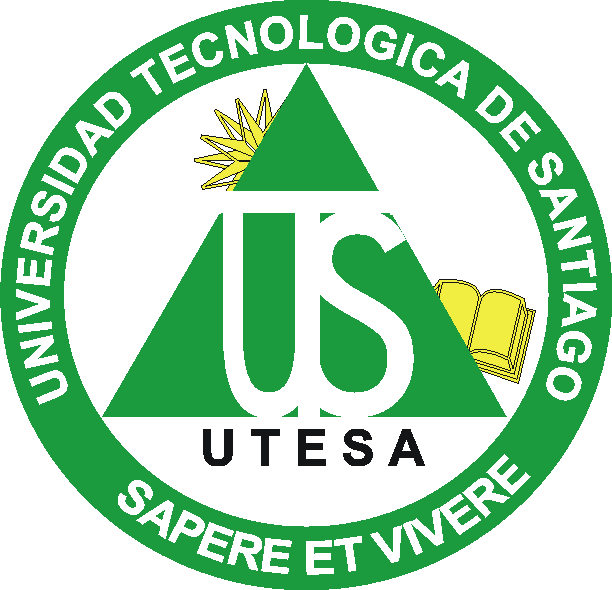
**Universidad Tecnológica de Santiago (UTESA)**



* **Estudiantes:**

-Edwin Acevedo Gómez (1-18-2709)

-Georges Gil (1-18-2363)

-Raymond Estrella (1-18-8353)

* **Asignatura**:

Programación de Videojuegos

* **Maestro**:

Iván Mendoza

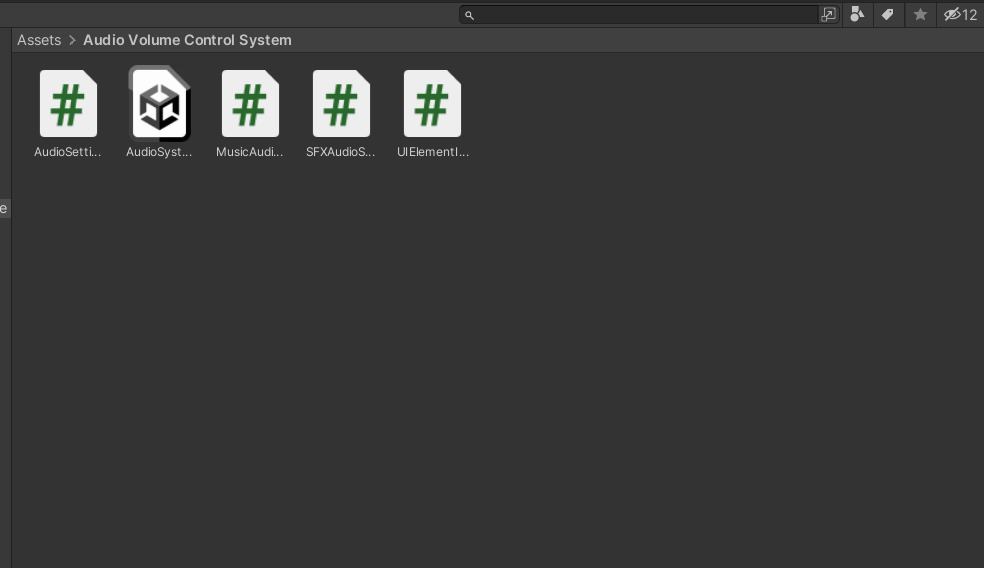
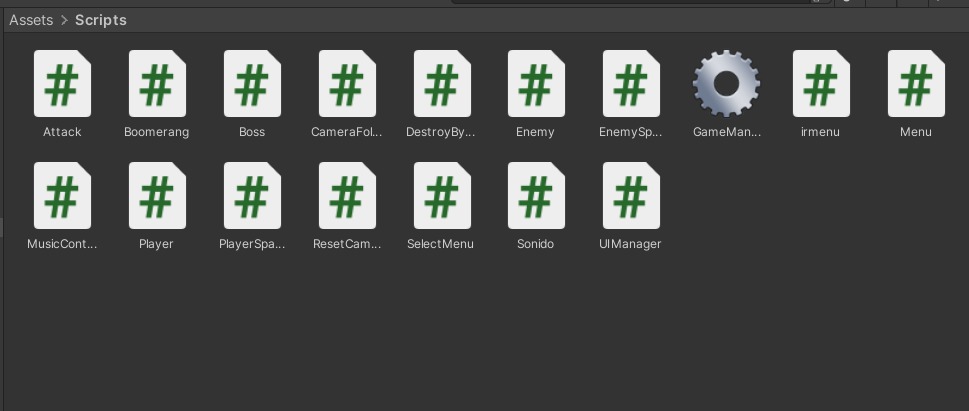
* **Tema**:

Capítulo 3- Desarrollo (Entrega Final)

* **Fecha de entrega:**

18/08/2022

**Capítulo III: Desarrollo**

******3.1 Capturas de la Aplicación Scripts, Sprites, Prefabs e imágenes) Capturas de la aplicación:**

**Script Atack:** sirve para que los personajes ataquen, tanto el jugador como también los enemigos

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**public class Attack : MonoBehaviour {**

**public int damage;**

**// Use this for initialization**

**void Start () {**

**}**

**// Update is called once per frame**

**void Update () {**

**}**

**private void OnTriggerEnter(Collider other)**

**{**

**Enemy enemy = other.GetComponent<Enemy>();**

**Player player = other.GetComponent<Player>();**

**if(enemy != null)**

**{**

**enemy.TookDamage(damage);**

**}**

**if (player != null)**

**{**

**player.TookDamage(damage);**

**}**

**}**

**}**

**Script Camerafollow.cs:** esto controla el movimiento de las cámaras, para que sigua el personaje y también para la intro donde muestra también el texto del prologo

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraFollow : MonoBehaviour {

public float xMargin = 1f; // Distance in the x axis the player can move before the camera follows.

//public float yMargin = 1f; // Distance in the y axis the player can move before the camera follows.

public float xSmooth = 8f; // How smoothly the camera catches up with it's target movement in the x axis.

public float ySmooth = 8f; // How smoothly the camera catches up with it's target movement in the y axis.

public Vector2 maxXAndY; // The maximum x and y coordinates the camera can have.

public Vector2 minXAndY; // The minimum x and y coordinates the camera can have.

private Transform m\_Player; // Reference to the player's transform.

private void Awake()

{

// Setting up the reference.

m\_Player = GameObject.FindGameObjectWithTag("Player").transform;

}

private bool CheckXMargin()

{

// Returns true if the distance between the camera and the player in the x axis is greater than the x margin.

return (transform.position.x - m\_Player.position.x) < xMargin;

}

//private bool CheckYMargin()

//{

// Returns true if the distance between the camera and the player in the y axis is greater than the y margin.

//return Mathf.Abs(transform.position.y - m\_Player.position.y) > yMargin;

//}

private void Update()

{

TrackPlayer();

}

private void TrackPlayer()

{

// By default the target x and y coordinates of the camera are it's current x and y coordinates.

float targetX = transform.position.x;

float targetY = transform.position.y;

// If the player has moved beyond the x margin...

if (CheckXMargin())

{

// ... the target x coordinate should be a Lerp between the camera's current x position and the player's current x position.

targetX = Mathf.Lerp(transform.position.x, m\_Player.position.x, xSmooth \* Time.deltaTime);

}

// If the player has moved beyond the y margin...

//if (CheckYMargin())

//{

// ... the target y coordinate should be a Lerp between the camera's current y position and the player's current y position.

//targetY = Mathf.Lerp(transform.position.y, m\_Player.position.y, ySmooth \* Time.deltaTime);

//}

// The target x and y coordinates should not be larger than the maximum or smaller than the minimum.

targetX = Mathf.Clamp(targetX, minXAndY.x, maxXAndY.x);

//targetY = Mathf.Clamp(targetY, minXAndY.y, maxXAndY.y);

// Set the camera's position to the target position with the same z component.

transform.position = new Vector3(targetX, transform.position.y, transform.position.z);

}

}

**Script Destroybytime.cs:** Destruye el gameObject luego de cierto tiempo, por ejemplo luego de que la vida llegue a cero.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DestroyByTime : MonoBehaviour {

public float destroyTime;

// Use this for initialization

void Start () {

Destroy(gameObject, destroyTime);

}

// Update is called once per frame

void Update () {

}

}

**Script Enemy.cs:** Este escrip contiene los códigos necesarios para que el enemigo se mueva ataque y cuando resiva daño le afecte

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Enemy : MonoBehaviour {

public float maxSpeed;

public float minHeight, maxHeight;

public float damageTime = 0.5f;

public int maxHealth;

public float attackRate = 1f;

public string enemyName;

public Sprite enemyImage;

public AudioClip collisionSound, deathSound;

public GameObject[] dropItem;

private static int chanceToDropItem = 0;

private int currentHealth;

private float currentSpeed;

private Rigidbody rb;

protected Animator anim;

private Transform groundCheck;

private bool onGround;

protected bool facingRight = false;

private Transform target;

protected bool isDead = false;

private float zForce;

private float walkTimer;

private bool damaged = false;

private float damageTimer;

private float nextAttack;

private AudioSource audioS;

public void Maxh(int heal){

if(maxHealth >= 1){

maxHealth = maxHealth - heal;

}

print(heal);

}

// Use this for initialization

void Start () {

rb = GetComponent<Rigidbody>();

anim = GetComponent<Animator>();

groundCheck = transform.Find("GroundCheck");

target = FindObjectOfType<Player>().transform;

currentHealth = maxHealth;

audioS = GetComponent<AudioSource>();

}

// Update is called once per frame

void Update () {

onGround = Physics.Linecast(transform.position, groundCheck.position, 1 << LayerMask.NameToLayer("Ground"));

anim.SetBool("Grounded", onGround);

anim.SetBool("Dead", isDead);

if (!isDead)

{

facingRight = (target.position.x < transform.position.x) ? false : true;

if (facingRight)

{

transform.eulerAngles = new Vector3(0, 180, 0);

}

else

{

transform.eulerAngles = new Vector3(0, 0, 0);

}

}

if(damaged && !isDead)

{

damageTimer += Time.deltaTime;

if(damageTimer >= damageTime)

{

damaged = false;

damageTimer = 0;

}

}

walkTimer += Time.deltaTime;

}

private void FixedUpdate()

{

if (!isDead)

{

Vector3 targetDitance = target.position - transform.position;

float hForce = targetDitance.x / Mathf.Abs(targetDitance.x);

if(walkTimer >= Random.Range(1f, 2f))

{

zForce = Random.Range(-1, 2);

walkTimer = 0;

}

if(Mathf.Abs(targetDitance.x) < 1.5f)

{

hForce = 0;

}

if(!damaged)

rb.velocity = new Vector3(hForce \* currentSpeed, 0, zForce \* currentSpeed);

anim.SetFloat("Speed", Mathf.Abs(currentSpeed));

if(Mathf.Abs(targetDitance.x) < 1.5f && Mathf.Abs(targetDitance.z) < 1.5f && Time.time > nextAttack)

{

anim.SetTrigger("Attack");

currentSpeed = 0;

nextAttack = Time.time + attackRate;

}

}

rb.position = new Vector3

(

rb.position.x,

rb.position.y,

Mathf.Clamp(rb.position.z, minHeight, maxHeight));

}

public void TookDamage(int damage)

{

if (!isDead)

{

damaged = true;

currentHealth -= damage;

anim.SetTrigger("HitDamage");

PlaySong(collisionSound);

FindObjectOfType<UIManager>().UpdateEnemyUI(maxHealth, currentHealth, enemyName, enemyImage);

if(currentHealth <= 0)

{

isDead = true;

chanceToDropItem+=3;

int random = Random.Range(0, 100);

//if random value is less than chance of drop and length of our items

if(random < chanceToDropItem && dropItem.Length > 0)

{

//create a random item between 0 index and our drop item list where ever enemy is going to die

Instantiate(dropItem[Random.Range(0, dropItem.Length)], transform.position, Quaternion.identity);

//reset chance of drop item back to 0 again

chanceToDropItem = 0;

}

rb.AddRelativeForce(new Vector3(3, 5, 0), ForceMode.Impulse);

PlaySong(deathSound);

}

}

}

public void DisableEnemy()

{

gameObject.SetActive(false);

}

void ResetSpeed()

{

currentSpeed = maxSpeed;

}

public void PlaySong(AudioClip clip)

{

audioS.clip = clip;

audioS.Play();

}

}

**Script EnemySpawn:** Este script contiene los condigos necesarios para que el enemigo se genere de manera aleatoria en distintos lugares del background

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemySpawn : MonoBehaviour {

public float maxZ, minZ;

public GameObject[] enemy;

public int numberOfEnemies;

public float spawnTime;

private int currentEnemies;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if(currentEnemies >= numberOfEnemies)

{

int enemies = FindObjectsOfType<Enemy>().Length;

if(enemies <= 0)

{

FindObjectOfType<ResetCameraScript>().Activate();

gameObject.SetActive(false);

}

}

}

void SpawnEnemy()

{

bool positionX = Random.Range(0, 2) == 0 ? true : false;

Vector3 spawnPosition;

spawnPosition.z = Random.Range(minZ, maxZ);

if (positionX)

{

spawnPosition = new Vector3(transform.position.x + 10, 0, spawnPosition.z);

}

else

{

spawnPosition = new Vector3(transform.position.x - 10, 0, spawnPosition.z);

}

Instantiate(enemy[Random.Range(0, enemy.Length)], spawnPosition, Quaternion.identity);

currentEnemies++;

if(currentEnemies < numberOfEnemies)

{

Invoke("SpawnEnemy", spawnTime);

}

}

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Player"))

{

GetComponent<BoxCollider>().enabled = false;

FindObjectOfType<CameraFollow>().maxXAndY.x = transform.position.x;

SpawnEnemy();

}

}

}

**Script GameManager.cs:** Este script controla el funcionamiento del juego en general ya sea saltar la intro, pasar de una escena a otra y manera el menú del volumen.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GameManager : MonoBehaviour {

public int lives;

public int characterIndex;

private static GameManager gameManager;

// Use this for initialization

void Awake () {

if(gameManager == null)

{

gameManager = this;

}

else if(gameManager != this)

{

Destroy(gameObject);

}

DontDestroyOnLoad(gameObject);

}

// Update is called once per frame

void Update () {

}

}

**Script Menu.cs**: este script junto con el de GameManager se encargan del funcionamiento optimo del menú para que sea dinamico y pueda mandarnos a la escena del gameplay.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Menu : MonoBehaviour {

// Update is called once per frame

void Update () {

if (Input.anyKeyDown)

{

LoadScene();

}

}

void LoadScene()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

}

**Script MusicControler.cs:** Este script se encargara de reproducir la canción de fondo del videojuego, dependiendo de la escena en cuestión.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MusicController : MonoBehaviour {

public AudioClip levelSong, bossSong, levelClearSong;

private AudioSource audioS;

// Use this for initialization

void Start () {

audioS = GetComponent<AudioSource>();

PlaySong(levelSong);

}

public void PlaySong(AudioClip clip)

{

audioS.clip = clip;

audioS.Play();

}

}

**Script Player.cs:** Este script se encargara de que nuestro personaje pueda moverse atacar y recibir daño.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Player : MonoBehaviour {

public float maxSpeed = 4;

public float jumpForce = 400;

public float minHeight, maxHeight;

public int maxHealth = 10;

public string playerName;

public Sprite playerImage;

public AudioClip collisionSound, jumpSound, healthItem;

private int currentHealth;

public int heal = 1;

private float currentSpeed;

private Rigidbody rb;

private Animator anim;

private Transform groundCheck;

private bool onGround;

private bool isDead = false;

private bool facingRight = true;

private bool jump = false;

private AudioSource audioS;

// Use this for initialization

void Start () {

rb = GetComponent<Rigidbody>();

anim = GetComponent<Animator>();

groundCheck = gameObject.transform.Find("GroundCheck");

currentSpeed = maxSpeed;

currentHealth = maxHealth;

audioS = GetComponent<AudioSource>();

}

// Update is called once per frame

void Update () {

onGround = Physics.Linecast(transform.position, groundCheck.position, 1 << LayerMask.NameToLayer("Ground"));

anim.SetBool("OnGround", onGround);

anim.SetBool("Dead", isDead);

if(Input.GetButtonDown("Jump") && onGround)

{

jump = true;

}

if (Input.GetButtonDown("Fire1"))

{

anim.SetTrigger("Attack");

}

}

private void FixedUpdate()

{

if (!isDead)

{

float h = Input.GetAxis("Horizontal");

float z = Input.GetAxis("Vertical");

if (!onGround)

z = 0;

rb.velocity = new Vector3(h \* currentSpeed, rb.velocity.y, z \* currentSpeed);

if (onGround)

anim.SetFloat("Speed", Mathf.Abs(rb.velocity.magnitude));

if(h > 0 && !facingRight)

{

Flip();

}

else if(h < 0 && facingRight)

{

Flip();

}

if (jump)

{

jump = false;

rb.AddForce(Vector3.up \* jumpForce);

PlaySong(jumpSound);

}

float minWidth = Camera.main.ScreenToWorldPoint(new Vector3(0, 0, 10)).x;

float maxWidth = Camera.main.ScreenToWorldPoint(new Vector3(Screen.width, 0, 10)).x;

rb.position = new Vector3(Mathf.Clamp(rb.position.x, minWidth + 1, maxWidth - 1),

rb.position.y,

Mathf.Clamp(rb.position.z, minHeight, maxHeight));

}

}

void Flip()

{

facingRight = !facingRight;

Vector3 scale = transform.localScale;

scale.x \*= -1;

transform.localScale = scale;

}

void ZeroSpeed()

{

currentSpeed = 0;

}

void ResetSpeed()

{

currentSpeed = maxSpeed;

}

public void TookDamage(int damage)

{

if (!isDead)

{

currentHealth -= damage;

anim.SetTrigger("HitDamage");

FindObjectOfType<UIManager>().UpdateHealth(currentHealth);

PlaySong(collisionSound);

if(currentHealth <= 0)

{

isDead = true;

FindObjectOfType<GameManager>().lives--;

if (facingRight)

{

rb.AddForce(new Vector3(-3, 5, 0), ForceMode.Impulse);

}

else

{

rb.AddForce(new Vector3(3, 5, 0), ForceMode.Impulse);

}

}

}

}

public void PlaySong(AudioClip clip)

{

audioS.clip = clip;

audioS.Play();

}

private void OnTriggerStay(Collider other)

{

if(other.CompareTag("Health Item"))

{

if (Input.GetButtonDown("Fire2"))

{

Destroy(other.gameObject);

anim.SetTrigger("Catching");

PlaySong(healthItem);

currentHealth = maxHealth;

FindObjectOfType<UIManager>().UpdateHealth(currentHealth);

}

}

else if(other.CompareTag("Damage"))

{

if (Input.GetButtonDown("Fire2"))

{

Destroy(other.gameObject);

anim.SetTrigger("Catching");

PlaySong(healthItem);

//currentHealth = maxHealth;

Enemy enemy = other.GetComponent<Enemy>();

if(enemy != null){

enemy.Maxh(heal);

}

//FindObjectOfType<UIManager>().UpdateHealth(currentHealth);

}

}

}

void PlayerRespawn()

{

if(FindObjectOfType<GameManager>().lives > 0)

{

isDead = false;

FindObjectOfType<UIManager>().UpdateLives();

currentHealth = maxHealth;

FindObjectOfType<UIManager>().UpdateHealth(currentHealth);

anim.Rebind();

float minWidth = Camera.main.ScreenToWorldPoint(new Vector3(0, 0, 10)).x;

transform.position = new Vector3(minWidth, 10, -4);

}

else

{

FindObjectOfType<UIManager>().UpdateDisplayMessage("Game Over");

Destroy(FindObjectOfType<GameManager>().gameObject);

Invoke("LoadScene", 2f);

}

}

void LoadScene()

{

SceneManager.LoadScene(0);

}

}

**Script PlayerSpawner.cs:** Este script se encarga de que cada vez que nuestro personaje muera vuelva a aparecer siempre y cuando le queden vidas sobrantes.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerSpawner : MonoBehaviour {

public GameObject[] player;

// Use this for initialization

void Awake () {

int index = FindObjectOfType<GameManager>().characterIndex - 1;

Instantiate(player[index], transform.position, transform.rotation);

}

}

**Script ResetCameraScript.cs:** Este script hace que la cámara vuelva a su posición inicial cuando se pasa de nivel o cuando se termina el juego y vuelve al menú principal.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ResetCameraScript : MonoBehaviour {

public void Activate()

{

GetComponent<Animator>().SetTrigger("Go");

}

void ResetCamera()

{

FindObjectOfType<CameraFollow>().maxXAndY.x = 200;

}

}

**Script SelectMenu.cs:** Este script se encarga del funcionamiento del menú para controlar el volumen de la música de fondo y el sonido de los golpes cuando el personaje principal o los enemigos se golpean.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class SelectMenu : MonoBehaviour {

public Image adamImage, axelImage;

public Animator adamAnim, axelAnim;

private Color defaultColor;

private int characterIndex;

private AudioSource audioS;

// Use this for initialization

void Start () {

characterIndex = 1;

audioS = GetComponent<AudioSource>();

defaultColor = axelImage.color;

}

// Update is called once per frame

void Update () {

if (Input.GetKeyDown(KeyCode.LeftArrow))

{

characterIndex = 1;

PlaySound();

}

else if (Input.GetKeyDown(KeyCode.RightArrow))

{

characterIndex = 1;

PlaySound();

}

if(characterIndex == 1)

{

adamImage.color = Color.yellow;

adamAnim.SetBool("Attack", true);

axelImage.color = defaultColor;

axelAnim.SetBool("Attack", false);

}

else if(characterIndex == 1)

{

axelImage.color = Color.yellow;

axelAnim.SetBool("Attack", true);

adamImage.color = defaultColor;

adamAnim.SetBool("Attack", false);

}

if (Input.GetKeyDown(KeyCode.Return))

{

FindObjectOfType<GameManager>().characterIndex = characterIndex;

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

}

void PlaySound()

{

if (!audioS.isPlaying)

{

audioS.Play();

}

}

}

**Script Sonido.cs:** Este script se encarga de reproducir en cierto momento los efectos de sonidos como son, golpes, saltos, cuando los enemigos son golpeados, cuando el jugador es golpeado y el sonido que hacen cuando mueren.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Audio;

using TMPro;

public class Sonido : MonoBehaviour

{

// Start is called before the first frame update

public GameObject window;

void Start()

{

}

// Update is called once per frame

void Update()

{

if(Input.GetKeyDown(KeyCode.V))

{

window.SetActive(!window.activeInHierarchy);

}

}

}

**Script UiManager.cs:** Este script manejas los cambas que salen en el gameplay, ya sea el nombre y la barra de vida del jugador y la de los enemigos y como son afectados cuando reciben ataques.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class UIManager : MonoBehaviour {

public Slider healthUI;

public Image playerImage;

public Text playerName;

public Text livesText;

public Text displayMessage;

public GameObject enemyUI;

public Slider enemySlider;

public Text enemyName;

public Image enemyImage;

public float enemyUITime = 4f;

private float enemyTimer;

private Player player;

// Use this for initialization

void Start () {

player = FindObjectOfType<Player>();

healthUI.maxValue = player.maxHealth;

healthUI.value = healthUI.maxValue;

playerName.text = player.playerName;

playerImage.sprite = player.playerImage;

UpdateLives();

}

// Update is called once per frame

void Update () {

enemyTimer += Time.deltaTime;

if(enemyTimer >= enemyUITime)

{

enemyUI.SetActive(false);

enemyTimer = 0;

}

}

public void UpdateHealth(int amount)

{

healthUI.value = amount;

}

public void UpdateEnemyUI(int maxHealth, int currentHealth, string name, Sprite image)

{

enemySlider.maxValue = maxHealth;

enemySlider.value = currentHealth;

enemyName.text = name;

enemyImage.sprite = image;

enemyTimer = 0;

enemyUI.SetActive(true);

}

public void UpdateLives()

{

livesText.text = "x " + FindObjectOfType<GameManager>().lives.ToString();

}

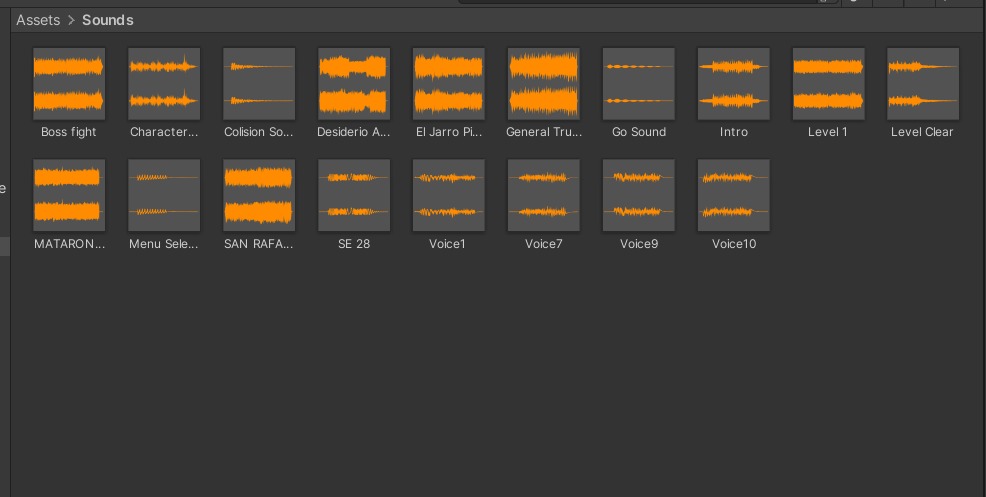
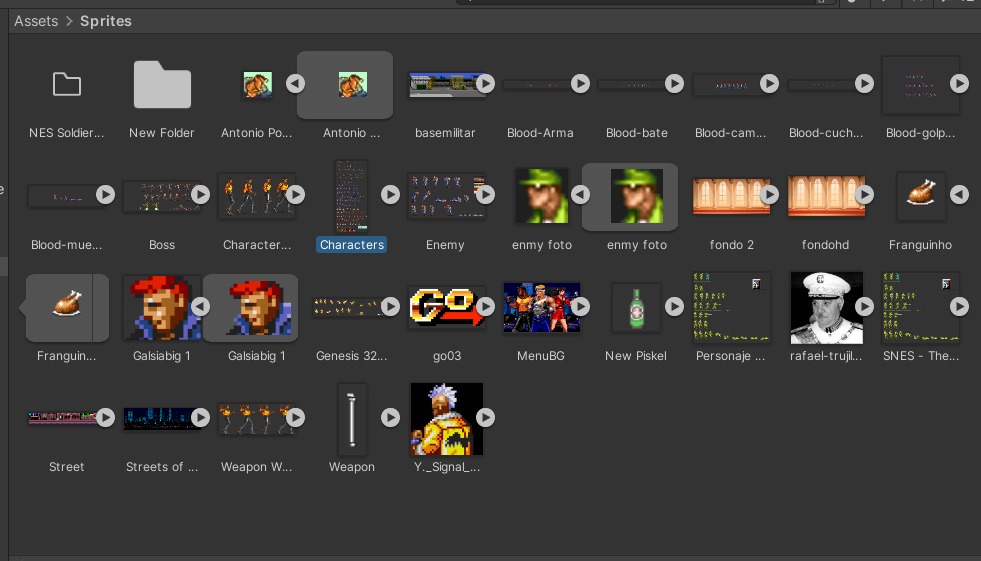
public void UpdateDisplayMessage(string message)

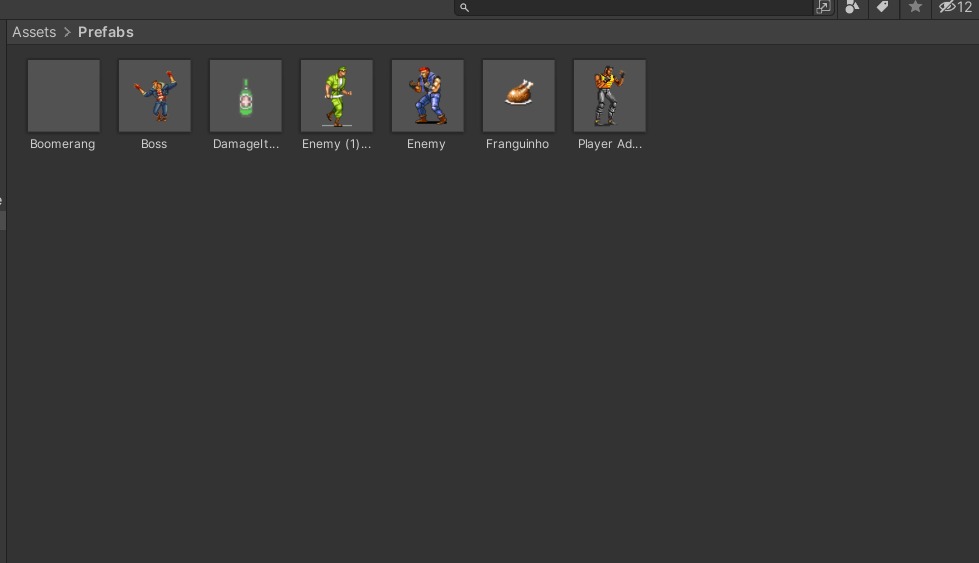
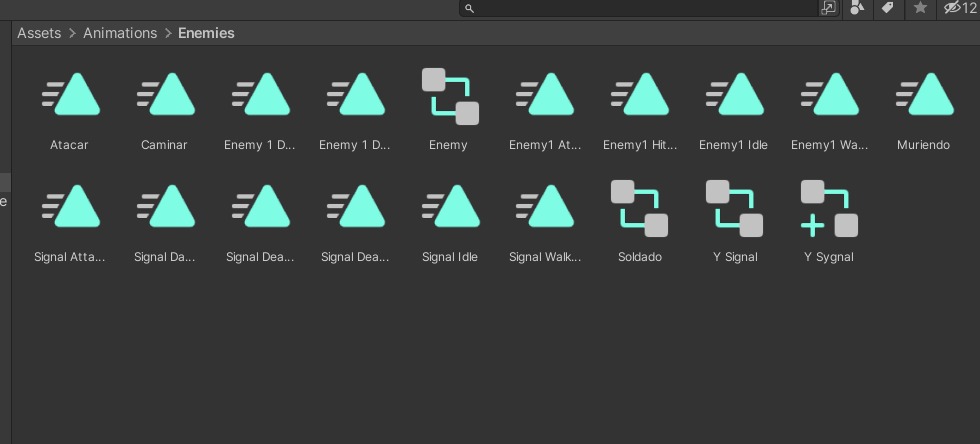
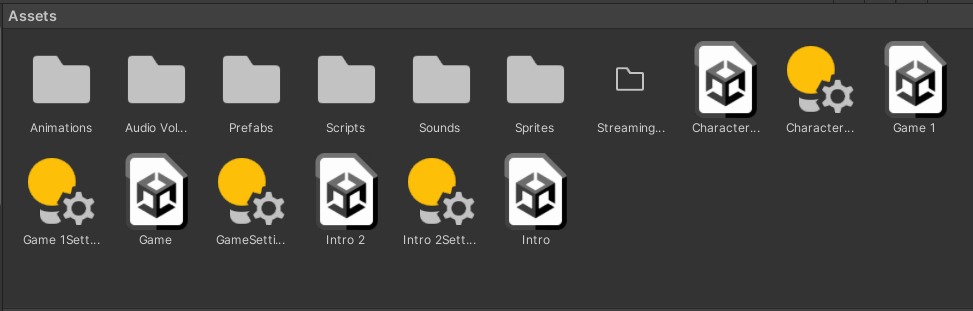
{

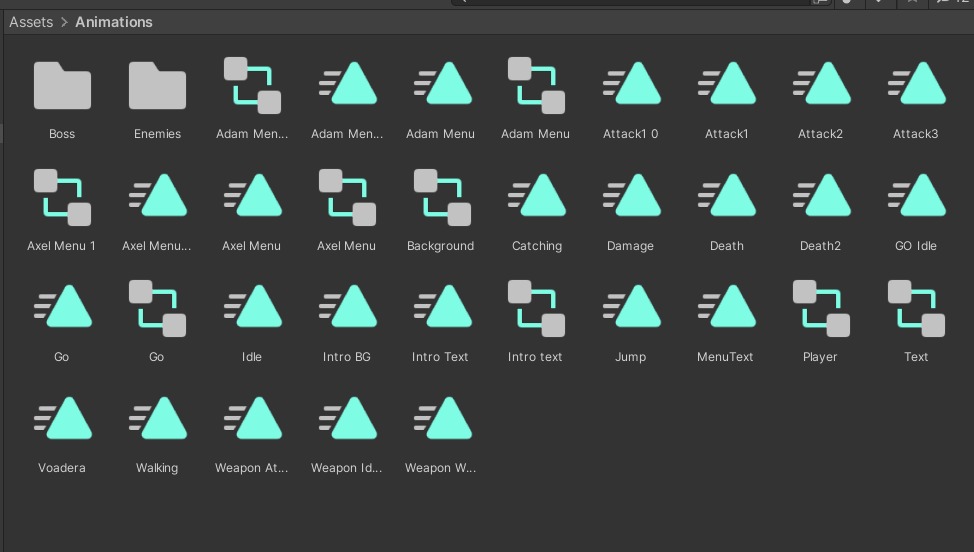
displayMessage.text = message;

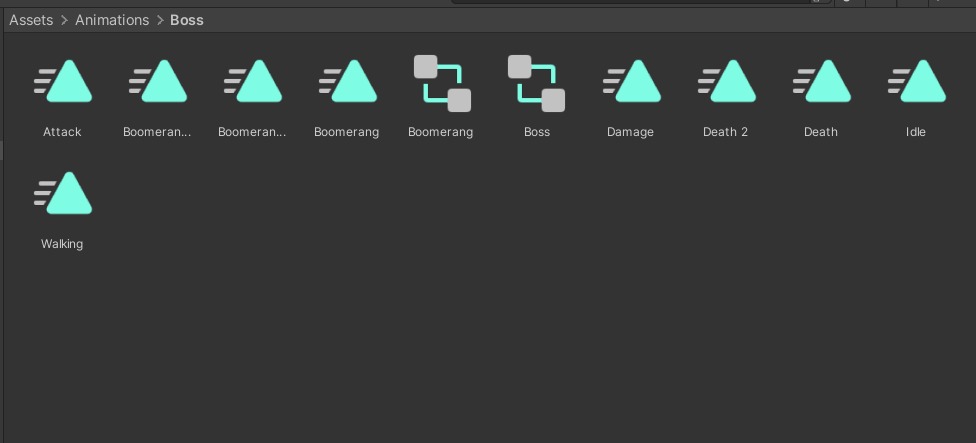
}

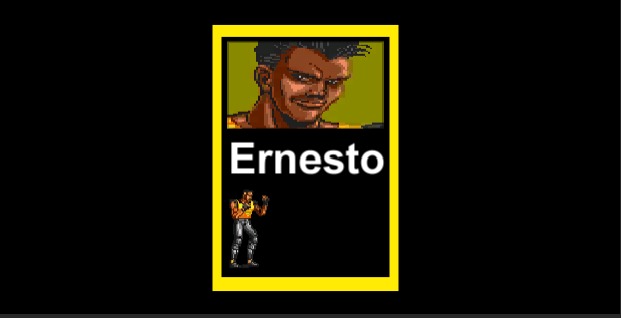
}

****

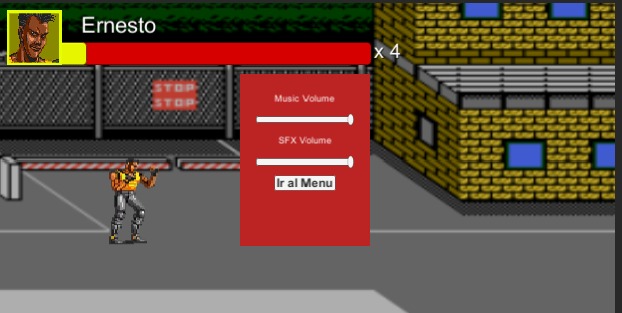
****

****

****

**Inicio del juego**

****

****

****

**Enemigo Final**

****

**3.2 Prototipos:**

Para la creación de dicho juego fue necesario crear 2 prototipos como son los prototipos (0.001P) y el (0.002P) los cuales fueron de gran ayuda para entender más los procesos y así lograr el mejor desempeño del juego Final.

**3.3 Perfiles de usuarios:**

El público objetivo del videojuego abarca:

* Personas entre 10 años de edad en adelante.
* Personas con interés en sucesos históricos.
* Personas con interés en el género Beat ‘Em Up.

**3.4 Usabilidad:**

Controles:

* + Se podrá mover el personaje mediante las teclas de desplazamiento o por las teclas (W) (A) (S) (D).
  + La barra de Espacio sirve para saltar.
  + Clic derecho o Alt para atacar.
  + Tecla (V) para abrir/cerrar menú volumen.
  + Clic izquierdo para usar un objeto.

**3.5 Test:**

|  |  |
| --- | --- |
| Sexo | Hombre |
| Edad | 22 |
| Nivel de estudios | Universitario |

Test 1:

Individuo 1:

**Resultados**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **3** |
| Dificultad | **2** |
| Control de personaje | **5** |
| Guía de usuario | **3** |
| Información proporcionada por el juego | **2** |
| Diseño visual | **4** |
| Coherencia | **4** |

Individuo 2:

|  |  |
| --- | --- |
| Sexo | Mujer |
| Edad | 30 |
| Nivel de estudios | Universitario |

**Resultados**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **3** |
| Dificultad | **3** |
| Control de personaje | **4** |
| Guía de usuario | **2** |
| Información proporcionada por el juego | **3** |
| Diseño visual | **4** |
| Coherencia | **2** |

Individuo 3:

|  |  |
| --- | --- |
| Sexo | Hombre |
| Edad | 17 |
| Nivel de estudios | Bachiller |

**Resultados**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **4** |
| Dificultad | **1** |
| Control de personaje | **4** |
| Guía de usuario | **3** |
| Información proporcionada por el juego | **2** |
| Diseño visual | **3** |
| Coherencia | **4** |

**Resultados test 1:**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **10/3 = 3.3** |
| Dificultad | **6/3 = 2** |
| Control de personaje | **13/3 = 4.3** |
| Guía de usuario | **8/3 = 2.7** |
| Información proporcionada por el juego | **7/3 = 2.3** |
| Diseño visual | **11/3 = 3.7** |
| Coherencia | **10/3 = 3.3** |

Test 2:

Individuo 1:

|  |  |
| --- | --- |
| Sexo | Hombre |
| Edad | 19 |
| Nivel de estudios | Universitario |

**Resultados**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **3** |
| Dificultad | **5** |
| Control de personaje | **4** |
| Guía de usuario | **4** |
| Información proporcionada por el juego | **4** |
| Diseño visual | **3** |
| Coherencia | **2** |

|  |  |
| --- | --- |
| Sexo | Hombre |
| Edad | 28 |
| Nivel de estudios | Universitario |

Individuo 2:

**Resultados**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **5** |
| Dificultad | **3** |
| Control de personaje | **5** |
| Guía de usuario | **3** |
| Información proporcionada por el juego | **3** |
| Diseño visual | **4** |
| Coherencia | **4** |

**Resultados test 2:**

|  |  |
| --- | --- |
| **Puntos a evaluar** | **Puntuación** |
| Jugabilidad | **8/2 = 4** |
| Dificultad | **8/2 = 4** |
| Control de personaje | **9/2 = 4.5** |
| Guía de usuario | **7/2 = 3.5** |
| Información proporcionada por el juego | **7/2 = 3.5** |
| Diseño visual | **7/2 = 3.5** |
| Coherencia | **6/2 = 3** |

**3.6 Versiones de la aplicación**

The Fall of Trujillo es la versión (1.001F).

-----------------------------------------------------------------------------------------------------

**Link de GitHub:**

<https://github.com/GeorgesGil/Proyecto-Final>

**Link de Itch.Io:**

<https://georgesgil.itch.io/the-fall-of-trujillo-windows>

<https://georgesgil.itch.io/the-fall-of-trujillo-web>