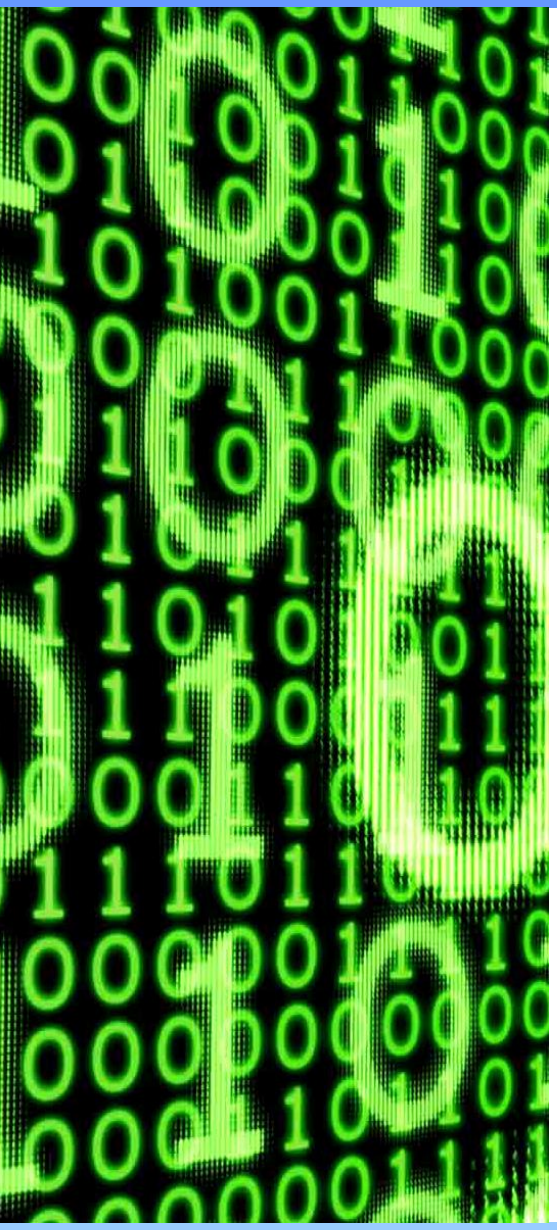


Programación de Juegos

Tema 3: Unity

Universidad de Huelva

Departamento de Tecnologías de la Información

Área de Ciencias de la Computación e Inteligencia Artificial

Objetivos:

- Conocer **uno de los principales motores de videojuegos actuales: Unity**.
- Aprender las **principales características** de Unity.
- Conocer los **principales lenguajes de programación** usados en Unity.
- Saber **crear un videojuego** en Unity.

Índice:

Índice:

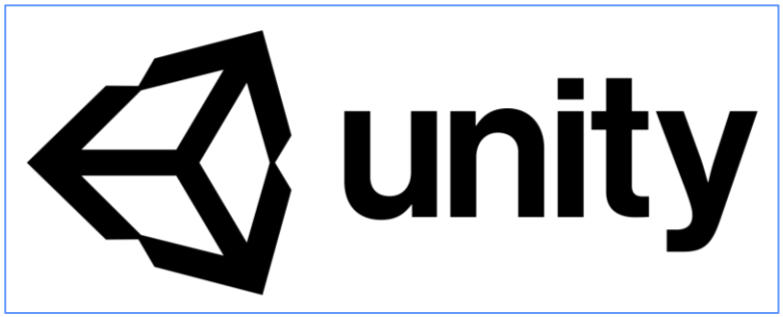
1. Introducción

- En la actualidad, **la herramienta preferida para desarrollar videojuegos son los Motores de Videojuegos**.

- Los Motores de Videojuegos nos ofrecen un entorno de desarrollo de propósito general y reutilizable que ***pueden*** abarcar:
  - Interfaz Gráfica intuitiva y amigable
  - Programación modular mediante scripts
  - Motor físico (Colisiones, gravedad, fricción…)
  - Motor gráfico (Renderizar la escena a imagen)
  - Implementación de técnicas de IA
  - Abstracción de Hardware mediante APIs como OpenGL, DirectX…

1. Introducción

- Los principales Motores de Juegos actualmente son:



1. Introducción

- Los principales Motores de Juegos actualmente son:

Unity	Unreal Engine
Posibilidad gratuita	Posibilidad gratuita
Interfaz Gráfica	Interfaz Gráfica
Abstracción Hardware	Abstracción Hardware
Material didáctico	Material didáctico
Comunidad activa	Comunidad activa
Fácil aprendizaje	**Aprendizaje costoso**
Poco optimizado y limitado	**Más optimizado y avanzado**

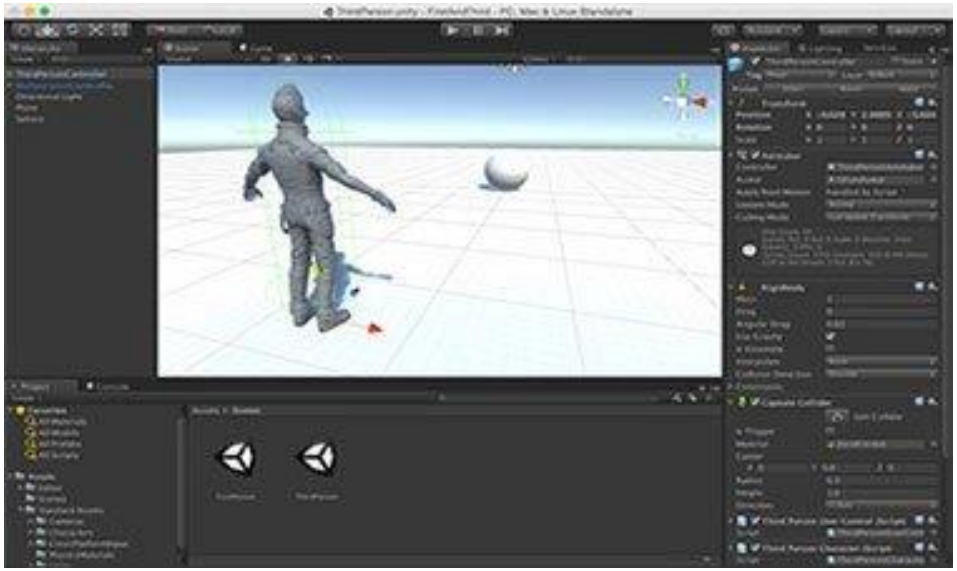
1. Introducción

- En nuestro caso, sacrificaremos la posibilidad de crear juegos con requisitos más exigentes, a cambio de poder crear juegos más fácilmente, con posibilidades muy amplias y gran portabilidad.

- Por tanto escogeremos **Unity**

1. Introducción

- **Unity** es un Motor de Videojuegos (game engine), es decir, una herramienta que nos permite diseñar, crear y hacer funcionar un entorno interactivo (videojuego).



MyScript.cs | 81-C# Script-NewBehaviourScript.cs.txt

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class #SCRIPTNAME# : MonoBehaviour
5  {
6      void Start()
7      {
8
9      }
10
11     void Update()
12     {
13
14     }
15 }
```

1. Introducción

- Surgió como herramienta para crear videojuegos para **Mac** en **2005**

- Gracias a su versatilidad, obtuvo el reconocimiento para crecer y convertirse en lo que es hoy: Un potente Motor de Videojuegos multiplataforma, tanto para grandes empresas como para usuarios ocasionales.

1. Introducción

- Versiones
  - Unity **3**.X (2011 – 2012): Partículas, Pathfinding, Web client…
  - Unity **4**.X (2012 – 2016): Mecanim (animaciones), Sombreado en tiempo real, Directx 11
  - Unity **5**.X (2015 – 2019): Iluminación en tiempo real, mejoras físicas, WebGL, Vulkan, VR y AR.
  - Unity **2017**.X: Mejoras de rendimiento y corrección de errores
  - Unity **2018**.X: Mejoras de rendimiento y corrección de errores
  - Unity **2019**.X: Mejoras de rendimiento y corrección de errores
  - Unity **2020**.X: Mejoras de rendimiento y corrección de errores
  - Unity **2021**.X: Mejoras de rendimiento y corrección de errores

1. Introducción

- En cuanto a las licencias de uso:
  - Individual (o empresa pequeña):
    - Estudiante: Gratis (Con algunas ventajas)
    - Personal: Gratis (Ingresos < 100.000 USD/año)
  - Empresas:
    - Plus: 369€/año por puesto (Ingresos < 200.000 USD/año)
    - Pro: 1.656€/año por puesto (Ingresos > 200.000 USD/año)
    - Empresa: 183,33€/mes por puesto (Ingresos > 200.000 USD/año)

- Comparativa de licencias

1. Introducción

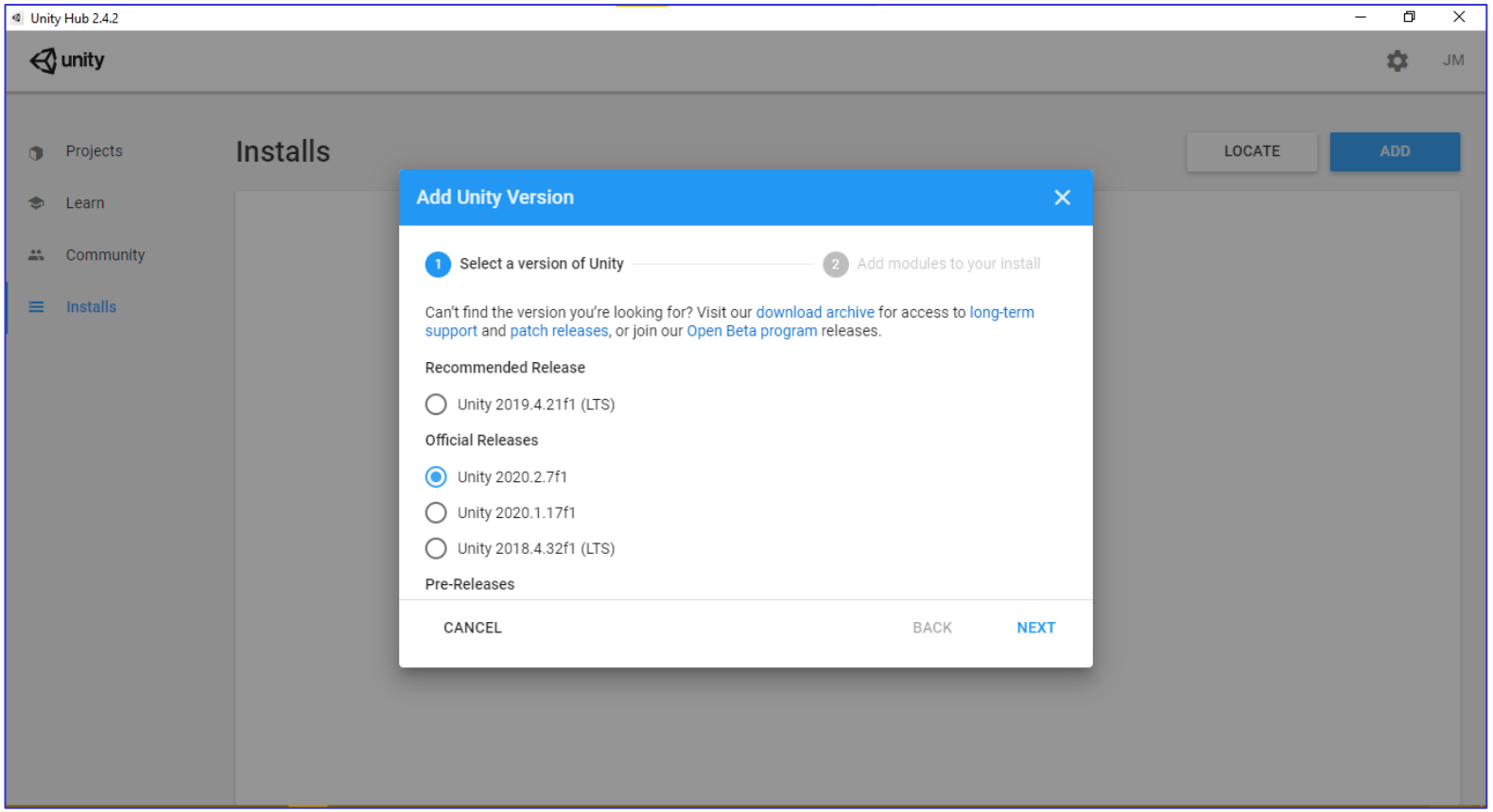
- Podemos descargarlo e instalarlo desde:
  - https://store.unity.com/download?ref=personal
  - https://unity3d.com/es/get-unity/download

- Lo recomendable es instalar Unity Hub, que es un **gestor de versiones de Unity**.

- Aunque opcionalmente, podemos escoger nosotros la versión o versiones que necesitemos, descargarlas e instalarlas directamente.

1. Introducción

- Si optamos por instalar el gestor de versiones Unity Hub.



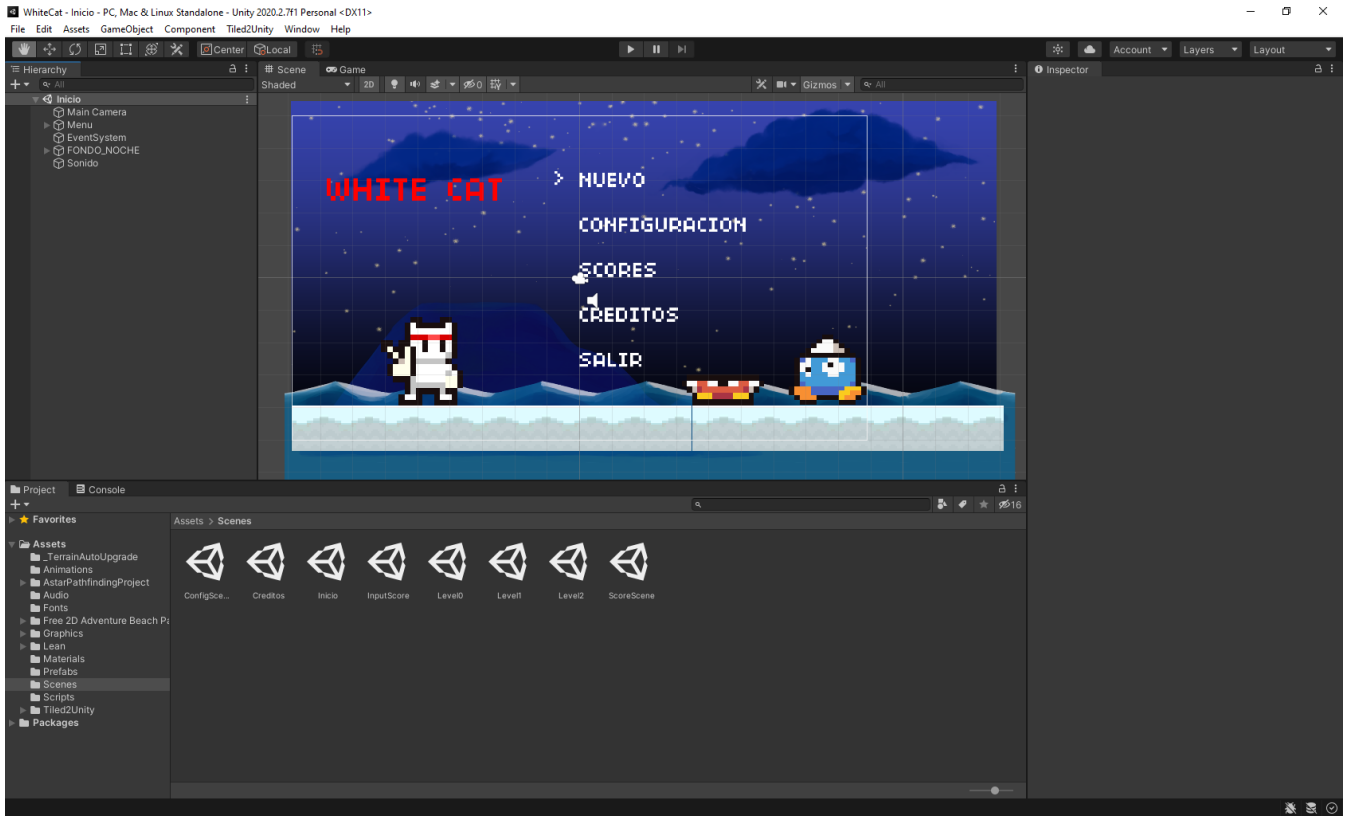
Índice:

2. Principales Características

- En este apartado entraremos en detalles sobre:
  - Interfaz gráfica
  - Motor gráfico
  - Motor físico
  - Abstracción hardware – Multiplataforma
  - Documentación oficial – Curva de aprendizaje
  - Lenguajes de programación
  - Asset Store
  - AR - VR

2. Principales Características

- La principal característica y la que nos encontraremos en primer lugar es su Interfaz Gráfica:

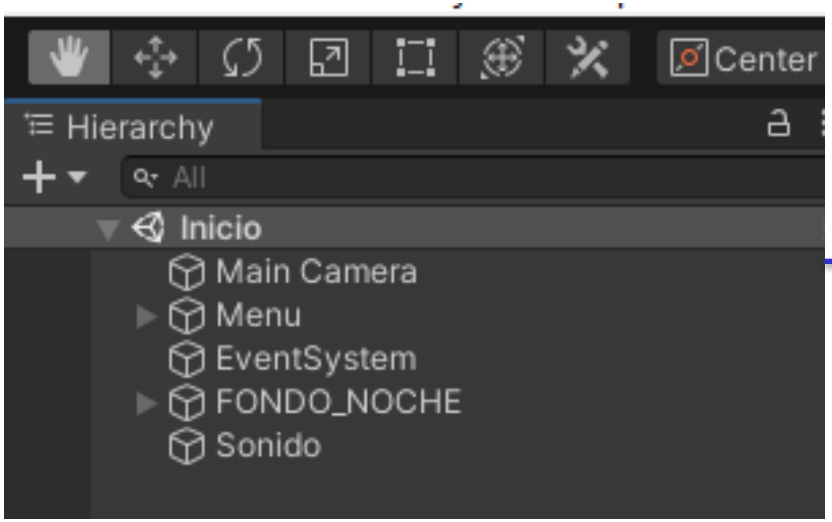


2. Principales Características

- Interfaz Gráfica: Scene

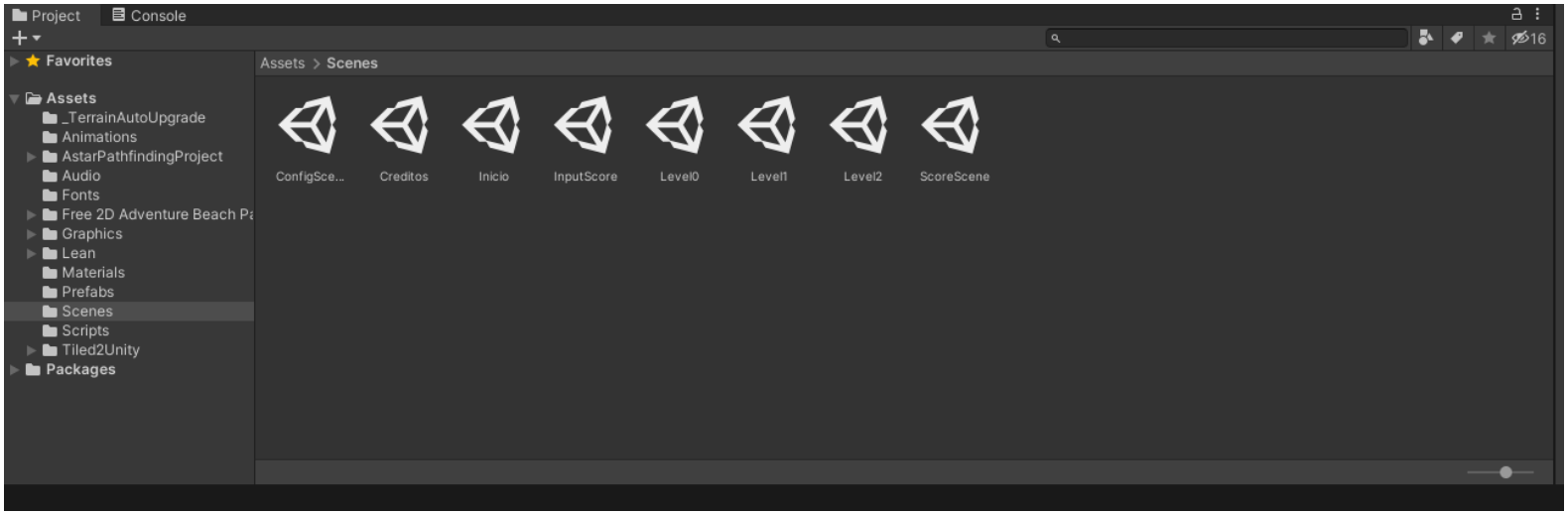
2. Principales Características

- Interfaz Gráfica: Hierarchy View



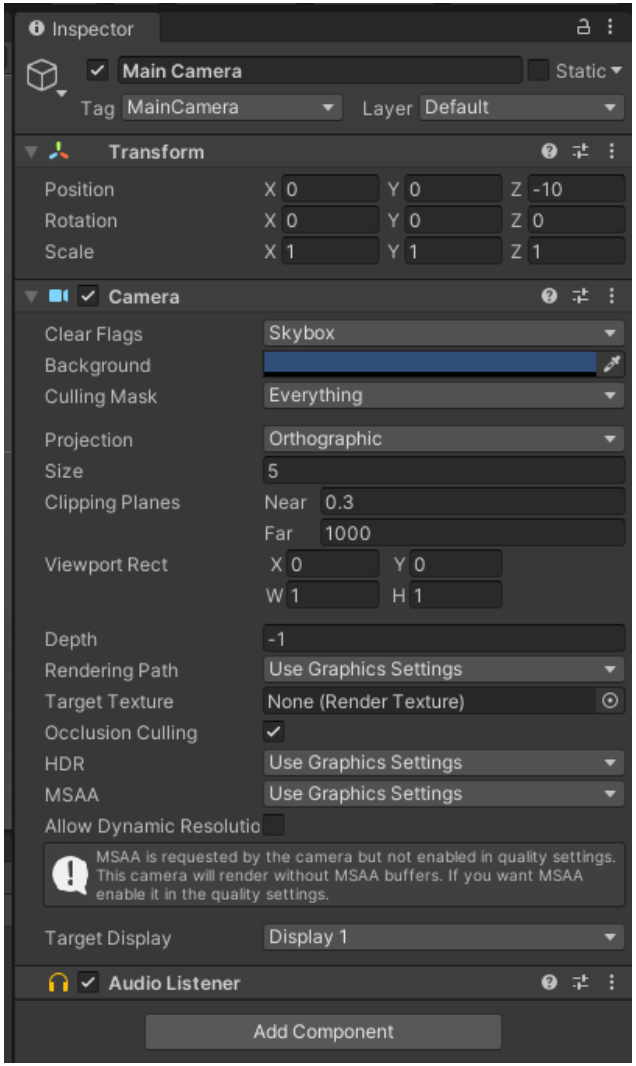
2. Principales Características

- Interfaz Gráfica: Project (Assets/Scripts/Animations/Prefabs…)



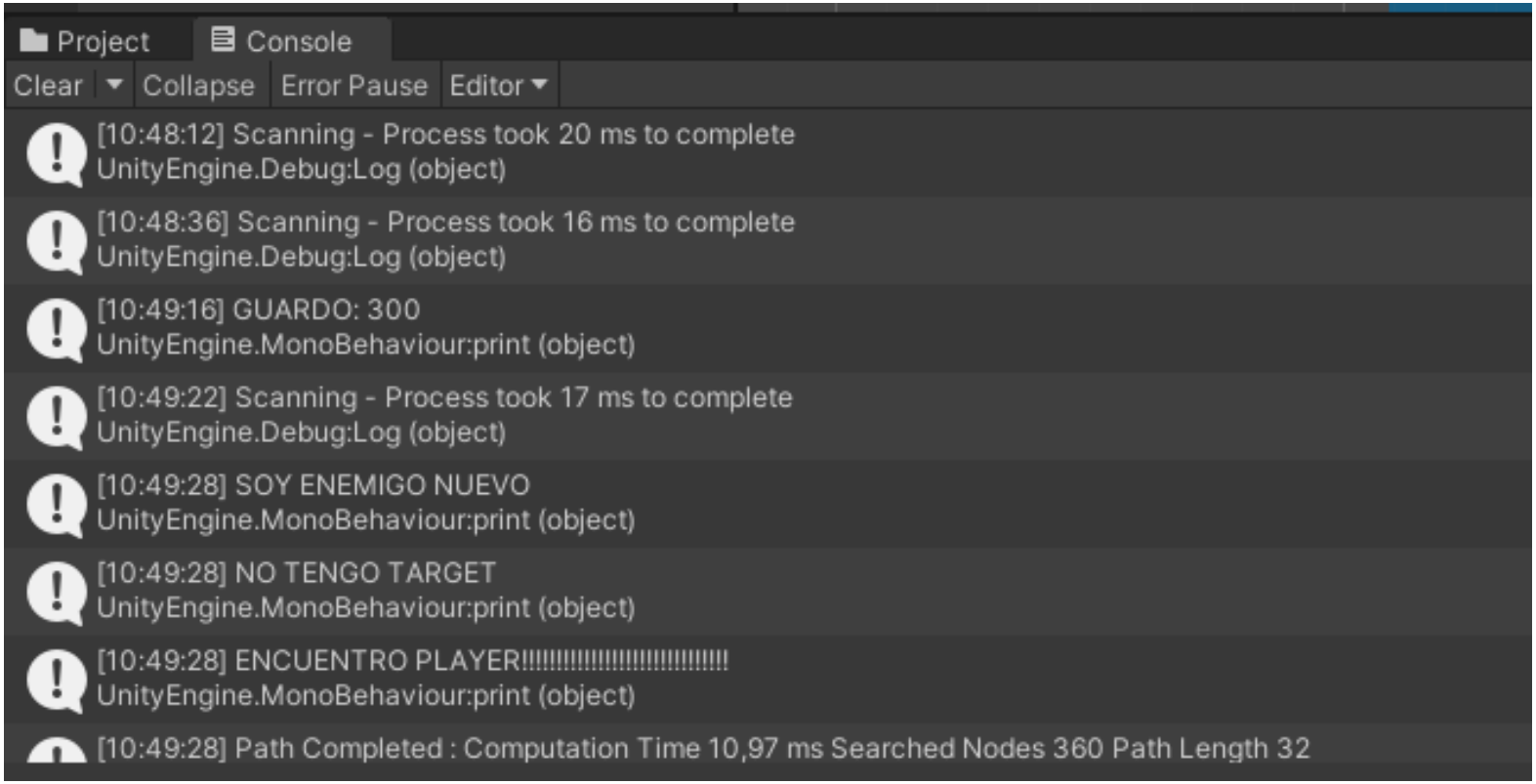
2. Principales Características

- Interfaz Gráfica: Inspector



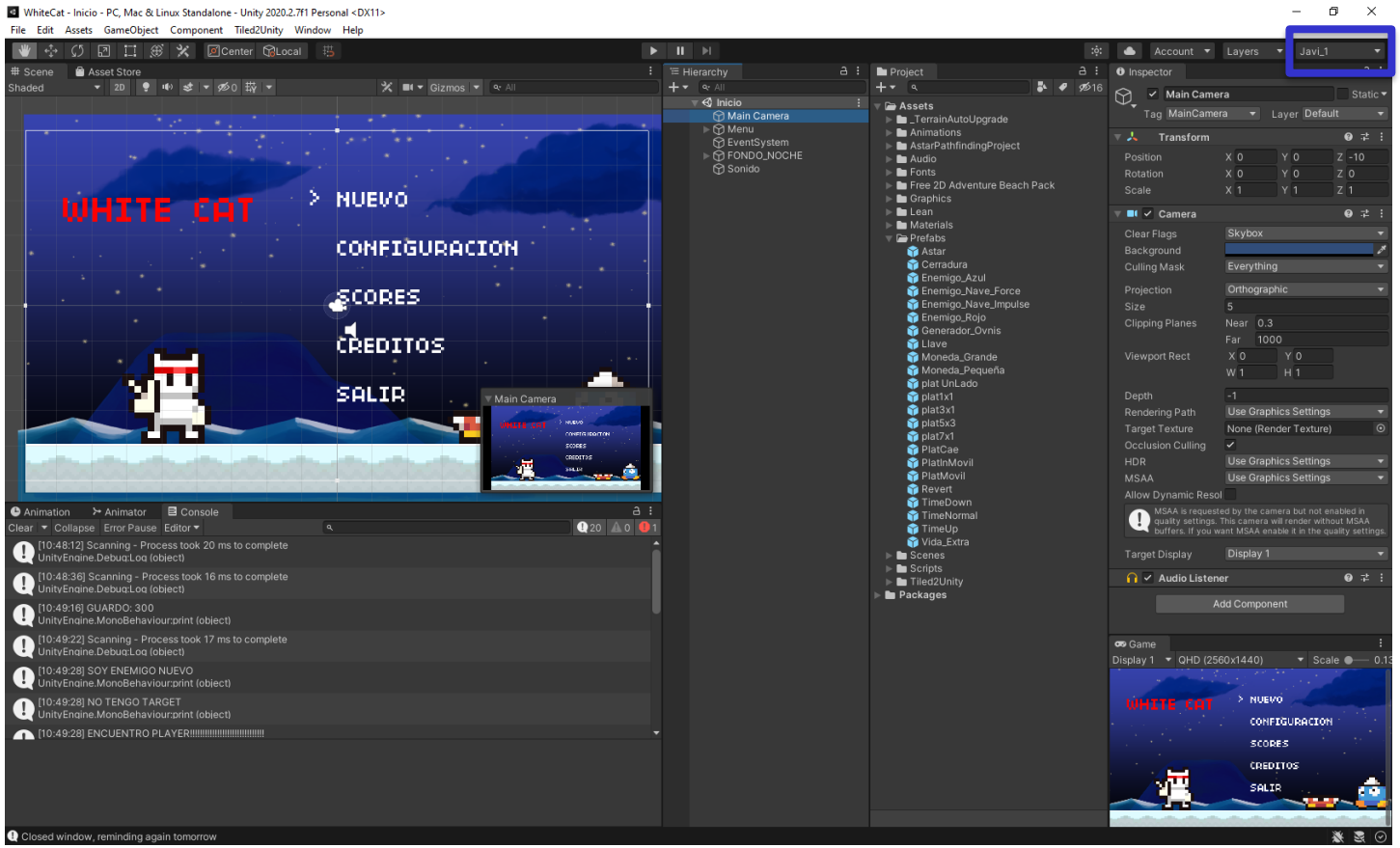
2. Principales Características

- Interfaz Gráfica: Console



2. Principales Características

- Interfaz Gráfica: Layouts

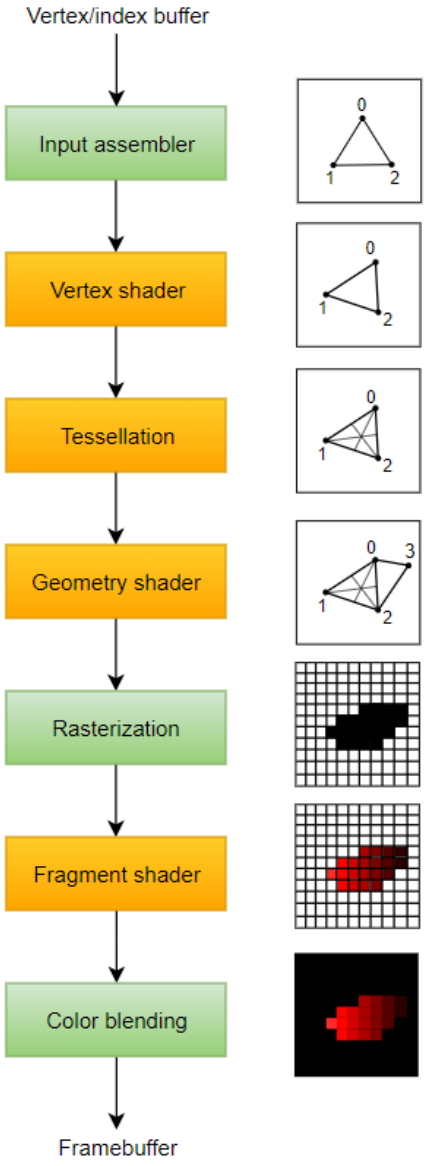


2. Principales Características

- Además, Unity cuenta con un potente Motor Gráfico.
- Nos permitirá **renderizar** la escena actual a la vista de la cámara **en tiempo real**.
- Ofrece diversos tipos de **iluminación**, para alcanzar el realismo deseado.
- **Raytracing en tiempo real**.
- **Cámaras** con perspectiva configurable.
- Diseño **2D/3D**.
- Efecto **partícula**.
- **Materiales, shaders, texturas, efectos visuales**…

2. Principales Características

- Motor Gráfico: Render Pipeline
  - **Built-in** Render Pipeline
  - **Universal** Render Pipeline
  - **High Definition** Render Pipeline
  - **Custom** Render Pipeline

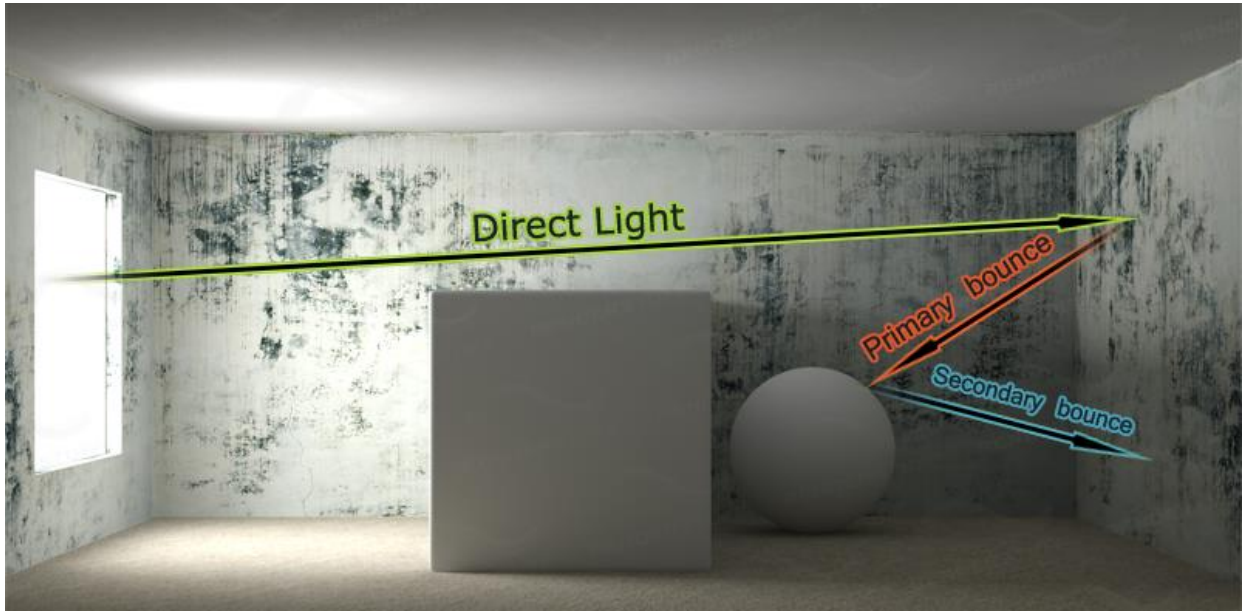


2. Principales Características

- Motor Gráfico: Iluminación

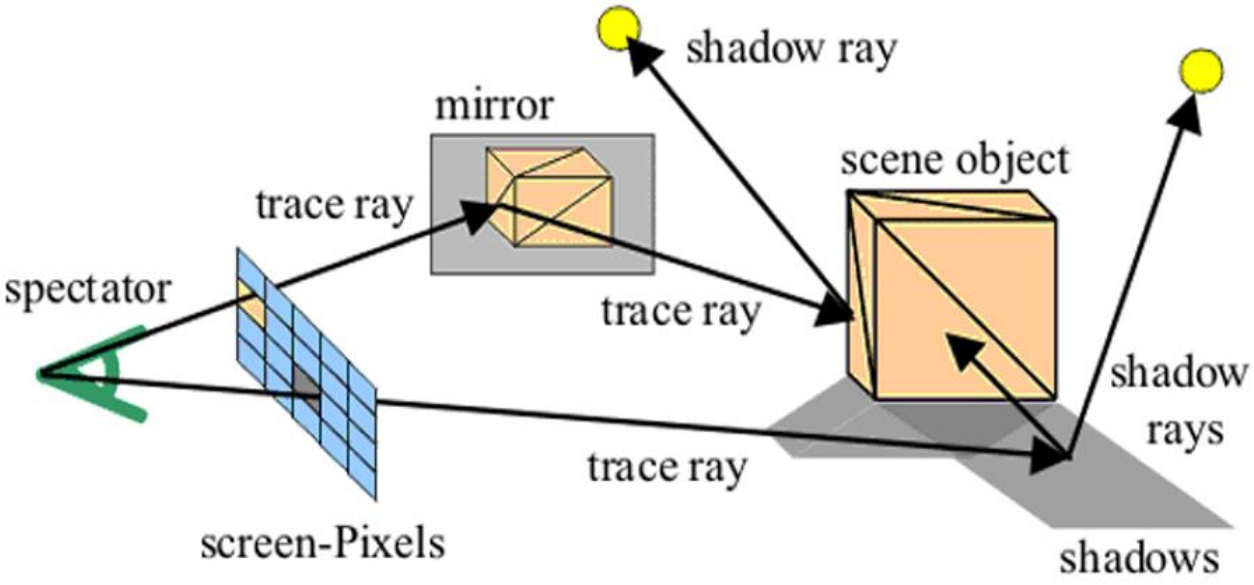
2. Principales Características

- Motor Gráfico: Iluminación
- Las luces pueden ser calculadas en **tiempo real o pueden estar precalculadas**. La elección dependerá de la complejidad y el nivel de realismo.



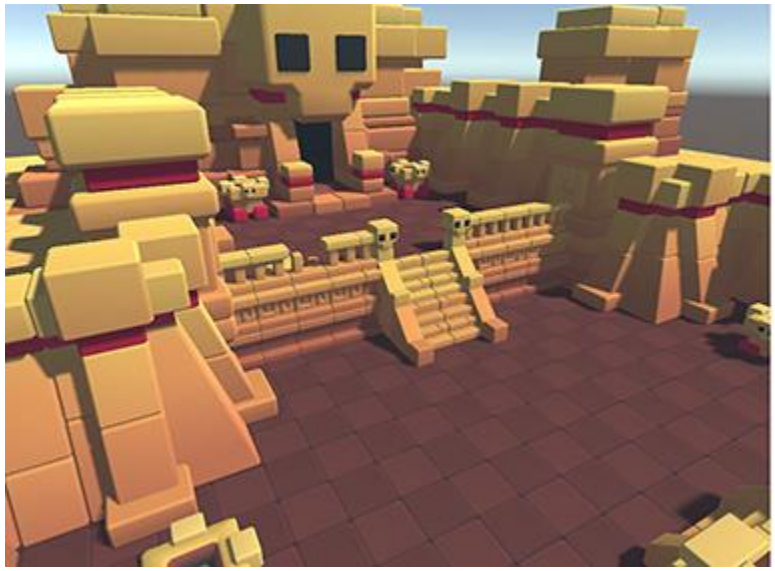
2. Principales Características

- Motor Gráfico: Raytracing en tiempo real



2. Principales Características

- Motor Gráfico: Cámaras



Perspectiva

Ortográfica

2. Principales Características

- Motor Gráfico: Diseño 2D/3D
  - En los proyectos **2D** se utilizan gráficos planos (sprites). La cámara es ortográfica, sin perspectiva.

2. Principales Características

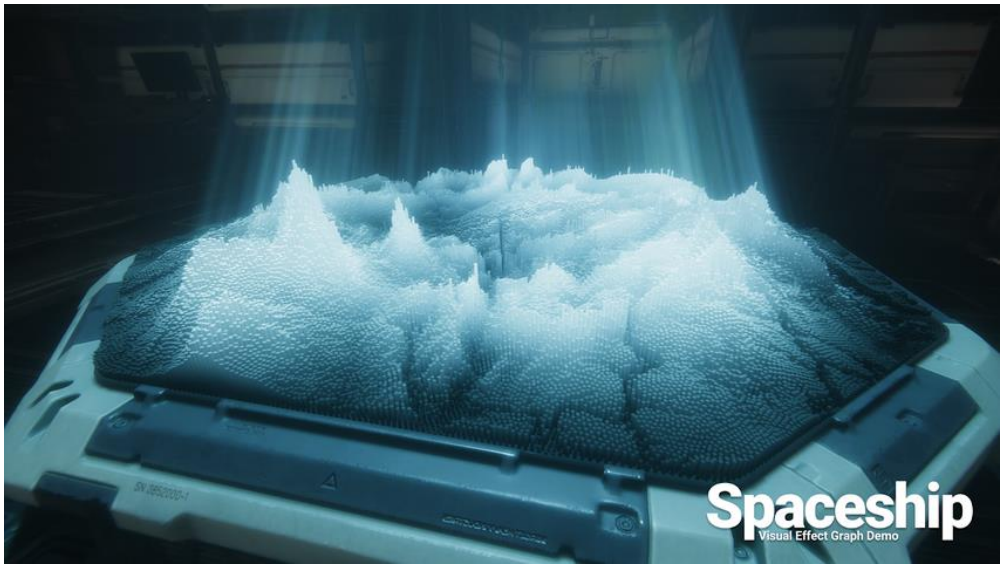
- Motor Gráfico: Diseño 2D/3D
  - En los proyectos **3D** se utilizan geometrías tridimensionales, con texturas y materiales que se renderizarán para obtener la imagen 2D en la pantalla. Habitualmente se usa una cámara con perspectiva.

2. Principales Características

- Motor Gráfico: Diseño 2D/3D
  - En los proyectos **2.5D** se utilizan geometrías tridimensionales pero se limita la jugabilidad a dos dimensiones

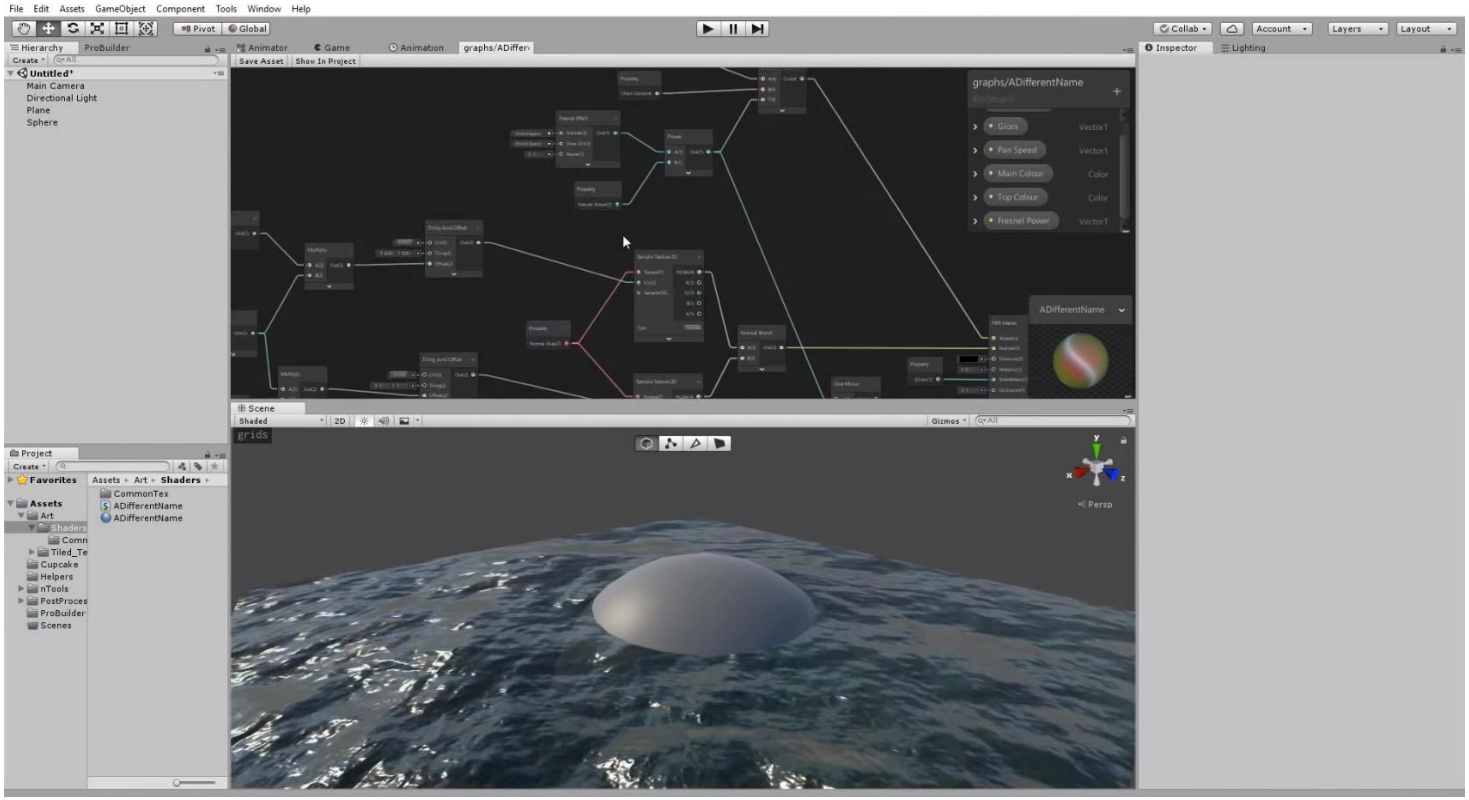
2. Principales Características

- Motor Gráfico: Partículas



2. Principales Características

- Motor Gráfico: Materiales, shaders, texturas, efectos visuales…



2. Principales Características

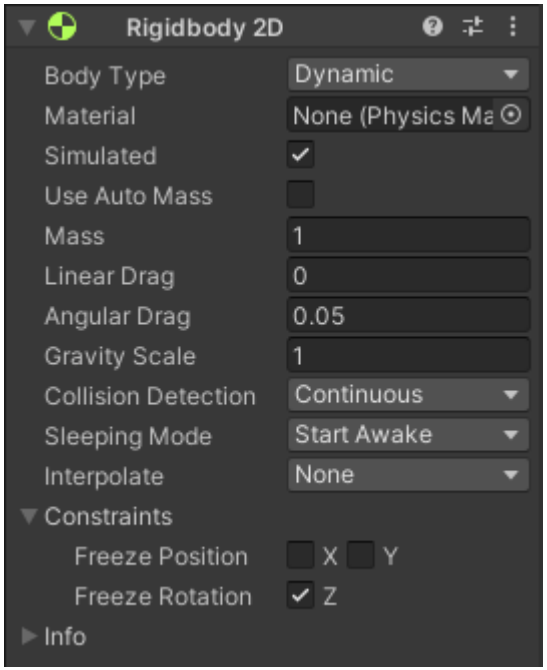
- Unity cuenta con un potente Motor Físico para implementar los comportamientos más realistas.
  - **Rigidbodies**
  - **Colliders**
  - **Joints**
  - **Character Controller**

2. Principales Características

- Motor Físico: Rigidbodies

- Es el componente que permite el comportamiento físico del GameObject.

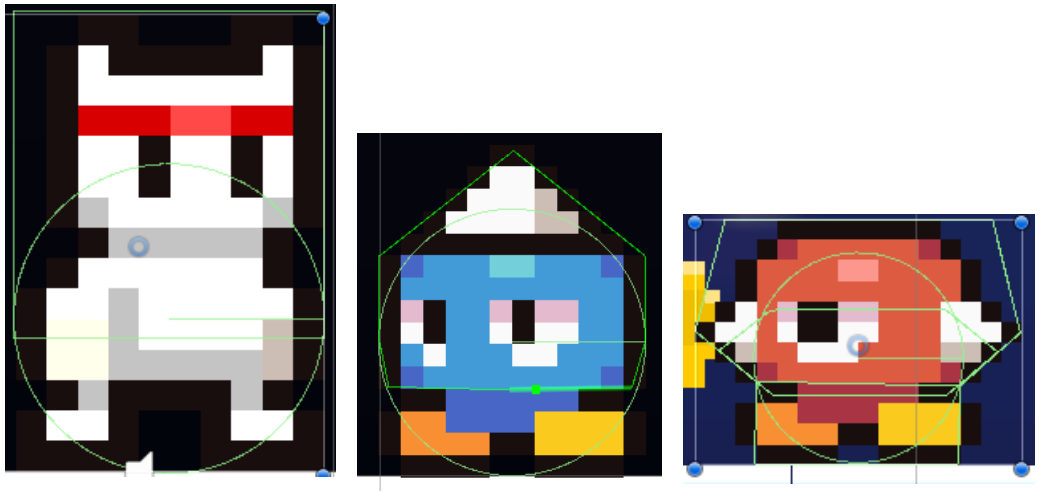
- Funciona aplicando fuerzas para generar movimiento realista como la gravedad, fuerza de salto, andar, correr, rebotar…



2. Principales Características

- Motor Físico: Colliders

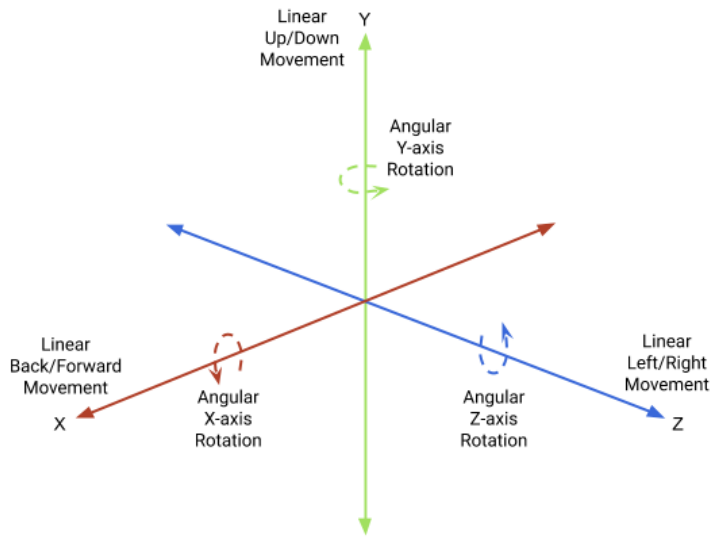
- Es el componente que define la forma GameObject frente a las colisiones con otros objetos. Para manejar las colisiones se usan Triggers.



2. Principales Características

- Motor Físico: Joints

- Es el componente que define las articulaciones de los GameObject uniendo varios Rigidbodies.



2. Principales Características

- Motor Físico: Character controller

- Es un componente predefinido que contiene el Rigidbody y los Colliders necesarios para un Jugador.

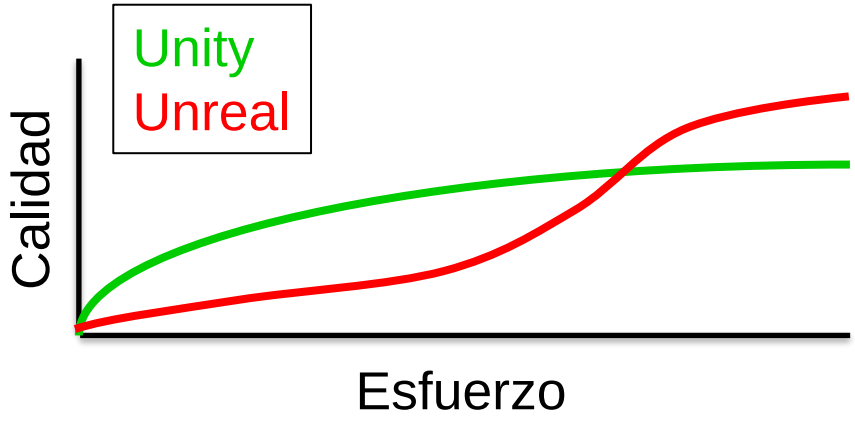
2. Principales Características

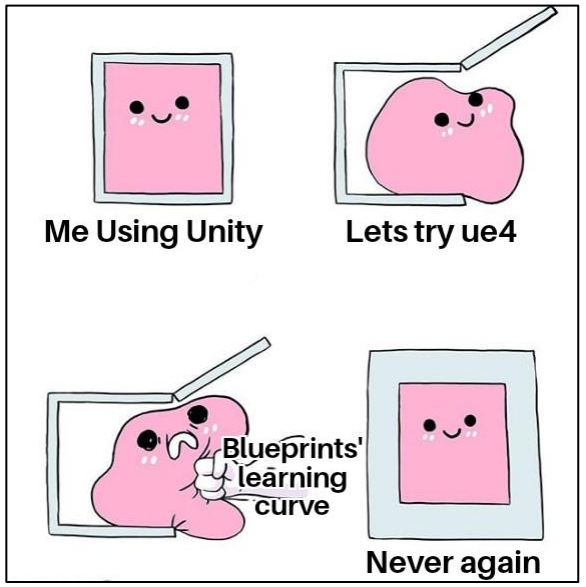
- Unity se trata de una herramienta multiplataforma, que además nos permite crear videojuegos para otras muchas plataformas.

- Gracias a las APIs disponibles totalmente integradas, para el programador/diseñador es totalmente transparente, simplemente escogemos a qué plataforma exportar el mismo juego.

2. Principales Características

- Respecto a su fácil aprendizaje, podemos destacar su gran cantidad de documentación y tutoriales oficiales, además de apoyarnos en los foros de la comunidad.



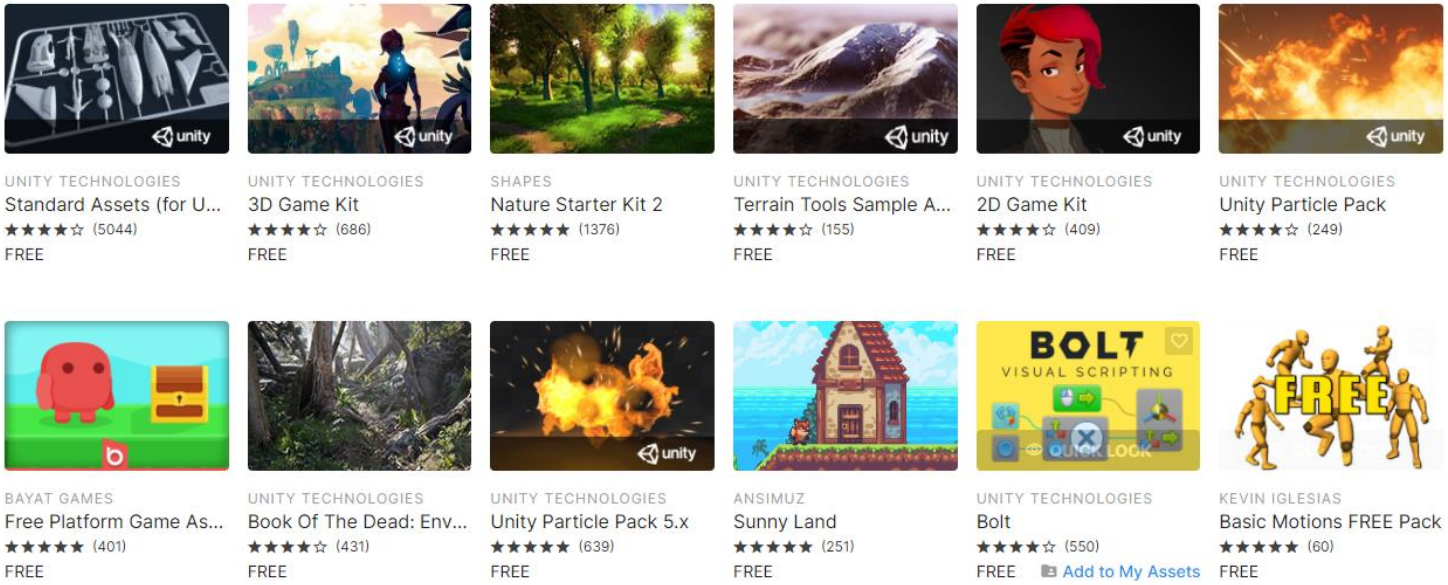


2. Principales Características

- En cuanto a la programación de la lógica del juego, ~~podemos usar diferentes lenguajes de programación de scripts~~. Este tipo de programación nos permite desarrollar un juego mediante módulos reutilizables.
  - **C#**
  - Boo (Deprecated)
  - JavaScript (Deprecated)

2. Principales Características

- Aunque no todo es “**picar código**”. Unity cuenta con la Asset Store, un lugar donde los usuarios suben los Assets que han diseñado para que otros usuarios puedan reusarlos de manera gratuita o pagando (a veces se usan demos).



2. Principales Características

- Hoy en día, las interfaces de Entrada/Salida son cada vez más amplias, encontrándonos frente a tecnologías de **Realidad Aumentada** y **Realidad Virtual**.

2. Principales Características

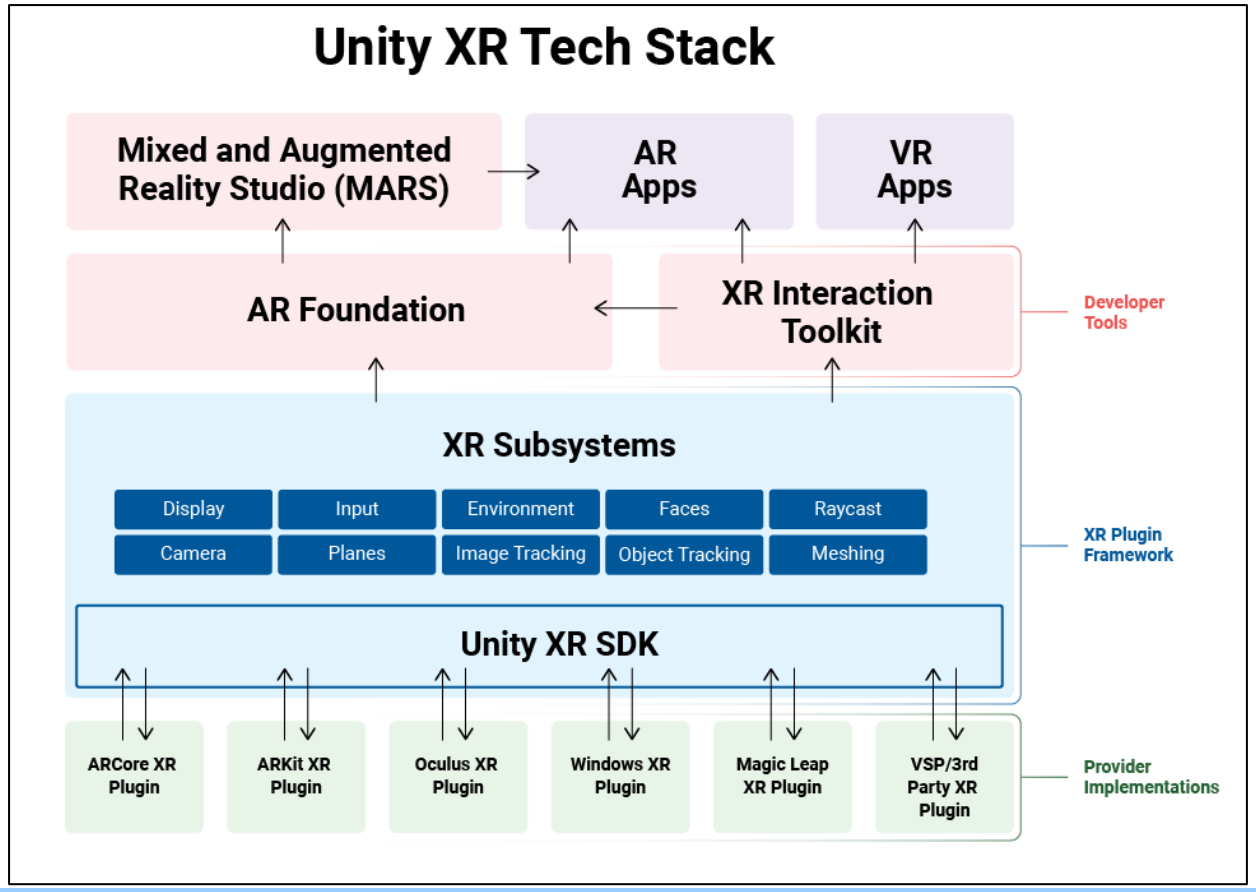
- **Realidad Aumentada**
- Unity ofrece varias formas de crear videojuegos basados en AR mediante la integración de herramientas como:
  - ARCore, desarrollado por Google para Android.
  - ARKit, desarrollado por Unity para iOS.
  - AR Foundation que aúna las principales funcionalidades de ambas.

2. Principales Características

- **Realidad Virtual**
- De igual manera, ofrece soporte para Realidad Virtual:
  - Oculus Rift, Casco virtual para PC.
  - Oculus Quest, Casco virtual para Android.
  - Google VR, Casco virtual para Android e iOS.
  - Open VR, SDK genérico para cascos virtuales.

2. Principales Características

- Unity XR: **Realidad Aumentada** y **Realidad Virtual**.



2. Principales Características

- En resumen:
  - Interfaz gráfica
  - Motor gráfico
  - Motor físico
  - Abstracción hardware – Multiplataforma
  - Documentación oficial – Curva de aprendizaje
  - Lenguajes de programación
  - Asset Store
  - AR - VR

- Actividad presencial entregable:

Redacte…

Mejor característica de Unity

Peor característica de Unity

Índice:

3. Lenguajes de Programación

- Como comentamos anteriormente, en Unity se puede programar la lógica del juego y los elementos que intervienen de manera modular mediante scripts.

- Para ello, usaremos el lenguaje de programación C#.

- C# es un lenguaje multiparadigma desarrollado por Microsoft, derivado de C/C++

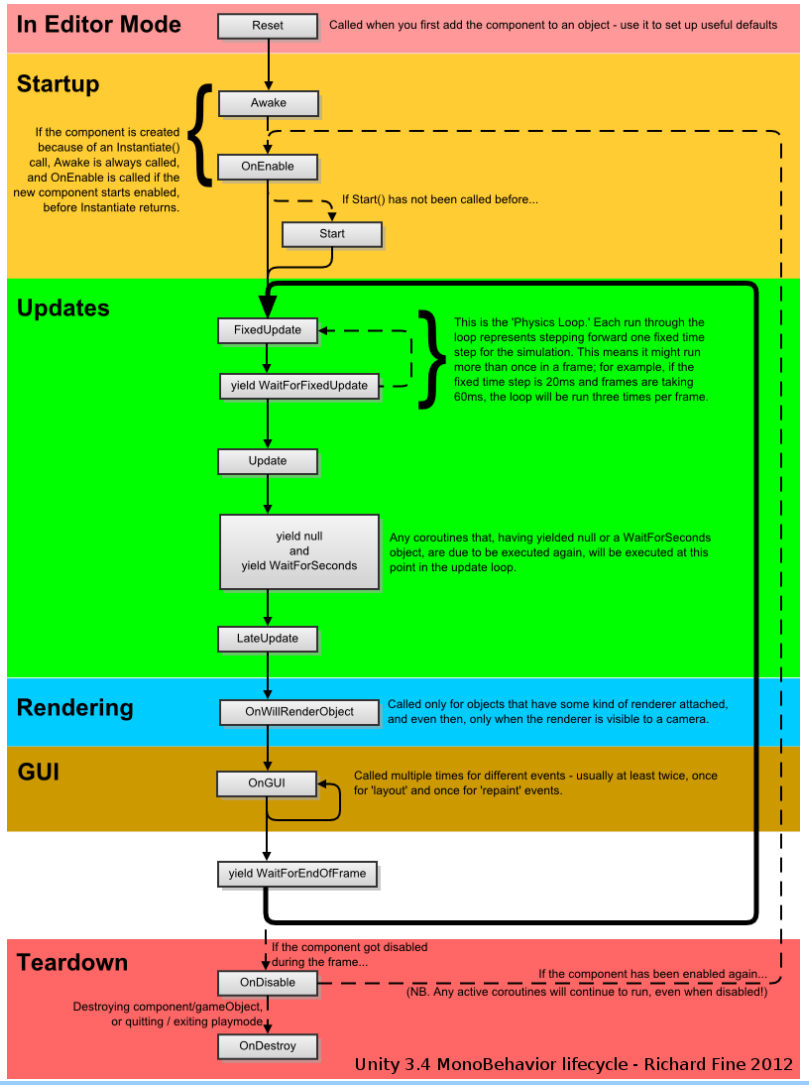
3. Lenguajes de Programación

- Por defecto, al añadir un script a cualquier GameObject, este script se estará ejecutando constantemente durante la vida de este GameObject.

- Es decir, la lógica del videojuego se producirá mediante la ejecución constante (bucle) de los scripts de cada GameObject en "*paralelo*" (para cada frame de manera secuencial) durante la partida.

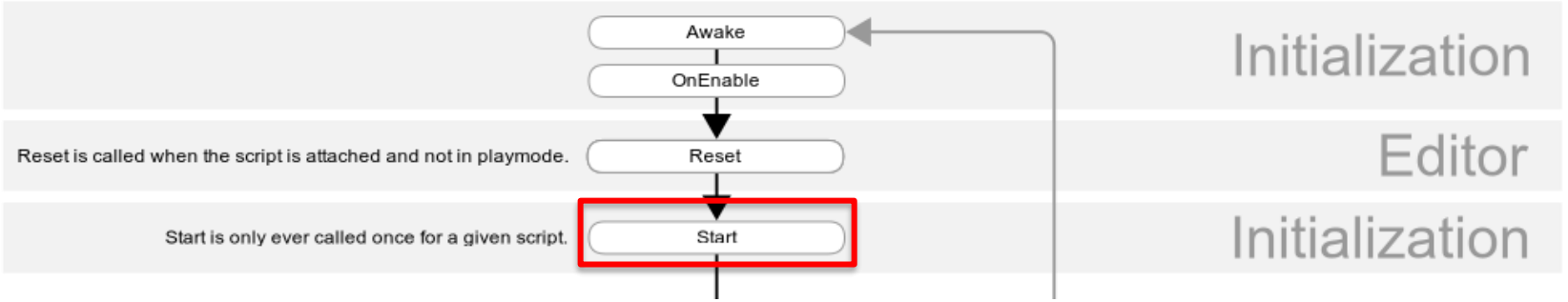
3. Lenguajes de Programación

- El ciclo de vida de un script de un **GameObject** es el siguiente:



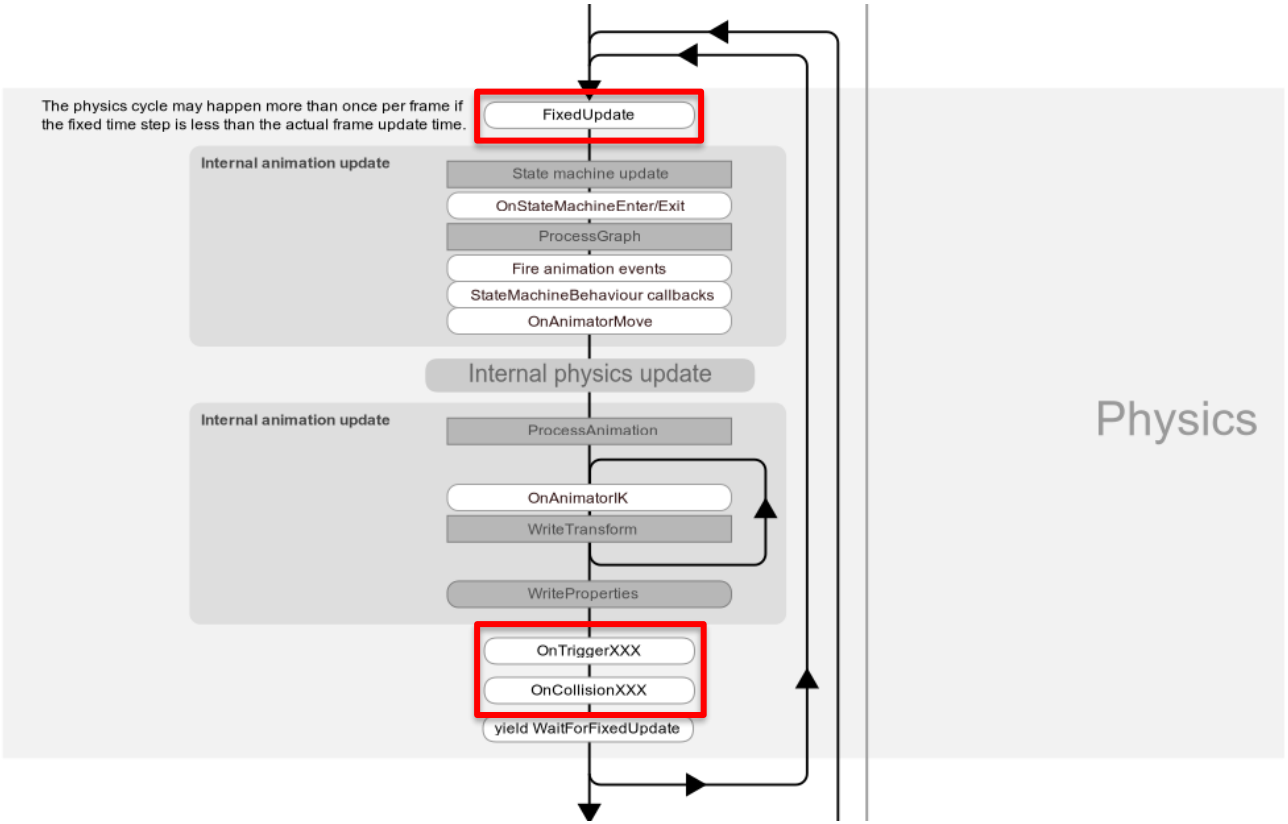
3. Lenguajes de Programación

- Inicialización



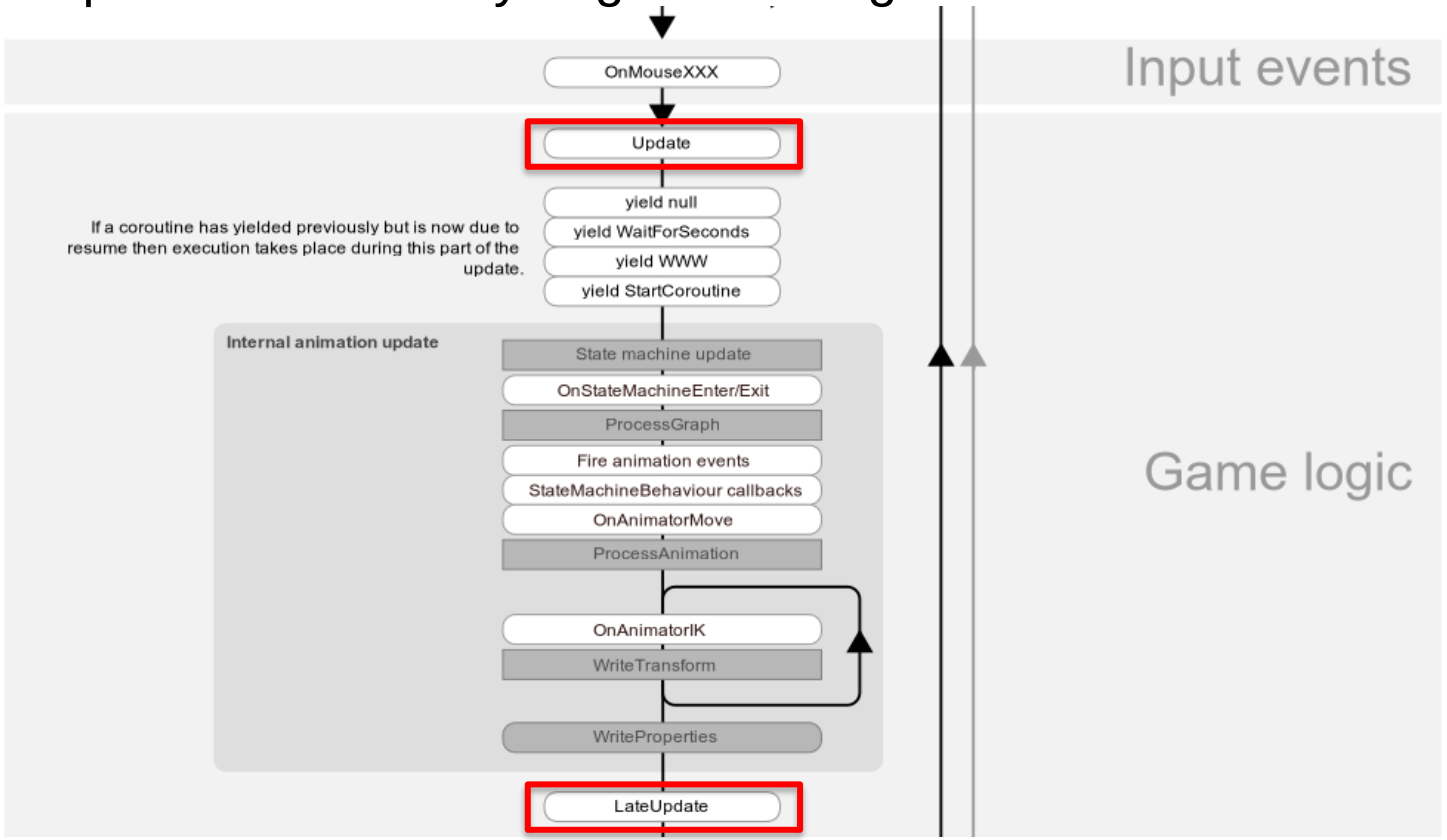
3. Lenguajes de Programación

- Actualización físicas



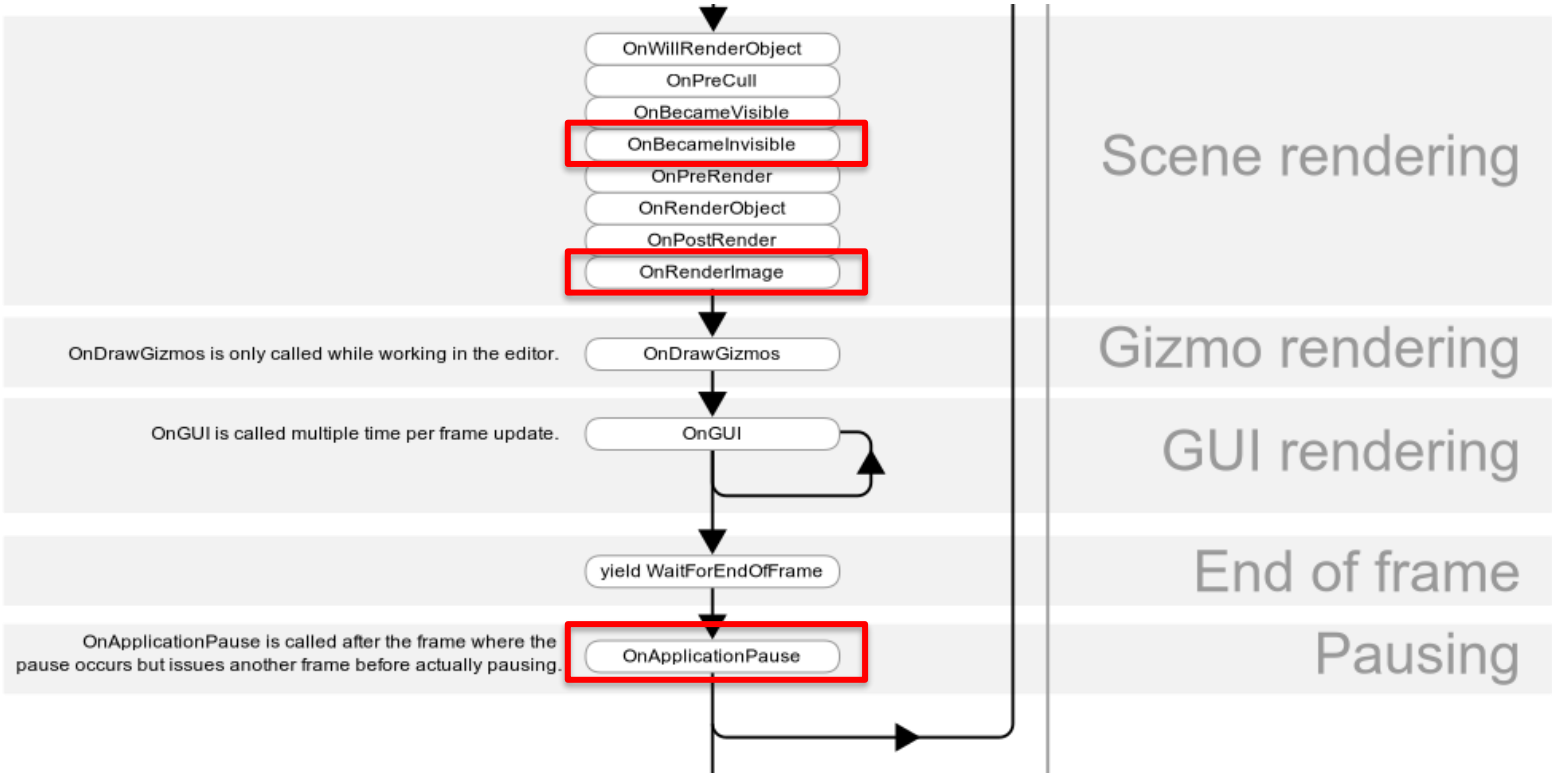
3. Lenguajes de Programación

- Captura de Entrada y Lógica de Juego



3. Lenguajes de Programación

- Renderizado y Fin del Frame

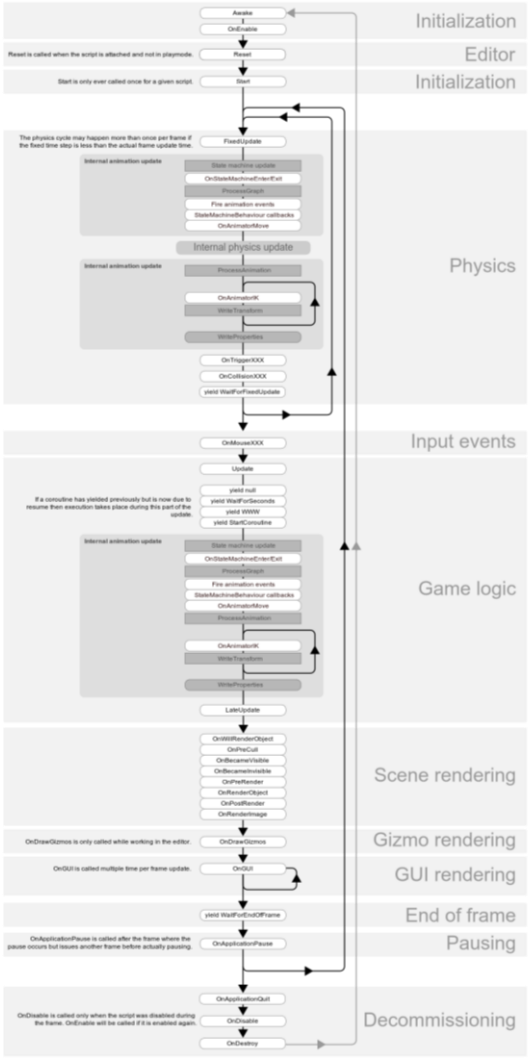


3. Lenguajes de Programación

- Liberación de recursos

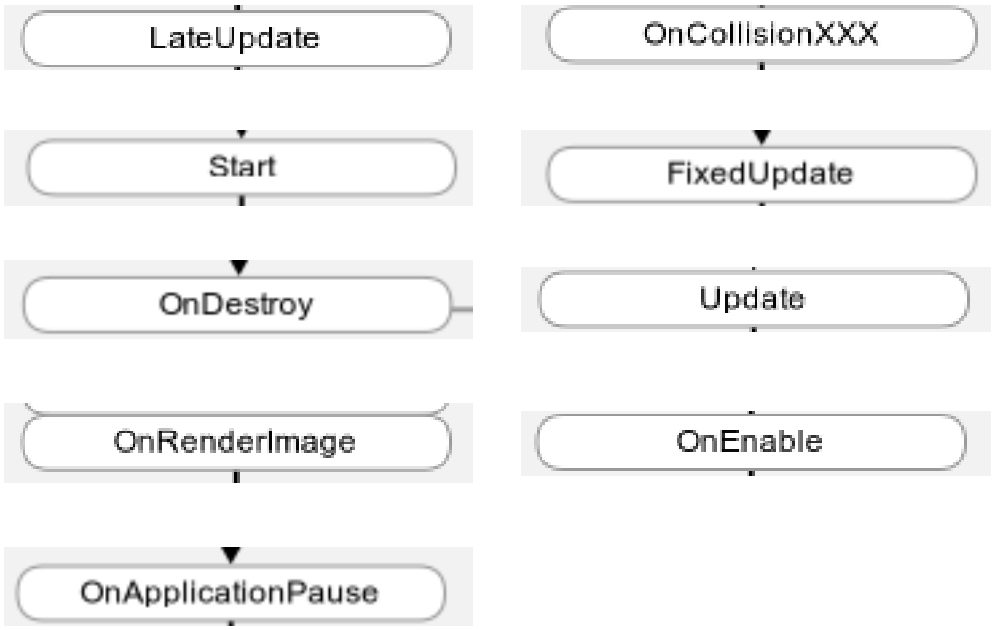
3. Lenguajes de Programación

- El ciclo de vida de un script de un GameObject es el siguiente:



- Actividad presencial entregable:

Organice el Game Loop:



3. Lenguajes de Programación

- Ejemplo script de enemigo simple (1/5):

```
public class enemigo_rojo_control : MonoBehaviour
{
    void Start()
    {
    }

    void Update()
    {
    }
}
```

3. Lenguajes de Programación

- Ejemplo script de enemigo simple (2/5):

```
public float maxVelocidad = 1f;
public float velocidad = 1f;

private Rigidbody2D rb2D;
private Animator anim;
```

3. Lenguajes de Programación

- Ejemplo script de enemigo simple (3/5):

```
void Start()
{
    rb2D = GetComponent<Rigidbody2D>();
    anim = GetComponent<Animator>();
}
```

3. Lenguajes de Programación

- Ejemplo script de enemigo simple (4/5):

```
void FixedUpdate()
{
        rb2D.AddForce(Vector2.right * velocidad);
        rb2D.velocity = new Vector2(Mathf.Clamp(rb2D.velocity.x, -maxVelocidad, maxVelocidad), rb2D.velocity.y);

        if (rb2D.velocity.x < 0.01 && rb2D.velocity.x > -0.01)
        {
            velocidad = -velocidad;
            rb2D.velocity = new Vector2(velocidad, rb2D.velocity.y);
        }

        if (velocidad < 0)
            transform.localScale = new Vector3(1f, 1f, 1f);
        if (velocidad > 0)
            transform.localScale = new Vector3(-1f, 1f, 1f);


}
```

3. Lenguajes de Programación

- Ejemplo script de enemigo simple (5/5):

```
void OnTriggerEnter2D(Collider2D col)
{
    if (col.gameObject.tag.Equals("Player"))
    {
        if (transform.position.y + 0.3 < col.transform.position.y)
        {
            velocidad = 0;
            rb2D.velocity= Vector2.zero;
            anim.SetBool("muerte", true);
            Invoke("eliminar", 0.7f);
            col.SendMessage("SaltoEnemigo");
        }
        else
            col.SendMessage("GolpeEnemigo",transform.position.x);
    }
}

void eliminar()
{
    Destroy(gameObject);
}
```

3. Lenguajes de Programación

- Ejemplo script de moneda:

```
public class Moneda_grande : MonoBehaviour
{
    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.tag.Equals("Player"))
        {
            col.SendMessage("AddScore", 300);
            Destroy(gameObject);
        }
    }
}
```

3. Lenguajes de Programación

- Ejemplo script de tiempo:

```
public class TimeNormal : MonoBehaviour
{
    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.tag.Equals("Player"))
        {
            Time.timeScale = 1f;
            Destroy(gameObject);
        }
    }
}
```

3. Lenguajes de Programación

- Mediante estos scripts podremos diseñar toda la lógica de nuestro juego y definir nuestras reglas de comportamiento, pero un elemento principal será el diseño de los NPC.

- Actualmente, se tiende a diseñar unos NPC realistas, con comportamiento inteligente, para lo que se utilizan técnicas de Inteligencia Artificial.

- Estas técnicas se pueden implementar mediante scripts o reutilizando Assets.

Índice:

4. Casos de Uso

- Cada vez es más común encontrarnos con juegos desarrollados con Unity. La lista es considerable, pero podemos destacar los más virales:

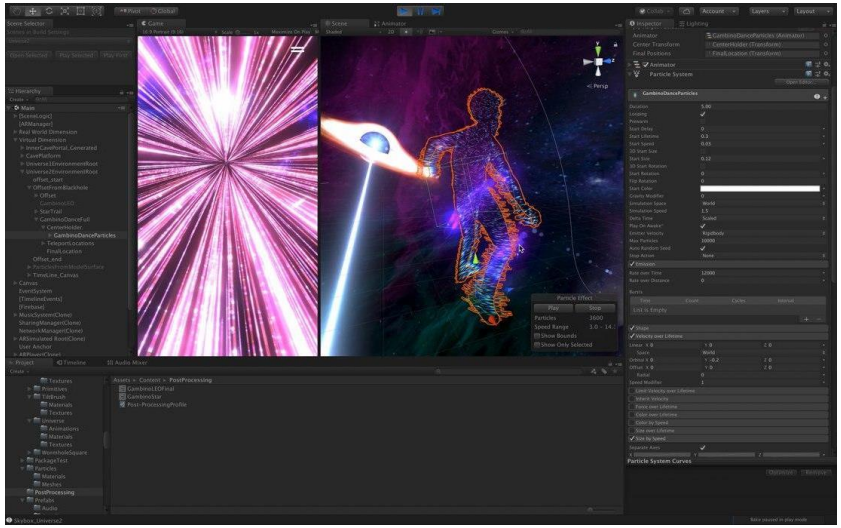
- Cup Head
- Fall Guys
- Hearthstone
- Among Us
- Beat Saber

4. Casos de Uso

- Pero no sólo se diseñan videojuegos en Unity:

PHAROS AR

Realidad Aumentada “Multijugador” gracias a ARCore y Unity



https://unity.com/es/madewith/pharos-ar

4. Casos de Uso

- Pero no sólo se diseñan videojuegos en Unity:

Configurador del Lexus LC 500

Máximo realismo gracias al HDRP en tiempo real de Unity

https://blogs.unity3d.com/es/2019/01/23/how-to-create-a-configurable-car-in-unity-hdrp/

Índice:

5. Casos prácticos

- A continuación, veremos dos ejemplos prácticos de videojuegos en Unity:
  - Por un lado, un videojuego de plataformas en 2D.
  - Luego, veremos un videojuego en tercera persona en 3D.

6. Bibliografía

https://docs.unity3d.com/Manual/index.html

GAME OVER