Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный университет»

Школа развития цифровых компетенций «Digital Up» (цифровая кафедра)

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**на производственную практику**

**по ДПП ПП «**Основы Gamedev и VR-разработки**»**

**Начало формы**

|  |  |  |  |
| --- | --- | --- | --- |
| п/п  № | Задание | Исполнитель | Рабочий график (план) выполнения |
| 1 | Разработать систему навыков | Шацких А. Е. Муравьёв Н. | 17.05.24 – 31.05.24 |

Руководитель проекта   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / «\_\_\_ » \_\_\_\_\_\_\_\_\_ 2025 г.

(подпись)

Соруководитель проекта *(при наличии)*   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / «\_\_\_ » \_\_\_\_\_\_\_\_\_ 2025 г.

(подпись)

СОГЛАСОВАНО:

Руководитель ДПП ПП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2024 г.

(подпись)

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный университет»

Школа развития цифровых компетенций «Digital Up» (цифровая кафедра)

Отчет о выполнении производственной практики по ДПП ПП

«Основы Gamedev и VR-разработки»

**«Разработка системы навыков»**

Исполнители:

Шацких А. Е.

Муравьёв Н. М.

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 г.

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 г.

г. Барнаул, 2025

1. Цель

Цель проекта – создание прототипа системы навыков для игры.

1. Задачи проекта и исполнители
2. Создание классов для обслуживания системы навыков. Исполнитель: Шацких А. Е.
3. Создание классов, взаимодействующих с системой навыков. Исполнитель: Муравьёв А. Е.
4. Общие сведения о проделанной работе

Для выполнения поставленной задачи было необходимо создать базовый класс **Skill**, который бы описывал логику навыка персонажа. В ходе выделения полей, свойств и методов класса Skill было принято решение о создании перечисления возможных типов навыков **SkillType**, а также создания дополнительного класса **SkillEffect** с перечислением **EffectType** для возможных типов эффектов навыка и перечисления **StatType** для указания характеристики, на которую действует навык.

Для работы с множеством навыков, которые могут появляться в игре, было принято решение о создании класса **SkillManager**, который хранит в себе информацию о всех навыках в игре, текущих доступных навыков игроку и выбранных игроком навыков, а также предоставляет методы для работы с навыками.

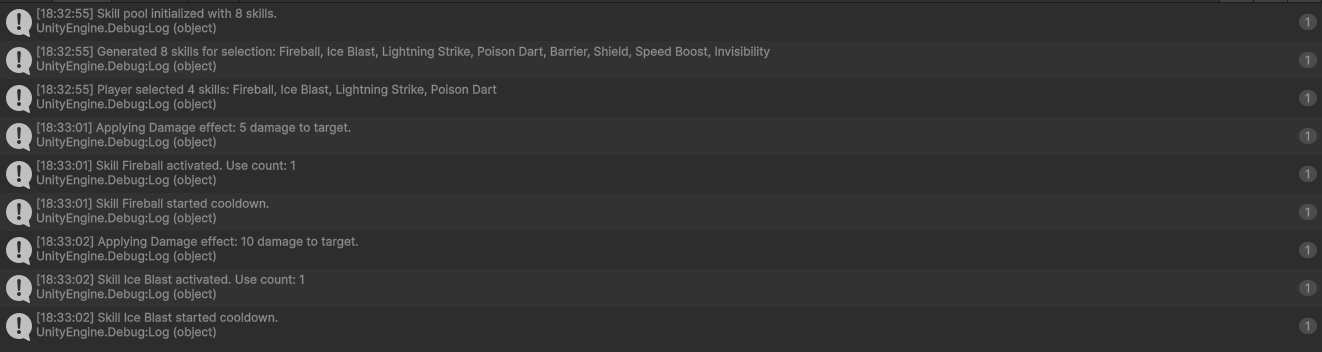
Был разработан класс **Pickup**, который реализует логику добавления нового навыка для игрока, посредством поднятия какого-либо предмета.

Был реализован класс **Shop**, который реализует логику магазина навыков, в котором игрок может покупать себе новые навыки посредством покупки каких-либо предметов.

1. Результаты проекта

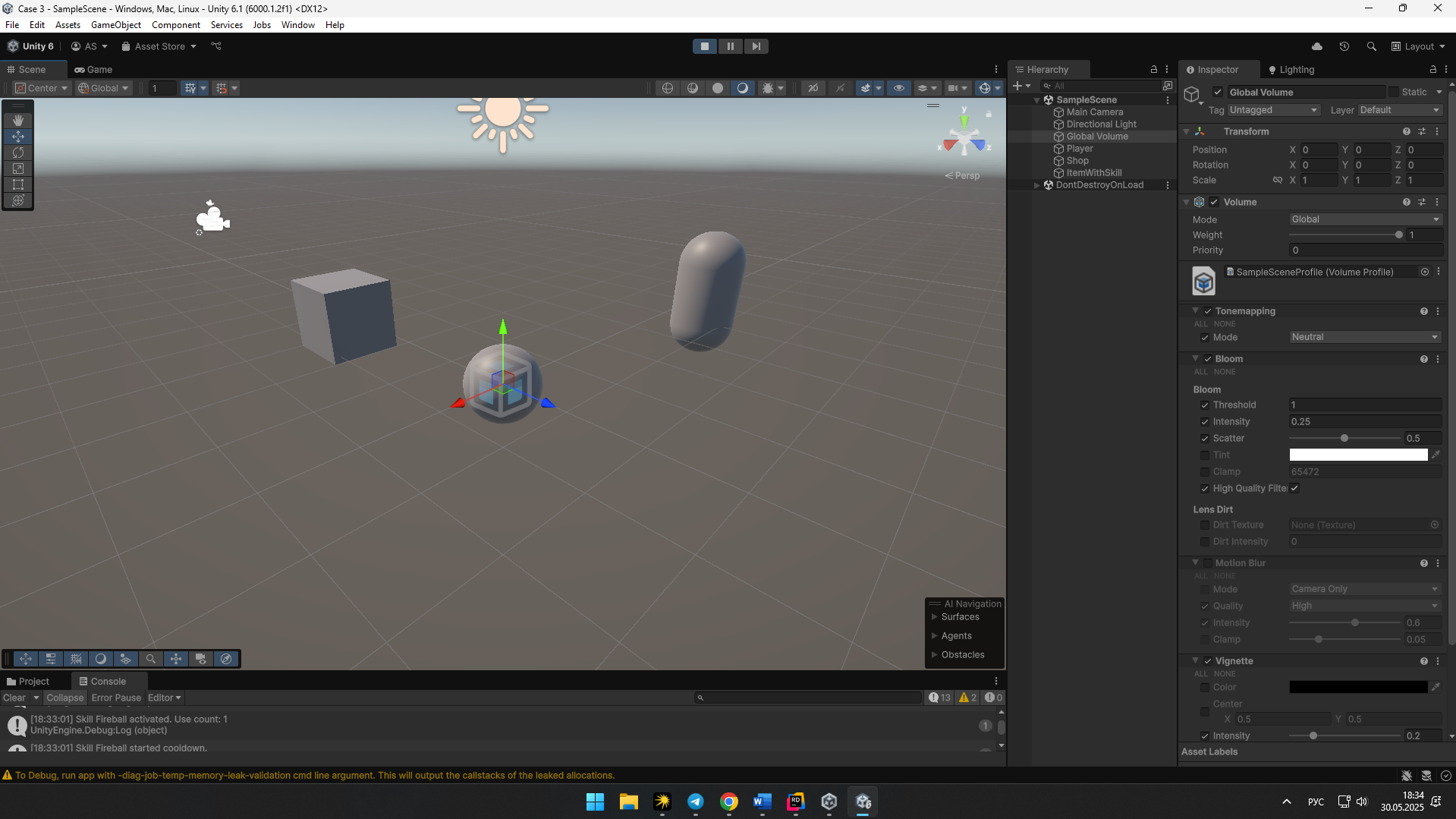
В ходе выполнения проекта были проделаны тесты имитаций действий игрока для демонстрации работы системы.

При входе в игру игроку дают 4 случайных навыка, которые при нажатии клавиш 1-4 он может использовать, что демонстрирует рисунок 1.

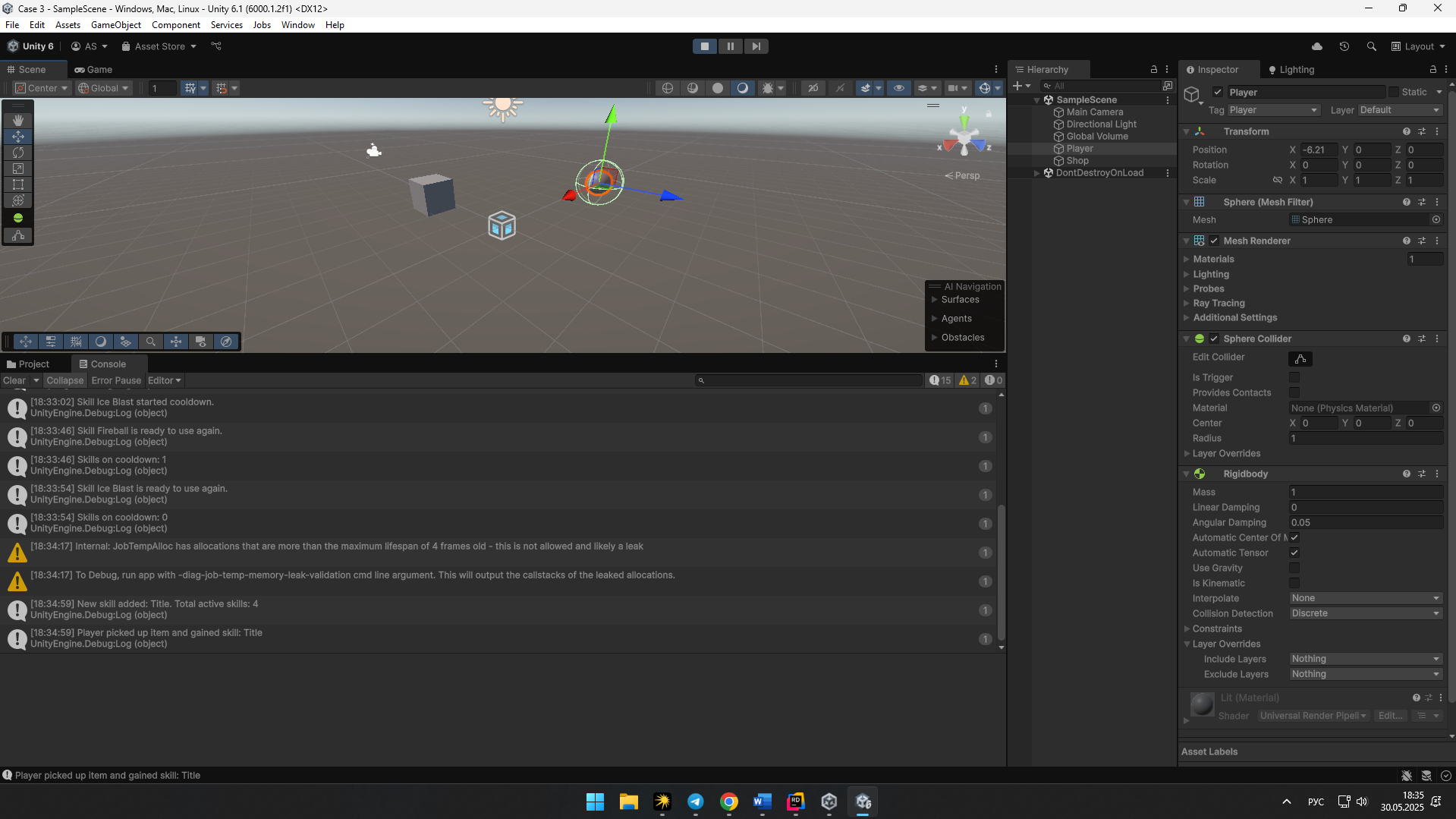
****

**Рисунок 1.**

Когда игрок задевает триггер предмета, в его общий пул навыков добавляется навык предмета, после чего предмет уничтожается, что демонстрирует рисунок 2 и 3.

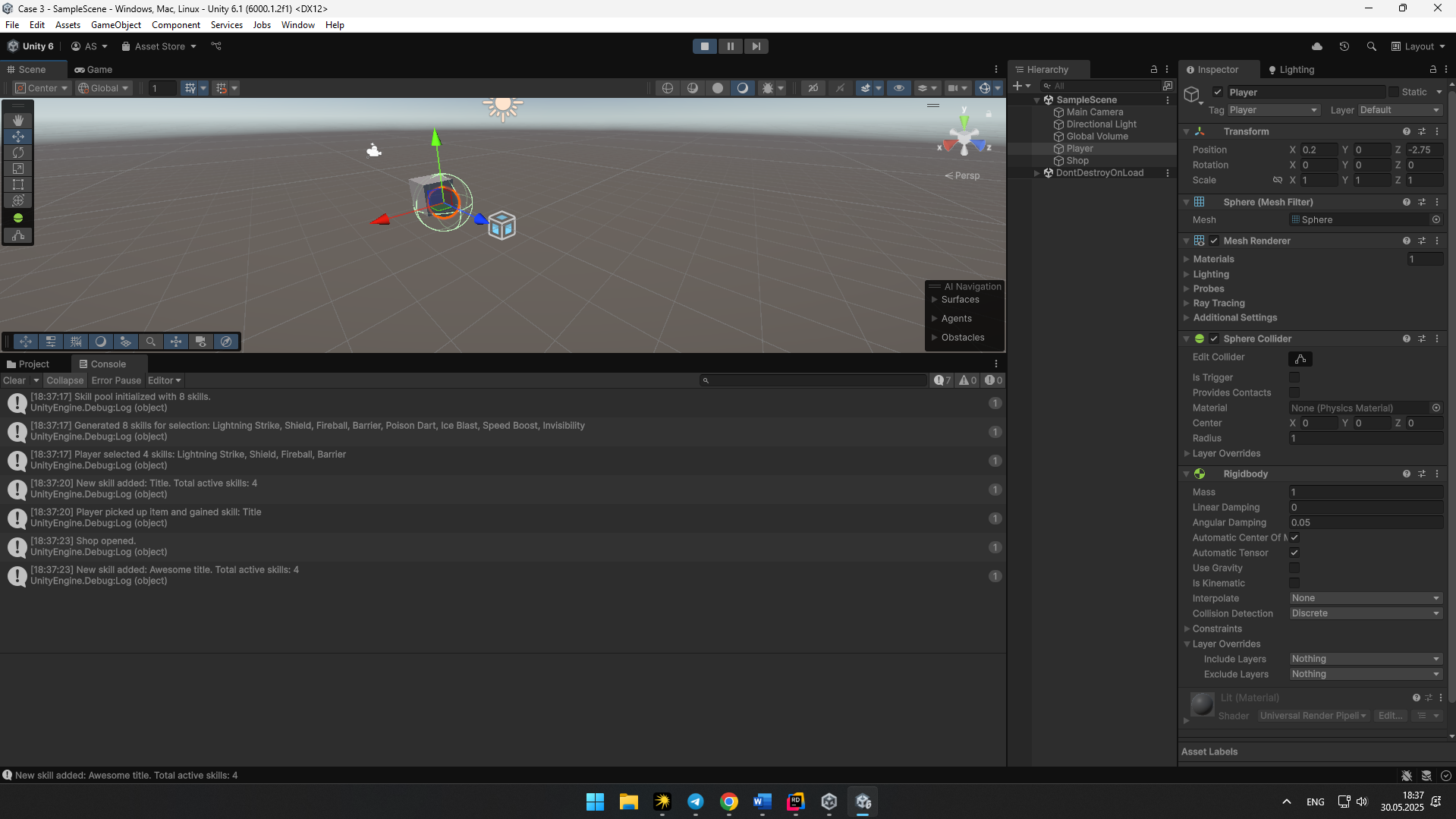


**Рисунок 2.**

****

**Рисунок 3.**

Когда игрок заходит в триггер магазина, имитируется покупка навыка и добавление его в пул доступных навыков, что демонстрирует рисунок 4.



**Рисунок 4.**

**Приложение 1.**

Программный код

namespace Enums  
{  
 public enum SkillType  
 {  
 **Attack**,  
 **Defense**,  
 Heal,  
 **Utility** }  
}

namespace Enums  
{  
 public enum EffectType  
 {  
 **Damage**,  
 **Heal**,  
 **Buff**,  
 **Debuff** }  
}

namespace Enums  
{  
 public enum StatType  
 {  
 Hp,  
 Mana,  
 **Armor**,  
 **Speed**,  
 **Visibility** }  
}

using System;  
using Enums;  
using UnityEngine;  
  
[Serializable]  
public class Skill  
{  
 [SerializeField] private string title;  
 [SerializeField] private SkillType type;  
 [SerializeField] private string description;  
 [SerializeField] private SkillEffect baseEffect;  
 [SerializeField] private SkillEffect enhancedEffect;  
 [SerializeField] private float baseCooldown;  
 [SerializeField] private float currentCooldown;  
 [SerializeField] private bool isOnCooldown;  
 [SerializeField] private int useCount;  
 [SerializeField] private bool isElit;  
  
 public string Title  
 {  
 get => title;  
 private set => title = value;  
 }  
  
 public SkillType Type  
 {  
 get => type;  
 private set => type = value;  
 }  
  
 public string Description  
 {  
 get => description;  
 private set => description = value;  
 }  
  
 public SkillEffect BaseEffect  
 {  
 get => baseEffect;  
 private set => baseEffect = value;  
 }  
  
 public SkillEffect EnhancedEffect  
 {  
 get => enhancedEffect;  
 private set => enhancedEffect = value;  
 }  
  
 public float BaseCooldown  
 {  
 get => baseCooldown;  
 private set => baseCooldown = value;  
 }  
  
 public bool IsOnCooldown  
 {  
 get => isOnCooldown;  
 private set => isOnCooldown = value;  
 }  
  
 public int UseCount  
 {  
 get => useCount;  
 private set => useCount = value;  
 }  
  
 public bool IsElit  
 {  
 get => isElit;  
 private set => isElit = value;  
 }  
  
 public Skill(string title, SkillType type, string description, SkillEffect baseEffect, SkillEffect enhancedEffect,  
 float baseCooldown, bool isElit)  
 {  
 Title = title;  
 Type = type;  
 Description = description;  
 BaseEffect = baseEffect;  
 EnhancedEffect = enhancedEffect;  
 BaseCooldown = baseCooldown;  
 IsOnCooldown = false;  
 IsElit = isElit;  
 }  
   
 public void Activate()  
 {  
 if (!isOnCooldown)  
 {  
 useCount++;  
 isOnCooldown = true;  
 var effectToApply = isElit ? enhancedEffect : baseEffect;  
 effectToApply.Apply();  
 Debug.Log($"Skill {title} activated. Use count: {useCount}");  
 }  
 else  
 {  
 Debug.Log($"Skill {title} is on cooldown. Time remaining: {currentCooldown}");  
 }  
 }  
  
 public void MakeElite()  
 {  
 if (!isElit)  
 {  
 isElit = true;  
 }  
 }  
   
 public void StartCooldown(float multiplier)  
 {  
 currentCooldown = baseCooldown \* multiplier;  
 isOnCooldown = true;  
 Debug.Log($"Skill {title} started cooldown.");  
 }  
  
 public void UpdateCooldown(float deltaTime)  
 {  
 if (!isOnCooldown) return;  
 currentCooldown -= deltaTime;  
 if (!(currentCooldown <= 0f)) return;  
 isOnCooldown = false;  
 currentCooldown = 0f;  
 Debug.Log($"Skill {title} is ready to use again.");  
 }  
}

using System;  
using Enums;  
using UnityEngine;  
  
[Serializable]  
public class SkillEffect  
{  
 private EffectType \_effectType;  
 private float \_value;  
 private float \_duration;  
 private StatType? \_targetStat;  
  
 public SkillEffect(EffectType type, float value, float duration)  
 {  
 \_effectType = type;  
 \_value = value;  
 \_duration = duration;  
 }  
  
 public SkillEffect(EffectType type, float value, float duration, StatType targetStat)  
 {  
 \_effectType = type;  
 \_value = value;  
 \_duration = duration;  
 \_targetStat = targetStat;  
 }  
  
 public void Apply()  
 {  
 switch (\_effectType)  
 {  
 case EffectType.**Damage**:  
 Debug.Log($"Applying {\_effectType} effect: {\_value} damage to target.");  
 break;  
 case EffectType.**Heal**:  
 Debug.Log($"Applying {\_effectType} effect: Restoring {\_value} HP.");  
 break;  
 case EffectType.**Buff**:  
 Debug.Log($"Applying {\_effectType} effect: Modifying {\_targetStat} by {\_value} for {\_duration} seconds.");  
 break;  
 case EffectType.**Debuff**:  
 Debug.Log($"Unknown effect type: {\_effectType}");  
 break;  
 default:  
 throw new ArgumentOutOfRangeException();  
 }  
 }  
}

using System.Collections.Generic;  
using System.Linq;  
using Enums;  
using UnityEngine;  
using UnityEngine.InputSystem;  
using Random = UnityEngine.Random;  
  
public class SkillsManager : MonoBehaviour  
{  
 public static SkillsManager Singleton { get; private set; }  
  
 private List<Skill> \_allSkills;  
 private List<Skill> \_availableSkills;  
 private List<Skill> \_activeSkills;  
 private int \_skillsOnCooldown;  
  
 private void Awake()  
 {  
 if (!Singleton)  
 {  
 Singleton = this;  
 DontDestroyOnLoad(this);  
 }  
  
 else  
 {  
 Destroy(gameObject);  
 }  
 }  
  
 private void Start()  
 {  
 InitSkillPool();  
 GenerateSkillSelection();  
 SelectInitialSkills();  
 }  
  
 private void Update()  
 {  
 UpdateSkillsCoolDown();  
  
 if (Keyboard.current.digit1Key.wasPressedThisFrame)  
 {  
 ActivateSkill(0);  
 }  
  
 if (Keyboard.current.digit2Key.wasPressedThisFrame)  
 {  
 ActivateSkill(1);  
 }  
  
 if (Keyboard.current.digit3Key.wasPressedThisFrame)  
 {  
 ActivateSkill(2);  
 }  
  
 if (Keyboard.current.digit4Key.wasPressedThisFrame)  
 {  
 ActivateSkill(3);  
 }  
 }  
  
 private void InitSkillPool()  
 {  
 \_allSkills = new List<Skill>  
 {  
 new(  
 "Fireball", SkillType.**Attack**,  
 "Description",  
 new SkillEffect(EffectType.**Damage**, 5f, 0),  
 new SkillEffect(EffectType.**Damage**, 10f, 0),  
 5f,  
 false),  
 new(  
 "Ice Blast", SkillType.**Attack**,  
 "Description.",  
 new SkillEffect(EffectType.**Damage**, 10f, 0),  
 new SkillEffect(EffectType.**Damage**, 20f, 0),  
 10f,  
 false),  
 new(  
 "Shield", SkillType.**Defense**,  
 "Description",  
 new SkillEffect(EffectType.**Buff**, 30f, 15f, StatType.**Armor**),  
 new SkillEffect(EffectType.**Buff**, 60f, 30f, StatType.**Armor**),  
 25f,  
 false),  
 new(  
 "Lightning Strike", SkillType.**Attack**,  
 "Description",  
 new SkillEffect(EffectType.**Damage**, 5f, 0),  
 new SkillEffect(EffectType.**Damage**, 10f, 0),  
 12f,  
 false),  
 new(  
 "Invisibility", SkillType.**Utility**,  
 "Description",  
 new SkillEffect(EffectType.**Buff**, 1f, 25f, StatType.**Visibility**),  
 new SkillEffect(EffectType.**Buff**, 1f, 50f, StatType.**Visibility**),  
 60f,  
 false),  
 new(  
 "Barrier", SkillType.**Defense**,  
 "Description",  
 new SkillEffect(EffectType.**Buff**, 8f, 15f, StatType.**Armor**),  
 new SkillEffect(EffectType.**Buff**, 16f, 30f, StatType.**Armor**),  
 5f,  
 false),  
 new(  
 "Poison Dart", SkillType.**Attack**,  
 "Description",  
 new SkillEffect(EffectType.**Damage**, 15f, 0),  
 new SkillEffect(EffectType.**Damage**, 30f, 0),  
 6f,  
 false),  
 new(  
 "Speed Boost", SkillType.**Utility**,  
 "Description",  
 new SkillEffect(EffectType.**Buff**, 5f, 15f, StatType.**Speed**),  
 new SkillEffect(EffectType.**Buff**, 10f, 30f, StatType.**Speed**),  
 15f,  
 false)  
 };  
 \_availableSkills = new List<Skill>();  
 \_activeSkills = new List<Skill>();  
 \_skillsOnCooldown = 0;  
 Debug.Log("Skill pool initialized with " + \_allSkills.Count + " skills.");  
 }  
  
 private void GenerateSkillSelection()  
 {  
 \_availableSkills.Clear();  
 var tempPool = new List<Skill>(\_allSkills);  
 for (var i = 0; i < 8; i++)  
 {  
 if (tempPool.Count == 0) break;  
 var randomIndex = Random.Range(0, tempPool.Count);  
 \_availableSkills.Add(tempPool[randomIndex]);  
 tempPool.RemoveAt(randomIndex);  
 }  
  
 Debug.Log("Generated " + \_availableSkills.Count + " skills for selection: " +  
 string.Join(", ", \_availableSkills.Select(s => s.Title)));  
 }  
  
 *// Выбор первых 4 навыков (эмуляция выбора игрока)* private void SelectInitialSkills()  
 {  
 for (var i = 0; i < 4 && i < \_availableSkills.Count; i++)  
 {  
 \_activeSkills.Add(\_availableSkills[i]);  
 }  
  
 Debug.Log("Player selected " + \_activeSkills.Count + " skills: " +  
 string.Join(", ", \_activeSkills.Select(s => s.Title)));  
 }  
  
 private void ActivateSkill(int index)  
 {  
 if (index < 0 || index >= \_activeSkills.Count)  
 {  
 Debug.Log("Invalid skill index: " + index);  
 return;  
 }  
  
 var skill = \_activeSkills[index];  
 if (!skill.IsOnCooldown)  
 {  
 skill.Activate();  
 \_skillsOnCooldown++;  
 var cooldownMultiplier = 1f + 0.1f \* \_skillsOnCooldown;  
 skill.StartCooldown(cooldownMultiplier);  
 }  
 else  
 {  
 Debug.Log($"Cannot activate {skill.Title}. It is on cooldown.");  
 }  
 }  
  
 public void AddSkillToPool(Skill newSkill)  
 {  
 \_allSkills.Add(newSkill);  
 Debug.Log($"New skill added: {newSkill.Title}. Total active skills: {\_activeSkills.Count}");  
 }  
  
 public void AddSkillToAvailable(Skill newSkill)  
 {  
 \_availableSkills.Add(newSkill);  
 Debug.Log($"New skill added: {newSkill.Title}. Total active skills: {\_activeSkills.Count}");  
 }  
  
 public void AddSkillToActive(Skill newSkill)  
 {  
 \_activeSkills.Add(newSkill);  
 Debug.Log($"New skill added: {newSkill.Title}. Total active skills: {\_activeSkills.Count}");  
 }  
  
 private void UpdateSkillsCoolDown()  
 {  
 var prevSkillsOnCooldown = \_skillsOnCooldown;  
 \_skillsOnCooldown = 0;  
  
 foreach (var skill in \_activeSkills)  
 {  
 skill.UpdateCooldown(Time.deltaTime);  
 if (skill.IsOnCooldown)  
 \_skillsOnCooldown++;  
 }  
  
 if (prevSkillsOnCooldown != \_skillsOnCooldown)  
 {  
 Debug.Log($"Skills on cooldown: {\_skillsOnCooldown}");  
 }  
 }  
  
 public List<Skill> GetSkillsByType(SkillType type)  
 {  
 var skills = \_allSkills.Where(s => s.Type == type).ToList();  
 if (skills.Count != 0)  
 return skills;  
  
 Debug.Log($"No skills found for category: {type}");  
 return null;  
 }  
}

using System;  
using Enums;  
using UnityEngine;  
  
public class Pickup : MonoBehaviour  
{   
 private SkillsManager \_skillsManager;  
  
 private void Start()  
 {  
 \_skillsManager = SkillsManager.Singleton;  
 }  
  
 *// Имитация подбора скилла с предмета* private void OnTriggerEnter(Collider other)  
 {  
 if (other.CompareTag("Player"))  
 {  
 var skill = new Skill(  
 "Title",   
 SkillType.**Attack**,   
 "Description",   
 new SkillEffect(EffectType.**Damage**, 5f, 0f),  
 new SkillEffect(EffectType.**Damage**, 10f, 0f),  
 5f,  
 false);  
   
 \_skillsManager.AddSkillToAvailable(skill);  
 Debug.Log($"Player picked up item and gained skill: {skill.Title}");  
 Destroy(gameObject);  
 }  
 }  
}

using Enums;  
using UnityEngine;  
  
public class Shop : MonoBehaviour  
{  
 private SkillsManager \_skillsManager;  
 private bool \_shopIsOpen;  
  
 private void Start()  
 {  
 \_skillsManager = SkillsManager.Singleton;  
 }  
  
 private void OnTriggerEnter(Collider other)  
 {  
 if (!other.CompareTag("Player")) return;  
   
 \_shopIsOpen = true;  
 Debug.Log("Shop opened.");  
 BuySkill();  
 }  
  
 private void OnTriggerExit(Collider other)  
 {  
 \_shopIsOpen = false;  
 Debug.Log("Shop closed.");  
 }  
   
 private void BuySkill()  
 {  
 var newSkill = new Skill(  
 "Awesome title",   
 SkillType.**Attack**,   
 "Awesome description",   
 new SkillEffect(EffectType.**Damage**, 5f, 0f),  
 new SkillEffect(EffectType.**Damage**, 10f, 0f),  
 5f,  
 false);  
 \_skillsManager.AddSkillToAvailable(newSkill);  
 }  
}