Instituto Tecnológico Superior de Guasave
Ing. Sistemas Computacionales

501 matutino

Graficacion
Profesor:
Graciela Lugo Rubio
Alumnos:
Contreras Vega Abelardo
Montoya Reyes Jorge de Jesus

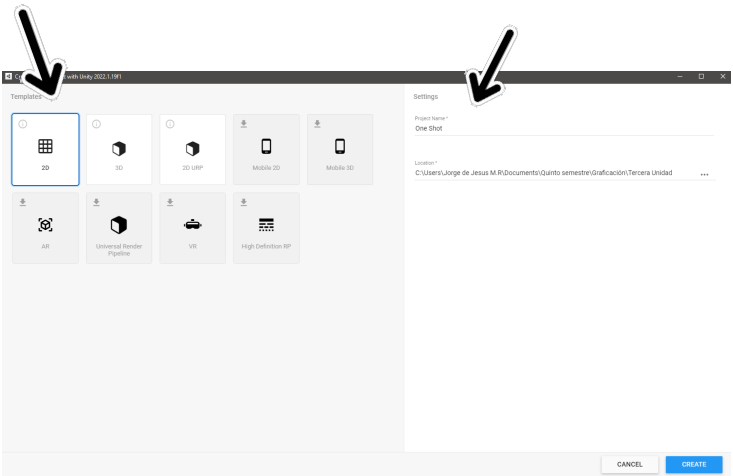
Proyecto Final
One Shot

Guasave, Sinaloa 29 de noviembre del 2022

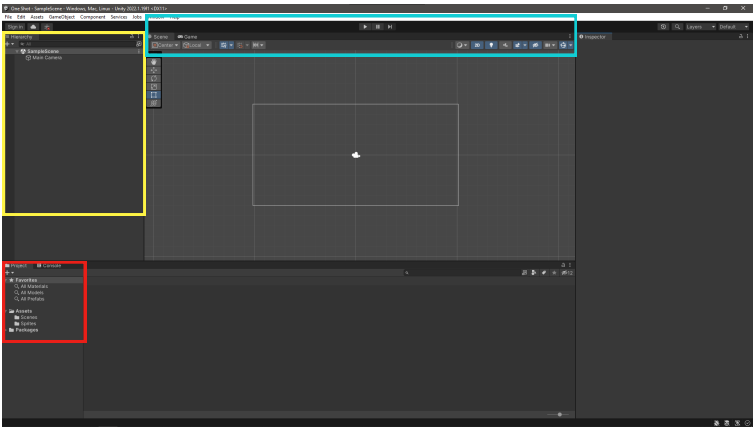
One Shot

UNITY
Es una herramienta que esta echa para desarrollar videojuegos para diversas plataformas mediante un editor y scripting

Para iniciar con la creación del proyecto de inicia un nuevo proyecto y se selecciona la dimensión del juego, que en nuestro caso es 2D y poner el nombre del proyecto que es el nombre del juego



Para empezar el proyecto damos en el botón de crear



Como se observa esta es la pestaña principal de Unity al crear un nuevo proyecto

Hierarchy: La ventana de Jerarquía contiene cada GameObject de la escena actual. Algunos de estos son instancias directas de archivos de asceta como modelos 3D, y otras son instancias de Prefabs, objetos personalizados que van hacer gran parte de su juego. Puede seleccionar objetos en la jerarquía y luego arrastrar un objeto a otro para hacer uso de Parenting (mirar abajo). Como hay objetos que están siendo agregados y eliminados en la escena, estos van a aparecer y desaparecer de la jerarquía también.

Project: Muestra la estructura de carpetas del proyecto como una lista de jerarquía. Cuando una carpeta es seleccionada de una lista haciendo clic, su contenido va a ser mostrado en el panel a la derecha. Usted puede hacer clic en el triángulo pequeño para expandir o colapsar la carpeta, mostrando cualquier carpetas anidadas que contenga.

Scene y Game: Las escenas contienen los objetos de su juego. Pueden ser usadas para crear un menú principal, niveles individuales, y cualquier otra cosa. Piense en cada archivo de escena, como un nivel único. En cada escena, usted va a colocar su ambiente, obstáculos, y decoraciones, el diseño esencial y la construcción de su juego en pedazos.

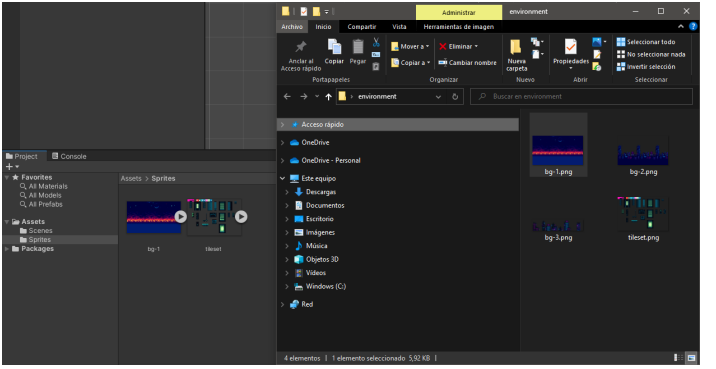
El **Game View** es renderizado por la cámara(s) en su juego. Es la representación de su juego ya finalizado. Necesitará utilizar una o más Cameras (Cámaras) para controlar lo que el jugador ve realmente cuando esté jugando su juego. Para más información acerca de Cámaras, por favor mirar la página del componente Camera.

Modo de Juego

Use los botones en la Barra de Herramientas para controlar el Editor Play Mode y ver cómo se jugará el juego publicado. Mientras en Play mode, cualquier cambio que haga será temporal, y se va reset cuando se salga de Play mode. El UI del Editor se va a oscurecer para avisarle de esto.

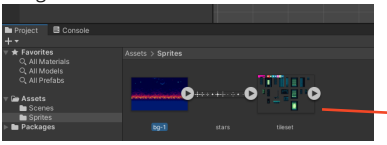
Conociendo esto Ya podemos crear el Proyecto de One Shot!!!

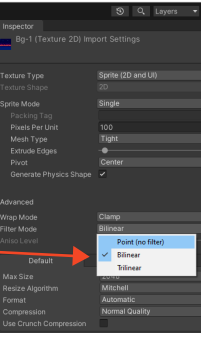
Primero tenemos que añadir las imágenes y los Sprites que seran usados

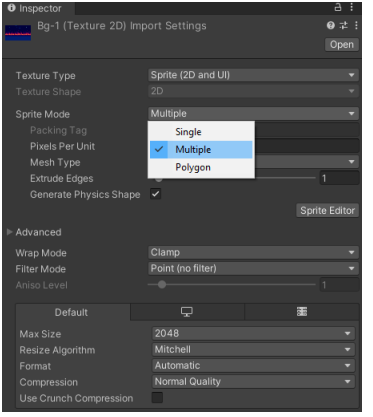


Para agregarlos simplemente agarramos las imágenes deseadas y las arrastramos a la carpeta de Sprites como se muestra en la pantalla.

generalmente cuando agregamos un sprite de pixeles suele verse mal al momento de ser usado en pantalla, para solucionar esto seleccionamos el Stprite y en el Inspector en la opción Filter Mode elegimos point(no filter) para que la imagen se mire clara.



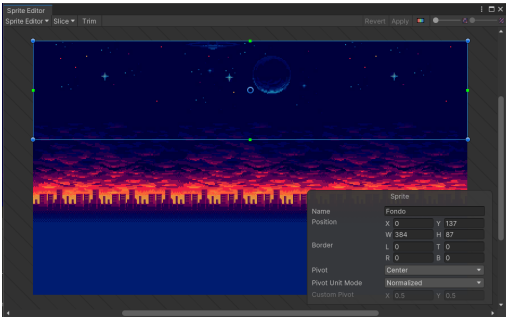




Para seguir modificando la imagen que queremos usar, en el inspector elegimos en el Sprite Mode la opción Multiple y después nos metemos en el sprite editor.

Este paso es para separar la imagen a como se necesite para ser usada en el proyecto desee.

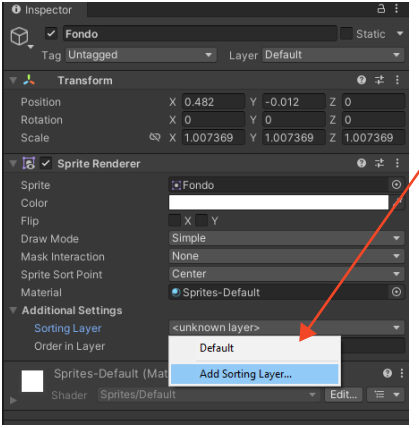
Aquí podemos ver la parte de la edición de la imagen donde recortamos la parte de arriba porque es la que elegimos.



Aplicamos los cambios con este botón  para guardar los cambios que realizamos.

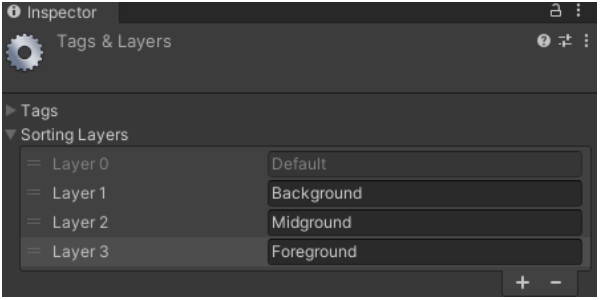
Como la imagen era muy pequeña, configuramos el tamaño de la cámara con la opción Size para hacer más pequeña la cámara elegimos 0.42.

Para seleccionar como queremos las escenas debemos meternos es este apartado, para acceder a este apartado se le da clic al objeto, para definir si este será acomodado en la capa que corresponde.

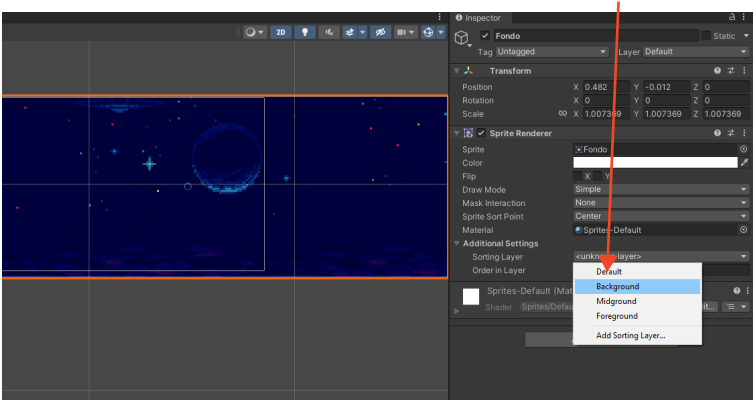


seleccionamos a opción Add Sorting layer...

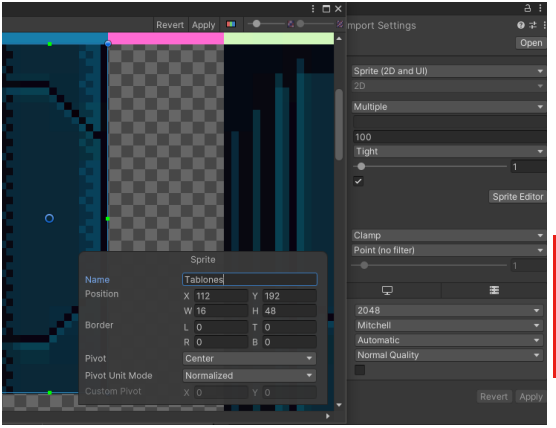
una vez seleccionada la escena aquí se elige donde será acomodado el objeto.



como este será el fondo que usaremos seleccionamos la opción Backgraund

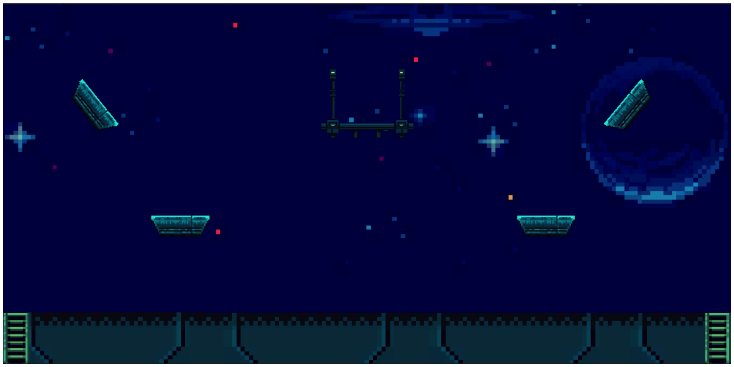


Para añadir otra parte del escenario hicimos el mismo proceso que el fondo y lo editamos de la forma mas adecuada para usarlo

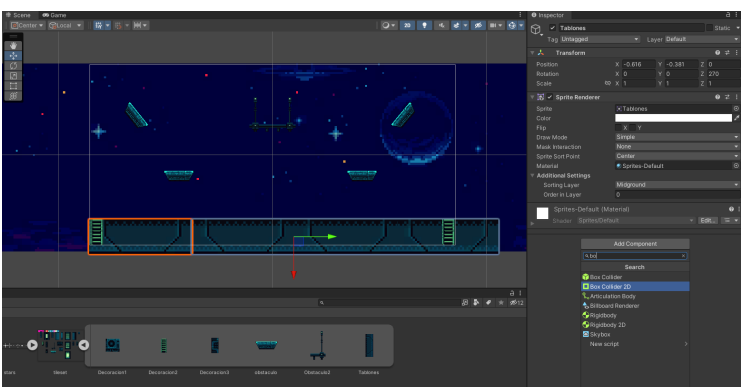


y se usara el mismo procedimiento para el resto de las imágenes

A continuación, creamos un nivel personalizado con las imágenes, del proyecto que decidimos usar, a continuación, se muestra como quedo el nivel.

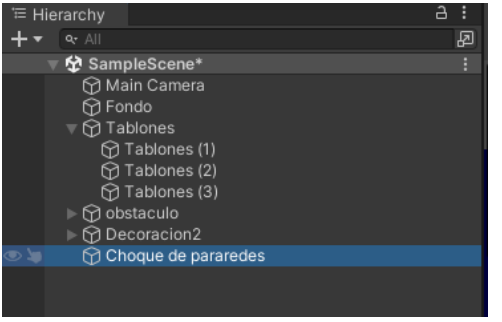


a continuación, vamos a agregar un box colaider 2D a cada parte solida del mapa, para agregarlo se selecciona el objeto y en la zona de Inspector en la parte de Add Component se añade, utilizaremos esto mismo para cada parte del nivel que lo requiera

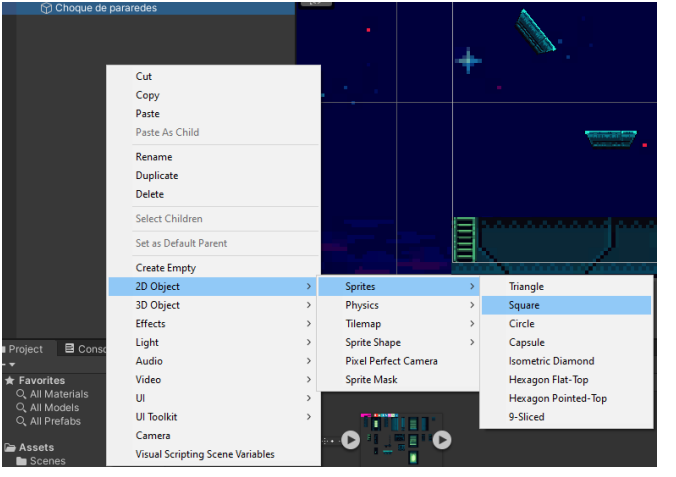


En esta parte podemos ver los objetos que añadimos al nivel.

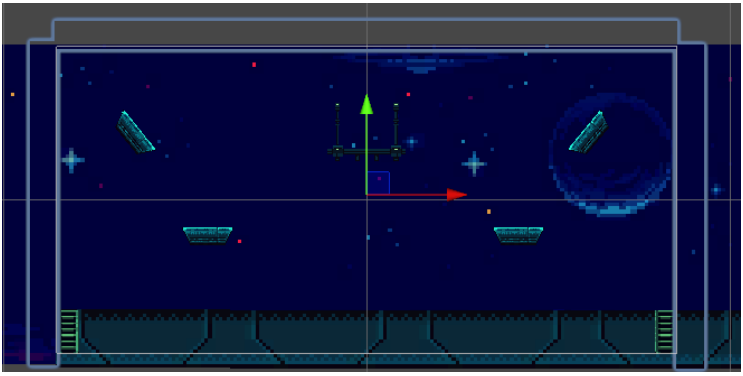
En este apartado creamos una nueva enti para los bordes del nivel



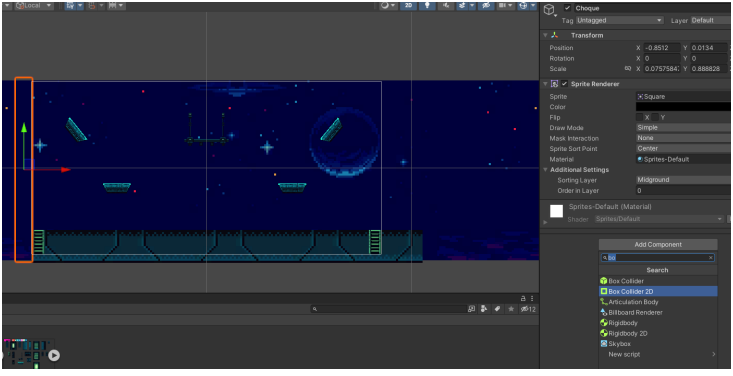
De esta manera se agregan como se muestra en la imagen, se le da clic izquierdo para abrir las opciones.



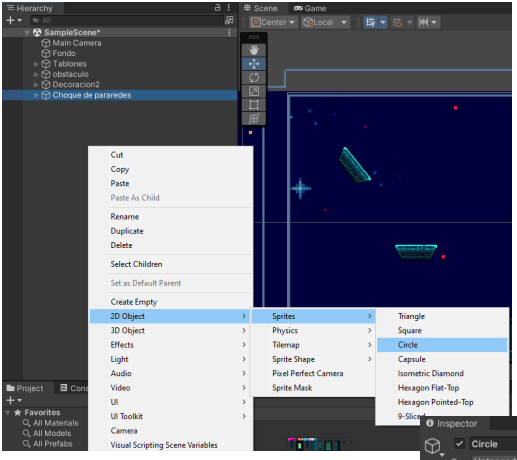
Aquí podemos ver como acomodamos las entidades para marcar los bordes del nivel



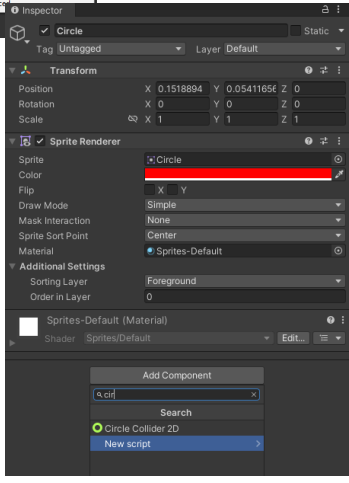
Agregamos un Box Colaider 2D a cada entidad para que tengan colisión con los objetos al igual que hicimos con los objetos.



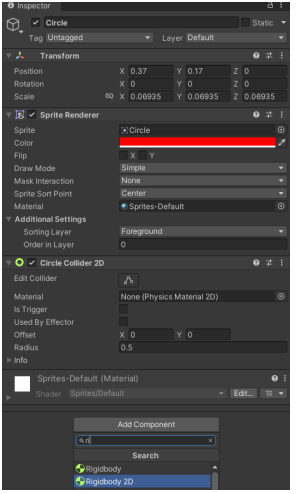
ya teniendo el mapa preparado podemos empezar a hacer los objetos con los que vamos a interactuar, que en este caso serán unos círculos de colores que se crearán directo del Unity y le daremos su propia colisión para que interactúe con los demás objetos.



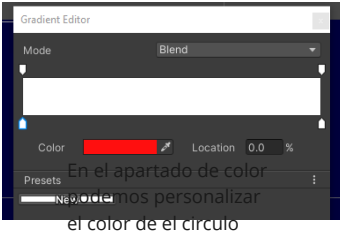
De esta manera creamos el circulo



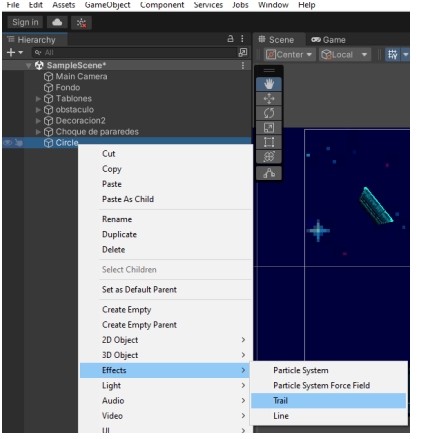
Al igual que los demás objetos le añadimos un Circle Colaider 2D porque es un circulo



También se le agrega un Rigibody 2D para que colisione con los demás Rigidbody del resto del mapa.

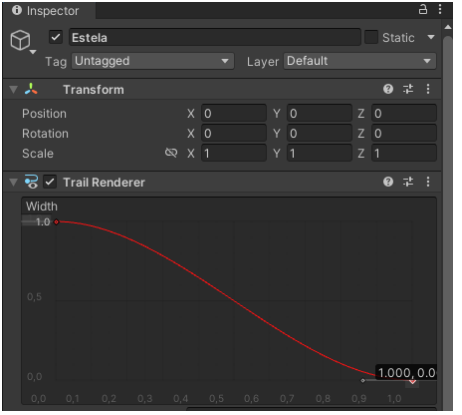


En el apartado de color podemos personalizar el color de el circulo para tener variación.



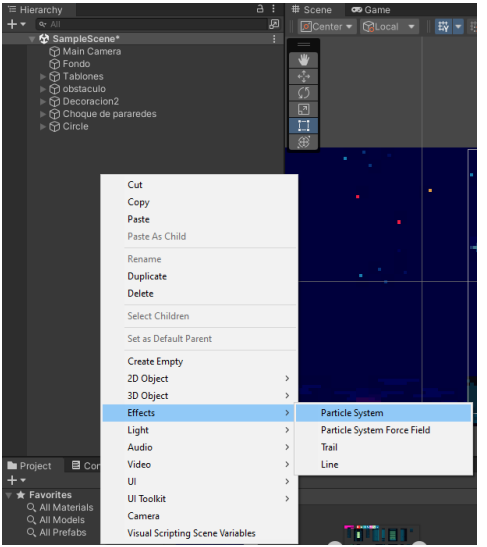
ya teniendo el circulo creado le añadimos un efecto llamado Trail que básicamente añade una trayectoria, como se mira en la imagen,

cuando seleccionamos el circulo en el inspector nos sale el efecto que agregamos que es el Trail renderer y en este le ajustamos el tamaño y el tiempo para crear la ilusión precisa de una trayectoria

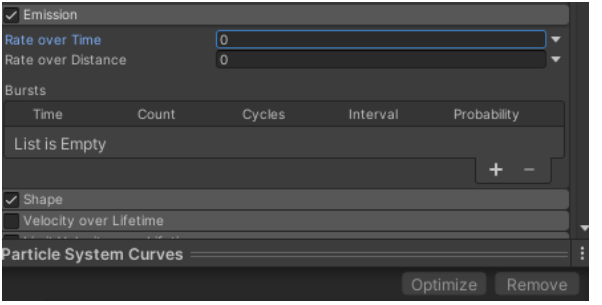


En la opción time elegimos que tanto permanece activa esta animación

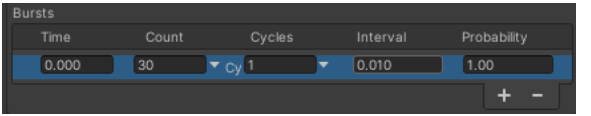
A continuación, añadimos efectos de partícula de la siguiente manera



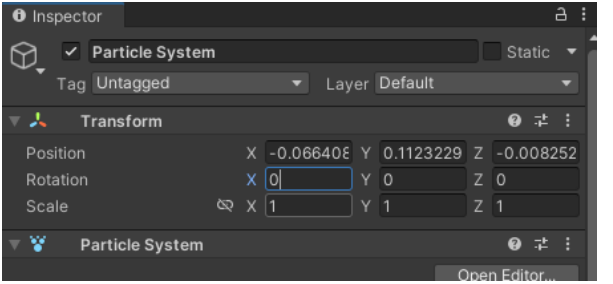
Damos clic derecho en el apartado de hierarchy y añadimos effets la parte de particle System, estas seran usadas para dar un efecto más agradable a la vista,



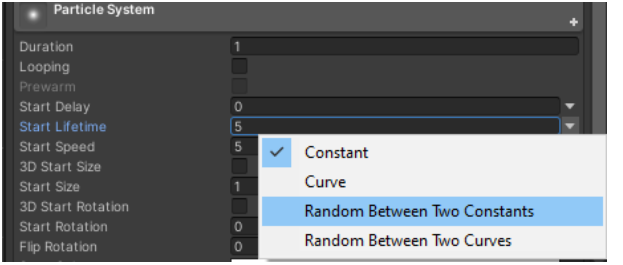
Estos son los parámetros que tienen las partículas



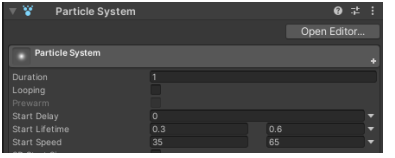
La probabilidad y los intervalos definen que tantas partículas salen



Modificamos los parámetros como se muestran en las imágenes

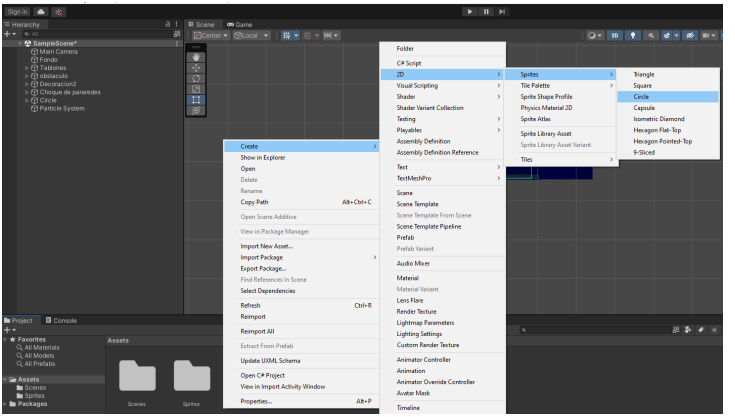


Y le metemos la opción Ramdom between two Constants para tener siempre un efecto aleatorio con estas particula.

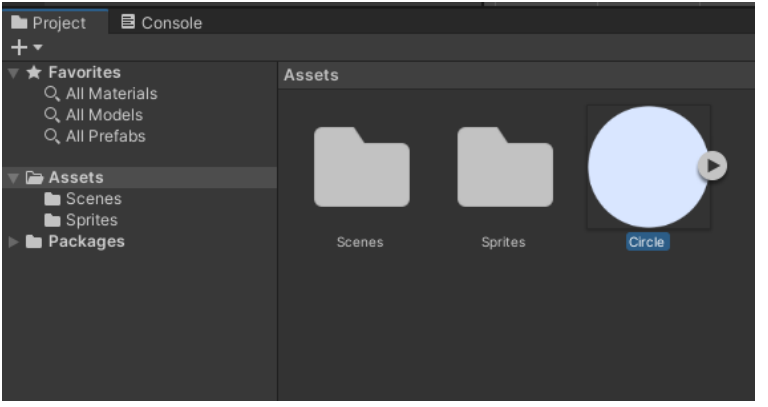


Estas son las opciones que elegimos en las partículas

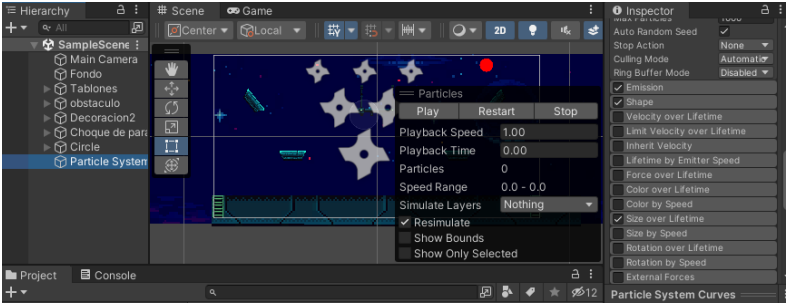
teniendo las partículas listas nos metemos a la carpeta Assets dando clic izquierdo y seleccionando las siguientes opciones Create>2D>Sprites>Circle



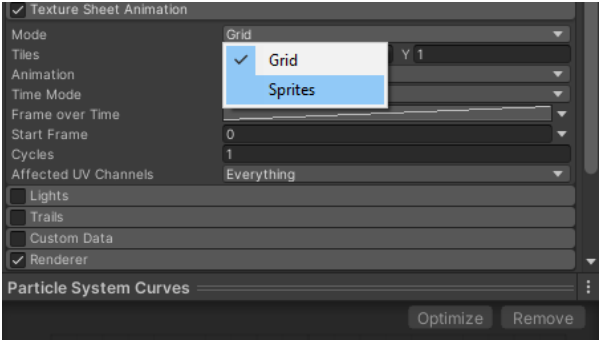
Así es como quedaría en los Assets

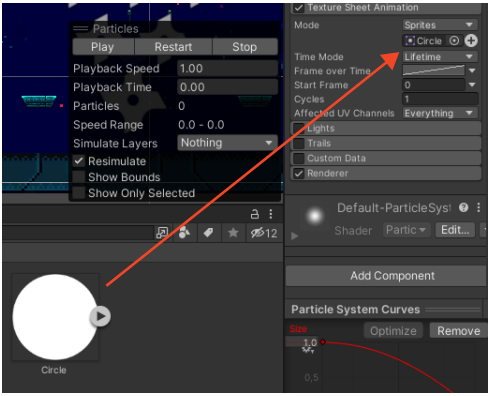


Se seleccionan las partículas para mover los ajustes y ponerlos como se muestras a continuación en la imagen, seleccionando Emission, Shape, Size over Lifetime.

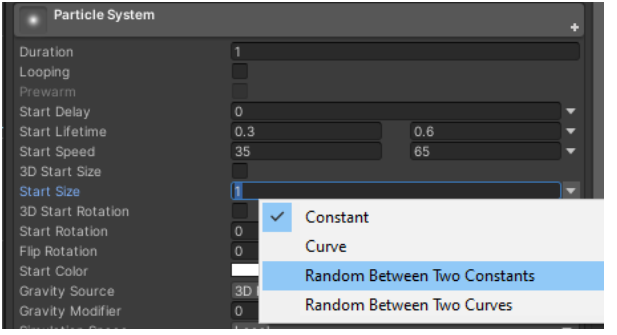


Tienes que cambiar el modo de Grid a Sprite



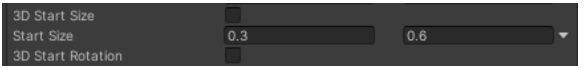


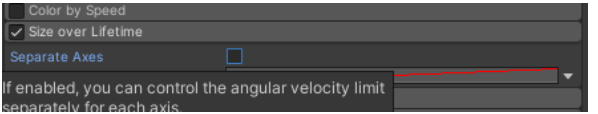
Arrastramos el Circulo a esta casilla debajo de los sprites



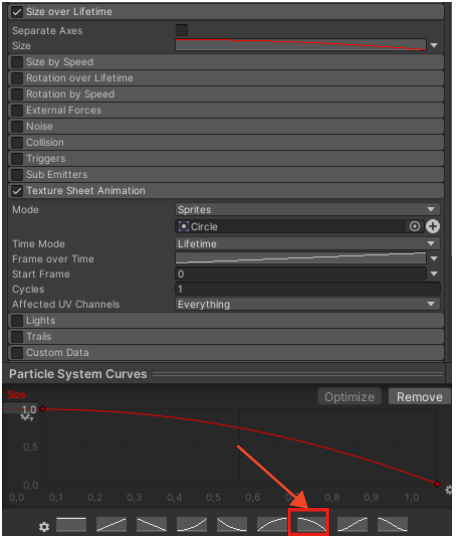
Seleccionamos esta casilla y elegimos la opción Random Between Two Constants para que las partículas se generen de manera aleatoria

A continuación, elegimos estos ajustes para la generación de las mismas.

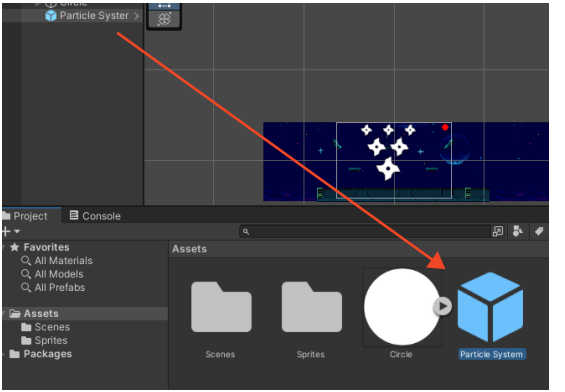




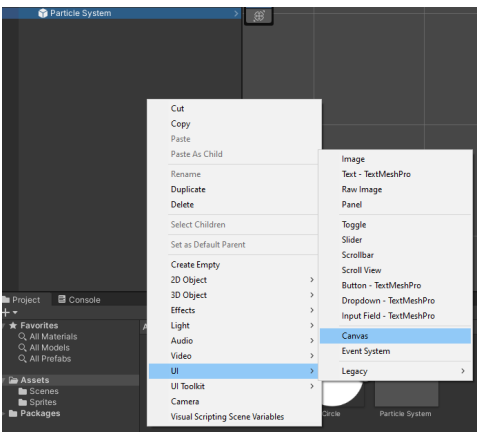
En el Particle System Curves elegimos la curva caída



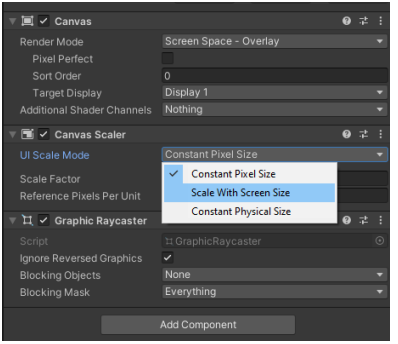
Arrastramos Particle System a el apartado de Assets de la parte de abajo como se muestra



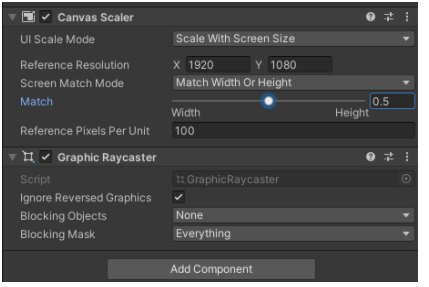
Añadimos un camvas de la siguiente manera a la particle System



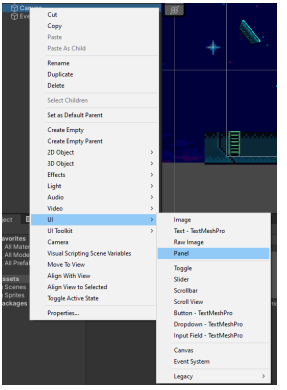
Seleccionamos el Canvas y en la parte de Canvas Scaler marcamos la opción Scale With Screen Size

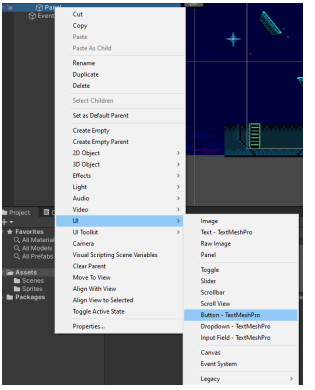


En los demás ajustes elegimos estas opciones para continuar

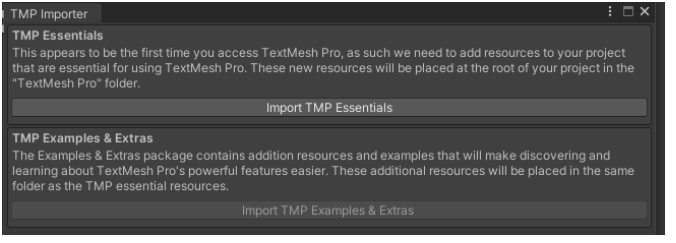


Agregamos un panel y un Button para que cuando el jugador gane el panel aparecerá con el texto de que gano

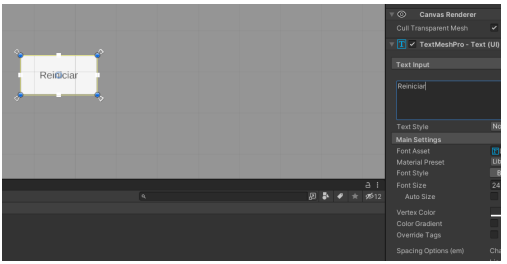




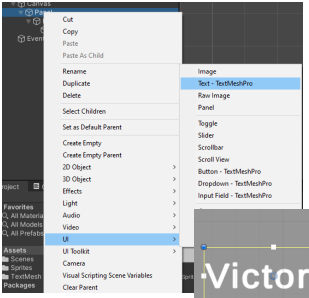
Seleccionamos la opción Importer, y importamos



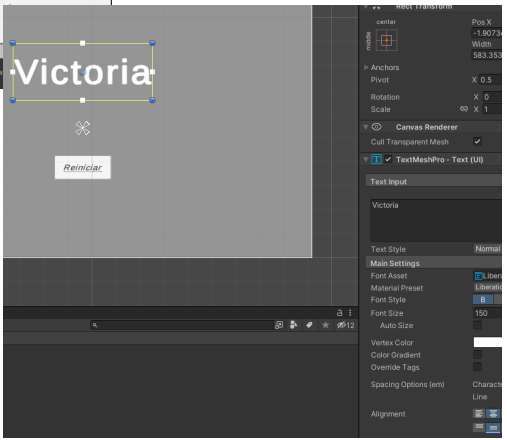
En el Text Input ponemos el texto que queremos que muestre el boton



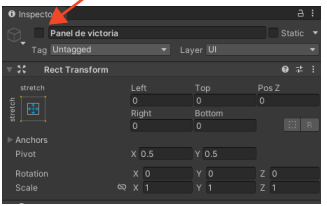
En el panel agregamos un TextMaeshpro



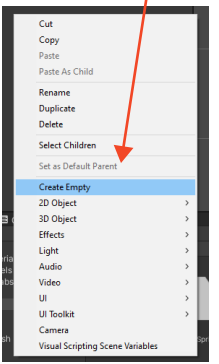
De la misma manera Ponemos el texto que en este caso es Victoria para marcar el fin del juego y que el jugador sepa que lo ha completado



Desmarcamos este punto

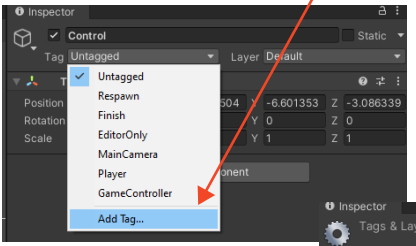


Seleccionamos el Create Empty

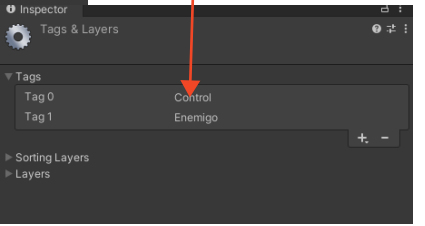


Al nombre le pusimos Control

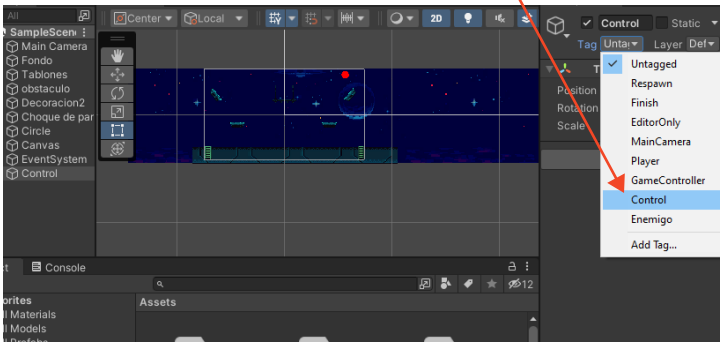
Lo seleccionamos y añadimos un Add Tag...



Lo siguiente es crear dos etiquetas, a una le vamos a poner Control y a la otra Enemigo

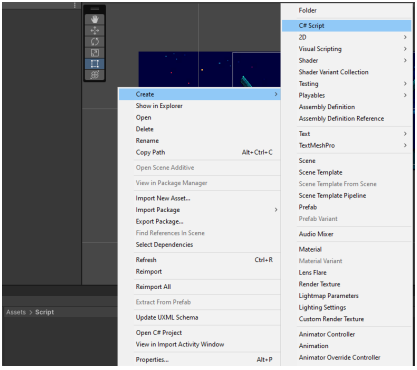


Al objeto control le añadimos la etiqueta control de la siguiente manera, primero elegim
el objeto control y en la opción tag seleccionamos control

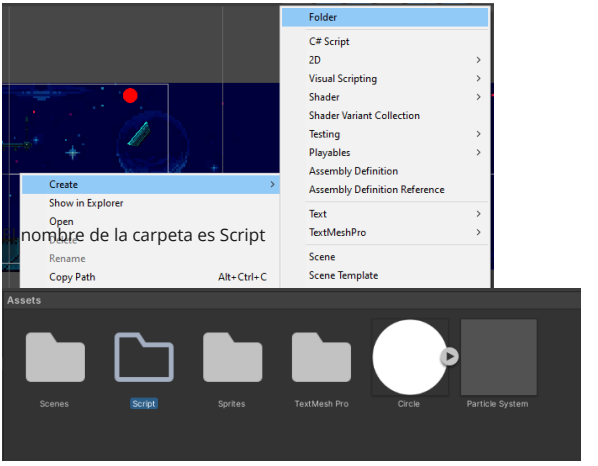


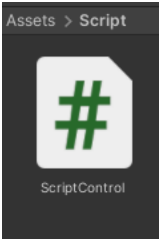
Y al circulo le ponemos la del enemigo de la misma manera

Ahora crearemos un C# Script para empezar
con la primera parte de la programación,
como se muestra en la imagen



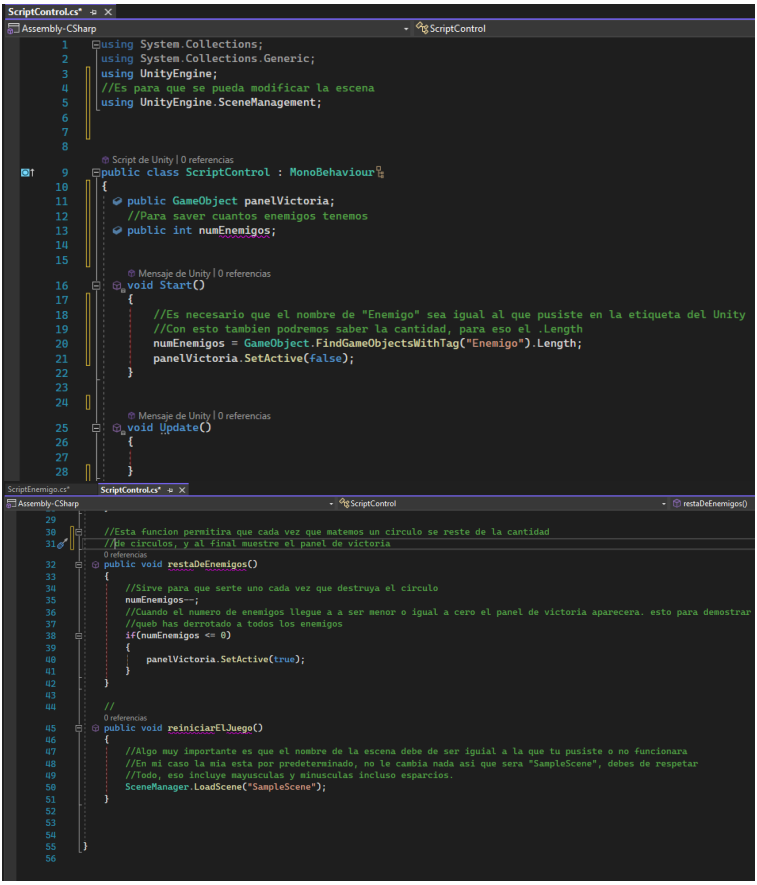
Ahora que vamos a empezar con el código para ello vamos a crear una carpeta





Le damos doble click y podemos empezar a programar

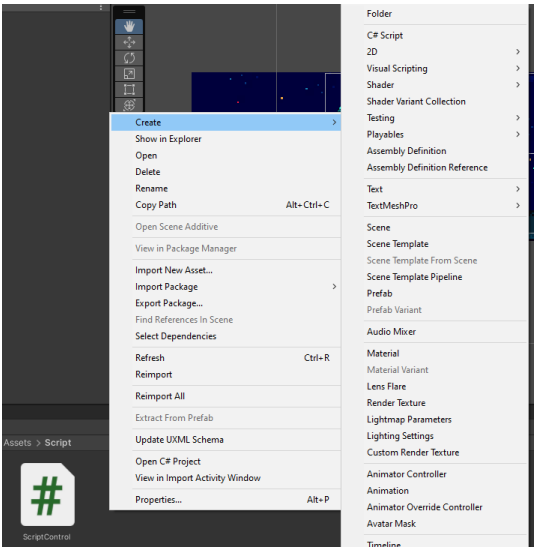
una vez adentro de la programación vamos a usar este código, la parte verde son los comentarios de como funciona el código, por lo cual se pueden omitir.



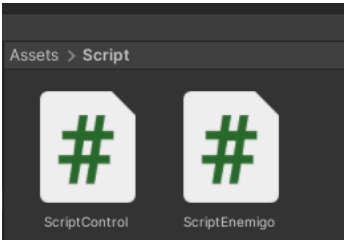
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
//Es para que se pueda modificar la escena
using UnityEngine.SceneManagement;



Script de Unity | 0 referencias
public class ScriptControl : MonoBehaviour
{
    public GameObject panelVictoria;
    //Para saver cuantos enemigos tenemos
    public int numEnemigos;


    Mensaje de Unity | 0 referencias
    void Start()
    {
        //Es necesario que el nombre de "Enemigo" sea igual al que pusiste en la etiqueta del Unity
        //Con esto tambien podremos saber la cantidad, para eso el .Length
        numEnemigos = GameObject.FindGameObjectsWithTag("Enemigo").Length;
        panelVictoria.SetActive(false);
    }


    Mensaje de Unity | 0 referencias
    void Update()
    {

    }
```

```
    //Esta funcion permitira que cada vez que matemos un circulo se reste de la cantidad
    //de circulos, y al final muestre el panel de victoria
    0 referencias
    public void restaDeEnemigos()
    {
        //Sirve para que serte uno cada vez que destruya el circulo
        numEnemigos--;
        //Cuando el numero de enemigos llegue a a ser menor o igual a cero el panel de victoria aparecera. esto para demostrar
        //queb has derrotado a todos los enemigos
        if(numEnemigos <= 0)
        {
            panelVictoria.SetActive(true);
        }
    }

    //
    0 referencias
    public void reiniciarElJuego()
    {
        //Algo muy importante es que el nombre de la escena debe de ser iguial a la que tu pusiste o no funcionara
        //En mi caso la mia esta por predeterminado, no le cambia nada asi que sera "SampleScene", debes de respetar
        //Todo, eso incluye mayusculas y minusculas incluso espacios.
        SceneManager.LoadScene("SampleScene");
    }


}
```

Ahora vamos a pasar el código de los enemigos, que en este caso son los enemigos.



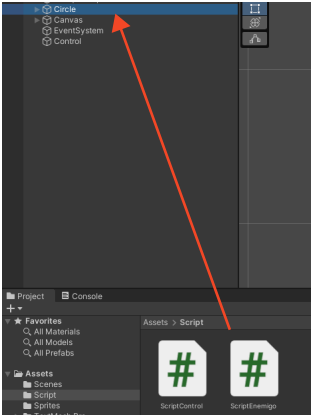
Creamos un script para el enemigo



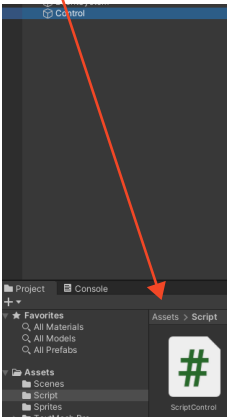
Y ponemos este código

```
ScriptEnemigo.cs  ScriptControl.cs
Assembly-CSharp                                   ScriptEnemigo
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    Script de Unity | 0 referencias
5   public class ScriptEnemigo : MonoBehaviour
6   {
7       //Creamos un dato tipo publico para las particulas
8       public GameObject animacionParticulas;
9       public ScriptControl Scontrol;
10
11      //para el el movimiento del mouse
        Mensaje de Unity | 0 referencias
12      private void OnMouseDown()
13      {
14          //instancia del efecto que emeos creado de particulas
15          Instantiate(animacionParticulas, transform.position, Quaternion.identity);
16          //Esto es para que reste uno despues de que instancie
17          Scontrol.restaDeEnemigos();
18          //Despues de instanciar, que s emuestre el efecto particulas al destruir el enemigo
19          Destroy(gameObject);
20
21      }
22
        Mensaje de Unity | 0 referencias
23      void Start()
24      {
25
26      }
27
28
        Mensaje de Unity | 0 referencias
29      void Update()
30      {
31
32      }
33
34
35  }
36
```

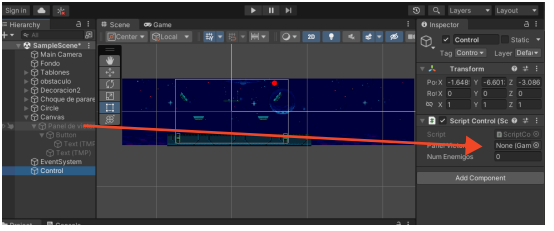
Seleccionamos y arrastramos el Script Enemigo al Circle



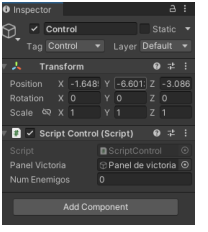
Hacemos lo mismo con el de control



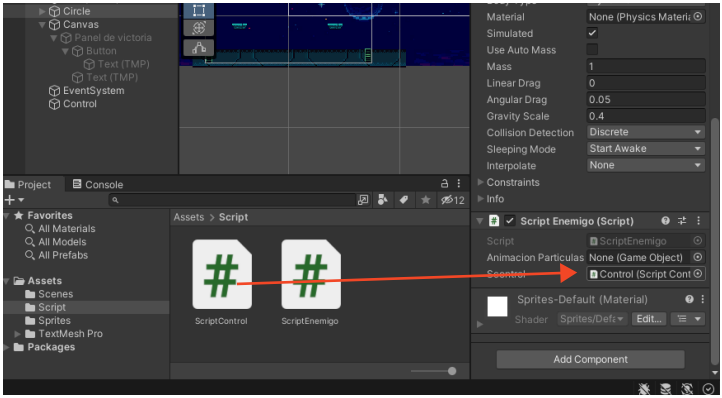
Seleccionamos el Control y a la derecha en inspector nos aparecen varias opciones, prestaremos atención en la opción panel de Victoria, una vez visualizado el panel vamos a seleccionar el panel de victoria que está a la izquierda dentro del Canvas y lo arrastraremos a la parte del panel de victoria de la derecha.



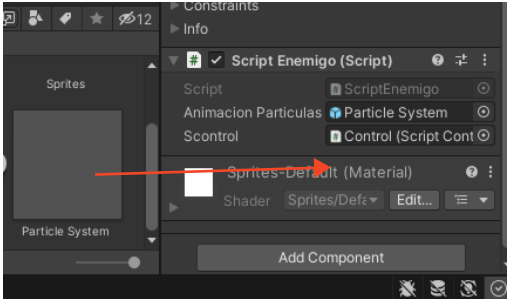
Quedando de la siguiente manera



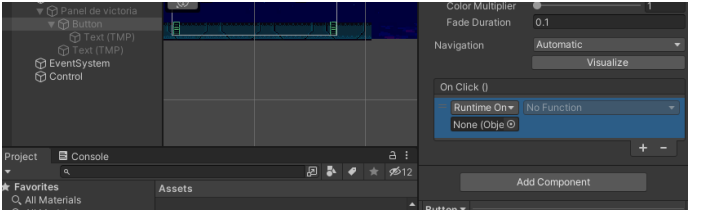
Seleccionamos el circle que es nuestro enemigo al que le vamos a meter las animaciones
el control por lo tanto lo seleccionamos y el script control lo ponemos dentro de la opció
Scontrol



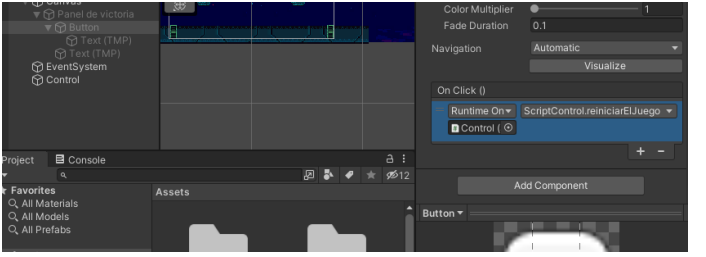
Aquí hacemos lo mismo, pero con las partículas como se muestra en la imagen.



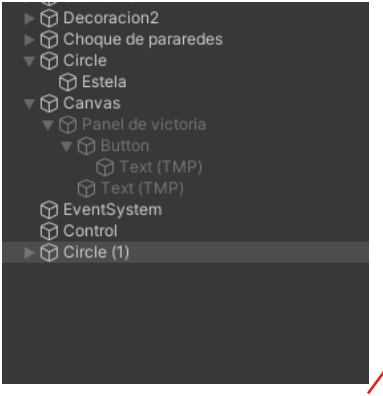
Agregamos un on Click



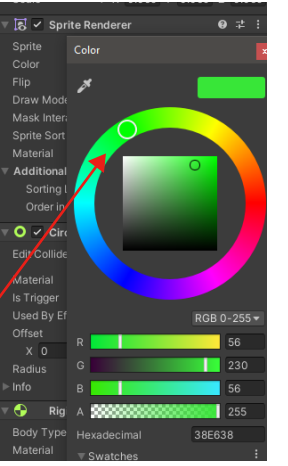
Agregas el controlador y el script reiniciar al juego para que este pueda reiniciar cada vez que presiones el botón y se comience otra vez el juego.
en esta parte ya estamos a punto de acabar de crear el juego.



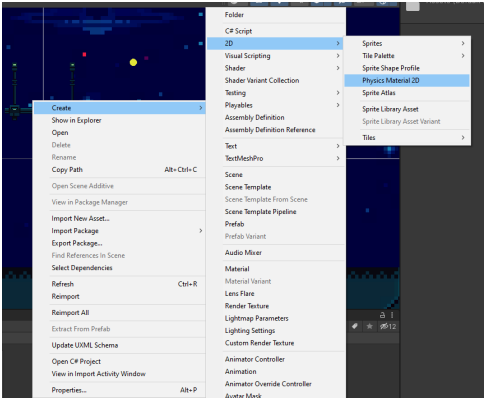
Copiamos y pegamos el circulo para tener más enemigos



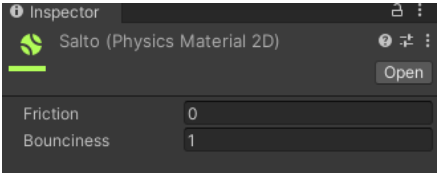
y le cambiamos el color para más variedad



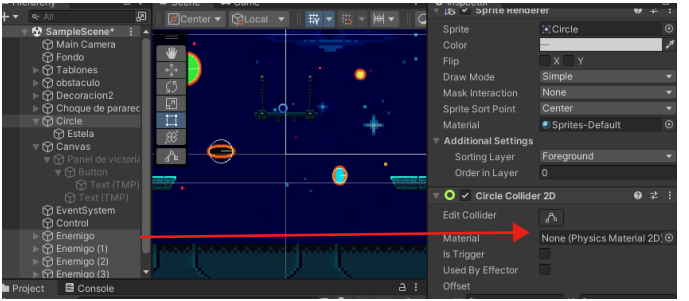
Creamos las físicas que vamos a usar para que los enemigos se muevan



Modificamos las físicas de esta manera para que las pelotas salten



Seleccionamos a todos los enemigos y le agregamos las físicas



Este sería el nivel terminado con toda la variedad de obstáculos y enemigos que se usaron