UNIVERSIDAD TECNOLÓGICA DE SANTIAGO
(UTESA)
SISTEMA CORPORATIVO

Facultad de Arquitectura e Ingeniería
Carrera de Ingeniería en Sistemas Computacionales

Programación de Videojuegos
Proyecto final.

Profesora:
Ivan Mendoza

Presentado Por:

Sadiel Henríquez	1-17-0903
Jeury Payano	2-15-0853
Randiel Arias	1-17-2703

Santiago de los Caballeros
República Dominicana
Diciembre 16, 2021

INTRODUCCIÓN

CAPÍTULO I: VIDEOJUEGO Y HERRAMIENTAS DE DESARROLLO

CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN

CAPÍTULO III: DESARROLLO

CAPÍTULO IV: PUBLICACIÓN

CONCLUSIONES, REFERENCIAS BIBLIOGRÁFICAS

1.1 Descripción

Este es un juego de aventura en el que Dino tendrá que pasar diferentes mundos con diferentes ambientes. Estos mundos finalizan en el momento en que Dino logra recolectar la cantidad suficiente de monedas para poder abrir la puerta hacia el siguiente mundo.

Cada mundo que va atravesando encuentra distintos enemigos y se dificulta más el alcance de las monedas. Ayuda a Dino a recolectar todas las monedas y a sobrevivir a todos los mundos por los que tiene que pasar.

1.2 Motivación

1.2.1 Originalidad de la idea

La idea surge de darnos cuenta que los dinosaurios son criaturas muy misteriosas y llamativas que les gusta mucho a las personas y por eso hemos decidido que nuestro personaje principal sea un dinosaurio, en concreto un tiranosaurio rex que es posiblemente el más conocido de todos.

Con esto podremos lograr tener un alcance mayor a las personas y niños que son el tipo de público al cual queremos llegar.

1.2.2 Estado del Arte

Este es un juego de aventuras parecido al de Mario Bros, en el que Dino atravesará distintos entornos recolectando monedas y pasando de nivel. El reto es sobrevivir a los ataques de diferentes enemigos y obstáculos. Es un juego muy intuitivo y fácil de entender por lo que no tendremos problemas para adaptarnos a él.

1.3 Objetivo general

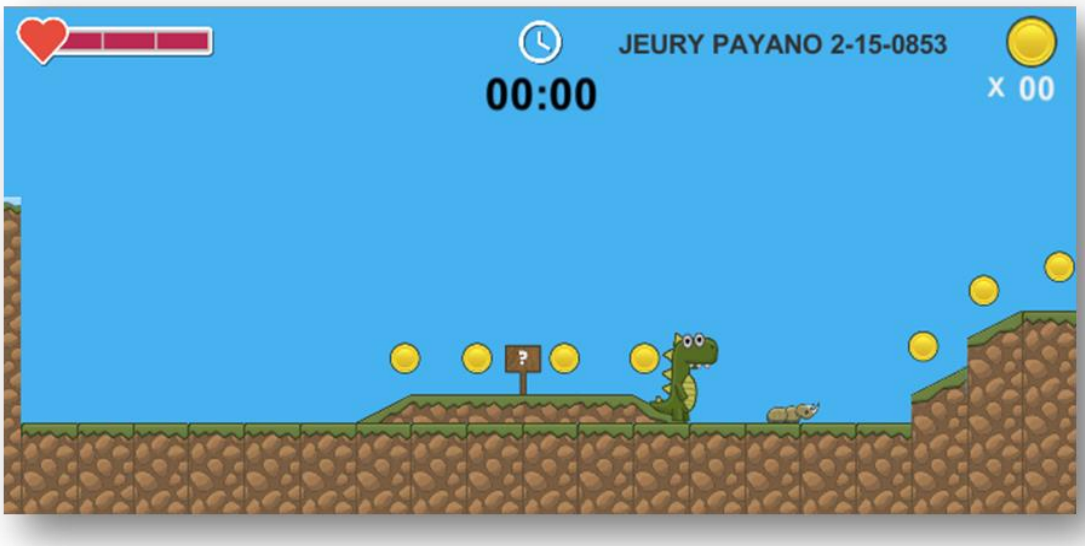
Desarrollar un videojuego 2D para desktop y móvil divertido en el que se tengan desafíos en base a esquivar enemigos y las trampas que tengan cada nivel, recolectando todas las monedas requeridas por el nivel en el menor tiempo posible.

1.4 Objetivos específicos

1. Creación de un videojuego apto para todo tipo de público.
2. Poder jugar en las plataformas móvil y web.
3. Monetización del videojuego por medio de bloqueo de niveles y habilidades y streaming.
4. Videojuego tipo plataforma divertido y emocionante.

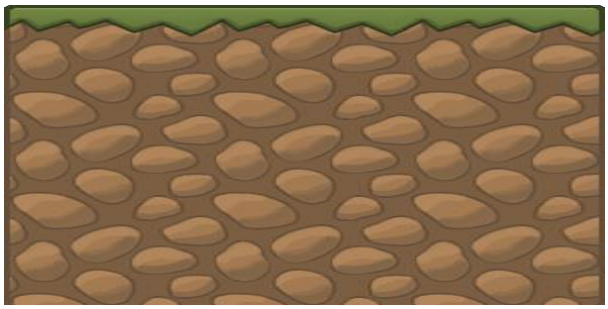
1.5 Escenario

El videojuego poseerá escenarios divertidos que tendrán distintos obstáculos, plataformas móviles y abismos en los cuales si caes pierdes, el personaje principal podrá saltarlos y esquivarlos o utilizar las plataformas como medio para llegar a otras monedas que falten por completar.



1.6 Contenidos

Terreno:



enemigos:

Personaje:

Dino

Prefab de moneda:

Control de vida del personaje:

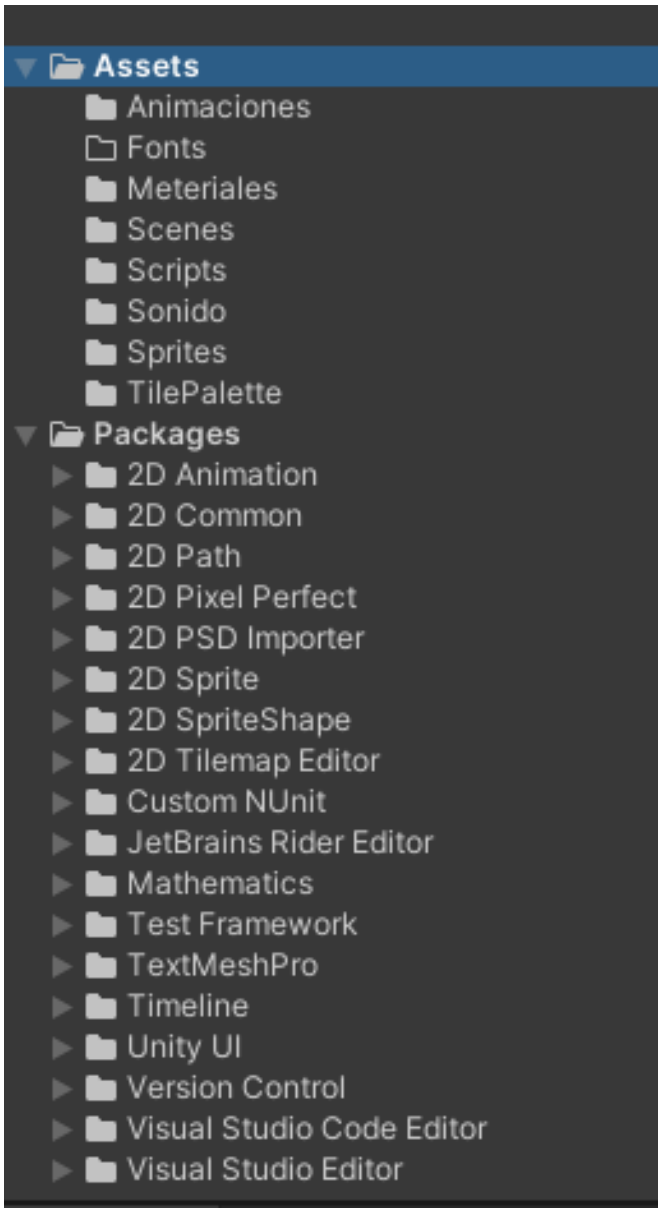
Temporizador:

Plataformas:

1.7 Metodología

El juego será desarrollado utilizando los assets que ofrece el blog de Curso de programación de videojuegos en Unity 2019 – Plataformas 2D más todas las herramientas para desarrollo que nos ofrece unity donde como grupo nos dividiremos roles de programación y diseño de mundos y personajes, así como la búsqueda de sonidos adecuados que se adapten al entorno.

1.8 Arquitectura de la aplicación



Estos vienen siendo todos los Sprite usados para el juego que fueron descargados desde una página que tenía esos recursos disponibles.

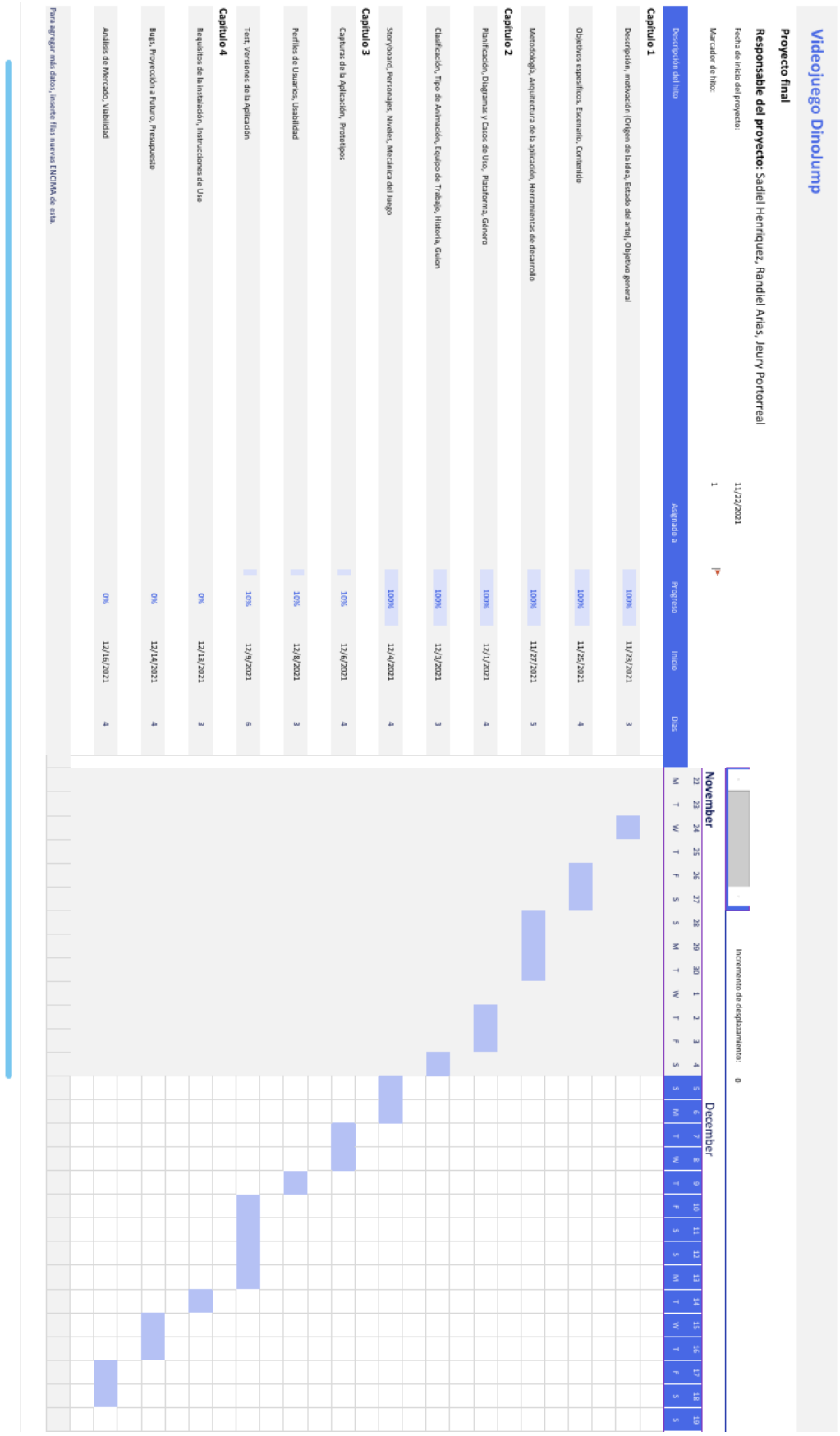
1.9 Herramientas de desarrollo

- **Unity:** utilizamos este motor de videojuegos debido a que es el motor de videojuego adecuado para crear este videojuego multiplataforma (celular y computadora), conocemos la estructura de manejo de Unity y utilizaremos un asset de este para la creación del juego.

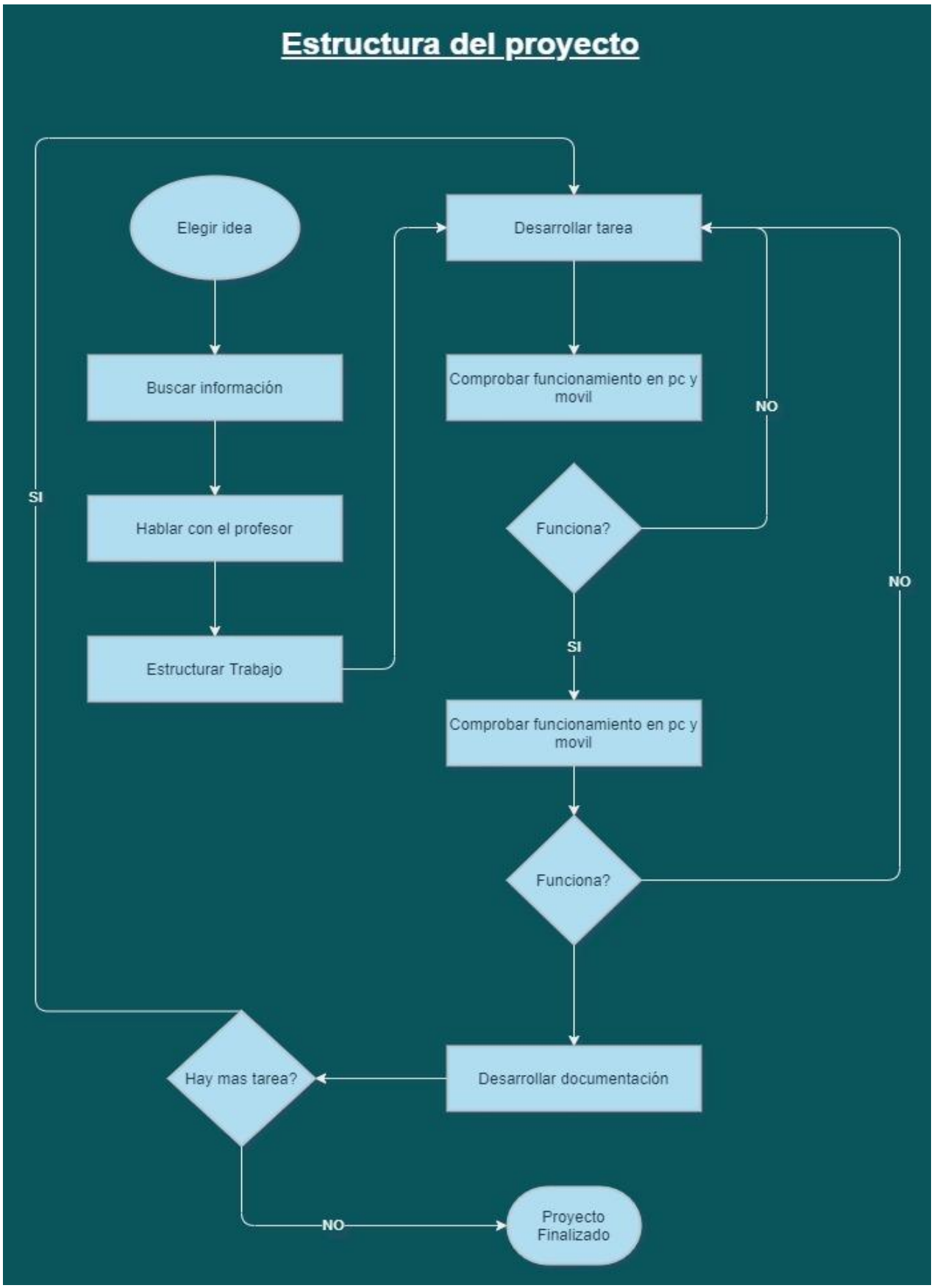
- **Git:** Para controlar y gestionar las versiones que tendrá nuestro proyecto, los cambios, mejoras, arreglos de bugs y versiones de acuerdo a etapas.

GitHub: Para alojar el proyecto en un repositorio remoto y asegurar los cambios que se vayan realizando.

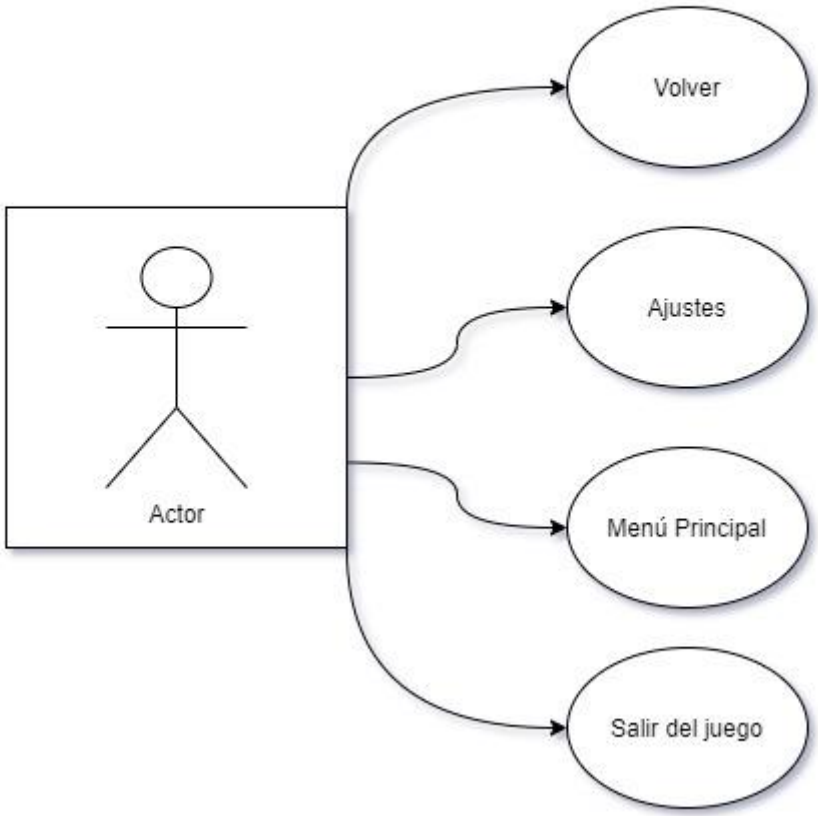
Diagrama de gantt



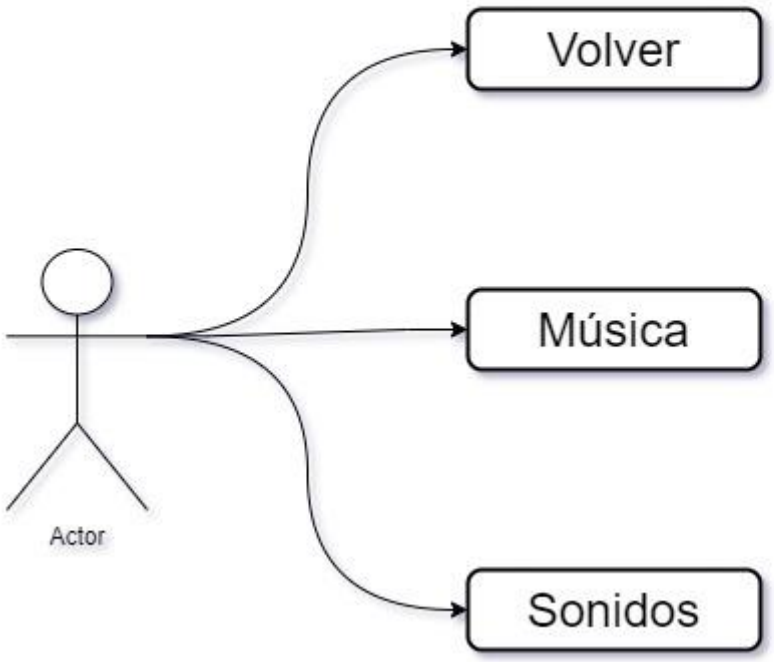
Diagramas de caso de Uso



Menú principal y opciones de niveles



Menú de ajustes



Menú principal

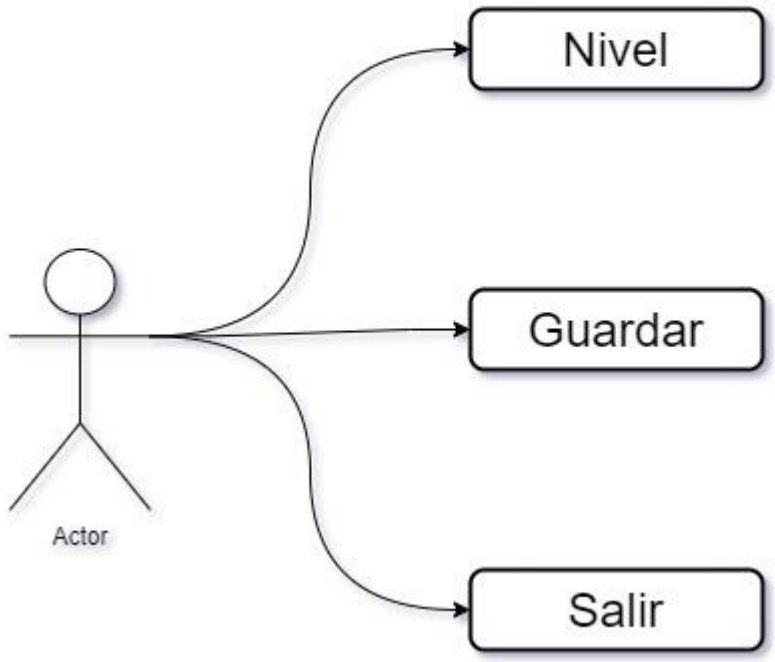
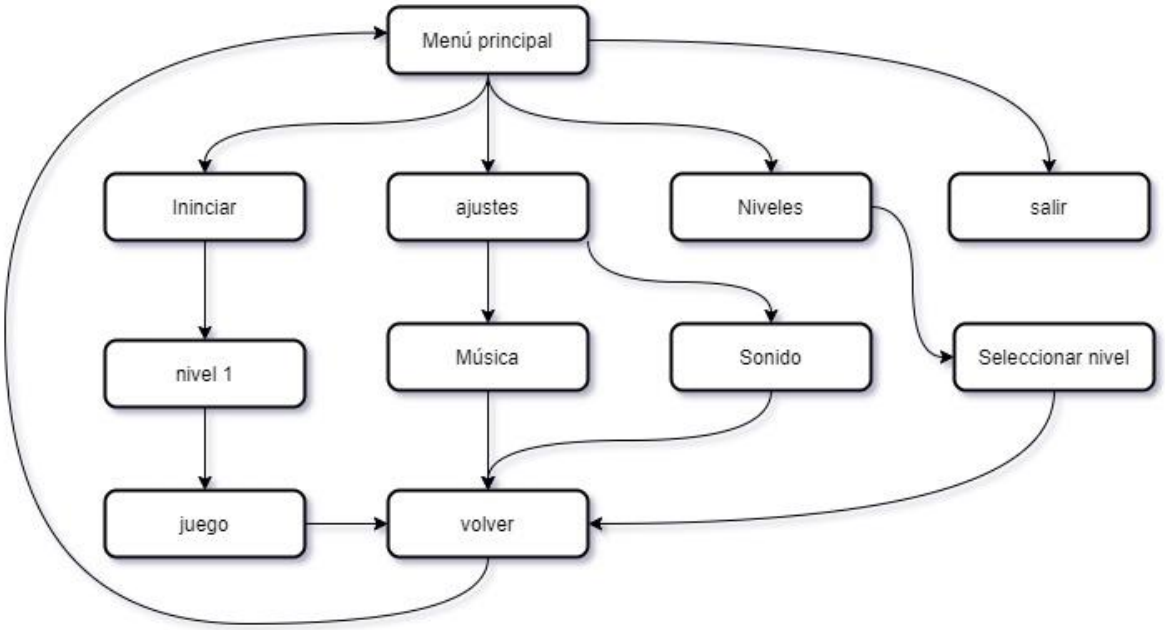


Diagrama de flujo



2.3 Plataforma

La plataforma en la que se está desarrollando el videojuego es PC, posiblemente además se incluirán controles para móvil.

2.4 Género

El género es de tipo plataformas ya que este va avanzando por los distintos niveles intentando sobrevivir a los diversos obstáculos del juego.

2.5 Clasificación

La clasificación es E según la ESRB. Apto para todas las edades, no posee violencia y posee personajes tipo caricatura.

2.6 Tipo de Animación

Animación Bidimensional 2D por Computadora. Se desarrollará de esta forma para la realización de mundos por medio de un TileMap que es como una herramienta para construir mapas por bloques y además aprovechar los personajes, backgrounds, plataformas y animaciones de este asset.

2.7 Equipo de Trabajo

Ingenieros de audio: Jeury Payano

Diseñadores: Sadiel Henríquez

Ilustradores: Sadiel Henríquez

Programadores: Jeury Payano, Randiel Arias

Animadores: Jeury Payano, Randiel Arias

Marketing: Jeury Payano, Randiel Arias, Sadiel Henriquez

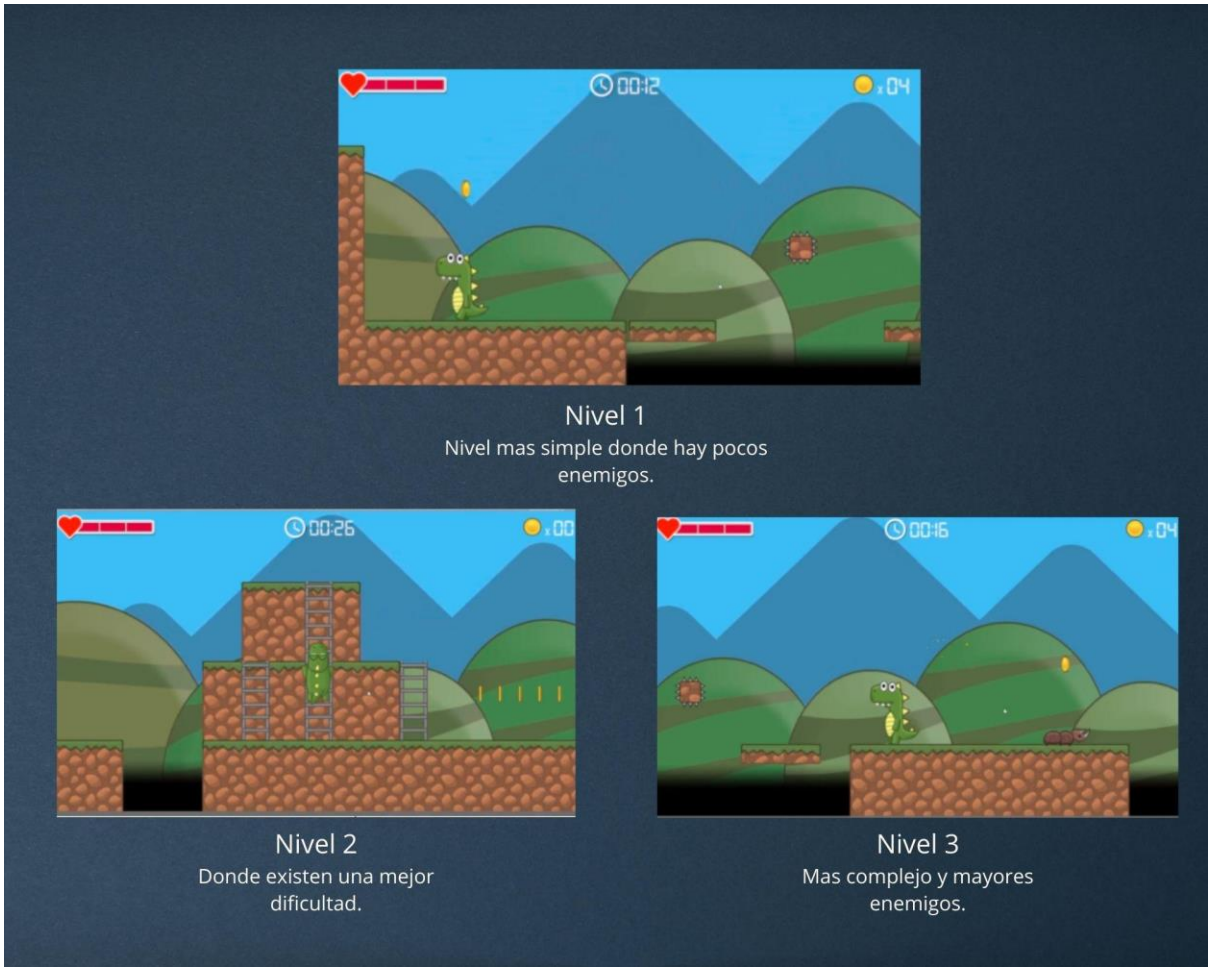
2.8 Historia

Dino es una criatura prehistórica que se encuentra en una dimensión con distintos ambientes. Este tiene que demostrar que es capaz de superar y sobrevivir a todos los obstáculos que posee la dimensión y sobrepasar todos sus ambientes recogiendo las monedas que son necesarias para abrir la puerta de pase a los siguientes ambientes.

2.9 Guión

El juego iniciara con un menú de opciones donde se elija el modo si es fácil, medio y difícil ya una vez seleccionada la opción Dino (Personaje principal del videojuego) inicia la partida caminando en el ambiente de juego que sería un bosque lleno de obstáculos y árboles donde se pueda escalar a la hora de escaparse de un ladrón o los ladrones, esos enemigos se trasladan de un lugar a otro en una misma distancia todo el momento como protegiendo ese terreno con el fin de que nadie pase sin que él los vea. Este deberá ir recogiendo todas las medicinas que se encuentren al aire libre en el ambiente en que se encuentre y no debe de ser visto por ningún peligro mientras recolecta una buena cantidad de medicina hasta poder completar la mayor cantidad de medicinas posibles en ese nivel, por cada vez que el personaje choque o roce con uno de los ladrones esta irá perdiendo un porcentaje de medicina, ya si este es atrapado será eliminado y devuelto a iniciar el juego o ese nivel nuevamente. Mientras más alto de nivel se encuentre Dino más difícil se le tornara poder conseguir las medicinas de su hija, debe de usar la inteligencia para plantear estrategias que puedan dar resultados en esta gran aventura. Al final del último nivel tendrá que saltar sin ser atrapado por el grande encargado de mantener los ladrones en busca de la medicina, si se logra durar un minuto sin ser atrapado conseguirá pasar de nivel, pero sino este tendrá que iniciar de nuevo la partida.

2.10 Storyboard



2.11 Personajes

enemigos:

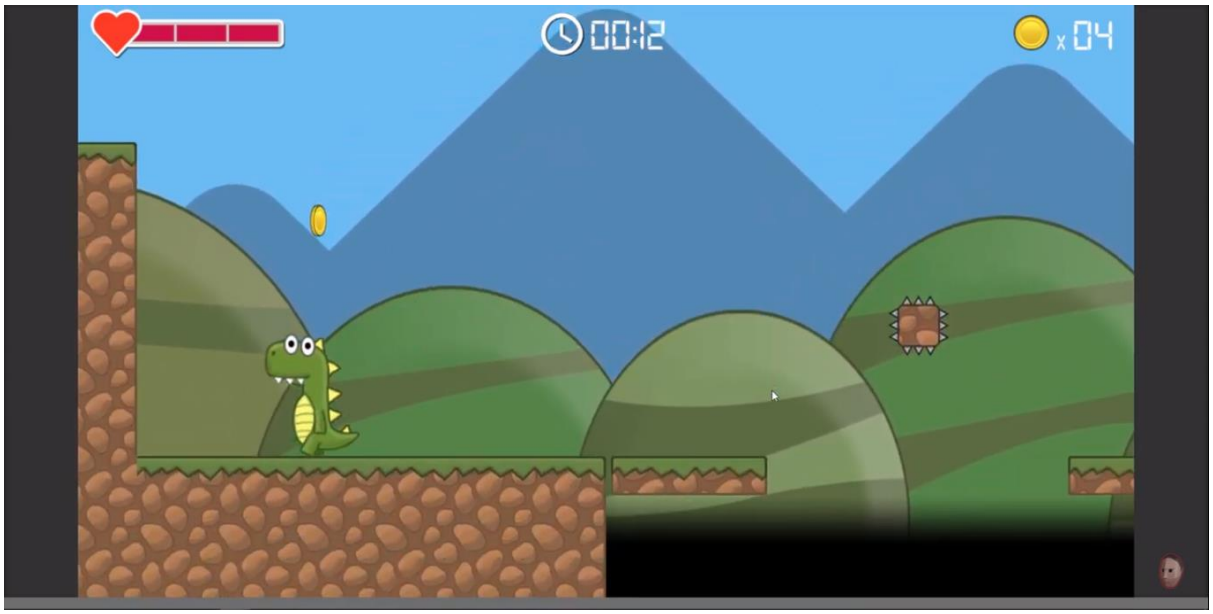
Escarabajo Subterráneo

Personaje principal:

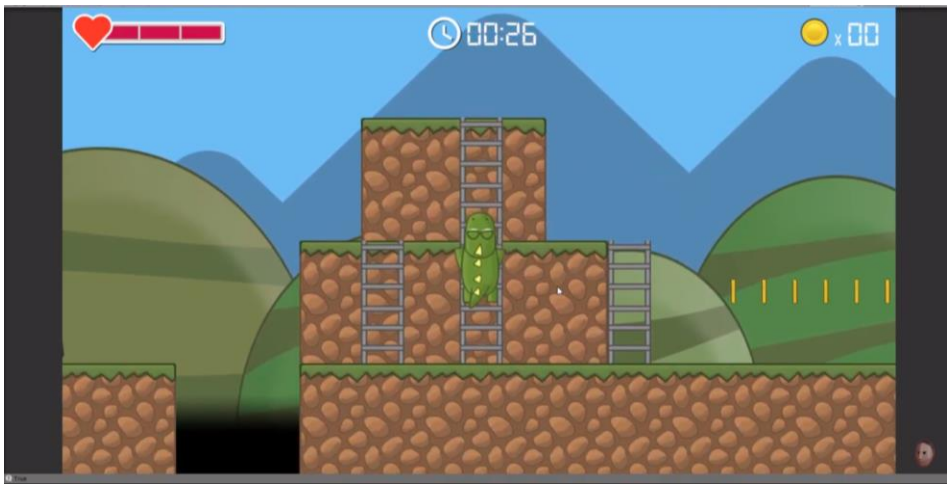
Dino

2.12 Niveles

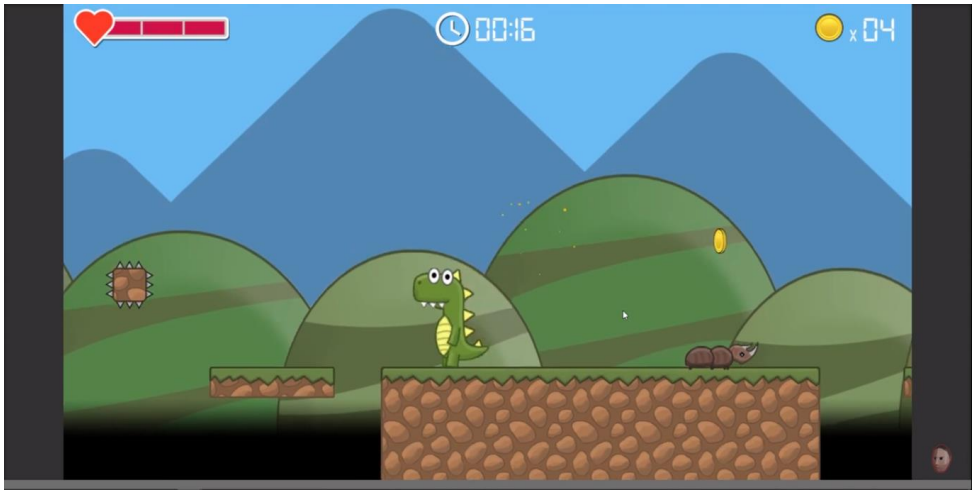
Nivel 1



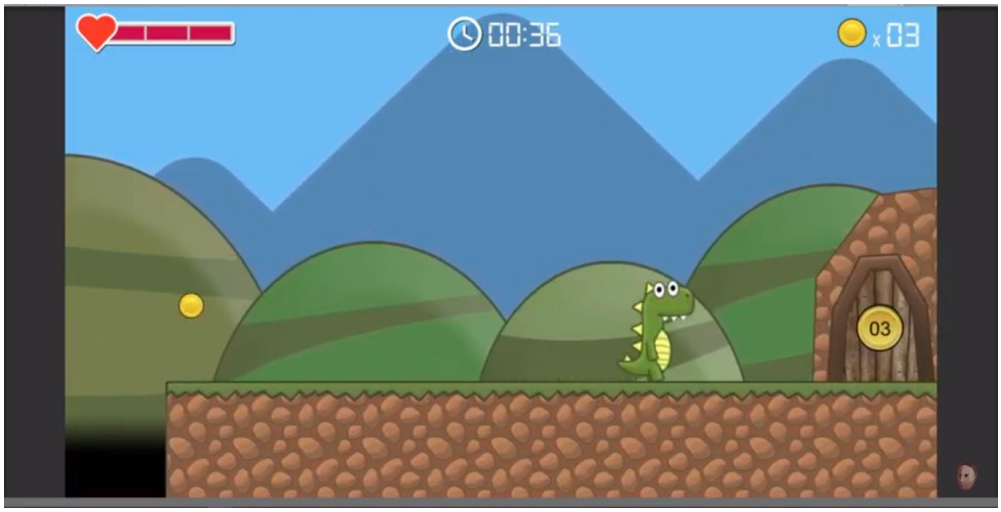
Nivel 2



nivel 3

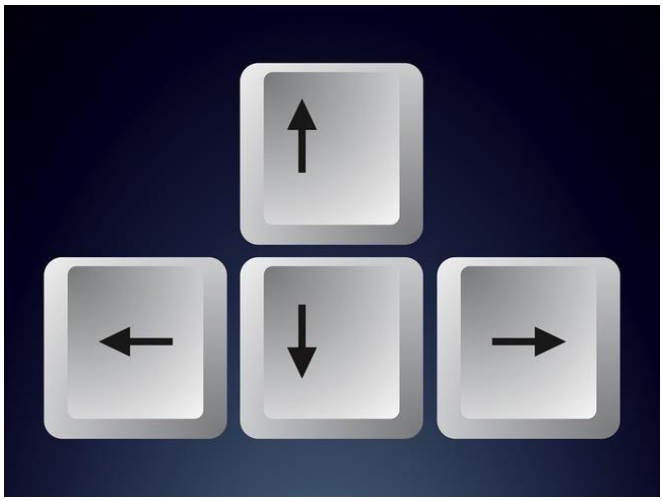


Puerta a siguiente nivel



2.13 Mecánica del Juego

El juego poseerá un script que permita mover al jugador por medio de las teclas de flechas del teclado, que es lo estándar en casi todos los juegos.



3.1 Capturas de la Aplicación

Scripts

Script para plataformas en movimiento.

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

public class PlansforMovil : MonoBehaviour
{
      [SerializeField] private Transform destino;
      [SerializeField] private float velocidad;
      private Vector3 posIni, posFin;
    // Start is called before the first frame update
    void Start()
    {
        destino.parent = null;
        posIni = transform.position;
        posFin = destino.position;
    }

    // Update is called once per frame
    void Update()
    {
        transform.position = Vector3.MoveTowards(transform.position, destino.position, velocidad * Time.deltaTime);
         if(transform.position == destino.position){
            //if(destino.position == posFin) destino.position = posIni; else destino.position = posFin;
            destino.position = (destino.position == posFin) ? posIni : posFin;
         }
    }
```

Script para las monedas.

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

public class CoindController : MonoBehaviour
{
  private void OntriggerEnter2D(Collider2D collision){
    if(collision.gameObject.tag == "Player"){
        GameController.sumaMoneda();
        Destroy(gameObject);
    }
  }

}
```

Ln 15, Col 1 Spaces: 2 UTF-8 with BOM CRLF C# Prettier

Script para controlar el tipo de salto del personaje.

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

public class MejorarSalto : MonoBehaviour
{

    public float saltoLargo = 1.5f;
    public float saltoCorto = 1f;

    private Rigidbody2D rd;

    // Start is called before the first frame update
    void Start()
    {
        rd = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        if(rd.velocity.y < 0)
        {
            rd.velocity += Vector2.up * Physics2D.gravity.y * saltoLargo * Time.deltaTime;

        } else if(rd.velocity.y > 0 && !Input.GetButtonDown("Jump")){

        rd.velocity += Vector2.up * Physics2D.gravity.y * saltoCorto * Time.deltaTime;

        }
```

Script para el controlador del tiempo.

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

public class TimeController : MonoBehaviour
{
    [SerializeField] int min, seg;
    [SerializeField] Text tiempo;
    // Start is called before the first frame update
    private float restante;
    private bool enMarcha;

    private void Awake(){
        restante = (min * 60) + seg;
        enMarcha = true;
    }
    // Update is called once per frame
    void Update()
    {
        if(enMarcha){
            restante -= Time.deltaTime;
            if(restante < 1){
             enMarcha = true;
             PlayerController.muerteExt = true;
            }
            int tiempoMin = Mathf.FloorToInt(restante / 60);
            int temSegundo = Mathf.FloorToInt(restante % 60);
            tiempo.text = string.Format("{00:00}:{01:00}", tiempoMin, temSegundo);
        }
```

Script para rocas en movimiento.

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

public class PlatRoca : MonoBehaviour
{
    [SerializeField] private Transform desde;
    [SerializeField] private Transform hasta;
    // Start is called before the first frame update
    private void OnDrawGizmosSelected(){
        Gizmos.color = Color.cyan;
        Gizmos.DrawLine(desde.position, hasta.position);
        Gizmos.DrawSphere(desde.position, 0.1f);
        Gizmos.DrawSphere(hasta.position, 0.1f);
    }
}

```

Scripts para los enemigos.

```
using UnityEngine;

public class Enemigos : MonoBehaviour
{

    [SerializeField] private Transform[] puntosMov;
    [SerializeField] private float velocidad;
    [SerializeField] private GameObject padre;
    private int i = 0;
    private Vector3 scalaIni, scalaTem;
    private float miraDer = 1;
    // Start is called before the first frame update
    void Start()
    {
        scalaIni = transform.localScale;
    }

    // Update is called once per frame
    void Update()
    {
        transform.position = Vector2.MoveTowards(transform.position, puntosMov[i].transform.position, velocidad * Time
      if(Vector2.Distance(transform.position, puntosMov[i].transform.position) < 0.1f){
            if(puntosMov[i] != puntosMov[puntosMov.Length - 1]) i++;
            else i = 0;
            miraDer = Mathf.Sign(puntosMov[i].transform.position.x - transform.position.x);

            gira(miraDer);
      }
    }
```

```
public void Muere(){
    Destroy(padre);
    }
    private void gira(float lado){
         if(miraDer == -1){
             scalaTem = transform.localScale;
             scalaTem.x = scalaTem.x * -1;
         }else scalaTem = scalaIni;

         transform.localScale = scalaTem;
    }
}
```

Script para controlar colisiones con el piso y demás objetos del entorno.

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

public class CheackGround : MonoBehaviour
{
    public static bool colPies;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Terreno") colPies = true;
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Terreno") colPies = false;
    }
}

```

Script que permite el seguimiento de la cámara.

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

public class SegumientoCamara : MonoBehaviour
{
    public Vector2 minCamPos, masCamPos, velocidad;
    public GameObject seguir;
    public float movSuave;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        float posX = Mathf.SmoothDamp(transform.position.x, seguir.transform.position.x, ref velocidad.x, movSuave);
        float posY = Mathf.SmoothDamp(transform.position.y, seguir.transform.position.y, ref velocidad.y, movSuave);

        transform.position = new Vector3(posX, posY, transform.position.z);
    }
}

```

Script que guarda un nuevo punto de inicio para el jugador.

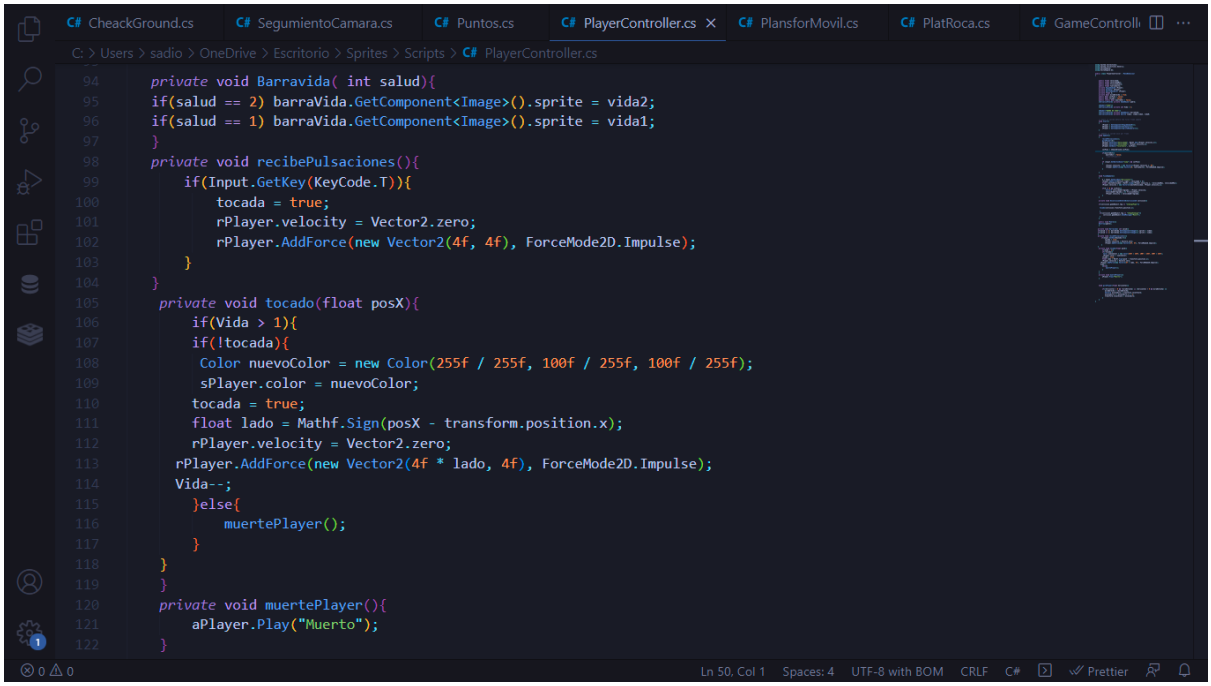
```cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Puntos : MonoBehaviour
{
    [SerializeField] private Transform origen;
    [SerializeField] private Transform destino;
    // Start is called before the first frame update
   private void OnDrawGizmosSelected(){
        Gizmos.color = Color.cyan;
        Gizmos.DrawSphere(origen.position, 0.1f);
        Gizmos.DrawSphere(destino.position, 0.1f);
    }
}

```

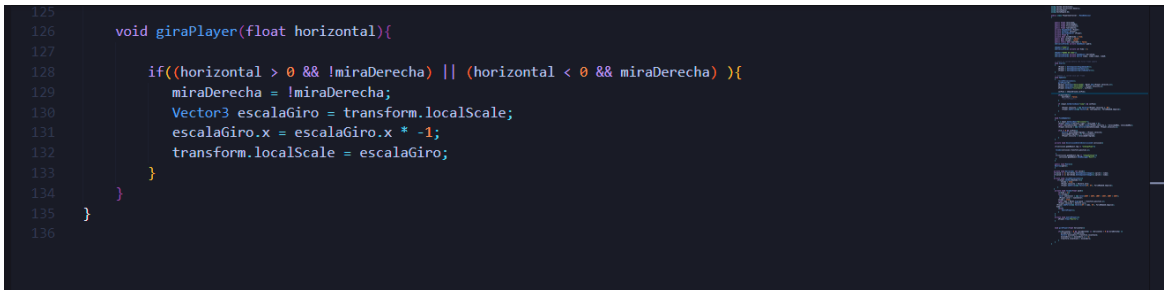
Script para manejar al jugador

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

public class PlayerController : MonoBehaviour
{


    public float velocidad;
    public float velocidadMax;
    public float friccionSuelo;
    public float fuerzaSalto;
    private Rigidbody2D rPlayer;
    private Animator aPlayer;
    private SpriteRenderer sPlayer;
    private float h;
    private bool miraDerecha = true;
    public bool colPies = false;
    public bool tocada = false;
    public static bool muerteExt = false;
    [SerializeField] private GameObject padre;

    [Header("VIDA")]
    [SerializeField] private int Vida = 2;

    [Header("BARRA DE VIDA")]
    [SerializeField] private GameObject barraVida;
    [SerializeField] private Sprite vida1, vida2,vida3, vida0;
```

```
void Start()
{
    rPlayer = GetComponent<Rigidbody2D>();
    aPlayer = GetComponent<Animator>();
    sPlayer = GetComponent<SpriteRenderer>();
}

// Update is called once per frame
void Update()
{
    recibePulsaciones();
    giraPlayer(h);
    aPlayer.SetFloat("VelocidadX", Mathf.Abs(rPlayer.velocity.x));
    aPlayer.SetFloat("VelocidadY", rPlayer.velocity.y);
    aPlayer.SetBool("TocarSuelo", colPies);

    colPies = CheackGround.colPies;

    if(muerteExt){
        muerteExt = false;
        //muertePlayer();
    }

    if (Input.GetButtonDown("Jump") && colPies)
    {
        rPlayer.velocity = new Vector2(rPlayer.velocity.x, 0f);
        rPlayer.AddForce(new Vector2(0, fuerzaSalto), ForceMode2D.Impulse);
    }
```

```
void FixedUpdate()
{
    h = Input.GetAxisRaw("Horizontal");
    rPlayer.AddForce(Vector2.right * velocidad * h);
    float limiteVelocidad = Mathf.Clamp(rPlayer.velocity.x, -velocidadMax, velocidadMax);
    rPlayer.velocity = new Vector2(limiteVelocidad, rPlayer.velocity.y);

    if(h == 0 && colPies){
        Vector3 velocidadArreglada = rPlayer.velocity;
        velocidadArreglada.x *= friccionSuelo;
        rPlayer.velocity = velocidadArreglada;
    }
}

private void OncollisionEnter2D(Collision2D collision){

if(collision.gameObject.tag == "enemigoPupa"){

 tocado(collision.transform.position.x);

 }
 if(collision.gameObject.tag == "ChepaEnemigo"){
     collision.gameObject.SendMessage("Muere");
 }
}

public void Muere(){
Destroy(padre);
}
```





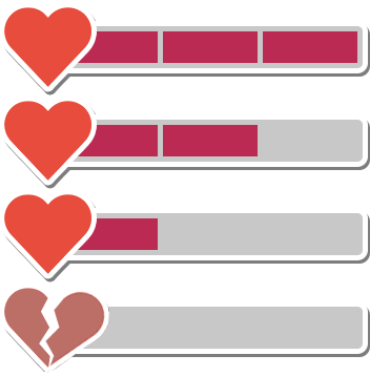
Sprints

Cartel de bienvenida

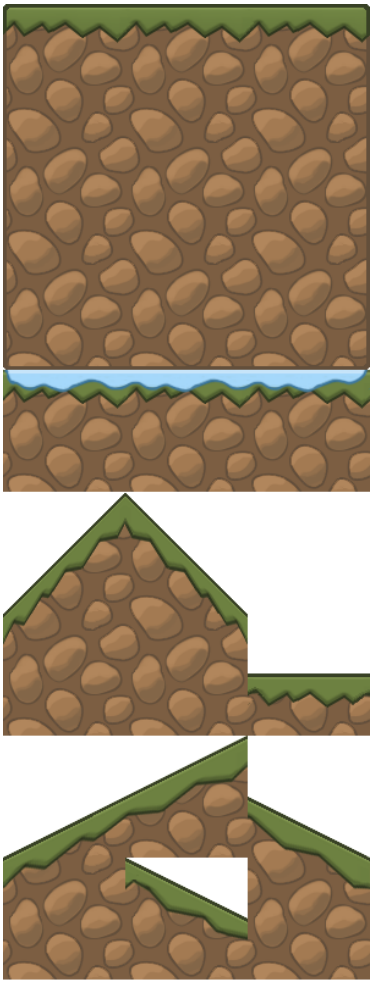
Enemigos:

Fondo:

Barra de vida:



Suelos:

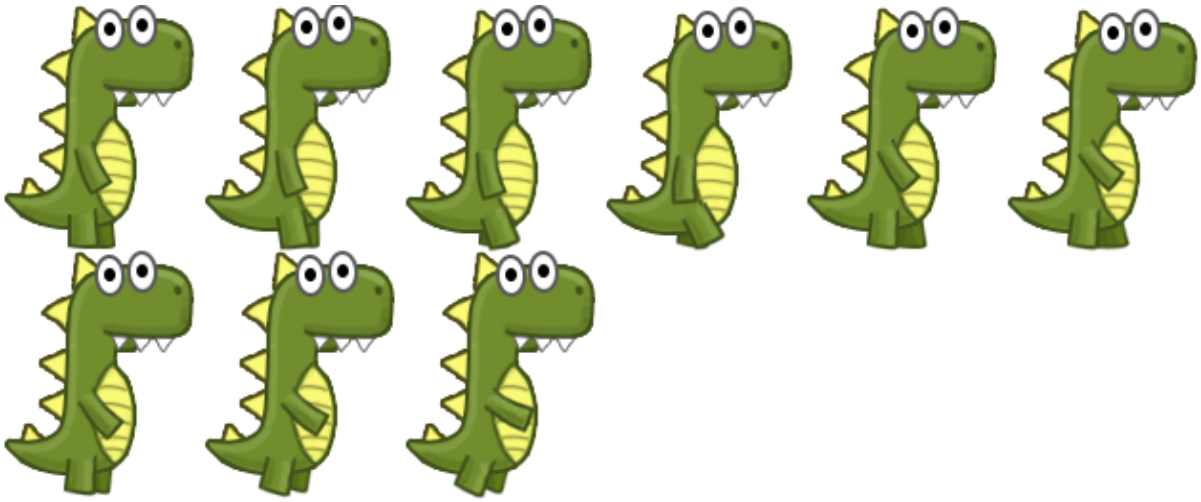


Objetos:

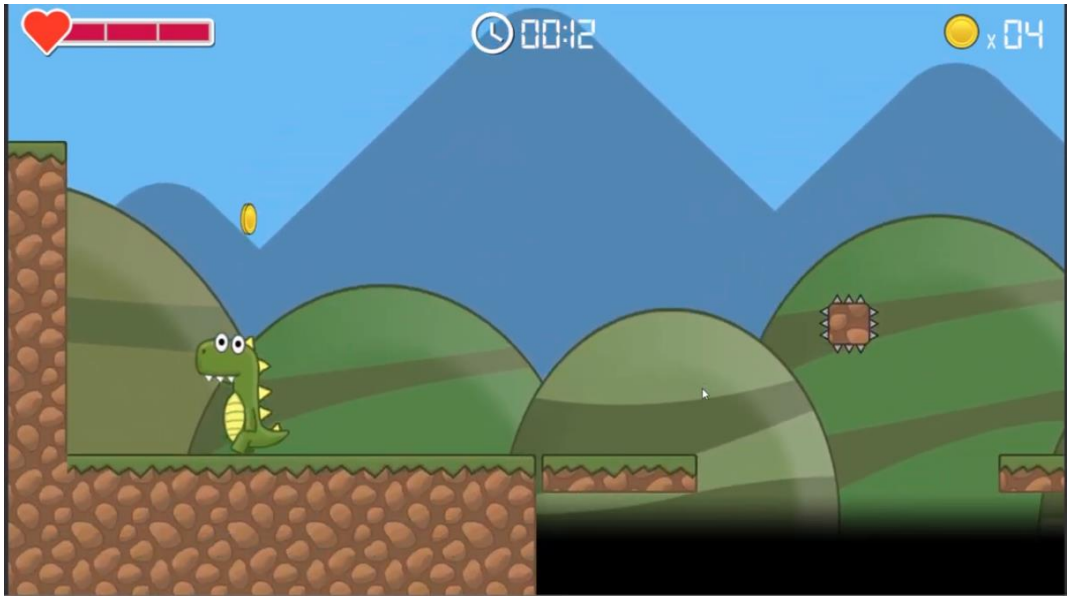
Monedas:

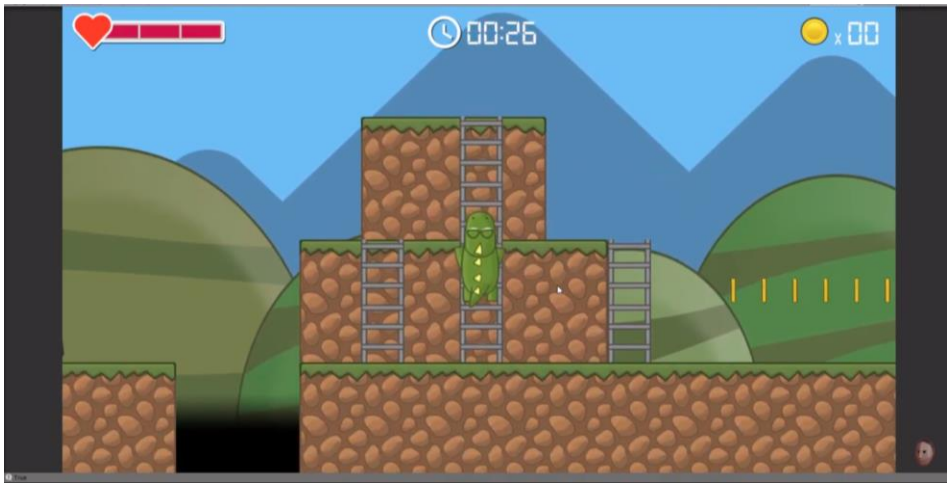
reloj:

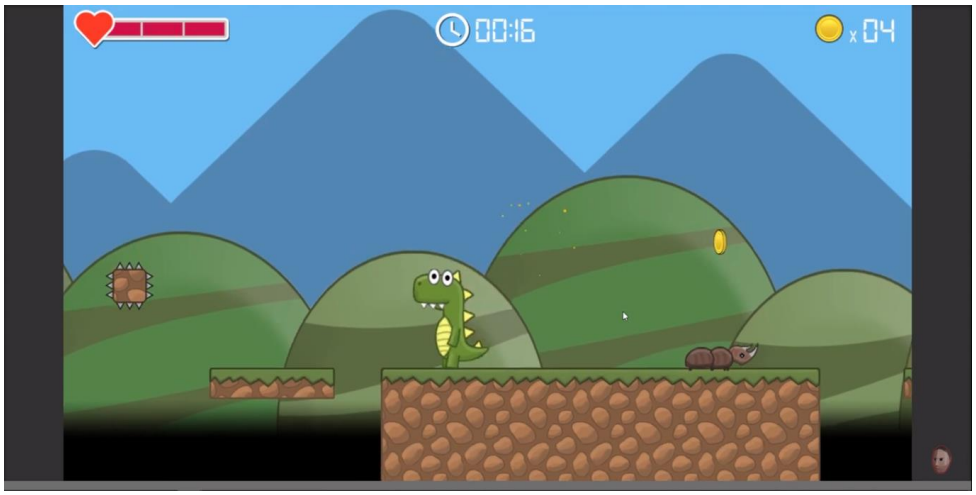
Personaje principal:

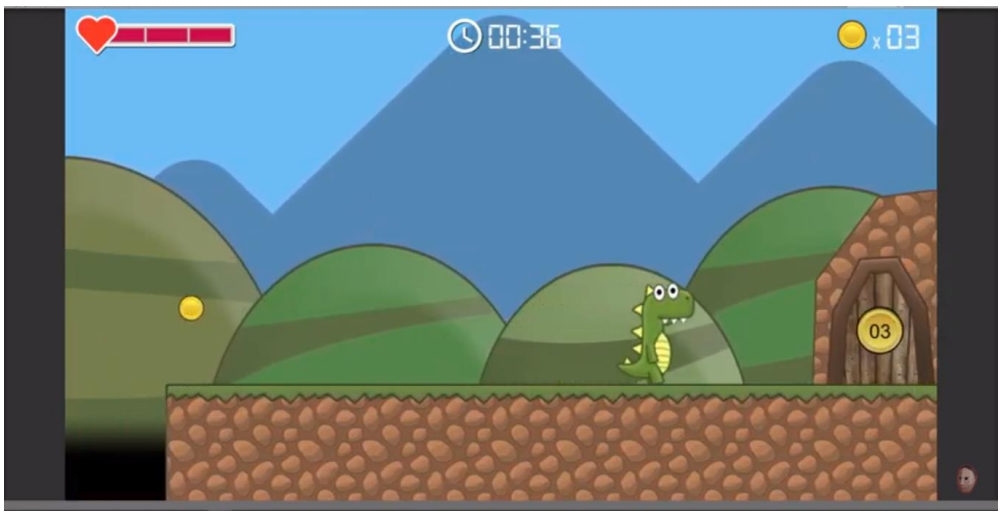


Imágenes:




00:26
x 00


00:16
x 04


00:36
x 03
03

3.2 Prototipos

Lo-Fi

- Primer prototipo con el primer nivel básico de varias monedas y movilidad del personaje principal y un enemigo.
- Segundo prototipo con el segundo nivel básico y mejora del salto del personaje y que mire para los lados y dos enemigos.
- Tercer prototipo con menú principal y menú de ajustes y varios enemigos y tercer nivel básico.
- Cuarto prototipo

Hi-Fi

- Primer prototipo con dos primeros niveles finalizados, realización de score de monedas y vidas del personaje.
- Segundo prototipo con el tercer y cuarto nivel finalizado, cuyos niveles tendrán trampolines, bolas de spikes movibles 360 grados, enemigos que vuelan, realización de mejoras en movimientos y animaciones en los personajes.

3.3 Perfiles de Usuarios

Los públicos objetivos del videojuego son:

- Personas de una edad comprendida entre los 6 y 50 años.
- Con el ingreso más básico existente.
- Personas con una formación mínima desde primero de básica o incluso cursos anteriores hasta carreras de educación superior.

3.4Usabilidad

La usabilidad del videojuego es bastante sencilla debido a que solo se trata de saber utilizar las teclas de direcciones para mover el personaje y la tecla de barra espaciadora para los saltos débiles o fuertes del personaje. Los mundos serán bastantes interactivos y entendibles pero que requerirán de paciencia para recolectar algunas monedas en algunos lugares porque habrá ciertos personajes moviéndose que no dejarán fácilmente tomarlas. En otras ocasiones hay que ser ágil debido a que algunos enemigos se moverán rápido y habrá que saltar y/o maniobrar en el momento adecuado.

3.5 Test

Sexo	Hombre
Edad	19
Nivel de estudio	junior developer
Aficiones	programación

Tareas	**Puntuación**
Jugabilidad	4
Dificultad	3
Control del personaje	4
Guía del usuario	2
La información proporcionada por el juego	1
Diseño visual	5
La coherencia	4

Sexo	**Hombre**
Edad	35
Nivel de estudio	Profesor
Aficiones	Lectura

Tareas	**Puntuación**
Jugabilidad	3
Dificultad	4
Control del personaje	4
Guía del usuario	1
La información proporcionada por el juego	1

Diseño visual	5
La coherencia	3

Sexo	Mujer
Edad	19
Nivel de estudio	Estudiante de ingeniería
Aficiones	Construcción

Tareas	Puntuación
Jugabilidad	5
Dificultad	2
Control del personaje	3
Guía del usuario	2
La información proporcionada por el juego	2
Diseño visual	5
La coherencia	3

Resultado del test:

Tareas	**Puntuación**
Jugabilidad	4
Dificultad	3
Control del personaje	3.7
Guía del usuario	1.7
La información proporcionada por el juego	1.3
Diseño visual	5
La coherencia	3.3

3.6 Versiones de la Aplicación

- **Versión Pre Alpha:** versión anterior donde se crearán niveles como prototipos básicos de pruebas y personaje principal con movimientos básicos.

- **Versión Alpha:** primera versión completa donde se realizan pruebas para verificar normalidad y búsqueda de bugs.

- **Versión Beta:** primera versión lanzada imperfecta pero que se podrá probar por público general.

- **Versión 1.0.0:** primera versión en ser lanzada como oficial y primera versión completa con los bugs testeados y recomendaciones realizadas donde se realizarán promociones por distintas vías como plataformas de streaming y redes sociales.

4.1 Requisitos de la instalación

Tener dispositivos regulares como PC y móvil para que así el juego tenga una mejor fluidez.

- 3 Gb de espacio libre en el disco.

- 4 Gb de memoria RAM en PC.

- 2 Gb de memoria RAM dispositivos móviles.

- Windows 7 o superior o MacOS X 10.8 o superior.

- Tarjeta gráfica con DX9 o DX11.

4.2 Instrucciones de Uso

En el videojuego se tendrá que manejar a Dino por diferentes pantallas, hacerlo saltar entre diferentes plataformas estáticas y/o movibles y saltarle encima a algunos enemigos para eliminarlos o simplemente esquivarlos en cada uno de los niveles. Para poder atravesar cada nivel necesitará recolectar todas las monedas requeridas en la puerta.

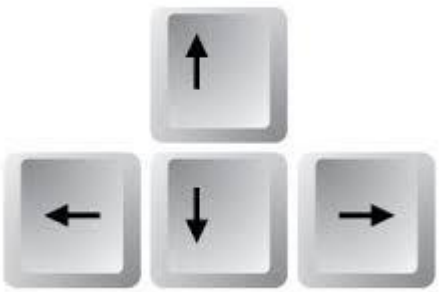
El personaje será controlado por las teclas de direcciones:

Tecla ESPACIO: saltar.

Tecla (->): para ir a la derecha.

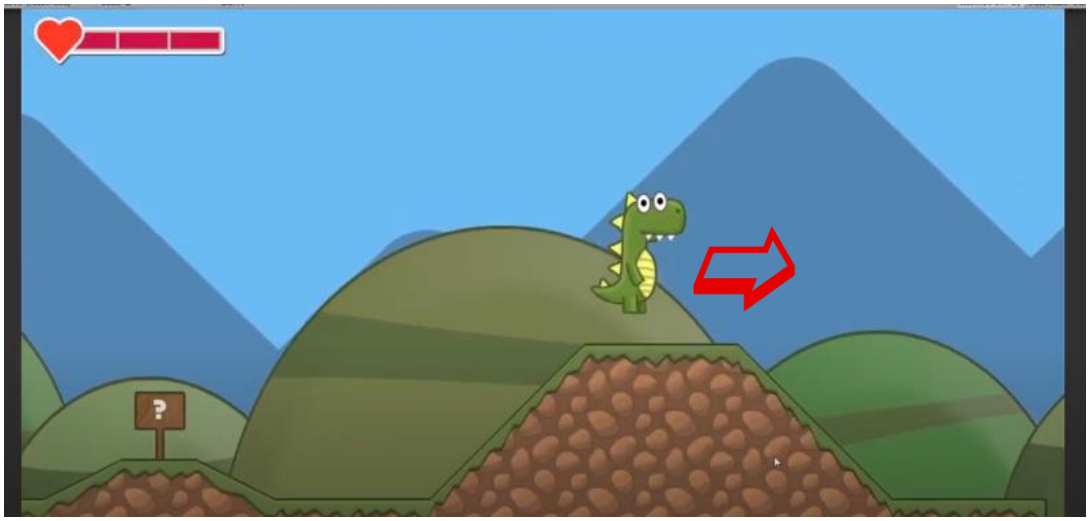
Tecla (<-): para ir a la izquierda.

Tecla de dirección abajo para caer de una plataforma.



Para atravesar la puerta sólo debe acercarse a esta y la misma tomará sus puntos recolectados.

4.3 Bugs



Cuando el personaje va a una alta velocidad(10) por una pendiente se sale del suelo como si fuera un vehículo.

4.4 Proyección a Futuro

Se realizará distribución del videojuego en diversas plataformas digitales, como en los smartphones y tablets por medio de las distintas tiendas de apps de cada uno de los sistemas operativos de estos dispositivos ya sea Google Play, App Store, etc. adaptando la resolución de la pantalla y modificar los controles del personaje a cada dispositivo nuevo.

4.5 Presupuesto

Inversión	Tiempo	Costo
Diseñador	2 Meses	500 USD
Programador	3 Meses	1500 USD
Técnico de sonido	1 Mes	250 USD
Testers	1 Mes	250 USD
Salida a tienda play store	3 Meses	25 USD
Total	10 Meses	2525 USD

Conclusiones

Sadiel Henriquez 1-17-0903

Mientras trabajas en el desarrollo de este videojuego puedes entender adecuadamente el desarrollo de los videojuegos y la importancia de hacer algo entretenido y cómodo para la audiencia sobre todo porque está dirigido a todo tipo de público lo que más destaca en parte para mí. Es la programación porque juega un papel muy importante en brindar la funcionalidad adecuada a los videojuegos y luego lo que más me gusta es el diseño porque tiene que destacar para el usuario.

Jeury Payano 2-15-0853

Para concluir con esta entrega se tocaron todos los puntos y temas tratados en la clase en cuanto a los conocimientos que se debían obtener en el transcurso para poder desarrollar el videojuego final, con esto quiero decir que en cada punto asignado para algún videojuego desarrollar he aprendido bastante cosas nuevas que nunca avisa visto y practicado. En el proyecto final se refleja lo aprendido en el curso de videojuegos ya que este se maneja en una plataforma totalmente abierta al aprendizaje.

Randiel Arias Canela 1-17-2703

En conclusión, gracias a este juego y a esta materia pudimos aprender mucho sobre el desarrollo de videojuegos usando la herramienta Unity, en esta aprendimos a desarrollar videojuegos tanto 2D como 3D, además de trabajar en un equipo para el desarrollo de este.

Con relevancia al videojuego presente la misión de este es pasar todo el nivel sin perder sus tres vidas ya sea cayendo al vacío, caer en pinchos o ser atacado por un enemigo de los que se encuentra a lo largo del mundo. Este juego el cual se podrá jugar desde la plataforma de itch.io y la finalidad de este es entretener a los usuarios o jugadores para sacarlos de la vida cotidiana mientras se entretienen con un excelente juego de plataformas.

LINK Github: https://github.com/jqblack/TercerParcialVideoJuego

LINK Itch.io: https://jeury1816.itch.io/dinojumpp

Referencias

https://pacobarba.com/blog/2020/juego-de-plataformas-2d-unity-tutorial-2020-capitulo-5-parte-1-pendientes-rampas-playercontroller-y-checkground/