**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 №3**

з дисципліни

"Розробка ігрових застосувань. Unity рішення"

на тему

"Дослідження типових патернів у скриптах ігрового застосунку"

Виконав: Перевірив:

студенти групи ІП-93 Катін П.Ю.

Дмитренко Р.В.

Домінський В.О.

Завальнюк М.Є.

Київ 2022

Зміст

[1. Постановка задачі 3](#_Toc118132297)

[2. Виконання 3](#_Toc118132298)

[3. Висновок 4](#_Toc118132299)

[4. Додатки 4](#_Toc118132300)

# 1. Постановка задачі

**Мета:** дослідити типові патерни у скриптах ігрового застосунку.

**Завдання до роботи:**

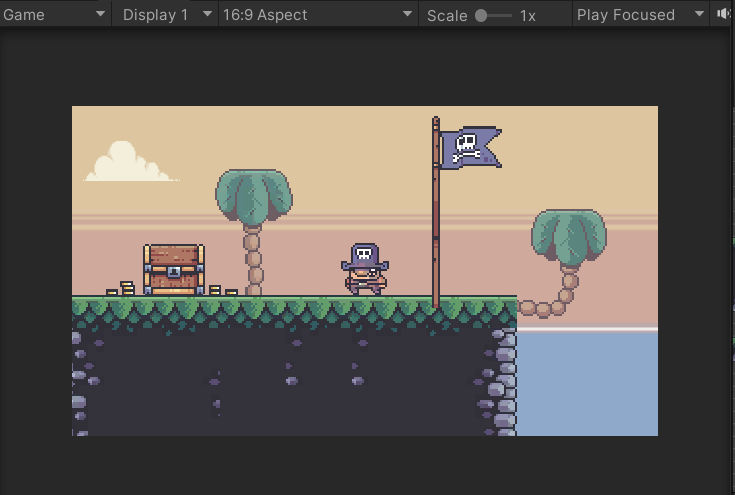
Створений проект IDE (2D або 3D) на основі рушія, що містить 1 сцену, об'єкт і скрипт управління, що виводить повідомлення, по можливості, скрипт керує об'єктом, проект розташовано у репозиторій на GitHub.

У проекті має бути застосований процедурний і ООП варіант скрипт на базі патерну Стан. Основна мета полягає у дослідженні і підтвердженні володіння обраною IDE і технологією розподіленої системи контролю версій.

# 2. Виконання

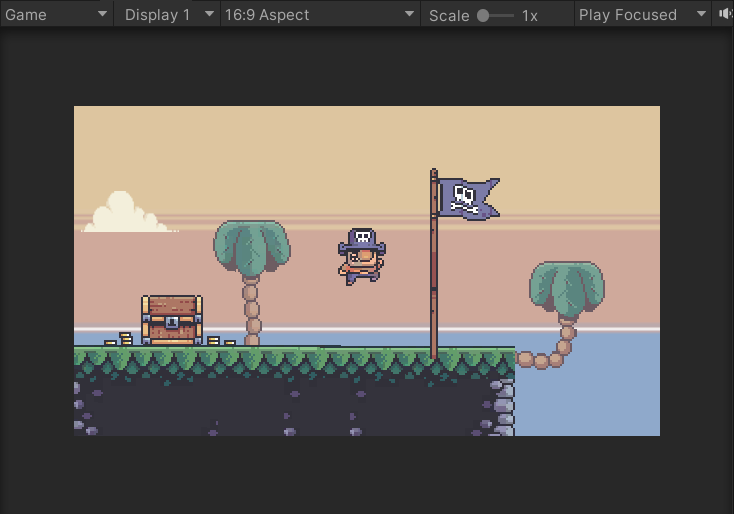
Для початку, за допомогою асета та Tilemap ми намалювали ігровий світ та головного персонажа, який буде з ним взаємодіяти.

*Демонстрація персонажу, який буде взаємодіяти зі світом*



Далі ми реалізували рух 2D персонажа за допомогою RigidBody2D. Скрипт до цього можна знайти в додатках.

*Демонстрація руху персонажа*



Після цього, ми реалізували шаблон State. В нашому випадку, він заключається в тому, що персонажу змінює спрайт залежно від різних дій.

# 3. Висновок

Виконуючи дану лабораторну роботу, ми дослідили типові патерни у скриптах ігрового застосунку.

# 4. Додатки

Код програми:

*PlayerMovement.cs:*  
using UnityEngine;

using UnityEngine.InputSystem;

namespace Labs

{

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 SpriteRenderer \_spriteRenderer;

private Animator \_animator;

private State \_state;

private void Awake()

{

\_rb = GetComponent<Rigidbody2D>();

\_colliderRadius = GetComponent<CircleCollider2D>().radius;

\_spriteRenderer = GetComponent<SpriteRenderer>();

\_animator = GetComponent<Animator>();

}

private void FixedUpdate()

{

\_rb.velocity = new(\_xMovement \* \_movementSpeed, \_rb.velocity.y);

CheckState();

}

private void CheckState()

{

if (!IsGrounded())

{

SetState(State.Jump);

return;

}

if (\_xMovement != 0)

{

SetState(State.Run);

return;

}

SetState(State.Idle);

}

private void SetState(State state)

{

\_state = state;

var stateName = \_state.ToString();

\_animator.Play(stateName);

Debug.Log(stateName);

}

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;

CheckFacingDirection();

}

private void CheckFacingDirection()

{

if (\_xMovement == 0)

{

return;

}

FlipCharacter();

}

private void FlipCharacter()

{

\_spriteRenderer.flipX = \_xMovement < 0;

}

public void Jump(InputAction.CallbackContext context)

{

if (context.performed && IsGrounded())

{

\_rb.velocity = new(\_rb.velocity.x, \_jumpSpeed);

}

}

}

}

*State.cs:*

namespace Labs

{

public enum State

{

Idle,

Run,

Jump

}

}