Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №1

з дисципліни

"Розробка ігрових застосувань. Unity рішення"

на тему

"Дослідження базового патерну ігрового рушія Unity на прикладі двовимірної технології"

Виконав:

студент групи ІП-93

Дмитренко Р.В.

Номер у списку: 5

Перевірив:

Катін П.Ю.

Київ 2022

Зміст

1. Постановка задачі 3
2. Виконання 3
    Вибір IDE 3
    Написання коду для керування персонажем 4
    Асети 5
    Скріншоти сцени 5
3. Висновок 6
4. Додатки 6

1. Постановка задачі

Мета: полягає у набутті знань, умінь та навичок з технології розроблення основ проекту з використанням обраної мови програмування у обраній парадигмі. Надається досвід створення репозиторію у системі контролю версій.

Варіант: 5

Завдання до роботи:

1. Репозиторій у системі контролю версій. Створити проект 2D. Загальні вимоги.

2. Акаунт на GitHub, на даному етапі за бажанням. Репозиторій на GitHub з проектом.

3. Назва GameProgLab1Group**Num**, де зафарбовано номер групи.

4. Установка ігрового рушія. Створений проект IDE (2D) на основі рушія, що містить 1 сцену, ігровий персонаж. Можуть бути включені інші елементи.

5. Розроблений і налагоджений скрипт для управління ігровим персонажем. Достатньо продемонструвати рух ліворуч, праворуч, стрибки, коректну фізику, зупинку перед перешкодою. Проект розташовано у репозиторій на GitHub, основна мета полягає у дослідженні і підтвердженні володіння обраною IDE (2D) і технологією розподіленої системи контролю версій.

Завдання відповідно до варіанту:

1. Примітив: квадрат.

2. Набор асетів: https://assetstore.unity.com/packages/2d/free-2d-platform-tileset-206082

2. Виконання

Вибір IDE

В якості IDE була обрана Visual Studio Code. На це є дві причини: в неї, на мій погляд, зручніший інтерфейс за Visual Studio і вона є більш легкою, не займаючи четверть диску пам'яті. Налаштовуючи Visual Studio Code для Unity, я користувався офіційною документацією від Visual Studio Code: Visual Studio Code and Unity.

Написання коду для керування персонажем

В додатках до даного звіту присутній саме цей код. Хотів би трішки його пояснити:

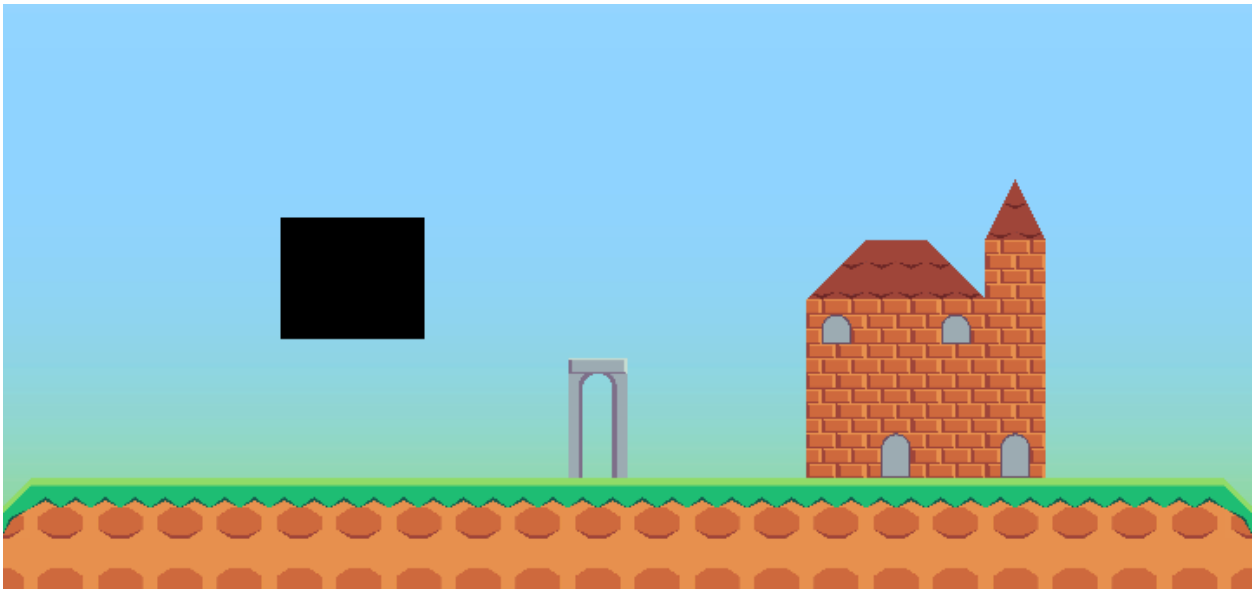
- Rigidbody2D - це компонент, який прикріплюється до ігрового персонажа. Цей компонент одразу надає йому базову фізику, написану під капото в Unity.
- BoxCollider2D - це компонент, який прикріплюється до ігрового персонажа та об'єктів з якими він буде взаємодіяти. Даний компонент потрібен для прорахування того, що буде, якщо персонаж зіткнеться з об'єктами.
- SerializeField - клас, який використовується для того, щоб змінну можна було змінювати в інтерфейсі Unity.
- _speed та jumpForce - змінні, які є атрибутами ігрового персонажа, позначають собою швидкість та силу стрибка.
- platformLayerMask - це LayerMask, який потрібен для того, щоб ігровий персонаж міг визначити, стоїть він на платформі чи ні.
- У функції Start() ми ініціалізуємо об'єкти класів Rigidbody2D та BoxCollider2D для того, щоб керувати діями нашими персонажами.
- У функції IsGrounded() визначається чи стоїть наш персонаж на платформі. Це робиться за допомогою RaycastHit2D, який запускає умовний промінь під нашого персонажа і в радіусі, вказаному в ініціалізації відповідної функції BoxCast() шукає якийсь об'єкт, в нашому випадку - це платформа.
- У функції Jump() реалізований стрибок ігрового персонажа. Зазначу, що Vector2.up - це направлення в якому має бути здійснений рух персонажа.
- У функції Update() перевіряється, що має відбуватись кожен кадр. В даному випадку в нас перевіряється, чи рухається персонаж вправо/вліво за допомогою Input.GetAxis("Horizontal"). Це вбудований в юніті Input, який відслідковує натискання стрілок вправо/вліво або клавіш "a" і "d". Далі в нас перевіряється чи був натиснутий "пробіл" та чи знаходиться ігровий персонаж на платформі. Навіщо остання перевірка? Щоб персонаж не зміг після стрибка ще раз підстрибнути в повітрі, що буде порушенням правил фізики.

Асети

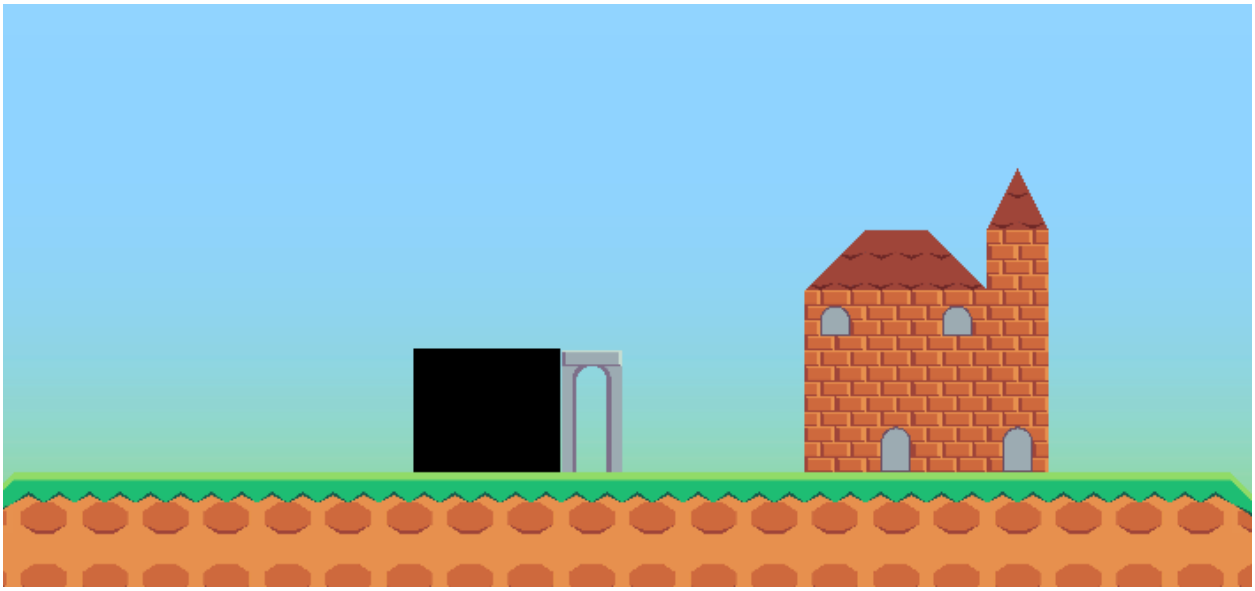
Відповідно до варіанту завдання, були завантажені та використані асети для сцени. Хочу зазначити, що до об'єкту Tilemap, який складається з асетів потрібно додати TilemapCollider2D і CompositeCollider2D, щоб ігровий персонаж не "прилипав боком" до об'єктів асету.

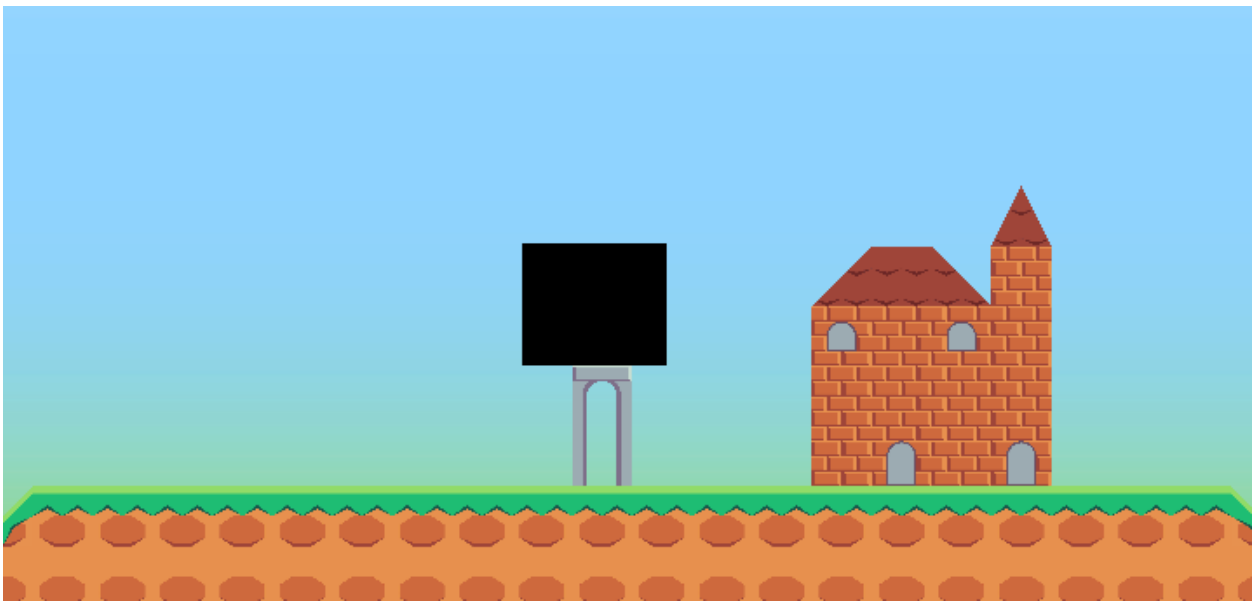
Скріншоти сцени

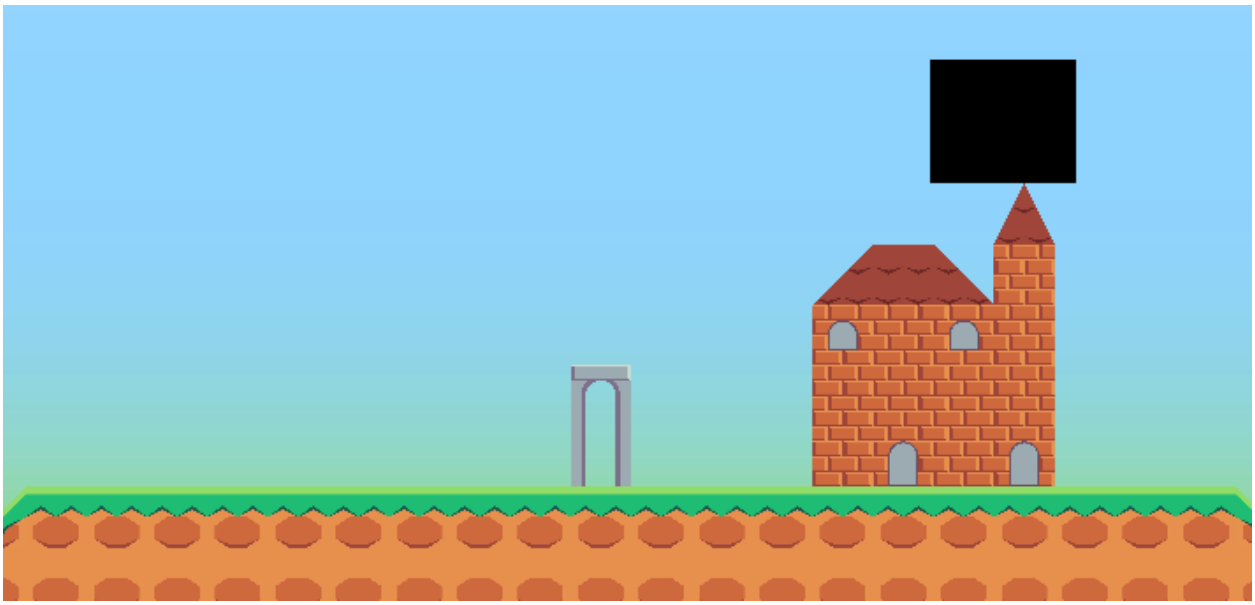
Стрибок персонажа:



Персонаж взаємодіє з об'єктами:







3. Висновок

В ході даної лабораторної роботи, я отримав навички елементарної роботи з Unity, створив одну сцену, де є ігровий персонаж та об'єкти-асети з якими він взаємодіє.

4. Додатки

Посилання на проект: GitHub

Код програми:

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;

public class SquareMovement : MonoBehaviour
{
    private Rigidbody2D rb2d;
    private BoxCollider2D bc2d;

    [SerializeField]
    private int _speed = 20;
    [SerializeField]
    private float jumpForce;
    [SerializeField]
    private LayerMask platformLayerMask;

    void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
        bc2d = GetComponent<BoxCollider2D>();
    }

    public bool IsGrounded()
    {
        RaycastHit2D rh2d = Physics2D.BoxCast(bc2d.bounds.center,
bc2d.bounds.size, 0f, Vector2.down, .1f, platformLayerMask);
        return rh2d.collider != null;
    }

    void Jump()
    {
        rb2d.velocity = Vector2.up * jumpForce;
    }

    void Update()
    {
        float moveX = Input.GetAxis("Horizontal");
        rb2d.velocity = new Vector2(
            moveX * _speed, rb2d.velocity.y
        );


        if (Input.GetKeyDown(KeyCode.Space) && IsGrounded()) {
            Jump();
        }
    }
}
```