



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем і технологій

Лабораторна робота №2
Технології розроблення програмного забезпечення
«Діаграма варіантів використання. Сценарії варіантів
використання. Діаграми uml. Діаграми класів.
Концептуальна модель системи»

Тема: «Гра у жанрі RPG»

Виконав:

Студент групи ІА-23

Ширяєв Д. Ю.

Перевірив:

Мягкий М. Ю.

Зміст

Діаграма прецедентів	3
Сценарії варіантів використання	4
Діаграма класів	7
Вихідні коди класів системи	8
Висновок	14

Діаграма прецедентів

Діаграма прецедентів для обраної теми зображена на рисунку 1:

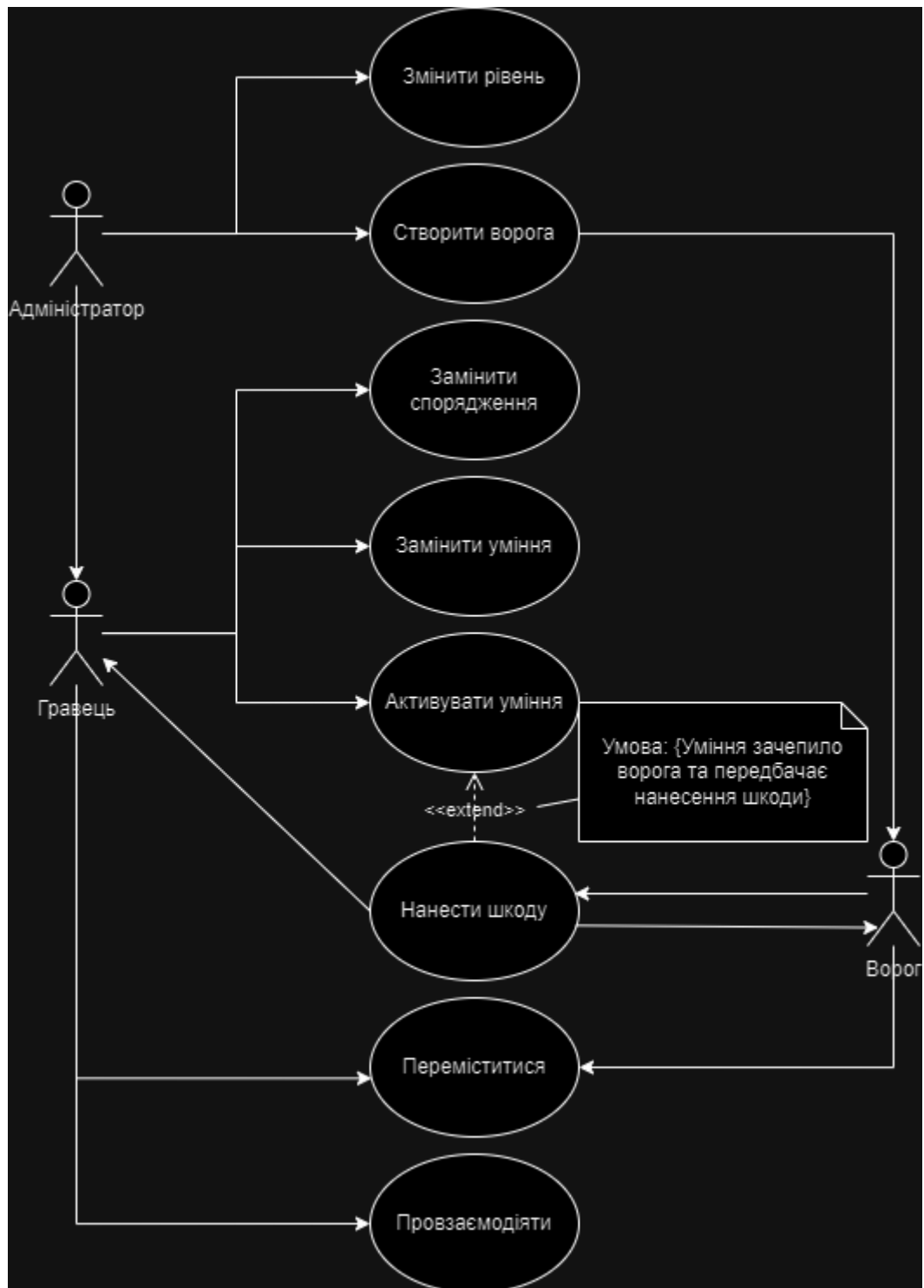


Рис. 1 – Діаграма прецедентів для гри у жанрі RPG

Сценарії варіантів використання

Для опису сценаріїв було обрано 3 варіанти використання системи:

1. Активувати уміння:

Передумови: Гравець у бойовому режимі.

Постумови: У випадку успішного виконання уміння активує свій ефект. У протилежному випадку на користувацькому інтерфейсі відобразиться повідомлення про причину невдачі.

Взаємодіючі сторони: Гравець, Система навичок.

Короткий опис: Даний варіант використання описує використання гравцем системи навичок.

Основний розвиток подій: Даний варіант використання починає виконуватись, коли гравець натискає на кнопку відповідного уміння.

1. Система навичок перевіряє умови активації уміння. Якщо персонаж гравця вже виконує інше уміння, Виняток №1. Якщо уміння на перезарядці, Виняток №2. Якщо персонажу не вистачає ресурсу на виконання уміння, Виняток №3.
2. З'являється область дії уміння, що дозволяє гравцю прицілитись у необхідне місце.
3. Гравець відпускає кнопку уміння.
4. Уміння активує свій ефект.

Винятки:

Виняток №1: Виконання іншого уміння. Якщо виявиться, що персонаж гравця наразі виконує інше уміння, система виведе повідомлення, що персонаж зараз зайнятий. Персонаж при цьому продовжить виконання попереднього уміння.

Виняток №2: Перезарядка. Якщо виявиться, що уміння, яке гравець намагається активувати перезаряджається, система виведе повідомлення про це.

Виняток №3: Нестача ресурсу. Якщо виявиться, що персонажу гравця не вистачає ресурсу на активацію уміння, система виведе повідомлення про це.

Примітки: Немає.

2. Переміститися:

Передумови: Немає.

Постумови: У випадку успішного розвитку подій персонаж гравця переміщається у вказаному напрямку. У протилежному випадку персонаж не зрушиться.

Взаємодіючі сторони: Гравець, Система управління персонажем, Система ефектів.

Короткий опис: Даний варіант використання описує переміщення персонажу гравця по рівню.

Основний розвиток подій: Даний варіант використання починає виконуватись, коли гравець натискає кнопки переміщення.

1. Система ефектів перевіряє, чи знаходиться персонаж гравця під впливом негативних ефектів, що його знерухомлюють. Якщо такі ефекти присутні, Виняток №1.
2. Система переміщення зчитує натискання гравця.
3. Персонаж гравця переміщується у вказаному напрямку на деяку відстань.

Винятки:

Виняток №1: Ефекти знерухомлення. Якщо виявиться, що персонаж гравця під впливом ефекту знерухомлення, він залишиться на місці.

Примітки:

Немає.

3. Провзаємодіяти:

Передумови: Гравець у звичайному режимі. Персонаж знаходиться поряд з об'єктом, з яким можна провзаємодіяти.

Постумови: У випадку успішного розвитку подій з'явиться вікно взаємодії з об'єктом. У протилежному випадку з'явиться повідомлення про причину неможливості взаємодії.

Взаємодіючі події: Гравець, Об'єкт взаємодії.

Короткий опис: Даний варіант використання описує взаємодію гравця з деяким об'єктом.

Основний розвиток подій: Даний варіант використання починає виконуватись, коли гравець натискає лівою кнопкою миші на об'єкт.

1. Об'єкт взаємодії перевіряє умови, чи можна з ним зараз взаємодіяти. Якщо персонаж гравця ще не має доступу до цієї взаємодії, Виняток №1. Якщо гравець вже взаємодіє з вказаним об'єктом, Виняток №2.
2. З'являється вікно з передбаченими варіантами взаємодії.

Винятки:

Виняток №1: Немає доступу. Якщо виявиться, що персонаж гравця не має доступу до взаємодії, з'явиться повідомлення, що він наразі не може цього зробити.

Виняток №2: Повторна взаємодія. Якщо виявиться, що гравець вже взаємодіє з вказаним об'єктом, вікно взаємодії підсвітиться.

Примітки:

Немає.

Діаграма класів

Діаграма класів для обраної теми зображена на рисунку 2:

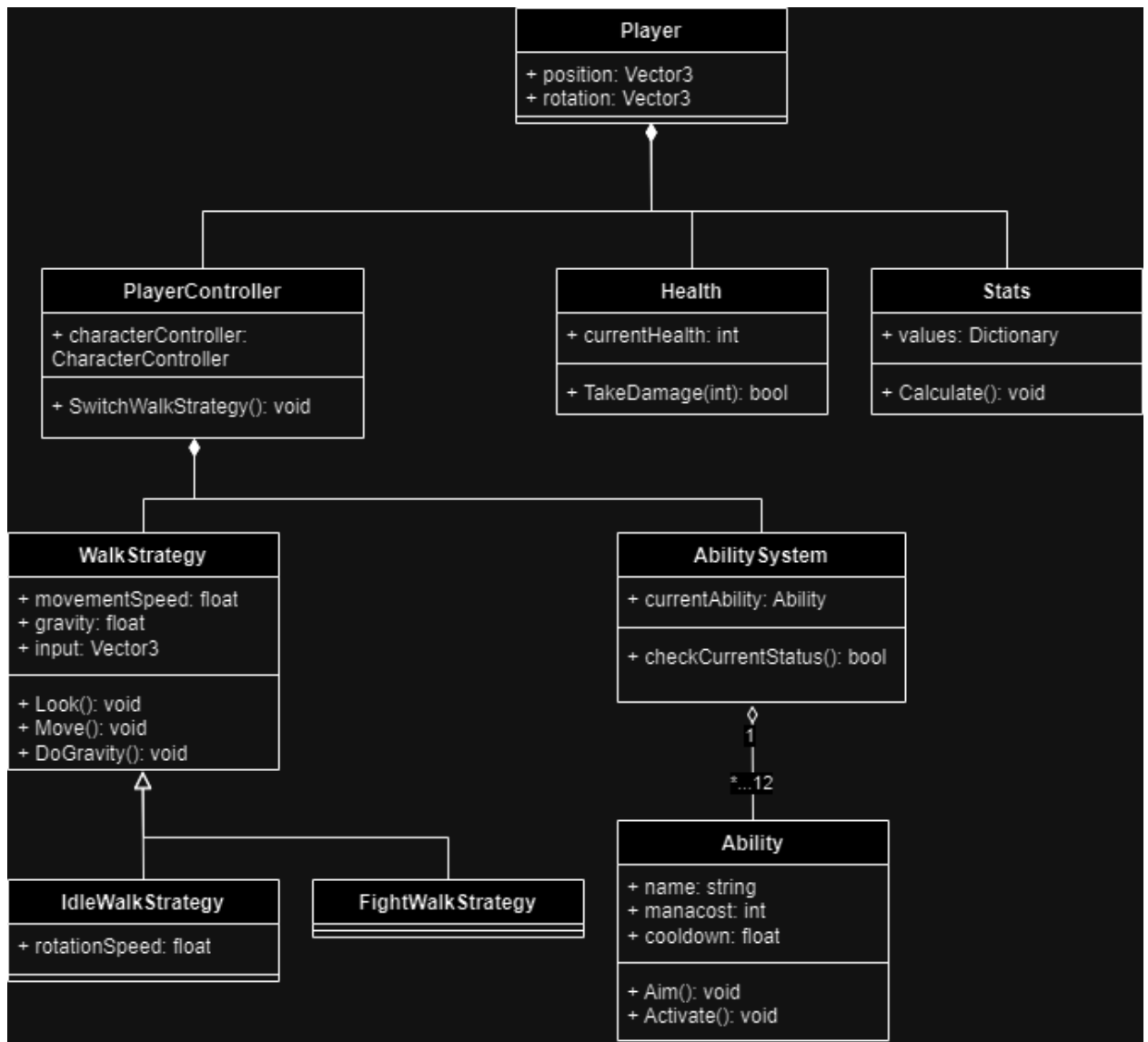


Рис. 2 – Діаграма класів для гри у жанрі RPG

Вихідні коди класів системи

Вихідний код класу PlayerController:

```
1 using UnityEngine;
2
3 [RequireComponent(typeof(CharacterController))]
4 public class PlayerController : MonoBehaviour
5 {
6     [SerializeField] private CharacterController _characterController;
7     [SerializeField] private Animator _animator;
8     private WalkStrategy _walkStrategy;
9     private WalkStrategy _fightWalkStrategy;
10    private WalkStrategy _idleWalkStrategy;
11
12    private void Awake()
13    {
14        _characterController = GetComponent<CharacterController>();
15        _fightWalkStrategy = new FightWalkStrategy(_characterController, transform, _animator);
16        _idleWalkStrategy = new IdleWalkStrategy(_characterController, transform, _animator);
17        _walkStrategy = _idleWalkStrategy;
18    }
19
20    private void Update()
21    {
22        if (Input.GetKeyDown(KeyCode.Tab))
23        {
24            SwitchWalkStrategy();
25        }
26        _walkStrategy.Look();
27    }
28
29    private void FixedUpdate()
30    {
31        _walkStrategy.Move();
32    }
33
34    private void SwitchWalkStrategy()
35    {
36        if (_walkStrategy == _idleWalkStrategy)
37        {
38            _walkStrategy = _fightWalkStrategy;
39        }
40        else
41        {
42            _walkStrategy = _idleWalkStrategy;
43        }
44    }
45 }
```


Вихідний код WalkStrategy:

```
1  public interface WalkStrategy
2  {
3      public void Look();
4      public void Move();
5      public void DoGravity();
6  }
7
```

Вихідний код IdleWalkStrategy:

```
using UnityEngine;

public class IdleWalkStrategy : WalkStrategy
{
    [SerializeField] private float _movementSpeed = 10.0f;
    [SerializeField] private float _gravity = -5f;
    [SerializeField] private float _velocity = -0.5f;
    [SerializeField] private float _rotationSpeed = 10f;
    private Vector3 _input;
    private CharacterController _characterController;
    private Transform _transform;
    private Animator _animator;

    public IdleWalkStrategy(CharacterController characterController, Transform transform, Animator animator)
    {
        _characterController = characterController;
        _transform = transform;
        _animator = animator;
    }

    public void Look()
    {
        if (_input.x != 0 || _input.z != 0)
        {
            var rotation = Quaternion.LookRotation(new Vector3(_input.x, 0, _input.z), Vector3.up);
            _transform.rotation = Quaternion.Lerp(_transform.rotation, rotation, _rotationSpeed * Time.deltaTime);
        }
    }
}
```

```
29  Ссылка: 1
30  public void DoGravity()
31  {
32      if (_characterController.isGrounded)
33      {
34          if (_velocity < -0.5f)
35          {
36              _animator.SetBool("IsFalling", false);
37              _animator.SetBool("IsGround", true);
38          }
39      }
40      else
41      {
42          _animator.SetBool("IsGround", false);
43          _velocity = -0.5f;
44      }
45      else
46      {
47          _velocity += _gravity * Time.fixedDeltaTime;
48          _animator.SetBool("IsFalling", true);
49      }
50  }
51  Ссылка: 2
52  public void Move()
53  {
54      DoGravity();
55      _input = new Vector3(Input.GetAxis("Horizontal"), _velocity, Input.GetAxis("Vertical"));
56      if (_input.x == 0 && _input.z == 0)
57      {
58          _animator.SetBool("IsWalk", false);
59      }
60      else
61      {
62          _animator.SetBool("IsWalk", true);
63      }
64      _characterController.Move(_input * Time.fixedDeltaTime * _movementSpeed);
65  }
66  }
```

Вихідний код FightWalkStrategy:

```
1  using UnityEngine;
2
3  public class FightWalkStrategy : WalkStrategy
4  {
5      [SerializeField] private float _movementSpeed = 7.0f;
6      [SerializeField] private float _gravity = -5f;
7      [SerializeField] private float _velocity = -0.5f;
8      private Vector3 _input;
9      private CharacterController _characterController;
10     private Transform _transform;
11     private Animator _animator;
12     private int _groundLayer = 1 << 6;
13
14
15     public FightWalkStrategy(CharacterController characterController, Transform transform, Animator animator)
16     {
17         _characterController = characterController;
18         _transform = transform;
19         _animator = animator;
20     }
21
22     public void Look()
23     {
24         var worldMousePosition = Camera.main.ScreenPointToRay(Input.mousePosition);
25         RaycastHit hit;
26         if (Physics.Raycast(worldMousePosition, out hit, 50f, _groundLayer))
27         {
28             _transform.LookAt(new Vector3(hit.point.x, _transform.position.y, hit.point.z));
29         }
30     }
31
32     Ссылка: 1
```

```
30     Ссылка: 1
31     public void DoGravity()
32     {
33         if (_characterController.isGrounded)
34         {
35             if (_velocity < -0.5f)
36             {
37                 _animator.SetBool("IsFalling", false);
38                 _animator.SetBool("IsGround", true);
39             }
40             else
41             {
42                 _animator.SetBool("IsGround", false);
43             }
44             _velocity = -0.5f;
45         }
46         else
47         {
48             _velocity += _gravity * Time.fixedDeltaTime;
49             _animator.SetBool("IsFalling", true);
50         }
51     }
52     Ссылка: 2
53     public void Move()
54     {
55         DoGravity();
56         _input = new Vector3(Input.GetAxis("Horizontal"), _velocity, Input.GetAxis("Vertical"));
57         if (_input.x == 0 && _input.z == 0)
58         {
59             _animator.SetBool("IsWalk", false);
60         }
61         else
62         {
63             _animator.SetBool("IsWalk", true);
64         }
65         _characterController.Move(_input * Time.fixedDeltaTime * _movementSpeed);
66     }
67 }
```

Вихідний код AbilitySystem:

```
1 using UnityEngine;
2
3 Скрипт Unity (1 ссылка на ресурсы) | Ссылок: 0
4 public class AbilitySystem : MonoBehaviour
5 {
6     [SerializeField] private Ability LMB_Ability;
7
8     Сообщение Unity | Ссылки: 0
9     private void Awake()
10     {
11         Animator animator = GetComponent<Animator>();
12         LMB_Ability = Resources.Load<Ability>("Ability/SwordSlash");
13         LMB_Ability.OnEquip(transform, animator);
14     }
15
16     Сообщение Unity | Ссылки: 0
17     private void Update()
18     {
19         if (Input.GetKeyDown(KeyCode.Mouse0))
20         {
21             LMB_Ability.Aim();
22         }
23         if (Input.GetKeyUp(KeyCode.Mouse0))
24         {
25             LMB_Ability.Activate();
26         }
27     }
28 }
```

Вихідний код Ability:

```
Assembly-CSharp Ability
1 using UnityEngine;
2 Скрипт Unity | Ссылки: 3
3 public abstract class Ability : ScriptableObject
4 {
5     [SerializeField] protected new string name;
6     [SerializeField] protected int manacost;
7     [SerializeField] protected float cooldown;
8     protected Transform ownerTransform;
9     protected Animator animator;
10     Ссылки: 3
11     public virtual void OnEquip(Transform transform, Animator animator)
12     {
13         ownerTransform = transform;
14         this.animator = animator;
15     }
16
17     Ссылки: 2
18     public abstract void Aim();
19
20     Ссылки: 2
21     public abstract void Activate();
22 }
```

Вихідний код Health:

```
1  using UnityEngine;
2
3  Ссылка: 0
4  public class Health
5  {
6
7      [SerializeField] private int currentHealth;
8
9      Ссылка: 0
10     public bool TakeDamage(int damage)
11     {
12         if (currentHealth <= damage)
13         {
14             currentHealth = 0;
15             return true;
16         }
17         else {
18             currentHealth -= damage;
19             return false;
20         }
21     }
22 }
```

Вихідний код Stats:

```
1  using System.Collections.Generic;
2
3  Ссылка: 0
4  public abstract class Stats
5  {
6      private Dictionary<string, int> _values = new Dictionary<string, int>();
7      Ссылка: 0
8      public Dictionary<string, int> Values => _values;
9
10     Ссылка: 0
11     protected abstract void Calculate();
12 }
```

Висновок

В ході виконання даної лабораторної роботи було розроблено базову структуру застосунку на тему «Гра у жанрі RPG». Для опису цієї структури на її основі побудовано діаграму прецедентів та діаграму класів, які у подальшому можуть бути поглиблені.

Спираючись на особливості обраної теми та середовища розробки Unity, інтеграція бази даних не є доцільною, оскільки необхідні для збереження об'єкти знаходяться безпосередньо у файлах застосунку у вигляді «Prefab», а збереження прогресу та налаштувань гравця реалізовується через Json.