INSTITUTO SUPERIOR TECNOLÓGICO QUITO

Formamos tu PROPÓSITO DE VIDA

Guía práctico experimental de

# DESARROLLO DE VIDEOJUEGOS

Autor: Alexis Xavier Urgíles Pillalaza

ISBN: 978-9942-562-51-7

2025

# GUÍA PRÁCTICO EXPERIMENTAL DE DESARROLLO DE VIDEOJUEGOS 2D Y 3D

**CARRERA:** ANIMACIÓN MULTIMEDIA

**PERIODO ACADÉMICO:** SEPTIEMBRE 2025 – FEBRERO 2026

**DOCENTE TUTOR:** ALEXIS URGILÉS PILLALAZA

**EDICIÓN: PRIMERA**

**AÑO: 2025**

**TRABAJO EN EDICIÓN:**

DOXA EDITION

**EQUIPO EDITORIAL:**

**MSc. KEYERMAN TOAPANTA CISNEROS**

**MSc. DAVID ALEJANDRO PEREIRA SALAZAR**

**ISBN: 978-9942-562-51-7**

ISBN: 978-9942-562-51-7
9 789942 562517

**QUITO – ECUADOR**

# 1 GUÍA PRÁCTICA EXPERIMENTAL DE DESARROLLO DE VIDEOJUEGOS

## 1.1. Descripción de la asignatura

La asignatura Desarrollo de Videojuegos 2D y 3D proporciona al estudiante las herramientas teóricas y prácticas necesarias para diseñar, desarrollar y montar videojuegos utilizando recursos gráficos en dos y tres dimensiones. Se abordan los conceptos fundamentales del diseño interactivo, la creación de personajes, escenarios, mecánicas y narrativas, así como el uso de software especializado en modelado, arte 2D, y programación básica de entornos jugables. El curso promueve el desarrollo de productos funcionales que integren estética, jugabilidad y eficiencia técnica.

El videojuego es hoy una de las expresiones más dinámicas e influyentes dentro de las industrias culturales y creativas. Esta asignatura permite al estudiante experimentar todas las fases del proceso de producción, desde la conceptualización hasta la implementación, considerando criterios de optimización de recursos, adaptabilidad multiplataforma y experiencia del usuario. Se fomenta el trabajo colaborativo y la resolución de problemas creativos con enfoque profesional.

Como parte clave del itinerario formativo en la carrera de Animación Multimedia, esta materia fortalece la capacidad del estudiante para integrar competencias técnicas, artísticas y narrativas en el desarrollo de productos interactivos. Contribuye a la construcción de un perfil profesional capaz de crear videojuegos innovadores, funcionales y visualmente atractivos, preparados para su inserción en el mercado digital.

## 1.1. Objetivos de la asignatura

Desarrollar videojuegos en entornos 2D y 3D, mediante el uso de herramientas de diseño, modelado, arte digital y programación, para crear productos interactivos funcionales que integrenjugabilidad, narrativa visual y eficiencia en el uso de recursos técnicos y creativos.

## 1.2. Contenidos mínimos

1. Conceptos básicos de juegos 2D y 3D.
2. Diseño, desarrollo y montaje de videojuegos
3. Creación de videojuegos 2D y 3D.
4. Arte 2D para videojuegos
5. Modelado 3D.
6. Herramientas 2D y 3D.

### 1.3. Resultados de aprendizaje

Desarrolla la creación de videojuegos en 2D y 3D, considerando eficiencia y eficacia en el uso de recursos.

# 2 UNIDAD 1: Introducción a Desarrollo 2D y 3D

## 2.1 Practica experimental 1

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Creación de un proyecto tipo plataforma en Construct 2 con tres elementos: Tile, Background y Player**

**Objetivos**

Crear un proyecto en Construct 2 del género plataforma implementando los elementos esenciales: un fondo, un sistema de tiles que construya el nivel y un jugador funcional con comportamiento de movimiento.

**Antecedentes/Escenario**

Construct 2 ofrece un sistema visual mediante eventos que facilita la creación de prototipos. Un juego de plataformas requiere una estructura específica: un fondo estático, un mapa decorativo o funcional formado con tilesets y un jugador capaz de desplazarse y saltar. Estos elementos permiten sentar la base técnica del gameplay y de la estructura del nivel.

**Recursos necesarios**

Construct 2 instalado.

Sprites o imágenes para fondo, tiles y personaje.

Plantilla de proyecto vacía.

**Planteamiento del problema**

Se requiere crear un proyecto funcional que represente un nivel inicial de plataformas, estableciendo los componentes visuales y jugables básicos antes de avanzar con mecánicas más complejas.

**Pasos por realizar**

Paso 1: Crear un proyecto nuevo en Construct 2 y configurar el layout principal.

Paso 2: Importar una imagen para el fondo y ajustar su posición y tamaño.

Paso 3: Crear un objeto Tilemap y cargar un tileset para construir plataformas y paredes.

Paso 4: Añadir un sprite del jugador y configurar su tamaño, puntos de animación y origen.

Paso 5: Asignar comportamientos al jugador, como Platform y Scroll To.

Paso 6: Construir una sección de nivel utilizando el Tilemap, asegurando colisiones adecuadas.

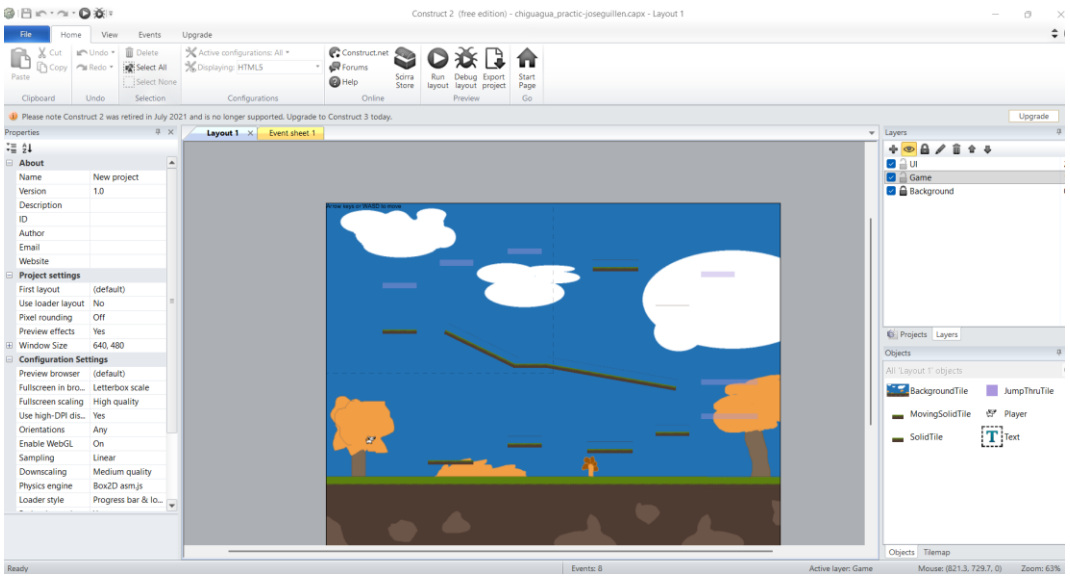
Paso 7: Probar el desplazamiento del jugador y ajustar parámetros.

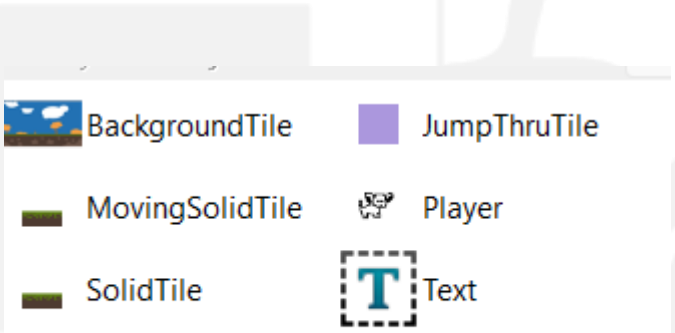
**Desarrollo**

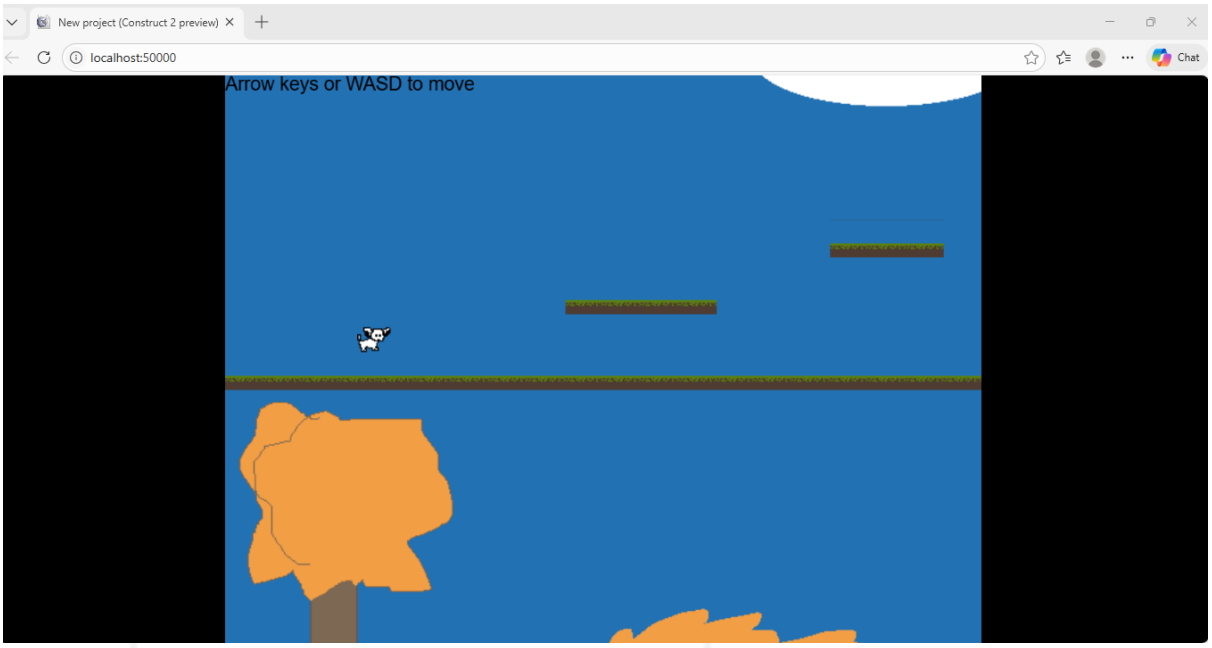
Se inicia un proyecto vacío y se cambia el nombre del layout a “Nivel1”. En el panel de propiedades se establece una resolución acorde al tamaño del juego, por ejemplo 1280x720. A continuación se importa una imagen de fondo que servirá como referencia visual, colocándola en la capa base. Se ajusta su parámetro de Parallax para que permanezca estático durante el desplazamiento del jugador.

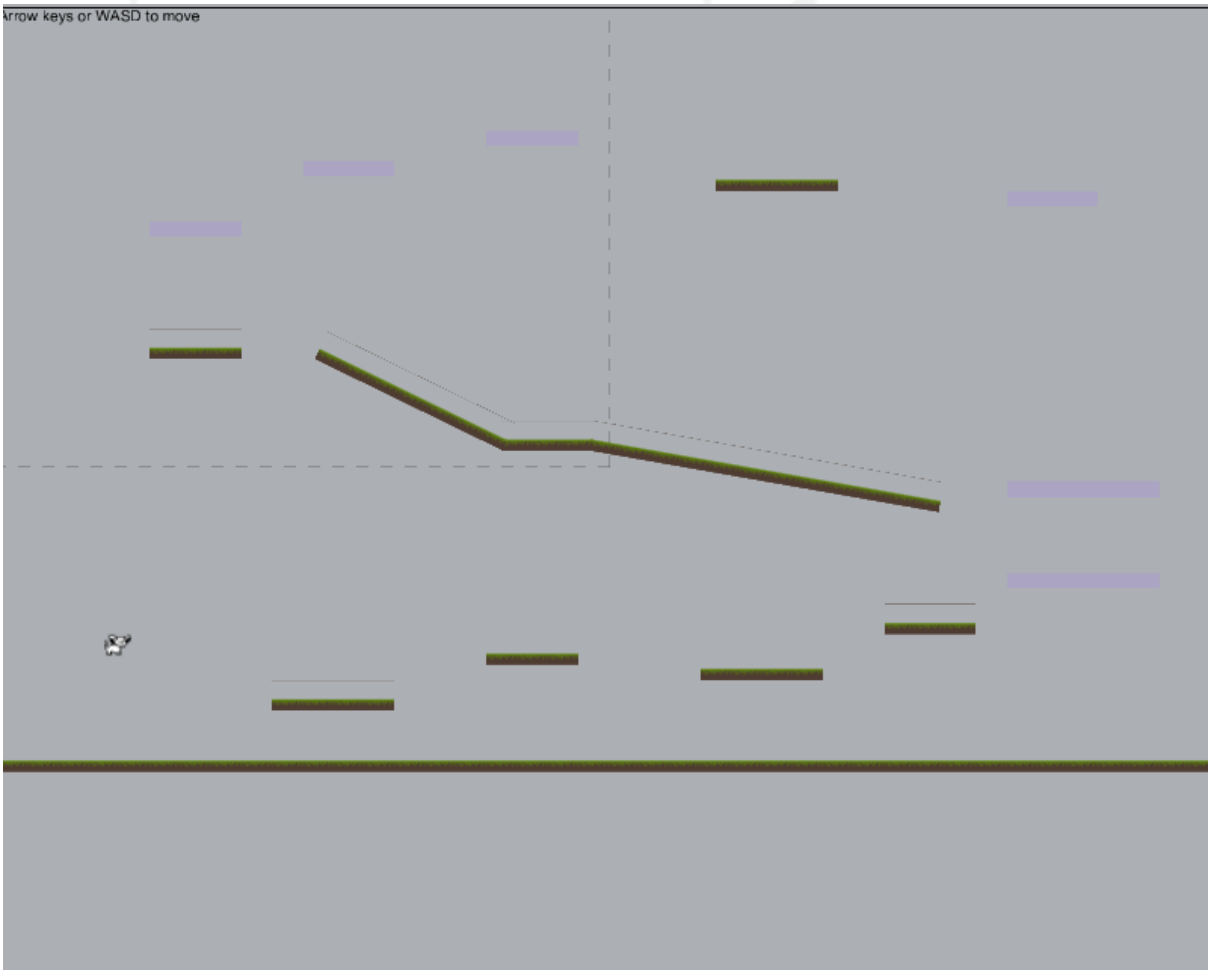
Luego se inserta un objeto Tilemap y se carga un tileset compuesto por bloques de terreno. Desde la herramienta de dibujo del Tilemap se construyen plataformas, suelos y paredes que formarán el nivel del juego.

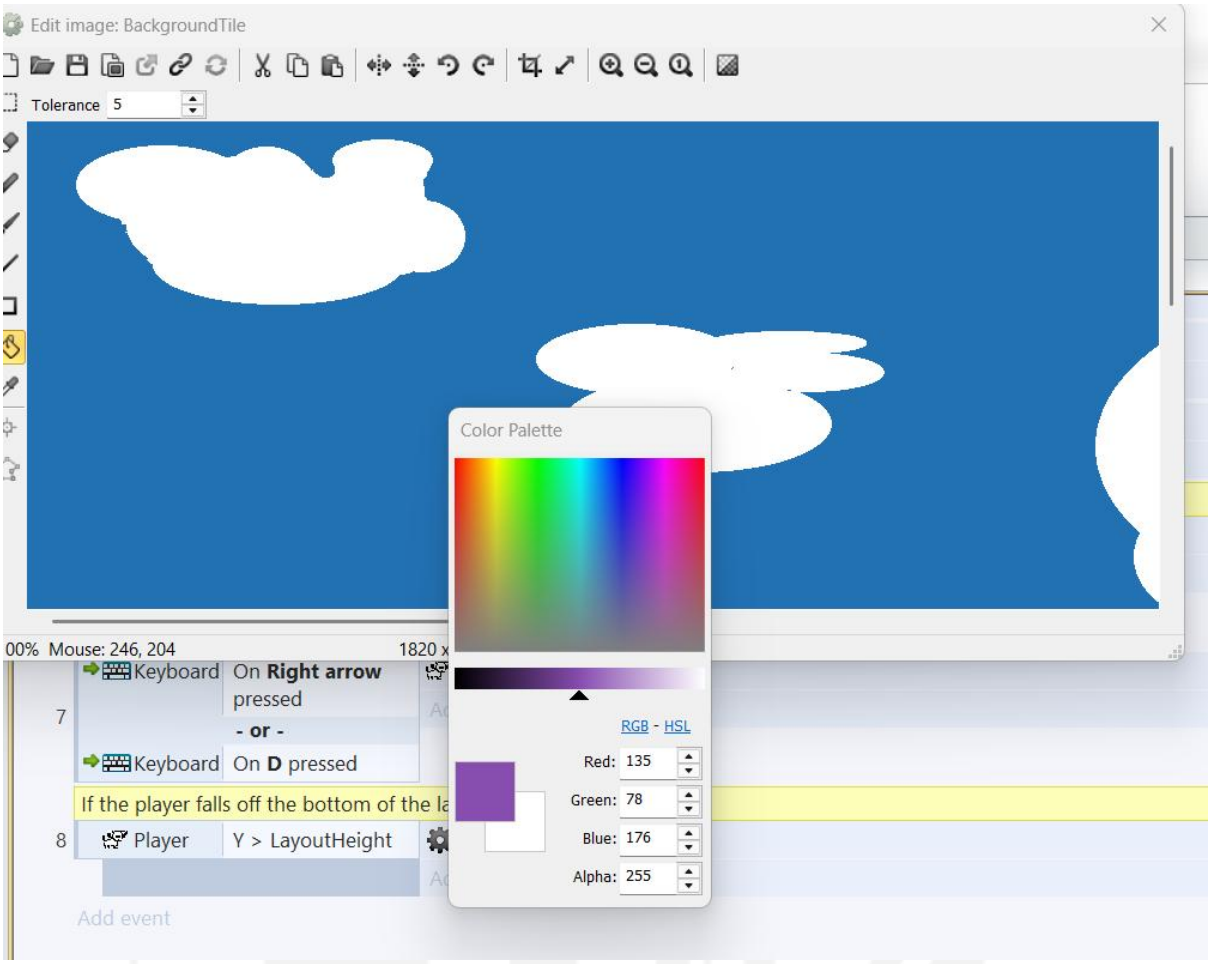
Para el jugador se importa un sprite simple con una o más animaciones. Se ajusta su punto de origen al centro inferior para una correcta detección de colisiones. Se añaden comportamientos como Platform para habilitar movimiento y saltos, y Scroll To para que la cámara siga al personaje. Posteriormente se prueban colisiones ajustando el Tilemap para que cada bloque actúe como una superficie sólida. Se realizan iteraciones de prueba modificando los valores de aceleración, desaceleración, gravedad y fuerza de salto para obtener un control satisfactorio. Finalmente se realiza un test para verificar que el jugador se mueva sin atravesar plataformas y que la cámara siga correctamente su desplazamiento.


Construct 2 (free edition) - chiguagua_practic-joseguillen.capx - Layout 1
File
Home
View
Events
Upgrade
Please note Construct 2 was retired in July 2021 and is no longer supported. Upgrade to Construct 3 today.
Upgrade
Properties
About
Name
New project
Version
1.0
Description
ID
Author
Email
Website
Project settings
First layout
(default)
Use loader layout
No
Pixel rounding
Off
Preview effects
Yes
Window Size
640, 480
Configuration Settings
Preview browser
(default)
Fullscreen scaling
High quality
Use high-DPI dis...
Yes
Orientations
Any
Enable WebGL
On
Sampling
Linear
Downscaling
Medium quality
Physics engine
Box2D asm.js
Loader style
Progress bar & lo...
Layout 1
Event sheet 1
Layers
UI
Game
Background
Projects
Layers
Objects
All 'Layout 1' objects
BackgroundTile
JumpThruTile
MovingSolidTile
Player
SolidTile
Text
Objects
Tilemap
Ready
Events: 8
Active layer: Game
Zoom: 63%


BackgroundTile
JumpThruTile
MovingSolidTile
Player
SolidTile
Text


New project (Construct 2 preview)
localhost:50000
Chat
Arrow keys or WASD to move


Arrow keys or WASD to move




**Resultados Esperados**

Al completar esta actividad, el estudiante habrá creado un prototipo funcional de un juego de plataformas en Construct 2 que integre correctamente un fondo, un sistema de Tilemap y un jugador con comportamiento de movimiento. Se espera que el alumno demuestre comprensión en la organización de capas, uso de tilesets para construir niveles y configuración adecuada del jugador mediante comportamientos como *Platform* y *Scroll To*. El proyecto final deberá mostrar un nivel navegable donde el personaje pueda desplazarse, saltar, interactuar con plataformas y mantener coherencia visual y jugable, estableciendo así la base para futuras mecánicas avanzadas.

## 2.2 Practica experimental 2

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Exploración de plantillas 2D y 3D en Unreal Engine**

**Objetivos**

Explorar plantillas predefinidas 2D y 3D en Unreal Engine para comprender su estructura interna, comportamiento del personaje, sistema de cámara y configuración inicial del proyecto.

**Antecedentes/Escenario**

Unreal Engine proporciona plantillas base que sirven como punto de partida para diferentes géneros. Estas plantillas incluyen configuraciones preestablecidas que permiten analizar cómo el motor estructura movimiento, cámara, colisiones, animaciones y lógica básica mediante Blueprints.

**Recursos necesarios**

Unreal Engine 5.

Computadora con capacidad 3D.

**Planteamiento del problema**

Se necesita comprender las diferencias entre proyectos 2D y 3D, así como sus configuraciones iniciales, para seleccionar la plantilla adecuada para futuros desarrollos.

**Pasos por realizar**

Paso 1: Crear un nuevo proyecto en Unreal Engine seleccionando la plantilla 2D.

Paso 2: Analizar la carpeta principal para identificar Blueprints, animaciones y sprites.

Paso 3: Abrir el personaje 2D y revisar su Event Graph para observar lógica de movimiento.

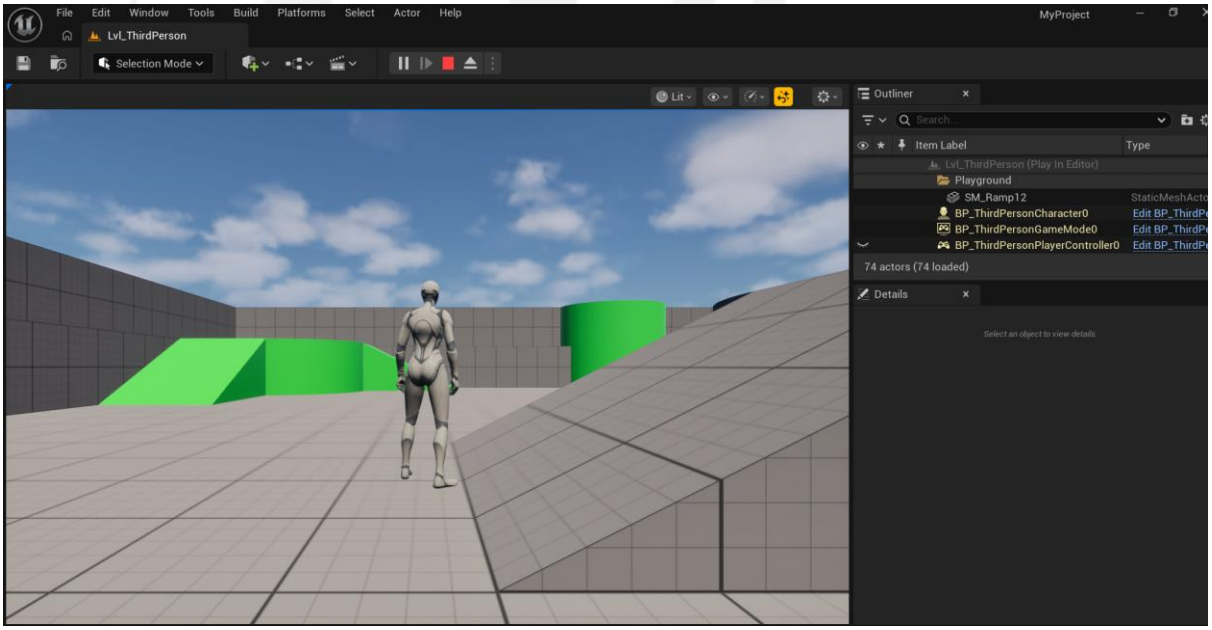
Paso 4: Crear otro proyecto utilizando una plantilla 3D.

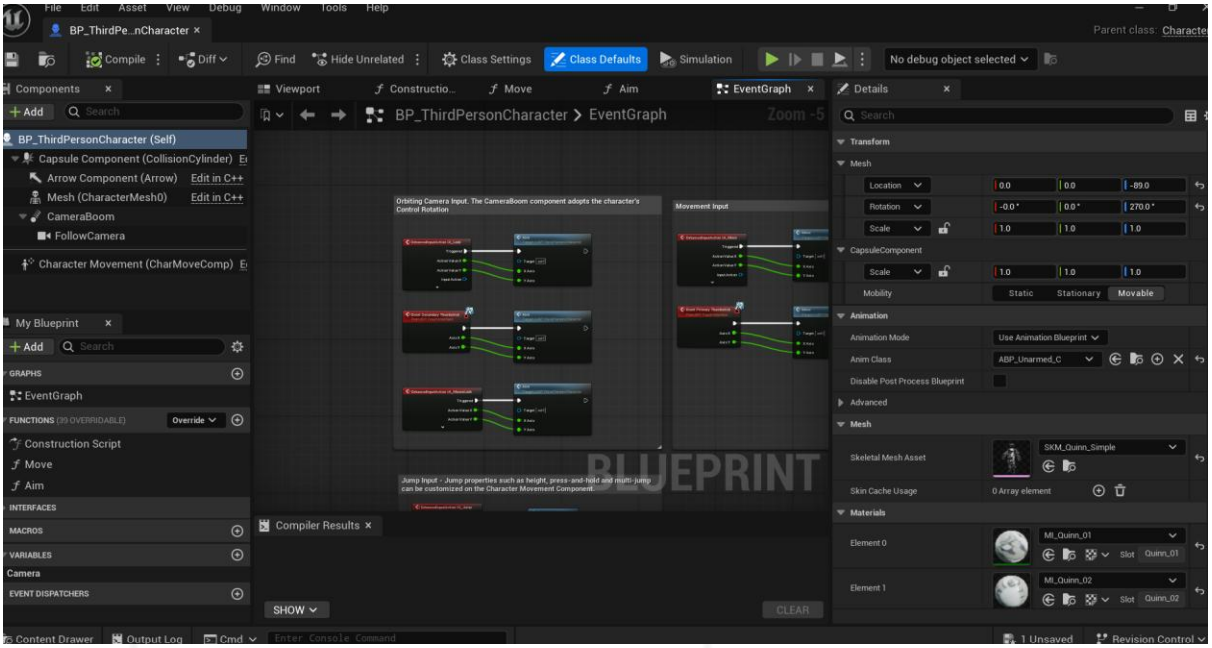
Paso 5: Observar el comportamiento del jugador, sistema de cámara y componentes.

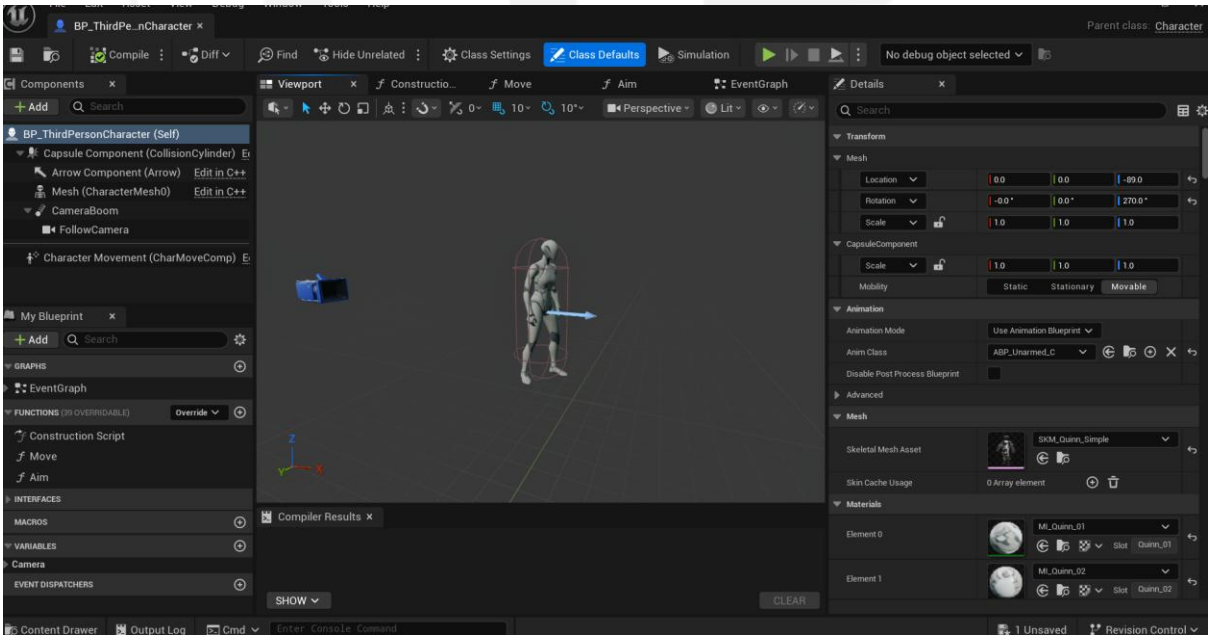
Paso 6: Comparar ambos proyectos en un documento descriptivo.

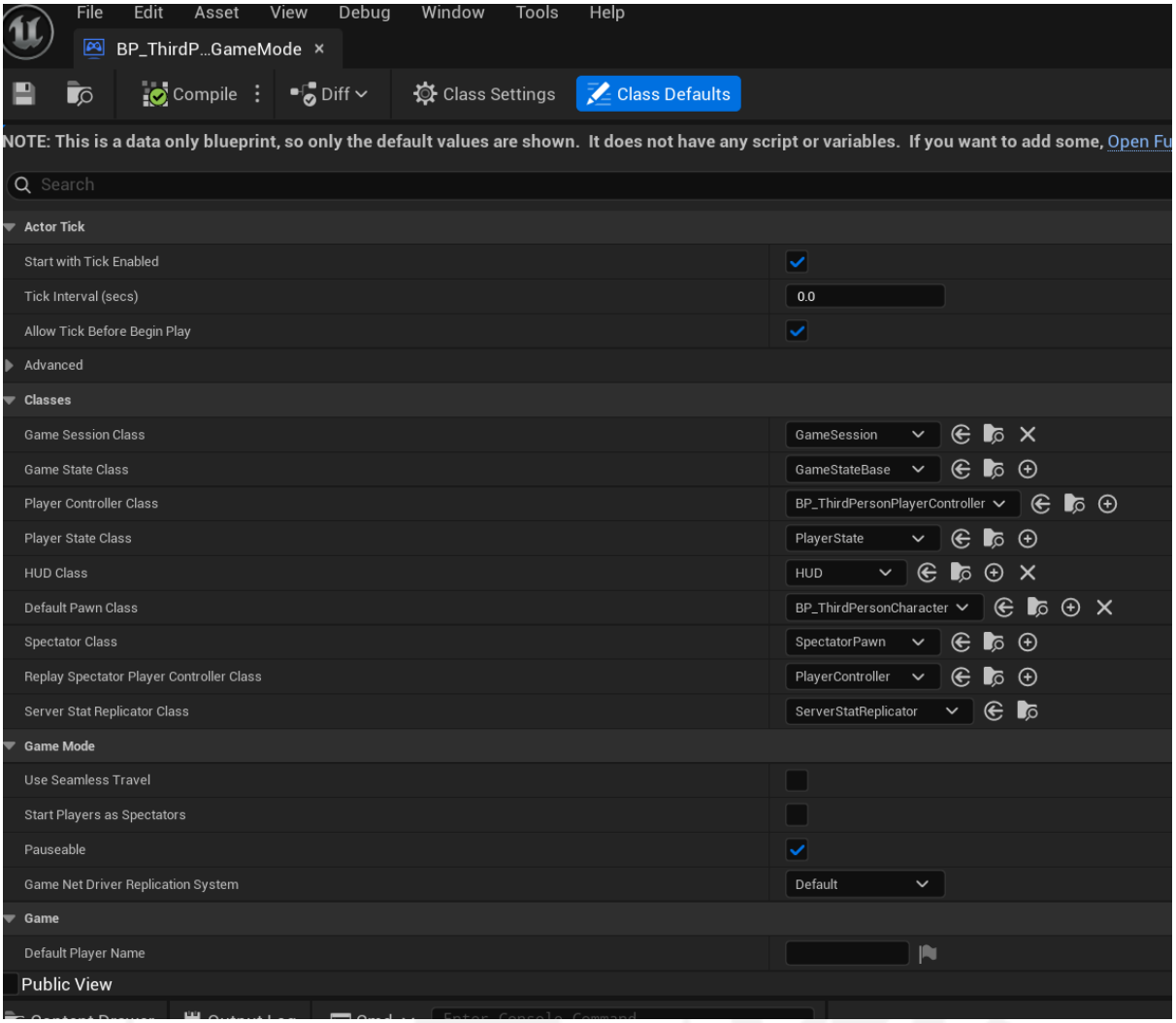
**Desarrollo**

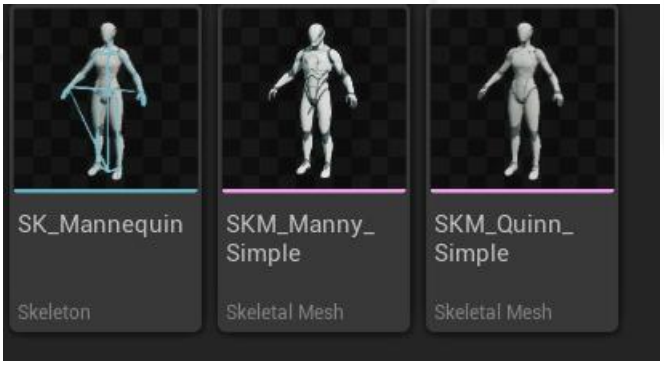
Se inicia Unreal Engine y se crea un proyecto utilizando la plantilla 2D. Dentro del proyecto se revisan carpetas como Blueprints, Maps y Sprites para identificar la estructura general. Al abrir el Blueprint del personaje se observa el uso de componentes como Capsule Component, Sprite y Camera Boom. En el Event Graph del personaje se estudian nodos encargados de desplazamiento, como InputAxis MoveRight y configuraciones de velocidad. Posteriormente se crea un segundo proyecto utilizando la plantilla 3D, como Third Person. Se examina el Blueprint del ThirdPersonCharacter observando componentes como la malla esquelética, la cámara montada en el Spring Arm y el Capsule Collider. Se ejecuta el proyecto apreciando cómo la cámara sigue al personaje desde atrás y cómo se comporta el movimiento en un espacio tridimensional. Finalmente se comparan ambas plantillas analizando sus diferencias en controles, dimensiones, colisiones, animación y lógica.

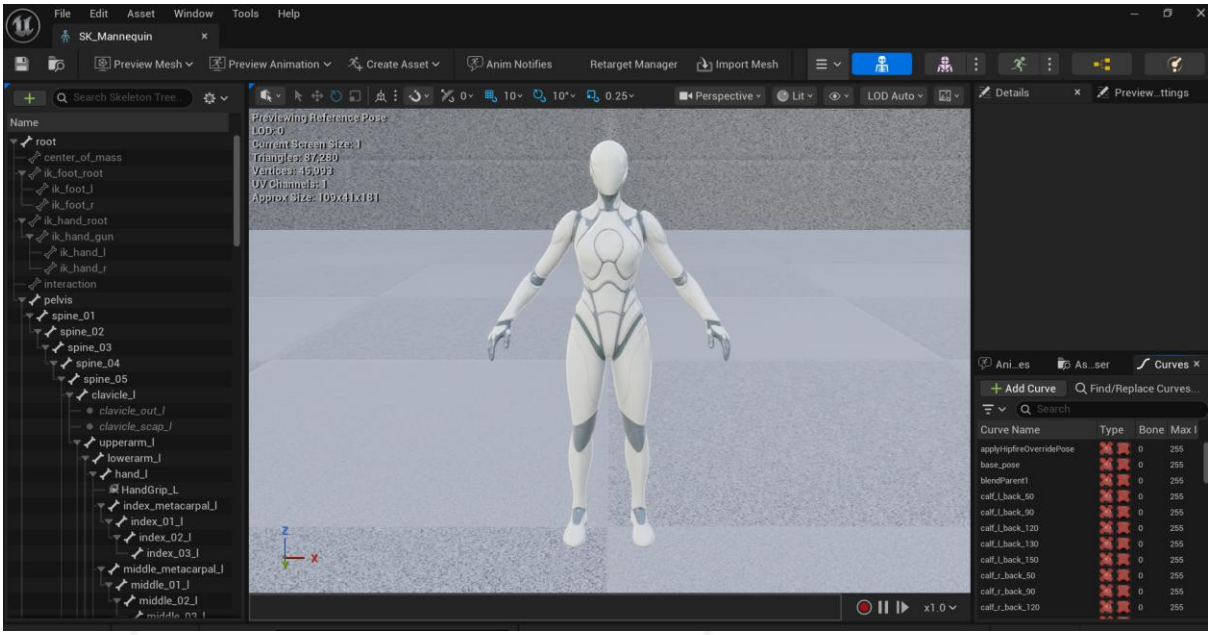



BP_ThirdPersonCharacter
EventGraph
Components
My Blueprint
Details
Compiler Results
BLUEPRINT


BP_ThirdPersonCharacter
Viewport
Components
My Blueprint
Details
Compiler Results


File Edit Asset View Debug Window Tools Help
BP_ThirdP...GameMode
Compile
Diff
Class Settings
Class Defaults
NOTE: This is a data only blueprint, so only the default values are shown. It does not have any script or variables. If you want to add some, Open Fu
Search
Actor Tick
Start with Tick Enabled
Tick Interval (secs)
0.0
Allow Tick Before Begin Play
Advanced
Classes
Game Session Class
GameSession
Game State Class
GameStateBase
Player Controller Class
BP_ThirdPersonPlayerController
Player State Class
PlayerState
HUD Class
HUD
Default Pawn Class
BP_ThirdPersonCharacter
Spectator Class
SpectatorPawn
Replay Spectator Player Controller Class
PlayerController
Server Stat Replicator Class
ServerStatReplicator
Game Mode
Use Seamless Travel
Start Players as Spectators
Pauseable
Game Net Driver Replication System
Default
Game
Default Player Name
Public View


SK_Mannequin
Skeleton
SKM_Manny_Simple
Skeletal Mesh
SKM_Quinn_Simple
Skeletal Mesh




**Resultados esperados**

Al finalizar la actividad, el estudiante será capaz de identificar con claridad las diferencias estructurales y funcionales entre las plantillas 2D y 3D de Unreal Engine. Se espera que comprenda cómo están organizadas las carpetas, qué componentes conforman cada tipo de personaje y cómo funciona su lógica interna mediante Blueprints. Además, deberá reconocer variaciones en el uso de cámaras, colisiones, animaciones y sistemas de movimiento entre ambas plantillas. Como resultado,

el alumno podrá seleccionar adecuadamente la plantilla más conveniente según el tipo de proyecto que desee desarrollar, demostrando criterio técnico y comprensión del funcionamiento base del motor.

## 2.3 Practica experimental 3

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Implementación de nuevas mecánicas en el proyecto de Construct 2**

**Objetivos**

Agregar nuevas mecánicas jugables al proyecto de plataformas previamente creado, ampliando las capacidades del jugador y la interacción con el mundo.

**Antecedentes/Escenario**

Una vez compuesto el nivel básico, Construct 2 permite extender el gameplay mediante eventos personalizados, nuevos objetos interactivos y condiciones lógicas.

**Recursos necesarios**

Proyecto creado previamente.

Nuevos sprites para ítems, plataformas móviles o enemigos.

**Planteamiento del problema**

El proyecto inicial necesita ampliarse mediante la implementación de mecánicas adicionales como colectables, plataformas dinámicas o enemigos simples.

**Pasos por realizar**

Paso 1: Revisar la estructura del proyecto existente.

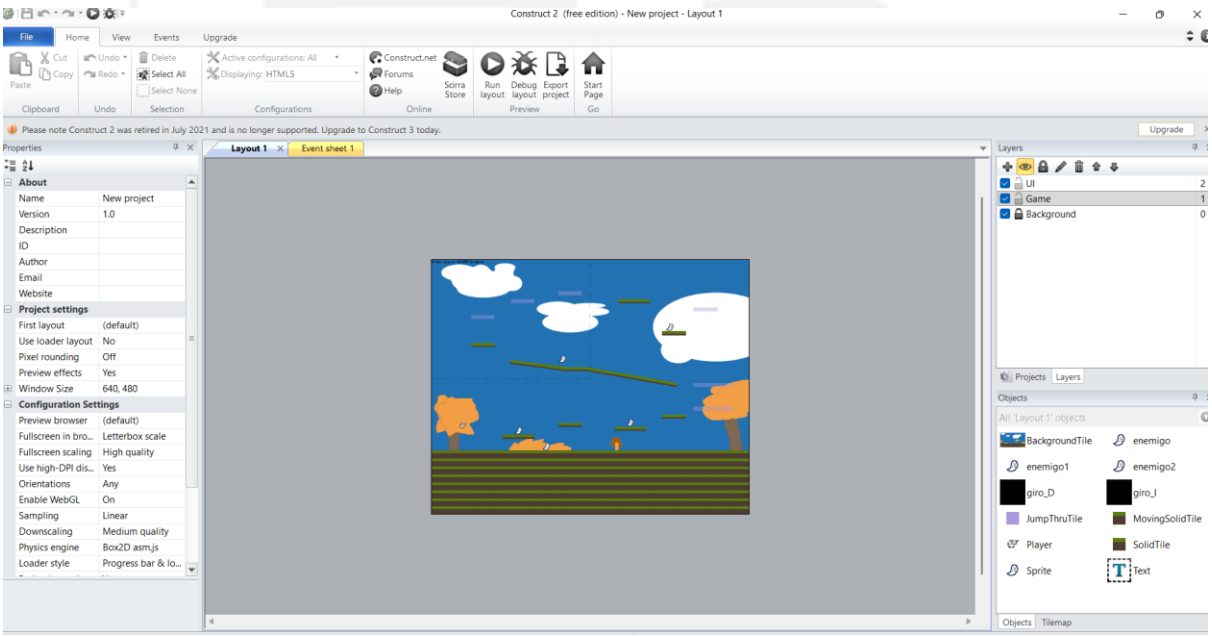
Paso 2: Importar nuevos sprites para los elementos a integrar.

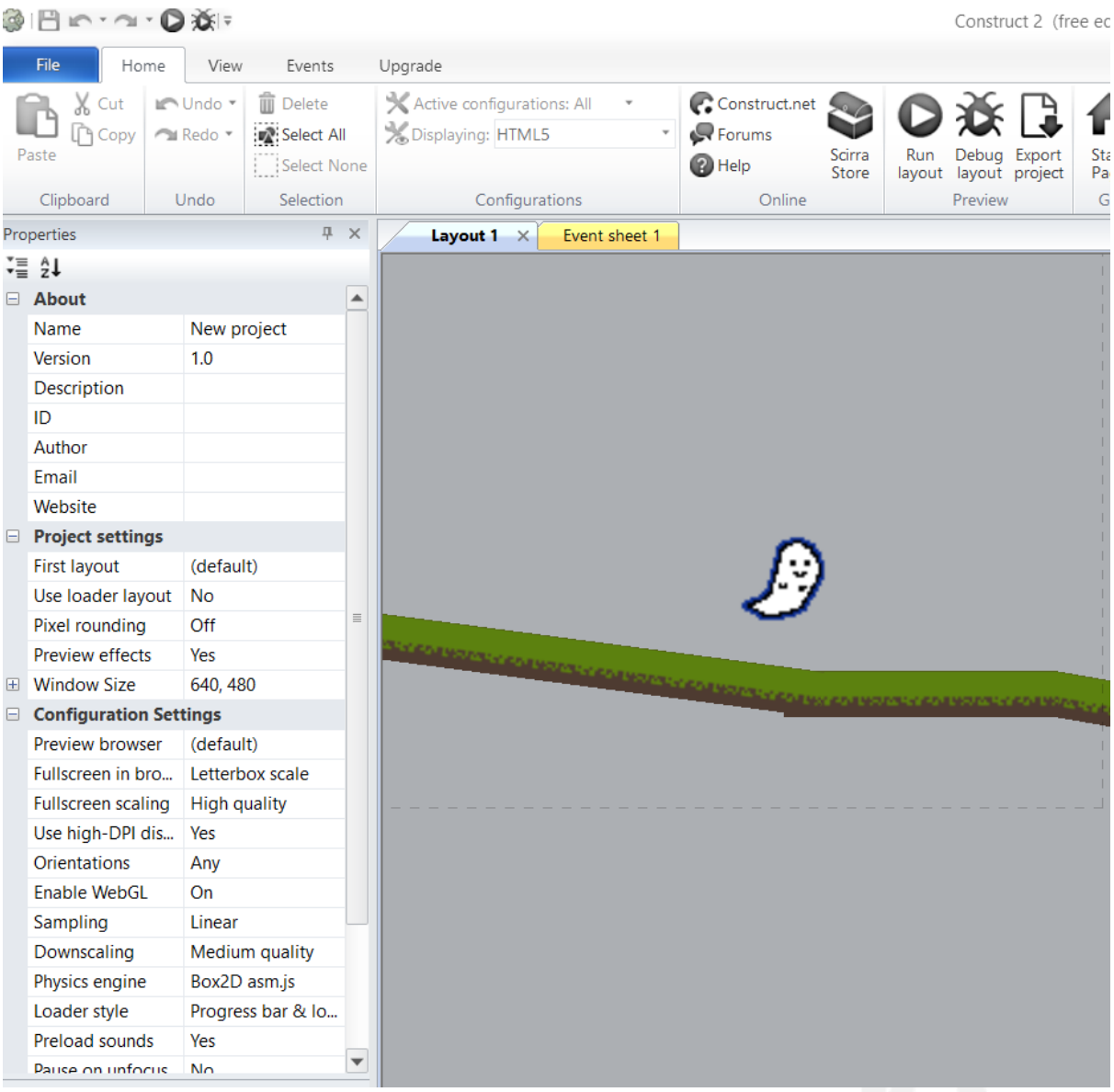
Paso 3: Crear eventos que permitan interacción con dichos elementos.

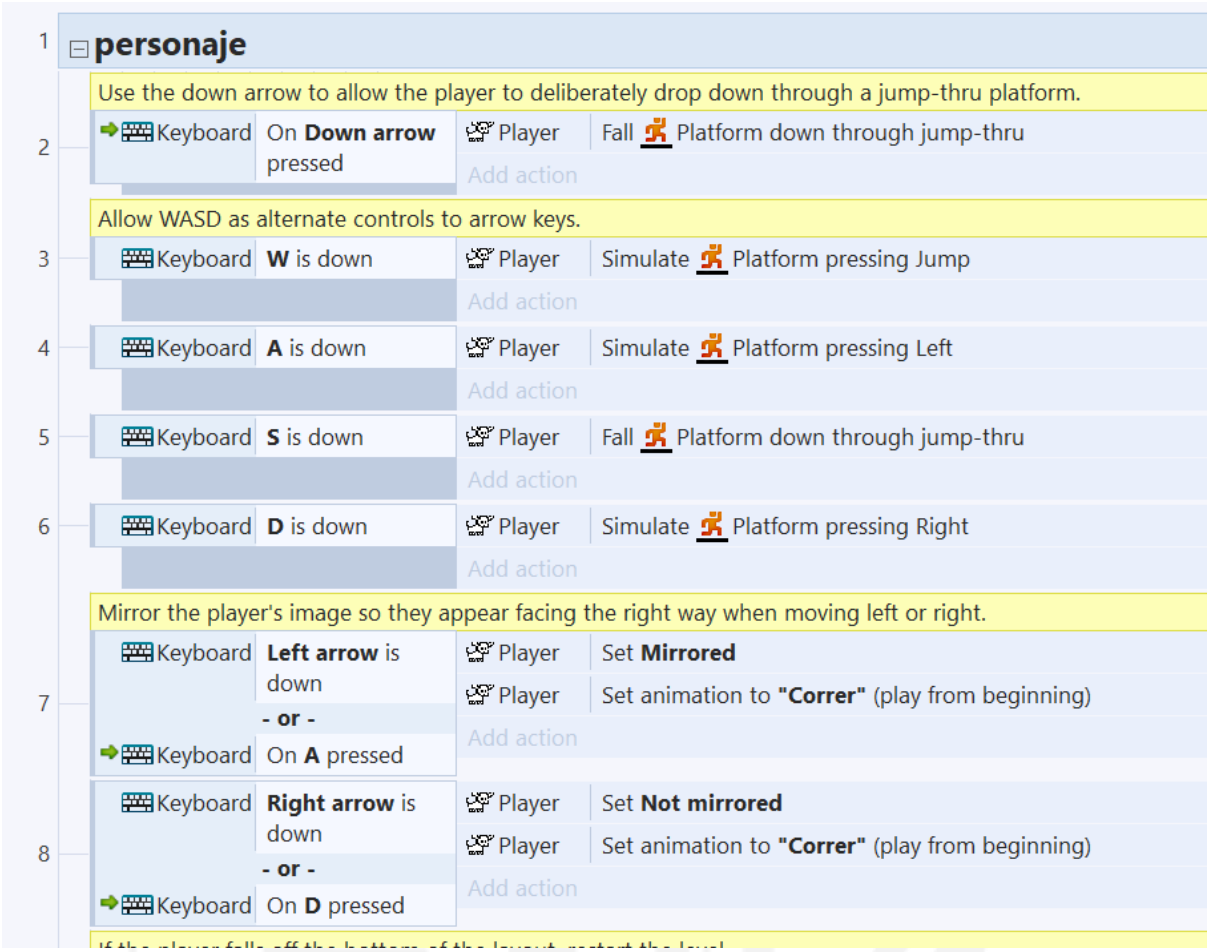
Paso 4: Probar funcionalidad y corregir comportamientos.

**Desarrollo**

Se abre el proyecto del nivel de plataforma. Se integra un nuevo sprite que representará un ítem coleccionable. Se crea un evento mediante la condición On collision between Player and Item que destruye el ítem y suma puntos a una variable global. Luego se añade una plataforma móvil mediante un sprite animado y se le asigna un comportamiento de movimiento utilizando el Behavior Sine o eventos personalizados con Set X. Se establecen límites de movimiento para evitar errores en la navegación del jugador. Finalmente se crea un enemigo sencillo que se mueve de un lado a otro mediante una variable de dirección. Se programa que si colisiona con el jugador por la parte superior, el enemigo desaparece, pero si el jugador es golpeado lateralmente, pierde vida o reinicia el nivel. Cada mecánica se prueba individualmente corrigiendo desplazamientos, tiempos y colisiones.




Construct 2 (free ec
File
Home
View
Events
Upgrade
Paste
Cut
Copy
Undo
Redo
Delete
Select All
Select None
Active configurations: All
Displaying: HTML5
Construct.net
Forums
Help
Scirra Store
Run layout
Debug layout
Export project
Clipboard
Undo
Selection
Configurations
Online
Preview
Properties
About
Name
New project
Version
1.0
Description
ID
Author
Email
Website
Project settings
First layout
(default)
Use loader layout
No
Pixel rounding
Off
Preview effects
Yes
Window Size
640, 480
Configuration Settings
Preview browser
(default)
Fullscreen in bro...
Letterbox scale
Fullscreen scaling
High quality
Use high-DPI dis...
Yes
Orientations
Any
Enable WebGL
On
Sampling
Linear
Downscaling
Medium quality
Physics engine
Box2D asm.js
Loader style
Progress bar & lo...
Preload sounds
Yes
Layout 1
Event sheet 1


1 personaje
Use the down arrow to allow the player to deliberately drop down through a jump-thru platform.
2 Keyboard On Down arrow pressed | Player Fall Platform down through jump-thru | Add action
Allow WASD as alternate controls to arrow keys.
3 Keyboard W is down | Player Simulate Platform pressing Jump | Add action
4 Keyboard A is down | Player Simulate Platform pressing Left | Add action
5 Keyboard S is down | Player Fall Platform down through jump-thru | Add action
6 Keyboard D is down | Player Simulate Platform pressing Right | Add action
Mirror the player's image so they appear facing the right way when moving left or right.
7 Keyboard Left arrow is down | Player Set Mirrored | Player Set animation to "Correr" (play from beginning)
- or -
Keyboard On A pressed | Add action
8 Keyboard Right arrow is down | Player Set Not mirrored | Player Set animation to "Correr" (play from beginning)
- or -
Keyboard On D pressed | Add action


**Resultados esperados**

Al concluir la actividad, el estudiante habrá ampliado el proyecto inicial mediante la integración de al menos tres nuevas mecánicas funcionales, demostrando dominio del sistema de eventos de Construct 2. Se espera que logre implementar correctamente ítems coleccionables que modifiquen variables del juego, plataformas dinámicas con trayectorias controladas y enemigos básicos con comportamientos predecibles y coherentes. Asimismo, el estudiante deberá evidenciar capacidad para depurar colisiones, ajustar movimientos y verificar el funcionamiento conjunto de todas las nuevas mecánicas dentro del nivel. El resultado final será un prototipo más completo, interactivo y técnicamente consistente, donde las nuevas dinámicas de juego amplían la experiencia del jugador y muestran comprensión de la lógica del motor.

# 3 UNIDAD 2: Entornos en Unreal Engine

## 3.1 Practica experimental 4

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Creación de un juego con gameplay completo en Construct 2**

**Objetivos**

Desarrollar un juego totalmente funcional en Construct 2 con inicio, desarrollo de nivel y finalización mediante las mecánicas impartidas en clase, integrando lógica de colisiones, físicas, interfaz básica y retroalimentación audiovisual.

**Antecedentes/Escenario**

Construct 2 permite estructurar un ciclo de juego mediante layouts para menús, niveles y pantallas finales. Integrar estos elementos forma un flujo completo de experiencia para el jugador, requiriendo coherencia en la progresión y consistencia en la lógica del proyecto.

**Recursos necesarios**

Sprites para jugador, plataformas, enemigos y objetos.

Sistema de eventos previamente trabajado.

Archivos de audio opcionales.

**Planteamiento del problema**

El desafío consiste en consolidar todos los elementos trabajados previamente para conformar un juego funcional que represente un flujo de principio a fin con reglas claras y mecánicas coherentes.

**Pasos por realizar**

Paso 1: Crear un layout para el menú inicial con opciones de inicio.

Paso 2: Construir el nivel completo utilizando tiles y objetos interactivos.

Paso 3: Definir las condiciones de victoria y derrota.

Paso 4: Crear un layout final que muestre un mensaje según el resultado.

Paso 5: Integrar eventos que conecten los layouts entre sí.

Paso 6: Probar el juego, detectar errores y corregir inconsistencias.

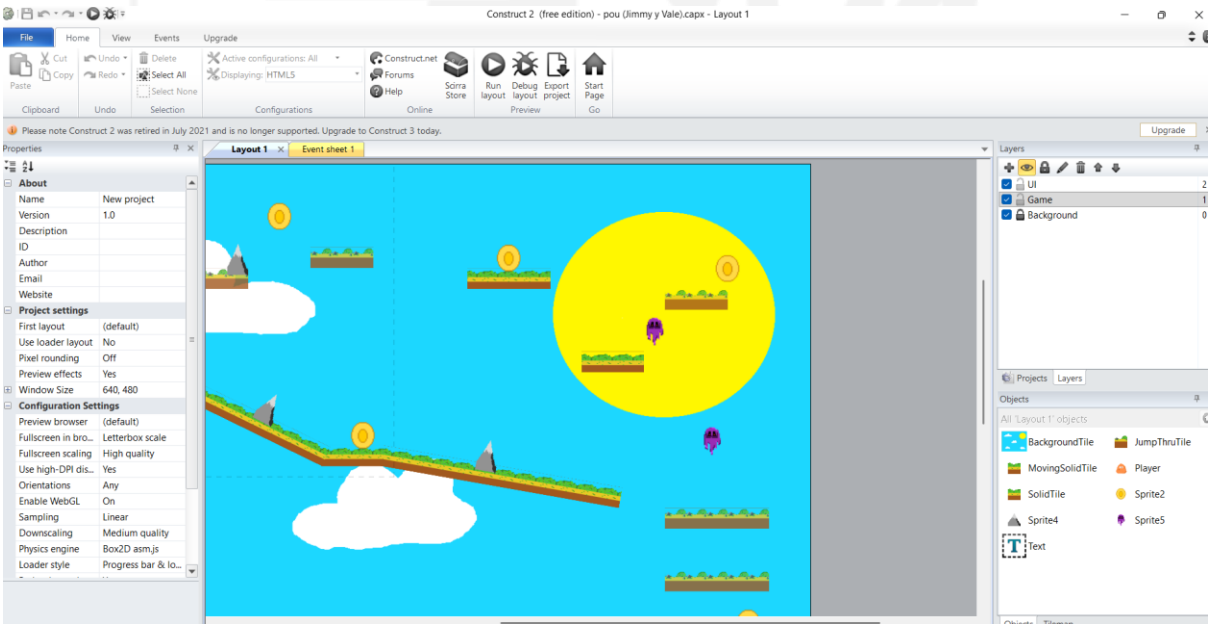
**Desarrollo**

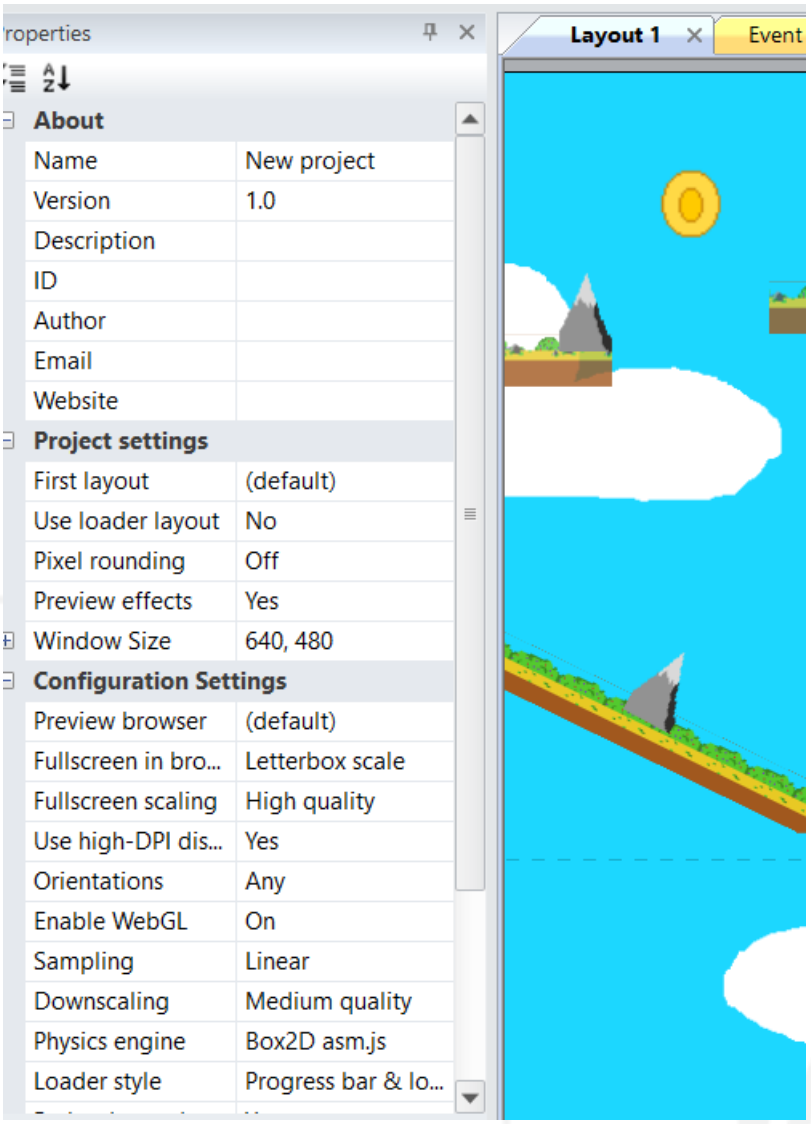
Se crea un layout titulado Menu con un sprite representando el botón “Jugar”. Se programa un evento basado en Touch o Mouse que, al hacer clic sobre él, redireccione al layout Nivel1. En Nivel1 se ensamblan plataformas, obstáculos y enemigos. El jugador se posiciona en el punto inicial del recorrido. Se integran elementos como ítems coleccionables, plataformas móviles y enemigos con rutas predefinidas.

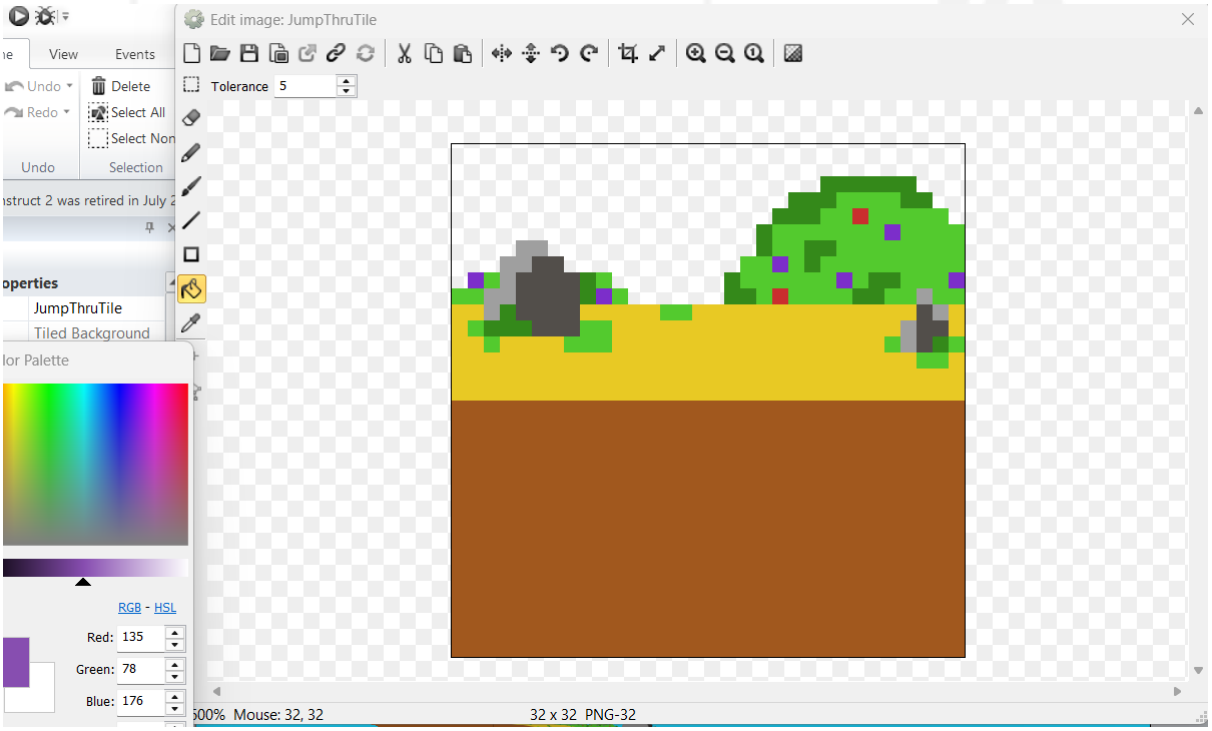
En los eventos se añaden condiciones de derrota, como caer fuera del mapa o recibir daño de un enemigo. Se programa una condición de victoria al alcanzar un objeto específico o una zona de meta, utilizando la condición On collision between Player and Meta.

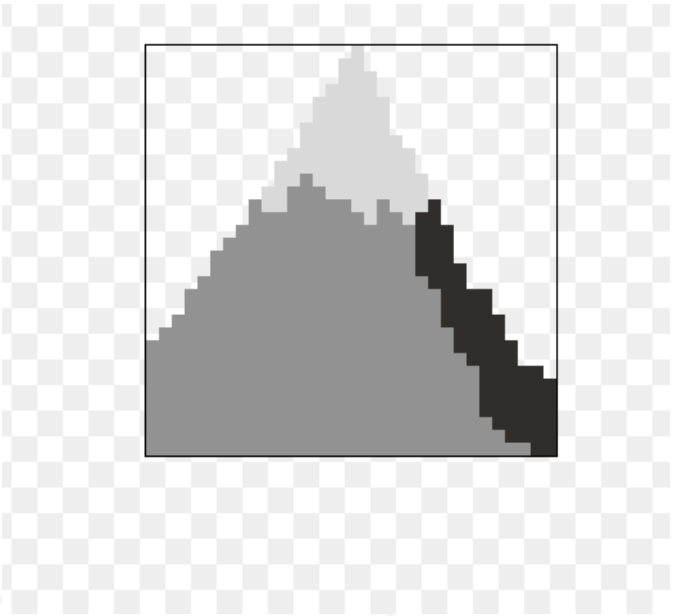
Se crea el layout Final con un mensaje que variará dependiendo de una variable global, como ResultadoJuego, que toma valor 1 para victoria y 0 para derrota.

Finalmente se realizan pruebas consecutivas para ajustar tiempos, gravedad, velocidad y posiciones, garantizando un ciclo de juego estable.




roperties
About
Name
New project
Version
1.0
Description
ID
Author
Email
Website
Project settings
First layout
(default)
Use loader layout
No
Pixel rounding
Off
Preview effects
Yes
Window Size
640, 480
Configuration Settings
Preview browser
(default)
Fullscreen in bro...
Letterbox scale
Fullscreen scaling
High quality
Use high-DPI dis...
Yes
Orientations
Any
Enable WebGL
On
Sampling
Linear
Downscaling
Medium quality
Physics engine
Box2D asm.js
Loader style
Progress bar & lo...
Layout 1
Event

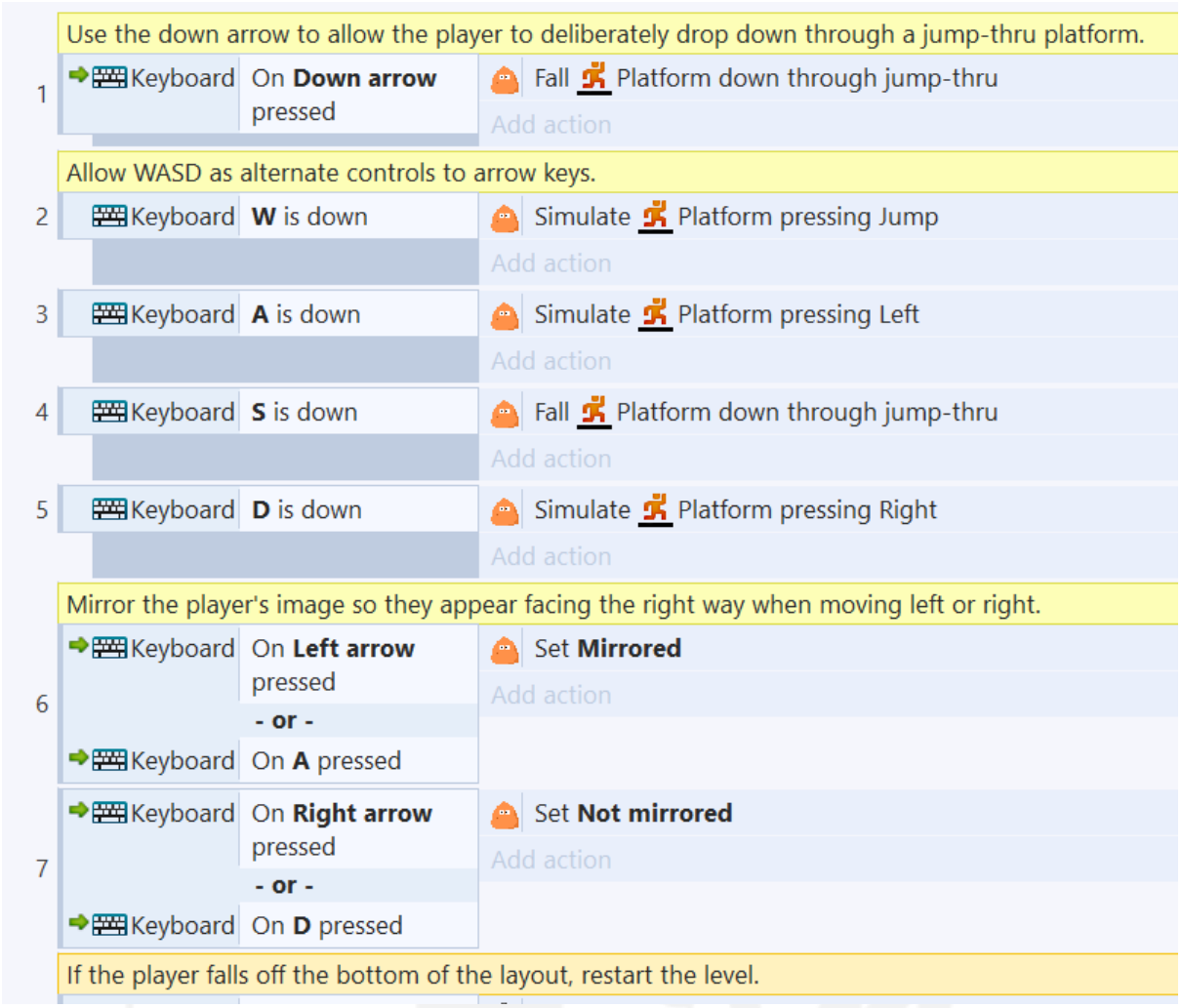

Edit image: JumpThruTile
Tolerance 5
JumpThruTile
Tiled Background
RGB - HSL
Red: 135
Green: 78
Blue: 176
Mouse: 32, 32
32 x 32 PNG-32



If the player falls off the bottom of the layout, restart the level.

| | | | |
|---|---|---|---|
| 8 | Player | Y > LayoutHeight | Restart layout |
| | | | Add action |
| 9 | | | Add action |
| 10 | Player | On collision with **Sprite2** | Destroy |
| | | | Add action |
| 11 | | | Add action |
| 12 | Player | On collision with **Sprite4** | Set position to (*147.777*, *684*) |
| | | | Add action |
| 13 | Player | On collision with **Sprite5** | Set position to (*147.777*, *684*) |
| | | | Add action |

Add event




**Resultados esperados**

El estudiante consolidará todos los conocimientos previos de Construct 2 para producir un juego completamente funcional con un flujo coherente desde el menú inicial hasta la pantalla final. Se espera que implemente correctamente la navegación entre layouts, un nivel jugable estructurado con plataformas, enemigos e ítems, así como condiciones claras de victoria y derrota basadas en eventos de colisión o estado del jugador.

El prototipo final deberá mostrar estabilidad en su lógica, con colisiones correctamente configuradas, animaciones funcionando según el comportamiento asignado y retroalimentación visual o sonora que acompañe las acciones principales. Además, el estudiante demostrará capacidad para depurar errores, ajustar parámetros de movimiento y asegurar que la progresión del juego sea fluida y comprensible para el jugador. El resultado previsto es un juego completo, consistente y jugable de inicio a fin, integrando todos los elementos fundamentales de un proyecto de plataformas.

### 3.2 Practica experimental 5

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Mapa visual de la interfaz de Unreal Engine con etiquetas de funciones**

**Objetivos**

Identificar cada sección del editor de Unreal Engine mediante un mapa visual que señale sus componentes, funciones principales y relación con el flujo de trabajo.

**Antecedentes/Escenario**

El editor de Unreal Engine integra numerosas ventanas y herramientas como el viewport, el outliner, el content browser y el details panel. Conocer su ubicación y propósito facilita la navegación y mejora la productividad.

**Recursos necesarios**

Unreal Engine 5.

Herramienta para anotación de imágenes.

**Planteamiento del problema**

Se requiere analizar visualmente la interfaz y generar una interpretación gráfica clara que facilite la comprensión del flujo de trabajo.

**Pasos por realizar**

Paso 1: Abrir Unreal Engine y cargar un proyecto vacío.

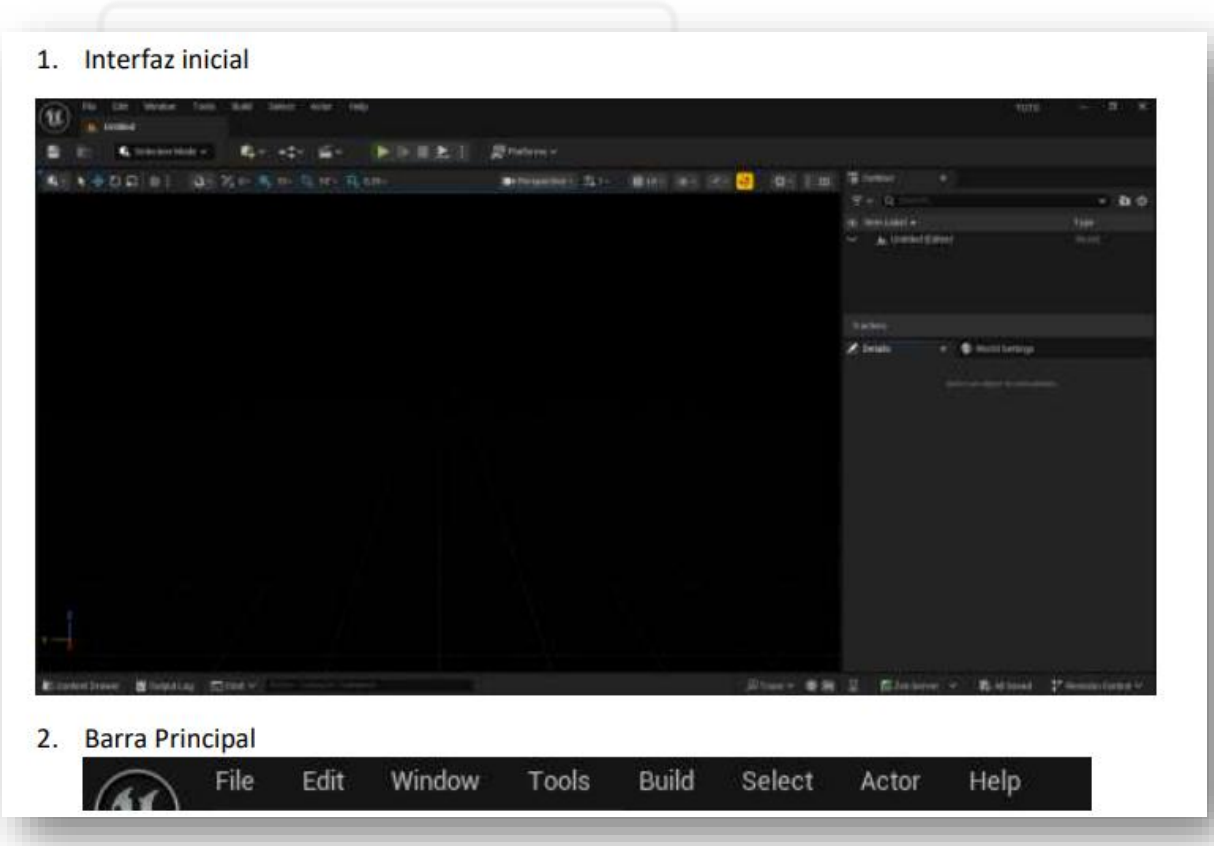
Paso 2: Capturar la interfaz completa desde el menú principal.

Paso 3: Identificar áreas fundamentales como el viewport, panel de contenido.

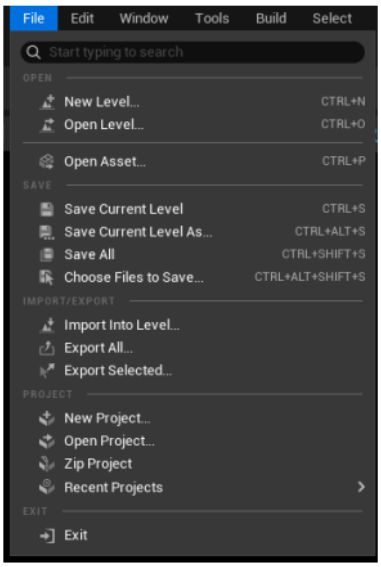
Paso 4: Añadir etiquetas descriptivas indicando la función de cada elemento.

**Desarrollo**

Se abre un proyecto vacío y se observa el editor completo. Se toma una captura de pantalla panorámica de la interfaz. Con la captura abierta en una herramienta de edición se insertan recuadros o flechas señalando áreas clave. El viewport se describe como la ventana en la que se previsualiza el mundo. El outliner se identifica como la lista jerárquica de los objetos presentes en la escena. El details panel muestra las propiedades específicas del objeto seleccionado. El content browser es la biblioteca donde se almacenan modelos, materiales, texturas, blueprints y más. Cada etiqueta se acompaña de una breve descripción operacional: qué permite hacer, cuándo se utiliza y cómo interactúa con las demás partes del editor.

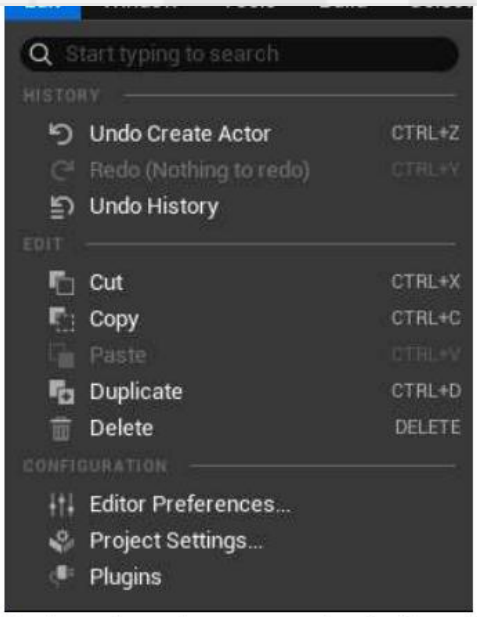


2.1. File




Opciones de guardado, creación de nuevos niveles, importación y exportación del proyecto, creación de nuevos proyectos y salida.

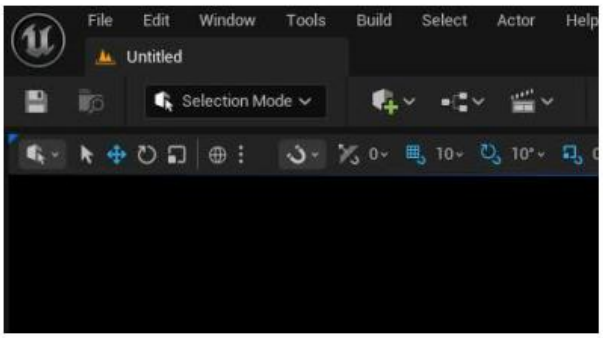
2.2. Edit




Opciones de copiar, pegar, cortar, duplicar, eliminar y rehacer.

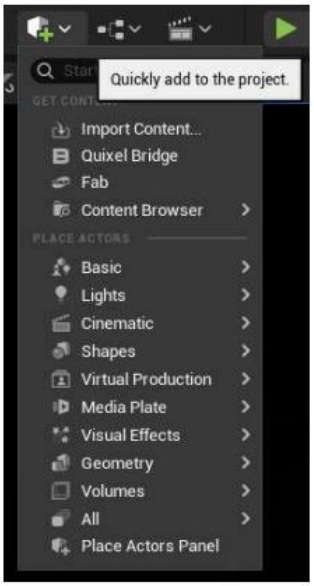
Acceder a la configuración del editor o sea del motor y del proyecto asi también como poder agregar o activar plugins.

3. Ventana gráfica



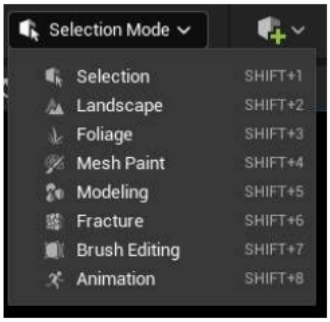

3.1. Botón de guardado

3.2. Agregar content



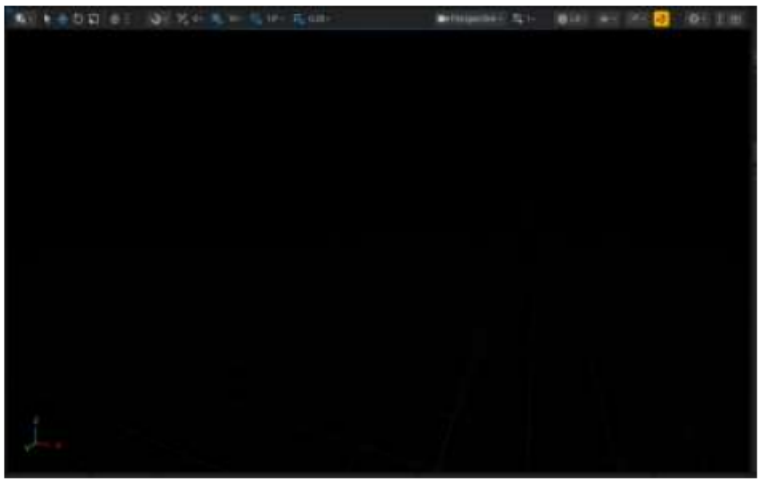

En este apartado se pueden agregar diversos elementos al proyecto como luces, elementos geométricos, volúmenes, cámaras, caracteres interactivos, etc.

3.3. Selection mode



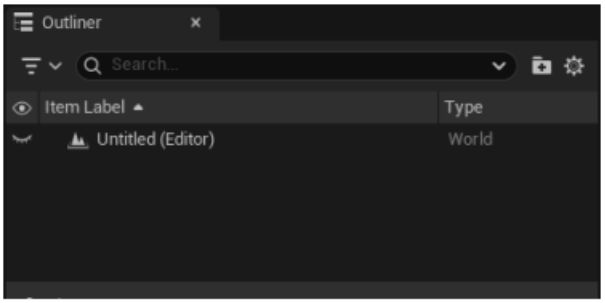

**Selection** es el modo predeterminado, permite la selección y movimiento de objetos en el espacio.
**Landscape** permite la creación de paisajes a través de la modificación de planos, pudiendo crear entornos como montañas, desiertos y otros y la libre asignación de materiales.
**Foliage** es un modo en el que se puede implementar y aplicar pasto a gran escala en un espacio/plano mediante diversos métodos como pintado o selección libre.
**Mesh** Paint ofrece opciones relacionado con la pintura de la malla ya sea para asignar pesos, cambiar colores y aplicar texturas, todos estos dependiendo de la cantidad de vértices.
**Modeling** es el modo que más opciones brinda, desde crear elementos tridimensionales y formas complejas, modificar coordenadas y hitbox y principalmente modificar y arreglar colisiones.

4. Viewport Editor



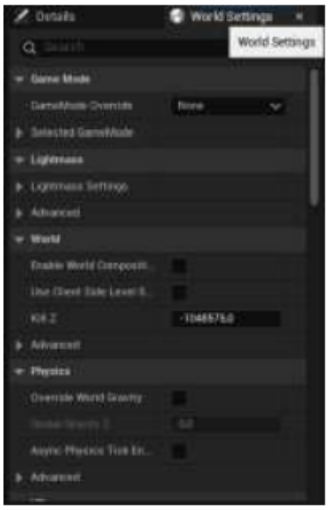
Permite la visualización de todo el proyecto y sus cambios en un plano tridimensional

5. Outliner




Visualización y enlistado del contenido aplicado en el espacio 3D o 2D

6. World Settings




Permite cambios en el world, especialmente verifica la aplicación de los Game Mods

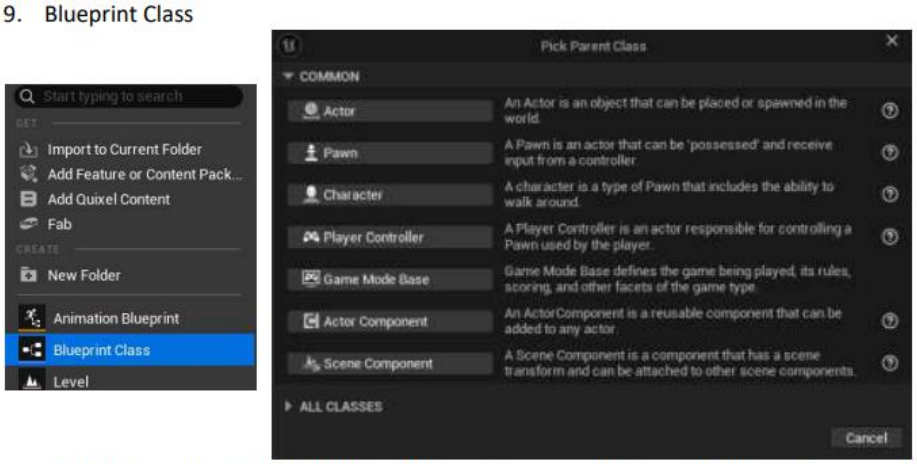
7. Details


Permite la modificación de los diversos elementos aplicados en el proyecto, permite cambios de parámetros como locación, escala y rotación, así también como cambios de mayas, iluminación y materiales.

8. Content Drawer

Muestra todo el contenido disponible para implementarlo en el proyecto asi también permite agregar archivos locales.

9. Blueprint Class




Permite la programación visual de diversos elementos que son fundamentales para el funcionamiento del proyecto, como personajes, elementos interactivos, obstáculos, cámaras, colisiones, mecánicas, animaciones, etc.

9.1. Ventana de edición/Blueprint

**Resultados esperados**

El estudiante identificará con claridad las áreas principales del editor de Unreal Engine y comprenderá la función específica de cada una dentro del flujo de trabajo. Se espera que el mapa visual elaborado muestre correctamente el viewport, el outliner, el details panel y el content browser con etiquetas precisas. El resultado debe reflejar una comprensión estructural que facilite la navegación, organización de activos y edición de escenas dentro del motor.

### 3.3 Practica experimental 6

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Construcción del nivel base 2D o 3D en Unreal Engine**

**Objetivos**

Construir el nivel inicial de un juego mediante la integración de un personaje, entorno básico y controles funcionales que permitan moverse dentro del espacio.

**Antecedentes/Escenario**

Un nivel base funciona como la plataforma fundacional sobre la cual se desarrollarán mecánicas, enemigos, interfaces y narrativa. Unreal Engine proporciona herramientas para generar escenarios, importar modelos y configurar movimiento.

**Recursos necesarios**

Unreal Engine 5.

Plantilla 2D o 3D.

Modelos básicos o geometría BSP.

**Planteamiento del problema**

Es necesario establecer una escena donde el jugador pueda interactuar adecuadamente con el entorno sin errores de colisión o movimiento.

**Pasos por realizar**

Paso 1: Crear un nuevo proyecto según el formato escogido (2D o 3D).

Paso 2: Añadir un personaje existente de la plantilla o importar uno.

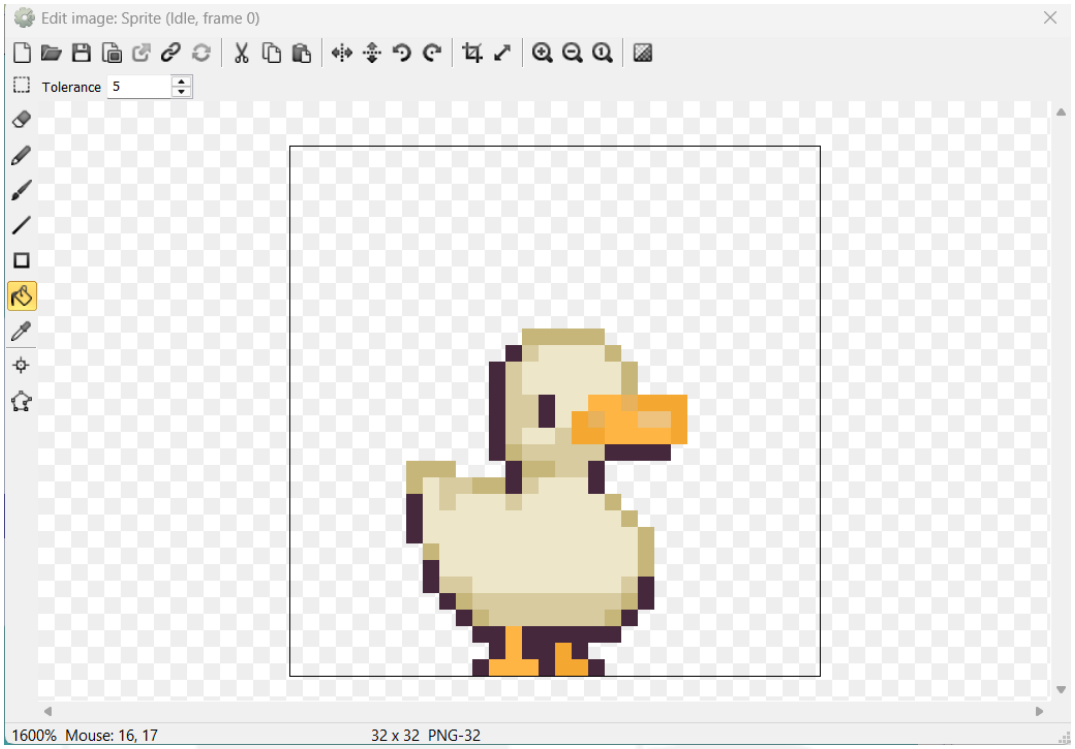
Paso 3: Crear el escenario básico empleando geometría simple o landscape.

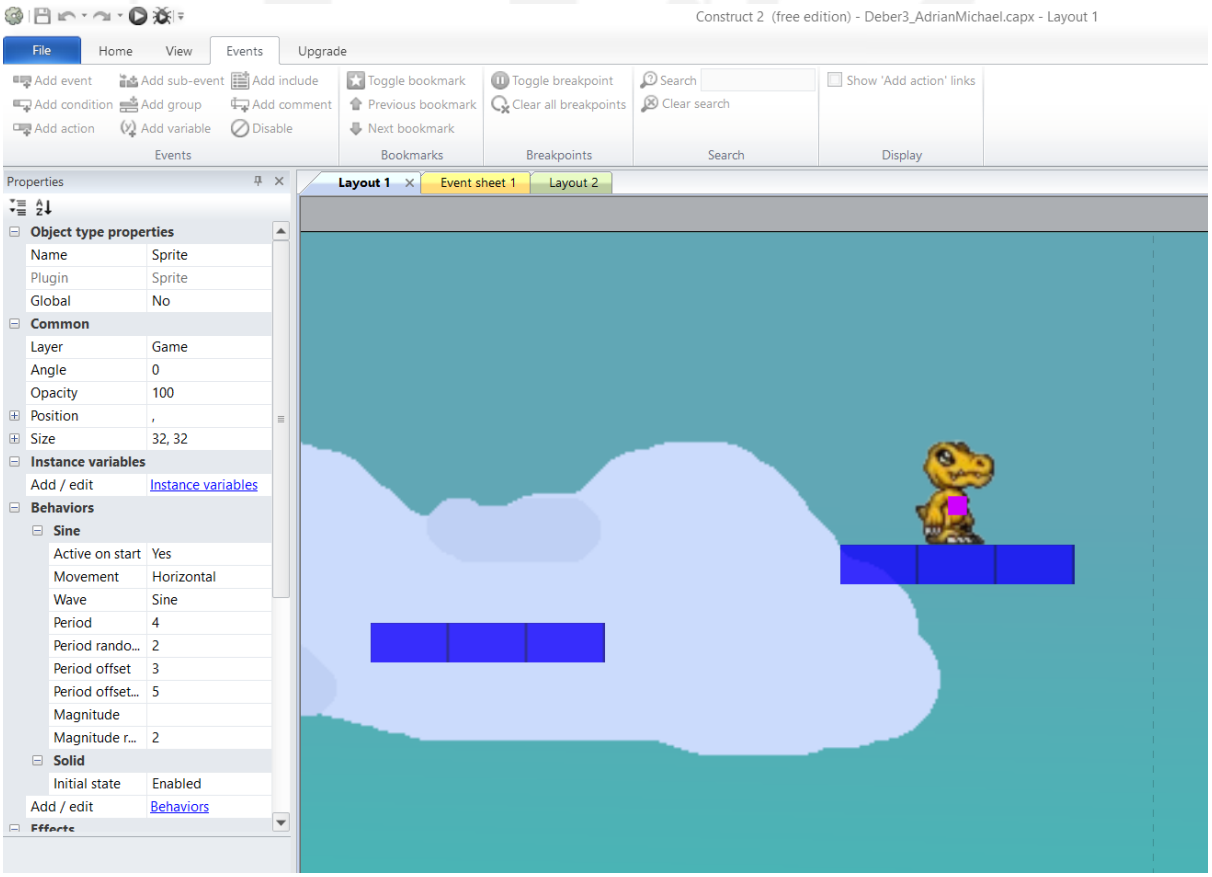
Paso 4: Configurar colisiones en los elementos del entorno.

Paso 5: Probar movimiento, colisiones y ajustes de cámara.

**Desarrollo**

El proyecto se crea seleccionando la plantilla adecuada. Si es 3D, se utiliza Third Person para obtener un modelo con animaciones y sistema de cámara. Se carga el mapa predeterminado y se reemplaza la geometría inicial con otros elementos si se desea. Un personaje principal es colocado en el escenario asegurando que la cápsula de colisión esté correctamente alineada. Se crean paredes, plataformas o pisos usando geometría. Se ajustan colisiones verificando que cada superficie tenga un preset BlockAll. Finalmente se ejecuta el juego para comprobar que el personaje se mueva correctamente, salte, colisione y sea seguido por la cámara. Cualquier error se corrige reubicando objetos o ajustando parámetros de movimiento.


Edit image: Sprite (Idle, frame 0)
Tolerance 5
1600% Mouse: 16, 17
32 x 32 PNG-32


Construct 2 (free edition) - Deber3_AdrianMichael.capx - Layout 1
File
Home
View
Events
Upgrade
Add event
Add sub-event
Add include
Add condition
Add group
Add comment
Add action
Add variable
Disable
Events
Toggle bookmark
Previous bookmark
Next bookmark
Bookmarks
Toggle breakpoint
Clear all breakpoints
Breakpoints
Search
Clear search
Search
Show 'Add action' links
Display
Properties
Object type properties
Name Sprite
Plugin Sprite
Global No
Common
Layer Game
Angle 0
Opacity 100
Position ,
Size 32, 32
Instance variables
Add / edit Instance variables
Behaviors
Sine
Active on start Yes
Movement Horizontal
Wave Sine
Period 4
Period rando... 2
Period offset 3
Period offset... 5
Magnitude
Magnitude r... 2
Solid
Initial state Enabled
Add / edit Behaviors
Layout 1
Event sheet 1
Layout 2

Use the down arrow to allow the player to deliberately drop down through a jump-thru platform.

1 — Keyboard: On **Down arrow** pressed → Fall Platform down through jump-thru; Add action

Allow WASD as alternate controls to arrow keys.

2 — Keyboard: **W** is down → Simulate Platform pressing Jump; Add action

3 — Keyboard: **A** is down → Simulate Platform pressing Left; Add action

4 — Keyboard: **S** is down → Fall Platform down through jump-thru; Add action

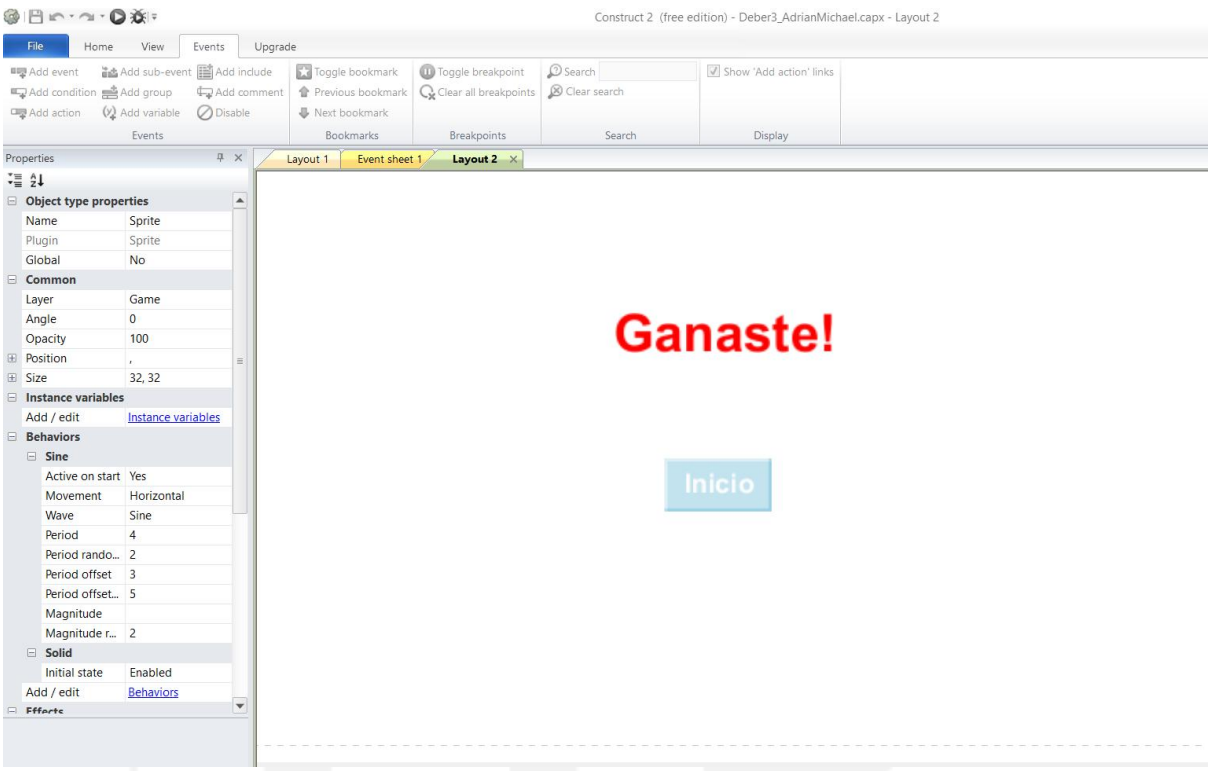
5 — Keyboard: **D** is down → Simulate Platform pressing Right; Add action

Mirror the player's image so they appear facing the right way when moving left or right.

6 — Keyboard: On **Left arrow** pressed **- or -** Keyboard: On **A** pressed → Set **Mirrored**; Add action

7 — Keyboard: On **Right arrow** pressed **- or -** → Set **Not mirrored**; Add action

| | | | |
|---|---|---|---|
| 8 | Player | On collision with **Colision_d** | Destroy<br>Wait **1** seconds<br>Spawn **Player** on layer **0** *(image point 0)*<br>Add action |
| 9 | Mouse | On **Left** button **Clicked** on **Boton** | Set animation frame to **1**<br>Wait **1** seconds<br>Go to **Layout 1**<br>Add action |
| 10 | Player | On collision with **Fin** | Destroy<br>Spawn **Particles** on layer **0** *(image point 0)*<br>Wait **1** seconds<br>Go to **Layout 2**<br>Add action |
| 11 | Player | On collision with **Sprite** | Destroy<br>Spawn **Particles** on layer **0** *(image point 0)*<br>Wait **1** seconds<br>Spawn **Player** on layer **0** *(image point 0)*<br>Add action |
| 12 | Player | Platform is moving | Set animation to **"Caminar"** (play from beginning)<br>Add action |


Construct 2 (free edition) - Deber3_AdrianMichael.capx - Layout 2
File
Home
View
Events
Upgrade
Add event
Add sub-event
Add include
Add condition
Add group
Add comment
Add action
Add variable
Disable
Events
Toggle bookmark
Previous bookmark
Next bookmark
Bookmarks
Toggle breakpoint
Clear all breakpoints
Breakpoints
Search
Clear search
Search
Show 'Add action' links
Display
Properties
Object type properties
Name
Sprite
Plugin
Sprite
Global
No
Common
Layer
Game
Angle
0
Opacity
100
Position
Size
32, 32
Instance variables
Add / edit
Instance variables
Behaviors
Sine
Active on start
Yes
Movement
Horizontal
Wave
Sine
Period
4
Period rando...
2
Period offset
3
Period offset...
5
Magnitude
Magnitude r...
2
Solid
Initial state
Enabled
Add / edit
Behaviors
Effects
Layout 1
Event sheet 1
Layout 2
Ganaste!
Inicio


Arrow keys or WASD to move

**Resultados esperados**

El estudiante construirá un nivel base completamente funcional donde el personaje pueda desplazarse sin errores, interactuando con superficies que cuenten con colisiones correctas. Se espera un escenario estructurado con geometría simple o elementos 3D coherentes, una cámara operativa y un personaje configurado adecuadamente. El nivel debe permitir pruebas estables de movimiento, saltos y navegación, sirviendo como base para futuras mecánicas, diseño de gameplay y ampliaciones del proyecto.

# 4 UNIDAD 3: Programación Nivel 1

## 4.1 Practica experimental 7

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Importar e integrar sprites en Unreal Engine**

**Objetivos**

Importar sprites creados externamente al motor, configurarlos como texturas 2D y asignar colisiones básicas para su uso en proyectos.

**Antecedentes/Escenario**

Unreal Engine permite trabajar con sprites mediante el plugin Paper2D que convierte las texturas en objetos con colisión y animación.

**Recursos necesarios**

Sprites externos en formato PNG.

Proyecto con Paper2D habilitado.

**Planteamiento del problema**

Es necesario integrar sprites con colisiones configuradas y animaciones básicas dentro de un mapa funcional.

**Pasos por realizar**

Paso 1: Activar Paper2D en Plugins.

Paso 2: Importar archivos PNG al Content Browser.

Paso 3: Convertir cada textura en sprite.

Paso 4: Configurar colisiones según requerimientos.

Paso 5: Colocar los sprites en escena y validar su comportamiento.

**Desarrollo**

Desde la sección de plugins se habilita Paper2D. Se importan los sprites y se convierten en objetos sprite desde el menú contextual. Se edita su shape de colisión ajustando la forma para adaptarse a la geometría visual. Se crean flipbooks si se desea animación mediante varias imágenes. Finalmente se arrastran los sprites al nivel verificando que su tamaño sea consistente.

**Resultados esperados**

El estudiante incorpora sprites externos al proyecto utilizando Paper2D, asegurando que cada textura sea convertida correctamente y cuente con colisión funcional. Los sprites se integran al nivel con proporciones adecuadas y, cuando se requiere, se combinan en flipbooks para animaciones básicas. Al finalizar, deben colocarse en la escena y responder correctamente al motor, listos para formar parte de un entorno 2D completamente operativo.

## 4.2 Practica experimental 8

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Creación de un proyecto ejecutable en UE5 para Windows**

**Objetivos**

Configurar un proyecto para que pueda generar un archivo ejecutable .exe mediante las herramientas de empaquetado de Unreal Engine.

**Antecedentes/Escenario**

UE5 permite generar compilaciones ejecutables para múltiples plataformas. El proceso de empaquetado requiere configuraciones específicas en Project Settings.

**Recursos necesarios**

ProUnreal Engine 5.

SDKs de Windows instalados.

**Planteamiento del problema**

Se debe configurar el proyecto para empaquetarlo sin errores, ajustando parámetros como mapas de inicio, compilación y dependencias.

**Pasos por realizar**

Paso 1: Abrir Project Settings y configurar mapas predeterminados.

Paso 2: Ajustar la sección Packaging.

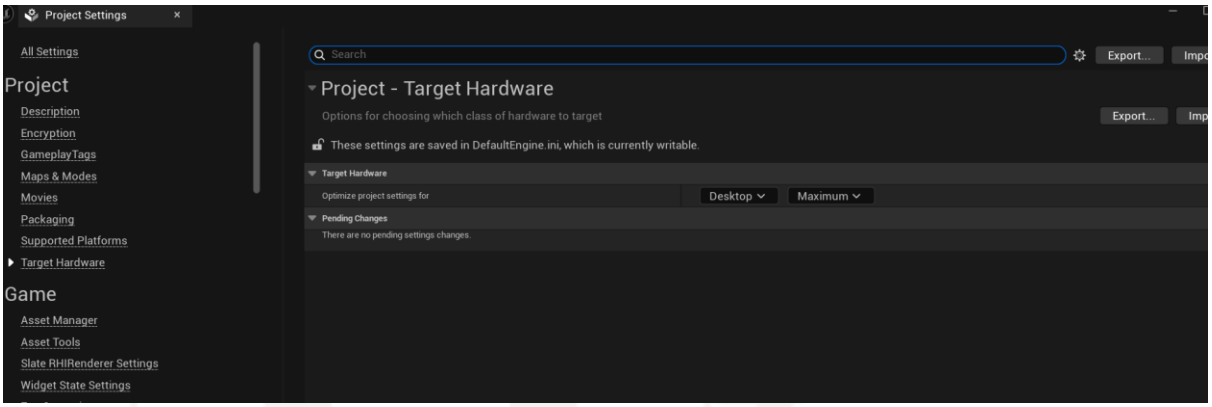
Paso 3: Verificar Plugins habilitados.

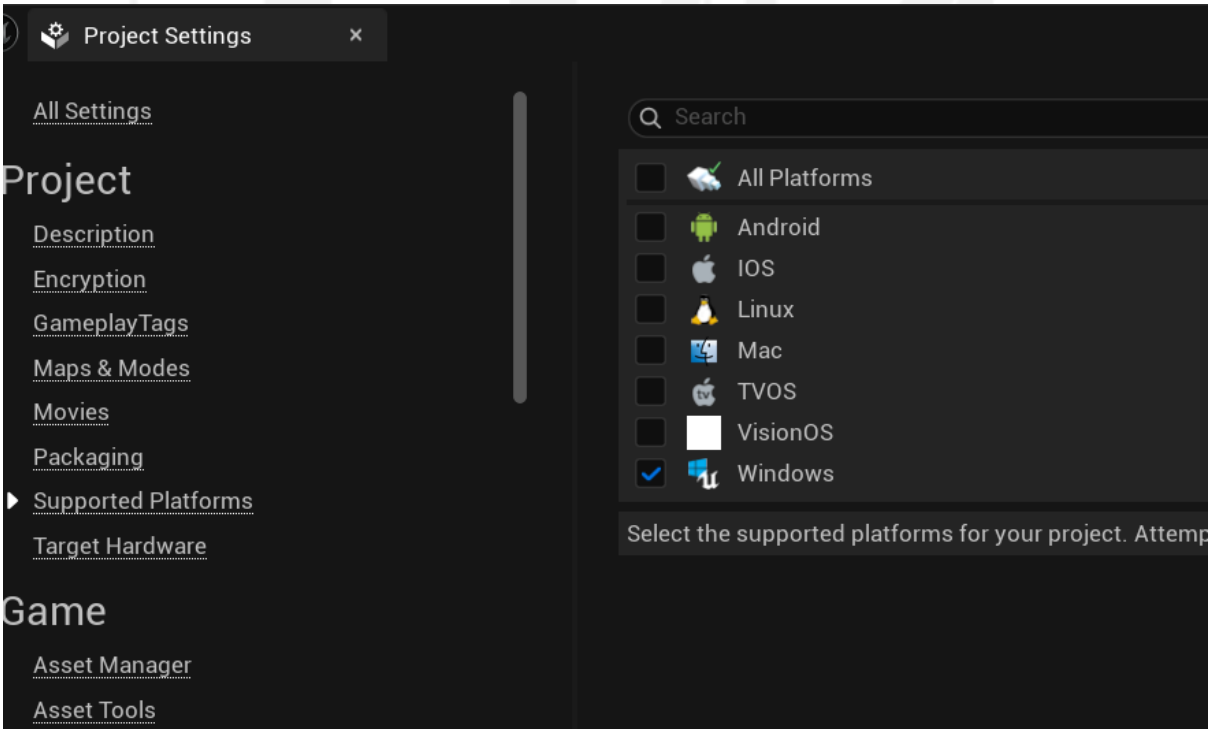
Paso 4: Empaquetar en File > Package Project > Windows.

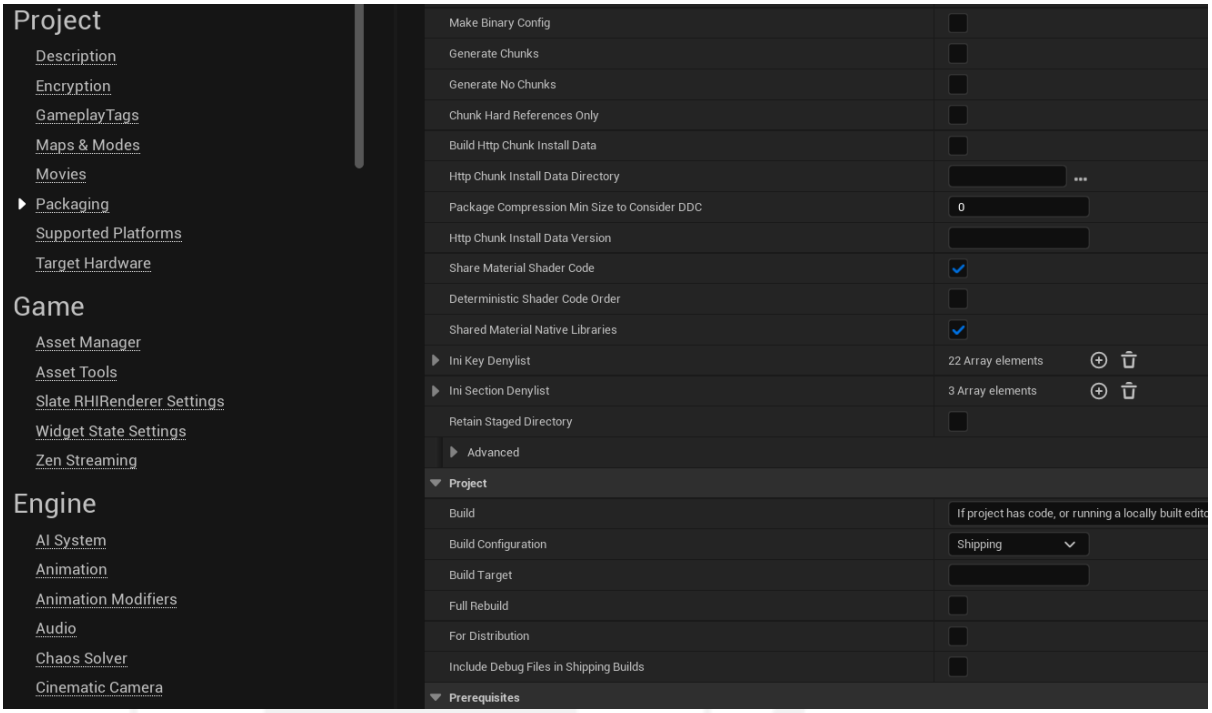
Paso 5: Ejecutar el archivo generado.

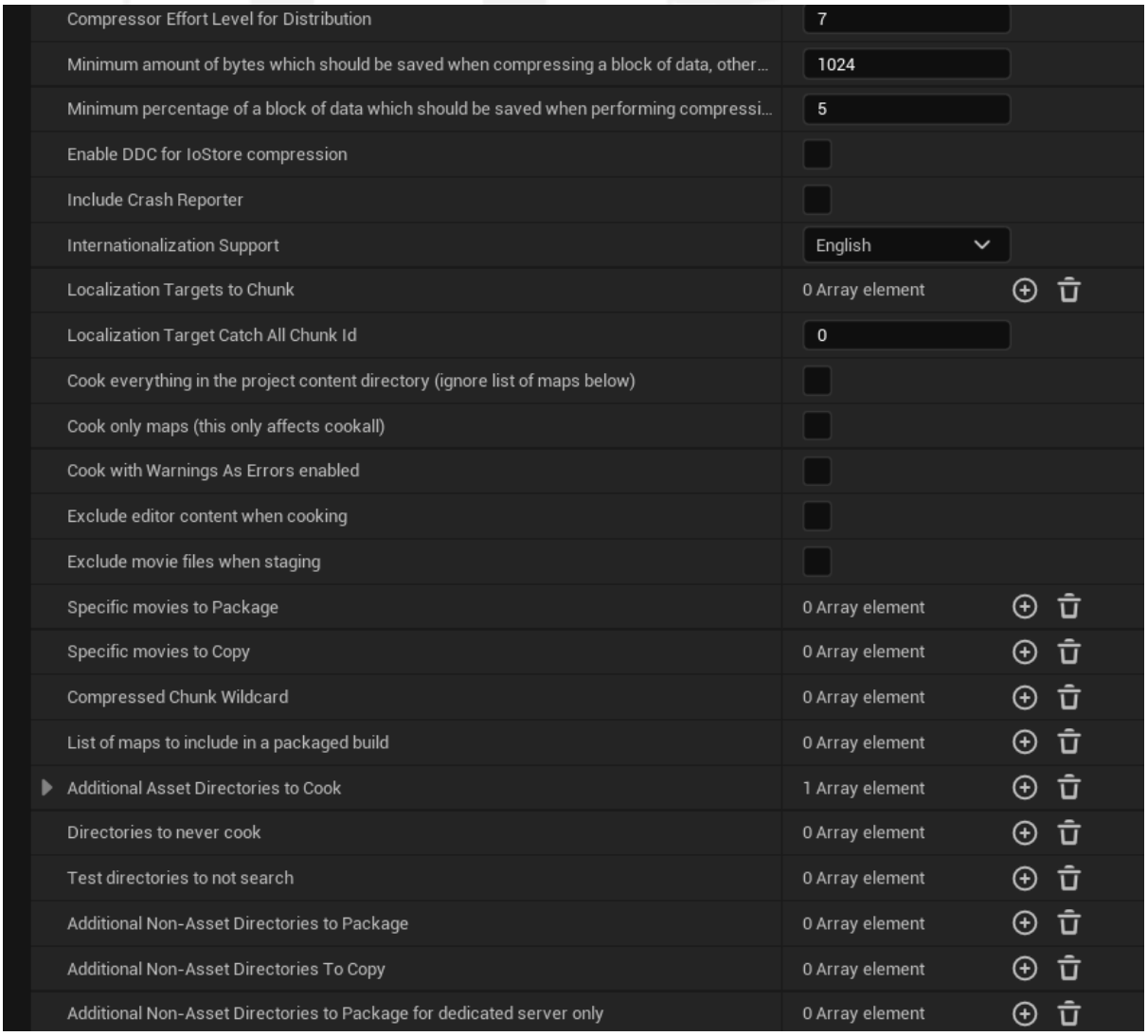
**Desarrollo**

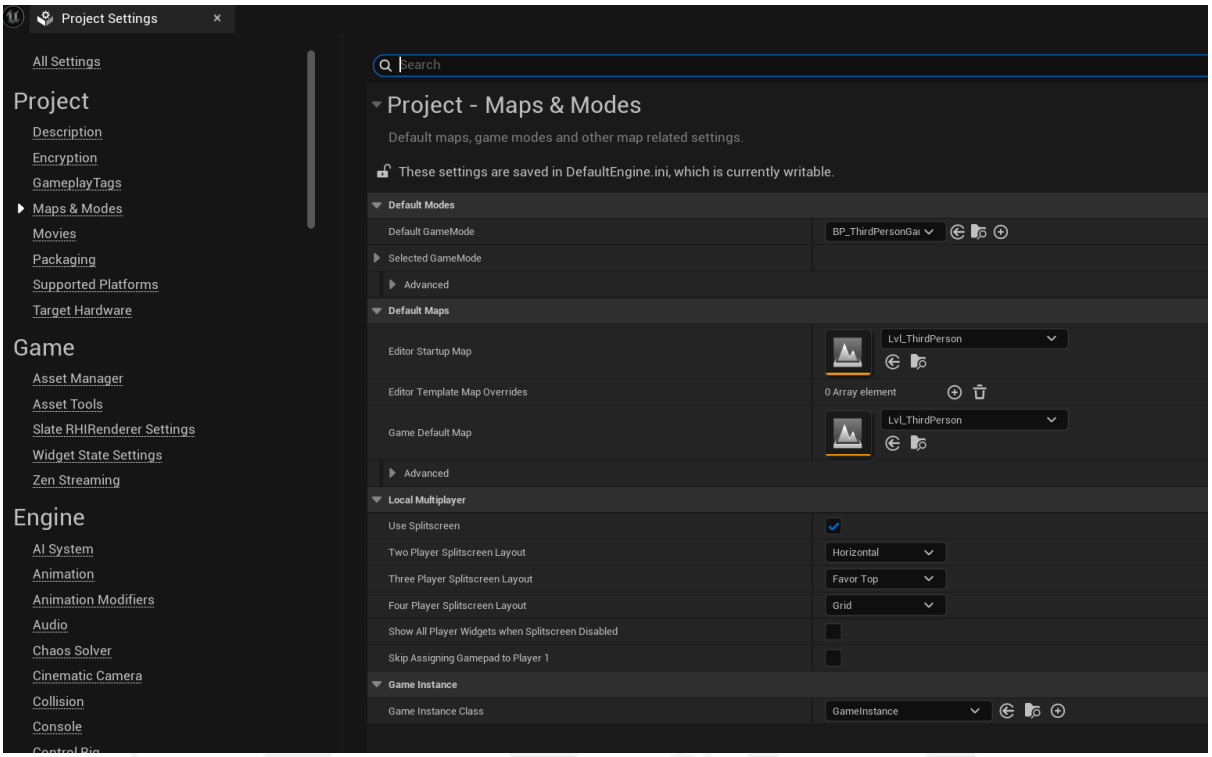
En Project Settings se define el mapa inicial y el mapa del editor. En Packaging se habilitan opciones como Full Rebuild y se desactivan archivos innecesarios. Se comprueba que no existan errores de referencias. Luego se selecciona la opción Package Project y se elige Windows. Una vez generado el ejecutable se prueba su funcionamiento.



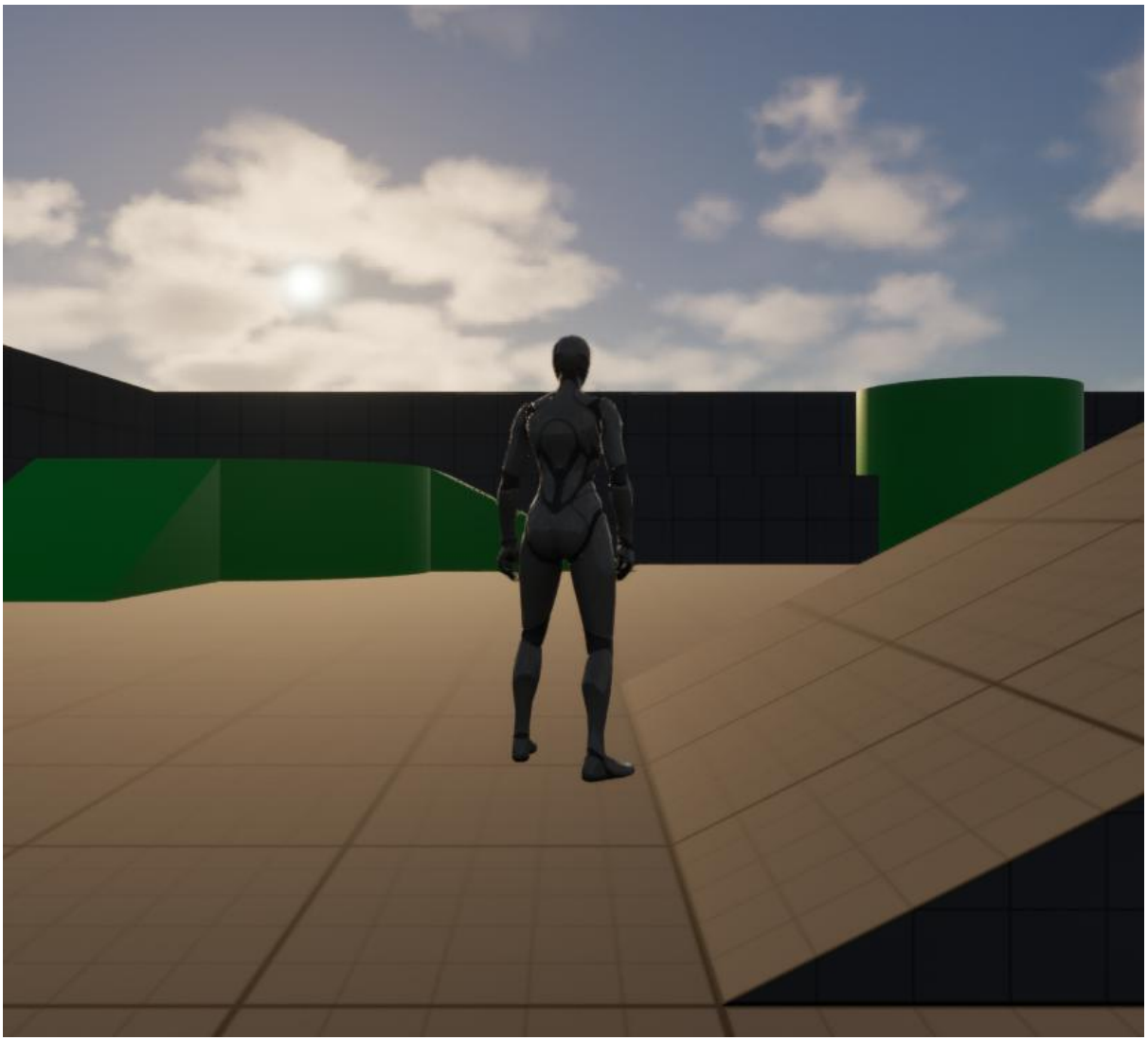




Project
Description
Encryption
GameplayTags
Maps & Modes
Movies
Packaging
Supported Platforms
Target Hardware
Game
Asset Manager
Asset Tools
Slate RHIRenderer Settings
Widget State Settings
Zen Streaming
Engine
AI System
Animation
Animation Modifiers
Audio
Chaos Solver
Cinematic Camera
Make Binary Config
Generate Chunks
Generate No Chunks
Chunk Hard References Only
Build Http Chunk Install Data
Http Chunk Install Data Directory
Package Compression Min Size to Consider DDC
0
Http Chunk Install Data Version
Share Material Shader Code
Deterministic Shader Code Order
Shared Material Native Libraries
Ini Key Denylist
22 Array elements
Ini Section Denylist
3 Array elements
Retain Staged Directory
Advanced
Project
Build
Build Configuration
Shipping
Build Target
Full Rebuild
For Distribution
Include Debug Files in Shipping Builds
Prerequisites


Compressor Effort Level for Distribution
7
Minimum amount of bytes which should be saved when compressing a block of data, other...
1024
Minimum percentage of a block of data which should be saved when performing compressi...
5
Enable DDC for IoStore compression
Include Crash Reporter
Internationalization Support
English
Localization Targets to Chunk
0 Array element
Localization Target Catch All Chunk Id
0
Cook everything in the project content directory (ignore list of maps below)
Cook only maps (this only affects cookall)
Cook with Warnings As Errors enabled
Exclude editor content when cooking
Exclude movie files when staging
Specific movies to Package
0 Array element
Specific movies to Copy
0 Array element
Compressed Chunk Wildcard
0 Array element
List of maps to include in a packaged build
0 Array element
Additional Asset Directories to Cook
1 Array element
Directories to never cook
0 Array element
Test directories to not search
0 Array element
Additional Non-Asset Directories to Package
0 Array element
Additional Non-Asset Directories To Copy
0 Array element
Additional Non-Asset Directories to Package for dedicated server only
0 Array element


Project Settings
All Settings
Project
Description
Encryption
GameplayTags
Maps & Modes
Movies
Packaging
Supported Platforms
Target Hardware
Game
Asset Manager
Asset Tools
Slate RHIRenderer Settings
Widget State Settings
Zen Streaming
Engine
AI System
Animation
Animation Modifiers
Audio
Chaos Solver
Cinematic Camera
Collision
Console
Search
Project - Maps & Modes
Default maps, game modes and other map related settings.
These settings are saved in DefaultEngine.ini, which is currently writable.
Default Modes
Default GameMode
BP_ThirdPersonGai
Selected GameMode
Advanced
Default Maps
Editor Startup Map
Lvl_ThirdPerson
Editor Template Map Overrides
0 Array element
Game Default Map
Lvl_ThirdPerson
Advanced
Local Multiplayer
Use Splitscreen
Two Player Splitscreen Layout
Horizontal
Three Player Splitscreen Layout
Favor Top
Four Player Splitscreen Layout
Grid
Show All Player Widgets when Splitscreen Disabled
Skip Assigning Gamepad to Player 1
Game Instance
Game Instance Class
GameInstance

| | | | |
|---|---|---|---|
| Engine | Carpeta de archivos | | |
| Proyecto_emp | Carpeta de archivos | | |
| Manifest_NonUFSFiles_Win64 | Documento de texto | 1 KB | No |
| Manifest_UFSFiles_Win64 | Documento de texto | 27 KB | No |
| Proyecto_emp | Aplicación | 89 KB | No |



**Resultados esperados**

El estudiante configura correctamente los ajustes de *Project Settings* para permitir el empaquetado del proyecto y genera un ejecutable funcional para Windows. Se establecen los mapas predeterminados, se ajustan las opciones de *Packaging* y se validan plugins y dependencias para evitar

errores durante la compilación. Finalmente, se obtiene un archivo .exe estable que inicia el juego fuera del editor y refleja fielmente el contenido desarrollado en Unreal Engine.

## 4.3 Practica experimental 9

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Implementación de Metahuman en UE5**

**Objetivos**

Integrar un personaje hiperrealista mediante Metahuman Creator y configurarlo dentro de un proyecto jugable.

**Antecedentes/Escenario**

Metahuman utiliza un sistema avanzado de mallas, texturas y rigs faciales. Su integración requiere el uso de Quixel Bridge.

**Recursos necesarios**

Cuenta de Epic Games.
Quixel Bridge.
Unreal Engine 5.

**Planteamiento del problema**

En la actualidad el aumento de creación de elementos 3D para la simulación de humanos ha crecido por esta razón es necesario aprender sobre las nuevas tecnologías que pueden ser aplicada y por esto se necesita tener la capacidad de importar un Metahuman funcional con su sistema de animación completo.

**Pasos por realizar**

Paso 1: Crear el Metahuman en la plataforma.

Paso 2: Descargarlo desde Quixel Bridge.

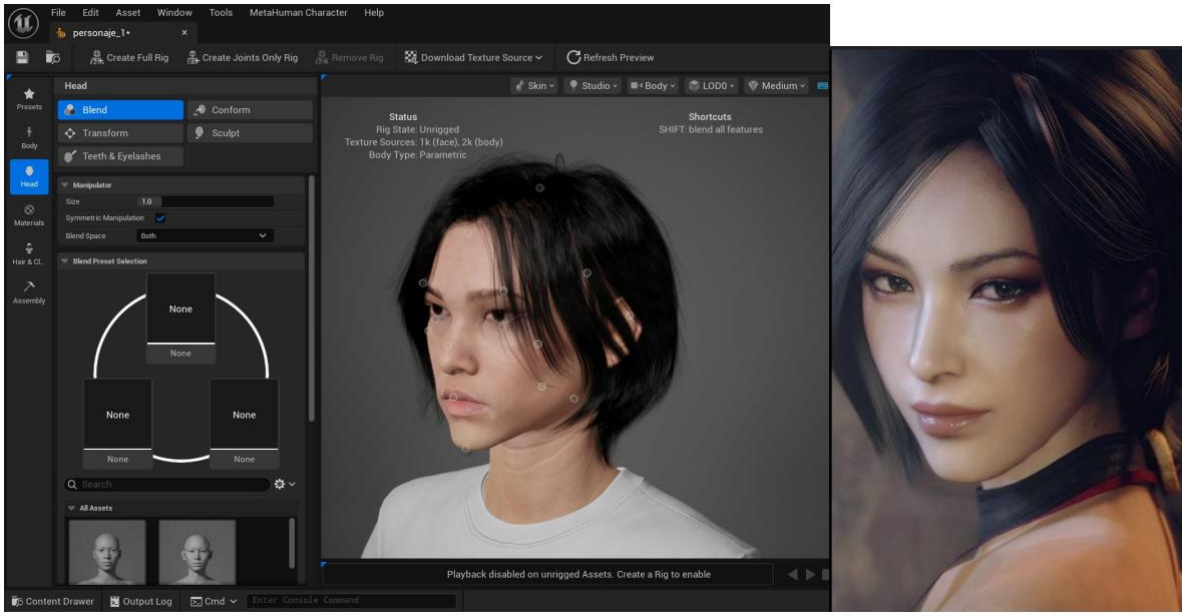
Paso 3: Importarlo al proyecto.

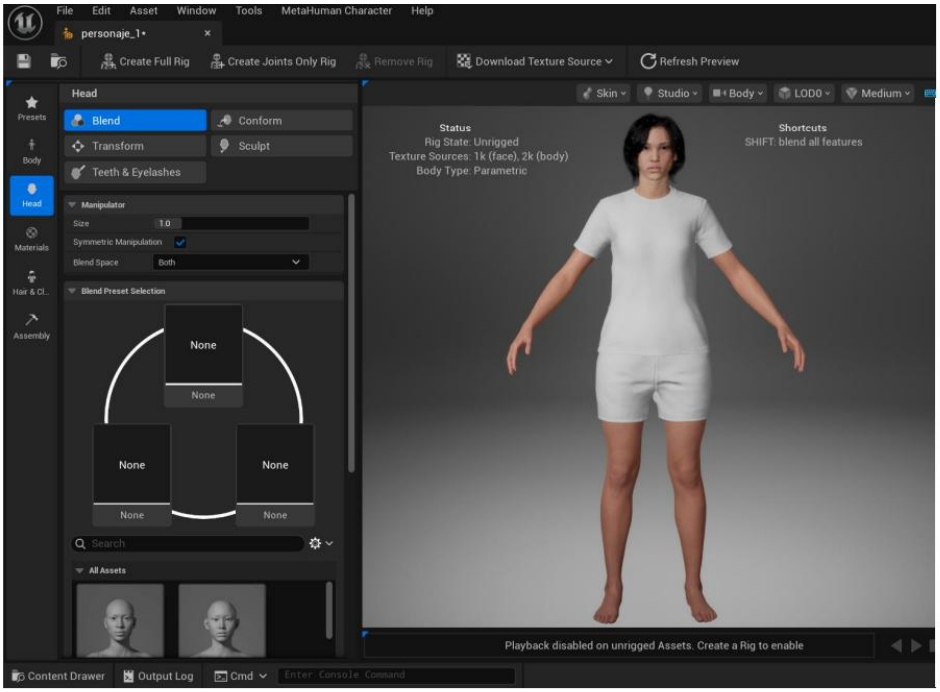
Paso 4: Revisar rig, materiales y blueprint.

Paso 5: Colocarlo en escena.

**Desarrollo**

Se crea un Metahuman seleccionando género, rostro y proporciones. Desde Bridge se descarga e importa al proyecto. El motor genera automáticamente su rig facial y esquelético. Se revisan sus materiales y se arrastra al nivel verificando colisiones y escala. Finalmente se configura un Blueprint para integrarlo con controles si se desea.




File
Edit
Asset
Window
Tools
MetaHuman Character
Help
personaje_1
Create Full Rig
Create Joints Only Rig
Remove Rig
Download Texture Source
Refresh Preview
Head
Blend
Conform
Transform
Sculpt
Teeth & Eyelashes
Manipulator
Size
Symmetric Manipulation
Blend Space
Both
Blend Preset Selection
None
None
None
None
None
None
Search
All Assets
Skin
Studio
Body
LOD0
Medium
Status
Rig State: Unrigged
Texture Sources: 1k (face), 2k (body)
Body Type: Parametric
Shortcuts
SHIFT: blend all features
Playback disabled on unrigged Assets. Create a Rig to enable
Content Drawer
Output Log
Cmd
Enter Console Command


**Resultados esperados**

El estudiante integra exitosamente un Metahuman completamente funcional dentro del proyecto, utilizando **Quixel Bridge** para su importación y verificando que el rig esquelético y facial se encuentren operativos. El personaje mantiene sus materiales avanzados, escala correcta y colisiones adecuadas. Asimismo, se valida su funcionamiento dentro del nivel, asegurando que el Blueprint asociado permita extender su uso para animación o control jugable según los requerimientos del proyecto.

# 5 UNIDAD 4: Programación Nivel 2

## 5.1 Practica experimental 10

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Importar y configurar modelo 3D en Unreal Engine**

**Objetivos**

Integrar un modelo 3D externo ajustando materiales, colisiones y escala para garantizar su funcionamiento en un entorno interactivo.

**Antecedentes/Escenario**

Modelos 3D suelen contener materiales y colisiones inconsistentes que deben ajustarse manualmente.

**Recursos necesarios**

Modelo en formato FBX.

Unreal Engine 5.

**Planteamiento del problema**

Un modelo mal importado puede generar errores de colisión, iluminación o textura.

**Pasos por realizar**

Paso 1: Importar el archivo FBX.

Paso 2: Revisar materiales y texturas.

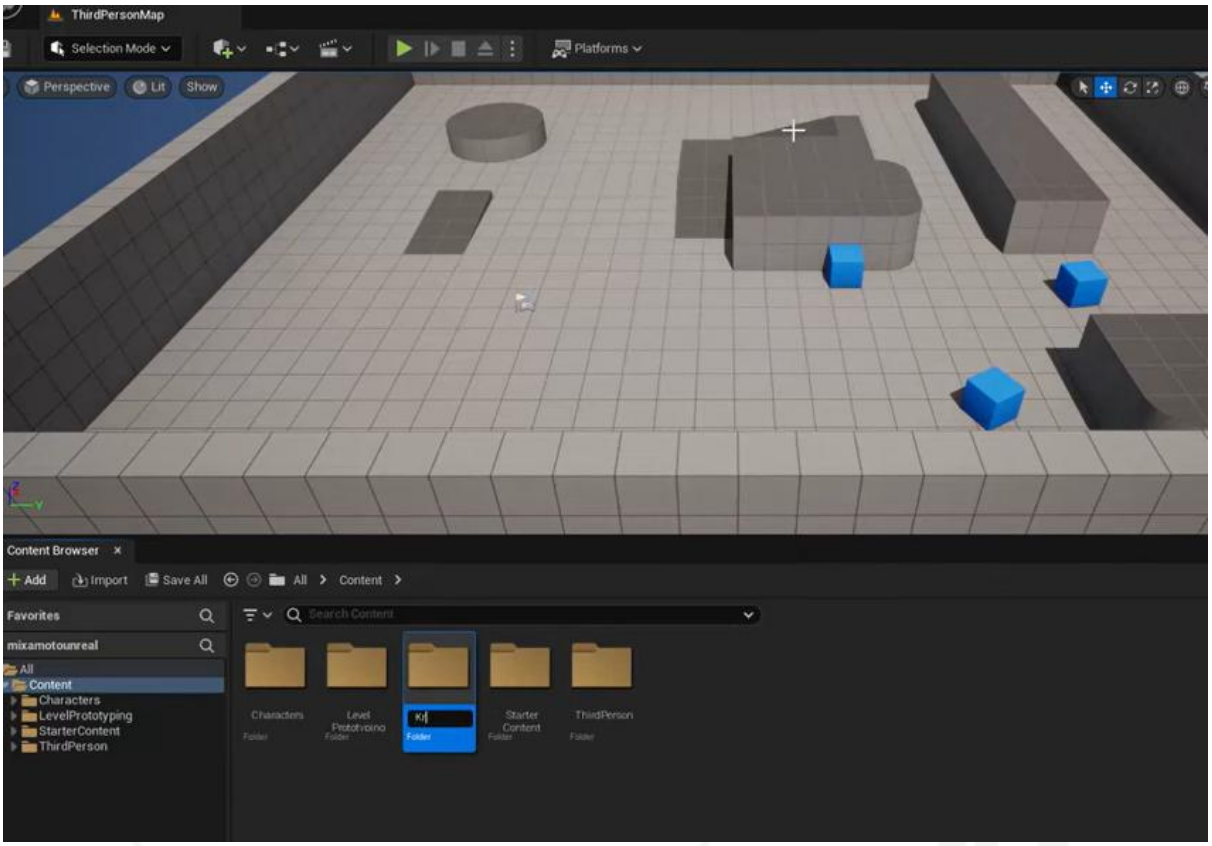
Paso 3: Ajustar colisiones desde Static Mesh Editor.

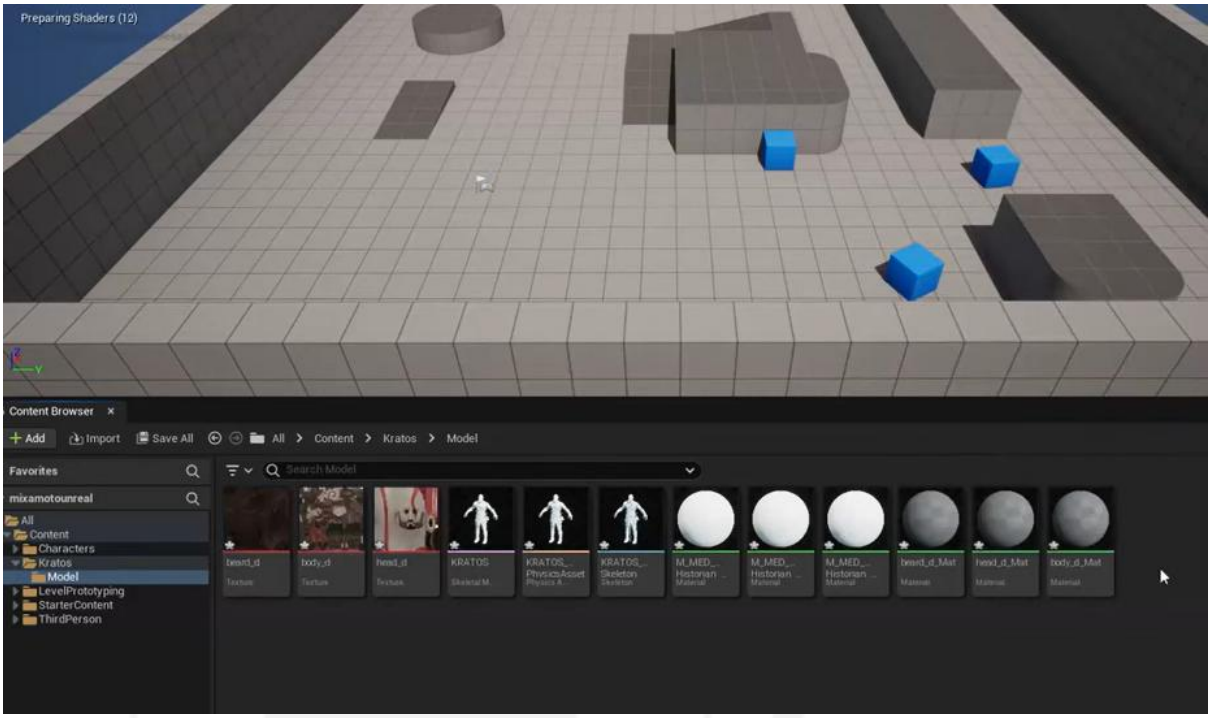
Paso 4: Ajustar escala.

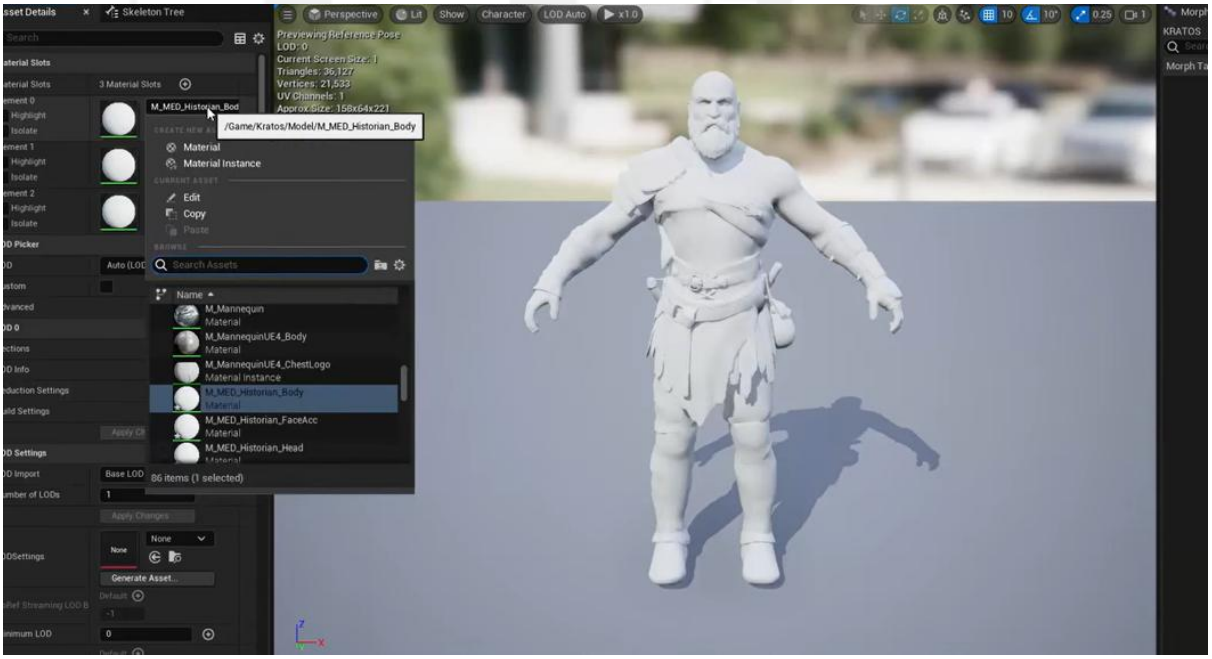
Paso 5: Colocar el modelo en la escena.

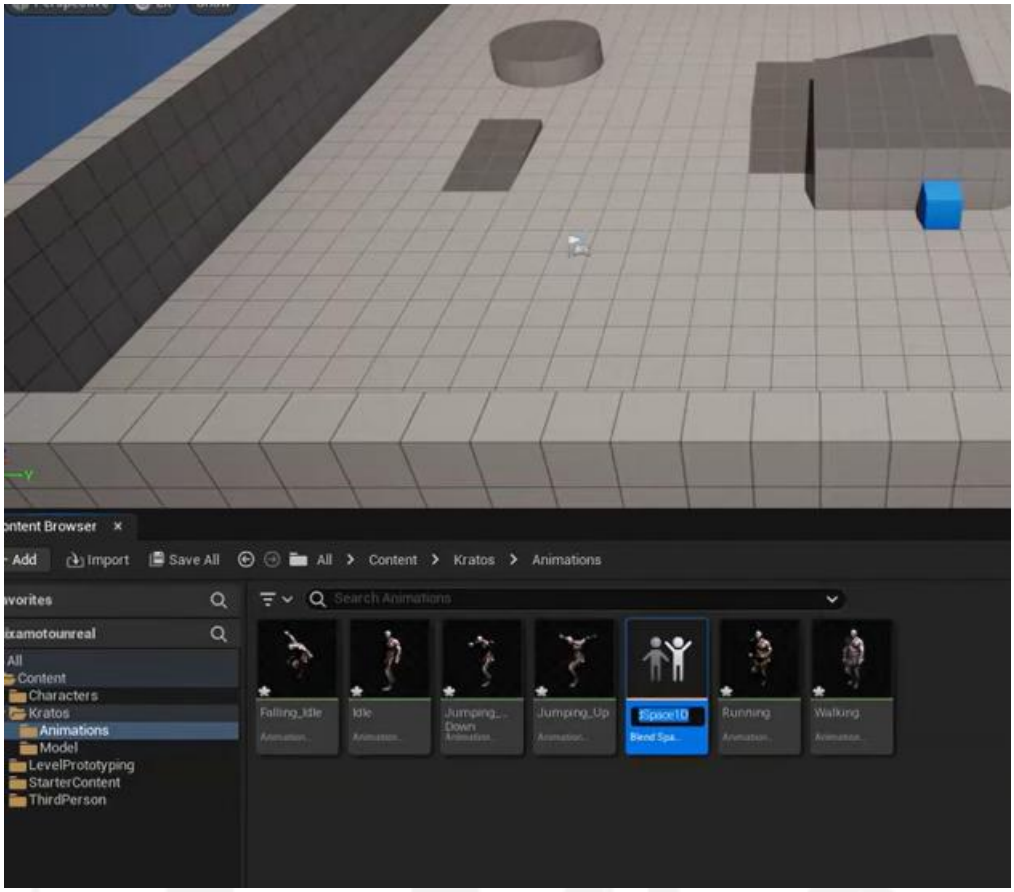
**Desarrollo**

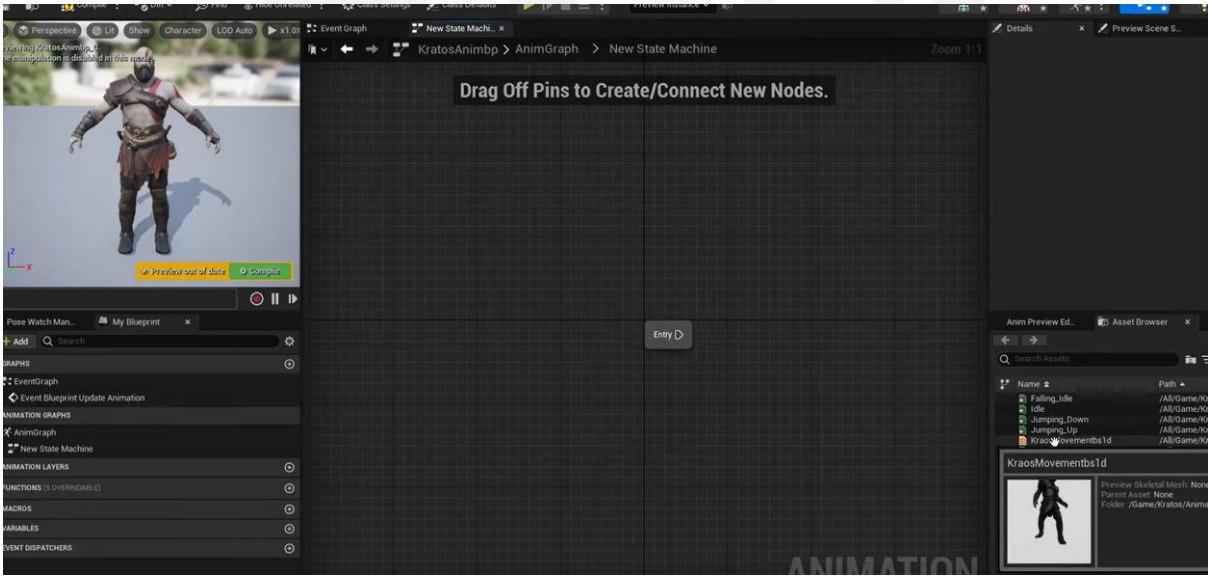
El modelo se importa verificando que la opción Import Materials esté activada. Luego se revisa cada material para verificar la configuración de Base Color, Roughness y Normal. Se genera una colisión automática o personalizada. Se ajusta la escala para que coincida con las proporciones del mundo y se posiciona el modelo dentro del nivel.

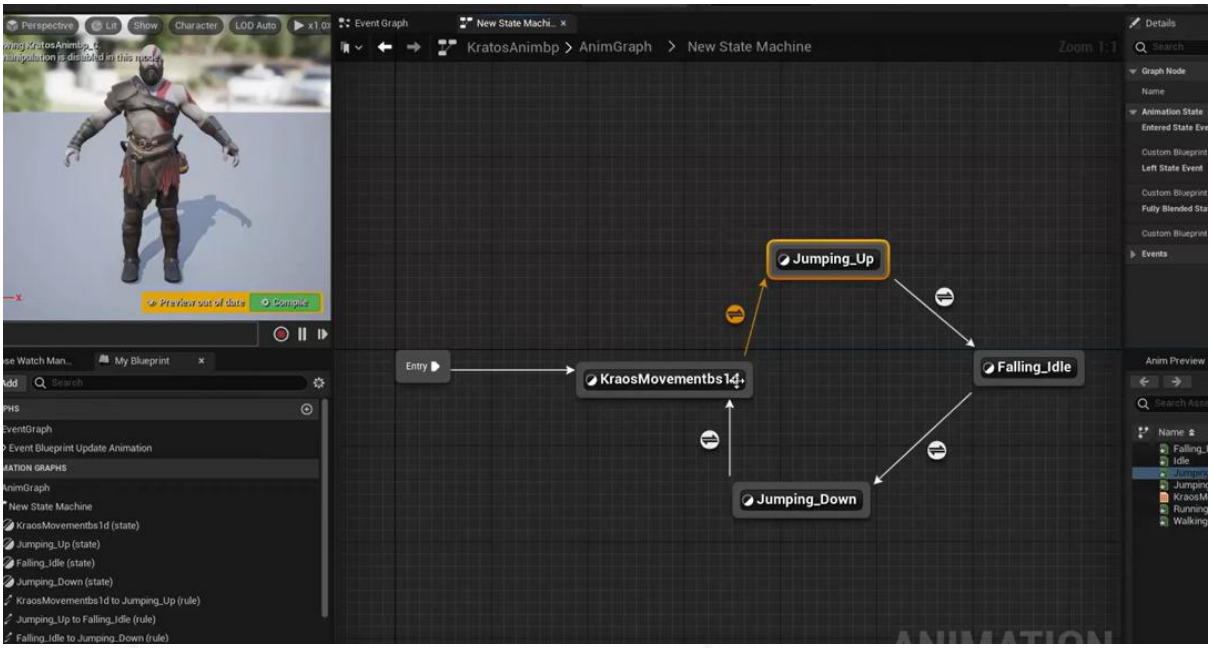



Preparing Shaders (12)
Content Browser
Add
Import
Save All
All
Content
Kratos
Model
Favorites
mixamotounreal
Search Model
Characters
LevelPrototyping
StarterContent
ThirdPerson


Skeleton Tree
Perspective
Lit
Show
Character
LOD Auto
Previewing Reference Pose
LOD: 0
Current Screen Size: 1
Triangles: 36,127
Vertices: 21,533
UV Channels: 1
Approx Size: 156x64x221
/Game/Kratos/Model/M_MED_Historian_Body
Material
Material Instance
Edit
Copy
Paste
Search Assets
M_Mannequin
M_MannequinUE4_Body
M_MannequinUE4_ChestLogo
M_MED_Historian_Body
M_MED_Historian_FaceAcc
M_MED_Historian_Head
86 items (1 selected)


Content Browser
Add
Import
Save All
All
Content
Kratos
Animations
Search Animations
Falling_Idle
Idle
Jumping_Down
Jumping_Up
Running
Walking
Content
Characters
Kratos
Animations
Model
LevelPrototyping
StarterContent
ThirdPerson


KratosAnimbp > AnimGraph > New State Machine
Drag Off Pins to Create/Connect New Nodes.
Entry
Event Blueprint Update Animation
AnimGraph
New State Machine
KraosMovementbs1d




**Resultados esperados**

El modelo 3D queda correctamente integrado en el proyecto, con materiales revisados y ajustados para asegurar una representación visual coherente. La colisión se configura desde el Static Mesh Editor, ya sea mediante generación automática o formas personalizadas según la geometría. La escala del objeto se normaliza para mantener proporciones consistentes dentro del nivel. Finalmente, el modelo se coloca en la escena y responde adecuadamente a iluminación, colisiones y comportamiento esperado dentro del entorno interactivo.

## 5.2 Practica experimental 11

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Manipulación de herramientas 2D y 3D en Unreal Engine**

**Objetivos**

Utilizar herramientas como brushes, landscape y materials para construir escenarios experimentales.

**Antecedentes/Escenario**

Unreal Engine proporciona herramientas para modelar terrenos, crear geometría y asignar materiales personalizados.

**Recursos necesarios**

Unreal Engine 5.

**Planteamiento del problema**

Se requiere reforzar dominio técnico manipulando elementos del editor.

**Pasos por realizar**

Paso 1: Crear geometría básica con brushes.

Paso 2: Generar un Landscape.

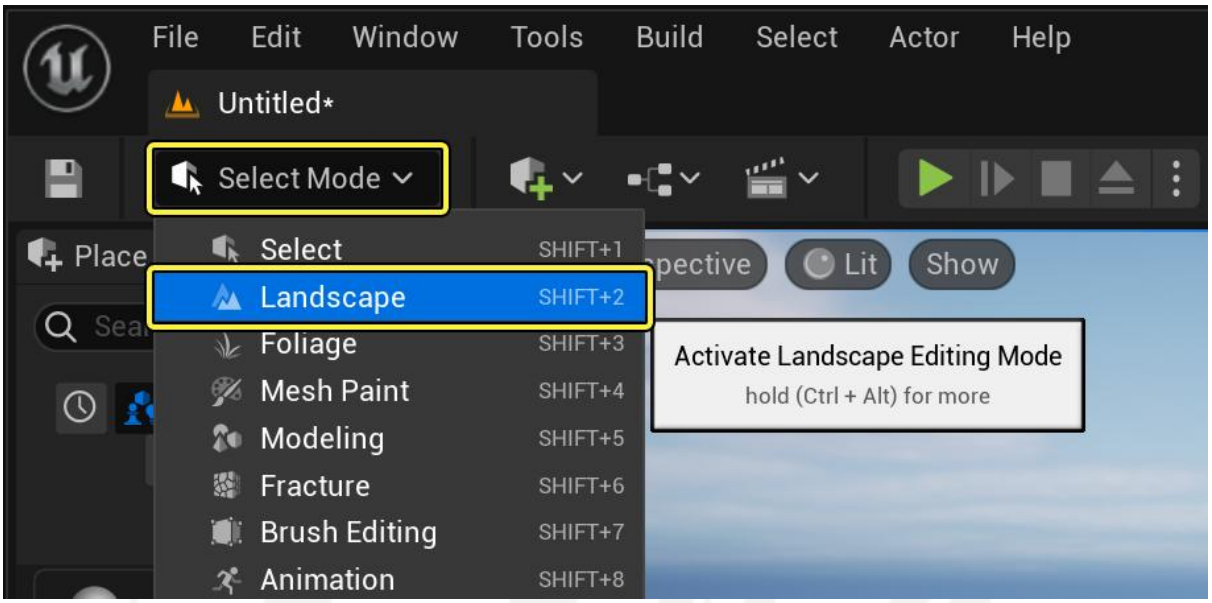
Paso 3: Asignar materiales a cada superficie.

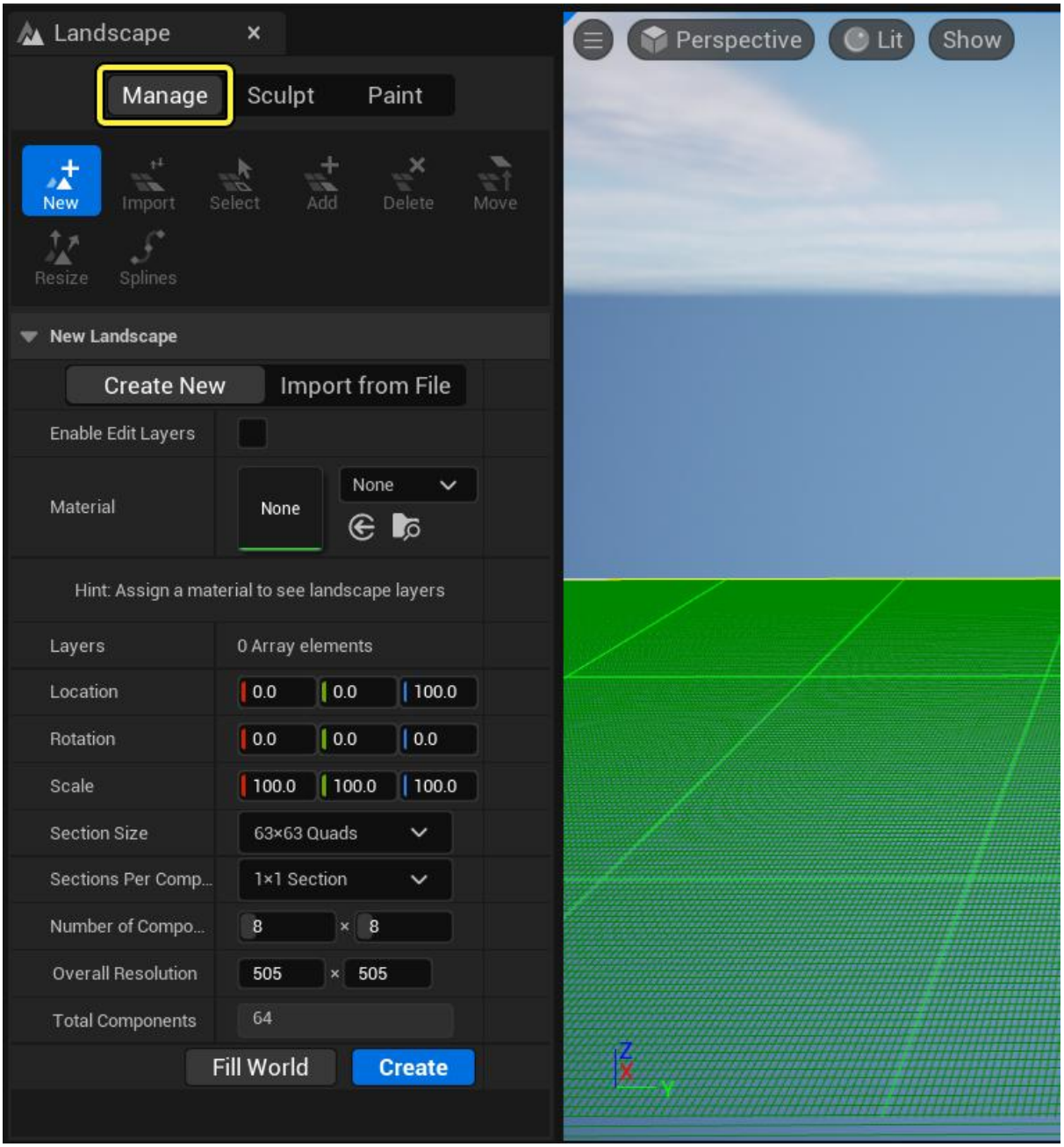
Paso 4: Ajustar tamaño y forma del entorno.

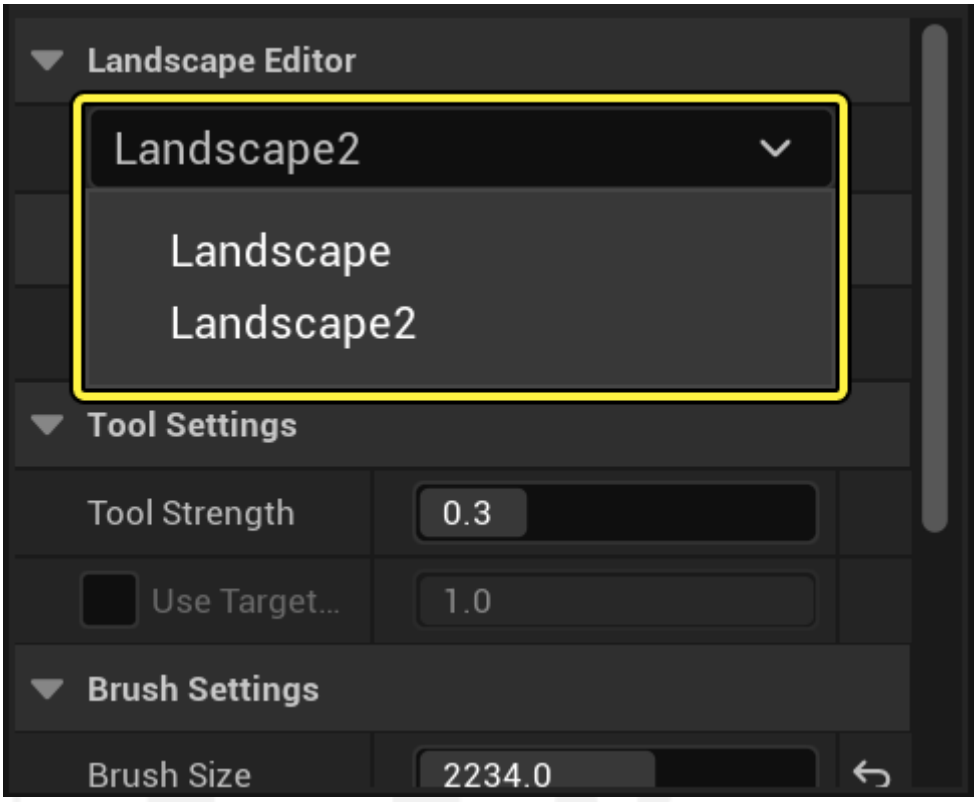
Paso 5: Probar navegabilidad.

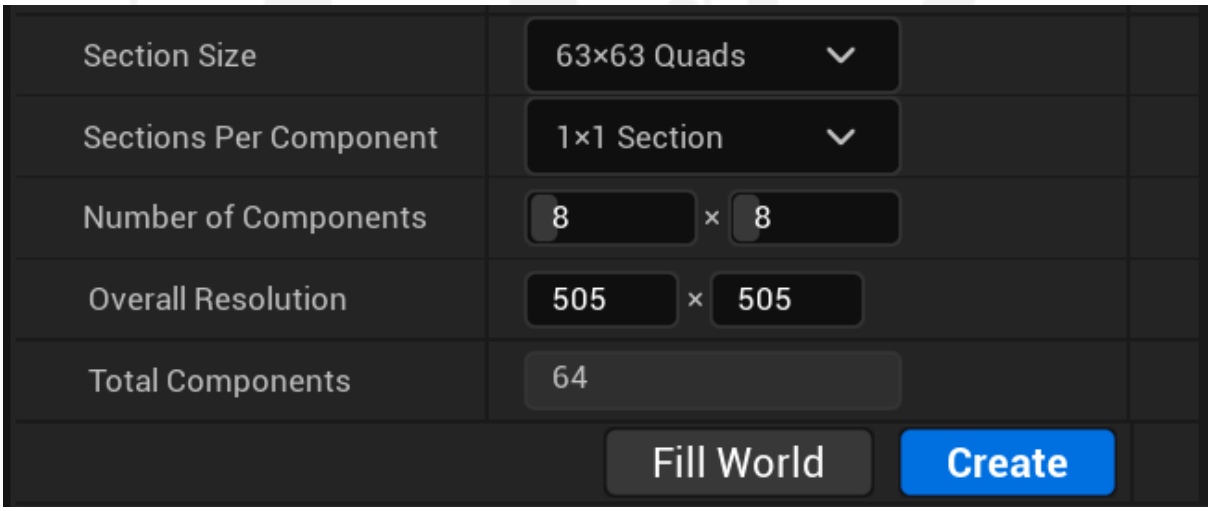
**Desarrollo**

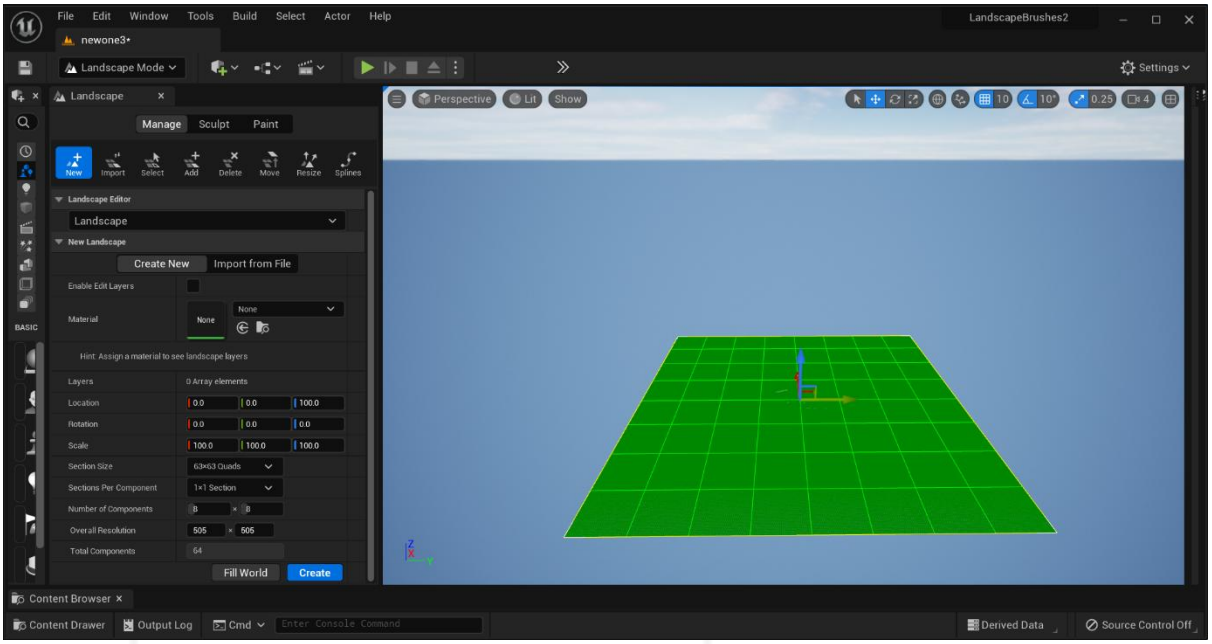
Se construyen muros, plataformas y habitaciones con geometry brushes. Luego se genera un Landscape ajustando su resolución. Se esculpe utilizando herramientas de Raise, Smooth y Flatten. Se crean materiales con texturas base, normales y roughness y se aplican al terreno y a los brushes. Finalmente se testea recorriendo el entorno con un personaje.




Landscape
Manage
Sculpt
Paint
New
Import
Select
Add
Delete
Move
Resize
Splines
New Landscape
Create New
Import from File
Enable Edit Layers
Material
None
None
Hint: Assign a material to see landscape layers
Layers
0 Array elements
Location
0.0
0.0
100.0
Rotation
0.0
0.0
0.0
Scale
100.0
100.0
100.0
Section Size
63×63 Quads
Sections Per Comp...
1×1 Section
Number of Compo...
8
8
Overall Resolution
505
505
Total Components
64
Fill World
Create
Perspective
Lit
Show

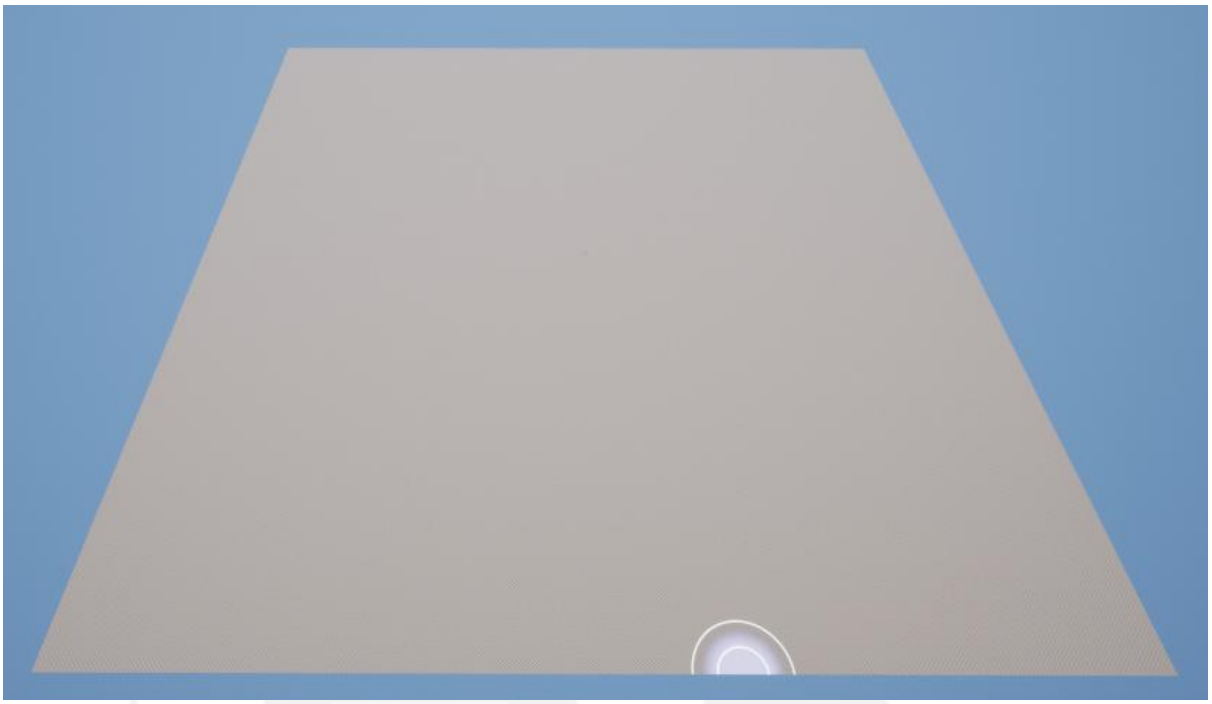

Landscape Editor
Landscape2
Landscape
Landscape2
Tool Settings
Tool Strength
0.3
Use Target...
1.0
Brush Settings
Brush Size
2234.0


Section Size
63×63 Quads
Sections Per Component
1×1 Section
Number of Components
8
×
8
Overall Resolution
505
×
505
Total Components
64
Fill World
Create


File Edit Window Tools Build Select Actor Help
newone3*
LandscapeBrushes2
Landscape Mode
Settings
Landscape
Manage Sculpt Paint
New Import Select Add Delete Move Resize Splines
Landscape Editor
Landscape
New Landscape
Create New Import from File
Enable Edit Layers
Material
None
Hint: Assign a material to see landscape layers
Layers 0 Array elements
Location
Rotation
Scale
Section Size 63x63 Quads
Sections Per Component 1x1 Section
Number of Components
Overall Resolution 505 x 505
Total Components 64
Fill World Create
Perspective Lit Show
Content Browser
Content Drawer Output Log Cmd Enter Console Command
Derived Data Source Control Off


Landscape
Manage
Sculpt
Paint
New
Import
Select
Add
Delete
Move
Resize
Splines
Landscape Editor
Landscape2
New Landscape
Create New
Import from File
Enable Edit Layers
Material
None
None
Hint: Assign a material to see landscape layers
Layers
0 Array elements
Location
0.0
0.0
100.0
Rotation
0.0
0.0
0.0
Scale
100.0
100.0
100.0
Section Size
63×63 Quads
Sections Per Component
1×1 Section
Number of Components
8
8
Overall Resolution
505
505
Total Components
64
Fill World
Create


Landscape
Manage
Sculpt
Paint
New
Import
Select
Add
Delete
Move
Resize
Splines
Landscape Editor
Landscape2
New Landscape
Create New
Import from File
Enable Edit Layers
Material
None
None
Hint: Assign a material to see landscape layers
Layers
0 Array elements
Location
0.0
0.0
100.0
Rotation
0.0
0.0
0.0
Scale
100.0
100.0
100.0
Section Size
63×63 Quads
Sections Per Component
1×1 Section
Number of Components
8
8
Overall Resolution
505
505
Total Components
64
Fill World
Create



**Resultados esperados**

El nivel queda conformado por geometría básica creada con brushes, un Landscape esculpido y superficies con materiales correctamente aplicados. Las herramientas de modelado permiten definir formas, alturas y volúmenes coherentes para la navegación. Los materiales incorporan texturas y parámetros visuales que mejoran el aspecto del entorno. Tras probar la escena con un personaje, se verifica que la navegación sea fluida, sin errores de colisión y con una distribución espacial funcional para futuros desarrollos.

## 5.3 Practica experimental 12

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Investigación y adaptación de Niagara Particles**

**Objetivos**

Comprender cómo funciona Niagara y adaptarlo para crear efectos visuales funcionales dentro de un proyecto.

**Antecedentes/Escenario**

Niagara es el sistema avanzado de partículas de UE5 que soporta simulaciones complejas.

**Recursos necesarios**

Unreal Engine 5.

**Planteamiento del problema**

Es necesario integrar un efecto visual creado en Niagara dentro de un proyecto.

**Pasos por realizar**

Paso 1: Investigar módulos básicos de Niagara.

Paso 2: Crear un sistema nuevo.

Paso 3: Ajustar emisores, tamaño, color y velocidad.

Paso 4: Colocar el efecto en el nivel.

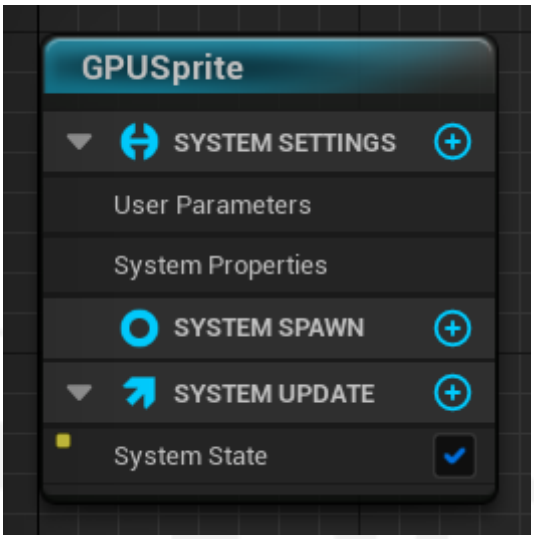
Paso 5: Vincularlo a un evento si es necesario.

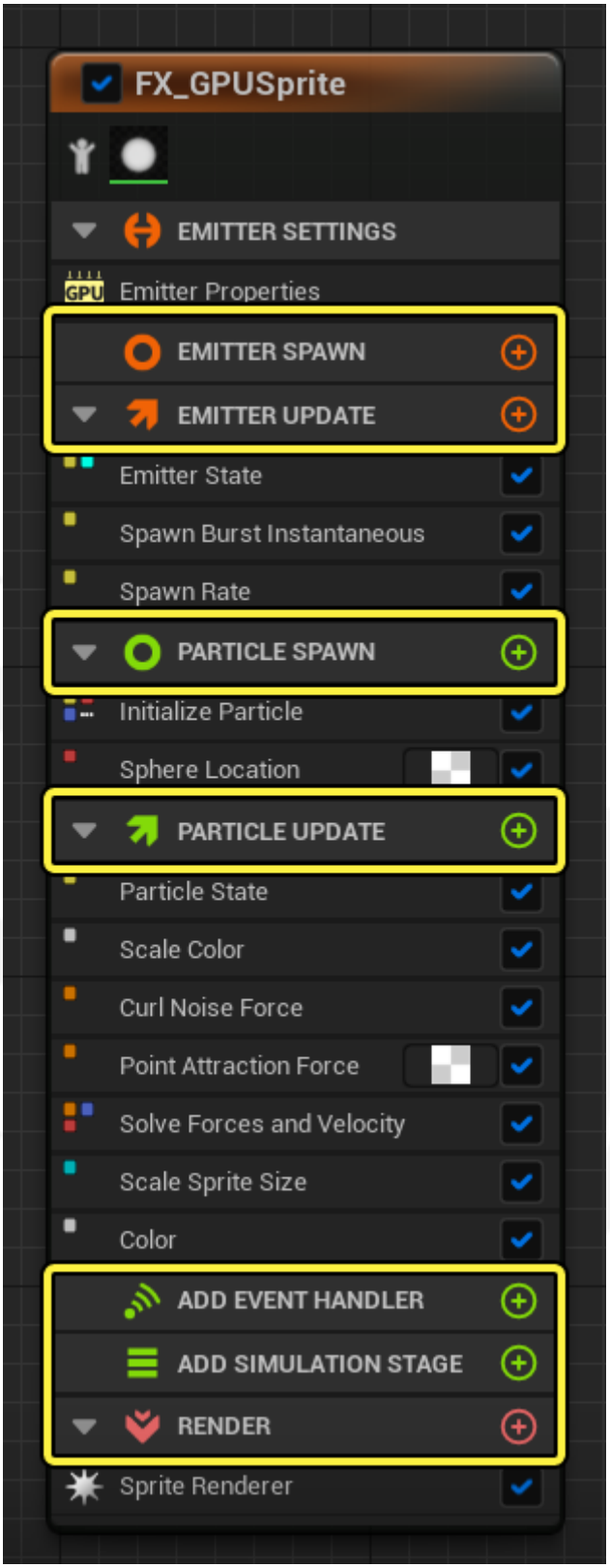


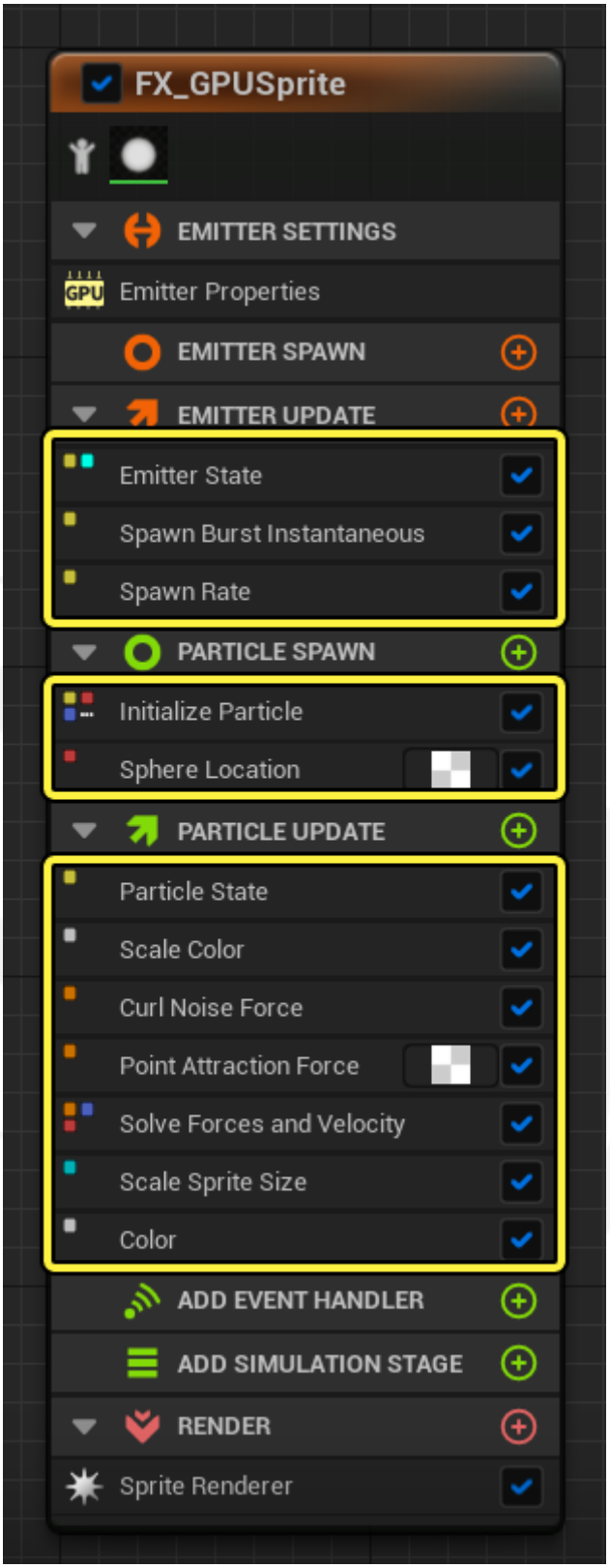
(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

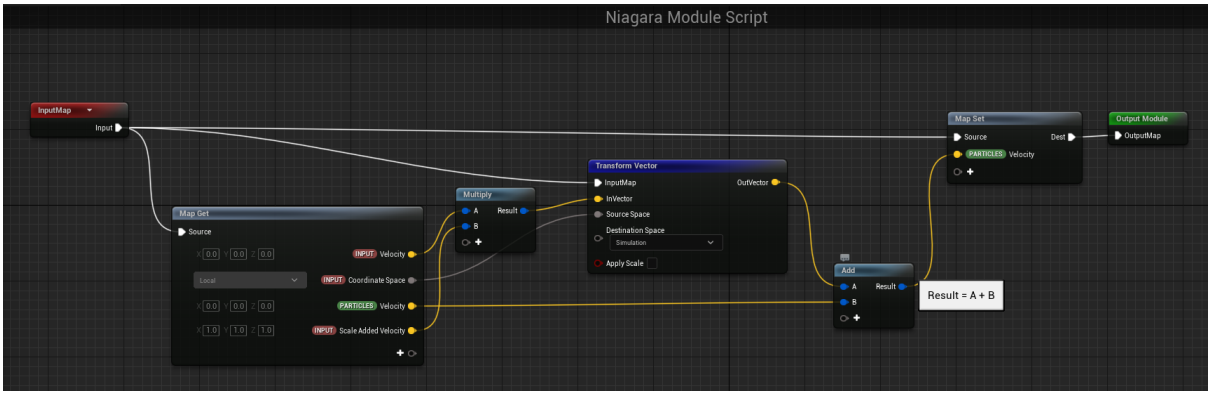
**Desarrollo**

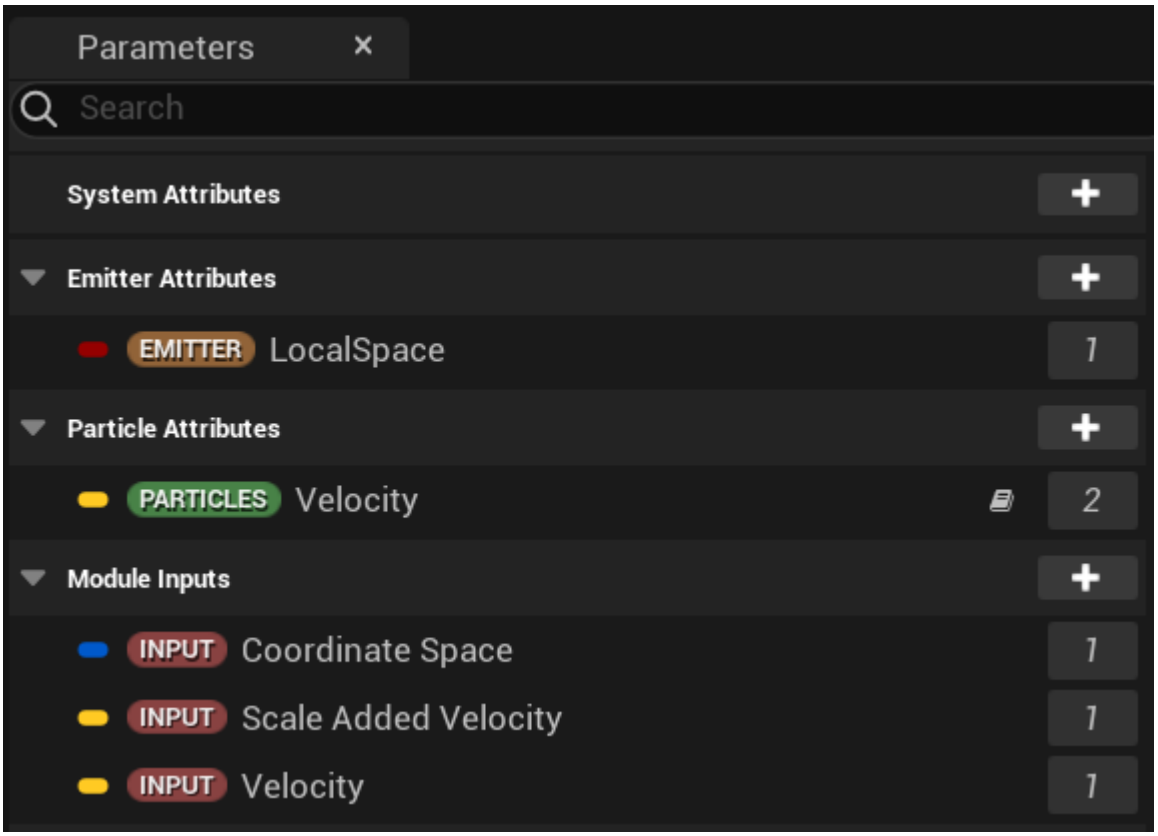
Se crea un Niagara System basado en una plantilla vacía. Se añaden módulos como Spawn Rate, Color, Scale Sprite Size y Velocity. Se ajustan parámetros para generar un efecto como humo, fuego o partículas de energía. Luego se coloca en el nivel y se vincula a un Blueprint si se requiere activación por evento.

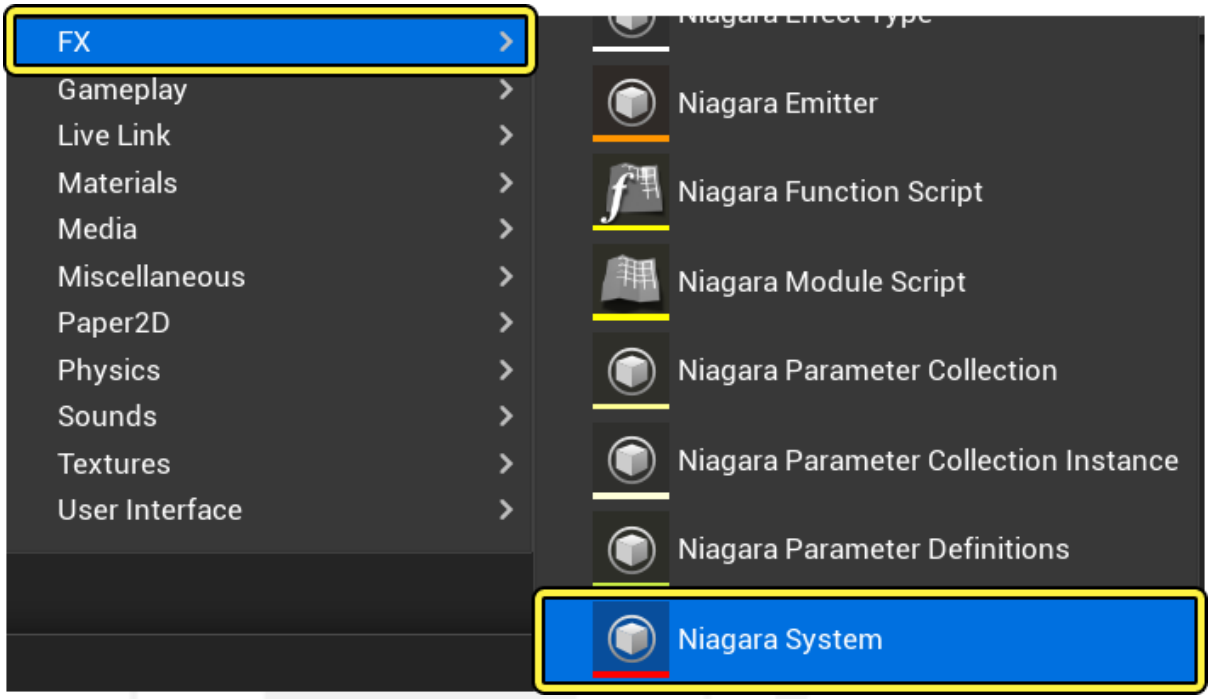



FX_GPUSprite
EMITTER SETTINGS
Emitter Properties
EMITTER SPAWN
EMITTER UPDATE
Emitter State
Spawn Burst Instantaneous
Spawn Rate
PARTICLE SPAWN
Initialize Particle
Sphere Location
PARTICLE UPDATE
Particle State
Scale Color
Curl Noise Force
Point Attraction Force
Solve Forces and Velocity
Scale Sprite Size
Color
ADD EVENT HANDLER
ADD SIMULATION STAGE
RENDER
Sprite Renderer


FX_GPUSprite
EMITTER SETTINGS
Emitter Properties
EMITTER SPAWN
EMITTER UPDATE
Emitter State
Spawn Burst Instantaneous
Spawn Rate
PARTICLE SPAWN
Initialize Particle
Sphere Location
PARTICLE UPDATE
Particle State
Scale Color
Curl Noise Force
Point Attraction Force
Solve Forces and Velocity
Scale Sprite Size
Color
ADD EVENT HANDLER
ADD SIMULATION STAGE
RENDER
Sprite Renderer


Niagara Module Script
InputMap
Input
Map Get
Source
Velocity
Coordinate Space
Velocity
Scale Added Velocity
Multiply
A
B
Result
Transform Vector
InputMap
InVector
Source Space
Destination Space
Simulation
Apply Scale
OutVector
Add
A
B
Result
Result = A + B
Map Set
Source
Dest
Velocity
Output Module
OutputMap


Parameters
Search
System Attributes
Emitter Attributes
EMITTER LocalSpace
1
Particle Attributes
PARTICLES Velocity
2
Module Inputs
INPUT Coordinate Space
1
INPUT Scale Added Velocity
1
INPUT Velocity
1




**Resultados esperados**

El sistema Niagara configurado produce un efecto visual coherente con la estética del proyecto, ya sea humo, fuego, energía u otro fenómeno. Los módulos ajustados permiten controlar emisión, color, tamaño y comportamiento de las partículas. Al colocarlo en el nivel, el efecto responde correctamente al entorno y, si está vinculado a un evento, se activa en el momento previsto. Con ello se refuerza el entendimiento práctico de Niagara dentro del flujo de trabajo de UE5.

# 6 UNIDAD 5: Ejecutable

## 6.1 Practica experimental 13

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Investigación de IA y sus componentes en Unreal Engine**

**Objetivos**

Analizar el funcionamiento del sistema de Inteligencia Artificial en Unreal Engine, identificando sus componentes principales y comprendiendo cómo se integran entre sí para que un personaje pueda percibir, decidir y actuar dentro del entorno interactivo.

**Antecedentes/Escenario**

Unreal Engine utiliza un sistema modular de IA compuesto por controladores, percepciones, navegación, tableros de datos y árboles de comportamiento. Estos elementos se coordinan para generar acciones autónomas en NPCs, permitiendo comportamientos como patrullar, perseguir objetivos, reaccionar a estímulos o tomar decisiones basadas en condiciones dinámicas del entorno.

**Recursos necesarios**

Documentación oficial de Unreal Engine.

Proyecto básico con al menos un personaje no jugable.

Acceso al editor para pruebas.

**Planteamiento del problema**

El desafío consiste en comprender cómo funcionan los subsistemas que permiten a un NPC actuar sin intervención humana, analizando la interacción entre percepción, navegación y toma de decisiones.

**Pasos por realizar**

Paso 1: Investigar qué es un AI Controller y su relación con un Pawn o Character.
Paso 2: Analizar el sistema de percepción de Unreal y sus formas de estímulo.
Paso 3: Estudiar el funcionamiento del sistema de navegación y la malla de recorrido.


(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

Paso 4: Revisar Behavior Trees y su integración con Blackboards.
Paso 5: Elaborar un resumen final unificando todos los componentes.

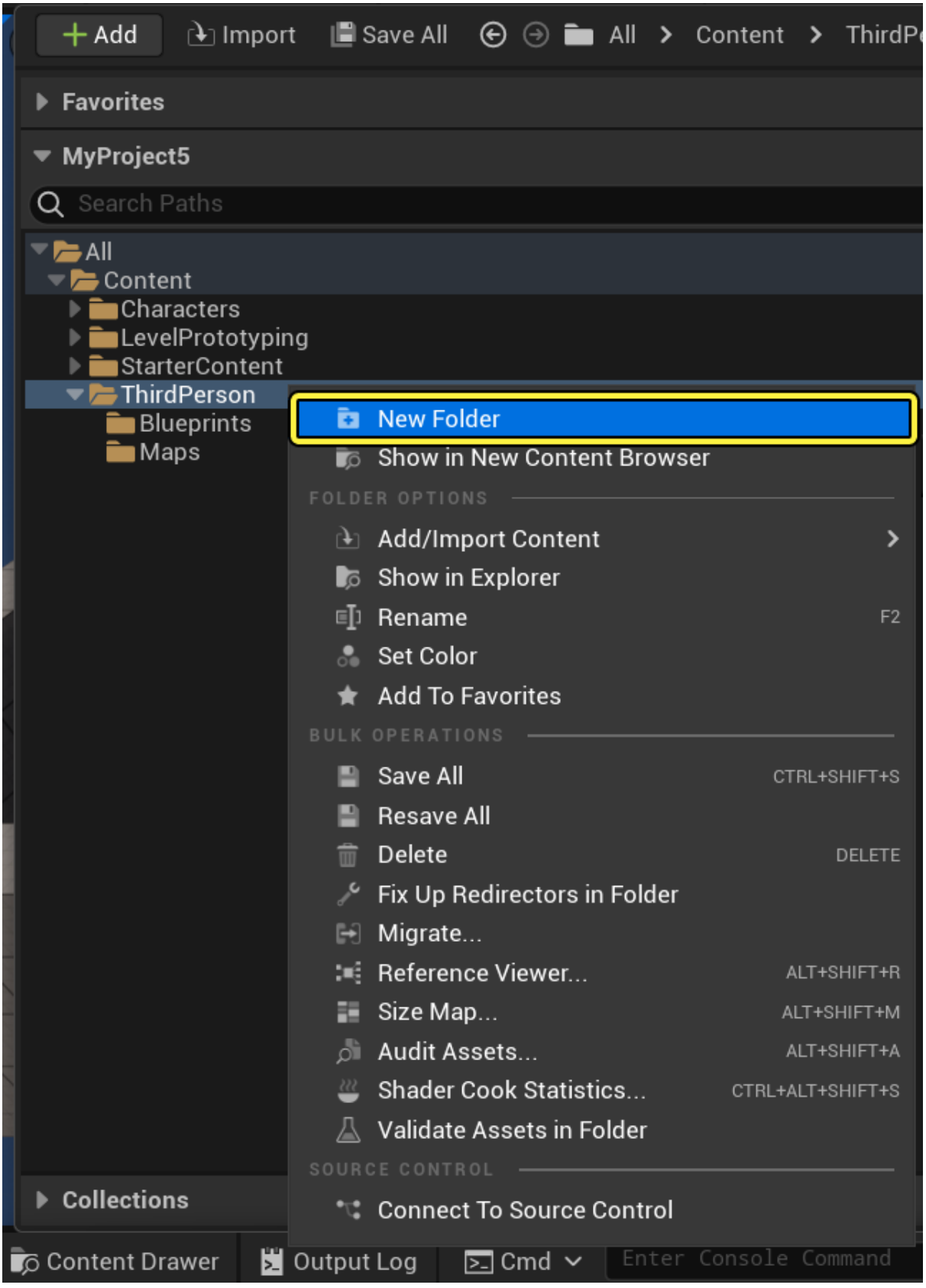
**Desarrollo**

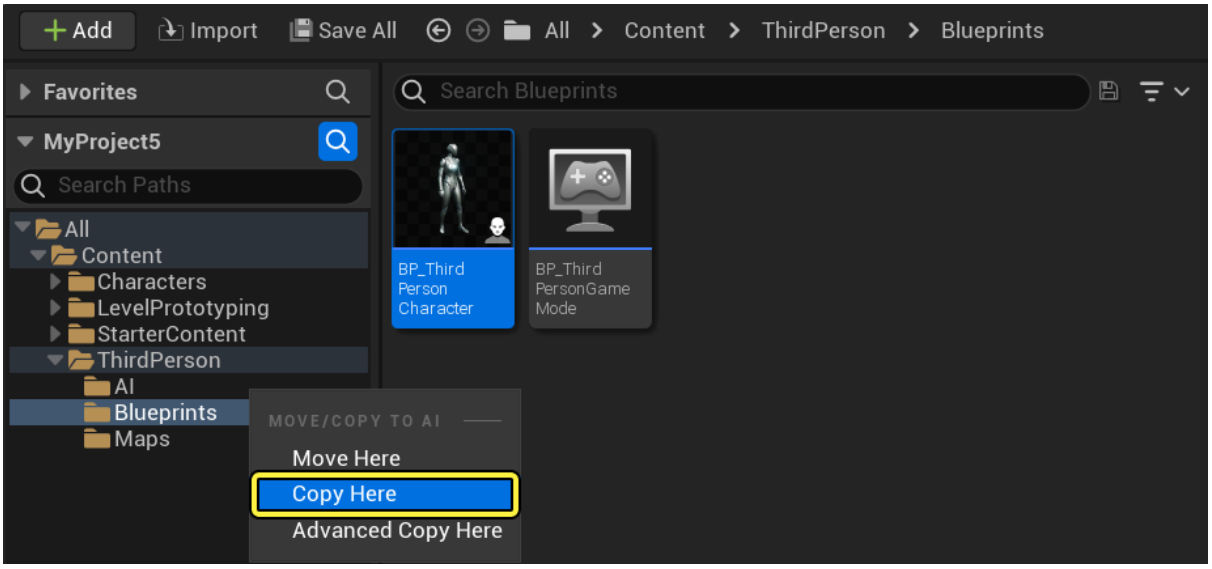
En la investigación se inició analizando el AI Controller, el cual reemplaza el control humano del personaje y administra las decisiones de la IA. Se descubrió que el controlador es indispensable para procesar las órdenes del Behavior Tree y coordinar movimientos con el sistema de navegación. Posteriormente se estudió el sistema de percepción. Este utiliza componentes capaces de detectar estímulos como visión, sonido o daño. El AIPerception permite definir rangos de visión, ángulos, tiempos de pérdida del objetivo y prioridades de estímulo. Cuando el jugador ingresa en ese rango, la IA recibe un evento que puede activar una transición de comportamiento. Luego se estudió la navegación, descubriendo que se genera mediante un NavMeshBoundsVolume. Cuando se compila la navegación, los NPC pueden calcular rutas automáticamente utilizando el nodo Move To.

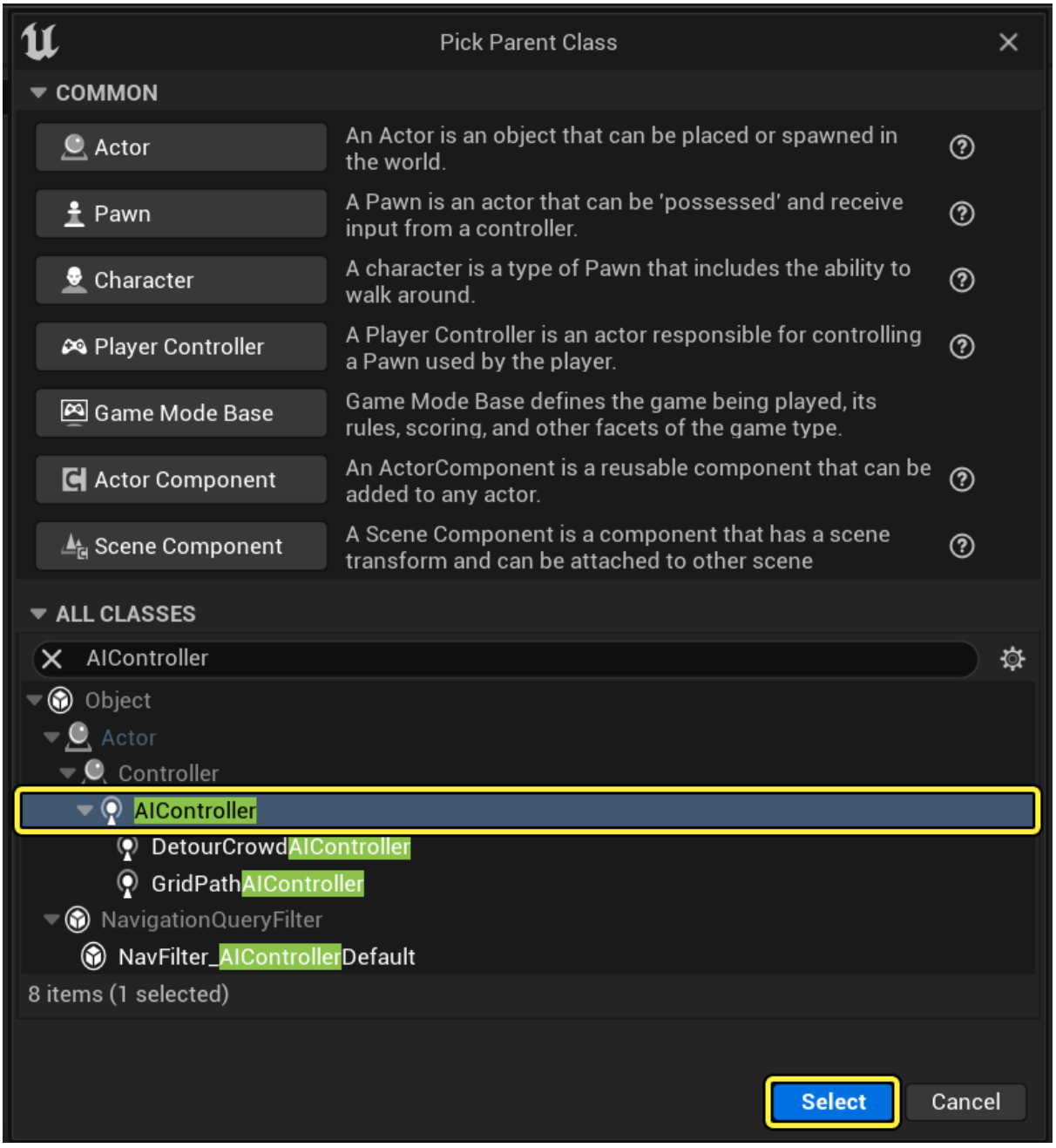
A continuación se investigaron los Behavior Trees. Estos organizan las decisiones del NPC mediante nodos secuenciales, selectores, tareas y condiciones. Se apoyan en un Blackboard donde se almacenan claves de información utilizadas por el árbol, como la ubicación del jugador, estados, contadores o defensas.

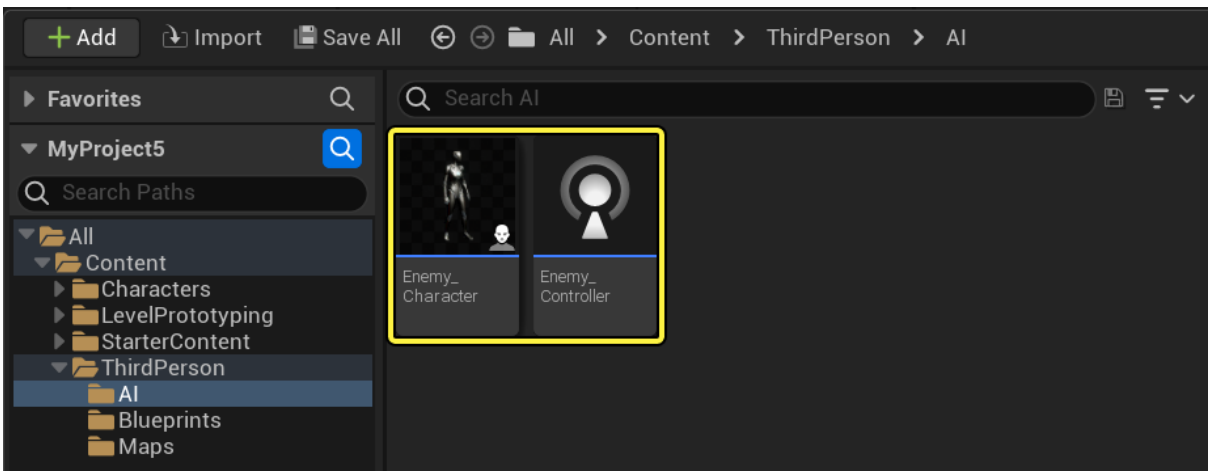
Finalmente se integraron todos los elementos determinando que el funcionamiento completo de la IA se basa en recibir estímulos, evaluar decisiones mediante el Behavior Tree, conservar información en el Blackboard y desplazarse con la malla de navegación usando el AI Controller como intermediario.

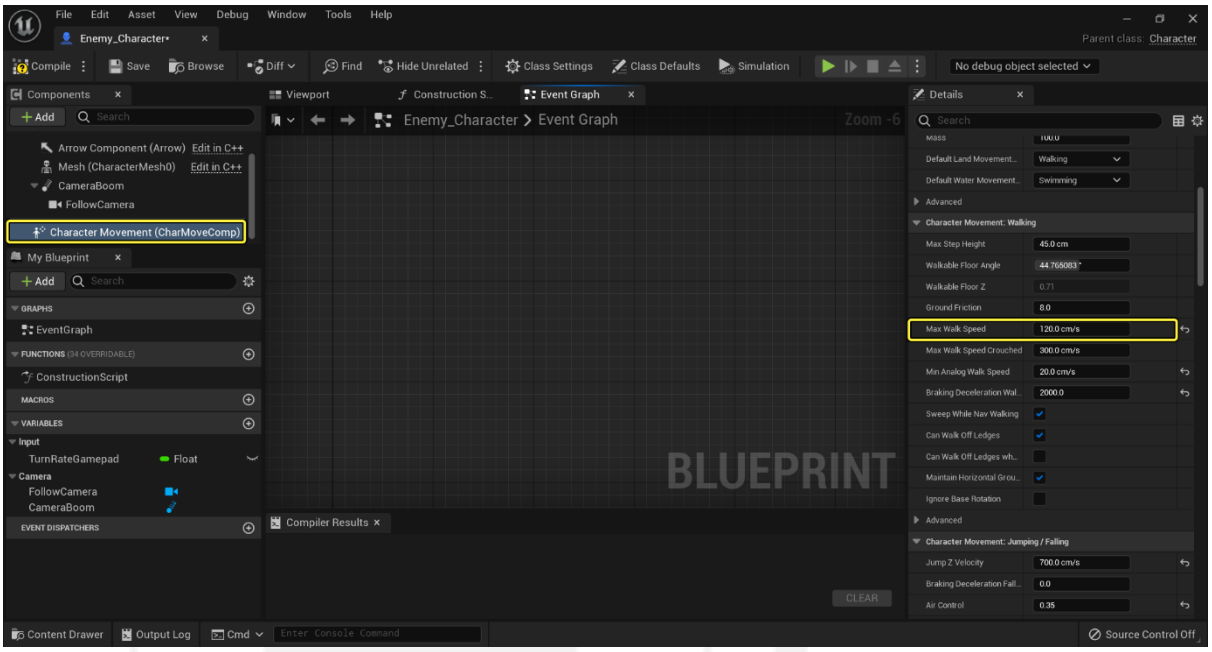

(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

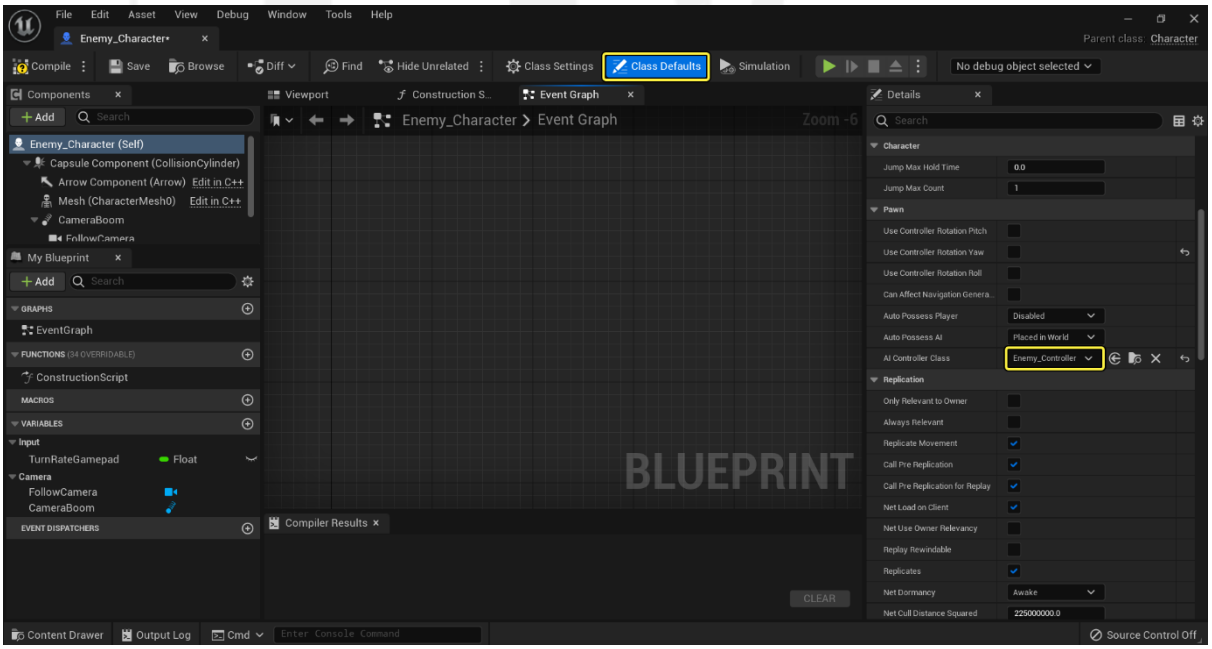

Add
Import
Save All
All
Content
Favorites
MyProject5
Search Paths
All
Content
Characters
LevelPrototyping
StarterContent
ThirdPerson
Blueprints
Maps
New Folder
Show in New Content Browser
FOLDER OPTIONS
Add/Import Content
Show in Explorer
Rename F2
Set Color
Add To Favorites
BULK OPERATIONS
Save All CTRL+SHIFT+S
Resave All
Delete DELETE
Fix Up Redirectors in Folder
Migrate...
Reference Viewer... ALT+SHIFT+R
Size Map... ALT+SHIFT+M
Audit Assets... ALT+SHIFT+A
Shader Cook Statistics... CTRL+ALT+SHIFT+S
Validate Assets in Folder
SOURCE CONTROL
Connect To Source Control
Collections
Content Drawer
Output Log
Cmd
Enter Console Command

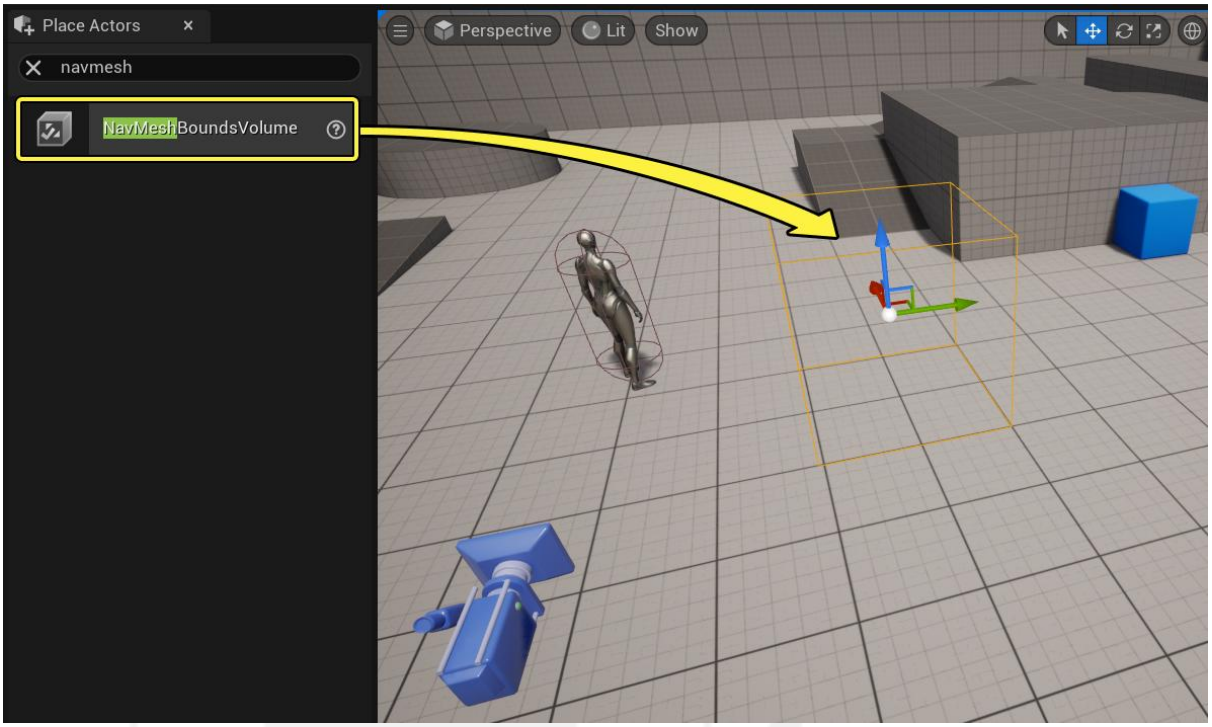

Add
Import
Save All
All
Content
ThirdPerson
Blueprints
Favorites
MyProject5
Search Paths
All
Content
Characters
LevelPrototyping
StarterContent
ThirdPerson
AI
Blueprints
Maps
Search Blueprints
BP_Third Person Character
BP_Third PersonGame Mode
MOVE/COPY TO AI
Move Here
Copy Here
Advanced Copy Here


Pick Parent Class
COMMON
Actor
An Actor is an object that can be placed or spawned in the world.
Pawn
A Pawn is an actor that can be 'possessed' and receive input from a controller.
Character
A character is a type of Pawn that includes the ability to walk around.
Player Controller
A Player Controller is an actor responsible for controlling a Pawn used by the player.
Game Mode Base
Game Mode Base defines the game being played, its rules, scoring, and other facets of the game type.
Actor Component
An ActorComponent is a reusable component that can be added to any actor.
Scene Component
A Scene Component is a component that has a scene transform and can be attached to other scene
ALL CLASSES
AIController
Object
Actor
Controller
AIController
DetourCrowdAIController
GridPathAIController
NavigationQueryFilter
NavFilter_AIControllerDefault
8 items (1 selected)
Select
Cancel


Add
Import
Save All
All
Content
ThirdPerson
AI
Favorites
MyProject5
Search Paths
Search AI
All
Content
Characters
LevelPrototyping
StarterContent
ThirdPerson
AI
Blueprints
Maps
Enemy_ Character
Enemy_ Controller


File Edit Asset View Debug Window Tools Help
Enemy_Character
Parent class: Character
Compile Save Browse Diff Find Hide Unrelated Class Settings Class Defaults Simulation
No debug object selected
Components
Add
Arrow Component (Arrow) Edit in C++
Mesh (CharacterMesh0) Edit in C++
CameraBoom
FollowCamera
Character Movement (CharMoveComp)
My Blueprint
GRAPHS
EventGraph
FUNCTIONS (34 OVERRIDABLE)
ConstructionScript
MACROS
VARIABLES
Input
TurnRateGamepad Float
Camera
FollowCamera
CameraBoom
EVENT DISPATCHERS
Viewport
Construction S...
Event Graph
Enemy_Character > Event Graph
Zoom -6
BLUEPRINT
Compiler Results
CLEAR
Details
Default Land Movement... Walking
Default Water Movement... Swimming
Advanced
Character Movement: Walking
Max Step Height 45.0 cm
Walkable Floor Angle
Walkable Floor Z
Ground Friction 8.0
Max Walk Speed 120.0 cm/s
Max Walk Speed Crouched 300.0 cm/s
Min Analog Walk Speed 20.0 cm/s
Braking Deceleration Wal... 2000.0
Sweep While Nav Walking
Can Walk Off Ledges
Maintain Horizontal Grou...
Ignore Base Rotation
Character Movement: Jumping / Falling
Jump Z Velocity 700.0 cm/s
Braking Deceleration Fall... 0.0
Air Control 0.35
Content Drawer
Output Log
Cmd
Enter Console Command
Source Control Off


File Edit Asset View Debug Window Tools Help
Enemy_Character
Parent class: Character
Compile Save Browse Diff Find Hide Unrelated Class Settings Class Defaults Simulation
No debug object selected
Components
Add
Enemy_Character (Self)
Capsule Component (CollisionCylinder)
Arrow Component (Arrow) Edit in C++
Mesh (CharacterMesh0) Edit in C++
CameraBoom
FollowCamera
My Blueprint
GRAPHS
EventGraph
FUNCTIONS (34 OVERRIDABLE)
ConstructionScript
MACROS
VARIABLES
Input
TurnRateGamepad Float
Camera
FollowCamera
CameraBoom
EVENT DISPATCHERS
Viewport
Construction S...
Event Graph
Enemy_Character > Event Graph
Zoom -6
BLUEPRINT
Compiler Results
CLEAR
Details
Character
Jump Max Hold Time 0.0
Jump Max Count 1
Pawn
Use Controller Rotation Pitch
Use Controller Rotation Yaw
Use Controller Rotation Roll
Can Affect Navigation Genera...
Auto Possess Player Disabled
Auto Possess AI Placed in World
AI Controller Class Enemy_Controller
Replication
Only Relevant to Owner
Always Relevant
Replicate Movement
Call Pre Replication
Call Pre Replication for Replay
Net Load on Client
Net Use Owner Relevancy
Replay Rewindable
Replicates
Net Dormancy Awake
Net Cull Distance Squared 225000000.0
Content Drawer
Output Log
Cmd
Enter Console Command
Source Control Off




**Resultados esperados**

Como resultado del análisis, se ha identificado la arquitectura modular de la inteligencia artificial en Unreal Engine, comprendiendo la interacción crítica entre el *AI Controller* y el *Pawn*. Se ha logrado determinar cómo el sistema de percepción sensorial alimenta al *Blackboard* con datos en tiempo real, permitiendo que el *Behavior Tree* ejecute una toma de decisiones lógica y jerárquica. Finalmente, se ha validado la dependencia del sistema de navegación (NavMesh) para la ejecución de desplazamientos autónomos, consolidando un flujo de trabajo técnico donde la detección, el procesamiento de datos y la acción física se integran de manera eficiente.


(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

## 6.2 Practica experimental 14

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Blueprint para movimiento controlado por IA**

**Objetivos**

Construir un Blueprint funcional que permita que un NPC se desplace de manera autónoma utilizando un AI Controller y la navegación de Unreal Engine.

**Antecedentes/Escenario**

Para que un NPC pueda moverse en un entorno interactivo, requiere un controlador asignado, una malla de navegación y un punto de destino. Unreal Engine facilita este proceso mediante Blueprints que permiten configurar movimiento sin necesidad de programar en C++.

**Recursos necesarios**

NPC configurado previamente.

Blueprint basado en AIController.

NavMeshBoundsVolume colocado en la escena.

**Planteamiento del problema**

El reto consiste en implementar un sistema de movimiento autónomo que permita al NPC desplazarse hacia un punto objetivo en el nivel, validando su integración con la navegación.

**Pasos por realizar**

Paso 1: Crear un Blueprint basado en AIController.

Paso 2: Asignar este controlador al NPC desde el panel de detalles.

Paso 3: Configurar Auto Possess AI para que el NPC active el controlador desde el inicio.

Paso 4: Colocar un NavMeshBoundsVolume en la escena y compilar navegación.

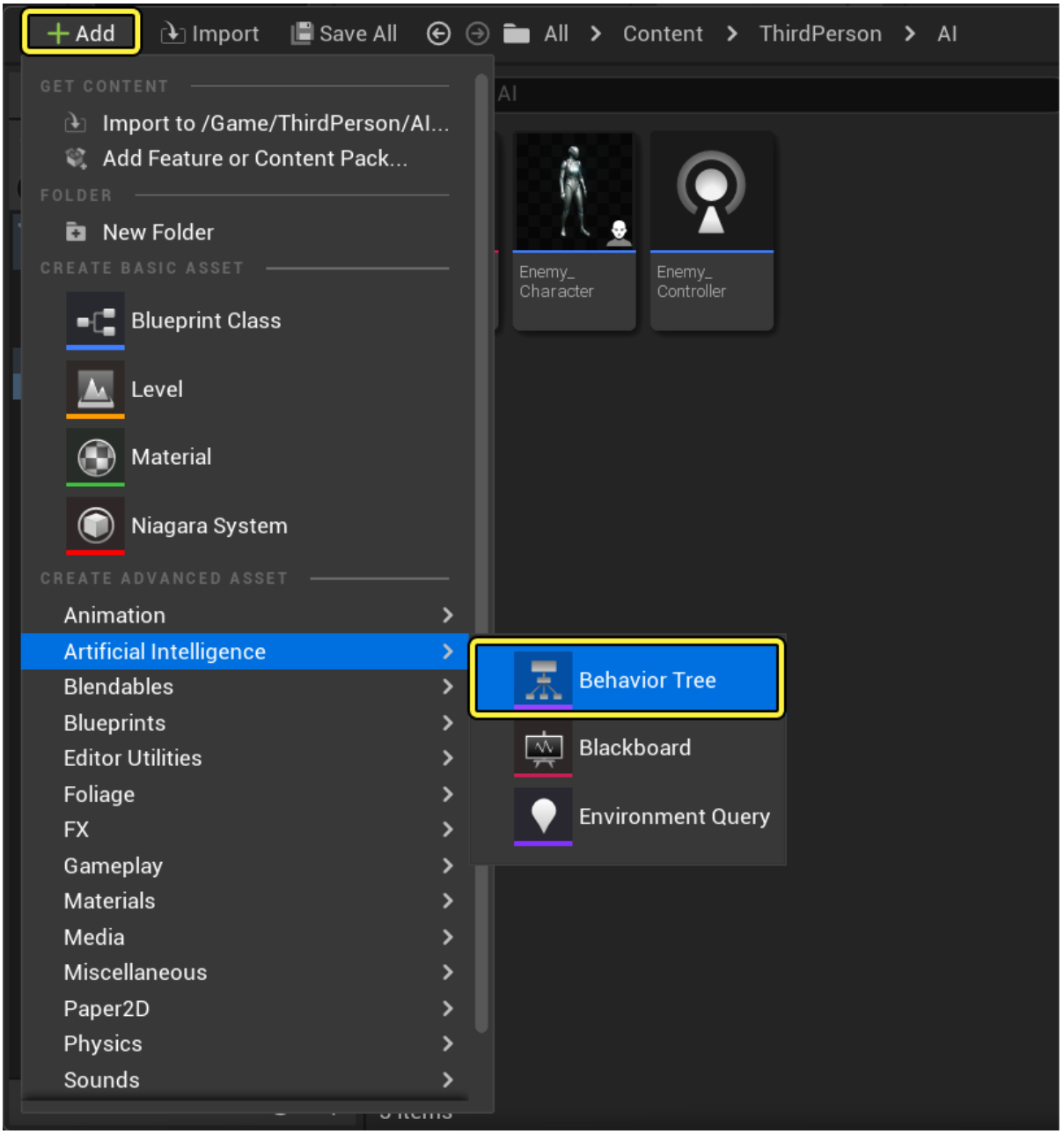
Paso 5: Crear un nodo de movimiento en el Event Graph que envíe al NPC hacia un objetivo.

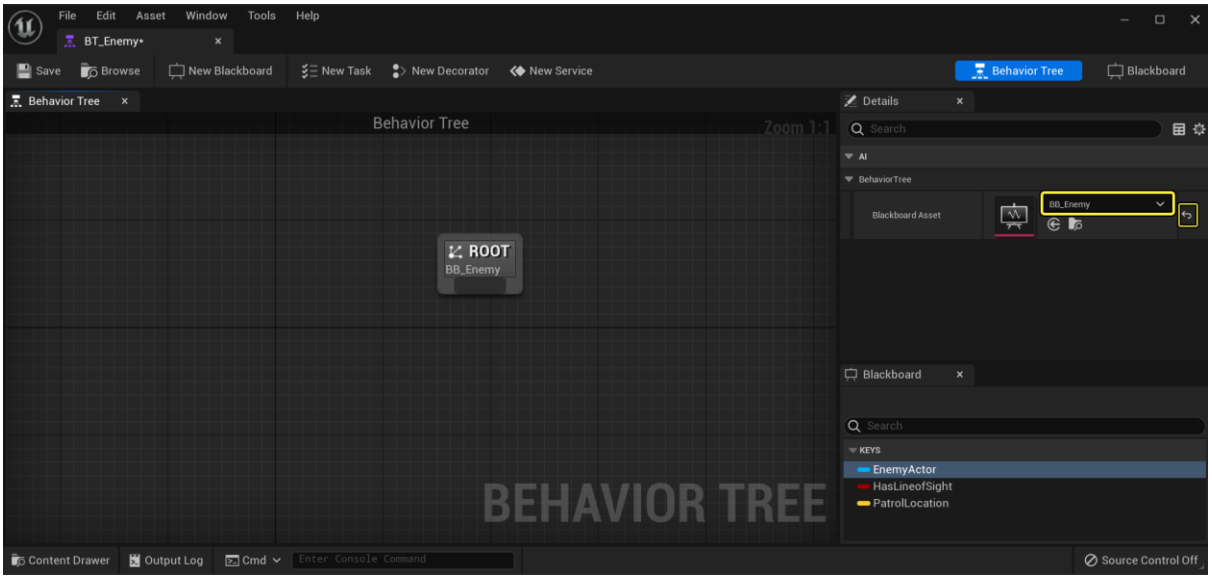
**Desarrollo**

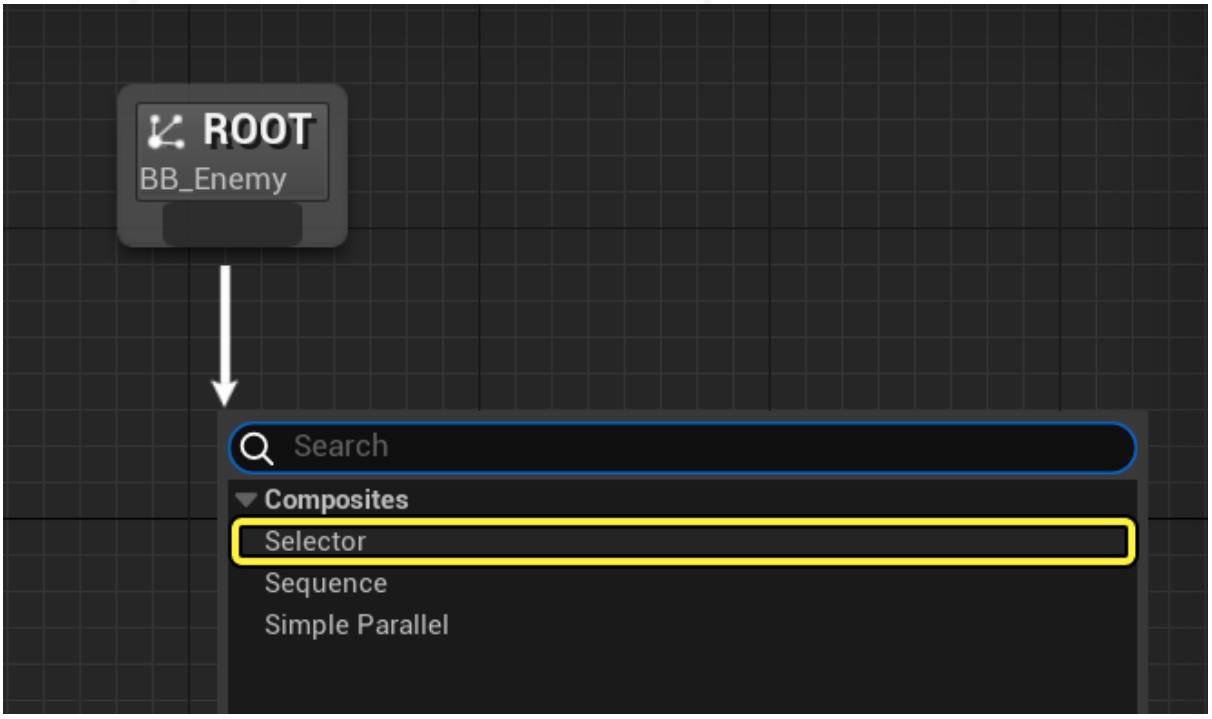
Se inició creando un Blueprint derivado de AIController y se nombró adecuadamente. En el NPC se abrió el panel de detalles y se asignó el controlador como AI Controller Class. Luego se activó la opción Auto Possess AI para “Placed in World or Spawned”, asegurando que el controlador tome control desde el comienzo.

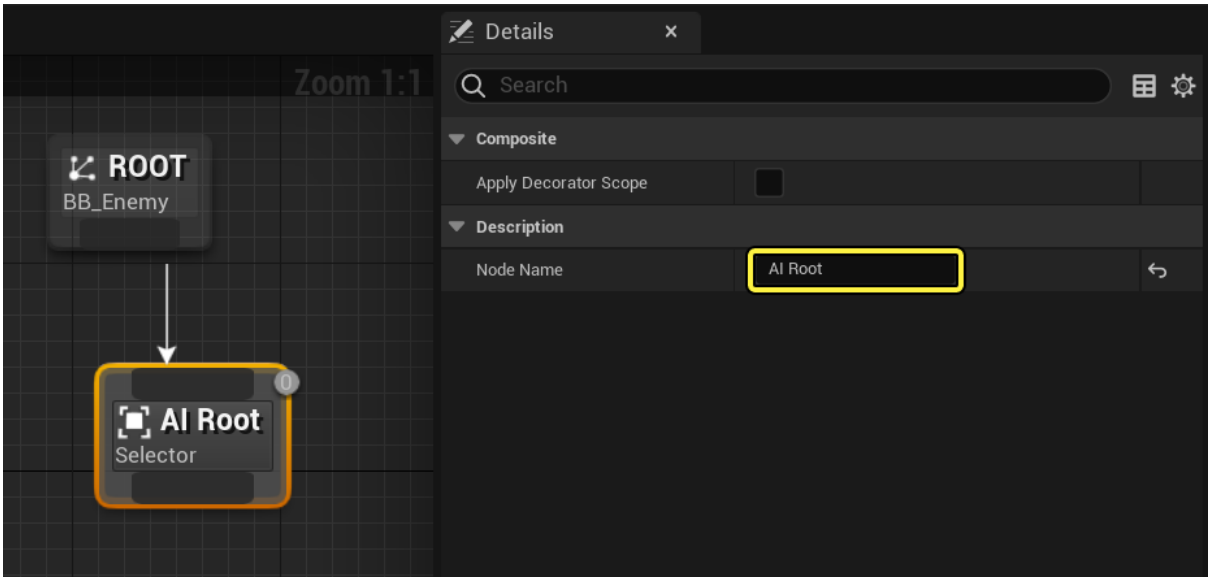
Posteriormente se insertó un NavMeshBoundsVolume en la escena cubriendo la zona donde se permitirá el movimiento. Al activar la vista de navegación, el área se mostró en verde, indicando que el NPC podría usarla.

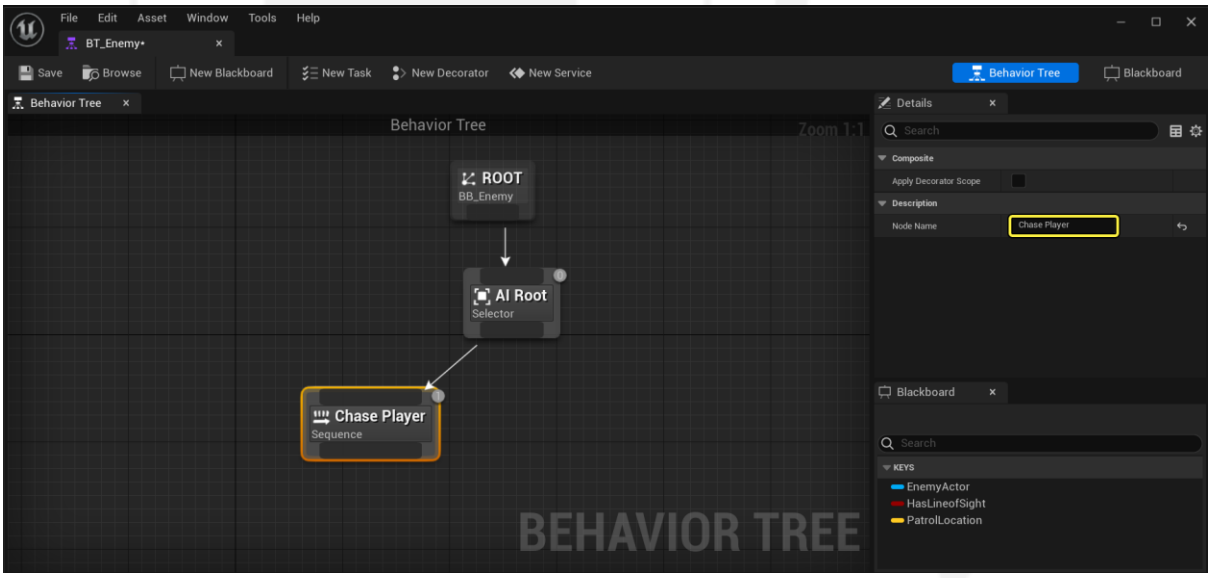
En el Blueprint del NPC se abrió el Event Graph y se agregó un nodo Event BeginPlay. Desde él se añadió un nodo AI MoveTo, configurando como destino una ubicación específica del mapa o un actor colocado como objetivo. Este nodo utiliza la malla de navegación para trazar una ruta. Al ejecutar el proyecto, el NPC inició su desplazamiento automáticamente siguiendo la ruta calculada. Esto confirmó la correcta interacción entre controlador, navegación y lógica de movimiento.

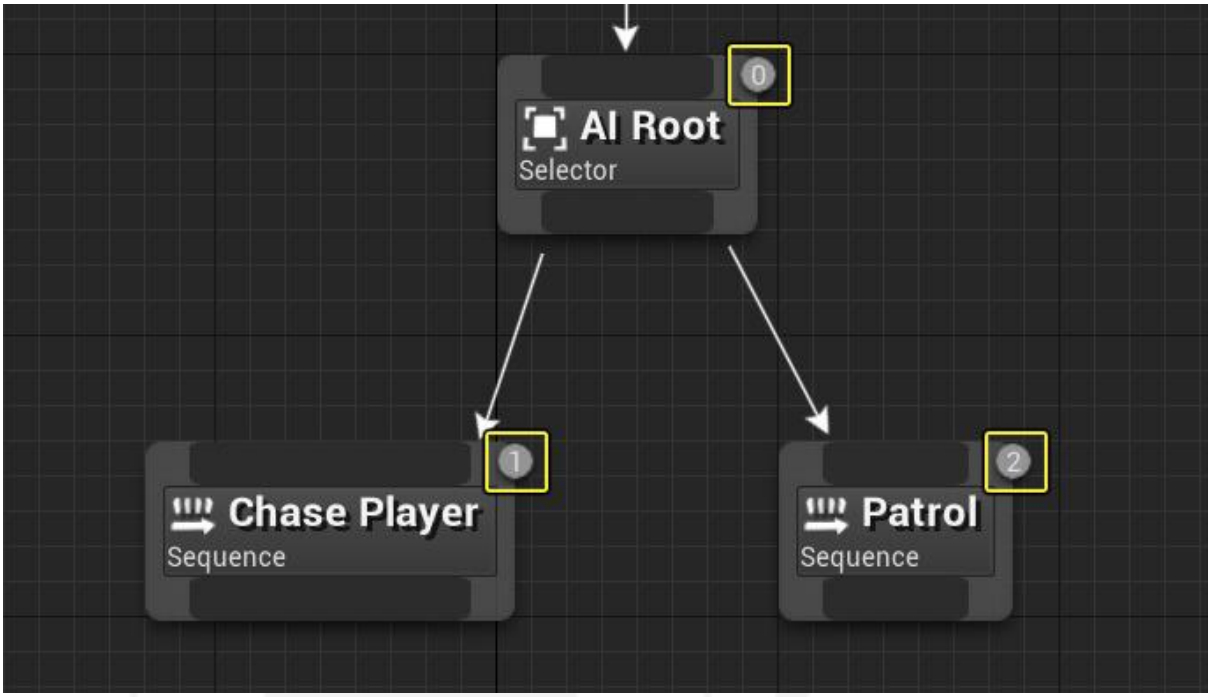

Add
Import
Save All
All
Content
ThirdPerson
AI
GET CONTENT
Import to /Game/ThirdPerson/AI...
Add Feature or Content Pack...
FOLDER
New Folder
CREATE BASIC ASSET
Blueprint Class
Level
Material
Niagara System
CREATE ADVANCED ASSET
Animation
Artificial Intelligence
Blendables
Blueprints
Editor Utilities
Foliage
FX
Gameplay
Materials
Media
Miscellaneous
Paper2D
Physics
Sounds
Behavior Tree
Blackboard
Environment Query
Enemy_ Character
Enemy_ Controller

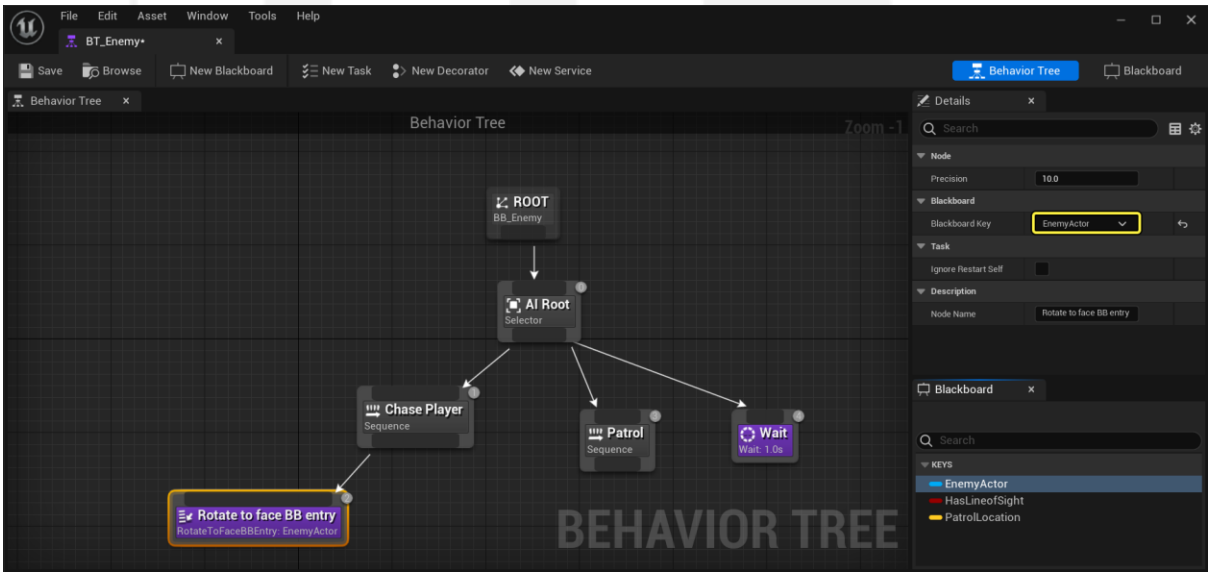

File
Edit
Asset
Window
Tools
Help
BT_Enemy*
Save
Browse
New Blackboard
New Task
New Decorator
New Service
Behavior Tree
Blackboard
Behavior Tree
Behavior Tree
Zoom 1:1
ROOT
BB_Enemy
Details
Search
AI
BehaviorTree
Blackboard Asset
BB_Enemy
Blackboard
Search
KEYS
EnemyActor
HasLineofSight
PatrolLocation
BEHAVIOR TREE
Content Drawer
Output Log
Cmd
Enter Console Command
Source Control Off

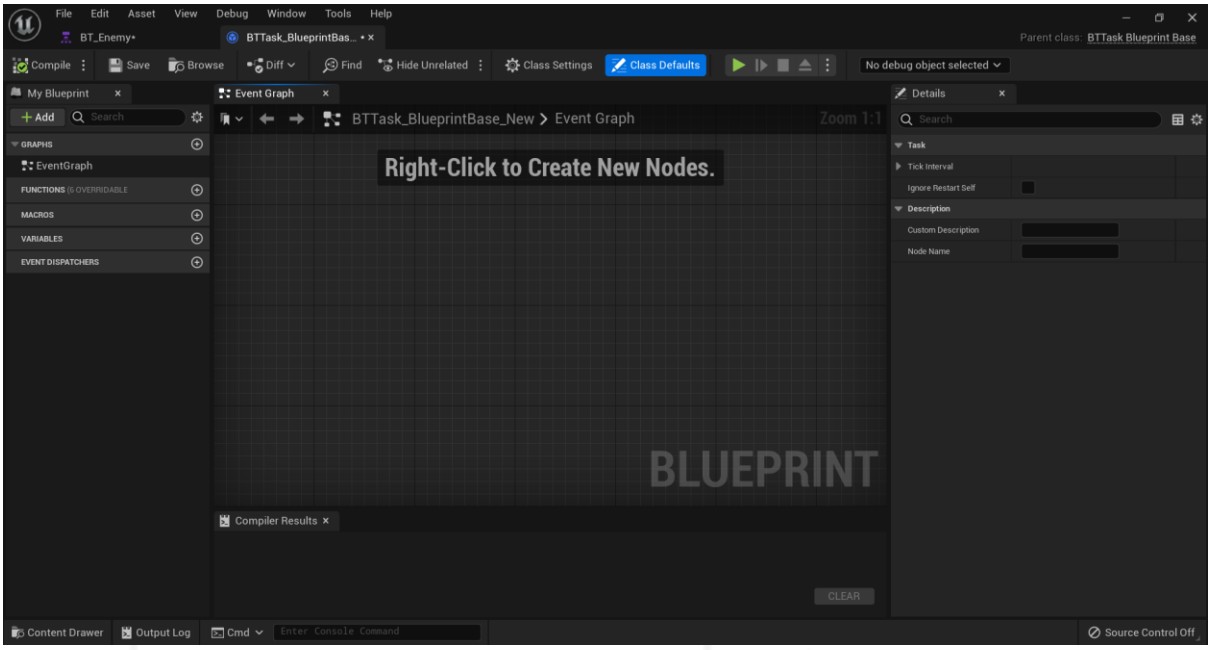

ROOT
BB_Enemy
Search
Composites
Selector
Sequence
Simple Parallel

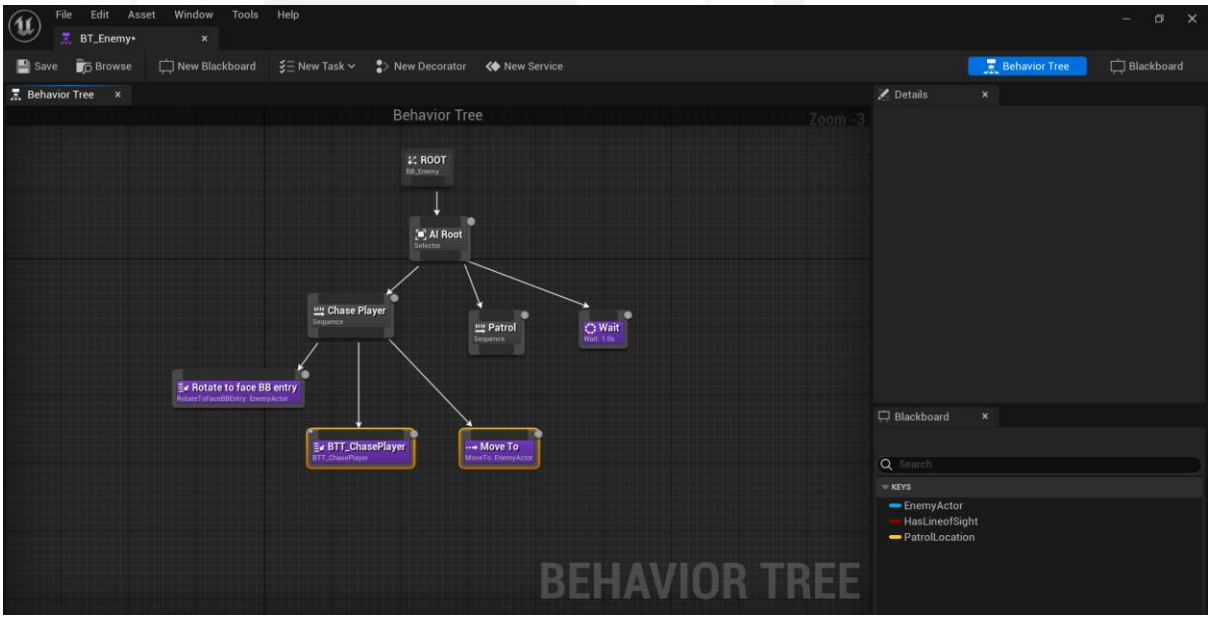

Details
Search
Zoom 1:1
ROOT
BB_Enemy
Composite
Apply Decorator Scope
Description
Node Name
AI Root
AI Root
Selector

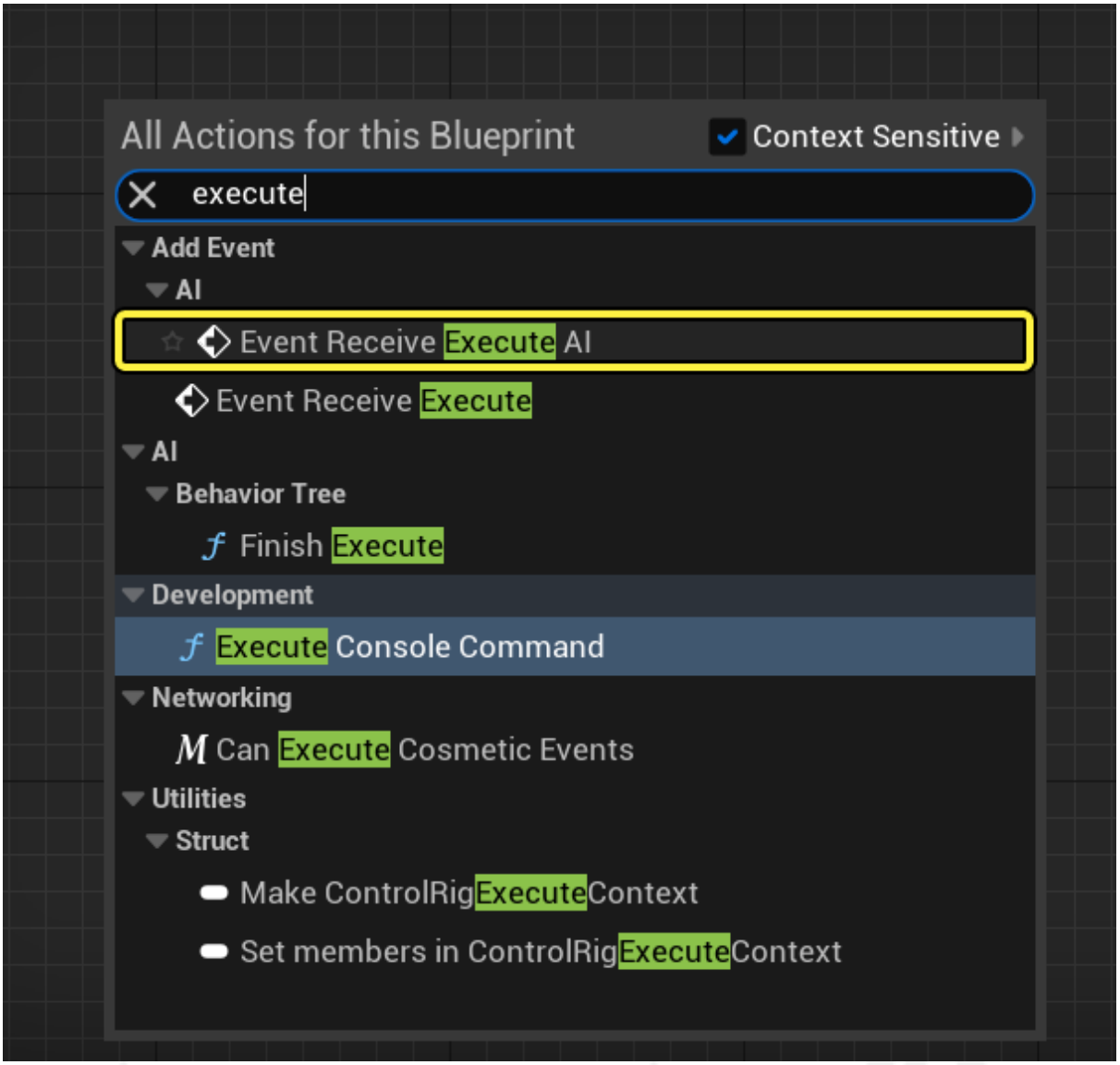

File Edit Asset Window Tools Help
BT_Enemy•
Save
Browse
New Blackboard
New Task
New Decorator
New Service
Behavior Tree
Blackboard
Behavior Tree
Zoom 1:1
ROOT
BB_Enemy
AI Root
Selector
Chase Player
Sequence
Details
Search
Composite
Apply Decorator Scope
Description
Node Name
Chase Player
Blackboard
Search
KEYS
EnemyActor
HasLineofSight
PatrolLocation
BEHAVIOR TREE


0
AI Root
Selector
1
Chase Player
Sequence
2
Patrol
Sequence


File Edit Asset Window Tools Help
BT_Enemy*
Save Browse New Blackboard New Task New Decorator New Service
Behavior Tree
Blackboard
Behavior Tree
Behavior Tree
Zoom -1
ROOT
BB_Enemy
AI Root
Selector
Chase Player
Sequence
Patrol
Sequence
Wait
Wait: 1.0s
Rotate to face BB entry
RotateToFaceBBEntry: EnemyActor
BEHAVIOR TREE
Details
Search
Node
Precision 10.0
Blackboard
Blackboard Key EnemyActor
Task
Ignore Restart Self
Description
Node Name Rotate to face BB entry
Blackboard
Search
KEYS
EnemyActor
HasLineofSight
PatrolLocation


Right-Click to Create New Nodes.
BLUEPRINT


Behavior Tree
ROOT
AI Root
Chase Player
Patrol
Wait
Rotate to face BB entry
BTT_ChasePlayer
Move To
EnemyActor
HasLineofSight
PatrolLocation
BEHAVIOR TREE




**Resultados esperados**

Como resultado del análisis, se ha identificado la configuración técnica necesaria para establecer un sistema de locomoción autónoma mediante el uso de *AI Controllers*. Se ha logrado la implementación de un flujo lógico que vincula el evento de inicio (*BeginPlay*) con la ejecución de rutas calculadas por el nodo AI MoveTo, aprovechando la malla de navegación (*NavMesh*) para evitar obstáculos de forma dinámica. Finalmente, se ha validado la correcta posesión automática del *pawn* y la conectividad entre el controlador y la escena, garantizando que el NPC responda a objetivos espaciales con precisión y eficiencia técnica.

## 6.3 Practica experimental 15

| | | | |
|---|---|---|---|
| **Cargo:** | Docente Especialista | | |
| **Nombre:** | Alexis Urgilés | | |
| **Asignatura:** | Desarrollo de Videojuegos | | |
| **Carrera:** | Alexis Urgilés | **Nivel:** | Cuarto Nivel |
| **Estudiante:** | | | |

**ACTIVIDAD PRÁCTICO EXPERIMENTAL EN EL ENTORNO ACADÉMICO**

**Testing del proyecto**

**Objetivos**

Evaluar el estado actual del proyecto mediante pruebas estructuradas que permitan identificar fallos en gameplay, colisiones, navegación, rendimiento y lógica general, con el fin de generar información útil para mejorar el desarrollo.

**Antecedentes/Escenario**

En la producción de videojuegos, el testing permite detectar errores que comprometen la jugabilidad. Unreal Engine proporciona herramientas internas que permiten observar rendimiento, colisiones, comportamiento de IA y estabilidad del proyecto.

**Recursos necesarios**

Proyecto ejecutado en modo PIE.

Herramientas internas de Unreal para medición de rendimiento.

Cuaderno o archivo digital para documentar errores.

**Planteamiento del problema**

Se requiere ejecutar un proceso de evaluación que permita identificar fallos que puedan afectar la experiencia del jugador o el comportamiento técnico del juego.



(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC

**Pasos por realizar**

Paso 1: Ejecutar el proyecto y observar el comportamiento del jugador y NPCs.

Paso 2: Probar el sistema de colisiones y detectar inconsistencias.

Paso 3: Analizar rendimiento utilizando herramientas internas.

Paso 4: Identificar errores visuales, lógicos o de navegación.

Paso 5: Registrar conclusiones y mejoras necesarias.

**Desarrollo**

El proceso inició ejecutando el proyecto en Play In Editor con el objetivo de observar el comportamiento general del personaje principal y de los NPCs controlados por IA. Se verificó la capacidad de movimiento, la respuesta a las colisiones y la fluidez de la interacción en el entorno. Luego se habilitó la visualización de colisiones para detectar superficies mal configuradas o comportamientos inesperados, como objetos atravesados o zonas sin colisión. Posteriormente se evaluó el rendimiento utilizando herramientas como Stat FPS, verificando la cantidad de frames por segundo, consumo de GPU, CPU y tiempos de render. Se registraron posibles caídas de rendimiento en zonas con demasiados objetos o iluminación pesada. Durante la prueba se identificaron posibles fallos como rutas incompletas de navegación, animaciones desincronizadas o materiales incorrectos. Cada error se registró detalladamente con fecha y descripción, acompañado de su posible causa. Finalmente se elaboró un resumen de mejoras que servirán como guía para optimizar el proyecto, reforzar la jugabilidad y garantizar estabilidad en versiones futuras.

**Resultados esperados**

Como resultado del análisis, se ha identificado la viabilidad técnica y operativa del proyecto mediante la ejecución de pruebas en entornos controlados (*PIE*). Se ha logrado la detección de inconsistencias en el sistema de colisiones y rutas de navegación, permitiendo una depuración precisa de la lógica del juego. Finalmente, se ha consolidado un registro de métricas de rendimiento (FPS y tiempos de render) que facilita la optimización de recursos, garantizando una experiencia de usuario estable y alineada con los estándares de calidad exigidos en el desarrollo profesional.


(+593) 96 356 1961 admisiones@itq.edu.ec Antonio de Ulloa N28-30 y Diego de Atienza (Esq). WWW.ITQ.EDU.EC