<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

Unity
Clase 2

Motores en videojuegos

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 1

Introducción

- Historia de los videojuegos
- Diferencias entre generaciones de consolas
- Ejemplos de juegos de diferentes géneros y sus mecánicas recurrentes.

Clase 2

Motor Gráfico y Físico - Instalación de Unity

- Motor gráfico y físico, sus herramientas, instalación y puesta en marcha de Unity (Unity Hub - Unity ID - Visual Studio).

Clase 3

GDD

- Documentación: GDD y ejemplos.

Motores en Videojuegos

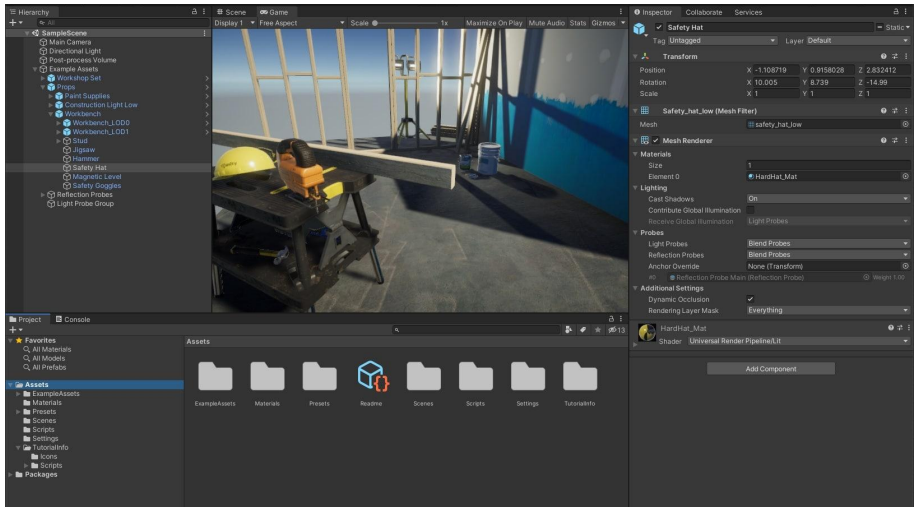
Fuentes: https://blogthinkbig.com

Motor Gráfico

Un **motor gráfico** es **un software usado por aplicaciones y programas para dibujar gráficos en la pantalla** de nuestro ordenador, smartphone o tablet.

Cuando pensamos en la palabra **motor**, pensamos en máquinas de producción mecánicas, como un motor de combustión. De hecho, los primeros ordenadores se diferenciaban entre los motores que lo constituían.

Desde hace unos años, **la palabra motor** en el mundo del software se usa de manera diferente, refiriéndonos al **software que ejecuta un determinado tipo de tareas comunes a muchas aplicaciones** de software: un *motor de base de datos*, un *motor de transcripción texto a escrito*, o un *motor gráfico*.



Motor Gráfico

Más concretamente, **se define como motor gráfico al framework de software diseñado para crear y desarrollar videojuegos.**

Los desarrolladores de videojuegos pueden usar los motores para crear videojuegos para tu consola, dispositivos móviles u ordenadores.

Todo motor gráfico ha de ofrecer al programador una funcionalidad básica, proporcionando normalmente:

- **Un motor de renderizado** (“render”) para gráficos 2D y 3D.
- **Un motor de interacciones físicas**, que detecte la colisión física de objetos y la respuesta a dicha colisión.
- **Sonidos y música.**
- **Animación.**
- **Inteligencia artificial.**
- **Comunicación con la red** para juegos multijugador.
- Posibilidad de **ejecución en hilos**.
- **Gestión de memoria o soporte para localización** (traducción de los textos y audios del juego según idioma).

Motor Gráfico

Las **capacidades gráficas** de motor gráfico **son una de las claves para su elección**, destacando motores gráficos como *CryEngine*.
Pero también son importantes:

- **La facilidad de desarrollo**
- **La plataforma para la que se va a desarrollar.**

Describir todas las funciones de un motor gráfico llevaría miles y miles de palabras, pero en esencia, un motor gráfico está ahí para que los desarrolladores no tengan que reinventar la rueda y se puedan centrar en lo importante: **su juego**.

Los desarrolladores de juegos no necesitan convertir sus modelos 3D a formatos crípticos para importarlos al juego, eso lo hará mejor el motor gráfico que ha sido desarrollado por un estudio con un equipo talentoso y grandes cantidades de recursos a lo largo de muchos años.

Un buen motor gráfico es el que traslada tus ideas creativas fácilmente a gráficos en una pantalla. Esto, combinado con efectos de postprocesado, creación de terrenos y construcciones y efectos de partículas, hará que el desarrollador pueda crear un mundo dentro de juego combinando las capacidades del motor con los modelos 3D que hayan podido crear los artistas responsables.

Motores de Juego

Unreal Engine

Es ampliamente considerado uno de los mejores game engine en general, principalmente debido a **los gráficos** que puede ofrecer y la **amplia gama de opciones de personalización disponibles**.

Algunos de los títulos más populares que se han construido en Unreal Engine son: *Fortnite, la serie Borderlands, Rocket League, la serie Gears of War, Bioshock, etc.*

Sin embargo, Unreal Engine no es ideal para todos los desarrolladores de juegos. En primer lugar, no es tan fácil de aprender como algunas de las otras opciones de game engines que existen y, si no está buscando construir juegos 3D robustos, probablemente sea mejor comenzar con un motor más simple (especialmente si estamos buscando construir juegos móviles.)

Unity

Si bien Unity y Unreal Engine a menudo se consideran los dos motores principales de juegos, **ambos motores tienen diferentes propósitos**. Unreal Engine es el más adecuado para juegos más robustos, **Unity es más versátil y puede ser una mejor opción para los desarrolladores que buscan crear juegos móviles, juegos 2D o juegos 3D story-driven.**

Para tener una idea de cuán versátil es Unity, aquí hay una muestra de algunos de los mejores juegos hechos con Unity: *City: Skylines, Kerbal Space Program, Hearthstone, Escape from Tarkov, la serie The Room, etc.*

Unity ofrece toneladas de recursos de aprendizaje gratuitos a través de Unity Learn, y el motor es lo suficientemente intuitivo para que los principiantes puedan comenzar con una curva de aprendizaje menos pronunciada.

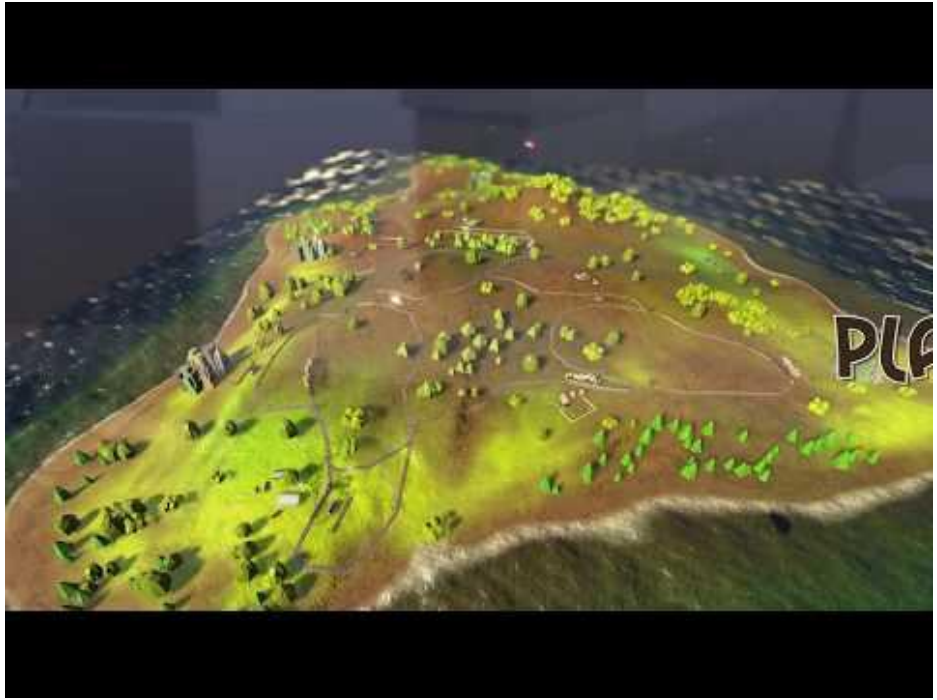
GameMaker Studio

Resulta **fácil de aprender y no requiere experiencia previa en programación.** En lugar de tener que escribir scripts o líneas de código, GameMaker le permite "arrastrar y soltar".

Por supuesto, *la naturaleza fácil de GameMaker Studio lo limita en el tipo de juegos que puedes hacer*. GameMaker admite el desarrollo de juegos en 3D, pero realmente es más adecuado para construir juegos en 2D. Algunos ejemplos de creaciones con GameMaker Studio son: *Spelunky, Undertale, Hyper Light Drifter y Hotline Miami*.

Sin embargo, a medida que el desarrollo de juegos independientes se ha vuelto más popular y debido a que hay un gran mercado para los juegos de estilo RPG 2D, GameMaker es un motor que vale la pena considerar para ciertos desarrolladores.

Godot



Otra opción si se desea construir juegos de estilo 2D o juegos 3D simples es Godot.

Godot no ha existido tanto tiempo como algunos de los otros motores en esta lista y realmente no ha habido ningún juego súper exitoso hecho con el motor.

Este motor es:

- **Código abierto**
- **Completamente gratuito**
- **Liviano**
- Tiene una **comunidad sólida** detrás
- Ofrece una **tonelada de herramientas** para desarrolladores
- Es compatible con **múltiples plataformas.**

CryEngine

Al igual que Unreal Engine, CryEngine es otro **motor de juegos creado para desarrollar juegos visualmente impresionantes.**

Como testimonio del potencial de CryEngine, aquí hay una lista de algunos de los juegos más populares construidos con él: la serie *Far Cry, la serie Crysis, Kingdom Come: Deliverance y Sniper Ghost Warrior*.

CryEngine es un motor de juego creado por la empresa Crytek, **originalmente un motor de demostración para la empresa Nvidia,** que al demostrar un gran potencial se implementa por primera vez en el videojuego *Far Cry*, desarrollado por la misma empresa creadora del motor.

Amazon Lumberyard

Amazon compró el código fuente de CryEngine y desarrolló su propio motor de juego: Amazon Lumberyard.

En otras palabras, **Lumberyard tiene el potencial de alta gama que tienen motores como Unreal Engine 4 y CryEngine,** pero con el respaldo de una gran empresa conocida por la innovación en una variedad de sectores.

Aun así, Lumberyard sufre algunas de las mismas caídas que sufre CryEngine: falta de usuarios, no muchos recursos de aprendizaje y no es tan confiable como otros motores.

Instalación de Unity

Fuente: Luis Canary Channel

Para la instalación de Unity, se requiere completar los siguientes pasos:

Instalación de Unity

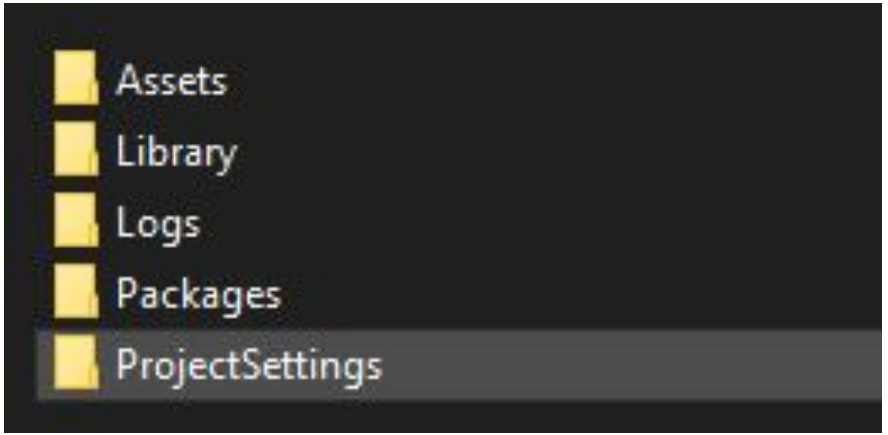
Instalación de Visual Studio

Instalación de Unity HUB

La estructura de un proyecto en Unity

Las carpetas más importantes a la hora de compartir un proyecto son **Assets, Packages y ProjectSettings.**

- **Assets** va a tener en su interior los archivos que se crean y se usan en el juego.
- **ProjectSettings** va a tener la configuración de los sistemas de Unity que se usan en el juego.
- Si instalamos algún módulo o paquete para expandir la funcionalidad de Unity, estos estarán en la carpeta **Packages**.
- Dentro de **Library** estarán los archivos importados convertidos al formato que utiliza Unity.



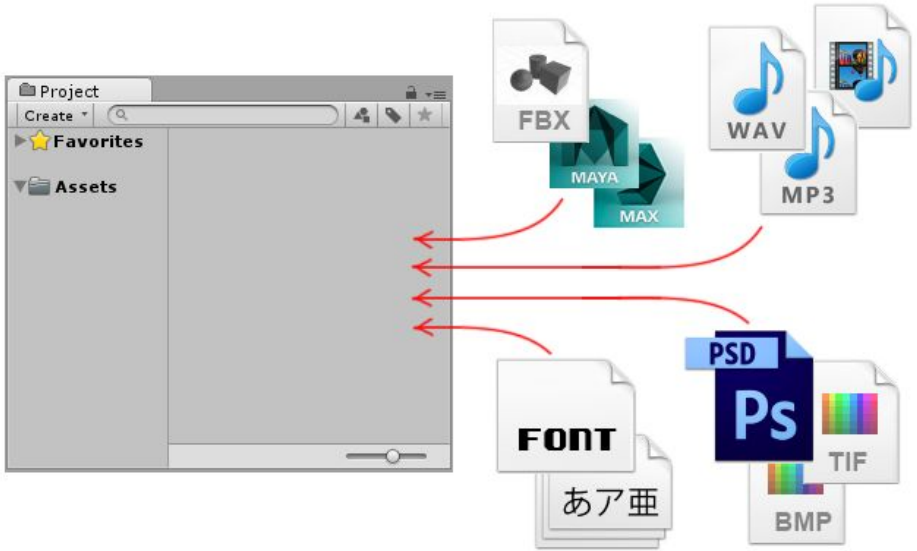
¿Qué son los Game Assets?

Fuentes: https://blogthinkbig.com

Game Assets

Casi todos los videojuegos son un paquete bastante complejo. Se debe desarrollar la historia, agregar música y sonido, y se deben diseñar e implementar los aspectos visuales del juego. Aparte de todo lo demás, desarrollar las visuales es un **GRAN TRABAJO**. Hay varios assets diferentes que los desarrolladores eligen para crear el paquete que ves cuando te sientas a jugar un juego en particular.

Uno de los primeros pasos para desarrollar un juego que coincida con la idea que tienes en mente es determinar qué assets incluir y cómo los desarrollarás.



Game Assets

Diseño 2D / 3D

- Personajes
- Objetos
- Ambientes
- Vehículos
- GUI

HUD

- Íconos
- Secuencias de comandos

AI

- Efectos especiales
- Redes
- Física

Audio:

- Música de fondo
- Efectos de sonido

Level Art

Level Art es una parte muy importante del diseño del juego.

Es fácil confundir la idea del arte de niveles con el medio ambiente, ya que parecen superponerse de alguna manera. Sin embargo, **el arte de niveles es esencialmente el lugar donde se encuentran el arte y la jugabilidad.**

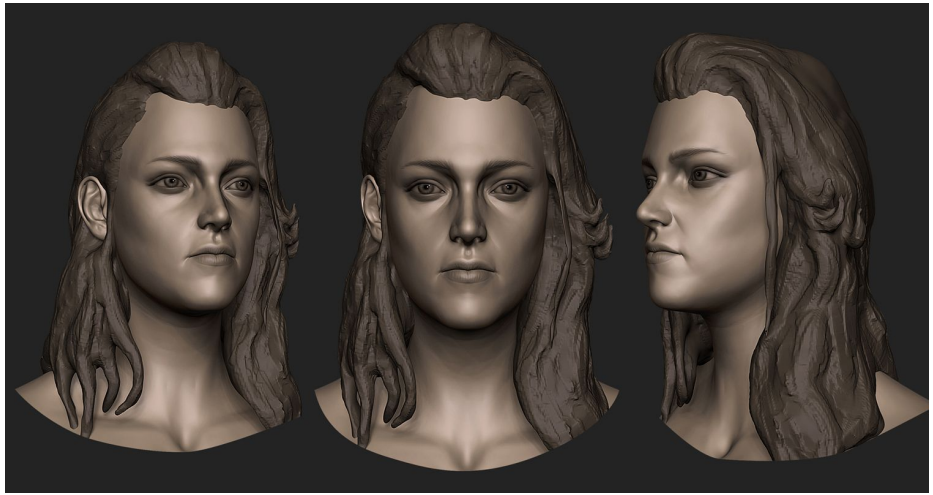
Un artista de niveles trabajará con los otros artistas para determinar dónde se ubica cada uno de estos assets en relación con el entorno, asegurándose de que el motor físico del juego no se vea comprometido.

Entorno

Un entorno bien diseñado puede crear una experiencia de juego altamente inmersiva.

Entornos simples y estáticos eran la norma hace mucho tiempo. Hoy, los entornos de juego son una extensión de los juegos mismos. Los juegos con entornos en constante cambio son muy atractivos y hacen que el jugador se sienta más como si realmente estuviera dentro del juego.

Diseño



El diseño de personajes es, por supuesto, muy importante. Sin embargo, un personaje no tiene que ser creado cuidadosamente para ser atractivo, incluso los personajes simples como los de la serie Super Mario Brothers son memorables y atractivos.

La clave más importante es asegurarse de que este asset se ajuste al entorno y a los accesorios, objetos y otras cosas del mundo del juego. En los últimos años, el recuento de polígonos dedicados a los personajes ha aumentado dramáticamente. Piensa en Lara Croft de *Tomb Raider* en 1996 en comparación con las iteraciones más recientes: la diferencia en su apariencia se debe principalmente a los juegos que admiten un mayor recuento de polígonos.

Props

Las props son cosas que no tienen vida que mejoran o cambian el gameplay.

Esto podría incluir elementos como un barril sobre el que salta el personaje, una bandera que el personaje toque al final de una carrera o las barreras que evitan que dos luchadores salgan de un ring de lucha libre.

Estas son una parte tan importante del juego que no pueden pasarse por alto.

Objetos destruibles

Los objetos destruibles han sido una parte crítica de los juegos durante muchos años.

Piensa incluso en lo que respecta a *Donkey Kong*. Había barriles que el personaje podía golpear con un martillo y explotar para reaccionar en la pantalla.

Si bien los objetos destructibles de hoy en día son a veces más creativos, son básicamente lo mismo, ¡pero con más polígonos y mejor física!

Autos, aviones y otros vehículos

Ésta es un área donde la tecnología ha tenido un gran impacto en los juegos. Los primeros juegos de carreras o juegos de vuelo fueron muy simples. Es posible que los autos anteriormente hayan podido girar de un lado a otro para evitar el tráfico que se les aproximaba, pero no daba la sensación de conducir realmente un automóvil.

Los juegos de hoy son totalmente inmersivos porque los diseñadores han descubierto cómo hacer que el vehículo se conduzca como un automóvil real (camión, avión, etc.) Considere un juego como *Gran Turismo*. La forma en que funcionan los frenos y el acelerador le permite acelerar y reducir la velocidad al igual que un vehículo real, además la dirección es más apretada y la respuesta es mucho más precisa.

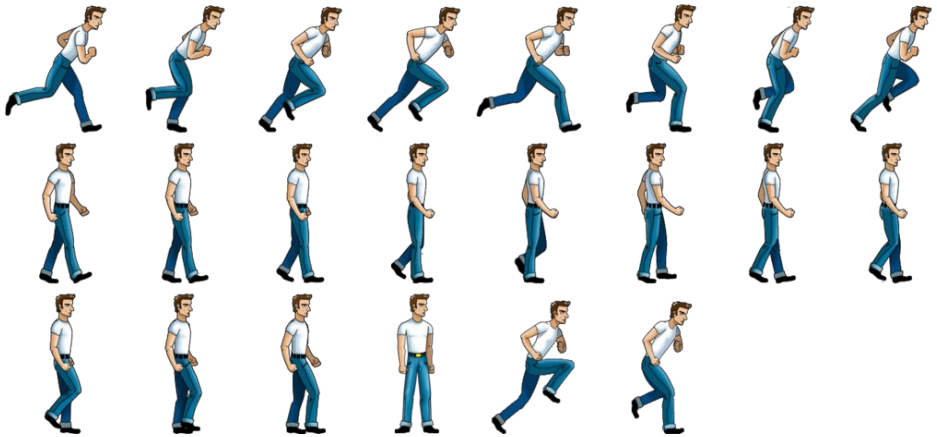
Armas

Las armas han sido parte de los juegos casi desde el principio.

Han cambiado tan dramáticamente a lo largo del tiempo, que ahora, en un juego de disparos, puedes sentir la diferencia entre una pistola, un rifle y un lanzallamas.

Además de cómo reacciona el arma, los diseños han mejorado para tener un aspecto diferente cuando están equipados o cuando se almacenan en la espalda o en una funda.

Sprites



Inicialmente, un sprite era una imagen de mapa de bits que era gestionada por un hardware especializado, independiente de la CPU de la máquina, usado generalmente para los protagonistas de los videojuegos u otros personajes, ya que permitía gestionarlos de manera independiente a los fondos.

Con el paso del tiempo el término sprite acabó por hacer referencia **a cualquier imagen de mapa de bits presente en pantalla, generalmente refiriéndose a los personajes del juego.**

En programación de videojuegos **es la máscara o imagen de un objeto que tiene la capacidad de colisión.**

Partículas

Fuente: GemukiDev

Las partículas son utilizadas para crear efectos. **Estos elementos utilizan una cantidad definida de imágenes, y mediante código, éstas son producidas con diferentes reglas.**

Por ejemplo en una explosión, salen volando reproducciones de una misma imagen para hacer parecer que ésta es muy grande.

Otro ejemplo tradicional es cuando un personaje va corriendo muy rápido y deja una estela de imágenes de él, o si un hada deja chispas brillando en el aire.

El fuego, el humo, la lluvia y muchos otros efectos son hechos de manera más eficiente utilizando partículas. Es importante recalcar que es indispensable conocer la capacidad del sistema al que apuntas, las características de tu motor gráfico y tus conocimientos, para identificar cuándo utilizarlas y cuando no.

Materiales, Shaders y Texturas

Los **materiales** definen cómo se debe representar una superficie, incluyendo referencias a las texturas que usa, información de mosaico, matices de color y más. Las opciones disponibles para un material dependen del shader que esté utilizando el material.

Los **shaders son pequeños scripts** que contienen los cálculos matemáticos y algoritmos para calcular el Color de cada píxel renderizado, basado en la entrada de iluminación y la configuración del material.

Las **texturas son imágenes de mapa de bits**. Un material puede contener referencias a texturas, de modo que el sombreador del material pueda usar las texturas mientras calcula el color de la superficie de un objeto.

Además del Color básico (Albedo) de la superficie de un objeto, las texturas pueden representar muchos otros aspectos de la superficie de un material, como su reflectividad o rugosidad.

Un **material especifica un shader específico para usar, y el shader utilizado determina qué opciones están disponibles en el material.** Un **shader especifica una o más variables de texturas** que espera usar en el objeto, y el inspector de materiales en Unity permite asignar sus propios assets de texturas a estas variables de texturas.

GUI (Graphical User Interface)

Todo lo que concierne a la interfaz del usuario que utiliza un conjunto de imágenes para representar la información y acciones disponibles.

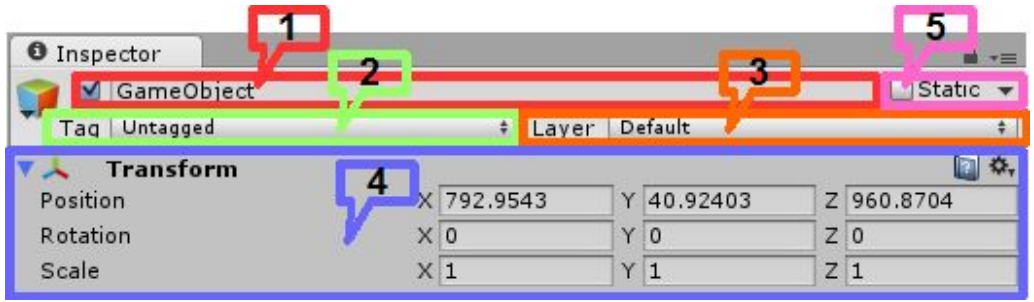
- **Menús**
  - Principales (inicio, opciones, salir, etc)
  - Extras (mapas, páginas de estadísticas de personajes, menús de rpg)
- **HUD** (Heads Up Display, es una indicación de todo lo que el jugador necesita saber: la vida, el tiempo restante, cuánto tiene de municiones restantes de un arma, su energía o cualquier otro recurso que sea necesario conocer en cualquier momento)
- **Mensajes** (errores, muerte o game over, cambio de nivel,etc)

GameObject

Son uno de los elementos indispensables en el desarrollo de videojuegos con Unity, ya que **representan cualquier objeto situado en la escena del juego.**

Todo GameObject estará formado por defecto por los siguientes elementos:

1. **Nombre** del GameObject en la escena (junto a un checkbox para habilitar o no este objeto en el juego).
2. **Tag** que permita identificar el objeto programáticamente, además de agrupar elementos que definen características similares.
3. **Layer**, que proporciona un desplegable donde se especifica la capa donde se representará el objeto.
4. Componente **Transform**, que describe la posición (coordenadas X, Y, Z), rotación (en grados alrededor del eje X, Y, Z) y escala (tamaño del objeto, que al definir valor 1, indica que es el tamaño original) del objeto en la escena.
5. Propiedad **Static (checkbox)**, que permite especificar si un objeto se moverá o no a lo largo del escenario.



GameObject

Además de los componentes y propiedades que intervienen en todo GameObject, aquí hay algunas de sus características básicas:

- Es posible desactivar un GameObject, afectando por igual a todos los elementos hijos de éste.
- El uso de un tag asignado al GameObject, permite vincular a varios objetos entre sí que implementen características similares (un ejemplo muy común son los tags «Player» o «Enemy», para identificar los personajes que maneja el usuario o los enemigos representados en el videojuego, respectivamente).
- Los componentes asignados a cada GameObject proporcionan su comportamiento y funcionalidad en la lógica del juego.

Prefab

Los **prefabs son objetos reutilizables**, y creados con una serie de características dentro de la vista proyecto, que serán instanciados en el videojuego cada vez que se estime oportuno y tantas veces como sea necesario. Es decir, nos permiten crear 'copias' fácilmente con una serie de ventajas:

- Es posible diferenciar qué objetos están conectados a un prefab desde la jerarquía de objetos (resaltados en color azul).
- Permite modificar los valores de un elemento del prefab instanciado, no afectando al resto de instancias implementadas, y sin romper la conexión con el prefab creado en los recursos del proyecto.
- Si se modifican las propiedades del prefab creado, y este es instanciado en diferentes zonas de la escena, los cambios realizados afectarán a todas las instancias implementadas, sin necesidad de tener que modificar de manera individual cada instancia.

Importando Custom Assets

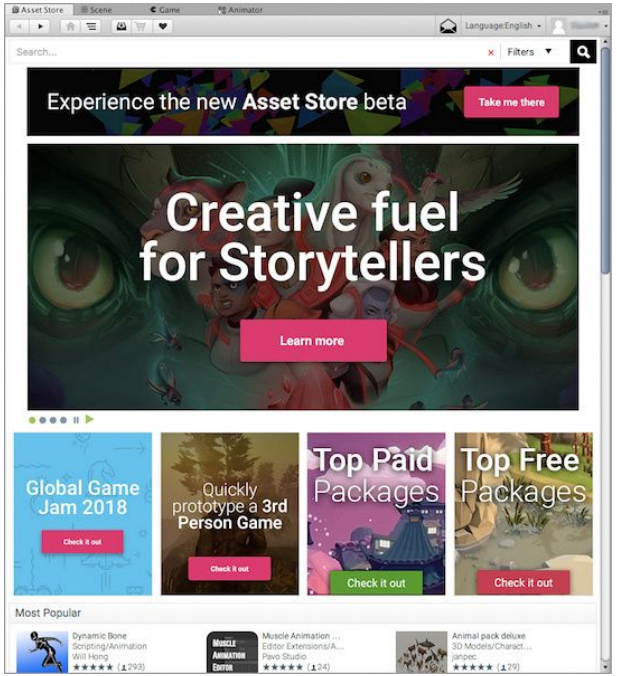
Asset Store

Unity Asset Store es el hogar de una **creciente biblioteca de assets gratuitos y comerciales creados por Unity Technologies** y también por miembros de la comunidad.

Se encuentra disponible una amplia variedad de assets, que abarcan desde **texturas, modelos y animaciones** hasta **ejemplos completos de proyectos, tutoriales y extensiones del editor**.

Se puede acceder a los assets desde una simple interfaz integrada en el Editor de Unity que permite descargar e importar assets directamente en el Proyecto.

Los usuarios de Unity pueden convertirse en editores en Asset Store y vender el contenido que han creado.



No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en los foros.

Todo en el Aula Virtual.