UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS

CARRERA LICENCIATURA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

ANIMACIÓN DIGITAL Y VIDEOJUEGOS

Parcial 2

Título: Videojuego en Unity

Integrante:
Anderson, Justmary 8-987-2340

Profesor:

Ing. Samuel Jiménez

SEMESTRE I, 2024

Introducción

"El que Busca Encuentra" es un emocionante juego de aventuras en el que los jugadores se embarcan en una travesía llena de color, desafíos y recompensas. La narrativa del juego se desarrolla en un mundo vibrante y misterioso, dividido en tres niveles, cada uno representando un nuevo reto lleno de sorpresas. Al final de cada nivel, los jugadores encuentran un tesoro, lo que les motiva a seguir avanzando y superar los obstáculos que se presentan en su camino. Este documento presenta una visión detallada del juego, incluyendo sus objetivos, mecánicas, diseño de niveles y activos.

Tabla de contenido

Contenido del Trabajo Escrito

Objetivos

El objetivo principal de "El que Busca Encuentra" es guiar al jugador a través de niveles llenos de obstáculos y desafíos mientras recolecta monedas y mantiene su vida.

Los objetivos específicos:

- Recolectar todas las monedas ocultas en cada nivel.
- Evitar caídas y golpes que reducen la vida del jugador.
- Alcanzar la meta final en cada nivel para descubrir el tesoro.
- Superar los desafíos y obstáculos únicos de cada nivel.
- Utilizar los corazones para recuperar vida y prolongar la aventura.
- Completar los tres niveles del juego y descubrir todos los tesoros ocultos.

Explicación de la Mecánica del Juego

La mecánica de juego en "El que Busca Encuentra" es intuitiva pero desafiante, diseñada para mantener al jugador inmerso y comprometido. Aquí se desglosan los elementos clave de la mecánica del juego:

Movimientos Básicos

El jugador, representado por un avatar de esqueleto robótico, puede realizar movimientos básicos como caminar y saltar. Estos movimientos son esenciales para navegar por los niveles y evitar obstáculos. La fluidez y la precisión en los controles son cruciales para que el jugador pueda superar los desafíos presentados.

Recolección de Monedas

Cada nivel del juego está lleno de monedas escondidas que el jugador debe recolectar. Las monedas son el principal objetivo de recolección y están estratégicamente ubicadas en áreas que requieren habilidad y estrategia para alcanzar. La cantidad total de monedas recolectadas se muestra en la esquina superior izquierda de la pantalla, permitiendo al jugador llevar un seguimiento constante de su progreso.

Pérdida de Vida

El jugador comienza cada nivel con una vida total de 100 puntos. La vida se ve afectada de las siguientes maneras:

- **Caídas:** Si el jugador cae, pierde 35 puntos de vida y es devuelto a la posición inicial del nivel. Las monedas recolectadas hasta ese momento se mantienen.

- **Golpes por Obstáculos:** Si el jugador es golpeado por un obstáculo, pierde 20 puntos de vida y también es devuelto a la posición inicial del nivel, conservando las monedas recolectadas.

Restauración de Vida

A lo largo de los niveles, el jugador puede encontrar corazones o símbolos de más + que restauran 25 puntos de vida. Estos corazones son recompensas por completar ciertos desafíos o alcanzar áreas específicas dentro del nivel.

Barra de Vida

La barra de vida se visualiza en la pantalla y comienza completamente blanca, indicando que el jugador tiene toda su vida. A medida que el jugador pierde vida por caídas o golpes, la barra se va llenando de colores, proporcionando una indicación visual clara del estado de salud del jugador.

Canvases y Menús

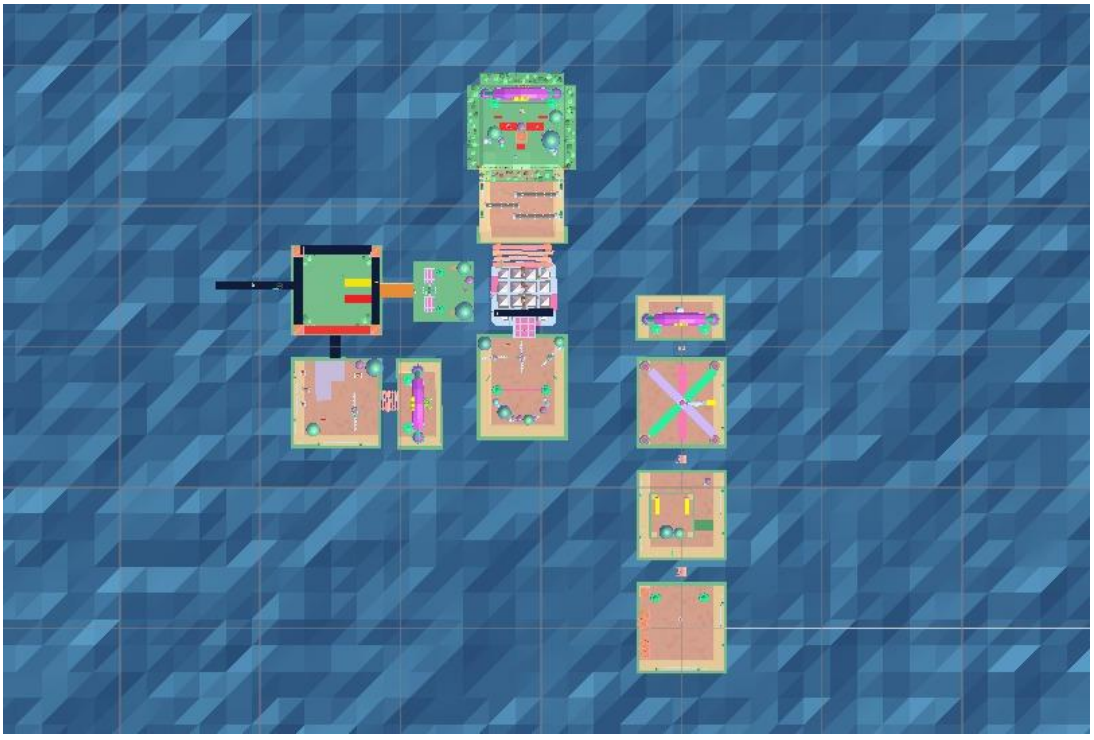
El juego incluye varios canvases y menús interactivos que mejoran la experiencia del jugador:

- **Canvas Felicitaciones:** Aparece al finalizar con éxito cada nivel, mostrando botones para regresar al menú principal o avanzar al siguiente nivel. En el nivel tres, el canvas incluye un botón para ver los créditos del juego.
- **Canvas Menú de Pausa:** Se accede presionando la tecla "Esc" y permite al jugador continuar, reiniciar el nivel, o acceder a configuraciones de audio.
- **Scene Menú:** El menú principal del juego, donde el jugador puede comenzar a jugar, ajustar opciones de audio, o salir del juego.
- **Scene Game Over:** Aparece cuando el jugador pierde toda su vida, ofreciendo un botón para regresar al menú principal.
- **Canvas Créditos:** Accesible desde el canvas felicitaciones que se muestra al completar el nivel tres, muestra los créditos del juego.

Puntos y Progresión

El jugador gana puntos al recolectar todas las monedas de un nivel y al llegar a la meta donde se encuentra el tesoro. La complejidad de los niveles incrementa gradualmente, introduciendo nuevos obstáculos y desafíos que requieren habilidades y estrategias avanzadas para superar.

Diseño de Niveles

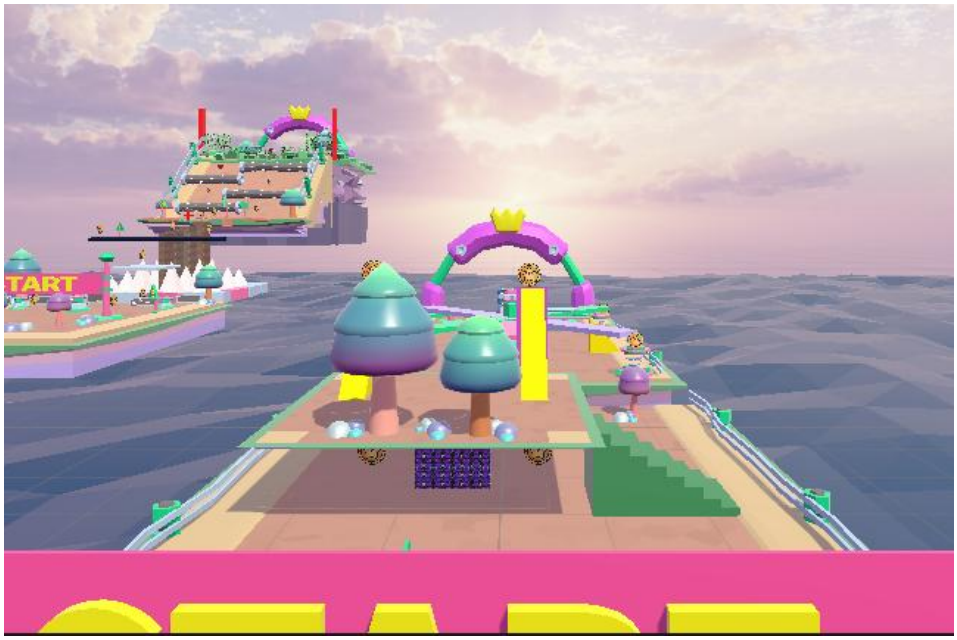


El juego consta de tres niveles, cada uno con obstáculos únicos y crecientes en dificultad:

Nivel 1: La Aventura Comienza

- **Breve descripción:** Tres pisos llenos de colores y desafíos iniciales. El jugador se enfrenta a una serie de obstáculos como un abanico gigante giratorio, un túnel con martillos colgantes, puentes y pisos falsos, una bola con pinchos que gira, y martillos gigantes.
- **Objetivo del jugador:** Encontrar y recolectar 9 monedas y llegar a la meta final.
- **Recompensa del nivel:** Avanzar al Nivel 2 y 3 corazones que suman 25 de vida adicionales.
- **Modos de juego principales:** Plataforma.
- **Obstáculos encontrados:**
    - Abanico gigante giratorio
    - Túnel con martillos colgantes
    - Puente falso
    - Piso falso
    - Bola con pinchos que gira
    - Martillos gigantes

Nivel 2: Desafíos en Movimiento



- **Breve descripción:** Tres pisos con obstáculos en constante movimiento, incluyendo barras con espinas que giran, puentes móviles, objetos que se desplazan y cuchillas giratorias.
- **Objetivo del jugador:** Encontrar y recolectar 10 monedas y llegar a la meta final.
- **Recompensa del nivel:** Avanzar al Nivel 3 y 4 corazones que suman 25 de vida adicionales.
- **Modos de juego principales:** Plataforma.
- **Obstáculos encontrados:**
    - Barras con espinas que giran

- Puentes que se mueven
- Objetos que se mueven
- Cuchilla giratoria

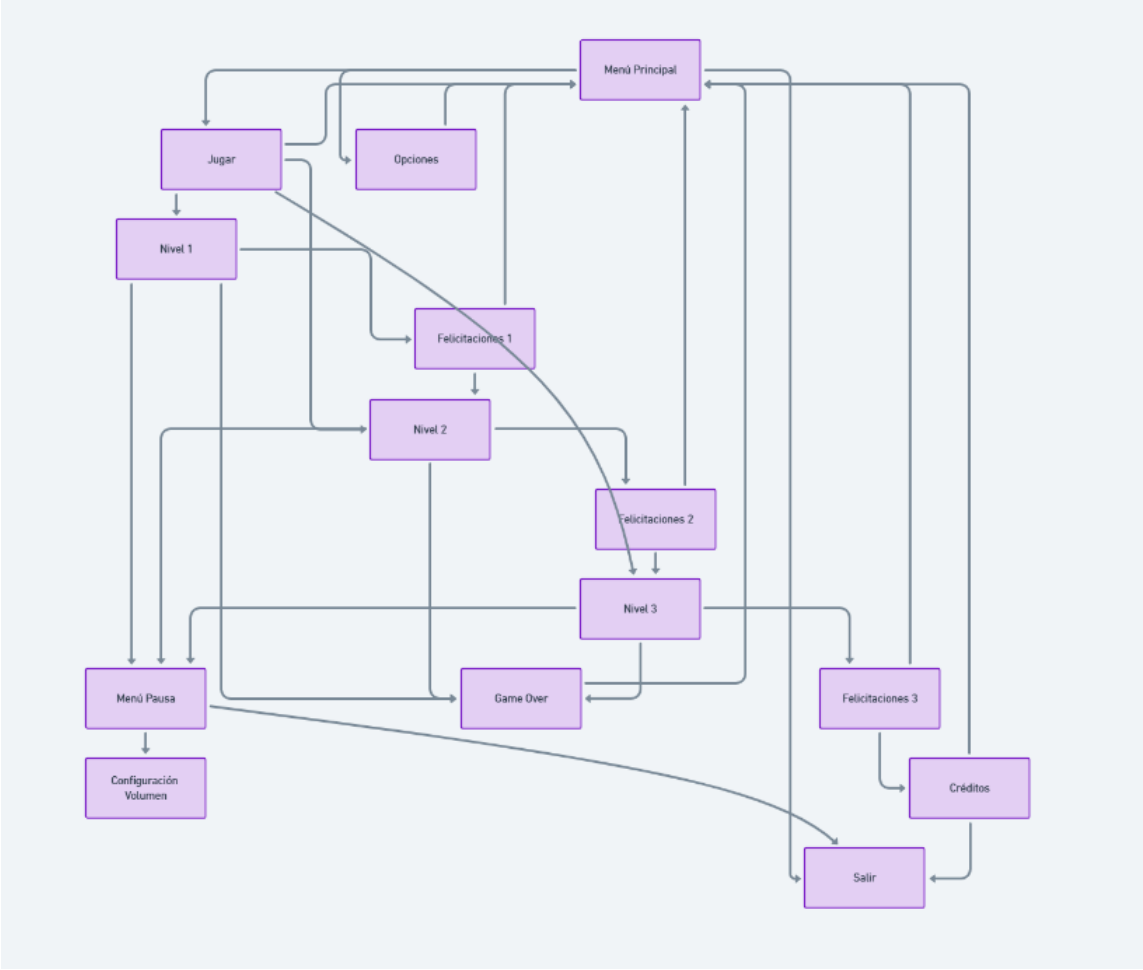
Nivel 3: La Batalla Final

- **Breve descripción:** Cuatro pisos con obstáculos complejos y variados, como aspas con pinchos que rotan, puentes que suben y bajan, pisos falsos, una escalera ascendente con peldaños móviles, y más.
- **Objetivo del jugador:** Encontrar y recolectar 19 monedas y llegar a la meta final.
- **Recompensa del nivel:** Completar el juego y visualizar los créditos del juego.
- **Modos de juego principales:** Plataforma.
- **Obstáculos encontrados:**
  - Aspas con pinchos que rotan
  - Puente que sube y baja
  - Piso falso
  - Escalera ascendente con peldaños móviles
  - Subida con barras que se mueven
  - Bola con pinchos que se mueve de lado a lado
  - Piso que aplasta moviéndose de arriba abajo
  - Púas que se abren y cierran

Cada nivel culmina con el jugador encontrando un tesoro y un canvas de felicitaciones que muestra los botones para avanzar al siguiente nivel o regresar al menú.

Diagrama de flujo de las Pantallas

Muestra cómo se conectan entre si todas las pantallas, desde pantalla de inicio hasta el game over.



Cámara del juego

Tipo de cámara específico:

• Tercera persona

Activos

"El que Busca Encuentra" incluye varios activos que enriquecen la experiencia de juego:

- **Avatar del jugador:** Representado por un esqueleto robótico.
- **Monedas:** Elementos recolectables que el jugador debe encontrar en cada nivel.
- **Obstáculos:** Diferentes tipos de trampas y peligros, como abanicos giratorios, martillos colgantes, bolas con pinchos, barras con espinas, puentes móviles, cuchillas giratorias, aspas con pinchos, escaleras móviles, y pisos aplastantes.
- **Corazones:** Objetos que restauran la vida del jugador.
- **Interfaz de usuario:** Incluye el cronómetro, barra de vida, contador de monedas, canvas de felicitaciones, menú de pausa, menú principal, escena de game over y créditos.
- **Sonido:** Música de fondo y efectos de sonido, incluyendo el cronómetro y sonidos asociados con recolectar monedas y recibir daño.
- **Diseño de niveles:** Ambientes coloridos y variados que ofrecen diferentes retos y recompensas en cada piso.

Código escrito en #C

Codigo del Canvas Menu Pausa:

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

public class codigopausa : MonoBehaviour
{
    [Header("Options")]
    public Slider VolumenMax;
    public Toggle mute;
    public AudioMixer mixer;
    public AudioSource fxSource;
    public AudioClip clickSound;

    [Header("Panels")]
    public GameObject panelprincipal;
    public GameObject panelopciones;
    public GameObject panelPausa; // El panel del menú de pausa

    private void Awake()
    {
        VolumenMax.onValueChanged.AddListener(ChangeVolumenMax);
        mute.onValueChanged.AddListener(ToggleMute);
        Time.timeScale = 1f; // Asegurar que el juego comience en estado activo
    }

    private void Update()
    {
        // Check if the Escape key is pressed to pause/unpause the game
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (Time.timeScale == 1f)
            {
                PausarJuego();
            }
            else
            {
                ReanudarJuego();
```

```
        }
    }
}

public void OpenPanel(GameObject panel)
{
    panelprincipal.SetActive(false);
    panelopciones.SetActive(false);

    panel.SetActive(true);
    PlaySoundButton();
}

public void Jugar1()
{
    SceneManager.LoadScene("Nivel1");
}

public void Jugar2()
{
    SceneManager.LoadScene("Nivel2");
}

public void Jugar3()
{
    SceneManager.LoadScene("Nivel3");
}

public void Salir()
{
    Debug.Log("SALIENDO.....");
    Application.Quit();
}

public void ChangeVolumenMax(float v)
{
    mixer.SetFloat("VolumenMaster", Mathf.Log10(v) * 20);
}

public void ToggleMute(bool isMuted)
{
    float volume = isMuted ? -80 : Mathf.Log10(VolumenMax.value) * 20;
    mixer.SetFloat("VolumenMaster", volume);
```

```
    }

    public void PlaySoundButton()
    {
        if (fxSource != null && clickSound != null)
        {
            fxSource.PlayOneShot(clickSound);
        }
    }

    // Función para reanudar el juego
    public void ReanudarJuego()
    {
        panelPausa.SetActive(false); // Desactivar el menú de pausa
        Time.timeScale = 1f; // Reanudar el tiempo
    }

    // Función para pausar el juego
    public void PausarJuego()
    {
        panelPausa.SetActive(true); // Activar el menú de pausa
        Time.timeScale = 0f; // Pausar el tiempo
    }

    // Función para reiniciar el nivel
    public void ReiniciarNivel()
    {
        Time.timeScale = 1f; // Asegurar que el tiempo esté en marcha antes de reiniciar
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```

Codigo para que al apretar ¨Esc¨ aparezca el Menu de Pausa:

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

public class escape : MonoBehaviour
{
    public GameObject ObjetoMenuPausa;
    public bool Pausa = false;
```

```
    void Start()
    {
        Time.timeScale = 1f; // Asegurarse de que el juego no está pausado al iniciar
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (Pausa == false)
            {
                ObjetoMenuPausa.SetActive(true);
                Pausa = true;

                // Asegurarse de que el cursor esté visible y desbloqueado
                Cursor.visible = true;
                Cursor.lockState = CursorLockMode.None;

                // Reanudar el juego antes de cargar la escena del menú
                Time.timeScale = 1f;
                SceneManager.LoadScene("Menu Pausa");
            }
        }
    }
}
```

Código para las funciones del canvas Felicitaciones:

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


public class felicitacion : MonoBehaviour
{
    public void regresar()
    {
        SceneManager.LoadScene("Menu");
    }

    public void Jugar1()
```

```
    {
        SceneManager.LoadScene("Nivel1");
    }

    public void Jugar2()
    {
        SceneManager.LoadScene("Nivel2");
    }

    public void Jugar3()
    {
        SceneManager.LoadScene("Nivel3");
    }

    public void Creditos()
    {
        SceneManager.LoadScene("Creditos");
    }
    void Start()
    {
        // Asegurarse de que el cursor esté visible y desbloqueado
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }
}
```

Código para Game Over:

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


public class GameOverManager : MonoBehaviour
{
    public void regresar()
    {
        SceneManager.LoadScene("Menu");
    }

void Start()
    {
```

```
        // Asegurarse de que el cursor esté visible y desbloqueado
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }
}
```

Código para manejar el conteo de Monedas y cronometro:

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

public class Monedas : MonoBehaviour
{
    public int contadorMoney = 0; // Inicializar el contador a 0
    public TextMeshProUGUI textoContador; // Referencia al TextMeshPro en el Canvas
    public GameObject canvasFelicitaciones; // Referencia al Canvas de felicitaciones
    public AudioSource audioSource; // Referencia al AudioSource para las monedas
    public AudioSource audioSourceCronometro; // Referencia al AudioSource para el
cronómetro
    public AudioClip sonidoMoneda; // Referencia al AudioClip para las monedas
    public int monedasParaGanar = 5; // Cantidad de monedas necesarias para ganar

    void Start()
    {
        Time.timeScale = 1f; // Asegurarse de que el juego no está pausado al iniciar

        if (textoContador == null)
        {
            Debug.LogError("TextoContador no está asignado en el Inspector.");
        }
        else
        {
            textoContador.text = "Monedas: " + contadorMoney; // Inicializar el texto
        }

        if (canvasFelicitaciones == null)
        {
            Debug.LogError("CanvasFelicitaciones no está asignado en el Inspector.");
        }
        else
        {
```

```
        canvasFelicitaciones.SetActive(false); // Inicialmente ocultar el Canvas de felicitaciones
    }

    if (audioSource == null)
    {
        Debug.LogError("AudioSource no está asignado en el Inspector.");
    }

    if (sonidoMoneda == null)
    {
        Debug.LogError("SonidoMoneda no está asignado en el Inspector.");
    }

    if (audioSourceCronometro == null)
    {
        Debug.LogError("AudioSourceCronometro no está asignado en el Inspector.");
    }
}

private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("Monedas"))
    {
        if (audioSource != null && sonidoMoneda != null)
        {
            audioSource.PlayOneShot(sonidoMoneda);
        }

        Destroy(other.gameObject);
        contadorMoney++;
        textoContador.text = "Monedas: " + contadorMoney; // Actualizar el texto
        Debug.Log("Monedas recogidas: " + contadorMoney);

        if (contadorMoney >= monedasParaGanar)
        {
            MostrarFelicitaciones();
        }
    }
}

private void MostrarFelicitaciones()
{
    canvasFelicitaciones.SetActive(true); // Mostrar el Canvas de felicitaciones
    Time.timeScale = 0f; // Pausar el juego
```

```
        // Pausar la música del cronómetro
        if (audioSourceCronometro != null)
        {
            audioSourceCronometro.Pause();
        }
    }
}
```

Mover objetos en el eje Z:

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

public class Mover : MonoBehaviour
{
    public float velocidadMovimiento = 5f;
    public float limiteMovimiento = 10f;

    private bool moviendoAdelante = true;
    private float posicionInicial;

    // Start is called before the first frame update
    void Start()
    {
        // Guardar la posición inicial del objeto en el eje Z
        posicionInicial = transform.position.z;
    }

    // Update is called once per frame
    void Update()
    {
        // Movimiento hacia adelante y hacia atrás automáticamente
        if (moviendoAdelante)
        {
            // Mover hacia adelante
            transform.Translate(Vector3.forward * velocidadMovimiento * Time.deltaTime);
            // Cambiar dirección al alcanzar el límite superior
            if (transform.position.z >= posicionInicial + limiteMovimiento)
            {
                moviendoAdelante = false;
            }
        }
        else
        {
```

```
            // Mover hacia atrás
            transform.Translate(Vector3.back * velocidadMovimiento * Time.deltaTime);
            // Cambiar dirección al alcanzar el límite inferior
            if (transform.position.z <= posicionInicial - limiteMovimiento)
            {
                moviendoAdelante = true;
            }
        }
    }
}
```

Código para mover en círculos en el eje Y:

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

public class MoverEnCirculo : MonoBehaviour
{
    public float radio = 5f;
    public float velocidadAngular = 1f;

    private float angulo = 0f;
    private Vector3 centro;

    // Start is called before the first frame update
    void Start()
    {
        // Guardar la posición inicial como el centro del círculo
        centro = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        // Calcular el nuevo ángulo
        angulo += velocidadAngular * Time.deltaTime;

        // Calcular la nueva posición
        float x = Mathf.Cos(angulo) * radio;
        float z = Mathf.Sin(angulo) * radio;

        // Establecer la nueva posición del objeto
        transform.position = new Vector3(centro.x + x, transform.position.y, centro.z + z);
    }
```

```
}
```

Código **para mover en círculos en el eje YX:**

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

public class MoverEnCirculoXY : MonoBehaviour
{
    public float radio = 5f;
    public float velocidadAngular = 1f;

    private float angulo = 0f;
    private Vector3 centro;

    // Start is called before the first frame update
    void Start()
    {
        // Guardar la posición inicial como el centro del círculo
        centro = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        // Calcular el nuevo ángulo
        angulo += velocidadAngular * Time.deltaTime;

        // Calcular la nueva posición
        float x = Mathf.Cos(angulo) * radio;
        float y = Mathf.Sin(angulo) * radio;

        // Establecer la nueva posición del objeto
        transform.position = new Vector3(centro.x + x, centro.y + y, transform.position.z);
    }
}
```

Código para mover en círculos en el eje ZY:

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

public class MoverEnCirculoZY : MonoBehaviour
{
    public float radio = 5f;
```

```
    public float velocidadAngular = 1f;

    private float angulo = 0f;
    private Vector3 centro;

    // Start is called before the first frame update
    void Start()
    {
        // Guardar la posición inicial como el centro del círculo
        centro = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        // Calcular el nuevo ángulo
        angulo += velocidadAngular * Time.deltaTime;

        // Calcular la nueva posición
        float z = Mathf.Cos(angulo) * radio;
        float y = Mathf.Sin(angulo) * radio;

        // Establecer la nueva posición del objeto
        transform.position = new Vector3(transform.position.x, centro.y + y, centro.z + z);
    }
}
```

Mover en el eje X:

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

public class MoverX : MonoBehaviour
{
    public float velocidadMovimiento = 5f;
    public float limiteMovimiento = 10f;

    private bool moviendoAdelante = true;
    private float posicionInicial;

    // Start is called before the first frame update
    void Start()
    {
        // Guardar la posición inicial del objeto en el eje X
```

```
        posicionInicial = transform.position.x;
    }

    // Update is called once per frame
    void Update()
    {
        // Movimiento hacia adelante y hacia atrás automáticamente en el eje X
        if (moviendoAdelante)
        {
            // Mover hacia adelante en el eje X
            transform.Translate(Vector3.right * velocidadMovimiento * Time.deltaTime);
            // Cambiar dirección al alcanzar el límite superior
            if (transform.position.x >= posicionInicial + limiteMovimiento)
            {
                moviendoAdelante = false;
            }
        }
        else
        {
            // Mover hacia atrás en el eje X
            transform.Translate(Vector3.left * velocidadMovimiento * Time.deltaTime);
            // Cambiar dirección al alcanzar el límite inferior
            if (transform.position.x <= posicionInicial - limiteMovimiento)
            {
                moviendoAdelante = true;
            }
        }
    }
}
```

Mover en eje Y:

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

public class MoverY : MonoBehaviour
{
    public float velocidadMovimiento = 5f;
    public float limiteMovimiento = 10f;

    private bool moviendoArriba = true;
    private float posicionInicial;

    // Start is called antes de la primera actualización del frame
```

```
    void Start()
    {
        // Guardar la posición inicial del objeto en el eje Y
        posicionInicial = transform.position.y;
    }

    // Update se llama una vez por frame
    void Update()
    {
        // Movimiento hacia arriba y hacia abajo automáticamente en el eje Y
        if (moviendoArriba)
        {
            // Mover hacia arriba en el eje Y
            transform.Translate(Vector3.up * velocidadMovimiento * Time.deltaTime);
            // Cambiar dirección al alcanzar el límite superior
            if (transform.position.y >= posicionInicial + limiteMovimiento)
            {
                moviendoArriba = false;
            }
        }
        else
        {
            // Mover hacia abajo en el eje Y
            transform.Translate(Vector3.down * velocidadMovimiento * Time.deltaTime);
            // Cambiar dirección al alcanzar el límite inferior
            if (transform.position.y <= posicionInicial - limiteMovimiento)
            {
                moviendoArriba = true;
            }
        }
    }
}
```

Utilizado para mover los martillos:

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

public class Pendulo : MonoBehaviour
{
    public float amplitud = 5f; // La distancia máxima a la que se mueve el péndulo desde el punto de origen
    public float frecuencia = 1f; // La velocidad de oscilación
```

```
    private Vector3 posicionInicial;

    // Start is called before the first frame update
    void Start()
    {
        // Guardar la posición inicial del objeto
        posicionInicial = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        // Calcular la nueva posición en el eje X usando una función seno para el movimiento de
péndulo
        float x = amplitud * Mathf.Sin(Time.time * frecuencia);

        // Actualizar la posición del objeto
        transform.position = new Vector3(posicionInicial.x + x, posicionInicial.y,
posicionInicial.z);
    }

}
```

Código para hacer rotar las Monedas:

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

public class Rotador : MonoBehaviour
{
    public float velocidad = 5;

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

    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(0, 0, velocidad * Time.deltaTime);
    }
}
```

Código para rotar en eje Y:

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

public class RotadorV : MonoBehaviour
{
    public float velocidad = 5;

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

    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(0, -velocidad * Time.deltaTime, 0); // Cambiar el signo para girar en
sentido contrario
    }

}
```

Código para rotar objetos en el eje X:

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

public class RotadorX : MonoBehaviour
{
    public float velocidad = 5f;

    // Update is called once per frame
    void Update()
    {
        // Rotar el objeto en el eje X
        transform.Rotate(velocidad * Time.deltaTime, 0, 0);
    }

}
```

Código que maneja la vida, daño, volver a la posición inicial, sonido de daño, etc del jugador:

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

public class VidaJugador : MonoBehaviour
{
    public int vidaMaxima = 100;
    public static int vidaActual;
    public Slider barraDeVida;
    public AudioSource audioSourceDaño; // Referencia al AudioSource para el sonido de daño
    public AudioSource audioSourceCaida; // Referencia al AudioSource para el sonido de caída
    public AudioClip sonidoDaño; // Referencia al AudioClip para el sonido de daño
    public AudioClip sonidoCaida; // Referencia al AudioClip para el sonido de caída
    public GameObject panelImpacto; // Referencia al panel que se mostrará al recibir daño
    private Vector3 posicionInicial; // Posición inicial del jugador
    private Quaternion rotacionInicial; // Rotación inicial del jugador
    private CharacterController characterController;

    void Start()
    {
        if (vidaActual == 0)
        {
            vidaActual = vidaMaxima; // Inicializa la vida solo si no está ya configurada
        }

        barraDeVida.maxValue = vidaMaxima;
        barraDeVida.value = vidaActual;

        if (audioSourceDaño == null)
        {
            Debug.LogError("AudioSourceDaño no está asignado en el Inspector.");
        }

        if (sonidoDaño == null)
        {
            Debug.LogError("SonidoDaño no está asignado en el Inspector.");
        }

        if (audioSourceCaida == null)
        {
            Debug.LogError("AudioSourceCaida no está asignado en el Inspector.");
```

```
    }

    if (sonidoCaida == null)
    {
        Debug.LogError("SonidoCaida no está asignado en el Inspector.");
    }

    if (panelImpacto == null)
    {
        Debug.LogError("PanelImpacto no está asignado en el Inspector.");
    }
    else
    {
        panelImpacto.SetActive(false); // Inicialmente ocultar el panel de impacto
    }

    characterController = GetComponent<CharacterController>();
    if (characterController == null)
    {
        Debug.LogError("CharacterController no está asignado en el objeto del jugador.");
    }

    posicionInicial = transform.position; // Guardar la posición inicial del jugador
    rotacionInicial = transform.rotation; // Guardar la rotación inicial del jugador
}

public void RecibirDanio(int cantidad)
{
    vidaActual -= cantidad;
    if (vidaActual <= 0)
    {
        vidaActual = 0;
        CargarEscenaGameOver(); // Cargar la escena de GAME OVER
    }
    else
    {
        ActualizarBarraDeVida();
        ReiniciarPosicionJugador(); // Reiniciar la posición del jugador al recibir daño
    }
}

public void Curar(int cantidad)
{
    vidaActual += cantidad;
    if (vidaActual > vidaMaxima)
```

```
    {
        vidaActual = vidaMaxima;
    }
    ActualizarBarraDeVida();
}

void ActualizarBarraDeVida()
{
    barraDeVida.value = vidaActual;
}

private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("Enemigo"))
    {
        RecibirDanio(20); // Aplica 30 puntos de daño
        if (audioSourceDaño != null && sonidoDaño != null)
        {
            audioSourceDaño.PlayOneShot(sonidoDaño);
        }
        StartCoroutine(MostrarPanelImpacto());
    }
    else if (other.gameObject.CompareTag("Caida"))
    {
        if (audioSourceCaida != null && sonidoCaida != null)
        {
            audioSourceCaida.PlayOneShot(sonidoCaida);
        }
        RecibirDanio(35); // Aplica 50 puntos de daño al caer
        StartCoroutine(MostrarPanelImpacto());
    }
    else if (other.gameObject.CompareTag("Curativo"))
    {
        Curar(25); // Curar 10 puntos de vida
        Destroy(other.gameObject); // Destruir el objeto curativo
    }
}

private IEnumerator MostrarPanelImpacto()
{
    panelImpacto.SetActive(true);
    yield return new WaitForSeconds(0.2f);
    panelImpacto.SetActive(false);
}
```

```
    void ReiniciarPosicionJugador()
    {
        if (characterController != null)
        {
            characterController.enabled = false; // Deshabilitar el CharacterController para mover al
jugador
        }
        transform.position = posicionInicial; // Reiniciar la posición del jugador
        transform.rotation = rotacionInicial; // Reiniciar la rotación del jugador
        if (characterController != null)
        {
            characterController.enabled = true; // Habilitar el CharacterController después de mover
al jugador
        }
    }

    void CargarEscenaGameOver()
    {
        SceneManager.LoadScene("GAME OVER");
    }

}
```

Código para el canvas de Creditos:

```
using UnityEngine;
using UnityEngine.SceneManagement;


public class Youwin : MonoBehaviour
{
    public void regresar()
    {
        SceneManager.LoadScene("Menu");
    }

    public void Salir()
    {
        Debug.Log("SALIENDO.....");
        Application.Quit();
    }
    void Start()
    {
        // Asegurarse de que el cursor esté visible y desbloqueado
        Cursor.visible = true;
```

```
        Cursor.lockState = CursorLockMode.None;
    }

}
```

Código para el Menu principal:

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

public class codigo : MonoBehaviour
{
    [Header("Options")]
    public Slider VolumenMax;
    public Slider VolumenMin;
    public Toggle mute;
    public AudioMixer mixer;
    public AudioSource fxSource;  // New
    public AudioClip clickSound;  // New

    [Header("Panels")]
    public GameObject panelprincipal;
    public GameObject panelopciones;
    public GameObject paneldejuego;

    private void Awake()
    {
        VolumenMax.onValueChanged.AddListener(ChangeVolumenMax);
        VolumenMin.onValueChanged.AddListener(ChangeVolumenMin);
        mute.onValueChanged.AddListener(ToggleMute); // Listen to toggle changes
    }

    public void OpenPanel(GameObject panel)
    {
        panelprincipal.SetActive(false);
        panelopciones.SetActive(false);
        paneldejuego.SetActive(false);

        panel.SetActive(true);
        PlaySoundButton();
    }
```

```
public void Jugar1()
{
    SceneManager.LoadScene("Nivel1");
}

public void Jugar2()
{
    SceneManager.LoadScene("Nivel2");
}

public void Jugar3()
{
    SceneManager.LoadScene("Nivel3");
}

public void Salir()
{
    Debug.Log("SALIENDO.....");
    Application.Quit();
}

public void ChangeVolumenMax(float v)
{
    mixer.SetFloat("VolumenMaster", v);
}

public void ChangeVolumenMin(float v)
{
    mixer.SetFloat("VolumenMax", v);
}

public void ToggleMute(bool isMuted)
{
    float volume = isMuted ? -80 : 0; // Mute volume (-80dB is effectively muted)
    mixer.SetFloat("VolumenMaster", volume);
}

public void PlaySoundButton()  // New
{
    fxSource.PlayOneShot(clickSound);
}
```

Conclusión

"El que Busca Encuentra" es un juego de aventuras vibrante y emocionante que desafía a los jugadores a explorar niveles llenos de color, recolectar monedas y superar obstáculos únicos. A través de su diseño de niveles ingenioso, el juego ofrece una experiencia de juego rica y envolvente.

Los objetivos claramente definidos permiten a los jugadores enfocarse en la recolección de monedas y la preservación de la vida mientras enfrentan desafíos crecientes. Las mecánicas de juego, que incluyen movimientos básicos y la habilidad de saltar, son fáciles de entender, pero difíciles de dominar, manteniendo a los jugadores comprometidos.

El diseño de niveles progresivamente difícil y los diversos activos, desde el esqueleto robótico del jugador hasta los complejos obstáculos, crean un mundo intrigante y estimulante. Los jugadores son recompensados por su habilidad y perseverancia, recibiendo corazones para restaurar vida y avanzando hacia la meta final de cada nivel para encontrar tesoros escondidos.

El juego también incorpora elementos de interfaz de usuario bien pensados, como el cronómetro, la barra de vida, y los menús de pausa y felicitaciones, que mejoran la experiencia del jugador. La inclusión de efectos de sonido y música de fondo añade una capa adicional de inmersión.

Bibliografia

- Unity Asset Store - The Best Assets for Game Making. (2024). @UnityAssetStore; Unity Asset Store. https://assetstore.unity.com/
- POLY STYLE - Platformer Starter Pack. (2024). @UnityAssetStore; Unity Asset Store. https://assetstore.unity.com/packages/3d/environments/poly-style-platformer-starter-pack-284167
- POLY STYLE - Fantasy Treasure Chest. (2021). @UnityAssetStore; Unity Asset Store. https://assetstore.unity.com/packages/3d/props/poly-style-fantasy-treasure-chest-280107
- Starter Assets - ThirdPerson | Updates in new CharacterController package. (2023). @UnityAssetStore; Unity Asset Store. https://assetstore.unity.com/packages/essentials/starter-assets-thirdperson-updates-in-new-charactercontroller-pa-196526
- ChatGPT. (2024). Chatgpt.com. https://chatgpt.com/c/1a827eec-69d2-45e7-962a-d7e983b47427