unity3d.com/pages/the-blacksmith

Introducción a Unity

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

ENTORNOS DE DESARROLLO DE VIDEOJUEGOS

Índice

- ¿Por qué Unity?
- Recordatorio de conceptos de Informática Gráfica
- El Editor de Unity
- Tu primera escena

¿Por qué Unity?

- Primera versión publicada en 2005
- El Editor se puede usar en Windows, Mac OS X y Linux (experimental)
- Motor multiplataforma:
  - Windows, OS X, Linux, Xbox, Wii U, Nintendo Switch, PlayStation, iOS, Android, Windows Phone, Web GL, Oculus Rift, Steam VR…
  - Direct3D, OpenGL, OpenGL ES, Vulkan, Metal, APIs propietarias…
- Para juegos 2D y 3D
- Muy usado en la industria (para videojuegos y otras áreas)
  - https://madewith.unity.com/
  - https://en.wikipedia.org/wiki/List_of_Unity_games

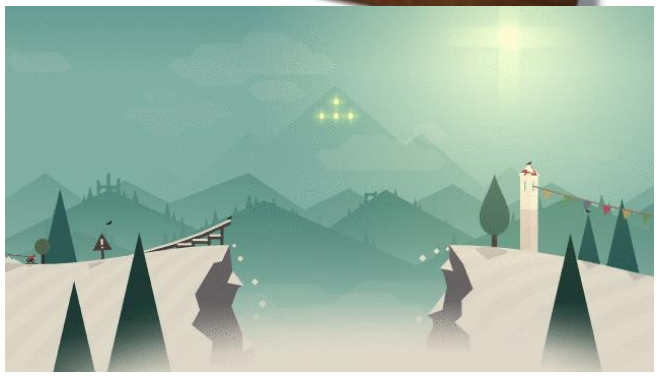
¡Cuidado! Es Unity, no Unity3D (unity3d es/era el nombre del dominio)

Juegos comerciales desarrollados con Unity

Hitman Go. Square Enix. 2014

Kerbal Space Program. Squad. 2015

Cuphead. StudioMDHR. 2017



Alto's Adventure. Snowman. 2015

Hearthstone. Blizzard Entert. 2014

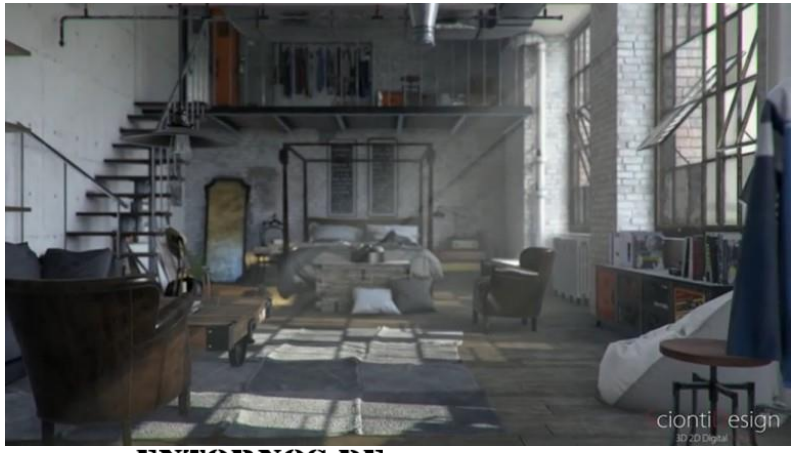
Angry birds 2. Rovio Entert. 2015

Serious games desarrollados con Unity

https://businesssimulations.com/Articles/new-3d-version-of-manage-business-simulation-game-launched-in-unity

https://archvirtual.com/project/river-home-bim-import-unity3d-oculus-rift/

https://www.oemoffhighway.com/operator-cab/operator-interface/press-release/20860010/jlg-industries-inc-jlg-featuring-vr-simulator-at-2017-unity-vision-summit

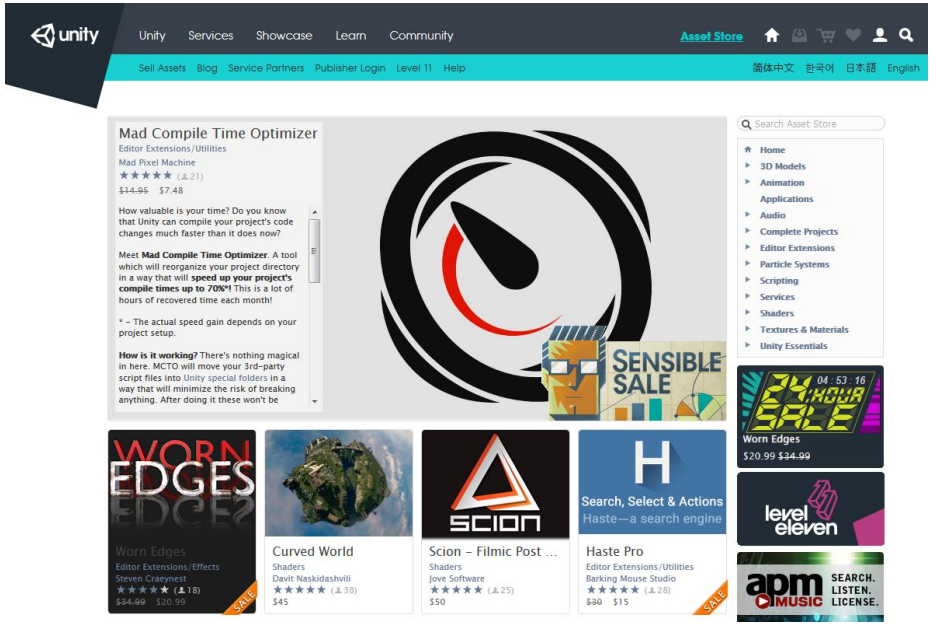


https://cinema-suite.com/architectural-visualization/

http://www.holovis.com/jobs/3d-gui-developer-c-and-unity/

¿Por qué Unity?

- El entorno de desarrollo está basado en el proyecto Mono (una implementación *open source* de .NET)
- Lenguaje de scripts: C# (y antes, Boo y UnityScript)
- Asset Store con montones de recursos (gratuitos y de pago)



¿Por qué Unity?

Personal

Comienza a crear con la versión gratuita de Unity

Gratis

Comencemos | Conoce más

Requisitos:
Ingresos o fondos inferiores a USD 100 mil en los últimos 12 meses

- Versión más reciente de la plataforma básica de desarrollo de Unity
- Recursos para aprender y comenzar a usar Unity

Compara los planes

Learn Premium

Domina Unity con aprendizaje en vivo y bajo demanda

15 $ Plan mensual, sin compromisos

Pruébalo gratis por 30 días | Conoce más

o suscríbete ahora

- Sesiones en vivo con instructores certificados por Unity
- Contenido bajo demanda actualizado en cada versión
- Recursos para los creadores en todas las etapas de aprendizaje

Incluido con los planes Plus, Pro y Enterprise

Plus

Más funcionalidad y recursos para potenciar tus proyectos

40 $ Plan anual, pagado mensualmente

Suscribirse | Conoce más

Requisitos:
Ingresos o fondos inferiores a USD 200 mil en los últimos 12 meses

- Versión más reciente de la plataforma básica de desarrollo de Unity
- Aprendizaje premium para dominar Unity
- Interfaz de usuario con tema oscuro
- Personalización de la pantalla de inicio
- Análisis de operaciones en vivo
- Diagnóstico en la nube en tiempo real

Pro LO MÁS POPULAR

Una solución completa para que los profesionales creen, operen y moneticen

150 $ Plan anual, pagado mensualmente

Suscribirse | Conoce más

Requerido si los ingresos o los fondos superan los USD 200 mil en los últimos 12 meses

- **Todo en Plus**
- Hasta tres licencias personales de Unity Teams Advanced
- Prioridad en el servicio al cliente
- Acceso prioritario a Unity Success Advisors
- Opciones de personalización disponibles para la compra:
  - + Soporte Premium
  - + Acceso al código fuente

https://store.unity.com/es/

Unity: ¿dónde empiezo?

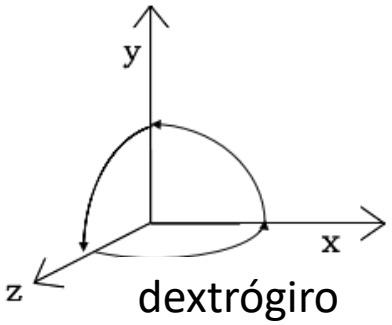
- https://unity.com
  - Página principal. Consigue tu versión personal aquí
- https://learn.unity.com
  - Repositorio con recursos como:
    - Tutoriales: https://learn.unity.com/tutorials
    - Documentación: https://docs.unity3d.com
    - Live training: http://unity3d.com/learn/live-training
    - Foros, Respuestas, Chat…: http://unity3d.com/community

Repaso de los fundamentos de Gráficos

Sistemas de coordenadas

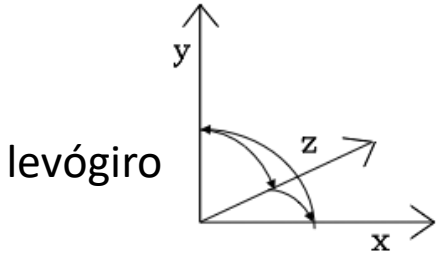
- Cualquier objeto en un espacio (2D o 3D) se define con respecto a un sistema de coordenadas
- Hay dos tipos de sistemas de coordenadas 3D:

el eje Z apunta hacia fuera de la pantalla

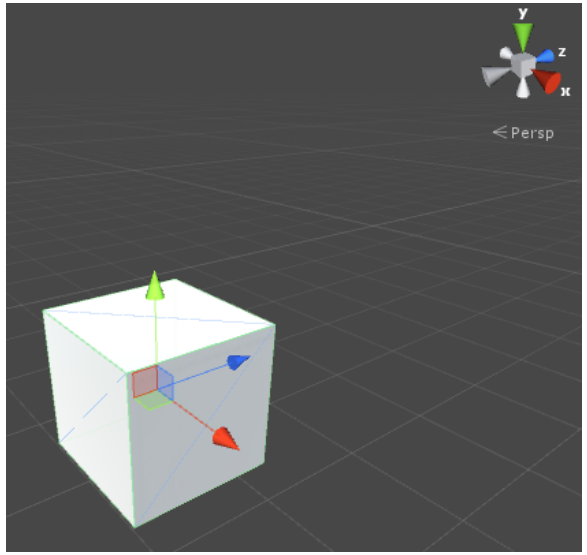


dextrógiro

el eje Z apunta hacia dentro de la pantalla



levógiro

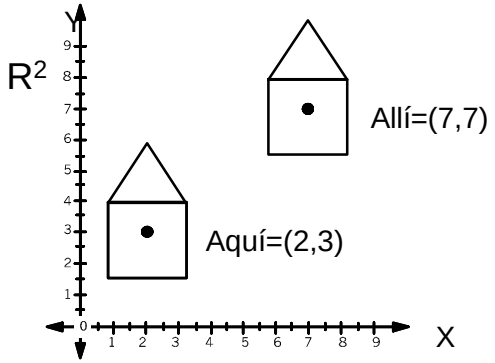


- ¿Quieres 2D? Descarta el eje Z

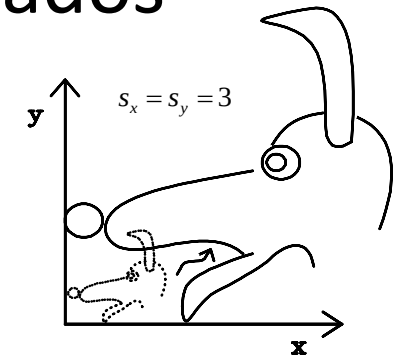
Repaso de los fundamentos de Gráficos

Transformaciones

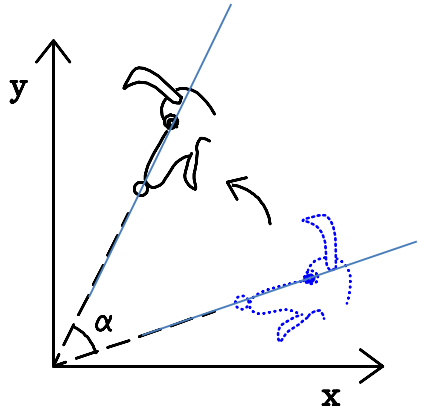
- Translaciones

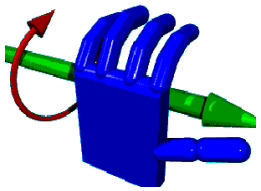


- Escalados



- Rotaciones

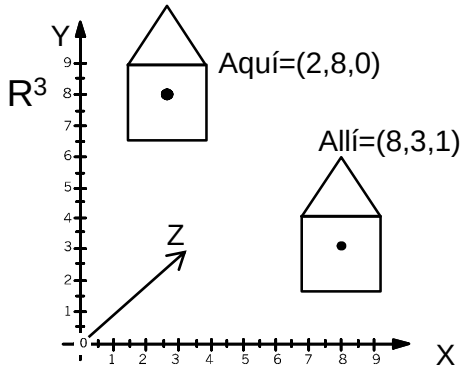




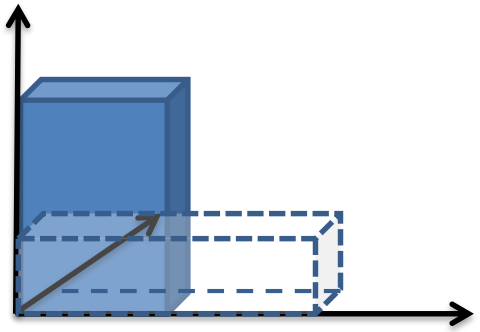
Repaso de los fundamentos de Gráficos

Transformaciones

- Pregunta



?

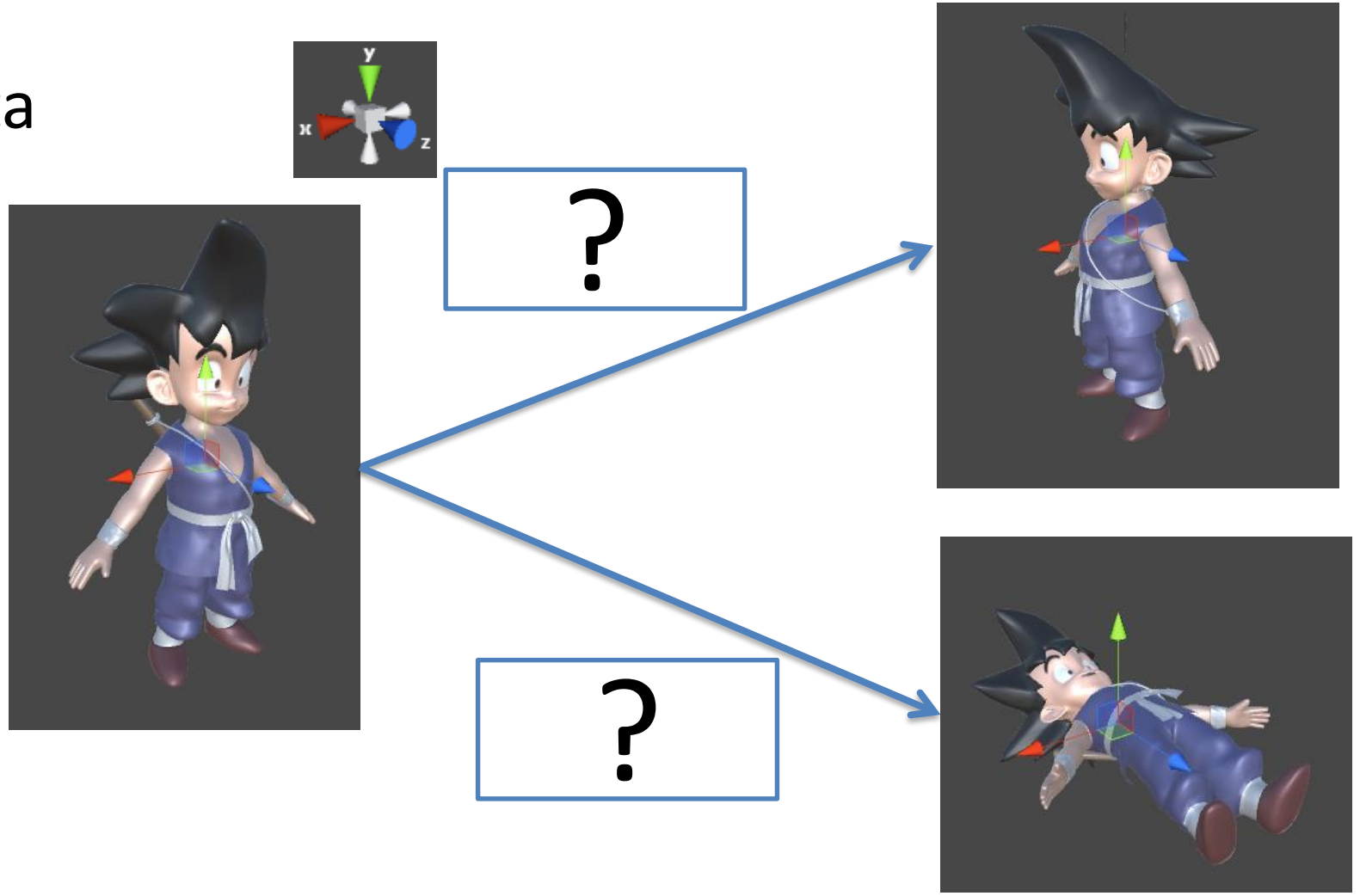


?

Repaso de los fundamentos de Gráficos

Transformaciones

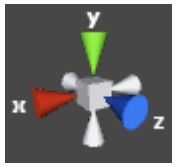
- Pregunta

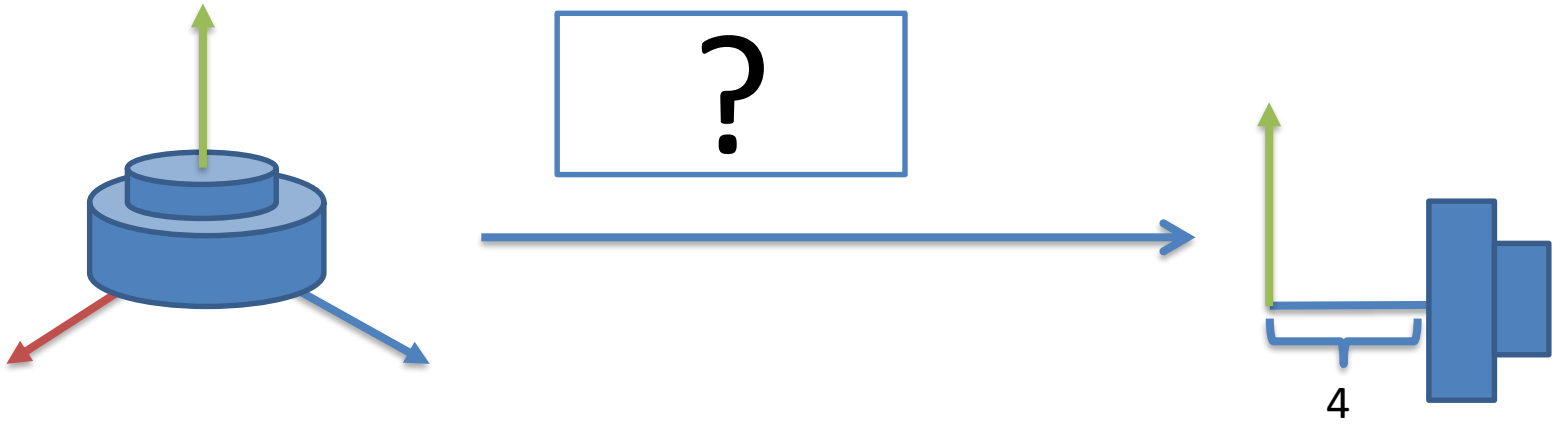


Repaso de los fundamentos de Gráficos

Transformaciones

- Pregunta

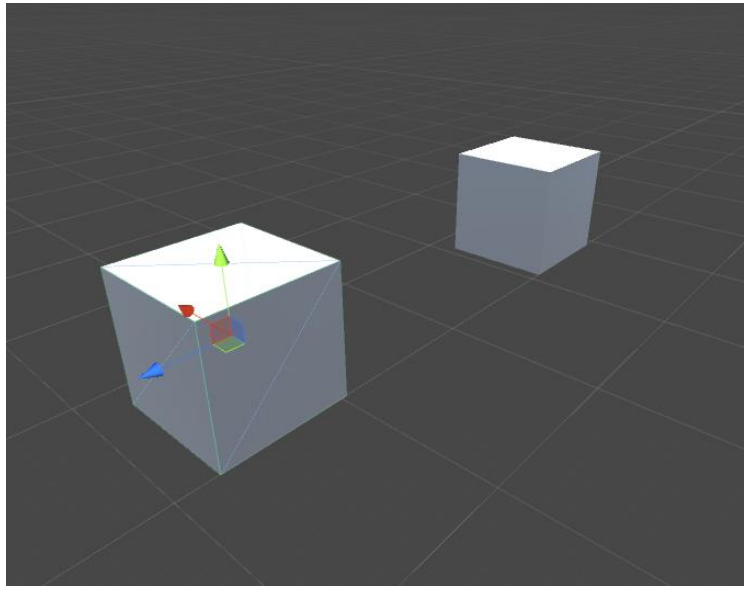




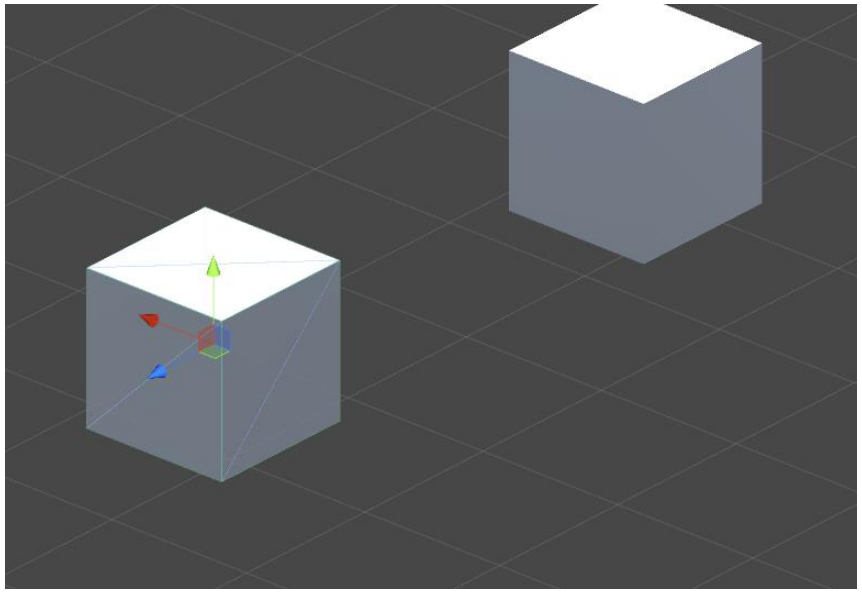
Repaso de los fundamentos de Gráficos

Cámaras

- Hay dos tipos de cámaras (para proyectar objetos 3D en imágenes 2D)



Perspectiva

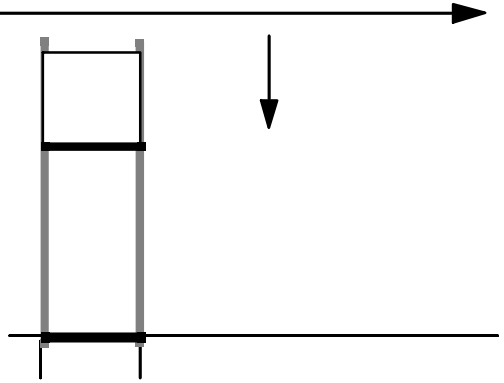


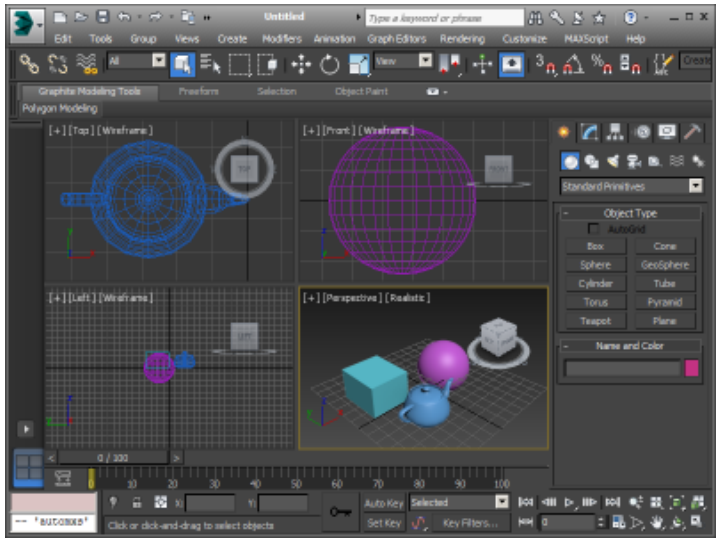
Paralela

Repaso de los fundamentos de Gráficos

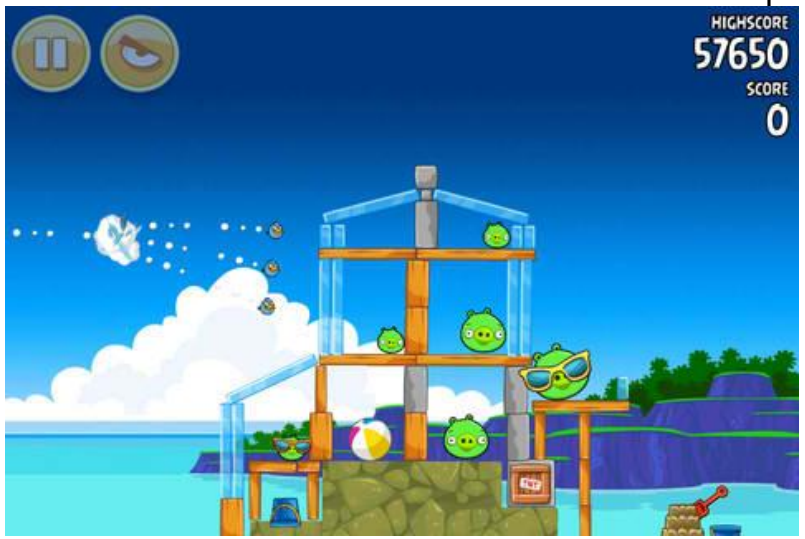
Cámaras

- Cámaras paralelas (u ortográficas o isométricas)
  - Usadas para representar tamaños exactos
  - Visualiza objetos sin distorsión perspectiva
- Usado en CAD, mapas, etc.





3D Studio Max



Angry Birds

Diablo

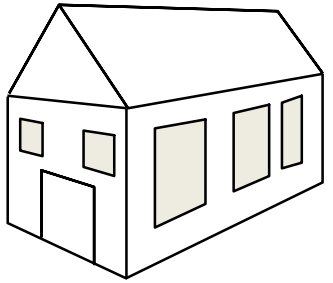
Repaso de Gráficos

Cámaras

- Cámara perspectiva
  - Proporciona realismo visual y pistas 3D
  - No preserva la forma ni escala de los objetos (excepto para los planos paralelos al plano de proyección)
  - Las líneas paralelas dejan de serlo en la proyección
  - El tamaño de un objeto disminuye con la distancia desde la cámara

http://flickr.com/photos/andykirk/

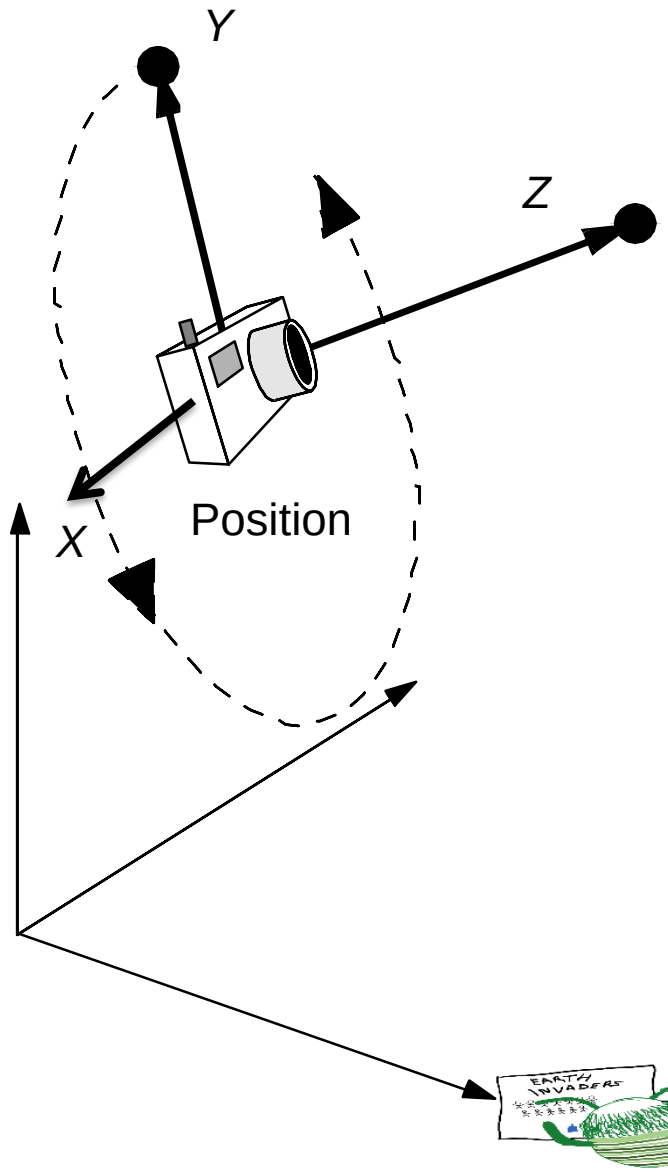
thomastnguyen.wordpress.com/2012/09/06/false-perspective/



Repaso de los fundamentos de Gráficos

Cámaras

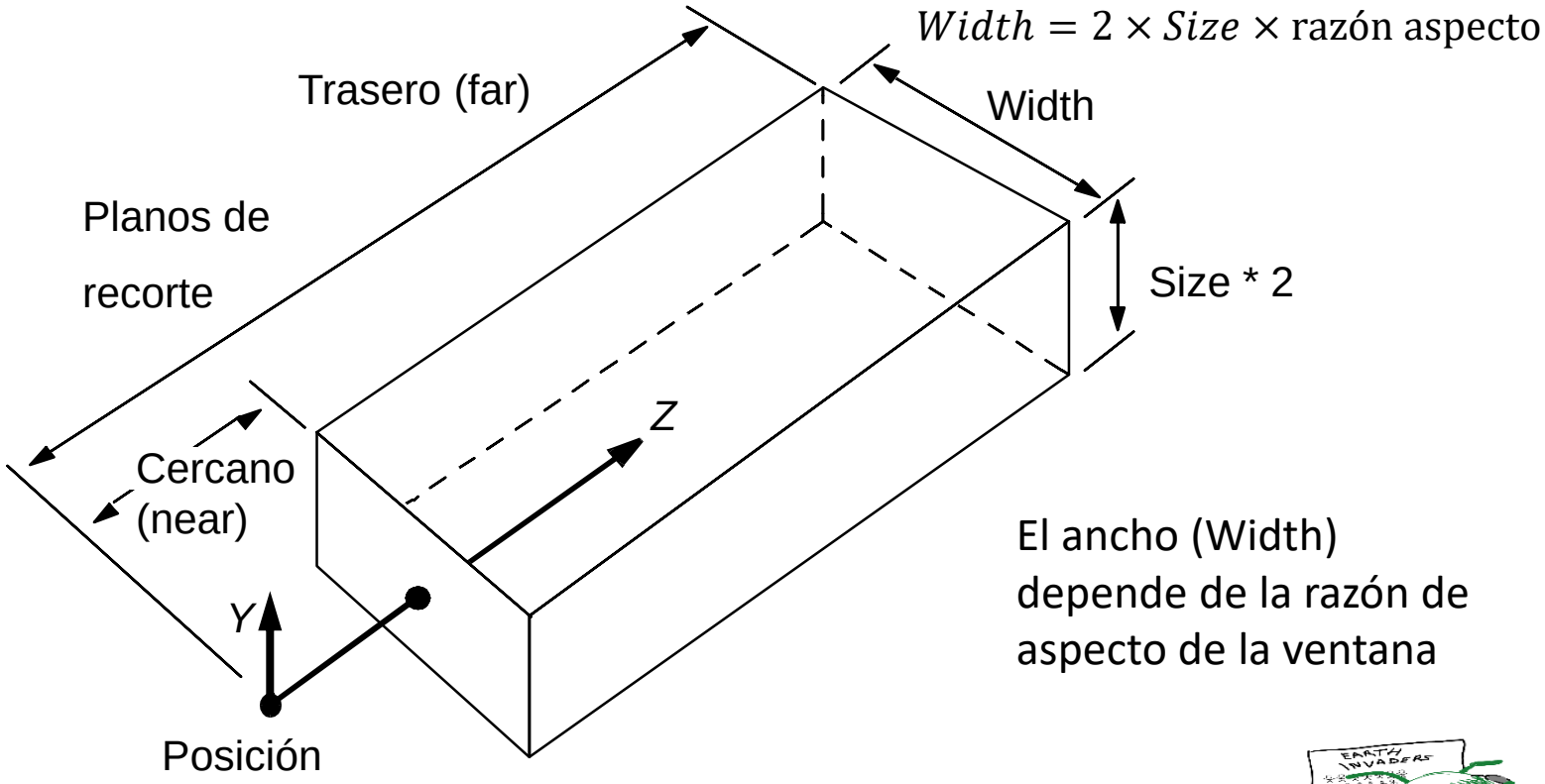
- Situando la cámara en la escena
  - Posición
    - Tres grados de libertad: las coordenadas (x, y, z) de la cámara en el espacio 3D
  - Orientación
    - La cámara mira a lo largo de su eje Z
    - Su eje Y define la inclinación de la cámara



Repaso de los fundamentos de Gráficos

Cámaras

- Volumen de la vista de una proyección ortográfica
  - El volumen de la vista define la parte de la escena que el usuario ve
  - Los objetos se recortan contra este volumen

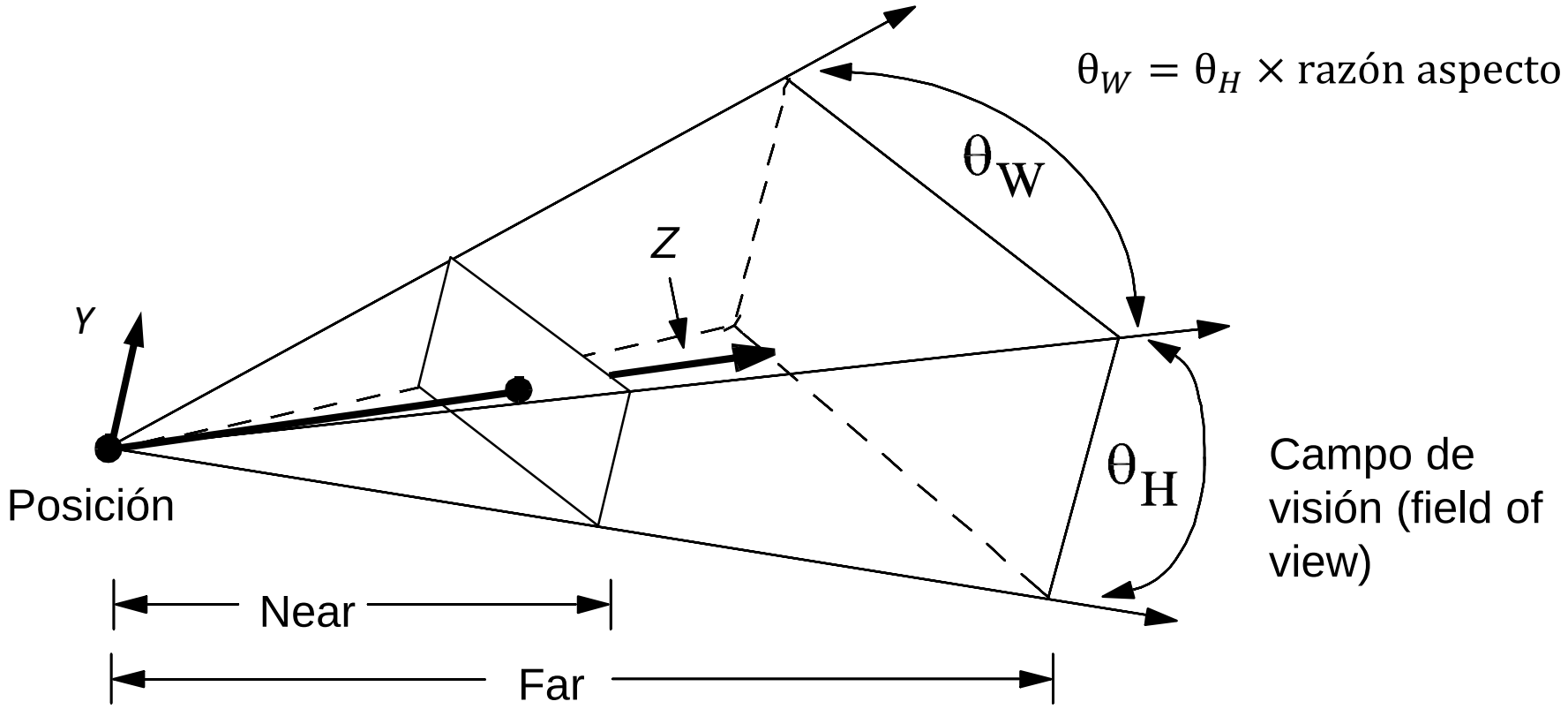


El ancho (Width) depende de la razón de aspecto de la ventana

Repaso de los fundamentos de Gráficos

Cámaras

- Pirámide truncada (frustum) de una proyección perspectiva



Repaso de los fundamentos de Gráficos

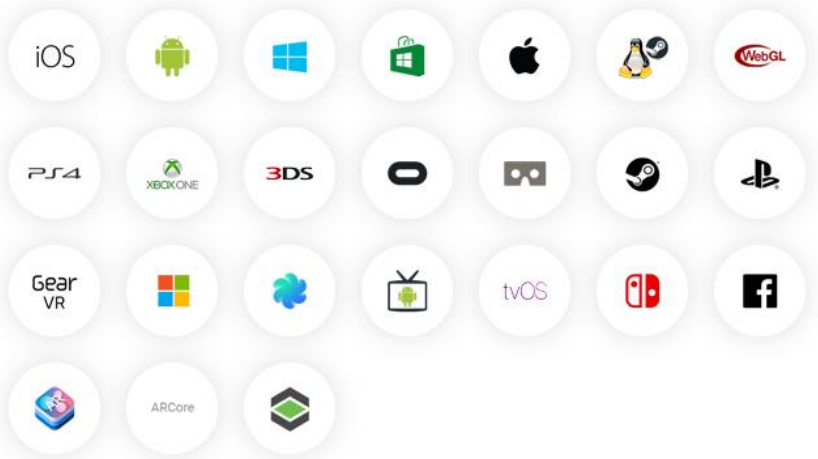
Interactividad

- El juego debería funcionar, como mínimo, a 30 FPS
- El problema es que el mismo ejecutable debe funcionar tanto en máquinas potentes, como en máquinas limitadas
  - El mismo proyecto de Unity se puede construir para móviles, consolas y ordenadores de escritorio, pero hay diferencias significativas en las capacidades de cada uno
- Parámetros a controlar
  - Tamaños de textura, complejidad geométrica de los modelos, efectos visuales, etc.

Unity

El editor de Unity

- El editor de Unity está disponible para Windows, Mac OS X y Linux



- Motores de física: Box2D y PhysX
- Puede importar la mayoría de formatos de fichero actuales*
- Características: atlas de sprites, física 2D, soporte para mundos con celdas rectangulares, hexagonales o isométricas, animación esquetal 2D…

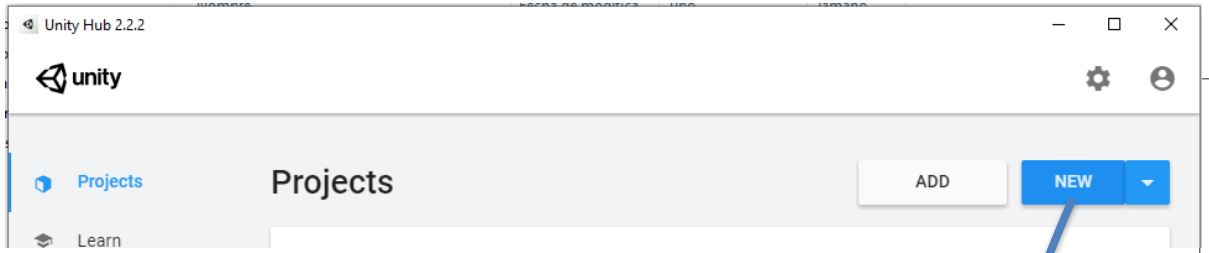
Unity

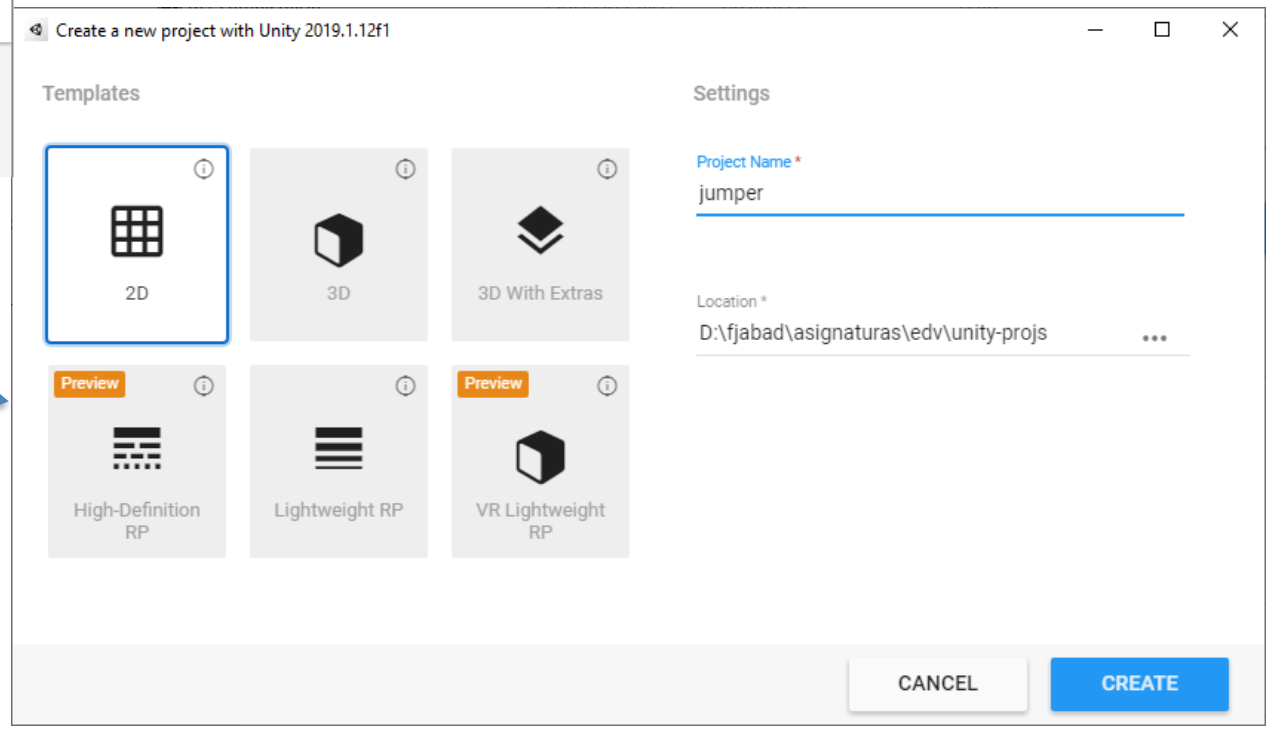
- Ventajas del editor
  - Permite prototipar el juego rápidamente
  - Muy flexible, se pueden añadir nuevas características y menús al editor
  - Se puede modificar el juego incluso en ejecución
  - Usa un sistema de componentes, que permite construir objetos a medida sin herencia
  - El editor trabaja por igual con componentes predefinidos (hay muchos) y los específicos (creados por código)

El editor de Unity

Creando un nuevo proyecto

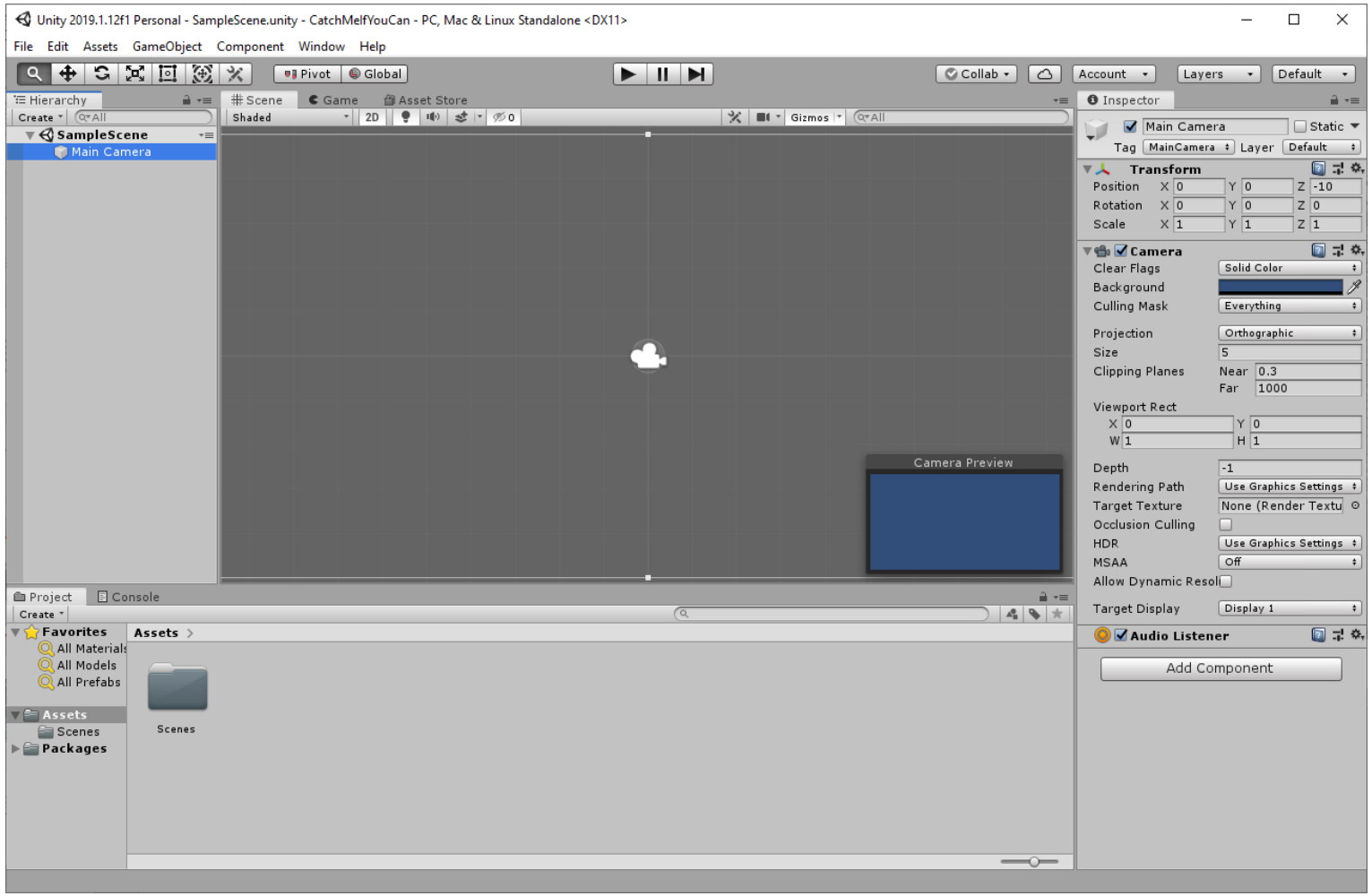
- Después de acceder, crear un nuevo proyecto:





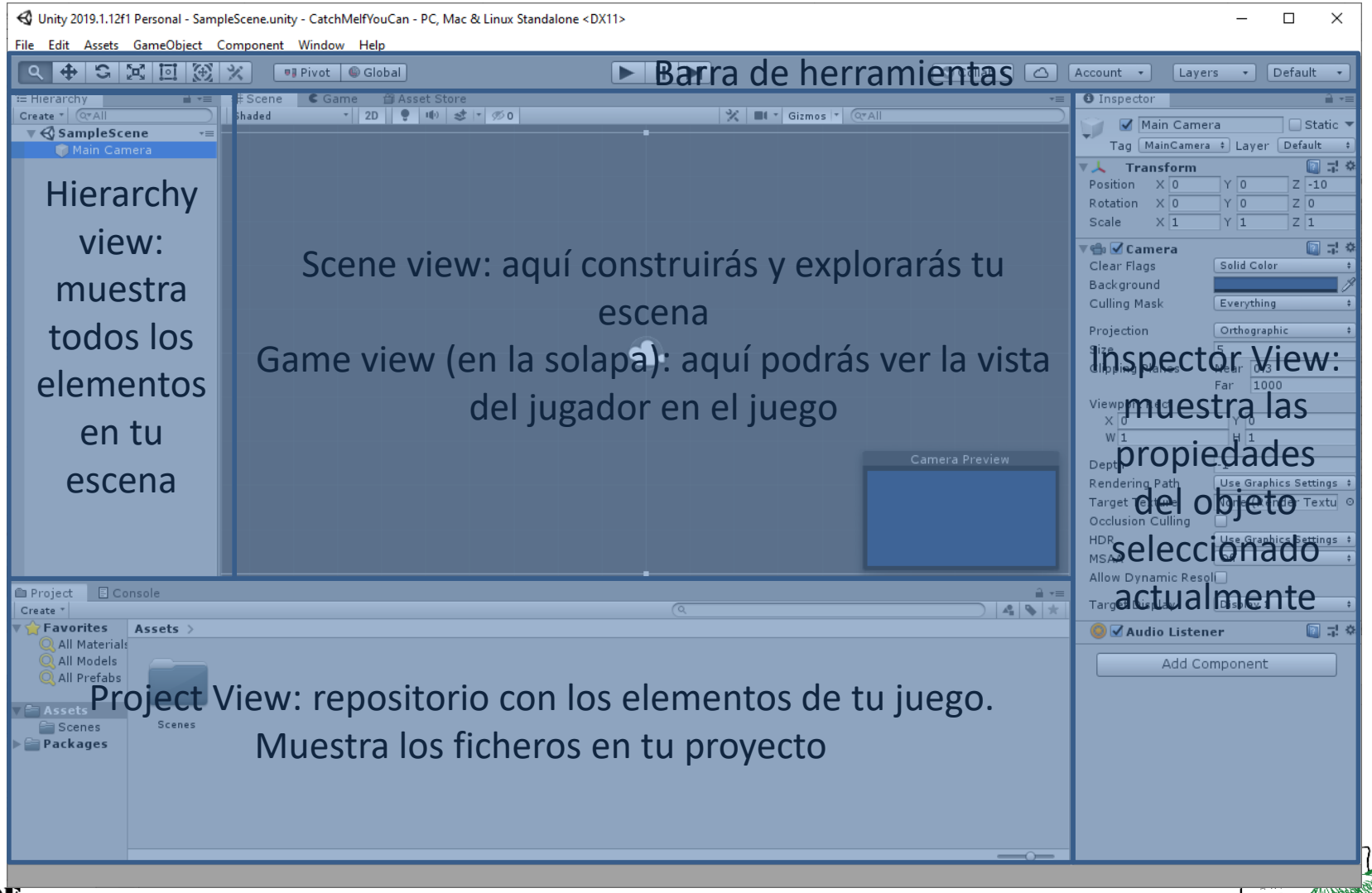
El editor de Unity

Organización



El editor de Unity

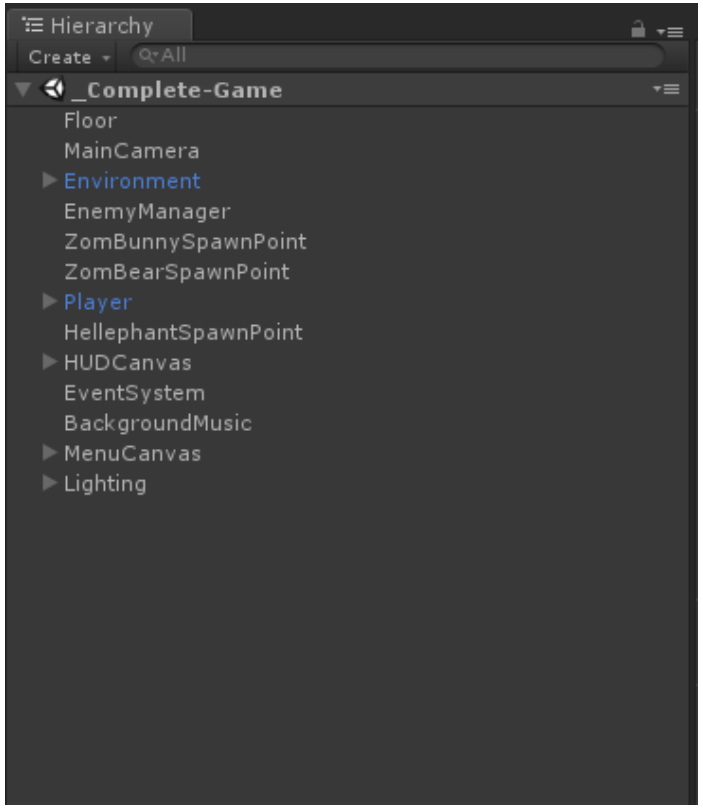
Organización



El editor de Unity

Hierarchy View

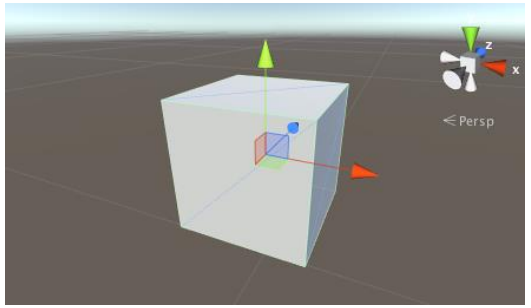
- Muestra los objetos en nuestra escena
- Se organiza jerárquicamente como un grafo de escena
  - Los objetos pueden tener relaciones padre/hijo
- El botón Create añade *GameObjets* a la escena



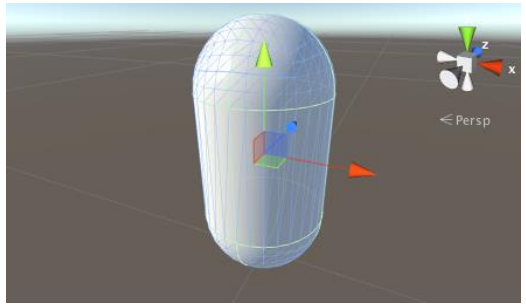
El editor de Unity

Objetos primitiva de Unity

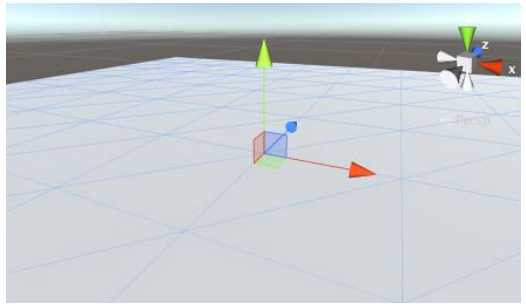
- Primitivas (3D)



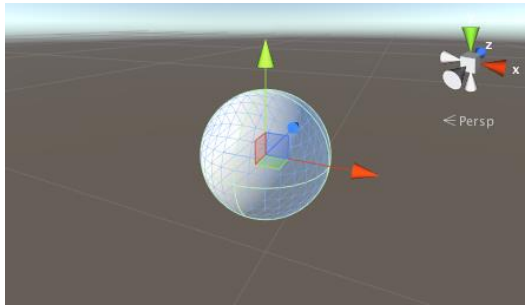
Cubo, 1x1x1



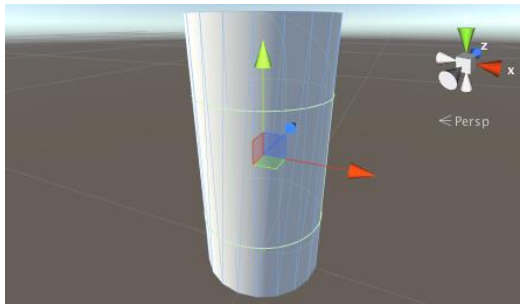
Cápsula, r=0.5, h=2



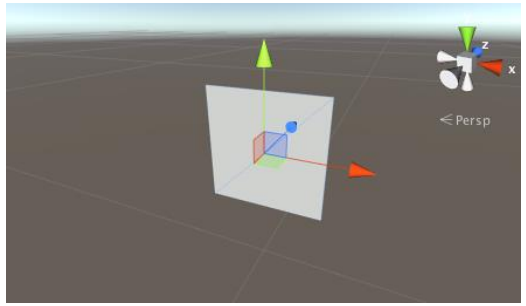
Plano, 10x10. Una cara, en XZ, 200 triángulos



Esfera, r=0.5 (diámetro = 1)



Cilindro, r=0.5, h=2

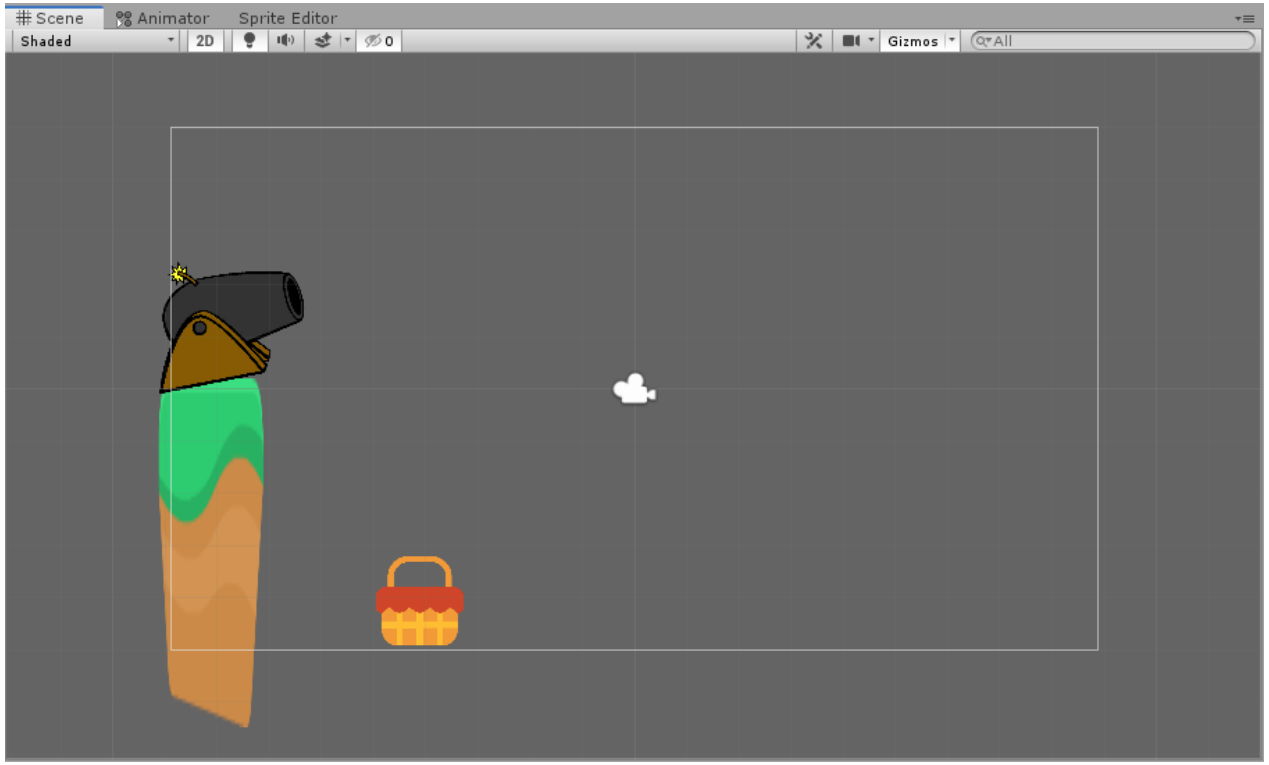


Cuadrilátero, 1x1. Una cara, en XY, 2 triángulos

El editor de Unity

Scene View

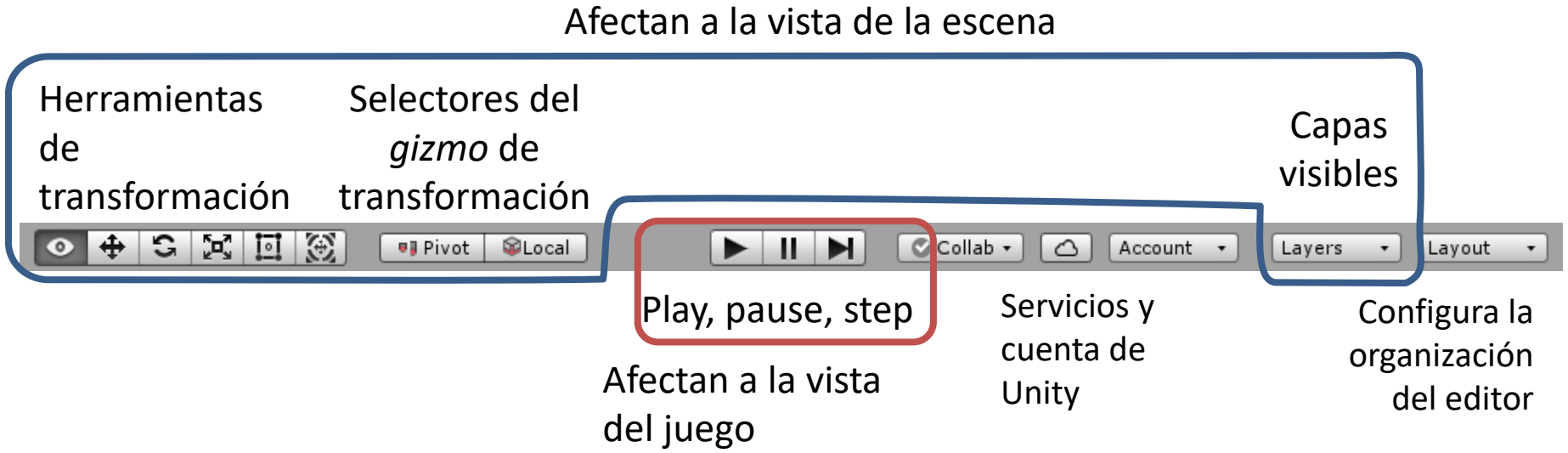
- Scene View
  - Donde construimos nuestras escenas (hechas de GameObjects)



El editor de Unity

Barra de herramientas

- Barras de herramientas



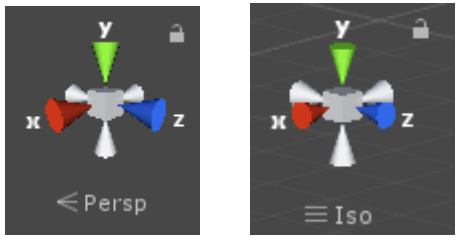
El editor de Unity

Scene View

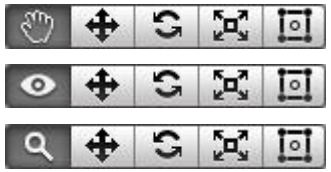
- Navegación en la vista de la escena
  - Navegación FPS:
    - Mantén pulsado el botón derecho del ratón
    - Teclas WASD (más Q y E para subir y bajar)
    - El ratón mueve la dirección de la vista
  - Los cursores para moverse en el plano XZ
  - Orbitar alrededor de un objeto
    - Alt + botón izquierdo del ratón
    - Se mueve alrededor del punto pivote
  - *Pan*
    - Alt + botón central o Botón central
  - *Zoom*
    - Rueda del ratón o Alt + botón derecho del ratón
  - Maximizar un panel
    - Mayúsculas + Espacio
  - Encontrar un GameObject
    - Seleccionarlo (por ejemplo, en la jerarquía) y pulsar F en la vista de la escena

El editor de Unity

Scene View



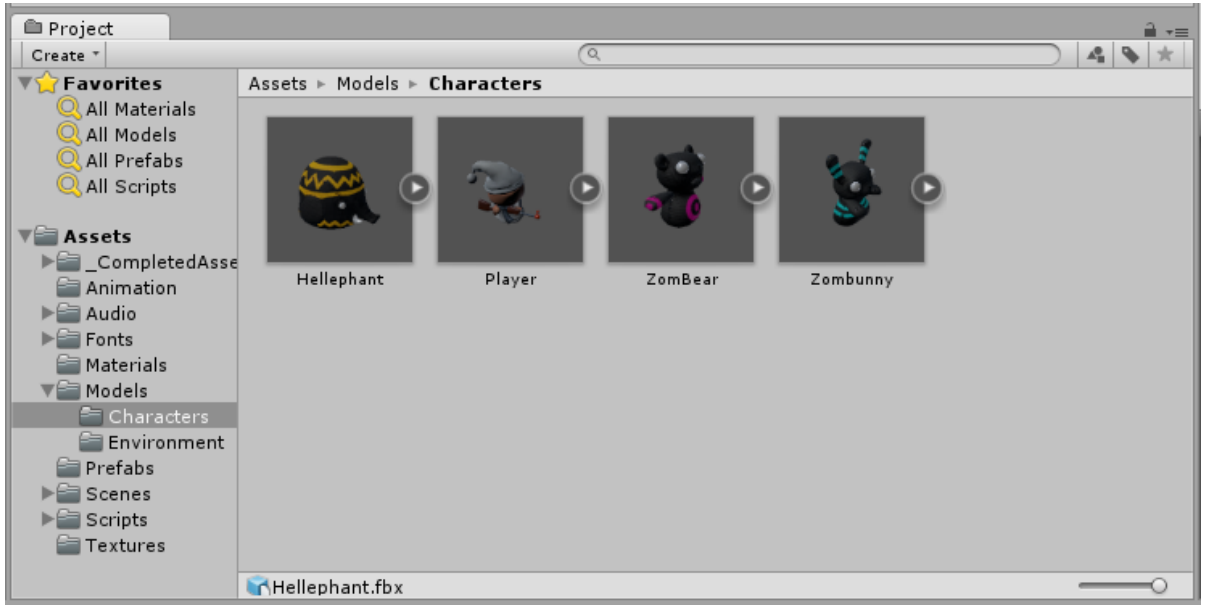
- Navegación en la vista de la escena
  - *Scene Wizmo*
    - Muestra la orientación de la cámara de la vista de la escena
    - Haz clic en una flecha para alinear la cámara con un eje principal
    - Haz clic en el cubo central o en el texto inferior para cambiar entre cámara perspectiva y paralela
    - Haz clic en el candado para bloquear la rotación de la vista
  - Barra de herramientas
    - Atajo: Q
    - Pan: Botón izquierdo del ratón
    - Orbitar: Alt + Botón izquierdo
    - Zoom: Alt + Botón derecho



El editor de Unity

Project View

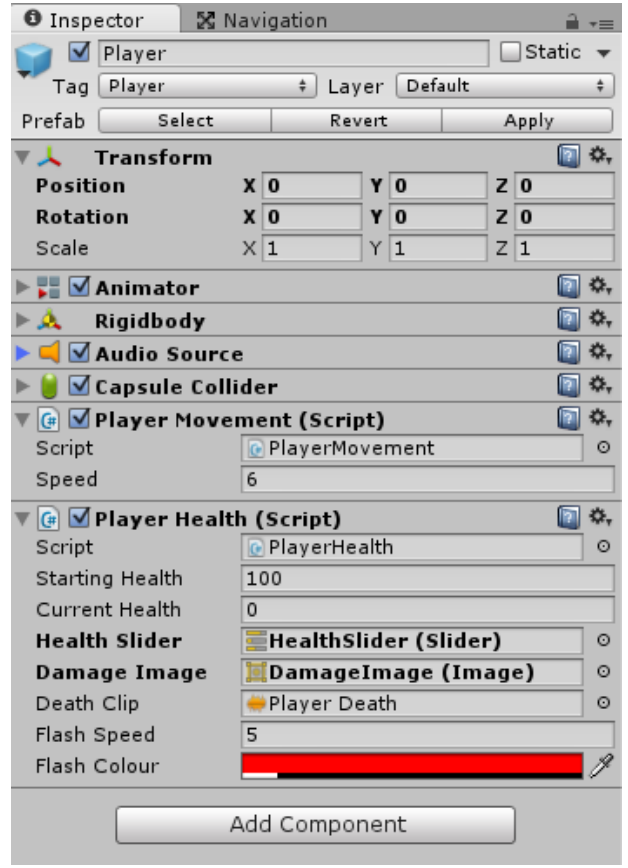
- Contiene todos los recursos (*assets*) del juego. Normalmente arrastraremos elementos desde esta vista a la vista de la escena
- Es una vista de la estructura de carpetas del proyecto en el disco
- Puede mostrar una o dos columnas, y una vista previa del recurso



El editor de Unity

Inspector

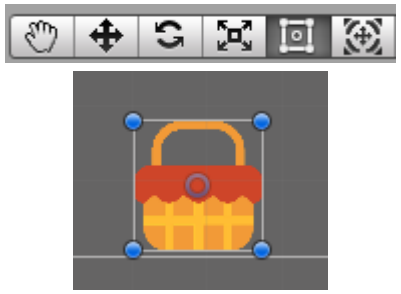
- Muestra las propiedades del GameObject seleccionado
- Los objetos se pueden seleccionar tanto en la vista de la Jerarquía como en la de la Escena



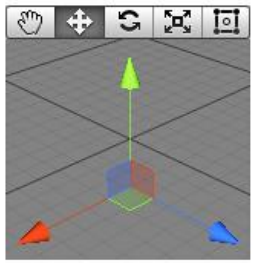
El editor de Unity

Transformaciones

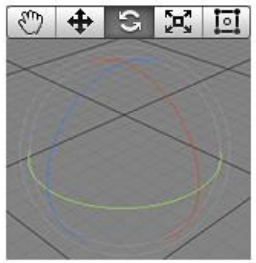
- Herramientas interactivas para las tres transformaciones básicas en la selección actual



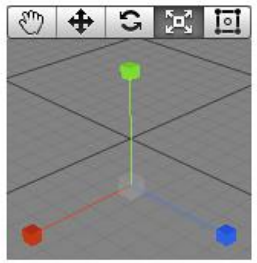
Rect (para 2D)



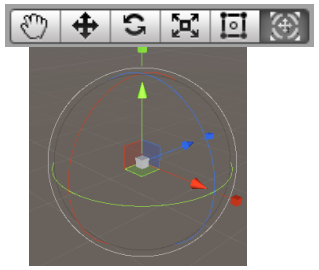
Translate (W)



Rotate (E)

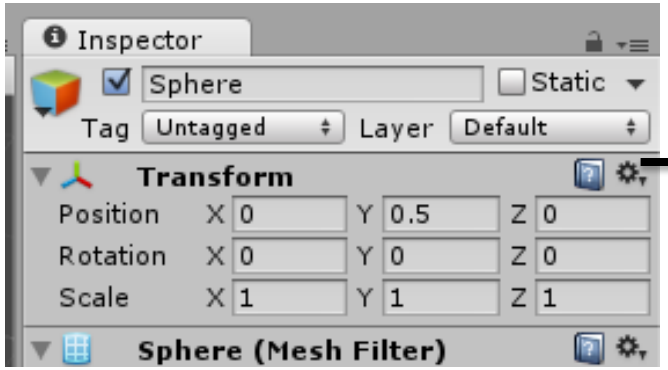


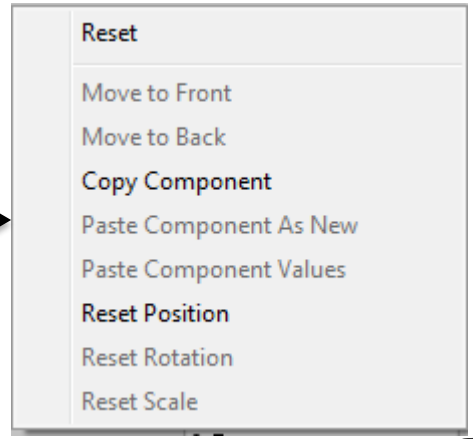
Scale (R)



Todo a la vez

- Y transformaciones precisas con el Inspector

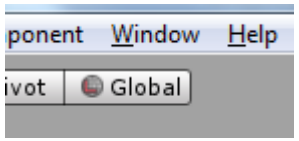


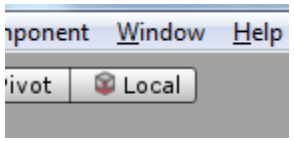


El editor de Unity

Transformaciones. Sistemas de referencia

- Podemos aplicar transformaciones en dos sistemas de coordenadas:
  - Global
  - Local



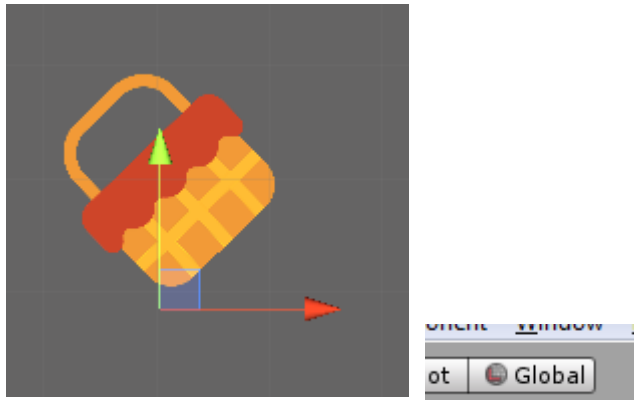


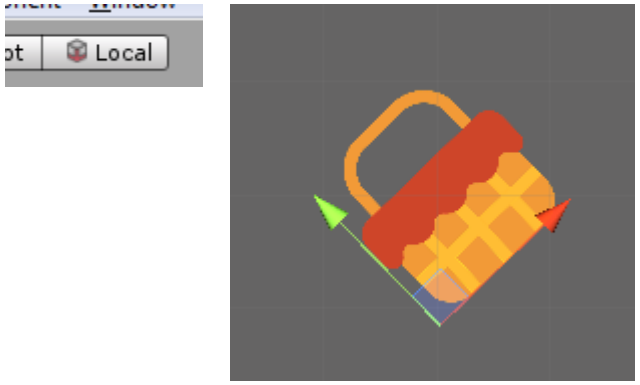
- Inicialmente, los *gameobjects* están alineados al sistema de coordenadas global (GCS)
  - El sistema de coordenada local es paralelo al global

El editor de Unity

Transformaciones. Sistemas de referencia

- Después de algunas transformaciones, el sistema local y global pueden dejar de coincidir
- Unity permite aplicar transformaciones con respecto a ambos sistemas de coordenadas

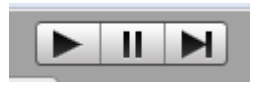




El editor de Unity

Modo de juego

- La vista del juego muestra la escena desde el punto de vista de la cámara principal

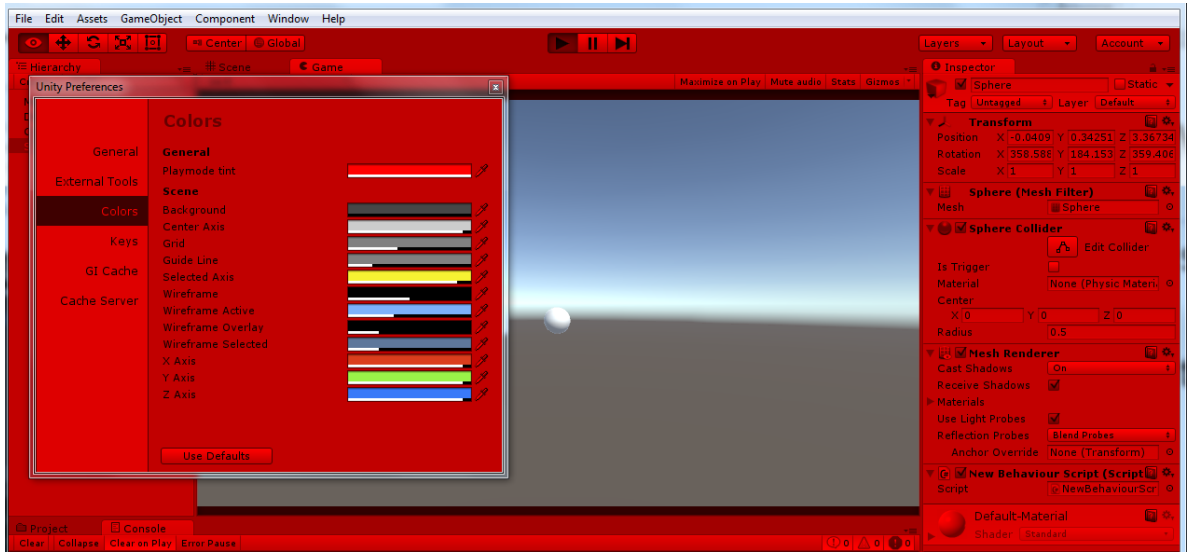


Haz clic en Play para empezar el juego

El editor de Unity

Modo de juego

- Entra en el modo de juego pulsando el botón de Play
- Se pueden modificar los elementos de la escena en el modo juego
  - Pero ¡no lo hagas! Los cambios no se pueden salvar, y se perderán
- Cambia el color del interfaz para que te avise cuando estés en modo juego para no perder los cambios

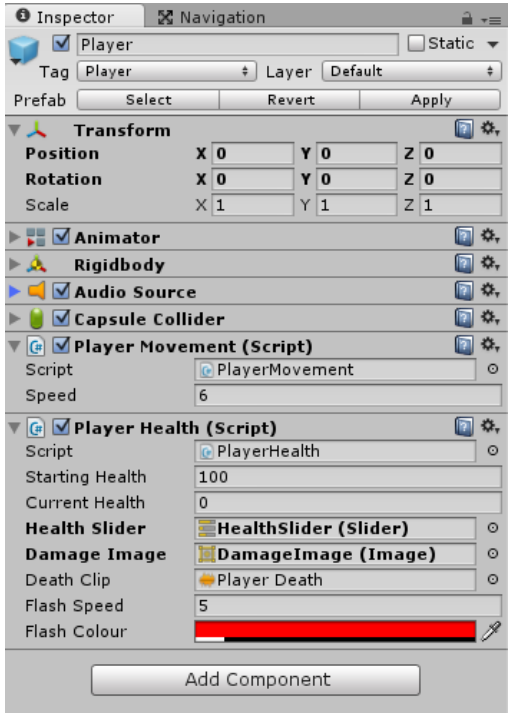


Edit\Preferences\Colors\Playmode tint

Elementos en un juego Unity

Escenas, GameObjects and Componentes

- Un juego está compuesto por diferentes escenas (niveles, pantallas de la GUI…)
- Las escenas están compuestas de GameObjects (cámaras, personajes, efectos de partículas, objetos…)
- Un GameObject es sólo un contenedor (de Componentes)
  - Los componentes son los que dan a un GameObject su comportamiento y aspecto
  - Hay muchos componentes predefinidos listos para usar
  - Se puede añadir nuevo comportamiento a los GameObjects usando un componente de tipo Script
  - El único componente común a todos los GameObjects es el Transform



Tu primera escena

Tu primera escena

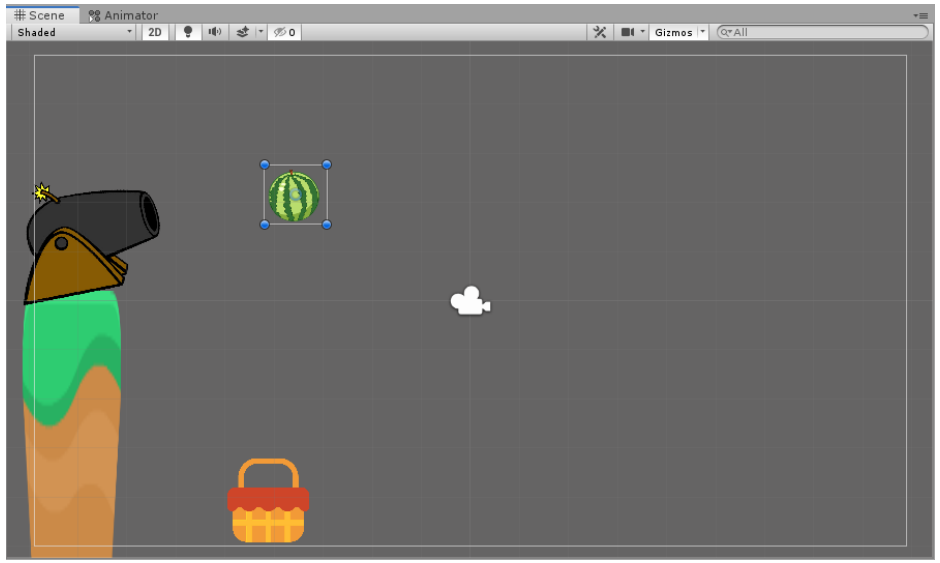
- Al crear un proyecto 2D, por defecto nuestra escena sólo tiene la cámara principal
- La escena por defecto se llama SampleScene, y se almacena en la carpeta Scenes
- Puedes guardar la escena con otro nombre más descriptivo con:
  - File\Save Scene As…

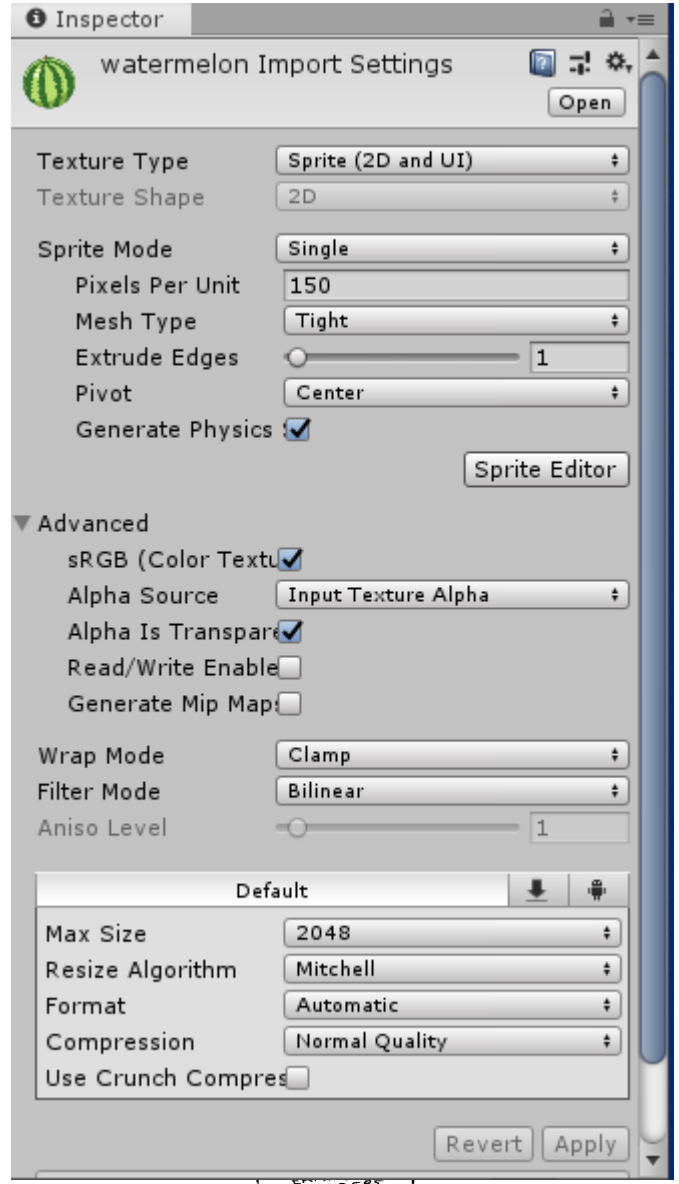
Tu primera escena

- El primer paso para construir un juego consiste en importar los recursos (assets)
- Crea un directorio llamado Sprites en el proyecto
  - Botón derecho\Create\Folder
- Dos opciones:
  - Dentro del nuevo directorio, botón derecho\Import New Asset
  - Con el explorador de ficheros de Windows, arrastra los ficheros al panel del proyecto

Tu primera escena

- Si has creado el proyecto en modo 2D, las imágenes se habrán importado en modo Sprite (2D and UI)
- Selecciona los sprites y comprueba su tipo en el Inspector
- Puedes arrastrar sprites a la escena desde el panel de proyecto





Tu primera escena

- Al importar un sprite, podemos especificar el tamaño que queremos que tenga en la escena
- En el Inspector, define en Pixels Per Unit el tamaño adecuado para dar a cada objeto su tamaño deseado en pantalla
  - Por ejemplo, si Pixels Per Unit (PPU) está a 100, estamos indicando que 100 píxeles equivalen a una unidad de Unity
  - El volumen de la cámara ortográfica por defecto tiene 10 unidades de alto
- Después de cambiar el PPU hay que pulsar en Apply

Scripts C#

Scripts como componentes

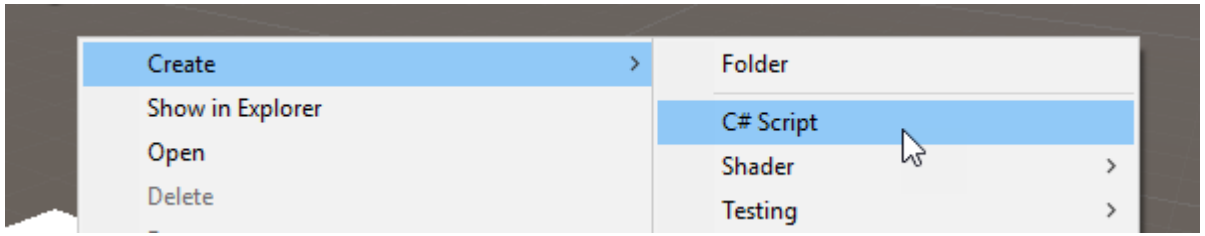
- Al iniciar una aplicación en Unity, se ejecuta el código asociado a los GameObjects de la escena
- Cada GameObject es una colección de componentes
  - Y los scripts pueden ser componentes
- Por ello, todo script que se desee ejecutar en Unity debe ser un componente de un GameObject
  - Para que un script C# pueda ser un componente, debe heredar de MonoBehaviour
- En Windows, usaremos Visual Studio para escribir los scripts
  - Hasta la versión 2018, Unity incluía el editor MonoDevelop

Scripts C#

Scripts como componentes

- Crea una nueva carpeta llamada Scripts en la carpeta de Assets
- Crea un Script C# y ábrelo con doble clic

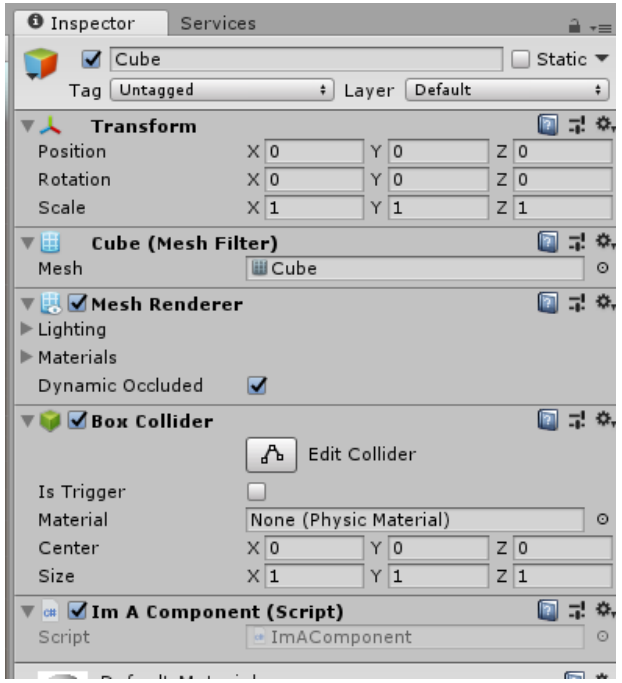
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ImAComponent : MonoBehaviour {
  // Use this for initialization
  void Start () {

  }
  // Update is called once per frame
  void Update () {

  }
}
```



Scripts C#

Scripts como componentes

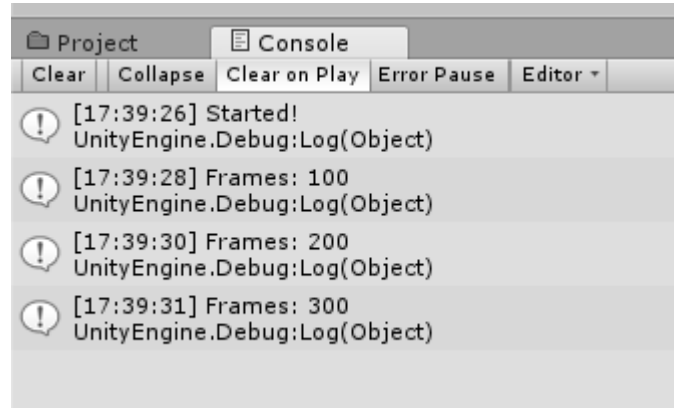
- Una vez se guarda un script, lo podemos asociar a cualquier GameObject como si fuera un componente más
  - Arrástralo sobre el GameObject



Scripts C#

Scripts como componentes

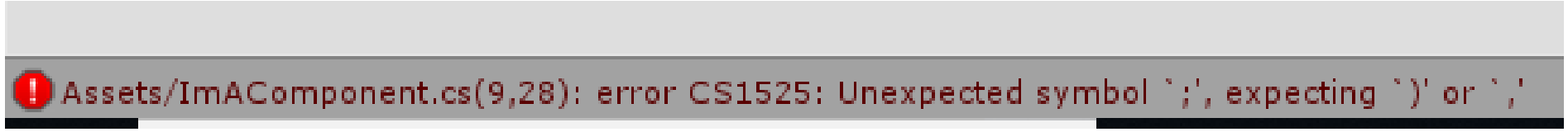
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ImAComponent : MonoBehaviour {
  private long frameCounter;

  void Start () {
    Debug.Log("Started!");
    frameCounter = 0;
  }

  void Update () {
    if (++frameCounter % 100 == 0) {
      Debug.Log("Frames: " + frameCounter.ToString());
    }
  }
}
```



Scripts C#

Errores

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

public class ImAComponent : MonoBehaviour {
  private long frameCounter;

  void Start () {
    Debug.Log("Started!";
    frameCounter = 0;
  }
[…]
```



Tu primera escena

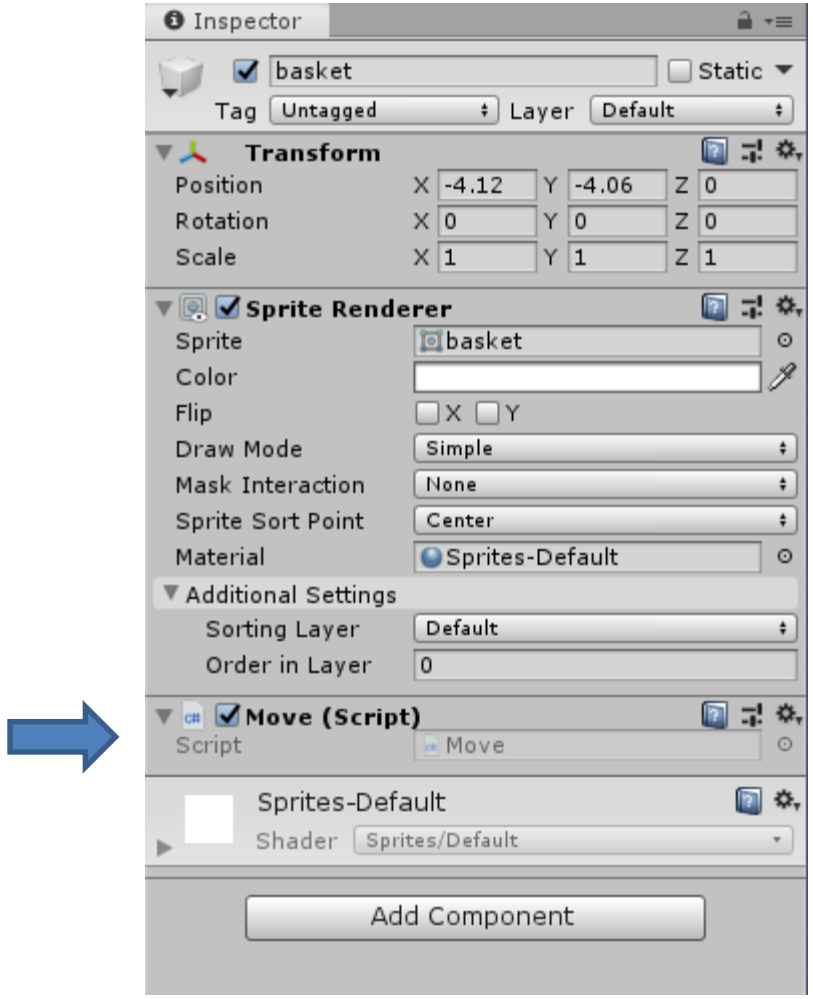
- Vamos a hacer que el jugador pueda mover la cesta con las teclas:
  - Dentro del directorio Scripts, botón derecho\Create\C# Script
  - Llámalo Move
  - Haz doble clic sobre su icono para editarlo con Visual Studio

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

public class Move : MonoBehaviour {
  // Start is called before the first frame update
  void Start() {
  }

  void Update() {
    transform.Translate(new Vector2(Input.GetAxis("Horizontal") * 0.1f, 0.0f));
  }
}
```


Tu primera escena

- Guarda el script desde Visual Studio
- Arrástralo sobre la cesta
- Ejecuta el juego, y mueve la cesta lateralmente
  - Usando A/D, los cursores Izquierda y Derecha, o
  - las teclas de dirección de izquierda y derecha en un gamepad

- ¿Qué pasa con la cesta?



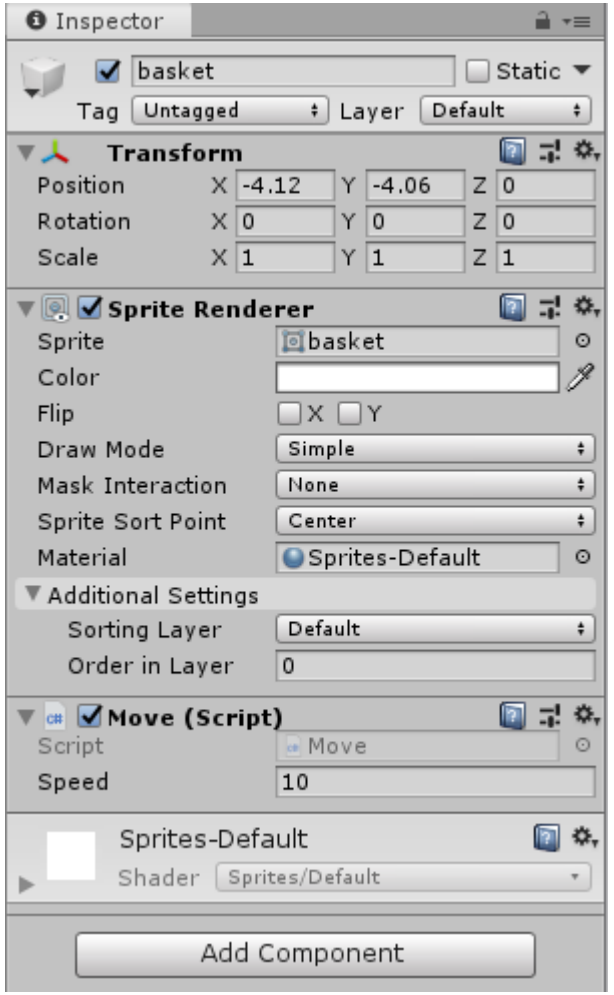
Tu primera escena

- `GetAxis` devuelve valores en el rango [-1,+1], y la función `Update` se ejecuta una vez por frame
  - Por lo tanto, el script anterior mueve la cesta a una velocidad que depende del número de frames por segundo que es capaz de dibujar el sistema
  - Por defecto, las unidades de medida de Unity son metros y segundos
- ¿Cómo podemos mover a una velocidad constante (en m/s), independientemente de la velocidad de refresco del dispositivo final?
  - `Time.deltaTime`, devuelve el tiempo en segundos desde el frame anterior (la última vez que se ejecutó el método Update() del GameObject)
  - Es muy importante en la animación cinemática, donde tenemos que actualizar el componente Transform de todos los objetos en movimiento
- Para mover N m/s a lo largo de un eje:
  - `N * Input.GetAxis(<direction>) * Time.deltaTime`

Tu primera escena

- Cambiar el script y recompilar para encontrar la velocidad adecuada es un proceso lento
  - Unity conecta las variables públicas de los scripts con el Inspector, para facilitar el prototipado

```
public class Move : MonoBehaviour {
  public float speed = 10.0f;
  void Update() {
    transform.Translate(new Vector2(
        Input.GetAxis("Horizontal") * Time.deltaTime * speed, 0.0f));
  }
}
```



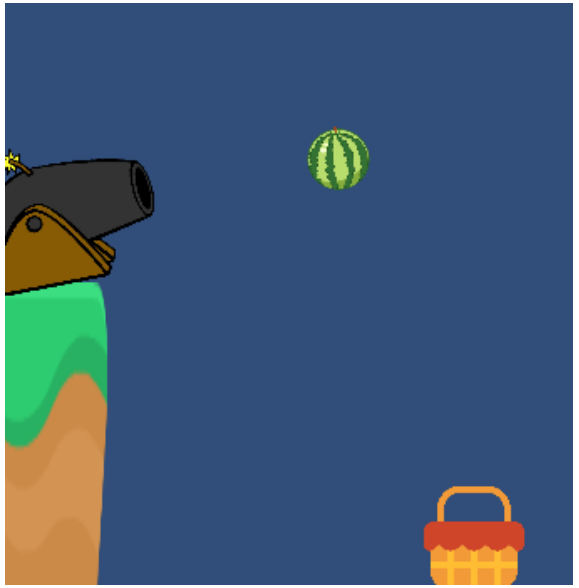
Tu primera escena

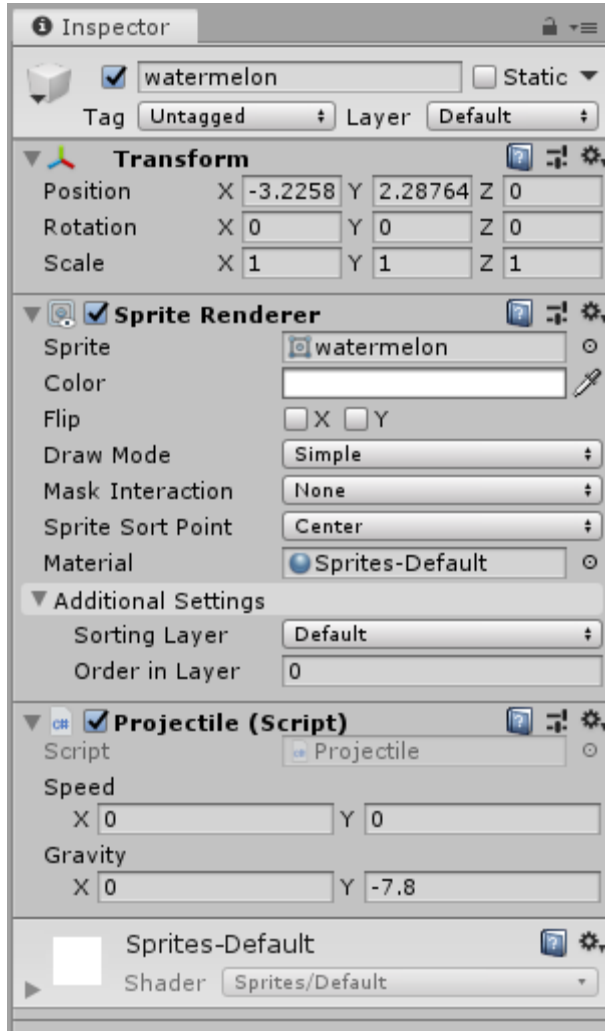
- También podemos limitar el movimiento de la cesta a la zona de juego

```
public class Move : MonoBehaviour {
  public float speed = 10.0f;
  public float leftBorder, rightBorder;
  void Update() {
    transform.Translate(new Vector2(Input.GetAxis("Horizontal")  * Time.deltaTime * speed, 0.0f));
    if (transform.position.x < leftBorder)
      transform.position = new Vector2(leftBorder, transform.position.y);
    else if (transform.position.x > rightBorder)
      transform.position = new Vector2(rightBorder, transform.position.y);
  }
}
```

Tu primera escena

- Vamos a crear un proyectil que siga un tiro parabólico
- Añade una sandía a la escena
- Crea el script de la siguiente transparencia, e insértalo a la sandía





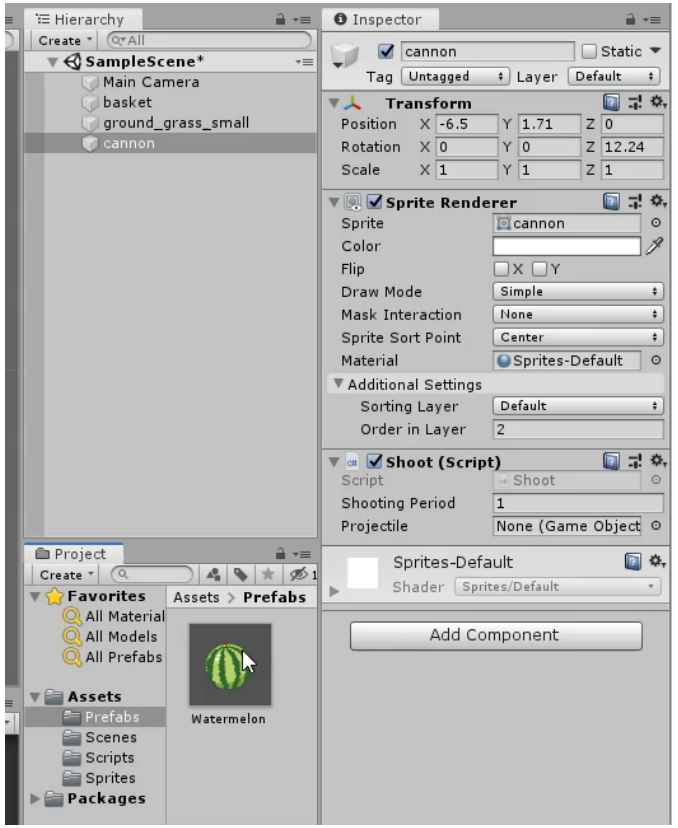
```
public class Projectile : MonoBehaviour {
  public Vector2 speed;
  public Vector2 gravity = new Vector2(0.0f, -7.8f);
  GameObject basket;
  static int counter = 0;
  void Start() {
    basket = GameObject.Find("basket");
  }
  void Update() {
    transform.Rotate(0.0f, 0.0f, -360.0f * Time.deltaTime);
    speed += gravity * Time.deltaTime;
    Vector2 off = speed * Time.deltaTime;
    transform.position = transform.position + new Vector3(off.x, off.y);
    float distanceToBasket = (transform.position - basket.transform.position).magnitude;
    if (distanceToBasket < 0.5f) {
      counter++;
      Debug.Log("Catched!: " + counter.ToString());
      Destroy(gameObject);
    } else if (transform.position.y < -15) {
      Destroy(gameObject);
    }
  }
}
```

Tu primera escena

- Ajusta la velocidad inicial y la gravedad de la sandía hasta conseguir el efecto deseado
- Para crear game objects en tiempo de ejecución, hay que construir un prefab:
  - Crea una carpeta llamada Prefabs y arrastra la sandía (la que tiene el script en la escena)
  - Puedes borrar la sandía de la escena

Tu primera escena

- Vamos a hacer que el cañón dispare sandias
  - Añádele el siguiente script:




```
public class Shoot : MonoBehaviour {
  public float shootingPeriod = 1.0f;
  public GameObject projectile;

  private void Start() {
    InvokeRepeating("Fire", 0.0f, shootingPeriod);
  }
  void Fire() {
    Projectile obj = Instantiate<GameObject>(projectile).GetComponent<Projectile>();
    obj.transform.position = transform.position;
    obj.speed = new Vector2(Random.Range(6, 12), Random.Range(-1, 0));
  }
}
```

Bibliografía

- Unity online manual
  - http://docs.unity3d.com/Manual/index.html
- S. J. Gortler. Foundations of 3D Computer Graphics. The MIT Press, 2012
- J. Hocking. Unity in action : multiplatform game development in C#. 2ª ed. Manning, 2018
  - Recursos:
    - http://apworldipedia.com/index.php?title=File:Canon_2.jpg
    - https://www.hiclipart.com/free-transparent-background-png-clipart-yvmmp/download
    - www.kenney.nl
    - https://www.svgrepo.com/svg/297747/picnic-basket-picnic

creative commons
COMMONS DEED

Documentación generada por
Grupo de Informática Gráfica
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Reconocimiento-NoComercial-CompartirIgual 2.5
Usted es libre de:

copiar, distribuir y comunicar públicamente la obra
hacer obras derivadas bajo las condiciones siguientes:

Reconocimiento. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador.

No comercial. No puede utilizar esta obra para fines comerciales.

Compartir bajo la misma licencia. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.