**ОТЧЕТ**

по дисциплине МДК 02.02. Инструментальные средства разработки программного обеспечения

**Группа:** ИСп351ДК

**Тема проекта:** Создание игрового приложения 2D Who's coming to the party

**Cameraddd.cs** предназначен для Unity и реализует простую систему следования камеры за целевым объектом с плавностью.

**Основные компоненты:**

**Класс Cameraddd**: Главный класс скрипта, который наследует от MonoBehaviour (базовый класс для всех скриптов Unity).

**Вложенный класс CameraFollow**: Внутри основного класса находится еще один класс, что является необычным для Unity (обычно используют отдельные файлы для каждого класса). Этот класс также наследует от MonoBehaviour.

**Поля класса CameraFollow:**

public Transform target  ссылка на объект, за которым должна следовать камера

public float smoothSpeed = 0.125f  коэффициент плавности движения камеры (чем меньше, тем плавнее)

public Vector3 offset  смещение камеры относительно целевого объекта

**Метод LateUpdate():**

Это стандартный метод Unity, который вызывается каждый кадр после выполнения всех Update(). Идеально подходит для движения камеры, так как гарантирует, что все объекты уже переместились.

**Логика работы:**

Vector3 desiredPosition = target.position + offset  вычисляет желаемую позицию камеры (позиция цели + смещение)

Vector3 smoothedPosition = Vector3.Lerp(transform.position, desiredPosition, smoothSpeed)  плавно интерполирует между текущей позицией камеры и желаемой позицией

Vector3.Lerp() выполняет линейную интерполяцию между двумя точками

transform.position = smoothedPosition  применяет вычисленную плавную позицию к камере

**CameraRig.cs** система следования камеры за игроком

Основные компоненты:

**Поля класса**:

dumping (вероятно, опечатка должно быть "damping") коэффициент плавности движения камеры

offset  смещение камеры относительно игрока по осям X и Y

isLeft  флаг, указывающий направление взгляда игрока

cat  ссылка на трансформ игрока (с тегом "Player")

lastX  предыдущая позиция игрока по X для определения направления

**Метод Start()**:

Гарантирует, что offset.x положительный (Mathf.Abs)

Вызывает FindPlayer с текущим значением isLeft

**Метод FindPlayer(bool CatIsLeft)**:

Находит игрока по тегу "Player"

Запоминает текущую позицию игрока по X

Устанавливает начальную позицию камеры с учетом направления

**Метод Update()**:

Определяет направление движения игрока (влево/вправо)

Вычисляет целевую позицию камеры с учетом направления и смещения

Плавно перемещает камеру к цели с помощью Vector3.Lerp

**Dialogue.cs** система триггеров для диалогов/облаков

Основные компоненты:

**Поле clouds**:

Массив аниматоров (вероятно, облаков с анимациями)

**Метод OnTriggerEnter2D**:

Срабатывает при входе объекта с тегом "Player" в триггер

Запускает анимацию "isTriggere" для всех облаков в массиве

**Метод OnTriggerExit2D**:

Срабатывает при выходе объекта с тегом "Player" из триггера

Также запускает ту же анимацию "isTriggere"

**CameraFollow.cs** реализует систему плавного слежения камеры за игроком с учетом направления его движения. Вот детальный разбор:

Основные параметры

**dumping** (скорее всего опечатка, должно быть damping) коэффициент плавности/затухания движения камеры (1.5f по умолчанию)

**offset**  смещение камеры относительно игрока (2f по X, 1f по Y)

**isLeft**  флаг, указывающий смотрит ли игрок влево

Приватные переменные

**cat**  ссылка на Transform игрока (имеющего тег "Player")

**lastX**  предыдущая координата X игрока для определения направления

**Методы**

**Start()**

Инициализация при старте:

* Делает offset.x абсолютным значением (на случай если было отрицательное)
* Вызывает FindPlayer() с текущим значением isLeft
* FindPlayer(bool CatIsLeft)
* Находит игрока и устанавливает начальную позицию камеры:
* Находит объект с тегом "Player" и сохраняет его Transform
* Запоминает текущую позицию X игрока
* Устанавливает позицию камеры с учетом:
* Смещения влево если CatIsLeft = true
* Смещения вправо если CatIsLeft = false

**Update()**

Основная логика слежения (вызывается каждый кадр):

* Проверяет что игрок существует
* Определяет направление движения игрока:
* Если текущий X больше предыдущего двигается вправо (isLeft = false)
* Если меньше двигается влево (isLeft = true)
* Вычисляет целевую позицию камеры с учетом направления
* Плавно перемещает камеру к цели используя Vector3.Lerp:
* Текущая позиция камеры
* Целевая позиция (позиция игрока + смещение)
* Коэффициент плавности (dumping \* Time.fixedDeltaTime)
* Особенности работы
* Камера автоматически определяет направление движения игрока
* Плавное перемещение без рывков благодаря Lerp
* Сохраняет Zкоординату камеры (важно для 2D камер)

**GameManager.cs**

Этот скрипт управляет основными игровыми функциями:

1. **Элементы:**

panelSetting ссылка на GameObject панели настроек

В Start() панель настроек изначально скрывается

1. **Методы:**

Play() загружает сцену с именем "Play"

Back() загружает сцену с именем "Game"

Setting() переключает видимость панели настроек

Exit() закрывает приложение

**Trampline.cs**

Этот скрипт реализует поведение трамплина:

1. **Параметры:**
   1. force сила прыжка (настраивается в инспекторе)
2. **Логика:**
   1. При столкновении с объектом с тегом "Player"
   2. Применяет импульсную силу вверх к Rigidbody игрока

Особенности:

Использует ForceMode2D.Impulse для мгновенного воздействия

Тег проверяется через сравнение строк (менее эффективно, чем CompareTag)

**MainCamera.cs**

Скрипт следования камеры:

**Логика:**

* Плавное слежение за игроком с учетом направления
* Смещение камеры в зависимости от направления взгляда

**Teleport.cs**

Скрипт телепортации игрока:

1. **Параметры:**
   1. pointTeleport целевая точка телепортации
2. **Логика:**
   1. При столкновении с игроком
   2. Перемещает игрока в позицию pointTeleport

**Приложения к отчету** (вся документация, макеты, и пр… которые были составлены в процессе разработки проекта. Указать название каждого приложения):

**Cameraddd.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Cameraddd : MonoBehaviour

{

    public class CameraFollow : MonoBehaviour

    {

                public Transform target;

                public float smoothSpeed = 0.125f;

                public Vector3 offset;

                void LateUpdate()

        {

            Vector3 desiredPosition = target.position + offset;

            Vector3 smoothedPosition = Vector3.Lerp(transform.position, desiredPosition, smoothSpeed);

            transform.position = smoothedPosition;

        }

    }

}

**CameraFollow.cs**

using System;

using UnityEngine;

public class MoveCamera : MonoBehaviour

{

    public float dumping = 1.5f;

    public Vector2 offset = new Vector2(2f, 1f);

    public bool isLeft;

    private Transform cat;

    private int lastX;

    void Start()

    {

        offset = new Vector2(Mathf.Abs(offset.x), offset.y);

        FindPlayer(isLeft);

    }

    public void FindPlayer(bool CatIsLeft)

    {

        cat = GameObject.FindGameObjectWithTag("Player").transform;

        lastX = Mathf.RoundToInt(cat.position.x);

        if (CatIsLeft)

            transform.position = new Vector3(cat.position.x offset.x, cat.position.y offset.y, transform.position.z);

        else

            transform.position = new Vector3(cat.position.x + offset.x, cat.position.y + offset.y, transform.position.z);

    }

    void Update()

    {

        if (cat)

        {

            int currentX = Mathf.RoundToInt(cat.position.x);

            if (currentX > lastX) isLeft = false; else if (currentX < lastX) isLeft = true;

            lastX = Mathf.RoundToInt(cat.position.x);

            Vector3 target;

            if (isLeft)

            {

                target = new Vector3(cat.position.x offset.x, cat.position.y offset.y, transform.position.z);

            }

            else

            {

                target = new Vector3(cat.position.x + offset.x, cat.position.y + offset.y, transform.position.z);

            }

            Vector3 Position = Vector3.Lerp(transform.position, target, dumping \* Time.fixedDeltaTime);

            transform.position = Position;

        }

    }

}

**CameraRig.cs**

using System;

using UnityEngine;

public class CameraRig : MonoBehaviour

{

    public float dumping = 1.5f;

    public Vector2 offset = new Vector2(2f, 1f);

    public bool isLeft;

    private Transform cat;

    private int lastX;

    void Start()

    {

        offset = new Vector2(Mathf.Abs(offset.x), offset.y);

        FindPlayer(isLeft);

    }

    public void FindPlayer(bool CatIsLeft)

    {

        cat = GameObject.FindGameObjectWithTag("Player").transform;

        lastX = Mathf.RoundToInt(cat.position.x);

        if (CatIsLeft)

            transform.position = new Vector3(cat.position.x offset.x, cat.position.y offset.y, transform.position.z);

        else

            transform.position = new Vector3(cat.position.x + offset.x, cat.position.y + offset.y, transform.position.z);

    }

    void Update()

    {

        if (cat)

        {

            int currentX = Mathf.RoundToInt(cat.position.x);

            if (currentX > lastX) isLeft = false; else if (currentX < lastX) isLeft = true;

            lastX = Mathf.RoundToInt(cat.position.x);

            Vector3 target;

            if (isLeft)

            {

                target = new Vector3(cat.position.x offset.x, cat.position.y + offset.y, transform.position.z);

            }

            else

            {

                target = new Vector3(cat.position.x + offset.x, cat.position.y + offset.y, transform.position.z);

            }

            Vector3 Position = Vector3.Lerp(transform.position, target, dumping \* Time.fixedDeltaTime);

            transform.position = Position;

        }

    }

}

**Dialogue.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Dialogue : MonoBehaviour

{

    public Animator[] clouds;

    public void OnTriggerEnter2D(Collider2D other)

    {

        if (other.CompareTag("Player"))

        {

            foreach(Animator anim in clouds)

            {

                anim.SetTrigger("isTriggere");

            }

        }

    }

    public void OnTriggerExit2D(Collider2D other)

    {

        if (other.CompareTag("Player"))

        {

            foreach (Animator anim in clouds)

            {

                anim.SetTrigger("isTriggere");

            }

        }

    }

}

**GameManager.cs**

using JetBrains.Annotations;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class GameMeneger : MonoBehaviour

{

    public GameObject panelSetting;

    private void Start()

    {

        panelSetting.SetActive(false);

    }

    public void Play()

    {

        SceneManager.LoadScene("Play");

    }

    public void Back()

    {

        SceneManager.LoadScene("Game");

    }

    public void Setting()

    {

        if (panelSetting.activeSelf == false)

        {

            panelSetting.SetActive(true);

        }

        else if (panelSetting.activeSelf == true)

        {

            panelSetting.SetActive(false);

        }

    }

        public void Exit()

        {

            Application.Quit();

        }

}

**Jumpp.cs**

using UnityEngine;

public class Trampline : MonoBehaviour

{

    [Header("Упругость батута")]

    [Space]

    [SerializeField] private float \_force;

    private void OnCollisionEnter2D(Collision2D collision)

    {

        if (collision.gameObject.tag == "Player")

        {

            collision.rigidbody.AddForce(Vector2.up \* \_force, ForceMode2D.Impulse);

        }

    }

}

**MainCamera.cs**

using UnityEngine;

using System.Collections;

public class MainCamera : MonoBehaviour

{

    public float damping = 1.5f;

    public Vector2 offset = new Vector2(2f, 1f);

    public bool faceLeft;

    private Transform player;

    private int lastX;

    void Start()

    {

        offset = new Vector2(Mathf.Abs(offset.x), offset.y);

        FindPlayer(faceLeft);

    }

    public void FindPlayer(bool playerFaceLeft)

    {

        player = GameObject.FindGameObjectWithTag("Player").transform;

        lastX = Mathf.RoundToInt(player.position.x);

        if (playerFaceLeft)

        {

            transform.position = new Vector3(player.position.x offset.x, player.position.y + offset.y, transform.position.z);

        }

        else

        {

            transform.position = new Vector3(player.position.x + offset.x, player.position.y + offset.y, transform.position.z);

        }

    }

    void Update()

    {

        if (player)

        {

            int currentX = Mathf.RoundToInt(player.position.x);

            if (currentX > lastX) faceLeft = false; else if (currentX < lastX) faceLeft = true;

            lastX = Mathf.RoundToInt(player.position.x);

            Vector3 target;

            if (faceLeft)

            {

                target = new Vector3(player.position.x offset.x, player.position.y + offset.y, transform.position.z);

            }

            else

            {

                target = new Vector3(player.position.x + offset.x, player.position.y + offset.y, transform.position.z);

            }

            Vector3 currentPosition = Vector3.Lerp(transform.position, target, damping \* Time.deltaTime);

            transform.position = currentPosition;

        }

    }

}

**Teleport.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Teleport : MonoBehaviour

{

    public GameObject pointTeleport;

    private void OnCollisionEnter2D(Collision2D collision)

    {

       if(collision.gameObject.tag == "Player")

        {

            collision.gameObject.transform.position = pointTeleport.gameObject.transform.position;

        }

    }

}