

Guía de desarrollo de videojuego 2D con Unity

Henry A. Gutierrez Condori

Ingeniería de Sistemas e Informática,
Universidad Nacional de Moquegua

Programación de Videojuegos I

Ing. Danyer A. Valencia Llamoca

23 de noviembre de 2020

Introducción

En el presente trabajo se detalla la mayoría de las acciones realizadas para crear el videojuego con el motor de desarrollo de unity.

Se inicia con acciones básicas y adentrando al lector al conocimiento del manejo de las herramientas que ha de utilizarse a menudo al desarrollar el proyecto.

Se cuenta con gran variedad de figuras, con detalles y opciones marcadas para responder y apoyar en las preguntas o dudas que el lector tuviera.

A medida que se vayan completando las etapas de desarrollo, se irán obviando algunos pasos, ya que en un inicio se hicieron detalladamente, se sugiere tomar apuntes, notas o regresar a pasos anteriores, de esa forma evitar retrasos o conflictos, procure leer con atención y no dejar pasar los detalles.

En los anexos se colocará el enlace hacia el repositorio para que pueda descargar el proyecto en su totalidad.

Espero que el trabajo le sea de gran ayuda.

Índice de contenido

Introducción.....	1
1. Herramientas a utilizar.....	10
1.1. Motor de desarrollo.....	10
1.2. Editor de código o Entorno de desarrollo.....	10
1.3. Requisitos mínimos del “Hardware”.....	10
2. Crear Proyecto.....	11
2.1. Iniciar Unity Hub.....	11
2.2. Crear proyecto nuevo.....	11
3. Creación de Carpetas.....	12
3.1. Creando una carpeta.....	12
3.2. Carpetas creadas.....	13
4. Importando Assets.....	13
4.1. Buscando un Assets gratuito.....	13
4.2. Descargando e importando “Pixel Adventure 1”.....	13
5. Configuración de las ventanas de trabajo “Layouts”.....	14
5.1. Vista de trabajo 2 by 3.....	14
5.2. Entorno de trabajo listo.....	14
6. Creando el mundo virtual del videojuego.....	15
6.1. Creando el piso.....	15
6.2. Escalar el bloque.....	16
6.3. Desplazar el bloque.....	16
6.4. Duplicando un bloque.....	17
6.5. Cambiar modelo del bloque.....	17
6.6. Acoplamiento de bloques.....	17
6.7. Creando nuevo objeto de juego.....	18
6.8. Configurar nuevo objeto de juego.....	18

	3
6.9. Agrupar objetos.....	19
6.10. Agregar Box Collider.....	19
6.11. Configurando el Box Collider.....	20
6.12. Crear un jugador.....	20
6.12.1. Creando un juador.....	21
6.12.2. Guardar animación del jugador.....	21
6.12.3. Agregar colisión y gravedad al jugador.....	22
6.12.4. Prueba en modo juego.....	22
6.13. Creando el objeto Canvas.....	22
6.13.1. Creando un Text Mesh Pro.....	23
6.13.2. Configurando Canvas.....	23
6.13.3. Vincular Canvas con la Cámara Principal.....	24
6.13.4. Agregar un objeto al Canvas de tipo Imagen.....	24
6.13.5. Agregar un sprite al objeto imagen.....	25
6.13.6. Importar imágenes.....	25
6.13.7. Agregar una imagen de fondo.....	26
6.14. Creando la barra de vida.....	26
6.14.1. Estructura de la barra de vida.....	26
6.14.2. Agregar un Sprite Render.....	27
6.14.3. Creando un Sprite cuadrado.....	27
6.14.4. Vincular los objetos con el sprite square.....	28
6.14.5. Cambiar la cantidad de pixel por unidad.....	28
6.14.6. Cambiar position X del objeto “BarraSprite”.....	29
6.14.7. Cambiar position X del objeto “Barra”.....	29
6.14.8. Diseño final de la barra de vida y agrupación.....	29
6.14.9. Agregar un tag a un objeto.....	30
6.15. Agregar zona de muerte.....	30

	4
6.16. Agregar frutas al juego.....	31
6.16.1. Primera fruta del juego.....	31
6.16.2. Escalar y agregar componente Circle Collider 2D.....	31
6.17. Agregar plugin para la cámara.....	32
6.17.1. Abrir el administrador de paquetes.....	32
6.17.2. Buscar e instalar Cinemachine.....	32
6.17.3. Agregar objeto cámara 2D desde virtual machine.....	33
6.17.4. Agregar jugador a la cámara de cinemachine.....	33
6.17.5. Configurar del área de la cámara.....	34
6.18. Agregar un enemigo.....	34
6.18.1. Agregar un detector de suelo para el enemigo.....	35
6.19. Crear nueva escena.....	35
6.19.1. Crear Scena.....	35
6.19.2. Diseñar scena gameover.....	36
7. Crear script para probar el desplazamiento del jugador.....	36
7.1. Script de prueba para el jugador.....	36
7.1.1. Editando el script creado.....	37
7.1.2. Agregar el script editado al objeto Player.....	37
8. Creando la estructura de scripts para el proyecto.....	38
8.1. Creando la estructura de los scripts.....	38
9. Editando los Scripts C#.....	39
9.1. Carpeta GameState.....	39
9.1.1. Script GameOver.cs.....	39
9.1.2. Agregar el escript al botón y el botón al script.....	40
9.1.3. Script GameTime.cs.....	40
9.1.4. Agregar el script al padre del texto y el texto al script.....	41
9.2. Carpeta Enemy.....	42

	5
9.2.1. Script Enemy.cs.....	42
9.2.2. Agregar el escript al enemigo y el detector de suelo del enemigo al script.....	43
9.3. Carpeta Player.....	44
9.3.1. Script PlayerHeartScore.cs.....	44
9.3.2. Agregamos el script anterior en el txtMPVidas.....	45
9.3.3. Script PlayerHeart.cs.....	45
9.3.4. Agregando el script anterior al objeto Heart.....	46
9.3.5. Script PlayerHelth.cs.....	47
9.3.6. Agregamos el script anterior al objeto barra de vida.....	47
9.3.7. Script PlayerFood.cs.....	48
9.3.8. Agregar el script anteroir a los objetos fruta.....	49
9.3.9. Script Player.cs.....	49
9.3.10. Agregando el script al objeto Player y componentes/objetos al script.....	54
10. Extras.....	55
10.1. Animar recojo de fruta.....	55
10.2. Player corriendo.....	55
10.2.1. Crear nueva animación.....	55
10.2.2. Agregando los sprites a la nueva animación.....	56
10.2.3. Configurar a +- 30 segundos.....	56
10.2.4. Abrir ventana animator y vincular una transición.....	57
10.2.5. Configurar animación de estado IDLE a RUN.....	57
10.2.6. Configurar animación de estado RUN a IDLE.....	58
11. Figura Final.....	58
12. Características del equipo utilizado para el desarrollo de el viejuego.....	59
13. Conclusiones.....	60
14. Referencias.....	61

15. Anexos.....	62
-----------------	----

Índice de figuras

Figura 1 . Requisitos Mínimos Unity.....	10
Figura 2 . Requisitos Mínimos Code Learn.....	11
Figura 3 . Ventana de Unity Hub.....	11
Figura 4 . Crear proyecto nuevo 2D.....	12
Figura 5 . Creando una carpeta nueva.....	12
Figura 6 . Total de carpetas creadas.....	13
Figura 7 . Buscando Pixel Adventure 1.....	13
Figura 8 . Importar Assets.....	14
Figura 9 . Confirmar los Assets necesarios.....	14
Figura 10 . Mejor entorno de trabajo.....	14
Figura 11 . Una buena disposición.....	15
Figura 12 . Creando el primer bloque del suelo.....	15
Figura 13 . Escalado del bloque.....	16
Figura 14 . Mover el bloque de una posición a otra.....	16
Figura 15 . Duplicando un bloque.....	17
Figura 16 . Cambio de modelo del bloque duplicado.....	17
Figura 17 . Acoplar bloque duplicado.....	18
Figura 18 . Creando un objeto vacío.....	18
Figura 19 . Configurar objeto de juego vacío.....	19
Figura 20 . Agrupar objetos.....	19
Figura 21 . Agregando un box collider.....	20
Figura 22 . Cubrir el Suelo1 con el box collider.....	20
Figura 23 . Creando jugador Mask Dude.....	21
Figura 24 . Guardando la animación del jugador.....	21
Figura 25 . Agregando componentes al objeto Player.....	22
Figura 26 . Jugador que colisiona con el Suelo1.....	22

Figura 27 . Creando un canvas y un text mesh pro.....	23
Figura 28 . Habilitar el renderizado del canvas a las dimensiones de la cámara.....	23
Figura 29 . Vincular cámara principal con el canvas.....	24
Figura 30 . Agregando un objeto de tipo imagen.....	24
Figura 31 . Agregando un sprite al objeto imagen en el canvas.....	25
Figura 32 . Importar dos imágenes a la carpeta Images.....	25
Figura 33 . Agregando frutas, canasta, vida, reloj, textos y un fondo.....	26
Figura 34 . Creando estructura de la barra de vida para el jugador.....	26
Figura 35 . Pasos para agregar un sprite render.....	27
Figura 36 . Pasos para crear un sprite cuadrado.....	27
Figura 37 . Pasos para agregar un sprite a un objeto con sprite renderer	28
Figura 38 . Pasos para cambiar la cantidad de pixel por unidad.....	28
Figura 39 . Pasos para cambiar la posición en x.....	29
Figura 40 . Pasos para cambiar la posición en -x.....	29
Figura 41 . Diseño final de la barra de vida del jugador.....	29
Figura 42 . Pasos para crear un tag y asignarlo a un objeto.....	30
Figura 43 . Pasos para agregar la zona de muerte.....	30
Figura 44 . Pasos para agregar una fruta.....	31
Figura 45 . Pasos para agregar un collider y escalar el objeto.....	31
Figura 46 . Pasos para abrir el administrador de paquetes.....	32
Figura 47 . Pasos para instalar cinemachine.....	32
Figura 48 . Pasos para agregar 2D camera desde virtual machine.....	33
Figura 49 . Pasos para agregar el jugador a la cámara de cinemachine...	33
Figura 50 . Definir área de seguimiento al player.....	34
Figura 51 . Agregando un enemigo.....	34
Figura 52 . Crear un objeto detector.....	35

Figura 53 . Pasos para crea una escena y renombrar escenas.....	35
Figura 54 . Configurar y diseñar escena gameover.....	36
Figura 55 . Pasos para crear un script.....	36
Figura 56 . Pasos para agregar el script al objeto player, animator y rigidbody.....	38
Figura 57 . Modelo de la estructura de los scripts para el proyecto.....	38
Figura 58 . Pasos para agregar el script al botón.....	40
Figura 59 . Pasos para agregar el script al padre del texto y el texto agregar al script.....	42
Figura 60 . Pasos para agregar el script al enemigo.....	44
Figura 61 . Pasos para agregar el script al text mesh pro.....	45
Figura 62 . Pasos para vincular el script con el objeto Heart.....	46
Figura 63 . Agregar el script playerhealth.....	48
Figura 64 . Agregar script a las frutas, darles un tag a las frutas por su tipo, naranja o fresa.....	49
Figura 65 . Vincular el script principal con el objeto principal.....	54
Figura 66 . Pasos para crear animación de explosión al tocar la fruta.....	55
Figura 67 . Pasos para crear una animación en el mismo objeto.....	55
Figura 68 . Agregando los sprites a la nueva animación en el mismo objeto.....	56
Figura 69 . Configurar los segundos de animación.....	56
Figura 70 . Pasos para vincular dos animaciones.....	57
Figura 71 . Pasos para configurar la animación de estado detenido a estado corriendo.....	57
Figura 72 . Pasos para configurar animación de estado corriendo a estado detenido.....	58
Figura 73 . Figura final con los componentes en marcha y trabajando.....	58

1. Herramientas a utilizar

1.1. Motor de desarrollo

El motor de desarrollo que a utilizar tiene por nombre “Unity”, instalado mediante “Unity Hub”, que es el administrador de proyectos y versiones para “Unity”.

Versión utilizada **2019.4.0f1**.

1.2. Editor de código o Entorno de desarrollo

Es necesario para realizar cambios o configuraciones mediante código “C#”. Los que se sugieren “Visual Studio Code”, “VSCodium”, “MonoDevelop” y si se cuenta con un equipo “Hardware” potente “Visual Studio”.

1.3. Requisitos mínimos de “Hardware”

Para un correcto uso del software de desarrollo, se recomienda las siguientes configuraciones.

- ✓ Recomendados por unity

Figura 1. Requisitos Mínimos Unity

Requisitos del sistema de Unity Editor

Esta sección enumera los requisitos mínimos para ejecutar Unity Editor. El rendimiento real y la calidad de la reproducción pueden variar según la complejidad de su proyecto.

Requerimientos mínimos	Windows	Mac OS	Linux (compatibilidad en versión preliminar)
Versión del sistema operativo	Windows 7 (SP1+) y Windows 10, solo versiones de 64 bits.	Sierra 10.12.6+	Ubuntu 16.04, Ubuntu 18.04 y CentOS 7
UPC	Arquitectura X64 con soporte de conjunto de instrucciones SSE2	Arquitectura X64 con soporte de conjunto de instrucciones SSE2	Arquitectura X64 con soporte de conjunto de instrucciones SSE2
API de gráficos	GPU compatibles con DX10, DX11 y DX12	GPU Intel y AMD con capacidad para metales	GPU Nvidia y AMD compatibles con OpenGL 3.2+ o Vulkan.
Requerimientos adicionales	Controladores oficialmente admitidos por el proveedor de hardware Controladores compatibles con Apple Para todos los sistemas operativos, Unity Editor es compatible con estaciones de trabajo o factores de forma de portátils, y se ejecuta sin emulación, contenedor o capa de compatibilidad.		

- ✓ Otras fuentes

Figura 2. Requisitos Mínimos Code Learn

Debido a que en este campus se hará uso del software Unity, existen unos requisitos mínimos para los equipos de los alumnos asistentes:

- 15 Gb de espacio libre en el disco.
- 4 Gb de memoria RAM.
- Windows 7 o superior o MacOS X 10.8 o superior.
- Tarjeta gráfica con DX9 o DX11.

Requisitos recomendados:

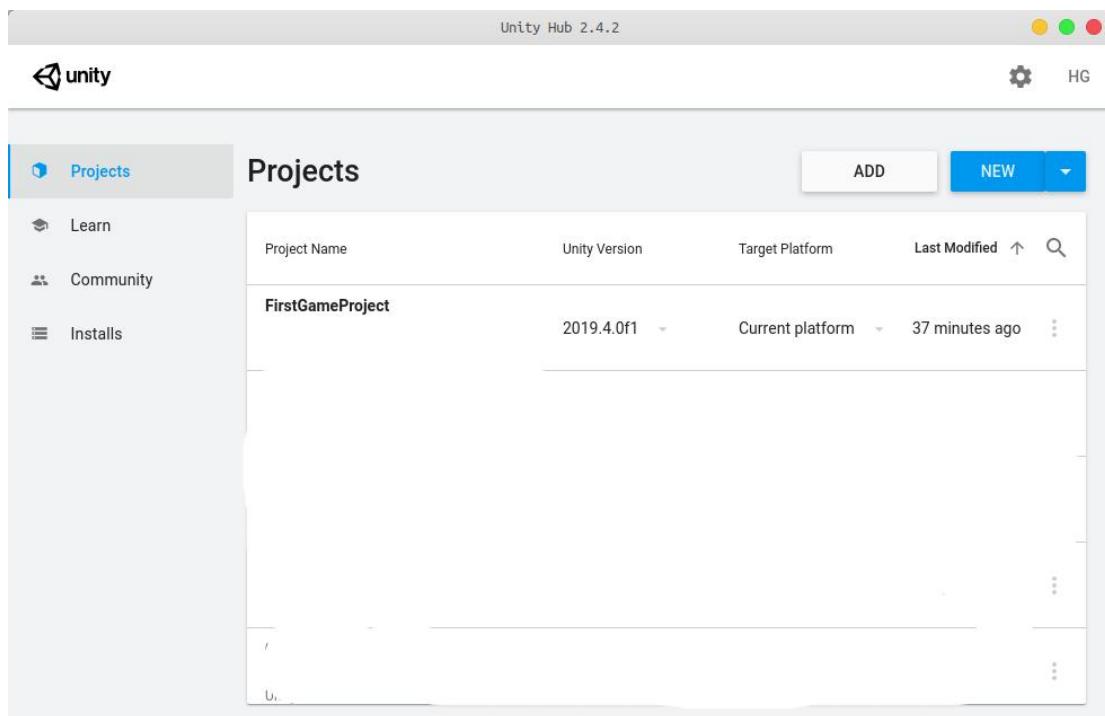
- 15 GB de espacio libre en el disco.
- 8 GB de memoria RAM.
- Windows 7 o superior o MacOSX 10.8 o superior.
- Tarjeta gráfica con DX9 o DX11.
- Tarjeta gráfica INTEL GRAPHIC 4000 o superior, o Nvidia o ATI con un 1G VRAM dedicada o superior.

2. Crear Proyecto

La ventana de “Unity Hub”, nos muestra una gran variedad de opciones.

2.1. Iniciar Unity Hub

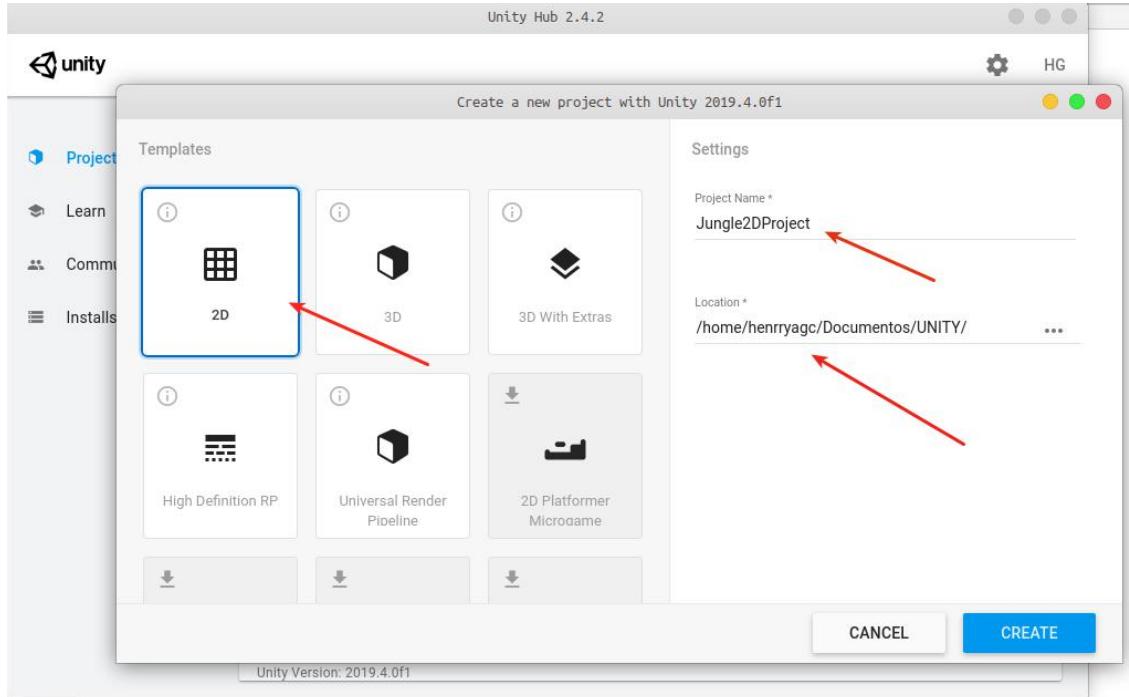
Figura 3. Ventana de Unity Hub



2.2. Crear proyecto nuevo

Digitamos el nombre del proyecto, la ubicación y en modo 2D.

Figura 4. Crear proyecto nuevo 2D

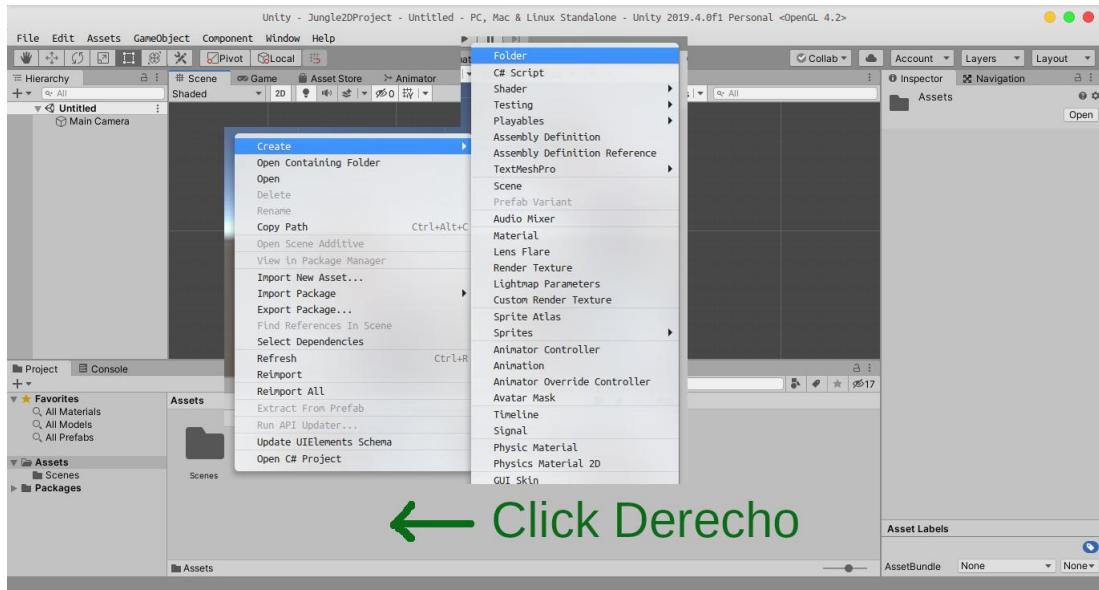


3. Creación de Carpetas

Se crean carpetas para scripts, imágenes, sonido y animación.

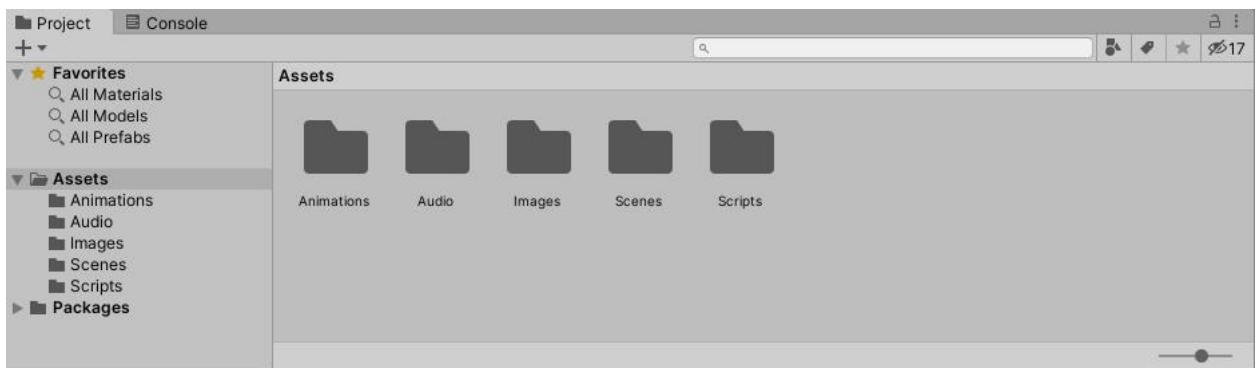
3.1. Creando una carpeta

Figura 5. Creando una carpeta nueva



3.2. Carpetas creadas

Figura 6. Total de carpetas creadas



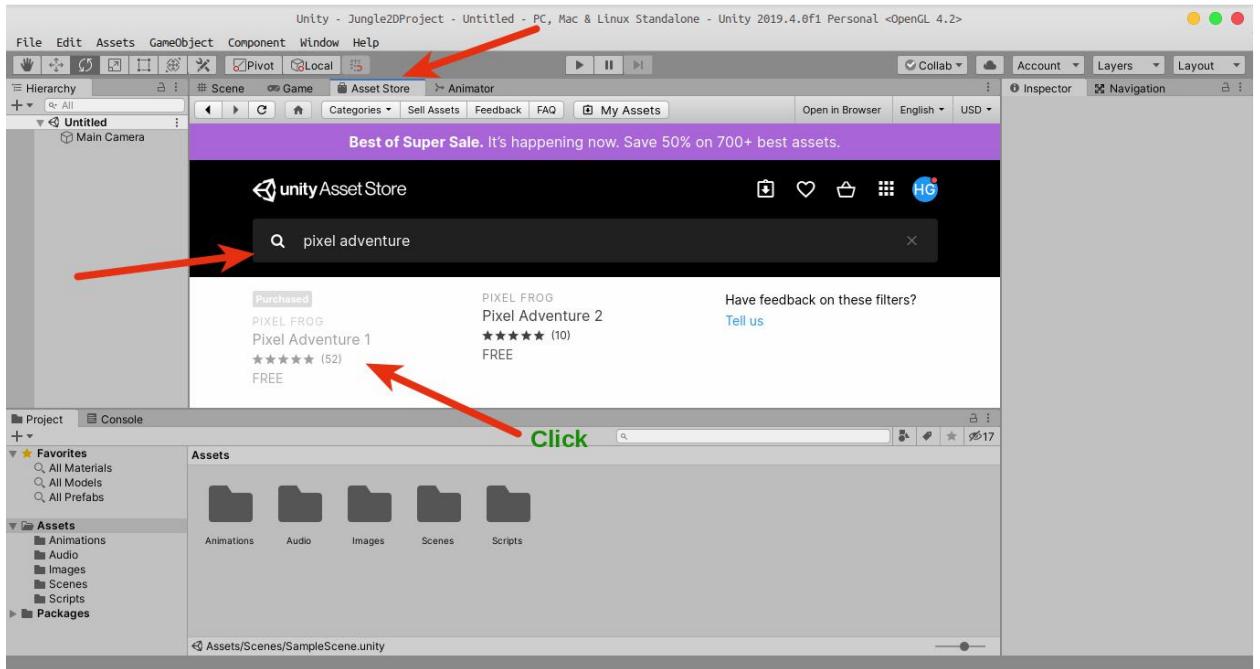
4. Importando Assets

La Asset Store de Unity cuenta con varios paquetes con imágenes, audio, modelos 2D, modelos 3D y otros complementos más.

4.1. Buscando un Assets gratuito

El paquete de Assets gratuito encontrado se llama “Pixel adventure 1”, pero existen otras alternativas, modelos diferentes, para el proyecto se tomará como ejemplo el paquete de Assets ya mencionado.

Figura 7. Buscando Pixel Adventure 1



4.2. Descargando e importando “Pixel Adventure 1”

Lo que sigue es descargar e importar el paquete de Assets de “Pixel Adventure 1”.

Figura 8. Importar Assets

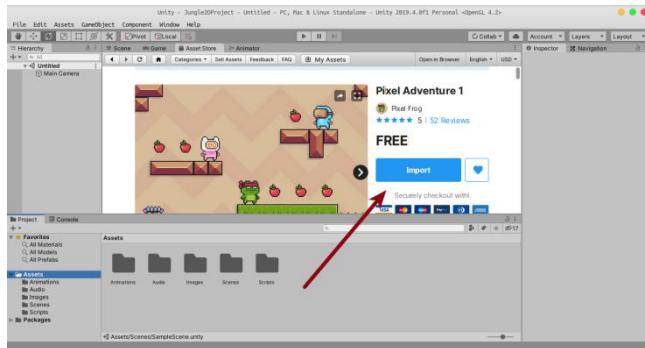
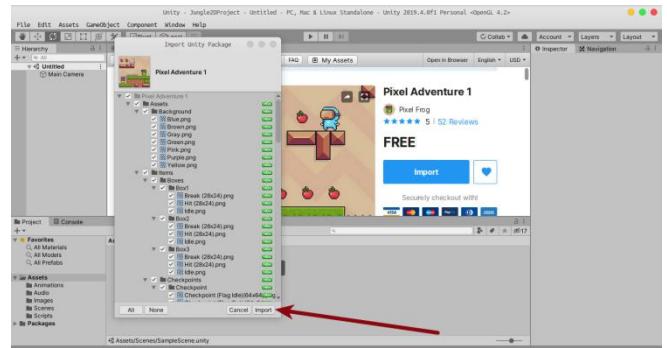


Figura 9. Confirmar los Assets necesarios



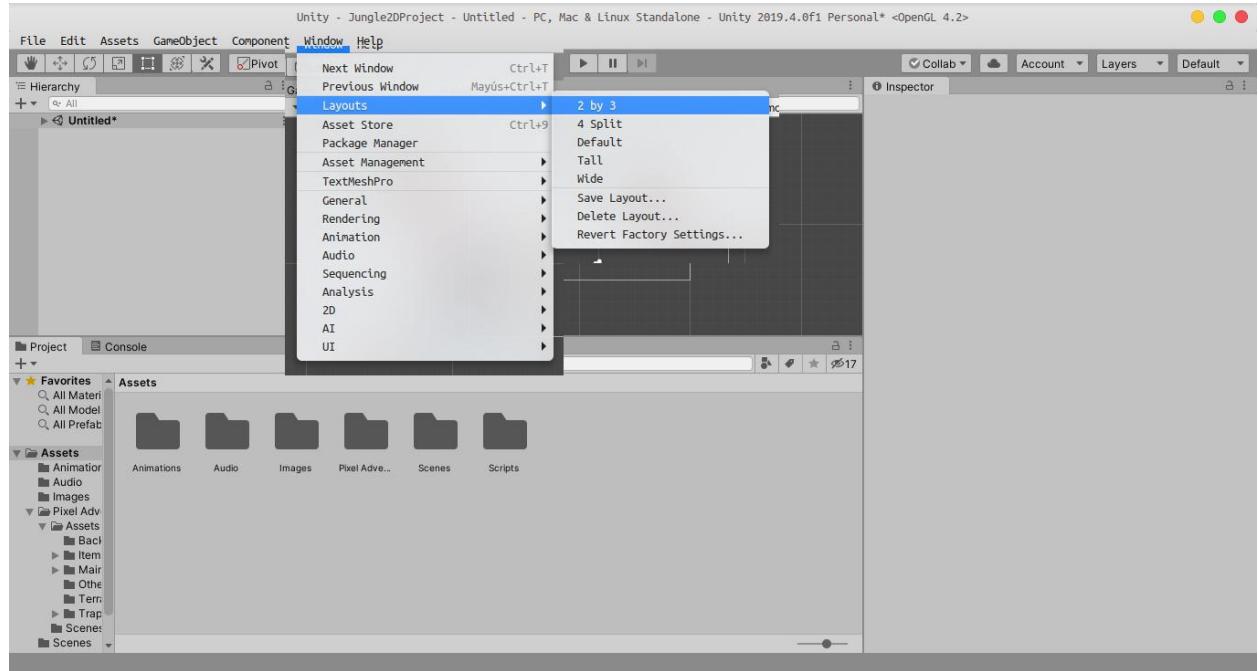
5. Configuración de las ventanas de trabajo “Layouts”

Para tener un campo de visión más general del proyecto, se obtará por una configuración mucho más adecuada.

5.1. Vista de trabajo 2 by 3

En el menú de opciones se encuentra la opción de ventana “Window”, que tiene un apartado de disposición “Layout”.

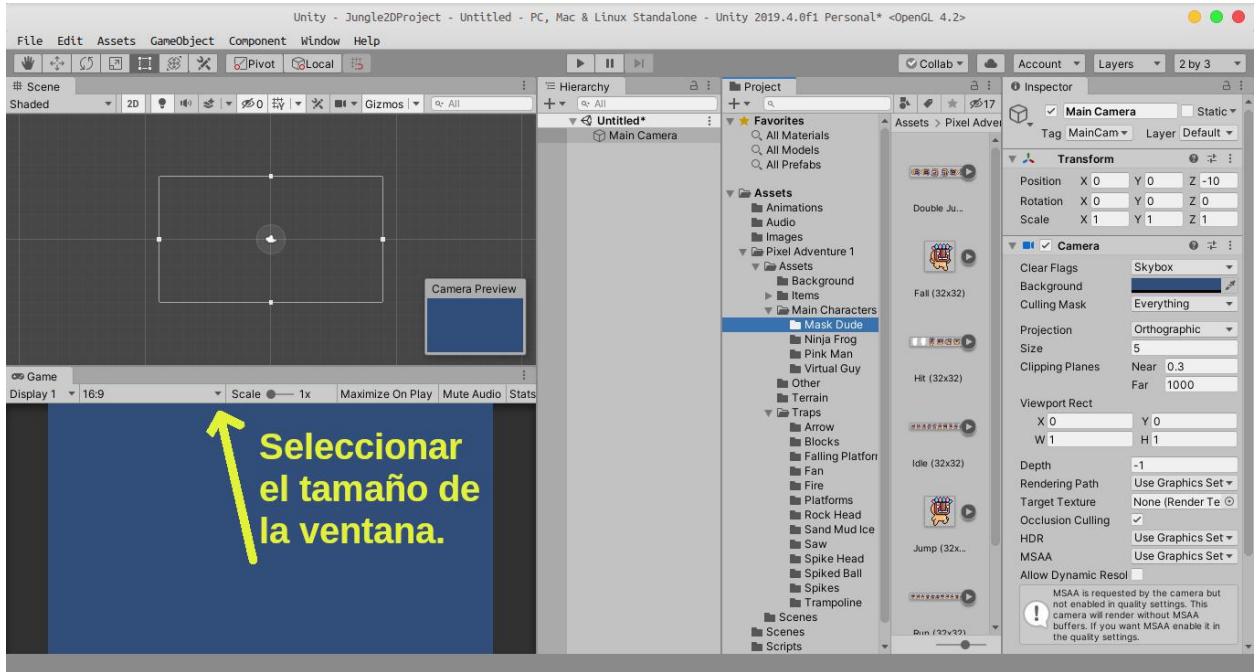
Figura 10. Mejor entorno de trabajo



5.2. Entorno de trabajo listo

Un entorno de trabajo bien estructurado, permite una mejor productividad y visión amplia del proyecto.

Figura 11. Una buena disposición



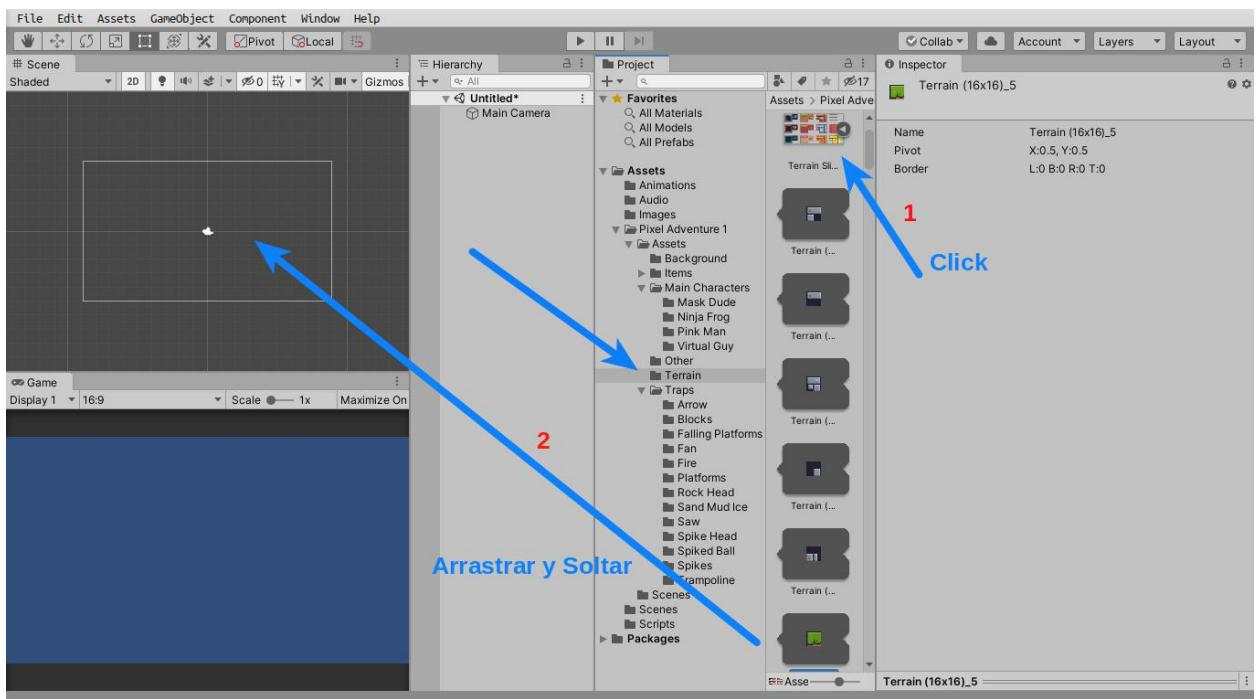
6. Creando el mundo virtual del videojuego

Para el mundo virtual se necesitaran suelos, paredes, gradas, jugador, enemigos, recompensas. La mayoria de modelos que necesitaremos lo vamos a encontrar en el paquete de “Pixel Adventure 1”.

6.1. Creando el piso

Empezamos creando el suelo, en el cual ocurriran todas las interacciones del jugador.

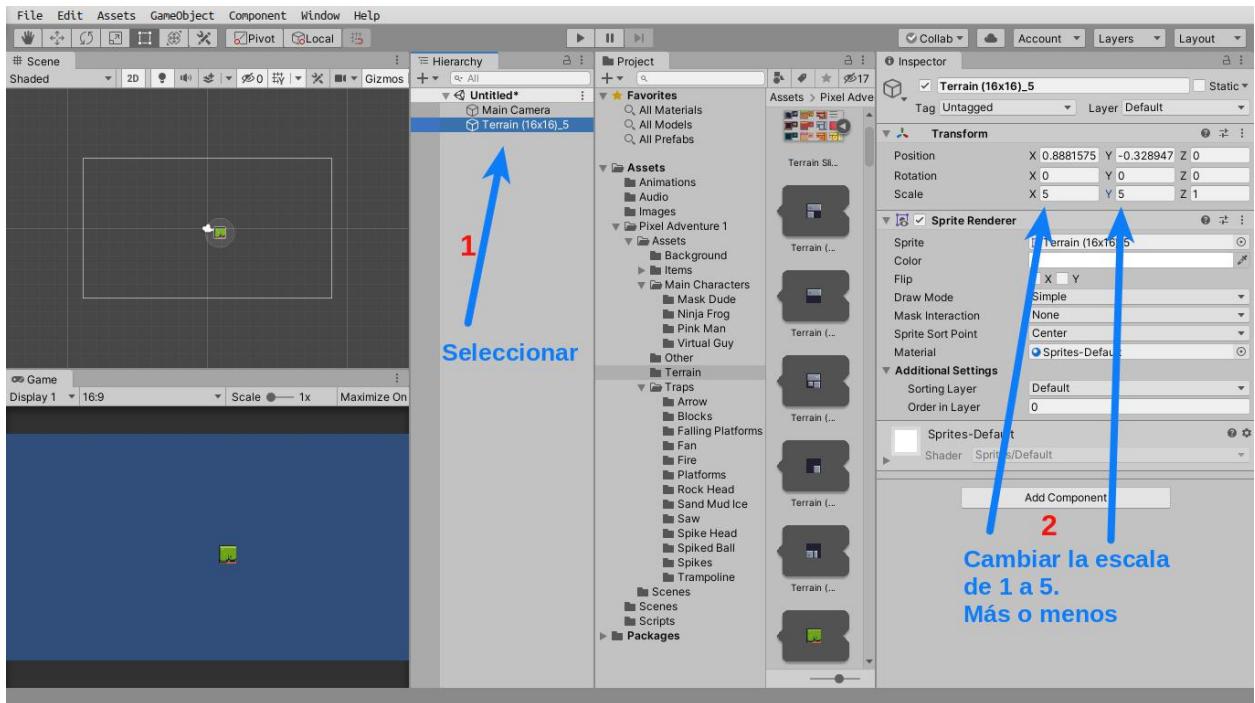
Figura 12. Creando el primer bloque del suelo



6.2. Escalar el bloque

Mediante los valores de transformación, se escalará el bloque para que se pueda visualizar de mejor manera

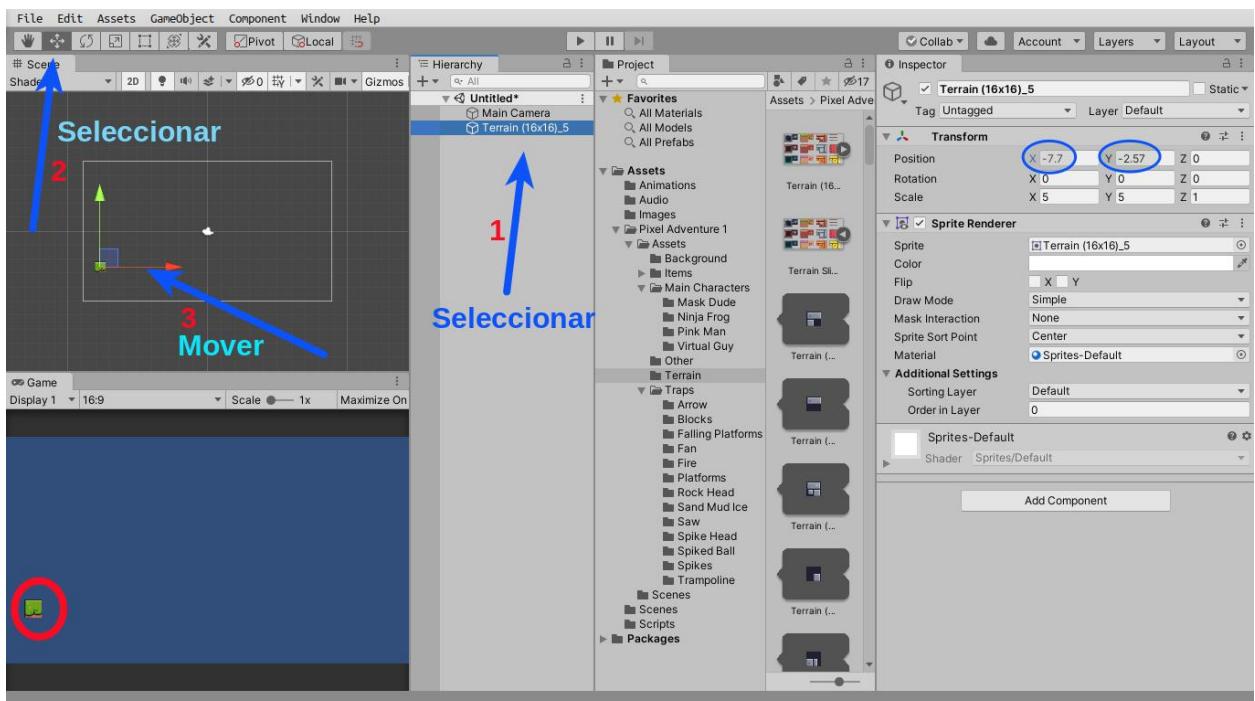
Figura 13. Escalado del bloque



6.3. Desplazar el bloque

La herramienta que nos permite desplazar objetos a distintas posiciones en el modo de vista (escena).

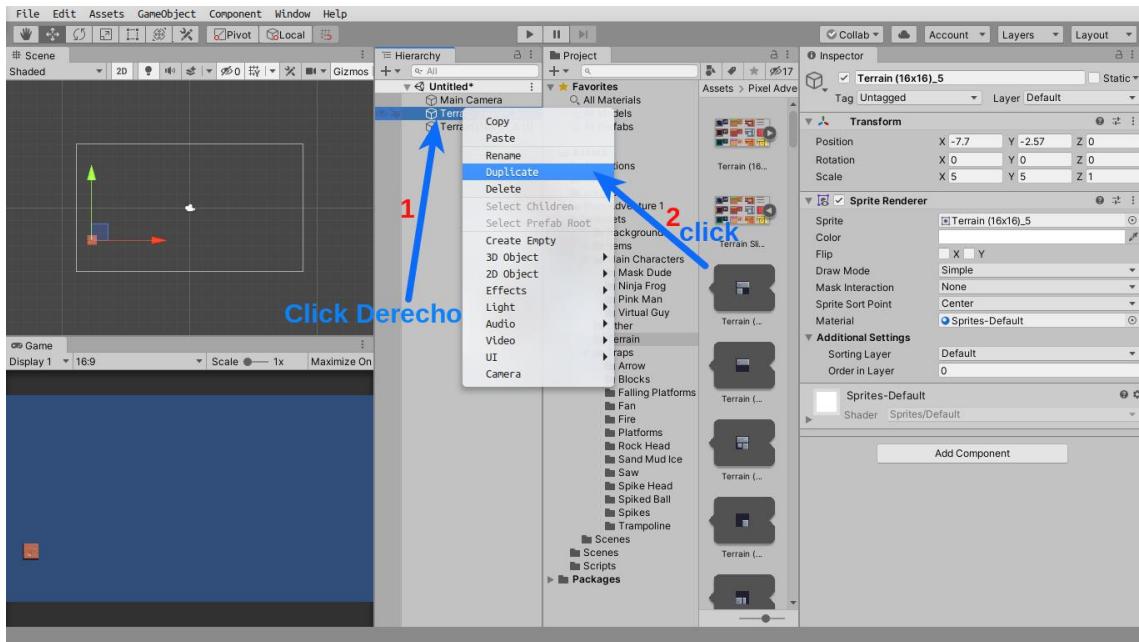
Figura 14. Mover el bloque de una posición a otra



6.4. Duplicando un bloque

Duplicar un objeto similar, nos permite de cierto modo acelerar el diseño.

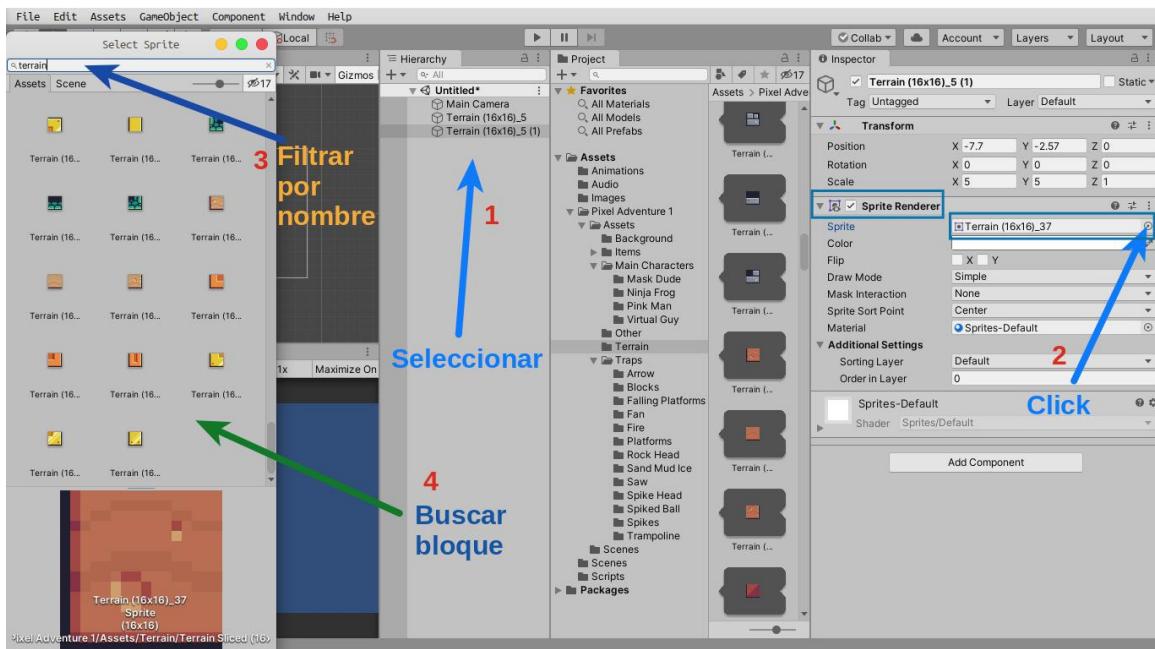
Figura 15. Duplicando un bloque



6.5. Cambiar modelo del bloque

Se cambia el modelo del bloque duplicado para que se complemente con el diseño del bloque inicial y se muestre una correlación.

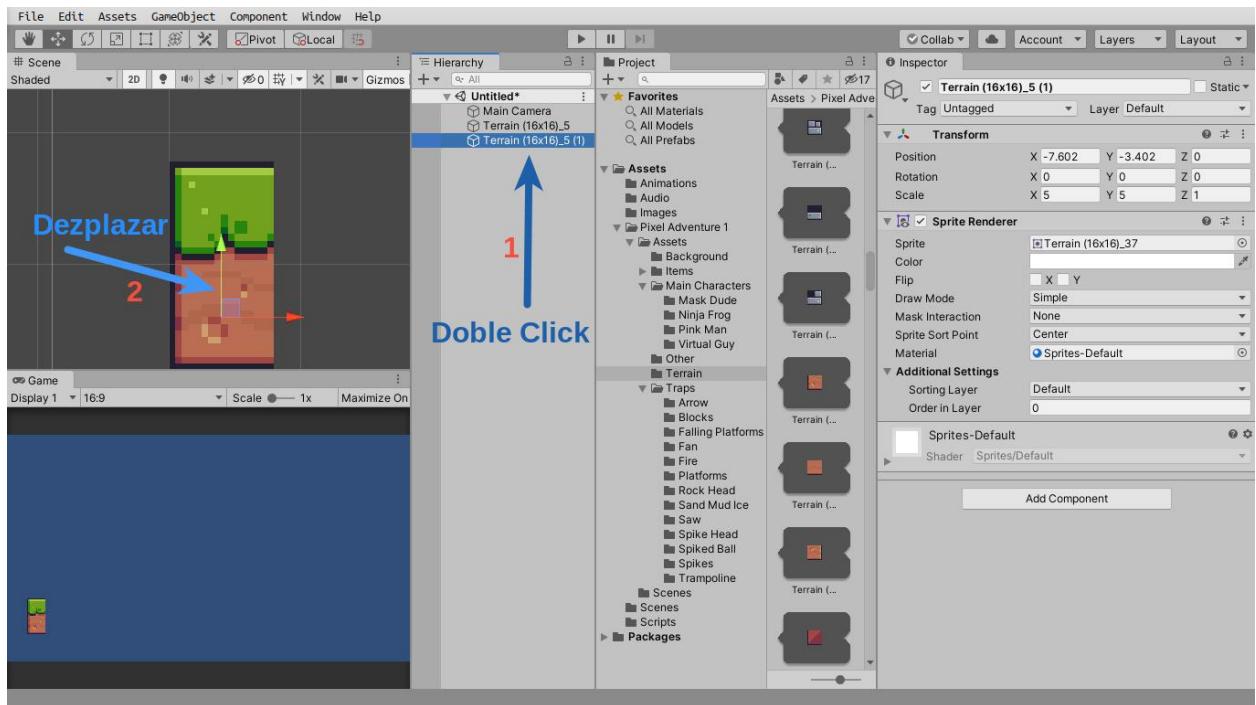
Figura 16. Cambio de modelo del bloque duplicado



6.6. Acoplamiento de bloques

Se acoplan los bloques para que el suelo se muestre con una mayor consistencia.

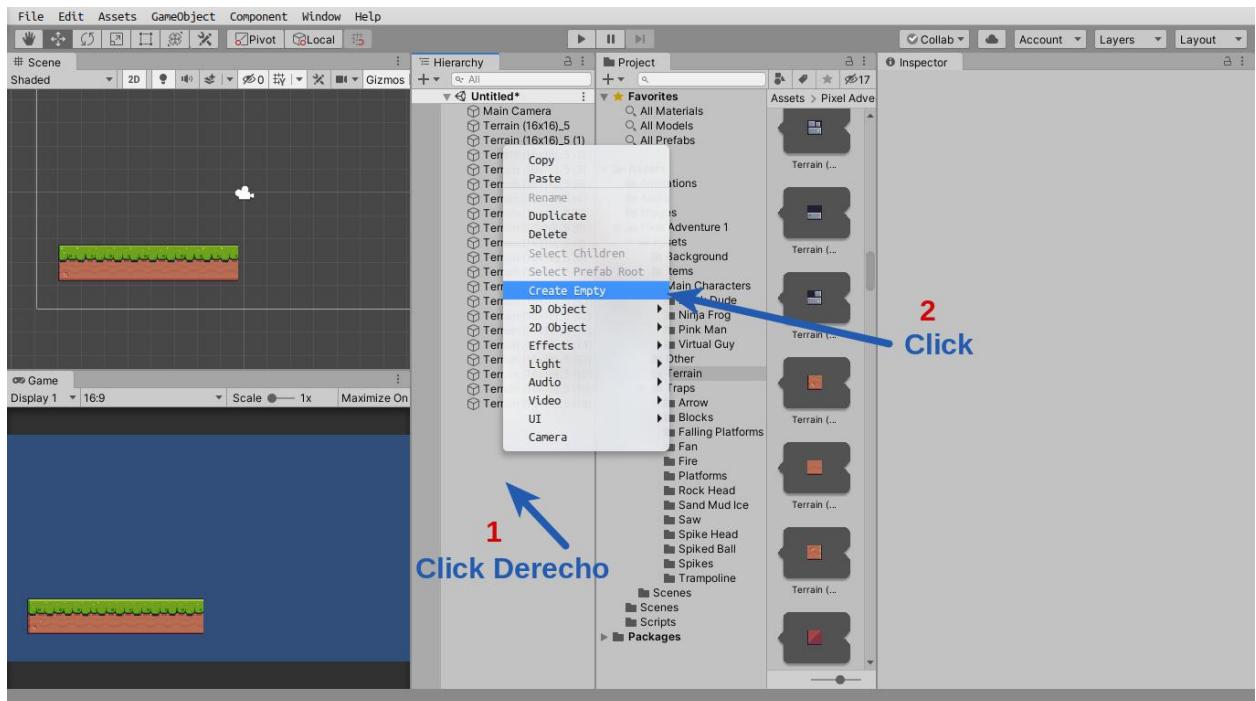
Figura 17. Acoplar bloque duplicado



6.7. Creando nuevo objeto de juego

El nuevo objeto de juego nos servirá para agrupar múltiples objetos.

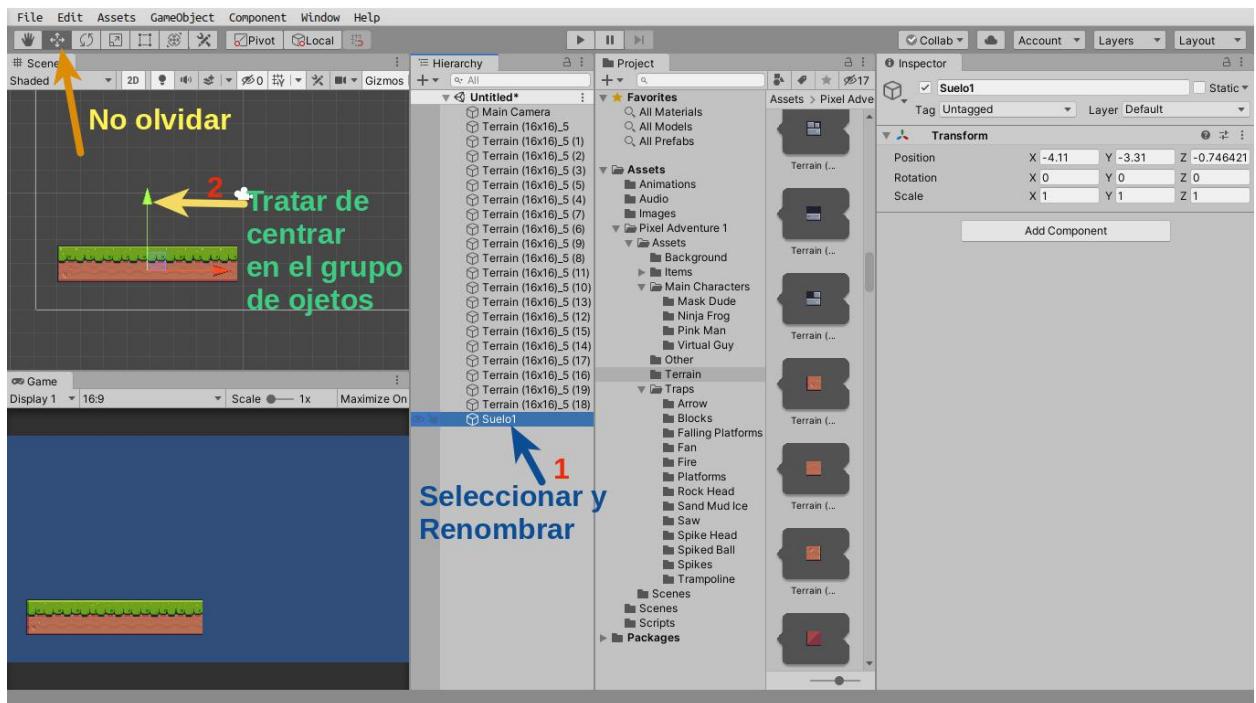
Figura 18. Creando un objeto vacío



6.8. Configurar nuevo objeto de juego

Las configuraciones son necesarias para tener el control de cada objeto.

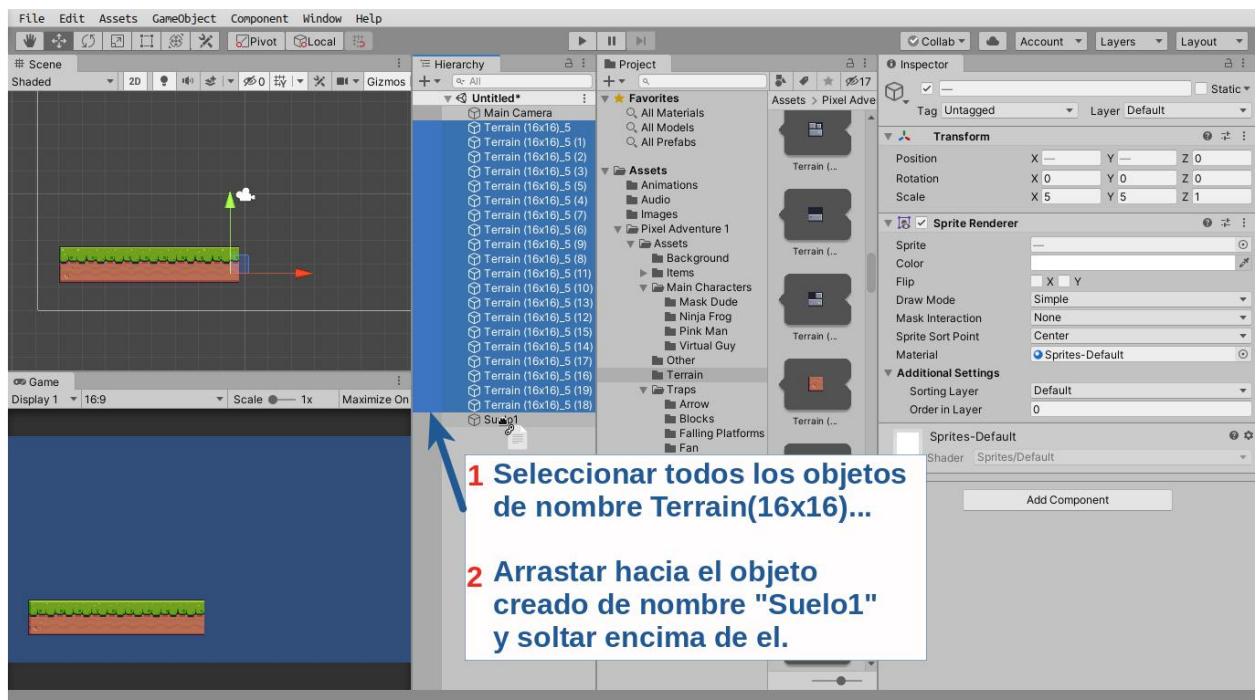
Figura 19. Configurar objeto de juego vacío



6.9. Agrupar objetos

Tener objetos agrupados nos va a permitir tener un orden y un entorno de trabajo menos saturado.

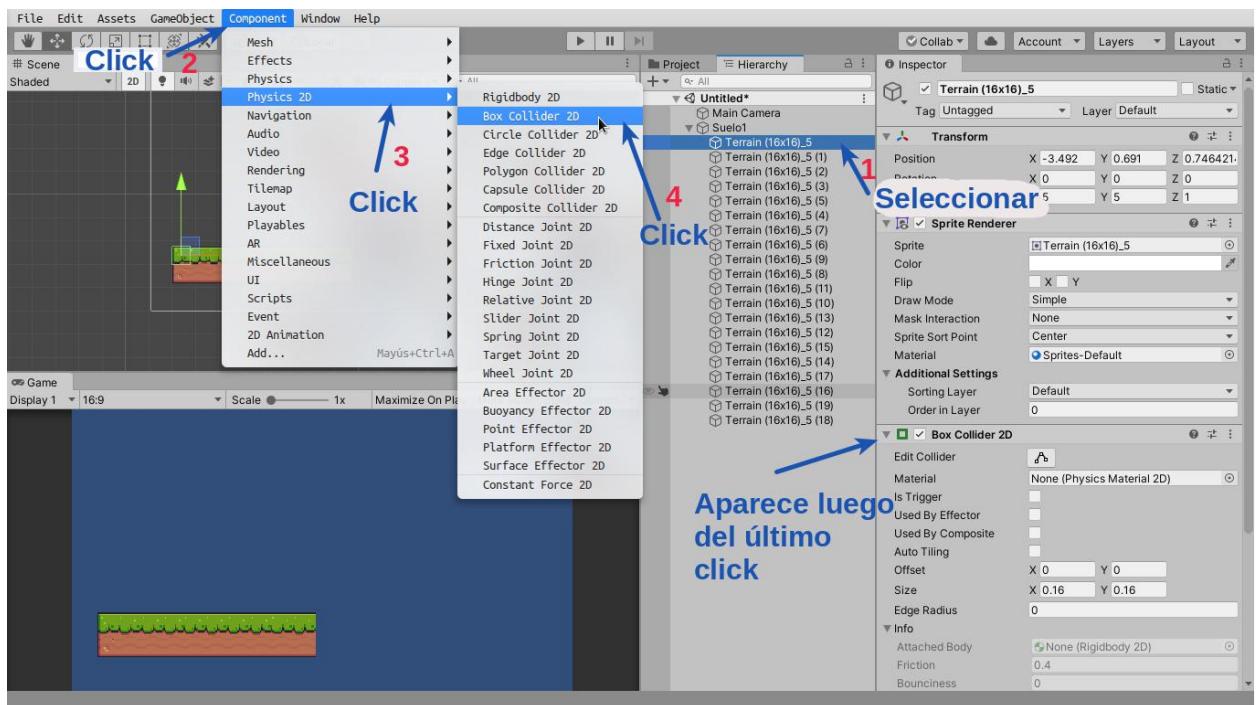
Figura 20. Agrupar objetos



6.10. Agregar Box Collider

Agregamos un componente de tipo Box Collider al primer objeto de nombre “Terrain(16x16)_5”, que se encuentra en el grupo de objetos Suelo1.

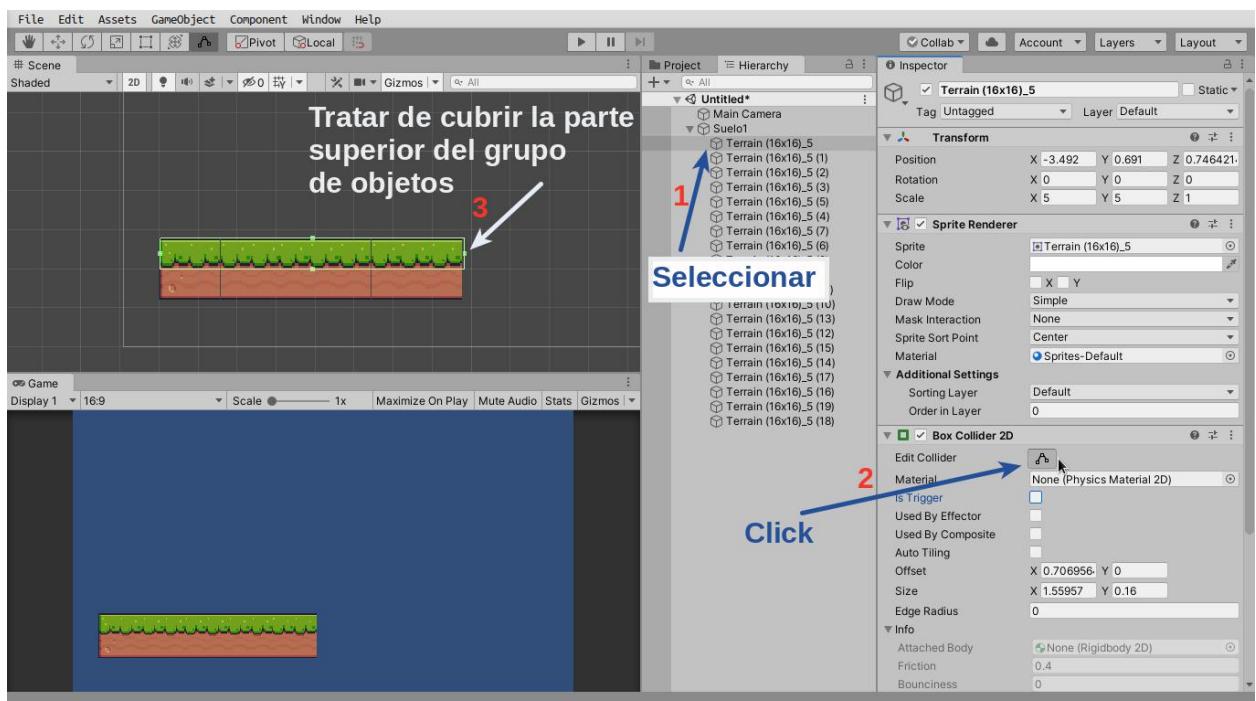
Figura 21. Agregando un box collider



6.11. Configurando el Box Collider

El Box Collider nos permite brindar capacidad de colisión.

Figura 22. Cubrir el Suelo1 con el box collider.



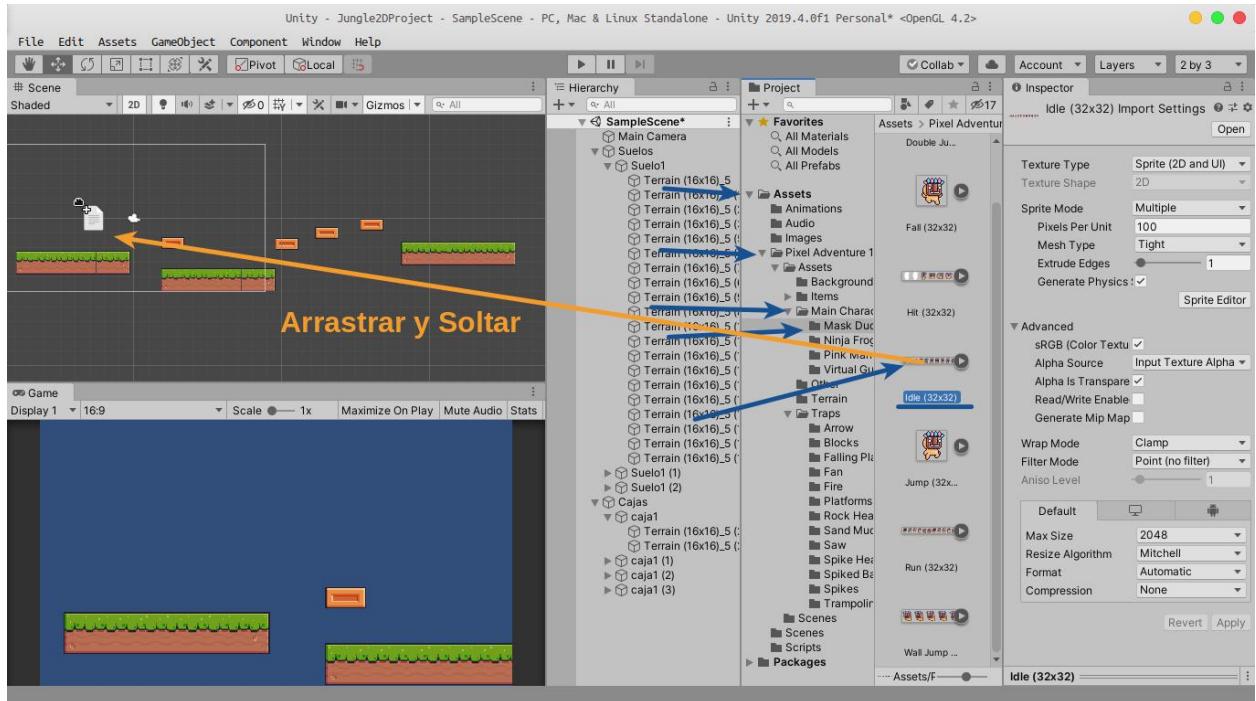
6.12. Crear un jugador

El jugador será el actor principal del juego.

6.12.1. Creando un jugador

Dentro del paquete de Pixel Adventure 1 existen cuatro modelos de jugadores, elegimos el que se deseé.

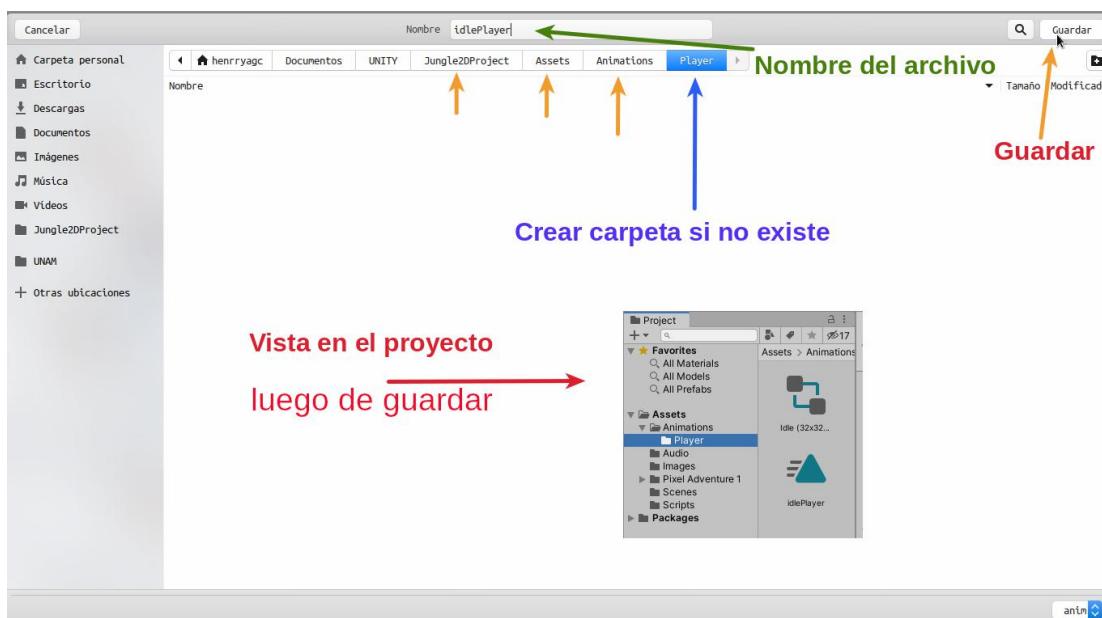
Figura 23. Creando jugador Mask Dude



6.12.2. Guardar animación del jugador

El modelo de jugador (Sprite) elegido cuenta con una animación por defecto. La animación se tiene que almacenar dentro de la carpeta “/Assets/Animations/Player/”, el nombre que se le pondrá al archivo será “idlePlayer”.

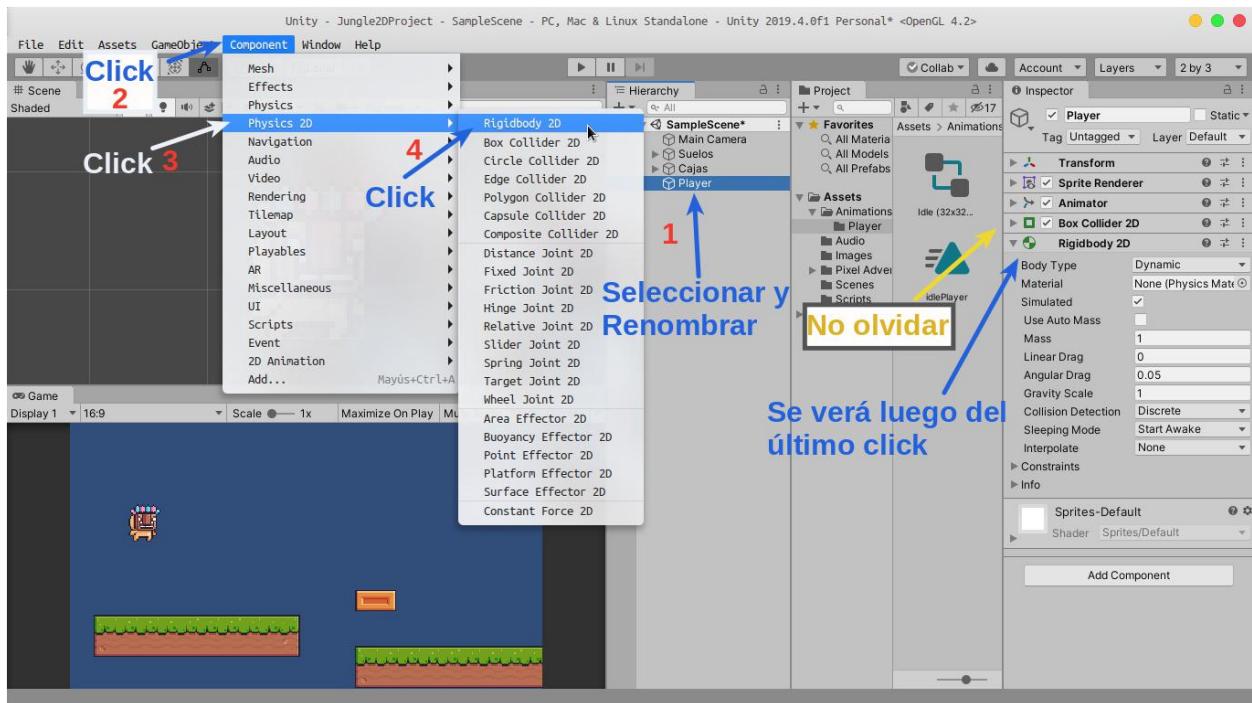
Figura 24. Guardando la animación del jugador



6.12.3. Agregar colisión y gravedad al jugador

Se agregará un Box Collider 2D y un Rigidbody 2D, esto para que el jugador pueda colisionar y también tenga gravedad.

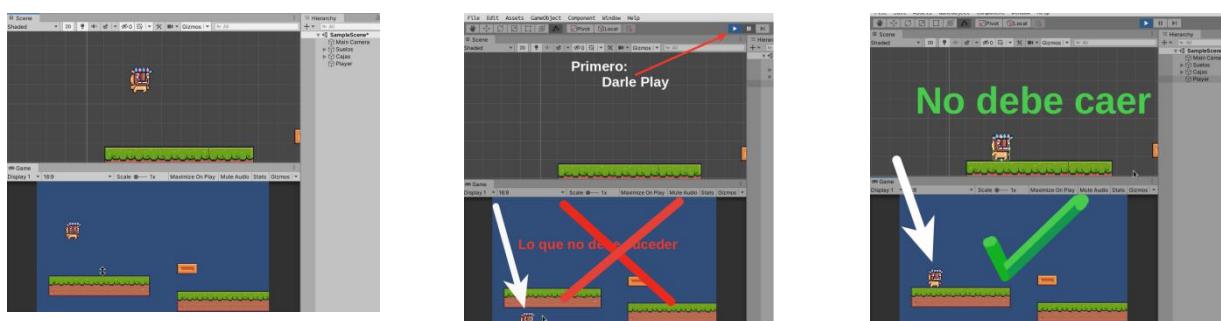
Figura 25. Agregando componentes al objeto Player



6.12.4. Prueba en modo juego

Presionar el botón “Play ►”, que se encuentra en la parte superior, en medio de la barra de herramientas. El Player debe interactuar con el suelo, además el jugador debe de mostrar una animación.

Figura 26. Jugador que colisiona con el Suelo



Antes de darle Click a Play

Player sin Box Collider

Player con Box Collider

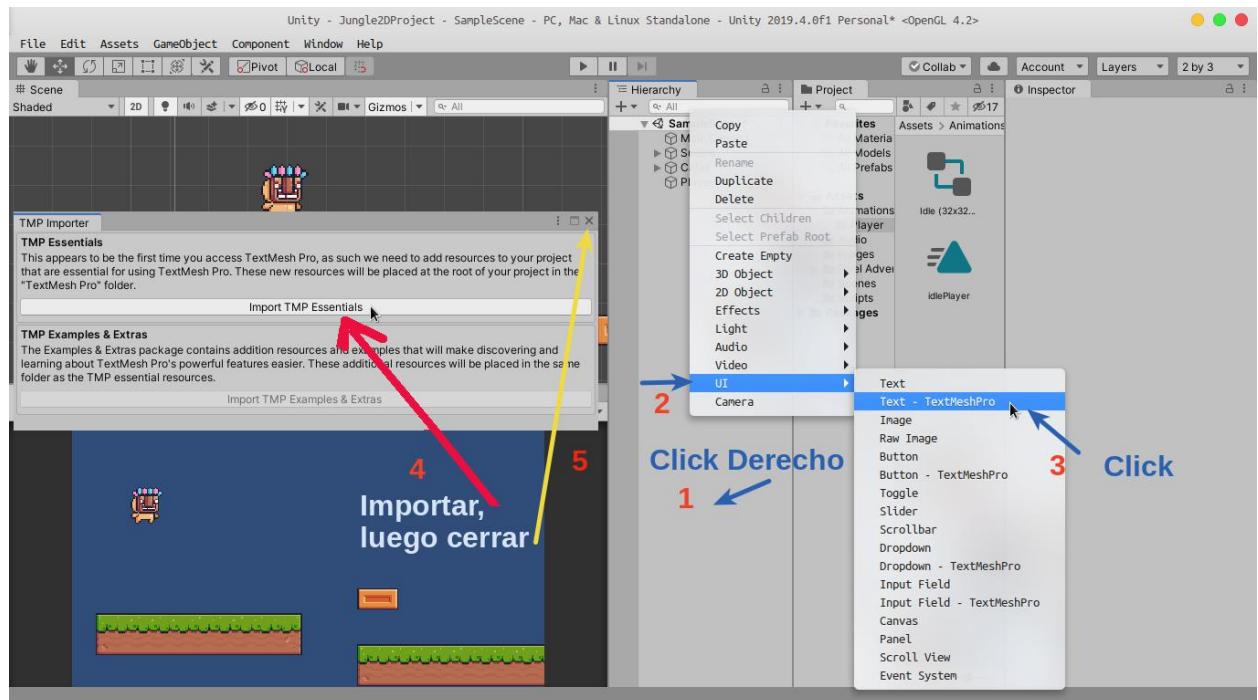
6.13. Creando el objeto Canvas

Un objeto de tipo Canvas nos va a permitir controlar texto, imagen, botón y otros.

6.13.1. Creando un Text Mesh Pro

Este objeto es de formato texto enriquecido.

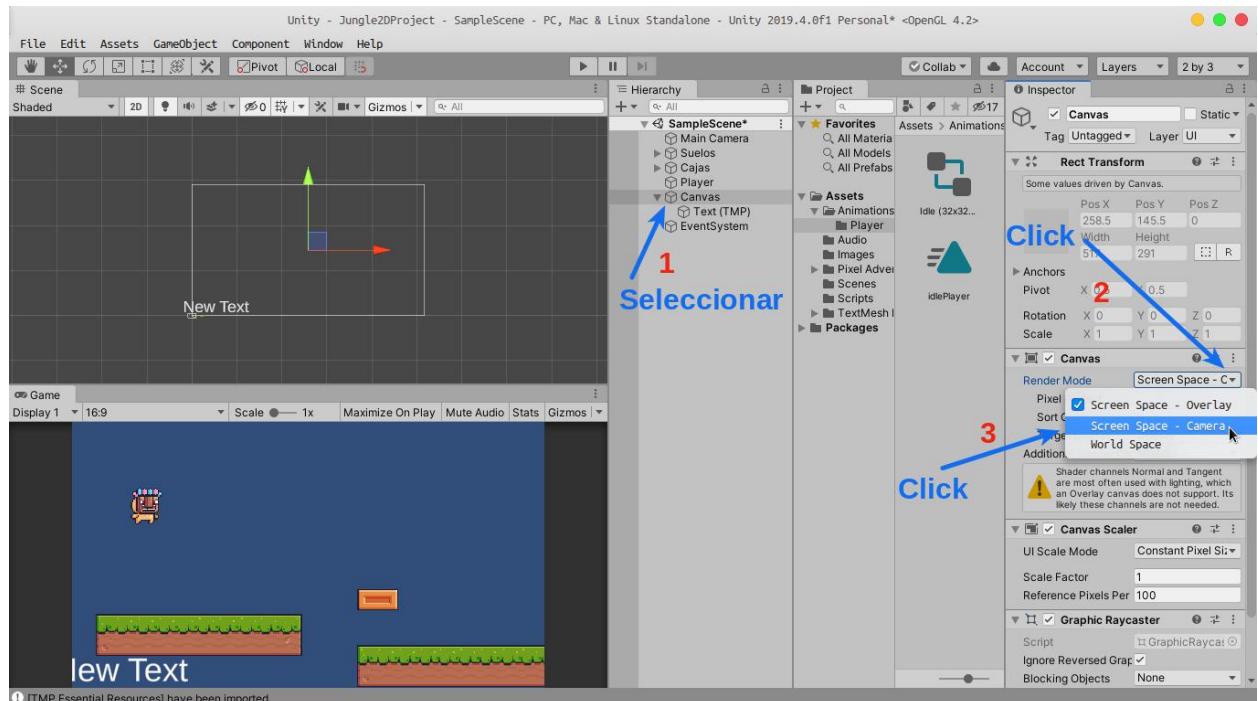
Figura 27. Creando un canvas y un text mesh pro



6.13.2. Configurando Canvas

Configurar el tamaño de Canvas al tamaño de la cámara

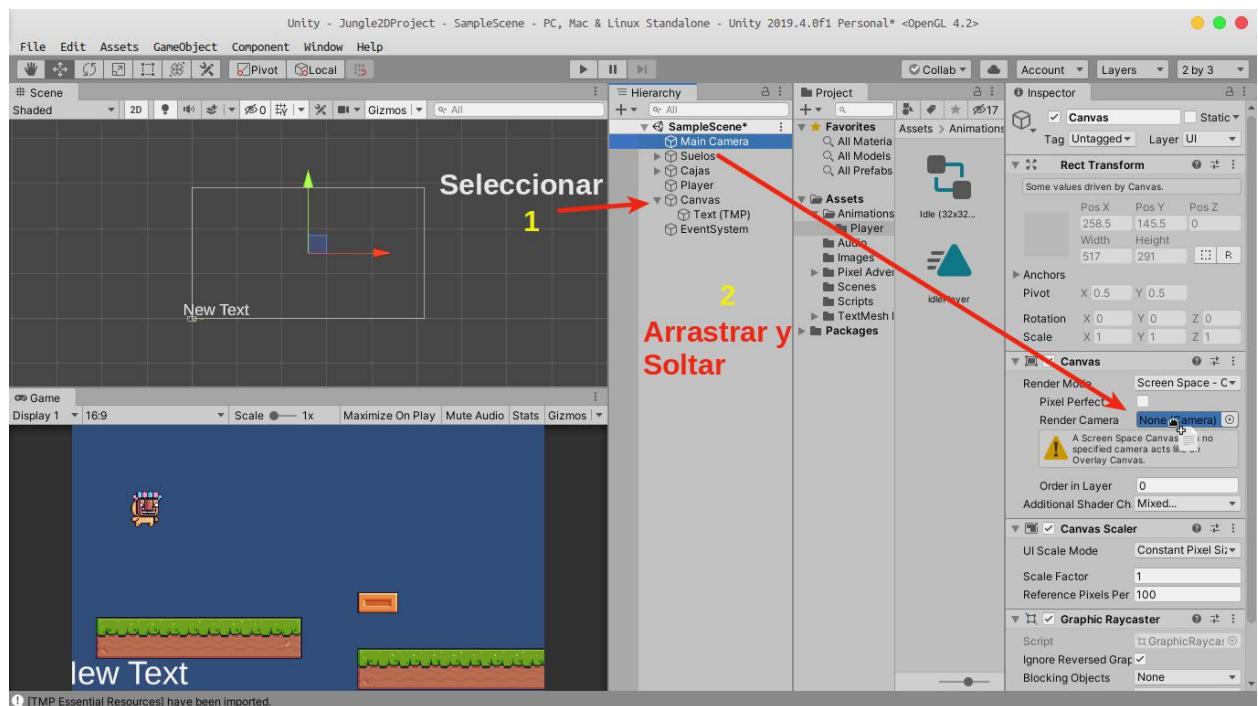
Figura 28. Habilitar el renderizado del canvas a las dimensiones de la cámara



6.13.3. Vincular Canvas con la Cámara Principal

Arrastar el objeto con nombre Main Camera hacia el apartado de Render Camera en Canvas.

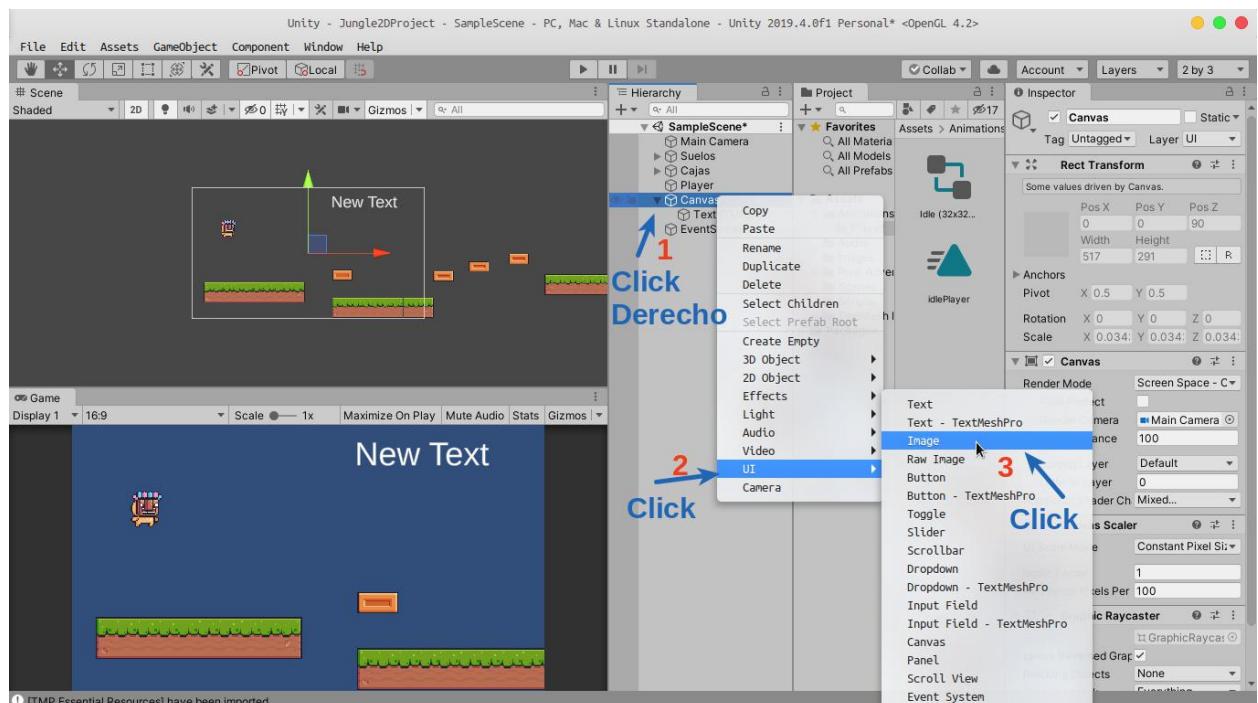
Figura 29. Vincular cámara principal con el canvas



6.13.4. Agregar un objeto al Canvas de tipo Imagen

Mediante este objeto podremos ingresar una imágenes.

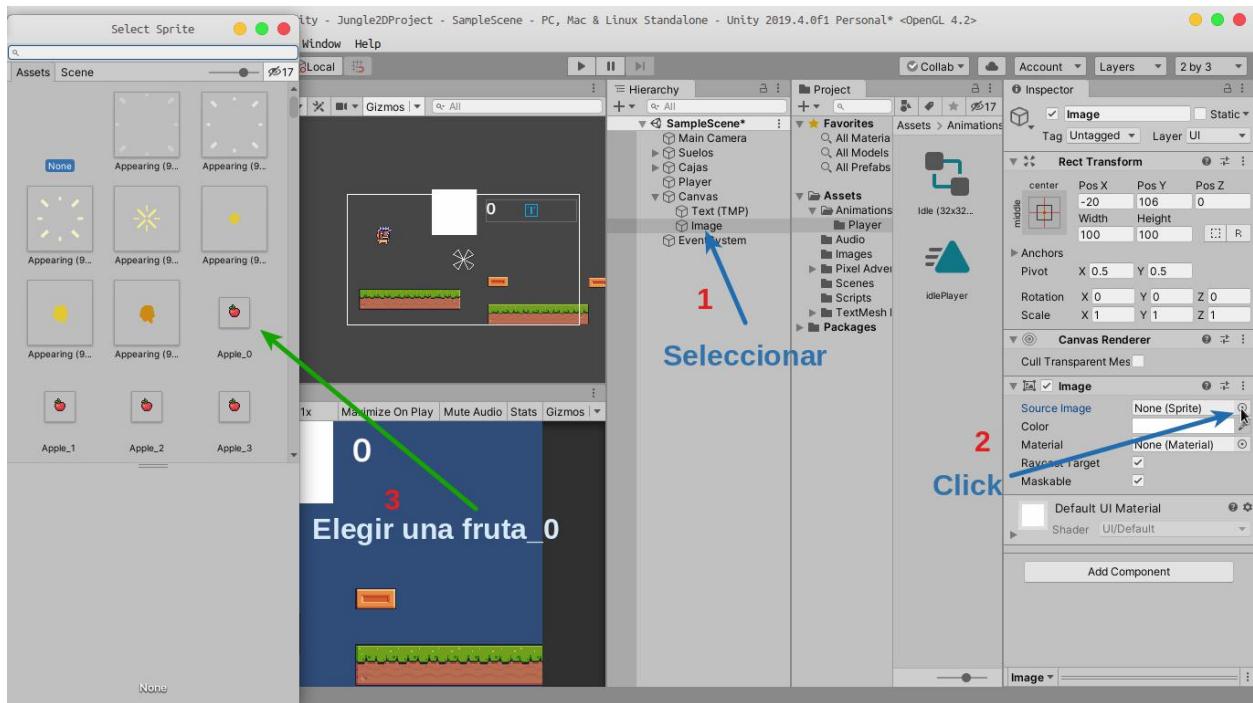
Figura 30. Agregando un objeto de tipo imagen



6.13.5. Agregar un sprite al objeto imagen

Agregamos una imagen con diseño de alguna fruta.

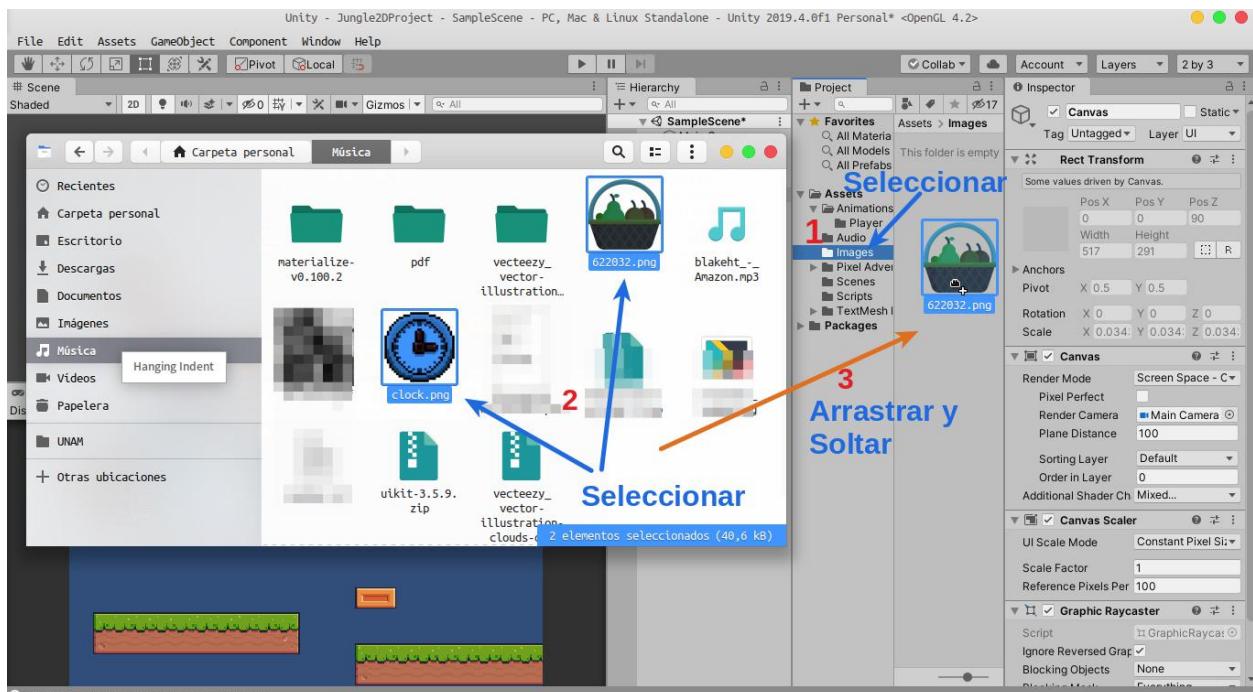
Figura 31. Agregando un sprite al objeto imagen en el canvas



6.13.6. Importar imágenes

Se van a importar algunas imágenes para agregarlas al proyecto, este modo también funciona para otro tipo de archivos (audio, scripts, etc.).

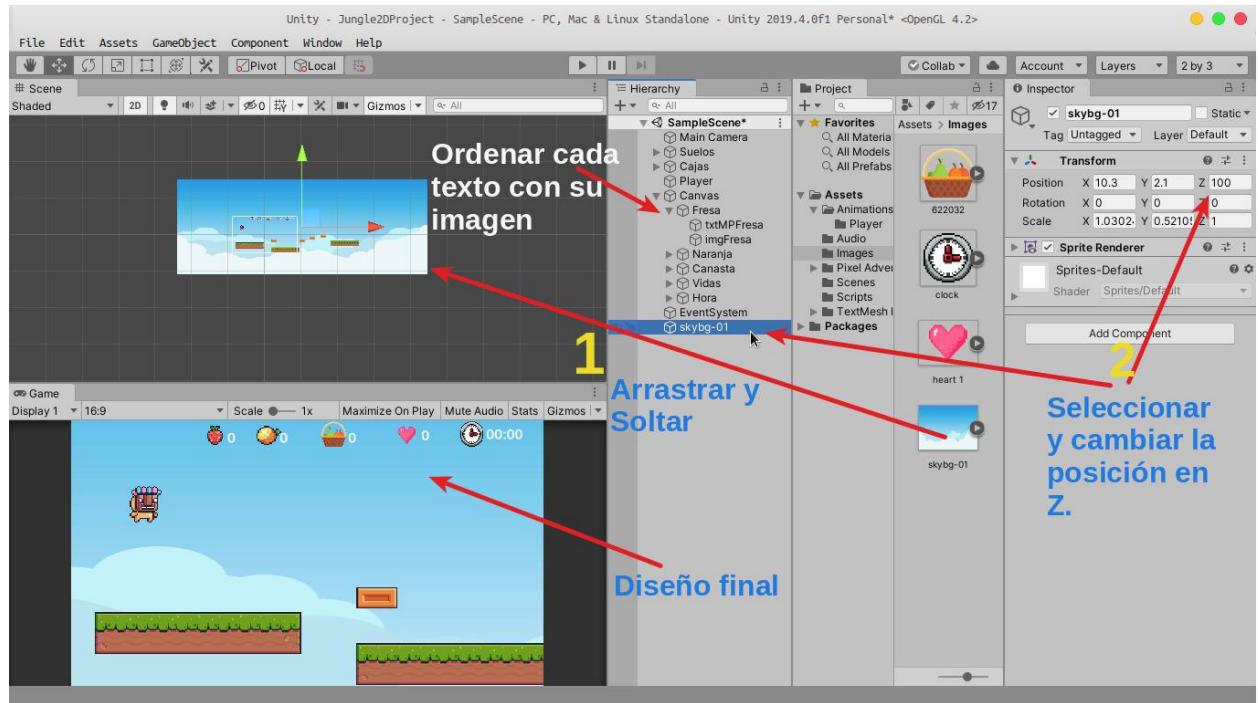
Figura 32. Importar dos imágenes a la carpeta Images



6.13.7. Agregar una imagen de fondo

Agregar un fondo hará que el entorno de juego se vea mejor.

Figura 33. Agregando frutas, canasta, vida, reloj, textos y un fondo



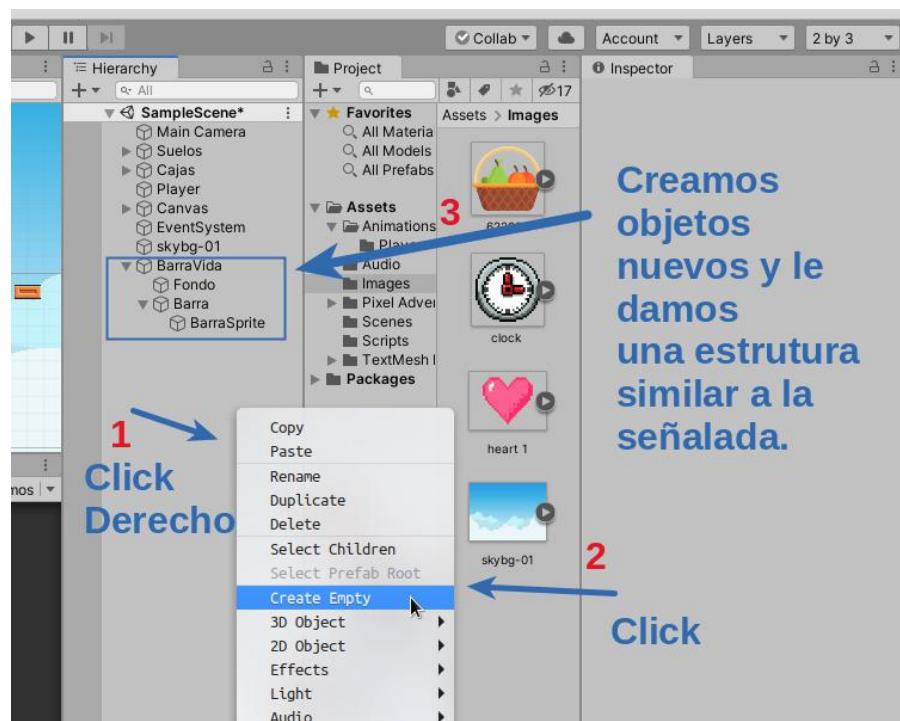
6.14. Creando la barra de vida

Crearemos una barra de vida para indicar la salud del jugador.

6.14.1. Estructura de la barra de vida.

Es una estructura básica y práctica.

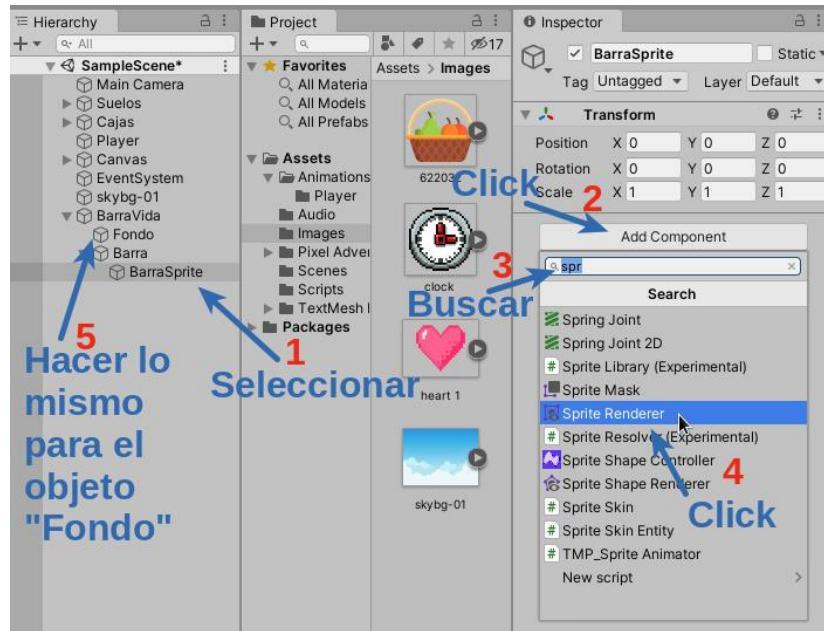
Figura 34. Creando estructura de la barra de vida para el jugador



6.14.2. Agregar un Sprite Render

Se han de agregar al objeto “BarraSprite y Fondo”.

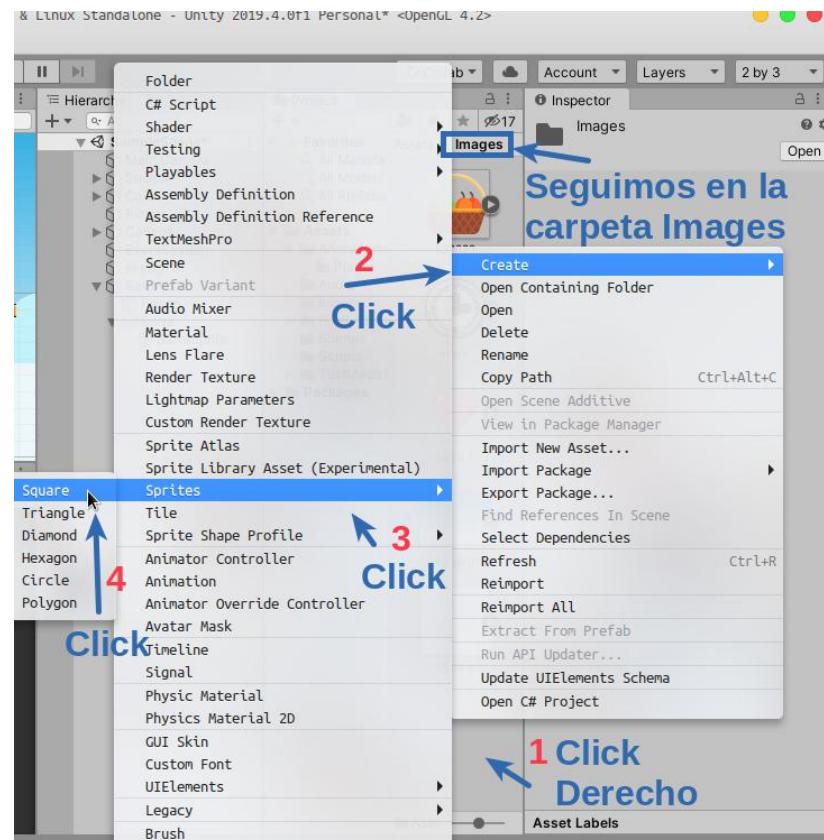
Figura 35. Pasos para agregar un sprite render



6.14.3. Creando un Sprite cuadrado

Este Sprite nos servirá para darle color a los objetos “Fondo y Barra Sprite”.

Figura 36. Pasos para crear un sprite cuadrado



6.14.4. Vincular los objetos con el sprite square

Se agrega el sprite square a los objetos “Fondo y BarraSprite”, para poder darles color y así la barra de vida va tomando forma. No olvidar que el objeto “Fondo”, debe de tener 1 de “Position” en “Z”.

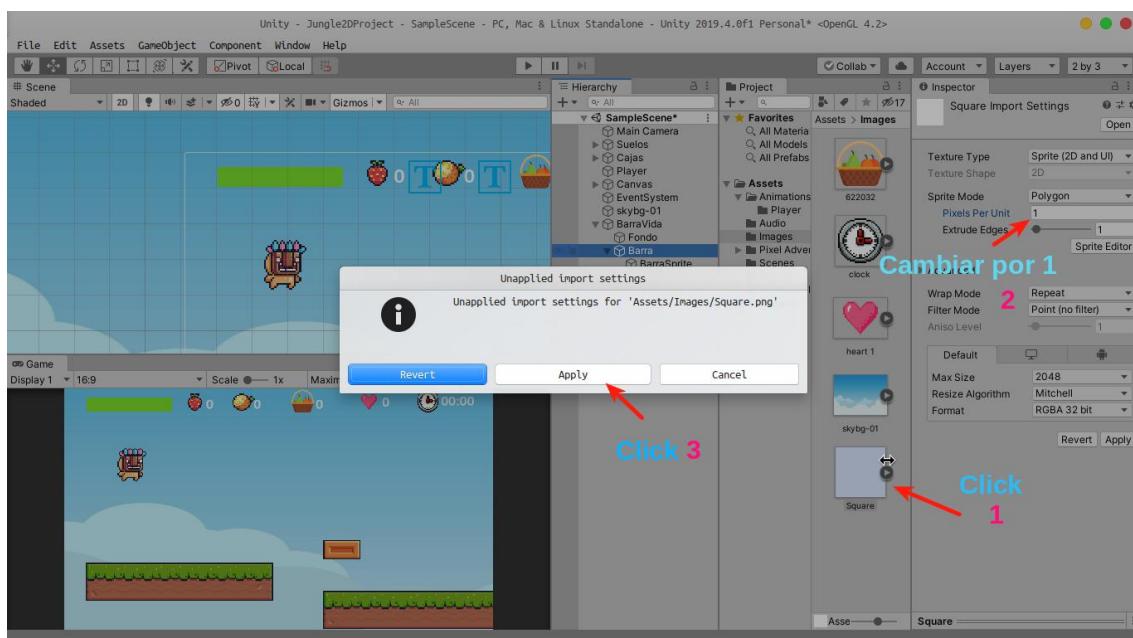
Figura 37. Pasos para agregar un sprite a un objeto con sprite renderer



6.14.5. Cambiar la cantidad de pixel por unidad

Cambiamos el tamaño de pixel por unidad del sprite creado con nombre Square.

Figura 38. Pasos para cambiar la cantidad de pixel por unidad



6.14.6. Cambiar position X del objeto “BarraSprite”

Se debe cambiar la posición para que al momento de escalar el objeto no tengamos problemas.

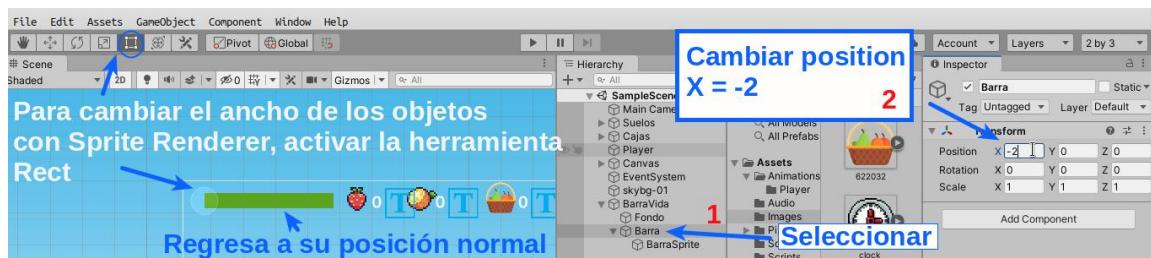
Figura 39. Pasos para cambiar la posición en x



6.14.7. Cambiar position X del objeto “Barra”

Como el objeto “Fondo” se ve descubierto, tenemos que mover la posición del objeto “Barra”, entonces, position X en el objeto “Barra” sera -2.

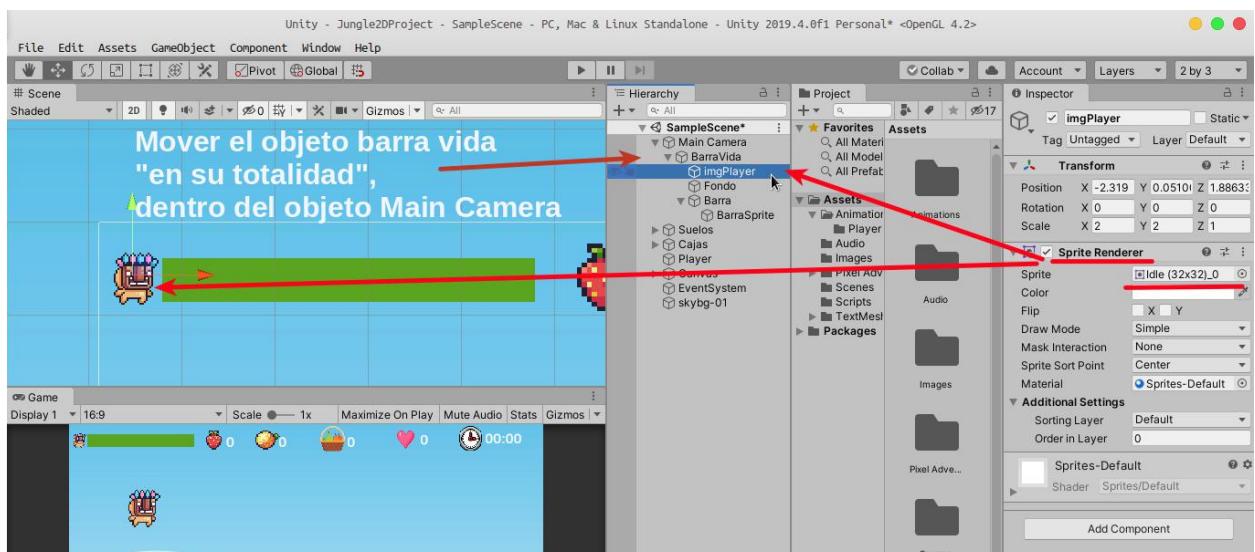
Figura 40. Pasos para cambiar la posición en x



6.14.8. Diseño final de la barra de vida y agrupación.

Mover los objetos dentro del objeto “BarraVida” hacia “Main Camera” será necesario para cuando el jugador se tenga que desplazar por los distintos suelos.

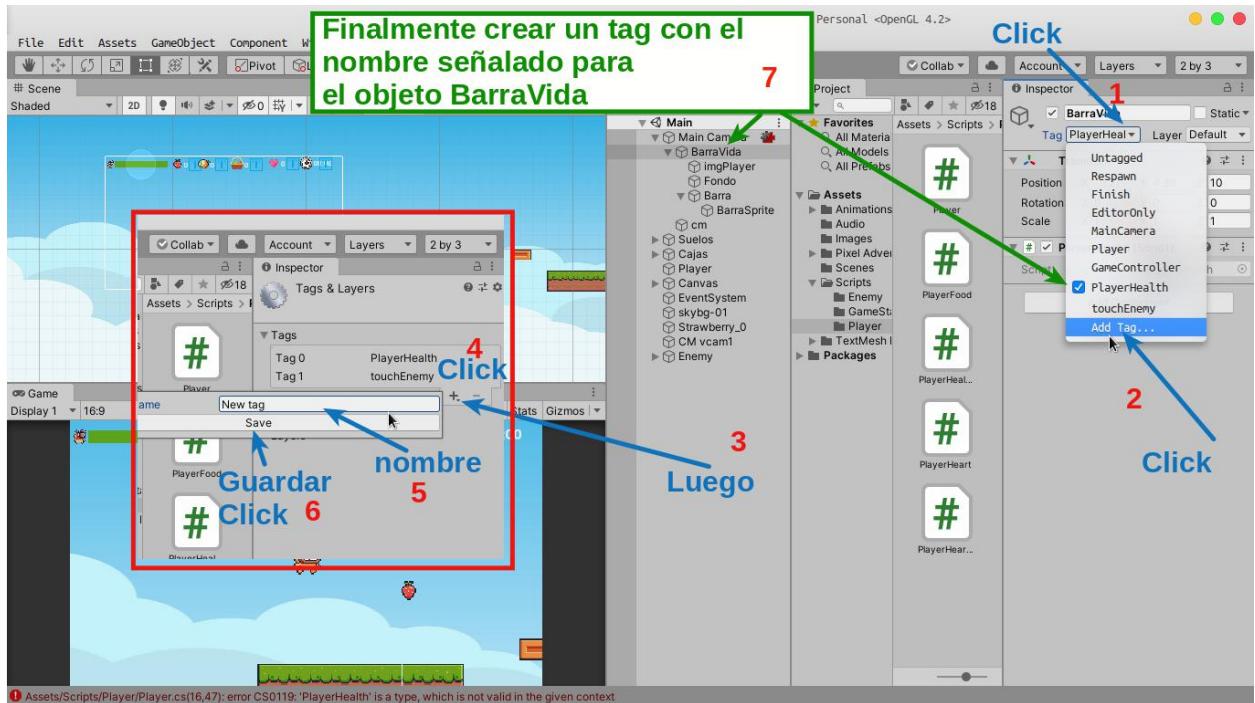
Figura 41. Diseño final de la barra de vida del jugador



6.14.9. Agregar un tag a un objeto

Creamos y agregamos el tag al objeto BarraVida.

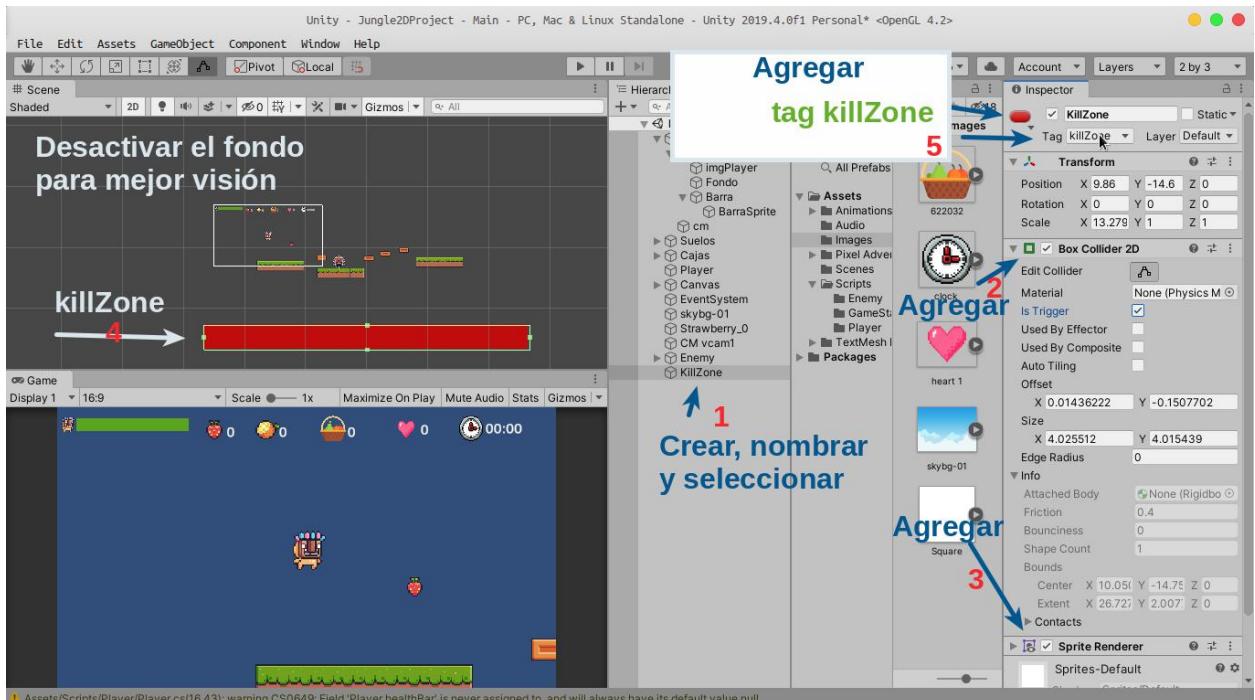
Figura 42. Pasos para crear un tag y asignarlo a un objeto.



6.15. Agregar zona de muerte

Creamos un objeto nuevo, le agregamos un tag, un box collider y un sprite para darle color.

Figura 43. Pasos para agregar la zona de muerte



6.16. Agregar frutas al juego

Se agregan de la misma manera que un jugador, contiene una animación por defecto. Por lo pronto bastará con uno, ya que se tiene que agregar un Script y luego se replicaran para no hacer doble trabajo agregando el Script a cada fruta luego.

6.16.1. Primera fruta del juego

En el paquete hay muchas frutas, como ejemplo se escogieron las fresas.

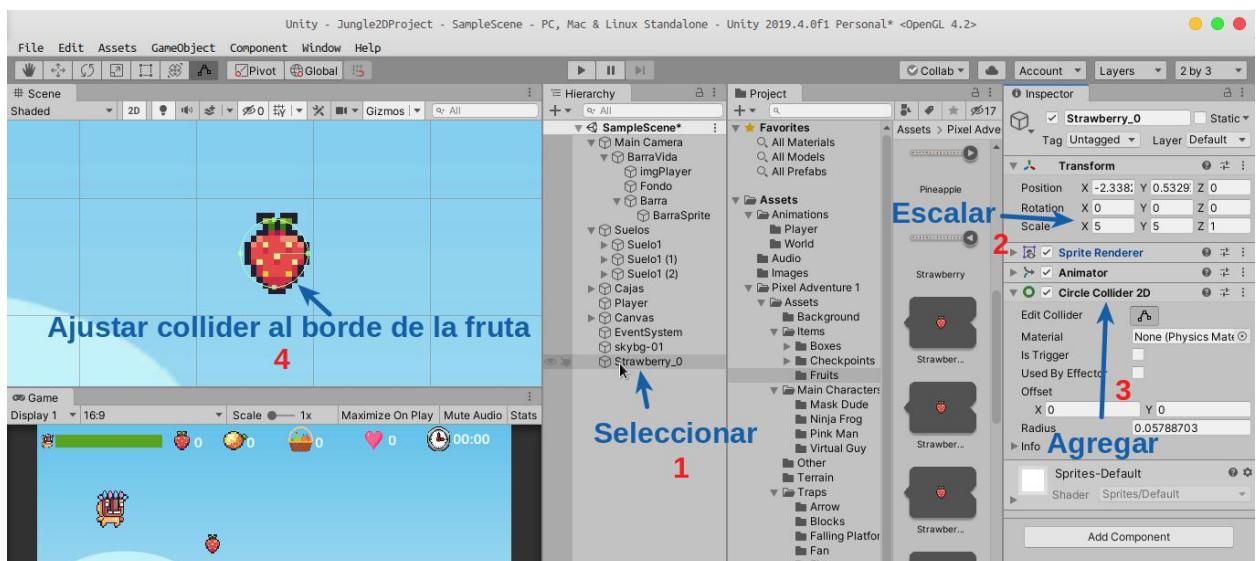
Figura 44. Pasos para agregar una fruta



6.16.2. Escalar y agregar componente Circle Collider 2D

El collider nos ayudara a identificar una colisión con cualquier otro objeto que tenga un collider.

Figura 45. Pasos para agregar un collider y escalar el objeto

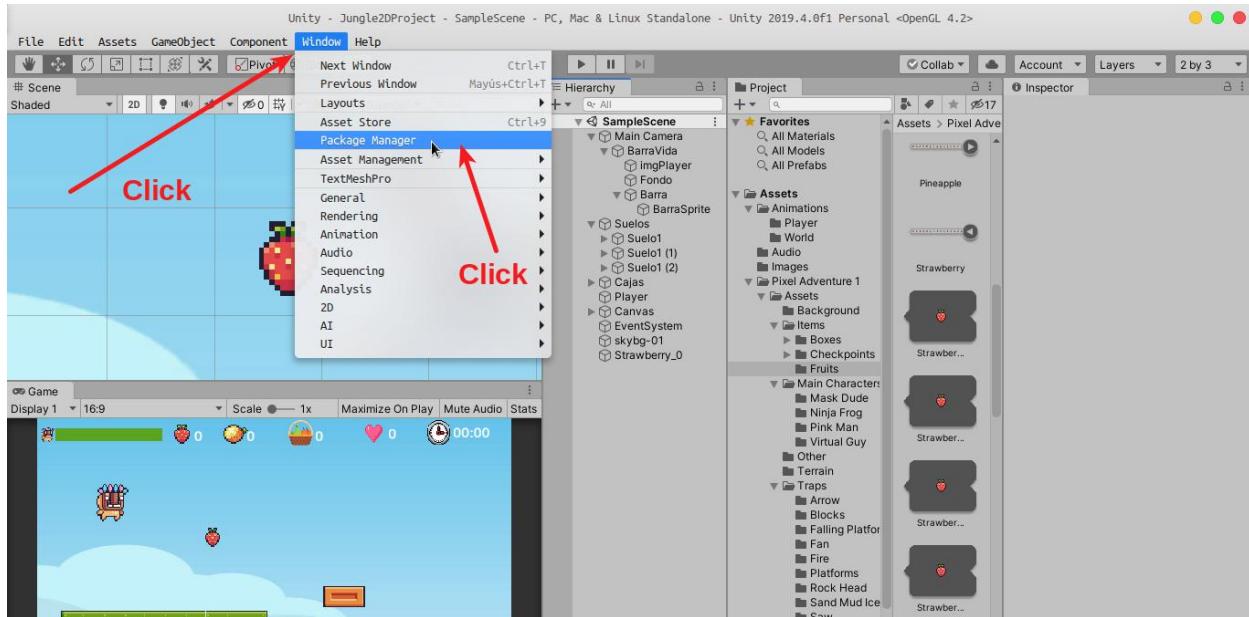


6.17. Agregar plugin para la cámara

6.17.1. Abrir el administrador de paquetes

Con el administrador de paquetes podremos instalar plugins especiales.

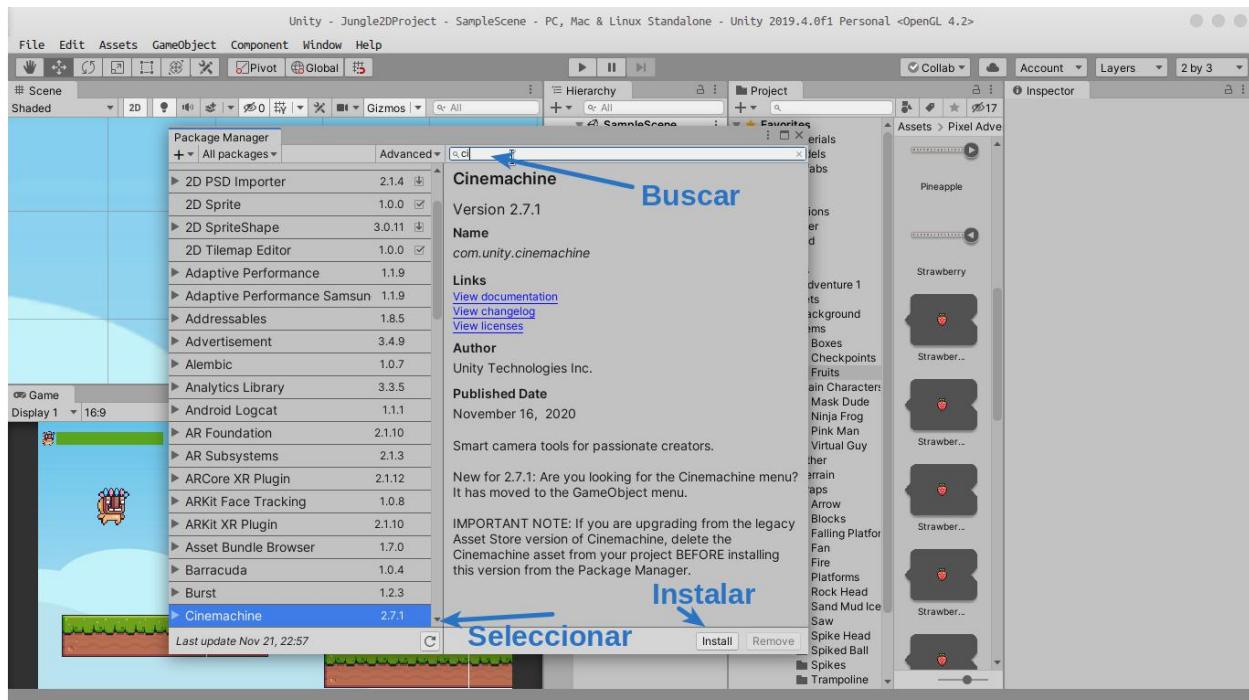
Figura 46. Pasos para abrir el administrador de paquetes



6.17.2. Buscar e instalar Cinemachine

Es un plugin que nos ayudara a seguir con la camara al jugador cuando este se mueva por el mapa o suelos.

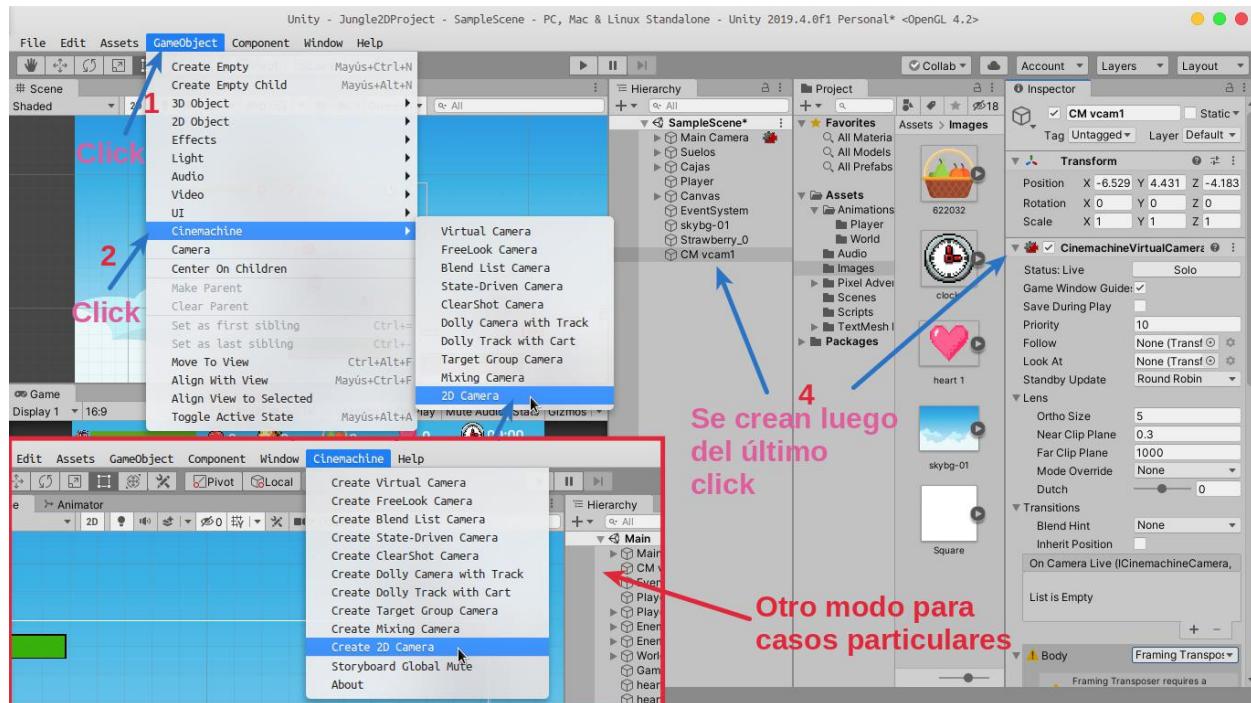
Figura 47. Pasos para instalar cinemachine



6.17.3. Agregar objeto cámara 2D desde virtual machine

Luego de instalar el plugin, lo que hacemos es crear un objeto de con nombre 2D Camera.

Figura 48. Pasos para agregar 2D camera desde virtual machine



6.17.4. Agregar jugador a la cámara de cinemachine

Se vinculan los dos objetos para que el jugador pueda ser seguido por la cámara.

Figura 49. Pasos para agregar el jugador a la cámara de cinemachine



6.17.5. Configurar del área de la cámara

Mediante esta configuración se definirá la activación del seguimiento de la cámara, cuándo el jugador salga de la área definida.

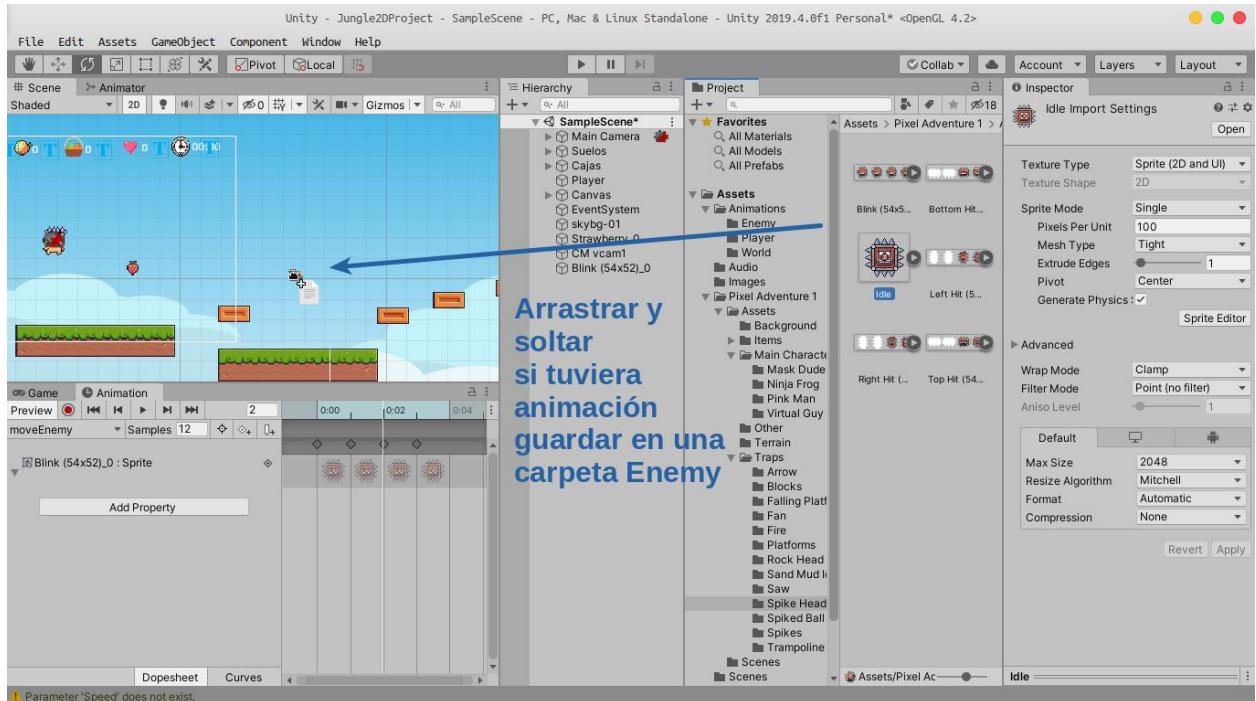
Figura 50. Definir área de seguimiento al player



6.18. Agregar un enemigo

Se agrega como cualquier otro objeto agregado anteriormente.

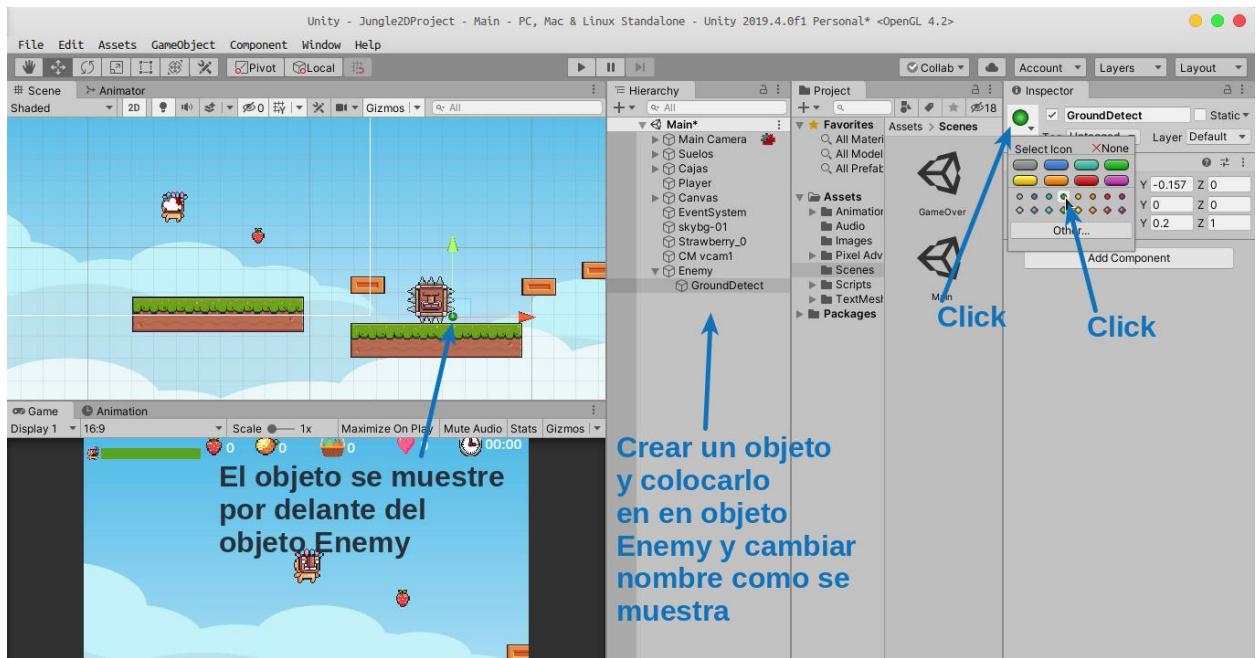
Figura 51. Agregando un enemigo



6.18.1. Agregar un detector de suelo para el enemigo

Crear el objeto detector de suelo.

Figura 52. Crear un objeto detector.



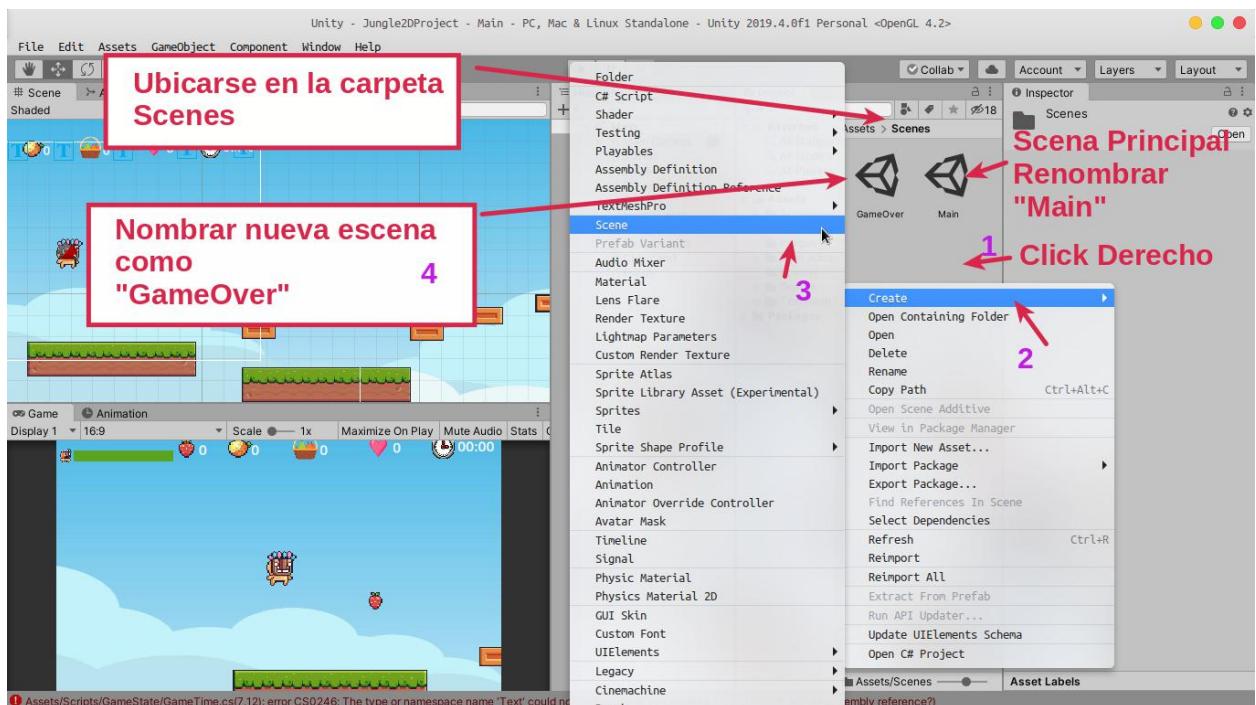
6.19. Crear nueva escena

Cambiamos de escenas para mostrar otros mapas, estados de juego, etc.

6.19.1. Crear Scene

Creamos un Scene para el juego terminado.

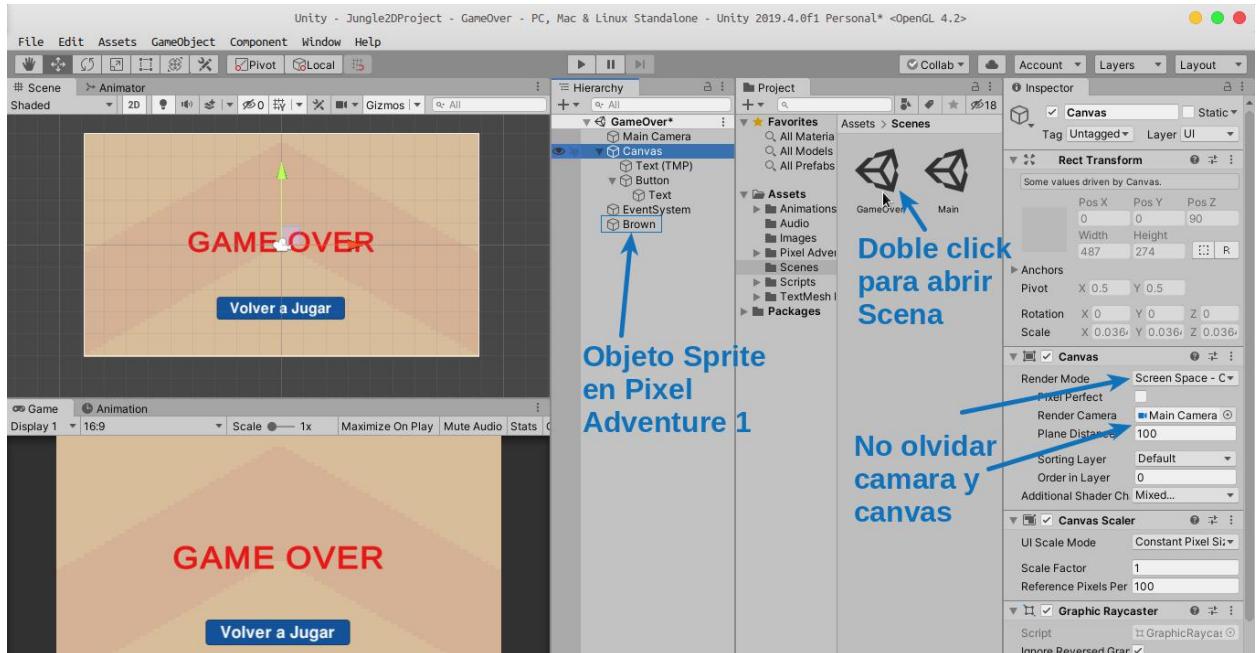
Figura 53. Pasos para crear una escena y renombrar escenas



6.19.2. Diseñar scene gameover

Se diseña la escena gameover y algunas configuraciones más, para posicionar objetos se utiliza Position en Z (10, 50, 100+ o -10, -20, -100+) .

Figura 54. Configurar y diseñar escena gameover



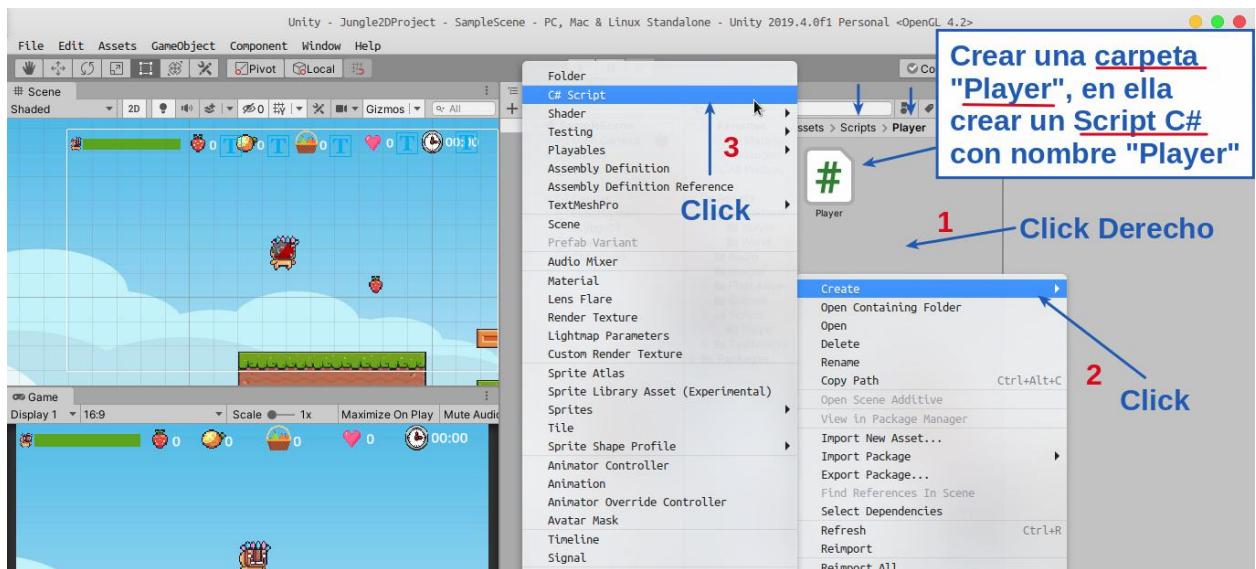
7. Crear script para probar el desplazamiento del jugador

Creamos scripts para controlar algunas acciones e interacciones del jugador con los objetos y viceversa.

7.1. Script de prueba para el jugador

Con este script haremos que el jugador salte, se pueda desplazar por los suelos.

Figura 55. Pasos para crear un script



7.1.1. Editando el script creado

Abrir el script del player con un editor de código e insertar el siguiente código, “Sólo agregar código no repetido”, una vez editado guardar los cambios hechos en el script.

Código C# 1. Código para el player

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player : MonoBehaviour
{
    /// <summary>
    /// Creando variables globales
    /// </summary>
    public float MovementSpeed;
    public float JumpForce;
    public Animator animator;
    public Rigidbody2D rigidbody;

    void Start()
    {

    }

    void Update()
    {
        var movement = Input.GetAxis("Horizontal");
        animator.SetFloat("Speed", Mathf.Abs(movement));
        transform.position += new Vector3(movement, 0, 0) * Time.deltaTime * MovementSpeed;

        // Girar, si movimiento no esta en cero
        if (!Mathf.Approximately(0, movement))
        {
            // Hacer que el jugador gire su cuerpo (Derecha o izquierda)
            transform.rotation = movement < 0 ? Quaternion.Euler(0, 180, 0) : Quaternion.identity;
        }

        // Se preciona la tecla espaciadora y que no salte en el aire.
        if (Input.GetButtonDown("Jump") && Mathf.Abs(rigidbody.velocity.y) < 0.01f)
        {
            // Hacer que el jugador salte
            rigidbody.AddForce(new Vector2(0, JumpForce), ForceMode2D.Impulse);
        }
    }
}
```

7.1.2. Agregar el script editado al objeto Player.

Ahora se vincula el script editado al objeto player, además de algunos componentes.

Figura 56. Pasos para agregar el script al objeto player, animator y rigidbody



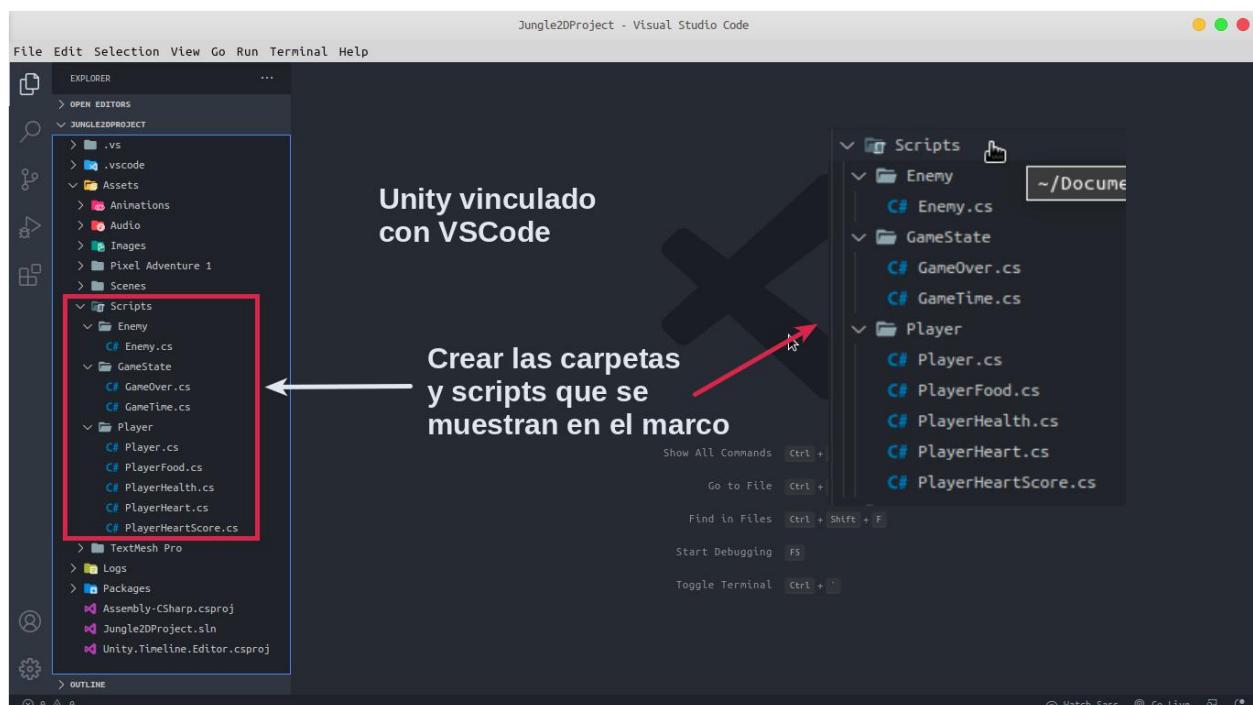
8. Creando la estructura de scripts para el proyecto

Creamos la estructura de scripts para tener un proyecto ordenado.

8.1. Creando la estructura de los scripts

Crear los scripts dentro del proyecto de unity, pero mantener la estructura mostrada en la “*Figura 52. Modelo de la estructura de los scripts para el proyecto*”.

Figura 57. Modelo de la estructura de los scripts para el proyecto



9. Editando los Scripts C#

Editamos el código de cada script.

9.1. Carpeta GameState

En esta carpeta se encuentran algunos estados del juego, como el gameover y el estado del tiempo.

9.1.1. Script GameOver.cs

Con este script podremos regresar a la escena main para reiniciar el juego, con el botón restart.

Código C# 2. Código para controlar el estado de game over

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class GameOver : MonoBehaviour
{
    public Button btnRestart;

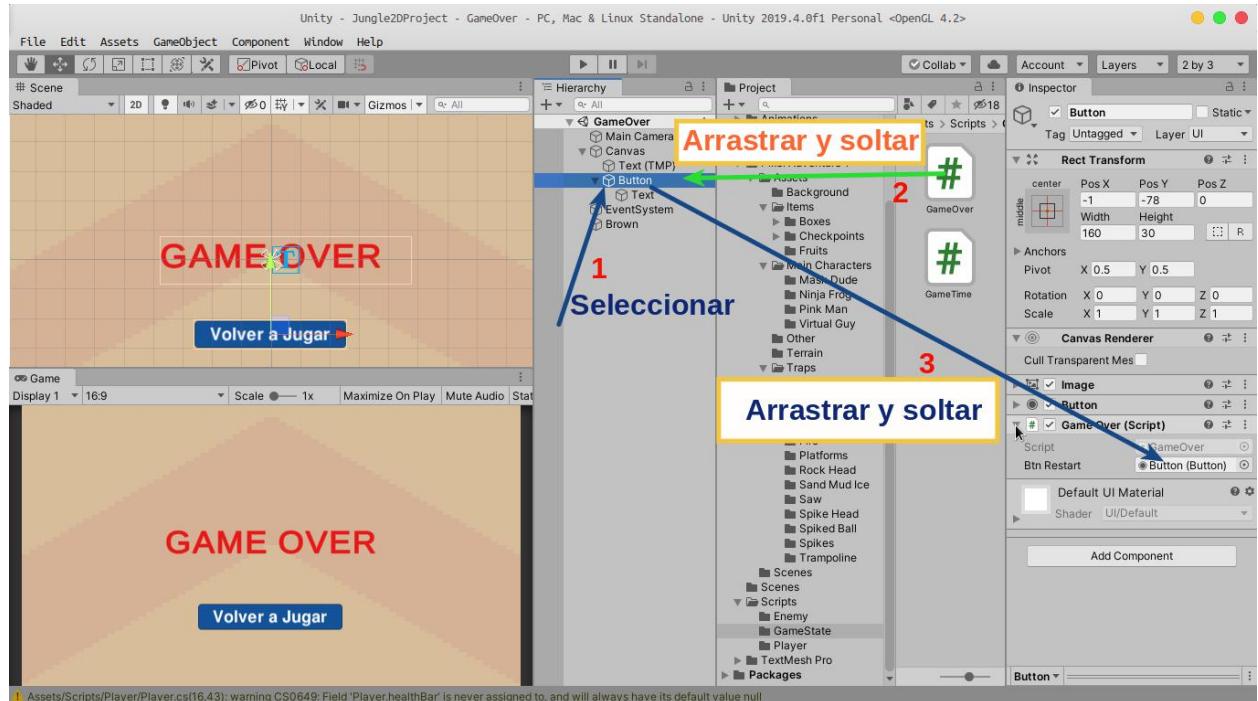
    void Start () {
        btnRestart.onClick.AddListener(TaskOnClick);
    }

    void TaskOnClick(){
        PlayerHeartScore.playerHeartScore = 3;
        SceneManager.LoadScene("Main");
    }
}
```

9.1.2. Agregar el script al botón y el botón al script

“En principio el script se creo para otra función, pero a fin de no alterar estructuras anteriores se continuará con el nombre dado”, lo que hace el botón es reiniciar el juego, no terminarlo.

Figura 58. Pasos para agregar el script al botón



9.1.3. Script GameTime.cs

Controlamos el tiempo transcurrido y lo mostramos en la ventana del jugador.

Código C# 3. Enviar una cadena de texto al text mesh pro en el canvas del proyecto

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class GameTime : MonoBehaviour
{
    public TextMeshProUGUI timeText;
    public float timeRemaining = 0; //el valor se manda desde unity
```

```
void Start()
{
    timeText.text = timeRemaining.ToString();
}

void Update()
{
    timeRemaining -= Time.deltaTime;

    float minutes = Mathf.FloorToInt(timeRemaining / 60);
    float seconds = Mathf.FloorToInt(timeRemaining % 60);

    timeText.text = string.Format("{0:00}:{1:00}", minutes,
        seconds);

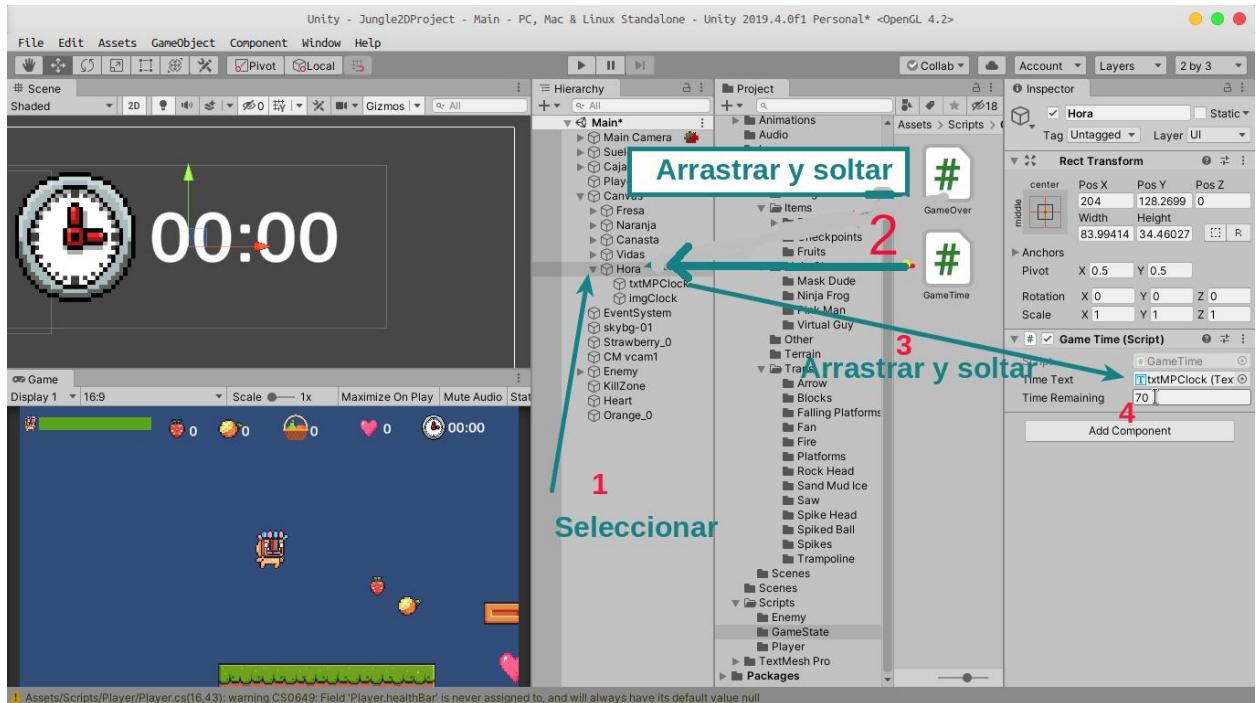
    if (timeRemaining <= 0) {
        // vuelve a cargar el juego
        SceneManager.LoadScene("Main");

        //Disminuye un corazon
        PlayerHeartScore.playerHeartScore -= 1;
    }
}
```

9.1.4. Agregar el script al padre del texto y el texto al script

Tener en cuenta que el texto tiene un objeto padre.

Figura 59. Pasos para agregar el script al padre del texto y el texto agregar al script



9.2. Carpeta Enemy

En esta carpeta están los scripts para el enemigo.

9.2.1. Script Enemy.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour
{
    public float enemySpeed;
    public float rayDist;
    private bool movingRight;
    public Transform groundDetect;
    public static float EnemyDamage = .2f;
    // Update is called once per frame
}
```

```
void Update()
{
    transform.Translate(Vector2.right * enemySpeed *
Time.deltaTime);

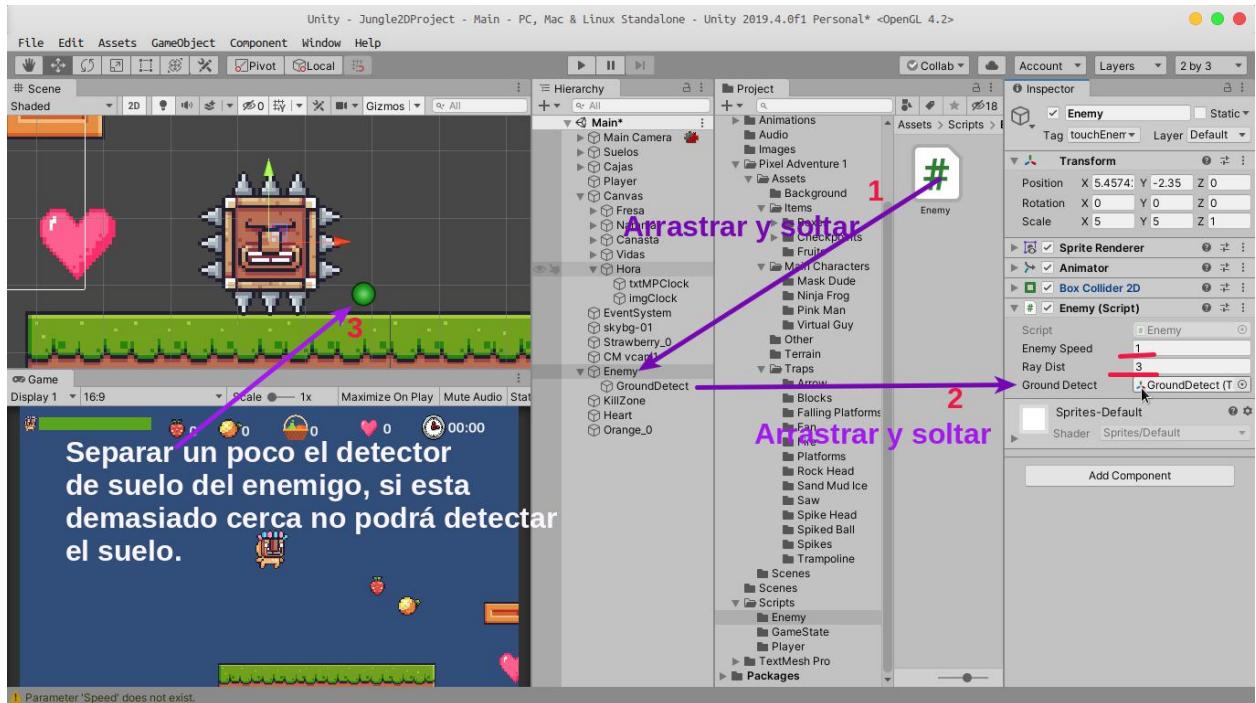
    RaycastHit2D groundCheck =
Physics2D.Raycast(groundDetect.position, Vector2.down,
rayDist);

    if (groundCheck == false){
        if (movingRight)
        {
            movingRight = false;
            transform.eulerAngles = new Vector3(0, 180, 0);
        }
    }
    else
    {
        movingRight = true;
        transform.eulerAngles = new Vector3(0, 0, 0);
    }
}
}
```

9.2.2. Agregar el script al enemigo y el detector de suelo del enemigo al script

Se muestra algunas indicaciones para que el detector de suelo pueda funcionar de modo correcto.

Figura 60. Pasos para agregar el script al enemigo



9.3. Carpeta Player

Editamos todos los scripts para el objeto Player.

9.3.1. Script PlayerHeartScore.cs

Controlamos la cantidad de vidas en el juego, es decir cantidad de juegos, también es un modo distinto para controlar un componente textMeshPro.

Código C# 4. Controlamos la cantidad de vidas y se muestra en la pantalla

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class PlayerHeartScore : MonoBehaviour
{
    public static int playerHeartScore = 3;
    TextMeshProUGUI textHeart;
```

```

void Start()
{
    textHeart = GetComponent<TextMeshProUGUI>();
}

// Update is called once per frame

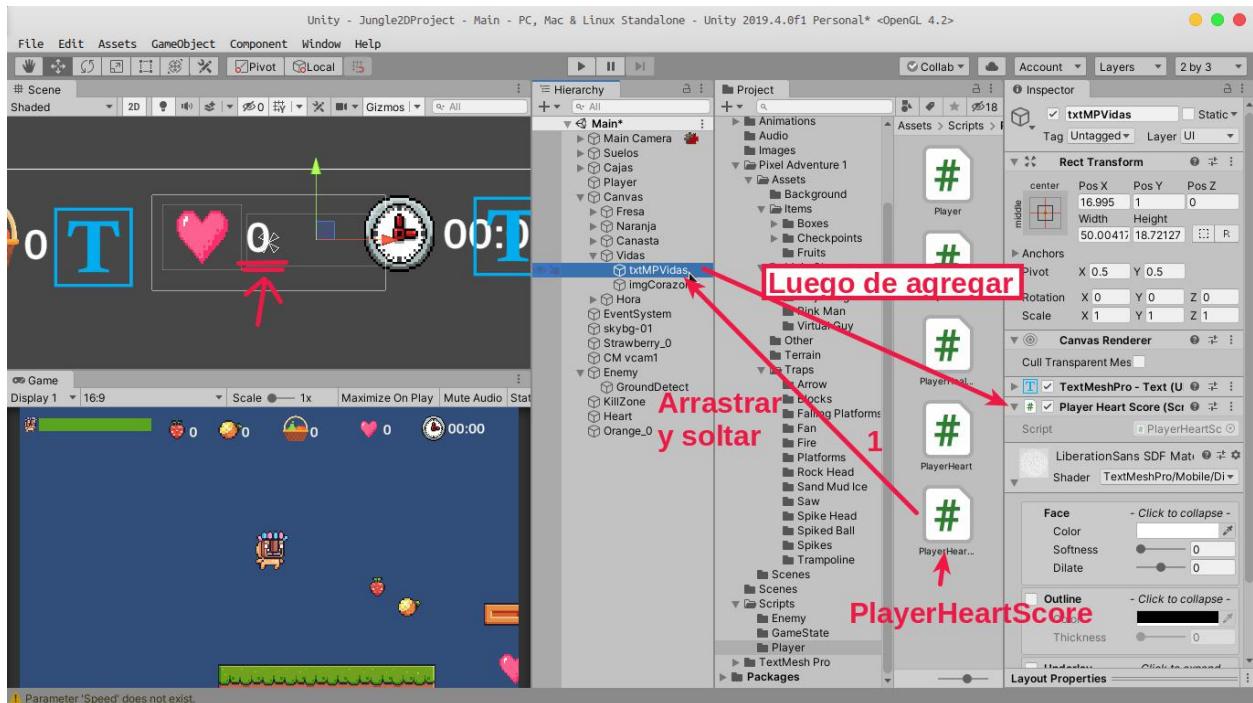
void Update()
{
    textHeart.text = playerHeartScore.ToString();
}

```

9.3.2. Agregamos el script anterior en el txtMPVidas

Este es un modo distinto para agregar script a un texto.

Figura 61. Pasos para agregar el script al text mesh pro



9.3.3. Script PlayerHeart.cs

Es un script para destruir el objeto Heart cuando el jugador colisione con el.

Código C# 5. Código para destruir el objeto Heart

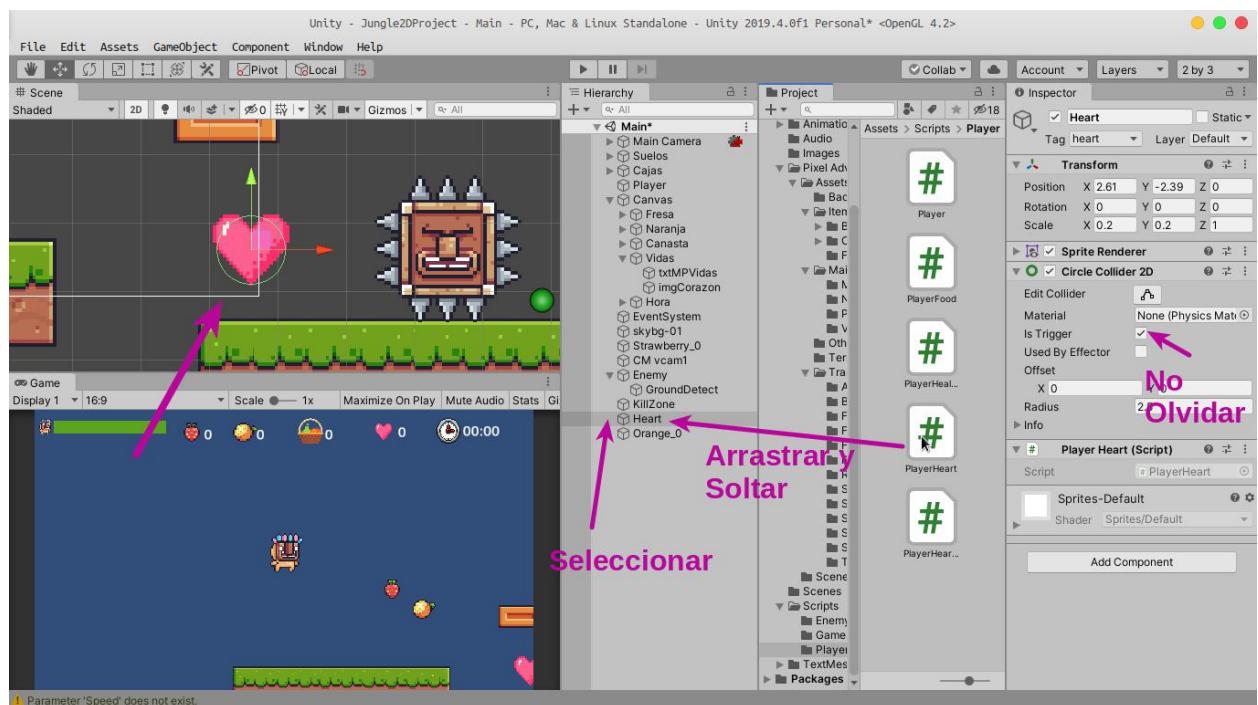
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerHeart : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        Destroy(gameObject);
    }
}
```

9.3.4. Agregando el script anterior al objeto Heart

No olvidar que ese objeto tiene un circle collider y el trigger esta marcado y tiene un Tag "heart".

Figura 62. Pasos para vincular el script con el objeto Heart



9.3.5. Script PlayerHealth.cs

Con este script controlaremos la barra de vida.

Código C# 6. Código para la barra de vida

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class PlayerHealth : MonoBehaviour
{
    private Transform transformBar;

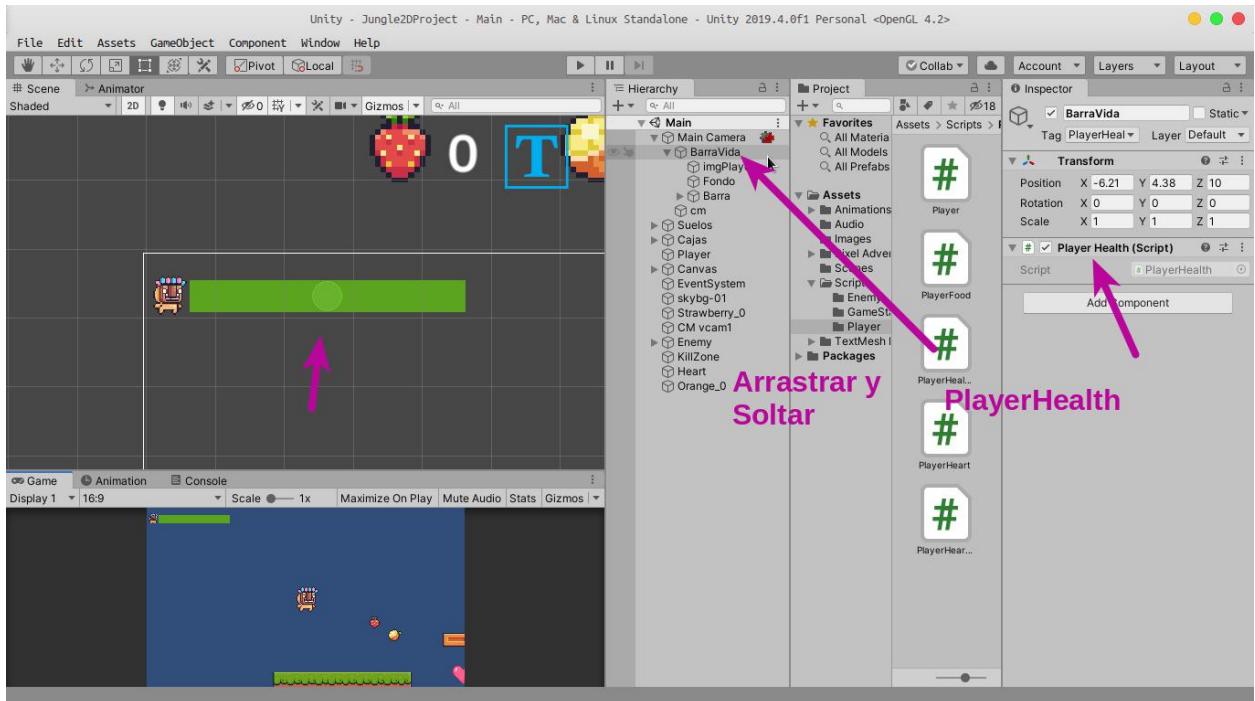
    void Start()
    {
        transformBar = transform.Find("Barra");
    }

    public void setScaleSize(float sizeNormalized)
    {
        transformBar.localScale = new Vector3(sizeNormalized, 1f);
    }
}
```

9.3.6. Agregamos el script anterior al objeto BarraVida

Se vincula el script y el objeto, de ese modo podremos capturar algunos atributos del objeto BarraVida e instanciarlo en otros scripts.

Figura 63. Agregar el script playerhealth



9.3.7. Script PlayerFood.cs

Con el script controlamos todas las frutas creadas, al duplicar cada fruta tendra el mismo script, de esa manera cada fruta se comportara de manera independiente.

Código C# 7. Código para controlar el valor de las frutas y destruir en caso de colisión

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerFood : MonoBehaviour
{
    // Para las fresas
    public static int playerFoodStrawberry = 5;
    // Para las naranjas
    public static int playerFoodOrange = 10;

    private void OnTriggerEnter2D(Collider2D collision) {
```

```

GetComponent<SpriteRenderer>().enabled = false;
gameObject.transform.GetChild(0).gameObject.SetActive(true);

Destroy(gameObject, .5f);
}

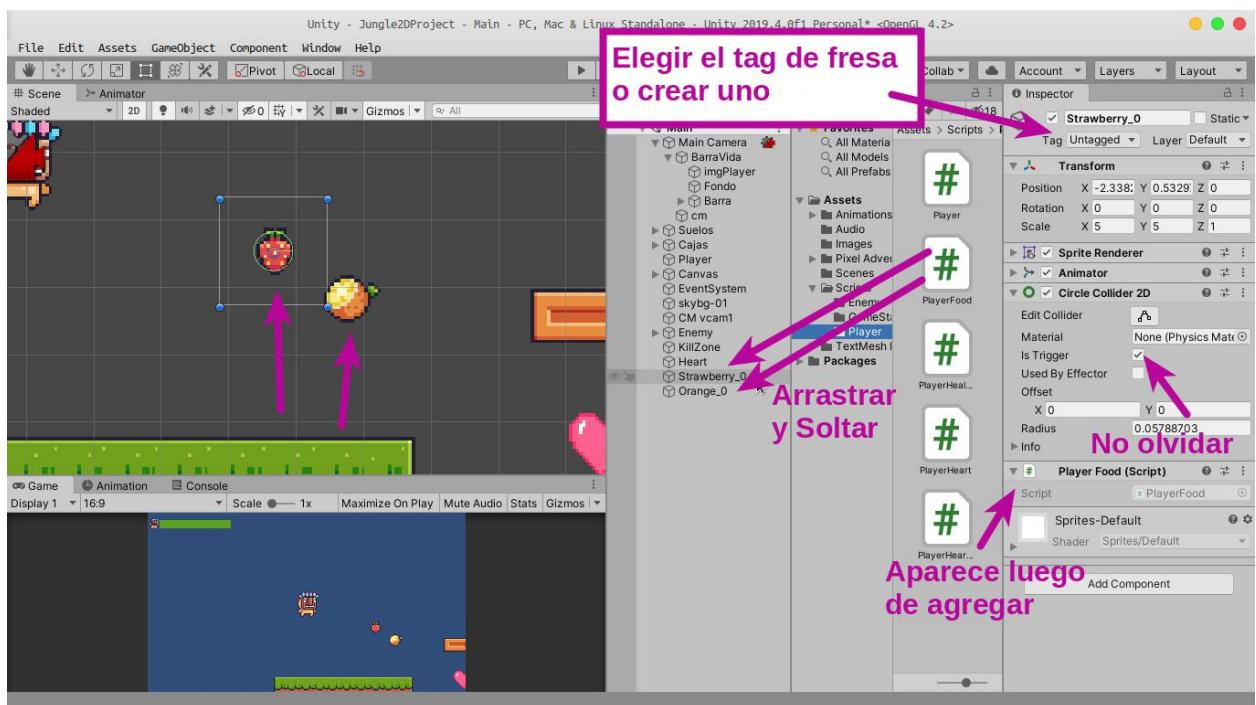
}

```

9.3.8. Agregar el script anterior a los objetos fruta.

El escript agregado nos permite controlar los estados de la fruta.

Figura 64. Agregar script a las frutas, darles un tag a las frutas por su tipo, [naranja](#) o [fresa](#)



9.3.9. Script Player.cs

Con este escript podremos controlar la mayor parte de eventos que interactúen con el jugador, como, recoger frutas, colisionar con el enemigo, caer del suelo y otros.

Código C# 8. Scritp para el objeto player

```

using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;

```

```
using UnityEngine.SceneManagement;

public class Player : MonoBehaviour
{
    // Creando variables globales
    public float MovementSpeed;
    public float JumpForce;
    public Animator animator;
    public new Rigidbody2D rigidbody;

    private Transform transformHealth;
    [SerializeField] private PlayerHealth healthBar;
    private float playerHealth = 1f;

    public TextMeshProUGUI textStrawberry, textOrange, textAllFruit;

    private int playerFoodStrawberry, playerFoodOrange, totalFood;

    void Start()
    {
        transformHealth =
        GameObject.FindGameObjectWithTag("PlayerHealth").transform;
    }

    void Update()
    {
        var movement = Input.GetAxis("Horizontal");
    }
}
```

```
animator.SetFloat("Speed", Mathf.Abs(movement));  
transform.position += new Vector3(movement, 0, 0) *  
Time.deltaTime * MovementSpeed;  
  
  
// Girar, si movimiento no esta en cero  
if (!Mathf.Approximately(0, movement))  
{  
    // Hacer que el jugador gire su cuerpo (Derecha o izquierda)  
    transform.rotation = movement < 0 ? Quaternion.Euler(0, 180,  
0) : Quaternion.identity;  
}  
  
  
// Se preciona la espaciadora del teclado y que no salte en el  
aire.  
if (Input.GetButtonDown("Jump") &&  
Mathf.Abs(rigidbody.velocity.y) < 0.01f)  
{  
    // Hacer que el jugador salte  
    rigidbody.AddForce(new Vector2(0, JumpForce),  
ForceMode2D.Impulse);  
}  
}  
  
  
// Evento para reiniciar el juego "Muerte"  
private void OnTriggerEnter2D(Collider2D collision) {  
    //  
    switch (collision.gameObject.tag)  
    {  
        // Si cae muere
```

```
case "killZone":  
    SceneManager.LoadScene("Main");  
    // Se reinicia las manzanas recogidas por el jugador  
    PlayerHeartScore.playerHeartScore -= 1;  
    break;  
  
    // Si tropieza con el enemigo  
    case "touchEnemy":  
        // Disminuir la vida del jugador  
        playerHealth -= Enemy.EnemyDamage;  
        // Si la vida es menor a 0.1f, termina el juego  
        if (playerHealth < 0.1f)  
        {  
            SceneManager.LoadScene("Main");  
  
            PlayerHeartScore.playerHeartScore -= 1;  
        }  
        // Enviar el daño a la barra de vida  
        healthBar.setScaleSize(playerHealth);  
        break;  
    case "heart":  
        //playerHeart += 1;  
        PlayerHeartScore.playerHeartScore += 1;  
        //textHeart.text = playerHeart.ToString();  
        break;  
    case "strawberry":  
        playerFoodStrawberry += PlayerFood.playerFoodStrawberry;
```

```
textStrawberry.text = playerFoodStrawberry.ToString();  
//  
  
totalFood += PlayerFood.playerFoodStrawberry;  
break;  
  
case "orange":  
  
playerFoodOrange += PlayerFood.playerFoodOrange;  
textOrange.text = playerFoodOrange.ToString();  
//  
  
totalFood += PlayerFood.playerFoodOrange;  
break;  
  
default:  
  
break;  
}  
  
  
// Número de frutas en la canasta  
  
textAllFruit.text = (playerFoodStrawberry +  
playerFoodOrange).ToString();  
  
  
  
if (totalFood >= 50)  
{  
  
totalFood = 0;  
playerHealth += 0.2f;  
  
  
  
if (playerHealth > 1.0f)  
{  
  
playerHealth = 1.0f;  
}
```

```

// Enviar el daño a la barra de vida

healthBar.setScaleSize(playerHealth);

}

// Cantidad de juegos "Corazones" = 0

if (PlayerHeartScore.playerHeartScore == 0)

{
    SceneManager.LoadScene("GameOver");
}
}

}

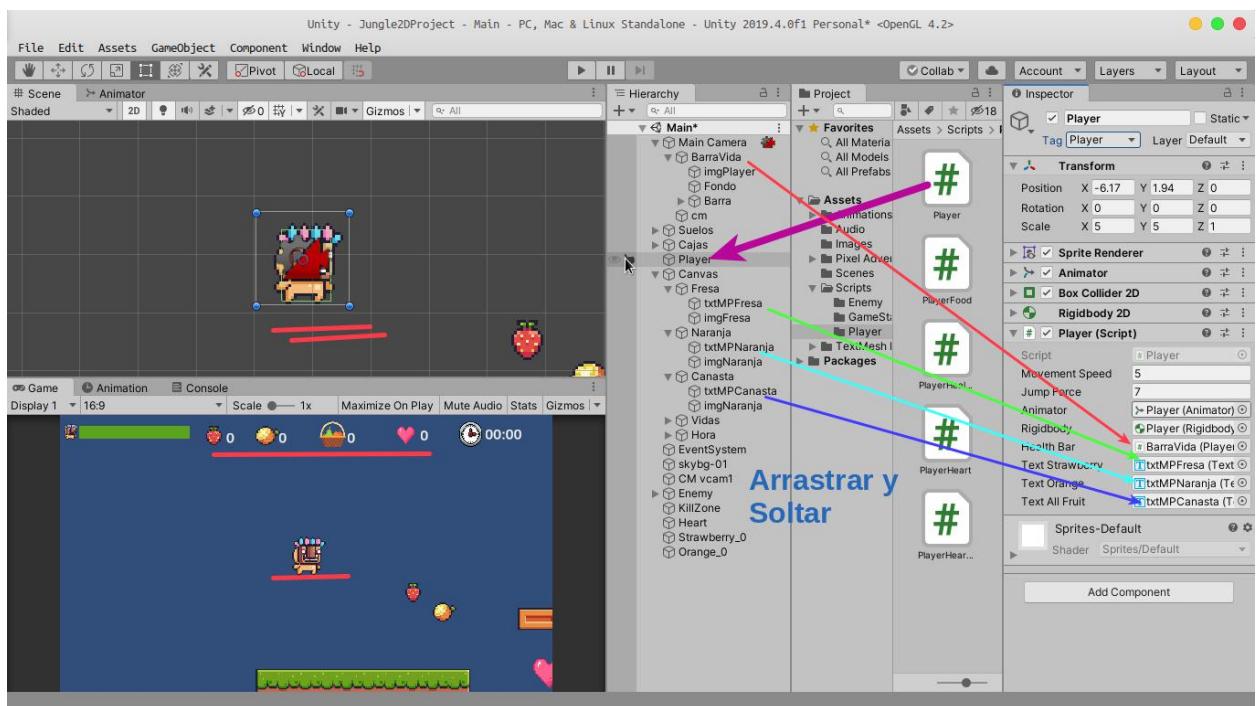
}

```

9.3.10. Agregando el script al objeto Player y componentes/objetos al script

Se vinculan los objetos que actúan como contadores de las frutas, también la barra de vida. Con ese vínculo se podrá controlar los distintos estados del jugador.

Figura 65. Vincular el script principal con el objeto principal



10. Extras.

Algunas animaciones extra.

10.1. Animar recojo de fruta

La fruta hará una pequeña explosión

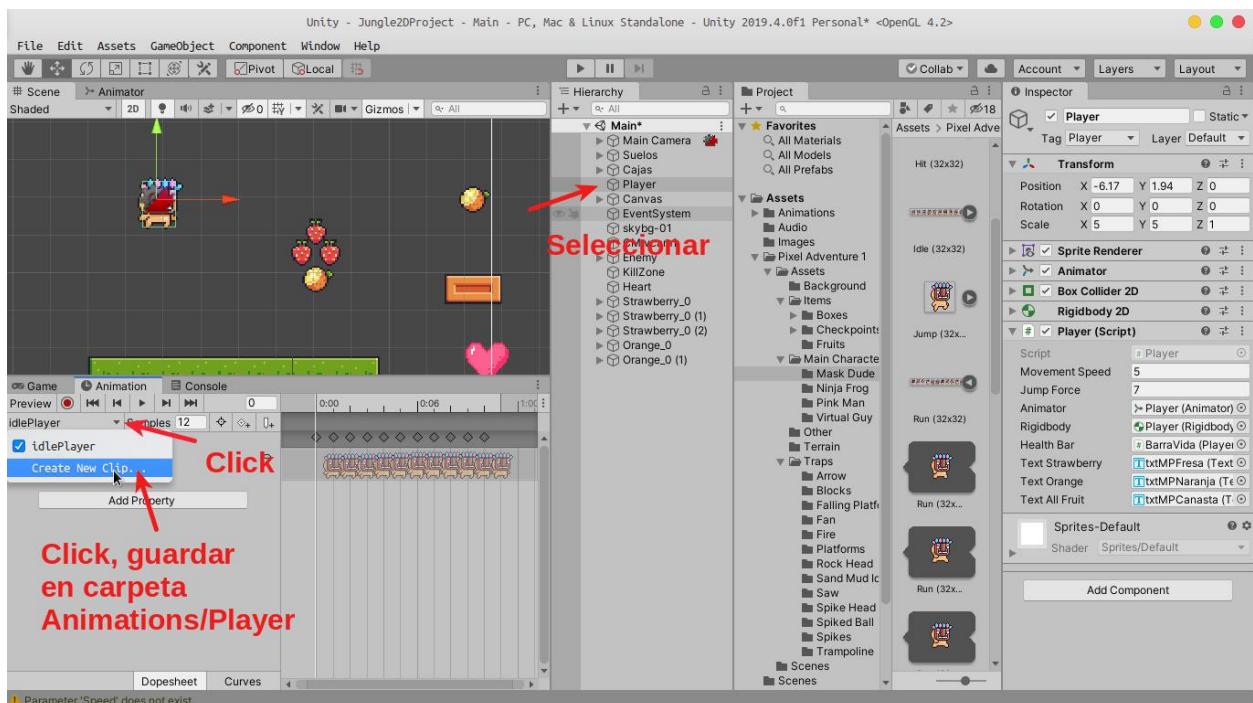
Figura 66. Pasos para crear animación de explosión al tocar la fruta



10.2. Player corriendo.

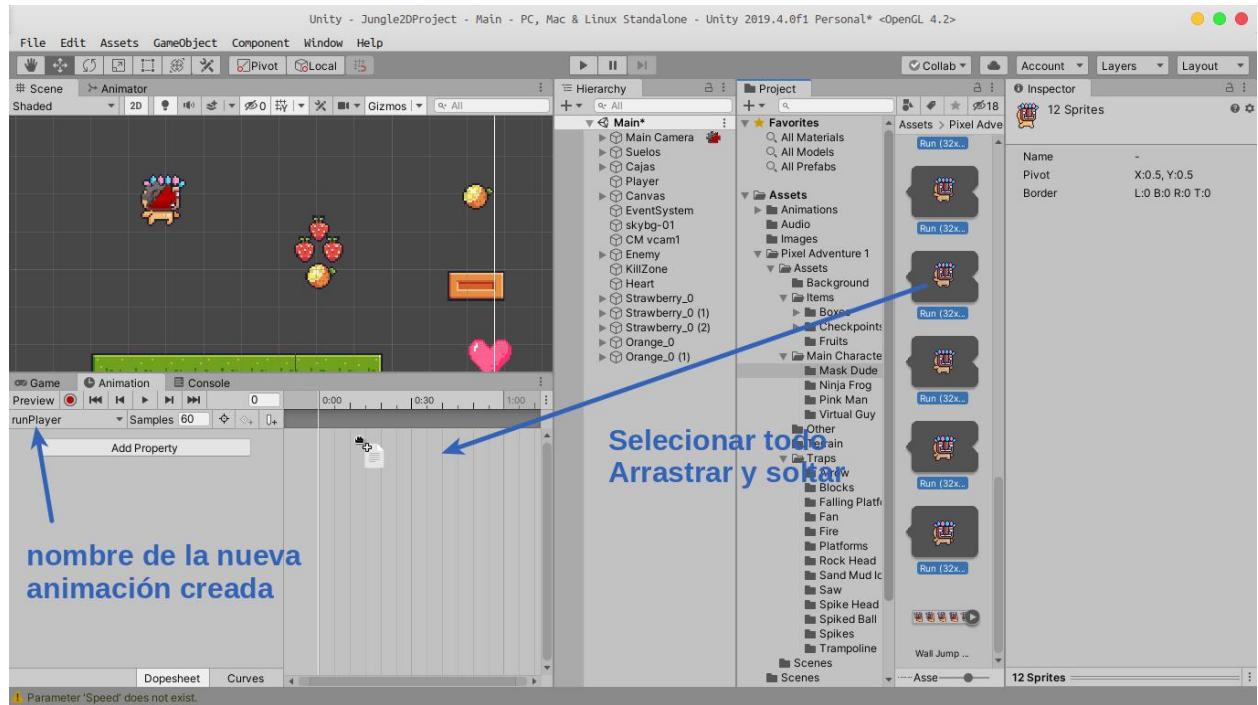
10.2.1. Crear nueva animación

Figura 67. Pasos para crear una animación en el mismo objeto



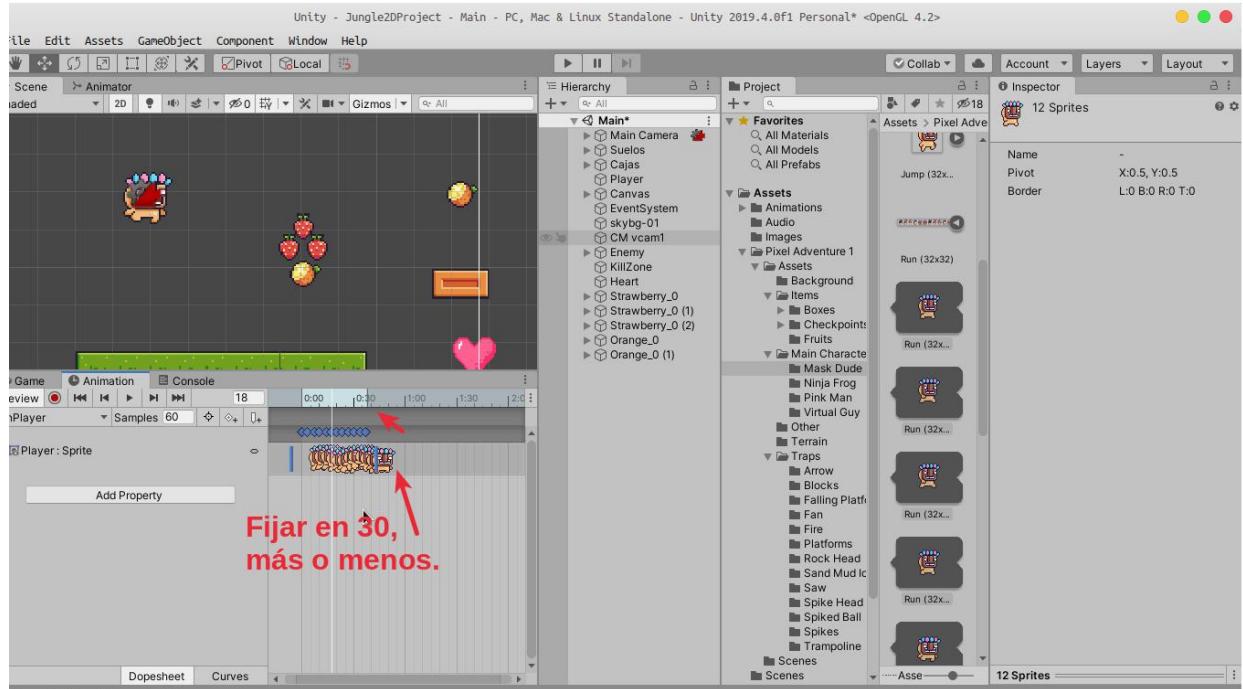
10.2.2. Agregando los sprites a la nueva animación

Figura 68. Agregando los sprites a la nueva animación en el mismo objeto



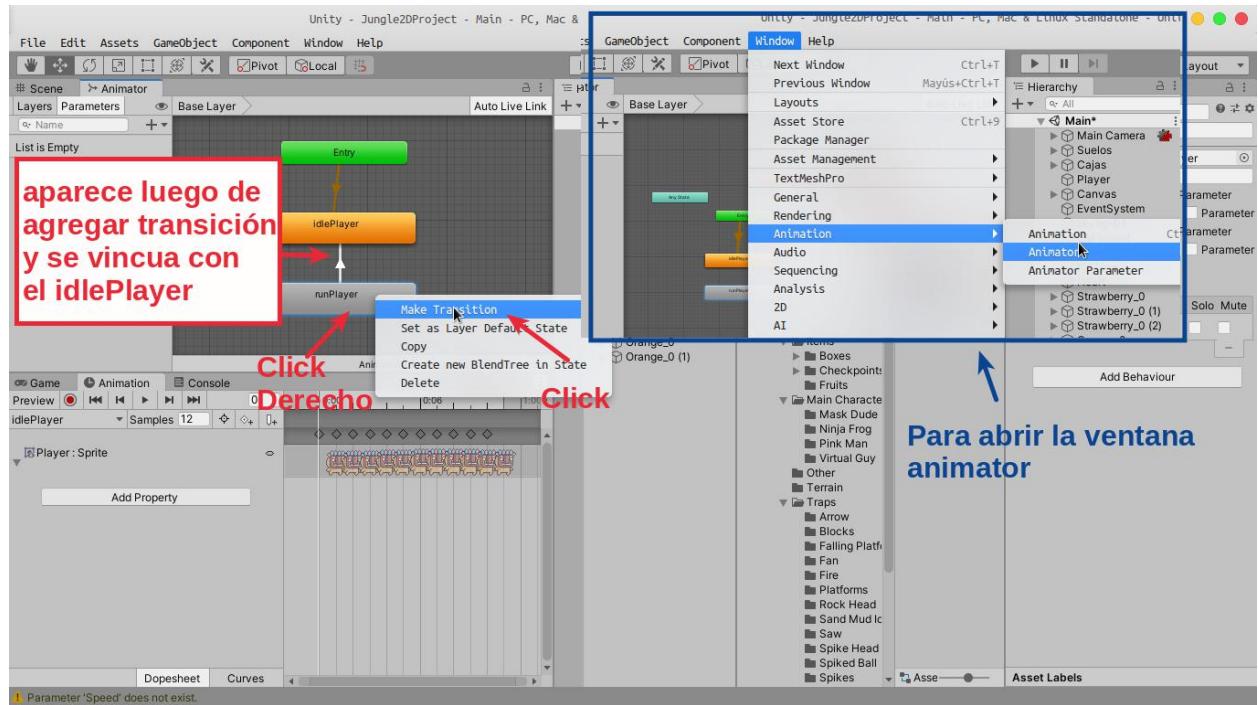
10.2.3. Configurar a +- 30 segundos

Figura 69. Configurar los segundos de animación



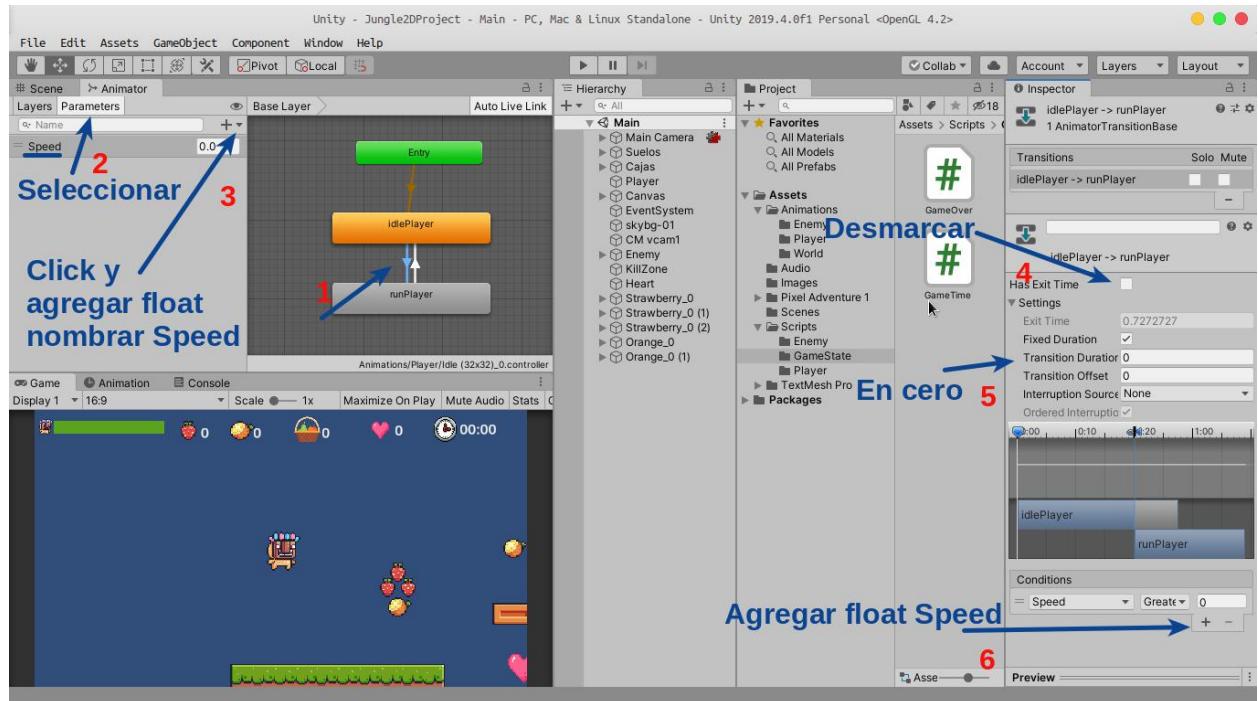
10.2.4. Abrir ventana animator y vincular una transición

Figura 70. Pasos para vincular dos animaciones



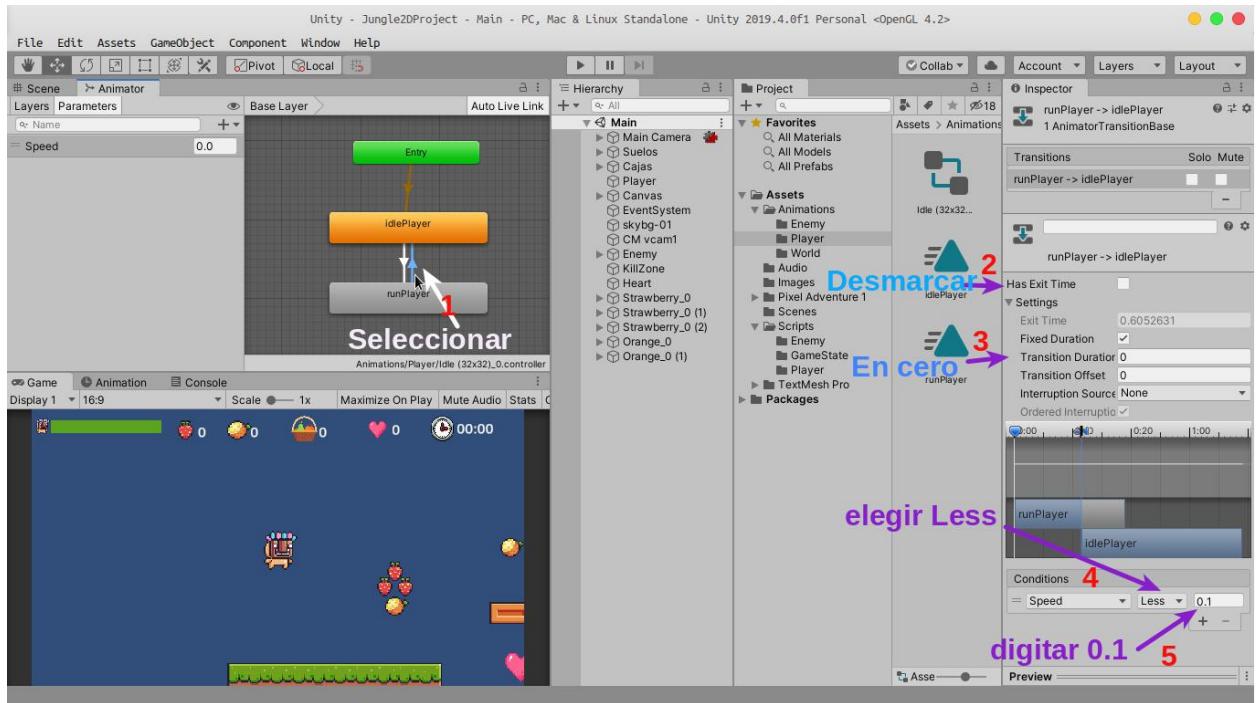
10.2.5. Configurar animación de estado IDLE a RUN

Figura 71. Pasos para configurar la animación de estado detenido a estado corriendo



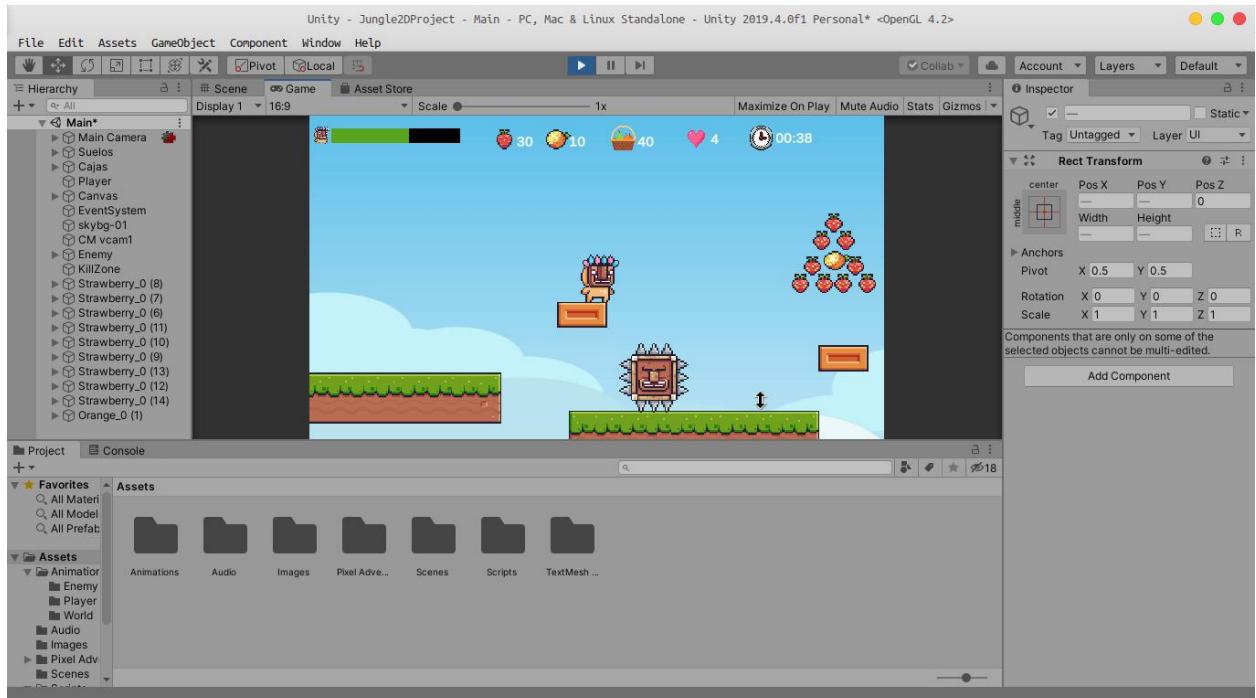
10.2.6. Configurar animación de estado RUN a IDLE

Figura 72. Pasos para configurar animación de estado corriendo a estado detenido



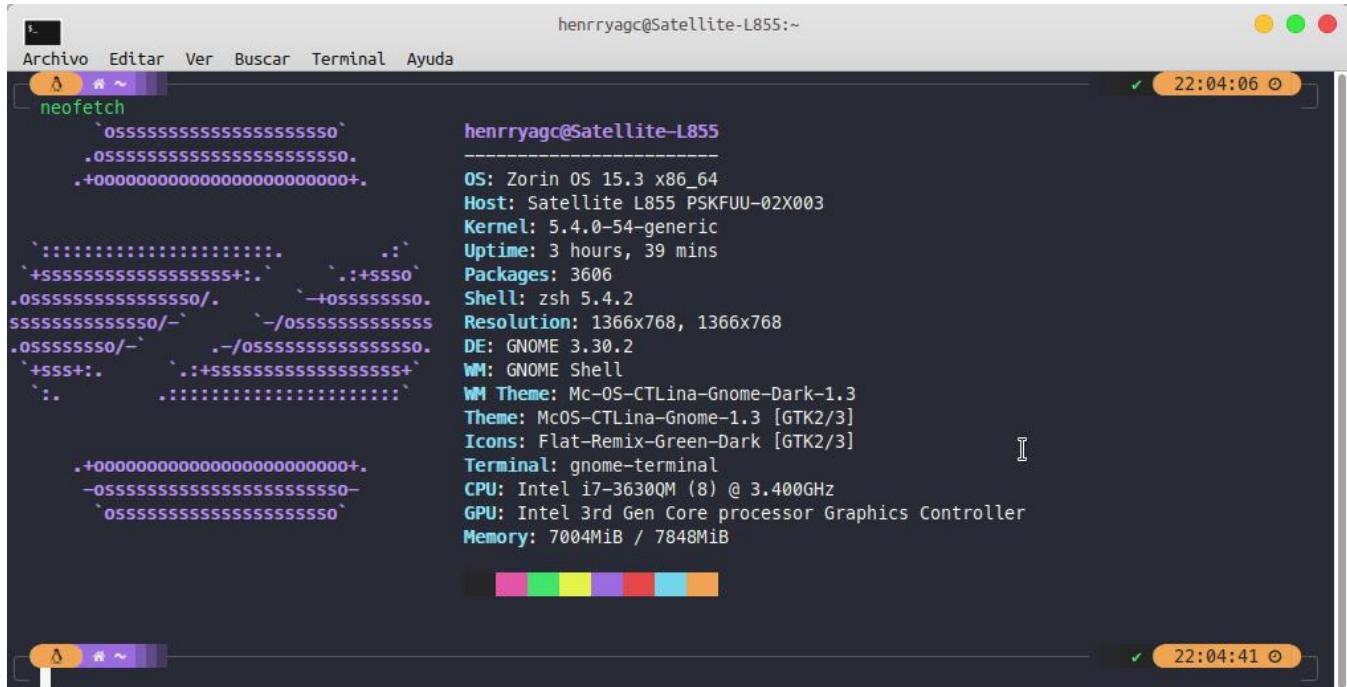
11. Figura Final

Figura 73. Figura final con los componentes en marcha y trabajando



12. Características del equipo utilizado para el desarrollo de el videojuego

SO: ZorinOs basado en ubuntu 18.04



The screenshot shows a terminal window titled "Terminal" with the command "neofetch" run. The output is a combination of a stylized ASCII art logo and system information. The logo is composed of various symbols like dots, dashes, and plus signs. The system information includes:

```
henrryagc@Satellite-L855:~  
-----  
OS: Zorin OS 15.3 x86_64  
Host: Satellite L855 PSKFUU-02X003  
Kernel: 5.4.0-54-generic  
Uptime: 3 hours, 39 mins  
Packages: 3606  
Shell: zsh 5.4.2  
Resolution: 1366x768, 1366x768  
DE: GNOME 3.30.2  
WM: GNOME Shell  
WM Theme: Mc-OS-CTLina-Gnome-Dark-1.3  
Theme: McOS-CTLina-Gnome-1.3 [GTK2/3]  
Icons: Flat-Remix-Green-Dark [GTK2/3]  
Terminal: gnome-terminal  
CPU: Intel i7-3630QM (8) @ 3.400GHz  
GPU: Intel 3rd Gen Core processor Graphics Controller  
Memory: 7004MiB / 7848MiB
```

The terminal window has a dark theme with orange and yellow accents. It shows the date and time at the top right (22:04:06) and bottom right (22:04:41). The title bar also displays the date and time.

13. Conclusiones

El juego no contiene una temática interesante, esto se debe a que el proyecto sólo está orientado al conocimiento y aprendizaje del uso del motor de desarrollo Unity.

Existen pocos objetos para interactuar, pero esto es mucho más sencillo de escalar ya que existen muchos objetos que se pueden replicar y esto haría que el proyecto sea de mucho mayor escala y tenga una jugabilidad más compleja, sólo es cuestión de invertir más tiempo en el desarrollo ya que la base ya está desarrollada.

De cierto modo, el trabajo está ordenado para el lector con poca experiencia y le sea sencillo e intuitivo seguir los pasos a medida que vaya avanzando con los pasos de la guía.

14. Referencias.

Code Learn. (S.F.). *Curso de Videojuegos 2D con Unity*. Consultado el 19 de Noviembre de 2020. <https://codelearn.es/cursos/curso-de-videojuegos-2d-con-unity/>.

Unity Documentation. (2020, 17 de noviembre). *System requirements for Unity 2019.4*. <https://docs.unity3d.com/Manual/system-requirements.html>.

BlackThornprod. (2018, 7 de febrero). *2D Platformer patrol AI with Unity And C# - easy tutorial [video]*. Youtube.

<https://www.youtube.com/watch?v=aRxuKoJH9YO>

LuisCanary. (2020, 22 de abril). *Juego de plataformas 2D/Recoger Frutas/Unity Tutorial/3-Capítulo/Programación Videojuegos* [video].
<https://www.youtube.com/watch?v=vlXF6XhAje8>

Brackeys. (2018, 15 de julio). *Movement in Unity (Tutorial)* [video].
<https://www.youtube.com/watch?v=dwcT-Dch0bA>

Brackeys. (2018, 5 de agosto). *Animation in Unity (Tutorial)* [video].
<https://www.youtube.com/watch?v=hkaysu1Z-N8>

Alexander Zotov. (2017, 228 de agosto). *How to create a simple coin counter in your Unity game? Easy Unity 2D tutorial.* [video].
<https://www.youtube.com/watch?v=-EIXQHxoicq>

Antarsoft. (2020, 9 de septiembre). *Unity 2D Platformer Tutorial 25 - Health Bar + Lives Counter* [video]. <https://www.youtube.com/watch?v=iX0BEiTjrE>

Press Start. (2018, 10 de agosto). *Working with Time in Unity* [video].
<https://www.youtube.com/watch?v=ijANOQI70UU>

John. (2020, 25 de febrero). How to make a countdown timer in Unity (in minutes + seconds). gamedevbeginner. <https://gamedevbeginner.com/how-to-make-countdown-timer-in-unity-minutes-seconds/>

Brackeys, (2018, 9 de septiembre). *2D Camera in Unity (Cinemachine Tutorial)* [video].
<https://www.youtube.com/watch?v=2jTY11Am0lg&list=PLPV2KyIb3jR5QFsefuO2RIAgWEz6EvVi6&index=37>

15. Anexos.

Repositorio: <https://github.com/Henrryagc/UnityGame2D>