Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра Обчислювальної Техніки

**Лабораторна робота №1**

з дисципліни "Розробка ігрових застосувань. Unity рішення"

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

Виконав:
студент групи ІП-93
Домінський Валентин
Олексійович

Перевірив:
Катін Павло Юрійович

Київ 2022

# Зміст

# Мета:

1. 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

# Завдання до роботи:

1. Репозиторій у системі контролю версій. Створити проект 2D. Загальні вимоги. Акаунт на GitHab, на даному етапі за бажанням. Репозиторій на GitHab з проектом. Назва GameProgLab1GroupNum, де зафарбовано номер групи.
2. Установка ігрового рушія. Створений проект IDE (2D) на основі рушія, що містить 1 сцену, ігровий персонаж. Можуть бути включені інші елементи. Розроблений і налагоджений скрипт для управління ігровим персонажем. Достатньо продемонструвати рух ліворуч, праворуч, стрибки, коректну фізику, зупинку перед перешкодою. Проект розташовано у репозиторій на GitHab

# Хід роботи:

Прізвище –> Домінський

Ім’я –> Валентин

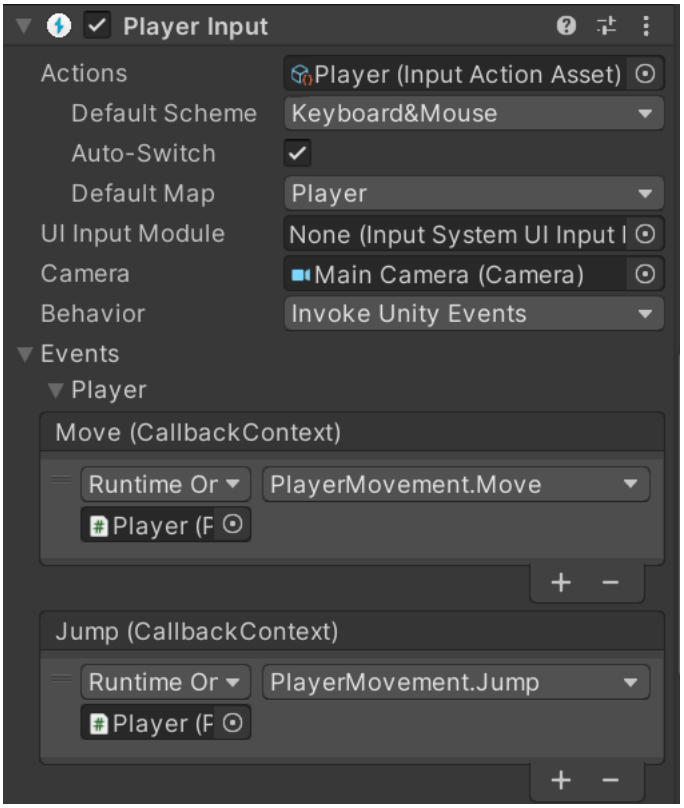
Шифр групи –> ІП-93

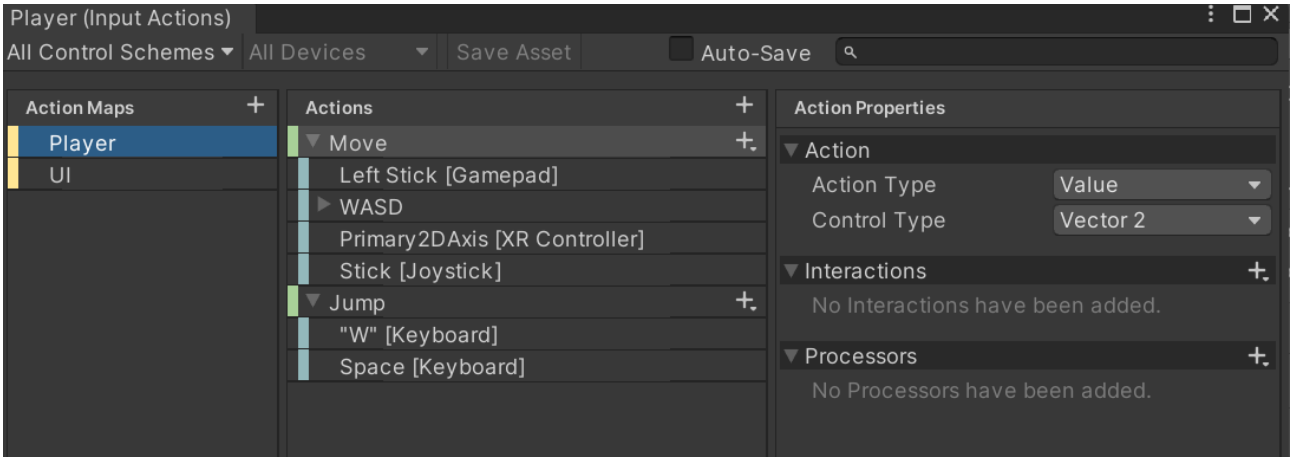
Назва факультету –> ФІОТ

Варіант –> 8 -> 2D Stone Age World Platformer Starter Kit та коло

## Рух:

Для даного завдання Я вирішив використати нову систему руху, яка заснована на подіях. Встановити її можна завдяки Package Manager. Щоб вона працювала, Нам треба створити Bindings файл, де буде прив’язка певних дій до значення, яке повертається та до кнопок:






Також у скрипті руху Ми повинні створити відповідні публічні методи:

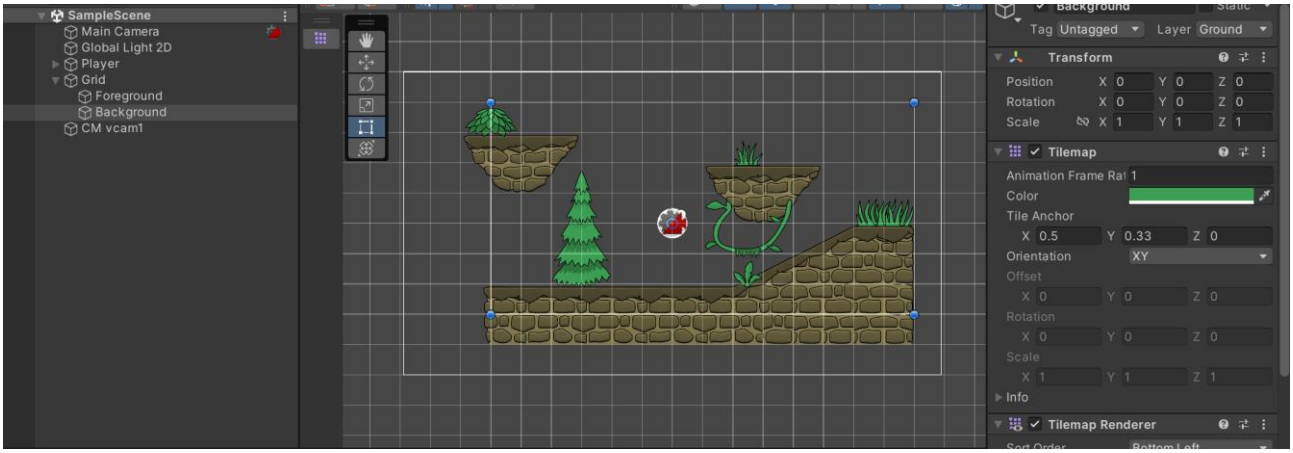
```
0 references | Valentyn-Dominskyi, 10 minutes ago | 1 author, 1 change
public void Move(InputAction.CallbackContext context)
{
    _xMovement = context.ReadValue<Vector2>().x;
}

0 references | Valentyn-Dominskyi, 10 minutes ago | 1 author, 1 change
public void Jump(InputAction.CallbackContext context)
{
    if (context.performed && IsGrounded())
    {
        _rb.velocity = new(_rb.velocity.x, _jumpSpeed);
    }
}
```

Як видно, кожен з них має параметр CallbackContext, який і має значення, які надходять від користувача

Середовище:

Оскільки асет, який Мені випав, ідельно підходив для системи тайлів, то Я вирішив скористатися цією нагодою. Для початку створив Grid, в якому буде 2 tilemap – один для переднього плану (об’єкти з колізією), інший – задній план (через ці об’єкти можна буде спокійно проходити)




На тайли з колізією додав компонент CompositeCollider2D, який об'єднує усі BoxCollider2D та PolygonCollider2D в один колайдер, аби гравець під час руху не застрягав між клітинками

Камера:

Використав Я пакет Cinemachine. Завдяки йому та різним віртуальним камерам Я можу дуже кінематографічно слідкувати за гравцем

## Висновки:

Я вперше спробував попрацювати з ШІМ та кнопкою. Зумів розібратися з проблемою «брязкоту». Пописав цикли (як звичайні, так і зворотні). Розібрався з умовними операторами, функціями користувача. Зрозумів різницю між глобальними змінними та локальними.

## Додатки:

## Вихідний код:

PlayerMovement.cs:

```
using UnityEngine;
using UnityEngine.InputSystem;

namespace Lab1
{
```

```
public class PlayerMovement : MonoBehaviour
{
    [SerializeField]
    private float _movementSpeed = 5f;
    [SerializeField]
    private float _jumpSpeed = 5f;
    [SerializeField]
    private float _groundCheckRadius = 0.3f;
    [SerializeField]
    private LayerMask _groundMask;

    private Vector2 _groundCheckPos;
    private float _colliderRadius;
    private float _xMovement;
    private Rigidbody2D _rb;

    private void Awake()
    {
        _rb = GetComponent<Rigidbody2D>();
        _colliderRadius = GetComponent<CircleCollider2D>().radius;
    }

    private void FixedUpdate()
    {
        _rb.velocity = new(_xMovement * _movementSpeed, _rb.velocity.y);
    }

    private bool IsGrounded()
    {
        _groundCheckPos = new(gameObject.transform.position.x,
            gameObject.transform.position.y - _colliderRadius);
             return Physics2D.OverlapCircle(_groundCheckPos,
             _groundCheckRadius, _groundMask);
    }

    public void Move(InputAction.CallbackContext context)
    {
        _xMovement = context.ReadValue<Vector2>().x;
     }

    public void Jump(InputAction.CallbackContext context)
    {
        if (context.performed && IsGrounded())
        {
            _rb.velocity = new(_rb.velocity.x, _jumpSpeed);
        }
    }
}
}
```

# Посилання:

Проект - посилання