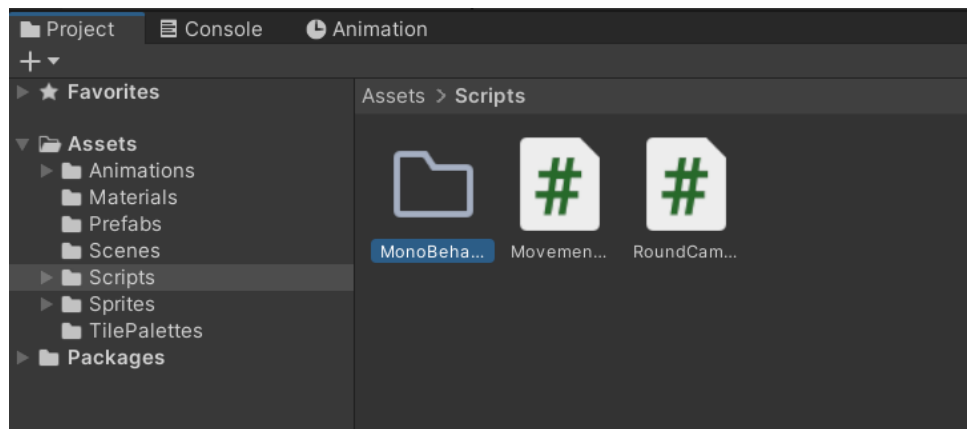
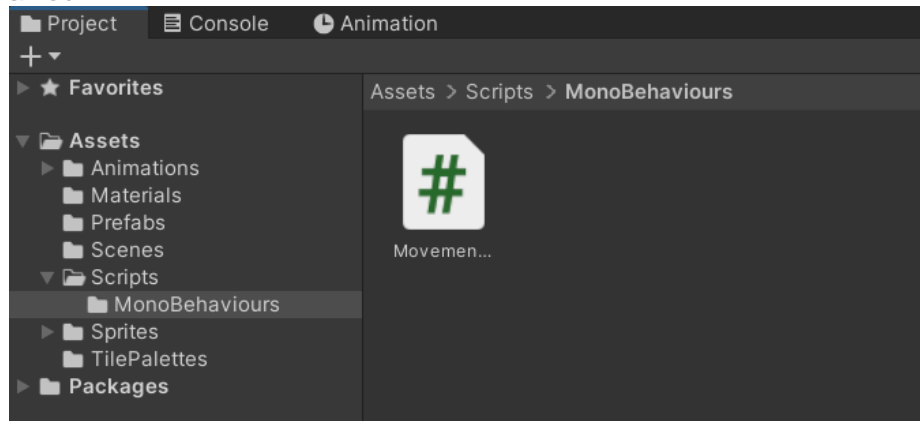


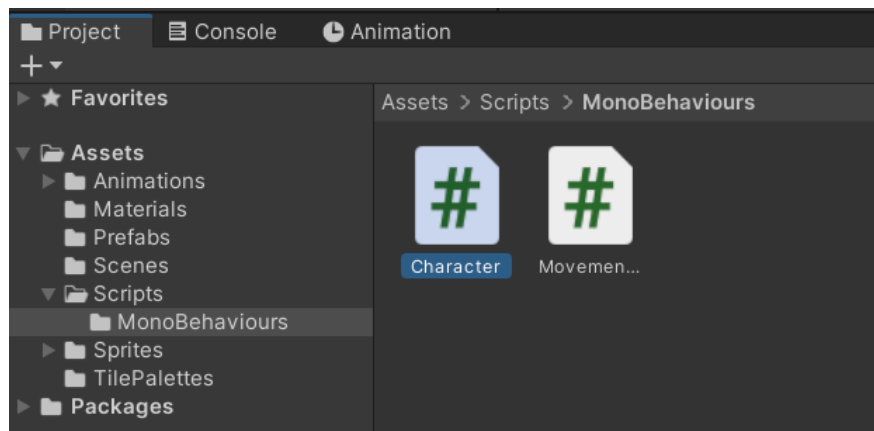
Cree una nueva carpeta en Scripts llamada **MonoBehaviours**.

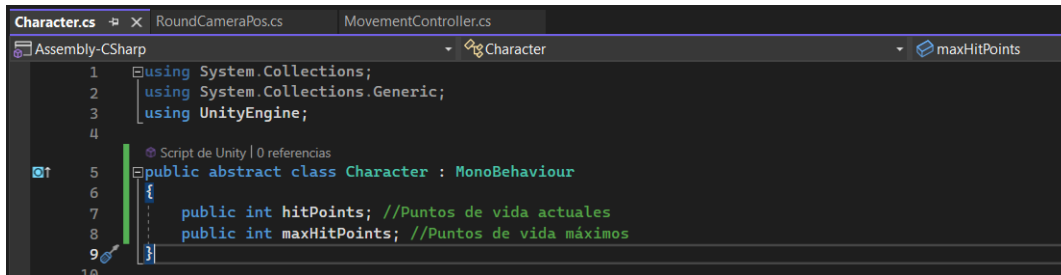


Mueva el script **MovementController.cs** a esta carpeta, porque hereda de MonoBehaviour.



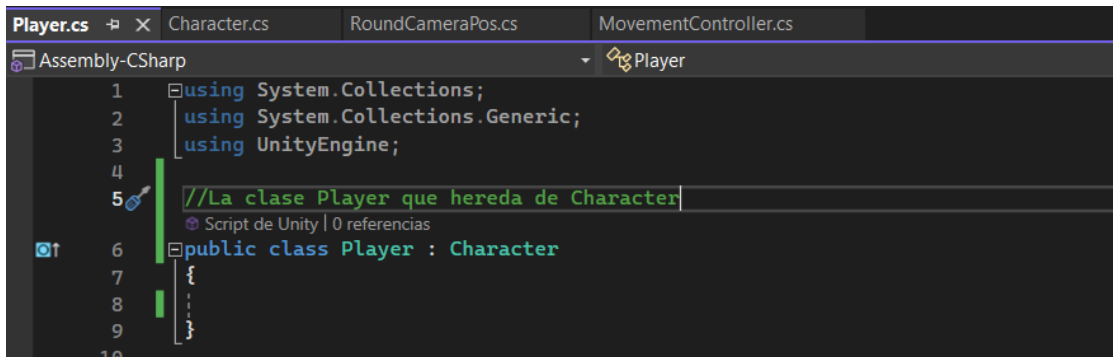
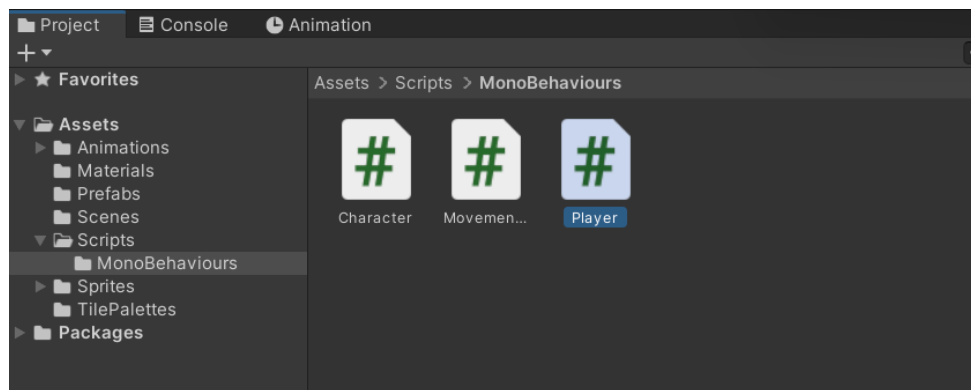
Dentro de la carpeta **MonoBehaviours**, cree un nuevo script C# llamado **Character.cs**. Haga doble clic en el script para abrirlo en nuestro editor y modifica el archivo fuente





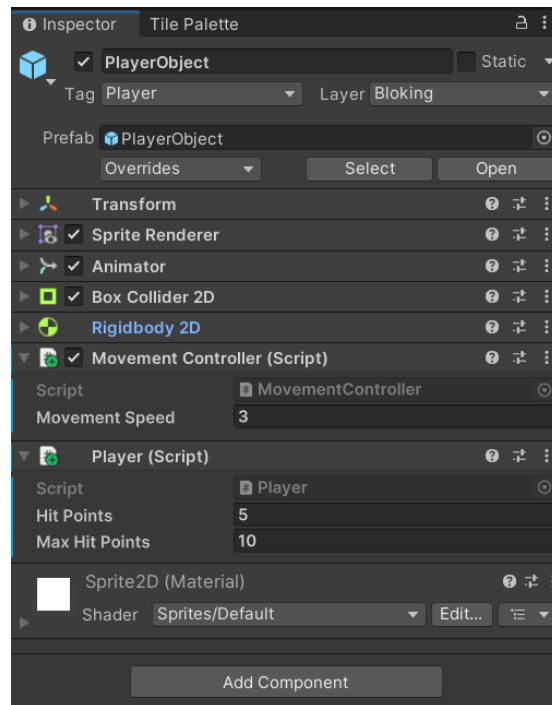
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public abstract class Character : MonoBehaviour
6 {
7     public int hitPoints; //Puntos de vida actuales
8     public int maxHitPoints; //Puntos de vida máximos
9 }
```

En nuestra carpeta **MonoBehaviours**, crea un nuevo script C# llamado **Player.cs**. Esta clase Player comenzará extremadamente simple, pero le agregaremos funcionalidad a medida que avancemos.

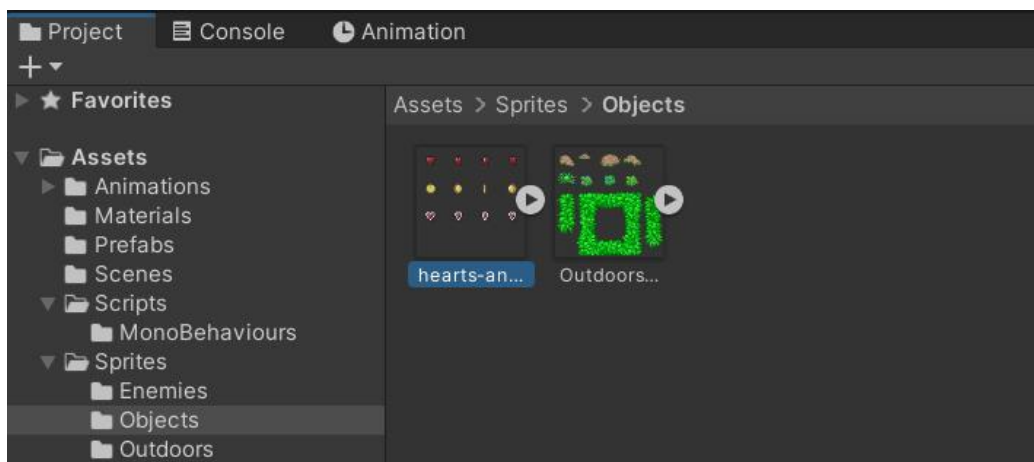


```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //La clase Player que hereda de Character
6 public class Player : Character
7 {
8 }
9 
```

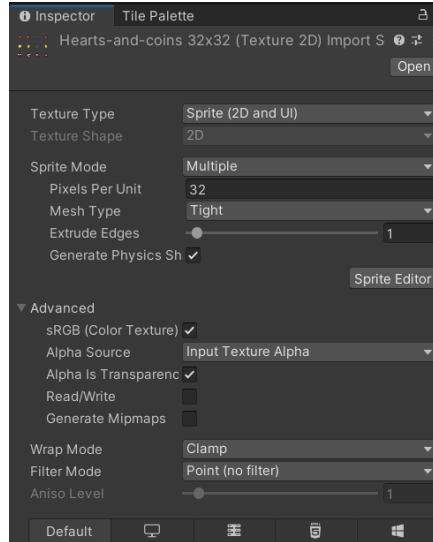
Selecciona el [Player prefab](#). Arrastre y suelte el script [Player.cs](#) al objeto Prefab. Da al jugador 5 puntos de golpe y 10 puntos de vida máximos para empezar.



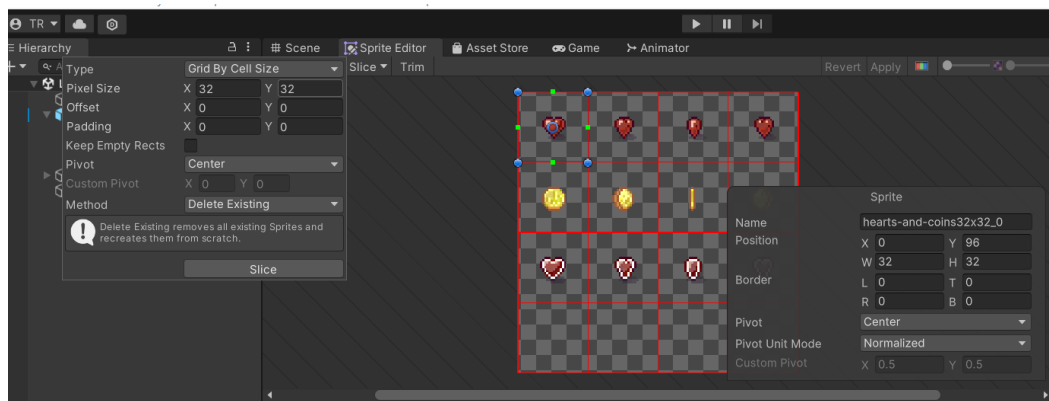
En la carpeta de recursos del tutorial, seleccione la hoja de sprites titulada "[hearts-and-coins32x32.png](#)", arrastrala a la carpeta [Assets](#) ► [Sprites](#) ► [Objects](#)



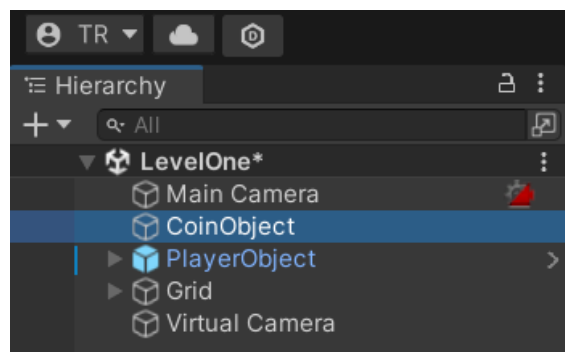
Selecciona la imagen "**hearts-and-coins32x32.png**", después en la vista Inspector modifique las propiedades,



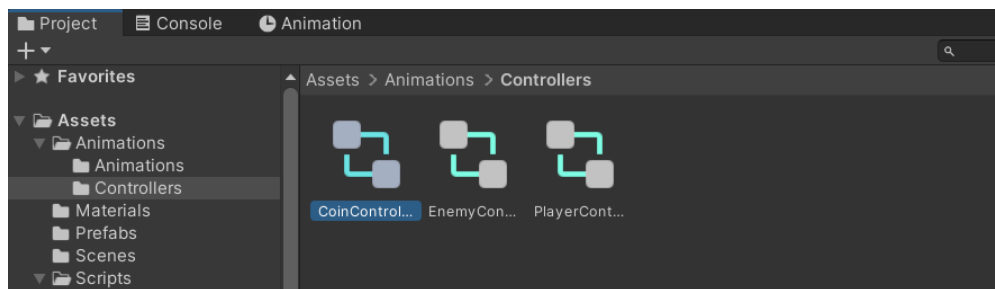
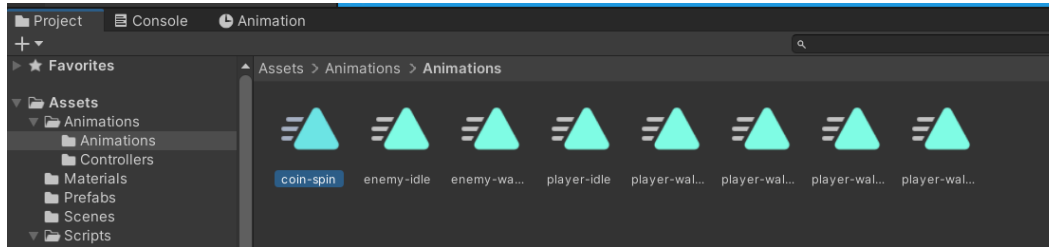
Abra el Editor de Sprite presionando el botón titulado **Sprite Editor**. En el menú **Slice**, seleccione **Grid by Cell Size** y configure el Tamaño del píxel en ancho: **32**, alto: **32**. Presione **Apply** y cierre el Editor de Sprite.



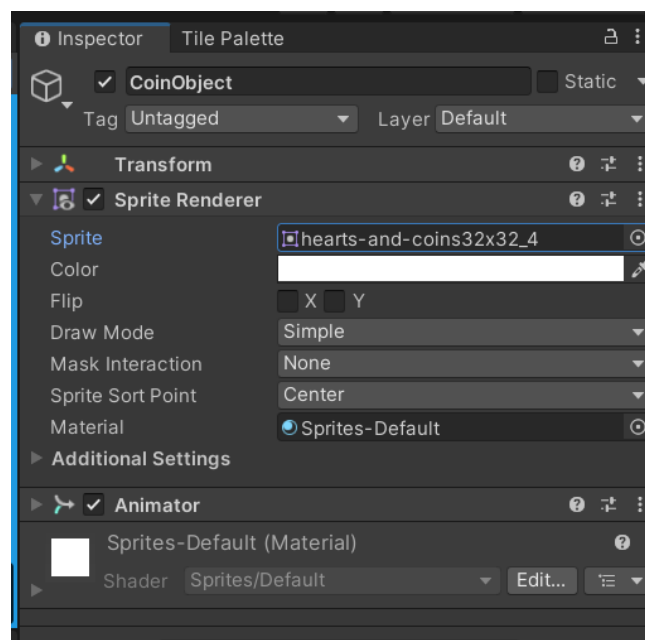
Crea un nuevo **GameObject** en la vista **Hierarchy** y cambiale el nombre a **CoinObject**.



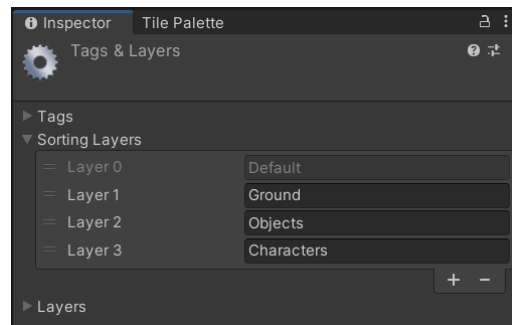
Selecciona los cuatro sprites de monedas individuales de la hoja de sprites corazón-moneda y arrastrarlos al **CoinObject** para crear una nueva animación. Cambia el nombre del clip de animación a "**coin-spin**" y guárdelo en la carpeta **Animations > Animations**. Cambia el nombre del controlador a "**CoinController**" y muévelo a la carpeta **Controllers**.



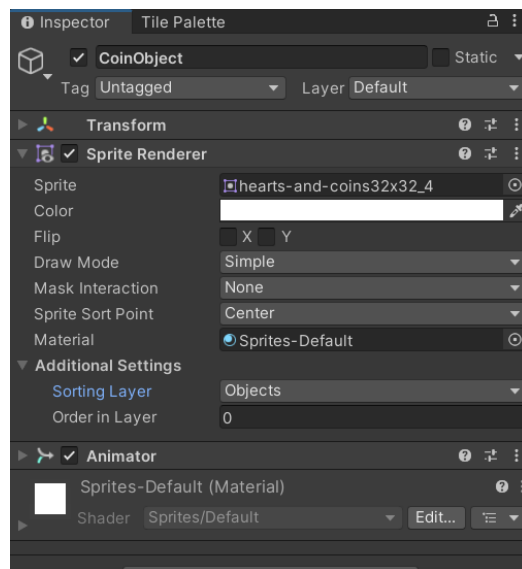
En el componente **Sprite Renderer** del **CoinObject**, haga clic en el pequeño punto junto a la forma "**Sprite**" y seleccione un Sprite para usar al obtener una vista previa de este componente en la vista Scene.



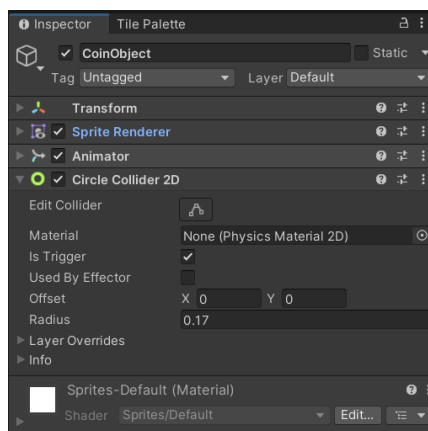
Crea una nueva capa de ordenamiento seleccionando el menú desplegable **Sorting Layer** en el componente **Sprite Renderer**, haga clic en "**Add Sorting layer**", luego agregue una nueva capa llamada, "**Objects**" entre la capa Ground y Character.



Seleccione el **CoinObject** nuevamente y establezca su Capa **Sorting Layer** en: **Objects**.



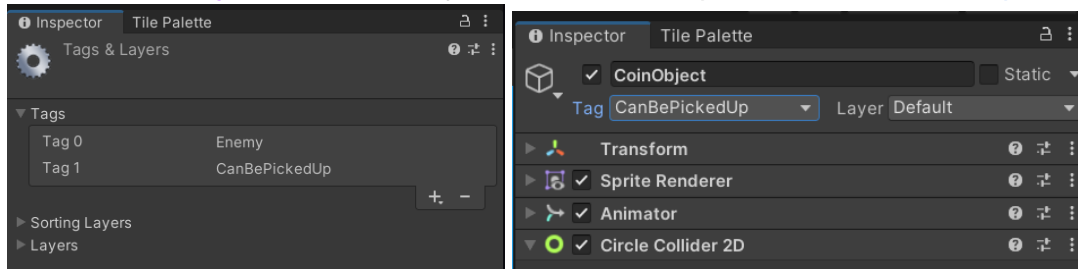
Seleccione **CoinObject** nuevamente y agregue un componente **Circle Collider 2D** a él. Establecer el radio del **Circle Collider 2D** a: **0.17**, por lo que es aproximadamente del mismo tamaño que el Sprite.



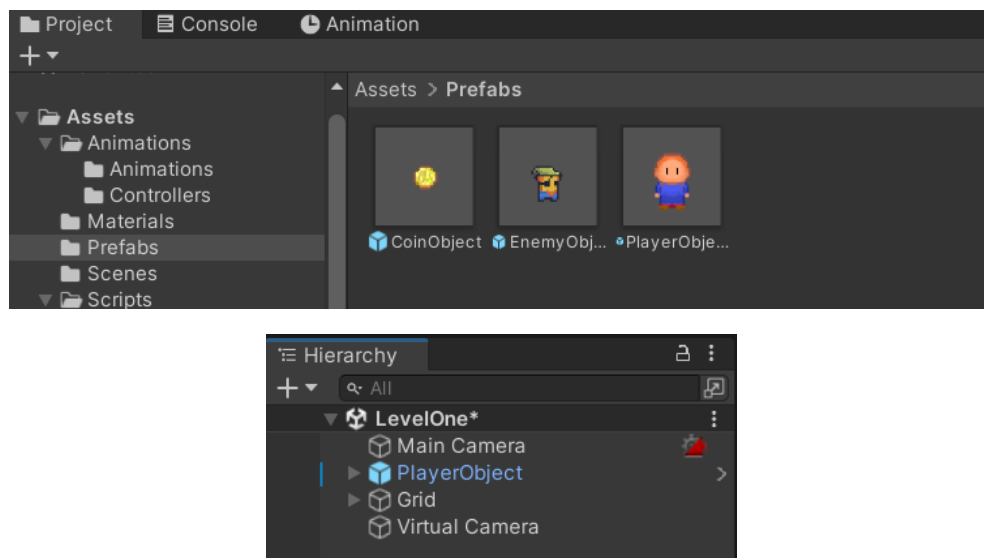
Creemos una nueva etiqueta en el menú **Tags & Layers** llamada "**CanBePickedUp**":

1. Seleccione **CoinObject** en la vista **Hierarchy**.
2. En la parte superior izquierda del Inspector, seleccione "**Add tag**" del menú Tag.
3. Cree la etiqueta **CanBePickedUp**.

Seleccione **CoinObject** nuevamente y establezca su Etiqueta a: **CanBePickedUp**.

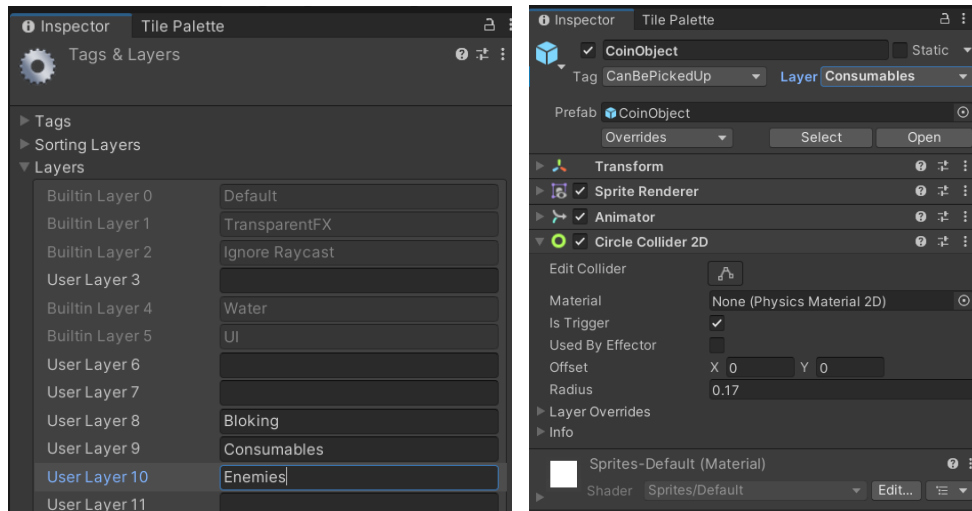


Cree un objeto Prefab a partir del objeto **CoinObject** arrastrándolo a la carpeta **Prefabs** desde la carpeta Hierarchy. Puede eliminar **CoinObject** de la vista Hierarchy después de haberlo creado.



Debemos crear y asignar capas a los GameObjects relevantes.

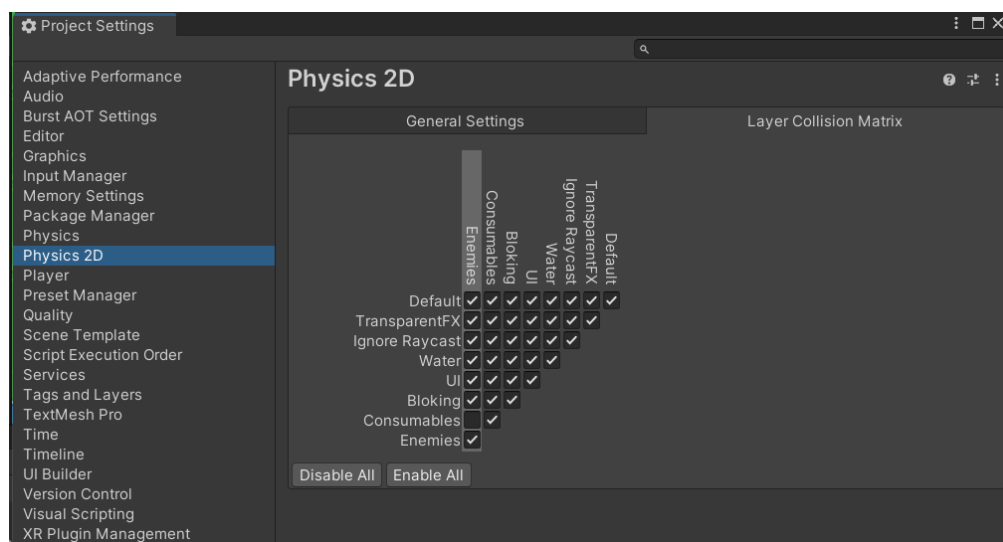
1. Seleccione **CoinObject** en la vista Hierarchy.
2. En el Inspector, seleccione el menú desplegable **Layer**.
3. Seleccione: "**Add Layer**".
4. Cree una nueva capa llamada: "**Consumables**".
5. Crea otra capa llamada: "**Enemies**".



Vaya al menú **Edit ► Project settings ► Physics 2D**. Observa la matriz de colisión de capas en la parte inferior de la vista **Physics2DSettings**. Aquí es donde configuraremos las capas para permitir que los enemigos caminen bien a través de monedas, potenciadores y cualquier otra cosa que elijamos.

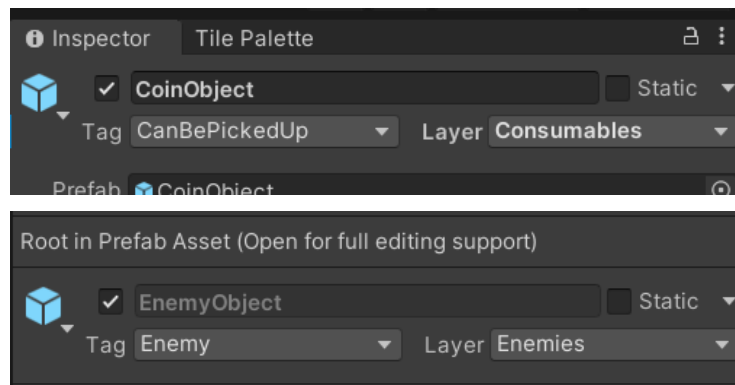
Queremos configurar el jugador y los objetos de moneda para que sus colisionadores sean conscientes el uno del otro. Queremos que los colisionadores enemigos no se den cuenta de los colisionadores de monedas.

Desmarque la casilla en la intersección entre **Consumables** y **Enemies** véase siguiente imagen.

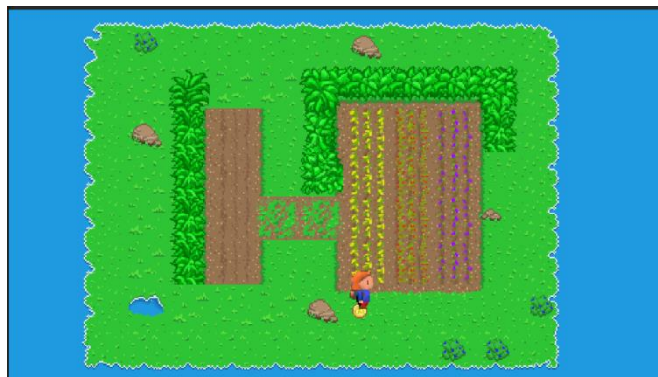




Seleccione el prefab **CoinObject** y cambie su capa para que sea: **Consumables**. Ya que estamos en eso, seleccione el objeto prefabricado **EnemyObject** en la carpeta **Prefabs** y cambie su capa para ser: **Enemies**.



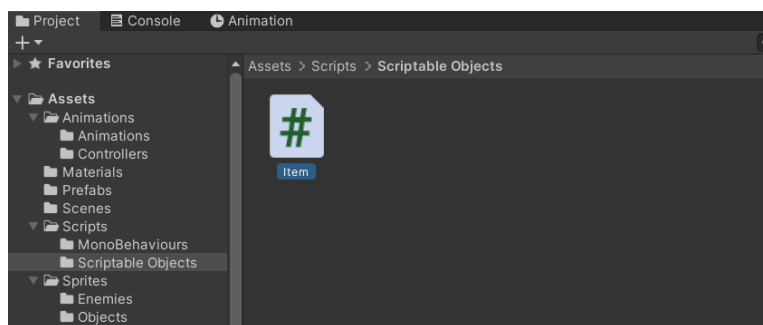
Ahora arrastre un objeto prefabricado **CoinObject** en algún lugar de la escena. Presiona play y lleva al personaje hacia la moneda.



La propiedad "**Is Trigger**" se utiliza para detectar cuando otro objeto ha entrado en el rango definido por el Colisionador. Cuando el colisionador del jugador toca el colisionador circular de la moneda, el método: **void OnTriggerEnter2D (Collider2D collision)** se llama automáticamente en ambos objetos unidos a los colisionadores.

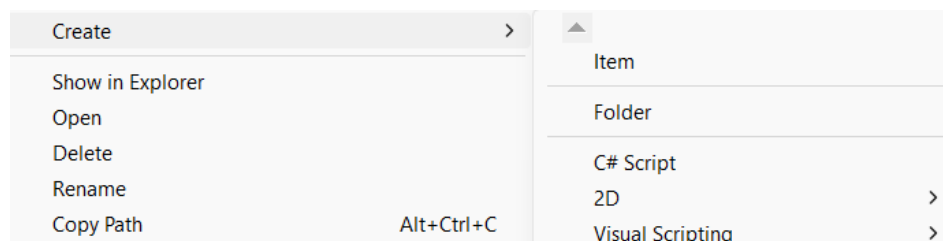
```
Player.cs | Character.cs | RoundCameraPos.cs | MovementController.cs
Assembly-CSharp | Player
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //La clase Player que hereda de Character
6 [Script de Unity | 0 referencias]
7 public class Player : Character
8 {
9     //Metodo invocado cuando otro colider colisiona.
10    [Mensaje de Unity | 0 referencias]
11    private void OnTriggerEnter2D(Collider2D collision)
12    {
13        //Verifica si el objeto colisionado tiene como etiqueta CanBePickedUp
14        if (collision.gameObject.CompareTag("CanBePickedUp"))
15        {
16            //Ocultamos el objeto de la escena
17            collision.gameObject.SetActive(false);
18        }
19    }
20 }
```

Crea una carpeta en el directorio de Scripts llamada "[Scriptable Objects](#)". Luego haga clic derecho y cree un nuevo script llamado [Item.cs](#).

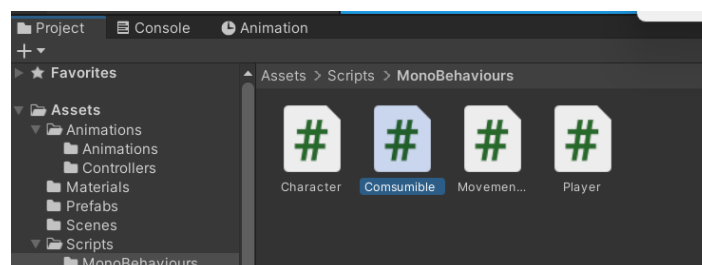


```
Item.cs | Player.cs | Character.cs | RoundCameraPos.cs | MovementController.cs
Assembly-CSharp | Item
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [CreateAssetMenu(menuName = "Item")] //Opción más en el menú
6 public class Item : ScriptableObject
7 {
8     public string objectName; //Nombre del personaje
9     public Sprite sprite; //Referencia a un Item Sprite
10    public int quantity; //Cantidad de un Item específico
11    public bool stackable; //Múltiples copias
12    public ItemType itemType; //Tipo de un elemento
13    1 referencia
14    public enum ItemType //Identifica el tipo objeto consumible
15    {
16        COIN,
17        HEALTH
18    }
19 }
20
```

Se crea una entrada en el submenú [Create](#)



Construyendo un script [Consumible.cs](#)

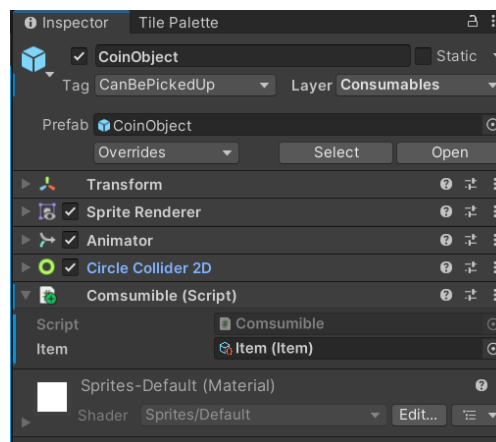
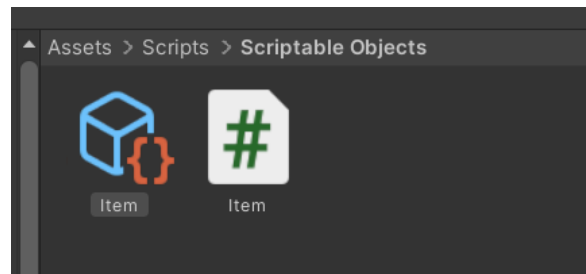
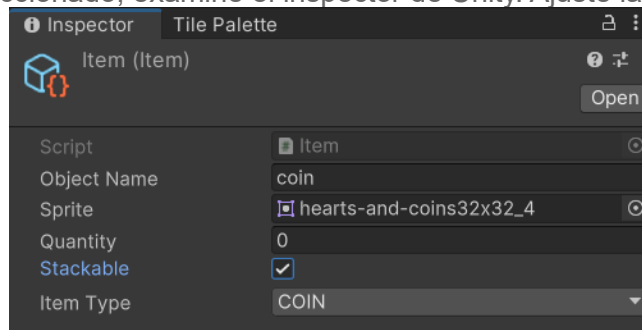


```
Comsumible.cs | Item.cs | Player.cs | Character.cs | RoundCameraPos.cs | MovementController.cs
Assembly-CSharp | Comsumible | Item
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //Hereda de MonoBehaviour lo podemos enlazar a un GameObject.
6 public class Comsumible: MonoBehaviour
7 {
8     //Referencia a un objeto Scriptable o programable
9     public Item item;
10 }
```

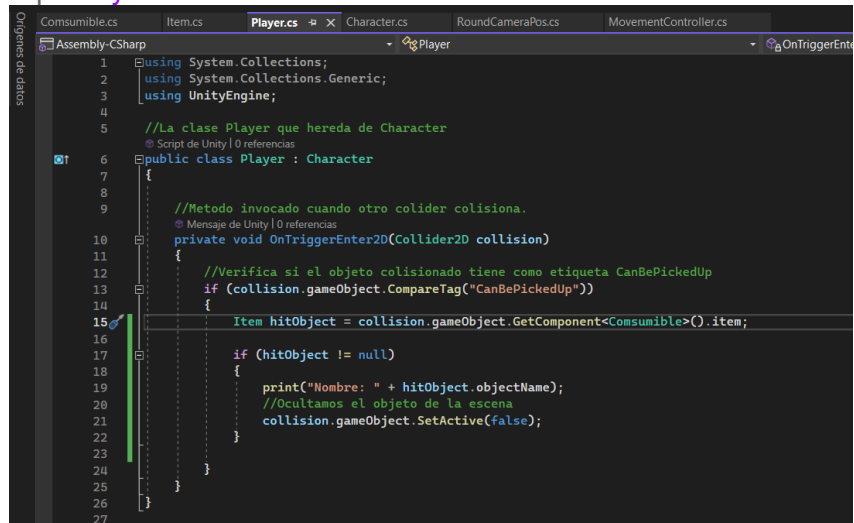
Seleccione el objeto prefabricado **CoinObject** y arrastre el script **Comsumible.cs** sobre él. Es necesario establecer la propiedad del **Item Comsumible**

En la carpeta **Scriptable Objects**, haga clic con el botón derecho y seleccione **Create ► Item** en la parte superior del menú **Asset** para crear un Objeto Item Scriptable. Si prefieres usar la barra de menú en la parte superior del Editor de Unity, puedes seleccionar **Assets ► Create ► Item**.

Cambie el nombre del objeto programable, "**Item**". Asegúrese de que el objeto **Item Scriptable** esté seleccionado, examine el inspector de Unity. Ajuste las propiedades

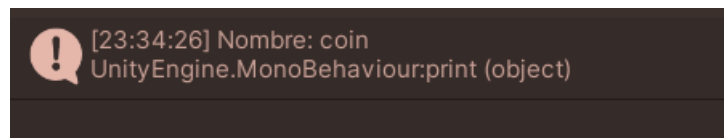


Modifica el script **Player.cs**



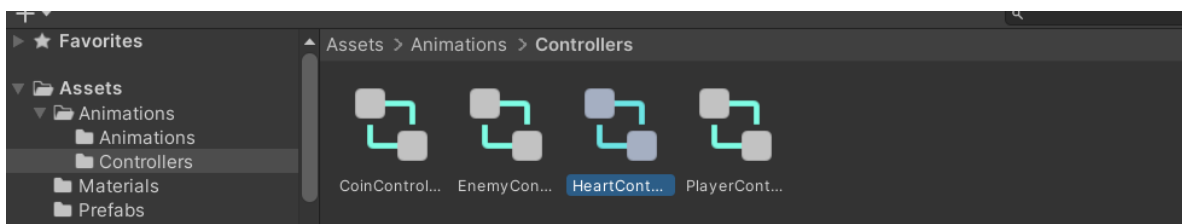
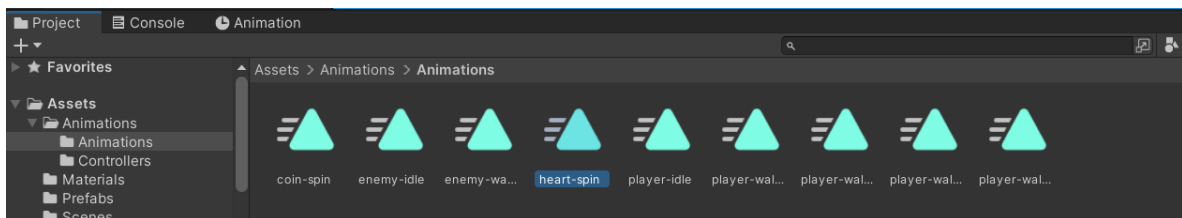
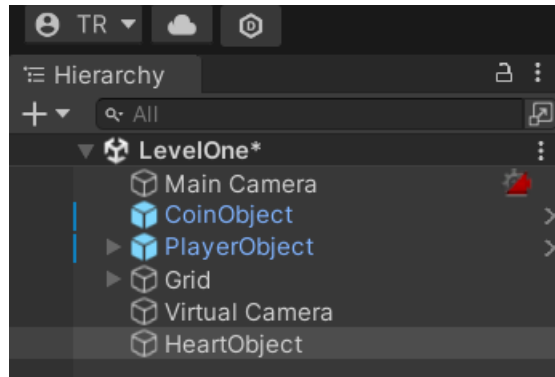
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 //La clase Player que hereda de Character
6 public class Player : Character
7 {
8
9     //Metodo invocado cuando otro colider colisiona.
10    private void OnTriggerEnter2D(Collider2D collision)
11    {
12        //Verifica si el objeto colisionado tiene como etiqueta CanBePickedUp
13        if (collision.gameObject.CompareTag("CanBePickedUp"))
14        {
15            Item hitObject = collision.gameObject.GetComponent<Comsumible>().item;
16
17            if (hitObject != null)
18            {
19                print("Nombre: " + hitObject.objectName);
20                //Ocultamos el objeto de la escena
21                collision.gameObject.SetActive(false);
22            }
23        }
24    }
25 }
26
27
```

**Guarda** el script y vuelve al editor de Unity. Presione el botón **Play** y lleva al jugador a una moneda. Observa la salida en la consola

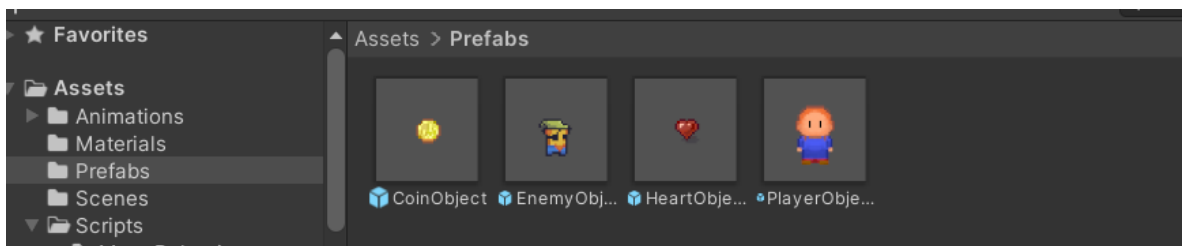


Creemos otro objeto que el jugador puede recoger: un corazón encendido. Usa los sprites que cortamos anteriormente de la hoja de sprites "[hearts-and-coins32x32.png](#)".

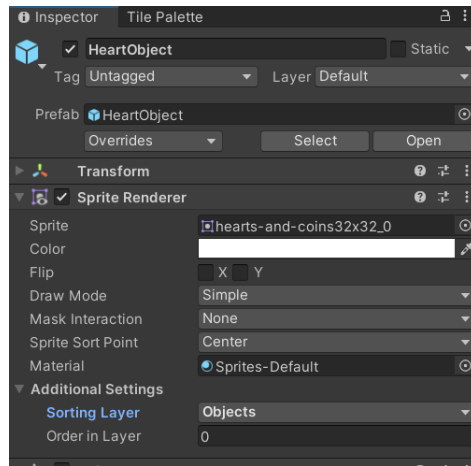
1. Crea un **GameObject** y cámbiale el nombre a "**HeartObject**".
2. Agrega sprites para la animación Prefab. Usa el sprite titulado: "**hearts-and-coins32x32**" terminando en 0, 1, 2 y 3. Nombra la animación recién creada como "heart-spin" guárdelo en la carpeta **Animations** ➤ **Animations**.



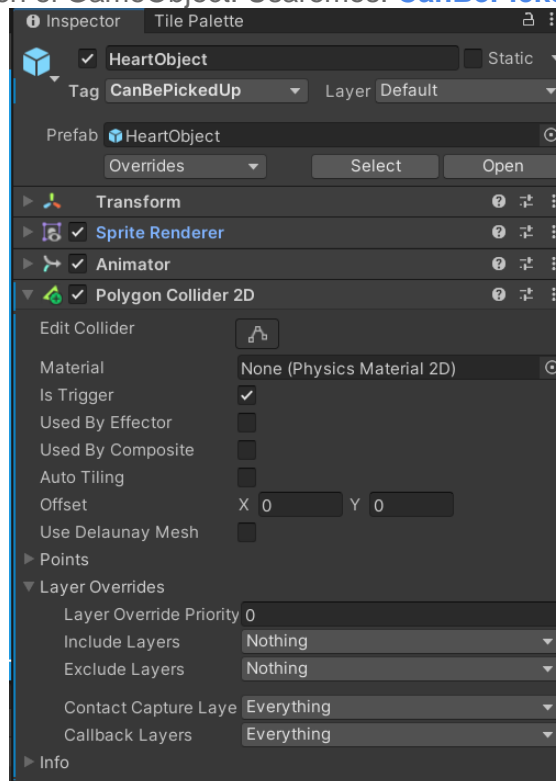
Crema un **Prefab** a partir de **HeartObject** arrastrándolo a la carpeta de **Prefabs**, luego borra el objeto original de la vista Hierarchy.



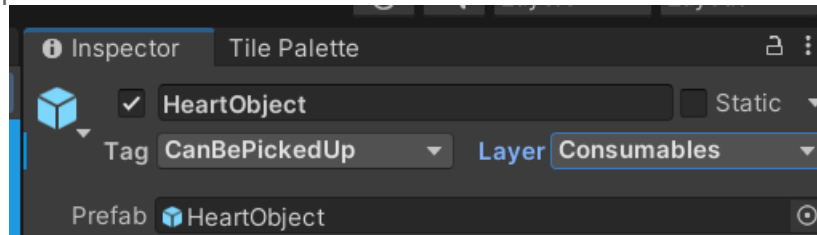
Selecciona el **Prefab Heart Object** de la carpeta y configura la propiedad **Sprite**. Esta propiedad se usa cuando se previsualiza la escena. En el componente **Sprite Renderer**, configure **Sorting Layer** a **Objects** para que el Prefab sea visible.



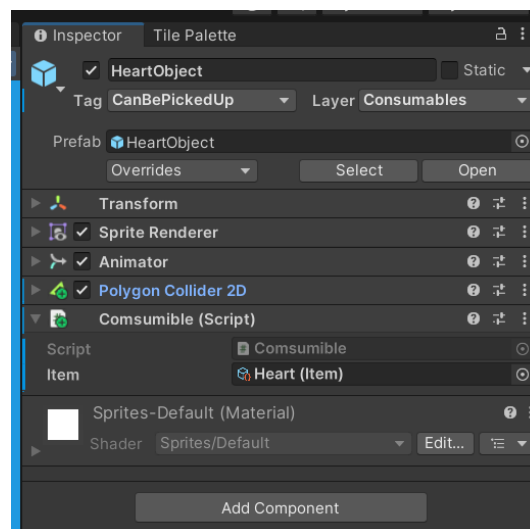
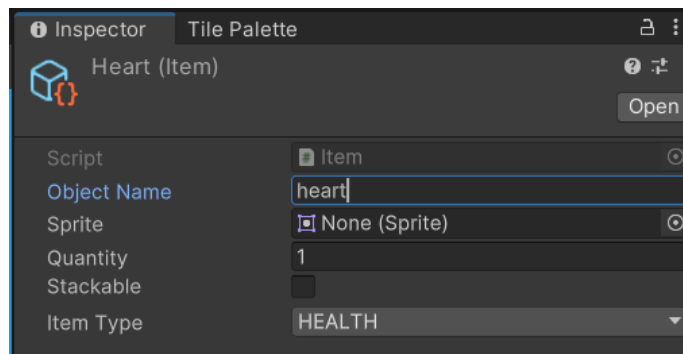
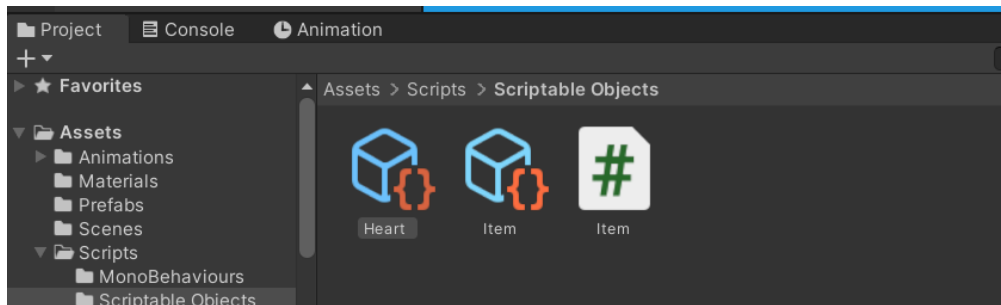
Agregue un componente **Collider 2D**. Podemos usar un círculo Collider, Box o Polygon 2D, pero para el sprite corazón sprite usaremos un **Polygon 2D** qué funcionará mejor. Edite la forma de colisionador si es necesario. Dependiendo del tipo de **Prefab** que se está creando, establezca: **Is Trigger** activado. Establezca la etiqueta en el GameObject. Usaremos: **CanBePickedUp**



Cambie la capa a "**Consumables**".



Vamos a configurar **Prefab Heart** para que contenga una referencia a un objeto programable de la misma manera que lo hace Coin. Añade el script **Consumable.cs** a Heart seleccionando Prefab, luego presione el botón "**Add component**" y escriba, "**Consumable**".



¡Eso es todo! Si presiona **Play** y lleve al jugador al heart Prefab sobre la pantalla,

```
[23:48:58] Nombre: heart
UnityEngine.MonoBehaviour:print (object)

Nombre: heart
UnityEngine.MonoBehaviour:print (object)
Player.OnTriggerEnter2D (UnityEngine.Collider2D) (at Assets/Scripts/MonoBehaviours/Player.cs:19)
```

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

```
Comsumible.cs  Item.cs  Player.cs  Character.cs  RoundCameraPos.cs  MovementController.cs
Assembly-CSharp  Player
6  public class Player : Character
7  {
8
9      //Metodo invocado cuando otro colider colisiona.
10     private void OnTriggerEnter2D(Collider2D collision)
11     {
12         //Verifica si el objeto colisionado tiene como etiqueta CanBePickedUp
13         if (collision.gameObject.CompareTag("CanBePickedUp"))
14         {
15             Item hitObject = collision.gameObject.GetComponent<Comsumible>().item;
16
17             if (hitObject != null)
18             {
19                 switch (hitObject.itemType)
20                 {
21                     case Item.ItemType.COIN:
22                         break;
23                     case Item.ItemType.HEALTH:
24                         AdjustHitPoints(hitObject.quantity);
25                         break;
26                     default:
27                         break;
28                 }
29                 print("Nombre: " + hitObject.objectName);
30                 //Ocultamos el objeto de la escena
31                 collision.gameObject.SetActive(false);
32             }
33         }
34     }
35
36     1 referencia
37     public void AdjustHitPoints(int amount)
38     {
39         hitPoints = hitPoints + amount;
40         print("Ajustando puntos: " + amount + " nuevo valor: " + hitPoints);
41     }
42 }
```

Guarde **Player.cs** vuelva al editor Unity.

Presiona **Play** y haz que el jugador corra hacia el objeto Prefab Heart. Debería ver el mensaje

```
[23:52:58] Ajustando puntos: 1 nuevo valor: 6
UnityEngine.MonoBehaviour:print (object)

[23:52:58] Nombre: heart
UnityEngine.MonoBehaviour:print (object)

[23:52:58] Nombre: coin
UnityEngine.MonoBehaviour:print (object)

Ajustando puntos: 1 nuevo valor: 6
UnityEngine.MonoBehaviour:print (object)
Player.AdjustHitPoints (int) (at Assets/Scripts/MonoBehaviours/Player.cs:39)
Player.OnTriggerEnter2D (UnityEngine.Collider2D) (at Assets/Scripts/MonoBehaviours/Player.cs:24)

Nombre: coin
```



