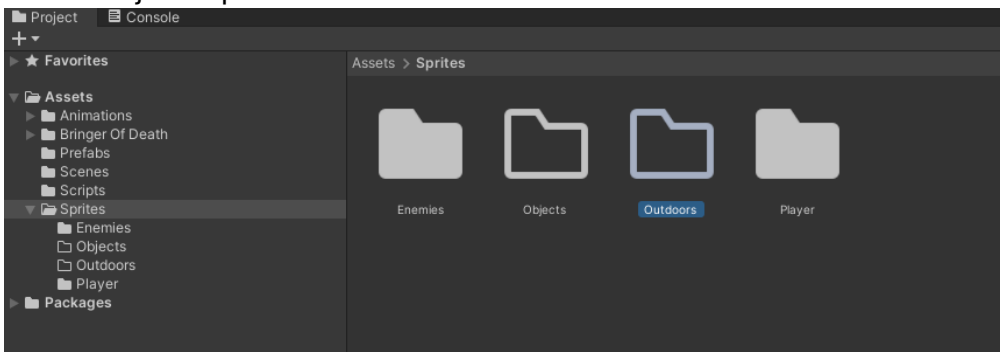
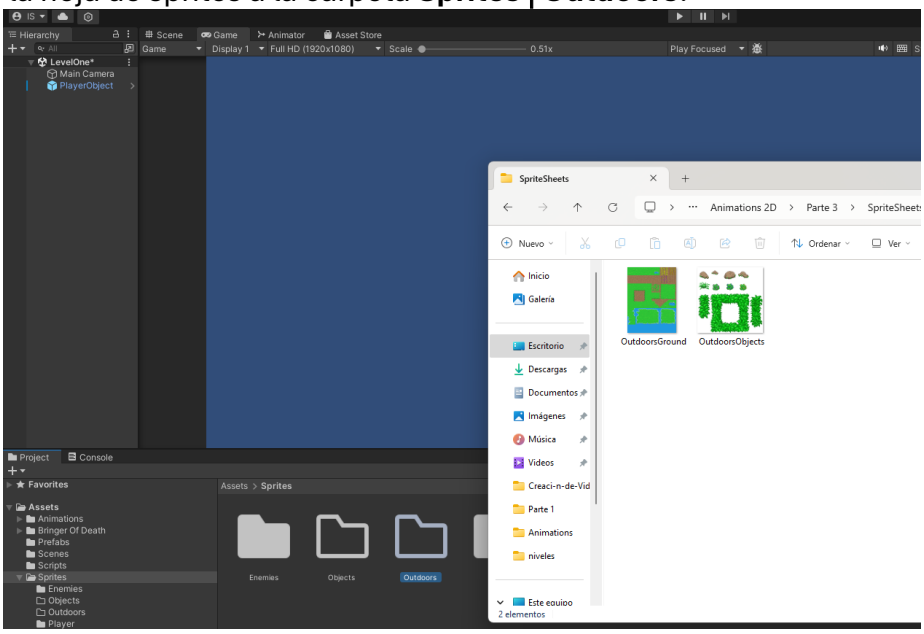
Creamos nuevas carpetas en el directorio **Sprites** tituladas: **"Objects"** y **"Outdoors"**. Usaremos estas carpetas para almacenar las hojas de sprites y las partes utilizadas para nuestros **Tilemaps** externos, así como para los diversos objetos que colocaremos en nuestro mundo.



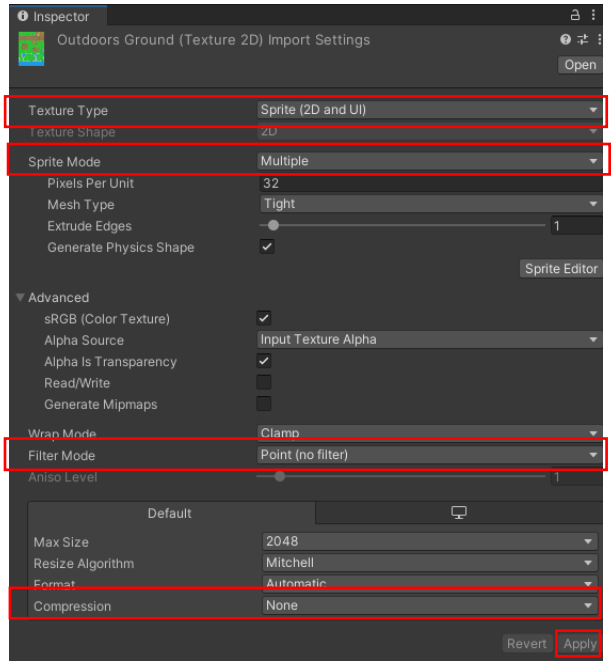
Buscamos la hoja de sprites titulada **"OutdoorsGround.png"**, que viene como recurso de este tutorial. Luego, arrastramos la hoja de sprites a la carpeta **Sprites | Outdoors**.



En la vista **Inspector**, establecemos la configuración de la imagen de la siguiente manera (según lo que se indique en la figura proporcionada):

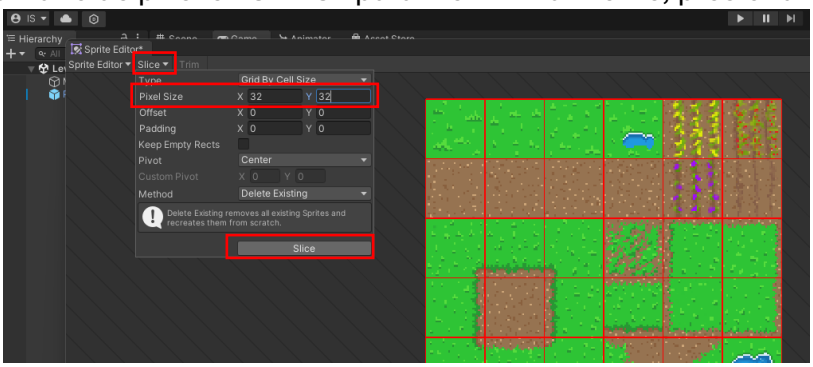
1. **Sprite Mode**: Cambiamos a **Multiple** (para usar varios sprites desde la misma hoja).
2. **Pixels Per Unit**: Aseguramos que esté configurado en el valor adecuado (por ejemplo, 100).
3. **Filter Mode**: Establecemos como **Point** (para evitar el suavizado de los píxeles).
4. **Compression**: Lo ponemos en **None** (si es necesario, para evitar la compresión de la imagen).

Una vez hecho esto, hacemos clic en el botón **Apply** para guardar la configuración.



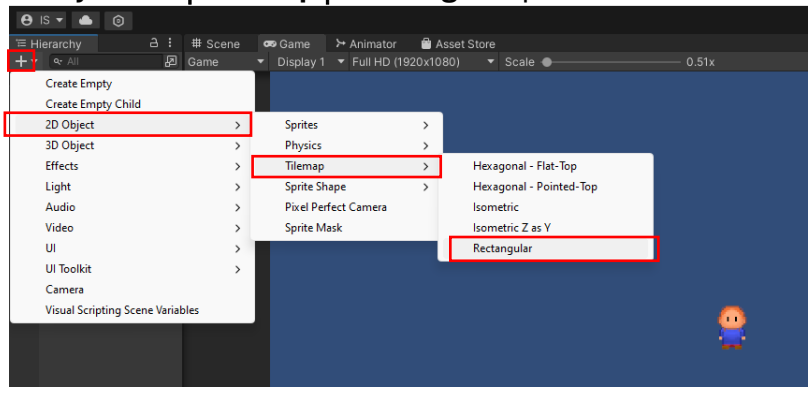
Ahora vamos a cortar la hoja de sprites que acabamos de importar. Entrar en el Sprite Editor haciendo clic en su botón respectivo en la vista Inspector.

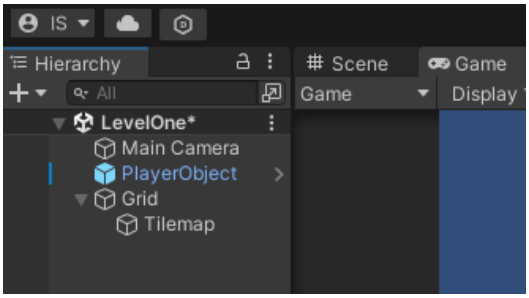
Presionamos el botón **Slice** en la parte superior izquierda y luego seleccionamos la opción **Grid By Cell Size**. Establecimos el tamaño de píxel en **32 × 32** para **X** e **Y**. Finalmente, presionamos el botón **Slice**.



Verificamos que las líneas de corte resultantes se vean bien y luego presionamos el botón **Apply** en la esquina superior derecha del **Sprite Editor**. Ahora tenemos nuestro conjunto de **tiles** externos listo para usarse.

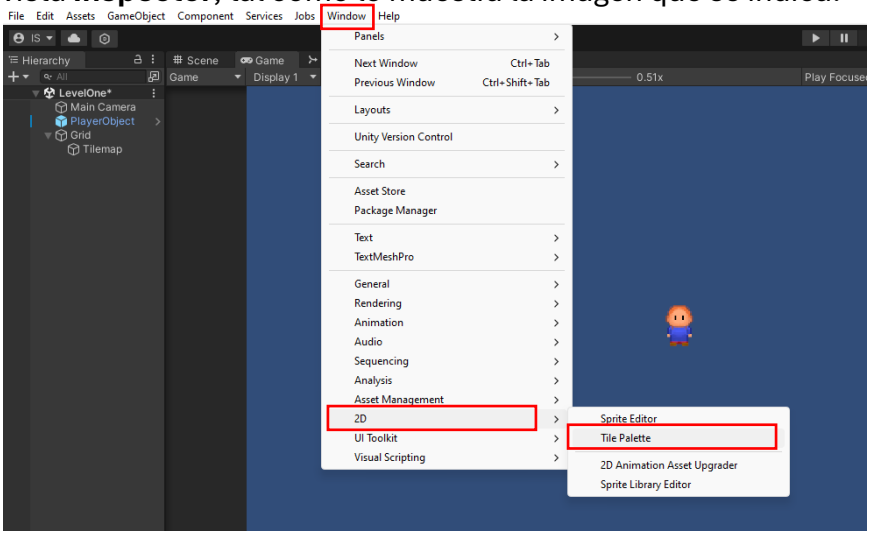
A continuación, para crear nuestros **Tilemaps**, en la vista **Hierarchy**, hicimos clic en el botón **+** y seleccionamos la opción **Objeto 2D | Tilemap | Rectangular** para crear un **GameObject** de **Tilemaps**.

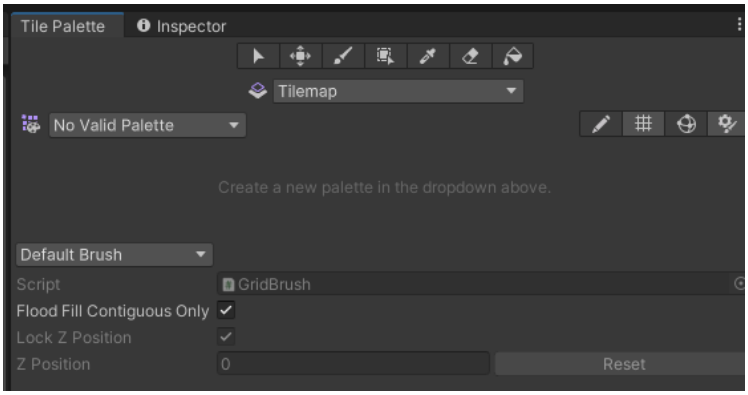




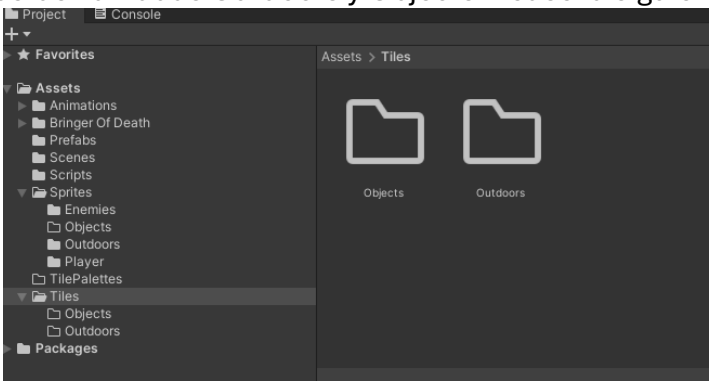
Antes que podamos dibujar, necesitamos crear Tilemaps, que están hechas de tiles individuales.

Vamos al menú **Window | 2D | Tile Palette** para mostrar la vista **Tile Palette**. Luego, acoplamos el panel en la misma área que la vista **Inspector**, tal como lo muestra la imagen que se indica.

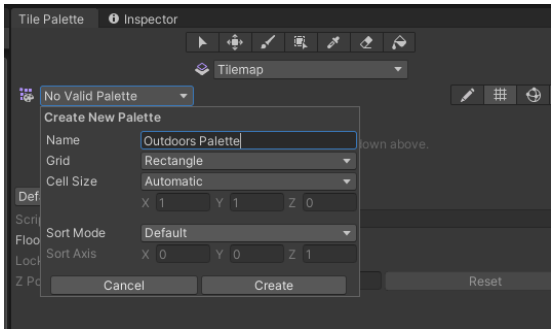




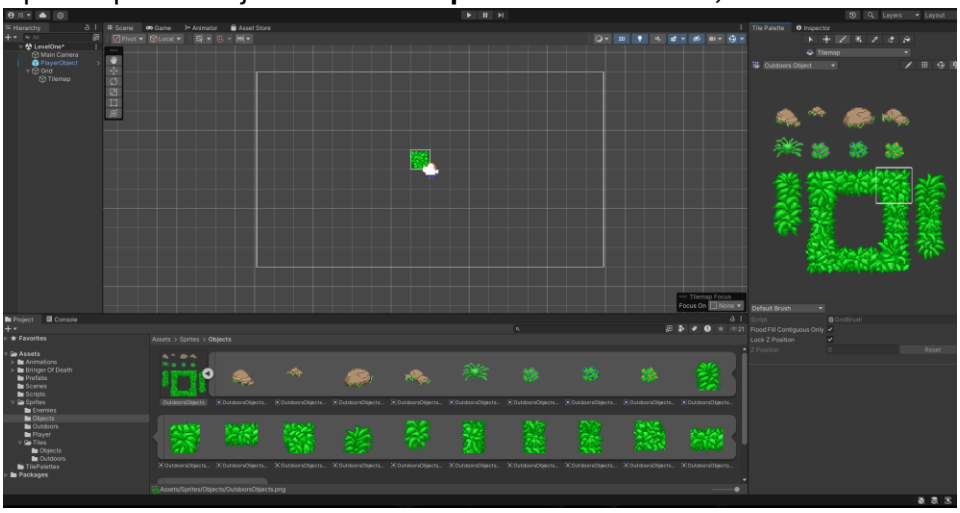
Queremos que el proyecto se mantenga organizado, así que crear una carpeta dentro del proyecto, dentro de la carpeta Assets llamada TilePalettes, luego cree otra carpeta llamada Tiles en la carpeta Sprites. En la carpeta Tiles, crea dos carpetas llamadas Outdoors y Objects. Véase la siguiente figura

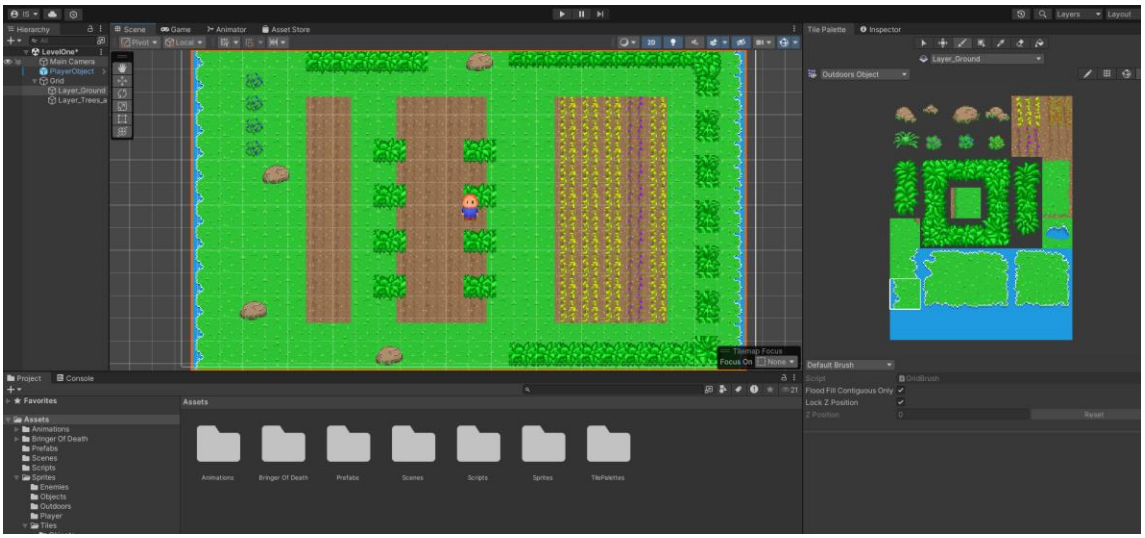


Seleccionamos el botón **Create New Palette** en la ventana **Tile Palette**, nombramos la paleta como **"Outdoors Palette"** y dejamos la configuración de **Cuadrícula** y **Tamaño de celda** tal como se asigna por defecto.



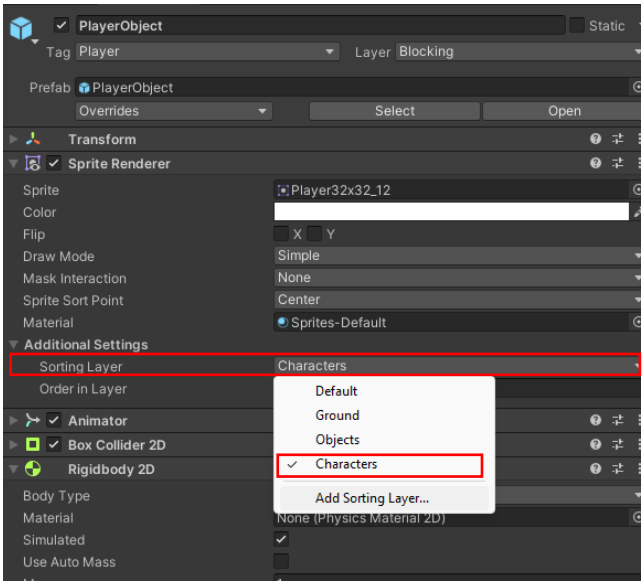
Seleccionamos la herramienta **Pincel** en la vista **Tile Palette**, luego seleccionamos un **Tile** de la vista **Tile Palette**. Usamos el pincel para dibujar en el **Tilemap** en la **Vista Scene**, colocando los tiles en el escenario.





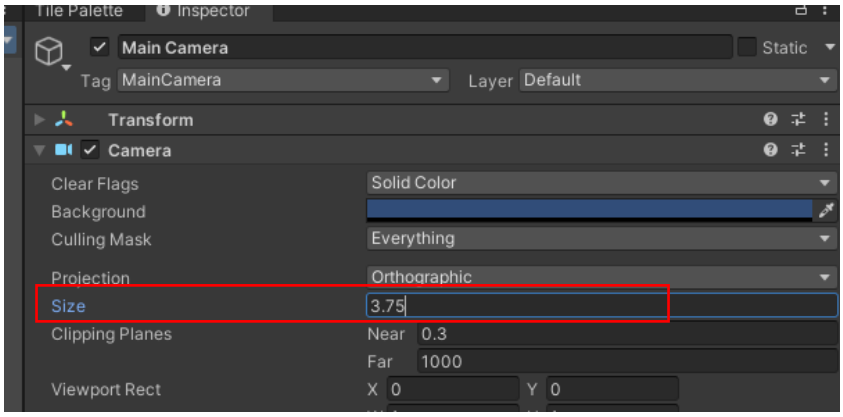
Si queremos asegurarnos de que el Player se renderice en la parte superior del suelo y rocas. Establezcamos Sorting Layer del jugador.

Seleccionamos el **PlayerObject**, luego localizamos la propiedad **Sorting Layer** en el componente **Sprite Renderer**. Presionamos el botón **Add Sorting Layer** y añadimos una nueva capa de orden llamada **"Characters"**. Posteriormente, movimos esta capa hacia la parte inferior, ubicándola después de las capas **Ground** y **Objects**.

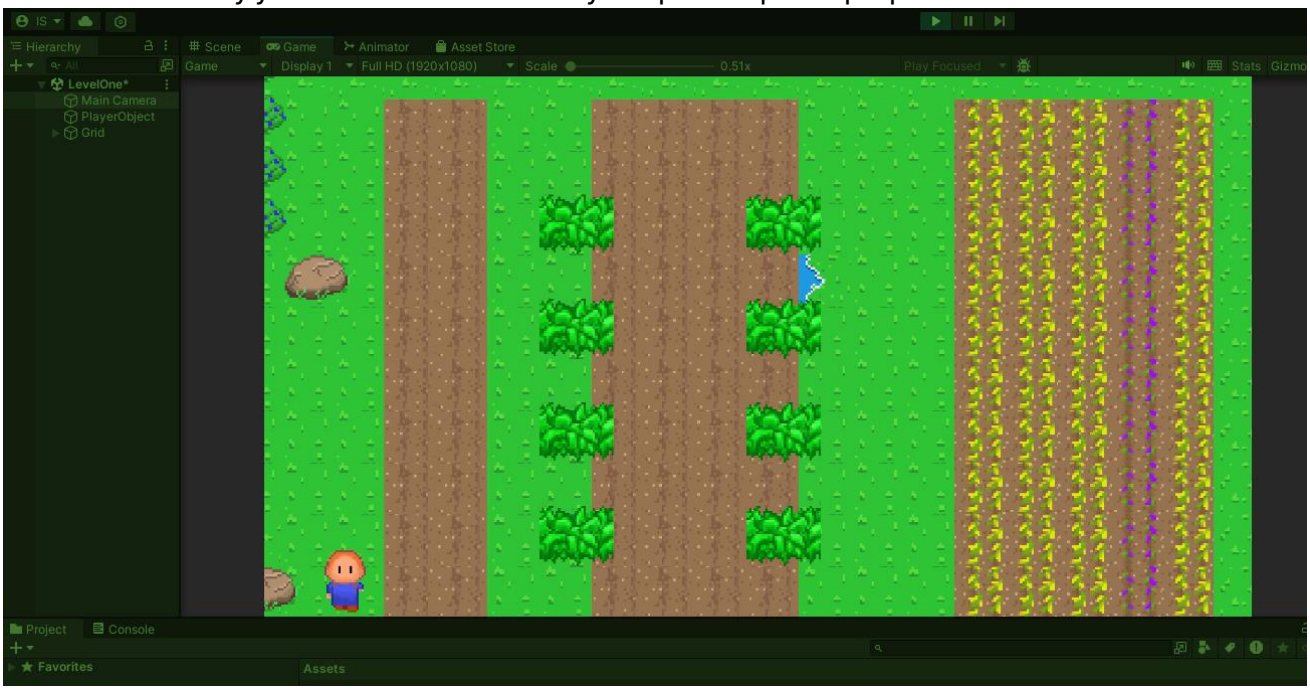


Ahora le hemos dicho al Sprite Renderer que renderice los objetos en orden desde la primera capa de clasificación, "Ground" hasta la última clasificación Capa "Characters".

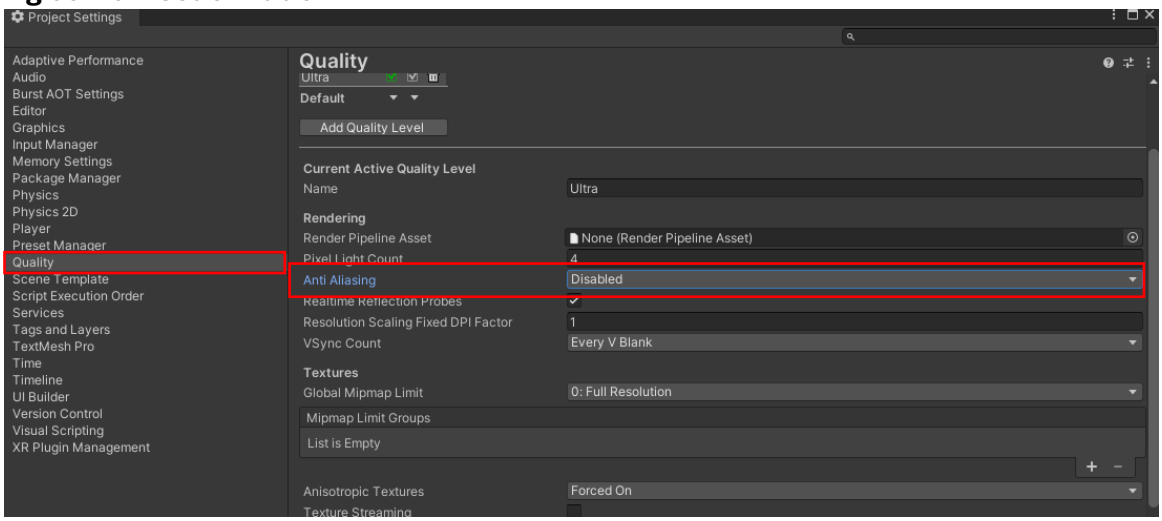
Por ahora, seleccionamos el objeto de la **Cámara** y cambiamos la propiedad **Size** a **3.75**. Más adelante explicaremos cómo funciona la cámara en detalle.



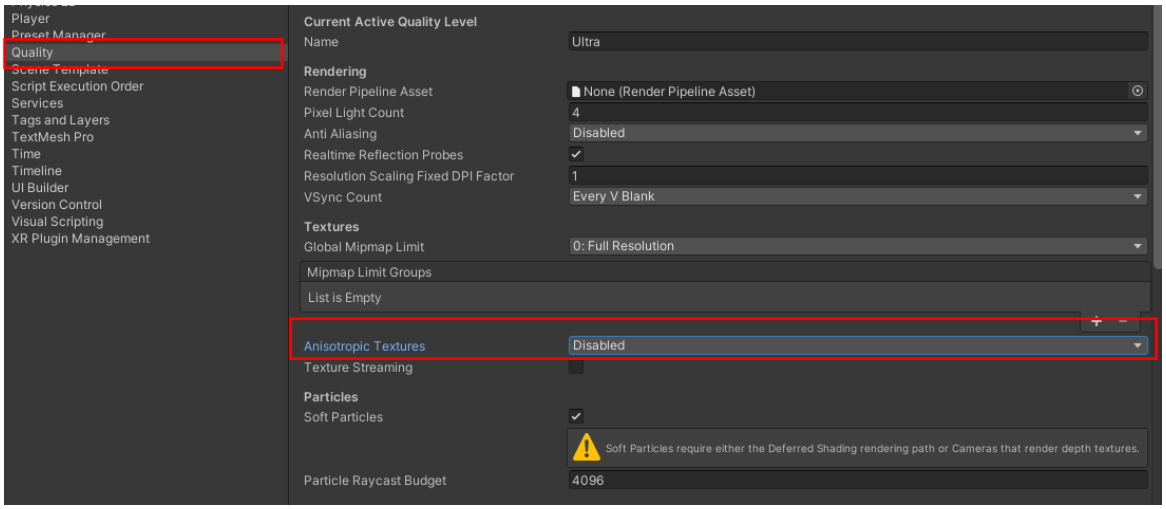
Pulsamos el botón Play y llevamos a nuestro Player a pasear por la pequeña isla.



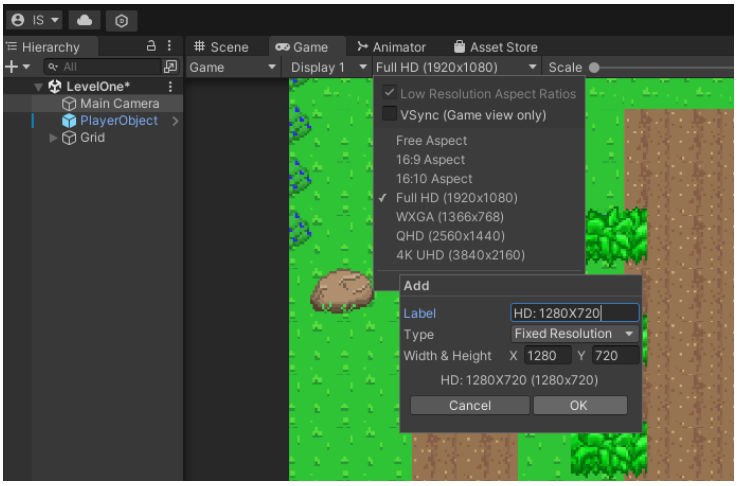
Para desactivar el suavizado, fuimos a la opción **Edit | Project Settings | Quality** y configuramos la opción **Anti-Aliasing** como **Desactivado**.



Desde el mismo menú **Edit | Project Settings | Quality**, desactivamos la opción **Anisotropic Textures**. El filtrado anisotrópico mejora la calidad de imagen para ciertas perspectivas de cámara, pero no es necesario para este proyecto.

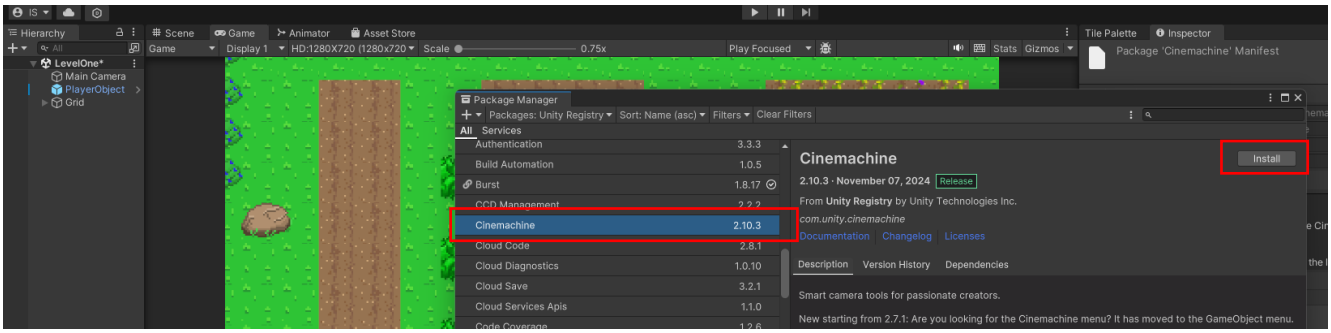


En la parte inferior del menú desplegable, presionamos el signo + para abrir una ventana donde se puede ingresar una nueva resolución. Creamos una resolución personalizada de 1280 × 720.

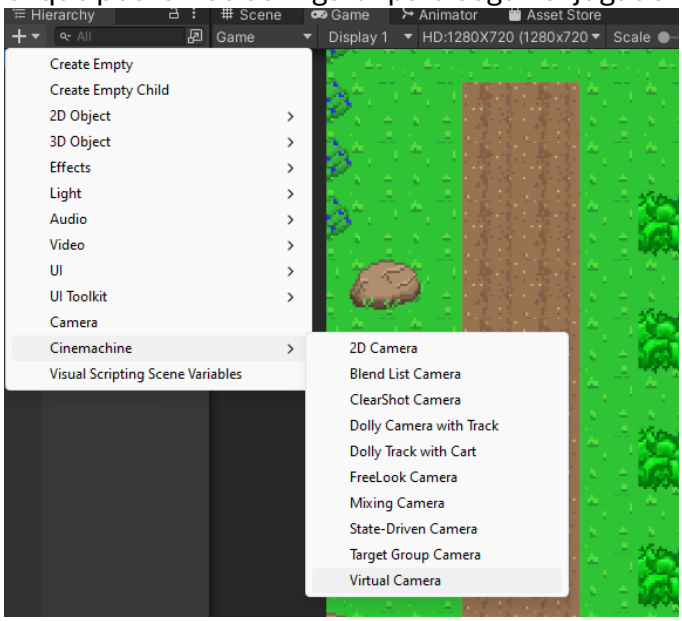


En los menús, seleccionamos **Window | Package Manager**, lo que hizo que apareciera la ventana del **Unity Package Manager**.

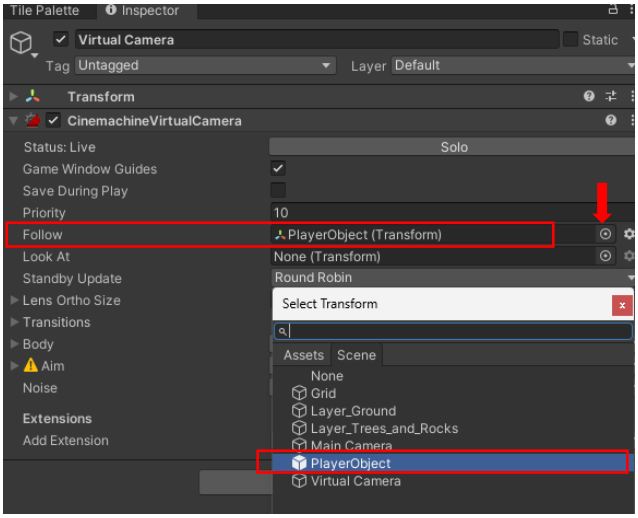
Hicimos clic en el botón **Install** en la parte inferior derecha para instalar **Cinemachine**. Una vez completada la instalación, cerramos la ventana del **Package Manager**. Ahora, podemos ver una nueva carpeta llamada **Packages** en la vista **Proyecto**, que incluye **Cinemachine**.



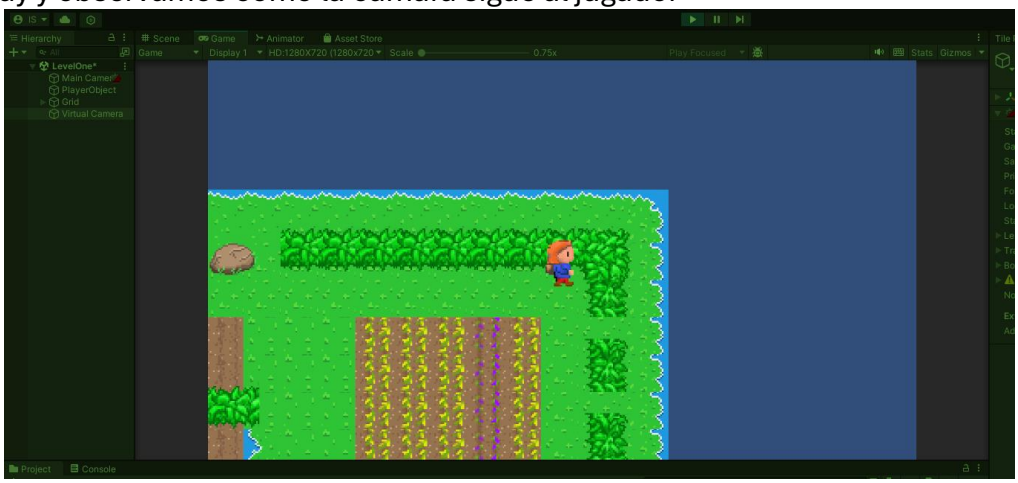
Seleccionamos **LevelOne** en la vista **Hierarchy** y añadimos una **Cinemachine | Virtual Camera** al nivel. Esto agrega una cámara virtual que podremos configurar para seguir al jugador u otros elementos del juego.



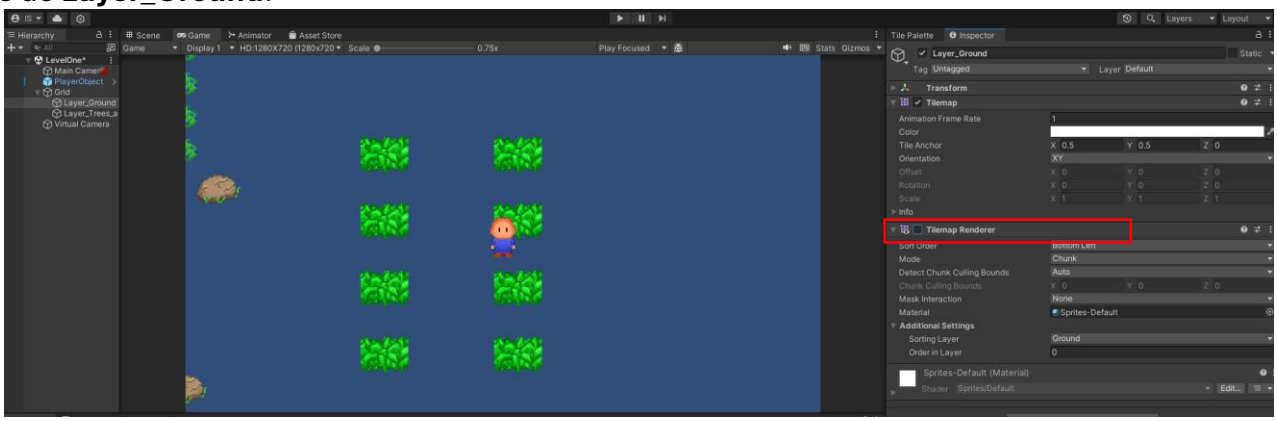
Seleccionamos la **Cámara Virtual** en la jerarquía y arrastramos el objeto **PlayerObject** a la propiedad llamada **Follow** en el Inspector. Esto configura la cámara para que siga al jugador mientras se mueve en el juego.

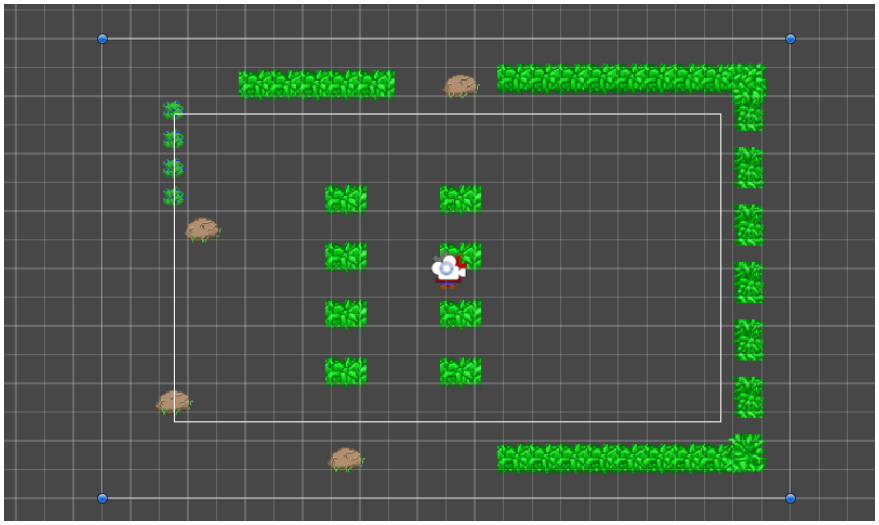


Presionamos play y observamos como la cámara sigue al jugador

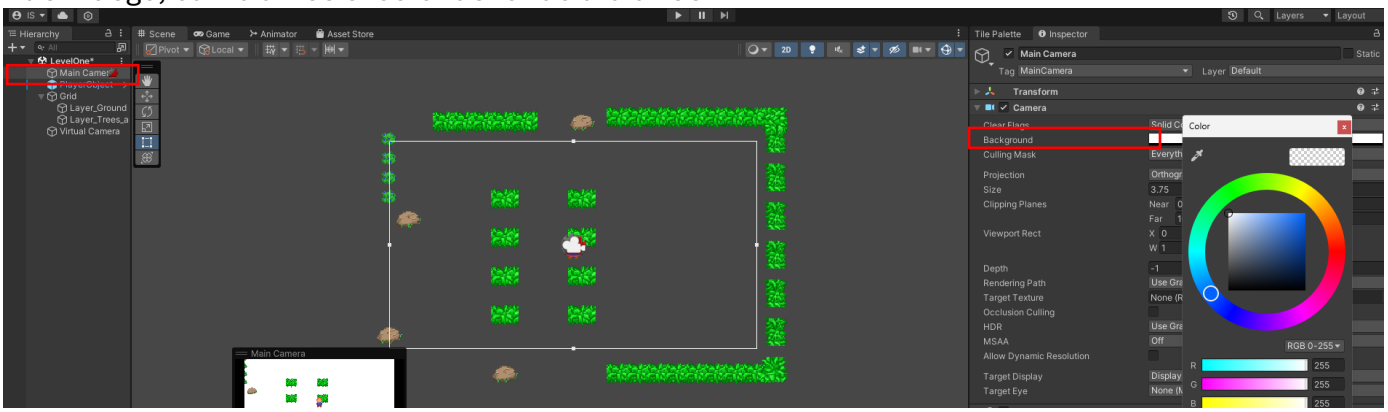


Para tener una mejor idea de los parámetros que rigen el movimiento de la cámara, ocultamos la capa **Ground**. Seleccionamos el objeto **Layer_Ground Tilemap** en la vista **Hierarchy** y desmarcamos el cuadro junto al componente **Tilemap Renderer** de **Tilemap** para desactivarlo. Ahora, Unity no renderiza el mapa de tiles de **Layer_Ground**.

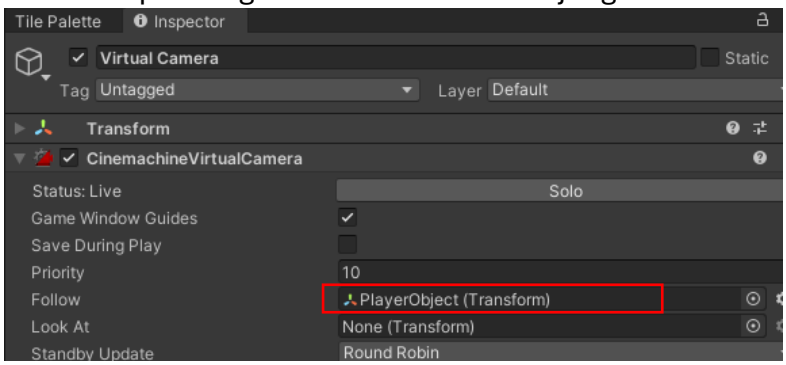




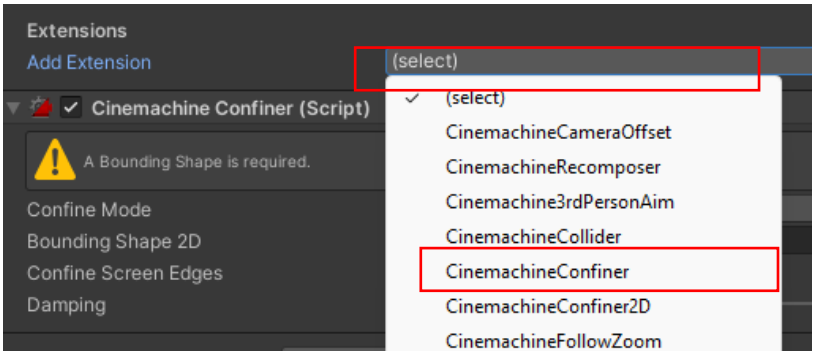
Hacemos clic en el objeto **Main Camera** en la vista **Hierarchy** y presionamos el cuadro de color que dice **Fondo**. Luego, cambiamos el color de fondo a **blanco**.



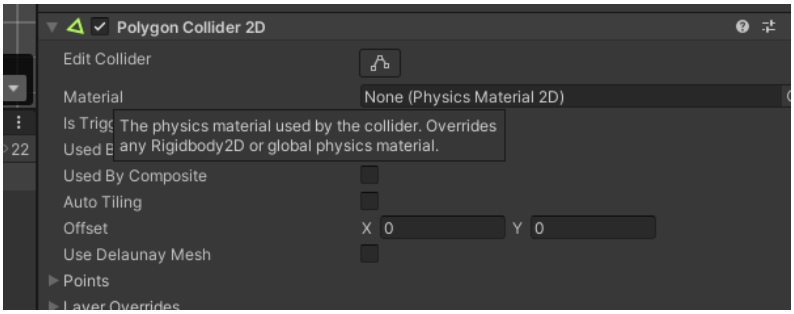
Seleccionamos la **Cámara Virtual** y nos aseguramos de que la propiedad **Game Window Guides** esté marcada. Luego, experimentamos moviendo al jugador para observar cómo la cámara sigue al jugador dentro de los límites establecidos por las guías de la ventana del juego.



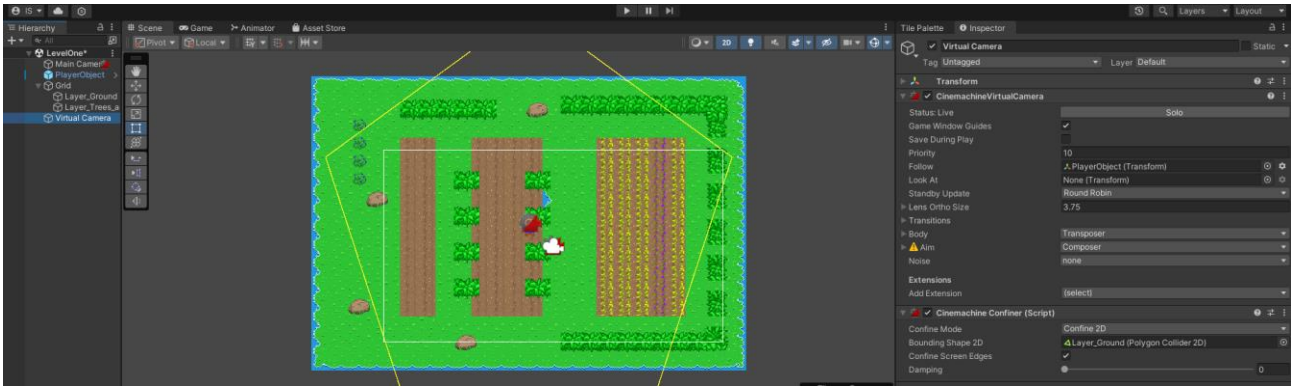
Seleccionamos la **Cámara Virtual** en la vista **Hierarchy**. En la ventana **Inspector**, junto a **Add Extension**, seleccionamos **CinemachineConfiner** en el menú desplegable. Esto agrega un componente **Cinemachine Confiner** a nuestra **Cámara 2D Cinemachine**, lo que limita el movimiento de la cámara dentro de un área específica.



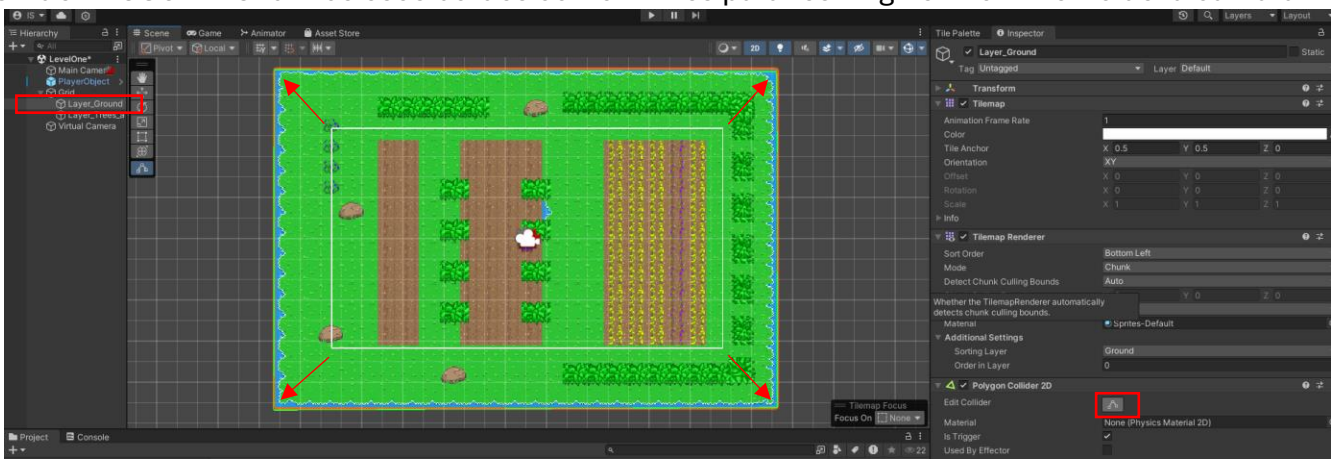
El **CinemachineConfiner** requiere un **Composite Collider 2D** o un **Polygon Collider 2D** para determinar los bordes del confinamiento. Para hacerlo, seleccionamos el objeto **Layer_Ground** y le agregamos un **Polygon Collider 2D**. Esto permitirá que el componente **Cinemachine Confiner** utilice los límites del collider para restringir el movimiento de la cámara.



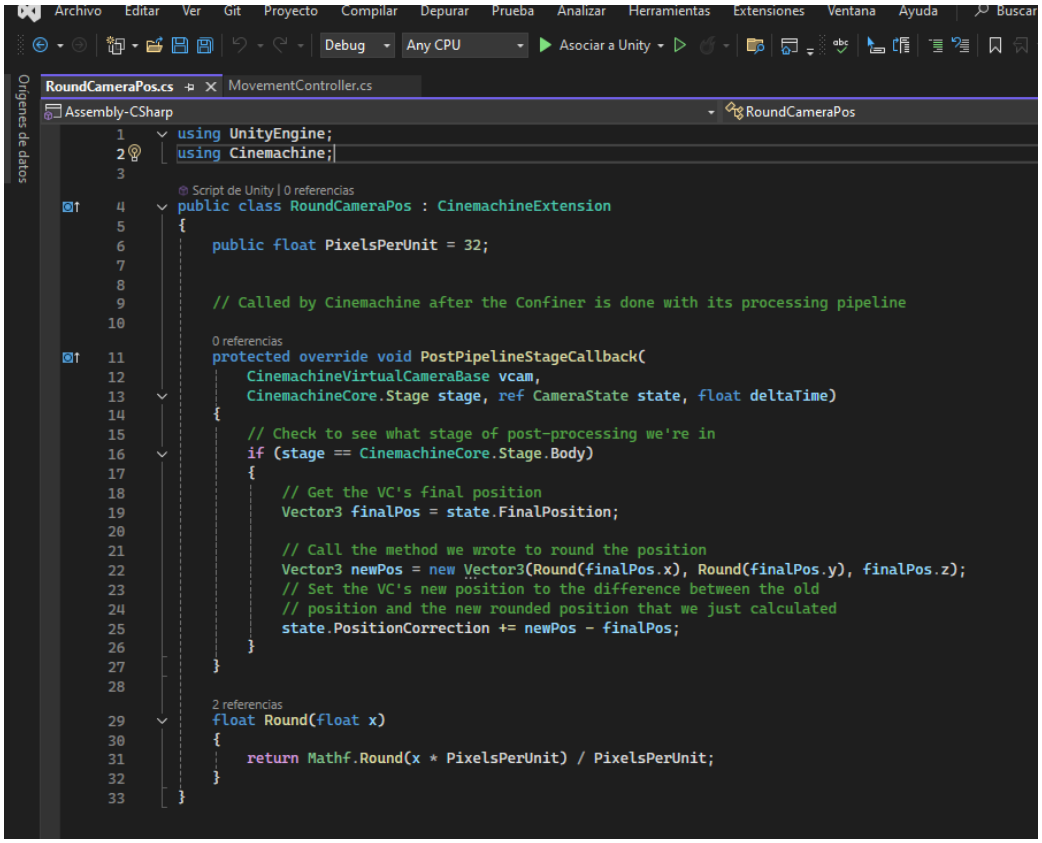
Arrastramos Layer_Ground a la pestaña inspector en el apartado de Cinemachine Confiner | Building Shape 2D

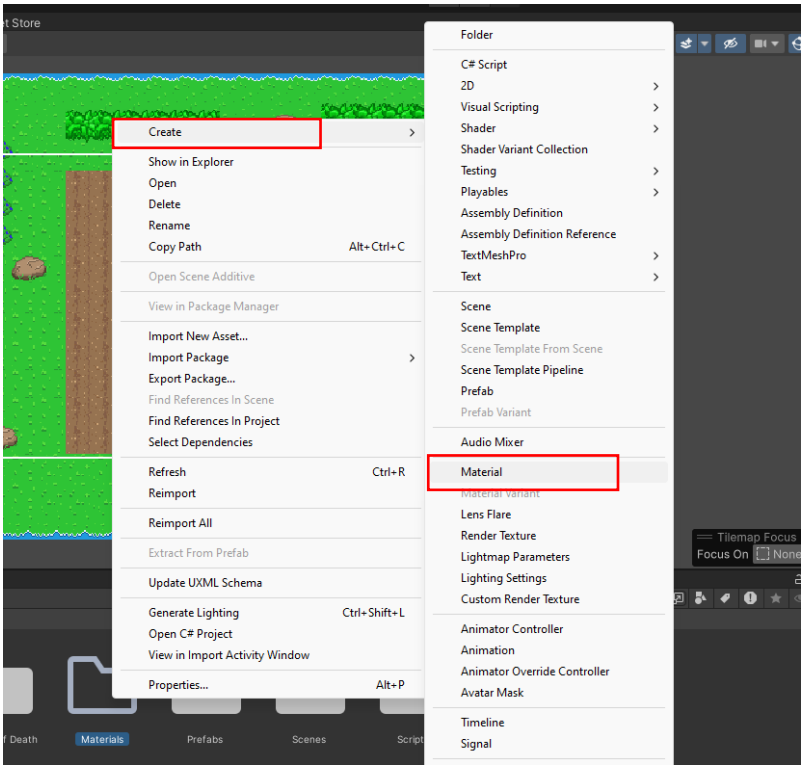


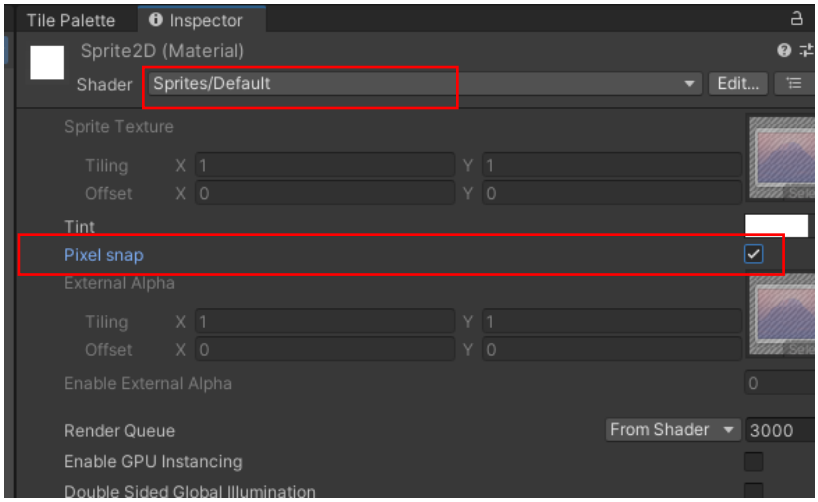
Hacemos clic en el botón **Editar Collider** en el componente **Polygon Collider 2D** y editamos el colisionador para que delimite los bordes de nuestro nivel **Layer_Ground**. Esto permite que el **CinemachineConfiner** utilice esos bordes como límites para restringir el movimiento de la cámara.



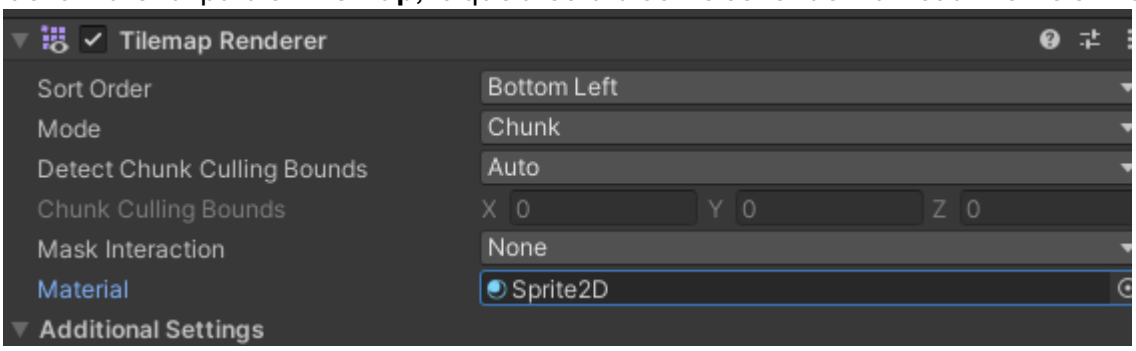
Presionamos el botón **Play** y caminamos hacia el borde de la pantalla. El jugador ha caminado fuera de la zona muerta, y mientras el punto de seguimiento ha seguido moviéndose con el jugador, la cámara se ha detenido, ya que el **Cinemachine Confiner** ha restringido su movimiento dentro de los límites del collider configurado.


Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda
Debug
Any CPU
Asociar a Unity
RoundCameraPos.cs
MovementController.cs
Assembly-CSharp
RoundCameraPos
Orígenes de datos
using UnityEngine;
using Cinemachine;
Script de Unity | 0 referencias
public class RoundCameraPos : CinemachineExtension
{
public float PixelsPerUnit = 32;
// Called by Cinemachine after the Confiner is done with its processing pipeline
0 referencias
protected override void PostPipelineStageCallback(
CinemachineVirtualCameraBase vcam,
CinemachineCore.Stage stage, ref CameraState state, float deltaTime)
{
// Check to see what stage of post-processing we're in
if (stage == CinemachineCore.Stage.Body)
{
// Get the VC's final position
Vector3 finalPos = state.FinalPosition;
// Call the method we wrote to round the position
Vector3 newPos = new Vector3(Round(finalPos.x), Round(finalPos.y), finalPos.z);
// Set the VC's new position to the difference between the old
// position and the new rounded position that we just calculated
state.PositionCorrection += newPos - finalPos;
}
}
2 referencias
float Round(float x)
{
return Mathf.Round(x * PixelsPerUnit) / PixelsPerUnit;
}
}

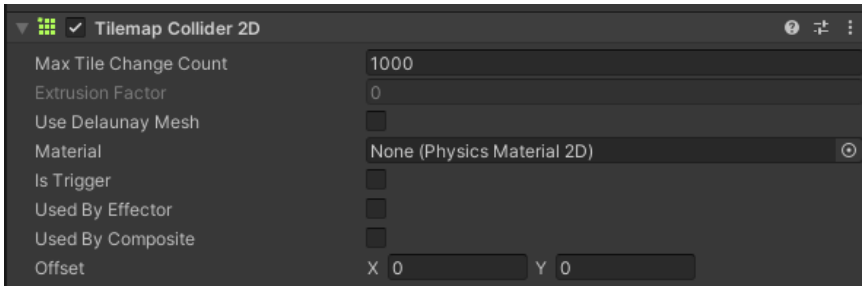

Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Find References In Project
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Update UXML Schema
Generate Lighting Ctrl+Shift+L
Open C# Project
View in Import Activity Window
Properties... Alt+P
Folder
C# Script
2D
Visual Scripting
Shader
Shader Variant Collection
Testing
Playables
Assembly Definition
Assembly Definition Reference
TextMeshPro
Text
Scene
Scene Template
Scene Template From Scene
Scene Template Pipeline
Prefab
Prefab Variant
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Timeline
Signal
Materials
Prefabs
Scenes

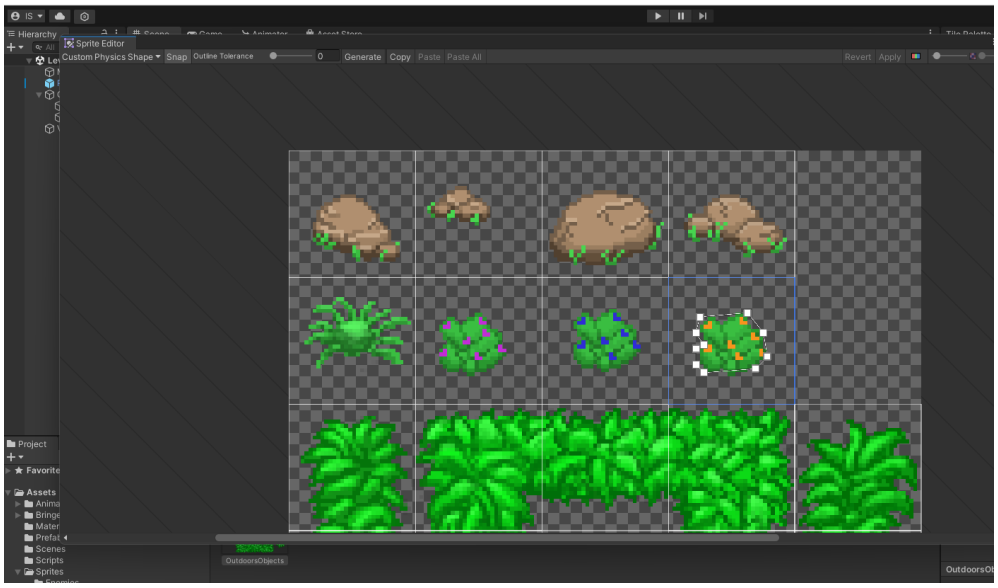


Seleccionamos el **Tilemap Layer_Ground** y cambiamos el material en el componente **Tilemap Renderer** haciendo clic en el punto junto a la propiedad **Material**. Esto abrirá un selector de material donde podemos asignar un nuevo material para el **Tilemap**, lo que afectará cómo se renderiza visualmente el mapa de tiles.



Seleccionamos **Layer_Trees_and_Rocks** en la vista **Hierarchy** y luego presionamos el botón **Add Component** en la vista **Inspector**. Buscamos y agregamos un componente llamado **Tilemap Collider 2D**. Esto permitirá que los objetos en este **Tilemap** interactúen con la física del juego, como la detección de colisiones.





Ahora presione el botón Play y vea como los colisionadores están funcionando