ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

Факультет информационных технологий

Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

Разработка игрового приложения «Этажи»

Выполнили: студенты группы 201-726

Денискина Елизавета Сергеевна

Ковтуненко Наталья Сергеевна

Карпушкин Сергей Евгеньевич

Калашян Венера Араевна

Курносова Арсения Валерьевна

Дата, подпись ___10.02.2022_____ _____

(Дата) (Подпись)

Проверил: _____ _____

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

Дата, подпись _____ _____

(Дата) (Подпись)

Москва

2023

Оглавление

Сюжет

Изначально было предложено два варианта сюжета, и путем голосования был выбран следующий:

Главный герой по имени Отто находит себя дома, и вроде бы все как обычно, но что-то точно не так: все страницы в книгах пустые, в квартире нет входной двери, а главное – перед ним стоит цветочный горшок с фиолетовым щупальцем. Отто ничего не остается, кроме как, взаимодействуя с активными предметами, раз за разом пытаться сбежать из странной квартиры.

Тема «Все симуляция» передана через небольшую локацию, которая кажется герою привычной, однако какие-то детали, которые постепенно начинают бросаться в глаза, и порой странные пути решения загадок показывают, что этот мир совсем не такой, каким Отто его видит.

1 уровень.

Главный герой пытается вспомнить, как именно он здесь оказался. Заметив отсутствие входной двери, он напрягается и решает сбежать. Чтобы это сделать, игроку необходимо взять лейку, набрать воду и полить щупальце, тогда оно разобьет окно, и Отто сможет спуститься по нему ниже.

2 уровень.

Отто снова оказывается в своей квартире, но на этот раз что-то поменялось. Чтобы выбраться в этот раз надо взять со стола расческу-гребешок и положить ее рядом с кукольным домиком так, чтобы она создавала подобие лестницы, подходящей к игрушечному окну. У реального окна также появляется лестница вниз, и главный герой переходит на новый уровень.

3 уровень.

Главный герой обращает внимания на свои простыни и наволочки и, связав их, спускается из окна. Он рад, что ему больше не нужно выполнять «квесты», и он надеется, что третий раз – счастливый.

4 уровень.

Отто холодеет, поняв, что снова оказался здесь. Решив, что попробует в последний раз, он находит отвертку, откручивает полку, берет ее крюкообразные крепления и, используя их, как кошки, спускается вниз.

5 уровень.

Главный герой потерял какую-либо надежду. Он находит ленту для мух и, хоть и понимает, что она его не выдержит, все равно ее использует.

Концовка:

Отто оказывается на улице. Сначала он не верит своему счастью, но затем, осмотревшись, понимает, что находится в окружении абсолютно идентичных домов – таких же, как из которого только что выбрался он. Камера отдаляется, и мы видим, что все это происходит на экране чьего-то ноутбука.

Дизайн

Стиль изначально был выбран карандашный, в нем же и нарисован главный герой. Для его анимации в сумме было отрисовано 16 кадров.

Рисунок 1- Дизайн главного героя

Между обычными объектами и объектами, необходимыми для прохождения квестов, есть и должна быть большая разница, поэтому за их рисование взялось два разных художника.

Обыкновенные объекты были черно-белыми, нарисованы довольно крупной набросочной кистью:

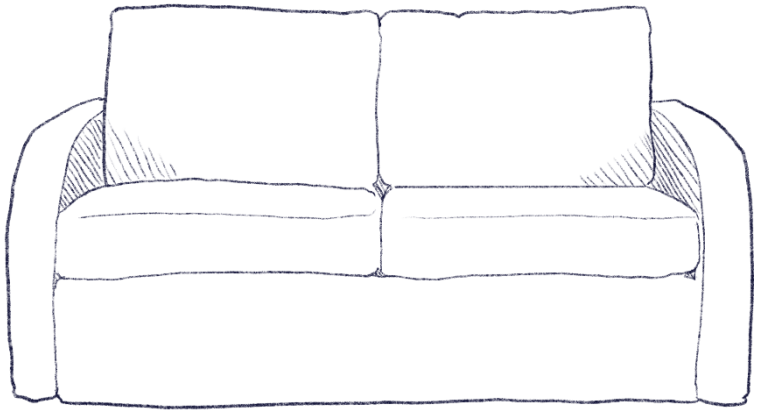

Рисунок 2 - Обыкновенный объект (1)

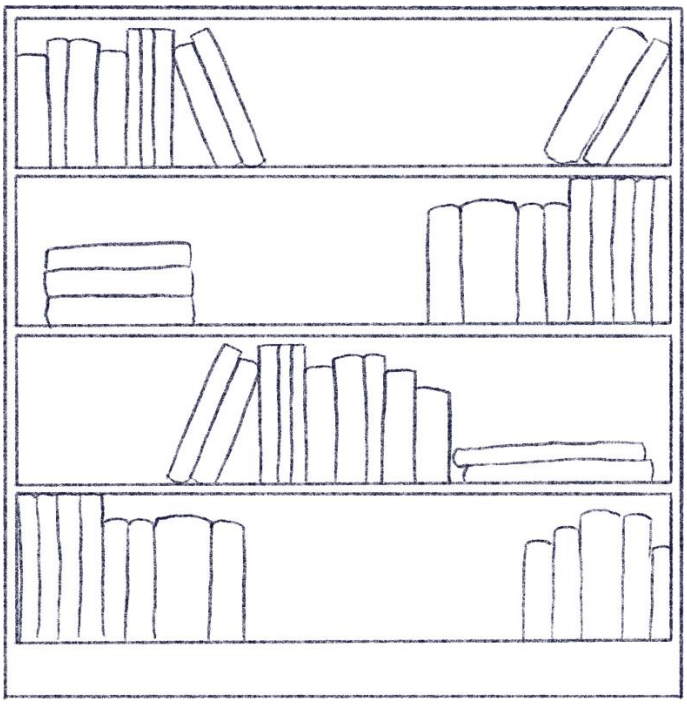

Рисунок 3 - Обыкновенный объект (2)

Во-первых, эти объекты – фон, так что не должны были слишком сильно притягивать взгляд. Во-вторых, их дизайн передает идею симуляции: все набросано на скорую руку, но оно чисто-белое, без лишних мелких деталей.

Объекты, необходимые для квестов, были яркими, и нарисованы немного в другом стиле, в том числе и использованием мелких деталей:

Рисунок 4 - Квестовый объект (1)

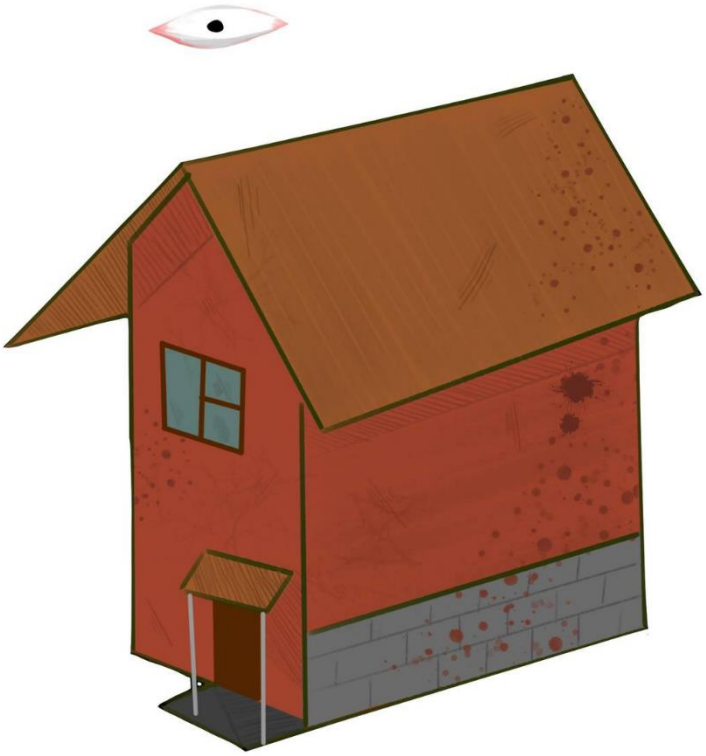

Рисунок 5 - Квестовый объект (2)

Именно на этих объектах пользователь должен был заострять внимание и понимать, с чем ему взаимодействовать, чтобы двинуться дальше.

Так же на пол были наложены текстуры, а в некоторых местах были добавлены источники света. Итог выглядел таким образом:

Рисунок 6 - Итоговая версия

Исследование этапов разработки игры

Этап 1 – Выбор среды разработки

В качестве среды разработки игры был выбран игровой движок «Unity». Данная среда удовлетворяет следующим требованиям:

- Менее требовательная (сравнительно с другими игровыми движками) по техническим характеристикам устройства, на котором будет происходить разработка;
- Позволяет реализовать запланированные в дизайнерском документе механики игры;
- Позволяет реализовать запланированные в дизайнерском документе дизайнерские решения игры.

Unity позволяет разработать продукт в соответствии с дизайнерским документом, без излишних затрат ресурсов носителя среды разработки и ресурсов носителя, на который в дальнейшем может быть установлен продукт разработки.

Этап 2 – Ознакомление с реализацией игровых механик

Игра не задействует уникальных игровых механик. Для реализации игровых механик были изучены реализации похожих механик в других игровых проектах.

Были рассмотрены:

- Механики управления главным героем в двухмерном игровом пространстве;
- Механики взаимодействия с интерактивными предметами в игровом пространстве;
- Механики «квестов» в игровых проектах;
- Механики диалогов с NPS, на основании которых была разработана механика описаний интерактивных объектов главным героем;
- Механики перехода между игровыми сценами;
- Механики разработки меню и других пользовательских интерфейсов.

На основании изученных механик, были разработаны механики для игрового проекта.

Этап 3 – Ознакомление с разработкой визуального наполнения

Для проекта были разработаны уникальные элементы игрового окружения. Элементы игрового окружения необходимо разрабатывать с учётом общих технических параметров игрового проекта. Для разработки элементов игрового окружения были изучены:

- Работа с графическими редакторами, удовлетворяющими требованиям разработки элементов игрового окружения данного игрового проекта;

- Технические параметры, которым должны удовлетворять разрабатываемые элементы: расширение и разрешение изображений;
- Перечень элементов, необходимых к разработке для данного игрового проекта;
- Принципы покадровой анимации;
- Адаптация принципов создания покадровой анимации под общие технические параметры игрового проекта.

Этап 4 – Сборка проекта

Для сборки игрового проекта, были изучены следующие этапы разработки:

- Язык программирования C#, встроенный в среду разработки «Unity»;
- Встроенные функции языка программирования, для реализации запланированных игровых механик;
- Встроенные инструменты среды разработки для реализации запланированных игровых механик;
- Встроенные инструменты, для реализации игрового окружения и корректного взаимодействия игрового персонажа с игровым оркужением;
- Работа с медиаконтентом в среде разработки;
- Параметры экспорта итогового продукта.

Код

Скрипт на C# для контроллера игрового персонажа. Он содержит такие компоненты, как «Rigidbody2D», «Animator» и «TextMeshProUGUI», которые обрабатывает движение и анимацию. Он содержит такие методы, как «StartCutscene» и «Pick Up Sound», которые управляют элементами игрового процесса, такими как запуск роликов, подбор объектов и воспроизведение звуков. Он также содержит такие переменные, как «canMove» и «movement»,

с помощью которых игрок перемещается и хранятся данные о перемещении игрока.

Листинг 1 - PlayerController

```
using System;
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.Serialization;

public class PlayerController : MonoBehaviour
{
    public static PlayerController Instance { get; private set; }



    private bool canMove = true;
    private float tipFadeTime = 0.5f;
    public float moveSpeed = 5f;
    public AudioSource audioSource;
    public AudioSource pickupAudioSource;
    public AudioClip pickupSound;
    public AudioClip walkSound;
    public Rigidbody2D rb;
    public Animator animator;
    private Vector2 movement;

    public List<GameObject> allCutscenePrefabs;
    [FormerlySerializedAs("cutScenePrefab")] public GameObject cutscenePrefab;
    public GameObject dialoguePanel;
    public TextMeshProUGUI dialogueTextUI;
    public TextMeshProUGUI characterNameUI;
    public TextMeshProUGUI pressEUI;
    public TextMeshProUGUI tipUI;

    private void Awake()
    {
        if (Instance != null && Instance != this)
        {
```

```
            Destroy(gameObject);
            return;
        }

        Instance = this;
        DontDestroyOnLoad(gameObject);

        rb = GetComponent<Rigidbody2D>();
    }

    private void OnEnable()
    {
        SceneManager.sceneLoaded += OnSceneLoaded;
    }

    private void Start()
    {
        dialoguePanel = CanvasSingleton.Instance.gameObject.transform.GetChild(0).gameObject;
        characterNameUI = dialoguePanel.transform.GetChild(0).GetComponent<TextMeshProUGUI>();
        dialogueTextUI = characterNameUI.transform.GetChild(0).GetComponent<TextMeshProUGUI>();
        pressEUI =
CanvasSingleton.Instance.transform.GetChild(1).GetComponent<TextMeshProUGUI>();
        tipUI = CanvasSingleton.Instance.transform.GetChild(2).GetComponent<TextMeshProUGUI>();

        if (cutscenePrefab)
        {
            StartCutscene(cutscenePrefab);
        }
    }

    private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
    {
        transform.position = new Vector3(-50, 12, 0);
        switch(scene.name)
        {
            case "Level1":
                StartCutscene(allCutscenePrefabs[0]);
                break;
```

```
            case "Level2" or "Level2_NEW":
                StartCutscene(allCutscenePrefabs[1]);
                break;
            case "Level3":
                StartCutscene(allCutscenePrefabs[2]);
                break;
            case "Level4":
                StartCutscene(allCutscenePrefabs[3]);
                break;
            case "Level5":
                StartCutscene(allCutscenePrefabs[4]);
                break;
            case "Ending":
                SetDialogueUIActive(false);
                StopAllCoroutines();
                Destroy(CanvasSingleton.Instance.gameObject);
                Destroy(CameraSingleton.Instance.gameObject);
                Destroy(gameObject);
                break;
        }
    }

    private void Update()
    {
        if (canMove)
        {
            movement.x = Input.GetAxisRaw("Horizontal");
            movement.y = Input.GetAxisRaw("Vertical");

            if (movement.sqrMagnitude > 0.01f)
            {
                if (!audioSource.isPlaying)
                {
                    audioSource.clip = walkSound;
                    audioSource.loop = true;
                    audioSource.Play();
                }
            }
```

```
        else
        {
            audioSource.Stop();
        }

        animator.SetFloat("Horizontal", movement.x);
        animator.SetFloat("Vertical", movement.y);
        animator.SetFloat("Speed", movement.sqrMagnitude);
    }
}

private void FixedUpdate()
{
    rb.MovePosition(rb.position + movement * moveSpeed * Time.fixedDeltaTime);
}

public void PickUpSound()
{
    pickupAudioSource.Play();
}

public void SetDialogueUIActive(bool setActive)
{
    dialoguePanel.SetActive(setActive);
    canMove = !setActive;
    if (setActive)
        rb.velocity = Vector2.zero;
}

public void StartCutscene(GameObject _cutscenePrefab)
{
    Instantiate(_cutscenePrefab);
    cutscenePrefab = null;
}

public void ShowTip(string text, float tipDisplayTime)
{
    StopAllCoroutines();
```

```
        StartCoroutine(ShowTipCoroutine(text, tipDisplayTime));
    }

    private IEnumerator ShowTipCoroutine(string text, float tipDisplayTime)
    {
        tipUI.text = text;
        tipUI.color = new Color(tipUI.color.r, tipUI.color.g, tipUI.color.b, 0f);
        tipUI.gameObject.SetActive(true);

        // Fade in
        float timer = 0f;
        while (timer < tipFadeTime)
        {
            float alpha = Mathf.Lerp(0f, 1f, timer / tipFadeTime);
            tipUI.color = new Color(tipUI.color.r, tipUI.color.g, tipUI.color.b, alpha);
            timer += Time.deltaTime;
            yield return null;
        }
        tipUI.color = new Color(tipUI.color.r, tipUI.color.g, tipUI.color.b, 1f);

        // Display
        yield return new WaitForSeconds(tipDisplayTime);

        // Fade out
        timer = 0f;
        while (timer < tipFadeTime)
        {
            float alpha = Mathf.Lerp(1f, 0f, timer / tipFadeTime);
            tipUI.color = new Color(tipUI.color.r, tipUI.color.g, tipUI.color.b, alpha);
            timer += Time.deltaTime;
            yield return null;
        }
        tipUI.color = new Color(tipUI.color.r, tipUI.color.g, tipUI.color.b, 0f);
        tipUI.gameObject.SetActive(false);
    }
}
```

Это скрипт обрабатывает взаимодействие игрока с объектами. Сценарий наследуется от класса «MonoBehaviour» и требуется для работы компонента «Collider2D». Метод «OnTriggerEnter2D» вызывается, когда коллайдер с тегом «Player» входит в триггер объекта. Метод «OnTriggerExit2D» вызывается, когда коллайдер покидает триггер объекта.

Листинг 2 – Interactable

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


[RequireComponent(typeof(Collider2D))]
public class Interactable : MonoBehaviour
{
    protected bool playerIsNear = false;


    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            playerIsNear = true;
            PlayerController.Instance.pressEUI.gameObject.SetActive(true);
        }
    }


    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            playerIsNear = false;
            PlayerController.Instance.pressEUI.gameObject.SetActive(false);
        }
    }
}
```

Данный скрипт обрабатывает диалоги с неигровыми персонажами (NPC). Он предназначен для отображения диалога конкретного NPC и управления им, и его функциональность включает в себя:

1. Создание и отслеживание диалогов.

2. Показ диалогов игроку.

3. Запуск диалога, когда игрок приближается к предмету.

4. Завершение диалога, когда игрок покидает зону диалога.

5. Разрешите игроку продолжить диалог, нажав клавишу «E».

6. Разрешение диалогу начинать диалоги квеста или непосредственно начинать задание.

7. Установите текущему состоянию диалога одно из трех возможных значений: «Не запущено», «В процессе» и «Готово».

Листинг 3 - NPCDialogue

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

public class NPCDialogue : MonoBehaviour
{
    [SerializeField] private List<Dialogue> _dialogues;
    private Dialogue _currentDialogue;
    private readonly Dialogue _nullDialogue = new Dialogue();
    private bool playerIsNear = false;

    public bool dialogueIsStarted = false;

    private void Start()
    {
        _currentDialogue = _nullDialogue;

        // Можно просто выключать панель перед началом игры, тогда этот код не понадобится
        //PlayerController.Instance.SetDialogueUIActive(false);

        // Запуск диалога при старте сцены
        /*if (gameObject.name == "StartDialogue")
        {
```

```
            playerIsNear = true;
            SetCurrentDialogue(0);
            _currentDialogue.StartDialogue();
        }*/
    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            Debug.Log("Игрок вошел в зону триггера");

            // Так можно задать нестандартные варианты диалогов, в зависимости от других условий
            /*switch (GetComponent<NPCQuest>().GetQuestState())
            {
                case NPCQuest.QuestState.NotStarted:
                    SetCurrentDialogue(0);
                    break;
                case NPCQuest.QuestState.InProcess:
                    SetCurrentDialogue(1);
                    break;
                case NPCQuest.QuestState.IsDone:
                    SetCurrentDialogue(2);
                    break;
            }*/

            playerIsNear = true;
            PlayerController.Instance.pressEUI.gameObject.SetActive(true);
            SetCurrentDialogue((int)GetComponent<NPCQuest>().GetQuestState());
        }
    }

    private void OnTriggerExit2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            Debug.Log("Игрок вышел из зоны триггера");
            playerIsNear = false;
            CloseDialogue();
        }
```

```
    }

    public void SetCurrentDialogue(int index)
    {
        _currentDialogue = _dialogues[index];
        _currentDialogue.characterNameUI = PlayerController.Instance.characterNameUI;
        _currentDialogue.dialogueTextUI = PlayerController.Instance.dialogueTextUI;
    }
    private void CloseDialogue()
    {
        PlayerController.Instance.pressEUI.gameObject.SetActive(false);
        _currentDialogue.EndDialogue();
        _currentDialogue = _nullDialogue;
        dialogueIsStarted = false;
        PlayerController.Instance.SetDialogueUIActive(false);
    }

    private void Update()
    {
        if (!playerIsNear) return;

        if (_currentDialogue != _nullDialogue && Input.GetKeyDown(KeyCode.E))
        {
            if (dialogueIsStarted)
            {
                _currentDialogue.NextLine();
            }
            else
            {
                dialogueIsStarted = true;
                _currentDialogue.StartDialogue();

                if (_currentDialogue.isStartQuestDialogue)
                    GetComponent<NPCQuest>().StartQuest();
            }
        }
    }
}
```

Можно разобрать интерактивный объект «горшок». Это класс реализует квестовый объект «Горшок». Метод `Update` вызывается один раз в каждом кадре и проверяет наличие игрока и текущее название сцены. Если взаимодействие игрока выполняется с помощью клавиши «E», код определяет соответствующее поведение на основе состояния задания. Метод «Поведение 1-го уровня» вводится, когда состояние задания "Выполнено" и игрок завершил предыдущие этапы задания. Код запускает диалоговое окно или сообщение с подсказкой в зависимости от доступных условий. Как только задание завершено, для объекта "Горшок" вызывается метод «Завершить задание», и соответствующие компоненты отключаются или скрываются.

Листинг 4 - Gorshok

```
using System;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.Serialization;

public class Gorshok : QuestObject
{
    public bool hasLeika;
    public bool leikaHasWater;

    public Sprite windowSprite;
    public Window window;

    void Update()
    {
        if (!playerIsNear) return;
        if (FindObjectOfType<CutScene>()) return;

        if (Input.GetKeyDown(KeyCode.E))
        {
            switch (SceneManager.GetActiveScene().name)
            {
                case "Level1":
```

```
                    Level1Behaviour();
                    break;
                default:
                    PlayerController.Instance.ShowTip("Опять это щупальце", 2f);
                    break;
            }
        }
    }
    private void Level1Behaviour()
    {
        switch (_quest.GetQuestState())
        {
            case NPCQuest.QuestState.NotStarted:
                // Запустится диалог
                break;
            case NPCQuest.QuestState.InProcess:
                if (hasLeika && leikaHasWater)
                {
                    PlayerController.Instance.ShowTip("Поливаем...", 2f);
                    window.gameObject.GetComponent<SpriteRenderer>().sprite = windowSprite;
                    GetComponent<AudioSource>().Play();
                    window.questIsDone = true;
                    GetComponent<SpriteRenderer>().enabled = false;
                    _quest.CompleteQuest();
                    break;
                }
                if (!GetComponent<NPCDialogue>().dialogueIsStarted)
                {
                    if (hasLeika)
                        PlayerController.Instance.ShowTip("Теперь нужно налить воды в лейку",
4f);
                    else
                        PlayerController.Instance.ShowTip("Ну щупальце, да", 2f);
                }
                break;
            case NPCQuest.QuestState.IsDone:
                // Запустится диалог
                break;
            default:
```

Листинг 4 - Продолжение

```
                throw new ArgumentOutOfRangeException();
            }
        }
    }
```

Список литературы

1. Разработка на Unity [Электронный ресурс]. - https://unity.com/ru/how-to/beginner-video-game-resources
2. Уроки разработки на Unity [Электронный ресурс]. - https://www.youtube.com/watch?v=GGsOU7sP0r4
3. Уроки разработки игр на Unity [Электронный ресурс]. - https://www.youtube.com/watch?v=IxAzjhTKFtA
4. Руководство разработки на Unity [Электронный ресурс]. - https://blog.skillfactory.ru/kak-sozdat-igru-na-unity/
5. Руководство для разработки на Unity [Электронный ресурс]. - https://proglib.io/p/razrabotka-igr-na-unity-s-nulya-do-professionala-2020-08-27