17-4-2020

Videojuego 3D

ACTIVIDAD 2 FCT

José Ángel Atoche Jovacho
2º DAM

Objetivos

-Crear mapa
-Crear Jugador
-Controles sencillos
-Pantalla de muerte
-Sonidos
-Varios enemigos
-Historia
-Final
-Poder reintentar

Razones del diseño

Quizá hay ciertas dudas del razonamiento que hay detrás del diseño tan sobrio que tiene el juego, a penas sin colores, o tan liso, las razones son muchas.

Entre ellas, crear un juego más detallado consume más gráfica, lo primero que se debe pensar en estos casos es que quizá las gráficas de quienes lo vayan a probar no tienen por qué ser muy potentes entonces pensando en este concepto, a más simpleza y menos polígonos mejor.

Posteriormente es por la historia, la historia, como después trataré, es como la cognición o representación conceptual de una vida pasar, y como a todos nos pasa, de nuestras vidas nunca recordamos detalles o colores de las cosas, y esta es la representación artística de este efecto.

Cabe señalar, y bajo mi punto de vista, que, aunque carezca de colores o texturas, el trabajo no es menor o inferior, pienso que buscar modelos en internet en 3d y colocarlos conlleva el mismo trabajo posiblemente que el diseño de este mapa que he diseñado, al fin y al cabo, se trata de lo mismo, coger elementos en 3D y arrastrarlos al mapa y colocarlos en la posición deseada.

He optado también por usar este recurso ya que facilita las cosas a la hora de la creación del escenario, ya que, si buscamos modelos en 3D de, por ejemplo, zonas de una casa, hay muchos que no combinan entre sí, por ser diversos tipos de muebles, o texturas, y quedarían muy muy extraños, por ello la mejor solución es representarlos sin colores, así la unión de estos modelos es bastante más sencilla

Historia

La historia quizá es lo más complejo de explicar o exponer, la historia cuenta el relato de una vida de una persona, una vida que no tiene nombre ni cara, esta historia se intenta centrar en el camino de una vida hasta llegar a la muerte, el mapa está diseñado sin detalles tal como lo representa la cognición cerebral a no recordar ciertos detalles de la vida.

La historia se relata en un segundo plano dentro del propio juego, ya que es un juego que resulta demasiado arcade, podemos encontrar ciertos detalles y notas que crean la historia del jugador para que entremos, solo si queremos, en la historia de este.

¿Por qué está así diseñado? son dos motivos simples:

1. Solo recordamos los humanos lo que queremos recordar, los “traumas” siempre se quedan ahí y se nos ponen por delante, los traumas en este caso son los enemigos
2. Así el jugador querrá jugar al juego en varias ocasiones para descubrir qué le pasó al protagonista, además es una forma de hacer los videojuegos hoy en día, una pseudo narrativa que se demuestra más por detalles en el juego que por medio de diálogos en este.

Esta historia, y este juego, sin lugar a dudas son caóticos, diseñados exclusivamente como un pequeño juego indie con principio y final con mecánicas que deben ser simples para que el jugador sea capaz de jugar con facilidad, y una historia rápida que pueda hacer que el jugador no pierda la paciencia si muere

La historia evoluciona junto al jugador, nuestro personaje se hace cada vez más grande en cuanto más avanzamos por el pasillo de la vida, la representación de nuestro camino escogido, un pasillo que representa nuestro destino y que no tenemos escapatoria de este, y unos problemas que siempre se interponen entre nosotros y nuestra meta, problemas, que son representado por medio de diversos colores en los enemigos.

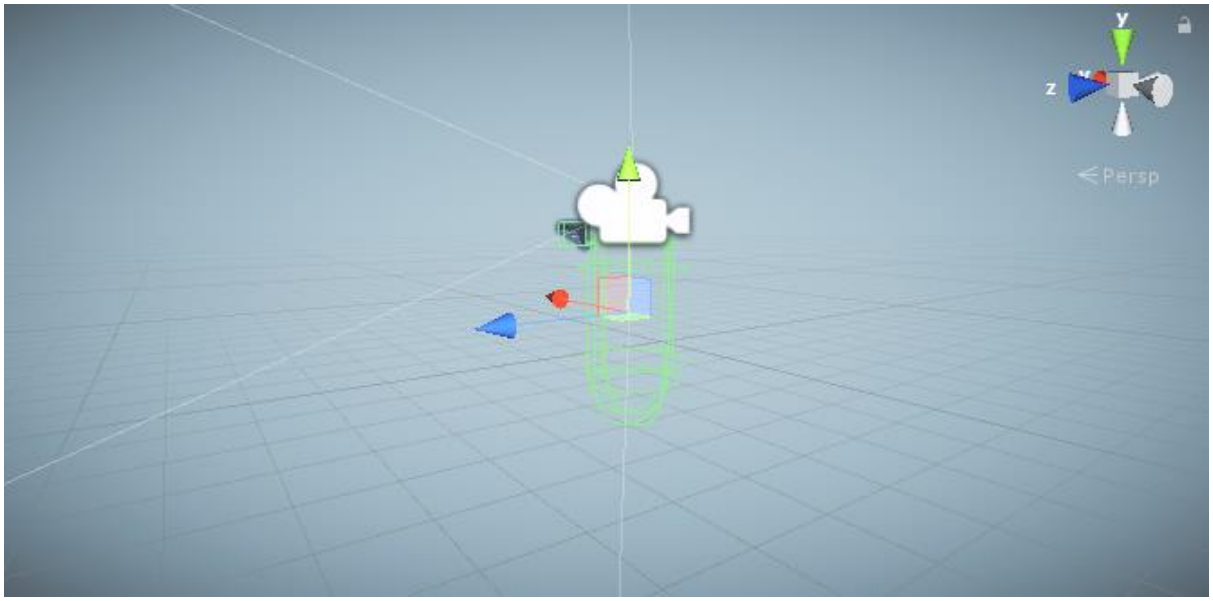
Rojos-muerte
azules- soledad
negros- abandono
amarillos- peligros
verdes- dinero
blancos- drogas
grises- enfermedades

Debemos derrotar a estos enemigos de la vida para poder avanzar por esta, y es eso lo que quiere narrar la historia desde un segundo plano, por ello cuando te derrota un enemigo aparece un mensaje señalando la razón concreta de la muerte:

Rojos-" Te han asesinado"
azules- "Moriste triste y solo"
negros- "No soportaste la muerte"
amarillos- "has sufrido un accidente"
verdes- "Tus deudas se pagan caras"
blancos- "Muerto por sobredosis"
grises- "Una enfermedad ha podido contigo"

Personaje Principal

Necesito un personaje principal para este juego, como su narrativa es la historia de una persona lo mejor será hacer el juego en 1 Persona

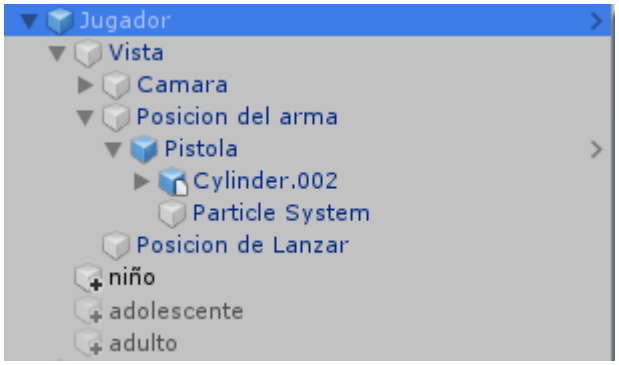


Lo más simple y recomendable es no hacer un modelo en 3d para el personaje, o molestaría con tu vista, y ya que no lo vas a ver nunca pues eso que ahorras.

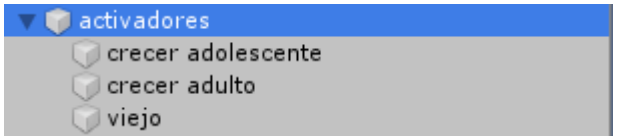
Este personaje lleva un arma para disparar a los enemigos, y un collider para tanto su movimiento como para recibir daños, el código es el siguiente:

```
1 referencia
public class ControladorPersonaje : MonoBehaviour
{
    1 referencia
    public static ControladorPersonaje Instance { get; protected set; }

    public Camera MainCamera;

    public Transform posicionDeCamara;
    public Transform posicionDeArma;

    public float sensibilidadRaton = 5;
    public float velocidadConstante = 5.0f;


    private float velocidadVertical = 0.0f;
    private float anguloVertical;
    private float anguloHorizontal;
    public float velocidadActual = 0.0f;

    private CharacterController controlador;

    0 referencias
    private void Start()
    {
        Instance = this;
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;

        MainCamera.transform.SetParent(posicionDeCamara, false);
        MainCamera.transform.localPosition = Vector3.zero;
        MainCamera.transform.localRotation = Quaternion.identity;
        controlador = GetComponent<CharacterController>();

        anguloVertical = 0.0f;
        anguloHorizontal = transform.localEulerAngles.y;
    }
```

```
/// <summary>
/// Permite el movimiento del personaje
/// </summary>
0 referencias
private void Update()
{
    velocidadActual = 0;
    Vector3 movimiento = Vector3.zero;

    movimiento = new Vector3(Input.GetAxisRaw("Horizontal"), 0, Input.GetAxisRaw("Vertical"));
    if (movimiento.sqrMagnitude > 1.0f)
    {
        movimiento.Normalize();
    }

    movimiento *= (velocidadConstante * Time.deltaTime);

    movimiento = transform.TransformDirection(movimiento);
    controlador.Move(movimiento);

    anguloHorizontal += Input.GetAxis("Mouse X") * sensibilidadRaton;

    anguloHorizontal += anguloHorizontal > 360 ? -360f : 360f;

    Vector3 anguloActual = transform.localEulerAngles;
    anguloActual.y = anguloHorizontal;
    transform.localEulerAngles = anguloActual;

    float giro = -Input.GetAxis("Mouse Y");
    giro *= sensibilidadRaton;
    anguloVertical = Mathf.Clamp(giro + anguloVertical, -89.0f, 89.0f);
    anguloActual = posicionDeCamara.transform.localEulerAngles;
    anguloActual.x = anguloVertical;
    posicionDeCamara.transform.localEulerAngles = anguloActual;

    velocidadActual = movimiento.magnitude / (velocidadConstante * Time.deltaTime);

    gravedad();
}

//Establece una gravedad continua hacia abajo
1 referencia
private void gravedad()
{
    velocidadVertical = velocidadVertical - 10.0f * Time.deltaTime;
    if (velocidadVertical < -10.0f)
    {
        velocidadVertical = -10.0f;
    }

    var verticalMove = new Vector3(0, velocidadVertical * Time.deltaTime, 0);
    var flag = controlador.Move(verticalMove);
    if ((flag & CollisionFlags.Below) != 0)
    {
        velocidadVertical = 0;
    }
}
```



Importante señalar que el personaje tiene 3 hitboxes, diseñadas para cuando crece conforme avanza



y tienen los siguientes scripts respectivamente:

```
public GameObject nino;
public GameObject adolescente;
bool one = false;
public CharacterController character;


0 referencias
private void OnTriggerEnter(Collider other)
{

    if (!one && other.tag == "Player")
    {
        character.height = 1.2f;
        character.center = new Vector3(0, 0.5f, 0);
        adolescente.active = true;
        nino.active = false;
        one = true;

    }
```

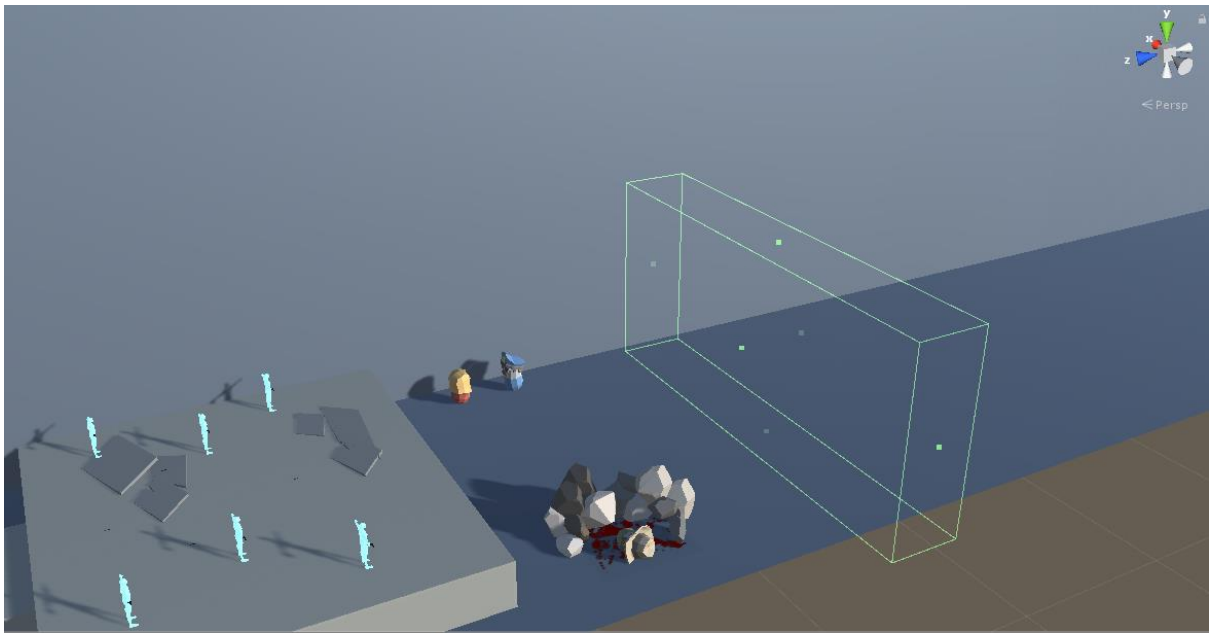
Este primero solo hace que el personaje crezca un poco y le pone una hitbox más grande

```
public GameObject adulto;
public GameObject adolescente;
bool one = false;
public CharacterController character;


0 referencias
private void OnTriggerEnter(Collider other)
{
    if (!one && other.tag == "Player")
    {
        character.height = 1.5f;
        character.center = new Vector3(0, 0, 0);

        adulto.active = true;
        adolescente.active = false;

        one = true;
    }
```

Este segundo hace que el personaje crezca un poco más y le pone una hitbox aún más grande

```
public ControladorPersonaje controller;
bool one = false;


0 referencias
private void OnTriggerEnter(Collider other)
{
    if (!one && other.tag == "Player")
    {
        controller.velocidadConstante = 10;
        one = true;
    }
```

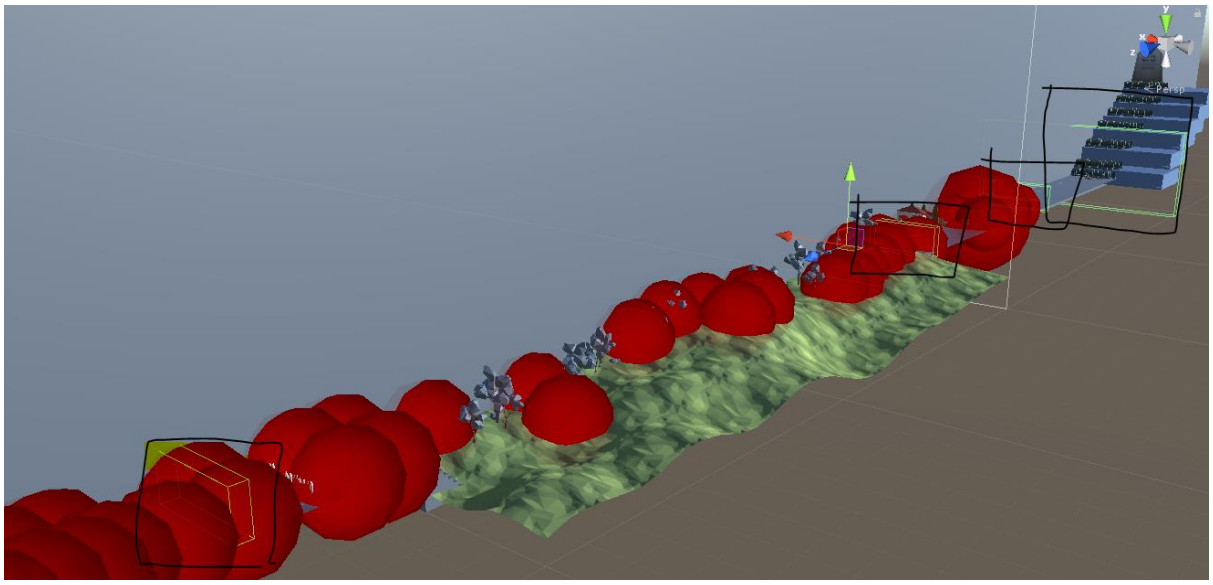
Este último solo hace al personaje más lento



Al pasar por esta hitbox se activan las diversas características programadas en los script anteriores.

En resumen… son los siguientes y de la siguiente manera:

Activadores:



Hice unos activadores para crecer y hacer que la historia siguiese, lo controlan los siguientes scripts

niño:

```
public class activar : MonoBehaviour
{

    public GameObject nino;
    public GameObject adolescente;
    bool one = false;
    public CharacterController character;


    0 referencias
    private void OnTriggerEnter(Collider other)
    {

        if (!one && other.tag == "Player")
        {
            character.height = 1.2f;
            character.center = new Vector3(0, 0.5f, 0);
            adolescente.active = true;
            nino.active = false;
            one = true;

        }
```

adolescente:

```
0 referencias
public class activar2 : MonoBehaviour
{

    public GameObject adulto;
    public GameObject adolescente;
    bool one = false;
    public CharacterController character;


    0 referencias
    private void OnTriggerEnter(Collider other)
    {
        if (!one && other.tag == "Player")
        {
            character.height = 1.5f;
            character.center = new Vector3(0, 0, 0);

            adulto.active = true;
            adolescente.active = false;

            one = true;
        }
```

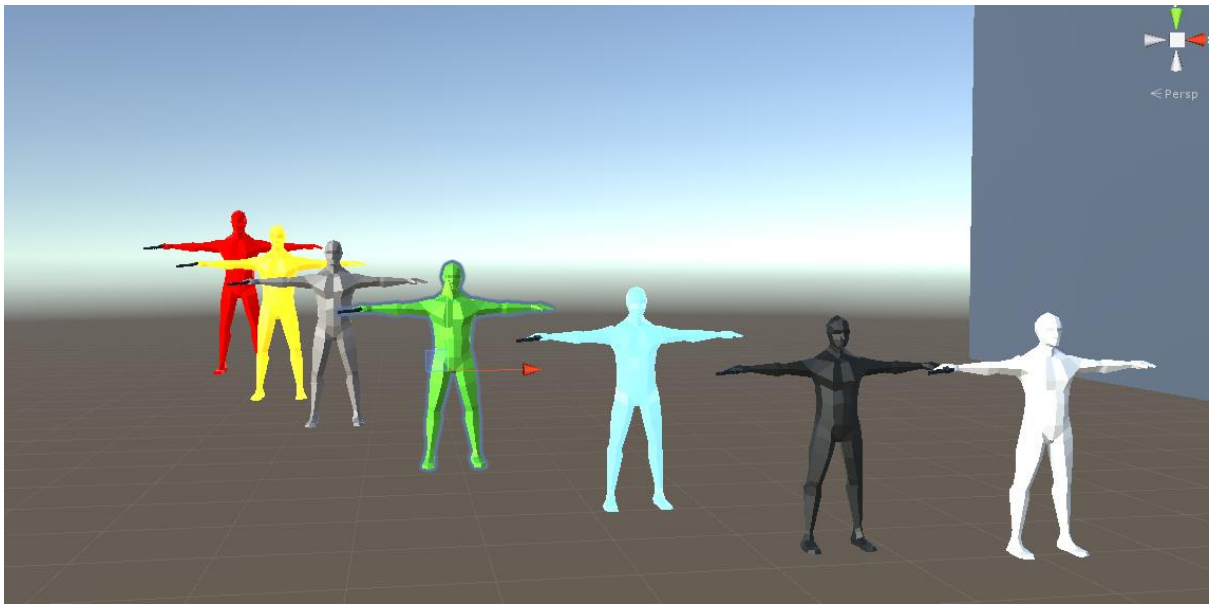
adulto:

```
public class activar3 : MonoBehaviour
{
    public ControladorPersonaje controller;
    bool one = false;


    0 referencias
    private void OnTriggerEnter(Collider other)
    {
        if (!one && other.tag == "Player")
        {
            controller.velocidadConstante = 10;
            one = true;
        }
```

Final:

```
0 referencias
public class fin : MonoBehaviour
{
    0 referencias
    private void OnTriggerEnter(Collider other)
    {


        GameController.muertescript.ModificarTexto("Moriste en el hospital al lado de tu mujer, con una sonrisa y mucho amor");


        GameController.imagen2.SetActive(true);



    }
}
```

Enemigo



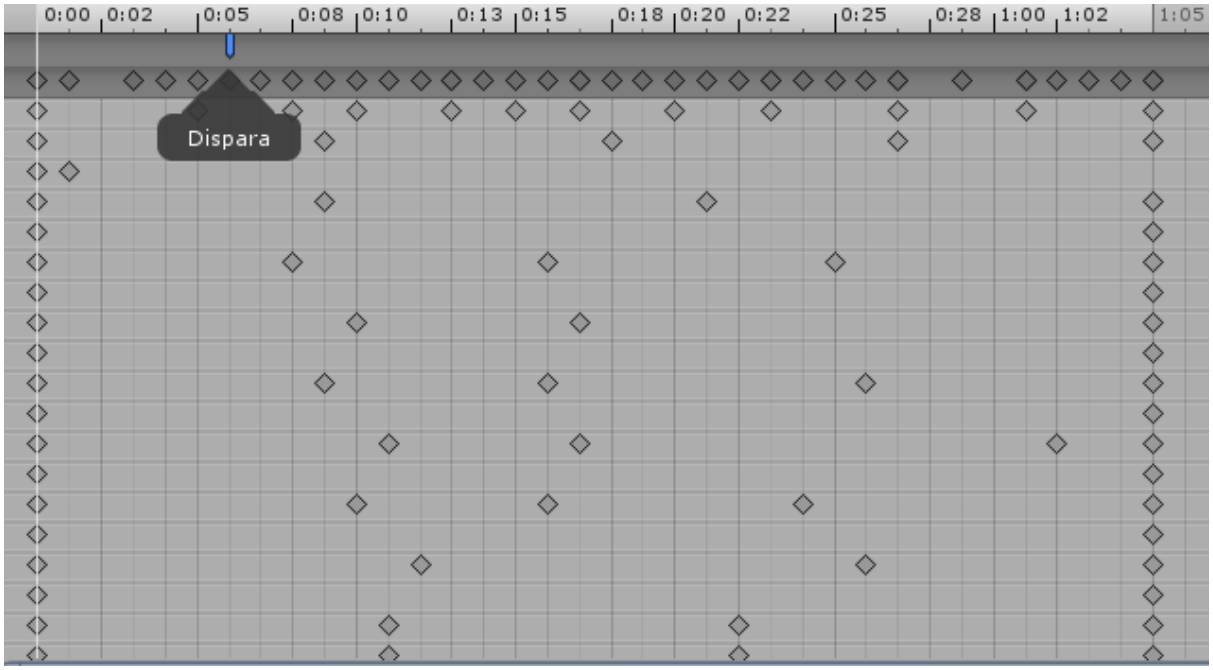
Aunque hay varios tipos de enemigos, todos comparten el mismo código interno, apuntar y disparar, tuve que diseñar algo también para que cuando muriese, cayese desplomado al suelo

```
[SelectionBase]
1 referencia
public class ControladorEnemigo : MonoBehaviour
{
    private Animator animacion;
    public bool muerto;
    public Transform posicionarma;

    0 referencias
    private void Start()
    {
        animacion = GetComponent<Animator>();

        if (posicionarma.GetComponentInChildren<ControladorArma>() != null)
        {
            posicionarma.GetComponentInChildren<ControladorArma>().suelto = false;
        }
    }

    0 referencias
    private void Update()
    {
        //Si no está muerto continuamente te mira

        if (!muerto)
        {
            transform.LookAt(new Vector3(Camera.main.transform.position.x, 0, Camera.main.transform.position.z));
        }
    }
    /// <summary>
    /// Se cae al suelo como si estuviese muerto
    /// </summary>
    1 referencia
    public void Muerte()
    {
        animacion.enabled = false;
        Rotura[] partes = GetComponentsInChildren<Rotura>();
        foreach (Rotura rotura in partes)
        {
            rotura.rigibody.isKinematic = false;
            rotura.rigibody.interpolation = RigidbodyInterpolation.Interpolate;
        }
        muerto = true;

        if (posicionarma.GetComponentInChildren<ControladorArma>() != null)
        {
            ControladorArma arma = posicionarma.GetComponentInChildren<ControladorArma>();
            arma.Soltar();
        }
    }
```

```
/// <summary>
/// Dispara siempre que no esté muerto
/// </summary>
0 referencias
public void Dispara()
{
    if (!muerto && posicionarma.GetComponentInChildren<ControladorArma>() != null)
    {
        posicionarma.GetComponentInChildren<ControladorArma>().Disparar(GetComponentInChildren<ParticleSystem>().transform.position, transform.rotation, true);
    }
}
```

De esta página cogí la animación del disparo https://www.mixamo.com/#/

El disparo se ejecuta en la animación y lo que hace es crear una bala que se dirige al jugador


0:00
0:02
0:05
0:08
0:10
0:13
0:15
0:18
0:20
0:22
0:25
0:28
1:00
1:02
1:05
Dispara

Bala

La bala es un prefab que al chocar con el enemigo genera unas partículas de choque, y al chocar con el jugador debe lanzar una pantalla de cómo ha muerto y posteriormente reiniciar el nivel.

```
public class MovimientoBala : MonoBehaviour
{
    public float velocidad = 5f;
    public string enemigo;


    0 referencias
    private void Update()
    {
        transform.position += transform.forward * velocidad * Time.deltaTime;

        if (GameController.imagen2.active == true)
        {
            if (Input.GetKeyDown(KeyCode.Space))
            {

                UnityEngine.SceneManagement.SceneManager.LoadSceneAsync(UnityEngine.SceneManagement.SceneManager.GetActiveScene().name);

            }
        }
        else
        {
            Destroy(gameObject, 5f);

        }

    }
    0 referencias
    private void OnTriggerEnter(Collider other)
    {

        if (other.gameObject.CompareTag("Enemy"))
        {
            Rotura rotura = other.gameObject.GetComponent<Rotura>();

            Instantiate(GameController.gamecontroller.particulas, transform.position, transform.rotation);

            rotura.Romper();
        }
        else if (other.gameObject.CompareTag("Player") && enemigo != null)
        {

            Debug.Log("asdasd");
            switch (enemigo)
            {

                case "red":
                    GameController.muertescript.ModificarTexto("Te han asesinado");
                    break;

                case "yellow":
                    GameController.muertescript.ModificarTexto("Has sufrido un accidente");

                    break;
                case "blue":
                    GameController.muertescript.ModificarTexto("Moriste triste y solo");

                    break;
                case "green":
                    GameController.muertescript.ModificarTexto("Tus deudas se pagan caras");
```

```
            break;
        case "white":
            GameController.muertescript.ModificarTexto("Muerto por sobredosis");

            break;
        case "black":
            GameController.muertescript.ModificarTexto("No soportaste la muerte de tu familia");

            break;
        case "grey":
            GameController.muertescript.ModificarTexto("Una enfermedad ha podido contigo");

            break;



    }
    GameController.imagen2.SetActive(true);
```

El prefab no es más que una bola simple:

Controlador

El controlador se encarga de varias cosas, principalmente de controlar tu disparo, o tu lanzamiento de arma, de recoger armas de lejos y también de escalar el tiempo para que dé la sensación de SlowMotion si te estás moviendo

```
public class GameController : MonoBehaviour
{
    public static GameController gamecontroller;

    public bool puedeDisparar = true;
    public bool accion;
    public GameObject bala;
    public Transform balaspawn;
    public AudioClip disparo;

    public ControladorArma arma;

    public Transform posiciondelarma;
    public LayerMask capa;

    public Image indicador;

    public GameObject particulas;

    public GameObject balaprefab;

    0 referencias
    private void Awake()
    {
        gamecontroller = this;
        if (posiciondelarma.GetComponentInChildren<ControladorArma>() != null)
        {
            arma = posiciondelarma.GetComponentInChildren<ControladorArma>();
        }
    }
```

```
// Update is called once per frame
0 referencias
private void Update()
{
    //Si pulsas el botón de disparar, disparas una bala
    if (puedeDisparar && Input.GetMouseButtonDown(0))
    {
        StopCoroutine(TiempoEspera(.03f));
        StartCoroutine(TiempoEspera(.03f));
        if (arma != null)
        {
            arma.Disparar(Camera.main.transform.position + (Camera.main.transform.forward * .5f) + (Camera.main.transform.up * -.02f), Camera.main.transform.rotation, false);
        }
    }
    //Si pulsas el otro botón tiras el arma
    if (Input.GetMouseButtonDown(1))
    {
        StopCoroutine(TiempoEspera(.4f));
        StartCoroutine(TiempoEspera(.4f));

        if (arma != null)
        {
            arma.Lanzar();
            arma = null;
        }
    }
    //Si no tienes arma, pero tu vista está sobre un arma, y pulsas el botón de disparar, el alma volará hacia ti
    RaycastHit hit;
    if (Physics.Raycast(Camera.main.transform.position, Camera.main.transform.forward, out hit, 10, capa) && Input.GetMouseButtonDown(0) && arma == null)
    {
        hit.transform.GetComponent<ControladorArma>().Recoger();
    }

    float x = Input.GetAxisRaw("Horizontal");
    float y = Input.GetAxisRaw("Vertical");

    float tiempo = (x != 0 || y != 0) ? 1f : .03f;
    float lerpTime = (x != 0 || y != 0) ? .05f : .5f;

    tiempo = accion ? 1 : tiempo;
    lerpTime = accion ? .1f : lerpTime;

    //Escala el tiempo segun tu movimiento, si te mueves  el tiempo se vuelve normal, si no, mas lento
    Time.timeScale = Mathf.Lerp(Time.timeScale, tiempo, lerpTime);
}
```

```
4 referencias
private IEnumerator TiempoEspera(float time)
{
    accion = true;
    yield return new WaitForSecondsRealtime(.06f);
    accion = false;
}
//Animacion en el HUD de como se recarga el arma segun el indicador
1 referencia
public void RecargarIndicador(float time)
{
    indicador.transform.DORotate(new Vector3(0, 0, 90), time, RotateMode.LocalAxisAdd).SetEase(Ease.Linear).OnComplete(() => indicador.transform.DOPunchScale(Vector3.one / 3, .2f, 10, 1).SetUpdate(true));
}
```

Controlador del arma

El arma también hará daño, al chocar con enemigo, así que debo permitir que se pueda soltar, agarrar, y que el enemigo la suelte si muere para ello tuve que hacer un controlador:

```
11 referencias
public class ControladorArma : MonoBehaviour
{
    public bool suelto = true;

    public bool recargando;

    private Rigidbody rigibody;
    private Collider collider;

    public float tiempoderecarga = .3f;

    0 referencias
    private void Start()
    {
        rigibody = GetComponent<Rigidbody>();
        collider = GetComponent<Collider>();

        CambiarEstado();
    }


    //Cambia el estado del arma segun la situación
    3 referencias
    private void CambiarEstado()
    {
        if (transform.parent == null)
        {
            rigibody.isKinematic = (GameController.gamecontroller.arma == this) ? true : false;
            rigibody.interpolation = (GameController.gamecontroller.arma == this) ? RigidbodyInterpolation.None : RigidbodyInterpolation.Interpolate;
            collider.isTrigger = (GameController.gamecontroller.arma == this);
        }
    }
```

```
/// <summary>
/// Disparar bala, dispara una bala en la direccion en la que se mira
/// </summary>
/// <param name="posicion"></param>
/// <param name="rotacion"></param>
/// <param name="enemigo"></param>
2 referencias
public void Disparar(Vector3 posicion, Quaternion rotacion, bool enemigo)
{
    if (!recargando)
    {
        GameObject bala = Instantiate(GameController.gamecontroller.balaprefab, posicion, rotacion);

        if (GetComponentInChildren<ParticleSystem>() != null)
        {
            GetComponentInChildren<ParticleSystem>().Play();
        }

        if (GameController.gamecontroller.arma == this)
        {
            StartCoroutine(Recargar());

            Camera.main.transform.DOComplete();
            Camera.main.transform.DOShakePosition(.2f, .01f, 10, 90, false, true).SetUpdate(true);

            transform.DOLocalMoveZ(-.1f, .05f).OnComplete(() => transform.DOLocalMoveZ(0, .2f));
        }
    }
}

/// <summary>
/// Lanza el arma hacia delante
/// </summary>
1 referencia
public void Lanzar()
{
    Sequence secuence = DOTween.Sequence();
    secuence.Append(transform.DOMove(transform.position - transform.forward, .01001f)).SetUpdate(true);
    secuence.AppendCallback(() => transform.parent = null);
    secuence.AppendCallback(() => transform.position = Camera.main.transform.position + (Camera.main.transform.right * .1001f));
    secuence.AppendCallback(() => CambiarEstado());
    secuence.AppendCallback(() => rigibody.AddForce(Camera.main.transform.forward * 10.01f, ForceMode.Impulse));
    secuence.AppendCallback(() => rigibody.AddTorque(transform.transform.right + transform.transform.up * 20.01f, ForceMode.Impulse));
}
```

```
/// <summary>
/// Recoge el arma
/// </summary>
1 referencia
public void Recoger()
{
    if (suelto)
    {
        GameController.gamecontroller.arma = this;
        CambiarEstado();

        transform.parent = GameController.gamecontroller.posiciondelarma;

        transform.DOLocalMove(Vector3.zero, .25f).SetEase(Ease.OutBack).SetUpdate(true);
        transform.DOLocalRotate(Vector3.zero, .25f).SetUpdate(true);
    }
}
/// <summary>
/// Cuando el enemigo muere suelta el arma
/// </summary>
1 referencia
public void Soltar()
{
    suelto = true;
    transform.parent = null;
    rigibody.isKinematic = false;
    rigibody.interpolation = RigidbodyInterpolation.Interpolate;
    collider.isTrigger = false;

    rigibody.AddForce((Camera.main.transform.position - transform.position) * 2, ForceMode.Impulse);
    rigibody.AddForce(Vector3.up * 1, ForceMode.Impulse);
}
/// <summary>
/// Controla que el arma se caargue un tiempo
/// </summary>
/// <returns></returns>
1 referencia
private IEnumerator Recargar()
{
    if (GameController.gamecontroller.arma == this)
    {
        GameController.gamecontroller.RecargarIndicador(tiempoderecarga);
        recargando = true;
        yield return new WaitForSeconds(tiempoderecarga);
        recargando = false;
    }
}
```

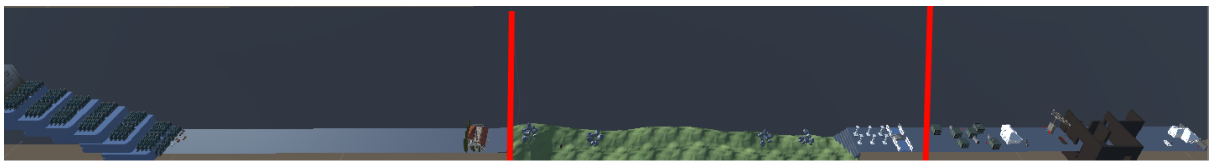
```
/// <summary>
/// Si choca con el enemigo lo mata
/// </summary>
/// <param name="collision"></param>
0 referencias
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Enemy") && collision.relativeVelocity.magnitude < 15)
    {
        Rotura rotura = collision.gameObject.GetComponent<Rotura>();

        if (!rotura.enemigo.muerto)
        {
            Instantiate(GameController.gamecontroller.particulas, transform.position, transform.rotation);
        }

        rotura.Romper();
    }
}
```

Objetivo del juego

Llegar al final del mapa sin morir, descubrir la historia del personaje y revivir su vida fraccionada en 3 actos, Niñez, adolescencia y decadencia.

También puedes elegir buscar ciertas pistas que te descubran la historia del personaje, o puedes elegir ir directo al final e imaginar la historia o buscar detalles para interpretarla esta.

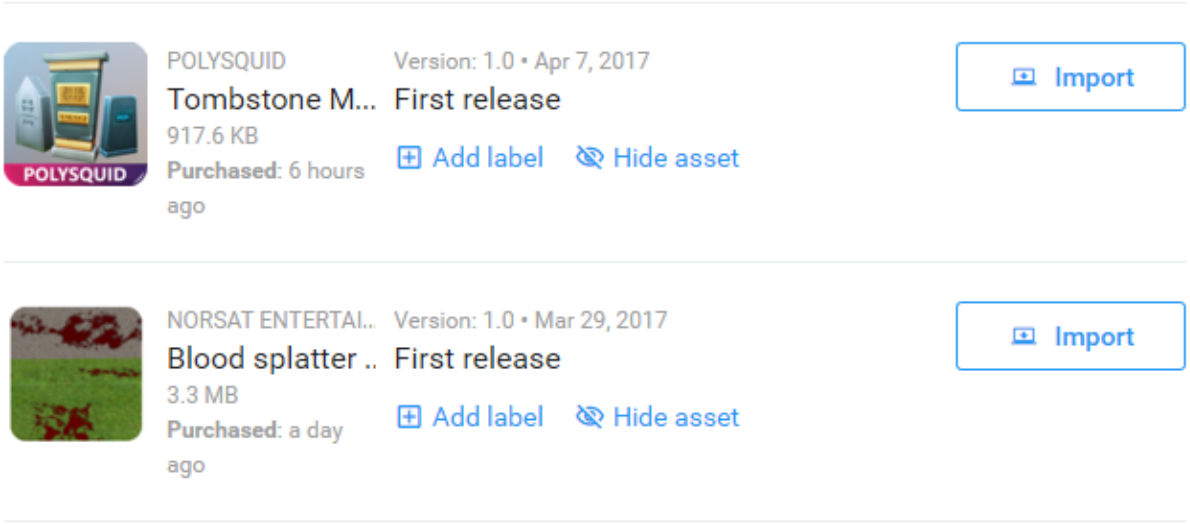
Mapeado



El juego puede estructurarse en 3 trozos, que serían la niñez, adolescencia y la decadencia, cada parte tiene sus dificultades, se nota como la niñez todo se ve más grande, más difícil y complejo, la adolescencia es todo como con más curvas y dificultades, todo se hace enorme pero no en tamaño si no en sentimiento, y la decadencia es superar mejorar eso vivido y morir.

La colocación de los enemigos tiene una cierta lógica, posicionando los enemigos de dinero más adelante que en la niñez, o eliminando ciertos enemigos en un momento concreto.

El escenario fue creado con recursos de la Store de Unity:



TERRA NOVA CREA.. Version: 1.3 • Apr 23, 2019
Low Poly Meg... Fix broken forest and greek prefabs
5.5 MB
Purchased: a day ago
Add label Hide asset
Import

WAND AND CIRCLE.. Version: 1.0 • Oct 17, 2017
Polygon City P.. First release
5.0 MB
Purchased: a day ago
Add label Hide asset
Import

VERTEX STUDIO Version: 1.3 • Dec 14, 2017
Big Furniture P. Removing legacy js scripts Clean up
more
11.3 MB
Purchased: a day ago
Add label Hide asset
Import

Conclusión

Ha sido un proyecto largo y algo tedioso, nunca había hecho un juego en 3D ni con mecánicas tan chulas, me ha gustado diseñar el mapeado y aprender todas las cosas y mecánicas que he descubierto con este juego.

Creo que ha sido un proyecto complicado, que me ha dado muchos problemas, y bastante largo, pero con este he aprendido muchas cosas positivas, aunque haya costado.