



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 №8
Технології розроблення програмного забезпечення
«Шаблони «Composite», «Flyweight», «Interpreter»,
«Visitor»»

Тема: «Гра у жанрі RPG»

Виконав:

Студент групи ІА-23

Ширяєв Д. Ю.

Перевірив:

Мягкий М. Ю.

Київ 2024

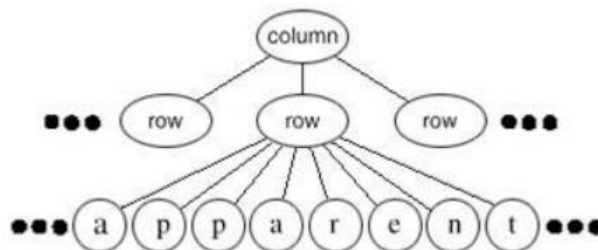
Зміст

| | |
|-----------------|---|
| Вступ..... | 3 |
| Реалізація..... | 4 |
| Висновок | 8 |

Вступ

В четвертій лабораторній роботі було розроблено систему умінь для реалізації шаблону Стратегія. Але також уміння з однаковим ефектом можуть мати різні значення параметрів (більше шкоди, більша дальність дії тощо), які заздалегідь завдаються і балансуються геймдизайнером. Для цього усі уміння наслідуються від класу ScriptableObject, який дозволяє через редактор Unity створювати, налаштовувати і завантажувати екземпляри класу. Такі уміння можуть одночасно використовуватись в різних точках програми (інвентар, гравці), тому важливо забезпечити спільні параметри незмінними, а індивідуальні незалежними один від одного. Наприклад, щоб при активації уміння одною сутністю перезарядка починалась лише в неї, а не в усіх одразу. Схожу задачу має система екіпірування, оскільки предмети працюють по такому ж принципу. Заради досягнення цієї мети обрано шаблон Flyweight, який виділяє спільні риси однакових предметів/умінь в окремий клас, а різні риси оброблюються індивідуально.

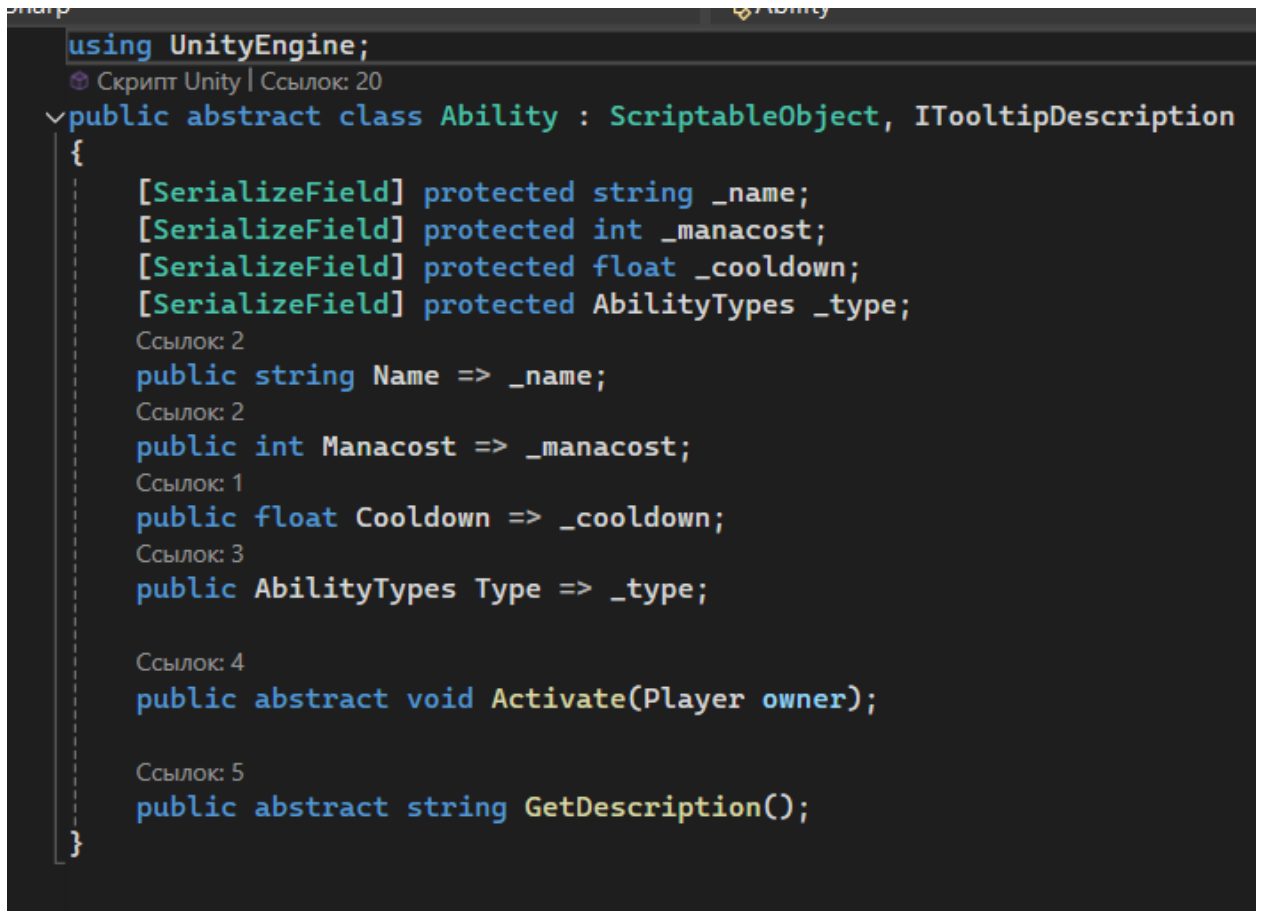
Шаблон використовується для зменшення кількості об'єктів в додатку шляхом поділу цих об'єктів між ділянками додатку. Flyweight являє собою поділюваний об'єкт. Хорошим прикладом є наступне зображення:



Здається, ніби для кожної літери існує окремий об'єкт. Насправді фізично об'єкт всього один, існує лише безліч посилань на нього. Дуже важливою є концепція «внутрішнього» і «зовнішнього» станів. Внутрішній стан відображає дані, характерні саме поділюваному об'єкту (наприклад, код букви); зовнішній стан несе інформацію про його застосування в додатку (наприклад, рядок і стовпчик). Внутрішній стан зберігається в самому поділюваному об'єкті, зовнішній - в об'єктах додатку (контексту використання поділюваного об'єкта).

Реалізація

На рисунках 1-2 наведено клас уміння та його застосування у системі умінь:



```
using UnityEngine;

Скрипт Unity | Ссылка: 20
public abstract class Ability : ScriptableObject, ITooltipDescription
{
    [SerializeField] protected string _name;
    [SerializeField] protected int _manacost;
    [SerializeField] protected float _cooldown;
    [SerializeField] protected AbilityTypes _type;

    Ссылка: 2
    public string Name => _name;

    Ссылка: 2
    public int Manacost => _manacost;

    Ссылка: 1
    public float Cooldown => _cooldown;

    Ссылка: 3
    public AbilityTypes Type => _type;

    Ссылка: 4
    public abstract void Activate(Player owner);

    Ссылка: 5
    public abstract string GetDescription();
}
```

Рис. 1 – Ability

Цей абстрактний клас є спільним для кожного виду уміння і оголошує його параметри і поведінку.

```

Ссылка: 4
public class AbilityHolder{
    private float _currentCooldown = 0;
    private Ability _ability;
    private Player _owner;
    Ссылка: 1
    public AbilityHolder(Ability ability, Player owner)
    {
        _owner = owner;
        SetAbility(ability);
    }
    Ссылка: 1
    public void TryActivate()
    {
        if (_owner.Mana >= _ability.Manacost && _currentCooldown == 0)
        {
            _ability.Activate(_owner);
            _currentCooldown = _ability.Cooldown;
            _owner.ConsumeMana(_ability.Manacost);
        }
    }
    Ссылка: 2
    public void SetAbility(Ability ability)
    {
        _ability = ability;
    }
    Ссылка: 0
    public Ability GetAbility()
    {
        return _ability;
    }
    Ссылка: 1
    public void ReduceCooldown(float cooldown)
    {
        if (_currentCooldown < cooldown){
            _currentCooldown = 0;
        }
        else{
            _currentCooldown -= cooldown;
        }
    }
}

```

Рис. 2 – AbilityHolder

Цей клас вміщує в собі посилання на уміння, але також зберігає індивідуальні параметри, такі як: власник та поточна перезарядка. Також цей клас керує активацією уміння. Таким чином різні об'єкти AbilityHolder можуть посилатися на однакове уміння, але мати різних власників та перезарядку, створюючи враження, що і об'єкти Ability є різними.

На рисунках 3-4 зображено клас екіпірування та його використання у системі інвентаря:

```
[CreateAssetMenu(fileName = "Equipment")]

public class Equipment: ScriptableObject, ITooltipDescription
{
    [SerializeField] private string _name;
    [SerializeField] private Stats _mainStat;
    [SerializeField] private Stats _secondaryStat;
    [SerializeField] private int _mainStatValue;
    [SerializeField] private int _secondaryStatValue;
    [SerializeField] private Types _type;
    Ссылка: 3
    public string Name => _name;
    Ссылка: 3
    public Stats MainStat => _mainStat;
    Ссылка: 3
    public Stats SecondaryStat => _secondaryStat;
    Ссылка: 3
    public int MainStatValue => _mainStatValue;
    Ссылка: 3
    public int SecondaryStatValue => _secondaryStatValue;
    Ссылка: 4
    public Types Type => _type;
    Ссылка: 8
    public enum Types
    {
        Weapon,
        Helmet,
        Chestplate,
        Boots,
    }
    Ссылка: 2
    public string GetDescription()
    {
        return $"{_name}\n {_type}\n {_mainStat}:{_mainStatValue}\n {_secondaryStat}:{_secondaryStatValue}";
    }
}
```

Рис. 3 – Equipment

Цей клас оголошує параметри предметів та спосіб їх опису.

```

using TMPro;
using UnityEngine;
using UnityEngine.EventSystems;

Скрипт Unity | Ссылка: 3
public abstract class InventorySlot : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IPointerClickHandler
{
    protected ITooltipDescription _content;
    protected Inventory _inventory;

    Ссылка: 0
    public void OnPointerEnter(PointerEventData eventData)
    {
        _inventory.ShowTooltip(_content.GetDescription(), eventData.position);
    }

    Ссылка: 0
    public void OnPointerExit(PointerEventData eventData)
    {
        _inventory.HideTooltip();
    }

    Ссылка: 5
    public ITooltipDescription GetContent()
    {
        return _content;
    }

    Ссылка: 9
    public virtual void InitSlot(ITooltipDescription content, Inventory inventory) {
        _inventory = inventory;
        _content = content;
        GetComponent<TextMeshProUGUI>().text = content.Name;
    }

    Ссылка: 3
    public abstract void OnPointerClick(PointerEventData eventData);
}

```

```

using UnityEngine.EventSystems;

Скрипт Unity (1 ссылка на ресурсы) | Ссылка: 6
public class EquipmentSlot : InventorySlot
{
    protected new Equipment _content;

    Ссылка: 5
    public override void InitSlot(ITooltipDescription content, Inventory inventory)
    {
        _content = (Equipment)content;
        base.InitSlot(content, inventory);
    }

    Ссылка: 1
    public override void OnPointerClick(PointerEventData eventData)
    {
        _inventory.EquipEquipment(this);
    }
}

```

Рис. 4 – EquipmentSlot

Цей клас вміщує в собі посилання на об'єкт екіпірування та керує виводом його опису на екран, або ж одяганням на персонажа. Відповідно, при наявності в інвентарі гравця однакових предметів, вони будуть відображатися в окремих слотах, але мати посилання на один і той самий об'єкт, замість створення ідентичних об'єктів в пам'яті.

Висновок

В ході виконання даної лабораторної роботи було реалізовано шаблон проєктування Flyweight для оптимізації пам'яті під час використання однакових об'єктів у системах умінь та інвентаря.