Universidad Tecnológica de Tijuana

Ingenieria en Desarrollo y Gestión de Software

Realizado por:

Toledo Perez Cristian Alejandro

Profesor:

Ing. Eduardo Fortino Garcia Zaragoza

Materia:

Optativa I: Creación de Videjuegos

Actividad:

“2.2 Animación de Rodar y Cinemáticas de Cámara”

Cuatrimestre: Decimo ***Grupo:*** 10-¿

Fecha de realización

Martes 06 y viernes 09 de Febrero del 2024

Parcial:

Segundo – Unidad 2

INDICE DE CONTENIDO

I. INTRODUCCIÓN DE LA ACTIVIDAD

En este análisis del desarrollo en Unity, exploramos la creación de un juego 2D, centrándonos en las animaciones y la configuración de la cámara para mejorar la experiencia del jugador. Desde la implementación de nuevas animaciones hasta la integración del paquete Cinemachine para efectos visuales, examinaremos en detalle el proceso de desarrollo y las mejoras realizadas en el proyecto. Este enfoque detallado nos permite comprender mejor las complejidades del desarrollo de juegos en Unity y cómo cada elemento contribuye a una experiencia de juego envolvente y emocionante. En este documento trataremos la animación de un dash o giro y configuraciones de la cámara, los cuales son aspectos fundamentales, ya que proporcionan al jugador una experiencia más inmersiva y dinámica.

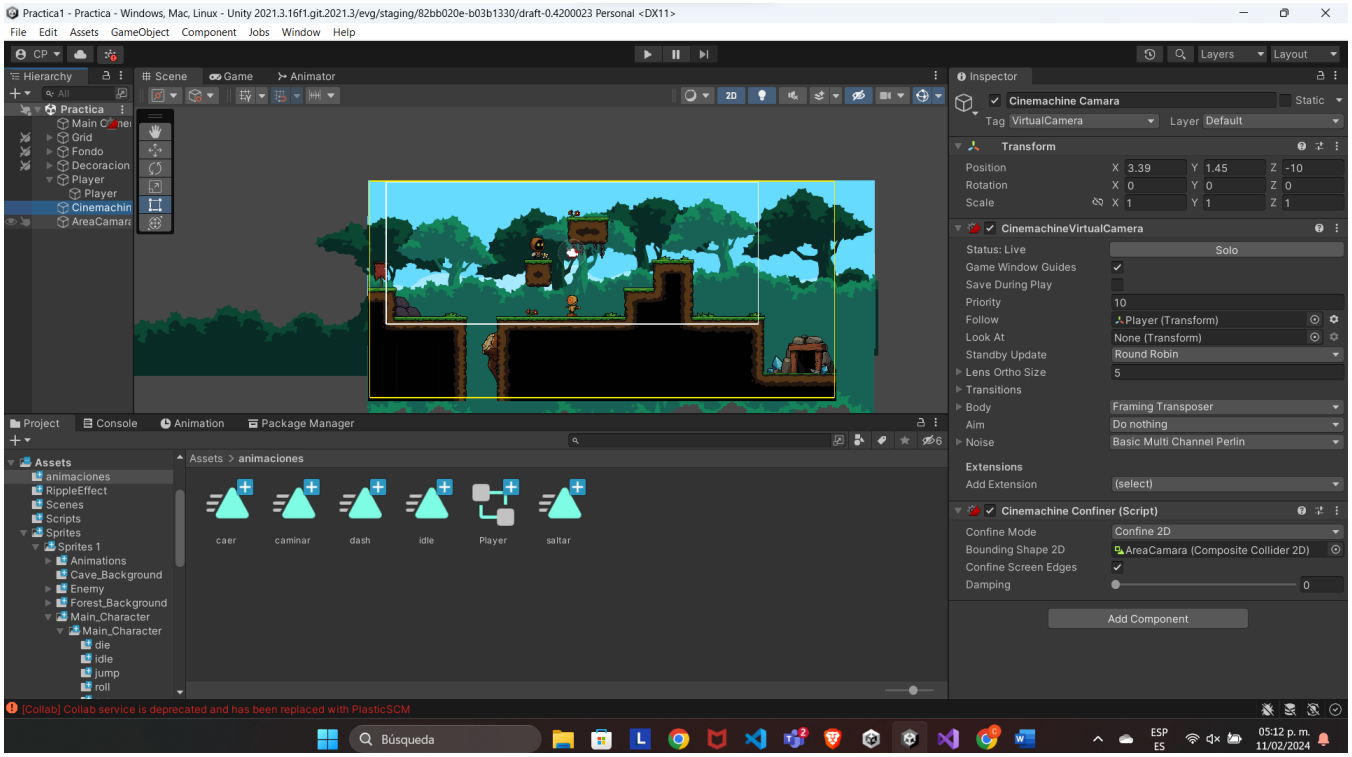
II. DEMOSTRACIÓN DEL DESARROLLO: UNITY

Comenzamos accediendo al Animator y creamos un nuevo Blend Tree para mejorar las animaciones existentes. Configuramos un parámetro de tipo Float para medir la velocidad vertical y añadimos dos animaciones al árbol: una para el salto y otra para la caída. Eliminamos las animaciones individuales previas y ajustamos las transiciones correspondientes, como de CAMINAR a SALTO y de SALTO a IDLE, para asegurar que se activen adecuadamente. En el script, realizamos modificaciones para actualizar las funciones y ajustar los comportamientos de las animaciones.

Posteriormente, integramos una nueva animación llamada "dash" al proyecto, la cual simula el efecto de rodar. Ajustamos las transiciones entre caminar y "dash", así como entre salto y "dash", asegurando que las condiciones sean coherentes. Agregamos nuevos parámetros y ajustamos las estadísticas para mejorar el comportamiento del personaje durante el "dash".

Continuamos configurando la cámara utilizando el paquete Cinemachine para lograr efectos cinematográficos. Creamos una nueva cámara 2D y la ajustamos para seguir al jugador, limitando su área de movimiento con un confinador Cinemachine. Modificamos el área del Collider para definir el espacio seguido por la cámara y configuramos los componentes necesarios para asegurar su funcionamiento óptimo.

Finalmente, agregamos vibración a la cámara durante ciertas acciones del jugador, utilizando scripts y métodos para controlar este efecto. Verificamos el funcionamiento del proyecto en Unity y ajustamos la física del juego para una experiencia más fluida y coherente.


Practica1 - Practica - Windows, Mac, Linux - Unity 2021.3.16f1.git.2021.3/evg/staging/82bb020e-b03b1330/draft-0.4200023 Personal <DX11>
File Edit Assets GameObject Component Jobs Window Help
Hierarchy
Practica
Main Camera
Grid
Fondo
Decoracion
Player
Player
Cinemachin
AreaCamara
Scene
Game
Animator
Inspector
Cinemachine Camara
Static
Tag VirtualCamera
Layer Default
Transform
Position X 3.39 Y 1.45 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
CinemachineVirtualCamera
Status: Live
Solo
Game Window Guides
Save During Play
Priority 10
Follow Player (Transform)
Look At None (Transform)
Standby Update Round Robin
Lens Ortho Size 5
Transitions
Body Framing Transposer
Aim Do nothing
Noise Basic Multi Channel Perlin
Extensions
Add Extension (select)
Cinemachine Confiner (Script)
Confine Mode Confine 2D
Bounding Shape 2D AreaCamara (Composite Collider 2D)
Confine Screen Edges
Damping 0
Add Component
Project
Console
Animation
Package Manager
Assets
animaciones
RippleEffect
Scenes
Scripts
Sprites
Sprites 1
Animations
Cave_Background
Enemy
Forest_Background
Main_Character
Main_Character
die
idle
jump
roll
Assets > animaciones
caer
caminar
dash
idle
Player
saltar
[Collab] Collab service is deprecated and has been replaced with PlasticSCM
Búsqueda
ESP
ES
05:12 p. m.
11/02/2024

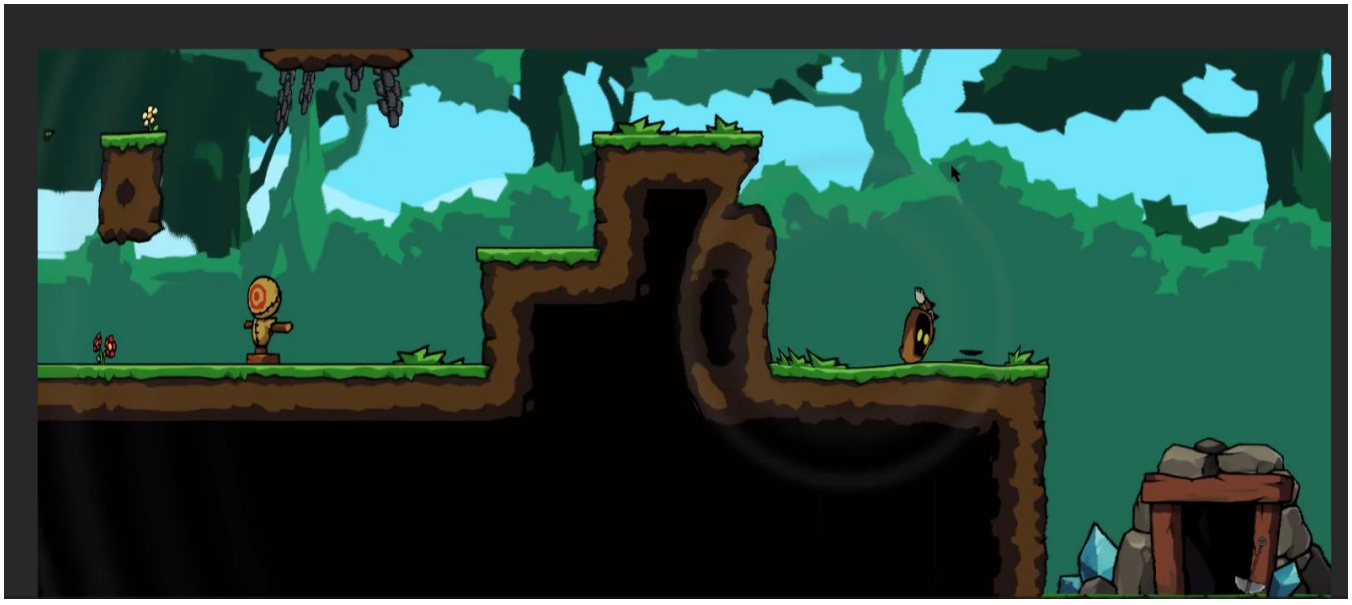
III. DEMOSTRACIÓN DE USABILIDAD: GAME EJECUTADO

Para poner en marcha nuestro proyecto, nos dirigimos a la opción "Game" y hacemos clic en el botón "Start". Alternativamente, podemos optar por la opción "Edit" y seleccionar "Play" o utilizar la combinación de teclas "Ctrl + P". Una vez en el proyecto ejecutado, podemos interactuar con las siguientes características:

Movimiento del personaje: Utilizamos las teclas de flecha para controlar el movimiento del personaje. Ahora, además de las animaciones previas de caminar, idle, saltar y caer, también podemos activar una animación de "dash" al presionar una tecla designada.

Cámara dinámica: Durante la ejecución del juego, la cámara sigue al jugador utilizando las funcionalidades proporcionadas por el paquete Cinemachine. Esto asegura que la cámara siga al personaje mientras se mueve por el escenario, manteniendo una perspectiva adecuada y mejorando la experiencia visual del jugador.

Estas características agregadas proporcionan una experiencia de juego más dinámica y emocionante, permitiendo al jugador explorar el mundo del juego con mayor fluidez y detalle.



IV. EXPLICAICÓN Y ANÁLISIS BREVE DEL SCRIPT

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

public class PlayerController : MonoBehaviour
{
    private Rigidbody2D rb;
    private Animator anim;
    private Vector2 direccion;
    private CinemachineVirtualCamera cm;

    [Header("Estadisticas")]
    public float velocidadDeMovimiento = 10;
    public float fuerzaDeSalto = 8;
    public float velocidadDash = 20;

    [Header("Colisiones")]
    public LayerMask layerPiso;
    public Vector2 abajo;
    public float radioDeColision;


    [Header("booleanos")]
    public bool puedeMover = true;
    public bool enSuelo = true;
    public bool puedeDash;
    public bool haciendoDash;
    public bool tocadoPiso;
    public bool haciendoShake;

    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        cm = GameObject.FindGameObjectWithTag("VirtualCamera").GetComponent<CinemachineVirtualCamera>();
    }
    void Start()
    {

    }

    void Update()
    {
        Movimiento();
        Agarres();
    }
```

```
    }

    private IEnumerable AgitarCamara()
    {
        haciendoShake = true;
        CinemachineBasicMultiChannelPerlin cinemachineBasicMultiChannelPerlin = cm.GetCinemachineComponent<CinemachineBasicMultiChannelPerlin>();
        cinemachineBasicMultiChannelPerlin.m_AmplitudeGain = 5;
        yield return new WaitForSeconds(0.3f);
        cinemachineBasicMultiChannelPerlin.m_AmplitudeGain = 0;
        haciendoShake = false;
    }

    private IEnumerable AgitarCamara(float tiempo)
    {
        haciendoShake = true;
        CinemachineBasicMultiChannelPerlin cinemachineBasicMultiChannelPerlin = cm.GetCinemachineComponent<CinemachineBasicMultiChannelPerlin>();
        cinemachineBasicMultiChannelPerlin.m_AmplitudeGain = 5;
        yield return new WaitForSeconds(tiempo);
        cinemachineBasicMultiChannelPerlin.m_AmplitudeGain = 0;
        haciendoShake = false;
    }



    private void Dash(float x, float y)
    {
        anim.SetBool("Dash", true);
        Vector3 posicionJugador = Camera.main.WorldToViewportPoint(transform.position);
        Camera.main.GetComponent<RippleEffect>().Emit(posicionJugador);

        puedeDash = true;
        rb.velocity = Vector2.zero;
        rb.velocity += new Vector2(x, y).normalized * velocidadDash;
        StartCoroutine(AgitarCamara());

    }

    private IEnumerable PrepararDash()
    {
        StartCoroutine(DashSuelo());
        rb.gravityScale = 0;
        haciendoDash = true;

        yield return new WaitForSeconds(0.3f);
        rb.gravityScale = 3;
        haciendoDash = false;
        FinalizarDash();
    }

    private void StartCoroutine(IEnumerable enumerable)
    {
        throw new NotImplementedException();
    }

    private IEnumerable DashSuelo()
    {
        yield return new WaitForSeconds(0.15f);
        if (enSuelo)
            puedeDash = false;
    }

    public void FinalizarDash()
    {
        anim.SetBool("Dash", false);
    }

    private void TocarPiso()
    {
        puedeDash = false;
        haciendoDash = false;
        anim.SetBool("Saltar", false);
    }


    private void Movimiento()
    {
        float x = Input.GetAxis("Horizontal");
        float y = Input.GetAxis("Vertical");

        float xRaw = Input.GetAxisRaw("Horizontal");
        float yRaw = Input.GetAxisRaw("Vertical");

        direccion = new Vector2(x, y);

        Caminar();

        MejorarSalto();
        if (Input.GetKeyDown(KeyCode.Space))
```

```
        Caminar();

        MejorarSalto();
        if (Input.GetKeyDown(KeyCode.Space))
        {
            if (enSuelo)
            {
                anim.SetBool("Saltar", true);
                Saltar();
            }
        }

        if (Input.GetKeyDown(KeyCode.X) && !haciendoDash)
        {
            if (xRaw != 0 || yRaw != 0)
                Dash(xRaw, yRaw);
        }

        if(enSuelo && !tocadoPiso)
        {
            TocarPiso();
            tocadoPiso = true;
        }
        if(!enSuelo && tocadoPiso)
        {
            tocadoPiso = false;
        }

        if (!enSuelo)
        {
            anim.SetBool("Caer", false);
        }

        if (Input.GetKeyDown(KeyCode.X))
        {
            Camera.main.GetComponent<RippleEffect>().Emit(transform.position);
        }

        float velocidad;
        if (rb.velocity.y > 0)
            velocidad = 1;
        else
            velocidad = -1;

        if (!enSuelo)
        {
            anim.SetFloat("velocidadVertical", velocidad);
        }
        else
        {
            if (velocidad == -1)
                FinalizarSalto();
        }
    }

    public void FinalizarSalto()
    {
        anim.SetBool("Saltar", false);
        anim.SetBool("Caer", true);
    }
    private void MejorarSalto()
    {
        if (rb.velocity.y < 0)
        {
            rb.velocity += Vector2.up * Physics2D.gravity.y * (2.5f - 1) * Time.deltaTime;
        }
        else if (rb.velocity.y > 0 && !Input.GetKey(KeyCode.Space))
        {
            rb.velocity += Vector2.up * Physics2D.gravity.y * (2.0f - 1) * Time.deltaTime;
        }

    }

    private void Agarres()
    {
        enSuelo = Physics2D.OverlapCircle((Vector2)transform.position + abajo, radioDeColision, layerPiso);
    }
    private void Saltar()
    {
        rb.velocity = new Vector2(rb.velocity.x, 0);
        rb.velocity += Vector2.up * fuerzaDeSalto;
    }
    private void Caminar()
    {
        if (puedeMover && !haciendoDash)
        {
            rb.velocity = new Vector2(direccion.x * velocidadDeMovimiento, rb.velocity.y);

            if (direccion != Vector2.zero)
```

```
219
220             if (direccion != Vector2.zero)
221             {
222                 if (!enSuelo)
223                 {
224                     anim.SetBool("Caer", true);
225                 }
226                 else
227                 {
228                     anim.SetBool("Caminar", true);
229                 }
230
231                 if (direccion.x < 0 && transform.localScale.x > 0)
232                 {
233                     transform.localScale = new Vector3(-transform.localScale.x, transform.localScale.y, transform.localScale.z);
234                 }
235                 else if (direccion.x > 0 && transform.localScale.x < 0)
236                 {
237                     transform.localScale = new Vector3(Mathf.Abs(transform.localScale.x), transform.localScale.y, transform.localScale.z);
238                 }
239             }
240             else
241             {
242                 anim.SetBool("Caminar", false);
243             }
244         }
245
246     }
247 }
248
```

Dash (Movimiento Rápido):

El personaje tiene la capacidad de realizar un movimiento rápido, conocido como dash. Cuando el jugador presiona la tecla designada (en este caso, la tecla "X"), el personaje activa una animación de dash y se desplaza con una velocidad incrementada en la dirección especificada por el jugador.

Para implementar el dash, se utilizan varias funciones y variables:

- La función Dash(float x, float y) se encarga de activar la animación de dash y aplicar una velocidad específica al personaje en la dirección determinada por los parámetros x e y.
- Se utiliza una corutina llamada PrepararDash() para controlar la secuencia de acciones durante el dash, como detener temporalmente la gravedad y finalizar la animación del dash.
- Durante el dash, se activa un efecto visual en la cámara mediante la función AgitarCamara(), que crea una sensación de dinamismo y movimiento en la escena.

Efecto de la Cámara:

Para mejorar la experiencia visual del jugador, se implementa un efecto de agitación en la cámara cuando el personaje realiza acciones especiales, como el dash.

- Las funciones AgitarCamara() y AgitarCamara(float tiempo) se encargan de activar y desactivar este efecto de agitación en la cámara. Este efecto se logra utilizando la clase CinemachineBasicMultiChannelPerlin del paquete Cinemachine.
- En el método Awake(), se busca y asigna el componente CinemachineVirtualCamera al objeto cm, permitiendo así el control de la cámara durante el juego.

V. CONCLUSIÓN DE LA ACTIVIDAD

En conclusión, el desarrollo en Unity requiere una combinación de creatividad, habilidad técnica y atención al detalle. La implementación de animaciones mejoradas, como el "dash", y la configuración dinámica de la cámara mediante Cinemachine, agregan profundidad y realismo al juego. Estas mejoras proporcionan una experiencia de juego más inmersiva y dinámica, permitiendo al jugador explorar el mundo del juego con mayor fluidez y detalle. El análisis detallado del script revela la complejidad y la importancia de cada función implementada, destacando la necesidad de un enfoque meticuloso en cada aspecto del desarrollo del juego en Unity.