# Curso Desarrollo de Juegos Unity 2D Equipo 5

## Descarga e instalación de Unity

Para instalar Unity es necesario bajar el software en la página oficial https://unity.com/download. Dentro de esta descarga se encuentra Unity Hub (descargar versión recomendada), esto sirve cómo un cliente e interfaz que permite empezar proyectos. Unity Hub te indica el proceso de instalación para utilizar la plataforma.

## Conoce la Herramienta

Para crear un proyecto podemos abrir Unity Hub donde existe un apartado arriba a la derecha [Nuevo Proyecto]. Le damos click a este apartado y se nos presentarán opciones fundamentales para el proyecto.

Para este proyecto, le debemos de dar click en la opción 2D. Dentro de este apartado en la parte inferior podemos nombrar el proyecto. El nombre del proyecto puede ser elaborado a tus gustos.

Al momento de crear el proyecto. Podremos entrar a la interfaz principal donde es posible observar todas las herramientas que nos ofrece Unity. Aquí se mencionan los apartados principales.

**Scene** - Esta es tu escena principal donde se simula visualmente los procesos que están involucrados en el desarrollo del juego. Aquí podemos modificar propiedades que se ubican en el [Inspector] cómo la posición, tamaño, rotación. Visualizar colisiones, physics, y más. En la escena también podemos arrastrar archivos que se ubican en la sección de [Project], estos usualmente son texturas y materiales, audios, sprites, entre otros.

**Project** - En esta sección puedes ubicar todos los archivos que estructuran el proyecto. Aquí también puedes visualizar diferentes escenas y crear nuevos objetos que nos permitirán construir nuestro videojuego.

**Console** - La función dentro de la consola es procesar información para ayudar con el debugging (Procesamientos al correr el juego) y se utiliza principalmente para identificar errores u otros mensajes que se necesiten para mejorar el sistema.

**Hierarchy** - Aquí se encuentran todos los objetos que se prestan dentro de la escena. Para generar un nuevo objeto podemos darle click al símbolo de [+] > [Add new Empty Object] para implementar un nuevo objeto vacío, este objeto no tendrá un aspecto visual debido a que no lo hemos implementado nodos hijos que representan el contenido del objeto. Este objeto lo podemos manipular en la misma escena o dentro del inspector.

**Inspector** - El Inspector es un pilar fundamental, aquí podemos generar componentes y propiedades. Por ejemplo, si tenemos un objeto dentro de la escena se le puede dar click a [Add component] en el inspector. Aquí nos ofrecen una gran cantidad de propiedades ya hechas por el motor de Unity, cómo el [RigidBody 2D] la cual nos permite modificar variables públicas que afectan la física del juego y el [Box Collider 2D] que le agrega colisiones al objeto (Cuando 2 diferentes Rigid Bodies intersectan las distancias de ambas figuras de collision disminuyen) estos componentes también pueden ser maleables dentro de la escena.

## Importacion de Assets

Para el proyecto al ejemplar se utilizan los assets PIXEL ART TOP DOWN 2D. Los assets son datos y contenido que incluyen archivos, modelos, audio, o imágenes que puedes implementar en tu proyecto de Unity.

Para importar los assets, podemos dirigirnos a la liga https://assetstore.unity.com/. Es importante crear una cuenta de Unity para utilizar los recursos que ofrece el asset store.

La cuenta se puede crear en la página oficial [Create Unity ID] en la parte superior. Para instalar el asset solo buscamos [Search for Assets] PIXEL ART TOP DOWN donde saldrá una opción gratuita [Add to My Assets].

Al momento de dar click surgirá una notificación para agregar los recursos al proyecto. La cual se presentará la opción de abrir la instalación del Asset en Unity. Finalmente dentro del motor Unity nos dirigimos a [Window] > [Package Manager] > [Download] > [Import] y tendremos los Assets dentro de la sección [Project].

## Sprites

Los sprites son imágenes que representan los assets del juego. Estas imágenes pueden variar dependiendo de los que el usuario quiera crear. Los sprites pueden visualizarse cómo el jugador, enemigos, proyectiles, tiles u otras imágenes que también pueden ser animadas con la herramienta [Animator]. No obstante, es necesario programar las funciones de las animaciones cuyas imágenes necesitan para actuar en su entorno.

PIXEL ART TOP DOWN Incluye los sprites del ambiente y del jugador. Sin embargo, el resultado no es ideal para desarrollar un juego funcional, tal que debemos crear nuestros propios sprites para agregarle más inmersión al Juego.

Unity siempre utiliza una textura al crear un nuevo sprite. Esta textura se aplica en el objeto y lo hace visible. Esto también causa que tenga propiedades relacionadas a la que puedes observar en el proyecto.

Para crear un sprite puedes arrastrar una nueva imagen (la que el usuario desee) en el apartado [Hierarchy] en la escena. Esto ocasiona que se cree un nuevo objeto con la textura tal que nos permite observar la imagen.

Para el ejemplo en este curso, se desarrollaron unos sprites para crear un tutorial y enemigos dentro de la escena. Al correr el juego se observan varias características agregadas en la interfaz para guiar al jugador. También se desarrollaron funciones para cambiar el aspecto de los sprites, ya sea con un cambio de color, o animaciones.

## Animaciones

Las animaciones en Unity son importantes para darle vida a la escena. El paquete de PIXEL ART solo incluye las animaciones default para el PF Player. La cual está dividida en 4 imágenes creando un sprite con 4 separaciones. Estas separaciones son imágenes que forman una secuencia los cuales puedes dividir y correr en una sucesión. Esto causa que el espectador capte una animación e intérprete un movimiento de las imágenes.

Para crear una animación podemos dirigirnos a [Window] > [Animation] > [Animation]. Para crear un nuevo clip con los sprites hacemos un click en [Create] dentro del apartado de [Animation]. Esto provoca que se cree un controlador para la animación. Tal que la podemos nombrar dependiendo del sprite que queremos animar. En el controlador se encuentra un diagrama de estados, cuyos estados representan el momento en que se corre la animación. Esto se mencionara en el subtema de **Scripts**.

Para incluir la animación el objeto del [Hierarchy] o Proyecto debe tener un [Animator Component] que estará en el controlador del asset asignado. Esto se puede representar en el apartado [Inspector] al modificar el Objeto, donde podemos asignar las imágenes del sprite dentro de la línea del tiempo. Para crear la secuencia podemos simplemente arrastrar las imágenes(Sprites) en los marcos que queramos para crear nuestra animación.

## 2D Physics

Los juegos deben de tener física para que los objetos se comporten como si estuvieran en el mundo real. Esto es un detalle muy importante para los juegos y simulaciones que se presentan en Unity. Unity nos ofrece propiedades para implementar 2D physics en un objeto. Sin embargo no siempre son exactas, tal que debemos de tomar en cuenta la creación de nuevos **Scripts** que permitan realizar esta tarea con más exactitud. En un ambiente 2D existen los siguientes componentes que podemos utilizar para recrear física en Unity.

**Rigidbody 2D -** Define cómo el Objeto y sus nodos hijos se comportan respecto a la posición, rotación, o tamaño dentro de la escena. Cuando los valores cambian, esto actualiza los componentes cómo los vectores de colisión. Al agregar un rigidbody a un sprite, causará que se mueva de manera convincente.

**Collider 2D -** Agrega colisiones, las cuales pueden ser invisibles. Estos se pueden adaptar al [Mesh] o puedes manipularlas por tu propia cuenta.

**Physics Material 2D** - Sirve para añadir un coeficiente de Fricción, normalmente es preferible tener un rigidbody antes de aplicar esta propiedad para que funcione a su totalidad.

**2D Joints -** Existen diferentes tipos de Joints. Sin embargo, este componente se utiliza para juntar objetos dentro del [Hierarchy] o la [Escena].

**Constant Force 2D -** Puede servir para aplicar fuerzas lineales y angulares (Fuerzas constantes). Por ejemplo, en caso de que se quiera desarrollar un proyectil que varía de velocidad al acelerar con el tiempo. El componente de Constant Force puede implementar tal acción.

**Effectors 2D -** Effectors 2D tiene varios usos. Sin embargo, el componente se utiliza principalmente para elaborar efectos de repulsión o atracción dado un punto fuente.

## Scripts

Para el juego en el ejemplo, se hicieron varios Scripts para darle funcionalidad al ambiente. Los scripts se conocen cómo parte del código y la programación, es una manera de escribir instrucciones para la ejecución del programa. Para desarrollar un Script en Unity es importante tener conocimientos del Lenguaje C sharp para poder crear un funcionamiento único. Estos scripts se pueden crear en la sección de [Add component] dentro del [Inspector]. Sin embargo, aquí se mencionan los scripts que se hicieron para el juego.

**Camera Follow -** Crea un offset para que la Cámara siga al Jugador (Target Position)

**Layer Trigger -** Consigue el Objeto del Juego para cambiar su respectiva Layer dependiendo de Del layer en que se encuentra.

**Sprite Color Animation -** Collision Trigger (Ocurre cuando el objeto interactúa con un Collision) para cambiar el color del sprite.

**Top Down Character -** Controles con GetKeyDown para WASD y EQ para Comer, también tiene un SetBool para el [Animator] isMoving de esta manera podemos saber si el jugador se está moviendo con una variable dir si la magnitud es positiva.

**DetectionZone -** Un trigger Collision que crea un lista de objetos y agrega los objetos que tienen un tag = “Movable” (un tag se puede generar en el Inspector) en el arreglo. De esta manera los enemigos pueden detectar si el objeto está cerca.

**Enemy -** Para este script se le agregaron varias funciones. Primero se creó una variable para el RigidBody y Movespeed. De esta manera podemos añadir [AddForce] sumando la dirección hacia el objeto que proporciona el script de deteccionZone multiplicando estos valores por MoveSpeed y Time.deltaTime. De esta manera el enemigo se puede mover a un objeto. En este script también se encuentra la vida/derrota del enemigo y funciones para llamar sus animaciones.

**PropsAltar -** Props altar también utiliza un collision trigger para confirmar que un objeto esta en la plataforma, tambien cambia el sprite de color y llama el objeto de ScoreManager para sumarle un valor al contador del interfaz.

**ScoreManager -** Aquí se cambia el textmesh cada vez que se entra el trigger Collision de PropsAltar. Esto se hace para cambiar el score a Máximo 1 por cada altar.

**TutorialManager -** Existen varios procesos dentro de este script. No obstante, este scripts se enfoca en hacer una lista de objetos que se crean dentro de la escena y corre cada uno de estos objetos de acuerdo a la condición que se le da.

**BackToMenu -** Sirve para sumar un valor al índice que se ubican en la lista de escenas, de esta manera se puede regresar al menú principal.

**EndGame -** Termina el juego y nos lleva a la última escena, esto se logra con una condición que se ejecuta con un collision trigger, ya sea obteniendo un score total de 3 y que el jugador esté cerca a al área de colisión.

**MainMenu -** Parecido a BackToMenu, sin embargo la suma del índice nos lleva a Juego en lugar de traernos a la escena principal.

**SwordAttack -** Script para agregar la función de comer enemigos. Checamos si podemos atacar a la derecha o a la izquierda usando vectores. También obtenemos el game Component del enemigo para poder hacer daño y finalmente restarle vida.