**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА №** 2

**Дисциплина: «Шаблоны проектирования»**

**Тема:** *Управление игровыми активами*

**Выполнил: студент группы: 231-339**

Карапетян Нвер Каренович\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)



**Дата, подпись:** 11.04.25 \_\_\_\_\_\_\_\_\_\_\_\_

(Дата)(Подпись)

**Проверил:** \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_*

(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_\_\_*

(Дата)(Подпись)

**Москва**

**2025**

****Цель****:

Разработать систему управления активами (например, управление различными скинами или оружием) с использованием шаблона Приспособленец (Flyweight). Его еще называют «Легковесом».

****Описание****:

В играх часто используется множество активов, таких как скины, оружие, и т.д. Шаблон Приспособленец позволяет эффективно управлять и рендерить множество игровых активов, минимизируя потребление памяти.

План работы

Теоретическая часть

Шаблон «Приспособленец» (англ. Flyweight) — это структурный шаблон проектирования, цель которого — минимизировать использование памяти за счет повторного использования уже существующих объектов.

Он особенно полезен, когда приложение оперирует большим количеством однотипных объектов, содержащих одинаковые данные (например, текстуры, анимации, спрайты, модели и т. д.).

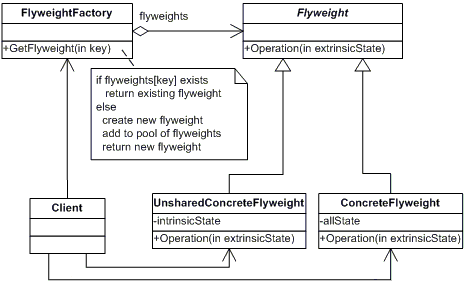


Рисунок 1. Схема шаблона.

Практическая часть

Была реализована система смены скинов для 2D-персонажа, основанная на подмене AnimatorController с помощью кэшированных экземпляров AnimatorOverrideController. Для соблюдения принципов Flyweight были выделены следующие компоненты:

1. **Flyweight (Легковес) — SkinFlyweight.cs**

Этот класс инкапсулирует общее состояние объекта — в нашем случае, AnimatorOverrideController. Он не содержит уникальных данных о конкретном объекте (например, о персонаже), а только разделяемый ресурс, используемый повторно.

public class SkinFlyweight

{

    private AnimatorOverrideController skinController;

    public SkinFlyweight(AnimatorOverrideController controller) => skinController = controller;

    public AnimatorOverrideController GetController() => skinController;

}

1. **Flyweight Factory (Фабрика легковесов) — SkinFactory.cs**

Отвечает за создание и кэширование SkinFlyweight-объектов. Если скин уже был загружен, он возвращается из словаря. В противном случае — загружается из ресурсов и сохраняется для повторного использования.

public class SkinFactory : MonoBehaviour

{

    private static Dictionary<string, SkinFlyweight> skins = new Dictionary<string, SkinFlyweight>();

    public static SkinFlyweight GetSkin(string skinPath)

    {

        if (!skins.ContainsKey(skinPath))

        {

            AnimatorOverrideController controller = Resources.Load<AnimatorOverrideController>(skinPath);

            if (controller is null)

                return null;

            skins[skinPath] = new SkinFlyweight(controller);

        }

        return skins[skinPath];

    }

}

1. **Client (Клиент шаблона) —** **PlayerSkinManager.cs**

Компонент, управляющий скином персонажа. Сохраняет оригинальный контроллер при запуске и предоставляет методы смены скина и возврата к изначальному виду. Использует фабрику SkinFactory для получения нужного SkinFlyweight.

public class PlayerSkinManager : MonoBehaviour

{

    private Animator animator;

    private RuntimeAnimatorController originalController;

    private void Awake()

    {

        animator = GetComponent<Animator>();

        if (animator is null)

            return;

        originalController = animator.runtimeAnimatorController;

    }

    public void ChangeSkin(string skinPath)

    {

        SkinFlyweight skin = SkinFactory.GetSkin(skinPath);

        if (skin is not null)

            animator.runtimeAnimatorController = skin.GetController();

    }

    public void ResetToOriginalSkin()

    {

        if (originalController is not null)

            animator.runtimeAnimatorController = originalController;

    }

    private void Update()

    {

        if (Input.GetKeyDown(KeyCode.Alpha1))

            ResetToOriginalSkin();

        if (Input.GetKeyDown(KeyCode.Alpha2))

            ChangeSkin("Skins/Skin2\_Controller");

        if (Input.GetKeyDown(KeyCode.Alpha3))

            ChangeSkin("Skins/Skin3\_Controller");

    }

}

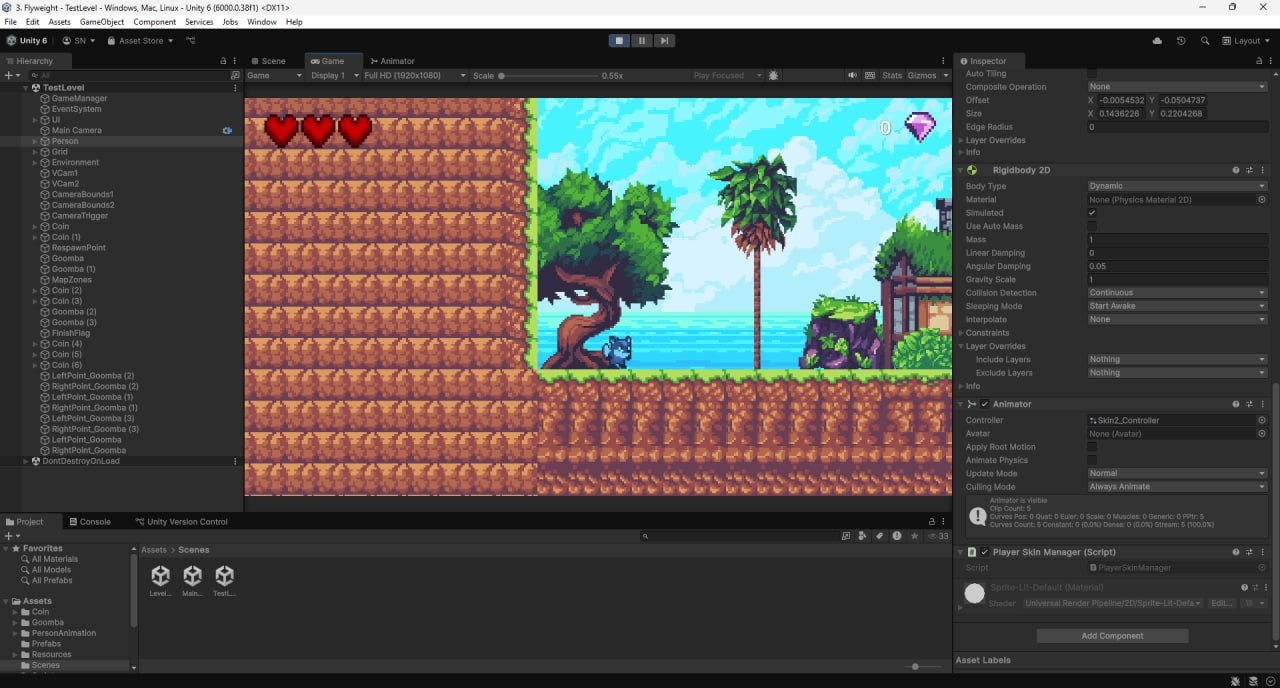


Рисунок 2. Скин персонажа перекрашен в синий цвет при нажатии на клавишу «2».

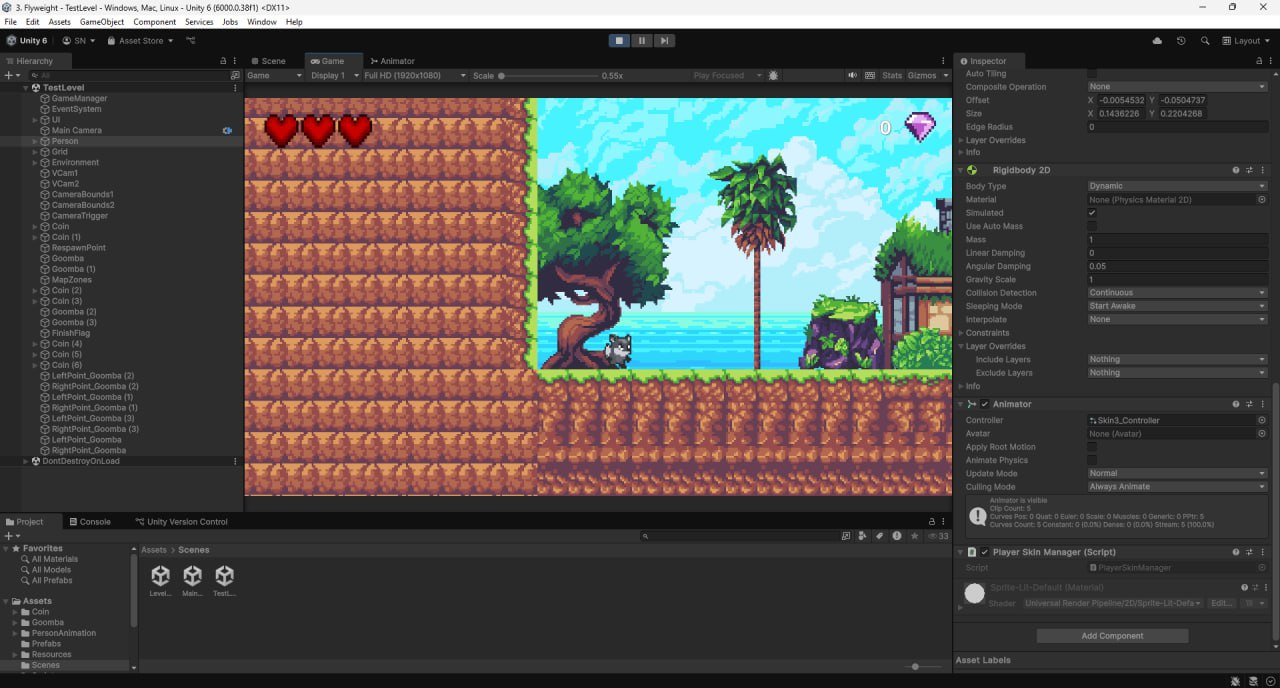


Рисунок 3. Скин персонажа перекрашен в черно-белый при нажатии на клавишу «3».