# Introducción

En este documento se detalla el desarrollo de un juego de romper bloques hecho con el motor de videojuegos Unity3D la versión 2020.3.27f1.

Este juego fue desarrollado como una entrega final de la materia de programación de videojuegos en la universidad tecnológica de Santiago (UTESA).

En este se detallan cada una de las etapas de desarrollo, planes futuros, presupuestos, componentes del juego, programación, salida al mercado, test de usuarios, ETC.

Espero que le sea de su agrado y utilidad.

# CAPÍTULO I: VIDEOJUEGO Y HERRAMIENTAS DE DESARROLLO

## Descripción

El juego va a llevar el nombre de Rompe Bloques, el juego va a tratar sobre romper todos los bloques haciendo rebotar una pelota con una raqueta que podrá mover el jugador, el juego para un solo jugador y este tendrá 3 vidas para pasar los 3 niveles que incluirá el juego.

## Motivación

La inspiración para el juego me llego la hacer uno de los proyectos pasados (el de Pong) que tiene una temática similar y por mi aprecio por la cultura japonesa ya que de ahí es que han salido muchos me mis videojuegos favoritos.

### Originalidad de la idea

El juego esta inspirado en otros juegos similares como BrickBreacker y otros pero se diferencia de estos en la estética la cual le he querido dar un toque japones con las paredes del juego siendo una imitación simplificada de un arco torii de los que se encuentran en Japón y también haciendo que el juego tenga gráficos 2.5D.

### Estado del Arte

Use imágenes de Google como inspiración para hacer la imitación del arco, imagenes como esta:

También use una fuente parecida a la del juego Monster Hunter Rise para la tipografía del juego por su esterica de escritura tradicional japonesa:


# Objetivo general

Tener un juego que sea entretenido para jugadores de todos tipo, principalmente jugadores casuales. Que tenga la menor cantidad de errores posibles y que se ejecute en una gran cantidad de dispositivos.

# Objetivos específicos

1. Crear escenarios con niveles de dificultad balanceados entre los niveles.
2. Minimizar el tiempo de arranque del juego y de carga de niveles.
3. Crear escenarios creativos y divertidos.

# Escenario

El escenario son tres pareces con curvaturas, una raqueta en la parte inferior que controla el jugador y una bola que rebota con las paredes y con la raqueta a cierta velocidad dependiendo del nivel.

El juego tiene como lugar de escenario un campo de japón con un arco de torii de diferentes colores.

La escena de inicio presenta una imagen del juego, el nombre del juego y mi nombre y matricula.

# Contenidos

El juego contara inicialmente con tres niveles, cada uno mas difícil que el anterior, estos niveles variaran en la disposición de los bloques a romper y en la velocidad en la que se mueve la bola, también para que no se siente que se esta repitiendo el mismo nivel se cambiara el color del escenario por cada nivel.

## Metodología

Para el desarrollo de este juego pienso usar una metodología de desarrollo en cascada, creando las partes del juego de manera estructurada, esto no sera un problema para el desarrollo porque estoy desarrollándolo yo solo y por lo tanto solo puedo desarrollar una parte a la vez, en cuanto a la consecución de proyecto pienso desarrollar los niveles de manera ascendente y probándolos para validar errores y la dificultad de este.

## Arquitectura de la aplicación

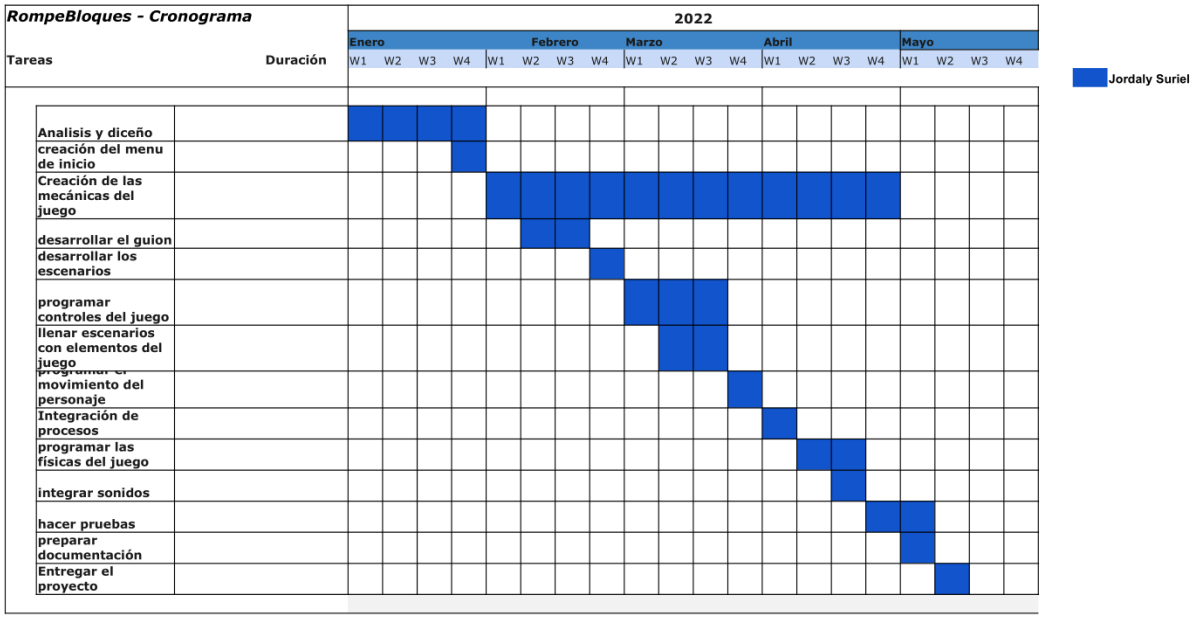
El juego estará compuesto de 4 escenas, una de inicio y cuatro niveles en su fase de lanzamientos, las escenas presentan escenarios en 3 dimensiones. Los elementos del juegos están formados por formas geométricas generadas en Unity 3d. Y toda la lógica del juegos se ejecuta dentro del motor de Unity.

## Herramientas de desarrollo

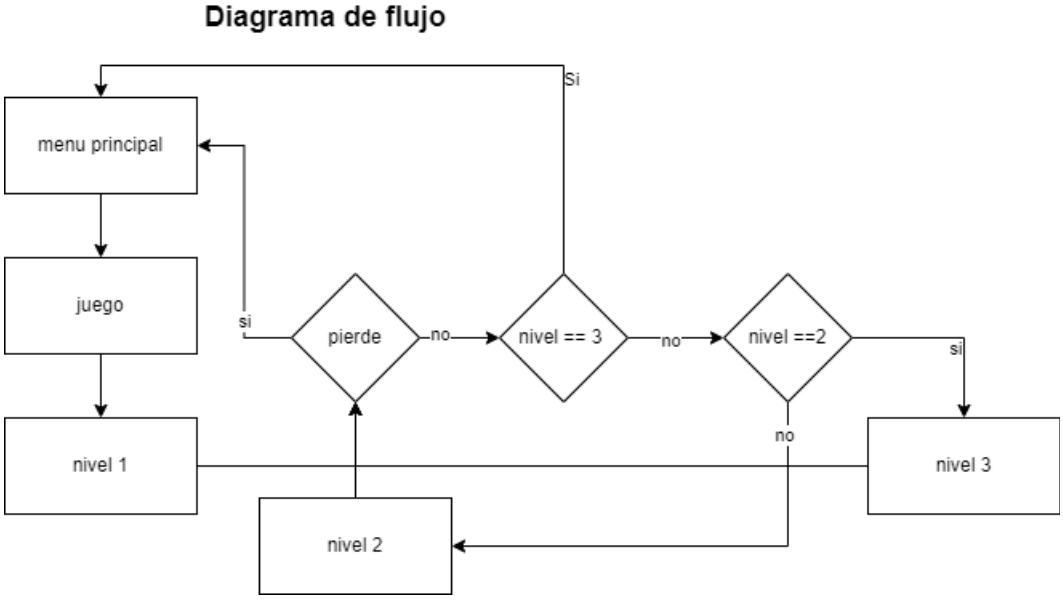
1. Unity en su versión 2020.3.27f1
2. Visual Studio Code
3. Google para las fuentes y la música del juego.

# CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN

## 2.1 Planificación (Diagrama de Gantt)




## 2.2 Diagramas y Casos de Uso




Los casos de uso solo son el menú principal y el juego.

Desde el menú principal se podrán visualizar los controles y la forma de inicial el juego y desde el juego se podrá jugar hasta que se pierda o gane para volver la menú principal.

## 2.3 Plataforma

Las plataformas para el juego serán las siguientes:

1. Windows
2. Linux
3. Mac
4. Web

Como se puede ver son todas las plataformas soportadas por Unity a las que no se le tiene que hacer cambios drásticos para su funcionamiento.

## 2.4 Género

El genero del juego será Arcade, un género de juegos sencillos, sin historia o muy poca y que se centran en una mecánica de juego entretenida.

## 2.5 Clasificación

La clasificación del juego sera E para toda clase de públicos, porque no incluirá violencia de ningún tipo.

## 2.6 Tipo de Animación

Las animaciones serán en 3D generada automáticamente por el motor de físicas de Unity cuando algún objeto colisiona con otro y cuando la bola choca con uno de los bloques.

## 2.7 Equipo de Trabajo

Director: Jordaly Suriel

Diseñador: Jordaly Suriel

Animador: Jordaly Suriel

Programador: Jordaly Suriel

Ing. Sonido: Jordaly Suriel

Administrador de la comunidad: Jordaly Suriel

## 2.8 Historia

El juego al ser del genero árcade no tiene una historia, pero se podría decir que eres un bloque que se revela contra los demás bloques y intenta destruir su ejercito de bloques soldados con tu bola demoledora para forjar tu propia dinastía.

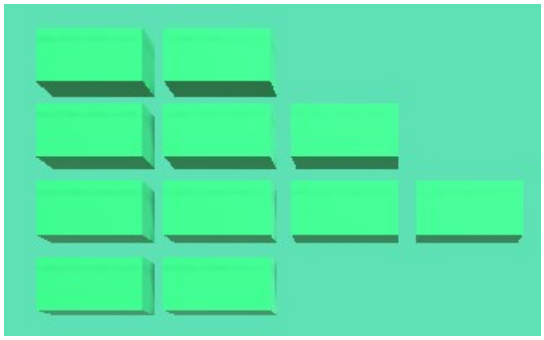
## 2.9 Guion

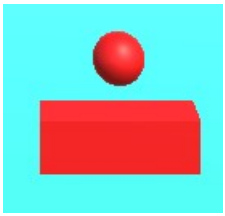
El objetivo del juego es derrotar a todos los bloques del nivel para pasar al siguiente evitan que la bola pase de la parte inferior de la pantalla para evitar perder vidas.

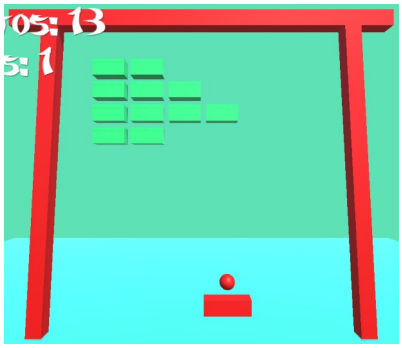
Los personales serán el bloque que controla el jugador y los bloques del nivel que se encontraran en la parte superior, también se puede considerar como personaje la bola que usa el jugador para destruir los otros bloques.

El juego se dividirá en dos partes, la pantalla de inicio y los niveles del juego.

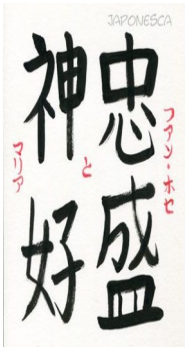
## 2.10 Storyboard













## 2.11 Personajes

1. Bloque controlado por el jugador.
2. Bloques enemigos en el escenario.
3. Bola.
4. las paredes del escenario.

## 2.12 Niveles

El juego contará con 3 niveles cada uno más difícil que el anterior, haciendo variar la velocidad de la pelota y el numero de cuadros en pantalla y el color del escenario.

## 2.13 Mecánica del Juego

Como mecánicas del juego tenemos las siguientes:

1. El jugador puede lanzan la pelota en el momento que quiere pero esta saldrá disparada en una dirección aleatoria.
2. La pelota destruirá los bloques enemigos con los que choque.
3. La pelota rebotará luego luego de destruir uno de los bloques enemigos.
4. La pelota revotará con las paredes del escenario.
5. La pelota rebotará si choca con el bloque controlado por el jugador.
6. Si la pelota pasa la posición del jugador en la parte inferior de la pantalla el jugador perderá una vida.
7. El jugador tendrá 3 vidas.
8. Cada vez que el jugador destruya un bloque se le sumara un punto al contador.
9. El jugador podrá recuperar una vida si destruye uno de los bloques especiales en el escenario.
10. Si el jugador se queda sin vidas pierde la partida y regresa al menú principal.

# CAPÍTULO III: DESARROLLO

## 3.1 Capturas de la Aplicación (Documentación completa del desarrollo, Scripts, Sprites, Prefabs e imágenes)

C# Barra.cs

Assets > Codigo > C# Barra.cs

```
1   using UnityEngine;
2   using System.Collections;
3
4   public class Barra : MonoBehaviour {
5
6       public float velocidad = 0.4f;
7
8       Vector3 posicionInicial;
9
10      // Use this for initialization
11      void Start () {
12          posicionInicial = transform.position;
13      }
14
15      public void Reset()
16      {
17          transform.position = posicionInicial;
18      }
19
20      // Update is called once per frame
21      void Update () {
22          float tecladoHorizontal = Input.GetAxisRaw("Horizontal");
23          float posX = transform.position.x + (tecladoHorizontal * velocidad * Time.deltaTime);
24          transform.position = new Vector3(Mathf.Clamp(posX, -8, 8), transform.position.y, transform.position.z);
25      }
26  }
27
```

C# Bloque.cs

Assets > Codigo > C# Bloque.cs

```
1   using UnityEngine;
2   using System.Collections;
3
4   public class Bloque : MonoBehaviour {
5
6       public GameObject efectoParticulas;
7       public Puntos puntos;
8
9       // Is Trigger DESACTIVADO
10      void OnCollisionEnter()
11      {
12          Instantiate(efectoParticulas, transform.position, Quaternion.identity);
13          Destroy(gameObject);
14          transform.SetParent(null);
15          puntos.GanarPunto();
16      }
17
18  }
```

C# BotonSalir.cs

Assets > Codigo > C# BotonSalir.cs

```
1   using UnityEngine;
2   using System.Collections;
3
4   public class BotonSalir : MonoBehaviour {
5
6       public bool salir;
7
8       // Update is called once per frame
9       void Update () {
10          if (Input.GetKeyDown(KeyCode.Escape))
11          {
12              if (salir)
13              {
14                  Debug.Log("Salimos del juego");
15                  Application.Quit();
16              }
17              else
18              {
19                  Application.LoadLevel("Portada");
20              }
21          }
22      }
23  }
24
```

C# EmpezarPartida.cs

Assets > Codigo > C# EmpezarPartida.cs

```
1   using UnityEngine;
2   using System.Collections;
3
4   public class EmpezarPartida : MonoBehaviour {
5
6       // Update is called once per frame
7       void Update () {
8           if (Input.GetButtonDown("Fire1"))
9           {
10              Puntos.puntos = 0;
11              Vidas.vidas = 3;
12              Application.LoadLevel("Nivel01");
13          }
14      }
15  }
16
```

C# Pelota.cs

Assets > Codigo > C# Pelota.cs

```
using UnityEngine;
using System.Collections;

public class Pelota : MonoBehaviour {

    public float velocidadInicial = 600f;

    public Rigidbody rig;

    bool enJuego;

    Vector3 posicionInicial;

    public Transform barra;

    // Use this for initialization
    void Start () {
        posicionInicial = transform.position;
    }

    public void Reset()
    {
        transform.position = posicionInicial;
        transform.SetParent(barra);
        enJuego = false;
        DetenerMovimiento();
    }

    public void DetenerMovimiento()
    {
        rig.isKinematic = true;
        rig.velocity = Vector3.zero;
    }

    // Update is called once per frame
    void Update () {
        if(!enJuego && Input.GetButtonDown("Fire1"))
        {
            enJuego = true;
            transform.SetParent(null);
            rig.isKinematic = false;
            rig.AddForce(new Vector3(velocidadInicial, velocidadInicial, 0));
        }
    }
}
```

Puntos.cs

Assets > Codigo > Puntos.cs

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

public class Puntos : MonoBehaviour {

    public static int puntos = 0;
    public Text textoPuntos;

    public GameObject nivelCompletado;
    public GameObject juegoCompletado;

    public SiguienteNivel siguienteNivel;

    public Pelota pelota;
    public Barra barra;

    public Transform contenedorBloques;

    public SonidosFinPartida sonidosFinPartida;

    // Use this for initialization
    void Start () {
        ActualizarMarcadorPuntos();
    }

    void ActualizarMarcadorPuntos()
    {
        textoPuntos.text = "Puntos: " + Puntos.puntos;
    }

    public void GanarPunto()
    {
        Puntos.puntos++;
        ActualizarMarcadorPuntos();

        if (contenedorBloques.childCount <= 0)
        {
            pelota.DetenerMovimiento();
            barra.enabled = false;

            if (siguienteNivel.EsUltimoNivel())
            {
                juegoCompletado.SetActive(true);
            }
            else
            {
                nivelCompletado.SetActive(true);
            }

            sonidosFinPartida.NivelCompletado();

            siguienteNivel.ActivarCarga();
        }
    }
}
```

SiguienteNivel.cs

Assets > Codigo > SiguienteNivel.cs

```
using UnityEngine;
using System.Collections;

public class SiguienteNivel : MonoBehaviour {

    public string nivelACargar;
    public float retraso;

    [ContextMenu("Activar Carga")]
    public void ActivarCarga()
    {
        Invoke("CargarNivel", retraso);
    }

    void CargarNivel()
    {
        if (!EsUltimoNivel())
        {
            Vidas.vidas++;
        }
        Application.LoadLevel(nivelACargar);
    }

    public bool EsUltimoNivel()
    {
        return nivelACargar == "Portada";
    }

}

```

SonidosFinPartida.cs

Assets > Codigo > SonidosFinPartida.cs

```
using UnityEngine;
using System.Collections;

public class SonidosFinPartida : MonoBehaviour {

    public AudioSource audioSource;
    public AudioClip completado;
    public AudioClip gameOver;

    public void GameOver()
    {
        ReproduceSonido(gameOver);
    }

    public void NivelCompletado()
    {
        ReproduceSonido(completado);
    }

    void ReproduceSonido(AudioClip sonido)
    {
        audioSource.clip = sonido;
        audioSource.loop = false;
        audioSource.Play();
    }
}
```

SonidosPelota.cs

Assets > Codigo > C# SonidosPelota.cs

```
using UnityEngine;
using System.Collections;

public class SonidosPelota : MonoBehaviour {

    public AudioSource rebote;
    public AudioSource punto;

    void OnCollisionEnter(Collision otro)
    {
        if (otro.gameObject.CompareTag("Bloque"))
        {
            punto.Play();
        }
        else
        {
            rebote.Play();
        }
    }

}
```

Suelo.cs

Assets > Codigo > C# Suelo.cs

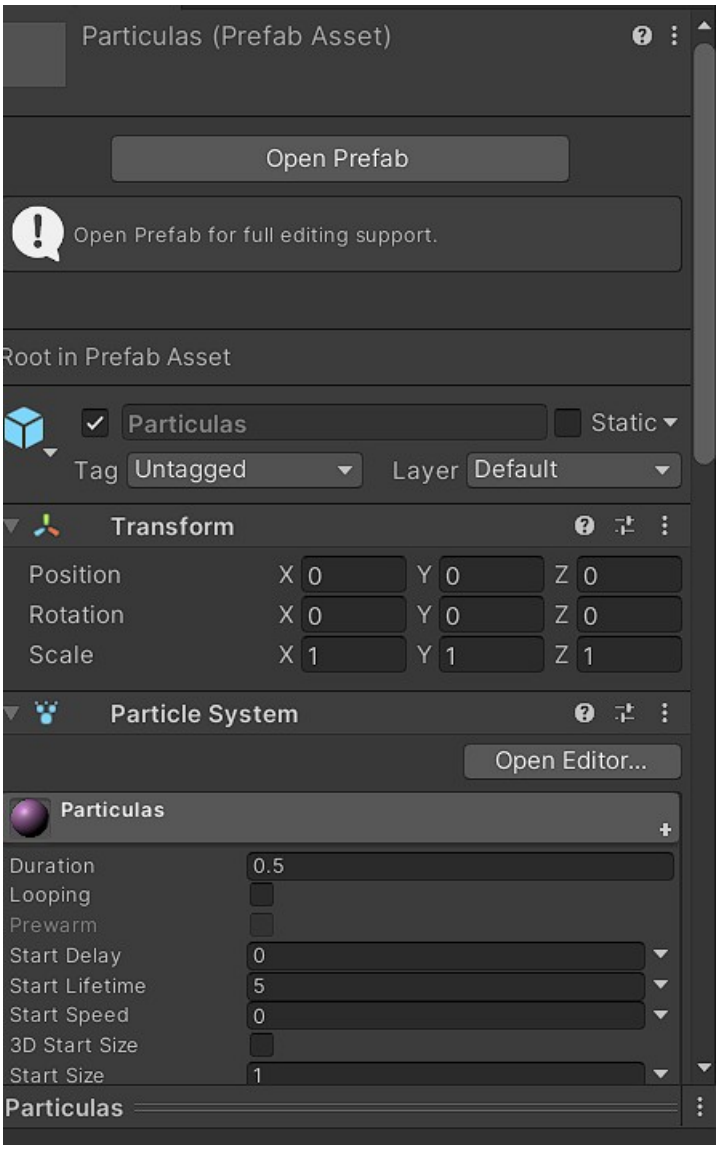
```
using UnityEngine;
using System.Collections;

public class Suelo : MonoBehaviour {

    public Vidas vidas;

    void OnTriggerEnter()
    {
        vidas.PerderVida();
    }
}

```

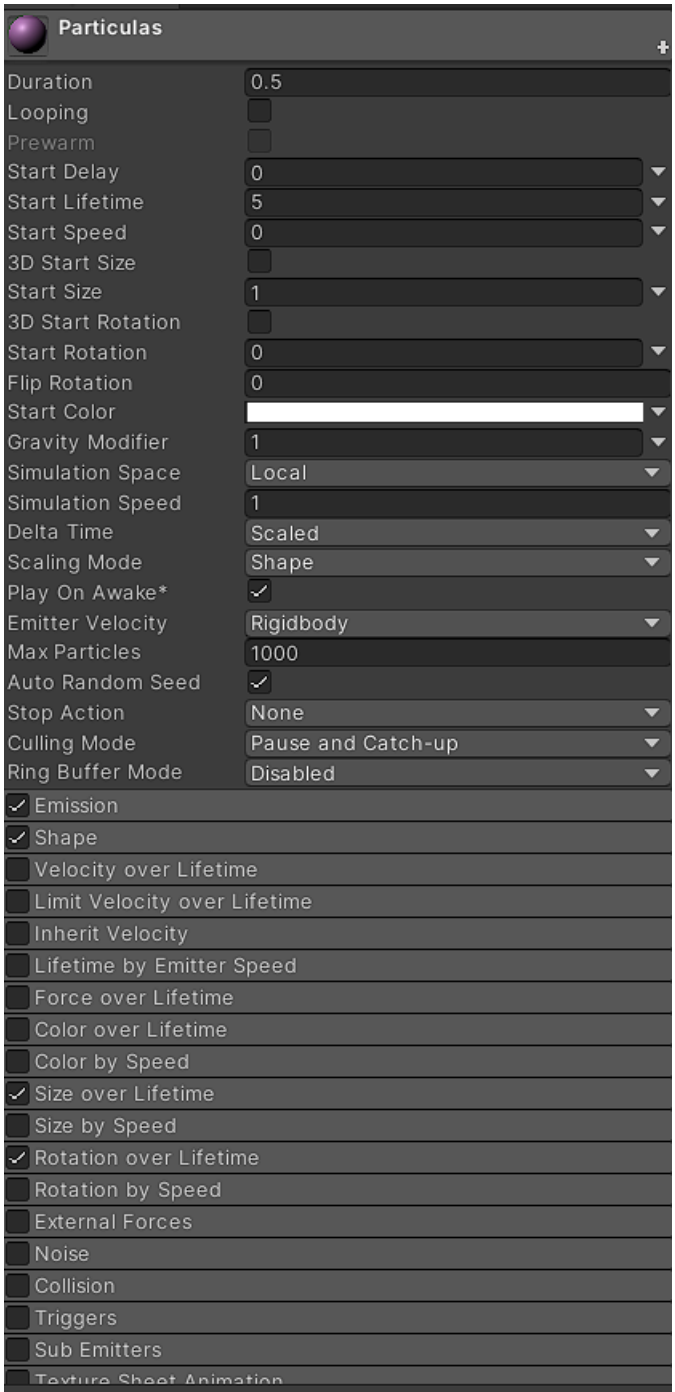
C# Vidas.cs

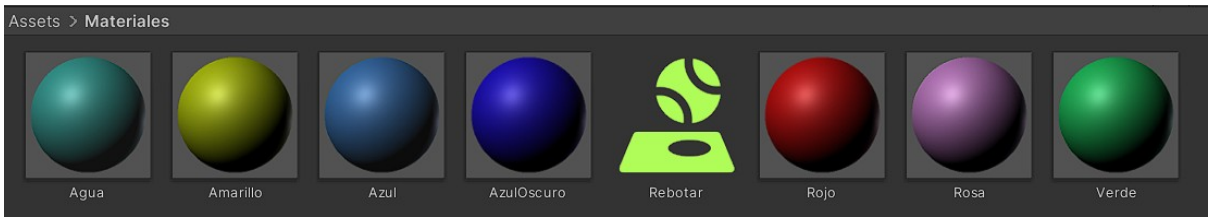
Assets > Codigo > C# Vidas.cs

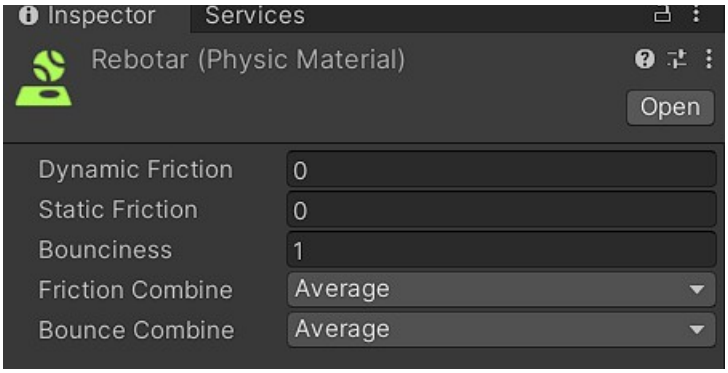
```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class Vidas : MonoBehaviour {

    public static int vidas = 3;

    public Text textoVidas;

    public Pelota pelota;
    public Barra barra;

    public GameObject gameOver;
    public SiguienteNivel siguienteNivel;

    public SonidosFinPartida sonidosFinPartida;

    // Use this for initialization
    void Start () {
        ActualizarMarcadorVidas();
    }

    void ActualizarMarcadorVidas()
    {
        textoVidas.text = "Vidas: " + Vidas.vidas;
    }

    public void PerderVida()
    {
        if (vidas <= 0) return;

        Vidas.vidas--;
        ActualizarMarcadorVidas();

        if (vidas <= 0)
        {
            sonidosFinPartida.GameOver();
            // Mostraremos GameOver
            gameOver.SetActive(true);
            pelota.DetenerMovimiento();
            barra.enabled = false;

            siguienteNivel.nivelACargar = "Portada";
            siguienteNivel.ActivarCarga();

        }
        else
        {
            barra.Reset();
            pelota.Reset();
        }
    }
}
```


Assets > Fonts
Aa
Karate


Particulas (Prefab Asset)
Open Prefab
Open Prefab for full editing support.
Root in Prefab Asset
Particulas
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Particle System
Open Editor...
Particulas
Duration 0.5
Looping
Prewarm
Start Delay 0
Start Lifetime 5
Start Speed 0
3D Start Size
Start Size 1
Particulas


Particulas
Duration 0.5
Looping
Prewarm
Start Delay 0
Start Lifetime 5
Start Speed 0
3D Start Size
Start Size 1
3D Start Rotation
Start Rotation 0
Flip Rotation 0
Start Color
Gravity Modifier 1
Simulation Space Local
Simulation Speed 1
Delta Time Scaled
Scaling Mode Shape
Play On Awake*
Emitter Velocity Rigidbody
Max Particles 1000
Auto Random Seed
Stop Action None
Culling Mode Pause and Catch-up
Ring Buffer Mode Disabled
Emission
Shape
Velocity over Lifetime
Limit Velocity over Lifetime
Inherit Velocity
Lifetime by Emitter Speed
Force over Lifetime
Color over Lifetime
Color by Speed
Size over Lifetime
Size by Speed
Rotation over Lifetime
Rotation by Speed
External Forces
Noise
Collision
Triggers
Sub Emitters
Texture Sheet Animation


Assets > Materiales
Agua
Amarillo
Azul
AzulOscuro
Rebotar
Rojo
Rosa
Verde


Inspector
Services
Rebotar (Physic Material)
Open
Dynamic Friction
0
Static Friction
0
Bounciness
1
Friction Combine
Average
Bounce Combine
Average


Assets > Sprites
BotonIzquierda
Icono


Assets > Sounds
completado
error
gameover
pelota
point

## 3.2 Prototipos

El juego tendrá dos prototipos:

- El primero se vio en la entrega del segundo parcial de la materia de programación de videojuegos el cual tenia una cantidad limitada de niveles y funcionalidad.
- Para el segundo prototipo se contara con un nivel mas de dificultad y con mas mecánicas de juego en la fecha del tercer parcial de programación de videojuegos.

## 3.3 Perfiles de Usuarios

El juego al ser del genero árcade no tendrá tipos de usuario solo jugadores sin hacer distensión entre ellos. Solo iniciar el juego y ganar o perder.

## 3.4 Usabilidad

El juego no tendrá mucha opciones, por lo tanto no sera complicado de entender y prodra ser usado incluso por niños pequeños.

## 3.5 Test

| | |
|---|---|
| Sexo | Mujer |
| Edad | 8 |
| Nivel de estudios | primaria |
| Aficiones | Jugar videojuegos |
| **Resultados** | |
| Tareas | Puntuación |
| Jugabilidad | 7 |
| Dificultad | 8 |
| Control del personaje | 8 |
| Guía de usuario | 3 |
| Información proporcionada por el juego | 2 |
| Diseño visual | 7 |
| Coherencia | 8 |

| | |
|---|---|
| Sexo | Hombre |
| Edad | 19 |
| Nivel de estudios | Universitarios |
| Aficiones | Jugar videojuegos |
| **Resultados** | |
| Tareas | Puntuación |
| Jugabilidad | 9 |
| Dificultad | 5 |
| Control del personaje | 6 |
| Guía de usuario | 1 |
| Información proporcionada por el juego | 1 |
| Diseño visual | 6 |
| Coherencia | 4 |

| | |
|---|---|
| Sexo | Mujer |
| Edad | 15 |
| Nivel de estudios | Secundaria |
| Aficiones | Ver películas, series, anime, leer manga |
| **Resultados** | |
| Tareas | Puntuación |
| Jugabilidad | 7 |

| | |
|---|---|
| Dificultad | 5 |
| Control del personaje | 6 |
| Guía de usuario | 0 |
| Información proporcionada por el juego | 0 |
| Diseño visual | 8 |
| Coherencia | 5 |

## 3.6 Versiones de la Aplicación

La aplicación va a tener tres versiones:

1. Una versión web que estará publicada en ITCH.io y en Github.
2. Una versión para Windows que se podrá descargar desde ITCH.io.
3. Una versión para Mac que se podrá descargar desde ITCH.io.

# CAPÍTULO IV: PUBLICACIÓN

## 4.1 Requisitos de la instalación

| Requisitos minimos | Windows | macOS | WEB |
|---|---|---|---|
| **Sistemas operativos** | Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only. | High Sierra 10.13+ (Intel editor)<br>Big Sur 11.0 (Apple silicon Editor) | Cualquier navegador actualizado en su ultima versión, se recomienda Chrome |
| **CPU** | X64 architecture with SSE2 instruction set support | X64 architecture with SSE2 instruction set support (Intel processors)<br>Apple M1 or above (Apple silicon-based processors) | Lo mismo que los sistemas operativos |
| **Graphics API** | DX10, DX11, and DX12-capable GPUs | Metal-capable Intel and AMD GPUs | Lo mismo que los sistemas operativos |
| **Almacenamiento** | 18 MB | 18 MB | Nada |
| **Requisitos adicionales** | Hardware vendor officially supported drivers | Apple officially supported drivers (Intel processor)<br>Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor. | Internet |

## 4.2 Instrucciones de Uso

- Para empezar el juego se tiene que presionar la flecha de arriba, la tecla w o la tecla de control izquierda.
- Para mover la raqueta presiona las teclas A-D o las flechas ← →.
- Para lanzar la pelota hacia la derecha y hacia la izquierda presiona las teclas W-S o las flechas ↑↓ respectivamente cuando el juego haya empezado.
- El objetivo del juego es destruir todos los bloques sin dejar que la bola toque el piso.

## 4.3 Bugs

- La pelota rebota infinitamente en el techo sin cambiar de posición.
- La pelota puede quedare atrapada en un bucle rebotando con las esquinas del escenario.

- La pelota rebota en ángulos inesperados cuando chocha con las esquinas de los bloques destructibles.
- La música del tercer nivel aveces no carga y se queda con la música del segundo nivel.
- La velocidad de la pelota no es constante y cambia al impactar con bloques o con la raqueta.

## 4.4 Proyección a Futuro

Pienso agregar más niveles para el juego (por lo menos 5 más), agregarle nuevas mecánicas como bloques que se muevan o que requieran mas de un golpe para ser destruidos. Y luego de que el juego sea un poco mas original lanzarlo al mercado.

## 4.5 Presupuesto

Actualmente tengo 200 dólares como presupuesto y lo pienso utilizar para contratar diseñadores para que me hagan niveles con mejores aspectos visuales y variedad.

## 4.6 Análisis de Mercado

El mercado de dispositivos móviles esta lleno de juegos de este tipo aunque no sean el mismo concepto, por eso pienso lanzar el juego cuanto tenga algo característico e innovador que ofrecerle a los jugadores.
Ademas puede que la empresa que tiene los derechos del juego origina me demande y tenga problemas con la ley, si el juego se parece demasiado al de ellos.

## 4.7 Viabilidad

El juego sera viable porque se incluirán anuncios en el menú de inicio y también se ofrecerá la opción de eliminar los anuncios pagando una pequeña cantidad por el juego.
Gracias al poco capital invertido el retorno y las ganancias están casi garantizadas diría yo.

# CONCLUSIONES

El desarrollo de este videojuego fue muy divertido, he aprendido mucho sobre como funciona Unity3D y las físicas, y sobre el desarrollo de videojuegos en general en esta materia, me ha gustado la experiencia y pienso que el futuro pueda terminar desarrollando videojuegos en un gran estudio como CAPCOM.

# REFERENCIAS BIBLIOGRÁFICAS


- https://docs.unity3d.com/Manual/index.html
- https://www.youtube.com/watch?v=gB1F9G0JXOo


Enlaces de Github:

- https://jordaly.github.io/Proyecto-final-programacionVideojuegos/
- https://github.com/jordaly/Proyecto-final-programacionVideojuegos