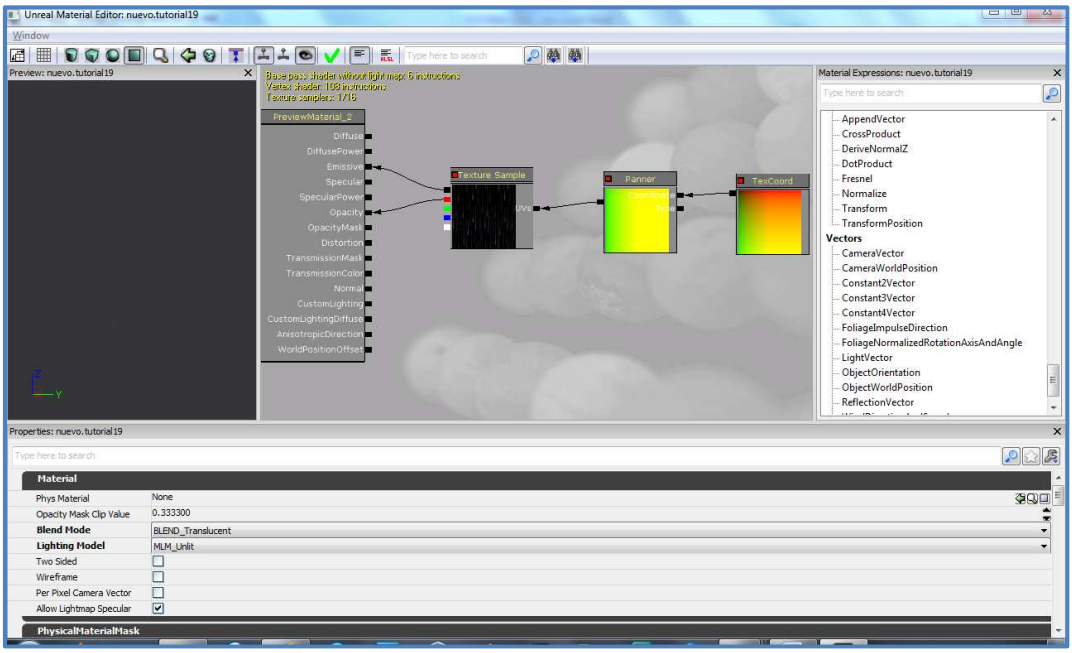


Imagen 390:Tutorial 19-Vista implementación en editor.

Para almacenar todos los cambios realizados en el material hay que pulsar el icono de la barra de herramientas superior.

11.El material creado puede ser utilizado en el nivel. El material de lluvia se acoplaría a planos verticales creados. Si el material se encuentra seleccionado en el Content Browser, al crear el plano ya tendría acoplado el material.

En la imagen 391 se muestra un ejemplo de un nivel con el material de lluvia acoplado:

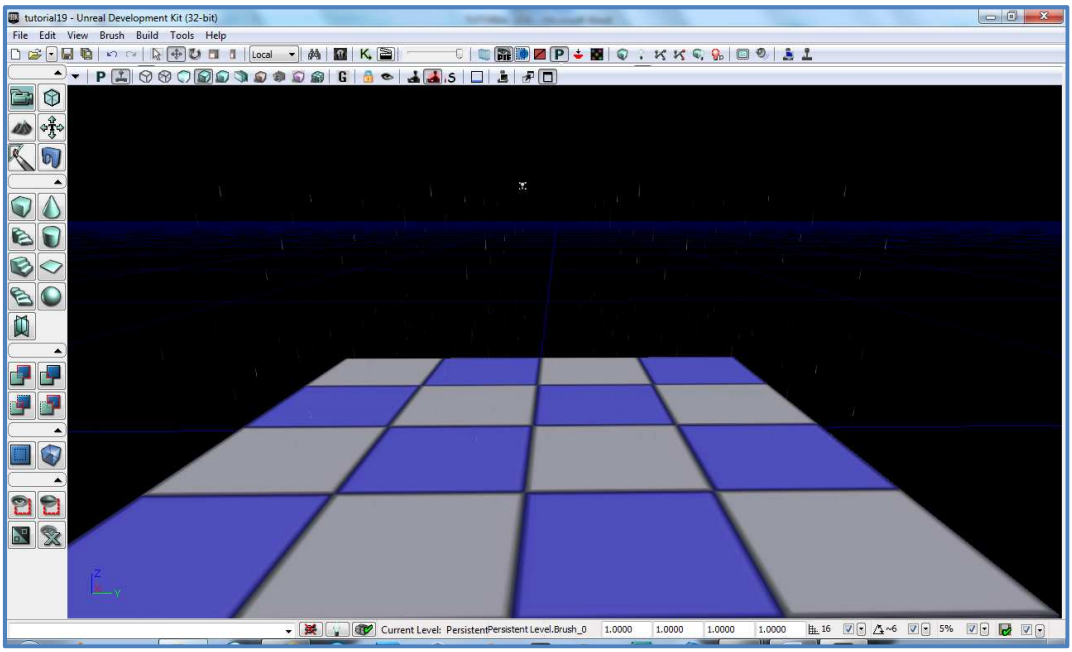


Imagen 391:Tutorial 19-Ejemplo de uso de lluvia.

8.20. Tutorial 20: Acoplar vídeo como material.

El motor UDK permite utilizar videos como material para acoplar en los objetos. El vídeo debe ser importado en el motor para poder utilizarlo. Para poder importar el vídeo, primero éste debe estar en la extensión correcta reconocida por el motor, siendo *".bink"*.

Existe la herramienta gratuita llamada *"RAD Video Tools"* [51], imagen 392, utilizada para poder transformar cualquier vídeo en la extensión *".bink"*:

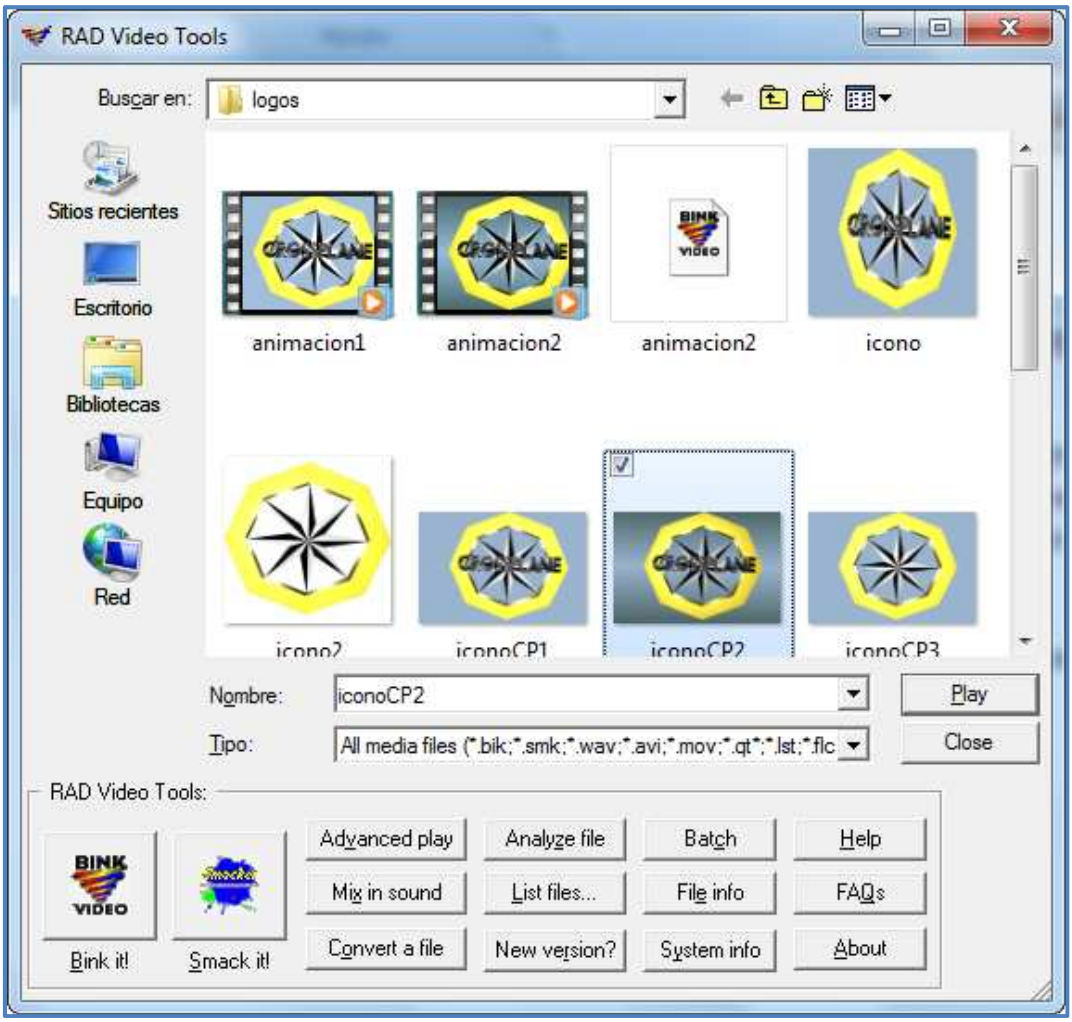


Imagen 392:Tutorial 20-Aplicación "RAD Video Tools".

Para poder transformar el vídeo simplemente seleccionarlo y pulsar el botón de la parte izquierda inferior *"Bink it"*.

Al pulsar este botón se mostrará una nueva ventana, imagen 393, donde se podrá indicar el nombre del archivo *".bik"* a crear y otras propiedades más referentes al vídeo:

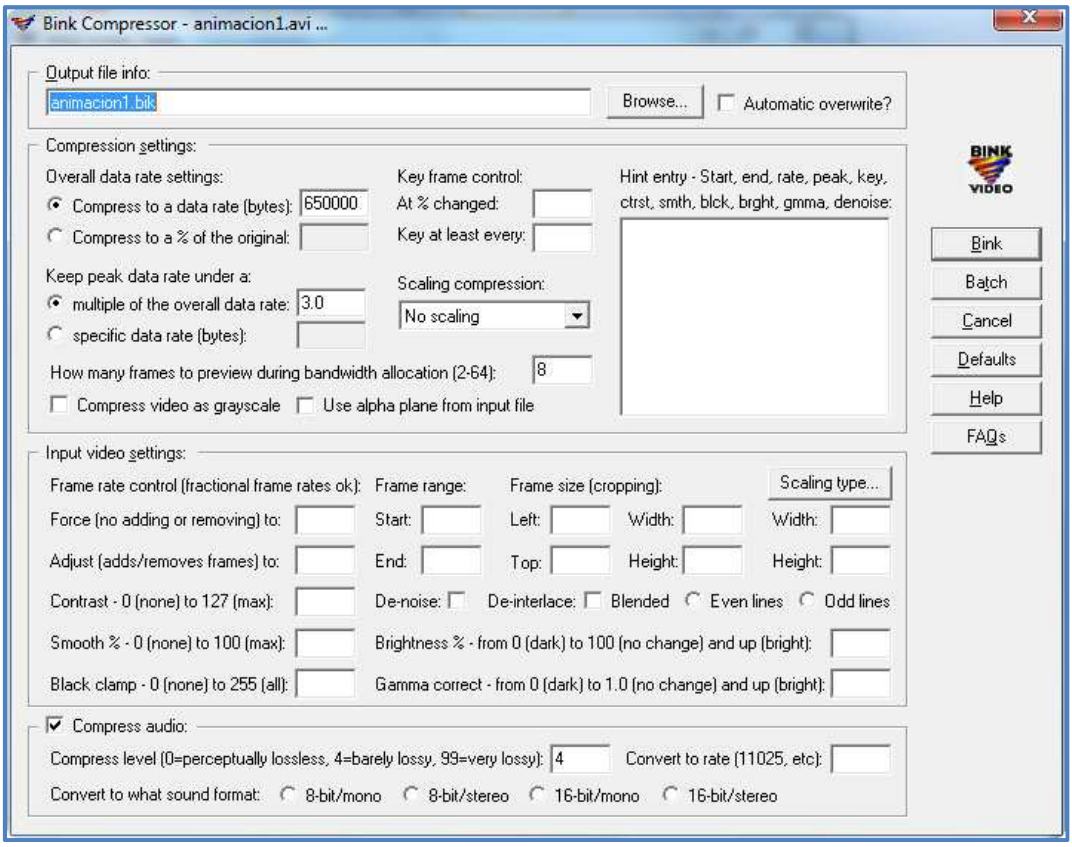


Imagen 393:Tutorial 20-Pantalla conversión a ".bink".

Pulsando en el botón *"Bink"* de la parte derecha de la ventana se obtendrá el archivo de vídeo con la extensión correcta para UDK.

El vídeo debe ser importado como elemento de tipo *"TextureMovie"* propio del editor. Al realizar la importación mediante los procedimientos proporcionados por la herramienta Content Browser, se creará automáticamente el elemento correcto del vídeo embebido en una textura. En este tipo de textura se reproduce el vídeo de forma continua.

Con la textura debe crearse un elemento de tipo *"Material"* que será el que se acople a los objetos del nivel.

Pasos a seguir:

1. Crear un nuevo elemento de tipo *"Material"* en la herramienta Content Browser accesible mediante el icono de la barra de herramientas superior del editor.

   Para crear un elemento de tipo *"Material"*, pulsar el botón derecho del ratón y en el menú mostrado seleccionar *"New Material"* proporcionando el nombre deseado:

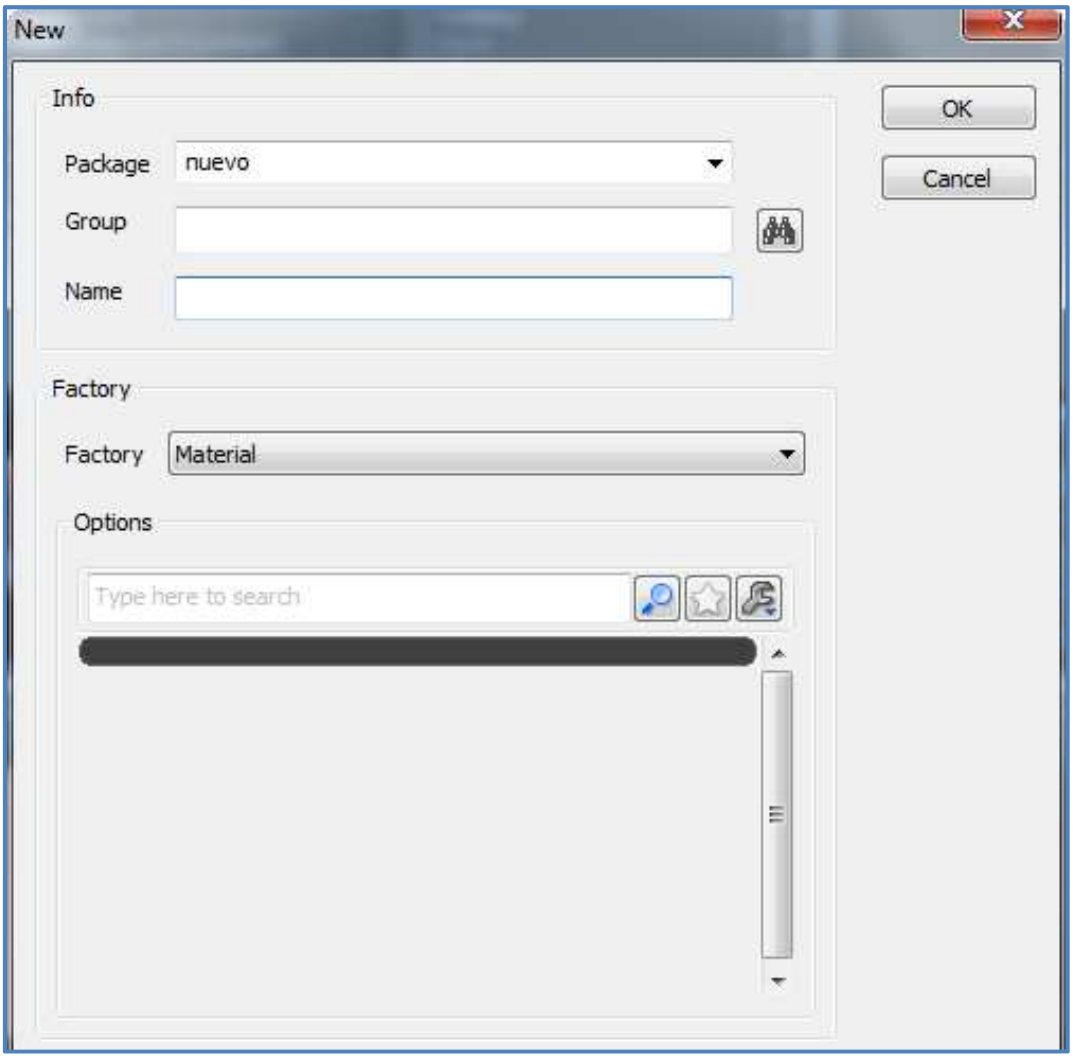


Imagen 394:Tutorial 20-Creación de material.

2. Un nuevo material es editado mediante la herramienta *"Unreal Material Editor"*:

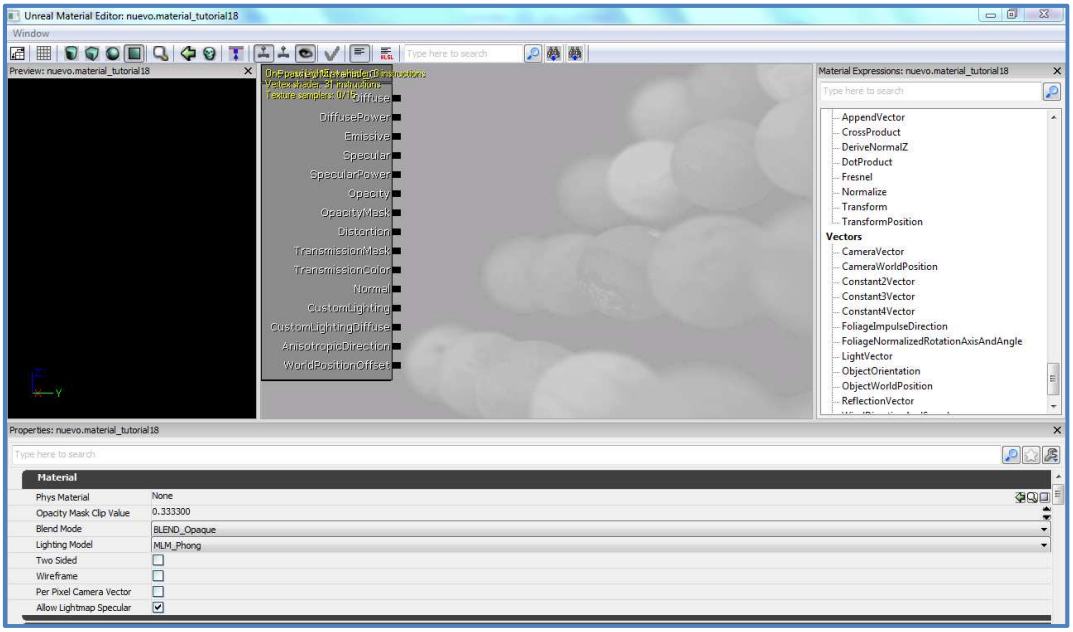


Imagen 395:Tutorial 20-Interfaz "Unreal Material Editor".

3. En el editor del material se debe añadir un elemento de tipo *"Texture Sample"* que es el que contendrá la textura del vídeo importada anteriormente.

   Se puede localizar este elemento mediante el buscador de la parte superior derecha y arrastrarlo a la parte central.

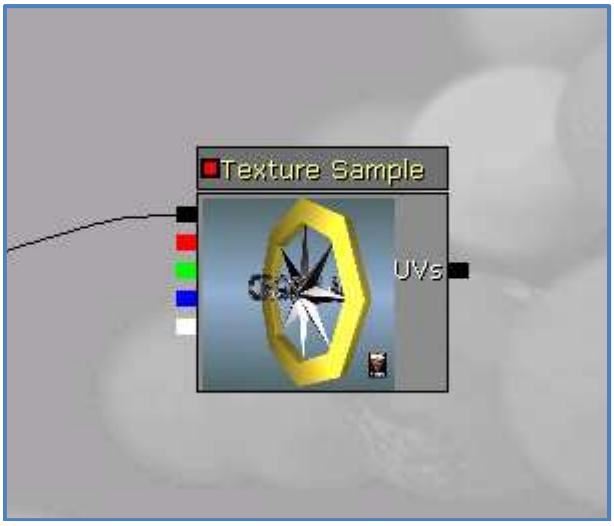


Imagen 396:Tutorial 20-Elemento"Texture Sample" con video acoplado.

   Al crear el elemento en el editor del material, si se tiene seleccionada la textura en el Content Browser, dicho elemento tendrá asociada dicha textura automáticamente.

   En el caso que no se asocie automáticamente, la textura hay que indicarla en la etiqueta *"Texture"* de las propiedades del elemento. En Content Browser se selecciona la textura y pulsar en el icono de la etiqueta, quedando añadida la textura.

4. Se debe asociar el elemento creado con las propiedades generales del material.
   La entrada de color negro del elemento *"Texture Sample"* se asociará con la salida *"Emissive"* de las propiedades generales del material:

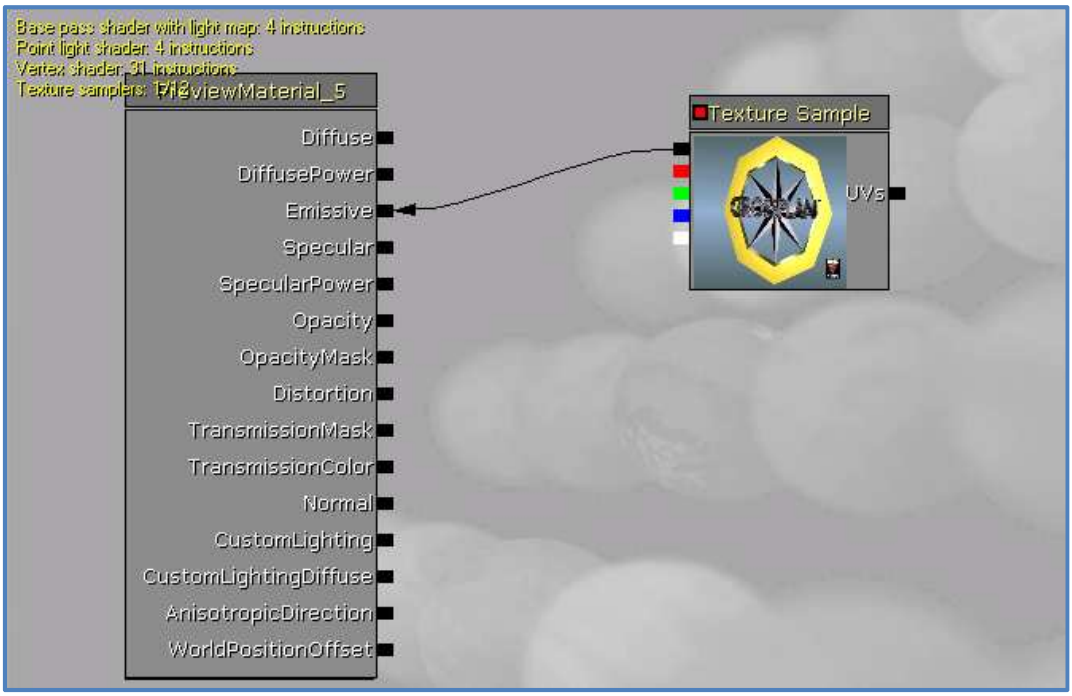


Imagen 397:Tutorial 20-Asociación del "Texture Sample" con material.

5. Mediante estas acciones se tiene completo el material que contiene el vídeo para poder acoplarlo en los objetos del nivel. La implementación completa de este material se presenta en la siguiente imagen:

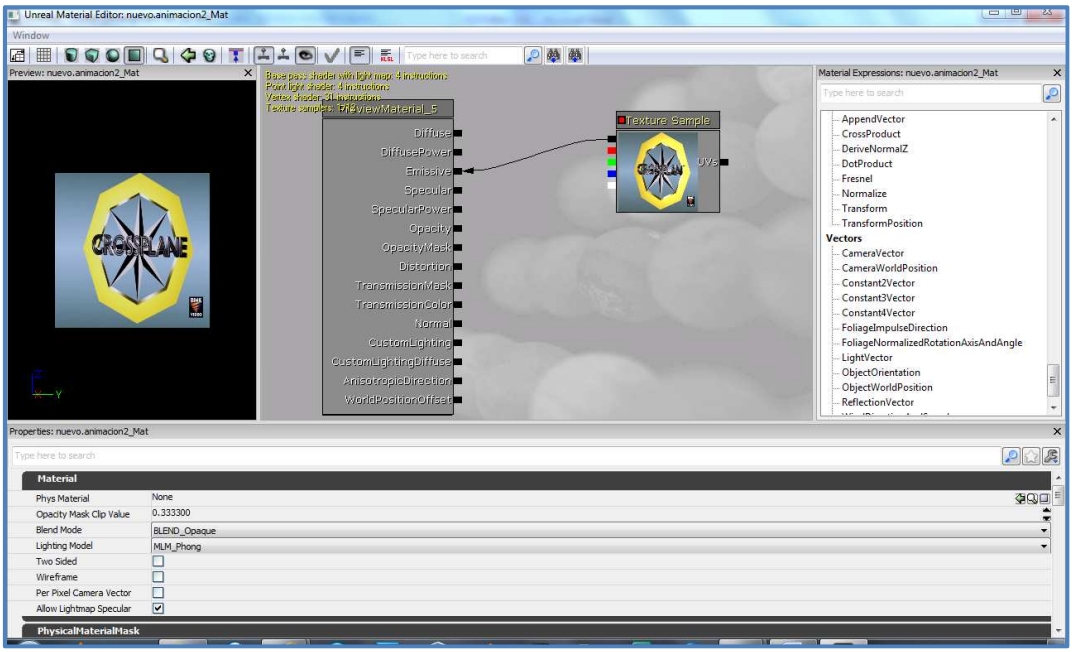


Imagen 398:Tutorial 20-Implementación completa en editor.

Para almacenar todos los cambios realizados hay que pulsar el icono de la barra de herramientas superior.

6. El material creado puede ser acoplado a cualquier objeto del nivel. Para poder acoplar el material se selecciona en el Content Browser y sobre el objeto pulsar el

botón derecho del ratón y en el menú mostrado seleccionar *"Apply material: nombre_material"*.

7. Al acoplar el material, el vídeo no es mostrado correctamente. Hay que modificar las propiedades del objeto para mostrarlo bien.

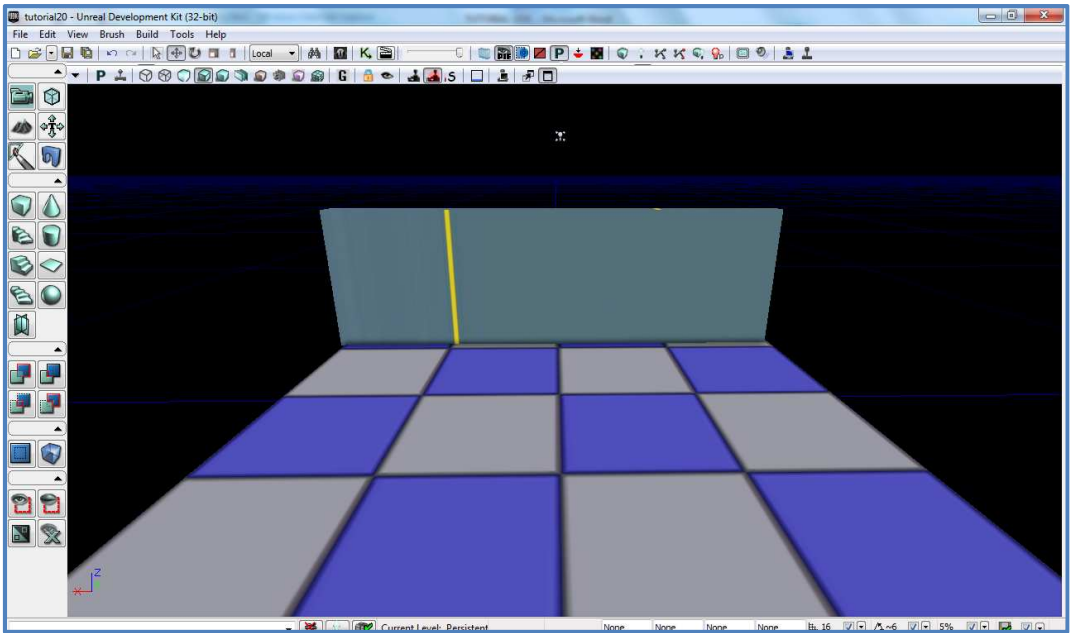


Imagen 399:Tutorial 20-Material acoplado incorrectamente.

Para mostrar las propiedades del objeto, es seleccionado y pulsar la tecla 'F4' del teclado:

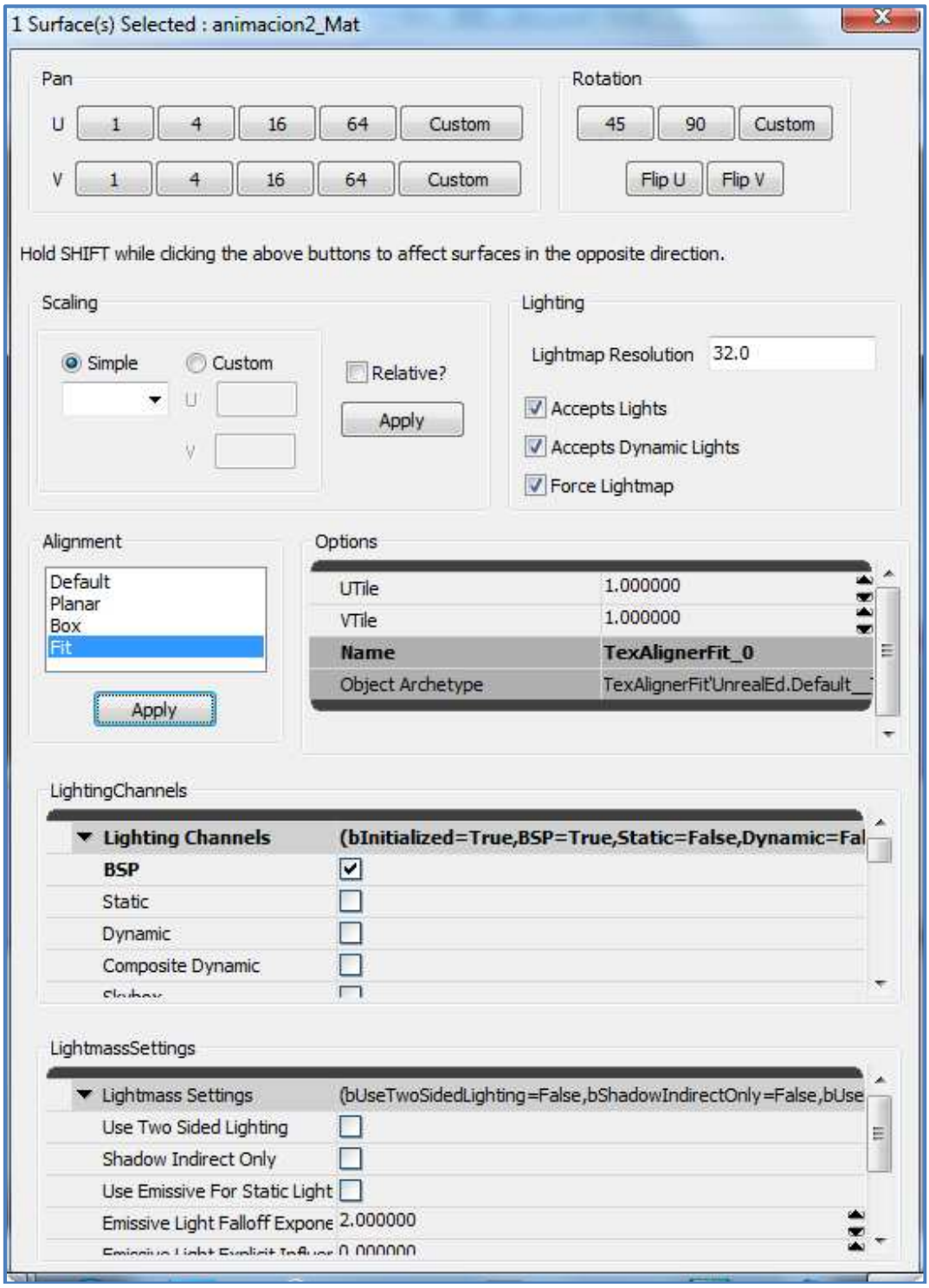


Imagen 400:Tutorial 20-Propiedades objeto.

En la sección *"Alignment"* seleccionar la opción *"Fit"* y pulsar el botón *"Apply"*.

8. Con este cambio realizado, el vídeo ya es mostrado correctamente como material, tal y como se muestra en la siguiente imagen:

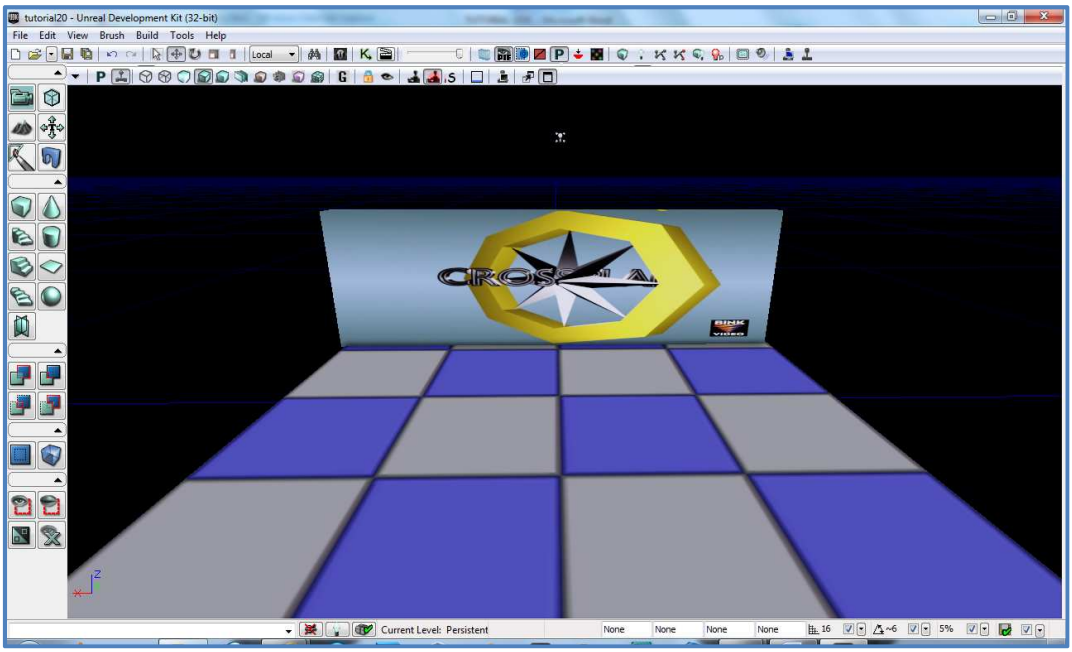


Imagen 401:Tutorial 20-Material acoplado correctamente.

8.21. Tutorial 21: Cambiar tamaño del personaje.

Según el nivel creado puede ser necesario modificar el tamaño de los personajes. Para modificar el tamaño del personaje principal se debe modificar el código de la clase del personaje. El personaje principal por defecto en el motor UDK es llamado *"Liandri"* y la clase que contiene toda la implementación está localizada en:

UDK / Development / Src / UTGame / Classes / UTFamilyInfo_Liandri.uc

En esta implementación puede modificarse la variable *"DefaultMeshScale"* que representa la escala del personaje:

```
//DefaultMeshScale=1.075
DefaultMeshScale=2.5
//BaseTranslationOffset=14
BaseTranslationOffset=80.0
```

Imagen 402:Tutorial 21-Variables respecto tamaño de personaje

Al modificar la escala del personaje principal, suele modificarse también la posición de éste respecto al suelo. En el caso que se amplíe la escala, parte del personaje puede quedar por debajo del suelo; y en el caso que se reduzca, puede quedar flotando en el aire.

Por ello hay que modificar la variable *"BaseTranslationOffset"* para situarlo en la posición correcta. Los otros tipos de personajes que se pueden acoplar al nivel, también pueden ser modificados.

Se deben modificar las mismas variables indicadas anteriormente pero en la clase *"UTPawn"*, la implementación por defecto de los posibles personajes acoplados al nivel. Esta clase puede ser localizada en:

UDK / Development / Src / UTGame / Classes / UTPawn.uc

En este tutorial se han nombrado las clases por defecto de los persoanjes en el código del motor UDK, pero pueden ser implementadas propias clases cumpliendo con el nombrado de las variables por defecto, deben ser las mismas ya que son propias del motor.

8.22. Tutorial 22: Unir dos niveles – Streaming.

Desde un nivel puede ser lanzado cualquier otro ya creado para continuar la ejecución en el nuevo nivel. Esta forma de cargar un nivel desde otro es llamada en UDK *"Streaming"*, donde la ejecución del juego pasa de un nivel a otro.

Esta forma de implementación puede ser útil para no crear niveles muy grandes que pueden dificultar la carga y la edición de ellos. Se realizarían niveles más pequeños pudiéndose hacer llamada entre ellos e ir cargándolos según necesidad.

Via "Kismet".

La herramienta Unreal Kismet proporcionada por el motor permite realizar la carga de otro nivel. Para acceder a esta herramienta se debe pulsar el icono de la barra de herramientas superior del editor. El elemento que permite preparar el nivel a cargar es llamado *"Prepare Map Change"*:

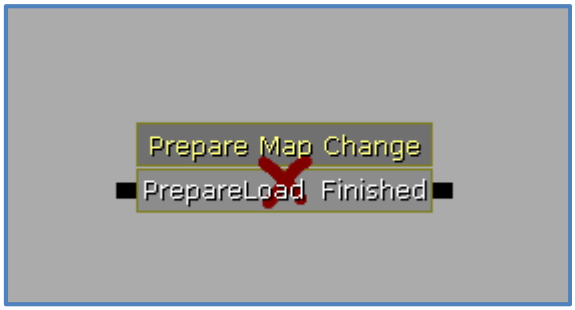


Imagen 403:Tutorial 22-Elemento "Prepare Map Change".

Se puede obtener este elemento mediante:

Botón derecho del ratón → New Action → Level → Prepare Map Change.

En las propiedades del elemento es donde se indicará el nivel a cargar, en la etiqueta *"Main Level Name"*:

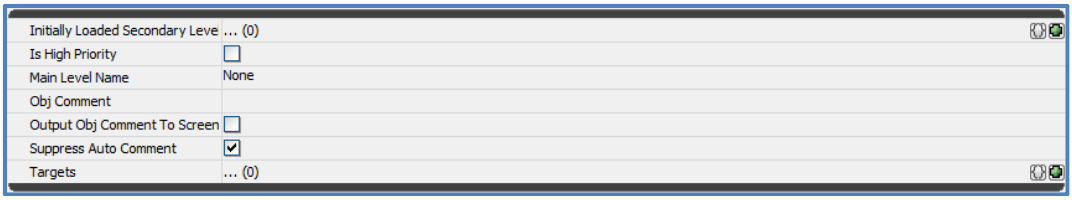


Imagen 404:Tutorial 22-Propiedades del elemento "Prepare Map Change".

Se debe introducir el mismo nombre que el archivo que contiene el nivel a cargar. Una vez preparado el nivel, éste puede ser cargado. El elemento que permite hacer el cambio de un nivel a otro es llamado *"Commit Map Change"*:

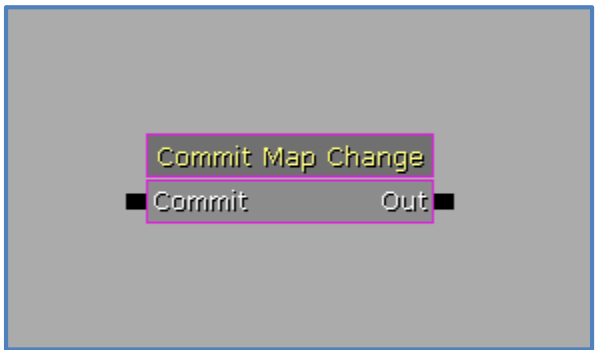


Imagen 405:Tutorial 22-Elemento "Commit Map Change".

Se puede obtener este elemento mediante:

Botón derecho del ratón → New Action → Level → Commit Map Change.

Por lo tanto, la asociación de estos dos elementos deber ser primero la preparación y luego ya la carga, no pudiéndose realizar de otra manera distinta:

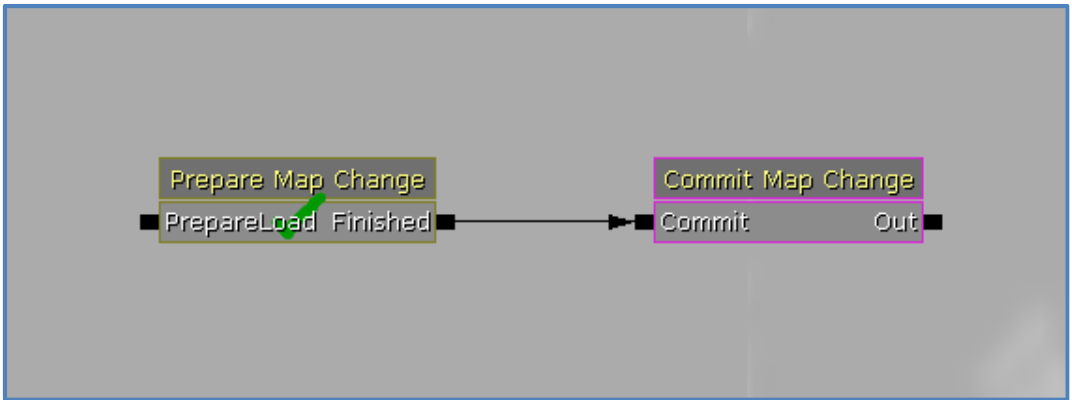


Imagen 406:Tutorial 22-"Prepare Map Change" y "Commit Map Change".

Mediante la ejecución de estos elementos, el nuevo nivel será cargado continuando la ejecución en el nuevo nivel. Esta secuencia de acciones puede ser lanzada por cualquier evento ó acción que se produzca en el nivel.

En el nivel a cargar mediante esta forma, es necesario acoplar el elemento *"Teleport"* indicando el punto de aparición del personaje mediante un objeto del tipo *"Note Actor"* en el editor. Además hay que indicar que el personaje a cargar en el nivel es el personaje principal mediante una variable *"Player"*. Para más información sobre el teletransporte del personaje consultar el tutorial 7.

Este elemento *"Teleport"* debe ser lanzado nada más cargar el nivel, utilizando el elemento *"Level Loaded"*.

Toda esta acción comentada sería implementada en Unreal Kismet de la siguiente manera:

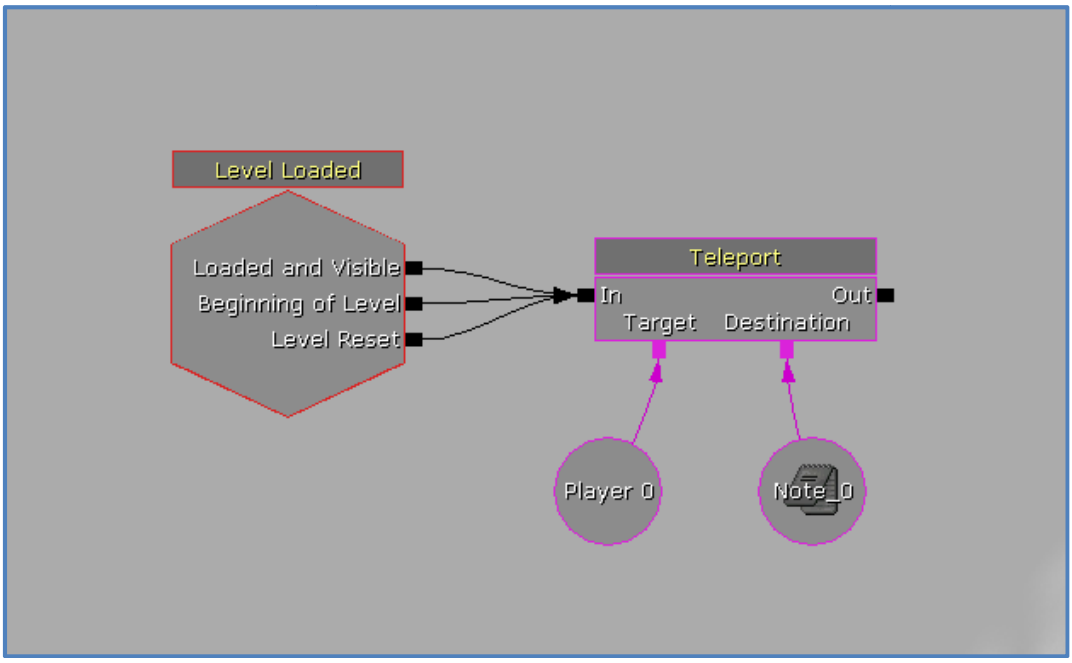


Imagen 407:Tutorial 22-Implementación en mapa de llegada.

Via Distance.

Esta forma de realizar *"Streaming"* utiliza *"volumes"* para cambiar de un nivel a otro. El jugador al hacer contacto sobre el volumen seleccionado se cargará el nuevo nivel.

Los *"volumes"* que se pueden utilizar para realizar *"Streaming"* son un tipo especial llamado *"Level Streaming Volume"*.

Para crear un "volumen" de este tipo se debe pulsar con el botón derecho del ratón sobre el icono de la parte izquierda. Se mostrará una nueva ventana con todos los tipos de volúmenes posibles:

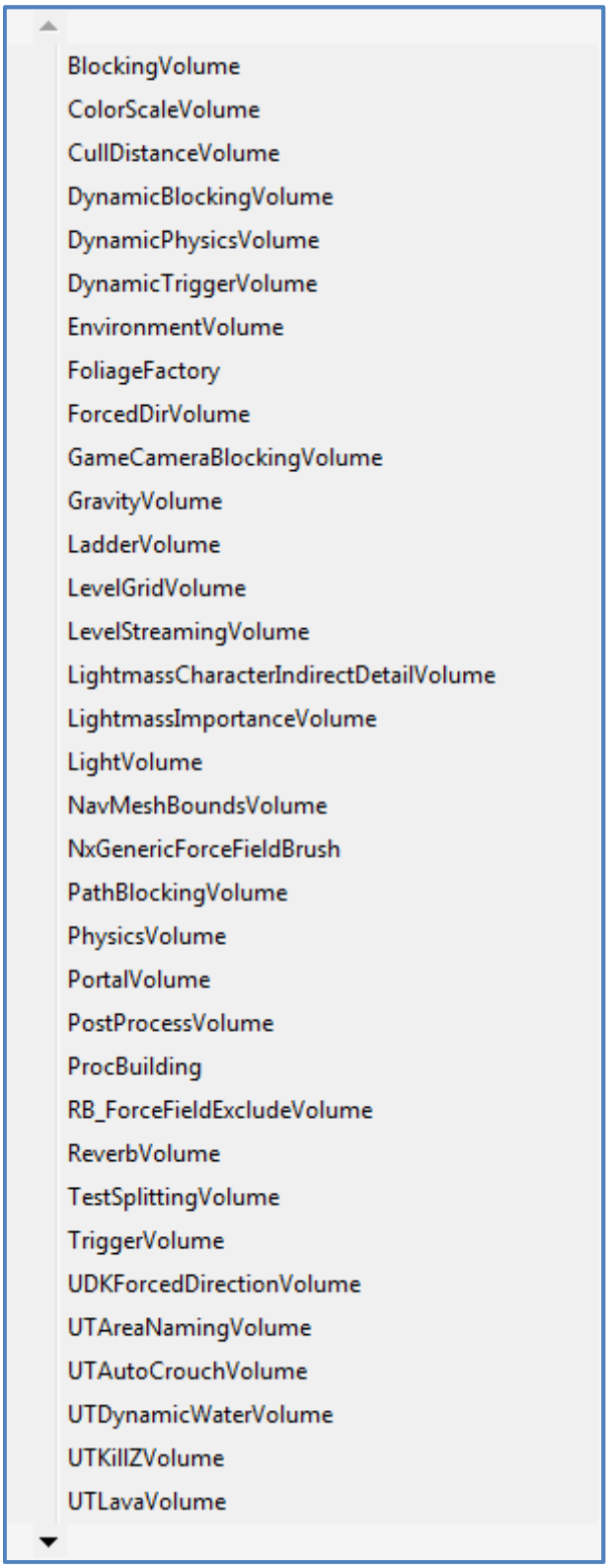


Imagen 408:Tutorial 22-Tipos de volumen.

Seleccionar *"LevelStreamingVolume"*, creándose así el volumen deseado. A este volúmen hay que asociar el nivel a cargar. Seleccionar el volumen creado y abrir la herramienta *"Levels"*. Esta herramienta puede abrirse mediante el icono de la barra de herramientas superior del editor y seleccionando la etiqueta con el mismo nombre:

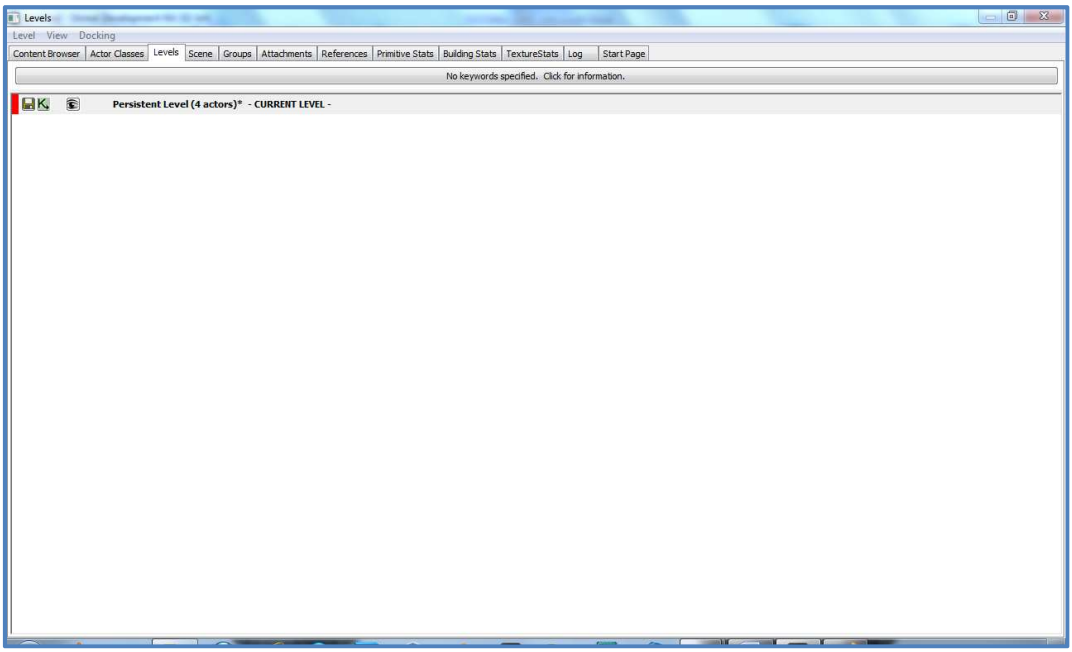


Imagen 409:Tutorial 22-Interfaz "Levels".

Se realiza una doble pulsación sobre el nivel actual para mostrar los niveles posibles a asociar. Sobre el nivel a asociar se pulsa con el botón derecho del ratón y seleccionar *"Set Streaming Volumes"* para quedar asociado el volumen con el nivel.

NOTA: El *"Streaming"* sólo se ejecuta cuando el juego es lanzado en modo *"Start level on PC"*, cuando los niveles se encuentran en toda la estructura de juego completo proporcionada por UDK.

8.23. Tutorial 23: Finalizar juego.

Una de las circunstancias importantes en la creación de un juego es cómo finalizarlo. La forma que se tiene en UDK de finalizar un juego es mostrando la pantalla de finalización por defecto en el motor y volver al menú también por defecto en toda la estructura de juego completo proporcionada por el motor.

Desde el nivel creado se puede hacer llamada a esta forma de finalizar un juego. Es realizada mediante la herramienta Unreal Kismet del editor pudiéndose abrir mediante el icono de la barra de herramientas superior del editor.

El elemento a utilizar es llamado *"Console Command"*:

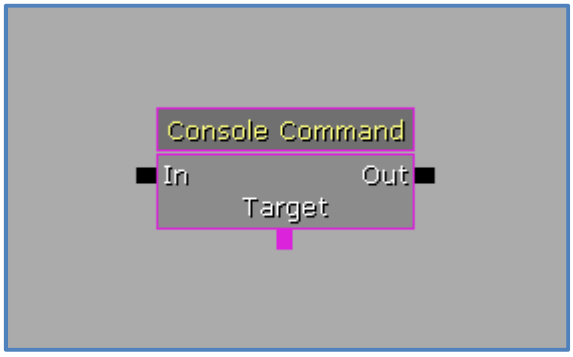


Imagen 410:Tutorial 23-Elemento "Console Command".

Este elemento puede ser insertado mediante:

Botón derecho del ratón → New Action → Misc → Console Command

Para realizar la finalización del juego se deben modificar las propiedades de este elemento mostradas en la parte inferior izquierda al seleccionar el elemento:

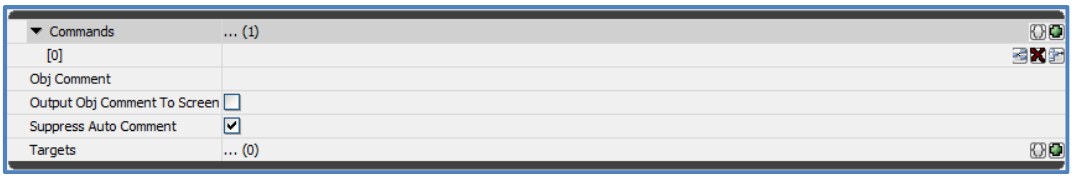


Imagen 411:Tutorial 23-Propiedades elemento "Console Command".

La etiqueta a modificar es llamada *"Commands"*. En la posición [0] hay que insertar el comando *"open UTFrontend"*:

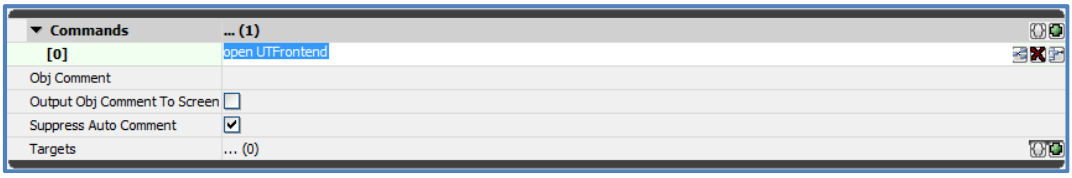


Imagen 412:Tutorial 23-Modificación de la etiqueta "Commands".

También en la entrada *"Target"* hay que asociar una variable que represente al personaje principal del nivel. Para obtener esta variable:

Botón derecho del ratón → New Variable → Player → Player

Y en las propiedades de esta variable hay que desactivar la etiqueta *"All Players"* para que represente sólo al personaje principal. La variable se asociará a la entrada *"Target"* del elemento *"Console Command"*, tal y como se muestra en la siguiente imagen:

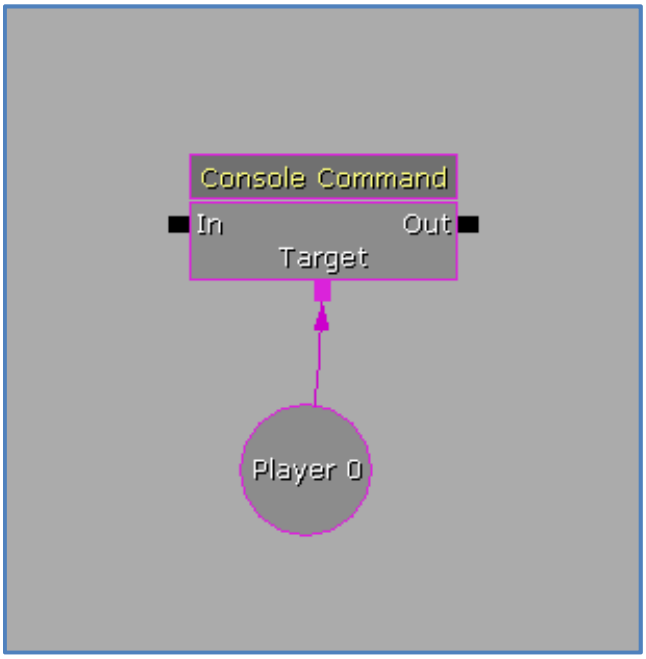


Imagen 413:Tutorial 23-Asociación jugador con "Console Command".

La finalización del juego puede ser lanzada por cualquier evento ó acción que se produzca en el nivel.

En el tutorial se ejecuta el fin del juego nada más cargarse el nivel utilizando el elemento *"Level Loaded"*:

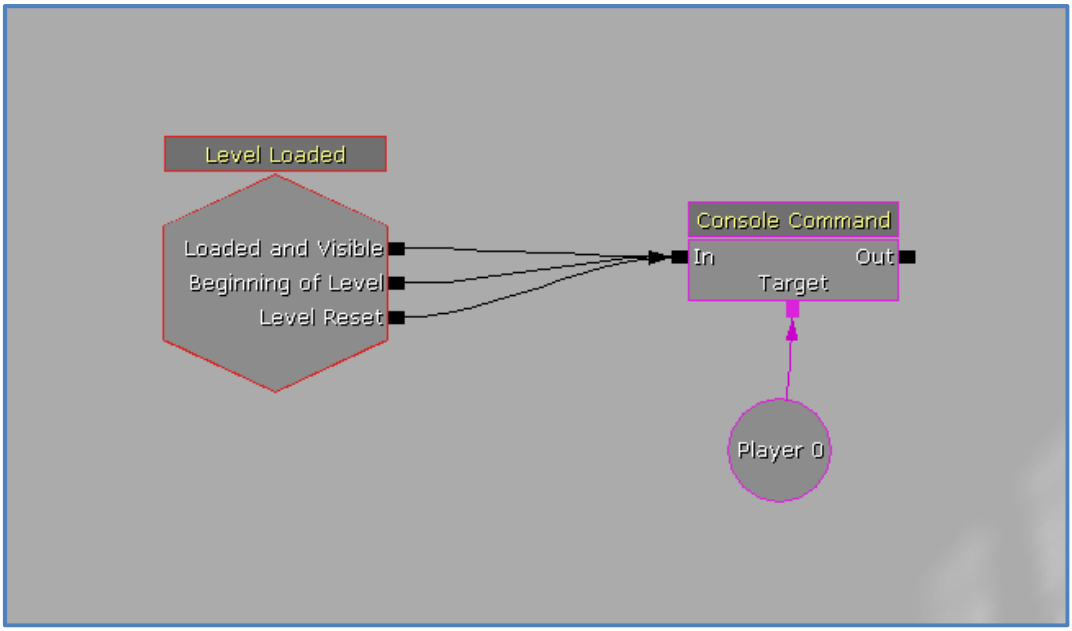


Imagen 414:Tutorial 23-Implementación de ejecución comando.

Antes de realizar la finalización del juego se puede mostrar cualquier información en pantalla mediante elementos *"UIScene"* ó realizar un fundimiento de la pantalla mediante *"Face Track"* del grupo *"DirGroup"* de la herramienta Matinee.

8.24. Tutorial 24: Cambiar/modificar la pantalla de "Bienvenida"

Mientras la carga de un juego realizado con el motor UDK, es mostrada una pantalla llamada *"pantalla de Bienvenida"*. Por defecto la pantalla de bienvenida es la siguiente:

Imagen 415:Tutorial 24-Pantalla de bienvenida.

Esta pantalla puede ser modificada para mostrar la deseada, para hacer más personal el juego creado.

Hay que tener en cuenta que *"por términos legales de Unreal, el logotipo de UDK debe aparecer en alguna parte de esta pantalla de inicio"*.

Esta pantalla de inicio del motor puede encontrarse en la siguiente jerarquía de directorios:

UDK / UDKGame / Splash / PC / Splash.bmp

Esta pantalla de inicio es una imagen con extensión *.bmp* con nombre *"Splash"* pudiéndola cambiar con cualquier editor de imágenes. Para modificar por completo la imagen, la imagen nueva debe ser nombrada de la misma manera y colocarla en el mismo directorio para poder ser lanzada por el motor.

8.25. Tutorial 25: Crear instalador del juego completo (.EXE).

Cuando se ha creado un nuevo juego con todos sus niveles implementados es necesario obtener un ejecutable con todo lo necesario para poder instalar el juego en cualquier equipo.

El motor UDK ofrece la posibilidad de obtener un ejecutable del juego con los niveles deseados. Para ello se proporciona la aplicación *"Unreal Frontend"* junto con el motor. La interfaz de esta herramienta es mostrada en la siguiente imagen:

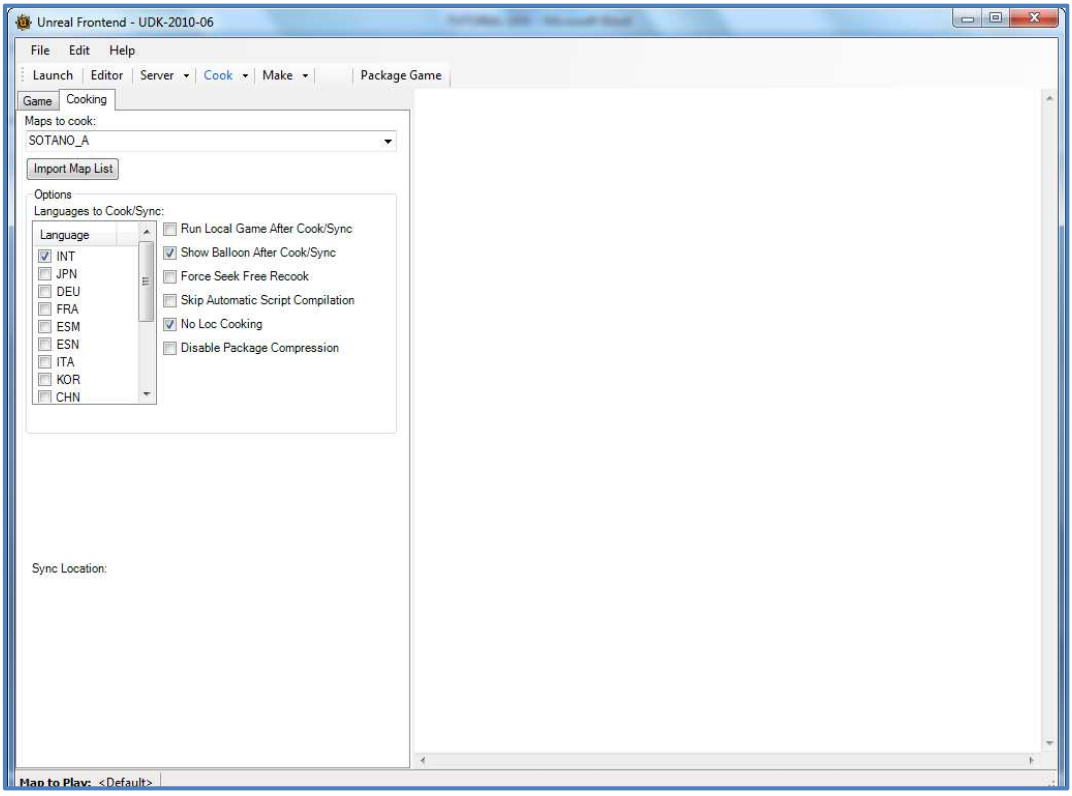


Imagen 416:Tutorial 25-Interfaz "Unreal Frontend".

En la pestaña *"Game"* hay que deseleccionar las etiquetas:

- Use cooked map.

- No VSync.
- Show Log.
- Remote Control.

Se debe dejar seleccionada sólo la etiqueta:

- Multi-threaded.

En la pestaña *"Cooking"*, en la etiqueta *"Maps to Cook"* se indican los nombres de los niveles que compondrán el juego completo. Para introducir varios niveles en dicha etiqueta, los nombres deben estar separados por el carácter espacio.

Pulsar en el triángulo invertido de la opción *"Cook"* de la aplicación

. En el menú mostrado seleccionar la opción *"Full Cook"* y en el mensaje mostrado elegir *"Sí"*.

La aplicación comenzará a trabajar en la compilación de los niveles indicados. Al finalizar todo el trabajo se muestra un resumen de todos los errores y avisos encontrados en la compilación:

```
Warning/Error Summary
---------------------
Warning, Found   67 script referenced static meshes!

Success - 0 error(s), 1 warning(s)
Execution of commandlet took:  108.98 seconds

[COMMANDLET 'UDK.exe CookPackages -platform=PC SOTANO_A mundo -full -languageforcookin
```

Imagen 417:Tutorial 25-Finalización de la compilación.

Al realizar la compilación, la opción *"Package Game"* queda activa, con la cual se creará el ejecutable. Pulsando en ella se muestra una ventana donde se podrá indicar el nombre del juego:

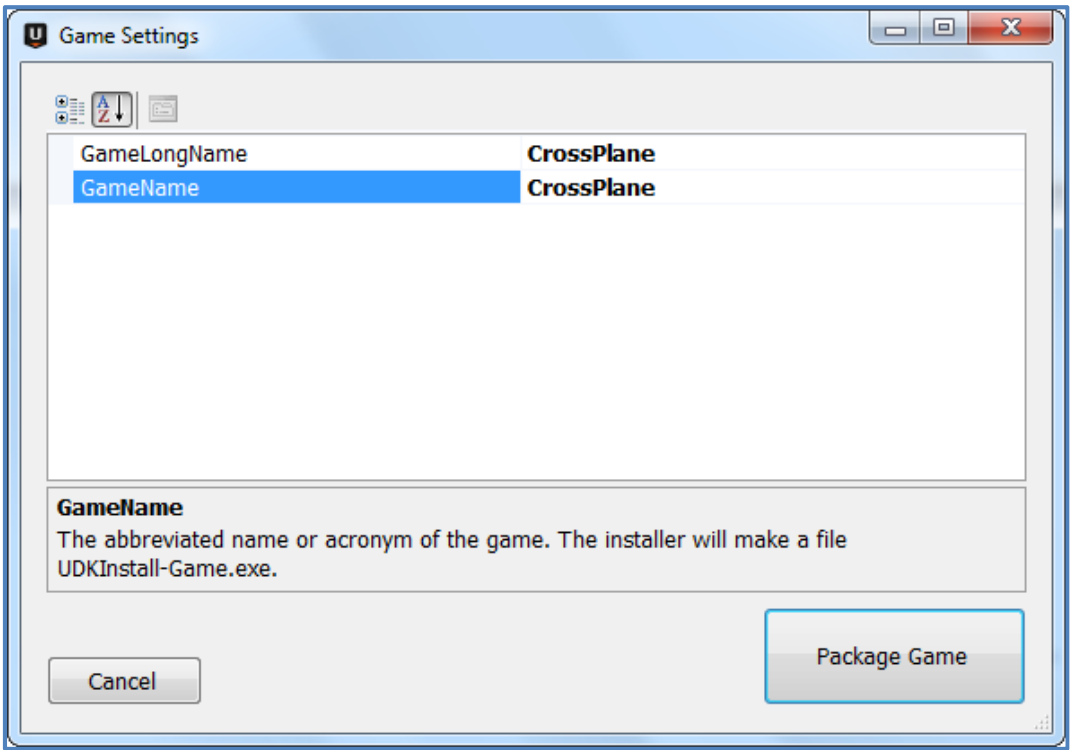


Imagen 418:Tutorial 25-Entrada del nombre del juego.

Cuando ya se ha indicado el nombre del juego, y por lo tanto el nombre del ejecutable, se debe pulsar sobre el botón *"Package Game"*. Con ello se empezará a crear toda la estructura necesaria para la obtención del ejecutable:

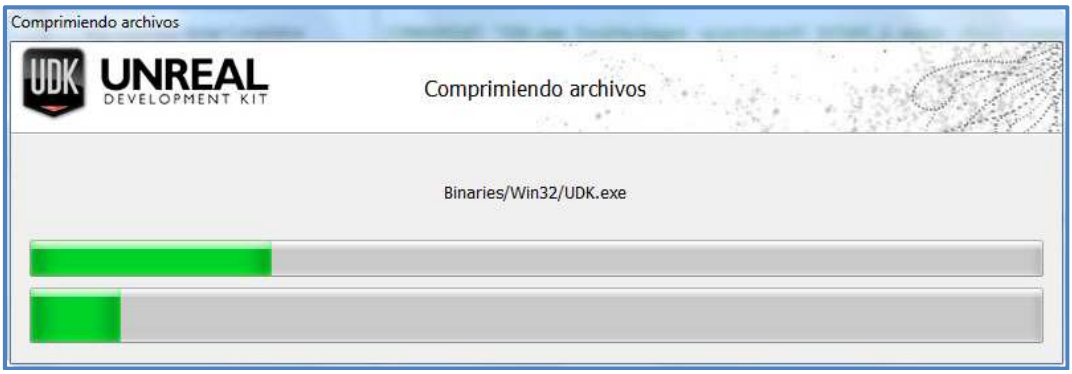


Imagen 419:Tutorial 25-Comprensión de archivos.

Al finalizar, se dispondrá del ejecutable *".exe"* necesario para instalar el juego en cualquier equipo. Este ejecutable se puede localizar en el directorio *"/UDK/"* con el nombre indicado previamente.

9. CREAR NUEVO PERSONAJE.

Se va a mostrar la forma de poder crear un nuevo personaje en UDK, desde su creación, importación y elementos necesarios en el motor para trabajar con él.

Todo personaje tiene la siguiente estructura:

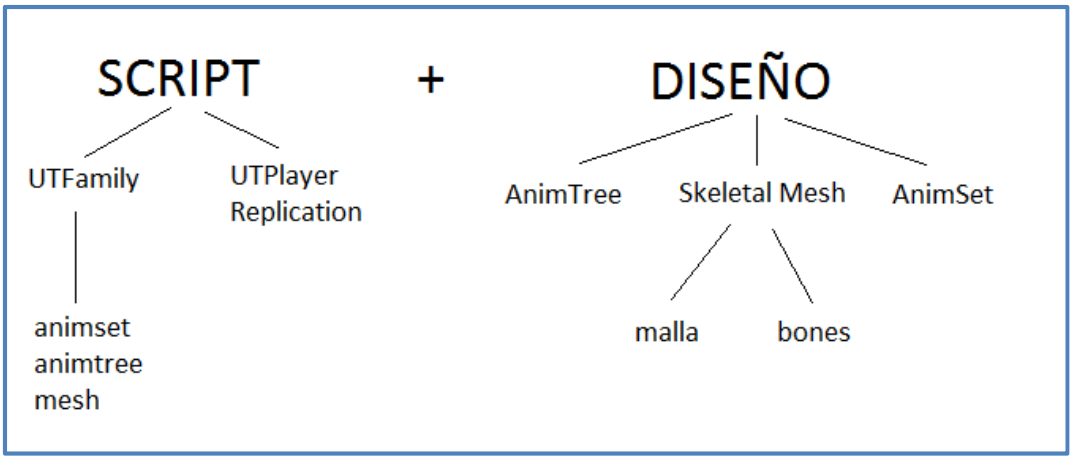


Imagen 420:Estructura creación de personajes.

9.1. Creación del "Skeletal Mesh".

Los *"Skeletal Mesh"* son mallas a la cuales se les puede dotar de movimiento ya que poseen un esqueleto internamente que permite realizar estas animaciones. [52] [53]

Para crear estos tipos de mallas se puede utilizar cualquier aplicación de modelado en 3D. En este manual se va a utilizar la herramienta *"3D Studio Max"* [54], que aunque es una aplicación de pago, es una de las más completas. [55] [56]

En la siguiente imagen se presenta la interfaz presentada por esta aplicación:

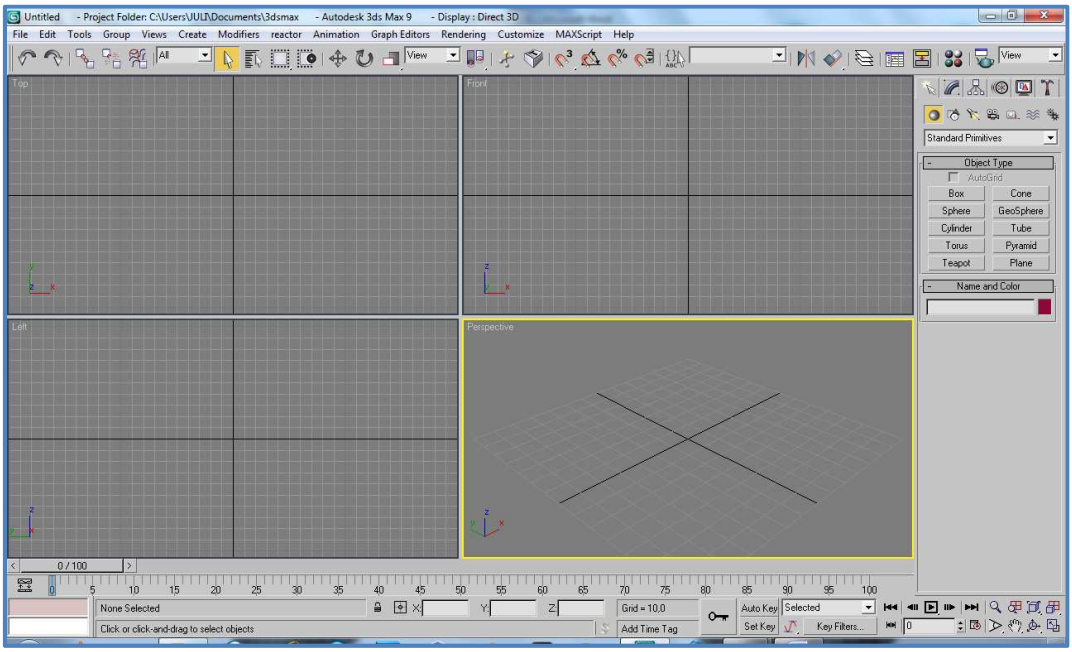


Imagen 421:Interfaz aplicación "3D Studio Max".

Lo recomendable a la hora de crear un *"Skeletal Mesh"* es primero obtener el esqueleto a partir del cual se va a ir realizando la malla y asociándola con los "huesos" de éste.

El personaje que se quiere crear es un bípedo, y *"3D Studio Max"* proporciona un esqueleto de este tipo. Se puede encontrar este tipo de esqueleto en las herramientas de la derecha: *Create → Systems → Biped*.

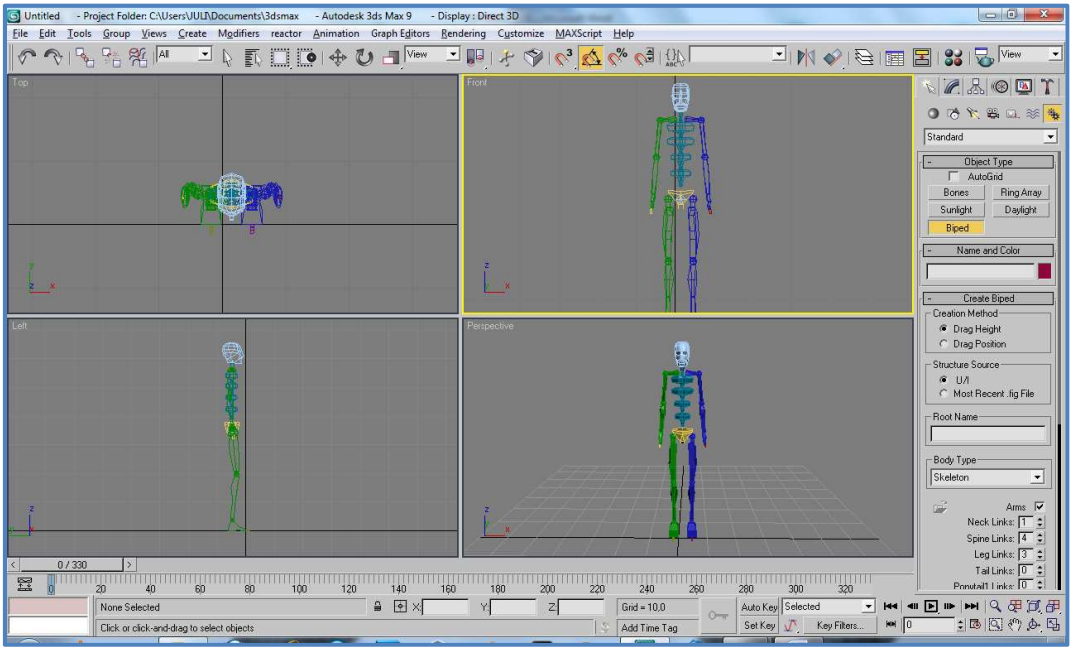


Imagen 422:Esqueleto.

Mediante la inserción de distintos elementos y su modificación, se puede ir creando la malla que recubrirá al esqueleto. En la imagen 423 se muestra el esqueleto y la malla a utilizar.

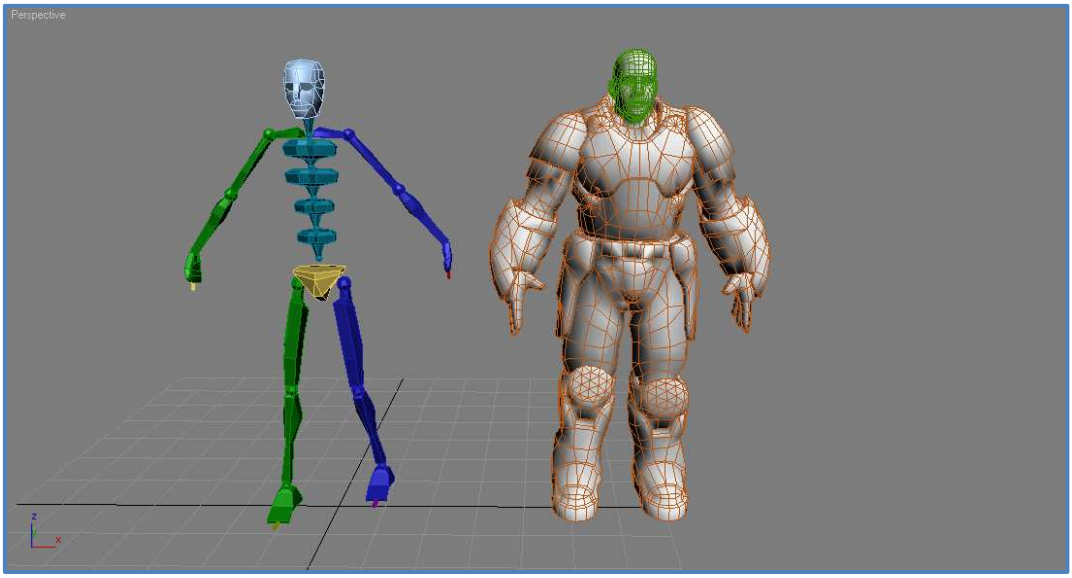


Imagen 423:Malla y esqueleto.

NOTA: No forma parte de este manual el describir la forma de crear distintas formas en el editor *"3D Studio Max"*. Para ello se tedría que consultar un manual específico de la herramienta.

Una vez creados el esqueleto y la *"piel"* hay que unirlos para que al realizar un movimiento en los huesos, la *"piel"* también realice el mismo movimiento:

El elemento de la *"piel"* se debe transformar a tipo *"Editable Poly"*.
Para ello se pulsa el botón derecho del ratón sobre el objeto y se selecciona *"convert to:"* en el menú mostrado. Se desplegará un nuevo menú en el que seleccionar *"Editable Poly"*.

En la aplicación se mostrarán las propiedades propias de estos tipos de elementos, tal y como se muestra en la siguiente imagen:

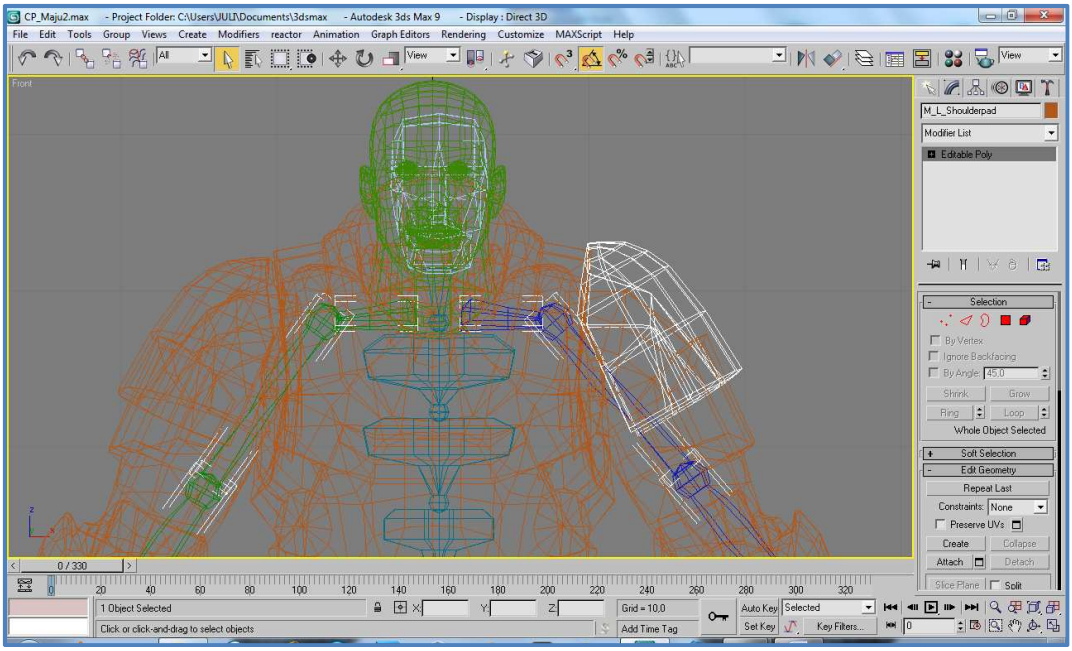


Imagen 424:Acople de malla y esqueleto.

En la parte de la derecha se muestra el conjunto de herramientas propias del tipo de objeto seleccionado.

En el desplegable de la parte superior, con nombre *"Modifier List"* hay que seleccionar *"Skin"*. Mediante esto, se podrá asociar al objeto de la piel el hueso que le corresponde. Para ello, en la sección *"Bones"* del menú de la derecha, se pulsa el botón *"Add"* mostrándose una nueva ventana para seleccionar el hueso que se quiere asocial al objeto de la piel:

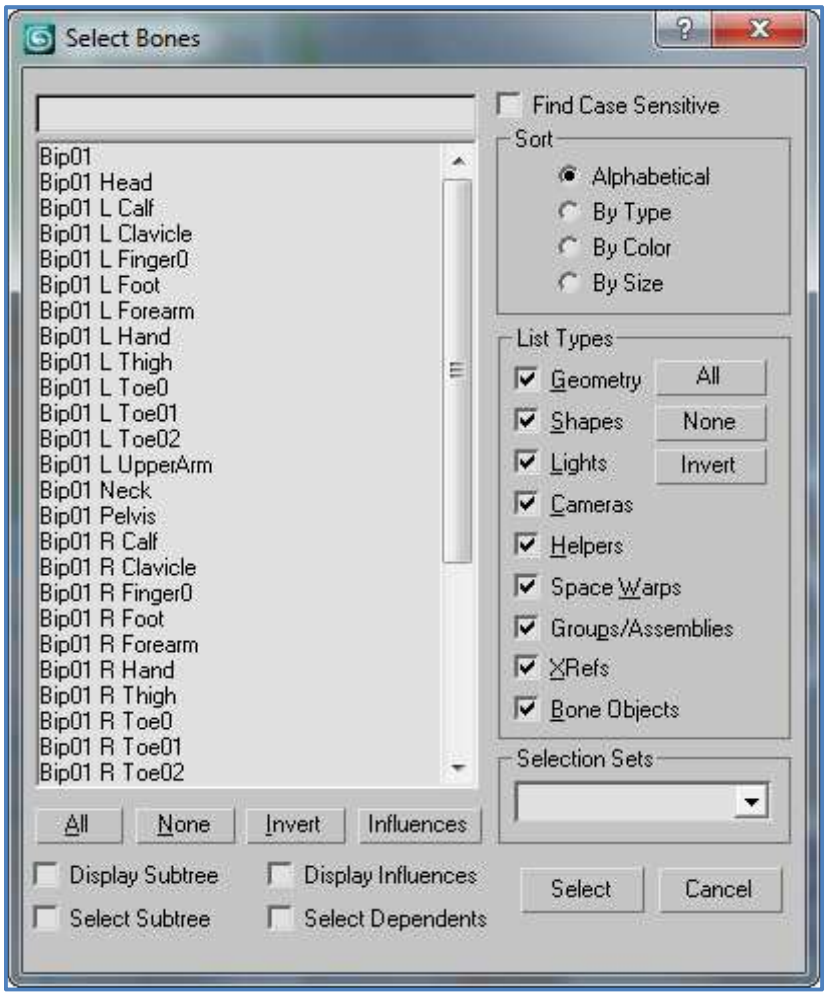


Imagen 425:Menú de asociación de objeto de malla con hueso del esqueleto.

Los huesos serán mostrados por su nombre en el esqueleto, con lo que para poder asociarlos con la piel habrá que saber el nombre de cada uno de ellos. Para obtener el nombre del hueso, simplemente con seleccionarlo se mostrará en las herramientas de la derecha.

El hueso ó los huesos seleccionados se indicarán en el panel de texto de la parte inferior, representada en la imagen 426.

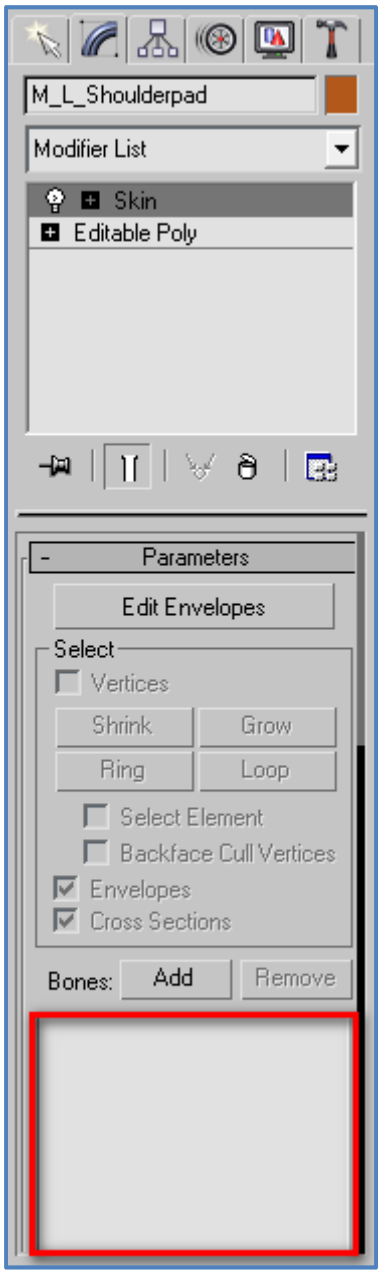


Imagen 426:Sección de nombres de huesos.

Al realizar un movimiento en el hueso, la piel también se moverá, comprobando así la correcta unión. Esta secuencia de acciones habría que realizarla con todos los objetos que forman parte de la piel y asociarlos con sus respectivos huesos.

Cuando ya se tiene perfectamente unidos el esqueleto y la piel, se pueden guardar ciertos movimientos del conjunto completo para reproducirlos posteriormente.

Los movimientos se guardarán en *"frames"*, y la secuencia de *"frames"* total, se encuentra en la parte inferior de la aplicación, que será utilizada para crear los distintos movimientos.

Para almacenar animaciones de un *"Skeletal Mesh"* se deben realizar los siguientes pasos:

- Seleccionar el hueso que se quiere animar.

- Para entrar en el modo de edición de animaciones hay que presionar el botón llamado *"Auto Key"* de la parte inferior, imagen 427.

  Se comprobará que la parte superior de la secuencia de frames cambia a color rojo, indicando la edición.

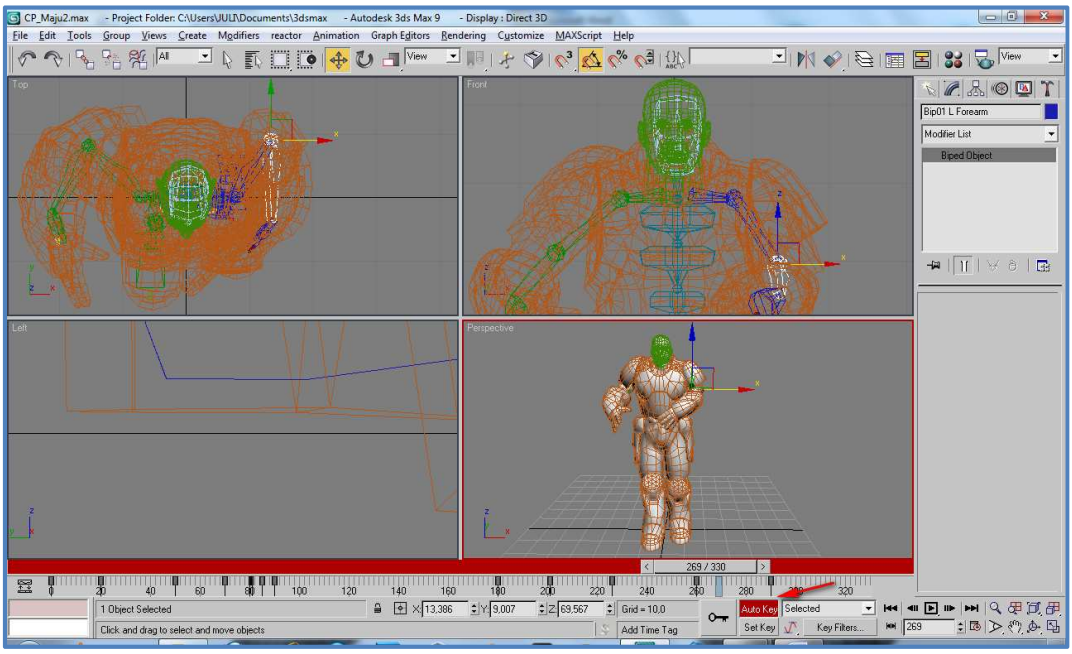


Imagen 427:Edición de secuencia de frames.

- Se avanza la barra al frame que será el frame final de la animación.

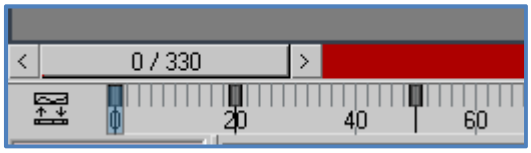


Imagen 428:Barra de frames.

- Presionar la tecla *"llave"* para guardar un punto de control en el frame indicado y para almacenar la animación a realizar.

Imagen 429:Botón para crear un punto de control de frames.

- Realizar el movimiento del hueso a grabar.
- Cuando ya se hayan guardado todas las animaciones, se puede desactivar la edición pulsando el botón *"Auto Key"* de nuevo.

Desplazando la barra de los frames de la parte inferior se pueden observar todas las animaciones guardadas del *"Skeletal Mesh"*.

Ya se tendría creado un *"Skeletal Mesh"* completo con sus animaciones guardadas que podrá ser exportado al motor UDK.

9.2. Exportar "Skeletal Mesh" de 3D Studio Max a UDK.

Los *"Skeletal Mesh"* y sus animaciones pueden ser exportadas en las extensiones adecuadas para ser utilizadas en UDK. La malla debe ser exportada a un archivo con extensión *".PSK"* y las animaciones a otro archivo independiente con extensión *".PSA"*.

Para realizar estas exportaciones se debe utilizar un plugin desarrollado por Epic llamado *"ActorX"* [57]. Se puede obtener este plugin para las principales aplicaciones de modelado, como 3DS Max, Maya y XSI.

Imagen 430:ActorX

Este plugin puede ser encontrado en la página:

http://udn.epicgames.com/Two/ActorX.html

pudiendo seleccionar la versión adecuada según la aplicación que se quiere utilizar.

Una vez descargado el archivo, se debe descomprimir en el directorio llamado *"plugins"* dentro de los directorios de 3DS Max.

El plugin instalado puede encontrarse dentro de la pestaña *"Utilities"* de las herramientas de la derecha. Presionando en el botón *"More…"* se mostrará una nueva ventana con todas las utilidades instaladas, siendo una de ellas *"ActorX"*:

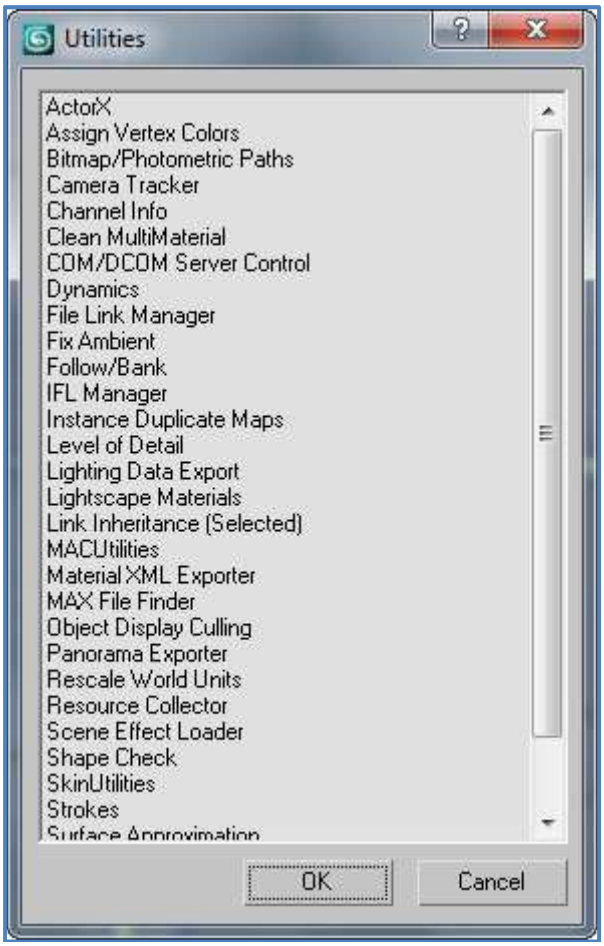


Imagen 431:Utilidades en la aplicación "3D Studio Max".

Al seleccionar la utilidad de *"ActorX"*, en la sección de la derecha, se mostrará dicha utilidad y las herramientas que proporciona:

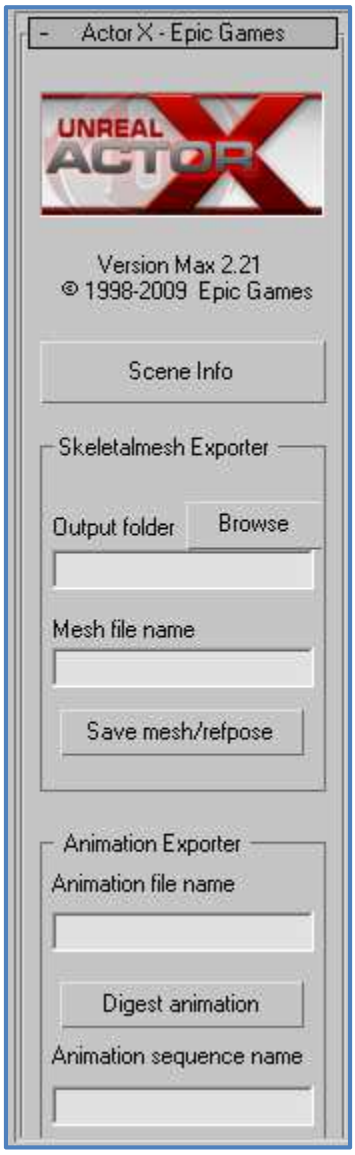


Imagen 432:Utilidad "ActorX" en aplicación "3D Studio Max".

Para obtener el archivo *.PSK* de la malla hay que rellenar los campos que se encuentran dentro de la sección *"Skeletalmesh Exporter"*:

- Output Folder.- Directorio donde se guardará el archivo.
- Mesh File name.- Nombre que se quiere dar al archivo.

Una vez rellenos estos campos hay que presionar el botón *"Save mesh/refpose"* creándose el achivo *.PSK* en el directorio indicado.

Cada una de las animaciones guardadas del mesh pueden ser exportadas con distinto nombre al mismo archivo *.PSA*. Para obtener este archivo se deben rellenar los campos que se encuentran dentro de la sección *"Animation Exporter"*:

- Animation file name.- Nombre que se le quiere dar al archivo de salida.

Cada una de las animaciones a exportar deben ser nombradas e indicar el rango de frames que forma dicha animación:

- Animation sequence name.- El nombre a dar a la animación.

- Animation range.- Rango de frames que componen la animación. El formato de esta entrada es *num_inicio – num_fin* (Ej. 0 – 10).
- Pulsar el botón *"Digest Animation"* para que la animación sea almacenada con el nombre dado.

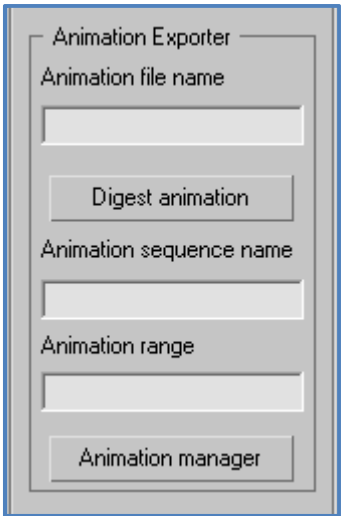


Imagen 433:"Animation Explorer".

Se realizarán estos pasos con todas las animaciones a exportar. Cuando se han guardado todas las animaciones con su respectivo nombre hay que pulsar en el botón *"Animation manager"*, ver imagen 433.

Se mostrará una nueva ventana (imagen 434), donde en la parte izquierda, en la sección *"Animations"*, se mostrarán todas las animaciones almacenadas con el nombre dado.

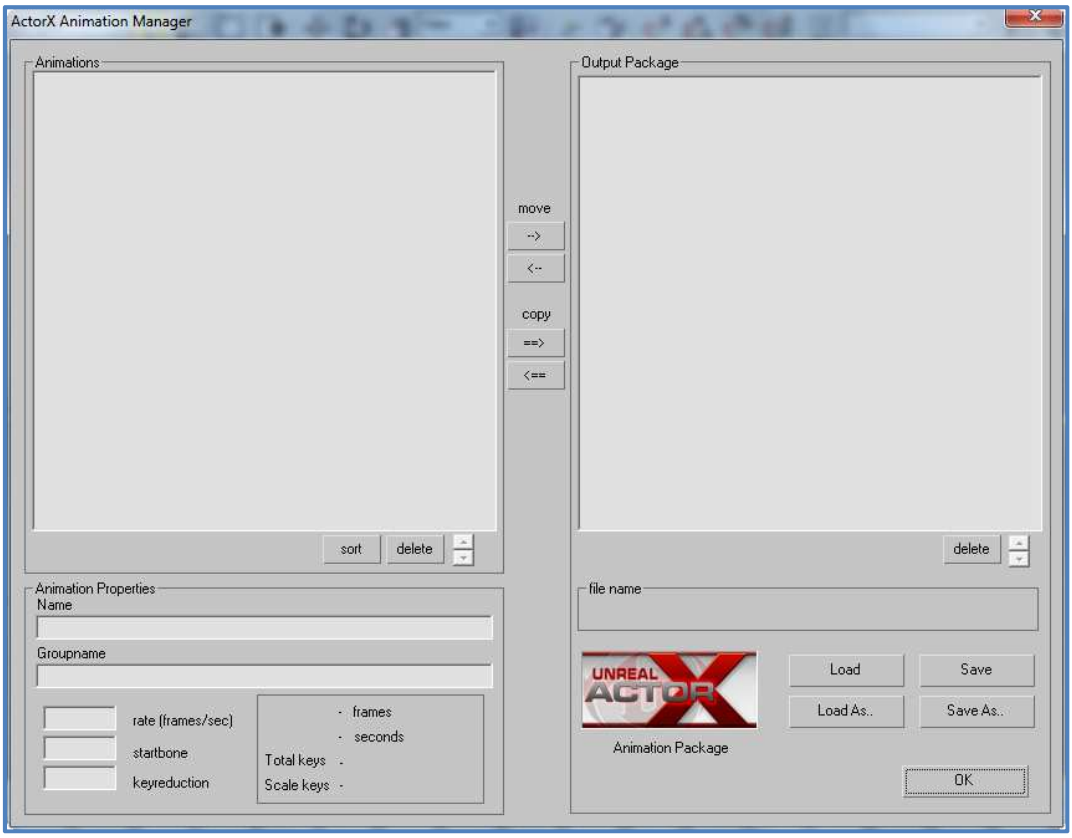


Imagen 434: "Animation Manager" de la utilidad ActorX.

En la sección de la derecha, *"Output Package"*, se deben colocar las animaciones a tener en el archivo *.PSA*. Para colocar las animaciones en esta sección se deben seleccionar de *"Animations"* y pulsar en los botones de *"move"* ó *"copy"*.

Cuando se tienen las animaciones en la sección *"Output Package"* se debe presionar el botón *"Save"* ó *"Save As…"* creando el archivo con las animaciones.

9.3. Importar "Skeletal Mesh" a UDK.

Los *"Skeletal Mesh"* se pueden importar a UDK desde un archivo con extensión *.PSK*. En la sección anterior se ha mostrado la forma de obtener este archivo desde cualquier aplicación de modelado.

La importación de este mesh se puede realizar mediante la herramienta Content Browser proprocionada por UDK. Para abrir dicha herramienta hay que pulsar sobre el icono de la barra de herramientas superior del editor.

Una vez finalizada la importación del archivo, se dispondrá en Content Browser de un elemento de tipo *"SkeletalMesh"* con el mesh importado:

Imagen 435:Elemento "SkeletalMesh" en "Content Browser".

Este tipo de elemento tiene su propio editor llamado *"Unreal AnimSet Editor"*. Se puede abrir este editor haciendo doble click sobre él:

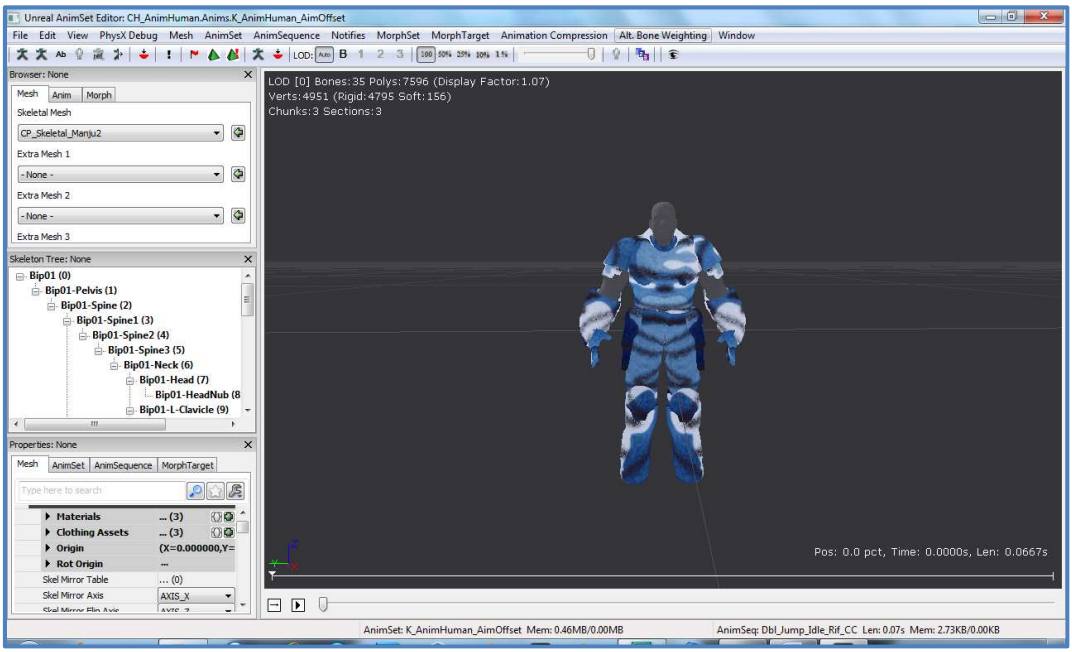


Imagen 436:Interfaz "Unreal AnimSet Editor".

Mediante este editor se pueden añadir/modificar los materiales del mesh, la posición, etc. Las modificaciones a realizar al mesh se deben hacer en las correspondientes etiquetas mostradas en la pestaña *"mesh"* de la sección *"properties"* mostradas en la parte inferior del editor, tal y como se muestra en la imagen 437.

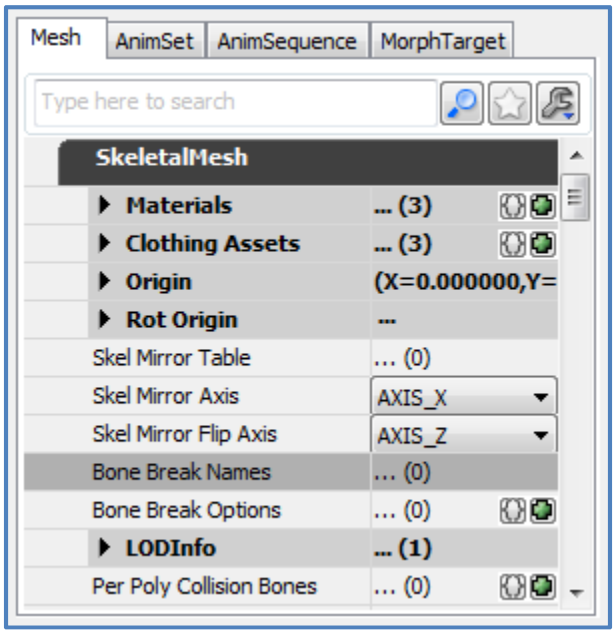


Imagen 437:Pestaña "Mesh" en el "Unreal AnimSet Editor".

9.4. Creación de elementos básicos a partir del "Skeletal Mesh".

Una vez creado el elemento *"SkeletalMesh"* con el archivo importado, es necesario crear otros tipos de elementos a partir de éste que ampliarán la funcionalidad del mesh:

- Physics Asset.
- AnimSet.
- AnimTree.

Physics Asset.

Este tipo de elemento contiene toda la física de objetos que componen al *"Skeletal Mesh"*. Para crear este tipo de elemento se debe seleccionar el elemento del *"SkeletalMesh"*. Al pulsar con el botón derecho del ratón se mostrará un menú que permite la creación del elemento, seleccionando la opción *"Create New Physics Asset"*.

El elemento de este tipo se creará mostrándose en el mismo paquete que el *"Skeletal Mesh"*:

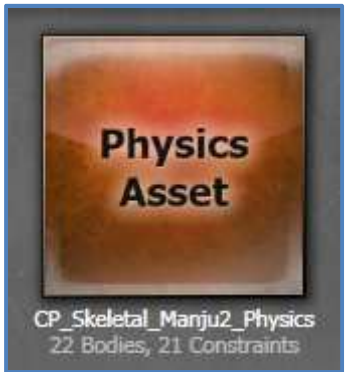


Imagen 438:Elemento "Physics Asset" en "Content Browser".

Al hacer doble click sobre este nuevo elemento creado se abrirá el editor propio donde se podrán modificar las propiedades de éste:

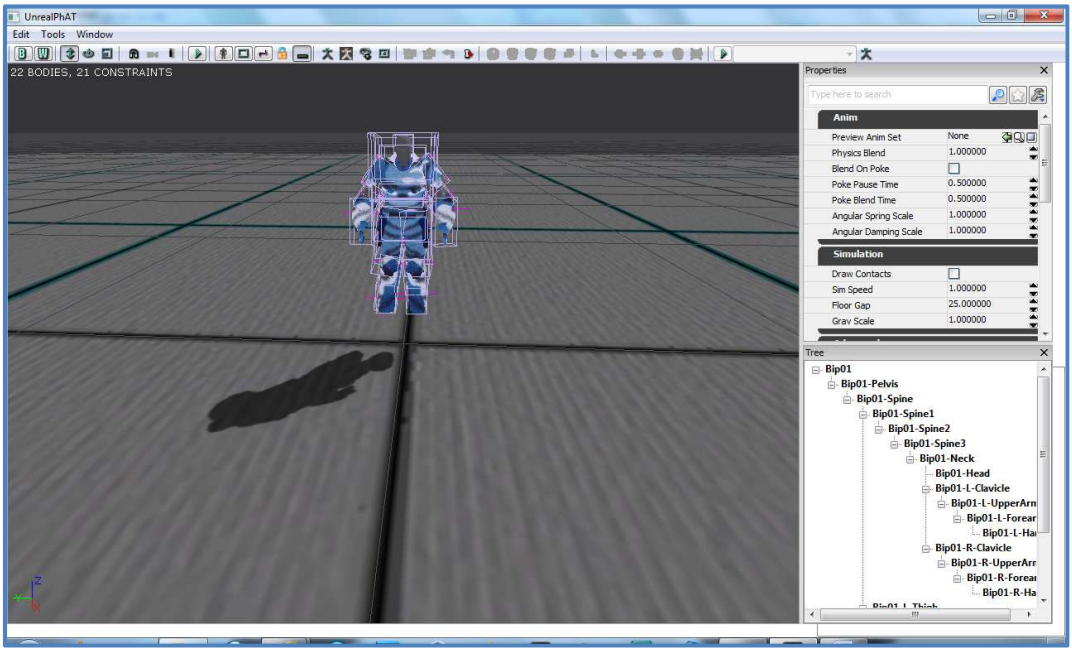


Imagen 439:Interfaz del editor de los elementos "Physics Asset".

Se puede observar todo el árbol de huesos que componen el esqueleto, en la parte derecha inferior, y las propiedades de los objetos en la parte superior derecha.

Anim Set.

Las animaciones creadas de un *"Skeletal Mesh"* en una aplicación de modelado son exportadas a un archivo con extensión *.PSA*.

Mediante este elemento se podrá importar el archivo que contiene las animaciones y asociarlas con el esqueleto. Para crear este tipo de elemento se debe seleccionar el elemento del *"SkeletalMesh"*. Al pulsar con el botón derecho del ratón se mostrará un menú que permite la creación del elemento, seleccionando la opción *"Create New AnimSet from Mesh"*.

El elemento de este tipo se creará mostrándose en el mismo paquete que el *"Skeletal Mesh"*:

Imagen 440:Elemento "AnimSet" en el "Content Browser".

En la creación del elemento se abrirá el editor de este tipo de elementos donde se podrá importar el archivo *.PSA*.

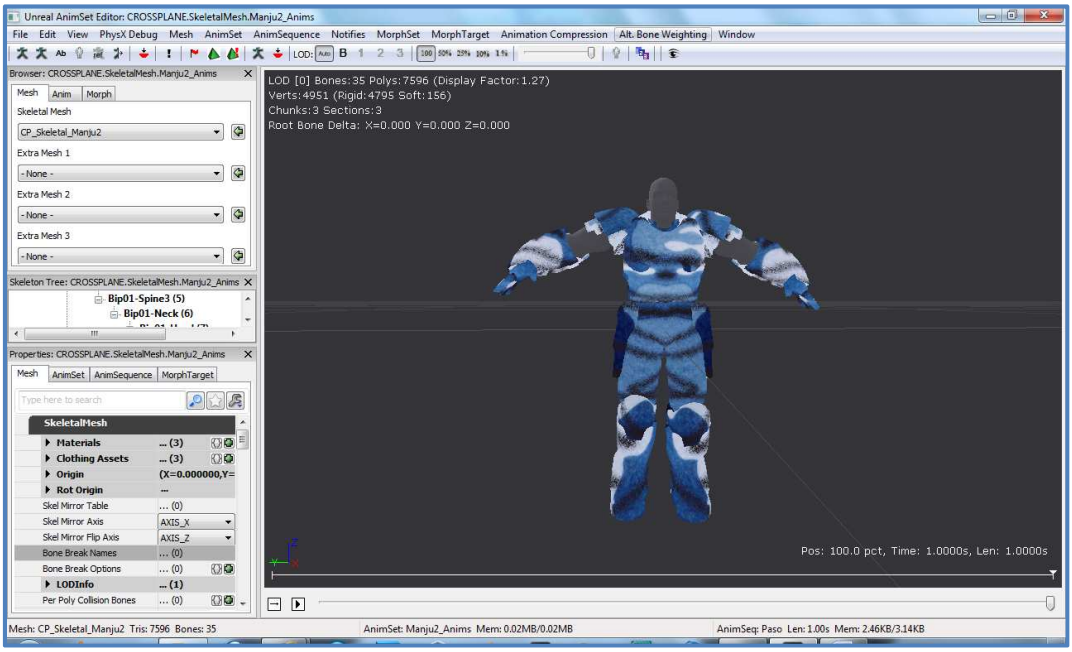


Imagen 441:Interfaz del editor de los elementos "AnimSet".

En la sección *"mesh"* del menú mostrado se puede importar el archivo *.PSA* correspondiente a las animaciones.

Una vez importado el archivo, podrán verse todas las animaciones que contiene el archivo y reproducirlas. En la etiqueta *"Anim"* de la parte izquierda se pueden observar todos los archivos de animaciones que se tienen.

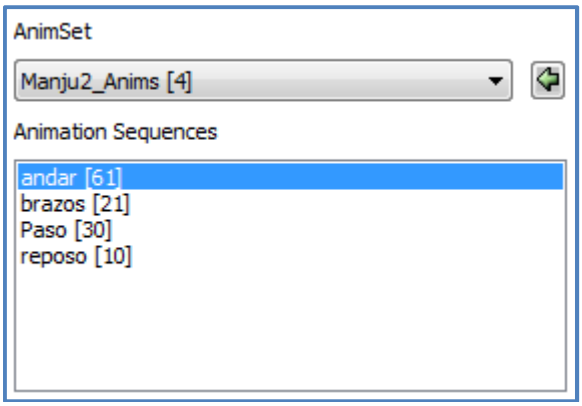


Imagen 442:Animaciones pertenecientes a un "AnimSet".

En el desplegable mostrado en *"AnimSet"* se tendrán todos los elementos de animaciones (Anim Set), y seleccionando cada uno de ellos se mostrarán las animaciones que poseen con su nombre en la sección *"Animation Sequences"*, mostrados en la imagen 442.

Seleccionando una animación, puede ser reproducida mostrando en el esqueleto el movimiento que se realiza. Para reproducir la animación se pulsa sobre el icono *"play"* mostrado en la parte inferior de la ventana del esqueleto, junto a una línea temporal:

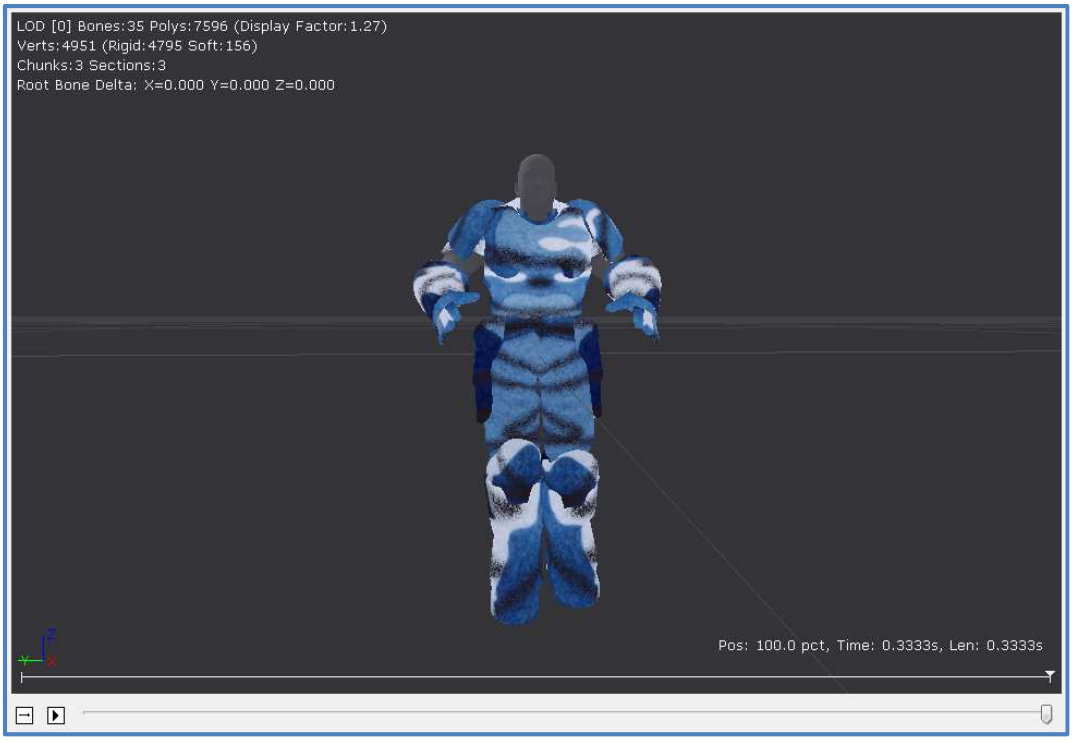


Imagen 443:Reproducción de animación.

En el *"Skeletal Mesh"* importado es necesario definir unos puntos especiales llamados *"sockets"*. Estos puntos representan la zona donde se podrán acoplar otros mesh, como un arma, etc.

Para crear estos puntos en el esqueleto hay que seguir los siguientes pasos:

- En la sección *"Mesh"* de la barra de herramientas superior del *"AnimSet Editor"* se selecciona la opción *"Socket Manager"*.

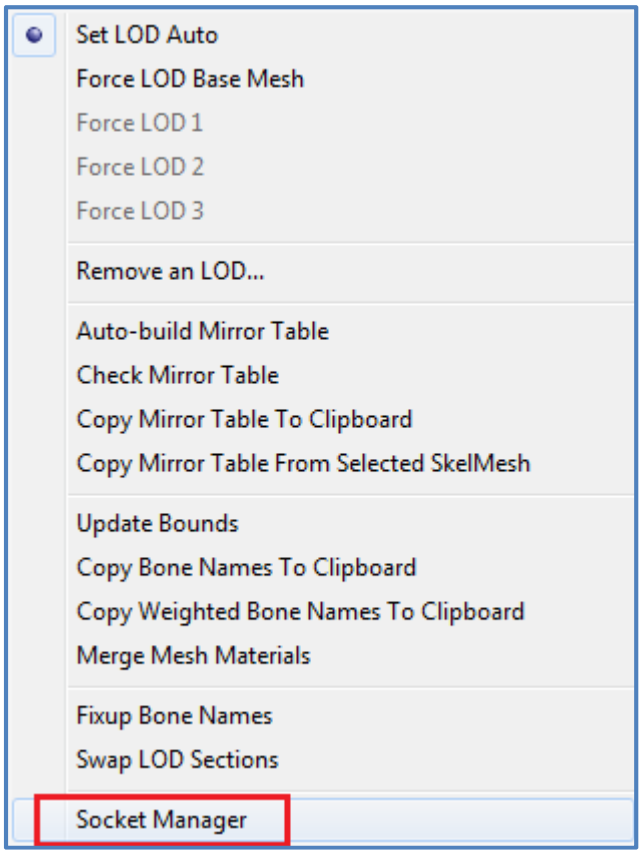


Imagen 444:Localización de "Socket Manager".

Se mostrará una nueva ventana en donde se seguirá la secuencia:

- New Socket → Se selecciona el *"hueso"* donde se quiere acoplar dicho punto del desplegable mostrado con todos ellos.

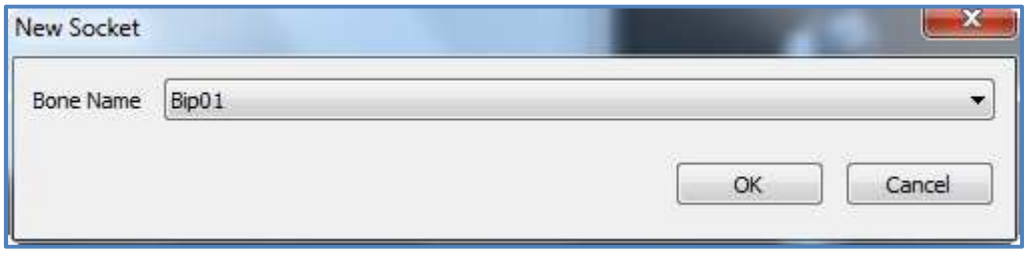


Imagen 445:Creación de "Socket".

- Al pulsar en botón *"OK"* se pide el nombre que se le quiere dar al socket.

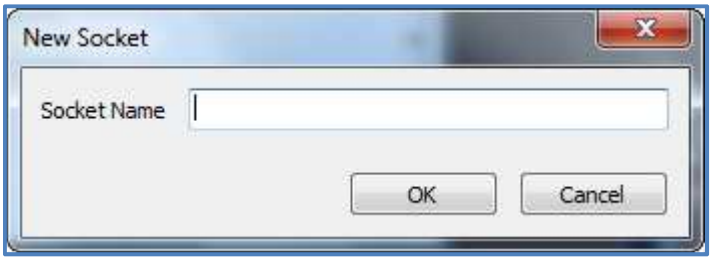


Imagen 446:Nombre del "Socket".

- Se puede introducir el nombre que se quiera al socket creado pero UDK tiene varios nombres prefijados para acoplar ciertos mesh:
  - WeaponPoint.- Punto donde se acoplará el arma principal.
  - DualWeaponPoint.- Punto donde se acoplará el arma secundaria.

En la pantalla *"Socket Manager"* se irán mostrando todos los sockets creados, en la sección *"Sockets"*.

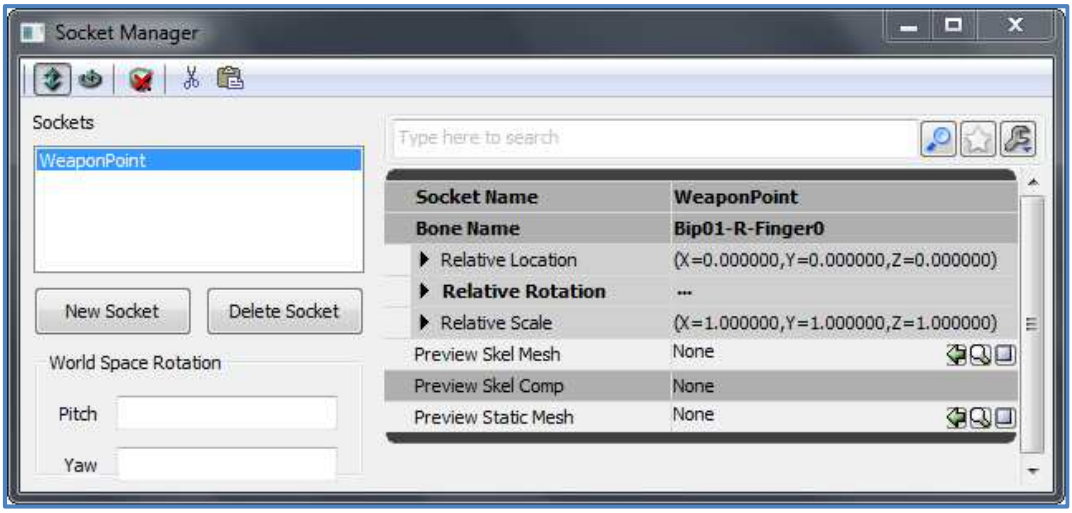


Imagen 447:Interfaz del "Socket Manager".

Al seleccionar un socket creado, se mostrarán sus propiedades en la parte derecha. En estas propiedades se puede modificar tanto la posición del punto como la rotación del mismo. Se puede hacer de forma textual, introduciendo los valores en las propiedades, ó de forma gráfica, con los elementos mostrados en el mesh y seleccionando la opción de *"translate/rotate"* de la parte superior izquierda.

Anim Tree.

Mediante el elemento *"Anim Tree"* se pueden especificar las animaciones que debe reproducir el *"Skeletal Mesh"* en cada una de las circunstancias que se produzcan en el juego.

Para crear un nuevo elemento de este tipo se debe pulsar el botón derecho del ratón sobre cualquier zona despejada en Content Browser y seleccionar *"New AnimTree"*.

Se abrirá una nueva herramienta llamada *"Unreal AnimTree Editor"* que permitirá modificar el contenido de este elemento para reproducir las animaciones en cada circunstancia:

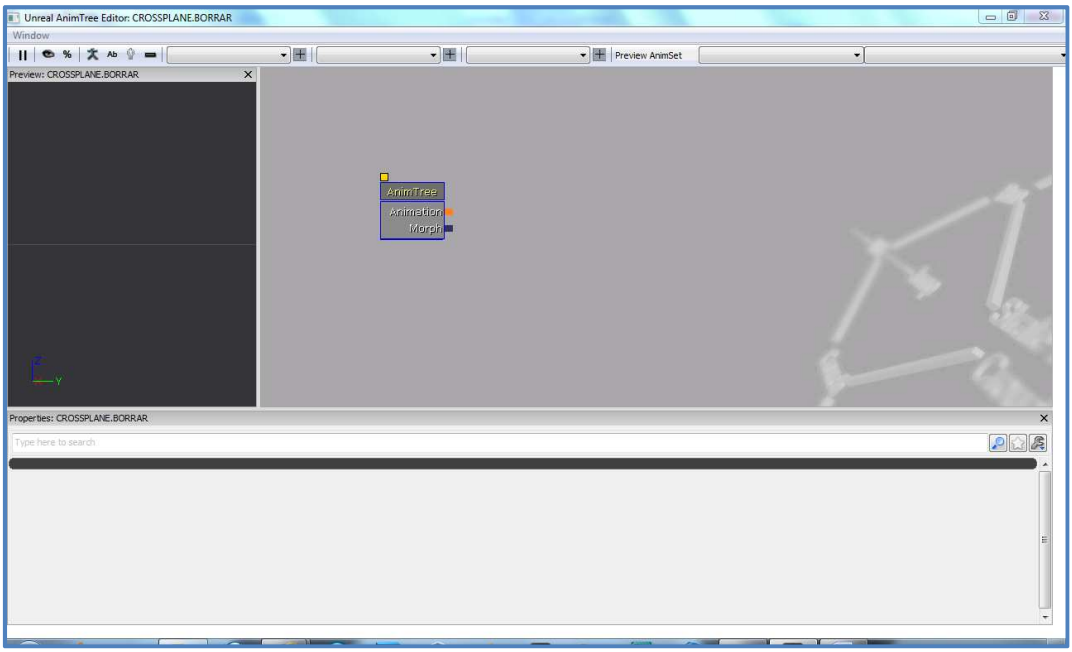


Imagen 448:Interfaz "Anim Tree Editor".

El *"Skeletal Mesh"* debe ser asociado en este elemento. En las propiedades del elemento *"Anim Tree"* es donde se acoplará el *"Skeletal"*:

- En la etiqueta *"Preview Mesh List"* se añadirá una nueva posición a la lista pulsando en la etiqueta , [0].
- En la nueva posición creada, en la etiqueta *"Preview Skell Mesh"*, se insertará el *"SkeletalMesh"* a utilizar, seleccionándolo en la herramienta Content Browser y pulsando el icono .

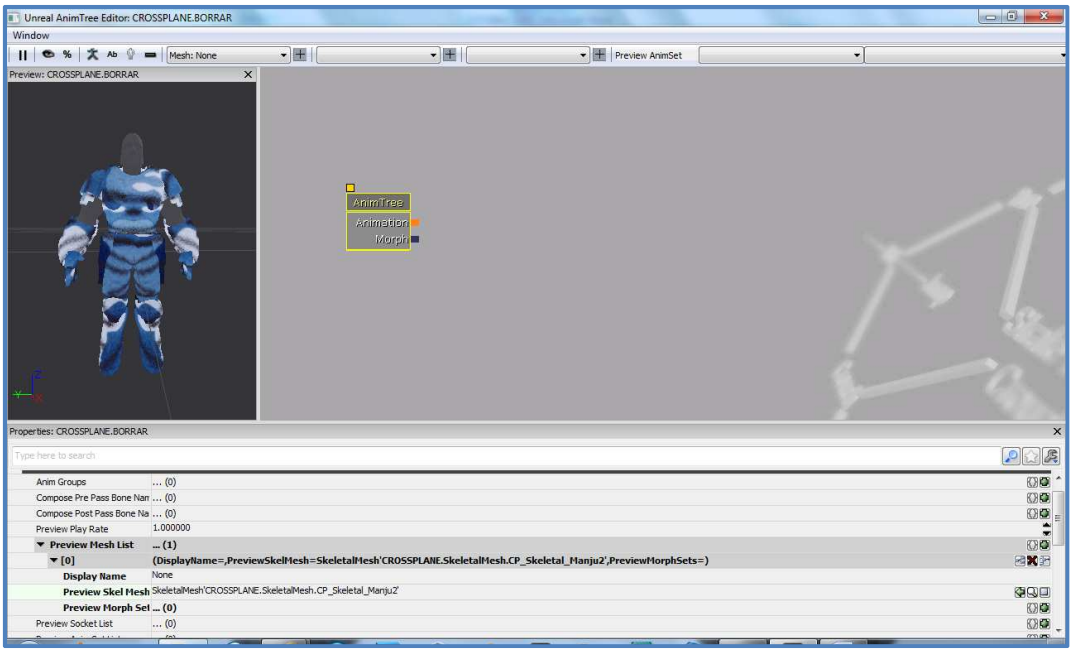


Imagen 449:Asociación "Skeletal Mesh" con "Anim Tree".

Con el mesh ya asociado se pueden ir acoplando los elementos que representarán cada una de las situaciones que se pueden producir en el juego y asociarles las animaciones:

- Se va a insertar el elemento que representa cada una de las condiciones físicas como *"andar"*, *"caer"* y *"nadar"*:

  Botón derecho del ratón → New Animation Node → BlendBy → UTAnimBlendByPhysics.

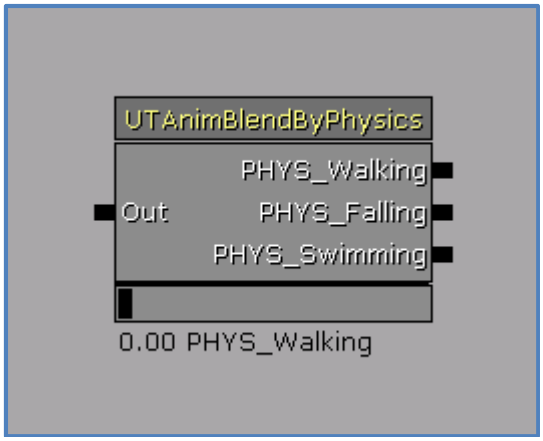


Imagen 450:Elemento "UTAnimBlendByPhysics".

  Y este nuevo elemento se unirá con *"Anim Tree"*.

- Es necesario insertar el elemento que hace distinción entre el estado de *"correr"* ó *"agacharse"*:

  New Animation Node → BlendBy → UDKAnimBlendByPosture.

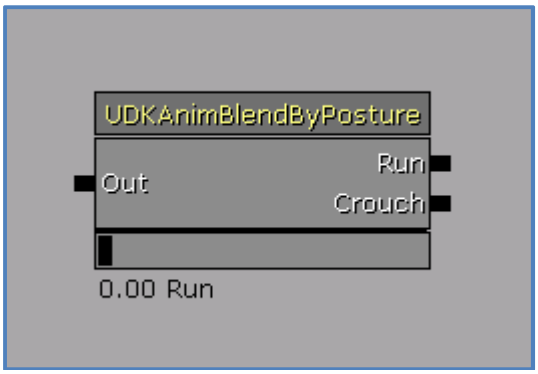


Imagen 451:Elemento "UDKAnimBlendByPosture".

  Se unirá su entrada con la salida *PHYS_Walking* del elemento anterior.

- Otro elemento que representa cada una de las circunstancias de caída del mesh:

  New Animation Node →BlendBy → UTAnimBlendByFall

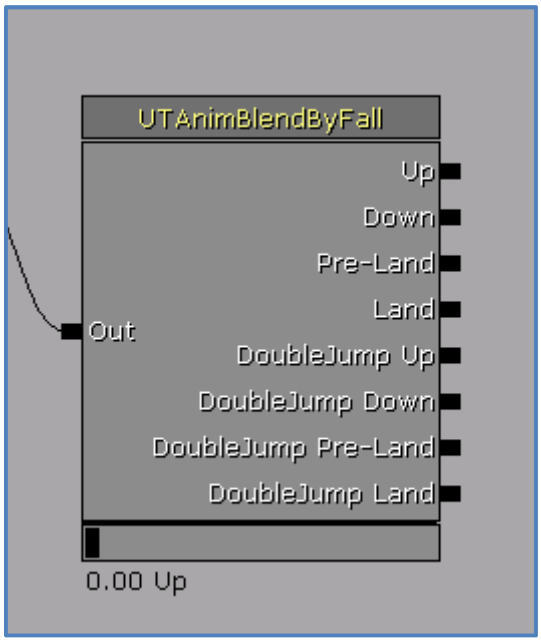


Imagen 452:Elemento "UTAnimBlendByFall".

Se unirá su entrada con la salida *"PHYS_Falling"* del elemento *"UTAnimBlendByPhysics"*.

- Cuando el personaje se encuentre *"ocioso"* se presenta mediante el elemento:

  New Animation Node → Blend By → UDKAnimBlendByIdle.

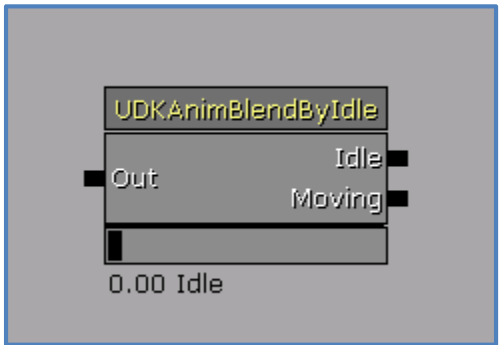


Imagen 453:Elemento "UDKAnimBlendByIdle".

Se unirá su entrada con la salida *"Run"* del elemento *"UDKAnimBlendByPosture"*.

- En cada circunstancia de movimiento del personaje se debe indicar la animación a reproducir cuando se mueve hacia delante, atrás, derecha ó izquierda.

  New Animation Node → Directional → AnimNodeBlendDirectional

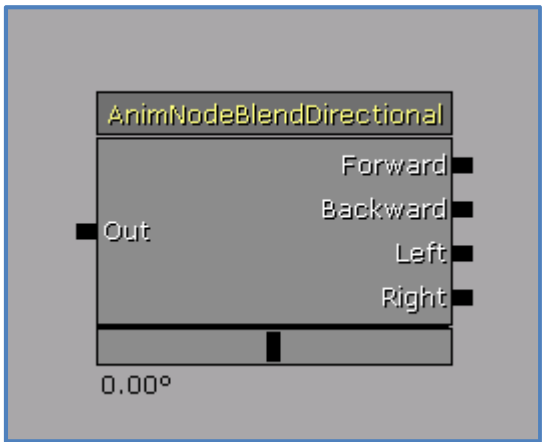


Imagen 454:Elemento "AnimNodeBlendDirectional".

Se unirá con la salida *"Moving"* del elemento *"…Idle"*.

Toda la secuencia de elementos quedaría de la siguiente manera, mostrada en la imagen 455.

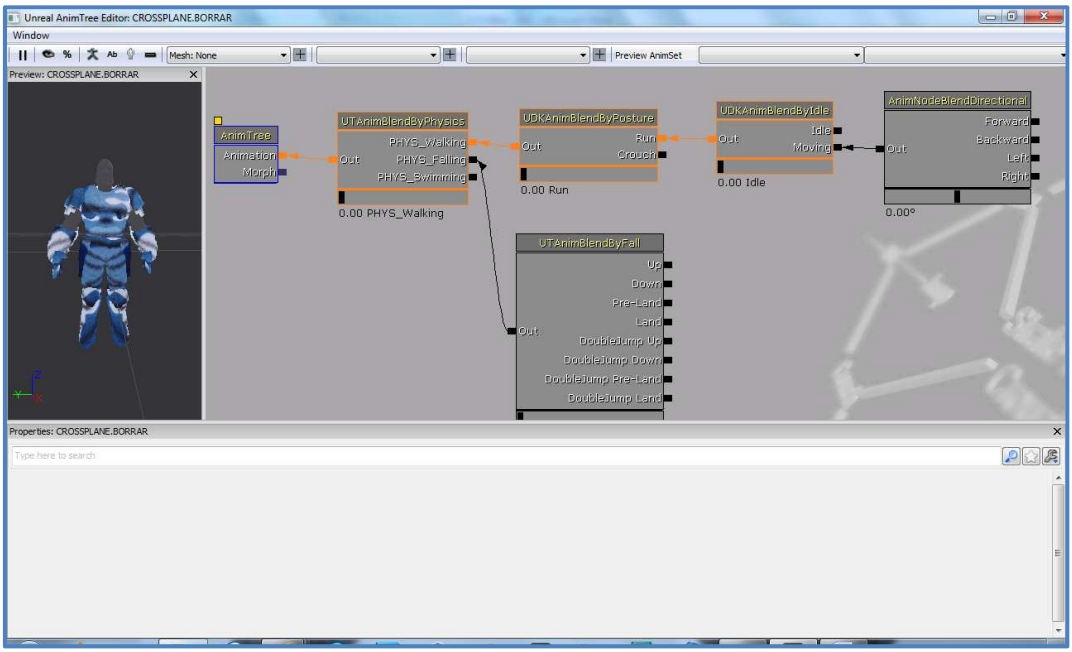


Imagen 455:Asociación completa de elementos en "Anim Tree".

En el elemento final de toda secuencia se debe indicar las animaciones a reproducir en cada una de las circunstancias, pero antes hay que indicar el *"Anim Set"* a utilizar.

Para ello se deben modificar las propiedades del elemento *"Anim Tree"*:

- En la etiqueta *"Anim Set List"* añadir tantas posiciones como animaciones se quieran utilizar del *"Anim Set"*, pulsando el icono por cada posición.
- En cada una de las posiciones creadas hay que adjuntar el elemento *"AnimSet"*.
- En la etiqueta de cada posición *"Preview Anim Sets"* se añade una posición, [0], mediante el icono. Con el elemento *"AnimSet"* seleccionado en la

herramienta Content Browser se añade a esta nueva posición [0] mediante el icono .

- En la etiqueta de cada posición *"Display Name"*, imagen 456, se inserta el nombre de la animación. Este nombre debe ser el mismo que se tiene en el *"AnimSet"*.

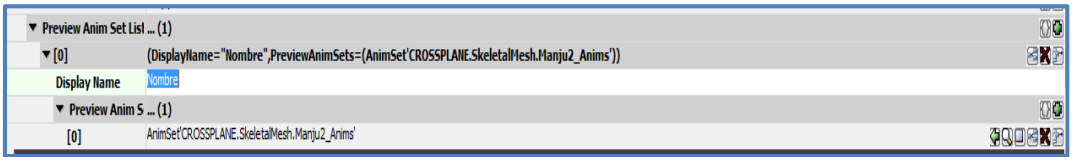


Imagen 456:Etiqueta "Display Name" del elemento "AnimSet".

Cada una de las animaciones puede ser utilizada en las circunstancias insertadas. El elemento que permite reproducir una animación es llamado *"UTAnimNodeSeqWeap"*:

New Animation Sequence → UTAnimNodeSeqWeap

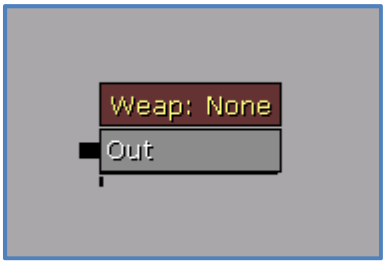


Imagen 457:Elemento "UTAnimNodeSeqWeap".

En las distintas propiedades de este elemento se indicará la reproducción de la animación deseada:

- AnimNodeSequence → Anim Seq Name.- Se indica el nombre de la animación a reproducir. Este nombre es el mismo que se tiene en el elemento *"AnimSet"*.
- AnimNodeSequence → Playing.- Para mostrar el movimiento.
- AnimNodeSequence → Looping.- La reproducción del movimiento sea en bucle.
- AnimNodeSequence → Rate.- Utilizado para cambiar la velocidad de reproducción del movimiento.

En cada salida se asociará un elemento de este tipo indicando la animación a reproducir cuando se produzca la circunstancia en el juego. Las animaciones asociadas se mostrarán en el *"Skeletal Mesh"* asociado en la parte izquierda del editor.

Es posible elegir la animación a mostrar en el *"Skeletal"*, es decir, elegir el flujo activando unas salidas u otras. Cada uno de los elementos de este editor, en la parte inferior, se tiene una barra de desplazamiento que activará una u otra opción para mostrar en el mesh.

En las siguientes imágenes se muestra un ejemplo de reproducción de una animación u otra mediante la barra de desplazaiento del elemento *"AnimNodeBlendDirectional"*:

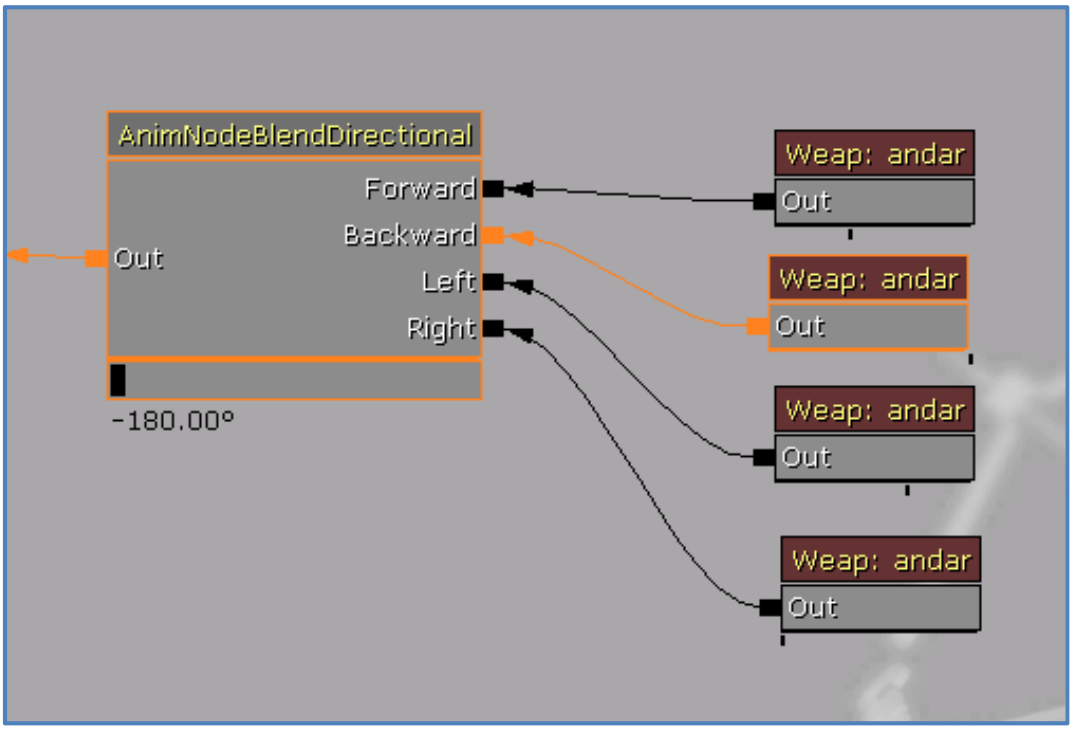


Imagen 458:Ejecución de la segunda salida.

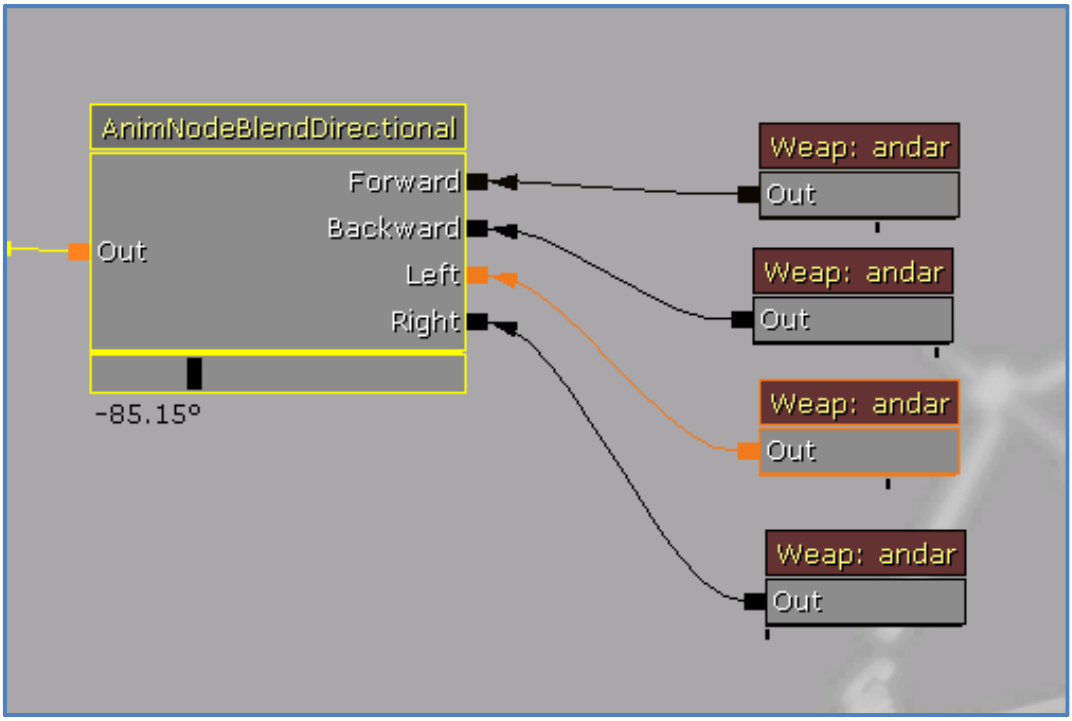


Imagen 459:Ejecución de la tercera salida.

El camino que se encuentra activo se representa de un color naranja intenso, mostrando desde el inicio, *"Anim Tree"*, hasta el final, *"Weap"*.

Cuando se produzcan cada una de las circunstancias en el juego, la animación asociada será reproducida. Los elementos que se pueden insertar en el editor pueden representar todas las circunstancias que se pueden producir durante el juego. En este manual sólo se ha representado la circunstancia de "andar" en todas las direcciones

posibles para no extender demasiado dicho manual, pero todas las demás circunstancias se realizarían de una forma similar.

9.5. Código a insertar/modificar para utilizar el nuevo personaje.

Cuando un *"Skeletal Mesh"* es importado en UDK y se tienen creados todos los elementos imprescindibles a partir de él, es necesario modificar ó insertar código en la implementación base de UDK.

Para un nuevo personaje hay que crear una nueva clase que extienda de *"UTFamilyInfo"*. Todos los personajes principales deben extender de esta clase para poder utilizarlo en la implementación.

Por defecto, el personaje principal de UDK está implementado en la clase *"UTFamilyInfo_Liandri"*:

UDK / Development / Src / UTGame / Classes / UTFamilyInfo_Liandri.uc

Las variables mínimas a crear en la implementación de un personaje son:

- CharacterMesh.- Variable donde se indicará el Skeletal Mesh que mostrará el personaje:

  CharacterMesh = SkeletalMesh '…'

- PhysAsset.- Variable donde se indicará el PhysicsAsset del personaje:

  PhysAsset = PhysicsAsset ' …'

- AnimSets(0).- Variable donde se indicará el AnimSet del personaje.

  AnimSets(0) = AnimSet '…'

La implementación de la clase del nuevo personaje sería:

```
/**
 * Copyright 1998-2010 Epic Games, Inc. All Rights Reserved.
 */

class UTFamilyInfo_Manju extends UTFamilyInfo
abstract;

defaultproperties
{
FamilyID="LIAM"
Faction="Manju"

CharacterMesh=SkeletalMesh'CROSSPLANE.SkeletalMesh.CP_Skeletal_Manju2'

AnimSets(0)=AnimSet'CROSSPLANE.SkeletalMesh.Manju2_Anims'

PhysAsset=PhysicsAsset'CROSSPLANE.SkeletalMesh.CP_Skeletal_Manju2_Physics'

ArmMeshPackageName="CH_Corrupt_Arms"
ArmMesh=CH_Corrupt_Arms.Mesh.SK_CH_Corrupt_Arms_MaleA_1P
ArmSkinPackageName="CH_Corrupt_Arms"
RedArmMaterial=CH_Corrupt_Arms.Materials.MI_CH_Corrupt_FirstPersonArms_VRed
BlueArmMaterial=CH_Corrupt_Arms.Materials.MI_CH_Corrupt_FirstPersonArms_VBlue

CharacterTeamHeadMaterials[0]=MaterialInterface'CH_Corrupt_Male.Materials.MI_CH_C
orrupt_MBody01_VRed'
```

```
CharacterTeamBodyMaterials[0]=MaterialInterface'CH_Corrupt_Male.Materials.MI_CH_C
orrupt_MHead01_VRed'
CharacterTeamHeadMaterials[1]=MaterialInterface'CH_Corrupt_Male.Materials.MI_CH_C
orrupt_MBody01_VBlue'
CharacterTeamBodyMaterials[1]=MaterialInterface'CH_Corrupt_Male.Materials.MI_CH_C
orrupt_MHead01_VBlue'

BaseMICParent=MaterialInstanceConstant'CH_All.Materials.MI_CH_ALL_Corrupt_Base'
BioDeathMICParent=MaterialInstanceConstant'CH_All.Materials.MI_CH_ALL_Corrupt_Bio
Death'
}
```

Imagen 460:Implementación de la clase del nuevo personaje.

Este código debe ser guardado en un archivo con extensión *".uc"* en el mismo directorio de todos los personajes:

UDK / Development / Src / UTGame / Classes /

Mediante esto se tiene el archivo que contiene la clase del nuevo personaje con las características propias creadas.

Con la clase creada del nuevo personaje se debe modificar la configuración del motor para que al lanzar un juego, el personaje principal mostrado sea el personaje creado. Para ello hay que modificar la implementación que se encuentra en el archivo:

UDK / Development / Src / UTGame / Classes / UTPlayerReplicationInfo.uc

En la sección *"defaultproperties"* localizada al final del archivo, se debe modificar la variable *"charClassInfo"* indicando la clase del nuevo personaje creado para lanzarlo como personaje principal en el juego.

```
defaultproperties
{
LastKillTime=-5.0
DefaultHudColor=(R=64,G=255,B=255,A=255)
VoiceClass=class'UTGame.UTVoice_Robot'
CharPortrait=Texture2D'CH_IronGuard_Headshot.HUD_Portrait_Liandri'

CharClassInfo=class'UTGame.UTFamilyInfo_Manju'
}
```

Imagen 461:Implementación de llamada a clase del personaje a utilizar.

Además, para tener completo el nuevo personaje, se debe indicar el *"Anim Tree"* creado para ese personaje en la implementación del motor.

En el archivo:

UDK / Development / Src / UTGame / Classes / UTPawn.uc

se debe modificar la variable *"AnimTreeTemplate"* del *"Object Class=SkeletalMeshComponent Name=WPawnSkeletalMeshComponent"* donde se indicará el nuevo *"Anim Tree"* a utilizar:

AnimTreeTemplate = AnimTree '…'

La implementación quedaría de la siguiente manera:

```
Begin Object Class=SkeletalMeshComponent Name=WPawnSkeletalMeshComponent
bCacheAnimSequenceNodes=FALSE
AlwaysLoadOnClient=true
AlwaysLoadOnServer=true
bOwnerNoSee=true
```

```
CastShadow=true
BlockRigidBody=TRUE
bUpdateSkelWhenNotRendered=false
bIgnoreControllersWhenNotRendered=TRUE
bUpdateKinematicBonesFromAnimation=true
bCastDynamicShadow=true
Translation=(Z=8.0)
RBChannel=RBCC_Untitled3
RBCollideWithChannels=(Untitled3=true)
LightEnvironment=MyLightEnvironment
bOverrideAttachmentOwnerVisibility=true
bAcceptsDynamicDecals=FALSE
//AnimTreeTemplate=AnimTree'CH_AnimHuman_Tree.AT_CH_Human'
AnimTreeTemplate=AnimTree'CROSSPLANE.CP_AnimTree_Manju2'
bHasPhysicsAssetInstance=true
TickGroup=TG_PreAsyncWork
MinDistFactorForKinematicUpdate=0.2
bChartDistanceFactor=true
//bSkipAllUpdateWhenPhysicsAsleep=TRUE
RBDominanceGroup=20
Scale=1.075
MotionBlurScale=0.0
bAllowAmbientOcclusion=false
End Object
```

Imagen 462:Implementación de la clase "UTPawn".

Esta clase es la utilizada para insertar nuevos personajes en un nivel en modo de ejecución mediante el elemento *"Actor Factory"* de la herramienta Unreal Kismet. Es recomendable implementar una nueva clase que extienda de esta última pero modificando los elementos básicos de un personaje para poder acoplar personajes diferentes al principal en el mapa. Esta nueva clase quedaría:

```
class UTPawnEnemigo extends UTPawn;

// members for the custom mesh
var SkeletalMesh defaultMesh;
var MaterialInterface defaultMaterial0;
var AnimTree defaultAnimTree;
var array<AnimSet> defaultAnimSet;
var AnimNodeSequence defaultAnimSeq;
var PhysicsAsset defaultPhysicsAsset;

simulated function SetCharacterClassFromInfo(class<UTFamilyInfo> Info)
{
Mesh.SetSkeletalMesh(defaultMesh);
Mesh.SetMaterial(0,defaultMaterial0);
Mesh.SetPhysicsAsset(defaultPhysicsAsset);
Mesh.AnimSets=defaultAnimSet;
Mesh.SetAnimTreeTemplate(defaultAnimTree);
Mesh.SetScale(DefaultMeshScale);
BaseTranslationOffset=80;

}

defaultproperties
{
defaultMesh=SkeletalMesh'CH_LIAM_Cathode.Mesh.SK_CH_LIAM_Cathode'
defaultAnimTree=AnimTree'CH_AnimHuman_Tree.AT_CH_Human'
defaultAnimSet(0)= AnimSet'CH_AnimHuman.Anims.K_AnimHuman_BaseMale'
defaultPhysicsAsset= PhysicsAsset'CH_AnimCorrupt.Mesh.SK_CH_Corrupt_Male_Physics'
```

```
Begin Object Name=WPawnSkeletalMeshComponent
AnimTreeTemplate=AnimTree'CH_AnimHuman_Tree.AT_CH_Human'
End Object

DefaultMeshScale=2.5
}
```

Imagen 463:Implementación de la nueva clase "UTPawn".

Si no se crease esta clase, todos los personajes que se acoplasen al nivel serían iguales al personaje principal. Esta clase permite seleccionar distintas opciones de personaje a acoplar.

9.6. Compilación del código

Cuando se modifica el código del motor, es necesario arrancar de nuevo toda la aplicación del motor. Aparecerá un mensaje indicando que se han producido cambios en el código y si se quiere recompilar todo. En caso afirmativo, se realizará la compilación de todas las clases en consola mostrando todos los avisos y errores en el caso que los hubiera, o indicando que todo se ha compilado correctamente.

En la siguiente imagen se muestra un ejemplo de avisos encontrados al realizar la compilación de todos los archivos:

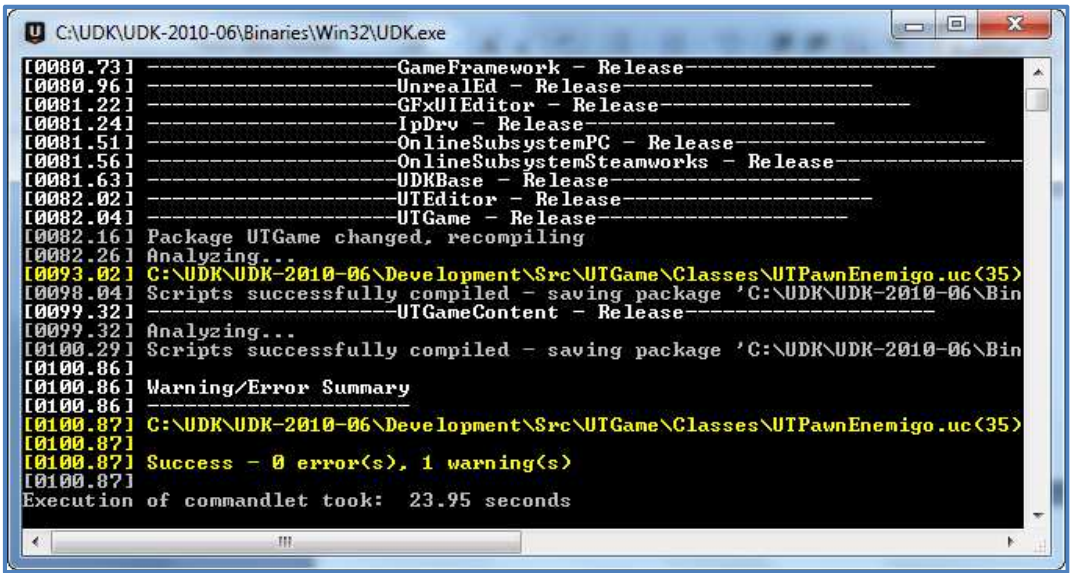


Imagen 464:Ejemplo de avisos en la compilación de los archivos del motor.

En la imagen 465 se muestra un ejemplo del mensaje mostrado por consola al realizar la correcta compilación de todos los archivos:

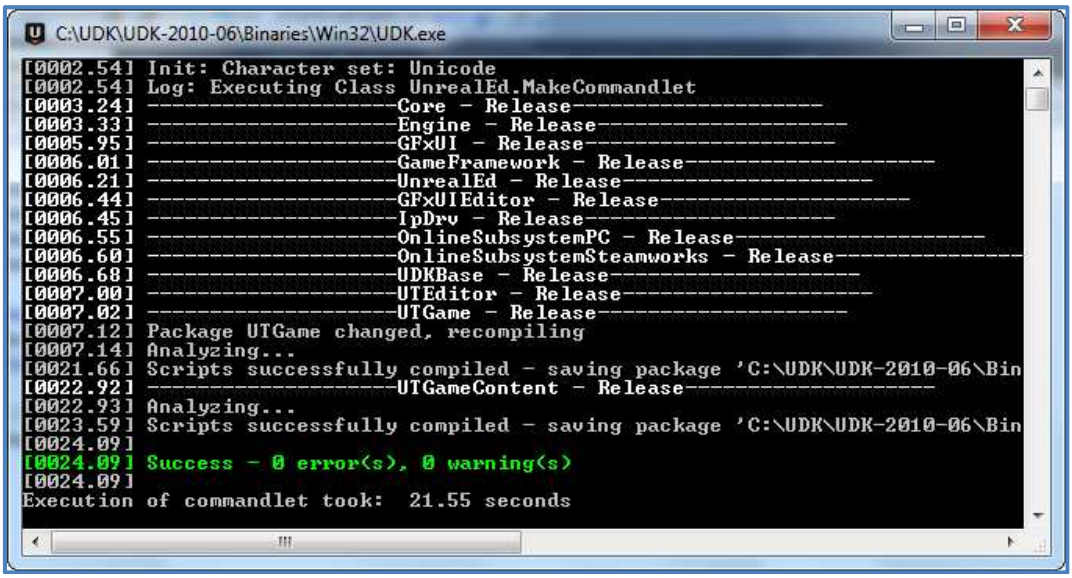


Imagen 465:Ejemplo de la correcta compilación de los archivos del motor.

10. CASO PRÁCTICO: JUEGO "CROSSPLANE".

Para mostrar un ejemplo de implementación de un juego con este motor, se ha realizado el juego *"CrossPlane"*.

Imagen 466:Juego "CrossPlane".

Para este juego se ha implementado un nuevo personaje que será utilizado como personaje principal, realizando todo su diseño y creación mediante la herramienta *"3D Studio Max"*. Este personaje, *"Manju"*, será el protagonista de todo el juego, el cual será el jugador que se controlará durante el juego para ir realizando todas las acciones.

El juego está compuesto de dos niveles, en los que se muestran las dos opciones de creación de mapas ó niveles, interior y exterior.

10.1. Nivel interior.

En este nivel del videojuego se ha utilizado el modo *"sustractive"* de creación de objetos para realizarlo en interior. Se va a especificar cada una de las características y decisiones tomadas en la implementación de dicho nivel, tomando como referencia los capítulos anteriores de este documento donde se especifica la forma de realización de cada propiedad.

Cámara en 3ª persona.

Se ha decidido acoplar la cámara en 3ª persona para obtener una jugabilidad diferente, tal y como se explicó en el capítulo 8.4.

En la siguiente imagen se muestra la perspectiva de la cámara en 3ª persona:

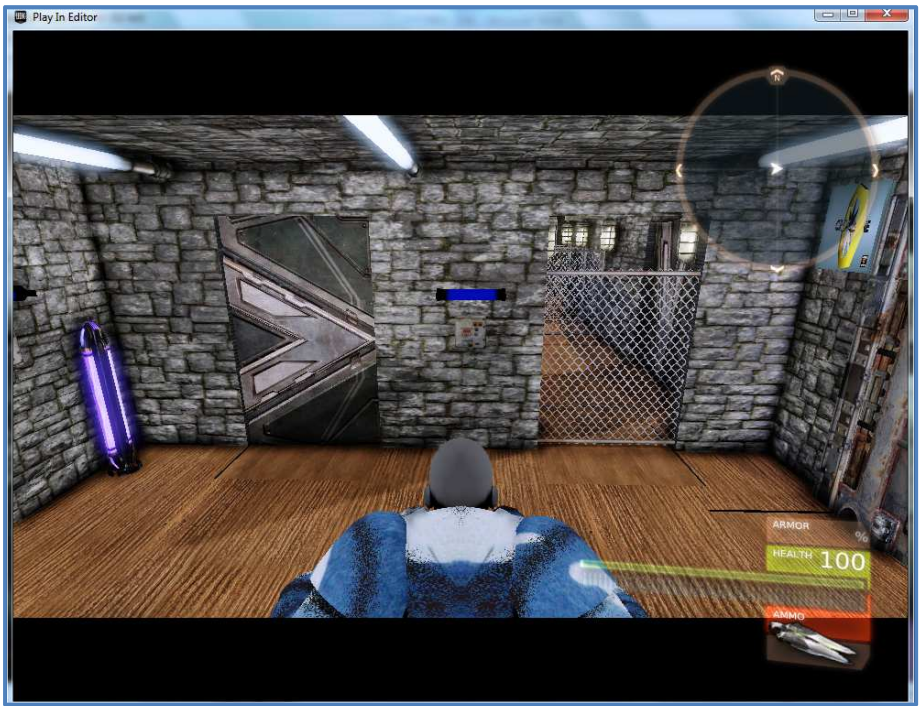


Imagen 467:Cámara en 3ª persona del juego.

Inventario de armas.

Al comienzo de este nivel el protagonista no dispone de armas. Se ha implementado esta circunstancia indicándolo en las propiedades del nivel en la sección *"World Properties"* (cosultar capítulo 7.5).

A lo largo del desarrollo del juego se podrá ir ampliando el inventario de armas. Para ello se añadieron al nivel las factorías necesarias para proporcionar al personaje de todos los tipos de armas posibles.

En la circunstancia en que el protagonista muera y se haya realizado cierta acción del nivel se ha decidido que el protagonista vuelva a reaparecer pero con el inventario de armas completo. La realización de la acción en el nivel se controla con una variable global de tipo *"boolean"* llamada *"Hab_Armas"* inicializada a *"false"* y modificado su valor al realizar la acción que controla.

Al reaparecer el protagonista se debe comprobar la variable global y en el caso correcto hay que dotar al personaje principal con las armas. La implementación en Unreal Kismet sería la mostrada en la siguiente imagen:

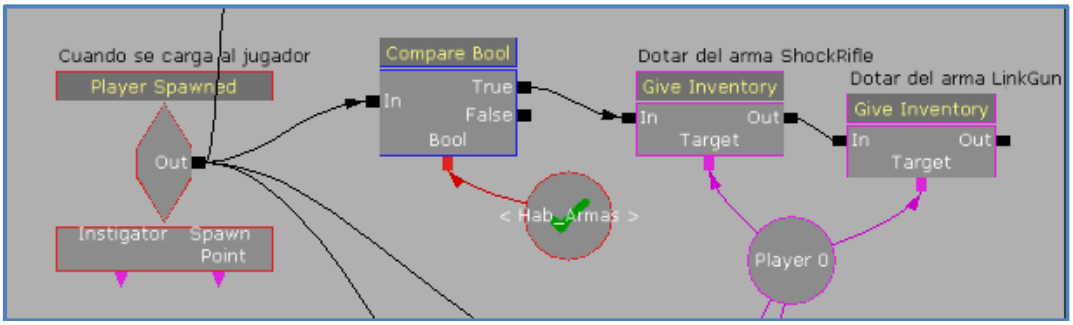


Imagen 468:Implementación dotación de armas del protagonista.

Apertura puertas.

Todas las salas de la instalación están cerradas mediante objetos *"static mesh"* que actúan como puertas. En la realización de ciertas acciones por parte del protagonista se muestra mediante cinemática la apertura de algunas puertas. Esta visualización de apertura de la puerta, y la propia apertura, se realiza mediante un elemento de tipo Matinee de la herramienta Unreal Kismet. (consultar capítulo 6).

Otras puertas son abiertas mediante *"triggers"* que interactúan con el protagonista, tanto de contacto como de uso. Además la acción de cada *"trigger"* estará condicionada según el valor de alguna variable global. Estas variables globales representarán ciertas acciones del nivel si han sido realizadas o no. En la imagen 469 se muestra un ejemplo de implementación de esta circunstancia:

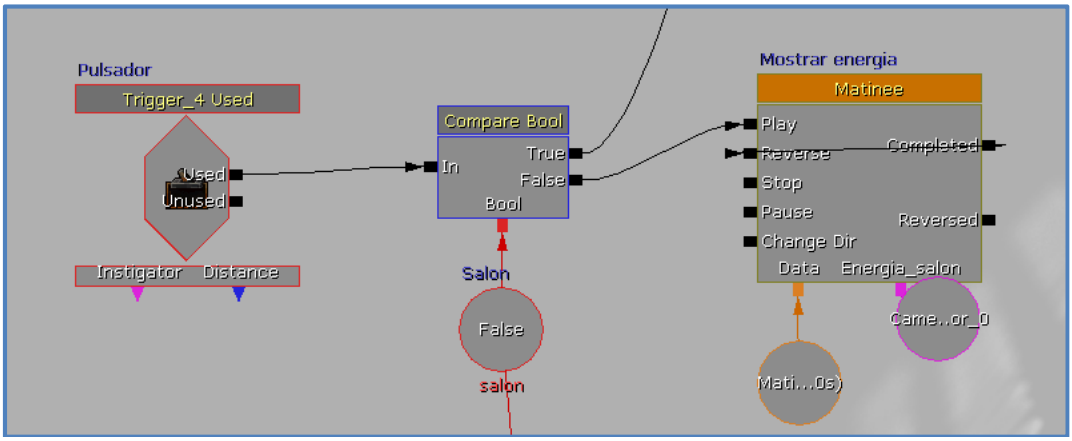


Imagen 469:Implementación de consulta de condición.

Una opción creada, al inicio del nivel, es la apertura de la puerta de modo aleatorio. Se tienen distintos trigger de uso y de manera aleatoria se abrirá la puerta ó se modificará el valor de vida del protagonista. En el juego se han utilizado tres trigger para poder abrir la puerta.

La aleatoriedad se ha conseguido mediante el elemento *"Random"* proporcionado por la herramienta Unreal Kismet. En las salidas de este elemento se asociarán las distintas opciones a realizar, realizándose una sóla de ellas elegida de forma aleatoria.

Como es un caso aleatorio, se puede llegar a tener que todos los trigger modifiquen la vida del protagonista sin llegar a abrir la puerta, quedando el juego bloqueado. Es un caso posible, pero no probable. Hay que controlar este hecho, ya que no se debe permitir. Para ello se usará una variable global se tipo *"int"*, llamada *"numero"*, que servirá como contador de las veces que se ha modificado la salud del protagonista. En el caso de usar un *"trigger"* y el contrador ya está a dos es obligatorio que se abra la puerta, no dejando que sea de manera aleatoria.

Por lo tanto, la secuencia de acciones en la implementación sería: al hacer uso de un trigger lo primero que se debe realizar es comprobar el valor de la variable *"numero"*. En el caso que el valor sea inferior a dos se pasaría a la elección aleatoria de abrir la puerta ó modificar la vida del protagonista. A su vez, en ambas opciones, hay que aumentar en una unidad el valor de la variable *"numero"*.

La implementación de todas estas acciones en Kismet sería:

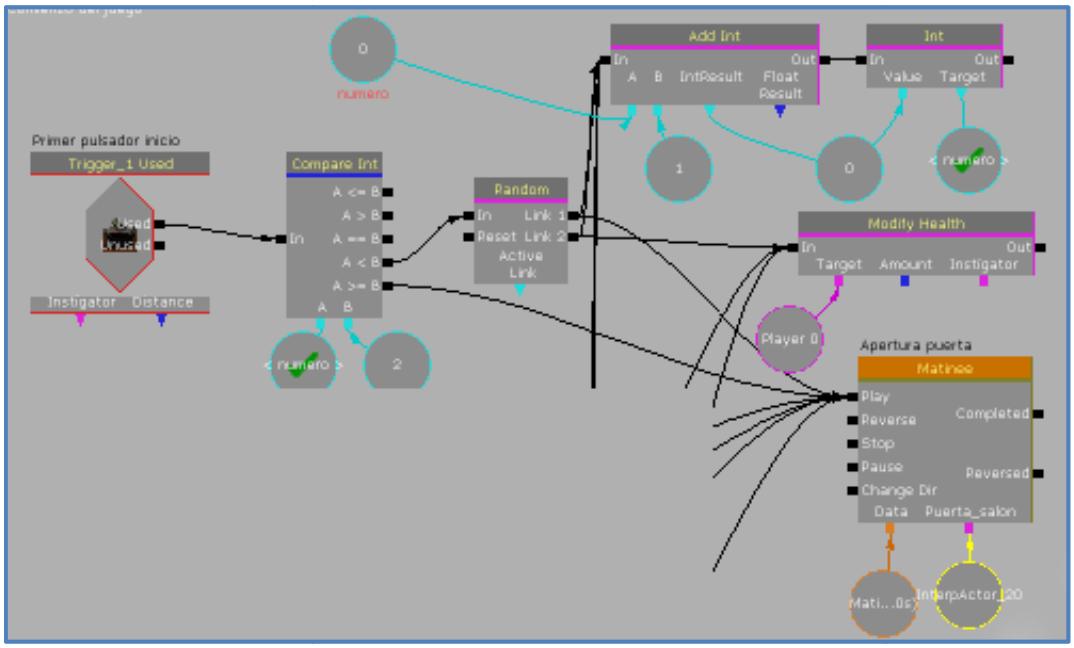


Imagen 470:Implementación de condiciones en apertura puerta.

Cinemáticas.

Para guiar al jugador sobre la secuencia de acciones que debe realizar, en muchas ocasiones, al realizar una acción, se muestra qué sucede en otra sala y a donde hay que ir, o la apertura de una puerta de una sala, o dónde se encuentra lo que se necesita para poder continuar. (consultar capítulo 6).

Imagen 471:Ejemplo visualización.

En la imagen 471 se muestra un ejemplo de cinemática visualizada por el juegador.

Todas estas visualizaciones se realizan mediante cinemáticas proporcionadas por los elementos Matinee de la herramienta Kismet. La implementación de uno se estos hechos se muestra en la siguiente imagen:

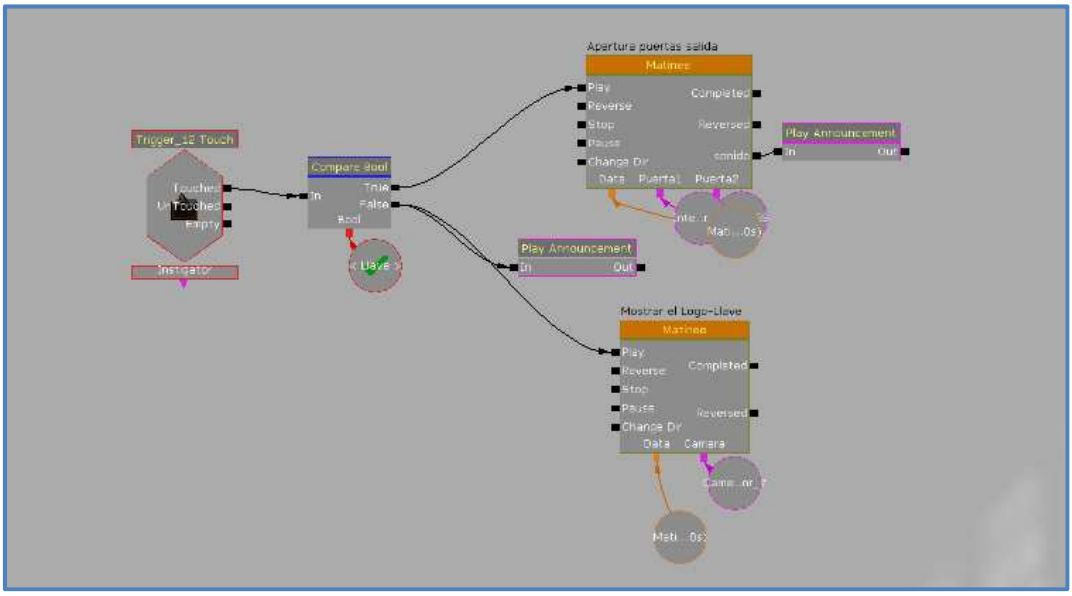


Imagen 472:Ejemplo de implementación de visualizaciones con condición.

La implementación de una apertura de puerta a modo de ejemplo en la herramienta Unreal Matinee sería de la siguiente manera:

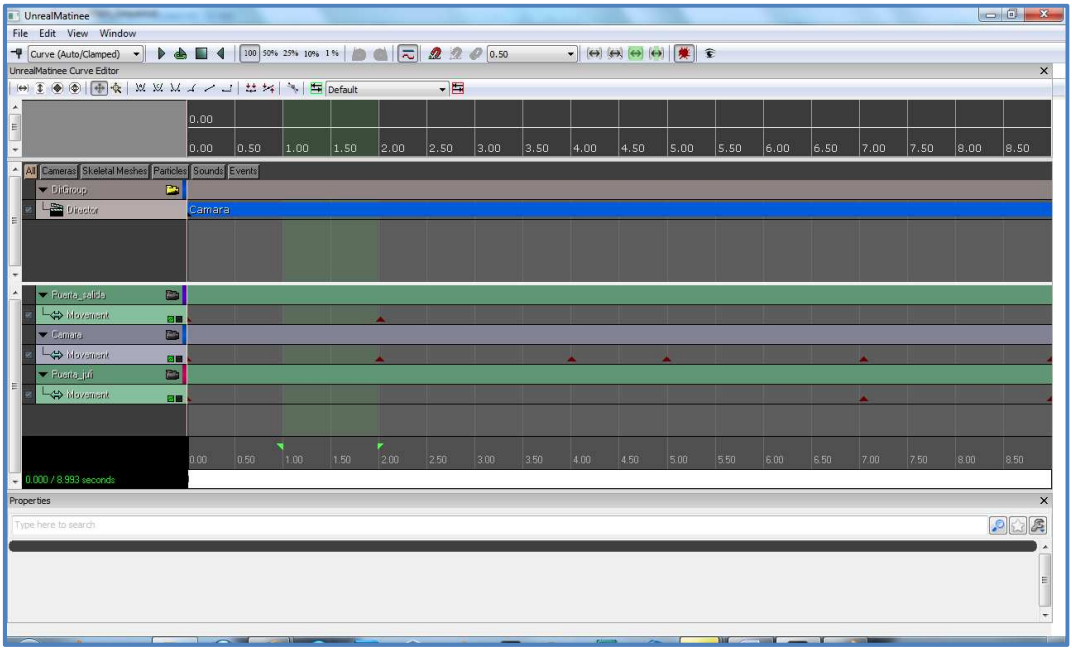


Imagen 473:Ejemplo de implementación de apertura de puerta en "Matinee".

Las restantes aperturas y sus correspondientes visualizaciones serían semejantes a la mostrada. Este ejemplo también se puede tomar como referencia para las simples visualizaciones ya que la implementación sería la misma pero sin los *"Asset"* de movimiento.

Muchas de estas visualizaciones están condicionadas por una variable global, que según el valor de ésta, se mostrará una opción u otra, como es lógico. Las variables globales son utilizadas para controlar si se ha realizado o no una acción y controlar el flujo de secuencia del juego.

Enemigo.

En un cierto momento del nivel se acoplará un nuevo personaje en el que se activará el *"DeathMach"*, con lo que será el enemigo del protagonista.

Antes de la aparición del enemigo se muestra una animación del mismo actor desafiando al jugador. Para crear este tipo de animación consultar el capítulo 6 y 8.8.

En la siguiente imagen se muestra la implementación para la visualización de la animación del personaje:

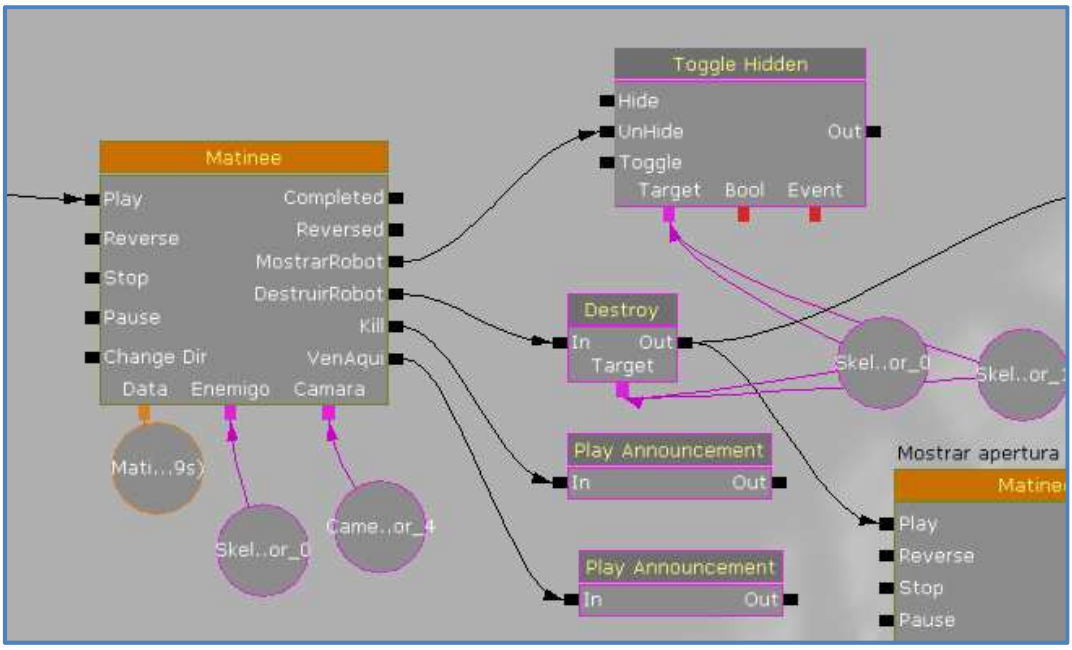


Imagen 474:Reproducción de animación de personaje.

Cuando se ha mostrado la animación, se acoplará al nivel el enemigo. (Consultar capítulo 8.9). Para este enemigo se ha implementado una nueva clase llamada *"PawnEnemigo"* indicando cada uno de los elementos necesarios para crear un nuevo personaje en el motor.

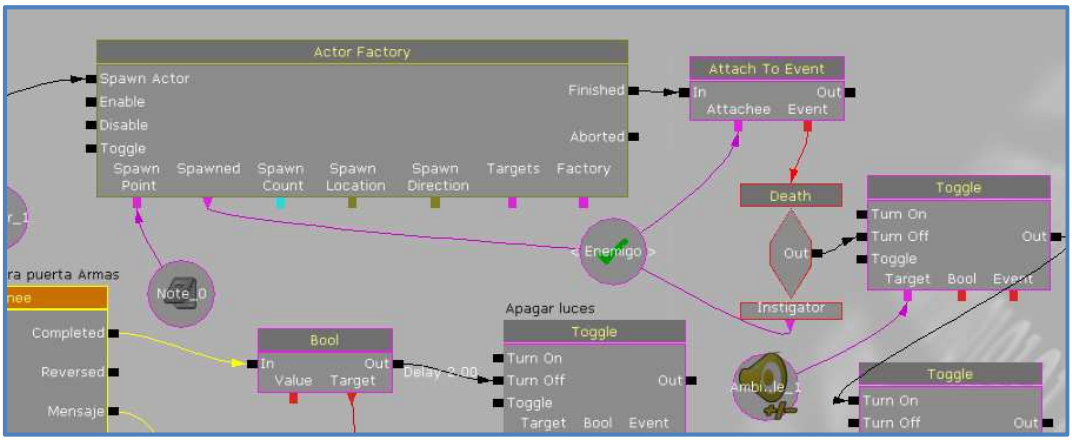


Imagen 475:Control de vida del enemigo acoplado.

También se quiere controlar en la implementación cuándo se ha conseguido matar al enemigo. (Consultar capítulo 8.10). En el nivel, cuando se ha matado al enemigo, se realiza el cambio de música reproducida

Iluminación.

Toda la iluminación utilizada en el nivel es del tipo *"punto de luz"* con la propiedad *"toggleable"*. (Consultar capítulo 7). Esta propiedad permite encender y apagar las luces mediante un *"trigger"*.

Se ha querido hacer en la implementación que se enciendan o se apaguen las luces con un único interruptor (trigger) (Consultar capítulo 5 y 8.2). Para ello es necesaria una variable de tipo *"boolean"*, llamada *"luz"*, que controle cómo se encuentran las luces para realizar la acción contraria. La implementación en Unreal Kismet sería:

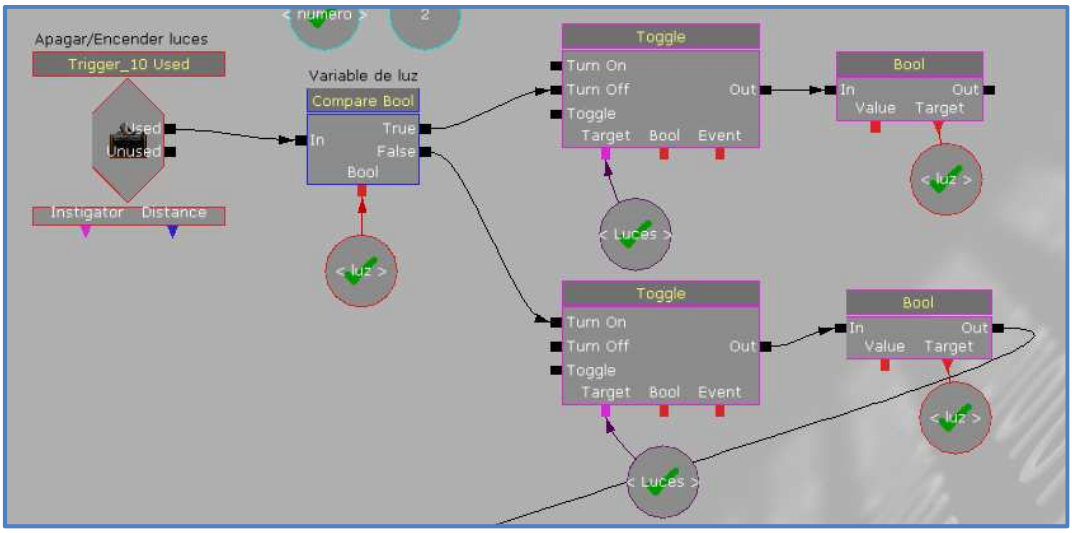


Imagen 476:Apagado y encendido de luces mediante pulsador.

Para el uso en la implementación de todas las luces se ha creado una lista de elementos de puntos de luz llamada *"luces"* que contiene a todas ellas, mostrada en la siguiente imagen:

Imagen 477:Lista de luces.

También se ha utilizado el tipo de iluminiación *"foco"* para resaltar el pulsador de la *"sala de las columnas"*. Mediante el color, en rojo o en verde, se indica si se encuentra activo o no.

Sonido y música.

Para ambientar el desarrollo del nivel, desde el comienzo se reproduce música. Esta música se irá modificando según transcurra la acción, cambiando entre más o menos energía.

En la imagen 478 se muestra un ejemplo de reproducción de música en el nivel.

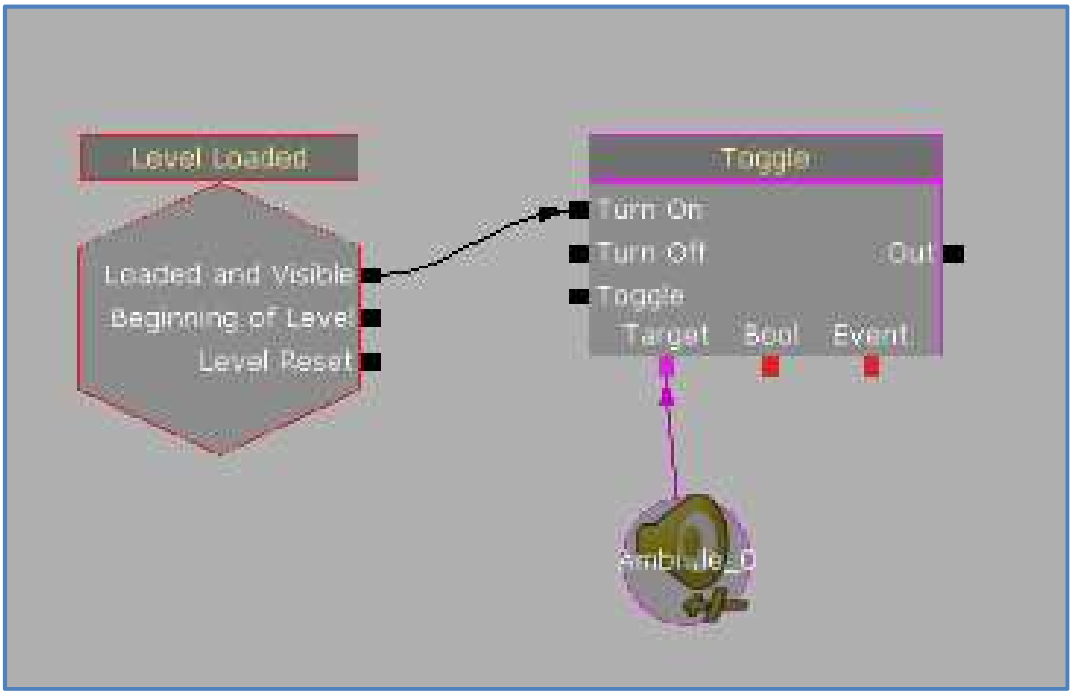


Imagen 478:Reproducción de música al inicio del nivel.

En la siguiente imagen se indica el cambio de un tipo de música a otro:

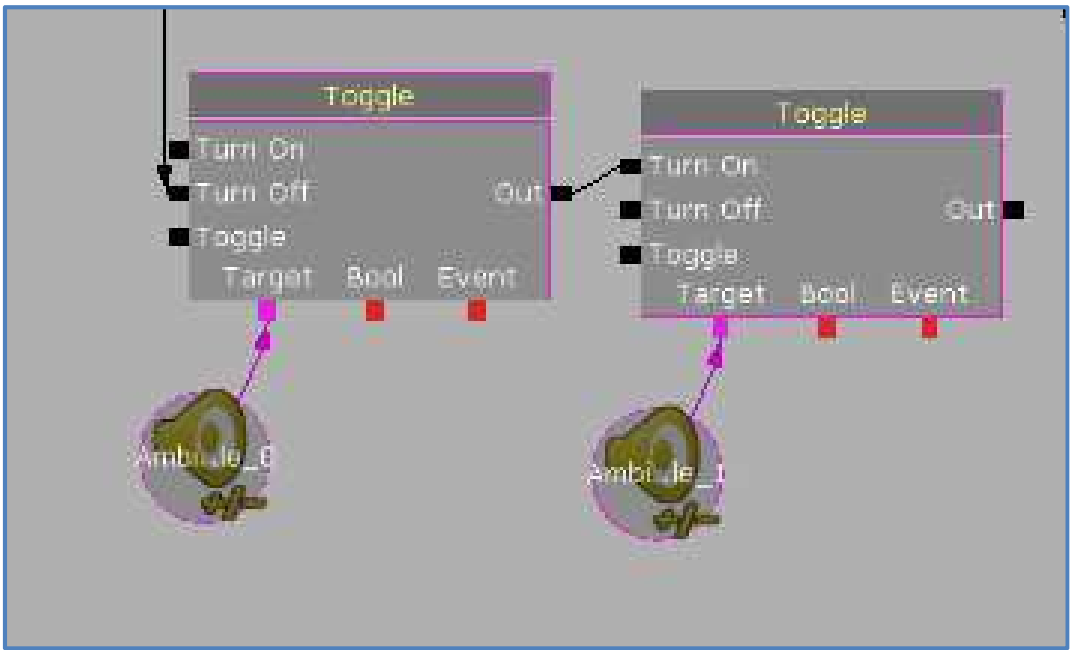


Imagen 479:Modificación de la música.

A su vez, para dotar de mayor realismo al nivel, en cada una de las acciones de apertura de puertas, dañar al protagonista o mostrar un mensaje informativo por pantalla, se reproduce un sonido acorde a la acción.

Un ejemplo de reproducción de sonidos mediante la herramienta Unreal Matinee se muestra en la siguiente imagen:

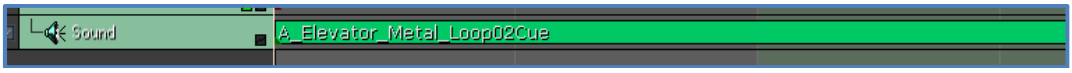


Imagen 480:Reproducción de sonido en "Matinee".

Para la reproducción de los sonidos se ha utilizado tanto la herramienta Unreal Kismet como la herramienta Unreal Matinee (Consultar capítulos 5 y 6).

Teletransporte aleatorio del protagonista.

En el transcurso del juego se realiza un teletransporte del personaje principal a las distintas salas de manera aleatoria. (Consultar capítulo 8.7) En la acción se quiere llegar a una sala concreta, pero puede darse la circunstancia que por la aleatoriedad no se llegue a conseguir. Esta situación no se puede permitir que suceda, con lo que será necesaria una variable contador, llamada *"Contador"*, que en el quinto intento que se haga, el protagonista se teletransporte al lugar deseado.

La secuenca en la implementación sería comprobar el valor del contador y si se ha llegado al máximo del contador, teletransportar a la sala deseada:

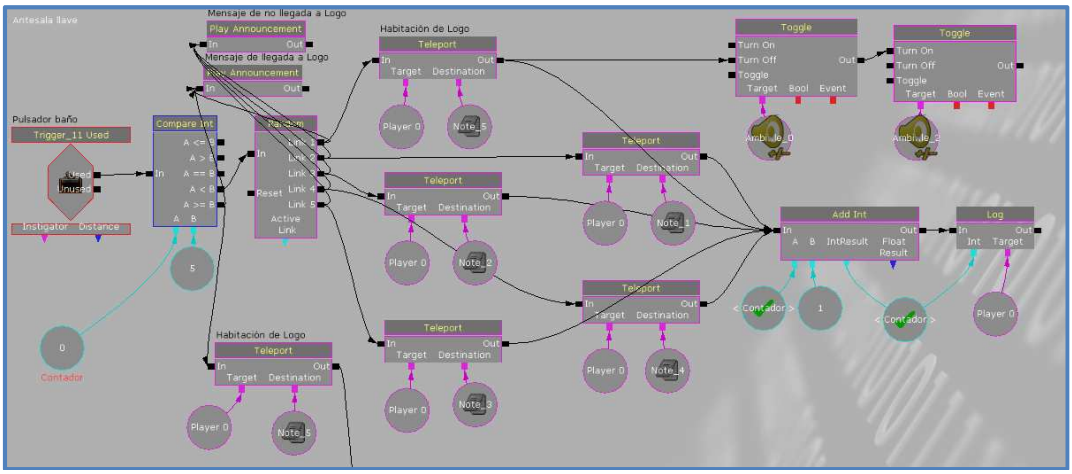


Imagen 481:Implementación de teletransporte aleatorio del protagonista.

Fin del nivel.

Al finalizar este nivel se quiere cargar el siguiente nivel del juego llamado *"CrossPlane_B"*.

La forma de poder cargar un mapa desde otro en ejecución es llamada *"Streaming"*, comentado en el capítulo 8.22.

La implementación en Unreal Kismet de la carga del siguiente nivel sería:

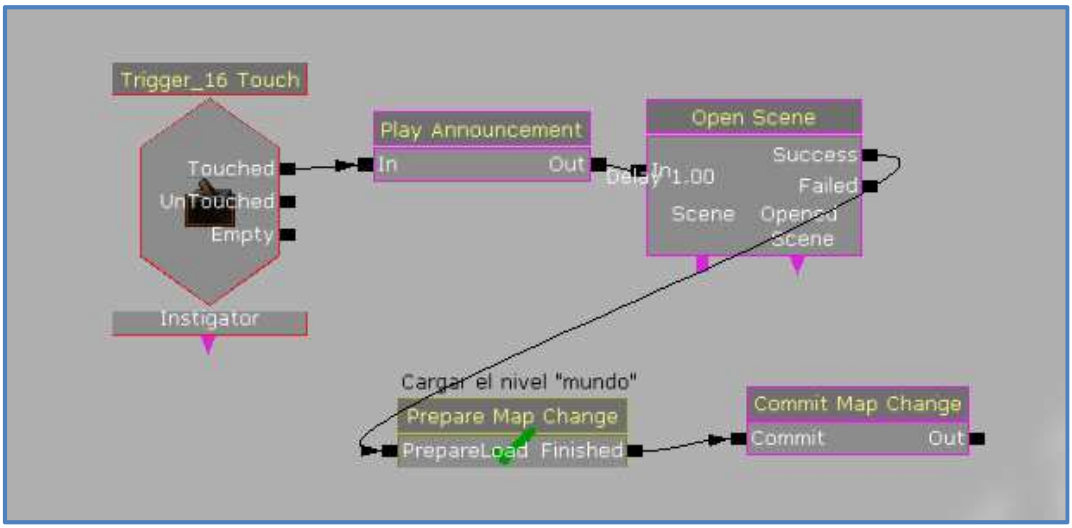


Imagen 482:Fin del nivel.

Dicha carga del nivel es lanzada mediante un trigger de contacto, como se muestra en la imagen 482 de la implementación.

10.2. Nivel exterior.

En este nivel del videojuego se ha utilizado el modo *"additive"* de creación de objetos para realizarlo en exterior. Se va a especificar cada una de las características y decisiones tomadas en la implementación de dicho nivel, tomando como referencia los capítulos anteriores de este documento donde se especifica la forma de realización de cada propiedad.

Cámara en 1ª persona.

En este nivel se ha decidido colocar la cámara en 1ª persona del jugador donde la vista es la imagen a través de los ojos del protagonista. En la siguiente imagen se muestra la imagen en 1ª persona del segundo nivel:

Imagen 483:Cámara en 1ª persona del juego.

Cielo.

Al ser un nivel exterior se ha acoplado un objeto que representará al cielo. La forma de poder acoplar un cielo a un nivel es comentado en el capítulo 8.16.

En la siguiente imagen se muestra el cielo utilizado en el nivel:

Imagen 484:Imagen del cielo.

Sonido.

En el nivel se han acoplado sonidos propios de la naturaleza para mejorar el realismo:

- Sonido de truenos: Como la imagen del cielo elegida asemeja tormentoso se ha añadido la reproducción de sonido de truenos.

  La reproducción de los sonidos se realiza mediante una secuencia de tiempo utilizando los elementos necesarios de la herramienta Unreal Kismet. La implementación sería la mostrada en la imagen:

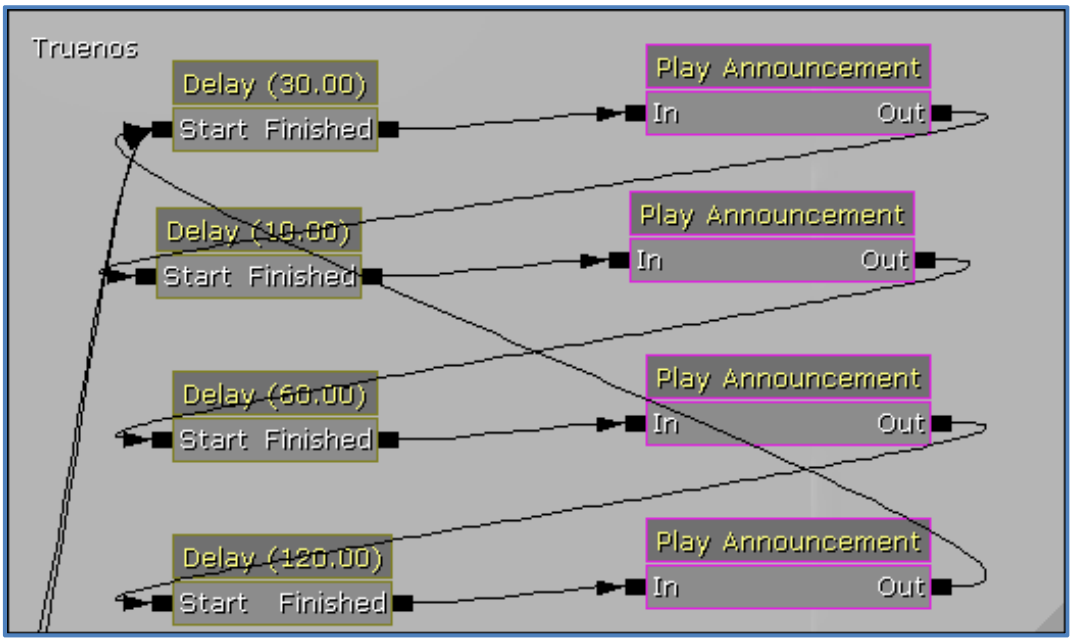


Imagen 485:Implementación de reproducción de truenos.

- Sonido del viento.- Otro sonido que se reproduce es el del viento. Para reproducir este sonido se utiliza uno de los elementos *"AmbientSoundSimple"* proporcionados por el editor (Consultar capítulo 7).

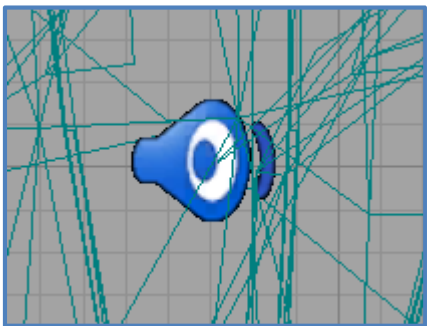


Imagen 486:Elemento de reproducción del sonido del viento.

El radio de acción para oir el sonido será de todo el nivel, ya que el viento se quiere escuchar en todos los puntos del mapa.

- Sonido del agua al caer.- Como en el mundo se ha creado agua y cascadas, se reproduce el sonido que se tendría en la naturaleza. Se utiliza un elemento tipo *"AmbienSound"* que permite indicar el radio de posibilidad de eschucha del sonido.

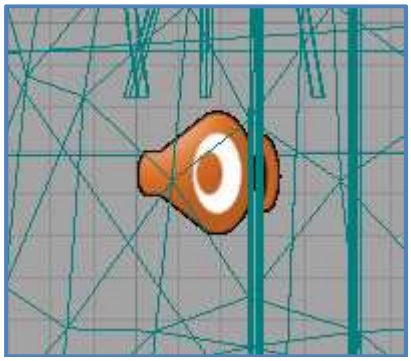


Imagen 487:Elemento de reproducción del sonido del agua.

Se colocará un elemento de este tipo en las inmedicaciones donde se encuentra el *"agua"* y con un radio de acción no muy amplio para que sólo se escuche el sonido cuando el protagonista se vaya acercando.

Viento y lluvia.

Para aportar más realidad al mundo creado se ha acoplado la simulación del viento de la naturaleza y un fenómeno atmosférico como es la lluvia.

El viento es acoplado mediante el elemento proporcionado por el motor el cual simula los efectos de éste. La forma de poder acoplar este elemento se indica en el capítulo 8.14.

Imagen 488:Elemento del viento.

En el nivel se pueden observar los movimientos de los objetos, los árboles y el follage de plantas, simulados como si fuese el viento el que los produjese.

Para poder acoplar la lluvia se ha tenido que crear un nuevo material que simule este fenómeno atmosférico. La creación de este material necesario se muestra en el capítulo 8.19 de este manual.

Imagen 489:Imagen de lluvia en el juego.

En la imagen 489 se muestra la vista de la lluvia simulada en el nivel, mediante el material creado para ello.

Cinemáticas.

Al igual que en nivel anterior, se van a utilizar cinemáticas para guiar al jugador sobre qué es lo debe realizar para ir consiguiendo los objetivos marcados.

Para mostrar las visualizaciones deseadas se utilizan los elementos Matinee proporcionados por la herramienta Unreal Kismet en las que se grabará el movimiento de una cámara y más tarde se visualizará lo captado por dicha cámara. La forma de realización de ello se muestra en el capítulo 6.

Una de estas visualizaciones se muestra justo al comienzo del nivel indicando el lugar al que hay que llegar.

La implementación en Unreal Kismet sería:

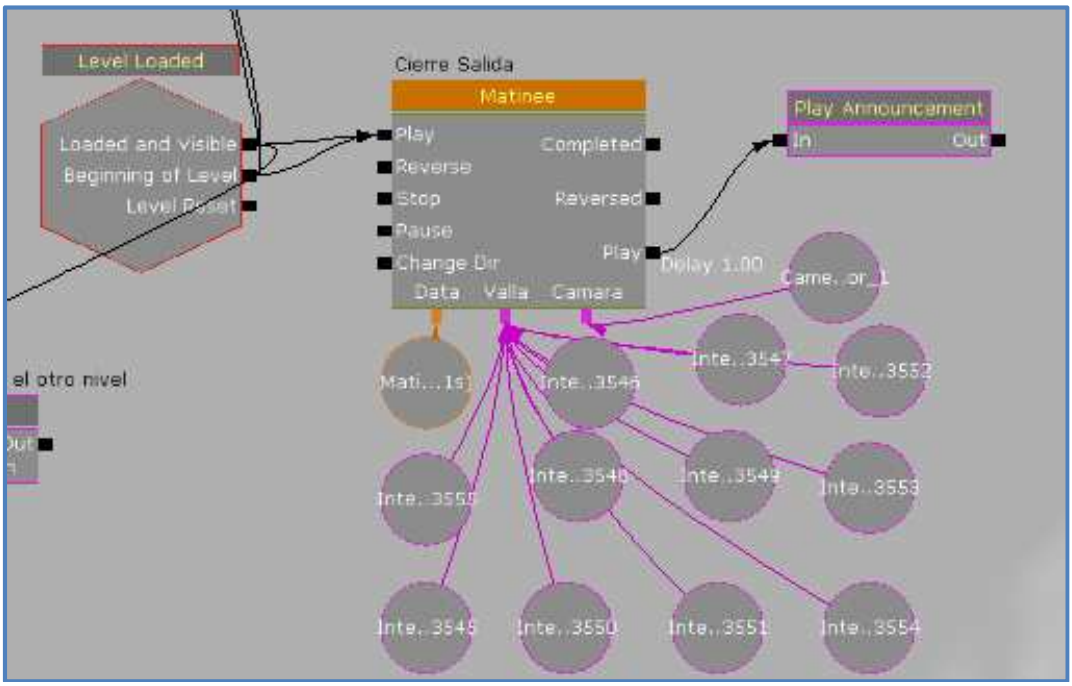


Imagen 490:Implementación de visualización.

Y a su vez la implementación en la herramienta Unreal Matinee del elemento Matinee sería:

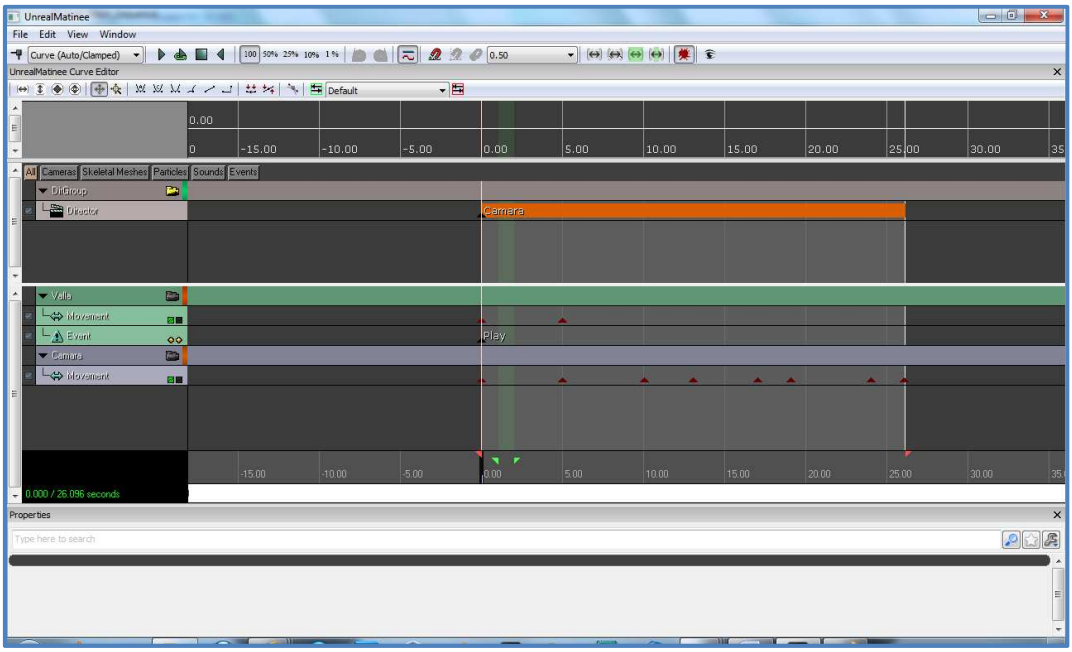


Imagen 491:Implementación de visualización en "Matinee".

Otra cinemática creada es la llegada de las naves enemigas realizando una animación de traslado de los objetos naves en el nivel simulando su vuelo y su aterrizaje.

En la herramienta Unreal Matinee esta implementación sería:

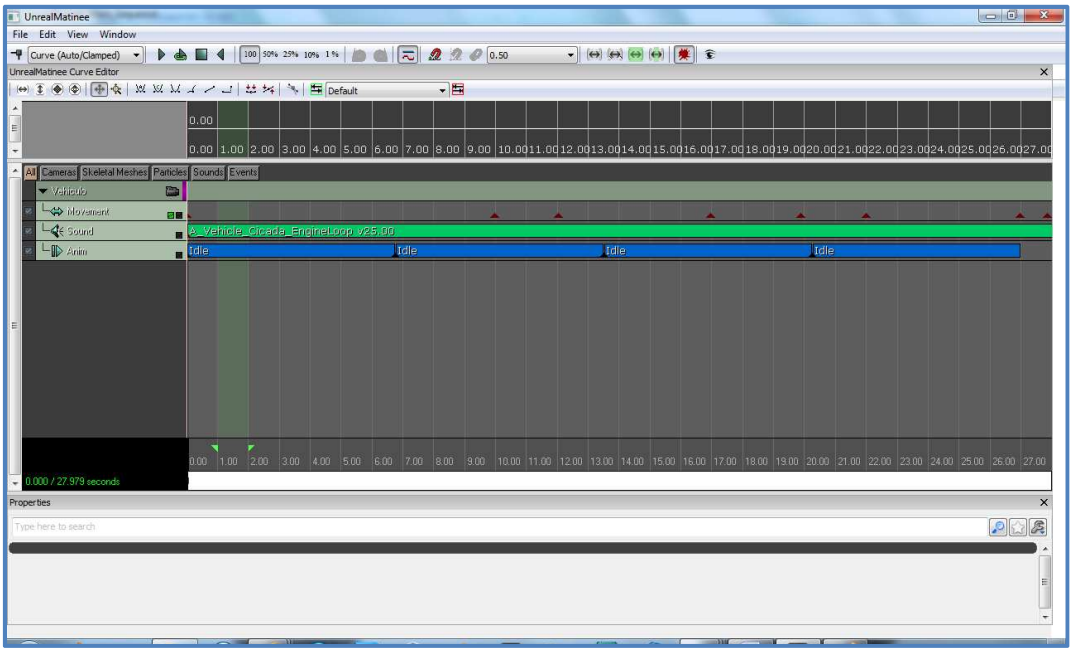


Imagen 492:Ejemplo de visualización de "llegada de enemigos".

Otra animación mostrada en el nivel es el montaje del puente. La implementación de esta acción es semejante a la llegada de enemigos ya que también es de traslado de objetos, pero en este caso, sólo de giro.

Enemigos.

Al igual que se realizó en el primer nivel, en éste se van a añadir ciertos personajes que actuarán como enemigos del protagonista.

En este caso no se va a activar la opción de *"DeathMatch"*, sino que el enemigo empiece a disparar al protagonista una vez que se encuentre a cierta distancia y no haya ningún obstáculo en medio. En el capítulo 8.12 se indica la forma de realizarlo.

La implementación en Unreal Kismet sería:

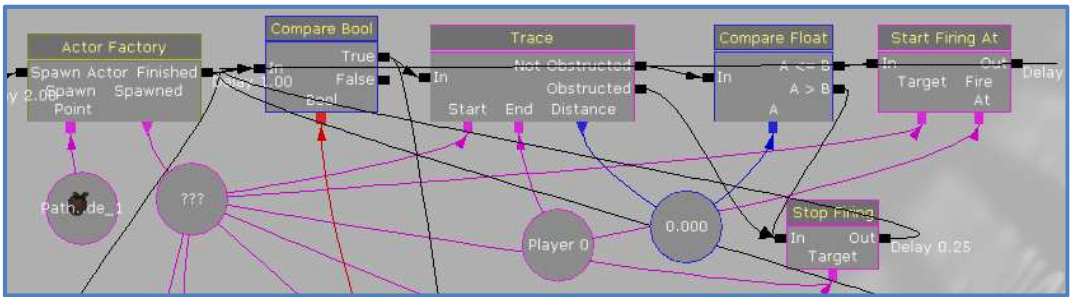


Imagen 493:Comienzo de disparo a una distancia dada.

A su vez, se quiere que los enemigos, nada más aparecer, se trasladen a unos puntos indicados. También algunos de los enemigos deben patrullar entre varios puntos dados moviéndose constantemente. La forma de poder realizar todo esto se indica en el capítulo 8.11 del manual.

La implementación en Unreal Kismet sería:

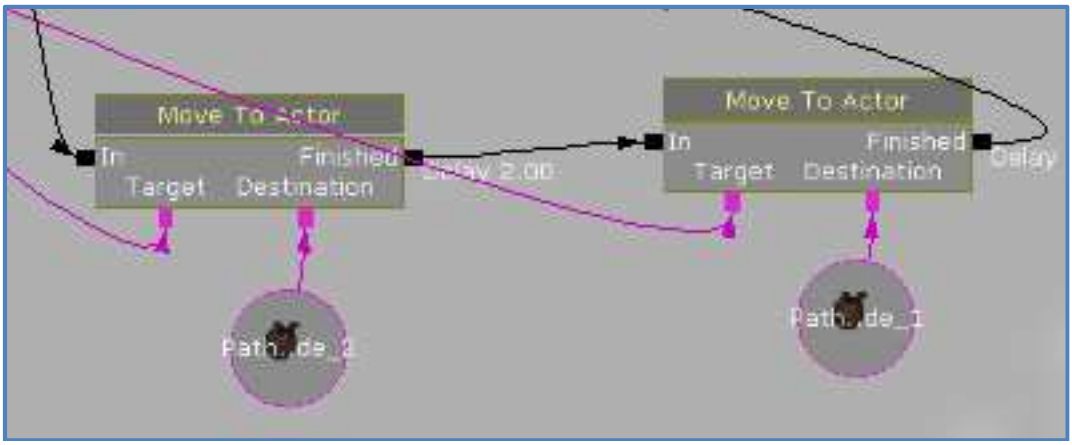


Imagen 494:Movimiento de patrulla de personaje acoplado.

Agua.

En el nivel se ha logrado simular la vista del agua mediante la creación de materiales que han sido acoplados a los objetos. Pero además de simular la vista, se ha acoplado un volumen el cual simule el agua pudiéndose nadar y bucear.

El volumen utilizado es de tipo *"WaterVolume"* y modificando sus propiedades se puede simular que cueste más o menos el desplazanrse en este volumen, nadar, y la velocidad en hundirse el protagonista en ella. Se puede consultar la forma de utilizar y acoplar este tipo de volumen en el capítulo 7 del manual.

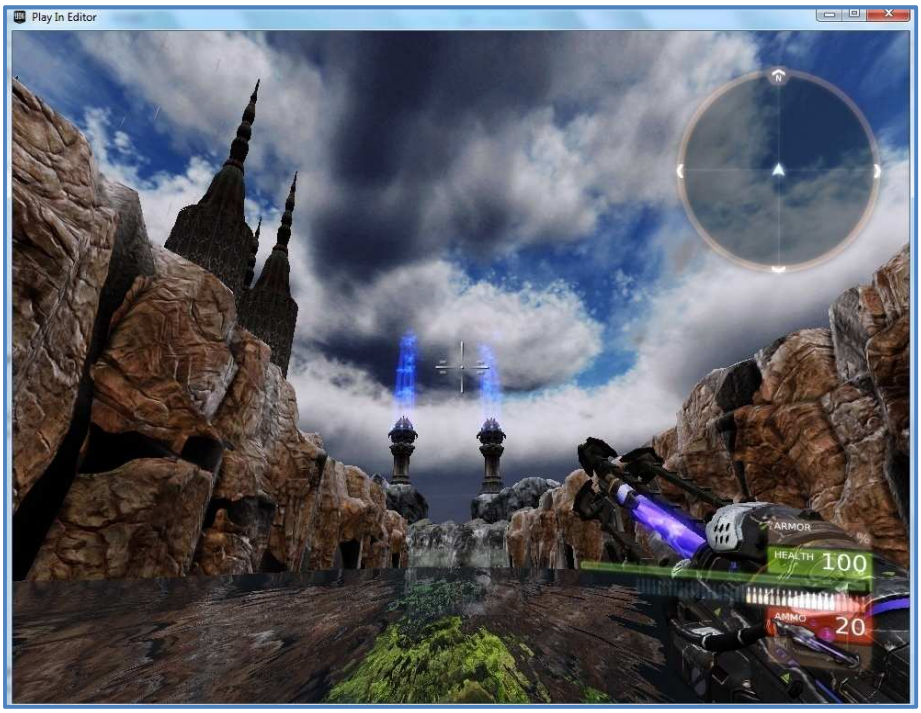


Imagen 495:Imagen del protagonista nadando en el agua.

También se ha querido realizar que al encontrarse el protagonista dentro del agua la imagen se difumine para dotar de mayor realismo. Para ello se ha creado otro volumen

de tipo *"PostProcessVolume"* de dimensiones exactamente igual que el anterior volumen y situado en la misma posición. En el capítulo 7 del manual se muestra cómo poder utilizar este tipo de volumen.

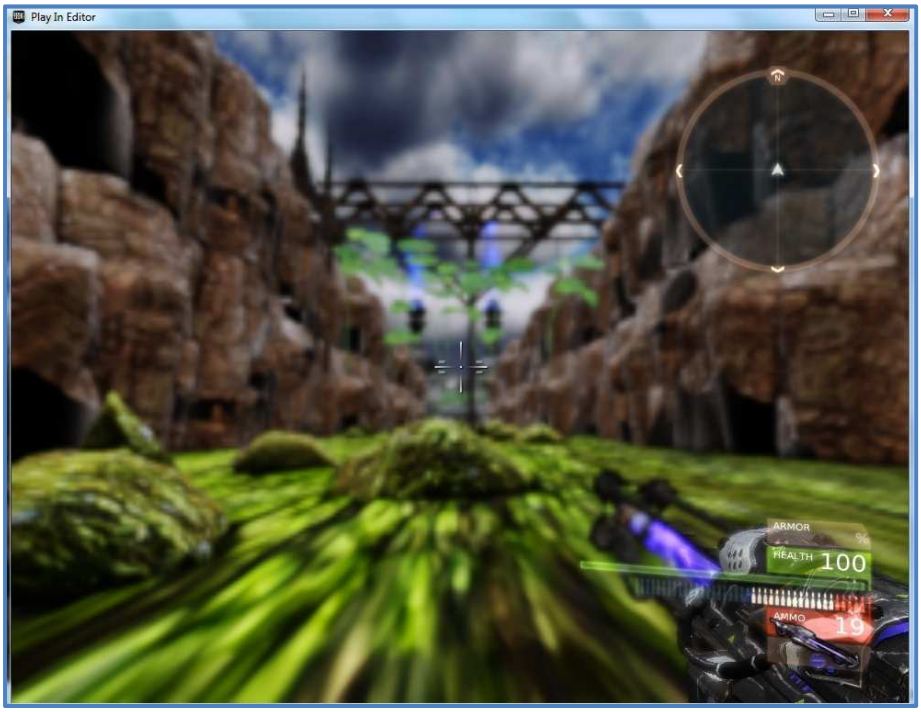


Imagen 496:Distorsión de la imagen al estar el protagonista dentro del agua.

Iluminación.

Al ser un nivel de un mundo exterior, la iluminación va a ser general para todo el nivel, iluminando todo el mundo por completo, simulando la luz del sol.

El motor proporciona un tipo de elemento de luz que representa a esta forma de iluminación. Este elemento es el llamado *"SkyLight"* y para obtener información de cómo utilizar este tipo de elemento consultar el capítulo 7 del manual.

Fin del nivel.

El fin de este nivel representa también el fin del juego. Se deberá ejecutar la orden de finalización de todo el juego. Esta orden es de consola y llamada *"open FrontEnd"*. Deberá ser lanzada desde la ejecución del nivel mediante el elemento *"console command"* proporcionado por la herramienta Unreal Kismet. Para saber cómo realizar la finalización del juego se debe consultar el capítulo 8.23 del manual.

La implementación de esta acción sería:

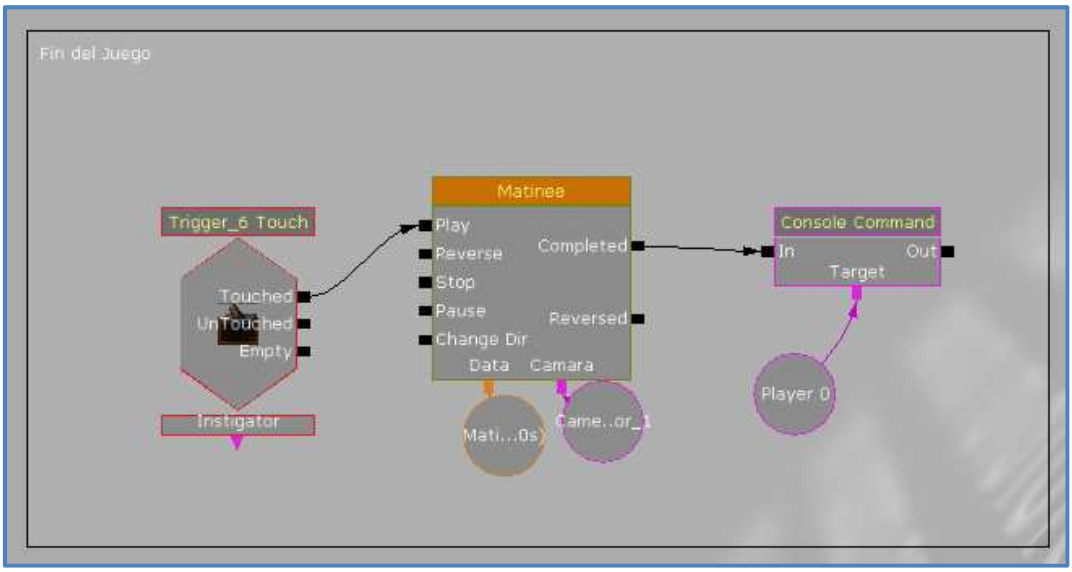


Imagen 497:Fin del nivel.

Y antes de la finalización del juego se muestra una visualización del final con un fundimiento de la imagen a negro. Esto se realiza gracias al elemento Matinee siendo su implementación:

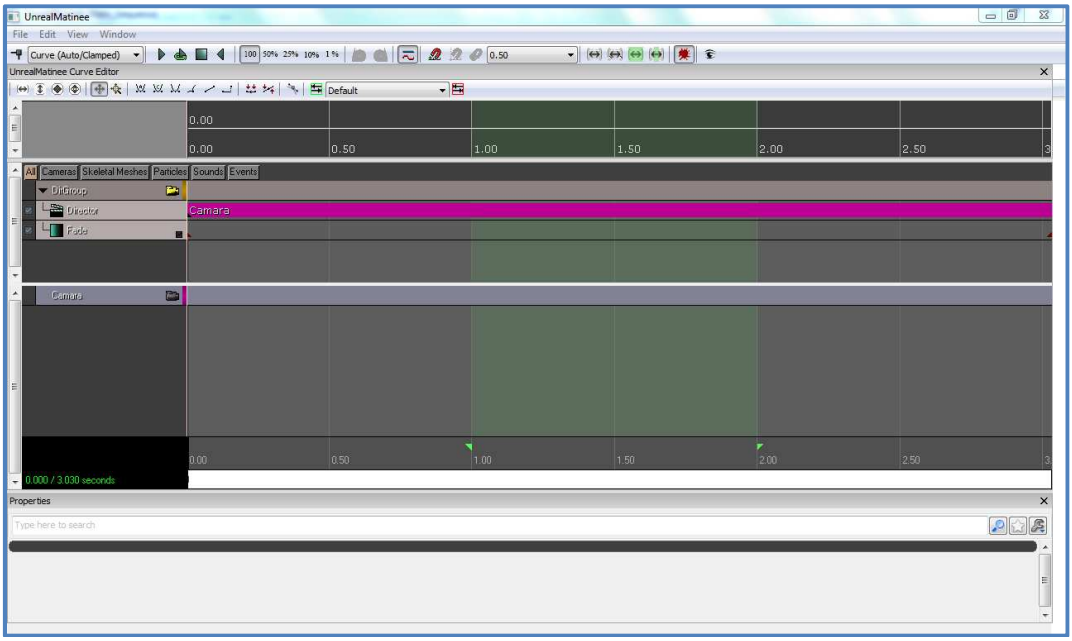


Imagen 498: Fundimiento de la imagen a negro.

11. GESTIÓN DEL PROYECTO.

11.1. Diagramas de Gantt.

En esta sección se va a indicar la planificación del proyecto donde se muestra el orden de las tareas a realizar y la estimación del tiempo necesario para llevarlas a cabo.

Para la presentación de dicha planificación se van a utilizar los diagramas de Gantt los cuales muestran explícitamente la duración de cada actividad y si se está realizando varias tareas a la vez. La creación de estos diagramas se ha realizado mediante la herramienta "*Microsoft Proyect*".

11.1.1. Planificación estimada.

Antes del comienzo del proyecto se hizo una estimación de la planificación para el proyecto mediante un diagrama de Gantt. En él se identificaron las tareas a realizar, el orden de su ejecución y su duración.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Estudio y revisión**	92 días?	vie 26/11/10	vie 01/04/11
2	**Creación de memoria y documentación**	**92 días?**	**jue 25/11/10**	**jue 31/03/11**
3	Estado de la cuestión	15 días?	jue 25/11/10	mié 15/12/10
4	Análisis del problema	29 días?	mar 21/12/10	sáb 29/01/11
5	Solución del problema	25 días?	lun 31/01/11	vie 04/03/11
6	Desarrollo del videojuego	20 días?	lun 07/03/11	jue 31/03/11
7	**Análisis del problema**	**26 días?**	**mar 21/12/10**	**mar 25/01/11**
8	Estilo del videojuego	6 días?	mar 21/12/10	mar 28/12/10
9	División en niveles	4 días?	mié 29/12/10	lun 03/01/11
10	Nuevo personaje	4 días?	mar 04/01/11	vie 07/01/11
11	Guión del videojuego	5 días?	vie 07/01/11	jue 13/01/11
12	Game Play	7 días?	jue 13/01/11	vie 21/01/11
13	Story Board	13 días?	vie 07/01/11	mar 25/01/11
14	**Diseño del problema**	**24 días?**	**jue 27/01/11**	**mar 01/03/11**
15	Historia	7 días?	jue 27/01/11	vie 04/02/11
16	Arte Conceptual	4 días?	vie 04/02/11	mié 09/02/11
17	Sonido	2 días?	jue 10/02/11	dom 13/02/11
18	Mecánica de juego	5 días?	lun 14/02/11	dom 20/02/11
19	Diseño de programación	7 días?	lun 21/02/11	mar 01/03/11
20	**Implementación del prototipo**	**17 días?**	**mié 02/03/11**	**jue 24/03/11**
21	Programación	11 días?	mié 02/03/11	mié 16/03/11
22	Ilustración	6 días?	lun 07/03/11	lun 14/03/11
23	Interfaz	6 días?	mar 15/03/11	mar 22/03/11
24	Modelado y animación 3D	10 días?	vie 11/03/11	jue 24/03/11
25	Diseño de sonido	3 días?	lun 21/03/11	mié 23/03/11
26	**Pruebas del prototipo**	**6 días?**	**sáb 26/03/11**	**vie 01/04/11**
27	Pruebas alpha	5 días?	lun 28/03/11	vie 01/04/11
28	Pruebas beta	6 días?	sáb 26/03/11	vie 01/04/11

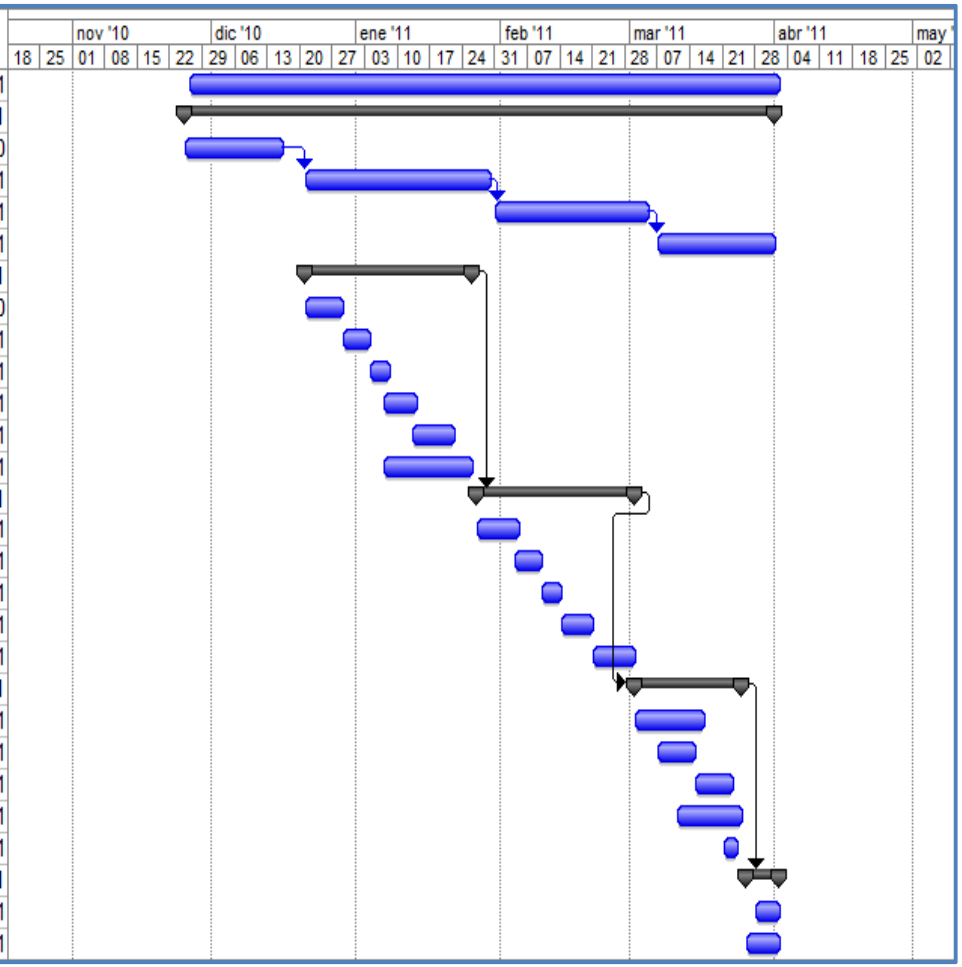


Imagen 499:Diagrama de Gantt – Planificación estimada.

Las tareas identificadas se agrupan en las fases lógicas de desarrollo de un proyecto:

- Análisis del problema.
  Desglose de los sub-problemas encontrados a la hora de la creación de un videojuego shoot'em up en 3D, estudiando la forma de resolver todos ellos mediante el motor UDK.
- Diseño de solución.
  Diseño que soluciona el problema mediante el motor UDK. Elementos básicos a desarrollar en el manual para solucionar el problema.
- Implementación.
  Esta fase se divide en dos subgrupos:
  - Creación del manual indicando la forma de realizar distintas acciones de un juego en 3D en el motor UDK.
  - Las mismas acciones desarrolladas en el manual implementarlas en el propio juego creado.
- Documentación.
  Tareas de documentación de cada uno de los pasos lógicos de desarrollo del proyecto.

11.1.2. Planificación real.

Durante la realización de todas las fases del proyecto y con el seguimiento de ellas, se fue modificando el diagrama de Gantt estimado con las duraciones reales de cada una de las tareas.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Estudio y revisión**	137 días?	lun 29/11/10	vie 03/06/11
2	**Creación de memoria y documentación**	**137 días?**	**lun 29/11/10**	**vie 03/06/11**
3	Estado de la cuestión	17 días?	lun 29/11/10	mar 21/12/10
4	Análisis del problema	47 días?	mié 22/12/10	mié 23/02/11
5	Solución del problema	33 días?	jue 24/02/11	vie 08/04/11
6	Desarrollo del videojuego	40 días?	lun 11/04/11	vie 03/06/11
7	**Análisis del problema**	**48 días?**	**mar 21/12/10**	**mié 23/02/11**
8	Estilo del videojuego	8 días?	mar 21/12/10	jue 30/12/10
9	División en niveles	5 días?	dom 02/01/11	jue 06/01/11
10	Nuevo personaje	6 días?	jue 06/01/11	jue 13/01/11
11	Guión del videojuego	6 días?	vie 14/01/11	sáb 22/01/11
12	Game Play	9 días?	lun 24/01/11	jue 03/02/11
13	Story Board	14 días?	vie 04/02/11	mié 23/02/11
14	**Diseño del problema**	**33 días?**	**jue 24/02/11**	**vie 08/04/11**
15	Historia	10 días?	jue 24/02/11	mié 09/03/11
16	Arte Conceptual	5 días?	mié 09/03/11	mar 15/03/11
17	Sonido	3 días?	mié 16/03/11	sáb 19/03/11
18	Mecánica de juego	10 días?	mar 15/03/11	dom 27/03/11
19	Diseño de programación	10 días?	lun 28/03/11	vie 08/04/11
20	**Implementación del prototipo**	**34 días?**	**lun 11/04/11**	**jue 26/05/11**
21	Programación	18 días?	lun 11/04/11	mié 04/05/11
22	Ilustración	7 días?	vie 22/04/11	lun 02/05/11
23	Interfaz	6 días?	jue 05/05/11	jue 12/05/11
24	Modelado y animación 3D	9 días?	vie 13/05/11	mié 25/05/11
25	Diseño de sonido	5 días?	vie 20/05/11	jue 26/05/11
26	**Pruebas del prototipo**	**6 días?**	**vie 27/05/11**	**vie 03/06/11**
27	Pruebas alpha	5 días?	vie 27/05/11	jue 02/06/11
28	Pruebas beta	6 días?	vie 27/05/11	vie 03/06/11

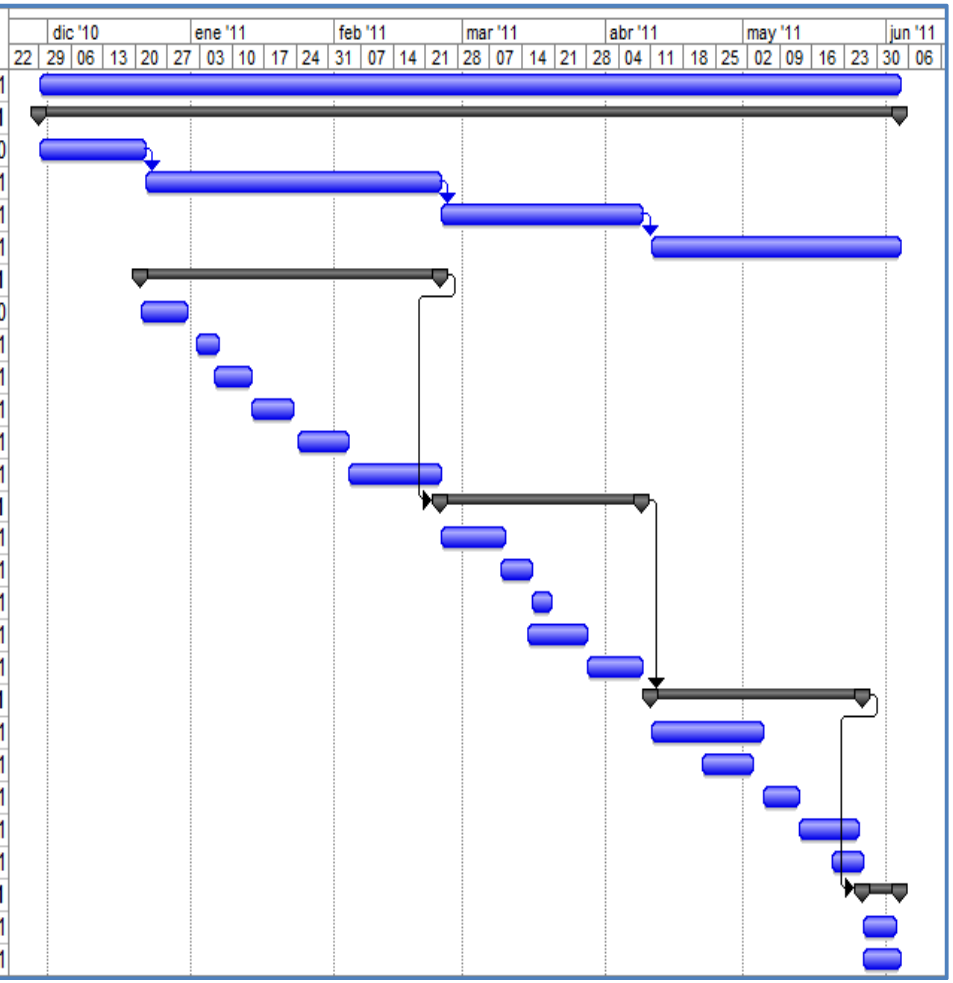


Imagen 500:Diagrama de Gantt – Planificación real.

11.1.3. Análisis de planificación.

Como se puede observar, la duración de la planificación real, 137 días, es mucho mayor a la calculada en la planificación estimada, 92 días, produciéndose una desviación de más o menos 2 meses.

Los motivos principales de esta desviación son los siguientes:

- **Evolución del manual y del caso práctico**: A la hora del desarrollo del caso práctico, el propio juego, surgieron complicaciones a resolver que a su vez tenían que ser plasmadas en el manual creado.
- **Documentación consultada**: Para la realización de un manual completo, el número de documentación consultada ha sido mucho mayor al estimado, para poder encontrar distintos puntos de vista y seleccionar la información correcta y más completa.

11.2. Presupuesto.

El presente presupuesto hace referencia a un proyecto de desarrollo de un videojuego mediante la herramienta Unreal Development Kit. Se van a especificar todos los costes asumidos al proyecto, como son el coste del personal integrante del equipo del proyecto, los equipos informáticos necesarios tanto para el desarrollo del juego como de la documentación y demás costes directos.

A priori el resultado del presupuesto puede resultar excesivo, no obstante se tendrá en cuenta que la ejecución del juego no se ha realizado con vistas a una posible comercialización que diera lugar a su amortizacíon y rentabilización, se hace única y exclusivamente bajo el pedido de un cliente, en este caso el Departamento de Informática de la Universidad Carlos III de Madrid. Se trata, por lo tanto, de un videojuego original realizado a modo de ejemplo de utilización del motor Unreal Development Kit.

AUTOR.

Julián Marchante de la Dueña.

DEPARTAMENTO.

Departamento de informática.

DESCRIPCIÓN DEL PROYECTO.

TÍTULO: Manual para la creación de videojuegos mediante el motor Unreal Development Kit.

DURACIÓN: 6 meses.

TASA DE COSTES INDIRECTOS: 20%.

PRESUPUESTO DETALLADO DEL PROYECTO.

Personal:

En la tabla 4 se muestra el personal involucrado en el proyecto indicando el coste de cada uno de ellos:

Apellidos y nombre	Categoría	Dedicación (hombres mes)	Coste hombre mes	Coste (Euro)
Juan Peralta Donate	Ingeniero Senior	1	4.289,54	4.289,54
Julián Marchante de la Dueña	Ingeniero	6	2.694,39	16.166,34
			Total	**20.455,88**

Tabla 4.Presupuesto detallado de personal.

Equipos:

En la siguiente tabla se muestran los equipos necesarios en el desarrollo del proyecto junto con el coste de cada uno de ellos:

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
HP Pavilion SlimLine	1.150,00	100	6	60	115,00
HP Pavilion tx2620es	1.099,00	100	6	60	109,90
Disco duro externo	100,00	100	6	60	10,00
Multifunción Escaner + Impresora	300,00	100	6	60	30,00
				Total	**264,90**

Tabla 5.Presupuesto detallado de equipos.

Los equipos detallados presentados en el presupuesto son:

- **HP Pavilion tx2620es**.- Ordenador para la creación de la documentación del proyecto:
  - Windows® 7 Professional original 64 bits.
  - Procesador i3-380UM Intel® Core™ 1,33 GHz, 3 MB de caché de nivel 3.
  - Chipset HM55 Intel®.
  - DDR3 de 4 GB (1 x 4.096 MB).
  - 2 ranuras de memoria accesible de usuario.
  - Pantalla HP de alta definición BrightView convertible táctil de 30,7 cm (12,1") en diagonal.
  - Resolución de pantal1a .280 x 800.
  - ATI Mobility Radeon™ HD 5450 Graphics (conmutable), hasta 1696 MB de memoria gráfica total con 512 MB DDR3 de memoria dedicada.
  - Lector de soportes digitales integrado "5 en 1" para tarjetas Secure Digital, tarjetas MultiMedia, Memory Stick, Memory Stick Pro o tarjetas xD Picture.
  - 1 puerto VGA; 1 puerto HDMI; 1 salida estéreo de auriculares; 1 conector combo de salida de auriculares / entrada de micrófono; 3 puertos USB 2.0; 1 puerto Ethernet RJ45.
  - Webcam HP TrueVision con micrófono digital integrado (VGA de luz baja).
  - Altavoces Altec Lansing - Dolby Advanced Audio.
  - Teclado en tamaño completo.
  - LAN Ethernet Gigabit 10/100/1000 integrada.
  - 802.11 b/g/n.
  - Adaptador de corriente de 65 W.
  - Batería de ion de litio (Li-Ion) de 6 celdas.
  - Lector de huellas digitales integrado.

- **HP Pavilion Slimline**.- Equipo necesario para el desarrollo del videojuego:
  - Monitor TFT de 19" teclado y raton óptico.
  - Procesador intel core 2 duo E4600 a 2.4 Ghz.
  - Memoria RAM de 2GB. DDR II.
  - Disco duro de 360 GB. Serial ATA 7200 RPM.
  - Tarjeta gráfica ATI Radeon HD 3450 hasta 250 MB.
  - Grabadora de DVD de doble capa Lightscribe.
  - Sistema operativo Microsoft Windows 7 Professional 64 bits.
  - Tarjeta de red 10 / 100 integrada.
  - Lector de tarjetas 15 en 1: admite, entre otras, Secure Digital (SD), Compact Flash (CF), Smart Media (SM), Multimedia Card (MMC), Sony Memory Stick y Microdrive.
  - 6 tomas USB (2 frontales).
- **Disco duro externo**.- Disco duro externo con conexión USB para realizar copias de seguridad del proyecto.
  - Modelo: FreeCom.
  - Capacidad: 320 Gb de almacenamiento de datos.
- **Multifunción impresora + escáner**.- Utilizada para la impresión y escáner de la documentación necesaria para el proyecto.
  - Modelo: HP photosmart C4180.

Otros costes directos:

Los costes directos achacados al desarrollo del proyecto se presentan en la siguiente tabla:

Descripción	Empresa	Costes imputable
Fungible	Matedi	200,00
Viajes	-	100,00
2x Licencia Windows 7 Professional 64 bits	Microsoft Store	599,98
Licencia Microsoft Office 2007 Basic	Microsoft Store	149,99
	Total	**1.049,97**

Tabla 6.Presupuesto detallado de costes directos.

- **Fungibles.-** Material de oficina utilizado en el desarrollo del proyecto fin de carrera y el presente documento.
- **Viajes.-** Coste de desplazamiento para la realización de las reuniones de seguimiento.
- **Licencias.-** Licencias de software necesarias para los equipos utilizados en el proyecto.

PRESUPUESTO TOTAL DEL PROYECTO.

Una vez presentados todos los costes asociacos al proyecto, en la siguiente tabla se presentan todos ellos a modo de resumen, indicando el coste total:

Presupuesto Costes Totales	
Personal	20.456
Amortización	265
Subcontratación de tareas	0
Costes de funcionamiento	1.050
Costes Indirectos	4.354
Total	**26.125**

Tabla 7.Presupuesto total del proyecto.

11.3. Comercialización del proyecto.

En el caso de querer comercializar un videojuego creado con el motor Unreal Development Kit (UDK) se tendrían varios puntos de posible rentabilidad:

- **Niveles exclusivos**: Se pueden comercializar niveles exclusivos para un juego creado con nuevas misiones, personajes, etc., ampliando las propiedades y características del juego.
- **Elementos exclusivos**: Algunos elementos proporcionados en el juego serían accesibles solamente mediante la compra de claves, que introduciéndolas en el juego ya se podría hacer uso de estos elementos. Los elementos exclusivos podrían ser nuevas armas mejoradas, personajes exclusivos, vehículos nuevos, vestimenta diferente, etc.
- **Mejoras de las propiedades del juego**: Comercialización de actualizaciones del juego donde se ampliarían las propiedades de éste, como ampliación del nivel de vida del jugador principal, dotar a las armas de mayor poder, ampliar el número de proyectiles de cada arma antes de recarga, etc.

12. CONCLUSIONES.

Una vez finalizado el proyecto fin de carrera se puede llegar a la conclusión que el motor de *"Unreal"* es una herramienta muy potente y completa para poder desarrollar videojuegos.

La aparción de los motores de videojuegos facilitó en gran medida el desarrollo de éstos, proporcionando un gran conjunto de herramientas que permiten desarrollar y crear todos los elementos y etapas necesarios de un videojuego.

Unreal es uno de estos motores potentes y completos que permiten realizar un videojuego de una forma muy rápida y cómoda. Para la realización de este manual se ha utilizado la versión gratuita de este motor, *"Unreal Development Kit"*. Al ser una versión gratuita se puede pensar que no será una herramienta completa, limitando mucho la posibilidad de utilizar ciertas herramientas, funcionalidades, etc. Pero en este caso, aún siendo una versión gratuita, sigue siendo una herramienta completa proporcionando una gran cantidad de herramientas para el desarrollo de un videojuego.

Una ventaja muy importante de este motor es que no es necesario tener conocimientos de informática ni de programación para su utilización. Su interfaz es tan intuitiva y sencilla que es muy fácil utilizar todas sus herramientas y relacionarlas entre sí. Cualquier persona podría crearse sus propios niveles de su videojuego de una manera rápida y sencilla gracias a este motor.

Otra ventaja que tiene este motor, es que su versión gratuita no es cerrada, cada cierto tiempo van apareciendo nuevas versiones con actualizaciones implementadas de mejora del motor. Un ejemplo de estas nuevas actualizaciones recientes es el poder compilar el videojuego para otras plataformas, como Iphone, ya que anteriormente el motor estaba desarrollado únicamente para desarrollar juegos para PC.

En resumen, Unreal Development Kit permite a todo tipo de público crear sus propios juegos de una manera casi profesional, ya que la potencia ofrecida por el motor es muy grande proporcionando herramientas muy completas.

13. LÍNEAS FUTURAS.

En este capítulo se van a presentar las líneas futuras de desarrollo de videojuegos mediante el motor Unreal Development Kit.

En el manual creado para el proyecto fin de carrera se indica la forma de desarrollar un mapa ó nivel de un videojuego utilizando todas las herramientas proporcionadas por el motor. Este mapa se insertará a toda la infraestructura de funciones de inicio y finalización de juego completo proporcionada por el motor.

Otra forma de creación de videojuegos mediante el motor sería la denominada *"creación desde cero"*. Con esta forma habría que implementar todo lo referente al juego, tanto las pantallas de inicio, menú se selección de modo de juego, menú de selección de mapas, pantallas de finalización, etc.

Mediante esta forma de desarrollo se tiene mucha más libertad de implementación, ya que todo lo que se pueda implementar se podrá realizar en el juego. El utilizar las herramientas del motor es más limitado ya que sólo se puede hacer todo aquello que la herramienta permite y proporciona.

Pero esta forma de desarrollo de videojuegos tiene un requisito indispensable, tener conocimientos de programación. Es necesario tener conocimientos de programación ya que todo lo que se quiera desarrollar es mediante implementación. El motor UDK posee un lenguaje propio llamado *"Unreal Script"* orientado a objetos. Mediante esta implementación pueden crearse videojuegos inéditos y propios con todos los elementos creados por el desarrollador.

Para esta forma de desarrollo, se podría utilizar un entorno de desarrollo. Para la aplicación "Visual Studio" puede utilizarse un módulo para poder utilizar el lenguaje propio de UDK, *Unreal Script*. Este módulo es llamado *"nFringe"* [58] mediante el cual es reconocido este lenguaje permitiendo implementar con él.

Puede encontrarse el módulo *nFringe* para su descarga en la página:

http://wiki.pixelminegames.com/index.php?title=Tools:nFringe:Releases

Otra línea de futura sería el aumentar y/o mejorar la Inteligencia Artificial de los personajes enemigos del principal. Se podría implementar un método de apredizaje automático de los mismos para ir modificando la actitud respecto a las acciones del personaje principal.

14. REFERENCIAS.

[1] Versión gratuita del motor de videojuegos Unreal. Disponible en World Wide Web: http://www.udk.com. Último acceso: 20 / 05 / 2011.

[2] Disponible en World Wide Web: http://en.wikipedia.org/wiki/Game_engine. Último acceso: 06 / 05 / 2011.

[3] Disponible en World Wide Web: http://www.niubie.com/2011/01/que-es-un-engine-para-videojuegos/. Último acceso: 10 / 05 / 2011.

[4] Componente que permite realizar una simulación de físicas y detecciones de colisiones. Disponible en World Wide Web: http://www.havok.com. Último acceso: 15 / 05 / 2011.

[5] Sistema software para la reproducción de sonido. Disponible en World Wide Web: http://www.radgametools.com/miles.htm. Último acceso: 15 / 05 / 2011.

[6] Códec de video para videojuegos. Disponible en World Wide Web: http://www.radgametools.com/bnkmain.htm. Último acceso: 29 / 04 / 2011.

[7] Motor de videojuegos desarrollado por “Criterion Software”.

[8] Disponible en World Wide Web: http://es.wikipedia.org/wiki/Motor_de_videojuego. Último acceso: 15 / 05 / 2011.

[9] Consola de videojuegos lanzada en octubre de 1977. Disponible en World Wide Web: http://www.atari2600.com/. Último acceso: 06 / 05 / 2011.

[10] Compañía de software. Disponible en World Wide Web: http://www.argonautsoftware.com/. Último acceso: 06 / 05 / 2011.

[11] Compañía de software. Disponible en World Wide Web: http://www.realitylab.at/wp/home/. Último acceso: 06 / 05 / 2011.

[12] Biblioteca para la programación de gráficos 3D.

[13] Empresa estadounidense desarrolladora y distribuidora de videojuegos fundada por Trip Hawkings. Disponible en World Wide Web: http://www.ea.com/es/1/index. Último acceso: 08 / 05 / 2011.

[14] Disponible en World Wide Web: http://www.chw.net/2006/02/que-es-un-game-engine/. Último acceso: 16 / 05 / 2011.

[15] Disponible en World Wide Web: http://es.wikipedia.org/wiki/Motor_de_videojuego. Último acceso: 16 / 05 / 2011.

[16] Disponible en World Wide Web: http://gpwiki.org/index.php/Game_Engines. Último acceso: 17 / 05 / 2011.

[17] Disponible en World Wide Web: http://www.gamemiddleware.org/. Último acceso: 11 / 05 / 2011.

[18] Disponible en World Wide Web: http://www.vorpaline.com/2011/05/14/las-cgi-en-los-videojuegos-actuales/. Último acceso: 12 / 05 / 2011.

[19] Disponible en World Wide Web: http://www.gamecareerguide.com/features/529/what_is_a_game_.php?page=2. Último acceso: 17 / 05 / 2011.

[20] Disponible en World Wide Web: http://en.wikipedia.org/wiki/Game_engine. Último acceso: 17 / 05 / 2011.

[21] Empresa desarrolladora de videojuegos establecida en Estados Unidos. Disponible en World Wide Web: http://www.epicgames.com/. Último acceso: 06 / 05 / 2011.

[22] Disponible en World Wide Web: http://en.wikipedia.org/wiki/Unreal_Engine. Último acceso: 21 / 05 / 2011.

[23] Disponible en World Wide Web: http://www.unrealengine.com/. Último acceso: 21 / 05 / 2011.

[24] Disponible en World Wide Web: http://www.unrealargentina.com/smf/index.php?topic=18657.0. Último acceso: 21 / 05 / 2011.

[25] Libro: “Mastering Unreal Technology, Volume I: Introduction to Level Design with Unreal Engine 3”.

[26] API utilizada para gráficos 3D. Disponible en World Wide Web: http://en.wikipedia.org/wiki/Glide_API. Último acceso: 22 / 05 / 2011.

[27] Disponible en World Wide Web:

http://udn.epicgames.com/Main/WebHome.html. Último acceso: 26 / 05 / 2011.

[28] Game Developers Conference del año 2009.

[29] *End User License Agreement*. Licencia por la cual el uso de un producto sólo está permitido para un único usuario (el comprador).

[30] Wiki del motor Unreal. Disponible en World Wide Web: http://wiki.beyondunreal.com/. Último acceso: 26 / 05 / 2011.

[31] Libro: "Mastering Unreal Technology, Volume II: Advanced Level Design Concepts with Unreal Engine 3".

[32] Foro sobre el motor Unreal Development Kit. Disponible en World Wide Web: http://unreales.forumcommunity.net/. Último acceso: 14 / 05 / 2011.

[33] Foro sobre el motor Unreal Development Kit. Disponible en World Wide Web: http://unreales.tk/. Último acceso: 14 / 05 / 2011.

[34] Disponible en World Wide Web: http://www.3dbuzz.com/vbforum/sv_videonav.php?fid=56797c0cd321ddc111e9059535cb3f24. Último acceso: 13 / 05 / 2011.

[35] Disponible en World Wide Web: http://www.solovagos.com/foro/programacion/451647-ogasoda-udk-video-tutorials-building-3d-world-set-1-a.html. Último acceso: 13 / 05 / 2011.

[36] Disponible en World Wide Web: http://eduardo777.wordpress.com/2009/11/12/videotutorial-comenzando-con-udk-en-espanol/. Último acceso: 13 / 05 / 2011.

[37] Disponible en World Wide Web: http://www.taringa.net/posts/videos/10711353/Videotutoriales-UDK-en-espanol.html. Último acceso: 14 / 05 / 2011.

[38] Disponible en World Wide Web: http://www.youtube.com/watch?feature=player_embedded&v=OcJ3J2Yf_Wg. Último acceso: 14 / 05 / 2011.

[39] Disponible en World Wide Web: http://www.youtube.com/results?search_query=UDK&aq=f. Último acceso: 14 / 05 / 2011.

[40] Disponible en World Wide Web: http://www.youtube.com/results?search_query=udk+tutorial+esp

a%C3%B1ol&aq=0&oq=UDK+tutorial. Último acceso: 15 / 05 / 2011.

[41] Disponible en World Wide Web: http://www.youtube.com/results?search_query=udk+tutorial&aq=f. Último acceso: 15 / 05 / 2011.

[42] Foro dedicado al motor Unreal Development Kit. Disponible en World Wide Web: http://forums.epicgames.com/. Último acceso: 15 / 05 / 2011.

[43] Foro dedicado al motor Unreal Development Kit. Disponible en World Wide Web: http://www.hl2spain.com/foro/index.php?showtopic=49112. Último acceso: 15 / 05 / 2011.

[44] Disponible en World Wide Web: http://www.ciroduran.com/inventad/foros/topic/udk. Último acceso: 17 / 05 / 2011.

[45] Disponible en World Wide Web: http://desarrolladoresdevideojuegos.es/2011/05/02/10-enlaces-recomendados-de-tutoriales-para-udk/. Último acceso: 17 / 05 / 2011.

[46] Disponible en World Wide Web: http://forums.epicgames.com/showthread.php?t=707152. Último acceso: 17 / 05 / 2011.

[47] Disponible en World Wide Web: http://wiki.beyondunreal.com/External_UDK_tutorials. Último acceso: 17 / 05 / 2011.

[48] Disponible en World Wide Web: http://hackerpedia.wordpress.com/2011/02/01/tutorial-de-udk-parte-1/. Último acceso: 18 / 05 / 2011.

[49] Disponible en World Wide Web: http://www.hourences.com/tutorials/. Último acceso: 18 / 05 / 2011.

[50] Disponible en World Wide Web: http://udk-tuts.blogspot.com/. Último acceso: 18 / 05 / 2011.

[51] Pack de utilidades para convertir videos a formato Bink. Disponible en World Wide Web: http://rad-video-tools.softonic.com/. Último acceso: 18 / 05 / 2011.

[52] Tutorial sobre el uso de la herramienta 3D Studio Max. Disponible en World Wide Web: Bipped rigging, Part 1. Último

acceso: 18 / 05 / 2011.

[53] Tutorial sobre el uso de la herramienta 3D Studio Max. Disponible en World Wide Web: Bipped rigging, Part 2. Último acceso: 18 / 05 / 2011.

[54] Programa de creación de gráficos y animación 3D desarrollado por la empresa "Autodesk". Disponible en World Wide Web: http://www.autodesk.es/adsk/servlet/pc/index?siteID=455755&id=14626995. Último acceso: 18 / 05 / 2011.

[55] Tutorial sobre el uso de la herramienta 3D Studio Max. Disponible en World Wide Web: Bipped rigging, Part 3. Último acceso: 18 / 05 / 2011.

[56] Tutorial sobre el uso de la herramienta 3D Studio Max. Disponible en World Wide Web: Biped Animation. Último acceso: 18 / 05 / 2011.

[57] Plugin utilizado para exportar objetos "Skeletal Mesh" y sus animaciones a archivos binarios. Disponibel en World Wide Web: http://udn.epicgames.com/Two/ActorX.html. Último acceso: 10 / 05 / 2011.

[58] Disponible en World Wide Web: http://wiki.pixelminegames.com/index.php?title=Tools:nFringe. Último acceso: 09 / 05 / 2011.

15. ANEXOS.

15.1. Anexo I: Glosario de términos.

En este anexo se van a presentar todos los términos y acrónimos junto con sus definiciones utilizados en el documento:

TÉRMINO	DEFINICIÓN
2D	Dos dimensiones.
3D	Tres Dimensiones.
ActorX	Herramienta desarrollada por "Epic Games" que permite exportar las mallas creadas en cualquier aplicación de modelado 3D al motor "Unreal Development Kit".
Anim Set	Elemento propio del motor "Unreal Development Kit" que almacena todas las animaciones propias de un objeto "Skeletal Mesh".
Anim Tree	Elemento propio del motor "Unreal Development Kit" que permite indicar la animación a reproducir del objeto "Skeletal Mesh" según se produzcan ciertas circunstancias de éste.
API	Application Programming Interface. Conjunto de funciones, procedimientos, clases, métodos (en función del paradigma de programación) que ofrece una biblioteca para ser utilizado en el desarrollo de aplicaciones.
Bink	Extensión de video utilizada por el motor "Unreal Development Kit" para poder utilizarlo.
Content Browser	Herramienta integrada en "Unreal Development Kit" que permite estructurar todos los elementos en paquetes, a la vez que crear ó importar nuevos.
CrossPlane	Juego desarrollado por Julián Marchante de la Dueña para su Proyecto Fin de Carrera.
Deathmatch	Tipo de videojuegos que consiste en lograr matar a los restantes jugadores.
Flujo de ejecución	Secuencia de acciones a realizar mediante la ejecución del juego.
Forma	Véase *Modelo*.
FPS	Shooter en primera persona.
Free	Gratuito.
Freeware	Cualquier software que no requiere pago ni otra compensación por parte de los usuarios que los usan. Que sean gratuitos no significa que se pueda acceder a su código fuente. El término fue acuñado por Andrew Fluegelman en 1982.
Game Engine	Motor de videojuegos. Conjunto de herramiestas proprocionadas para el desarrollo de videojuegos.
GPU	Unidad de procesamiento gráfico.
IA	Inteligencia Artificial.
Indie	Abreviación en inglés de la palabra "independiente". Un juego indie es básicamente uno en que el autor recibe todas las ganancias por su venta. Son juegos diseñados usualmente por una sola persona (con ayuda externa) que se encarga de escribirlo, armarlo y hasta a veces de venderlo personalmente.

Kismet	Herramienta integrada en "Unreal Development Kit" en la cual se implementa el flujo de ejecución del mapa desarrollado.
Lenguaje de scripting	Tipo de lenguaje de programación que es generalmente interpretado.
Mapa	Modelo en 3D del mundo creado mediante el editor.
Matinee	Herramienta integrada en "Unreal Development Kit" con la que se pueden implementar animaciones y cinemáticas para reproducirlas en ejecución.
Mesh	Malla. Objetos creados mediante modelado 3D utilizados en el motor "Unreal Development Kit".
Modelo	Elemento proporcionado por "Unreal Editor" con el que se pueden crear objetos básicos, como paredes, suelo, etc. Su apariencia por defecto es un hexágono con aristas de color rojo. Puede ser trasladado, rotado y escalado según preferencias, al igual que modificada su forma.
NFringe	Modulo desarrollado por "Epic Games" para utilizar en el entorno de desarrollo "Visual Studio" y poder implementar con el lenguaje "UnrealScript".
Nivel	Modelo en 3D del mundo creado mediante el editor.
Open Source	El software OpenSource se define por la licencia que lo acompaña, que garantiza a cualquier persona el derecho de usar, modificar y redistribuir el código libremente.
Path	Camino que toma el software para acceder a los datos contenidos en una unidad de almacenamiento.
Physics Asset	Elemento propio del motor "Unreal Development Kit" que almacena toda la estructura de esqueleto de un objeto "Skeletal Mesh".
Propietario	Software privativo en el cual el usuario tiene limitaciones para usarlo, modificarlo o redistribuirlo (esto último con o sin modificaciones).
RTS	Estrategia en tiempo real.
Shoot'em up	"Mátalos a tiros". Género de videojuegos en los que el jugador controla al personaje principal y se debe disparar contra todos los enemigos que van apareciendo.
Shooter	Estilo de videojuegos de disparos en el que el jugador toma el control de un personaje armado con el que interactuará con el entorno disparando de forma voluntaria.
Skeletal Mesh	Tipo especial de "Mesh" en cual posee un esqueleto internamente compuesto por huesos. Mediante este esqueleto se pueden realizar animaciones del objeto.
Socket	Punto de acople de un "Skeletal Mesh" con otros objetos diferentes.
UDK	Acrónimo de "Unreal Development Kit". Versión gratuita del motor "Unreal" desarrollado por "Epic Games".
Unreal Editor	Herramienta integrada en "Unreal Development Kit" en la cual se puede realizar el modelado 3D del mundo.
Unreal Engine	Motor de videojuegos "Unreal". Motor desarrollado por "Epic Games" siendo su primera versión de Junio de 1998.
UnrealScript	Lenguaje de programación propio del motor "Unreal Development Kit" que permite realizar cambios en el motor, sin tener que modificar el código fuente.

Tabla 8.Glosario de términos.

15.2. Anexo II: Guía del juego "CrossPlane".

Mediante estas guías se va a presentar una ayuda al jugador a la hora del desarrollo de todo el juego. Se van a explicar cada uno de los niveles que componen el juego: CrossPlane_A y CrossPlane_B.

CrossPlane_A

Empieza la acción y te encuentras en una sala cuadrada (Sala 1). En la pared de enfrente se muestra una puertas gris y a la derecha una valla que bloquea el paso; en la parte central se aprecia un pulsador que al acercarte y pulsar la tecla "acción" se apaga y enciende la luz de todo el nivel.

En la parte de la derecha se muestran unas compuertas, grandes, semiabiertas, con un símbolo en cada extremo, y que parecen atascadas y no se pueden abrir. Cuando te acercas a ellas, se muestra por un tiempo una sala donde se encuentra una llave y aparece un mensaje diciendo "*primero tienes que conseguir la llave*". Parece ser que hay que conseguir esa llave para poder abrir las compuertas.

A la izquierda se observan tres pulsadores. Al utilizarlos algunos causan daño al protagonista, restando la vida de éste, y uno de ellos abre la puerta que se tiene al lado.

Se entra a una sala nueva (Sala 2) repleta de columnas. En la pared de la izquierda se tiene un pulsador rodeado de dos luces rojas que al utilizarlo se muestra la imagen de una columna de fuego. Está localizada al fondo de la sala a la izquierda después de un laberinto de cristales. Al acercarte aparece el mensaje "*YA SE TIENE ENERGÍA*" con lo que parece que ha sucedido algo. Al salir nuevamente del laberinto, se puede observar que las luces del pulsador han cambiado de color. Al utilizar de nuevo el pulsador se muestra la apertura de otra puerta de una sala, y la imagen te lleva a otra nueva sala con otro pulsador, y nada más terminar la visualización se muestra el mensaje: "*TIENES DOS MINUTOS*". ¡Se tiene tiempo para hacer lo que se tenga que hacer!

Así que corres y corres hasta llegar a la sala mostrada (Sala 2.5). Al utilizar el pulsador ¡desaparece! y se muestra el mensaje "*A POR OTRA*". Mirando por toda la sala se observa que ha aparecido un nuevo pulsador en otra pared. Al utilizar este nuevo pulsador vuelve a suceder lo mismo. Así hay que actuar hasta que uno de los pulsadores muestra la apertura de una sala cerca de la Sala 1 y frente a la Sala 2. Al llegar a esta sala (Sala 3) se aprecia al fondo unos grandes tanques con un pulsador. Al utilizarlo aparece un mensaje diciendo "*JUSTO A TIEMPO*", parece que se ha realizado todo correctamente y en el tiempo indicado. Pero todo no ha acabado…la cámara muestra la sala inicial donde se encuentra un robot armado dando un mensaje aterrador, se muestra la apertura de otra sala donde se encuentran las armas para poder combatir con el enemigo; y de repente se apaga la luz, lo que faltaba. Pero ¿dónde están las armas?

Hay que salir de esta sala, girar a la derecha y al final del pasillo de nuevo girar a la derecha para encontrar la sala donde se encuentran las armas, que se encuentran en el fondo, se recogen y por fin ya te puedes enfrentar al enemigo.

Así que vas a por él disparándole desde el primer momento hasta conseguir matarlo, pero sigues a oscuras. Para poder encender la luz hay que volver al pulsador de la sala inicial. Al utilizarlo se muestra la apertura de otra puerta.

Así que hay que encontrarla: al final del pasillo al lado de la sala de las armas, sala 5. Al entrar te encuentras con otro pulsador, y una cristalera a la izquierda que no se puede atravesar ni saltar, mostrando otra sala con pulsador. Al utilizar el pulsador, la puerta de la sala del otro lado del cristal empieza a abrirse y cerrarse sola, pero ¿por dónde se llega a esa sala?

Al salir al pasillo te encuentras una puerta a la derecha que impide el paso pero al acercarnos a ella se muestra el mensaje: "*ESTA PUERTA PARECE UN HOLOGRAMA*", ¿holograma? ¿En una puerta que parece blindada?

Pues sí, se puede atravesar; al final no es una puerta, es un holograma. Al atravesar el holograma ya se tiene acceso a la sala que se quería llegar (sala 6) pero también se ve una gran puerta al fondo del pasillo. Como no, esta sala tiene otro pulsador. Al accionarlo, el protagonista será teletransportado a una de las salas ya visitadas ó a una nueva sala, a la que hay que llegar, mostrando el mensaje "*SI, POR FIN*". Se aparece en otra sala con una fuente de fuego y detrás de ella la llave. Hay que coger la llave desapareciendo ésta para indicar que ya se posee. La puerta de esta sala es la misma que se vio al pasar por el holograma. Sí, te encuentras al otro lado. Para abrirla hay que pulsar la tecla 'acción'.

Con la llave en posesión se pueden abrir ya las compuertas de la sala inicial. Al abrirse las compuertas se muestra un foso con una especie de fuego saliendo de él. Parece que habrá que saltarlo. En el caso de caer en el foso se vuelve a aparecer en la puerta de esta sala, para poder intentar el salto. Cuando se consigue saltar el foso y al acercarse a las compuertas se muestra el mensaje: "*Enhorabuena, has conseguido el primer nivel*".

La imagen se fundirá a negro indicando que el primer nivel ha sido completado.

CrossPlane_B

Nada más arrancar el nivel se muestra el cierre de un túnel y el movimiento de cámara a lo largo de todo el nivel hasta llegar al protagonista. El túnel debe ser la salida a la que llegar.

El protagonista aparece en un pequeño bosque con un palacio a la espalda. Según el movimiento de cámara hay que ir hacia la izquierda, impidiendo el paso un pequeño barranco con un río al fondo.

Recorriendo el borde del barrando se descubre que existe un puente, pero que no tiene suelo, con lo que es imposible pasar. En la entrada del puente aparece un letrero diciendo: "TENGO QUE MONTAR EL SUELO DEL PUENTE", pero ¿cómo?

Para conseguir el montaje del suelo del puente hay que encontrar la misma llave que en el nivel anterior. Una buena forma de esconder la llave sería en el fondo del agua del río. ¿Sabrá el protagonista nadar? La mejor forma es comprobarlo tirándose al agua.

Parece que sí se puede nadar y hasta bucear, pero con ¡cuidado de no ahogarse! Buceando por el fondo se observan piedras y más piedras y a orillas de la cascada se puede observar algo distinto, ¡se ha encontrado la llave!

Al tocarla desaparece, indicando que ya se posee. Pero ahora hay que salir del río. Al otro lado del río se observa una enredadera de plantas, que al aproximarnos a ella se produce un teletransporte del protagonista a la parte superior del barranco, la única

forma de poder salir del río. En este caso, al llegar al puente, el suelo se compone pudiendo cruzar al otro lado, pero ¿y ahora?

Te encuentras en un gran campo en el que parece que no hay salida: a la derecha se tiene grandes rocas que no puedes escalar, a la izquierda agua que impide el paso, y en frente un gran muro infranqueable. Paseando por todo el campo y buscando una salida se encuentra una especie de árbol seco en el que en su interior se observa algo raro. Justo cuando se va a tocar este objeto, el protagonista sale volando, literalmente, hasta llegar encima del muro, pudiendo pasar al otro lado.

Hay otro descampado repleto de rocas, pero el túnel mostrado al comienzo del nivel se puede observar al fondo. Pero está cerrado, no pudiendo continuar. Hay que ir al minarete que se puede observar en el centro del descampado. En uno de sus lados se encuentra un pulsador, que al utilizarlo se produce un teletransporte del protagonista a lo alto de la torre.

Desde esta posición se puede ver todo el paisaje, las dos torres gigantes, el muro, el mar, las rocas y el palacio. En una columna del minarete hay otro pulsador que al utilizarlo se muestra la apertura del túnel de salida. Pero en ese momento aparecen en el horizonte dos naves voladoras que aterrizan en el campo apareciendo dos enemigos de ellas, uno de cada nave, que se van a hacer guardia al túnel de salida, que ya está abierta. Seguidamente con un sonido ensordecedor se ven otras dos naves que también aterrizan, aparecen otros dos enemigos, pero éstos ya te están esperando para aniquilarte.

¿Y ahora?

Aparece otro pulsador en lo alto del torreón, que antes no estaba, y al utilizarlo el protagonista se teletransporta otra vez, pero esta vez a la parte baja del minarete. ¡Al lado de los enemigos! ¡Cuidado que si te matan apareces al principio del nivel!

Tienes que ser rápido, ya que hay enemigos que te están esperando. Se pueden quitar de en medio ó esquivarlos, pero siendo esto úlitmo bastante difícil. Una vez hayas acabado con ellos, sobre todo los guardianes de la puerta, puedes atravesar el túnel libremente.

¡CONSEGUISTE FINALIZAR EL JUEGO CON ÉXITO!