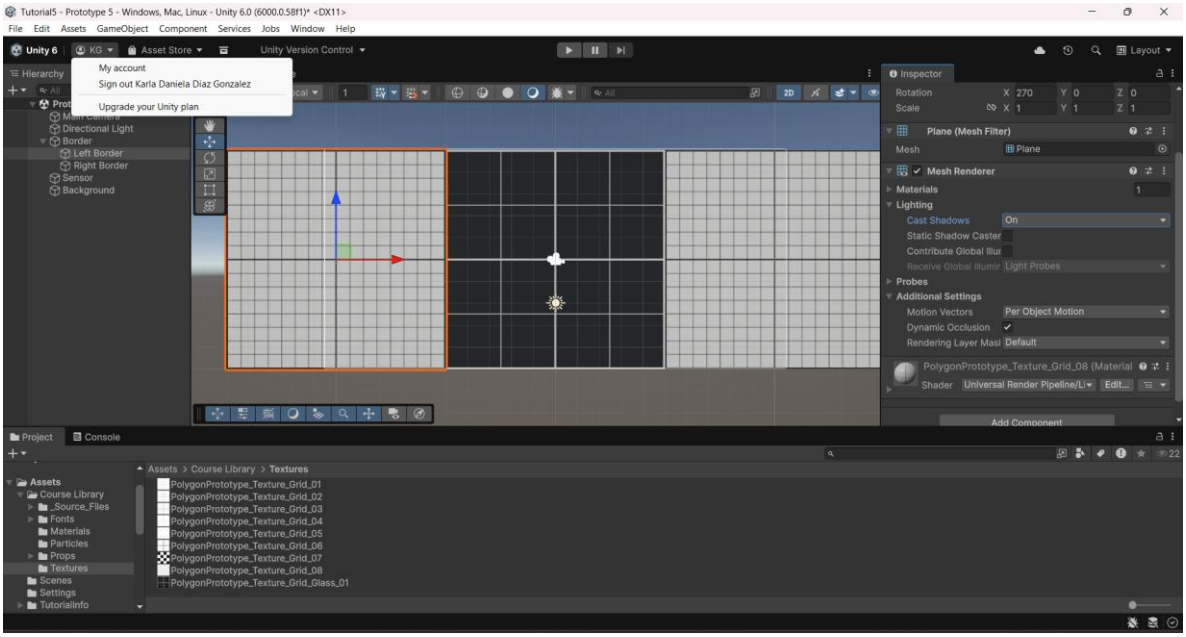
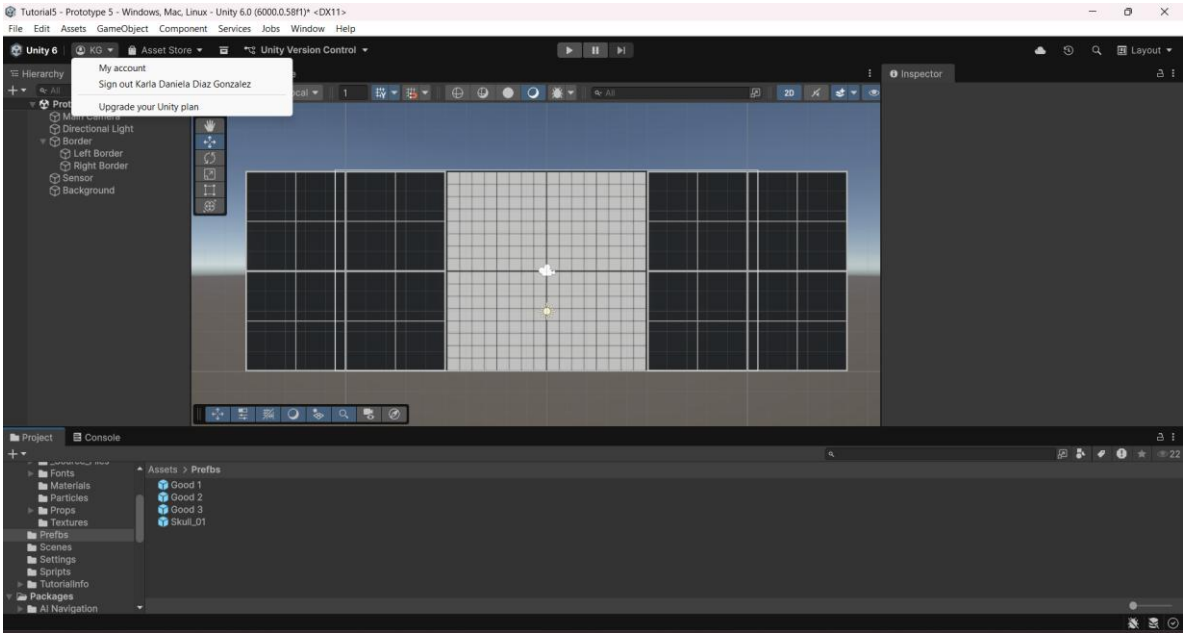
- Abre Unity Hub y crea un proyecto vacío llamado « Prototype 5 » en el directorio de tu curso, utilizando la versión correcta de Unity. Si no recuerdas cómo hacerlo, consulta las instrucciones de la Lección 1.1 - Paso 1.
- Haz clic para descargar los archivos iniciales del prototipo 5 , extrae la carpeta comprimida e importa el archivo .unitypackage a tu proyecto. Si no recuerdas cómo hacerlo, consulta las instrucciones de la Lección 1.1 - Paso 2.
- Abre la escena del Prototipo 5 y, a continuación, elimina la escena de muestra sin guardar.
- Haz clic en el icono 2D en la vista de escena para poner la vista de escena en 2D.

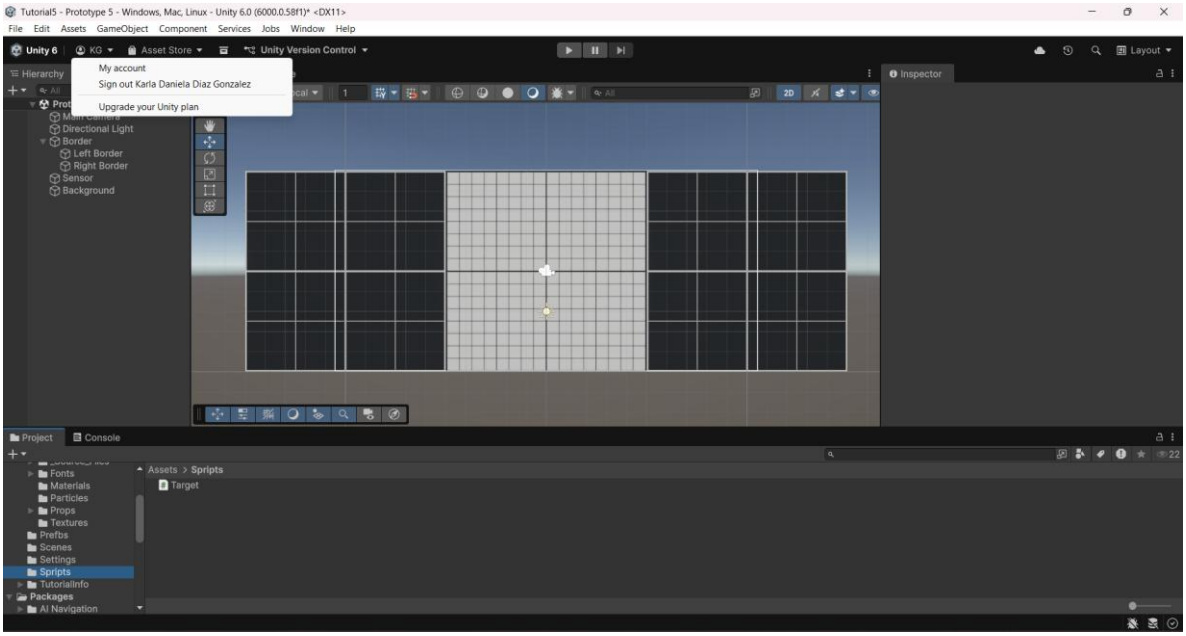
  (opcional) Cambia la textura y el color del fondo y el color de los bordes.



- Desde la Biblioteca , arrastra 3 objetos “buenos” y 1 objeto “malo” a la Escena, y cámbiales el nombre a “ Bueno 1 ”, “ Bueno 2 ”, “ Bueno 3 ” y “ Malo 1 ”.
- Agregar componentes de cuerpo rígido y colisionador de caja
- Crea una nueva carpeta Scripts, un nuevo script " Target.cs " dentro de ella y adjúntalo a los objetos Target.

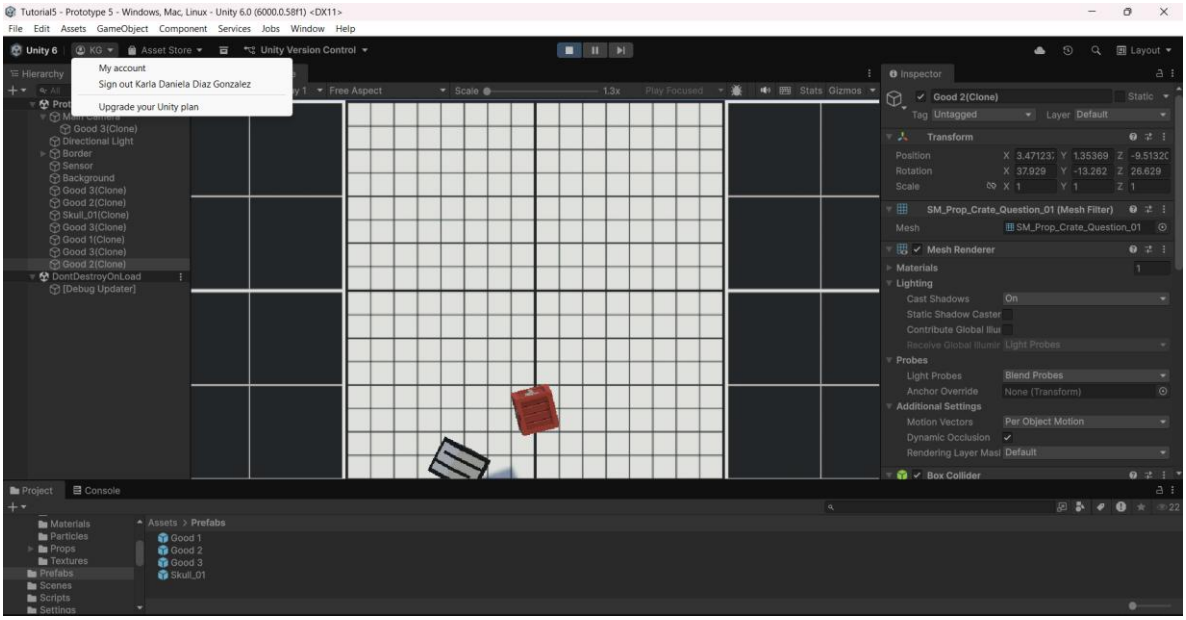
- Arrastra los 4 objetivos a la carpeta Prefabs para crear los "prefabs originales" y, a continuación, bórralos de la escena.

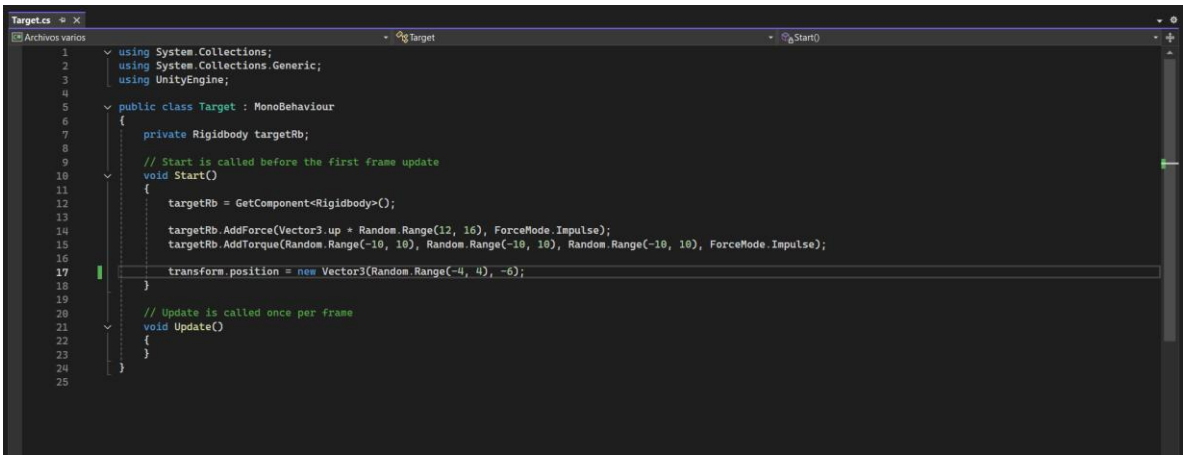




- En Target.cs , declara un nuevo Rigidbody privado llamado targetRb; e inicialízalo en Start().
- En Start() , añade una fuerza ascendente multiplicada por una velocidad aleatoria.
- Añadir un par de torsión con valores xyz aleatorios

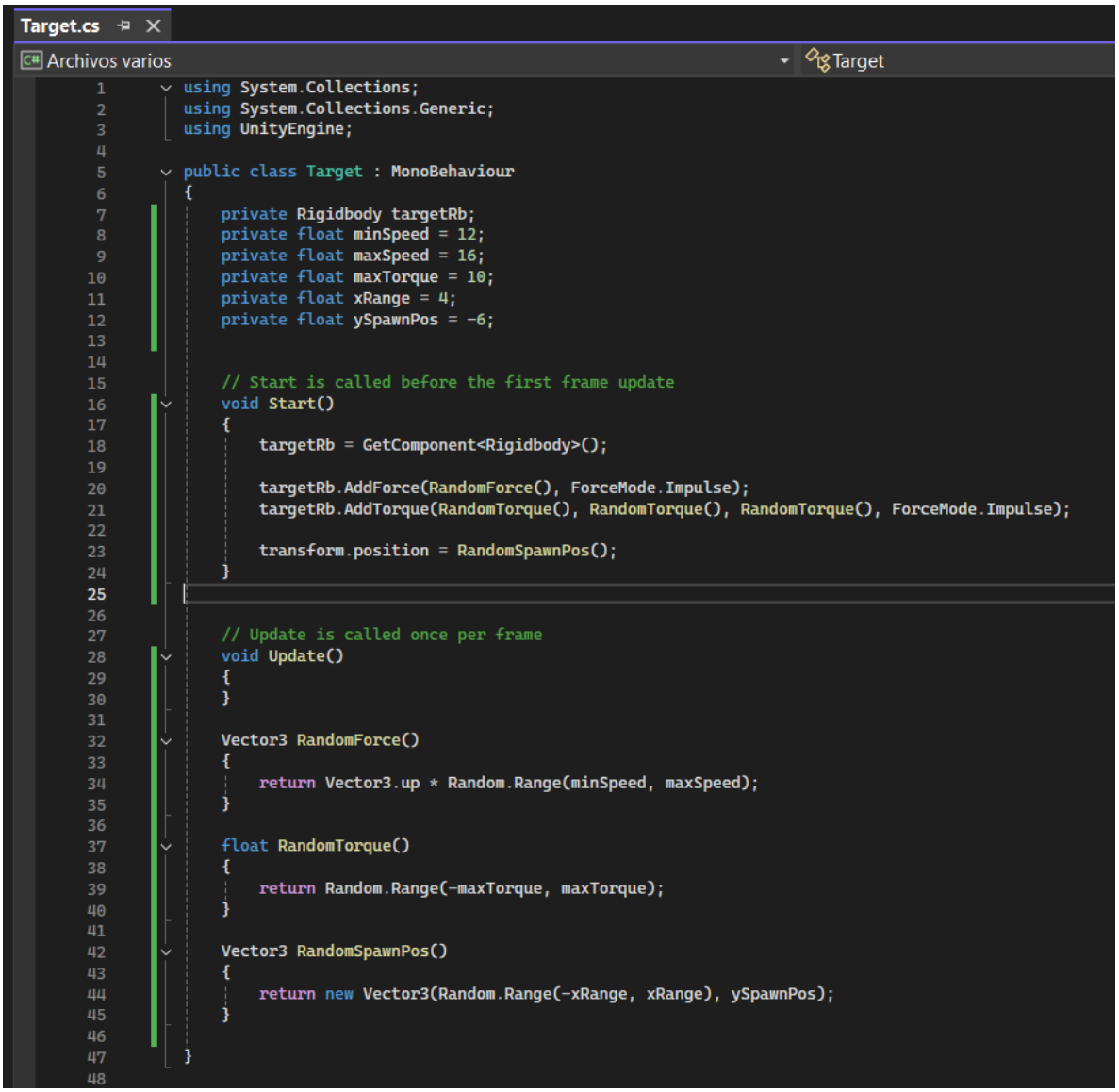
- Establece la posición con un valor X aleatorio.





C:\Users\karla\Downloads\Prototype 5 - Starter Files.zip\Prototype 5 - Starter Files

- Declarar e inicializar nuevas variables privadas de tipo float para minSpeed, maxSpeed, maxTorque, xRange e ySpawnPos ;
- Crea una nueva función para Vector3 RandomForce() y llámala en Start().
- Crea una nueva función para float RandomTorque() y llámala en Start().
- Crea una nueva función para RandomSpawnPos() , haz que devuelva un nuevo Vector3 y llámala en Start().

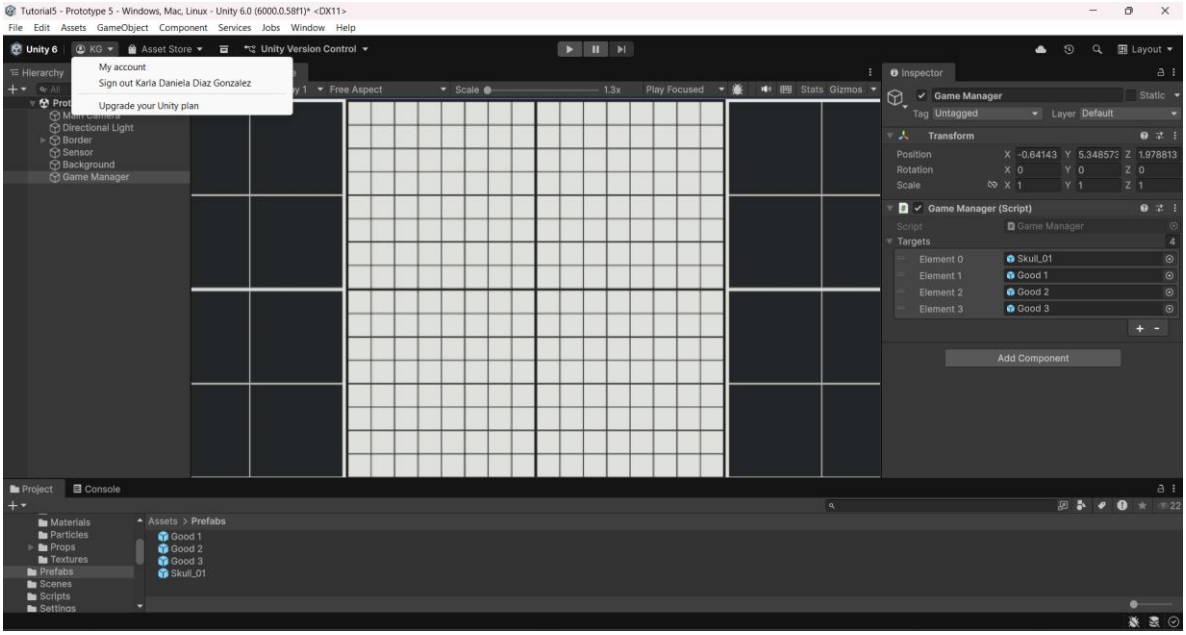


- Crea un nuevo objeto vacío llamado “ Gestor de juegos ” .
- Crea un nuevo script GameManager.cs , adjúntalo al GameObject Game Manager en la ventana Jerarquía y, a continuación, ábrelo.
- Agregue una nueva directiva using a la lista de espacios de nombres en la parte superior de su script: using System.Collections.Generic;
- Declara una nueva lista pública List<GameObject> targets; , luego en el inspector del Administrador de juegos, cambia el tamaño de la lista a 4 y asigna tus prefabs .

Game Manager.cs | Target.cs

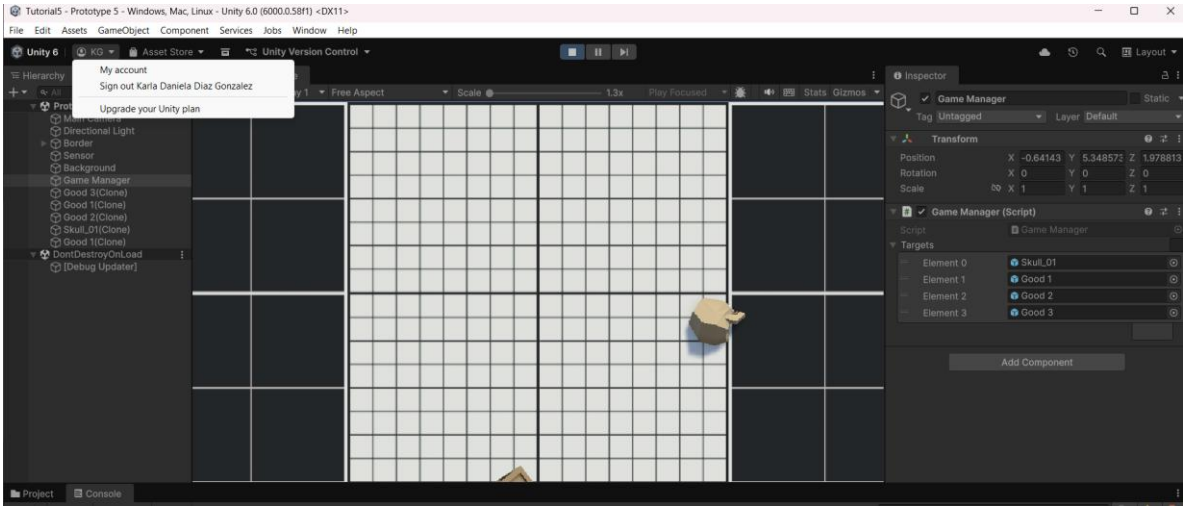
Archivos varios | GameManager

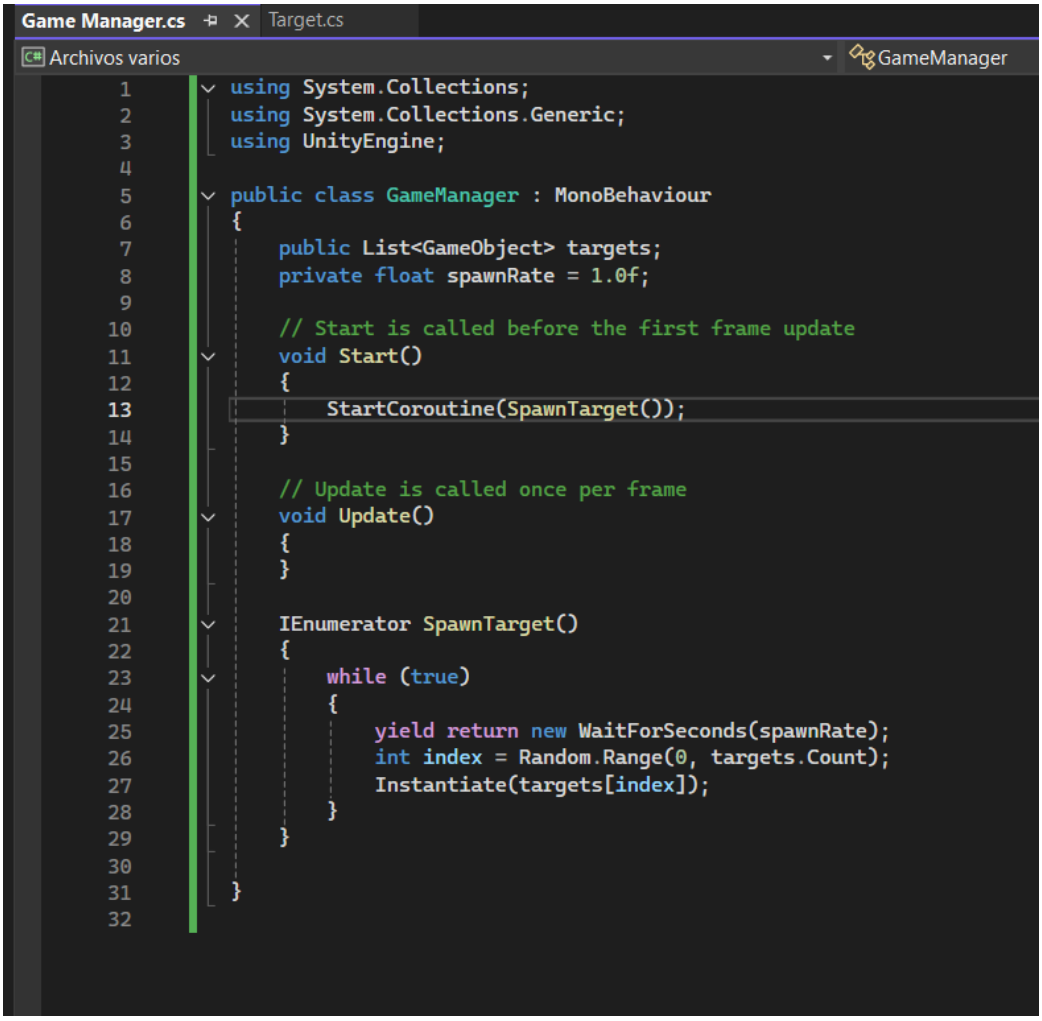
```
1  using System.Collections.Generic;
2  using UnityEngine;
3
4  public class GameManager : MonoBehaviour
5  {
6      public List<GameObject> targets;
7
8      // Start is called before the first frame update
9      void Start()
10     {
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16     }
17 }
18
```



- Declarar e inicializar una nueva variable privada de tipo float llamada spawnRate .
- Crea un nuevo método IEnumerator SpawnTarget().
- Dentro del nuevo método, while(true), espera 1 segundo , genera un índice aleatorio y crea un objetivo aleatorio .
- En Start() , utilice el método StartCoroutine para comenzar a generar objetos.

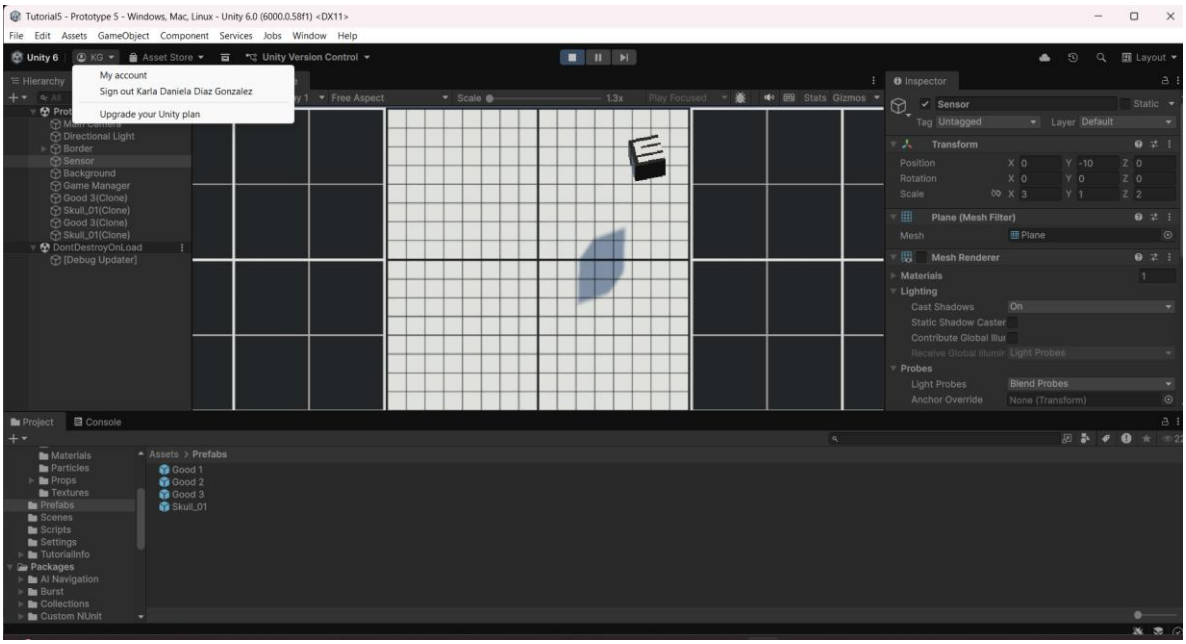
- Para corregir el error en IEnumerator, agregue la siguiente directiva using al principio de su código:
  using System.Collections;



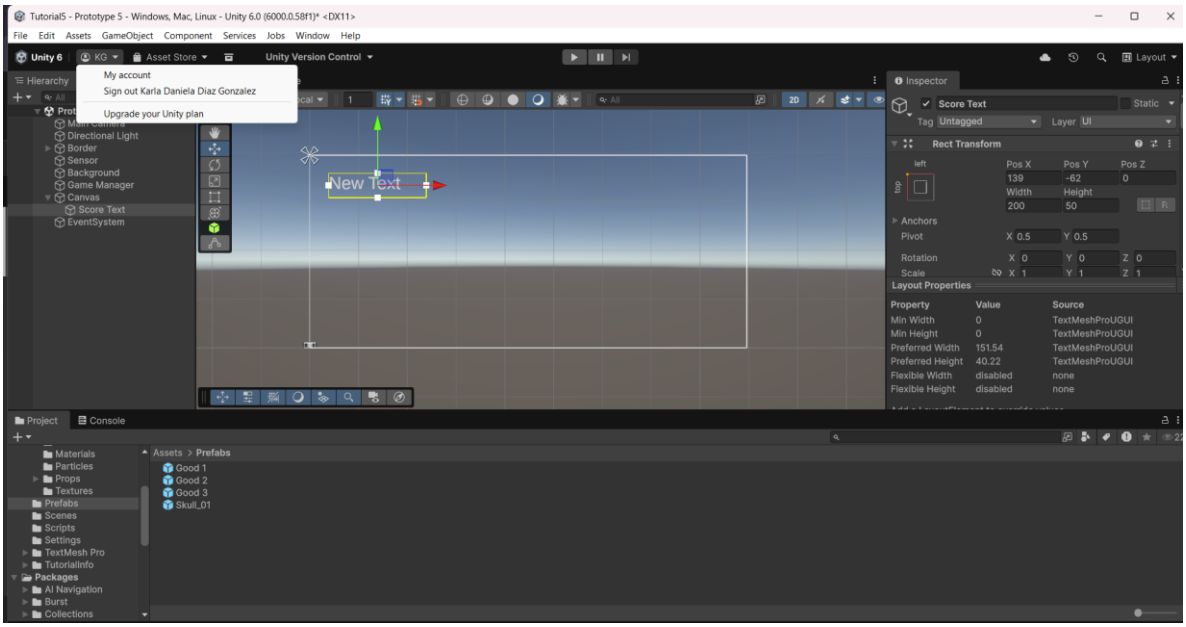


- En Target.cs , agrega un nuevo método para private void OnMouseDown() { } y, dentro de ese método, destruye el objeto del juego.
- Agrega un nuevo método para `private void OnTriggerEnter(Collider other)` y dentro de esa función, destruye el objeto del juego

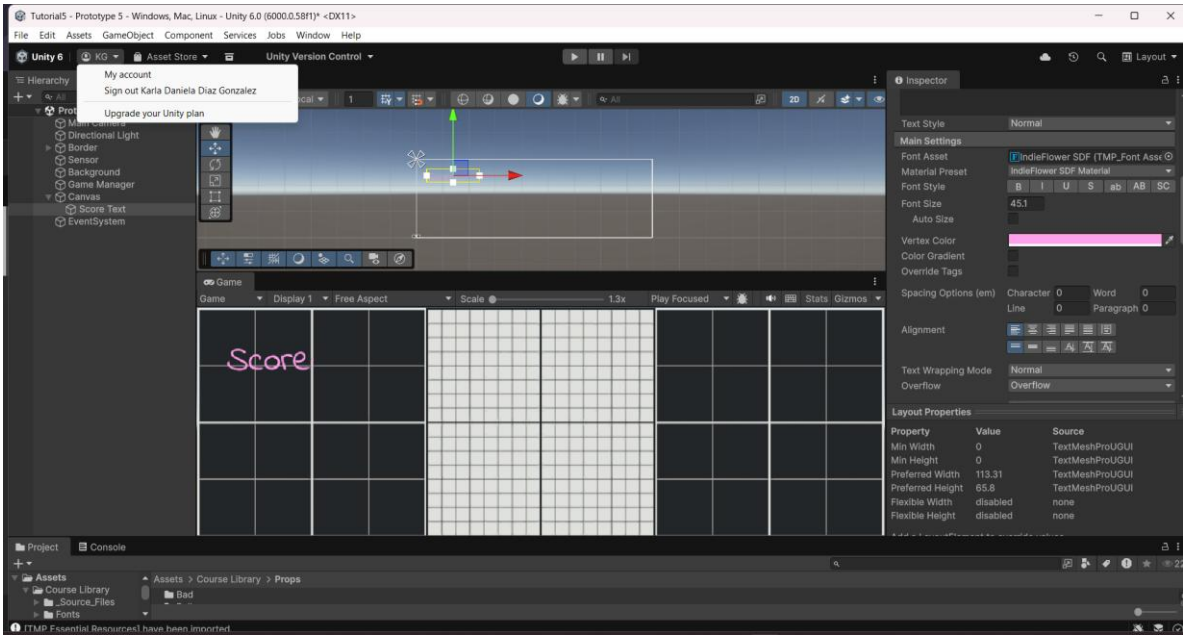
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Target : MonoBehaviour
{
    private Rigidbody targetRb;
    private float minSpeed = 12;
    private float maxSpeed = 16;
    private float maxTorque = 10;
    private float xRange = 4;
    private float ySpawnPos = -2;


    // Start is called before the first frame update
    void Start()
    {
        targetRb = GetComponent<Rigidbody>();

        targetRb.AddForce(RandomForce(), ForceMode.Impulse);
        targetRb.AddTorque(RandomTorque(), RandomTorque(), RandomTorque(), ForceMode.Impulse);

        transform.position = RandomSpawnPos();
    }


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

    }

    private void OnMouseDown()
    {
        Destroy(gameObject);
    }

    private void OnTriggerEnter(Collider other)
    {
        Destroy(gameObject);
    }

    Vector3 RandomForce()
    {
        return Vector3.up * Random.Range(minSpeed, maxSpeed);
    }

    float RandomTorque()
    {
        return Random.Range(-maxTorque, maxTorque);
    }

    Vector3 RandomSpawnPos()
    {
        return new Vector3(Random.Range(-xRange, xRange), ySpawnPos);
    }

}
```



- En la ventana Jerarquía , haga clic con el botón derecho o seleccione + > IU > Texto - TextMeshPro y, a continuación, si se le solicita, seleccione el botón para importar TMP Essentials .
- Cambie el nombre del nuevo objeto a “ Texto de la partitura ” y, a continuación, aleje la imagen para ver el lienzo en la vista de escena .
- Cambie el punto de anclaje para que esté anclado desde la esquina superior izquierda.
- En la ventana Inspector , cambie su Pos X y Pos Y para que se encuentre en la esquina superior izquierda.

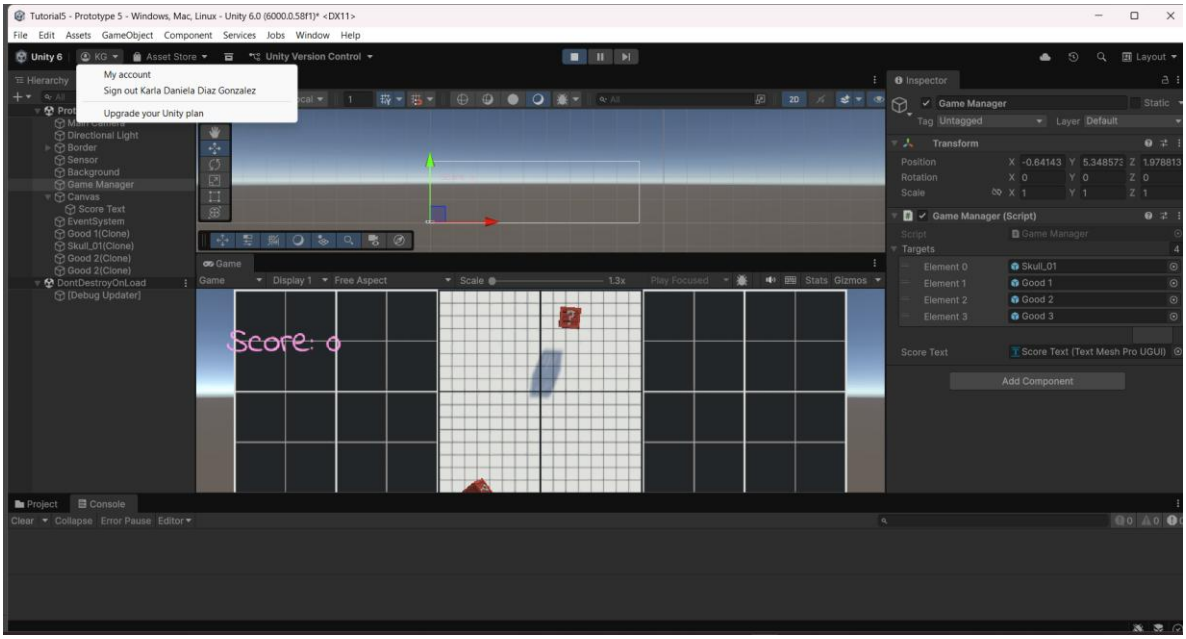


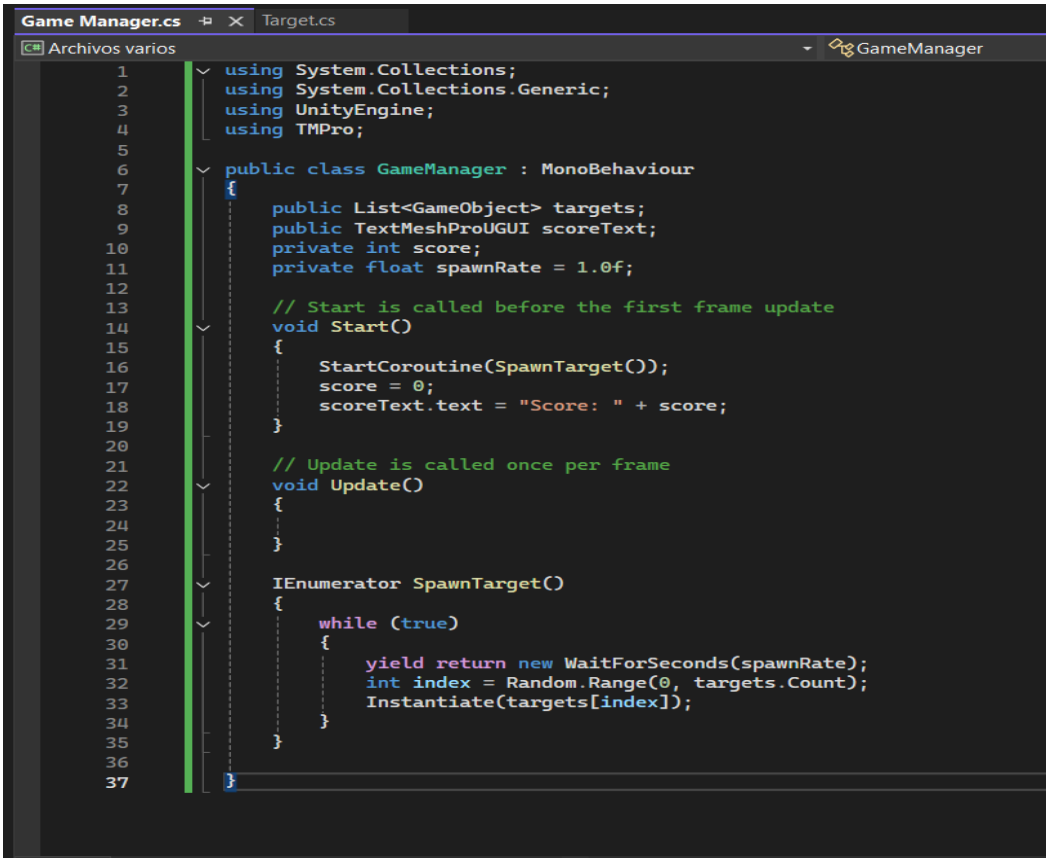
- Cambia su texto a “ Puntuación :”
- Elige una fuente , un estilo , un tamaño y un color de vértice que combinen bien con tu fondo.



- En la parte superior de GameManager.cs , agregue “ using TMPro; ”
- Declara una nueva variable pública TextMeshProUGUI scoreText y, a continuación, asígnale esa variable en el inspector.

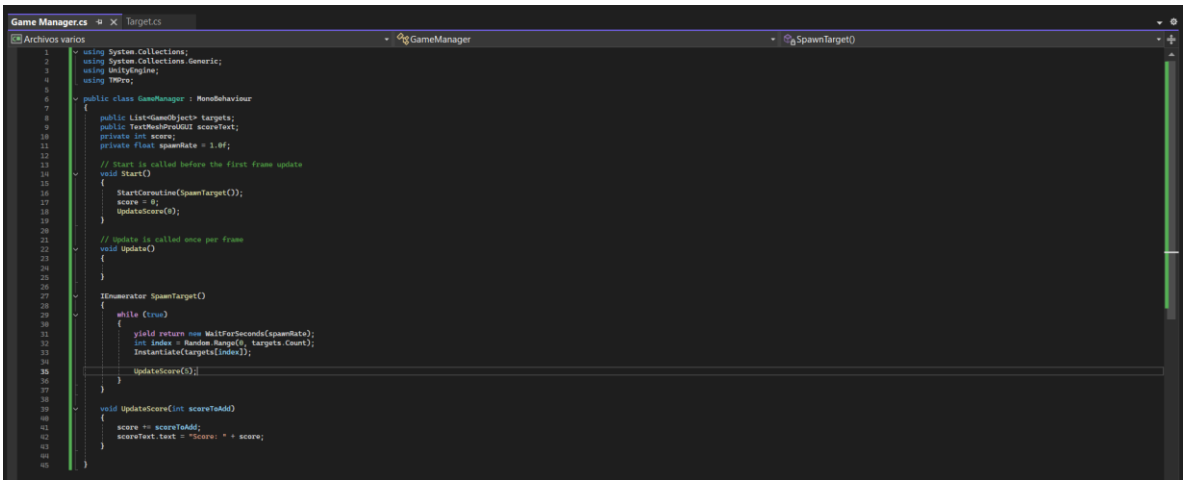
- Crea una nueva variable privada de tipo entero llamada score e inicialízala en Start() como score = 0;
- También en Start() , establezca scoreText.text = "Puntuación: " + score;

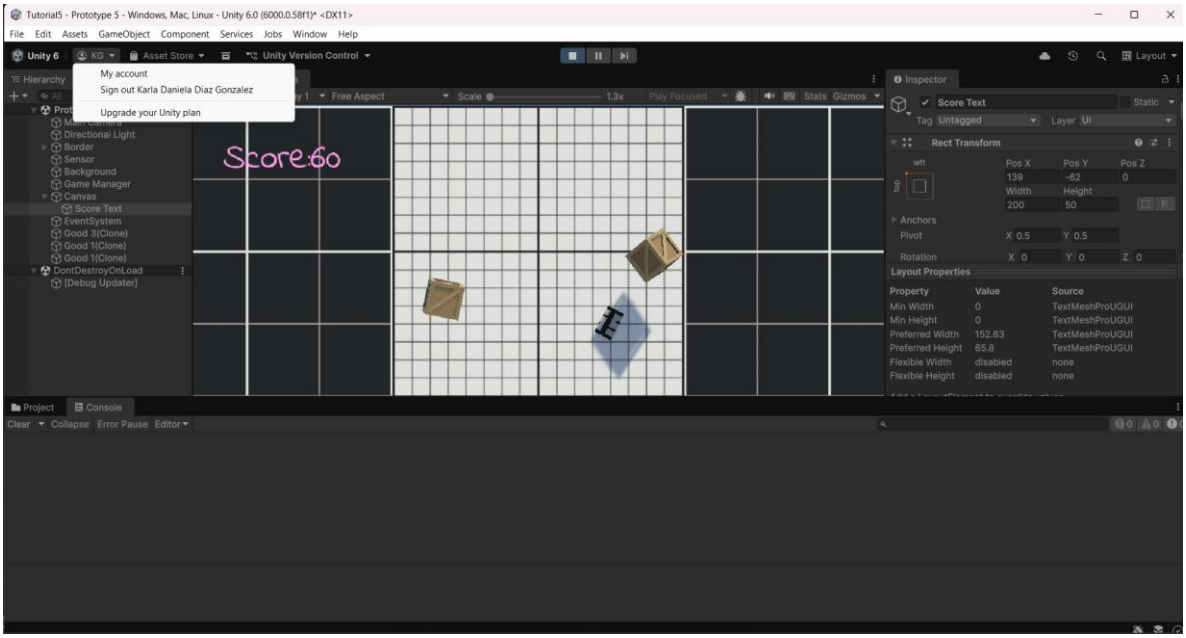




```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class GameManager : MonoBehaviour
{
    public List<GameObject> targets;
    public TextMeshProUGUI scoreText;
    private int score;
    private float spawnRate = 1.0f;

    // Start is called before the first frame update
    void Start()
    {
        StartCoroutine(SpawnTarget());
        score = 0;
        scoreText.text = "Score: " + score;
    }

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

    }

    IEnumerator SpawnTarget()
    {
        while (true)
        {
            yield return new WaitForSeconds(spawnRate);
            int index = Random.Range(0, targets.Count);
            Instantiate(targets[index]);
        }
    }

}
```

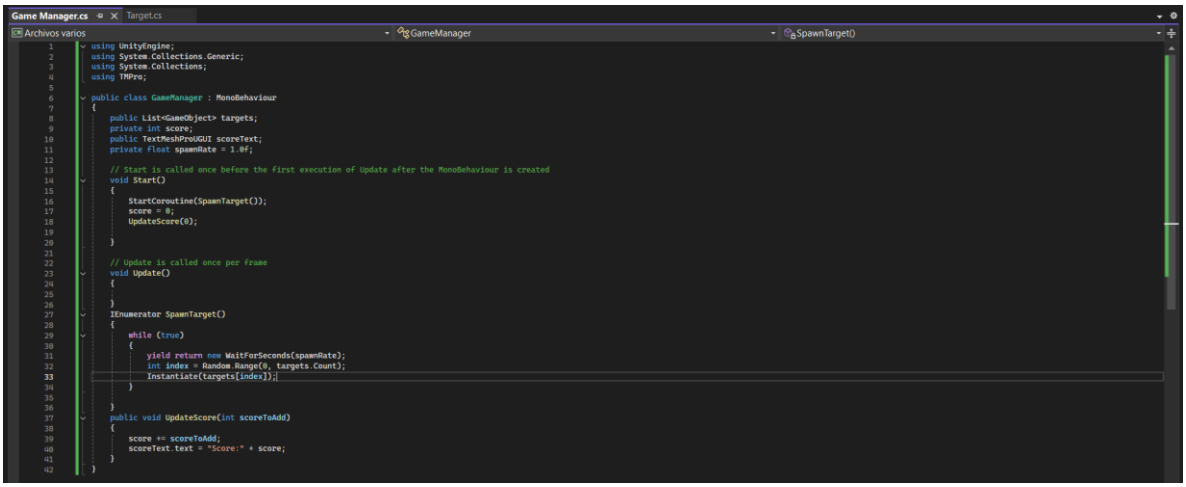
- Crea un nuevo método privado void UpdateScore que requiera un parámetro int scoreToAdd.
- Copia y pega scoreText.text = "Puntuación: " + score; en el nuevo método, luego llama a UpdateScore(0) en Start().
- En UpdateScore() , incremente la puntuación sumando score += scoreToAdd;
- Llama a UpdateScore(5) en la función spawnTarget()

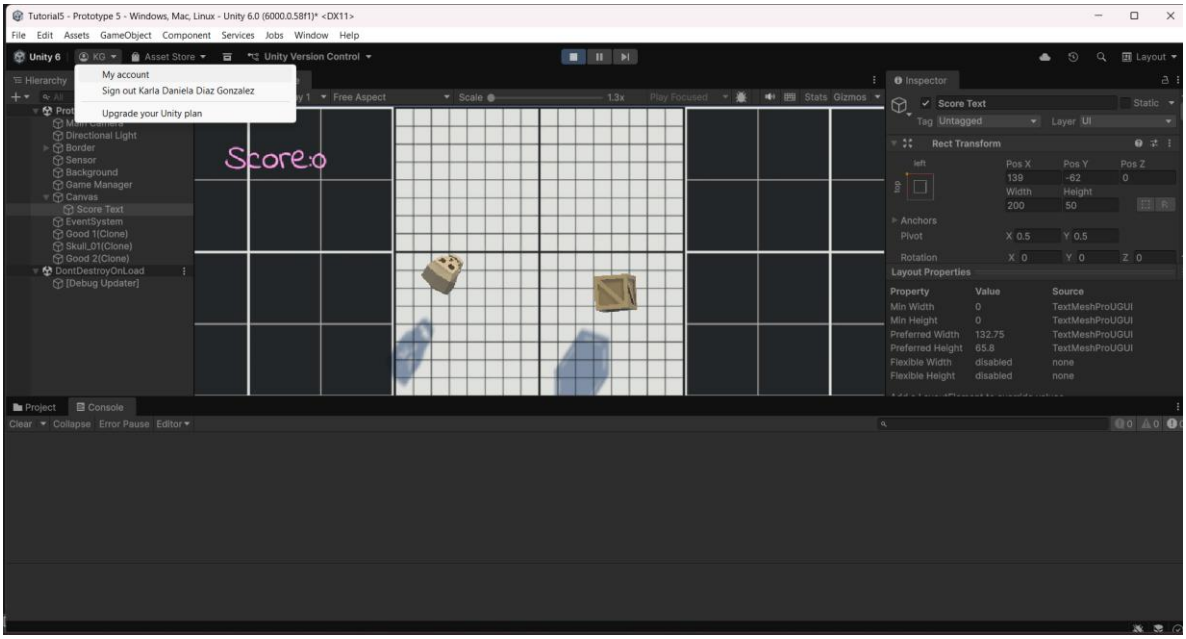




- En GameManager.cs, haga público el método UpdateScore.

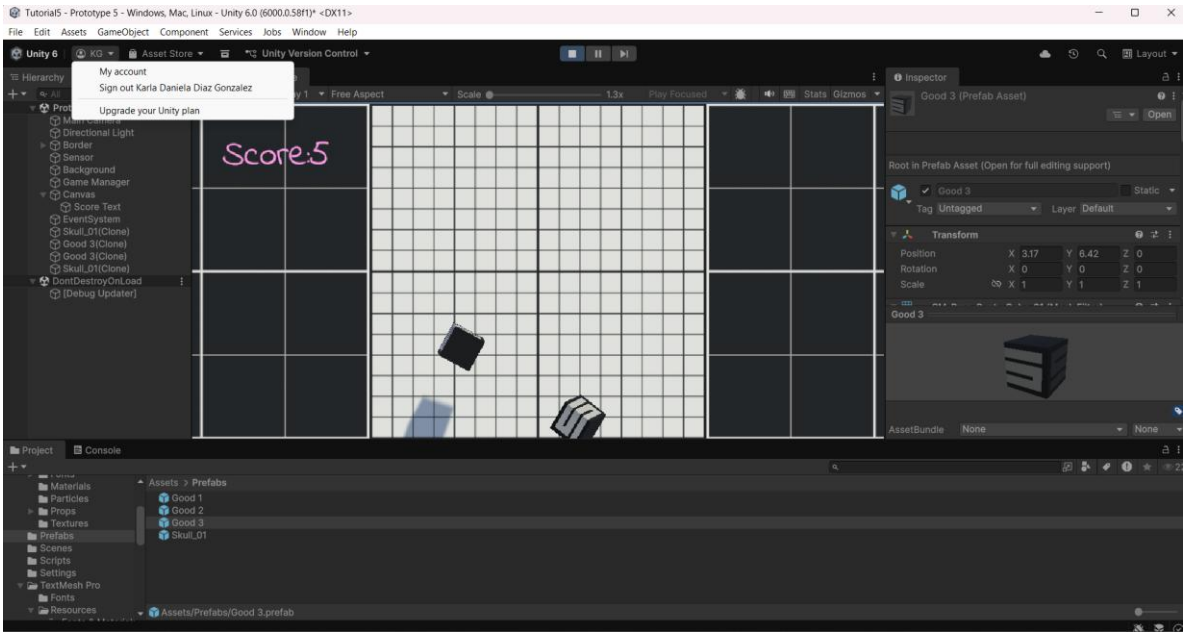
- En Target.cs, cree una referencia a la variable privada GameManager gameManager;
- Inicializa GameManager en Start() usando el método Find()
- Cuando se destruye un objetivo , llama a UpdateScore(5); y luego elimina la llamada al método de SpawnTarget().

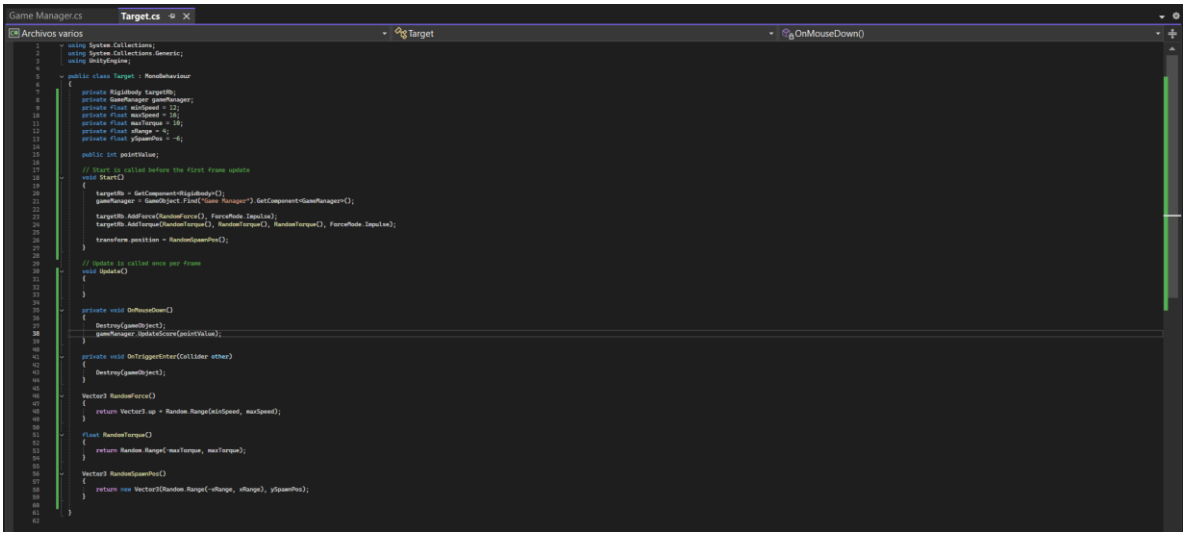




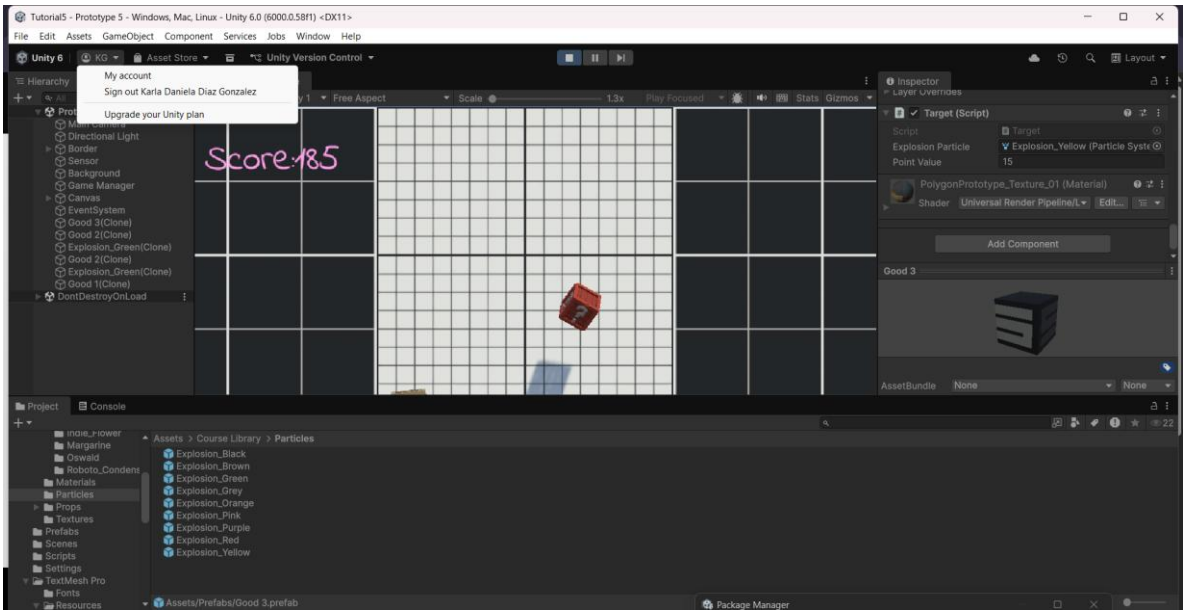
- En Target.cs, crea una nueva variable pública int pointValue
- En cada uno de los inspectores del prefabricado Target , establezca el Valor del Punto en su valor real, incluyendo el valor negativo del objetivo defectuoso.

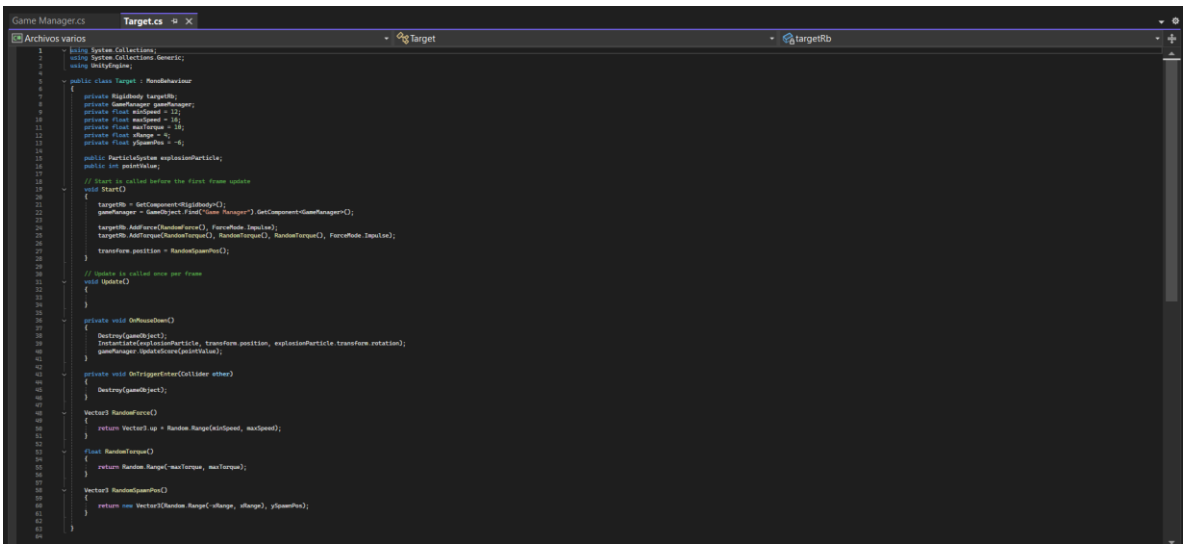
- Agrega la nueva variable a UpdateScore(pointValue);



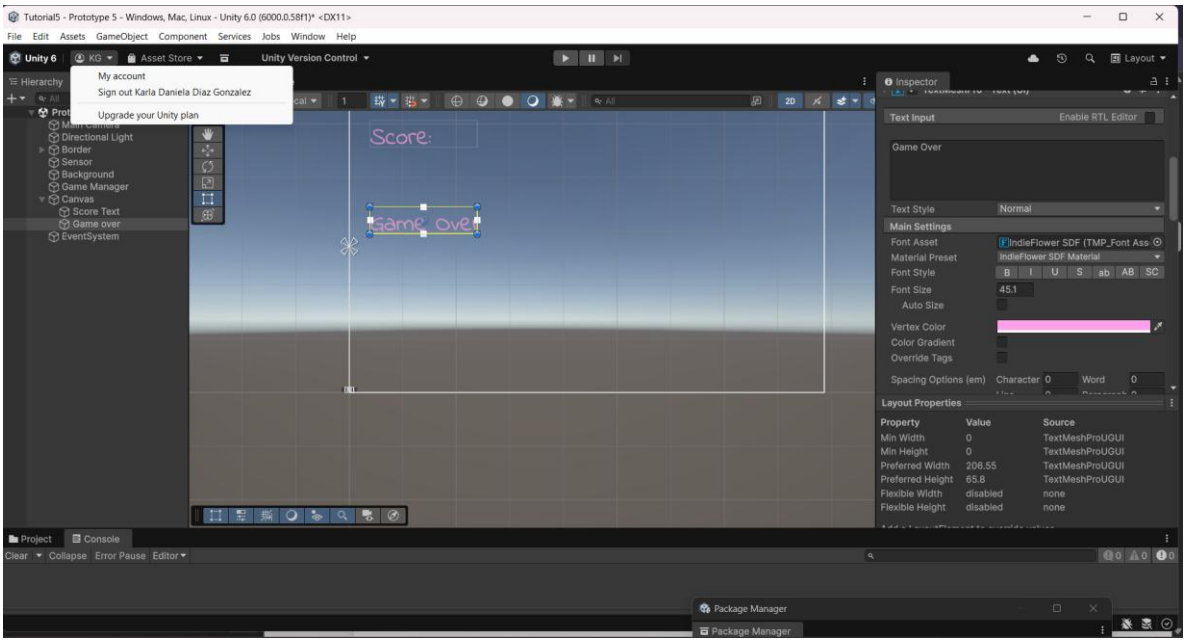


- En Target.cs, agrega una nueva variable pública llamada ParticleSystem explosionParticle
- Para cada uno de tus prefabricados objetivo, asigna un prefabricado de partículas de la Biblioteca de cursos > Partículas a la variable Partícula de explosión .
- En la función OnMouseDown() , crea una nueva instancia de un prefab de explosión.

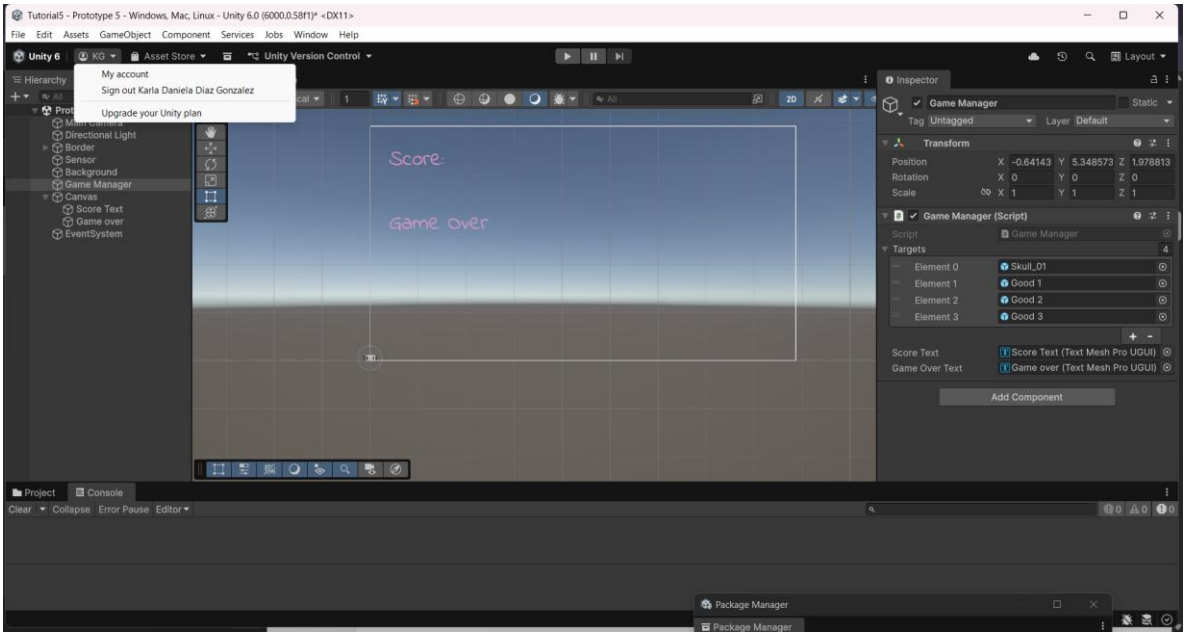


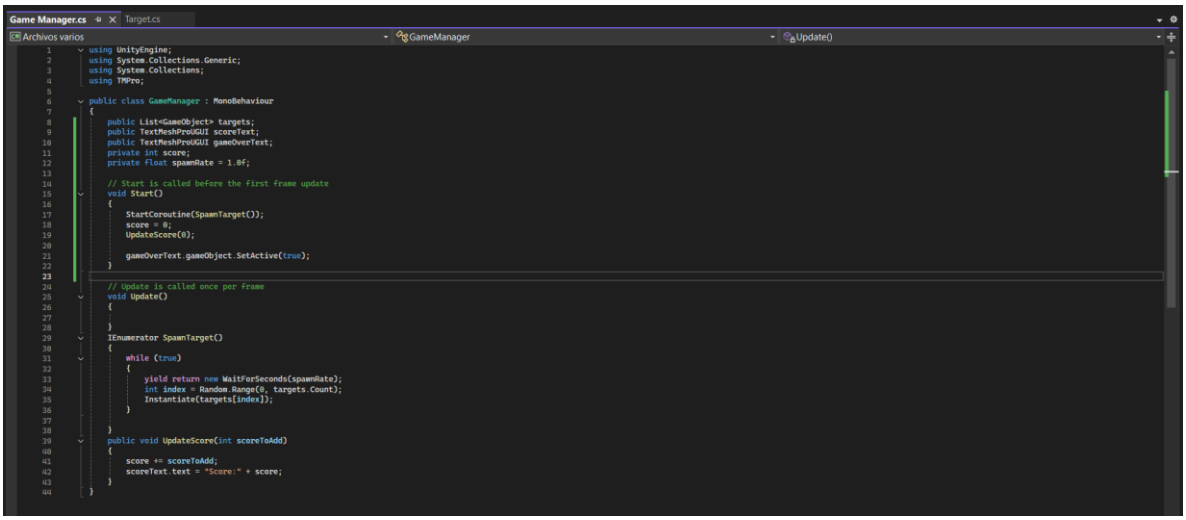


- Haz clic derecho en el lienzo , crea un nuevo objeto UI > Texto - TextMeshPro y cámbiale el nombre a “ Texto de fin de juego ”.
- En el inspector, edite su Texto , Pos X , Pos Y , Fuente , Tamaño , Estilo , Color y Alineación.
- Cambia la configuración de Ajuste de texto de Normal a Sin ajuste

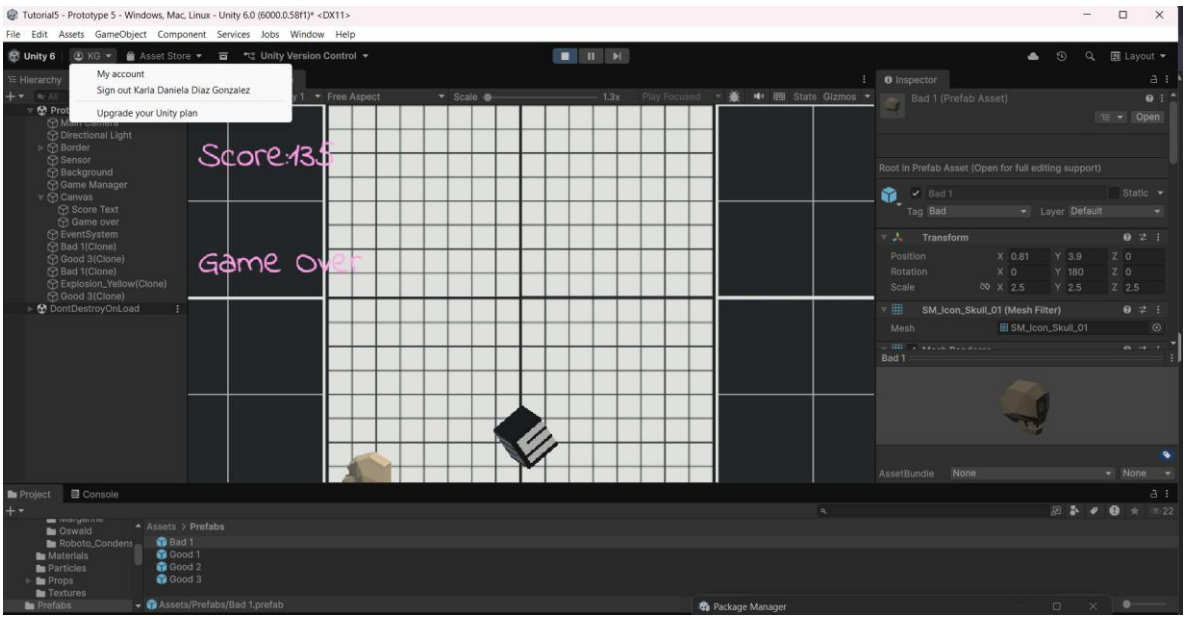


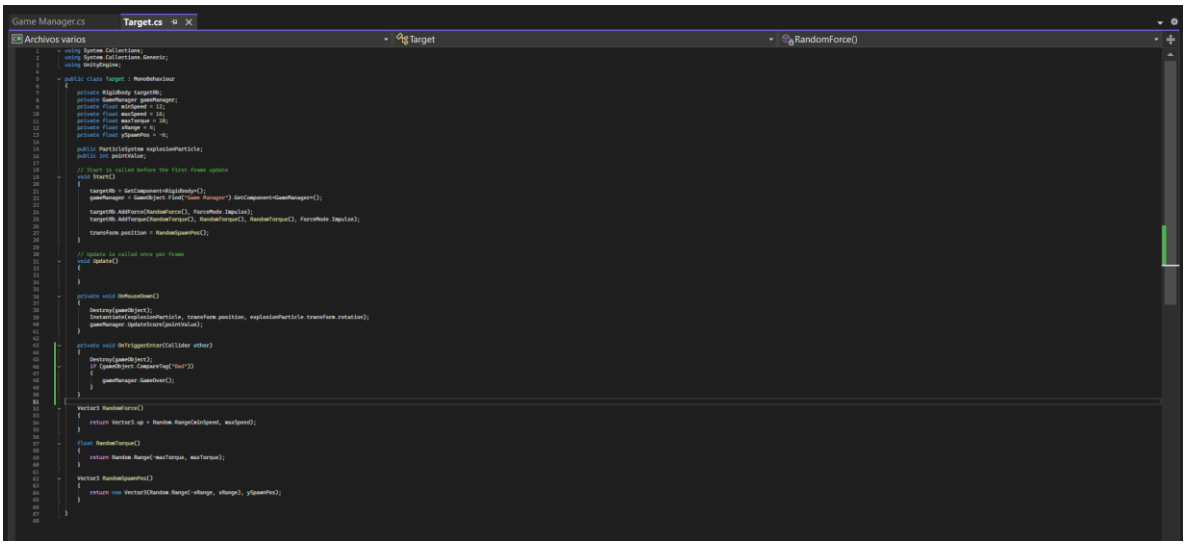
- En GameManager.cs, crea un nuevo objeto público TextMeshProUGUI gameOverText y asígnale el objeto Game Over en el inspector.
- Desmarca la casilla Activo para desactivar el texto "Fin del juego" por defecto.
- En Start() , activa el texto de Fin del juego

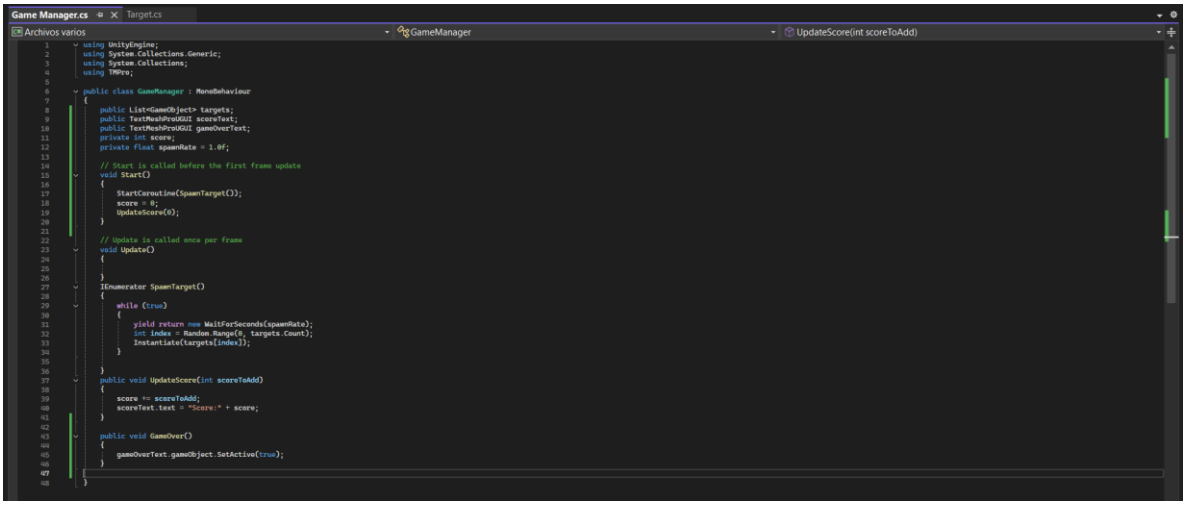




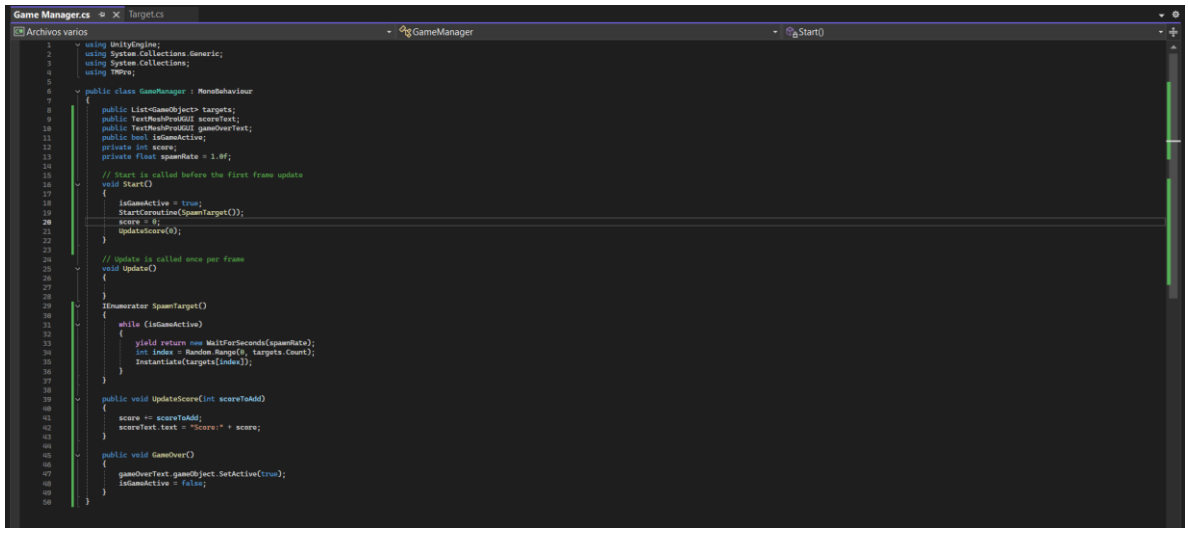
- Crea una nueva función pública `GameOver()` y mueve dentro de ella el código que activa el texto de fin de juego.
- En Target.cs, llama a gameManager.GameOver() si un objetivo colisiona con el sensor.
- Agrega una nueva etiqueta " Malo " al objeto Malo y una condición que solo active el fin del juego si no se trata de un objeto malo.



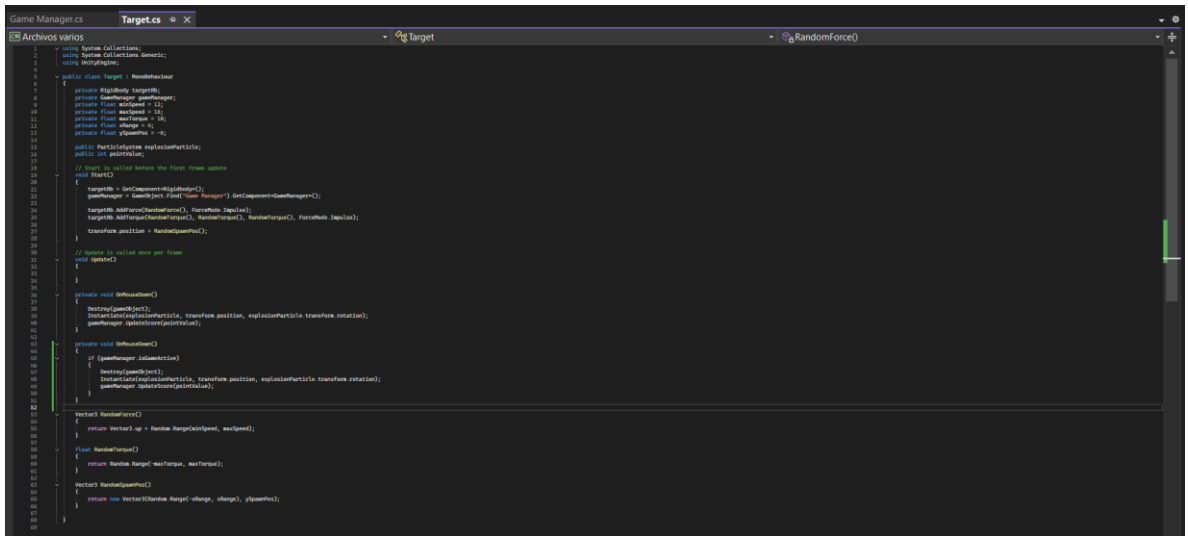




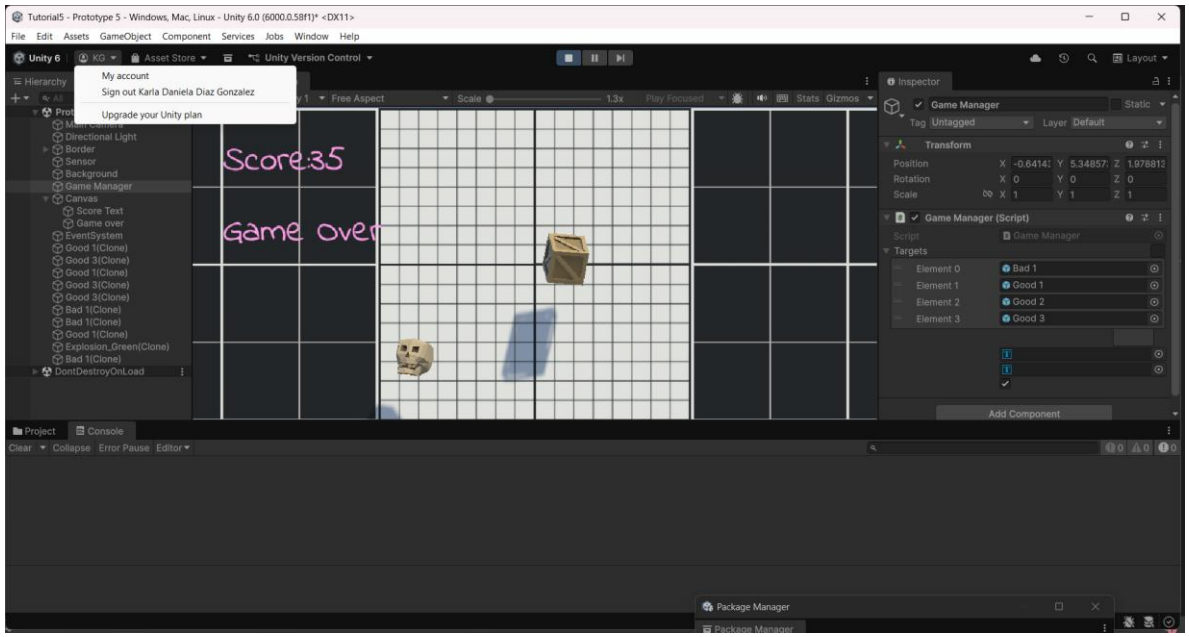
- Crea una nueva variable pública booleana isGameActive;
- En la primera línea de Start() , establezca isGameActive = true; y en GameOver() , establezca isGameActive = false;
- Para evitar la aparición de objetos, en la corrutina SpawnTarget() , cambie while (true) por while (isGameActive).
- Para evitar que se anoten puntos, en Target.cs, en la función OnMouseDown() , agregue la condición if (gameManager.isGameActive) {



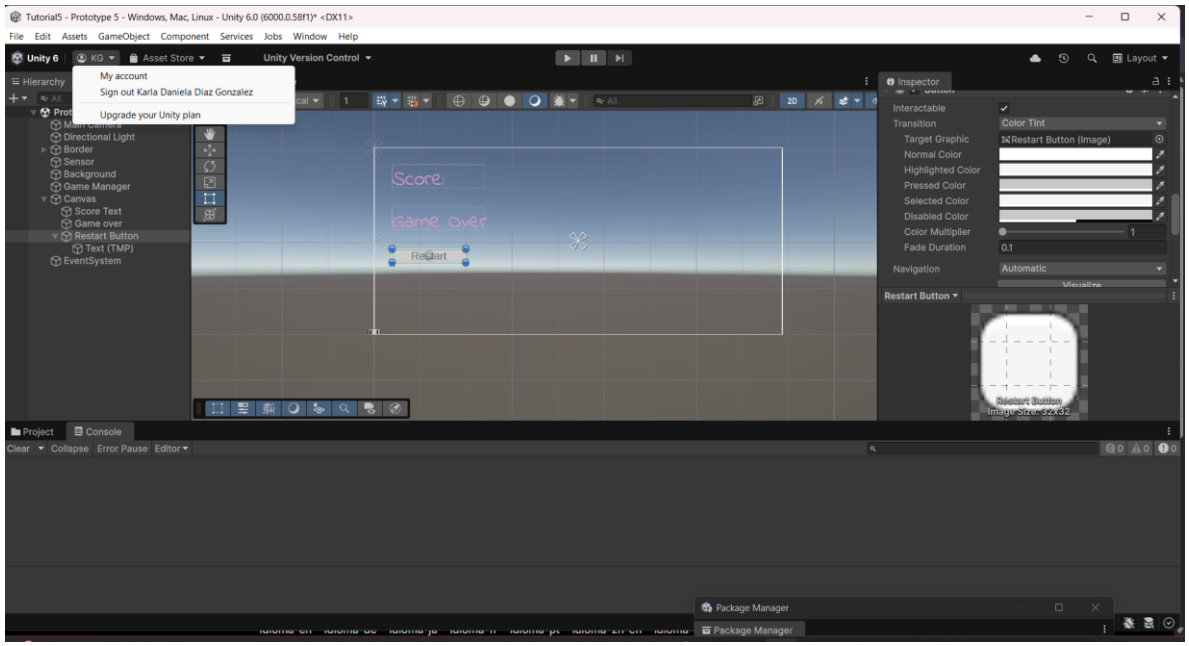
```
using UnityEngine;
using System.Collections.Generic;
using System.Collections;
using TMPro;

public class GameManager : MonoBehaviour
{
    public List<GameObject> targets;
    public TextMeshProUGUI scoreText;
    public TextMeshProUGUI gameOverText;
    public bool isGameActive;
    private int score;
    private float spawnRate = 1.0f;

    // Start is called before the first frame update
    void Start()
    {
        isGameActive = true;
        StartCoroutine(SpawnTarget());
        score = 0;
        UpdateScore(0);
    }

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

    }

    IEnumerator SpawnTarget()
    {
        while (isGameActive)
        {
            yield return new WaitForSeconds(spawnRate);
            int index = Random.Range(0, targets.Count);
            Instantiate(targets[index]);
        }
    }

    public void UpdateScore(int scoreToAdd)
    {
        score += scoreToAdd;
        scoreText.text = "Score:" + score;
    }

    public void GameOver()
    {
        gameOverText.gameObject.SetActive(true);
        isGameActive = false;
    }
}
```



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

public class Target : MonoBehaviour
{
    private Rigidbody targetRb;
    private GameManager gameManager;
    private float minSpeed = 12;
    private float maxSpeed = 16;
    private float maxTorque = 10;
    private float xRange = 4;
    private float ySpawnPos = -6;

    public ParticleSystem explosionParticle;
    public int pointValue;

    // Start is called before the first frame update
    void Start()
    {
        targetRb = GetComponent<Rigidbody>();
        gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();

        targetRb.AddForce(RandomForce(), ForceMode.Impulse);
        targetRb.AddTorque(RandomTorque(), RandomTorque(), RandomTorque(), ForceMode.Impulse);

        transform.position = RandomSpawnPos();
    }

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

    }

    private void OnMouseDown()
    {
        Destroy(gameObject);
        Instantiate(explosionParticle, transform.position, explosionParticle.transform.rotation);
        gameManager.UpdateScore(pointValue);
    }

    private void OnMouseDown()
    {
        if (gameManager.isGameActive)
        {
            Destroy(gameObject);
            Instantiate(explosionParticle, transform.position, explosionParticle.transform.rotation);
            gameManager.UpdateScore(pointValue)
        }
    }

    Vector3 RandomForce()
    {
        return Vector3.up * Random.Range(minSpeed, maxSpeed);
    }

    float RandomTorque()
    {
        return Random.Range(-maxTorque, maxTorque);
    }

    Vector3 RandomSpawnPos()
    {
        return new Vector3(Random.Range(-xRange, xRange), ySpawnPos);
    }
}
```

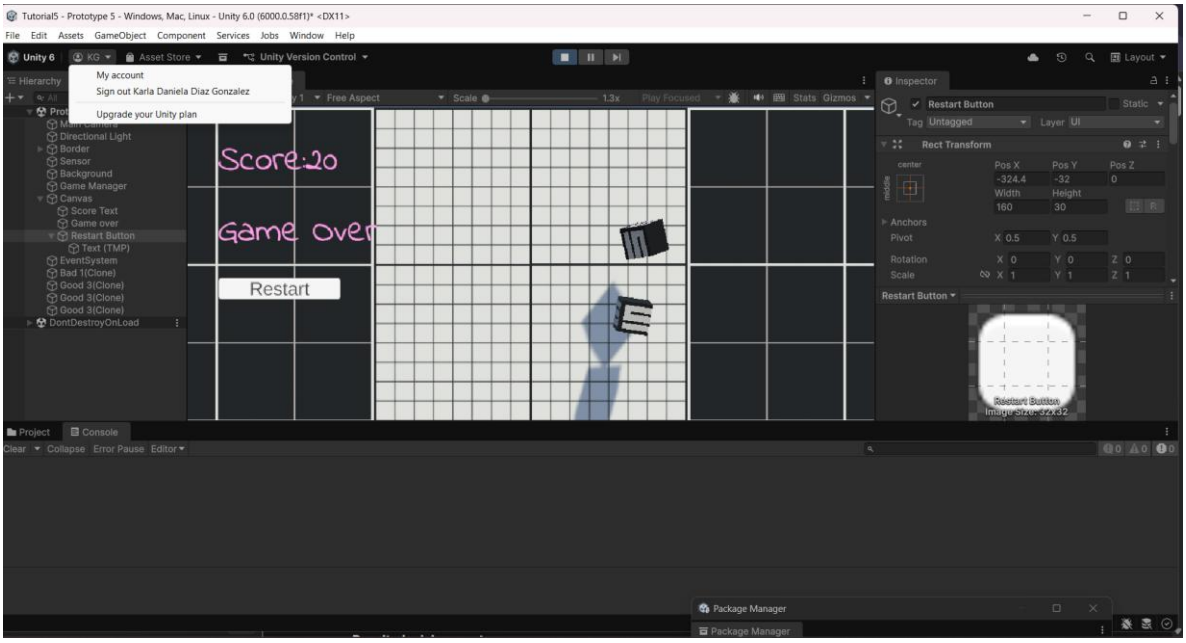


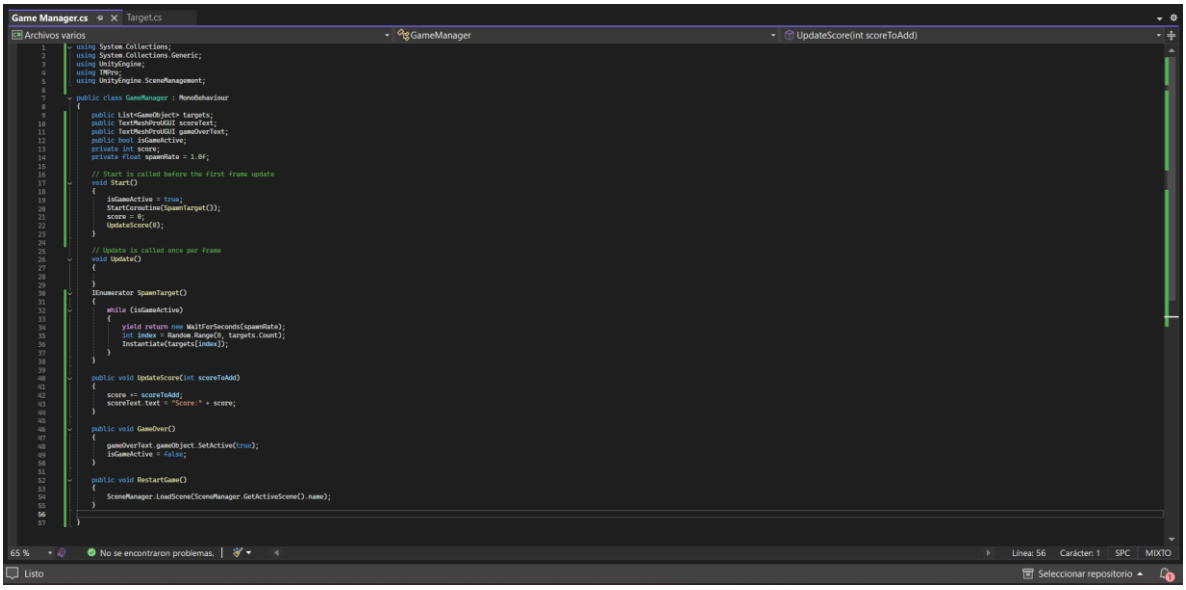
- Haz clic con el botón derecho en el lienzo y selecciona Crear > Interfaz de usuario > Botón - TextMeshPro .
- Cambiar el nombre del botón a “ Botón de reinicio ”
- Reactiva temporalmente el texto "Game Over" para reposicionar el botón de reinicio junto al texto y luego desactívalo de nuevo .
- Seleccione el objeto secundario Texto y, a continuación, edite su texto para que diga « Reiniciar » , su fuente, estilo y tamaño.



- En GameManager.cs, agregue using UnityEngine.SceneManagement;

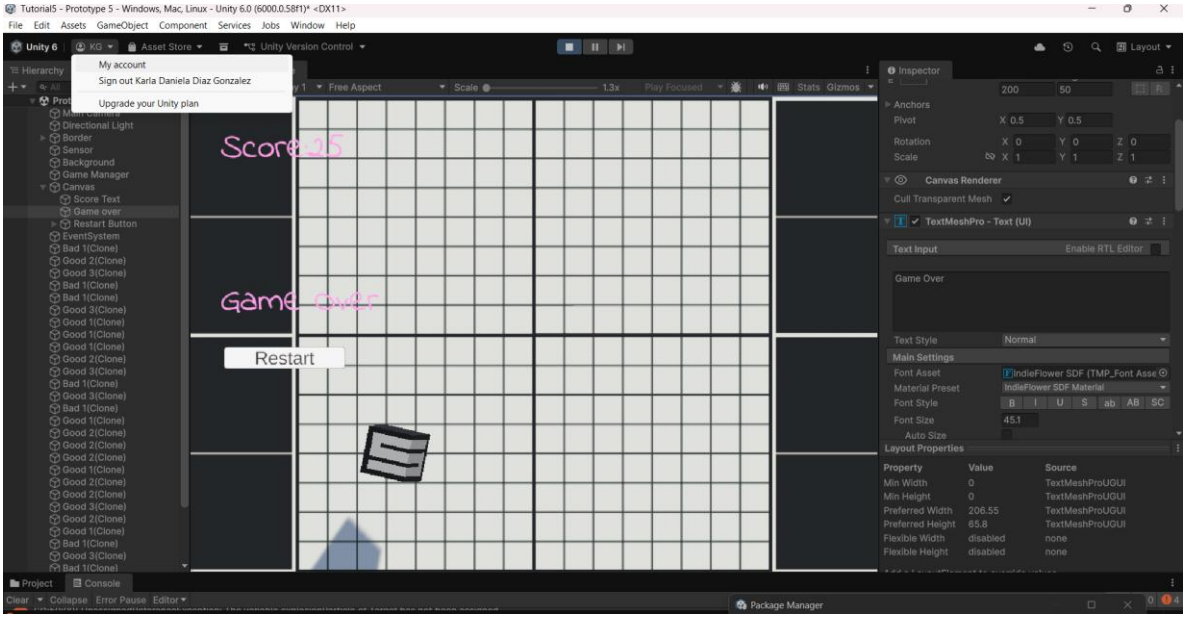
- Crea una nueva función pública void RestartGame() que recargue la escena actual.
- En el inspector del botón, haga clic en + para agregar un nuevo evento Al hacer clic, arrástrelo al objeto Administrador de juegos y seleccione la función GameManager.RestartGame.

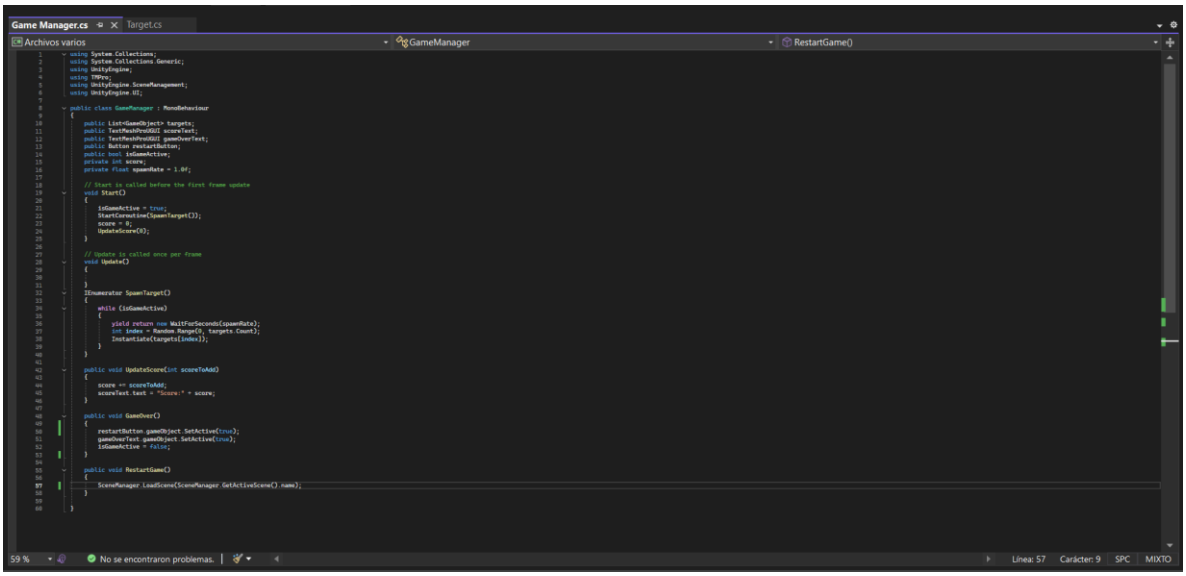




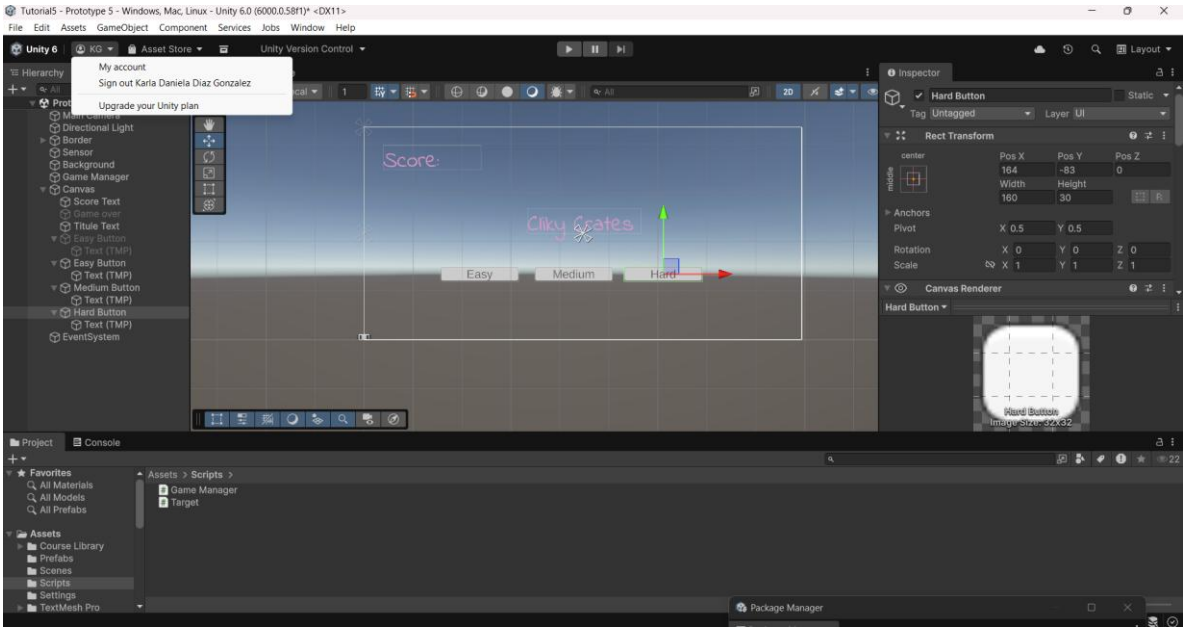
- En la parte superior de GameManager.cs agregue using UnityEngine.UI;
- Declara un nuevo botón público llamado restartButton; y asígnale el botón Reiniciar en el inspector.
- Desmarque la casilla de verificación “Activo” del botón Reiniciar en el inspector

- En la función GameOver , activa el botón Reiniciar.

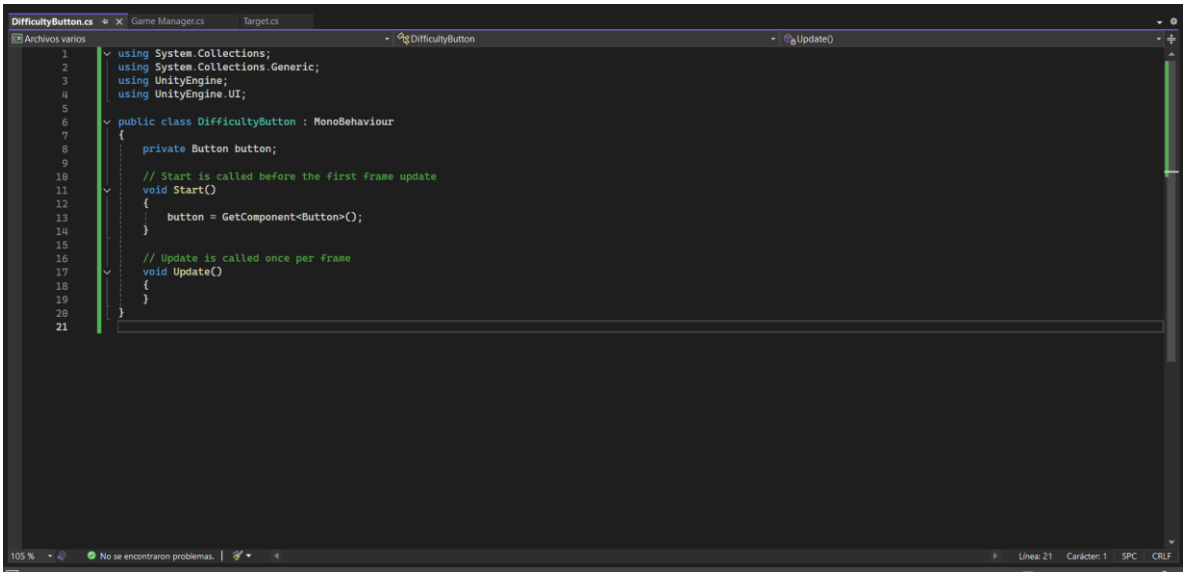




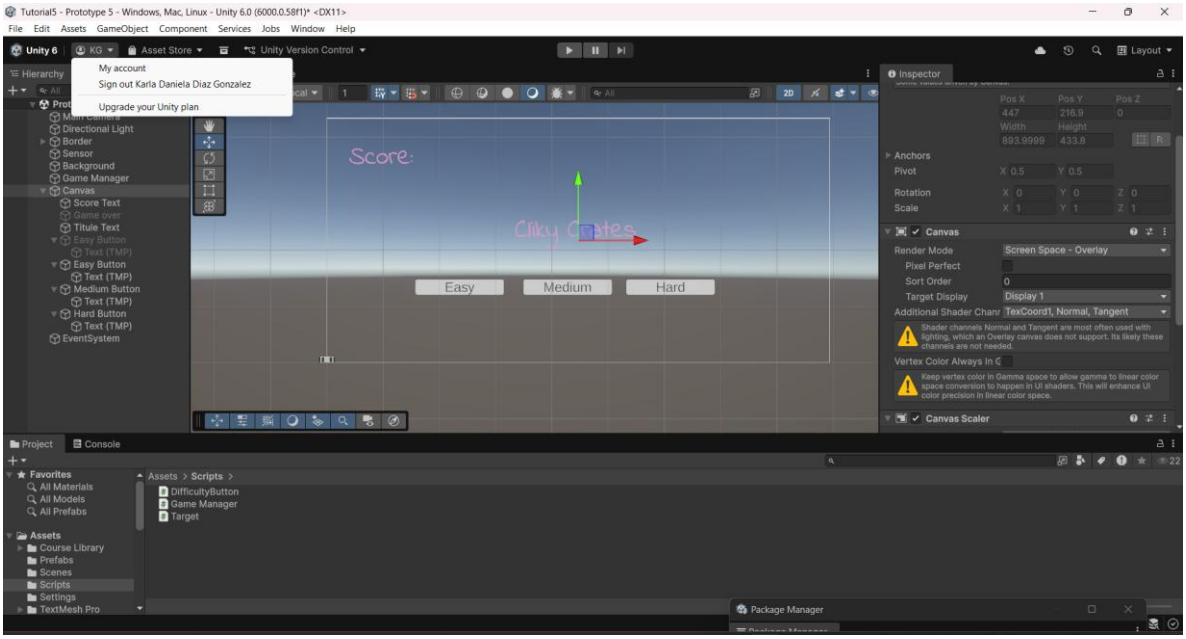
- Duplica tu texto de "Fin del juego" para crear tu texto de título, editando su nombre, texto y todos sus atributos.
- Duplica tu botón de reinicio y edita sus atributos para crear un botón " fácil ".
- Edita y duplica el nuevo botón Fácil para crear un “ Botón Medio ” y un “ Botón Difícil ”.



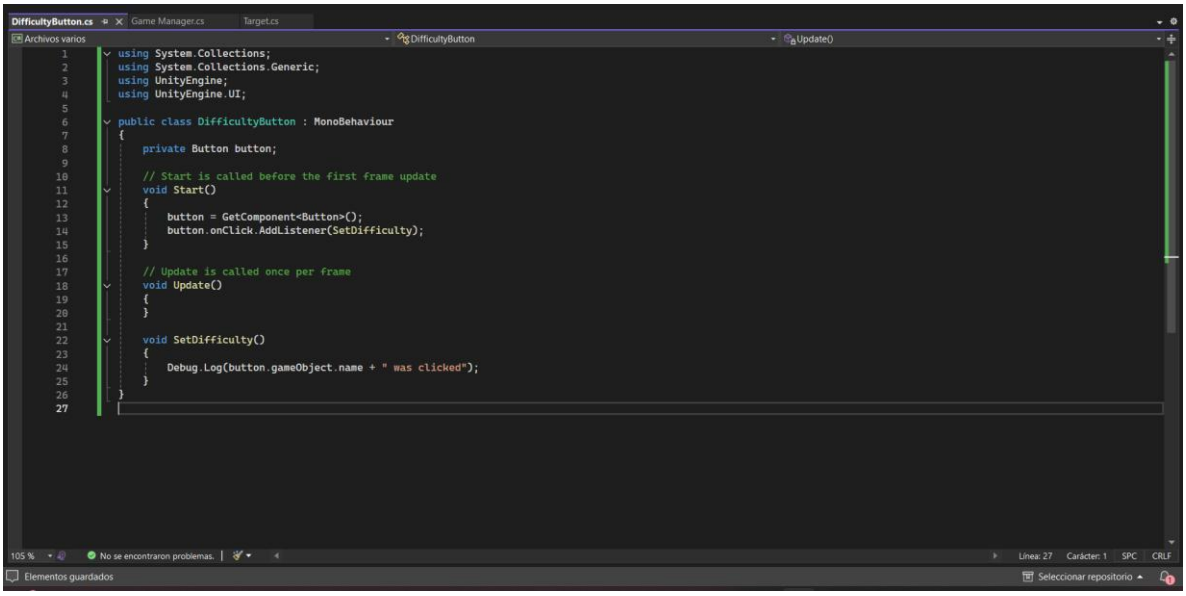
- Para los 3 botones nuevos, en el componente Botón, en la sección Al hacer clic () , haga clic en el botón menos (-) para eliminar la funcionalidad Reiniciar juego.
- Crea un nuevo script DifficultyButton.cs y adjúntalo a los 3 botones.
- Agrega using UnityEngine.UI a tus importaciones
- Crea una nueva variable privada llamada Button y inicialízala en Start().

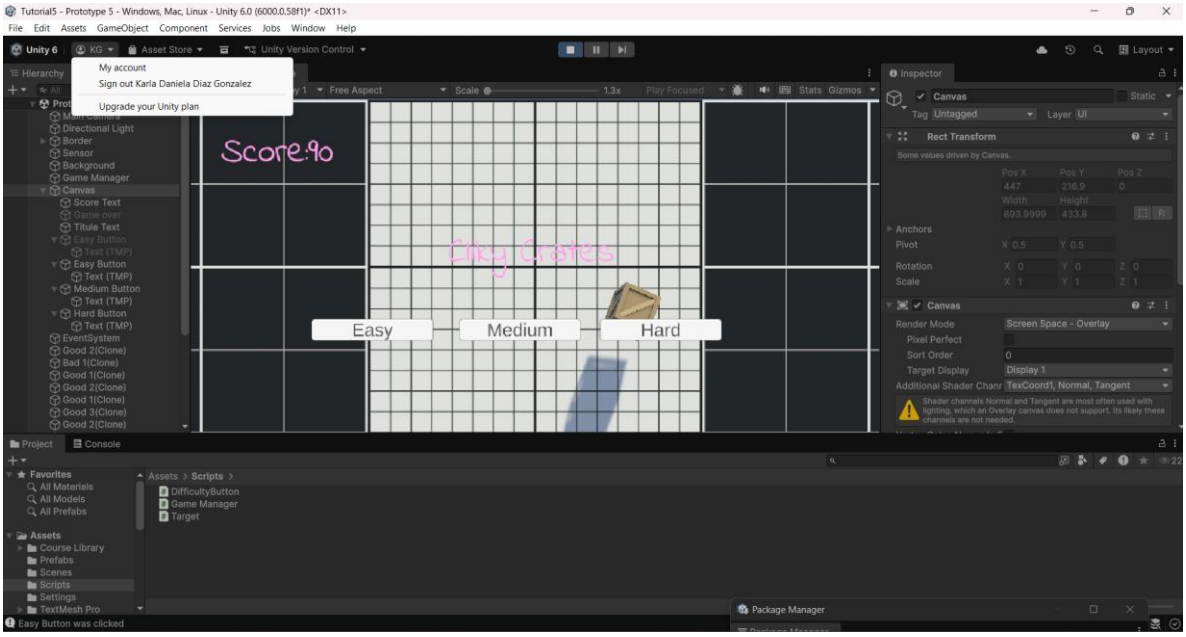


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DifficultyButton : MonoBehaviour
{
    private Button button;

    // Start is called before the first frame update
    void Start()
    {
        button = GetComponent<Button>();
    }

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

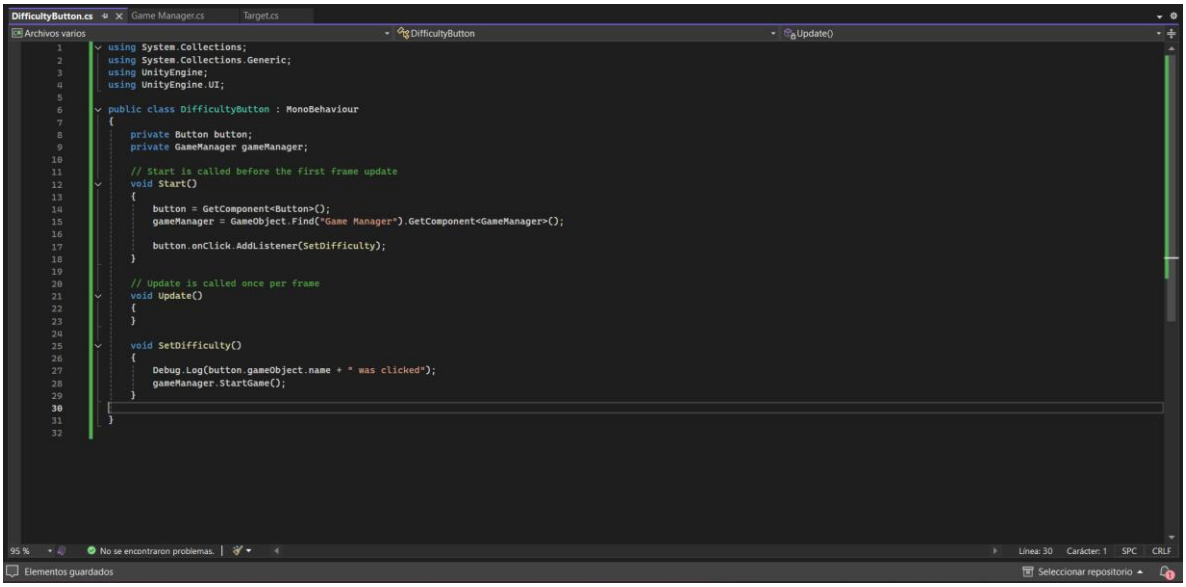


- Crea una nueva función void SetDifficulty y, dentro de ella, Debug.Log(gameObject.name + " se hizo clic");
- Agrega el detector de eventos del botón para llamar a la función SetDifficulty .

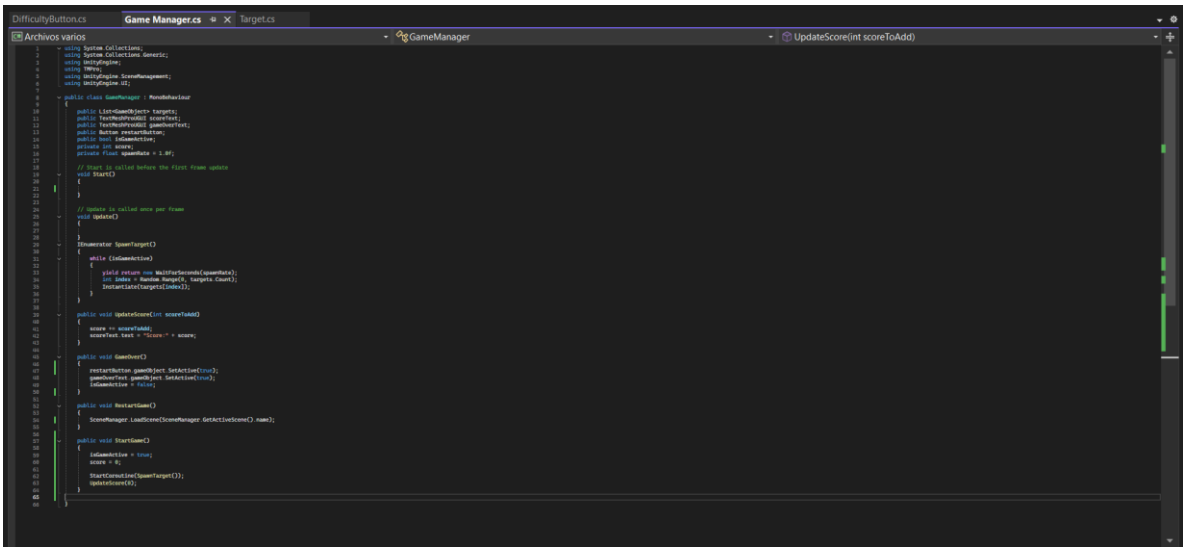


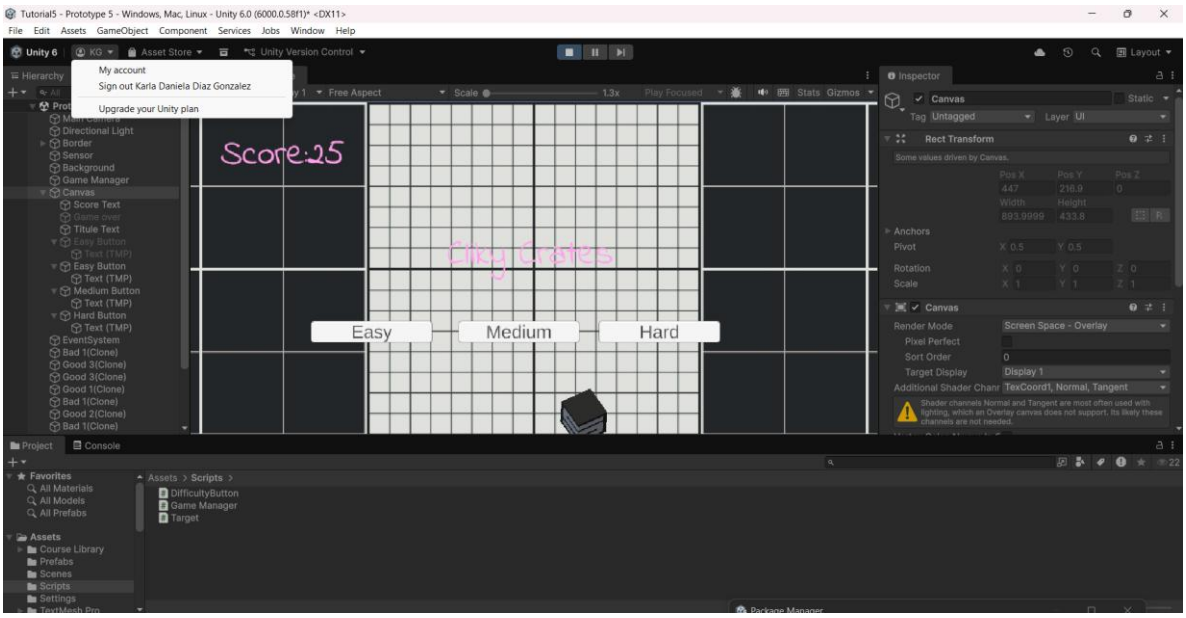


- En GameManager.cs, crea una nueva función pública void StartGame() y traslada todo el código de Start() a ella .
- En DifficultyButton.cs, crea un nuevo GameManager privado llamado gameManager; e inicialízalo en Start().
- En la función SetDifficulty() , llame a gameManager.StartGame();

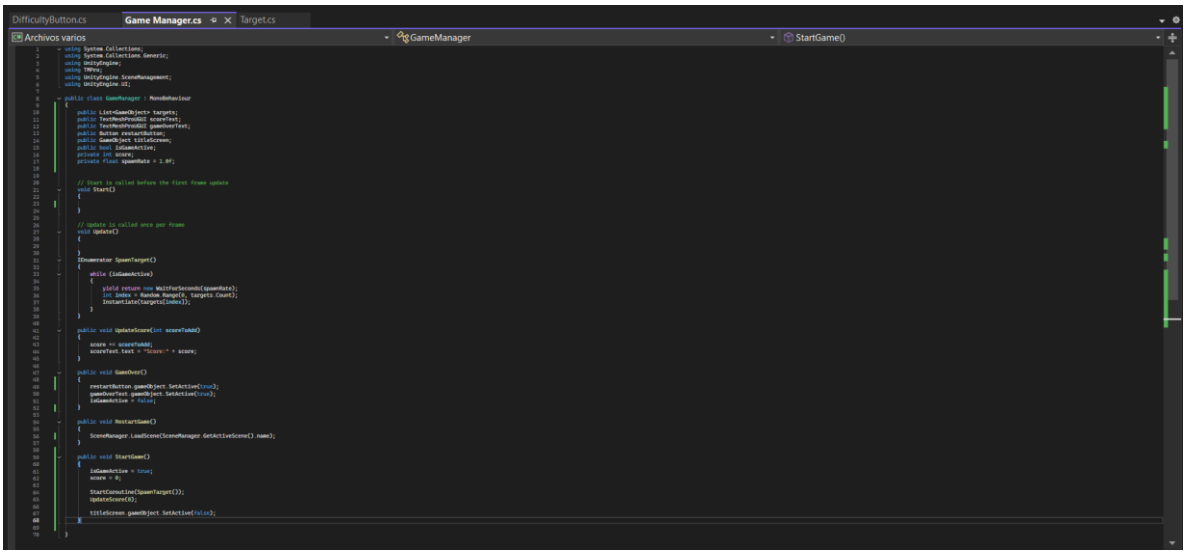


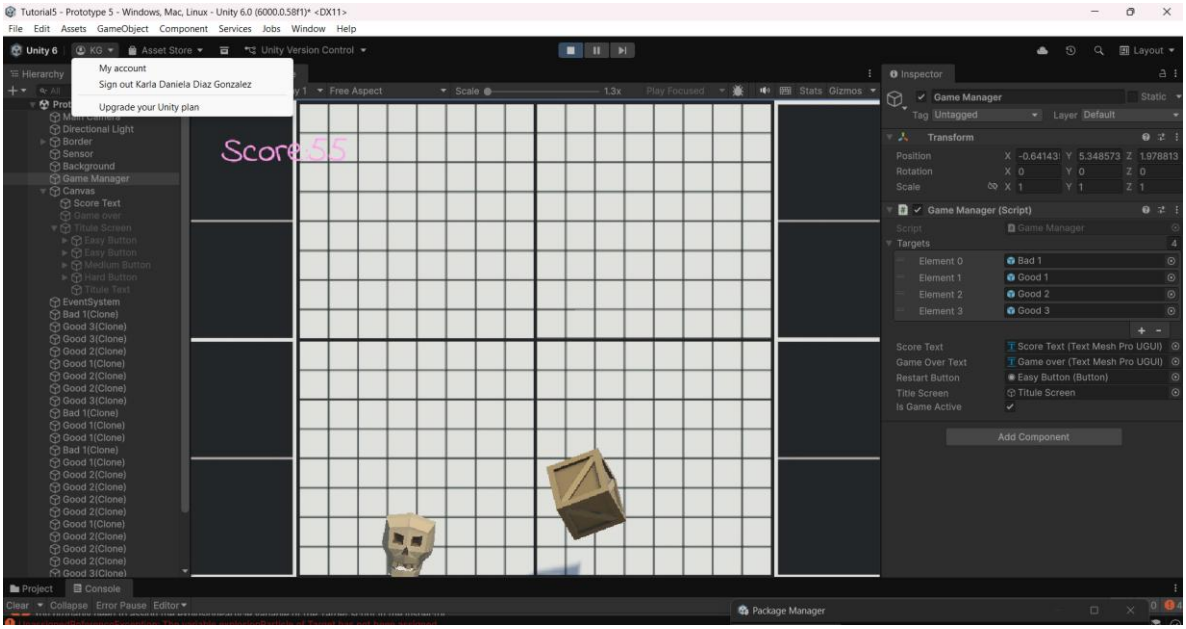
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DifficultyButton : MonoBehaviour
{
    private Button button;
    private GameManager gameManager;

    // Start is called before the first frame update
    void Start()
    {
        button = GetComponent<Button>();
        gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();

        button.onClick.AddListener(SetDifficulty);
    }

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

    void SetDifficulty()
    {
        Debug.Log(button.gameObject.name + " was clicked");
        gameManager.StartGame();
    }
}
```



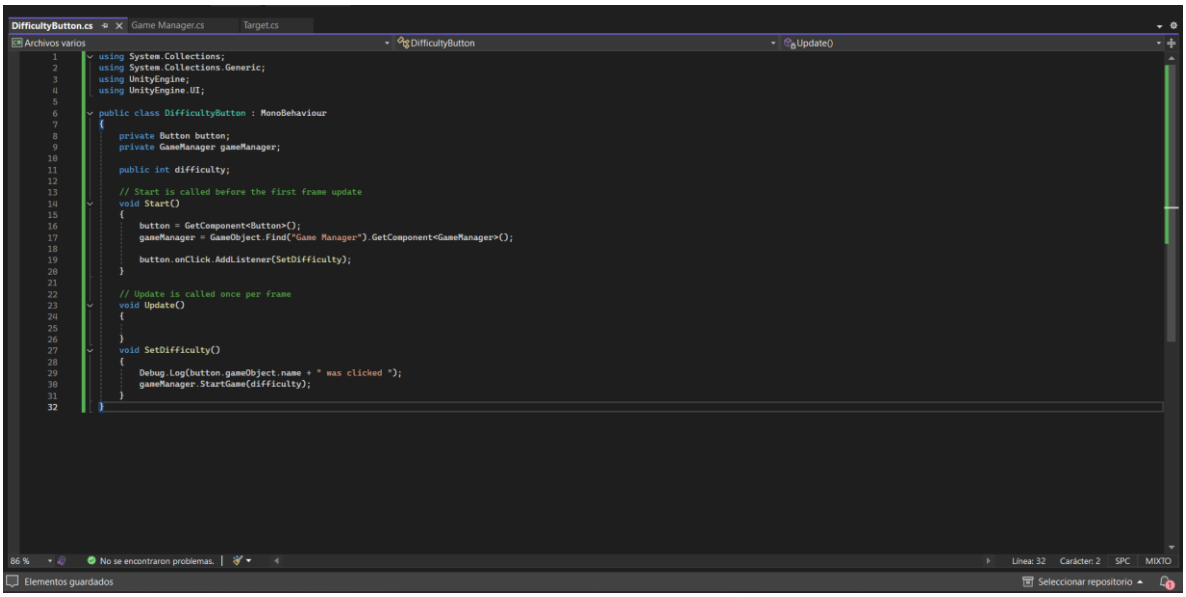


- Haz clic derecho en el lienzo y selecciona Crear > Objeto vacío , cámbiale el nombre a « Pantalla de título » y arrastra los 3 botones y el título a la pantalla.
- En GameManager.cs, crea un nuevo GameObject público llamado titleScreen y asígnalo en el inspector.
- En StartGame() , desactiva el objeto de la pantalla de título





- En DifficultyButton.cs, crea una nueva variable pública de tipo entero llamada difficulty y, a continuación, en el Inspector, asigna el valor 1 a la dificultad Fácil , 2 a la Media y 3 a la Difícil .
- Agrega un parámetro de dificultad int a la función StartGame().
- En StartGame() , establezca spawnRate /= difficulty;
- Corrija el error en DifficultyButton.cs pasando el parámetro de dificultad a StartGame(difficulty).


DifficultyButton.cs
Game Manager.cs
Target.cs
Archivos varios
DifficultyButton
Update()
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class DifficultyButton : MonoBehaviour
{
private Button button;
private GameManager gameManager;
public int difficulty;
// Start is called before the first frame update
void Start()
{
button = GetComponent<Button>();
gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();
button.onClick.AddListener(SetDifficulty);
}
// Update is called once per frame
void Update()
{
}
void SetDifficulty()
{
Debug.Log(button.gameObject.name + " was clicked ");
gameManager.StartGame(difficulty);
}
}
86 %
No se encontraron problemas.
Línea: 32
Carácter: 2
SPC
MIXTO
Elementos guardados
Seleccionar repositorio