Main Camera componentes:

Transform
Position X 0 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camara Script Target Personaje (Transform)
Y offset 1.5

CREAR ESCENARIO
Para crear el escenario de nuestro juego tenemos que utilizar los sprites que hemos añadido al proyecto y empezar a dibujar.
Seleccionamos la carpeta Terrain que es donde tengo la parte de suelo y plataformas y vemos que aparecen dos elementos.

● Terrain 16x16. Aparecen todos los sprites en una única imagen para que nosotros podamos trocearla como queramos. Para ello, la seleccionamos y en la **ventana de Inspector** cambiamos el valor de Sprite Mode a Multiple. A continuación pulsamos el botón de **Open Sprite Editor**

En la **ventana de Sprite editor** puedo realizar los cortes de cada componente según prefiera. Pulsamos sobre Slice y tenemos varias opciones de corte.

Automático. Realiza el corte de los componentes donde detecta un espacio. Una vez realizado el corte, podré modificar el elemento obtenido por si quiero hacerlo más pequeño o más grande.

Grid By Cell Size: En esta opción puedo indicarle los pixeles que va a ocupar cada uno de los cortes.
A continuación, empezamos a dibujar nuestro escenario. Para ello añadimos un nuevo GameObject llamado Tilemap.

Los objetos Tilemap, Tilemap Fondo y Blue tienen que estar dentro del Objeto Grid

Grid Transform Position x 0 y 0 z 0
Rotation x 0 y 0 z 0
Scale x 1 y 1 z 1
Grid
Cell Size x 0.16 y 0.16 z 0
cell gap x 0 y 0 z 0

cell layout rectangle
cell swizzle XYZ

Necesitamos mostrar la ventana llamada Tile Palette para empezar a dibujar mi escena. En caso de que no aparezca por defecto, podemos añadirla desde Windows → 2D → Tile Palette.

Una vez aparezca la ventana, arrastramos el sprite que troceamos en los pasos anteriores a la ventana del Tile Palette y a continuación, comenzamos a dibujar.

Para añadir físicas y colisiones al personaje ya sabemos que tenemos que añadirle los componentes RigidBody (En este caso 2d) y un Collider 2D (Que puede ser de tipo Box o Capsule).

Debemos modificar algunos de los parámetros del RigidBody y del Collider para evitar problemas futuros en cuanto a que el personaje de trabe al chocar contra las plataformas.

Personaje Rigibody
Dynamic
MASS 1
lINEAR Damping 0 Angular 0.05
Gravity Scale 1
Collision Detection Continous
Sleeping Star Awake
Interpolate
Constraints Freeze Rotation Z

Capsule Collider
Material New Physics Material 2D

Composite Operation None
Offset x 0 y 0 otra opcion Offset X -3.5762 y -0.0279
size x 1.1 y 0.0001 x 0.20826 y 0.26413
Direction Vertical

Para crear Material New Physics Material tengo que ir a Assets > Create > 2D > Physics Material 2D

Friction 0
Bounciness 0
Friction Combine Mean
Bounce Combine Maximun

Esto se añade al Material de Capsule Collider 2D del Personaje donde pone Material.

Transform todo a 0 pero Scale X 1 Y 1 z 1

Componentes de Tilemap
Tilemap Animation 1
Tile Anchor x 0.5 y 0.5 z 0
Orientation XY

Tilemap Renderer Sort Bottom Left
Mode Chunk
Detect Chunk Auto
Mask Interation None
Material Blue
ADditional Settings Order in Layer 1

Tilemap Collider 2D
Max Tile 1000

y hay que añadir el objeto Blue

Tilemap Fondo
Scale todo 1
Tilemap Animation 1
Tile Anchor 0.5 0.5 z 0
Xy
Material Blue
Order in Layer -1 hay que añadirle Blue

Blue
Position -2.84 y -4.67 z 0
rotation todo a 0
Scale todo a 1

Sprinter Renderer

Material Sprite Lit Default

Movimiento de jugador
añadir el componente al personaje Player Input.
Lo único que tengo que tener en cuenta en este caso, es que estamos en un juego 2D, con lo que sólo necesito utilizar los valores de la coordenada X para moverme por el juego.

Orientar Personaje
Hemos añadido movimiento a mi personaje pero nos damos cuenta que siempre mira hacia la misma dirección. Deberíamos poder cambiar la dirección para que el personaje mire hacia la derecha si he pulsado la flecha derecha y mire a la izquierda si he pulsado la flecha izquierda.
Este comportamiento lo implementamos cambiando de signo el valor de la propiedad X del componente transform.localScale.

Transform del Personaje
Position -3.5 Y -3.4 Z 0 otra opción x -1.983 y -0.592 z 0 tag Player
rOTATION TODO 0 todo 0
Scale todo 0.5 scale todo 0

dentro del componente Script

Aparece el Script
Velocidad
Fuerza Salto
Layer Suelo Suelo
Comprueba Sueldo se añade el objeto Comprueba Suelo
Audio Source
Get Recolectar el sonido cuando se consigue el punto
Sonido Muerto cuando te mata el enemigo

Añadir el Animator con controller Personaje
Orientar personaje

Dentro del método OnMove, tendríamos

Salto de personaje

Para programar el salto del personaje utilizaremos el método OnJump que nos proporciona el Player Input de Unity.

Tendremos un comportamiento parecido al del movimiento horizontal, pero lo que tenemos que hacer ahora es modificar el valor de la propiedad Y del personaje.
Creamos una variable pública de tipo float llamada fuerzaSalto para controlar la potencia del salto del personaje.

El problema que tenemos ahora es que cada vez que pulsamos la tecla de espacio el personaje salta, da igual que esté en el aire, con lo que podemos encadenar dobles o triples saltos.
Tenemos que asegurarnos que para que se produzca el salto, nuestro personaje esté en el suelo. Crearemos un objeto vacío debajo de nuestro personaje y detectaremos si ese objeto está tocando el suelo.
Para ello, tendremos que realizar varias configuraciones.

Vamos a añadirle a todo el Tilemap una capa con el nombre “Suelo” para que podamos comprobar si nuestro personaje está tocando esa capa.
Seleccionamos el Tilemap y en la parte de Layer, seleccionamos Add Layer.
Creamos la capa con el nombre Suelo

Vamos a crear un objeto vacío dentro de nuestro personaje y lo posicionamos en sus pies. Lo llamamos CompruebaSuelo.

Salto del personaje

Creamos cuatro variables:
● bool estaEnSuelo: Nos dice si está tocando el suelo o no.
● LayerMask layerSuelo: Capa que hemos añadido al Tilemap para indicarle que parte de la escena es el suelo.
● float radioEsferaTocaSuelo: Radio de la esfera que creamos a los pies del personaje para que detecte si toca el suelo o no.
● Transform compruebaSuelo: Posición se esa esfera.

Modificamos el método OnJump para comprobar que estamos en el suelo antes de saltar.

Salto del personaje

Para comprobar que estamos en el suelo. Podríamos ponerlo en el método Update pero el FixedUpdate se ejecuta antes. De esta forma nos aseguramos que comprueba correctamente si está en el suelo.

Movimiento de la cámara

Vamos a modificar el comportamiento de la cámara para que siga a nuestro personaje. Creamos un nuevo script que asignaremos a la cámara, y la idea es ir modificando la propiedad position y asignarle los valores de X e Y que tenga nuestro personaje.

Recolectar elementos

añadimos un objeto para recolectar. Le añadimos una etiqueta para identificarlo correctamente y un Collider para detectar cuando colisiona con el personaje.
Cuando el personaje colisione con el objeto, destruimos el objeto. En el Collider marcamos la opción de Is Trigger para poder pasar “por encima” de la manzana y el evento salte.

Añadimos un UI Image para añadir el dibujo de la manzana y un Text Mesh Pro para los puntos y los posicionamos en el lugar que queramos.

creamos un GameManager, donde controlaremos la suma de puntos y la actualización del texto.

Enemigo

La parte de colisionar con el enemigo y que reinicie la escena ya sabemos hacerla. Lo que tenemos que implementar aquí es el movimiento del enemigo.
Vamos a realizar un movimiento sencillo y continuo de arriba a abajo.
Para ello, podemos crear una animación con dicho movimiento y asignarla a nuestro enemigo.

Enemigo

Añadimos la ventana de Animation. Windows → Animation → Animation

Enemigo

Creamos una nueva carpeta animaciones y creamos la animación Enemigo. Nuestra animación va a consistir en sumar y restar valores a y para que realice un movimiento vertical, con lo que tenemos que añadir la propiedad Transform Position, que es donde tengo esos valores.

Enemigo

Pulsamos el botón de grabar y creamos el movimiento

Animaciones

La primera animación que vamos a crear va a ser la de Idle, cuando nuestro personaje está en espera.
Seleccionamos a nuestro objeto jugador y en la pestaña de Animation pulsamos el botón de Create para crear la animación. Nos pedirá una ruta donde almacenar el fichero que se genera así que creamos una carpeta animaciones dentro de los Assets de nuestro proyecto. Llamamos a la animación Idle y se crean dos ficheros, la propia animación y un animation controller asociado al jugador.
Usaremos ese controller para gestionar todas las animaciones del jugador

Animaciones

Seleccionamos los sprites que van a formar parte de la animación Idle y los arrastramos a la pestaña de Animation.

Animaciones

Para realizar la animación de correr o de salto pulsamos sobre Create New Clip en la pestaña de Animation y repetimos el proceso.

Animaciones

Una vez creadas las animaciones tenemos que configurar cuando se va a mostrar cada una. Para ello hacemos doble click sobre el elemento Jugador Controller y se abrirá una máquina de estados con las diferentes animaciones en la pestaña de Animator.

Animaciones

Vamos a crear una nueva transición desde el estado de Idle al estado Correr. Esto sucederá cuando pulsemos las teclas para mover al personaje.
Botón derecho sobre el estado Idle y seleccionamos Make Transition y lo conectaremos con el estado Correr.
También creamos la transición inversa, ya que desde correr tendré que ir a Idle en caso de que dejemos de mover al personaje.

Animaciones

Ya tenemos la transición hecha, nos queda indicar al controlador cuando debe realizar esta transición. Para eso vamos a usar parámetros.
Los parámetros son como variables donde le vamos a indicar si está pasando la acción que queremos y así poder ejecutar la transición.

Animaciones

Tenemos que configurar la condición para que se realice la transición entre estados. Pulsamos sobre la flecha del cambio de estado de Idle a Correr y le indicamos que debe activarse esa transición cuando el parámetro estaCorriendo sea true.

Animaciones

Por último, tenemos que modificar el parámetro estaCorriendo desde el script del jugador para indicar cuando está en espera y cuando corriendo.
Declaramos una variable privada tipo Animator para poder acceder a las animaciones y la inicializamos desde el método Start.

Animaciones

Si no detectamos movimiento en X el valor del parámetro estaCorriendo debe ser false.

Animaciones

Si detectamos movimiento en X el valor del parámetro estaCorriendo debe ser true.

Música y sonidos

Música y sonidos

Música y sonidos

Creamos una carpeta dentro de Assets para organizar nuestros ficheros de audio.
Para agregar una música de fondo, creamos un objeto vacío y le añadimos el componente AudioSource. También podemos arrastrar el fichero de música que queremos que suene en la pestaña de Inspector del proyecto.

Música y sonidos

Para añadir efectos de sonido a algún objeto, cuando recolectamos elementos, o tocamos algún enemigo, etc. tenemos que añadir un componente AudioSource al objeto.
En el script, declaramos las variables del audiosource que he añadido y los sonidos que quiero usar.

Música y sonidos

Tengo que localizar en el código dónde deben reproducirse los sonidos. Lo usaremos por ejemplo cuando recolectemos una manzana.

Música y sonidos

En el componente Script, tendré que inicializar las variables con los sonidos que deben reproducirse.

Comprueba Sueldo Personaje Position -3.5 y -3.41 z 0
rotation todo 0
scale todo 0.5

recolectar (para los puntos)

Sprite Renderer

Capsule Collider 2D
Is Trigger

GameManger Script Game Manger dentro del componente Scriptr se añade el Texto Numero Puntos

Crear Canvas para poner el texto de puntos
Puntos objeto dentro de Canvas

crear el objeto event system en GamObject > UI > Event System

Enemigo

Sprite Renderer
Animator Controller Enemigo hay que crear la animacion para el salto en animation
Capsule COllider is trigger
Size x 0.32 y 0.32

recolectar
Sprite renderer
capsule collider 2d Is trigger size 0.64 0.64

Adornos
Objeto START ponerlo al comienzo del juego

Audio
Audio Source poner en Audio Resource el sonido del juego

Zona Muerte cuando el personaje cae al vacio tiene que morir y reiniciar
Tag Zona Muerte
Box COllider 2d

Escena
Menu Inicio

hay que crear el Canvas en canvas se pone el Script para el Menu para el Panel
texto PONER EL NOMBRE DE MENU O titulo
Boton Jugar en On Click hay que poner el objeto Canvas y seleccionar la opcion para Jugar de acuerdo a los metodos del script
el objeto de texto se escribe Jugar por ejemplo
Boton Salir Boton CREDITOS
EVENT SYSTEM

Escena Victoria
crear un canvas

Con una Panel
y un texto con la palabra Victoria
Event System
se crea un objeto controladorVictoria con el script para volver al menu

Escena Credito

Canvas SCRIPT boton Menu
Panel
Texto se ponen todos los datos autor, curso, etc
boton Menu
EVENT SYSTEM

En Build Profiles
Scene List hay que poner el orden de las Escenas.

Funcionamiento de Animator hay que unir El personaje con Salto y Salto con el Personaje tambien el personaje con Correr y correr con Personaje

donde poner Blend
hay que crear esta Corriendo
estaSaltando