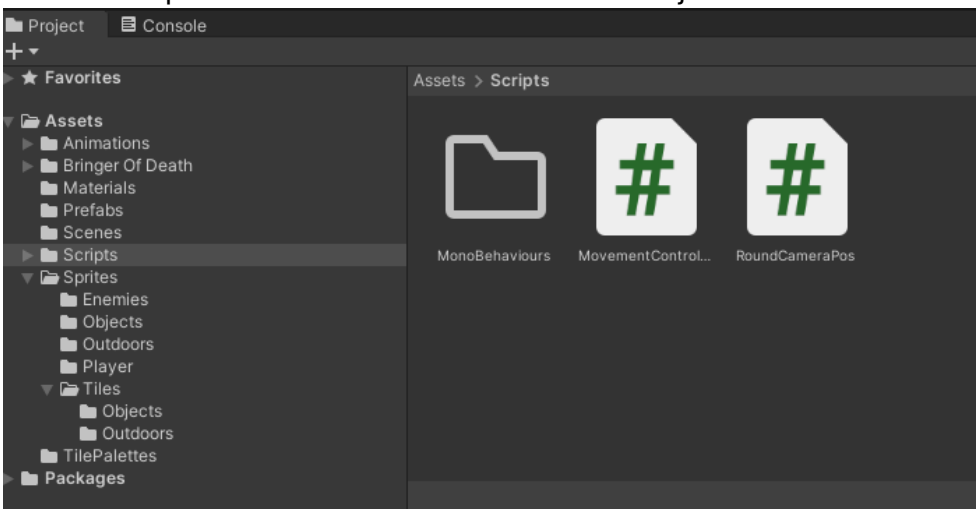
Hay ciertos rasgos que cada personaje "vivo" de nuestro juego tendrá, como el concepto de salud. Los puntos de vida o "puntos de golpe" se utilizan para medir cuánto daño un personaje puede tomar antes de morir.

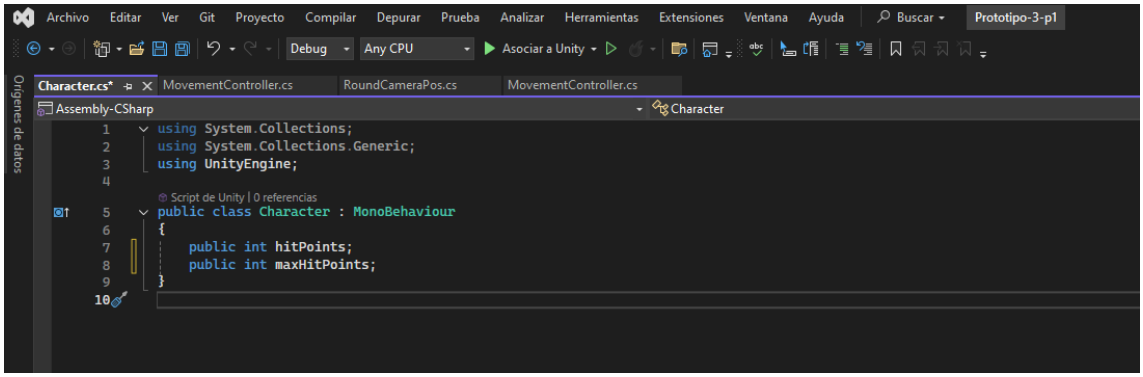
Creamos una nueva carpeta en **Scripts** llamada **MonoBehaviours** para organizar mejor los scripts que contienen comportamientos personalizados asociados a GameObjects.



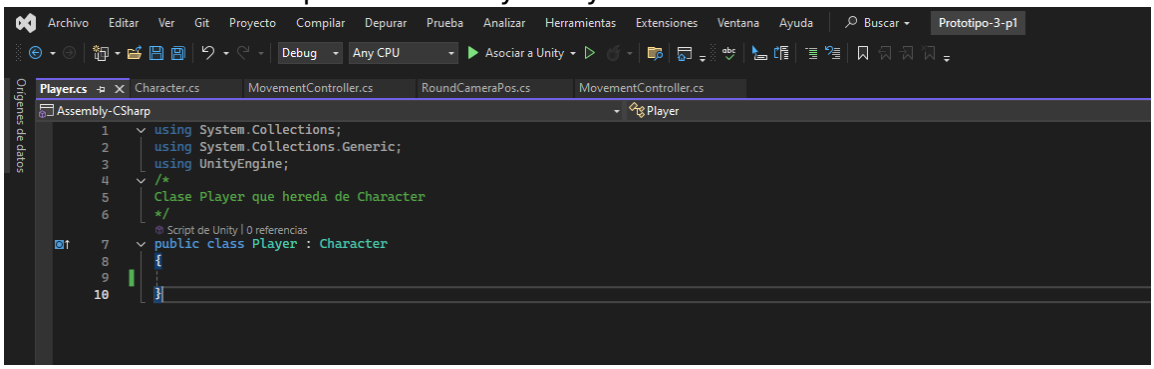
Movemos el script **MovementController.cs** a la carpeta **MonoBehaviours**, ya que hereda de **MonoBehaviour**.

Dentro de la carpeta **MonoBehaviours**, creamos un nuevo script en C# llamado **Character.cs**. Luego, hacemos doble clic en el script **Character.cs** para abrirlo en nuestro editor de código (como Visual Studio o cualquier otro configurado).

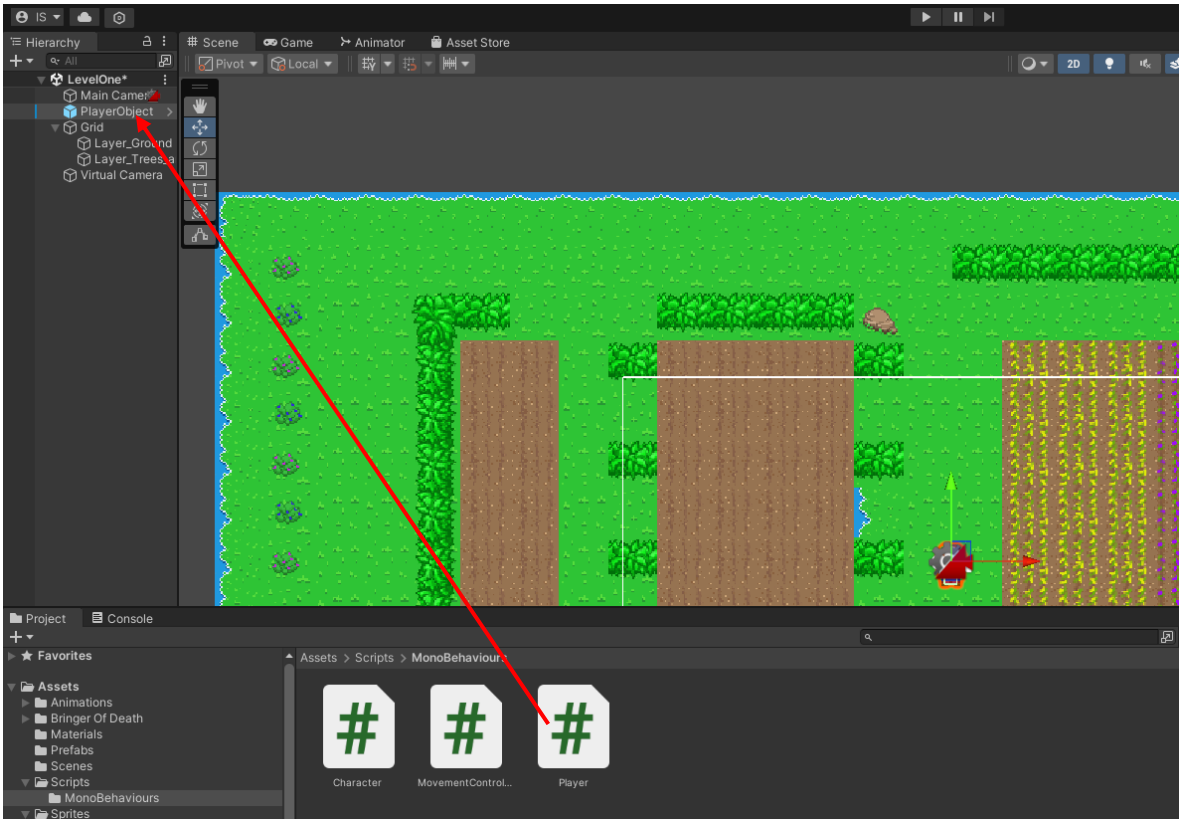
Modificamos el archivo fuente



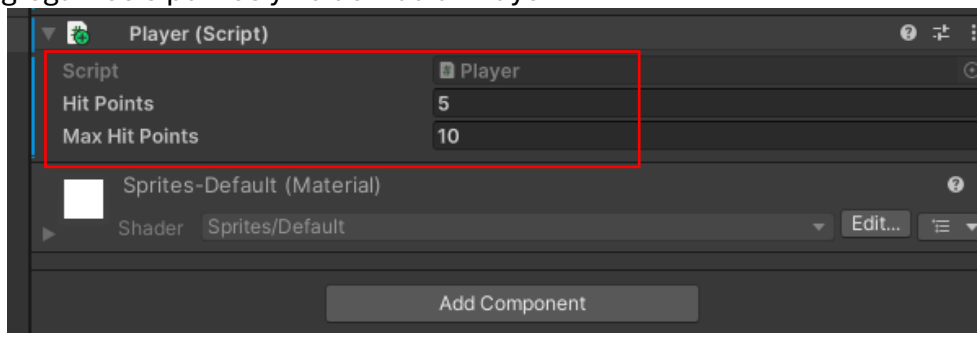
De igual manera creamos un script llamado Player.cs y lo modificamos



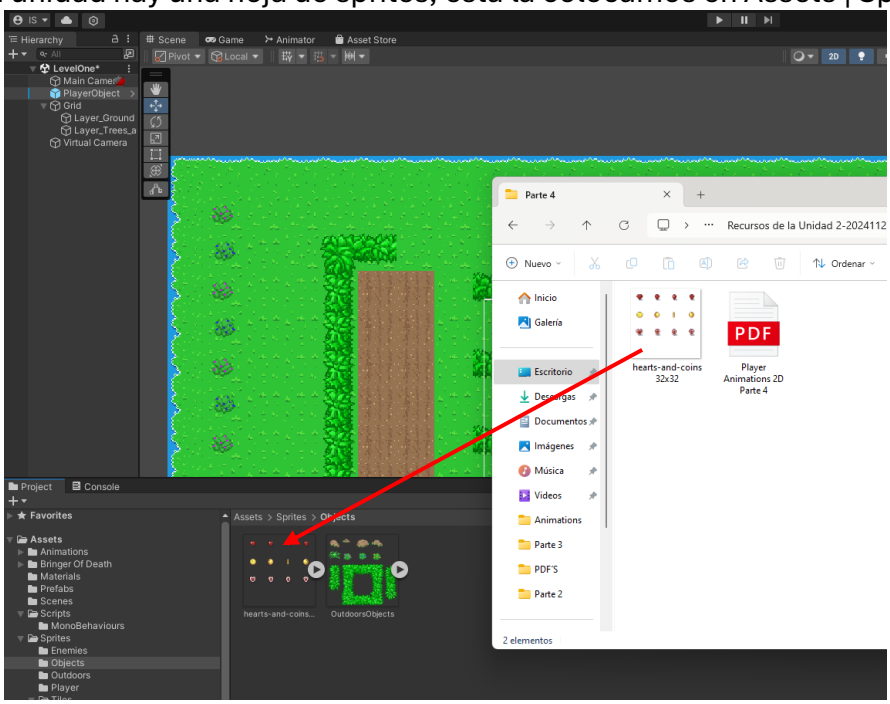
Arrastramos el script al PlayerObject de Hierarchy



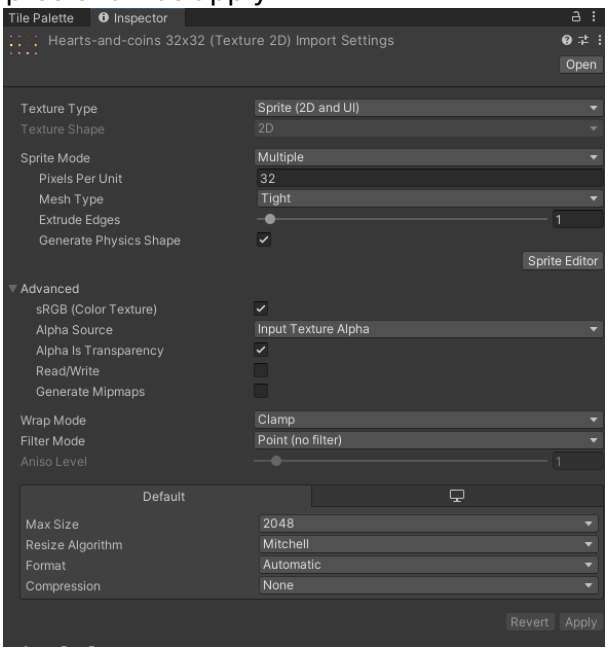
En el inspector, agregamos 5 puntos y 10 de vida al Player



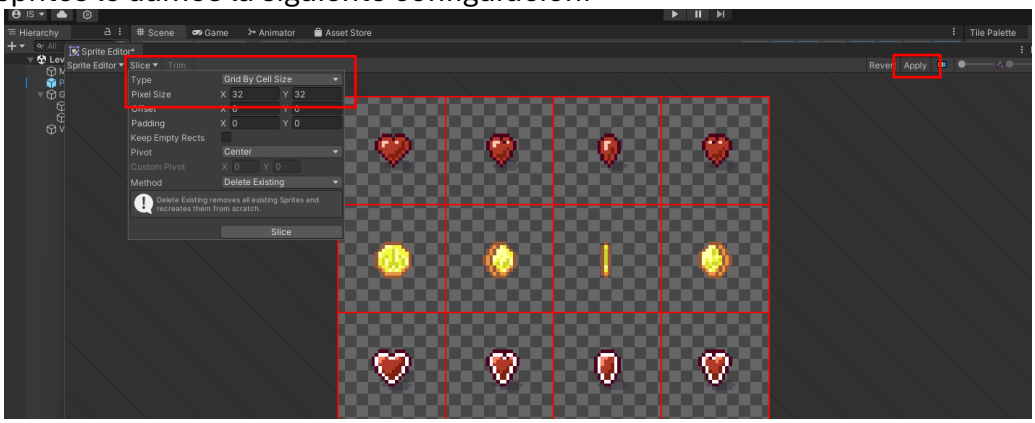
En los recursos de la unidad hay una hoja de sprites, esta la colocamos en Assets | Sprites | Objects



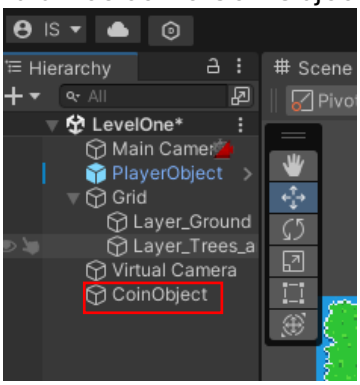
Ya que tenemos el Sprite, lo seleccionamos y en la ventana de Inspector damos la siguiente configuración y cundo se ve como la imagen, presionamos apply



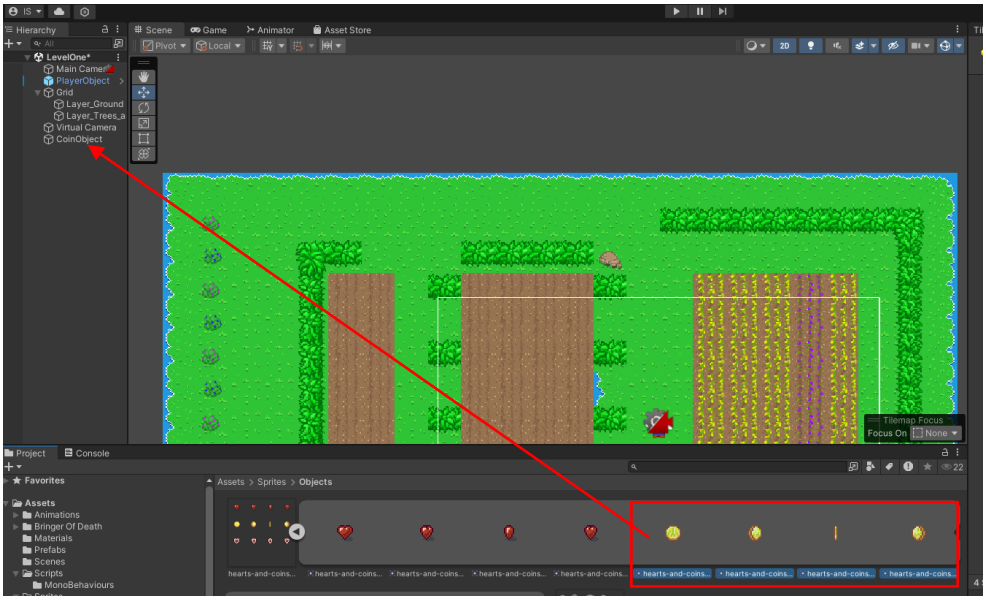
En el editor de sprites le damos la siguiente configuración.



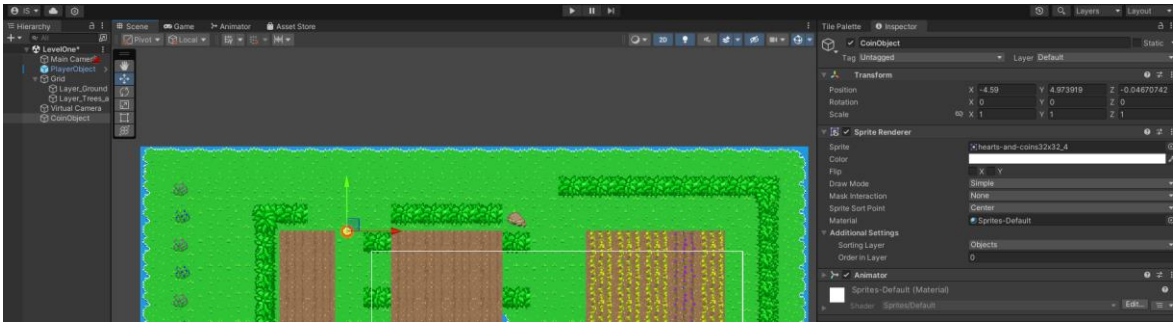
Ya que tenemos esa configuración, presionamos el botón Slice y finalmente en apply.
Ahora creamos un GameObject y lo nombramos como CoinObject



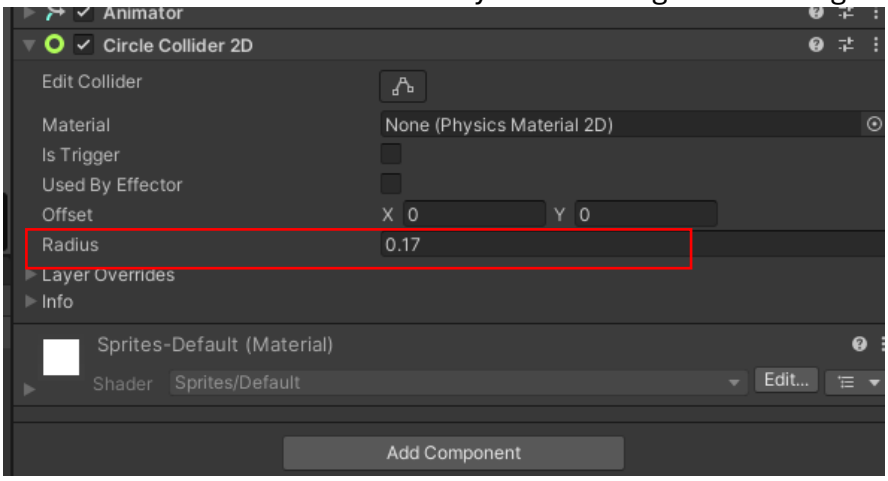
Seleccionamos los 4 sprites de moneda que se crearon y los arrastramos al objecto coin que creamos, esto nos creara una animación, la guardamos como coin-spin



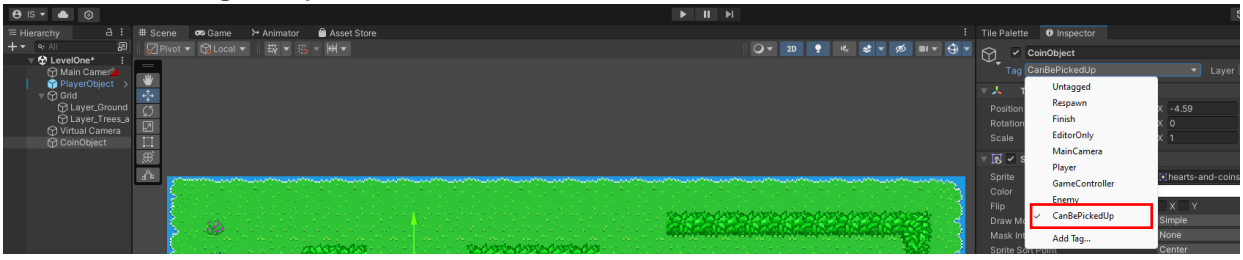
Presionamos sobre CoinObject y en Sprite elegimos el de coin-and-hearts y en Sorting layer establecemos como object



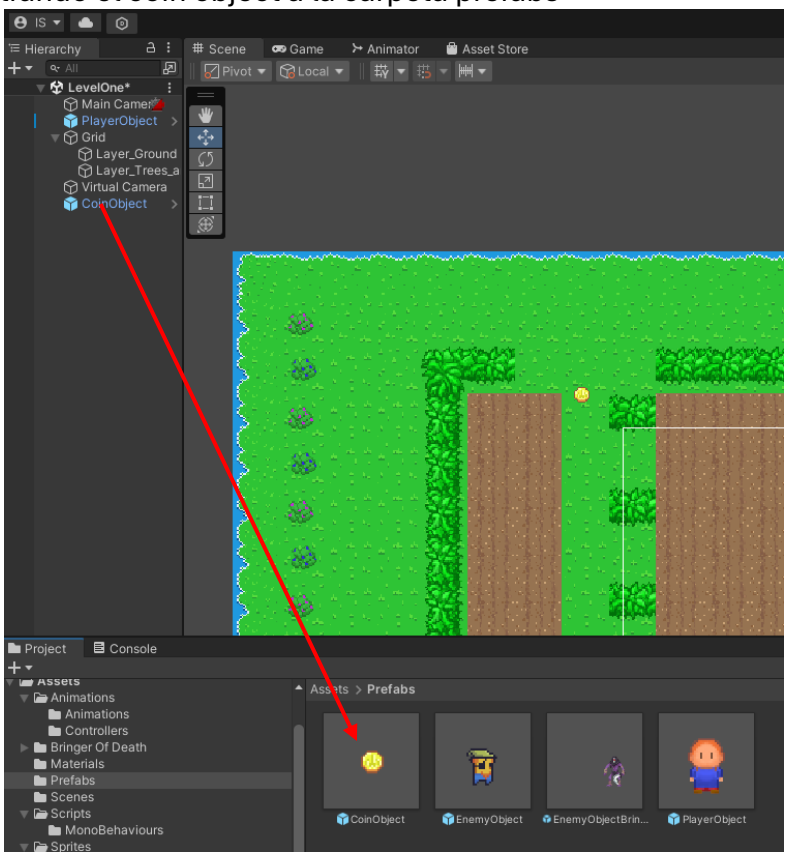
Agregamos un componente llamado Circle Collider 2D y damos la siguiente configuracion



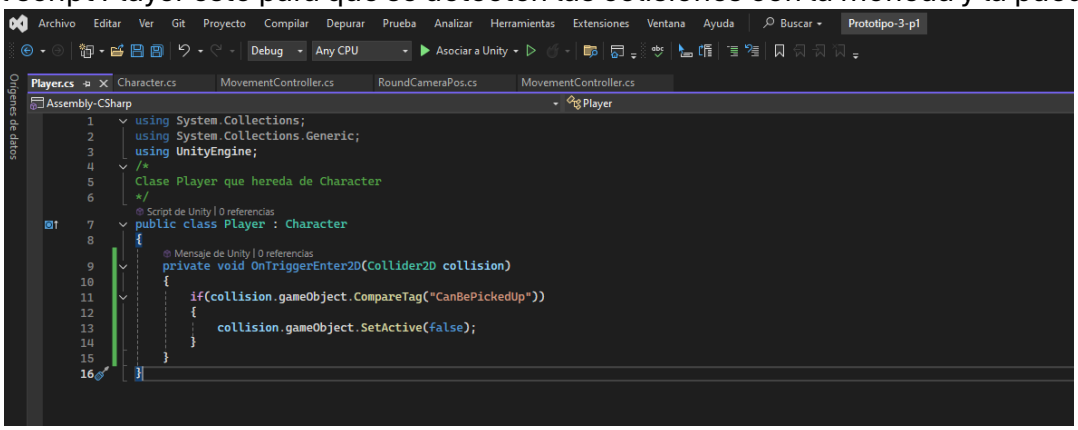
Agregamos un nuevo tag al objeto coin



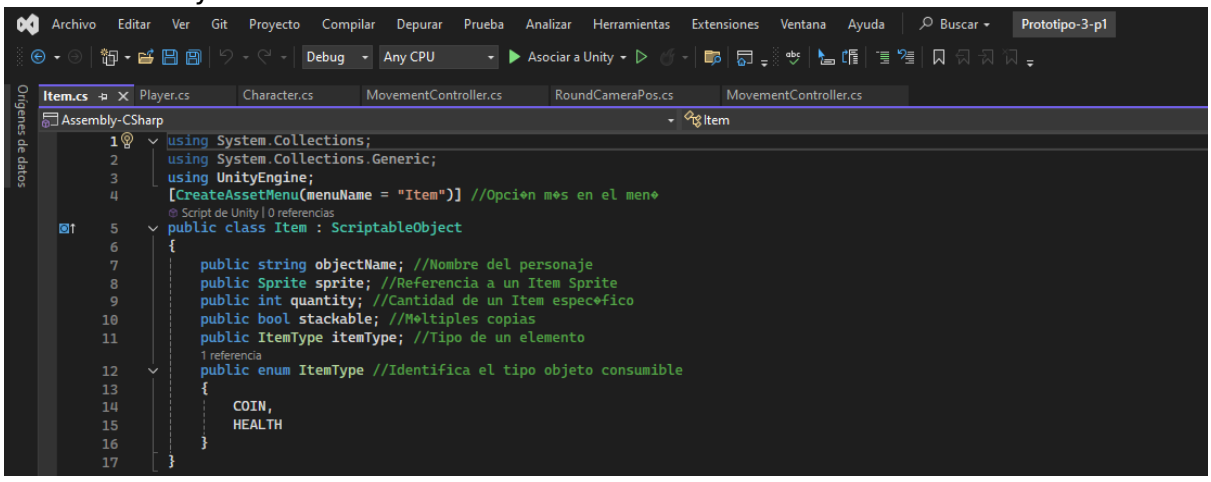
Creamos el Prefab arrastrando el coin object a la carpeta prefabs



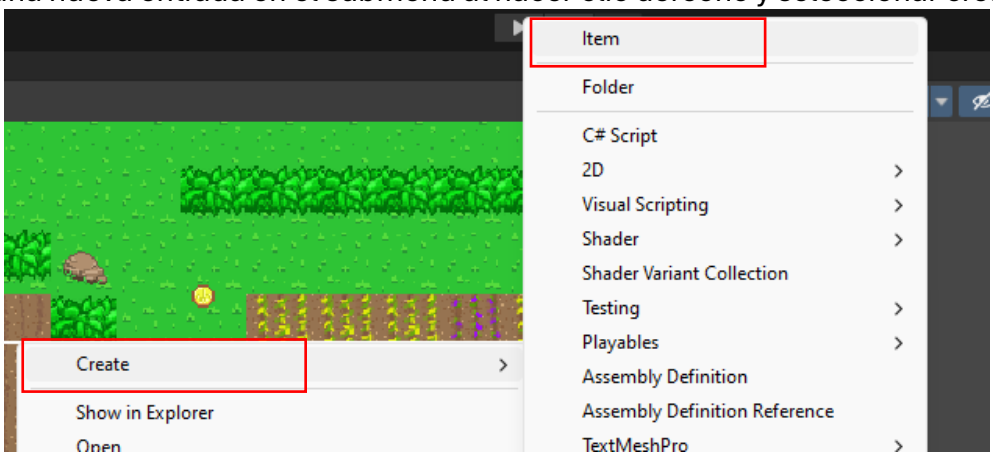
Se modifico el script Player esto para que se detecten las colisiones con la moneda y la pueda recoger



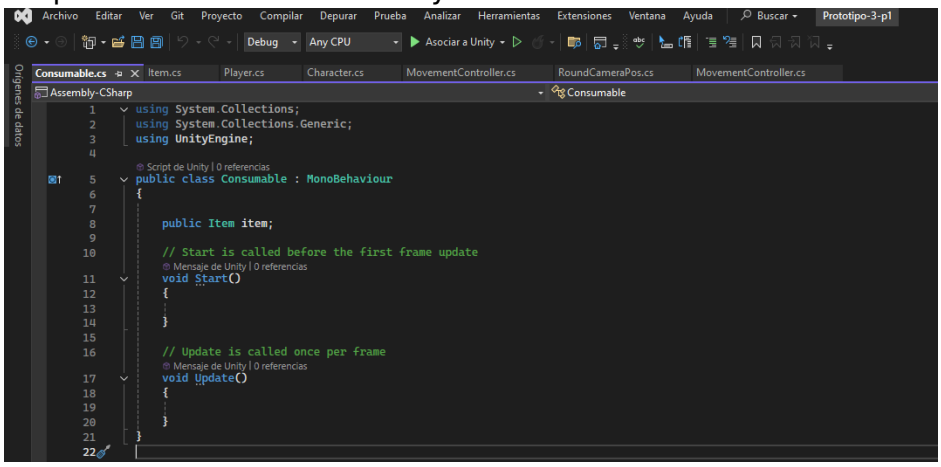
Creamos una carpeta en el directorio de scripts llamada Scriptable Objects, posteriormente creamos un script llamado Item.cs y lo editamos en VS



Esto nos creara una nueva entrada en el submenú al hacer clic derecho y seleccionar create

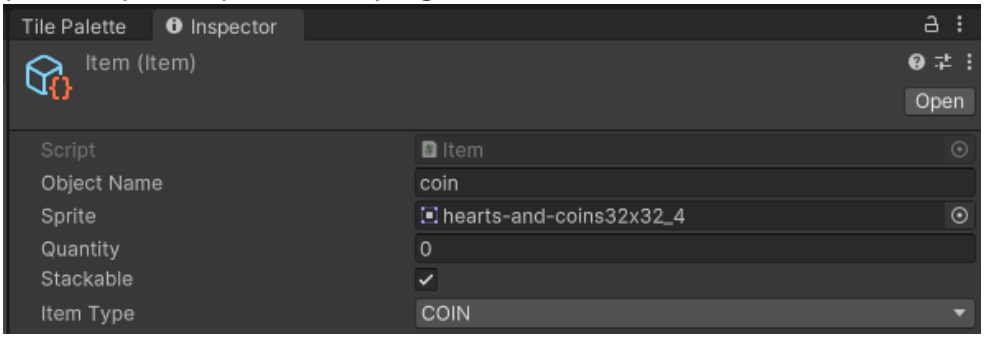


Ahora creamos un script llamado Consumable.cs y lo modificamos

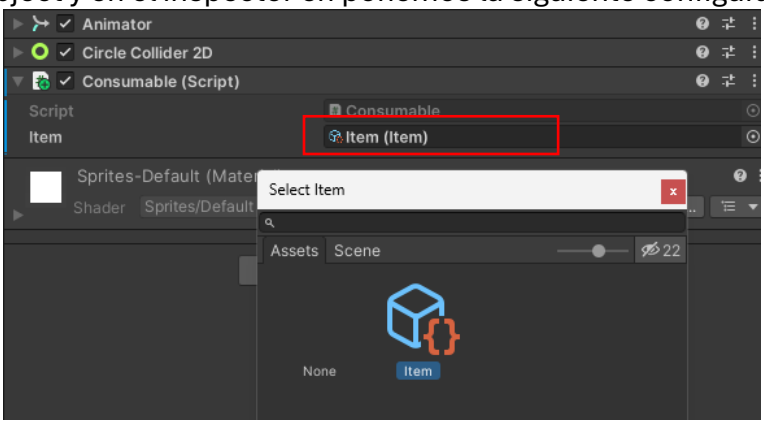


En la carpeta **Scriptable Objects**, seguimos estos pasos:

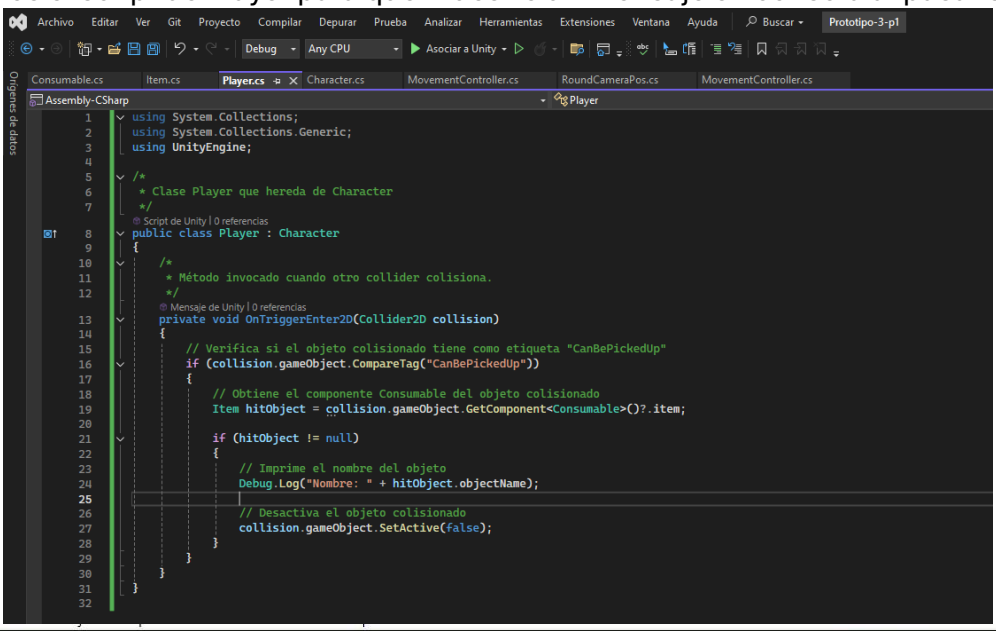
1. Hacemos clic derecho y seleccionamos **Create ➤ Item** en el menú desplegable para crear un **Objeto Item Scriptable**. Alternativamente, desde la barra de menú, seleccionamos **Assets ➤ Create ➤ Item**.
2. Renombramos el objeto programable recién creado a **"Item"**.
3. Nos aseguramos de que el **Objeto Item Scriptable** esté seleccionado. Luego, en el **Inspector**, ajustamos sus propiedades según sea necesario para definir los atributos del objeto. Esto puede incluir propiedades como el nombre, descripción, icono, y cualquier otra información específica que este tipo de objeto requiera en el juego.



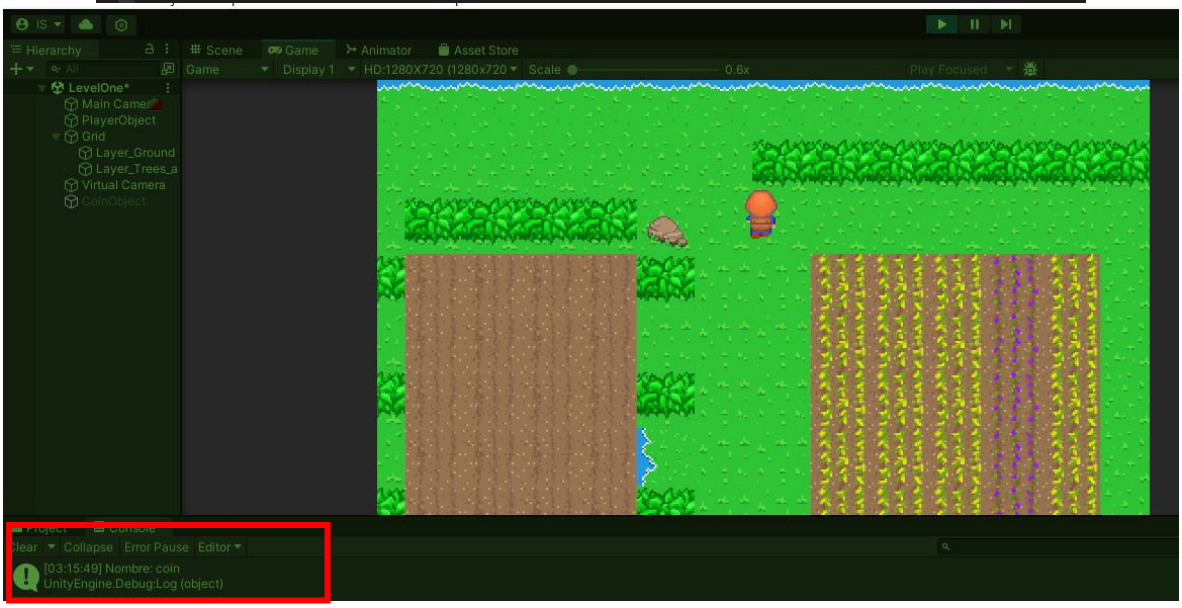
Seleccionamos el coin object y en el inspector en ponemos la siguiente configuracion



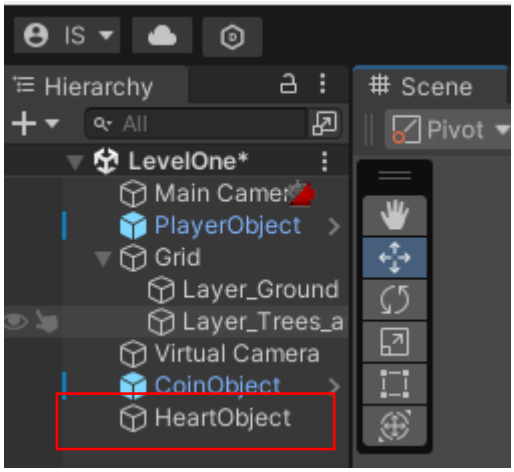
Ahora modificamos el script de Player para que muestre un mensaje en consola al pasar sobre el coin



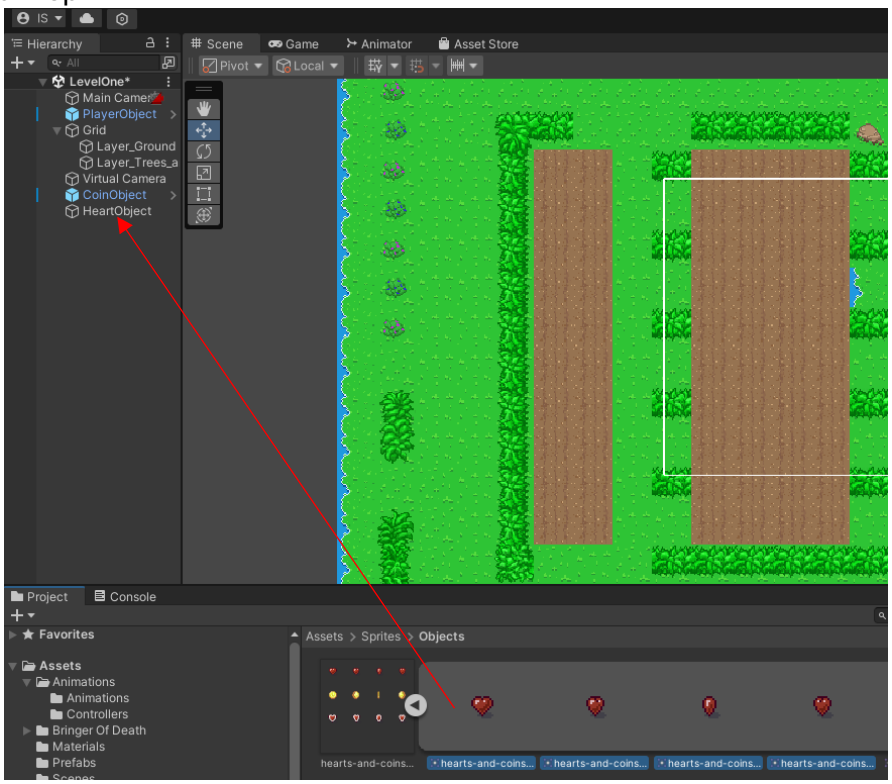
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/*
 * Clase Player que hereda de Character
 */
public class Player : Character
{
    /*
     * Método invocado cuando otro collider colisiona.
     */
    private void OnTriggerEnter2D(Collider2D collision)
    {
        // Verifica si el objeto colisionado tiene como etiqueta "CanBePickedUp"
        if (collision.gameObject.CompareTag("CanBePickedUp"))
        {
            // Obtiene el componente Consumable del objeto colisionado
            Item hitObject = collision.gameObject.GetComponent<Consumable>()?.item;

            if (hitObject != null)
            {
                // Imprime el nombre del objeto
                Debug.Log("Nombre: " + hitObject.objectName);

                // Desactiva el objeto colisionado
                collision.gameObject.SetActive(false);
            }
        }
    }
}
```



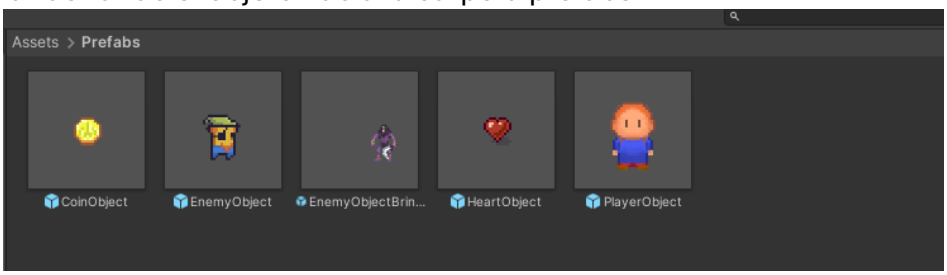
Creamos otro objeto llamado HeartObject



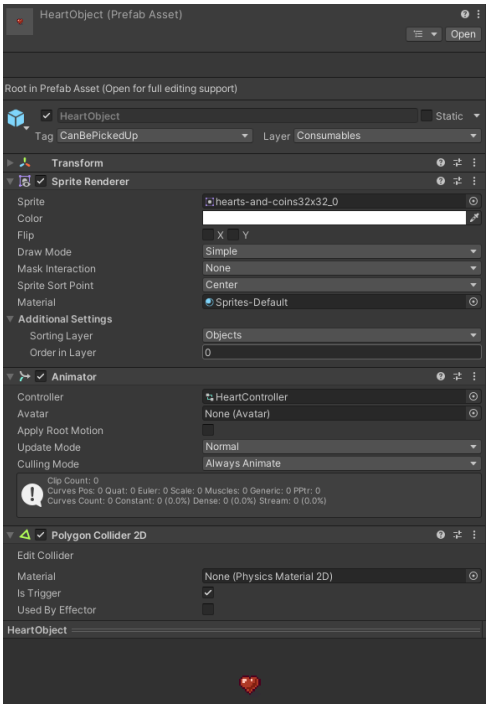
Ahora crearemos la animación para el objeto de heart como se hizo anteriormente con las monedas y a este lo llamamos heart-spin



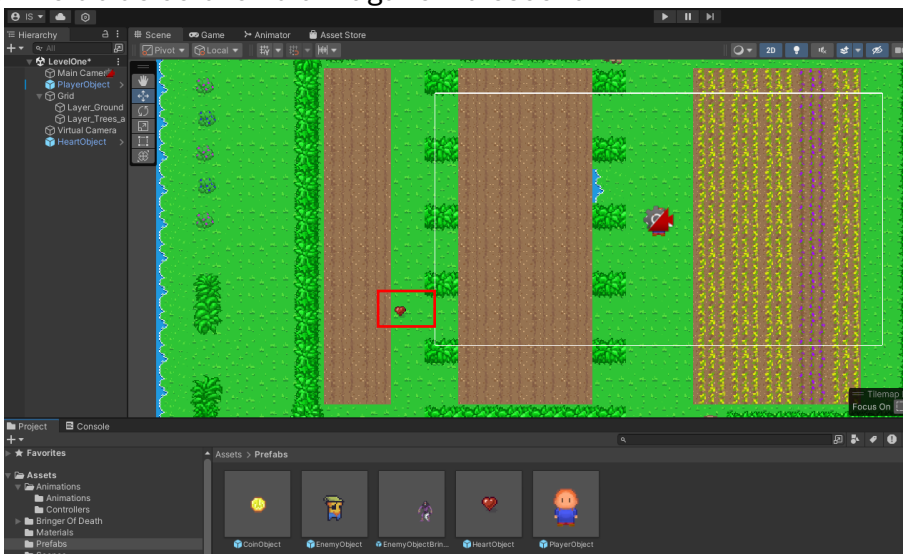
Creamos el Prefab arrastrando el objeto hacia la carpeta prefabs



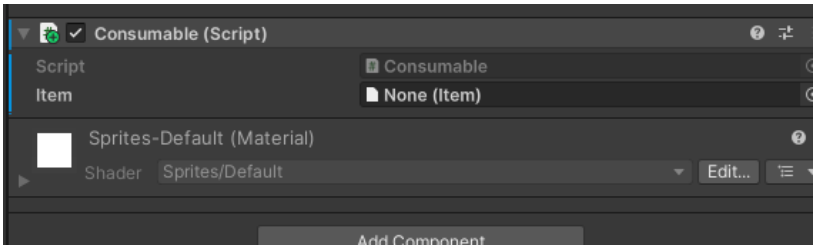
Al Prefab de heart le damos la siguiente configuración en la ventana inspector, se agrego un componente Polygon Collider 2D



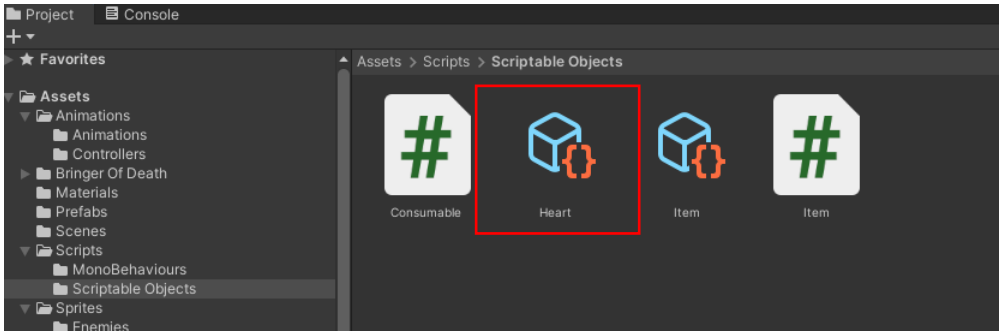
Ahora arrastramos el Prefab de corazón a un lugar en la escena



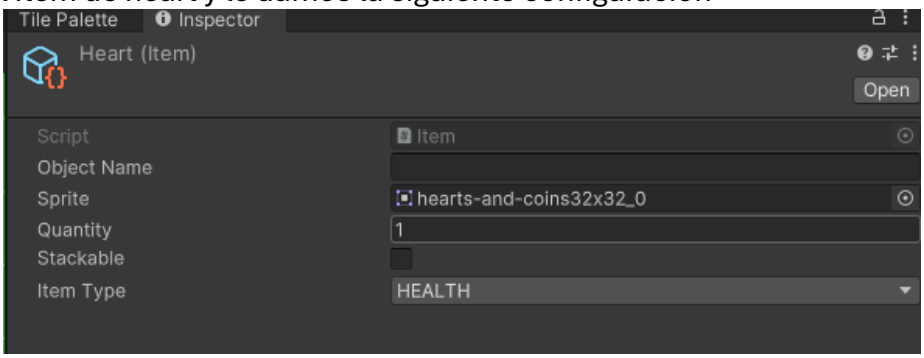
Agregamso el script consumable a heart



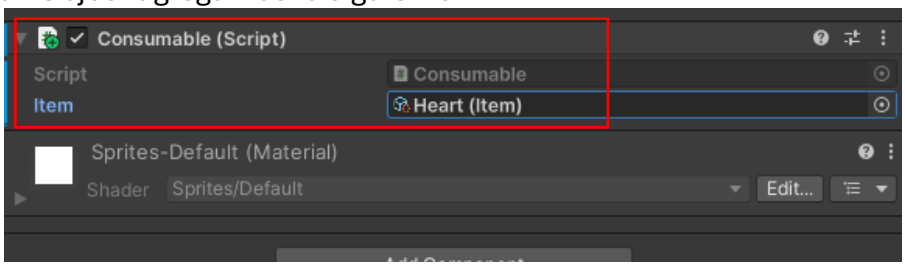
Creamos un nuevo Item llamado Heart



Damos clic sobre el Item de heart y le damos la siguiente configuración

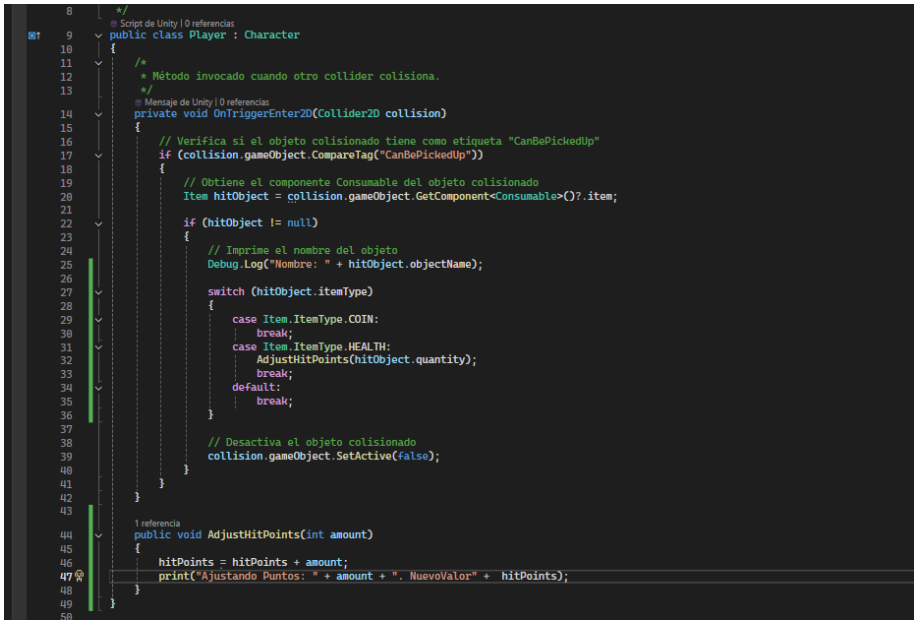


En el inpector de heartObject agregamos lo siguiente



Presionamos play y dirigimos al jugador hacia el heart, veremos la consola que printea el objeto seleccionado

Queremos incrementar los hitPoints del jugador cada vez que obtiene un corazón. Vuelvamos a modificar el script Player.cs



```
*/
Script de Unity | 0 referencias
public class Player : Character
{
    /*
     * Método invocado cuando otro collider colisiona.
     */
    Mensaje de Unity | 0 referencias
    private void OnTriggerEnter2D(Collider2D collision)
    {
        // Verifica si el objeto colisionado tiene como etiqueta "CanBePickedUp"
        if (collision.gameObject.CompareTag("CanBePickedUp"))
        {
            // Obtiene el componente Consumable del objeto colisionado
            Item hitObject = collision.gameObject.GetComponent<Consumable>()?.item;

            if (hitObject != null)
            {
                // Imprime el nombre del objeto
                Debug.Log("Nombre: " + hitObject.objectName);

                switch (hitObject.itemType)
                {
                    case Item.ItemType.COIN:
                        break;
                    case Item.ItemType.HEALTH:
                        AdjustHitPoints(hitObject.quantity);
                        break;
                    default:
                        break;
                }

                // Desactiva el objeto colisionado
                collision.gameObject.SetActive(false);
            }
        }
    }

    1 referencia
    public void AdjustHitPoints(int amount)
    {
        hitPoints = hitPoints + amount;
        print("Ajustando Puntos: " + amount + ". NuevoValor" +  hitPoints);
    }
}
```

Con esto, una vez que pasamos sobre el heart, nos printea en consola el nuevo valor que obtendremos