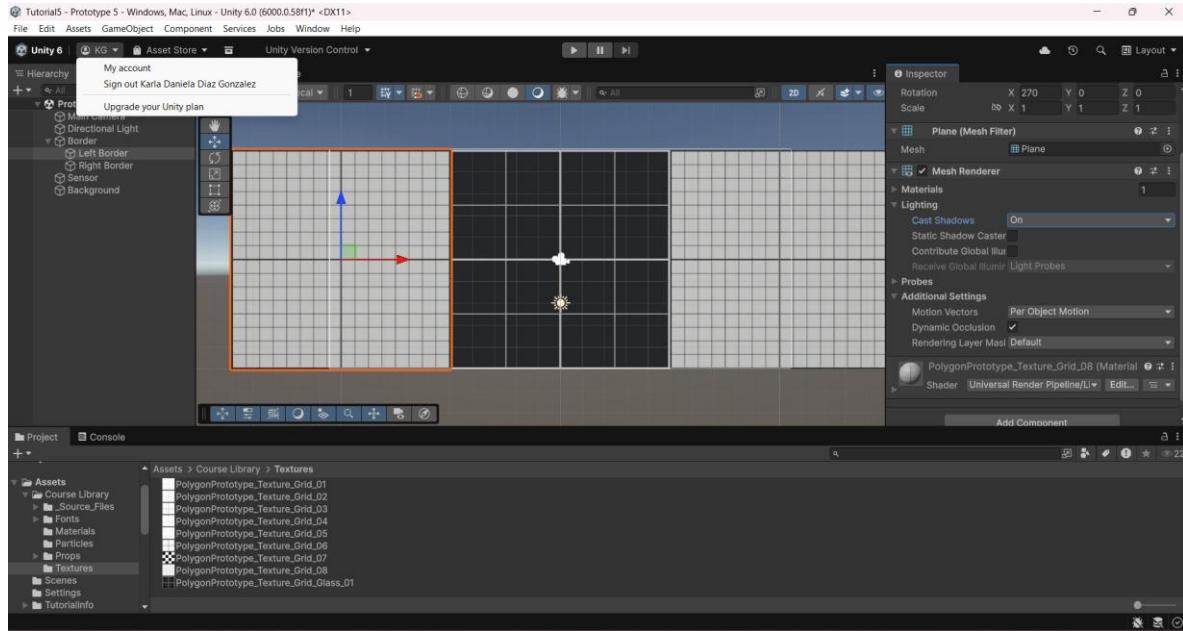


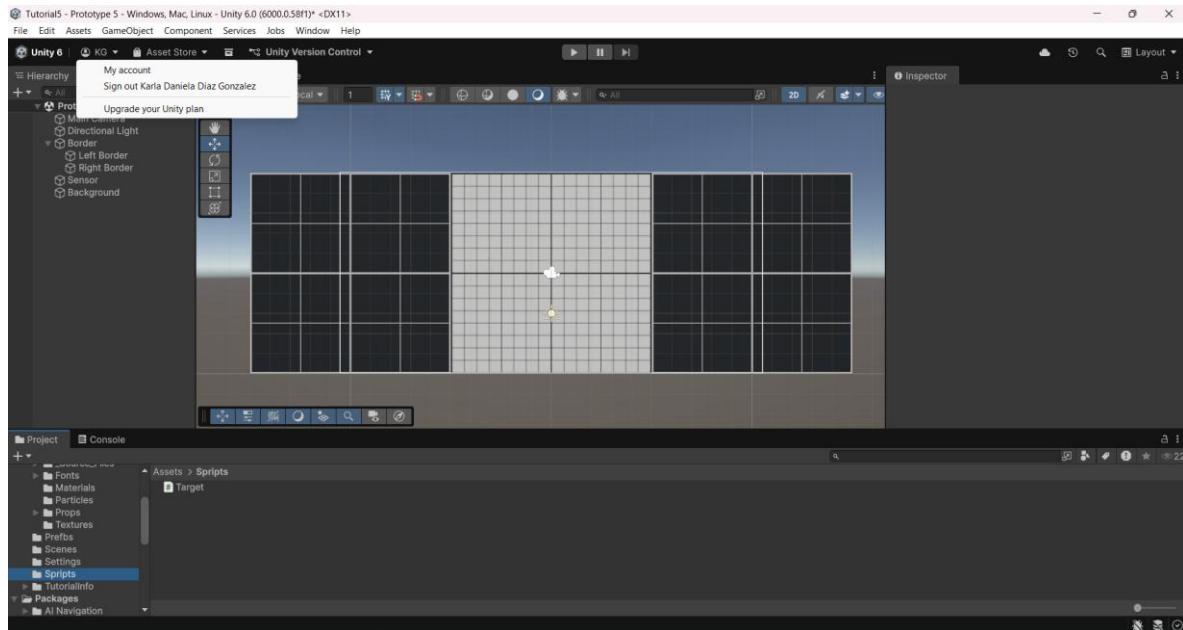
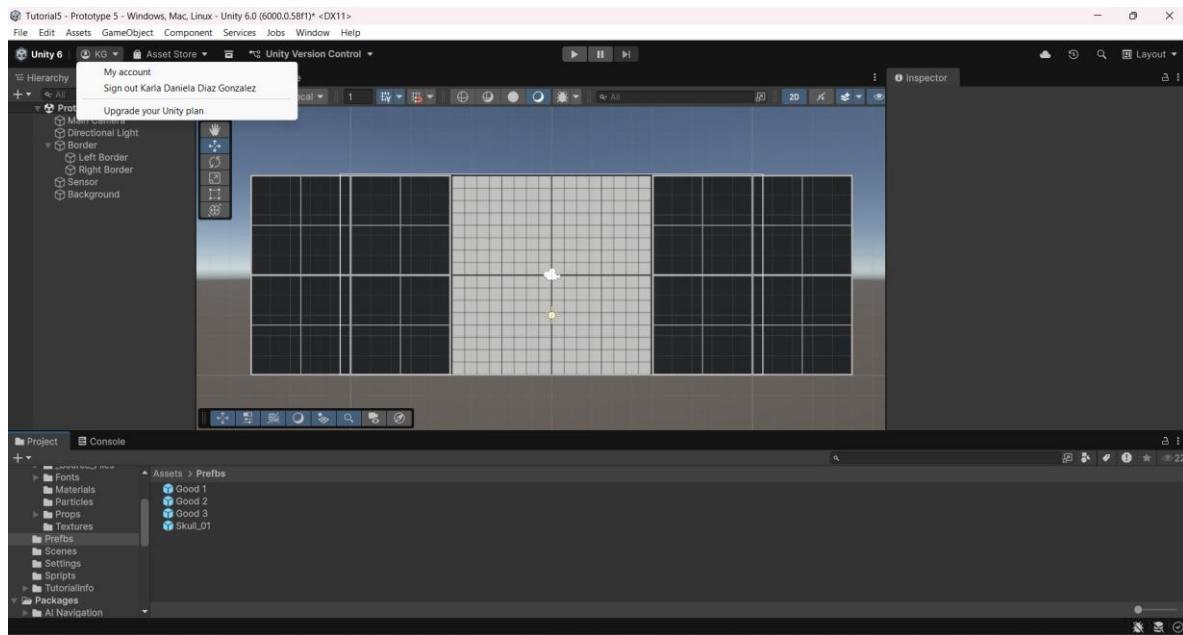
- Abre Unity Hub y crea un proyecto vacío llamado « Prototype 5 » en el directorio de tu curso, utilizando la versión correcta de Unity. Si no recuerdas cómo hacerlo, consulta las instrucciones de la Lección 1.1 - Paso 1.
- Haz clic para descargar los [archivos iniciales del prototipo 5](#), extrae la 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 ícono 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.



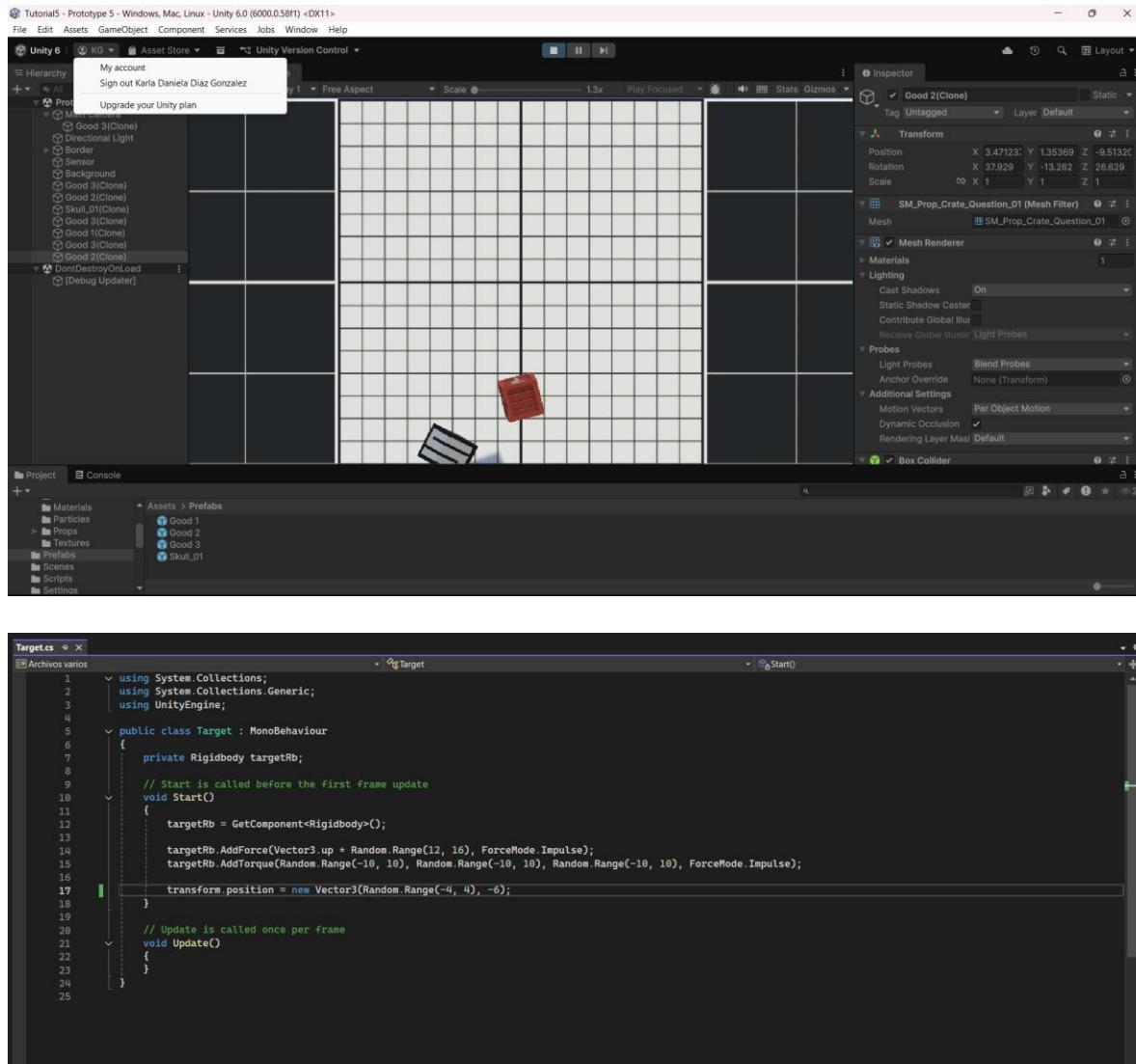
- Desde la Biblioteca , arrastra 3 objetos “buenos” y 1 objeto “malo” a la Escena, y cámbiales el nombre a “ Bueno 1 ”, “ Bueno 2 ”, “ Bueno 3 ” y “ Malo 1 ”.
- Agregar componentes de cuerpo rígido y colisionador de caja
- Crea una nueva carpeta Scripts, un nuevo script " Target.cs " dentro de ella y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

The screenshot shows the Unity Editor's Text Editor window with the script 'Target.cs' open. The code defines a MonoBehaviour named 'Target' with methods for Start and Update, and helper functions for RandomForce, RandomTorque, and RandomSpawnPos.

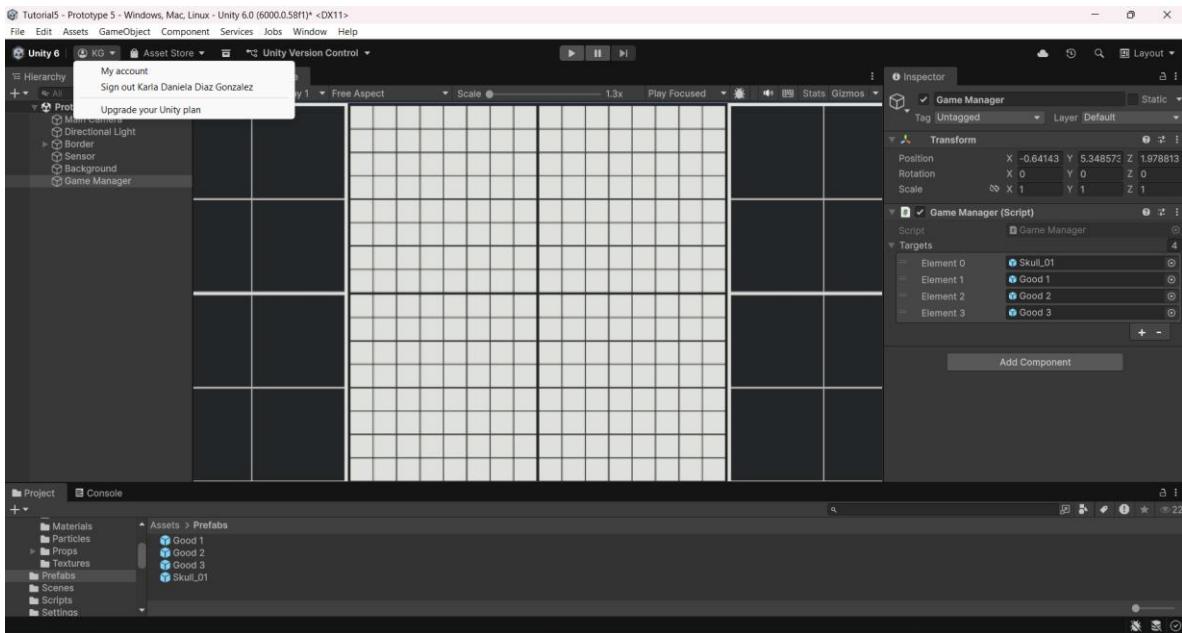
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Target : MonoBehaviour
6  {
7      private Rigidbody targetRb;
8      private float minSpeed = 12;
9      private float maxSpeed = 16;
10     private float maxTorque = 10;
11     private float xRange = 4;
12     private float ySpawnPos = -6;
13
14
15     // Start is called before the first frame update
16     void Start()
17     {
18         targetRb = GetComponent<Rigidbody>();
19
20         targetRb.AddForce(RandomForce(), ForceMode.Impulse);
21         targetRb.AddTorque(RandomTorque(), RandomTorque(), RandomTorque(), ForceMode.Impulse);
22
23         transform.position = RandomSpawnPos();
24     }
25
26
27     // Update is called once per frame
28     void Update()
29     {
30     }
31
32     Vector3 RandomForce()
33     {
34         return Vector3.up * Random.Range(minSpeed, maxSpeed);
35     }
36
37     float RandomTorque()
38     {
39         return Random.Range(-maxTorque, maxTorque);
40     }
41
42     Vector3 RandomSpawnPos()
43     {
44         return new Vector3(Random.Range(-xRange, xRange), ySpawnPos);
45     }
46
47 }
48
```

- 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 .

```

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 }

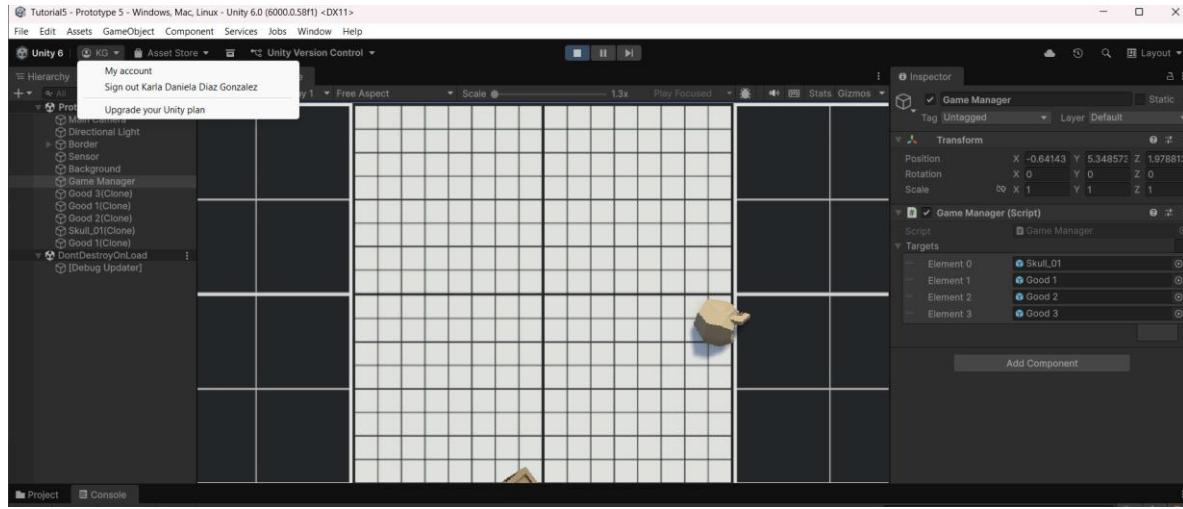
```



- 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;
```



```
GameManager.cs Target.cs
Archivos varios
Game Manager

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManager : MonoBehaviour
6  {
7      public List<GameObject> targets;
8      private float spawnRate = 1.0f;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         StartCoroutine(SpawnTarget());
14     }
15
16     // Update is called once per frame
17     void Update()
18     {
19     }
20
21     IEnumerator SpawnTarget()
22     {
23         while (true)
24         {
25             yield return new WaitForSeconds(spawnRate);
26             int index = Random.Range(0, targets.Count);
27             Instantiate(targets[index]);
28         }
29     }
30
31 }
32
```

- 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);
    }

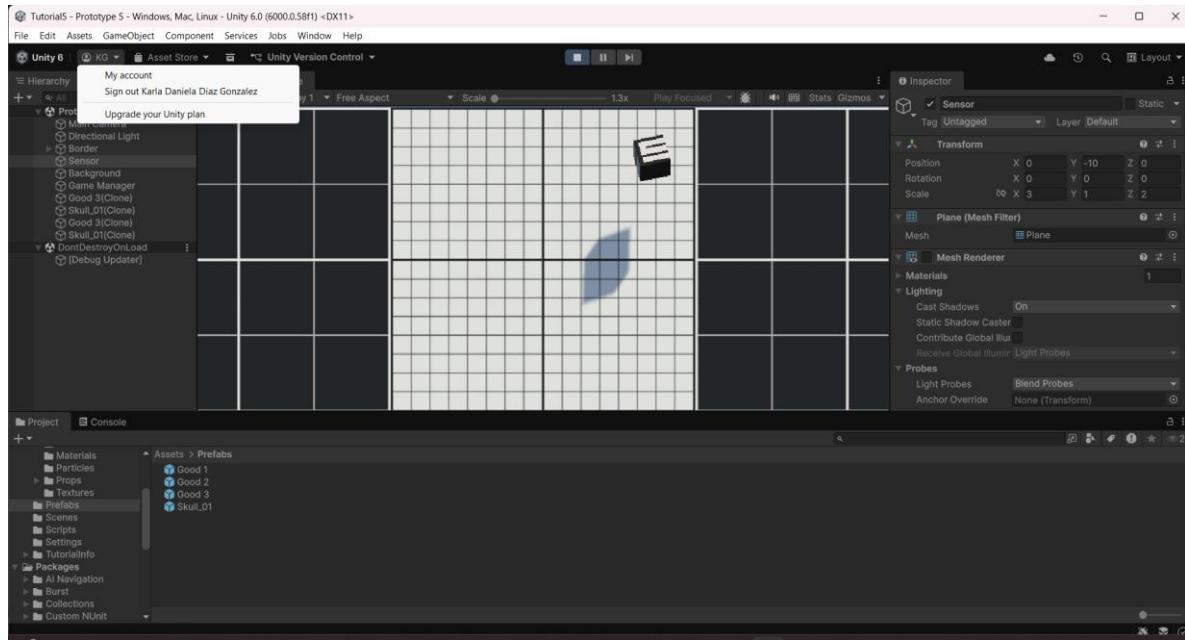
    private void OnTriggerEnter(Collider other)
    {
        Destroy(gameObject);
    }

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

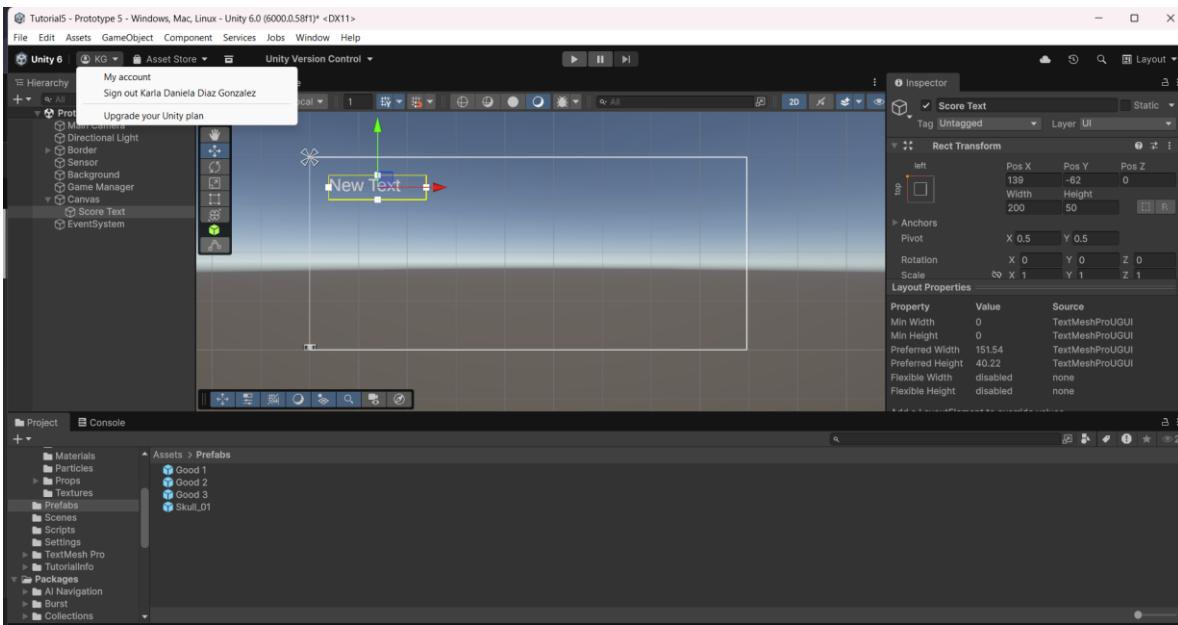
    float RandomTorque()
    {
        return Random.Range(-maxTorque, maxTorque);
    }

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

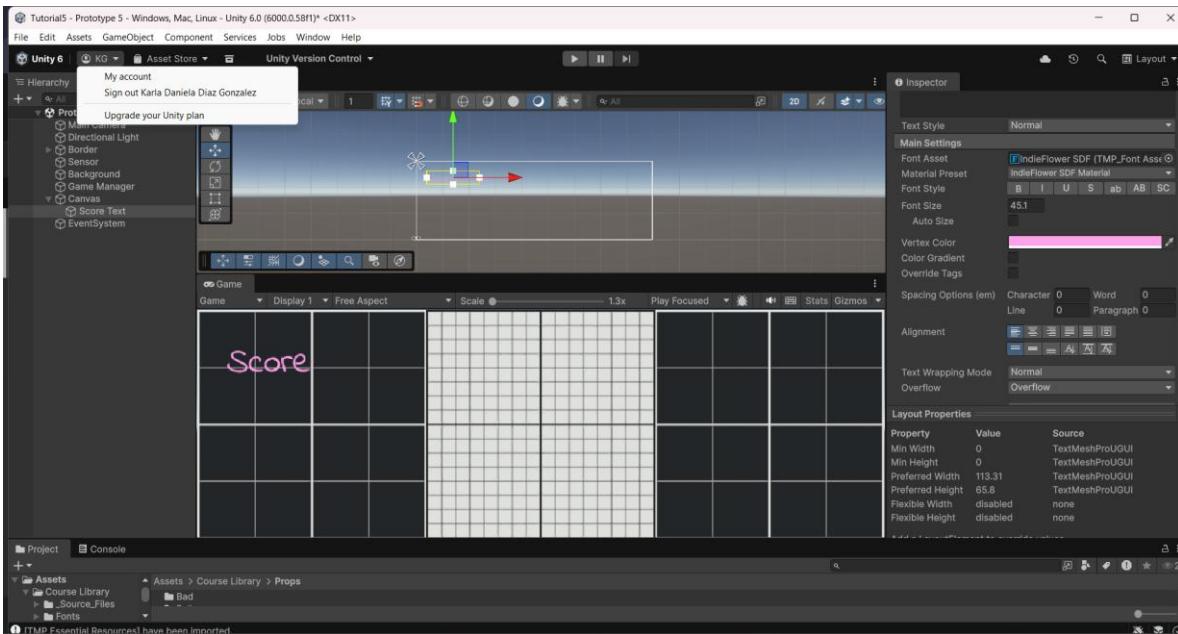
```



- 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.

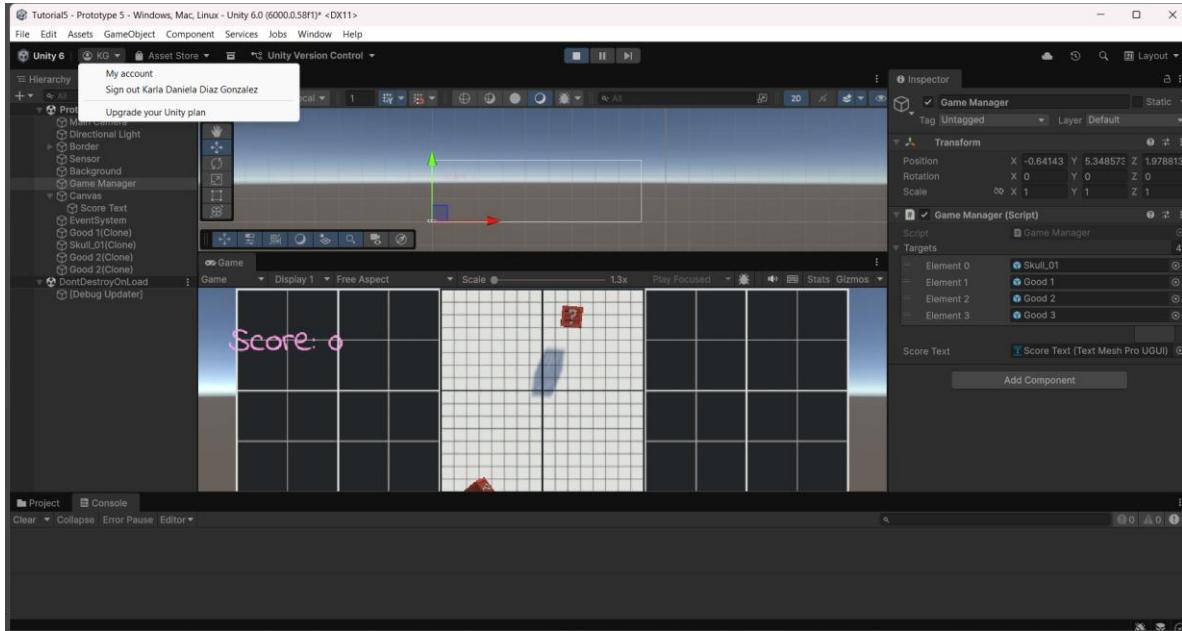


- Cambia su texto a “ Puntuación :”
- Elige una fuente , un estilo , un tamaño y un color de vértice que combinen bien con tu fondo.



- 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;

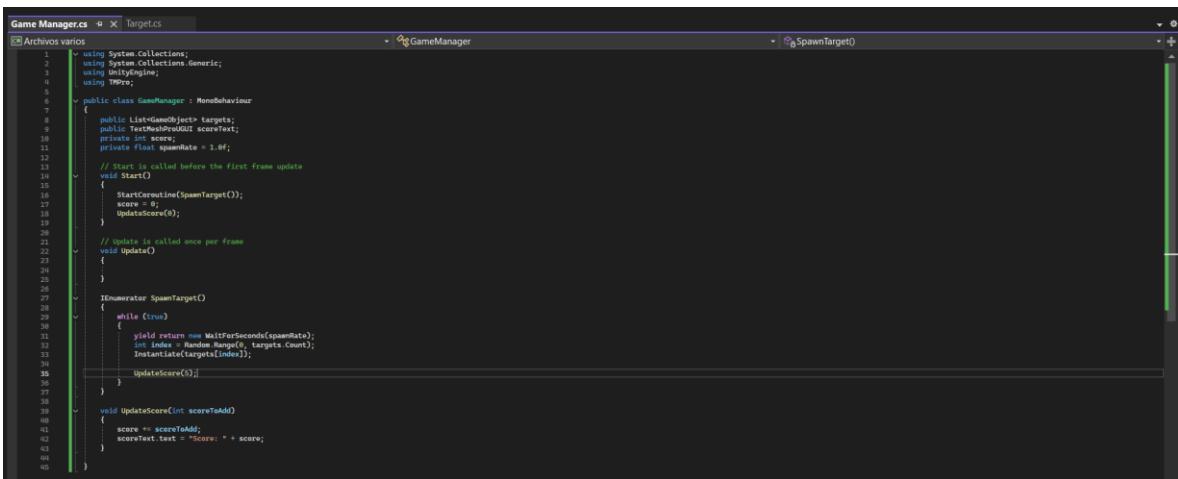


```

Game Manager.cs  Target.cs
Archivos varios
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5
6  public class GameManager : MonoBehaviour
7  {
8      public List<GameObject> targets;
9      public TextMeshProUGUI scoreText;
10     private int score;
11     private float spawnRate = 1.0f;
12
13     // Start is called before the first frame update
14     void Start()
15     {
16         StartCoroutine(SpawnTarget());
17         score = 0;
18         scoreText.text = "Score: " + score;
19     }
20
21     // Update is called once per frame
22     void Update()
23     {
24     }
25
26     IEnumerator SpawnTarget()
27     {
28         while (true)
29         {
30             yield return new WaitForSeconds(spawnRate);
31             int index = Random.Range(0, targets.Count);
32             Instantiate(targets[index]);
33         }
34     }
35
36
37

```

- 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()



```

using System.Collections;
using System.Linq;
using UnityEngine;
using TMPro;

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

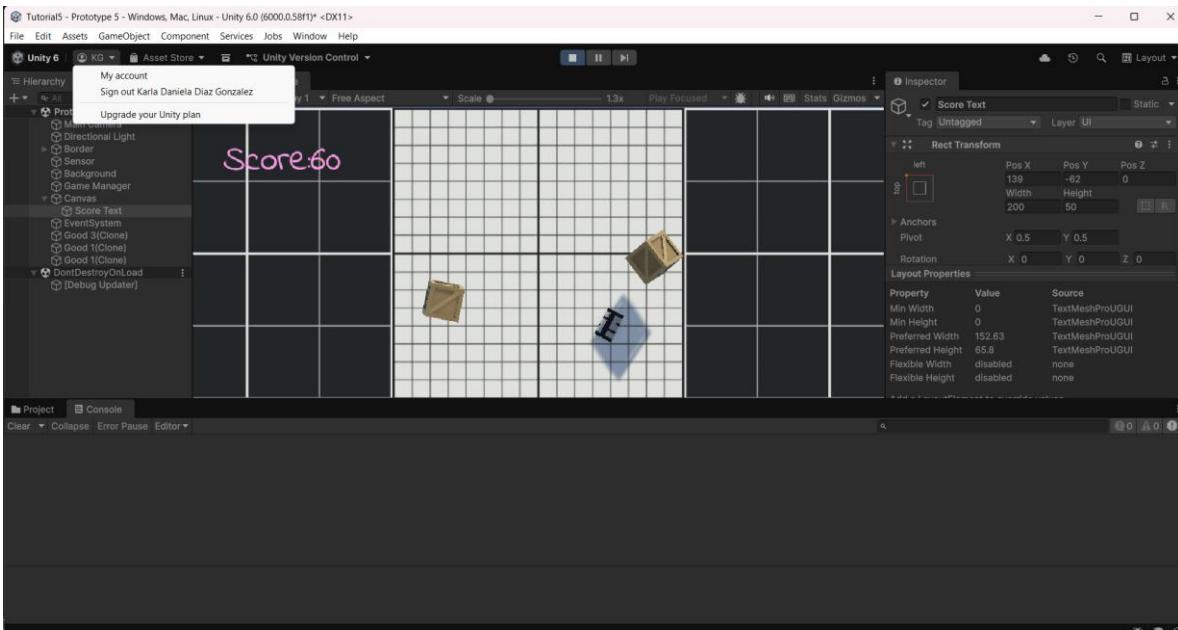
    void Start()
    {
        StartCoroutine(SpawnTarget());
        score = 0;
        UpdateScore(0);
    }

    void Update()
    {
    }

    IEnumerator SpawnTarget()
    {
        while (true)
        {
            yield return new WaitForSeconds(spawnRate);
            int index = Random.Range(0, targets.Count);
            Instantiate(targets[index]);
            UpdateScore(5);
        }
    }

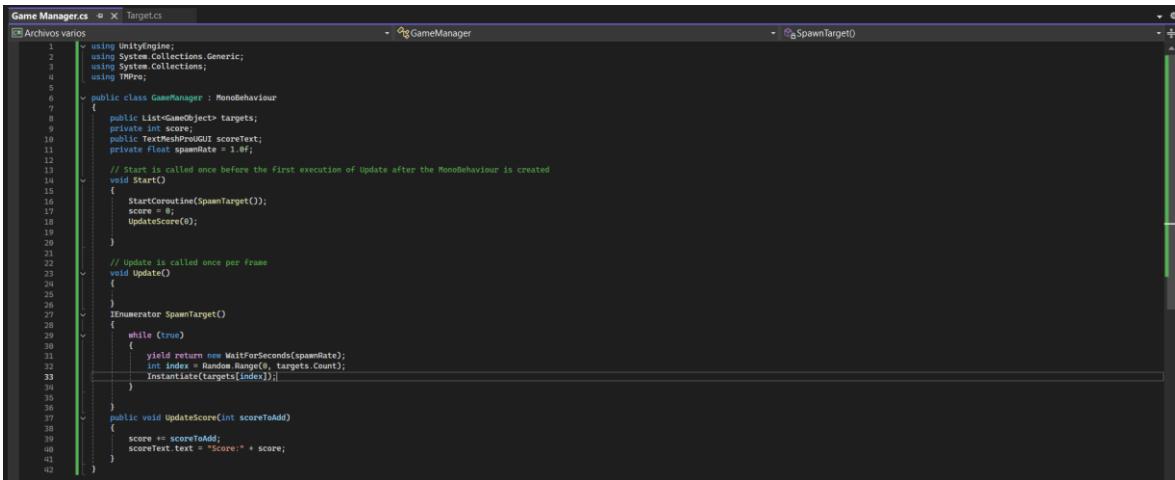
    void UpdateScore(int scoreToAdd)
    {
        score += scoreToAdd;
        scoreText.text = "Score: " + score;
    }
}

```



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

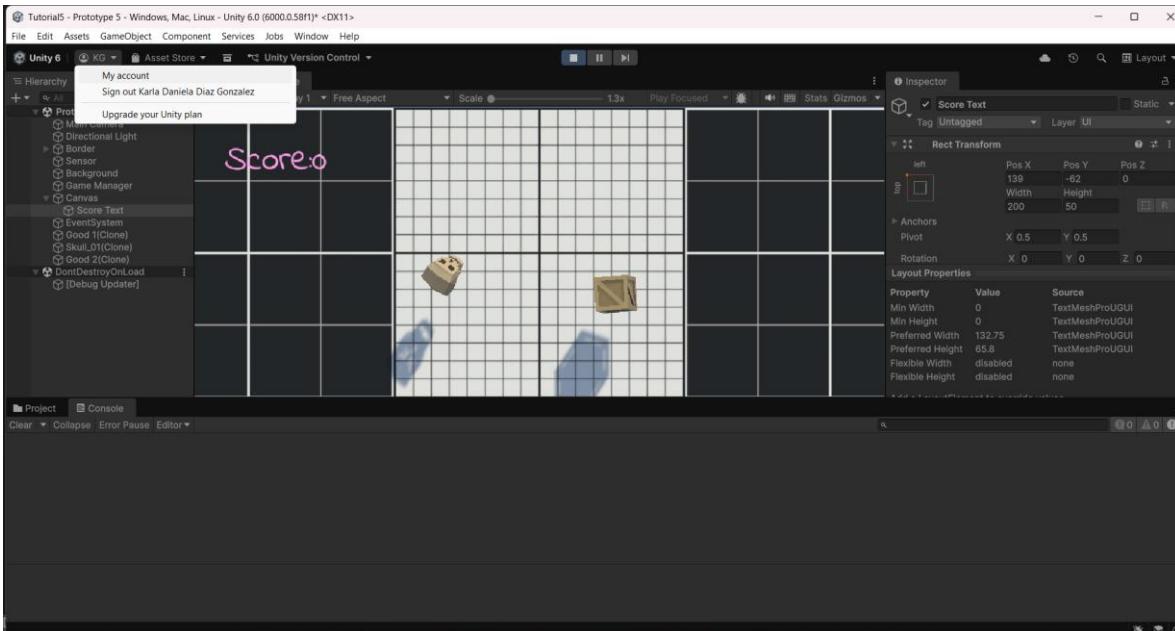
- En Target.cs, cree una referencia a la variable privada GameManager gameManager;
- Inicializa GameManager en Start() usando el método Find()
- Cuando se destruye un objetivo , llama a UpdateScore(5); y luego elimina la llamada al método de SpawnTarget().



```

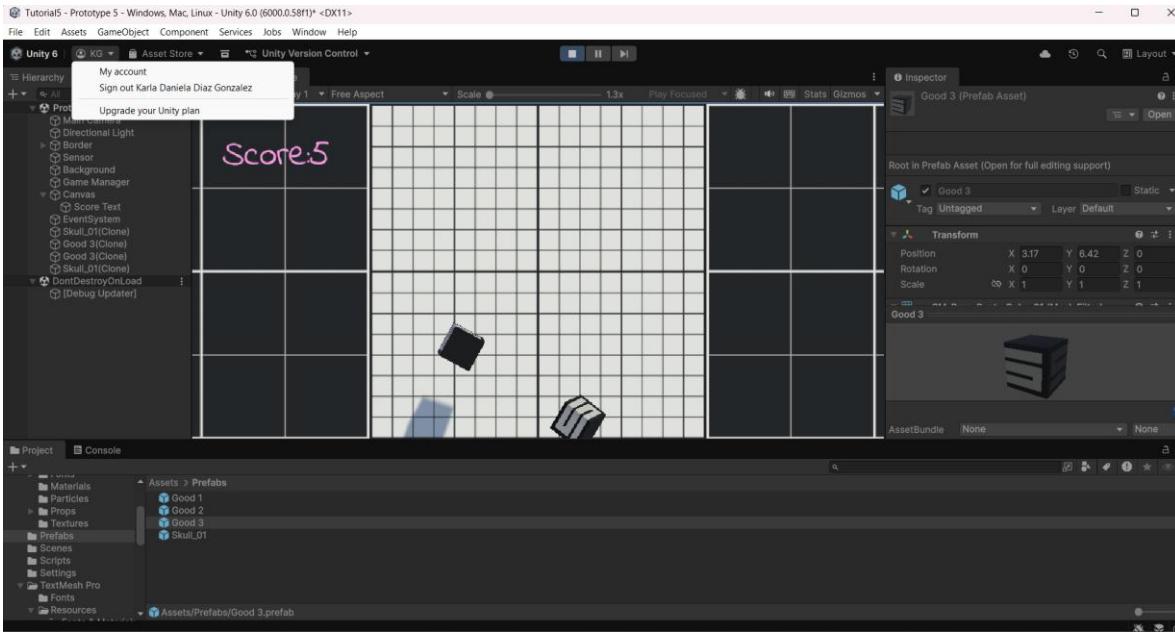
Game Manager.cs  Target.cs
Archivos varios
1  using UnityEngine;
2  using System.Collections.Generic;
3  using System.Collections;
4  using TMPro;
5
6  public class GameManager : MonoBehaviour
7  {
8      public List<GameObject> targets;
9      private int score;
10     public TextMeshProUGUI scoreText;
11     private float spamRate = 1.6f;
12
13     // Start is called once before the first execution of Update after the MonoBehaviour is created
14     void Start()
15     {
16         StartCoroutine(SpawnTarget());
17         score = 0;
18         UpdateScore();
19     }
20
21     // Update is called once per frame
22     void Update()
23     {
24     }
25
26     IEnumerator SpawnTarget()
27     {
28         while (true)
29         {
30             yield return new WaitForSeconds(spamRate);
31             int index = Random.Range(0, targets.Count);
32             Instantiate(targets[index]);
33         }
34     }
35
36     public void UpdateScore(int scoreToAdd)
37     {
38         score += scoreToAdd;
39         scoreText.text = "Score: " + score;
40     }
41 }
42

```



- 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);



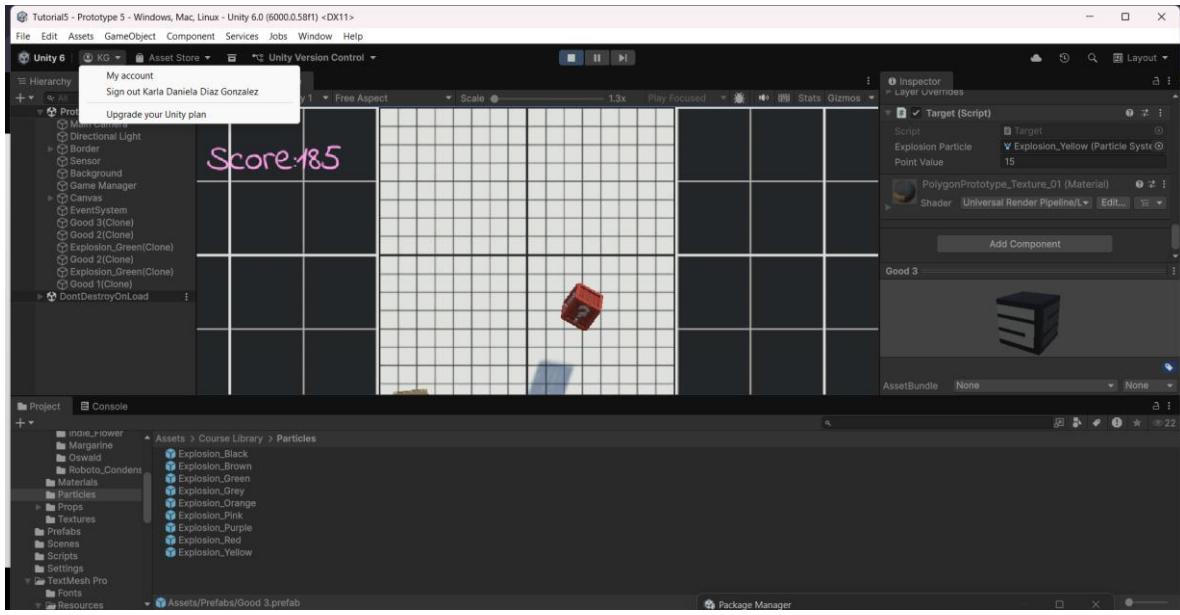
```

Game Manager.cs Target.cs

Archivos varios Target.cs
1 //using System;
2 //using System.Collections.Generic;
3 //using UnityEngine;
4
5 public class Target : MonoBehaviour
6 {
7     private Rigidbody targetRigidbody;
8     private Vector3 targetPosition;
9     private float minSpeed = 12;
10    private float maxSpeed = 15;
11    private float maxTorque = 10;
12    private float xSpan = 10;
13    private float ySpan = -6;
14
15    public int pointValue;
16
17    // Start is called before the first frame update
18    void Start()
19    {
20        targetRigidbody = GetComponent();
21        targetPosition = transform.parent.GetComponent("GameManager").GetRandomTarget();
22        targetRigidbody.AddForce(RandomVector(), ForceMode.Torque);
23        targetRigidbody.AddTorque(RandomTorque(), RandomTorque(), RandomTorque(), ForceMode.Impulse);
24        transform.position = RandomPosition();
25    }
26
27    // Update is called once per frame
28    void Update()
29    {
30    }
31
32    private void OnMouseDown()
33    {
34        Destroy(gameObject);
35        GameManager.UpdateScore(pointValue);
36    }
37
38    private void OnTriggerEnter(Collider other)
39    {
40        Destroy(gameObject);
41    }
42
43    Vector3 RandomForce()
44    {
45        return Vector3.up * Random.Range(minSpeed, maxSpeed);
46    }
47
48    float RandomTorque()
49    {
50        return Random.Range(-maxTorque, maxTorque);
51    }
52
53    Vector3 RandomSpanPos()
54    {
55        return new Vector3(Random.Range(-xSpan), ySpan);
56    }
57
58}

```

- 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.



```

Game Manager.cs Target.cs

public class Game Manager : MonoBehaviour
{
    public Transform spawnPoint;
    public Vector3 spawnRange;
    public float spawnTime;
    public int maxEnemies = 10;
    public float enemyHealth = 10;
    public float enemyDamage = -6;
    public ParticleSystem explosionParticle;
    public int pointValue;
}

public class Target : MonoBehaviour
{
    private Rigidbody targetRb;
    private float shielded = 12;
    private float health = 10;
    private float maxForce = 10;
    private float changeForce = 10;
    private float changeAngle = -6;

    public ParticleSystem explosionParticle;
    public int pointValue;

    // Start is called before the first frame update
    void Start()
    {
        targetRb = GetComponent<Rigidbody>();
        GameManager = FindObjectOfType<GameManager>();
        targetRb.AddForce(RandomForce(), ForceMode.Impulse);
        targetRb.AddTorque(RandomTorque(), RandomTorque(), ForceMode.Impulse);
        transform.position = RandomSpawnPos();
    }

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

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

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

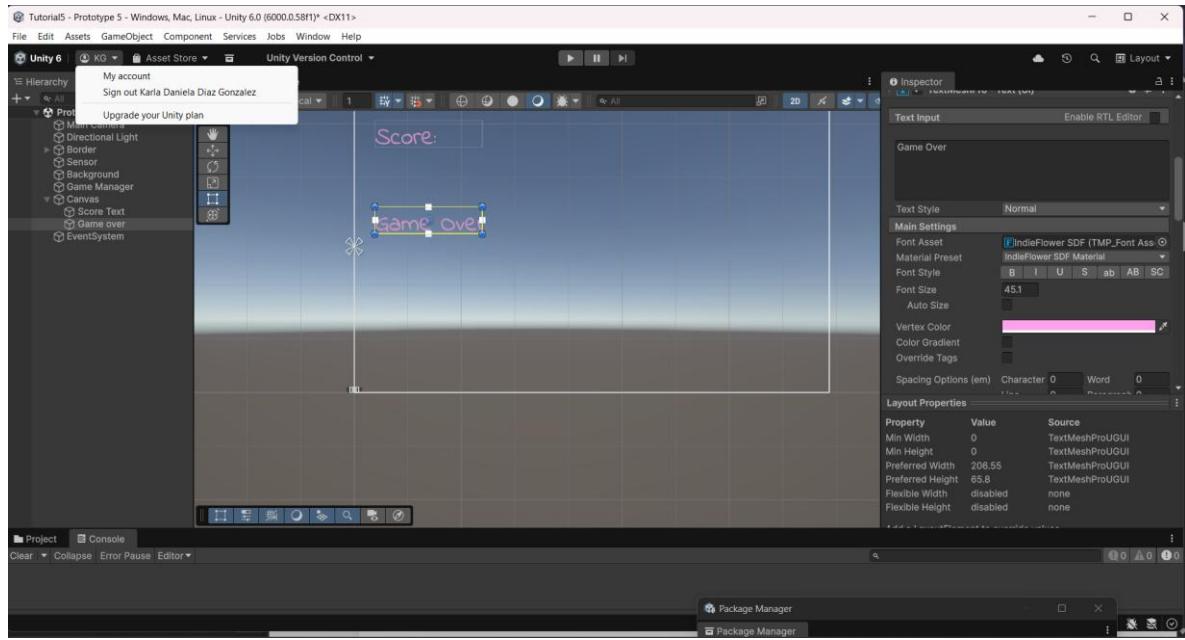
    Vector3 RandomForce()
    {
        return Vector3.up * Random.Range(minSpeed, maxSpeed);
    }

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

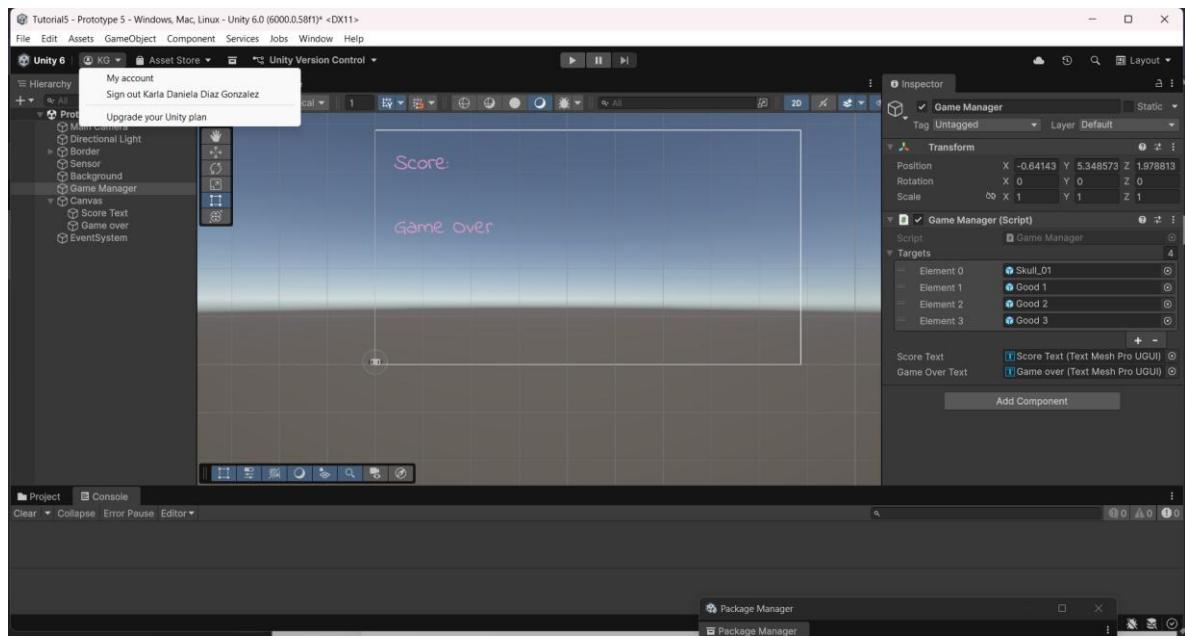
    Vector3 RandomSpawnPos()
    {
        return new Vector3(Random.Range(-changeX, changeX), ySpawn);
    }
}

```

- 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

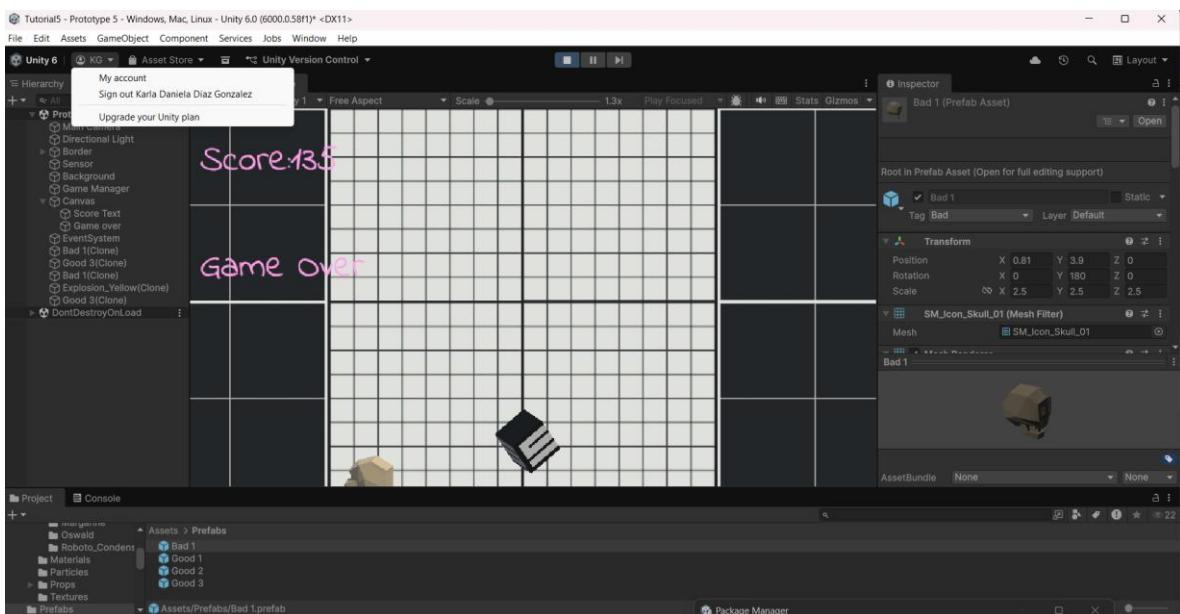


```

Game Manager.cs
Archivos varios
1  using UnityEngine;
2  using System.Collections.Generic;
3  using System.Collections;
4  using TMPro;
5
6  public class GameManager : MonoBehaviour
7  {
8      public List<GameObject> targets;
9      public TextMeshProUGUI scoreText;
10     public TextMeshProUGUI gameOverText;
11     private float spawnRate = 1.0f;
12
13     // Start is called before the first frame update
14     void Start()
15     {
16         StartCoroutine(SpawnTarget());
17         score = 0;
18         UpdateScore(0);
19
20         gameOverText.gameObject.SetActive(false);
21     }
22
23     // Update is called once per frame
24     void Update()
25     {
26     }
27
28     IEnumerator SpawningTarget()
29     {
30         while (true)
31         {
32             yield return new WaitForSeconds(spawnRate);
33             int index = Random.Range(0, targets.Count);
34             Instantiate(targets[index]);
35         }
36     }
37
38     public void UpdateScore(int scoreToAdd)
39     {
40         score += scoreToAdd;
41         scoreText.text = "Score:" + score;
42     }
43 }

```

- 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.



```

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

public class Target : MonoBehaviour
{
    private Rigidbody targetRB;
    private Transform targetTransform;
    private float maxSpeed = 12;
    private float maxTorque = 10;
    private float maxForce = 10;
    private float yPosition = -6;

    public ParticleSystem explosionParticle;
    public ParticleSystem deathParticle;

    // Start is called before the first frame update
    void Start()
    {
        targetRB = GetComponent();
        targetTransform = GetComponent();
        GetComponent<RandomForce>().GetComponent<GameManager>();
    }

    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Instantiate(explosionParticle, transform.position, explosionParticle.transform.rotation);
            GameManager.SpawnScoreText();
        }
    }

    void OnTriggerEnter(Collider other)
    {
        Destroy(gameObject);
        if (other.CompareTag("Wall"))
        {
            gameManager.GameOver();
        }
    }

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

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

    Vector3 RandomPosition()
    {
        return new Vector3(Random.Range(-xRange, xRange), yPosition);
    }
}

```

```

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

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

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

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

    IEnumerator SpawnerTarget()
    {
        while (true)
        {
            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;
    }

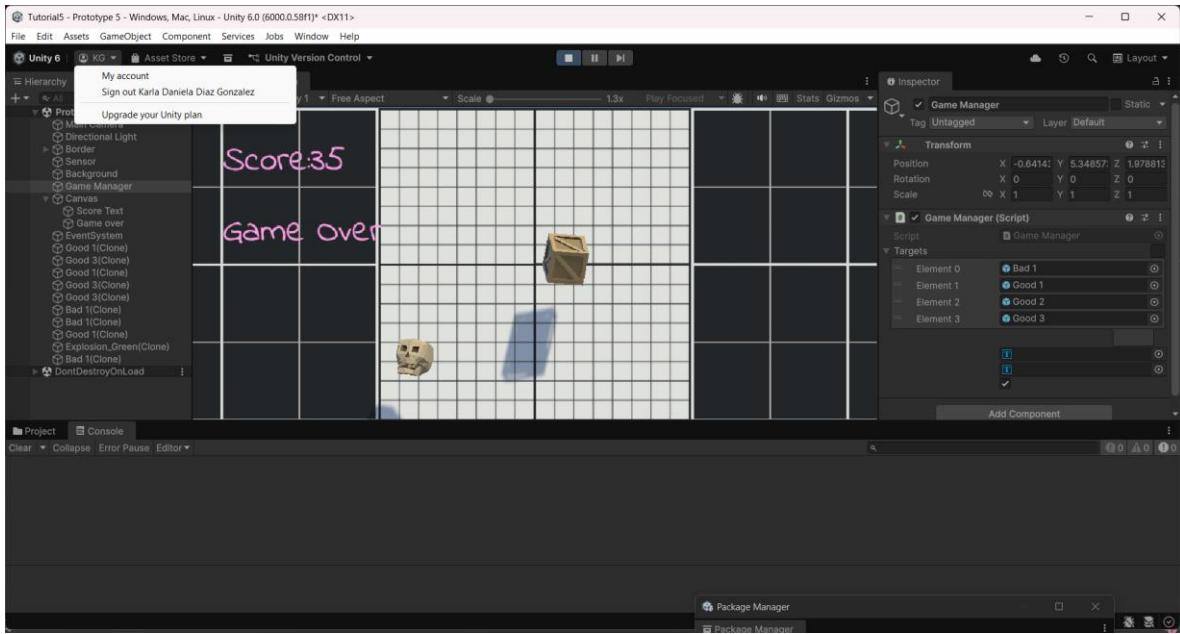
    public void GameOver()
    {
        gameOverText.gameObject.SetActive(true);
    }
}

```

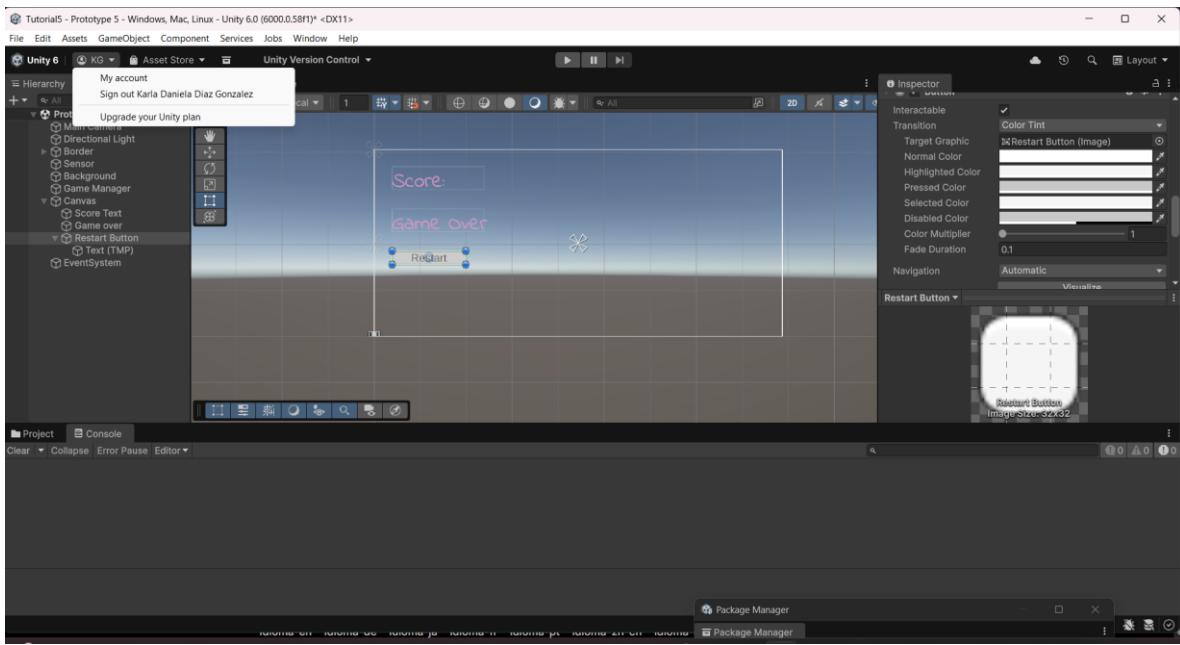
- Crea una nueva variable pública booleana isActive;
- En la primera línea de Start() , establezca isActive = true; y en GameOver() , establezca isActive = false;
- Para evitar la aparición de objetos, en la corutina SpawnerTarget() , cambie while (true) por while (isActive).
- Para evitar que se anoten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

```
GameManager.cs  Target.cs  Archivos varios
Archivos varios
using UnityEngine;
using System.Collections.Generic;
using System.Collections;
using TMPro;
using Random;
public class GameManager : MonoBehaviour
{
    public List<GameObject> targets;
    public TextMeshProUGUI scoreText;
    public TextMeshProUGUI gameOverText;
    public bool isGameOver;
    private int score;
    private float spamRate = 1.0f;
    // Start is called before the first frame update
    void Start()
    {
        isGameActive = true;
        Instantiate(SpawnTarget);
        score = 0;
        UpdateScore();
    }
    // Update is called once per frame
    void Update()
    {
        Ienumerator SpamTarget()
        {
            while (isGameActive)
            {
                yield return new WaitForSeconds(spamRate);
                int index = Random.Range(0, targets.Count);
                Instantiate(targets[index]);
            }
        }
        UpdateScore(scoreToAdd);
    }
    public void UpdateScore(int scoreToAdd)
    {
        score += scoreToAdd;
        scoreText.text = "Score:" + score;
    }
    public void GameOver()
    {
        gameOverText.gameObject.SetActive(true);
        isGameActive = false;
    }
}
```

```
GameManager.cs  Target.cs  Archivos varios
Archivos varios
using UnityEngine;
using System.Collections.Generic;
using Unity.Entities;
public class Target : MonoBehaviour
{
    private Rigidbody targetRb;
    private float targetSpeed = 10f;
    private float targetAngle = 30f;
    private float targetRadius = 5f;
    private float targetY = 0f;
    private float targetX = 0f;
    private float targetZ = 0f;
    private ParticleSystem explosionParticle;
    public ParticleSystem explosionParticle;
    // Start is called before the first frame update
    void Start()
    {
        targetRb = GetComponent();
        GameManager gameManager = GameObject.Find("Game Manager").GetComponent();
        targetRb.AddForceRandomForce();
        targetRb.AddTorqueRandomTorque();
        targetRb.AddTorqueRandomTorque();
        targetRb.AddTorqueRandomTorque();
        transform.position = RandomPosition();
    }
    // Update is called once per frame
    void Update()
    {
    }
    private void Initialize()
    {
        Destroy(gameObject);
        Instantiate(explosionParticle, transform.position, explosionParticle.transform.rotation);
        GameManager gameManager = GetComponent(<T>());
    }
    private void Initialize()
    {
        if (GameManager.isGameOver)
        {
            Instantiate(explosionParticle, transform.position, explosionParticle.transform.rotation);
            GameManager.UpdateScore(scoreValue);
        }
    }
    Vector3 RandomForce()
    {
        return Vector3.up * Random.Range(minSpeed, maxSpeed);
    }
    float RandomTorque()
    {
        return Random.Range(-maxTorque, maxTorque);
    }
    Vector3 RandomPosition()
    {
        return new Vector3(Random.Range(-targetX, targetX), targetY, targetZ);
    }
}
```

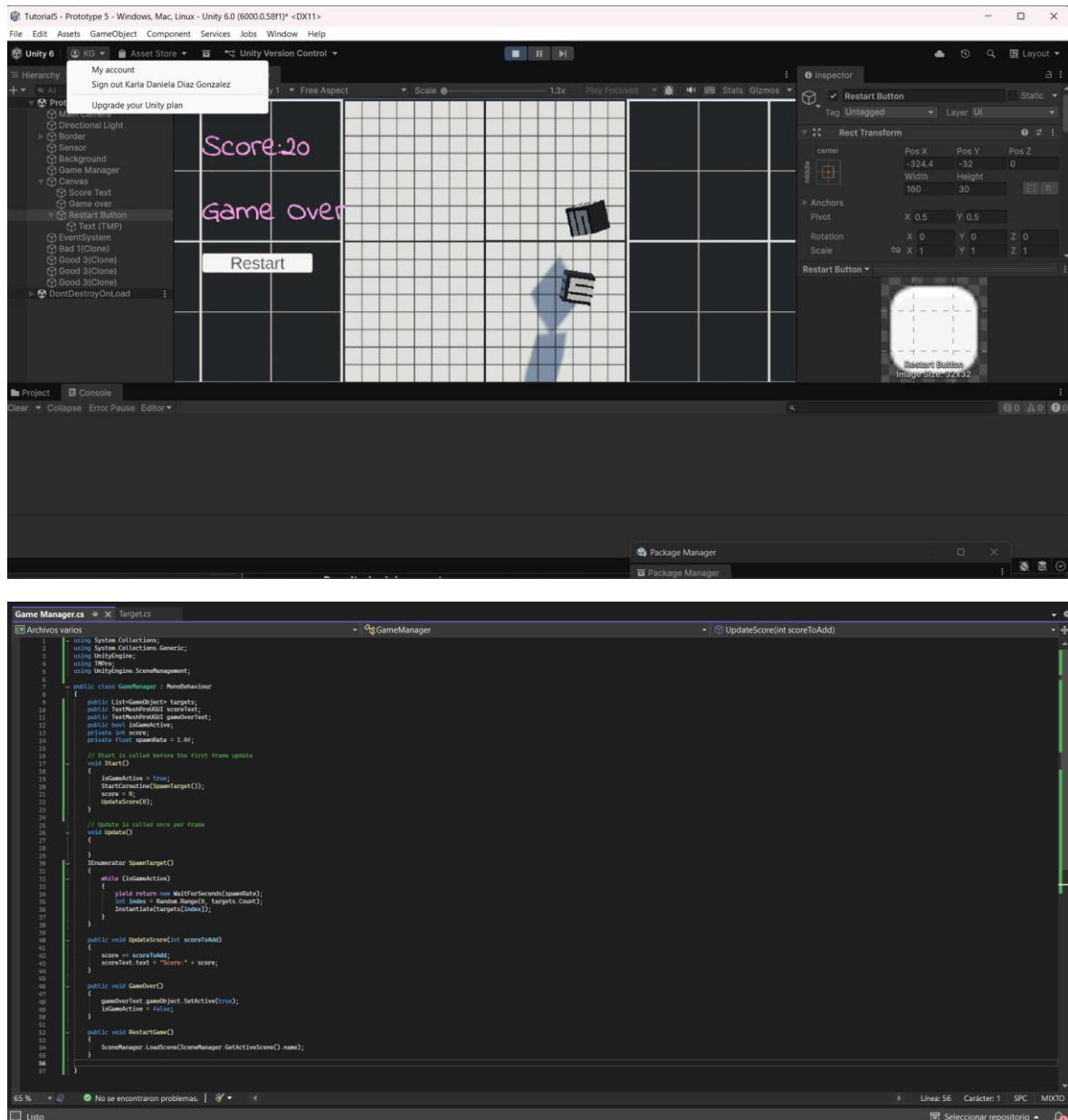


- Haz clic con el botón derecho en el lienzo y selecciona Crear > Interfaz de usuario > Botón - TextMeshPro .
- Cambiar el nombre del botón a “ Botón de reinicio ”
- Reactiva temporalmente el texto "Game Over" para reposicionar el botón de reinicio junto al texto y luego desactívalo de nuevo .
- Seleccione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



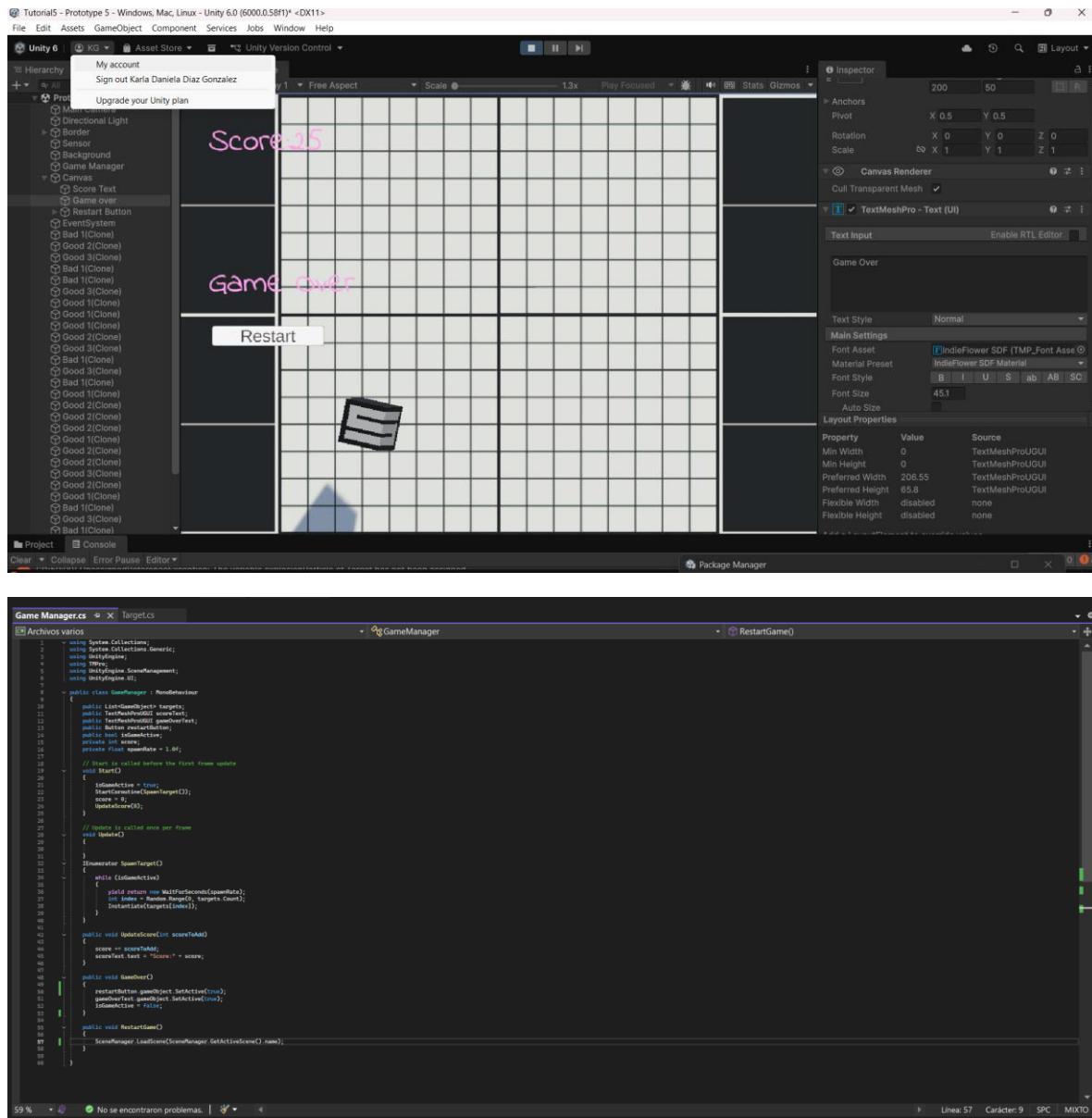
- En GameManager.cs, agregue using UnityEngine.SceneManagement;

- 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.

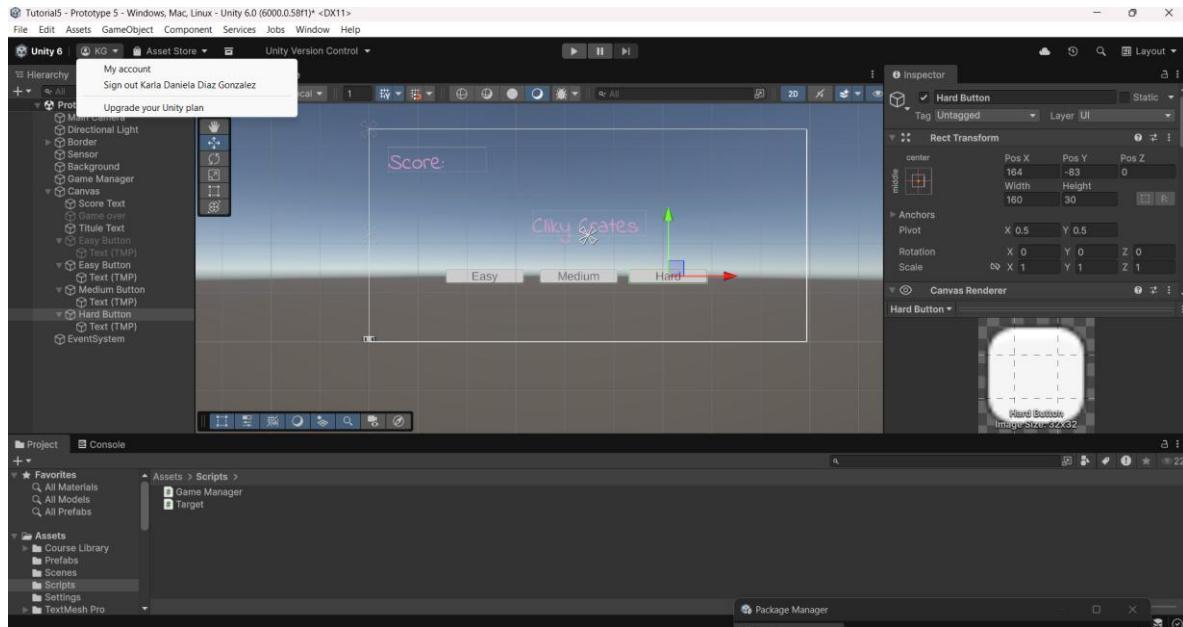


- En la parte superior de GameManager.cs agregue using UnityEngine.UI;
- Declara un nuevo botón público llamado restartButton; y asígnale el botón Reiniciar en el inspector.
- Desmarque la casilla de verificación “Activo” del botón Reiniciar en el inspector

- 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 ”.

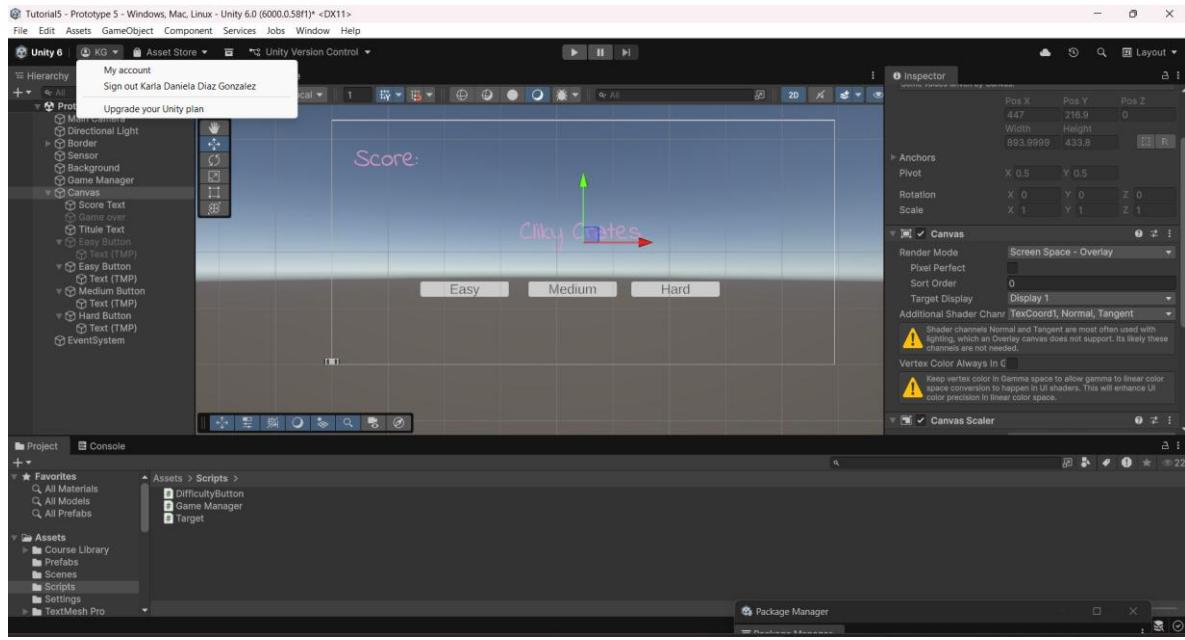


- Para los 3 botones nuevos, en el componente Botón, en la sección Al hacer clic () , haga clic en el botón menos (-) para eliminar la funcionalidad Reiniciar juego.
- Crea un nuevo script DifficultyButton.cs y adjúntalo a los 3 botones.
- Agrega using UnityEngine.UI a tus importaciones
- Crea una nueva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

```

DifficultyButton.cs  Game Manager.cs  Target.cs
Archivos varios
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class DifficultyButton : MonoBehaviour
7  {
8      private Button button;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         button = GetComponent<Button>();
14     }
15
16     // Update is called once per frame
17     void Update()
18     {
19     }
20
21 }
```

105 % No se encontraron problemas. Linea: 21 Carácter: 1 SPC CRLF



- 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 .

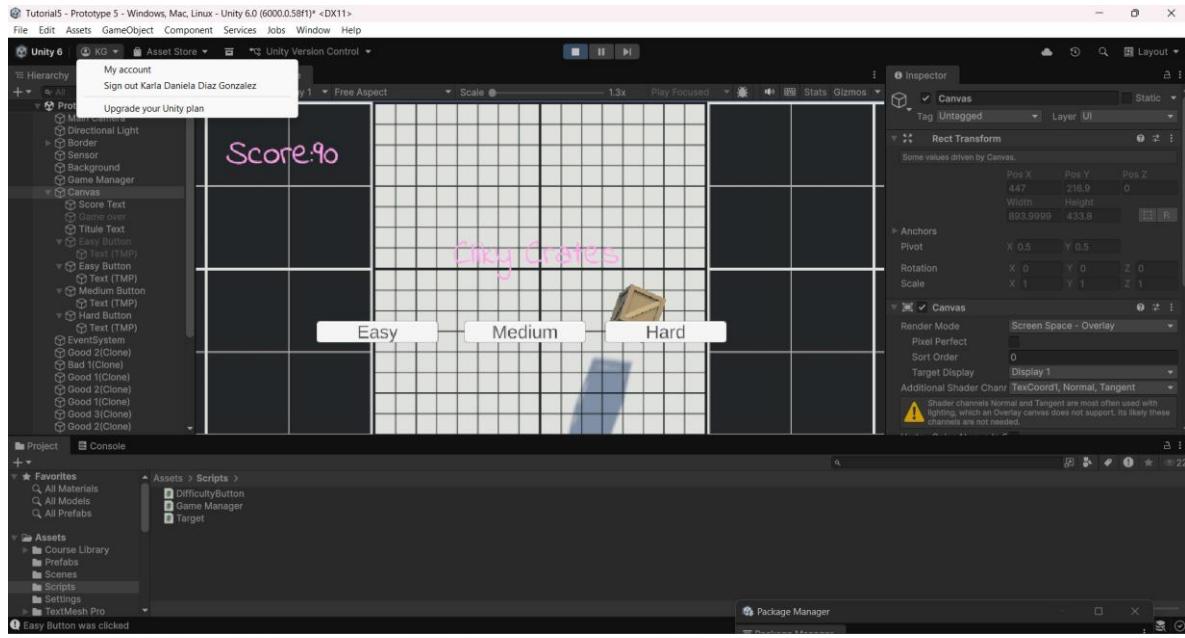
```

DifficultyButton.cs  X GameManager.cs  Target.cs
Archivos varios
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class DifficultyButton : MonoBehaviour
7  {
8      private Button button;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         button = GetComponent<Button>();
14         button.onClick.AddListener(SetDifficulty);
15     }
16
17     // Update is called once per frame
18     void Update()
19     {
20     }
21
22     void SetDifficulty()
23     {
24         Debug.Log(button.gameObject.name + " was clicked");
25     }
26 }
27

```

105 % No se encontraron problemas. Linea: 27 Carácter: 1 SPC CRLF

Elementos guardados Seleccionar repositorio



- 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();
    }
}

```

The screenshot shows the Unity code editor with the DifficultyButton.cs script open. The script defines a public class DifficultyButton that inherits from MonoBehaviour. It contains a private Button variable and a private GameManager variable named gameManager. The Start() method initializes the button and finds the GameManager component in the scene. The SetDifficulty() method is triggered by the button's onClick event and logs a message to the console indicating which button was clicked, then calls the StartGame() method on the gameManager. The code editor shows syntax highlighting and code completion suggestions.

```

using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections.Generic;
using System.Linq;
using System;

public class GameManager : MonoBehaviour
{
    public List targets;
    public TextMeshProUGUI scoreText;
    public Button restartButton;
    public Button resetButton;
    private int score;
    private float spawnRate = 1.0f;

    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
        if(score >= 100)
        {
            Application.Quit();
        }
    }

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

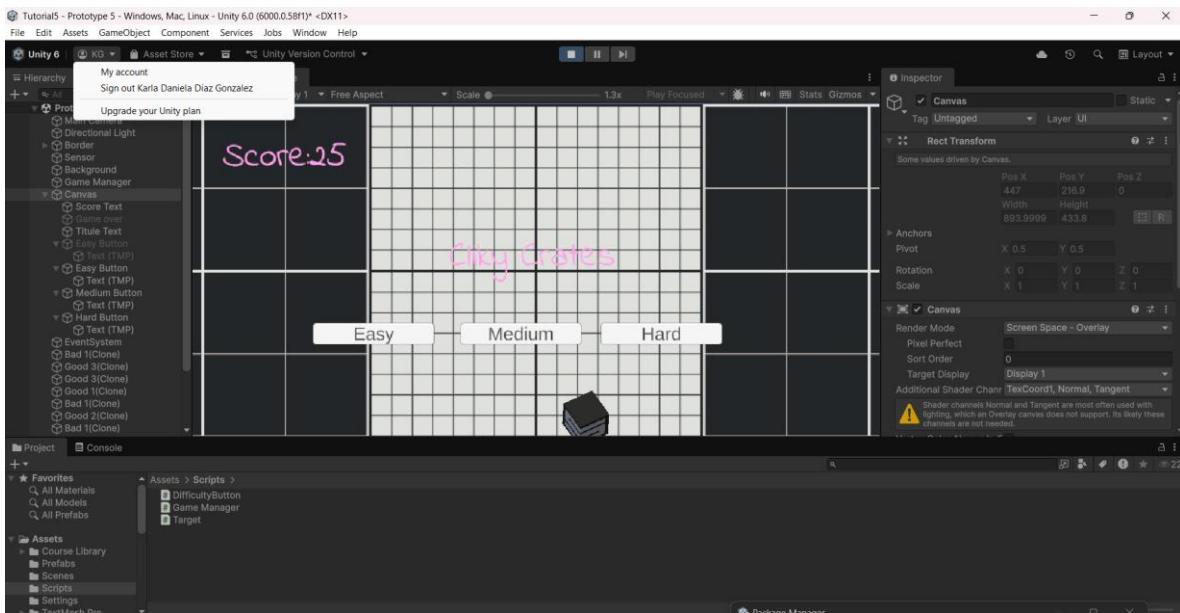
    public void UpdateScore(int scoreToAdd)
    {
        score += scoreToAdd;
        scoreText.text = "Score: " + score;
    }

    public void GameOver()
    {
        restartButton.gameObject.SetActive(true);
        resetButton.gameObject.SetActive(true);
        idleCount += 1;
    }

    public void RestartGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

    public void Interact()
    {
        idleCount += 1;
        StartCoroutine(SpawnTarget());
        UpdateScore(10);
    }
}

```

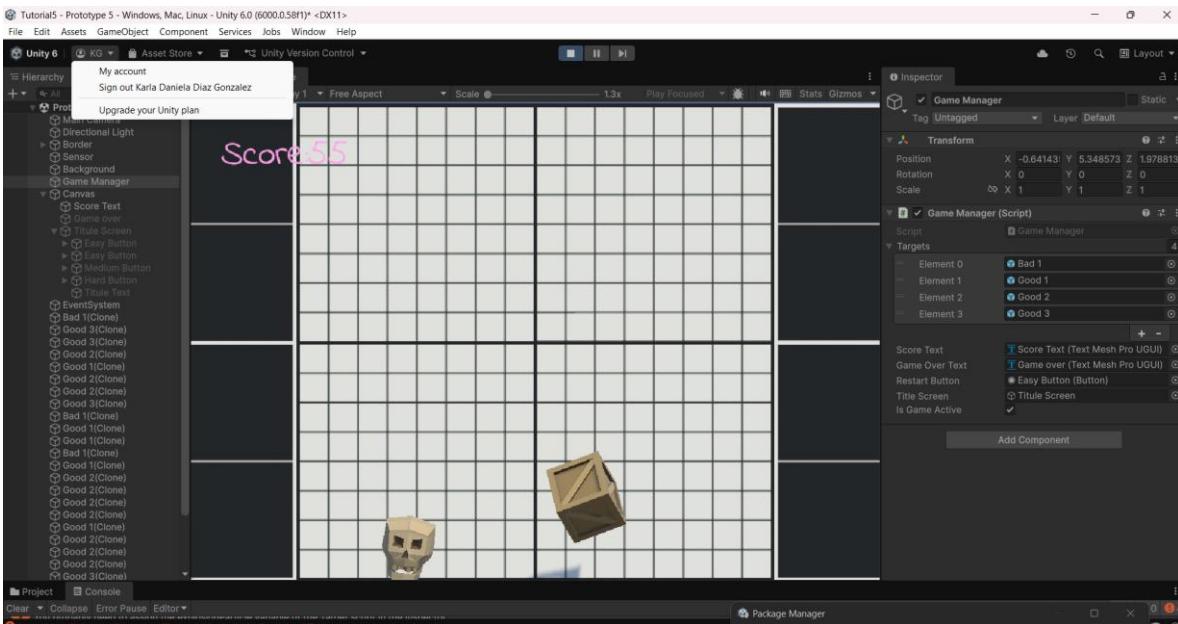


- Haz clic derecho en el lienzo y selecciona Crear > Objeto vacío , cámbiale el nombre a « Pantalla de título » y arrastra los 3 botones y el título a la pantalla.
- En GameManager.cs, crea un nuevo GameObject público llamado titleScreen y asígnalo en el inspector.
- En StartGame() , desactiva el objeto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

```

1  // using System.Collections;
2  // using System.Collections.Generic;
3  // using UnityEngine;
4  // using UnityEngine.SceneManagement;
5  // using UnityEngine.UI;
6  // using UnityEngine.Events;
7  // using System.Linq;
8
9  public class GameManager : MonoBehaviour
10 {
11     // public LineRenderer[] targets;
12     // public TextMeshPro[] scoreText;
13     // public Button restartButton;
14     // public TextMeshProUGUI scoreText;
15     // public float spawnRate;
16     // private float spawnRate = 1.0f;
17
18     // If Start() is called before the first frame update
19     void Start()
20     {
21     }
22
23     // Update is called once per frame
24     void Update()
25     {
26         //IEnumerator spawnTarget()
27         //while (true)
28         //{
29             //yield return new WaitForSeconds(spawnRate);
30             //Instantiate(targets[Random.Range(0, targets.Length)]);
31         //}
32     }
33
34     public void UpdateScore(int scoreToAdd)
35     {
36         score += scoreToAdd;
37         scoreText.text = "Score: " + score;
38     }
39
40     public void GameOver()
41     {
42         restartButton.gameObject.SetActive(true);
43         scoreText.gameObject.SetActive(true);
44         isGameOver = true;
45     }
46
47     public void Restart()
48     {
49         ScoreManager.LoadScene(ScoreManager.GetActiveScene());
50     }
51
52     public void StartGame()
53     {
54         isGameOver = false;
55         score = 0;
56         ScoreManager.LoadScene(ScoreManager.GetActiveScene());
57         titleScreen.gameObject.SetActive(false);
58     }
59 }

```



- 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
```

```
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. Lines: 32 Carácter: 2 SPC MIXTO Seleccionar repositorio

DifficultyButton.cs Game Manager.cs Target.cs

```
Archivos varios
```

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

public class GameManager : MonoBehaviour
{
    public Client gameClient;
    public List<GameObject> targets;
    public TextMeshProUGUI scoreText;
    public TextMeshProUGUI timeText;
    public TextMeshProUGUI titleText;
    public TextMeshProUGUI levelText;
    private float generateTime = 1.0f;

    // Start is called before the first frame update
    void Start()
    {
    }

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

    void GenerateTarget()
    {
        while (index < max)
        {
            yield return new WaitForSeconds(generateTime);
            CreateTarget(targets.Count);
            index++;
        }
    }

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

    public void GameOver()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        gameClient.gameObject.SetActive(false);
        gameClient.gameObject.SetActive(true);
        SceneManager.LoadScene(0);
    }

    public void RestartLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

    public void StartGame(bool difficulty)
    {
        index = 0;
        generateTime = difficulty;
        scoreText.text = "0";
        titleText.text = "Title";
        levelText.text = "Level 0";
        titleScreen.gameObject.SetActive(false);
    }
}
```

49 % No se encontraron problemas. Lines: 70 Carácter: 1 SPC MIXTO Seleccionar repositorio

