TUTORIAL PRIMEROS PASOS CON UNITY.

DESCRIPCIÓN DEL EDITOR.

- **Jerarquía** (hierarchy):
  - Escena.
  - Cámara y Luz.
  - Propiedades de los objetos. (inspector): Apiladas en categorías.
- **Proyecto** (Project): Componentes del proyecto, que puedo subir a la escena. (assets).
- **Consola**: Irán apareciendo mensajes o podremos ponerlos nosotros.
- **Escena** o Juego (scene, game): En escena vamos diseñando, y en Juego vamos viendo cómo queda aproximadamente.
  - Al pulsar Play, es conveniente hacer un cambio de color del fondo (por defecto no lo incorpora), ya que, si mientras estás en Play, los cambios que hagas en Scene, no quedan guardados. Así, lo percibes. Ese cambio se realiza así:
    - Edit→Preferences→Color→Play Mode tint.

EJEMPLO INICIAL

Crearemos un personaje.

- **Game Object**→3D→Capsule: Solemos representar así al personaje principal. (posteriormente se le añadirán texturas).
  - Desplazaros con el ratón, hacer zoom con la rueda, probad a arrastrar sin ALT, con ALT, con SHIFT, con el botón derecho, etc.
  - Doble clic en el objeto (parte de la jerarquía), te lleva a ese objeto.
  - Puedes pinchas directamente en un eje, para tener una vista desde un ángulo concreto.
  - Dentro de las propiedades, que ya iremos viendo, una muy importante es el Collider (periferia de reacción al impactar contra otro objeto). Según sea el objeto (neutro, amigo, enemigo, etc), puede hacerme algún efecto negativo o no. Se puede ampliar que no sea solamente la que tienen el propio objeto, aumentando o disminuyendo la zona de influencia.
  - Cambiar el nombre de la cápsula (Personaje, por ejemplo).
  - Mover: cambiando coordenadas, o activando la herramienta Move Tool y arrastrando cuando sale la doble flecha al ir a las Coordenadas en la ventana propiedades. También se puede mover, tirando directamente de los ejes (así, te aseguras que sólo desplazas en ese eje). Si pincho en el centro, lo muevo en cualquier dirección.
  - Probad el resto de herramientas de esa barra: Rotar, Escalar, Punto de rotación (más adelante se entenderá).
  - Probad también que se puede mover o girar la cámara. Darle al play para ver lo que saldría.

- Añadir ahora 3 cubos (puedes crear uno y copiarlo o duplicarlo). Verás como si utilizas la tecla V, si los intento unir, quedan como "pegados".
- Si creo un Emty, lo llamo "Ladrillo" y meto los 3 cubos, pasan a ser un solo objeto.
- Este Ladrillo yo ya podría copiar y pegarlo, pero cuando un objeto se va a reutilizar muchas veces, lo mejor es hacer un Prefab, para que así, si haces cambios en el Prefab, se modifiquen todos los objetos de ese tipo que hayas utilizado por las escenas, si te interesa que se modifiquen y no tener que ir luego uno a uno.

Para ello es mejor empezar a organizar bien nuestro proyecto. Dentro de Assets, crearemos 3 carpetas:

  - Prefabs.
  - Scripts.
  - Materials.

- **Prefab**: Añadid el objeto dentro de la carpeta Prefabs. Veremos cómo hasta le cambia el aspecto. Ya arrastrando desde los Prefabs a la escena, puedo incorporar los que necesite.
  Ahora vamos a darles textura, para ello en:
- **Materials**: Dentro de la carpeta Materials, crear un material, por ejemplo, rojizo metálico (en el menú Assets o con botón derecho, créate→material.
  - Desde ahora arrastrando el material al elemento que quieras.
  - No se aplica a todo el Prefab. Para ello tendrías que editarlo (con doble clic) o haberlo creado ya así. Si editas un Prefab, cambian los que añadiste anteriormente. (dando a la flechita de la izquierda cuando estás editando un Prefab, se vuelve a la escena y se guarda la edición realizada).
  - Ahora probaremos a crear una textura a partir de una imagen. Busquemos unos ladrillos (ojo que sean de licencia que permita su uso). Los arrastramos a la carpeta Materials. (verás que te crea un nuevo material que puedes arrastrar a los objetos)
  - Aunque esa imagen se puede usar y unity crearía un material temporal, lo mejor es asignarlo a un material que tú crees para poder tener todas las propiedades. Por ejemplo, arrastra ese ladrillo al albedo del material que creaste (renombra el material que se llame ladrillo)

  Ahora haremos un pequeño Script.
- **Scripts**: Lo primero comprobar que tenemos vinculado el Visual Studio como editor (Preferences→External Tools). Si no lo tienes vinculado, al dar doble clic en el script, te lo tratará de abrir y puedes decir pulsando en Siempre, que ya te lo estavblezca vinculado por defecto.
  - Crearemos un script que se llame mover, al darle doble clic nos tiene que abrir el editor.

  Problemas que pueden surgir al instalar por primera vez VS Code:

- Instalar los 2 paquetes que recomienda VS Code.
- Si no localiza el SDK, descargarlo desde el enlace que recomienda. Cerrar VS Code e instalar el SDK. Ya debería funcionar y no dar errores.
    - Para crear un script el nombre de la clase debe de heredar de MonoBehaviour (al crear el script elige ese tipo y ya et crea el script de esa manera y con los eventos Start y Update.
    - Luego se puede poner código en el evento Start() se ejecuta al inicarse, Update() en cada frame etc.
    - Añade esto a Start(). Arrastra el script al Personaje (con eso se le asigna, se puede hacer también desde el inspector). Luego ejecuta y verás cómo se ha desplazado un poco la cápsula respecto a la posición inicial.

```
transform Translate (0,2,2); //al objeto que le asignes este script, se mueve 2 desde donde esté en el eje Z.
```

    - Has podido comprobar que al guardar el script y volver a Unity, se compila.
    - Ahora añadir unas variables **públicas** para controlar la velocidad. Veremos que al ser públicas se muestran como propiedades en Unity y puedes cambiar el valor en cada ejecución.
    - Se hace así:

```
using UnityEngine;

public class Mover : MonoBehaviour
{
    public float velX = 1f; // Velocidad en el eje X
    public float velY = 1f; // Velocidad en el eje Y
    public float velZ = 1f; // Velocidad en el eje Z

    // Start is called before the first frame update
    void Start()
    {
        // Mueve el objeto solo una vez cuando comienza el juego (solo en el primer frame)
        transform.Translate(velX, velY, velZ);
    }

    // Update is called once per frame
    void Update()
    {
        // Mover el objeto constantemente según la velocidad especificada en cada eje
        transform.Translate(velX * Time.deltaTime, velY * Time.deltaTime, velZ * Time.deltaTime);
    }
}
```

`Time.deltaTime` es una propiedad en Unity que representa el tiempo (en segundos) que ha pasado desde el último frame (cuadro) renderizado. Es muy útil para hacer que los movimientos, animaciones y efectos sean independientes de la tasa de fotogramas (FPS), lo que asegura que el juego funcione igual en cualquier dispositivo o configuración de hardware.

Veremos como ahora no para de moverse.

Probad ahora haciendo otro script para rotar (con Rotate)

```
using UnityEngine;

public class Rotar : MonoBehaviour
{
    public float velX = 30f; // Velocidad de rotación en el eje X
    public float velY = 30f; // Velocidad de rotación en el eje Y
    public float velZ = 30f; // Velocidad de rotación en el eje Z

    // Start is called before the first frame update
    void Start()
    {
        // Mueve el objeto una vez en la rotación inicial cuando
comienza el juego
        transform.Rotate(velX, velY, velZ);
    }

    // Update is called once per frame
    void Update()
    {
        // Rota el objeto continuamente según la velocidad en cada eje
        transform.Rotate(velX * Time.deltaTime, velY * Time.deltaTime,
velZ * Time.deltaTime);
    }
}
```

- Ver que puedo:
  - activar o desactivar los scripts desde Unity.
  - cambiar las propiedades.
  - resetear el script.
  - Cambiar propiedades en tiempo de ejecución (con la flechita)
- Añadid el mismo script a otro objeto cambiar propiedades y veréis que a cada uno le afecta de forma diferente, aunque compartan script de origen. (yo por ejemplo se lo he añadido al ladrillo y al cubo central del ladrillo y así cada uno va rotando)

- **Add component**: Veremos un ejemplo. Añadir componente RigidBody con gravedad, a algún elemento y elevadlo respecto del origen. Veremos que al ejecutar caen. Se pueden modificar parámetros de rozamiento, masa etc.

  Si no quieres que caiga de manera infinita, pon un plano como suelo. El objeto caerá hasta ese límite. Ponle un material al plano. ¿por qué no cae el objeto? Investiga por qué mirando sus propiedades y explícalo al resto de la clase.