UNIVERSIDAD TECNOLÓGICA DE SANTIAGO (UTESA)

Área de Arquitectura e Ingeniería
Carrera de Ingeniería en sistemas computacionales.

Proyecto Final

Entrega Para La Calificación Del Tercer Parcial
De La Asignatura Programación de Videojuegos.

PRESENTADO POR:

José Rodolfo Morel. 1-16-0328.
Rudelvi Valenzuela. 1-17-1005.
Adrián Grullón 1-16-1745.

ASESOR:

Ing. Iván Mendoza.

Santiago De Los Caballeros
República Dominicana
Diciembre, 2020.

INTRODUCCIÓN

CAPÍTULO I: VIDEOJUEGO Y HERRAMIENTAS DE DESARROLLO

1.1 Descripción
1.2 Motivación
  1.2.1 Originalidad de la idea
  1.2.2 Estado del Arte
1.3 Objetivo general
1.4 Objetivos específicos
1.5 Escenario
1.6 Contenidos
1.7 Metodología
1.8 Arquitectura de la aplicación
1.9 Herramientas de desarrollo.

CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN
2.1 Planificación (Diagrama de Gantt)
2.2 Diagramas y Casos de Uso

CAPÍTULO III: DESARROLLO
3.1 Capturas de la Aplicación (Documentación completa del desarrollo, Scripts, Sprites, Prefabs e imágenes)
3.2 Prototipos
3.3 Perfiles de Usuarios
3.4 Usabilidad
3.5 Test
3.6 Versiones de la Aplicación

INTRODUCCIÓN

A continuación, en este documento presentaremos los diferentes aspectos relacionados al desarrollo del proyecto propuesto, el videojuego “BallEnd”, los cuales serán planteados más adelante en los diferentes capítulos subsecuentes.

En el primer capítulo podremos observar a detalle las diferentes herramientas utilizadas en el diseño y desarrollo del juego, así como los objetivos y motivos para el desarrollo del juego propuesto, además del contenido mismo del juego y las diferentes metodologías empleadas en su desarrollo.

En el Segundo capitulo trataremos la planificación del proyecto, además presentaremos diagramas de los diferentes procesos y actividades del juego de forma que estos puedan ser contempladas de forma perspectivas, también se realizaron varios Test de las diferentes versiones y prototipos del juego los cuales se detallaran los resultados de dichas encuestas mas adelante.

CAPÍTULO I: VIDEOJUEGO Y HERRAMIENTAS DE DESARROLLO

1.1 DESCRIPCIÓN

Nuestro videojuego consiste en el desplazamiento de un objeto en este caso sería una bola la cual moveremos de un punto hacia otro mediante las teclas flecha arriba, abajo, izquierda y derecha tratando de esquivar obstáculos con el fin de llegar hacia una meta final para pasar a un siguiente nivel con más dificultad que el anterior hasta que el usuario pueda superar todos los niveles.

1.2 MOTIVACIÓN

En este videojuego el subir de nivel sería la motivación principal para el usuario llevando a este a sentir sensaciones y ansias de poder lograr el objetivo de llegar a la meta para poder seguir o preguntarse a sí mismo que habrá más adelante.

1.2.1 ORIGINALIDAD DE LA IDEA

El contacto visual del usuario hacia el videojuego contempla una idea muy única y motivacional ya que los obstáculos que el usuario debe de evadir estarán mayormente con un movimiento tanto de izquierda como derecha con el objetivo de que el usuario sea hábil y preciso a la hora de mover la bola hacia adelante.

1.2.2 ESTADO DEL ARTE

En el estado del arte pretendemos buscar que los usuarios puedan a través de nuestra estructura de diseño poder despejar o quitar el estrés siempre dejando a un lado que el videojuego no sea un aspecto vicioso para el mismo ya que eso influye a la salud mental de la persona.

1.3 OBJETIVO GENERAL

- Lograr desplazar la bola hacia la meta final, sin chocar con ningún obstáculo.

1.4 OBJETIVOS ESPECÍFICOS

- Realizarlo en el menor tiempo posible.
- Alcanzar la meta en el menor número de intentos.
- Recolectar todos los coleccionables.
- Descubrir el patrón de algunos niveles.
- Superar todos los niveles del juego
- Desbloquear nuevos objetos

1.5 ESCENARIO

En este video juego se utilizará un plano en 2 dimensiones, donde los usuarios podrán desplazarse de izquierda a derecha y de arriba hacia abajo.

Todos los objetos estarán limitados a una superficie (plana) de 40 cm, la cual contendrá: bola, obstáculos, coleccionable, línea de meta y cuadro de juego.

El escenario también incluirá información instructiva de las acciones de las acciones a realizar dentro del juego, así como otras de interés para el usuario, utilizando para esto los ‘Text’ y otros componentes.

Este escenario será minimalista, donde a través de la intuición los usuarios podrían guiarse para lograr los objetivos anteriormente expuestos.

1.6 CONTENIDOS

- Música
  - Inicio
  - transcurso(timer)
  - obtención de coleccionable
  - colisión con obstáculo
  - Fin
- Obstáculos
- Pelota(jugador)
- Paredes
- Línea de meta
- Coleccionable
  - Diferente valor(puntos)
- Timer
- Diferentes niveles
- Puntuación alcanzada

1.7 METODOLOGÍA

A continuación, se explica cómo es la metodología que se usa habitualmente en el desarrollo de videojuegos: cuáles son sus fases, cuáles son los miembros del equipo y los métodos de gestión del desarrollo que se suelen adoptar.

Estas metodologías están pensadas para potenciar la velocidad y la calidad de los desarrollos, por lo que su correcta implementación resulta fundamental a la hora del desarrollo de videojuegos dado que nos permite evaluar que nuestro juego se haya construido de acuerdo a las necesidades y requerimientos especificados previamente.

La metodología empleada para el desarrollo de videojuegos consta de 6 fases:

Fase 1: Selección De Las Tecnologías Empleadas Al Desarrollo

En este primer paso debemos hacer un estudio de las diferentes herramientas y plataformas de desarrollo de videojuegos disponibles en el mercado y que mejor se ajusten al tipo de videojuegos que queremos desarrollar, que dependiendo la temática y mecánicas del juego a desarrollar resultan más factibles algunas herramientas de desarrollo sobre otras, es importante tener en cuenta los aspectos más destacables de cada herramientas y cuáles serían los más adecuados para el proyecto que desarrollaremos.

Fase 2: Formación

Después de haber seleccionado la herramienta de desarrollo más adecuada para nuestro juego en la fase anterior, en nuestro caso **Unity** es necesario contar con los conocimientos para el manejo de dichas herramientas, por lo que resulta muy útil realizar cursos y tutoriales sobre el manejo de las últimas versiones de estas herramientas.

Fase 3: Definición del videojuego

En esta etapa he de plasmar el primer borrador de nuestro juego donde definiremos aquellos aspectos más básicos de nuestro juego, como el nombre, el objetivo, los elementos que intervienen en el juego los escenarios que conforman el juego, los guiones que seguirán los personajes dentro de la trama o historia si el juego contiene alguna, así como la forma y estilo de los personajes.

Fase 4: Desarrollo del videojuego

Esta fase en el trabajo principal, se basa en el uso de las herramientas seleccionadas para diseñar y desarrollar el juego propuesto guiándose de los borradores y especificaciones definidas anteriormente, en esta fase se procede a crear los elementos gráficos que compondrán el juego además del programa la lógica y reglas del juego.

Fase 5: Test con usuarios

En esta fase se procede a realizar pruebas al primer prototipo del juego, estas pruebas deben ser realizadas preferiblemente por un equipo externo al desarrollo del juego, para esto es muy útil contar con comunidades de jugadores que realicen las pruebas betas del juego y den sus comentarios al equipo de desarrollo para poder ir corrigiendo los bugs, una de las mejores comunidades para poder publicar un juego beta en una comunidad para que pueda ser testeado.

Fase 6: Publicación

Finalmente, la última etapa es la publicación del juego, el juego puede ser publicado en una o varias plataformas específicas como consolas de video juegos o aplicaciones web o de escritorio, también es importante tener en cuenta la página donde se publicará el juego para su posterior acceso.

1.8 ARQUITECTURA DE LA APLICACIÓN.

Esta aplicación estará disponible como una aplicación web ejecutable desde el navegador y una versión ejecutable para Windows.

Los requisitos mínimos para ejecutar este juego son:

CPU	Intel Core i3-3210 3.2 GHz/ AMD A8-7600 APU 3.1 GHz o equivalente
GPU (integrado)	Intel HD Graphics 4000 (Ivy Bridge) o serie AMD Radeon R5 (Kaveri line) con OpenGL 4.4*
GPU (discreta)	Serie Nvidia GeForce 400 o serie AMD Radeon HD 7000 con OpenGL 4.4
RAM	4 GB
HDD	Como mínimo 1 GB para núcleo del juego, mapas y otros archivos

1.9 HERRAMIENTAS DE DESARROLLO.

Las herramientas empleadas para el desarrollo de este video juego son:

Unity 2019.1 como herramienta de desarrollo de videojuegos.

Visual Studio Code como editor de código

git, como gestor de código y versiones

itch.io como página de publicidad y marketing.

Lenguaje de programación C#

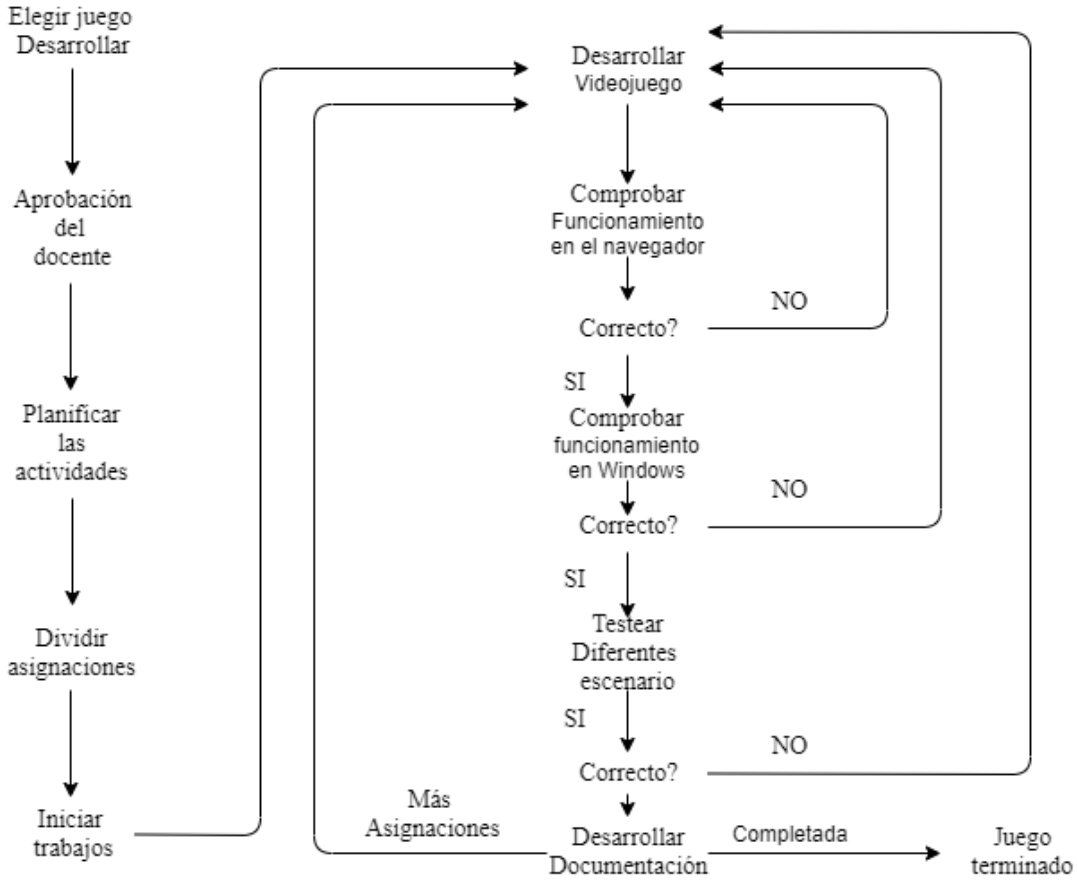
GITHUB como almacén de repositorios remotos.

CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN

2.1 PLANIFICACIÓN (DIAGRAMA DE GANTT)

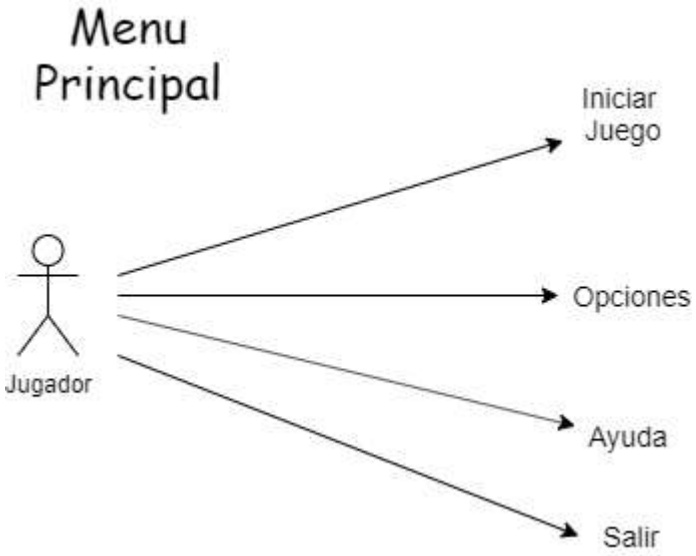
Ver Documento adjunto: “Cronograma.xlsx”.

2.2 DIAGRAMA DE LA ESTRUCTURA DEL PROYECTO

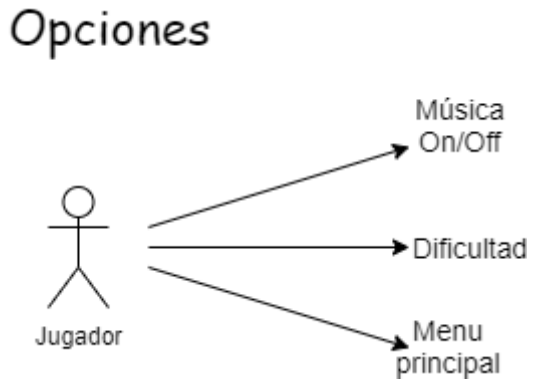


CASOS DE USO

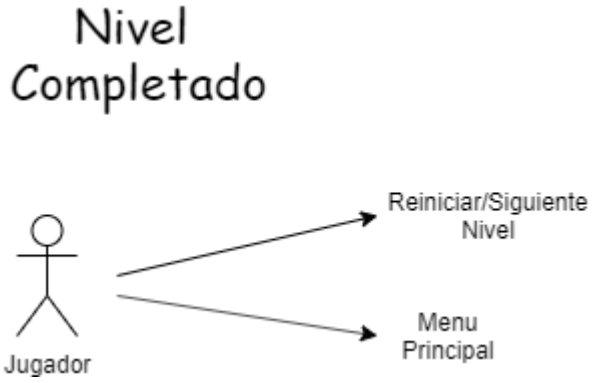
- **Menú principal**



- **Opciones**



- **Nivel Completado**



- **Dentro del juego**

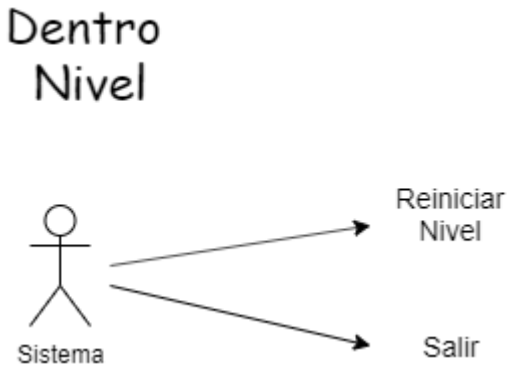
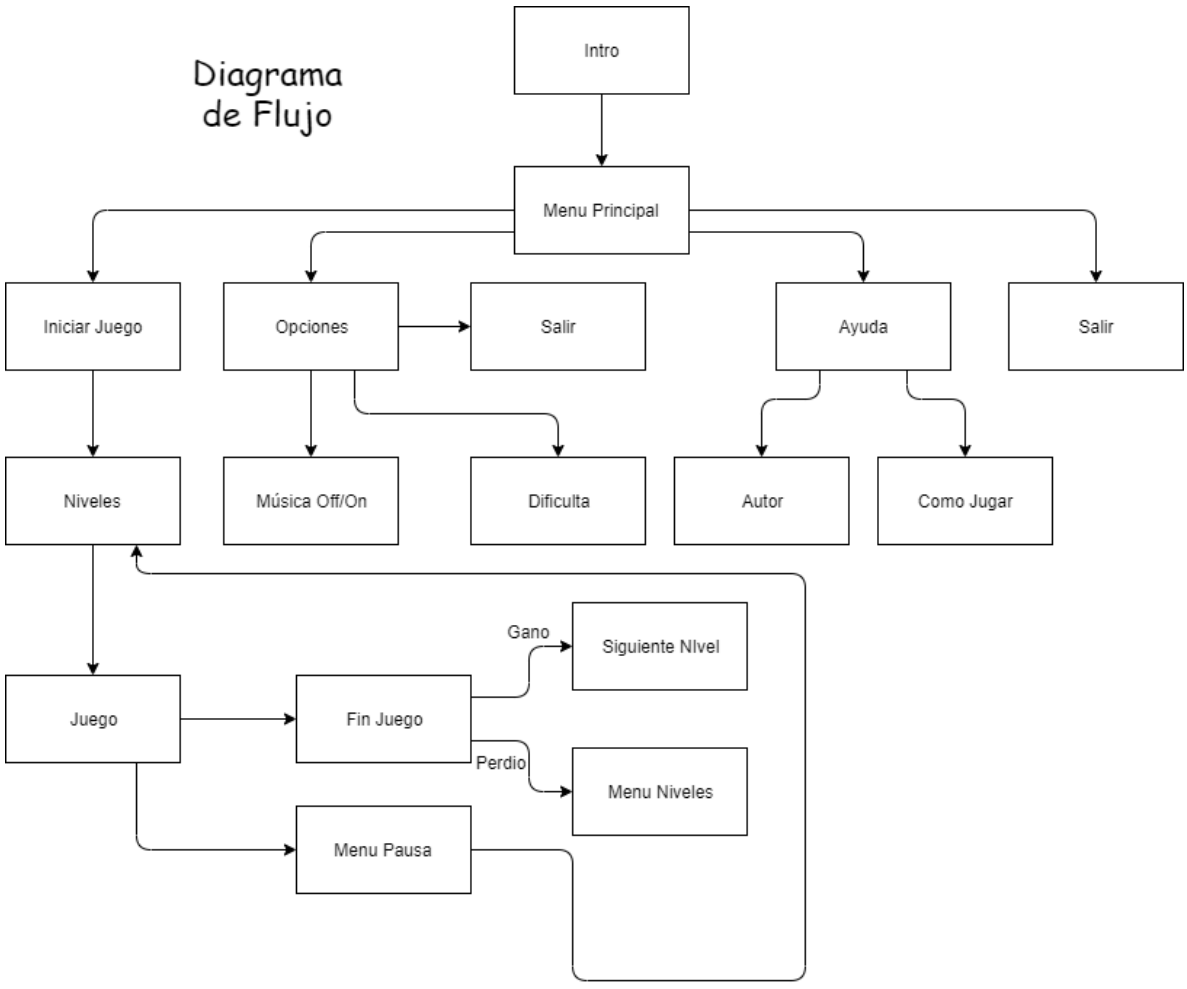


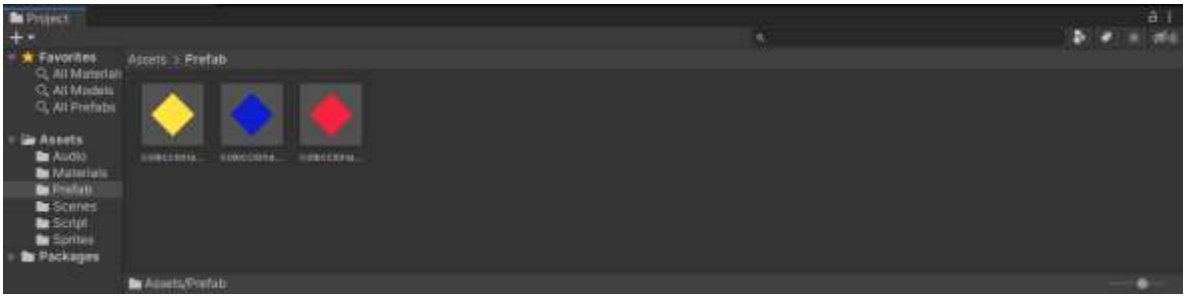
DIAGRAMA DE FLUJO DEL JUEGO

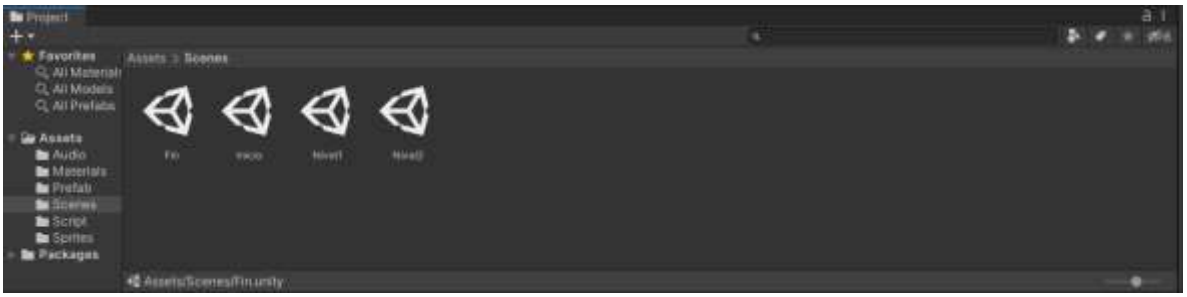


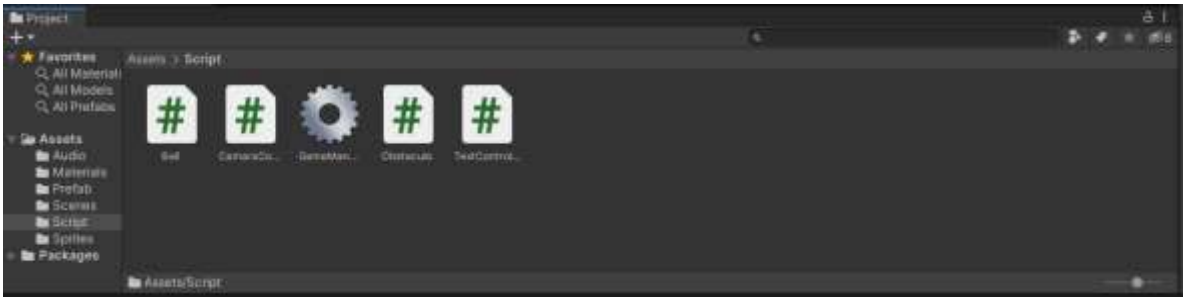
CAPÍTULO III: DESARROLLO

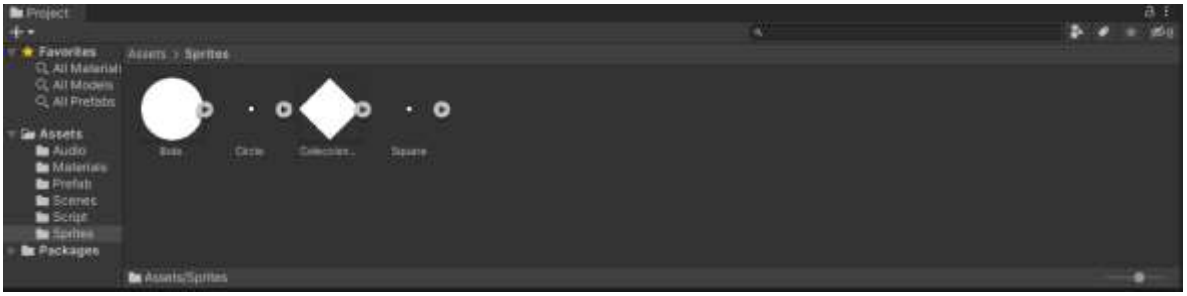
3.1 CAPTURAS DE LA APLICACIÓN

Proyect Windows



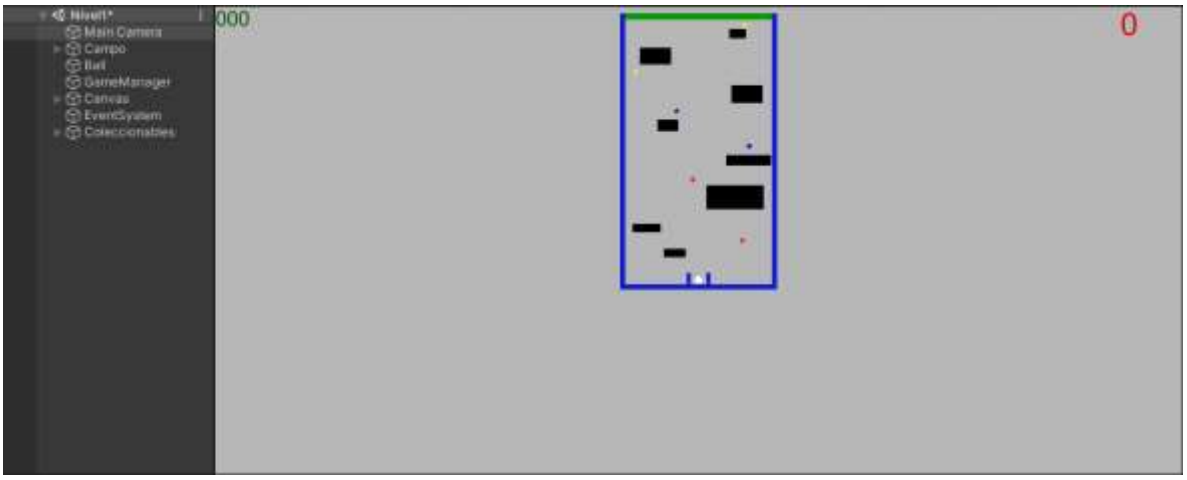

Project
Favorites
All Materials
All Models
All Prefabs
Assets
Audio
Materials
Prefab
Scenes
Script
Sprites
Packages
Assets > Scenes
Assets/Scenes/Fin.unity

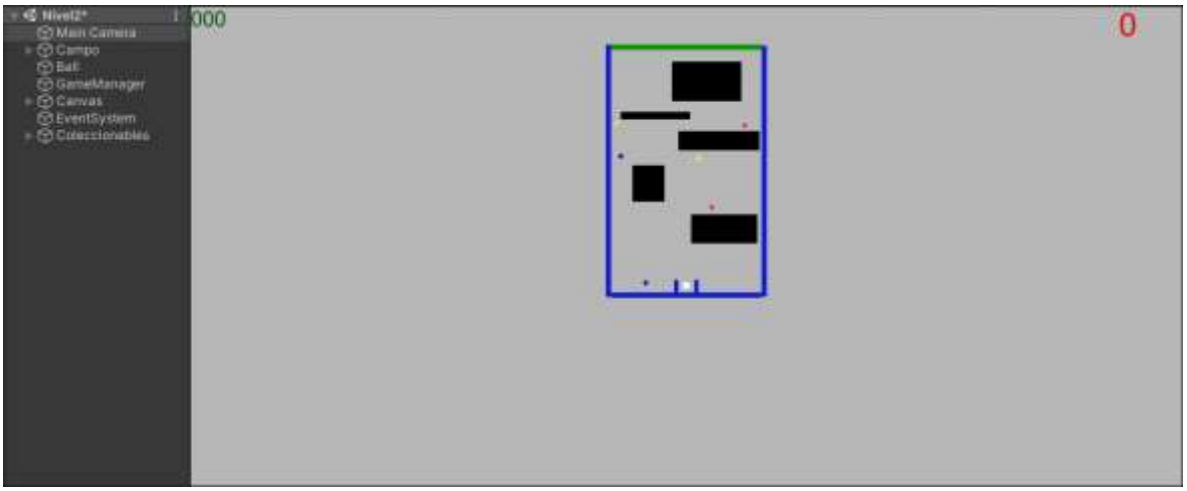

Project
Favorites
All Materials
All Models
All Prefabs
Assets
Audio
Materials
Prefab
Scenes
Script
Sprites
Packages
Assets > Script
Assets/Script


Project
Favorites
All Materials
All Models
All Prefabs
Assets
Audio
Materials
Prefab
Scenes
Script
Sprites
Packages
Assets > Sprites
Assets/Sprites

Scenes


Inicio
Main Camera
Cerbo
Canvas
EventSystem
GameManager
BallEnd
Utilizar cursor para desplazar Bola
Haz click para Jugar
o pulsa Esc para salir.
Created By:
José Rodolfo Morel, 1-18-0328.
Rudelvi Valenzuela, 1-17-1005.
Adrián Grullón, 1-16-1745.


Nivel1*
Main Camera
Campo
Ball
GameManager
Canvas
EventSystem
Coleccionables
000
0


Nivel2*
Main Camera
Campo
Ball
GameManager
Canvas
EventSystem
Coleccionables
000
0


Fin
Main Camera
Centro
Canvas
EventSystem
GameManager
BallEnd
Created By:
Rudelvi Valenzuela, 1-17-1005.
Adrian Grullon, 1-16-1745.
Utilizar cursor para desplazar Bola
Haz click para Jugar
o pulsa Esc para salir.

Script

Ball

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


public class Ball : MonoBehaviour
{
    public float velocidad= 30.0f;
    public float puntajeAcumulado = 0.0f;
    public bool finJuego = false;
    public bool gano = false;
    public Text textGanador;
    public Text textPerdedor;
    public Text textInstrucciones;
    public Text textPuntaje;
    public Text textTemporizador;
    public Text textPuntajeFinal;
    public float timeRemaining = 10;
    public float defaultBallPosicion=0;

    AudioSource fuenteDeAudio;

    public AudioClip audioInicio, audioPerdedor, audioGanador, audioPuntos;

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

        fuenteDeAudio.clip = audioInicio;
        fuenteDeAudio.Play();
    }

    void FixedUpdate()
    {
         float v = 0;
         float h = 0;

        if (!finJuego){

            v = Input.GetAxisRaw("Vertical");
            h = Input.GetAxisRaw("Horizontal");
```

```
        }

        GetComponent<Rigidbody2D>().velocity = new Vector2(h*velocidad , v *
 velocidad);


    }

     void Update()
        {
            if (timeRemaining > 0 && finJuego==false)
            {
                timeRemaining -= Time.deltaTime;
                textTemporizador.text = "Tiempo restante: "+System.Math.Roun
d(timeRemaining,0) + "";

            }else if(finJuego!=true)
            {
                textPerdedor.text = "HAS PERDIDO!";
                textInstrucciones.gameObject.SetActive(true);

                fuenteDeAudio = GetComponent<AudioSource>();
                fuenteDeAudio.clip = audioPerdedor;
                fuenteDeAudio.Play();

                finJuego = true ;
                transform.position = new Vector2(0 , defaultBallPosicion);
            }
        }

     void OnCollisionEnter2D(Collision2D micolision){

        if (micolision.gameObject.tag == "obstaculo")
        {


            textPerdedor.text = "HAS PERDIDO!" ;

            textInstrucciones.gameObject.SetActive(true);
            fuenteDeAudio = GetComponent<AudioSource>();
             fuenteDeAudio.clip = audioPerdedor;
             fuenteDeAudio.Play();
             finJuego = true ;
             transform.position = new Vector2(0 , defaultBallPosicion);
            // this.gameObject.SetActive (false);
```

```
        }else if(micolision.gameObject.tag == "objetivo"){



            textGanador.text = "HAS GANADO!" ;
            fuenteDeAudio.clip = audioGanador;
            fuenteDeAudio.Play();
           /// this.gameObject.SetActive (false);
           finJuego =  true;
           gano = true;
            textInstrucciones.gameObject.SetActive(true);
            textPuntajeFinal.gameObject.SetActive(true);

            textPuntajeFinal.text = "Puntaje Final: "+puntajeAcumulado+"/100
";


            transform.position = new Vector2(0 , defaultBallPosicion);
        }else if( micolision.gameObject.tag == "coleccionableA"){
            fuenteDeAudio = GetComponent<AudioSource>();
            fuenteDeAudio.clip = audioPuntos;
            fuenteDeAudio.Play();

           puntajeAcumulado = puntajeAcumulado + 25;
           textPuntaje.text = puntajeAcumulado + "";
           micolision.gameObject.SetActive(false);

        }else if( micolision.gameObject.tag == "coleccionableB"){

            fuenteDeAudio = GetComponent<AudioSource>();
            fuenteDeAudio.clip = audioPuntos;
            fuenteDeAudio.Play();

           puntajeAcumulado = puntajeAcumulado + 20;
           textPuntaje.text = puntajeAcumulado + "";

            micolision.gameObject.SetActive(false);

        }else if( micolision.gameObject.tag == "coleccionableR"){

            fuenteDeAudio = GetComponent<AudioSource>();
            fuenteDeAudio.clip = audioPuntos;
            fuenteDeAudio.Play();

           puntajeAcumulado = puntajeAcumulado + 5;
```

```
            textPuntaje.text = puntajeAcumulado + "";
             micolision.gameObject.SetActive(false);

        }


    }



}
```

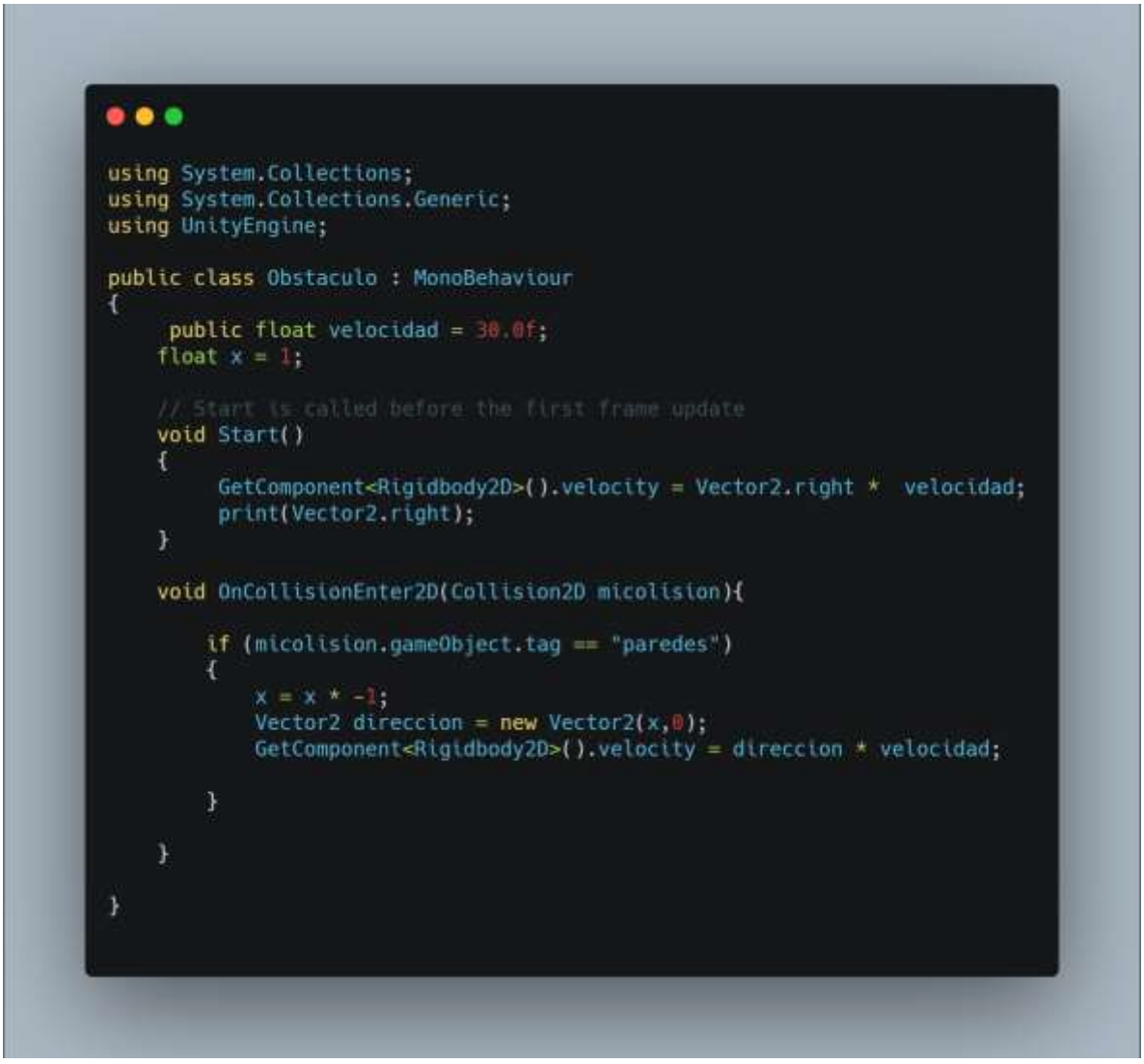
CamaraController

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

public class CamaraController : MonoBehaviour
{
    //Referencia a nuestro jugador
    public GameObject jugador;
    //Para registrar la diferencia entre la posicion de la cámara y la del jugador

    private Vector3 offset;
    // Use this for initialization
    void Start ()
    {
        //diferencia entre la posicion de la cámara y la del jugador
        offset = transform.position - jugador.transform.position;
    }

// Se ejecuta cada frame, pero después de haber procesado todo. Es más exacto para la cámara
    void LateUpdate ()
    {
        //Actualizo la posición de la cámara
        transform.position = jugador.transform.position + offset;
    }
}
```

GameManager

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

public class GameManager : MonoBehaviour
{
    public string nivel;
    public string siguienteNivel;
    public Ball ball;
    // Start is called before the first frame update

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButton(0)){

            if(nivel!="Inicio"){

                if(ball.finJuego){

                    if(ball.gano){
                        SceneManager.LoadScene(siguienteNivel);
                    }else
                    {
                        SceneManager.LoadScene(nivel);
                    }

                }
            }else
            {
                SceneManager.LoadScene(siguienteNivel);
            }

        }

        if(Input.GetKeyDown(KeyCode.R)){
            if(ball.finJuego)
            {
                SceneManager.LoadScene(nivel);
            }

        }

        if(Input.GetKeyDown(KeyCode.M)){
            SceneManager.LoadScene("Inicio");
        }

        if(Input.GetKeyDown(KeyCode.Escape)){
            Application.Quit();
            Debug.Log("salio del juego");
        }

    }
}
```

Obstáculo



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

public class Obstaculo : MonoBehaviour
{
     public float velocidad = 30.0f;
    float x = 1;

    // Start is called before the first frame update
    void Start()
    {
         GetComponent<Rigidbody2D>().velocity = Vector2.right *  velocidad;
         print(Vector2.right);
    }

    void OnCollisionEnter2D(Collision2D micolision){

        if (micolision.gameObject.tag == "paredes")
        {
            x = x * -1;
            Vector2 direccion = new Vector2(x,0);
            GetComponent<Rigidbody2D>().velocity = direccion * velocidad;

        }

    }

}
```

TextController

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

public class TextController : MonoBehaviour
{
    public Text creditos;
    public Text titulo;
    public Text instrucciones;
    public float defaultCredictPosition = -81.0f;
    public float defaultIntruccionesPosition = -81.0f;
    public float defaultTittlePosition = -81.0f;

    // Start is called before the first frame update
    void Start()
    {
        transform.position = new Vector3(460 , defaultCredictPosition,0);
        titulo.transform.position = new Vector3(460 , defaultTittlePosition,0);
        instrucciones.transform.position = new Vector3(460 , defaultIntruccionesPosition,0);

    }

    // Update is called once per frame
    void FixedUpdate()
    {
        if(defaultCredictPosition<500){
            transform.position = new Vector3(460 , defaultCredictPosition,0);
            defaultCredictPosition = defaultCredictPosition+0.5f;

        }

        if(defaultCredictPosition>=130 && defaultTittlePosition<200){
            titulo.transform.position = new Vector3(460 , defaultTittlePosition,0);
            defaultTittlePosition = defaultTittlePosition+0.5f;

        }

        if(defaultTittlePosition>=130 && defaultIntruccionesPosition<165){
            instrucciones.transform.position = new Vector3(460 , defaultIntruccionesPosition,0);
            defaultIntruccionesPosition = defaultIntruccionesPosition+0.5f;

        }

    }
}
```

3.2 PROTOTIPOS

Los prototipos son versiones preliminares del juego cuyo objetivo es tener una idea previa del acabado final del juego, así como sus aspectos de jugabilidad y estabilidad.

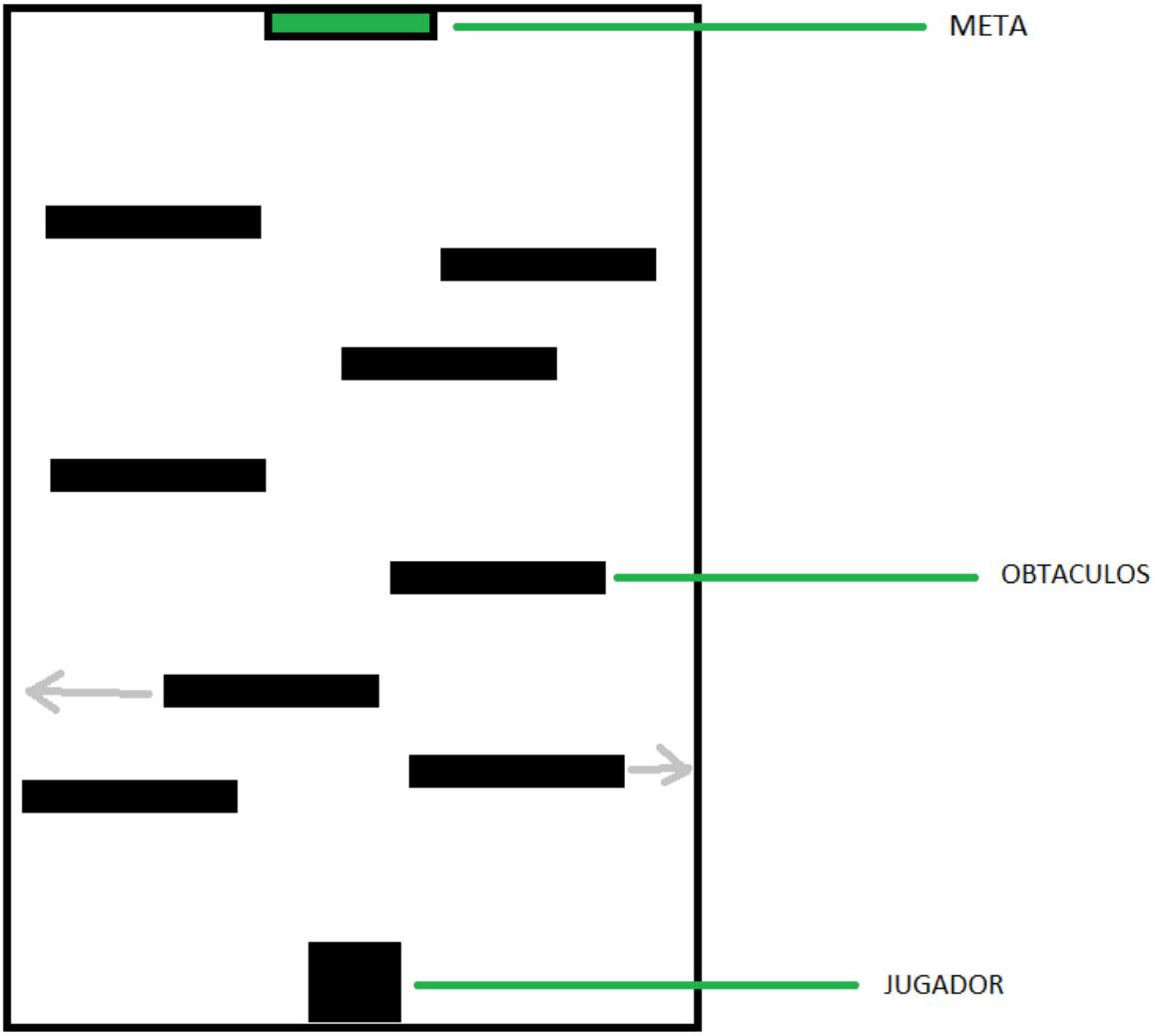
A lo largo de este proyecto se realizarán diferentes prototipos del juego donde se irán incorporando las diferentes funcionalidades del juego de forma incremental, esto ayudara a poder evaluar diferentes aspectos del juego sin necesidad de esperar la versión final del mismo.

Los prototipos que serán desarrollados en las diferentes etapas del juego pueden dividirse en 2 tipos **LO-FI** o de baja fidelidad, estos son borradores de muy baja calidad que solo sirven como un mockup o platilla demostrativa de algunos aspectos visuales o funcionales del juego, por ejemplo, un dibujo a papel.

Luego están los **HI-FI** o de alta fidelidad son prototipos funcionales del juego que cuentan con una o varias de las características finales del juego, estas son prácticamente versiones del juego que aun contienen errores, al refinar estos prototipos es que tendremos nuestro juego final.

Dentro de los prototipos de Baja calidad (**LO-FI**) se platea el diseño de los siguientes prototipos:

Primer prototipo: una imagen donde se visualizar la estructura general del juego, así como el escenario y el ambiente donde interactúa el usuario.



Segundo Prototipo: En el segundo prototipo se comienza a desarrollar la idea en el ambiente propuesto, de diseñan y se programan los aspectos mas fundamentales del juego como el jugador y su movilidad.

Tercer Prototipo: En el tercer prototipo se agregan los obstáculos y la meta junto con sus propiedades fiscas de movimiento, además se programa al jugador para que pierda al colisionar con algunos de los obstáculos y ganar al colisionar con la meta.

Cuarto Prototipo: Aquí se agregan los efectos de sonido al juego además de las etiquetas con información al ganar o perder la partida.

Quito prototipo: se agrega el menú principal con las opciones de inicio del juego.

Dentro de los prototipos de **HI-FI** tenemos los prototipos de la sexta generación en adelante.

Sexto prototipo: se desarrollan prelab y platillas de los diferentes elementos a agregar como obstáculos y coleccionables para facilitar la producción de múltiples niveles en el juego.

Séptimo Prototipo: Se implementa el uso de un Temporizador y el movimiento de la cámara en conjunto con el jugador.

3.3 PERFILES DE USUARIO

El público objetivo del juego a desarrollar comprende en su mayoría a:

- Personas de cualquier edad, ya que es un juego para todo público, sin sexo, ni violencia.
- Personas sin mucha experiencia en el mundo de los videojuegos.
- Personas con o sin formación académico ya que es un juego intuitivo que no requiere mayor interpretación.
- Personas que les justen los juegos de agilidad mental.

3.5 TEST

Para realizar el test de los diferentes prototipos del juego se tomarán en cuenta las opiniones de varios usuarios mediante la evaluación de ciertos aspectos definidos del juego.

Los usuarios deberán evaluar 7 aspectos referentes a las características más importantes del juego entre ellos tenemos: La jugabilidad, la dificultad de los niveles, el control del personaje, la guía del usuario, información brindad al usuario, diseño visual y coherencia.

Para realizar la evaluación del juego se le presentara un formulario a varios usuarios los cuales deberán evaluar cada uno de los aspectos anteriormente mencionados del juego con una puntuación entre 1 y 5 puntos, donde uno no es favorable y 5 es muy de acuerdo.

Formularios:

TEST 1

INDIVIDUO 1	
Sexo	HOMBRE
Edad	18
Nivel de Educación	BACHILLER
Aficiones	VIDEO JUEGOS

RESULTADOS	
TAREA	PUNTUACIÓN
La jugabilidad	5
La dificultad de los niveles	3
El control del personaje	5
La guía del usuario	4
Información brindada al usuario	4
Diseño visual	2
Coherencia.	5

INDIVIDUO 2	
Sexo	HOMBRE
Edad	22
Nivel de Educación	UNIVERSITARIO
Aficiones	SERIES DE TELEVISION

RESULTADOS	
TAREA	PUNTUACIÓN
La jugabilidad	4
La dificultad de los niveles	4
El control del personaje	4
La guía del usuario	3
Información brindada al usuario	3
Diseño visual	3
Coherencia.	4

INDIVIDUO 3	
Sexo	MUJER
Edad	27
Nivel de Educación	UNIVERSITARIO
Aficiones	TELENOVELAS

RESULTADOS	
TAREA	PUNTUACIÓN
La jugabilidad	5
La dificultad de los niveles	4
El control del personaje	5
La guía del usuario	3
Información brindada al usuario	4
Diseño visual	3
Coherencia.	5

INDIVIDUO 4	
Sexo	HOMBRE
Edad	9
Nivel de Educación	PRIMARIA
Aficiones	JUEGOS Y CARICATURAS

RESULTADOS	
TAREA	PUNTUACIÓN
La jugabilidad	5
La dificultad de los niveles	5
El control del personaje	4
La guía del usuario	2
Información brindada al usuario	3
Diseño visual	4
Coherencia.	5

RESULTADO FINAL	
TAREA	PUNTUACIÓN
La jugabilidad	4.75
La dificultad de los niveles	4
El control del personaje	4.5
La guía del usuario	3
Información brindada al usuario	3.5
Diseño visual	3
Coherencia.	4.75

3.6 Versiones de la aplicación

Se llama control de versiones a la gestión de cambios efectuados en un documento, programa, imagen, website y otros archivos que contengan información. Los cambios se registran de forma automática y pueden ser identificados mediante números o combinaciones alfanuméricas.

Para nuestro proyecto utilizaremos el formato **Versiones X.Y.Z**

Un método bastante habitual de numerar las versiones es utilizando dos o tres cifras decimales para indicar la importancia de los cambios realizados. El cambio de la primera cifra indica cambios más importantes que el de la segunda. El criterio más habitual es seguir las siguientes normas:

• La primera cifra (X) indica la versión mayor del proyecto. Si empieza con un cero significa que el videojuego aún no está listo o no cumple con los requerimientos mínimos. Cada cambio en esta cifra denota una reescritura o la incompatibilidad con versiones mayores anteriores.

• La segunda cifra (Y) indica la versión menor del proyecto. Denota cambios en el contenido o en la funcionalidad del videojuego, pero no lo suficientemente importantes como para decir que ya no es el mismo. Cuando se estrena una

versión mayor se deja la versión menor a cero, pero aun así se incluye de modo que la segunda versión mayor sería la 2.0

• La tercera cifra (Z) indica la segunda versión menor. Indica que el videojuego se ha corregido pero que no se ha añadido ni eliminado nada relevante. Cuando se estrena una versión menor, es decir, cuando la segunda versión menor es igual a cero; suele omitirse.

Bugs

Los bugs, esos fallos en los videojuegos que muchas partidas han fastidiado. Desde bugs menores como una textura atravesando otra que no debe o un funcionamiento incorrecto de algunos materiales, pasando por algunas anomalías que provocan reiniciar la partida, hasta los errores críticos que provocan las pérdidas de datos o la imposibilidad de continuar la aventura pese a que trates de buscar soluciones desesperadamente.

En esta generación, a diferencia de las anteriores, está todo repleto de bugs y parches tratando de corregir errores de todo tipo, y desde que empezó, se ha vuelto costumbre el sacar juegos incompletos y luego corregir errores menores con parches que requieren conexión a Internet, ya que el temor de pasadas generaciones a sacar un producto defectuoso se ha desvanecido y se preocupan poco por errores insignificantes.

En nuestro caso los bugs se trabajarán mediante informes que los usuarios puedan expresar mediante comentarios, también a través de test que se harán en el transcurso del proyecto para poder corregir esos errores que a veces no estamos conscientes en que existen para así brindarles un videojuego confiable a través de las versiones recientes.

REFERENCIAS

Link del repositorio del prototipo del juego en GitHub:
https://github.com/Rudelvi/BallEnd.git

Link del repositorio de la documentación en GitHub:
https://github.com/RodolfoMH/Proyecto-Final.git

Link del juego en ITCH.IO:

https://rodolfomh.itch.io/ballend