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 №5

Технології розроблення програмного забезпечення

«Шаблони «adapter», «builder», «command», «chain of responsibility», «prototype»»

Тема: «Гра у жанрі RPG»

Виконав:

Студент групи ІА-23

Ширяєв Д. Ю.

Перевірив:

Мягкий М. Ю.

Київ 2024

Зміст

Вступ

В ході розробки системи було створено ворогів, з якими персонаж гравця має боротись. Кожен тип ворога має різні стани та характеристики в процесі існування, але при створені вони повинні мати однакову конфігурацію. Для забезпечення цієї цілі було обрано застосувати шаблон «Прототип».

Шаблон "prototype" (прототип) використовується для створення об'єктів за "шаблоном" (чи "кресленню", "ескізу") шляхом копіювання шаблонного об'єкту. Для цього визначається метод "клонувати" в об'єктах цього класу.

Цей шаблон зручно використати, коли заздалегідь відомо як виглядатиме кінцевий об'єкт (мінімізується кількість змін до об'єкту шляхом створення шаблону), а також для видалення необхідності створення об'єкту - створення відбувається за рахунок клонування, і зухвалій програмі абсолютно немає необхідності знати, як створювати об'єкт.

Також, це дозволяє маніпулювати об'єктами під час виконання програми шляхом настроювання відповідних шаблонів; значно зменшується ієрархія спадкоємства (оскільки в іншому випадку це були б не шаблони, а вкладені класи, що наслідують).

Реалізація

На рисунках 1-3 наведено класи, які реалізують цей шаблон:

```
Ссылок: 20
public class EnemyData : ISpawnable
{
    public int maxHealth;
    public int damage;
    public int currentHealth;
    public int speed;
    public bool isAttacking = false;
    public string type;

    Ссылок: 5
    public EnemyData(int maxHealth, int speed, int damage)
    {
        this.maxHealth = maxHealth;
        this.speed = speed;
        this.damage = damage;
        currentHealth = maxHealth;
        type = "Skeleton_Minion";
    }
    Ссылок: 5
    public virtual ISpawnable Clone()
    {
        return new EnemyData(maxHealth, speed, damage);
    }

    Ссылок: 3
    public virtual IEnumerator Attack(Transform targetPosition, Enemy owner)
    {
        IHealth target = targetPosition.GetComponent<IHealth>();
        while (true)
        {
            if ((targetPosition.position - owner.transform.position).magnitude <= 2)
            {
                isAttacking = true;
                EventBus.OnAttacking(owner);
                yield return new WaitForSeconds(0.5f);
                DealDamage(target, owner);
                yield return new WaitForSeconds(0.7f);
                isAttacking = false;
                yield return new WaitForSeconds(2f);
            }
            yield return new WaitForSeconds(1f);
        }
    }

    Ссылок: 4
    protected virtual void DealDamage(IHealth target, Enemy owner)
    {
        Collider[] colliders = Physics.OverlapBox(owner.transform.position + owner.transform.forward * 2 + Vector3.up,
            new Vector3(1, 2, 1), owner.transform.rotation, 1 << 3);
        if (colliders.Any())
        {
            foreach (Collider collider in colliders)
            {
                if (collider.gameObject.GetComponent<IHealth>() == target)
                {
                    target.TakeDamage(damage);
                }
            }
        }
    }
```

Рис. 1 – EnemyData

Клас базового ворога. Реалізує інтерфейс ISpawnable, який має єдиний метод Clone(), що повертає склонований об’єкт.

```
Ссылок: 2
public WarriorEnemyData(int maxHealth, int speed, int damage) : base(maxHealth, speed, damage)
{
    type = "Skeleton_Warrior";
}
Ссылок: 2
public override IEnumerator Attack(Transform targetPosition, Enemy owner)
{
    IHealth target = targetPosition.GetComponent<IHealth>();
    while (true)
    {
        if ((targetPosition.position - owner.transform.position).magnitude <= 3)
        {
            isAttacking = true;
            EventBus.OnAttacking(owner);
            yield return new WaitForSeconds(1.0f);
            DealDamage(target, owner);
            yield return new WaitForSeconds(0.6f);
            DealDamage(target, owner);
            yield return new WaitForSeconds(0.7f);
            isAttacking = false;
            yield return new WaitForSeconds(2f);
        }
        yield return new WaitForSeconds(1f);
    }
}
Ссылок: 4
protected override void DealDamage(IHealth target, Enemy owner)
{
    Collider[] colliders = Physics.OverlapSphere(owner.transform.position,
            2.5f, 1 << 3);
    if (colliders.Any())
    {
        foreach (Collider collider in colliders)
        {
            if (collider.gameObject.GetComponent<IHealth>() == target)
            {
                target.TakeDamage(damage);
            }
        }
    }
}
Ссылок: 4
public override ISpawnable Clone()
{
    return new WarriorEnemyData(maxHealth, speed, damage);
}
}
```

Рис. 3 – WarriorEnemyData

Ворог-воїн. У відповідь на виклик Clone(), повертає клона свого ж типу.

```
Ссылок: 2
public NecromancerEnemyData(int maxHealth, int speed, int damage) : base(maxHealth, speed, damage)
{
    _spawnCapacity = 3;
    _spawnedEntities = new List<GameObject>();
    _spawnableEntities = new List<EnemyData>();
    _spawnableEntities.Add(new EnemyData(10, 5, 5));
    type = "Skeleton_Mage";
}
```

EnemyManager.cs | EnemyData.cs | WarriorEnemyData.cs | **NecromancerEnemyData.cs** | Enemy.cs

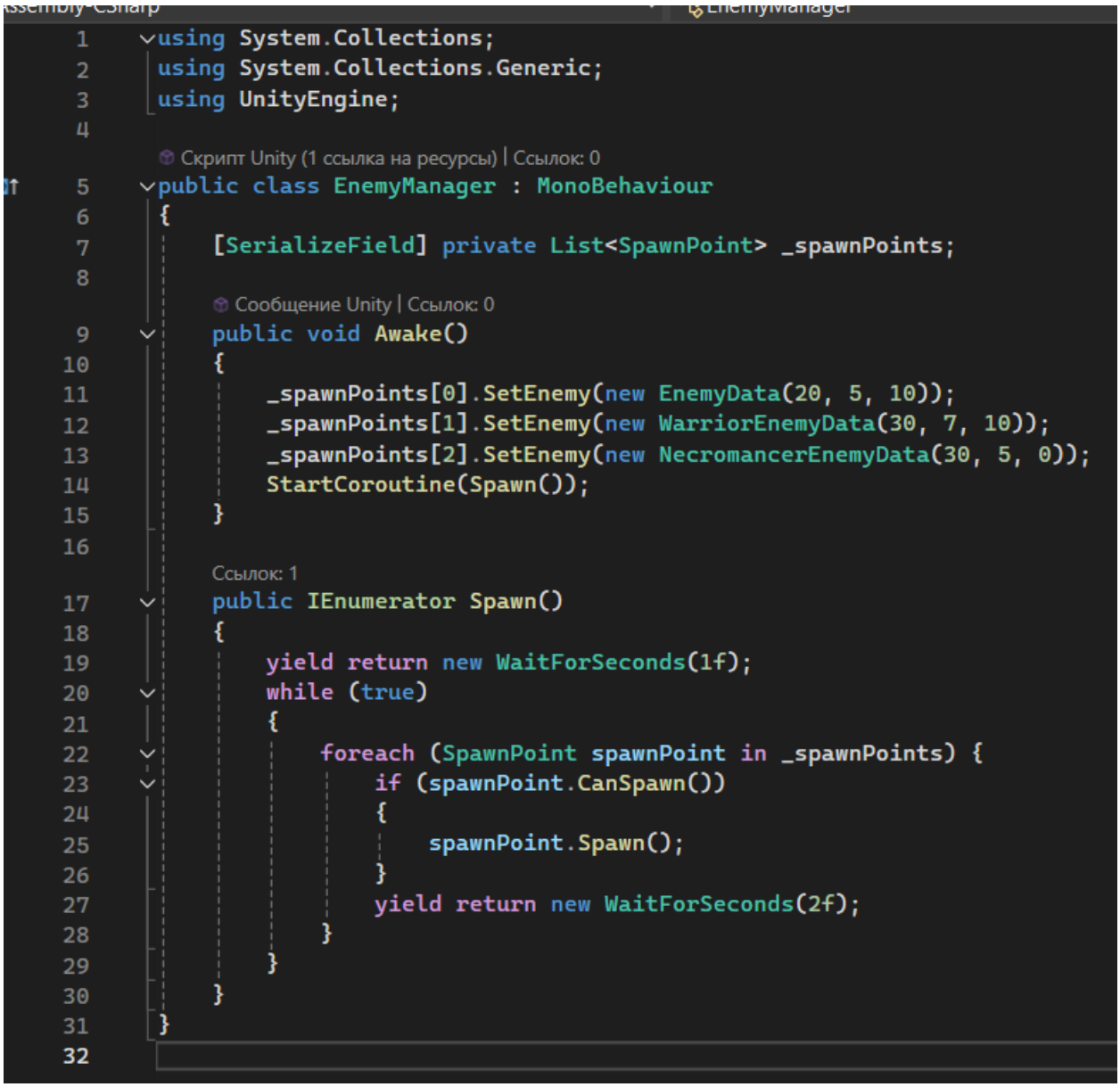
Assembly-CSharp | NecromancerEnemyData

```
            yield return new WaitForSeconds(2f);
        }
        yield return new WaitForSeconds(1f);
    }
}

Ссылок: 2
public bool CanSpawn()
{
    if(_spawnedEntities.Count < _spawnCapacity)
    {
        return true;
    }
    foreach (GameObject entity in _spawnedEntities)
    {
        if (!entity.activeSelf)
        {
            return true;
        }
    }
    return false;
}

Ссылок: 1
public void Spawn(Enemy owner)
{
    if(_spawnedEntities.Count < _spawnCapacity)
    {
        GameObject minion = Resources.Load<GameObject>($"Enemy/{_spawnableEntities[0].type}");
        GameObject SpawnedMinion = Object.Instantiate(minion);
        _spawnedEntities.Add(SpawnedMinion);
    }
    foreach (GameObject entity in _spawnedEntities) {
        if (!entity.activeSelf) {
            entity.transform.position = owner.transform.position + owner.transform.forward * 3;
            entity.GetComponent<Enemy>().SetData((EnemyData)_spawnableEntities[0].Clone());
            break;
        }
    }
}
Ссылок: 4
public override ISpawnable Clone()
{
    return new NecromancerEnemyData(maxHealth, speed, damage);
}
Ссылок: 1
```

Рис. 3 – NecromancerEnemyData

Ворог, який може породжувати інших. На виклик Clone() повертає клон свого типу, а також може викликати Clone() у збереженого шаблону ворога. По замовчуванню створює базового ворога з характеристиками 10, 5, 5.

На рисунку 4 зображено об’єкт менеджеру ворогів, який відповідає за їх створення:



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Скрипт Unity (1 ссылка на ресурсы) | Ссылок: 0
public class EnemyManager : MonoBehaviour
{
    [SerializeField] private List<SpawnPoint> _spawnPoints;

    Сообщение Unity | Ссылок: 0
    public void Awake()
    {
        _spawnPoints[0].SetEnemy(new EnemyData(20, 5, 10));
        _spawnPoints[1].SetEnemy(new WarriorEnemyData(30, 7, 10));
        _spawnPoints[2].SetEnemy(new NecromancerEnemyData(30, 5, 0));
        StartCoroutine(Spawn());
    }

    Ссылок: 1
    public IEnumerator Spawn()
    {
        yield return new WaitForSeconds(1f);
        while (true)
        {
            foreach (SpawnPoint spawnPoint in _spawnPoints) {
                if (spawnPoint.CanSpawn())
                {
                    spawnPoint.Spawn();
                }
                yield return new WaitForSeconds(2f);
            }
        }
    }
}
```

Рис. 4 – EnemyManager

Зберігає список точок спавну, в яких по черзі кожні 2 секунди створює ворогів відповідного типу. За допомогою шаблону «Прототип» та наслідування кожна конфігурувати кожну точку спавну, щоб вона породжувала ворогів по конкретно вказаному шаблону.

Висновок

У даній лабораторній роботі було розглянуто використання шаблону проектування «Prototype», з урахуванням особливостей обраної теми. В результаті шаблон допоміг досягти поставленої мети створення ворогів по заздалегідь вказаним налаштуванням.