

**ОТЧЕТ**

по производственной практике ПП.01.01 по модулю ПМ.01  
«Разработка программных модулей программного обеспечения   
для компьютерных систем»

по специальности 09.02.03 «Программирование в компьютерных системах»

Студент 3 курса группы П1-17

Форма обучения: очная

Карандашева Николая Константиновича

Место прохождения практики

Государственном бюджетном образовательном учреждении высшего образования Московской области «Технологический университет»

(название организации)

Срок прохождения практики с «27» октября 2019 г. по «28» декабря 2019 г.

Руководители практики

От организации: заведующий мастерской \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Попов В.Н.

(Должность) (Подпись) (ФИО)

От колледжа: преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Эшанов А.А.

(подпись)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Оглавление**

[**Введение** 3](#_Toc28167025)

[**1.** **Общие сведения о организации.** 4](#_Toc28167026)

[**1.1.** **Структура организации характеристика основных видов деятельности**. 4](#_Toc28167027)

[**1.2.** **Должностные обязанности оператора ЭВМ, техника – программиста, инженера – программиста.** 4](#_Toc28167028)

[**1.2.1.** **Должностные обязанности оператора ЭВМ.** 4](#_Toc28167029)

[**1.2.2.** **Должностные обязанности техника – программиста.** 5](#_Toc28167030)

[**1.2.3.** **Должностные обязанности инженера – программиста.** 5](#_Toc28167031)

[**1.3.** **Основные функции отдела.** 6](#_Toc28167032)

[**1.4.** **Документооборот предприятия, структурного подразделения.** 7](#_Toc28167033)

[**2.** **Содержание выполняемых видов работ** 9](#_Toc28167034)

[**2.1.** **Разработка спецификаций отдельных компонентов**. 9](#_Toc28167035)

[**2.2.** **Скрипты для игры.** 9](#_Toc28167036)

[**2.3.** **Отладка.** 15](#_Toc28167037)

[**2.4.** **Тестирование программного продукта** 15](#_Toc28167038)

[**2.4.1.** **Юнит тестирование.** 15](#_Toc28167039)

[**2.4.2.** **Тестирование методом черного ящика.** 15](#_Toc28167040)

[**2.4.3.** **Отчет о проведении тестирования.** 16](#_Toc28167041)

[**2.5.** **Оптимизация программного кода.** 16](#_Toc28167042)

[**2.6.** **Обращение к программе.** 17](#_Toc28167043)

[**3.** **Выводы** 18](#_Toc28167044)

[**4.** **Заключение** 19](#_Toc28167045)

[**5.** **Дневник практики** 20](#_Toc28167046)

[**6.** **Список использованной литературы.** 21](#_Toc28167047)

[**7.** **Приложения.** 22](#_Toc28167048)

# 

# **Введение**

На 3 курсе обучения в ККМТ, студентом группы П1-17 Карандашевым Николаем была проведена производственная практика по модулю ПМ.01  
«Разработка программных модулей программного обеспечения   
для компьютерных систем». Студент получил задание разработать игру.

Во время прохождения практики я поставил для себя следующие цели:

* Приобрести опыт работы по специальности.
* Закрепить теоретические знания, полученные во время учебы.
* Проанализировать работы отдела.
* Закрепить навыки в разработке проектной и технической документации.
* Закрепить навыки отладки и тестирования программных модулей.

Для выполнения вышеупомянутых мной целей я выдвинул следующие задачи:

* Изучить специфику деятельности организации.
* Установить необходимые инструменты для работы.
* Найти подходящую литературу.

# **Общие сведения о организации.**

## **Структура организации характеристика основных видов деятельности**.

Данное предприятие работает в сфере образования. Университет образован 16 июля 1998 года в форме некоммерческой организации с названием: Негосударственное образовательное учреждение «Королевская академия управления, экономики и социологии».

Технологический университет (ранее Финансово-технологическая академия; Королевский институт управления, экономики и социологии) создан для подготовки кадров новой информации, воспроизводства интеллектуальных ресурсов, формирования инновационных проектов и технологий. Академия находится в наукограде Королеве Московской области – уникальном центре интеллектуальных ресурсов, которые используются для интеграции важнейших знаний и создания систем глобального масштаба.

20 января 2015 года постановлением Правительства Московской области Академии присвоен статус «университета» и вуз переименован в Государственное бюджетное образовательное учреждение высшего образования Московской области «Технологический университет».

Организационная структура колледжа представлена на Рис. 6.1 в Приложении 1.

## **Должностные обязанности оператора ЭВМ, техника – программиста, инженера – программиста.**

### **Должностные обязанности оператора ЭВМ.**

* осуществляет техническую подготовку документации, необходимой в процессе работы компании. Выполняет копирование документов на ксероксе;
* выполняет набор различных текстов с соблюдением правил орфографии и пунктуации, а также стандартов оформления организационно-распорядительной документации;
* осуществляет работу с электронной почтой, принимает входящие электронные письма и следит за своевременной отправкой исходящих;
* распечатывает и систематизирует нужные документы;
* заносит в компьютерные базы данных различную информацию, важную и необходимую для работы компании;
* следит за состоянием компьютера и копировальной техники;
* своевременно информирует руководство о необходимости приобретения материалов, непосредственно относящихся к производственному процессу.

### **Должностные обязанности техника – программиста.**

* выполняет работу по обеспечению механизированной и автоматизированной обработки, поступающей в ВЦ (ИВЦ) информации, разработки технологии решения экономических и других задач производственного и научно-исследовательского характера;
* принимает участие в проектировании систем обработки данных и систем математического обеспечения машины;
* выполняет подготовительные операции, связанные с осуществлением вычислительного процесса, ведет наблюдение за работой машин;
* составляет простые схемы технологического процесса обработки информации, алгоритмы решения задач, схемы коммутации, макеты, рабочие инструкции и необходимые пояснения к ним;
* разрабатывает программы решения простых задач, проводит их отладку и экспериментальную проверку отдельных этапов работ;
* выполняет работу по подготовке технических носителей информации, обеспечивающих автоматический ввод данных в вычислительную машину, по накоплению и систематизации показателей нормативного и справочного фонда, разработке форм исходящих документов, внесению необходимых изменений и своевременному корректированию рабочих программ;
* участвует в выполнении различных операций технологического процесса обработки информации (прием и контроль входной информации, подготовка исходных данных, обработка информации, выпуск исходящей документации и передача ее заказчику);
* ведет учет использования машинного времени, объемов выполненных работ;
* выполняет отдельные служебные поручения своего непосредственного руководителя.

### **Должностные обязанности инженера – программиста.**

* на основе анализа математических моделей и алгоритмов решения экономических и других задач разрабатывает программы, обеспечивающие возможность выполнения алгоритма и соответственно поставленной задачи средствами вычислительной техники, проводит их тестирование и отладку;
* разрабатывает технологию решения задач по всем этапам обработки информации;
* осуществляет выбор языка программирования для описания алгоритмов и структур данных;
* определяет информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля;
* выполняет работу по подготовке программ к отладке и приводит отладку;
* определяет объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению;
* осуществляет запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач;
* проводит корректировку разработанной программы на основе анализа выходных данных;
* разрабатывает инструкции по работе с программами, оформляет необходимую техническую документацию;
* определяет возможность использования готовых программных продуктов;
* осуществляет сопровождение внедрения программ и программных средств;
* разрабатывает и внедряет системы автоматической проверки правильности программ, типовые и стандартные программные средства, составляет технологию обработки информации;
* выполняет работу по унификации и типизации вычислительных процессов;
* принимает участие в создании каталогов и картотек стандартных программ, в разработке форм документов, подлежащих машинной обработке, в проектировании программ, позволяющих расширить область применения вычислительной техники.

## **Основные функции отдела.**

* Производственно-технологическая: разработка алгоритма решения задачи на основе предложенной модели; программная реализация алгоритма; отладка и тестирование программных продуктов; модификация программных продуктов; адаптация и настройка программных продуктов; сопровождение программных продуктов; разработка и эксплуатация баз данных; обеспечение достоверности информации при использовании баз данных;
* Организационно-управленческая: организация работы коллектива исполнителей; планирование и организация работ; выбор оптимальных решений при планировании работ в условиях нестандартных ситуаций; участие в оценке качества и экономической эффективности деятельности; обеспечение техники безопасности.

## **Документооборот предприятия, структурного подразделения.**

Документооборот Отдела в сфере поставленной мне на практике задачи состоит из нескольких этапов:

* получение приказа и распределение работы между сотрудниками;
* перечень существующих дел в Отделе;
* годовой план работ;
* годовой отчет по проделанной работе.

Вид построенной IDEF модели по плану документооборота представлен на рисунках 1 – 3:

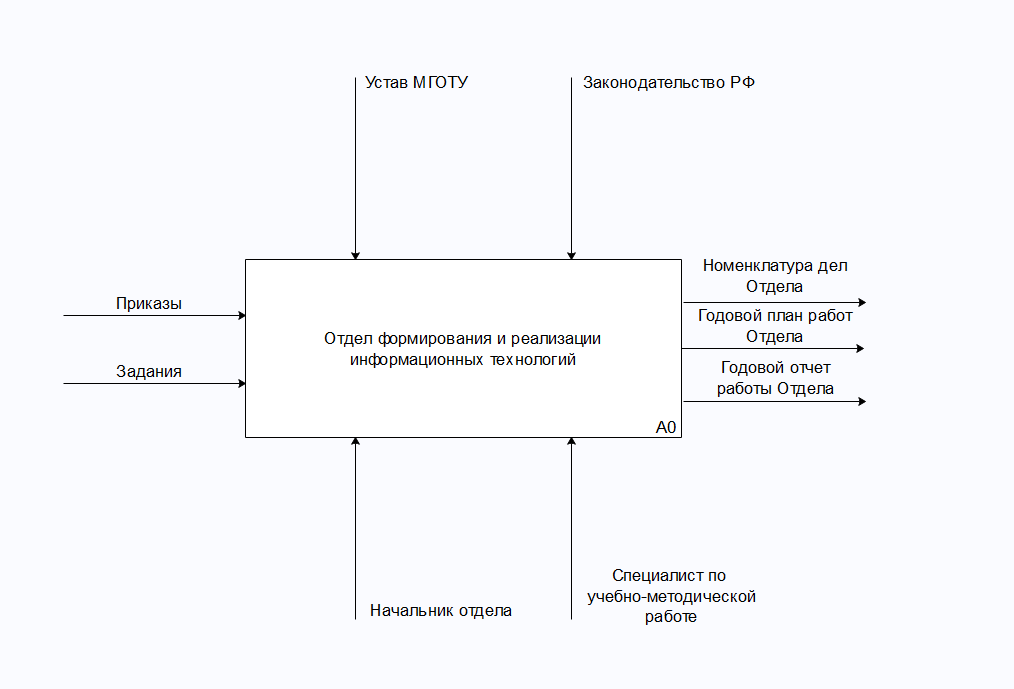


Рис. 1. - IDEF - модель 1 уровень

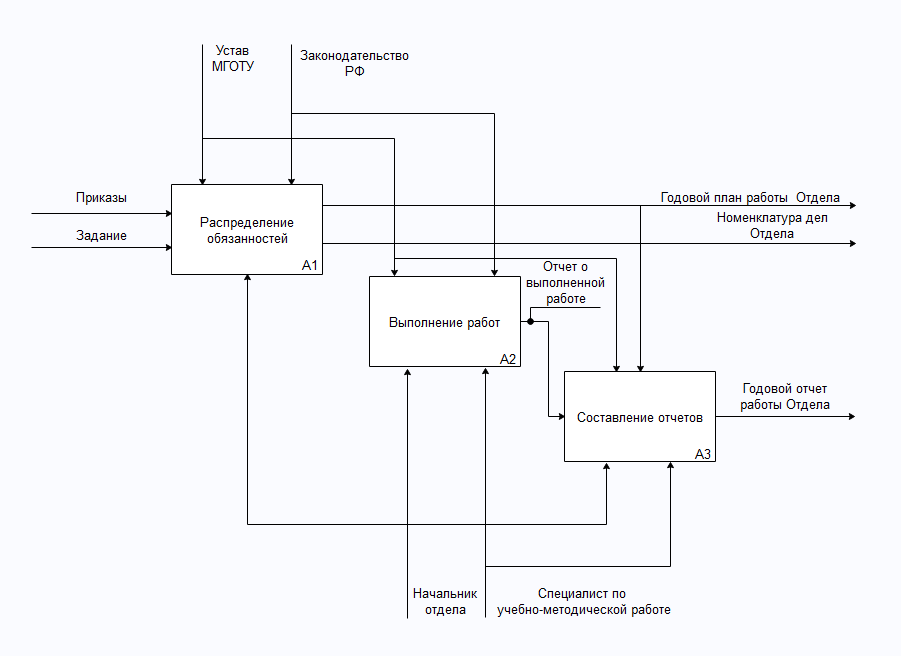


Рис. 2 - IDEF - модель 2 уровень

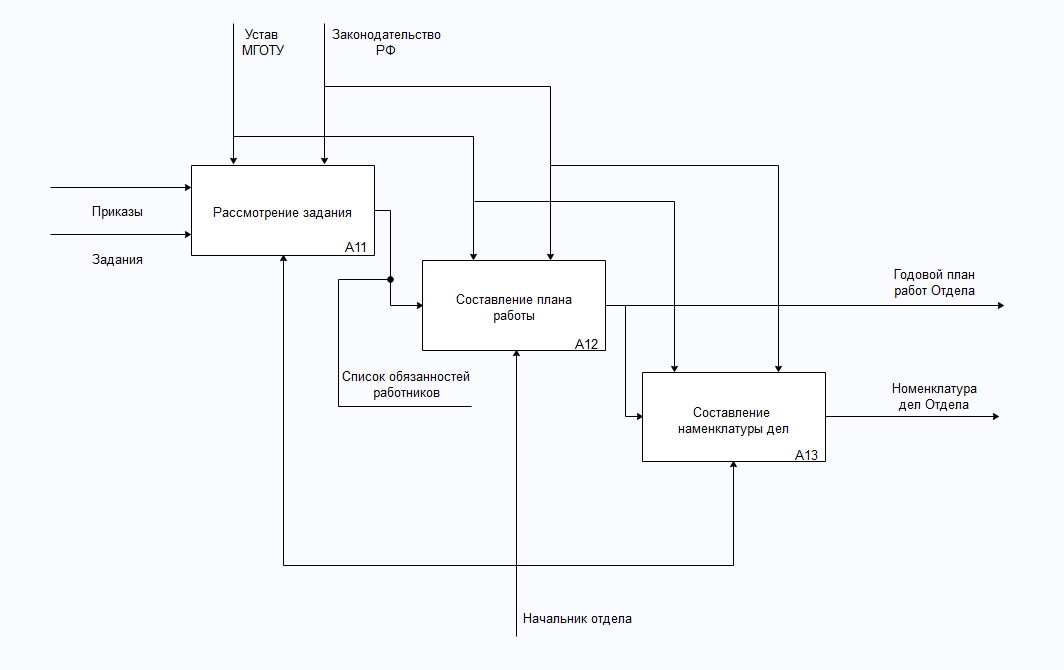


Рис. 3 – IDEF – модель подуровня блока «Распределение задания.

# **Содержание выполняемых видов работ**

## **Разработка спецификаций отдельных компонентов**.

Общее задание было разделено на 3 этапа:

1. Концепция. На основе выданного задания было принято решение писать игру в стиле «horror».
2. Написание сценария. В течение некоторого времени писался сценарий в последствии которого создавалась игра.

Входные данные: Word

1. Скрипты.

## **Скрипты для игры.**

**Листинг 1. Центральная обработка id**

using System.Collections;

using System.Collections.Generic;

using UnityEngine.UI;

using System;

using UnityEngine;

public class inventory : MonoBehaviour {

//public string time;

public GameObject[] wall;

public bool[] have\_keys = new bool [20] { false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false};

public bool[] open\_door = new bool[20] { false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false };

public bool[] sobution = new bool[25] { false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false };

public Text text\_message;

public lengths sav = new lengths();

public void Awake()//получаем индекс языка

{

if (PlayerPrefs.HasKey("LENGSV"))

{

sav = JsonUtility.FromJson<lengths>(PlayerPrefs.GetString("LENGSV"));

Debug.Log(sav.Language);

}

}

public void On(int index) обрабатываем сообщения и массивы в соответствии с индексом языка и пришедшего элемента

{

if (index < 1000 && index >= 0)

{

have\_keys[index] = true;

if (index == 19)

{

have\_keys[11] = true;

if (sav.Language == 1)

text\_message.text = "эта книга странная. я должен ее взять";

else

text\_message.text = "this book is strange. I should take it";

}

}

if (index >= 1000 && index < 2000)

{

if (have\_keys[index - 1000])

{

open\_door[index-1000] = true;

}

else

if (sav.Language == 1)

text\_message.text = "вам нужен ключ";

else

text\_message.text = "you need a key";

}

if(index >= 300000 && index < 400000)

{

if(index != 300004)

wall[index - 300000].SetActive(sobution[index - 300000]);

}

}

private void Update()

{

//text\_message.text = "вам нужен ключ";

if (Input.GetKeyDown(KeyCode.Space))

{

text\_message.text = "";

this.GetComponent<MainMove>().enabled = true;

this.GetComponent<SecondMove>().enabled = true;

}

}

[Serializable]

public class lengths

{

public int Language;

}

}

**Листинг 2. Первичная обработка.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using UnityEngine.Animations;

using UnityEngine.SceneManagement;

public class MainInv : MonoBehaviour

{

public GameObject script;

int ins = 0;

private void OnMouseEnter()//ловим наведение мыши на объект и красим его.

{

if (Vector3.Distance(script.transform.position, this.transform.position) < 7 )

{

if (this.tag == "message" || this.name== "200000")

GetComponent<Renderer>().material.color = Color.blue;

else if(this.tag != "door")

GetComponent<Renderer>().material.color = Color.gray;

ins = 1;

}

else ins = 0;

}

private void OnMouseExit()//ловим наведение мыши на объект и красим его.

{

GetComponent<Renderer>().material.color = Color.white;

ins = 0;

}

private void Update()//происходят анимации в соответствии с индексом

{

if (ins == 1 && Vector3.Distance(script.transform.position, this.transform.position) < 7)

{

if (Input.GetKeyDown(KeyCode.E))

{

int k = Convert.ToInt32(this.name);

Debug.Log(k);

if (k < 1000 && k >= 0)

{

script.GetComponent<inventory>().On(k);

if (this.name == "19")

{

if (GameObject.Find("130000") != null)

{ GameObject.Find("130000").GetComponent<Animation>().Play("statue");

}

}

if (this.name != "7")

Destroy(this.gameObject);

else

{

this.GetComponent<BoxCollider>().enabled = false;

script.GetComponent<inventory>().On(1007);

GameObject.Find("1007").GetComponent<Animation>().Play("open");

}

}

else if (k >= 1000 && k < 2000)

{

if (script.GetComponent<inventory>().have\_keys[k - 1000] == true && script.GetComponent<inventory>().open\_door[k - 1000]==false)

{

if (this.name == "1008")

{

script.GetComponent<save>().Save();

SceneManager.LoadScene("test horor");

}

this.GetComponent<AudioSource>().Play();

script.GetComponent<inventory>().On(k);

this.GetComponent<Animation>().Play("open");

if (this.name == "1005")

{ script.GetComponent<inventory>().sobution[2] = true;

script.GetComponent<inventory>().wall[2].GetComponent<Animation>().Play("go\_manik");

}

}

else script.GetComponent<inventory>().On(k);

}

else if (k >= 3000 && k < 100000)

{

script.GetComponent<inventory>().On(k);

Destroy(this.gameObject);

}

else if (k == 200000)

{

this.GetComponent<Animation>().Play("new");

this.GetComponent<BoxCollider>().enabled = false;

}

else if (k >= 1000000 && k < 4000000)

{

script.GetComponent<text\_col>().doing(k);

script.GetComponent<MainMove>().enabled = false;

script.GetComponent<SecondMove>().enabled = false;

}

else if (k == -1)

{

script.GetComponent<save>().Save();

SceneManager.LoadScene("SampleScene");

//else script.GetComponent<inventory>().On(k);

}

else if (k == -2)

{

script.GetComponent<save>().Save();

SceneManager.LoadScene("its\_house");

//else script.GetComponent<inventory>().On(k);

}

else if (k == -4)

{

//script.GetComponent<save>().Save();

SceneManager.LoadScene("first");

//else script.GetComponent<inventory>().On(k);

}

else if (k == -100)

{

script.GetComponent<save>().Save();

SceneManager.LoadScene("THE\_END");

//else script.GetComponent<inventory>().On(k);

}

}

ins = 1;

}

}

}

**Листинг 3. Обработка текста**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEditor;

using System;

using System.IO;

using UnityEngine.UI;

public class text\_col : MonoBehaviour {

public Text message;

private string lang = "RUS";

public GameObject pan;

public lengths sav = new lengths();

public void Awake()//проверяем язык

{

if (PlayerPrefs.HasKey("LENGSV"))

{

sav = JsonUtility.FromJson<lengths>(PlayerPrefs.GetString("LENGSV"));

Debug.Log(sav.Language);

}

}

public void doing(int k)//достаем из файла соответствующий текст и выводим его на лист

{

string sentence = "";

pan.SetActive(true);

if (sav.Language == 1)

lang = "RUS";

else

lang = "ENG";

string str = k.ToString();

this.GetComponent<MainMove>().enabled = false;

var textFile = Resources.Load<TextAsset>("text/" + lang + "/" + str);

sentence = textFile.name;

message.text = textFile.text;

}

private void OnTriggerEnter(Collider other)

{

int k = Convert.ToInt32(other.name);

if (k < 4000000 && k >= 1000000) ;

}

private void Update()//закрываем лист

{

if (Input.GetKeyDown(KeyCode.Space))

{

pan.SetActive(false);

this.GetComponent<MainMove>().enabled = false;

}

}

[Serializable]

public class lengths

{

public int Language;

}

}

## **Отладка.**

До отладки проведения отладки:

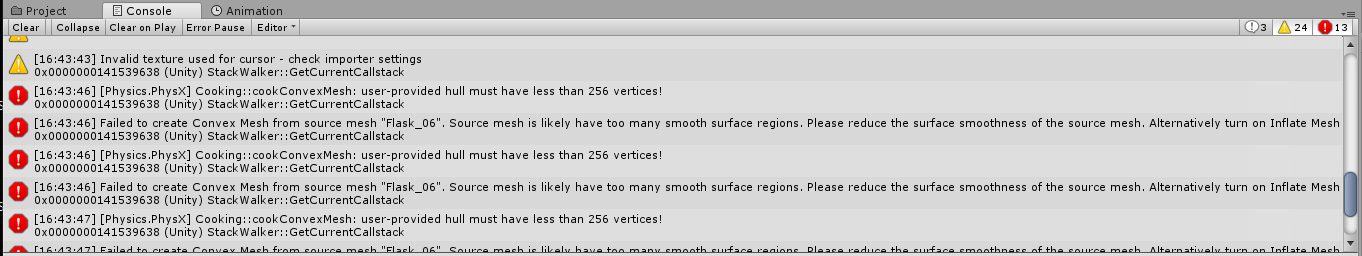


Рис.4 Ошибки до проведения отладки

После проведения отладки:

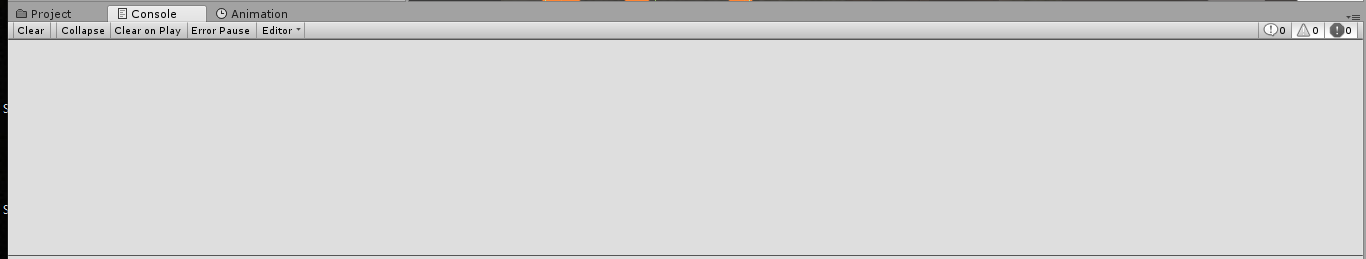


Рис.5 Ошибки после проведения отладки

## **Тестирование программного продукта**

### **Юнит тестирование.**

Для проверки на корректность работы кода, был создан юнит тест “Test1” его код предоставлен ниже, который демонстрирует корректность настройки уровня звука и графики.

**Пример - Теста программы.**

[TestMethod]

public void Test1()

{

float sound = 5.5;

int quality = 4;

Sounds.SetFloat("Sounds", sound);

QualitySettings.SetQualityLevel(quality);

}

### **Тестирование методом черного ящика.**

Тестирование корректной работы программы методом черного ящика проводятся по следующим сценариям:

Сценарий 1: Включение фонарика. Тестирования работы фонарика при нажатии определенной клавиши.

1. Нажать F.

Результат: Включение фонарика в игре.

Сценарий 2: Взаимодействие с ключами. Тестирование работы взаимодействия с ключами.

1. Подойти к ключу.
2. Навести на него.
3. Нажать Е.

Результат: Добавления ключа в инвентарь и удаление его со сцены.

Сценарий 3: Взаимодействие с дверьми. Тестирование работы взаимодействия с дверьми.

1. Подойти в двери.
2. Навести на нее.
3. Нажать Е.

Результат: Если есть ключ – откроется дверь, если нет – появится сообщение об отсутствии ключа.

### **Отчет о проведении тестирования.**

При тестировании программы методом черного ящика по вышеуказанным сценариям прошли успешно.

## **Оптимизация программного кода.**

**Пример неоптимизированного кода:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class faal : MonoBehaviour {

private void Start()//анимация падения

{

int[] mass = new int[7];

mass[0] = 1;

mass[1] = 2;

mass[2] = 3;

mass[3] = 4;

mass[4] = 5;

mass[5] = 6;

mass[6] = 7;

}

}

**Этот же код после оптимизации:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class faal : MonoBehaviour {

private void Start()//анимация падения

{

int[] mass = new int[7];

for(int i = 0;i < 7;i++)

mass[i] = i + 1;

}

}

## **Обращение к программе.**

Для запуска программы не обходимо выполнить следующие действия:

* Запустить на ПК двойным щелчком мыши.
* Запустится preview (превью).
* Откроется главное меню.
* Начать игру.

Для игроков, не имеющих сохранение, будет не доступно возможность продолжить игру. Для игроков, имеющих сохранение, при запуске новой игры сохранения будет удалены.

# **Выводы**

Полученные навыки:

* Дизайн уровней Unity3D
* Разработка скриптов в Visual Studio 2019 для игр на базе Unity3D
* Проектирование и создание 3D моделей в 3Ds Max

Полученные умения:

* Работа в Unity 3D
* Работа в Visual Studio 2019
* Работа с 3D моделями в 3Ds Max

# **Заключение**

Перед прохождением производственной практики в Государственном бюджетном образовательномучреждении высшего образования Московской области «Технологический университет» мной были поставлены следующие основные цели:

* Приобрести опыт работы по специальности.
* Закрепить теоретические знания, полученные во время учебы.
* Выполнение требований и действий, предусмотренных программой производственной практики и заданий руководителя.
* Закрепить навыки в разработке проектной и технической документации.
* Закрепить навыки отладки и тестирования программных модулей.

По окончанию практики я добился, чего хотел. Для достижения своих целей я использовал интернет – источники, документацию средств разработки.

Во время прохождения практики я приобрел опят работы по специальности. Так же был закреплен навык разработки проектной и технической документации и навык отладки и тестирования программных модулей.

По окончанию практики был составлен отчёт.

# **Дневник практики**

Таблица 1. Дневник практики.

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Содержание работы | Отметка о выполнении  работы | Подпись руководителя практики |
| 28.10 – 4.11 | Обсуждение концепции игры. Выбор языка программирования, выбор среды разработки, выбор программного обеспечения для 3D-моделирования. |  |  |
| 1.11 – 24.11 | Обсуждение и создание сюжета для игры. Разработка первичных UI объектов, зарисовки главных сцен проекта. |  |  |
| 25.11 – 6.12 | Определение масштабов в игре. Проектирование и создание 3D моделей в 3Ds Max. Разработка текстур и материалов объекта. |  |  |
| 2.12 – 13.12 | Работа в Unity3D. Конструирование игровых сцен, скачивание дополнительных assets, наложение текстур и материалов на соответствующие объекты. |  |  |
| 25.11 – 2.12 | Разработка музыкального сопровождения, подбор, обработка и написание звуковых эффектов. |  |  |
| 25.11 – 20.12 | Работа со скриптами. Система меню и настроек, система сохранения и загрузки, система взаимодействия с объектами, система работы с триггерами, система контроля анимации. |  |  |
| 2.12 – 27.12 | Оформление отчёта. Обработка листингов, группировка отчёта. |  |  |

# **Список использованной литературы.**

1. <https://habr.com/ru/post/346370/>
2. <https://docs.unity3d.com/ru/530/Manual/>
3. <http://www.cyberforum.ru/unity/>
4. <http://www.unity3d.ru/distribution/index.php>
5. <https://www.youtube.com/playlist?list=PLpyssslyeRz6Yd4SdrY-O_kyFiyeK8w6l>
6. <http://docs.unity3d.ru/>

# **Приложения.**

**Приложение 1.**

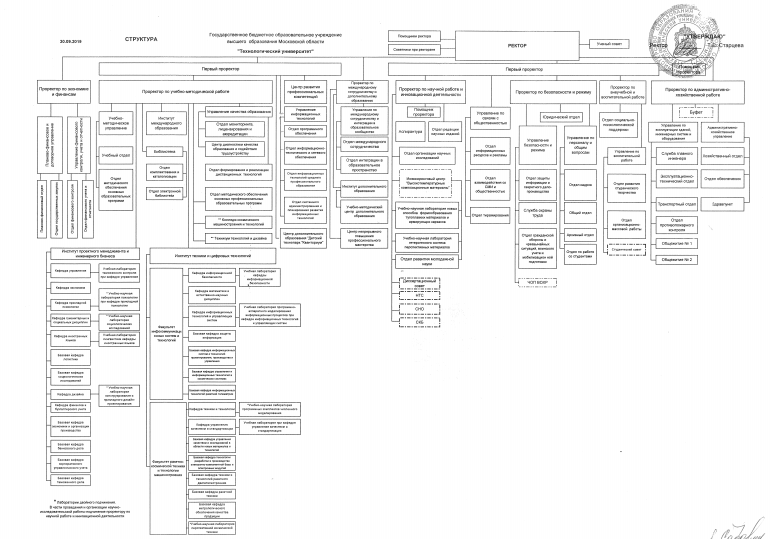


Рис. 6.1. Организационная структура колледжа

**Приложение 2.**

**Листинг 7.1 Центральный скрипт сохранения**

using System.Collections;

using System.Collections.Generic;

using System;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.Animations;

public class save : MonoBehaviour

{

public sv sav = new sv();

public void Save()

{

sav.position\_player = GameObject.Find("time\_pleyer").transform.position;

sav.rotation\_player.x = GameObject.Find("time\_pleyer").transform.rotation.x;

sav.rotation\_player.z = GameObject.Find("time\_pleyer").transform.rotation.z;

sav.rotation\_player.y = 90;

sav.sve\_keys = GameObject.Find("time\_pleyer").GetComponent<inventory>().have\_keys;

sav.sve\_door = GameObject.Find("time\_pleyer").GetComponent<inventory>().open\_door;

sav.sve\_sobution= GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution;

sav.name = Application.loadedLevelName;

PlayerPrefs.SetString("SV", JsonUtility.ToJson(sav));

PlayerPrefs.Save();

}

public void Load()//загрузка сохраненной сцены

{

if (PlayerPrefs.HasKey("SV"))

{

sav = JsonUtility.FromJson<sv>(PlayerPrefs.GetString("SV"));

Screen.lockCursor = true;

Cursor.visible = false;

SceneManager.LoadScene(sav.name);

}

}

public void Awake()//загрузка компонентов

{

if (PlayerPrefs.HasKey("SV"))

{

sav = JsonUtility.FromJson<sv>(PlayerPrefs.GetString("SV"));

GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution = sav.sve\_sobution;

if (Application.loadedLevelName == sav.name)

{

GameObject.Find("time\_pleyer").transform.position = sav.position\_player;

GameObject.Find("time\_pleyer").transform.rotation = sav.rotation\_player;

GameObject.Find("time\_pleyer").GetComponent<inventory>().have\_keys = sav.sve\_keys; GameObject.Find("time\_pleyer").GetComponent<inventory>().open\_door = sav.sve\_door;

for (int i = 0; i < 20; i++)

{

int k = i + 1000;

if (sav.sve\_door[i])

{

string u = k.ToString();

if (GameObject.Find(u) != null)

{

GameObject.Find(u).GetComponent<Animation>().Play("open\_moment");

}

}

if (sav.sve\_keys[i])

{

string name = i.ToString();

if (GameObject.Find(name) != null)

if (i == 7)

{ GameObject.Find("7").GetComponent<BoxCollider>().enabled = false;

}

else if (i == 8)

{

if(GameObject.Find("132000") != null) Destroy(GameObject.Find("132000"));

Debug.Log("jghihu");

}

else if (i == 19)

{

if (GameObject.Find("130000") != null)

{

GameObject.Find("130000").GetComponent<Animation>().Play("statue");

}

}

else GameObject.Find(name).SetActive(!sav.sve\_keys[i]);

}

}

}

for (int i = 0; i < 25; i++)

{

if (sav.sve\_sobution[i])

{

if (i == 0)

{

if (GameObject.Find("300000") != null)

{

Destroy(GameObject.Find("300000"));

GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[i].SetActive(true);

}

}

else if (i == 2)

{

if (GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[i] != null)

GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[i].GetComponent<Animation>().Play("moment\_going\_manicen");

}

else if (i == 4)

{

if (GameObject.Find("300004") != null)

Destroy(GameObject.Find("300004"));

}

else if (i == 1)

{

if (GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[i] != null)

GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[i].SetActive(true);

}

else if (i == 5)

{

if (GameObject.Find("dors") != null)

{ GameObject.Find("dors").GetComponent<Animation>().Play("fals\_moment"); GameObject.Find("dors").GetComponent<BoxCollider>().enabled = false;

}

}

else

{

string n = Convert.ToString(300000 + i);

if (GameObject.Find(n) != null)

Destroy(GameObject.Find(n));

}

}

} GameObject.Find("time\_pleyer").GetComponent<inventory>().have\_keys = sav.sve\_keys; GameObject.Find("time\_pleyer").GetComponent<inventory>().open\_door = sav.sve\_door;

}

if(sav.name== "its\_house" && Application.loadedLevelName == "SampleScene")

{

GameObject.Find("time\_pleyer").transform.position = new Vector3(810, 146, 703);

//Destroy(GameObject.Find("car"));

}

}

}

[Serializable]

public class sv

{

public Vector3 position\_player;

public Quaternion rotation\_player;

public bool[] sve\_keys;

public bool[] sve\_door;

public bool[] sve\_sobution;

public string name;

}

**Приложение 3.**

**Листинг 7.2. Сохранение меню.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using UnityEngine.UI;

using UnityEngine.Audio;

public class SaveMenu : MonoBehaviour

{

public stn settings = new stn();

public lng leng = new lng();

public void BackPressed()//сохранение значений настроек

{

settings.FullScreen = GetComponent<SettingsControl>().full;

settings.Sounds = GetComponent<SettingsControl>().sounds;

settings.Music = GetComponent<SettingsControl>().music;

settings.Quality = GetComponent<SettingsControl>().q1;

settings.ScreenResolution = GetComponent<SettingsControl>().r1;

settings.Language = GetComponent<SettingsControl>().language;

leng.Language = GetComponent<SettingsControl>().language;

PlayerPrefs.SetString("SETSV", JsonUtility.ToJson(settings));

PlayerPrefs.SetString("LENGSV", JsonUtility.ToJson(leng));

PlayerPrefs.Save();

gameObject.SetActive(false);

}

public void Awake()//загрузка значений настроек

{

if (PlayerPrefs.HasKey("SETSV"))

{

settings = JsonUtility.FromJson<stn>(PlayerPrefs.GetString("SETSV"));

GameObject.Find("Toggle").GetComponent<Toggle>().isOn = settings.FullScreen; GameObject.Find("SoundsSlider").GetComponent<Slider>().value = settings.Sounds; GameObject.Find("MusicSlider").GetComponent<Slider>().value = settings.Music; GameObject.Find("QualityDropdown").GetComponent<Dropdown>().value = settings.Quality; GameObject.Find("ScreenResolutionDropdown").GetComponent<Dropdown>().value = settings.ScreenResolution; GameObject.Find("LanguageDropdown").GetComponent<Dropdown>().value = settings.Language;

}

}

[Serializable]

public class stn

{

public bool FullScreen;

public float Sounds;

public float Music;

public int Quality;

public int ScreenResolution;

public int Language;

}

[Serializable]

public class lng

{

public int Language;

}

}

**Приложение 4.**

**Листинг 7.3. Контроль первой сцены**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEditor;

using System;

using System.IO;

using UnityEngine.UI;

public class first\_animation\_player : MonoBehaviour {

int k = 1;

bool h = false;

private string lang = "RUS";

public GameObject phone;

public Text mes;

public lengths sav = new lengths();

public sv savs = new sv();

private void Awake()//запускаем анимацию при соответствии настроек

{

if (PlayerPrefs.HasKey("SV"))

{

savs = JsonUtility.FromJson<sv>(PlayerPrefs.GetString("SV"));

if ((savs.sve\_keys[8] == true))

{

this.GetComponent<inventory>().sobution[0] = false;

this.GetComponent<Animation>().Play("stand up");

phone.GetComponent<Animation>().Play("on");

phone.GetComponent<BoxCollider>().enabled = true;

GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[0].SetActive(true);

}

}

if (!PlayerPrefs.HasKey("SV"))

{

h = true;

this.GetComponent<Animation>().Play("stand up");

phone.GetComponent<Animation>().Play("on");

h = true;

}

if (PlayerPrefs.HasKey("LENGSV"))

{

sav = JsonUtility.FromJson<lengths>(PlayerPrefs.GetString("LENGSV"));

if (sav.Language == 1)

lang = "RUS";

else

lang = "ENG";

//Debug.Log(sav.Language);

}

}

private void Update()//монолог

{

if (!PlayerPrefs.HasKey("SV"))

{

//mes.text = "sfgZVdZGzFDGzasrfg";

/\*string sentence;

StreamReader sr = new StreamReader("Assets\\text\\RUS\\4000001.txt");

sentence = sr.ReadToEnd();

mes.text = sentence;\*/

if (k == 1)

{

if (sav.Language == 1)

mes.text = "5 лет назад мой брат ушел из дома и без вести пропал. Надежда найти Томаса с каждым годом становится все меньше и меньше. Я чувствую, что он жив и не теряю надежды найти его.";

else

mes.text = "Five years ago, my brother left home and went missing. The hope of finding Thomas every year becomes less and less. I feel like he's alive and I'm not giving up hope of finding him.";

}

if (k == 2)

{

if (sav.Language == 1)

mes.text = "Какого черта, 5 утра!";

else

mes.text = "What the hell, 5 in the morning!";

}

if (Input.GetKeyDown(KeyCode.Space) && k <= 4)

{

k++;

}

if (k == 3)

{

if (sav.Language == 1)

mes.text = "- Да, слушаю. Кто это?\n- У меня есть информация о твоем брате.\n-Кто вы? Откуда у вас мой номер?\n-Слушай меня.Если тебе не безразличен твой брат, то можешь приехать и посмотреть на его смерть.";

else

mes.text = "-Hello. Who is that?\n- I have information about your brother.\n- Who are you? Where did you get my phone number ?\n-Listen to me.If you care about your brother, you can come and see his death.";

phone.SetActive(false);

}

if (k == 4)

{

if (sav.Language == 1)

mes.text = "Что здесь происходит? Почему этот человек звонит мне и говорит о моем брате спустя столько лет?";

else

mes.text = "What's going on here? Why does this man call me and talk about my brother after so many years?";

phone.SetActive(false);

}

if (k == 5)

{

mes.text = "";

h = false;

}

}

}

//Destroy(GameObject.Find("car"));

[Serializable]

public class sv

{

public Vector3 position\_player;

public Quaternion rotation\_player;

public bool[] sve\_keys;

public bool[] sve\_door;

public bool[] sve\_sobution;

public string name;

}

[Serializable]

public class lengths

{

public int Language;

}

}

**Приложение 5.**

**Листинг 7.4. Универсальный скример.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

public class scream\_room\_2 : MonoBehaviour {

private void OnTriggerEnter(Collider other)//функция на вход в зону

{

if(other.tag == "Player")//проверка что это был игрок

{

int k = Convert.ToInt32(this.name); GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution[k - 300000] = true;//вычитаем из индекса коэффициент и присваиваем значение 1 в соответствующем элементе

GameObject.Find("time\_pleyer").GetComponent<inventory>().On(k);

//передаем на обработку

if (k == 300000)

{ GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution[k - 300000] = true; GameObject.Find("time\_pleyer").GetComponent<inventory>().open\_door[1] = false;//закрываем дверь GameObject.Find("1001").GetComponent<Animation>().Play("close\_1001");//анимация закрывания GameObject.Find("1001").GetComponent<AudioSource>().Play();

//музыка

Destroy(this.gameObject);//уничтожение триггера

}

if (k == 300001)

{ GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution[k - 300000] = true; GameObject.Find("time\_pleyer").GetComponent<inventory>().wall[k - 300000].SetActive(true);//скрытие триггера

}

if (k == 300004)

{ GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution[4] = true; GameObject.Find("fly\_room").GetComponent<Animation>().Play();

Destroy(this.gameObject);//уничтожение триггера

}

}

}

}

**Приложение 6.**

**Листинг 7.5. Музыка**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

public class music : MonoBehaviour {

private void OnTriggerEnter(Collider other)

{

if (other.tag == "Player")

{

int k = -300000 + Convert.ToInt32(this.name);

if (!this.GetComponent<AudioSource>().isPlaying && other.GetComponent<inventory>().sobution[k] == false)//проверяем что этот триггер не включался

{

other.GetComponent<inventory>().sobution[k] = true;//соответствующему элементу даем 1

this.GetComponent<AudioSource>().Play();//проигрывание звука

}

}

}

}

**Приложение 7.**

**Листинг 7.6. Триггер падения**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class faal : MonoBehaviour {

private void OnTriggerEnter(Collider other)//анимация падения

{

if (GameObject.Find("time\_pleyer").GetComponent<inventory>().have\_keys[8]) {//проверка на условие падение в подвал

GameObject.Find("dors").GetComponent<Animation>().Play("fals");

//запуск анимации

this.GetComponent<BoxCollider>().enabled = false;//отключение триггера

GameObject.Find("time\_pleyer").GetComponent<inventory>().sobution[5] = true;//передаем 1;

}

}

}

**Приложение 8.**

**Листинг 7.7. Титры**

using System.Collections;

using System.Collections.Generic;

using System;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

using UnityEngine.Animations;

public class end : MonoBehaviour {

public Text rus;

public bool k=false;

public lengths sav = new lengths();

public void Awake()//запуск титров в соответствии с языком

{

if (PlayerPrefs.HasKey("LENGSV"))

{

sav = JsonUtility.FromJson<lengths>(PlayerPrefs.GetString("LENGSV"));

if (sav.Language == 1)

rus.GetComponent<Animation>().Play();

else

{

rus.text = "Screenwriter Popkova A.I. \n\n Artist Sapoletova A.V. \n\n Responsible for the music of Surkova V.A. \n\n SecomdProgrammer Yushakov N.R. \n\n Main programmer Karandashev N.K. \n\n We Express our gratitude for the help Popovu VN and all who passed this game";

rus.GetComponent<Animation>().Play();

}

}

else

{

rus.text = "Screenwriter Popkova A.I. \n\n Artist Sapoletova A.V. \n\n Responsible for the music of Surkova V.A. \n\n Second Programmer Yushakov N.R. \n\n Main programmer Karandashev N.K. \n\n We Express our gratitude for the help Popovu VN and all who passed this game";

rus.GetComponent<Animation>().Play();

}

}

private void Update()

{

if (k)

{

Screen.lockCursor = false;

Cursor.visible = true;

SceneManager.LoadScene("SceneMainMenu");

}

}

[Serializable]

public class lengths

{

public int Language;

}

}

**Приложение 9.**

**Листинг 7.8. Анимация камеры.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class camera\_trans : MonoBehaviour {

void Update () {//запускает анимацию в любой момент при запуске игры

if ((Input.GetKey(KeyCode.W) || (Input.GetKey(KeyCode.S))) && !GetComponent<Animation>().IsPlaying("camera"))//проверка на условия

GetComponent<Animation>().Play("amera\_2");//запуск анимации

}

}

**Приложение 10.**

**Листинг 7.9. скрипт двери из красной сцены.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class door : MonoBehaviour

{

public GameObject left;

public GameObject right;

public GameObject left\_2;

public GameObject right\_2;

private void OnTriggerEnter(Collider other)//запускает действия триггера при касании

{

if (other.tag == "Player")//проверка на игрока

{

left.GetComponent<Animation>().Play("of");//запуск анимации

left\_2.SetActive(true);//активация объекта

right.GetComponent<Animation>().Play("on");

left.GetComponent<AudioSource>().Play();

right\_2.SetActive(true);

Destroy(this.gameObject);//уничтожение триггера

}

}

}

**Приложение 11.**

**Листинг 7.10. Тестовый скрипт звуков**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class our\_scream\_sound: MonoBehaviour

{

public GameObject scream;

public GameObject screamNext;

private AudioSource clip;

bool have = false;

void Start()//запускаем музыку при соприкосновении с объектом

{

clip = GetComponent<AudioSource>();

}

private void Update()

{

if(have && !clip.isPlaying)

Destroy(scream.transform.gameObject);

}

private void OnTriggerEnter(Collider other)

{ if (have == false)

{

clip.Play();

screamNext.SetActive(true);

have = true;

}

}

}

**Приложение 12.**

**Листинг 7.11. Тестовый скрипт записок.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class our\_text\_1 : MonoBehaviour {

public GameObject scream;

public GameObject panel;

public GameObject screamNext;

private bool have = false;

private void Update()

{

if (have && Input.GetKeyDown(KeyCode.Space))//скрываем текстовое окно

{

Destroy(scream.transform.gameObject);

panel.SetActive(false);

}

}

private void OnTriggerEnter(Collider other)//получаем в текстовое окно скример

{

if (have == false)

{

panel.SetActive(true);

screamNext.SetActive(true);

have = true;

}

}

}

**Приложение 13.**

**Листинг 7.12. Тестовый скрипт перехода**

using UnityEngine;

using UnityEngine.SceneManagement;

public class first\_scene : MonoBehaviour {

private void OnTriggerEnter(Collider other)//запускаем новую сцену при касании с триггером

{

if(other.tag =="Player")

SceneManager.LoadScene("SampleScene");

}

}

**Приложение 14.**

**Листинг 7.13. Второй скрипт управления.**

using System.Collections;

using System.Collections.Generic;

using System;

using UnityEngine.UI;

using UnityEngine;

public class SecondMove : MonoBehaviour {

public GameObject flashLight;

bool have\_light = false;

public Text message;

// Use this for initialization

void Start () {//ВКЛЮЧАЕМ ФОНАРИК

}

// Update is called once per frame

void Update () {

if (Input.GetKeyDown(KeyCode.F))

{

flashLight.SetActive(have\_light);

have\_light = !have\_light;

}

}

private void OnTriggerEnter(Collider other)//держим закрытые зоны + пасхалка

{

int k = Convert.ToInt32(other.name);

if (k >= 2000 && k < 3000)

message.text = "я не хочу туда идти. мне там ничего не нужно";

if (k == 10001)

message.text = "эти статуи в крови. мне это не нравится";

}

}

**Приложение 15.**

**Листинг 7.14. Центральный скрипт движения.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MainMove : MonoBehaviour {

public float senityX = 3f;

public float senityY = 3f;

public float minX = -360;

public float maxX = 360;

public float minY = -60;

public float maxY = 60;

private Quaternion OriginalRoat;

private float rotx = 0;

private float roty = 0;

//передвижение

public float speed = 6f;

public float speedSteep = 6f;

public float speedShift = 29f;

public float gravity = 20f;

public float rotation = 4f;

private float now\_speed;

private Vector3 moveDir = Vector3.zero;

private CharacterController controll;

void Start () {

controll = GetComponent<CharacterController>();

OriginalRoat = transform.localRotation;

//SetCursorLocked();

}

// https://www.youtube.com/watch?v=1WaeNaq5AJM#action=share

//https://www.youtube.com/watch?v=w9JV1R\_VyOs

void Update () {//само управление

rotx += Input.GetAxis("Mouse X") \* senityX;

roty += Input.GetAxis("Mouse Y") \* senityY;

rotx %= 360;

roty %= 360;

rotx = Mathf.Clamp(rotx, minX, maxX);

roty = Mathf.Clamp(roty, minY, maxY)

Quaternion xQ = Quaternion.AngleAxis(rotx, Vector3.up);

Quaternion yQ = Quaternion.AngleAxis(roty, Vector3.left);

transform.localRotation = OriginalRoat \* xQ \* yQ;

//moveDir = transform.TransformDirection(moveDir) \* speed;

if (Input.GetKey(KeyCode.LeftShift))

now\_speed = speedShift;

else

now\_speed = speed;

moveDir = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));

moveDir = transform.TransformDirection(moveDir);

moveDir \*= now\_speed;

moveDir.y = 0;

moveDir.y -= gravity \* Time.deltaTime;

controll.Move(moveDir \* Time.deltaTime);

}

}

**Приложение 16.**

**Листинг 7.15. Смена языка в меню.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using UnityEngine.UI;

using UnityEngine.Audio;

public class swap\_menu : MonoBehaviour

{

public Text[] Eng;

lng len = new lng();

private void Awake()//получаем язык

{

if (PlayerPrefs.HasKey("LENGSV"))

{

len = JsonUtility.FromJson<lng>(PlayerPrefs.GetString("LENGSV"));

if (len.Language == 1)

{

Eng[0].text = "Новая игра";

Eng[1].text = "Продолжить игру";

Eng[2].text = "Настройки";

Eng[3].text = "Выход";

Eng[4].text = "ваш прогресс будет удален, продолжить?";

Eng[5].text = "Да";

Eng[6].text = "Нет";

Eng[7].text = "полный экран";

Eng[8].text = "Звуки";

Eng[9].text = "Музыка";

Eng[10].text = "Качество";

Eng[11].text = "Расширение";

Eng[12].text = "Язык";

Eng[13].text = "Назад";

Eng[14].text = "у вас нет сохранения";

Eng[15].text = "сейчас начнем!";

}

else

{

Eng[0].text = "New game";

Eng[1].text = "Load game";

Eng[2].text = "Settings";

Eng[3].text = "Exit";

Eng[4].text = "After this action, all unsaved progress will be lost. Continue?";

Eng[5].text = "Yes";

Eng[6].text = "No";

Eng[7].text = "FullScreen";

Eng[8].text = "Sounds";

Eng[9].text = "Music";

Eng[10].text = "Quality";

Eng[11].text = "Screen Resolution";

Eng[12].text = "Language";

Eng[13].text = "Back";

Eng[14].text = "you have no save";

Eng[15].text = "now brgin!";

}

}

}

public void swap\_settings(int k)//меняем в соответствии с языком

{

if (k == 1)

{

Eng[0].text = "Новая игра";

Eng[1].text = "Продолжить игру";

Eng[2].text = "Настройки";

Eng[3].text = "Выход";

Eng[4].text = "ваш прогресс будет удален, продолжить?";

Eng[5].text = "Да";

Eng[6].text = "Нет";

Eng[7].text = "полный экран";

Eng[8].text = "Звуки";

Eng[9].text = "Музыка";

Eng[10].text = "Качество";

Eng[11].text = "Расширение";

Eng[12].text = "Язык";

Eng[13].text = "Назад";

Eng[14].text = "у вас нет сохранения";

Eng[15].text = "сейчас начнем!";

}

else

{

Eng[0].text = "New game";

Eng[1].text = "Load game";

Eng[2].text = "Settings";

Eng[3].text = "Exit";

Eng[4].text = "After this action, all unsaved progress will be lost. Continue?";

Eng[5].text = "Yes";

Eng[6].text = "No";

Eng[7].text = "FullScreen";

Eng[8].text = "Sounds";

Eng[9].text = "Music";

Eng[10].text = "Quality";

Eng[11].text = "Screen Resolution";

Eng[12].text = "Language";

Eng[13].text = "Back";

Eng[14].text = "you have no save";

Eng[15].text = "now brgin!";

}

}

[Serializable]

public class lng

{

public int Language;

}

}

**Приложение 17.**

**Листинг 7.16. Настройки.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Audio;

public class SettingsControl : MonoBehaviour

{

public AudioMixer Sounds;

public AudioMixer Music;

Resolution[] rsl;

List<string> resolutions;

public Dropdown dropdown;

public bool full = true;

public int language;

public float sounds, music;

public int q1;

public int r1;

public void Awake() [///создание](file:///\\\создание) всех доступных разрешений ///экрана

{

resolutions = new List<string>();

rsl = Screen.resolutions;

foreach (var i in rsl)

{

resolutions.Add(i.width + "x" + i.height);

}

resolutions.Reverse();

dropdown.ClearOptions();

dropdown.AddOptions(resolutions);

}

public void FullScreenToggle() ///Включение и выключение ///полного экрана

{

Screen.fullScreen = !Screen.fullScreen;

full = !full;

}

public void SoundsVolume(float sliderValue)///Настройка звука

{

sounds = sliderValue;

Sounds.SetFloat("Sounds", sliderValue);

}

public void MusicVolume(float slidervalue)///Настройка музыки

{

music = slidervalue;

Music.SetFloat("Music", slidervalue);

}

public void Quality(int q)///Настройка графики

{

q1 = q;

QualitySettings.SetQualityLevel(5 - q);

}

public void Resolution(int r)///Настройка разрешения экрана

{

r1 = r;

Screen.SetResolution(rsl[resolutions.Count - 1 - r].width, rsl[resolutions.Count - 1 - r].height, true);

}

public void Language(int l)///Выбор языка

{

language = l;

GameObject.Find("Main Camera").GetComponent<swap\_menu>().swap\_settings(l);

}

}

**Приложение 18.**

**Листинг 7.17. Настройки меню.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.Audio;

public class MenuControls : MonoBehaviour

{

public start set = new start();

public AudioMixer Sounds;

public AudioMixer Music;

public void Start()

{

set = JsonUtility.FromJson<start>(PlayerPrefs.GetString("SETSV"));

SoundsVolume(set.Sounds);

MusicVolume(set.Music);

}

public void PlayPressed()///Начало новой игры

{

PlayerPrefs.DeleteKey("SV");

SceneManager.LoadScene("first");

Screen.lockCursor = true;

Cursor.visible = false;

}

public void ExitPressed()///Выход из игры

{

Application.Quit();

}

public void SoundsVolume(float sliderValue)///Изменение звука

{

Sounds.SetFloat("Sounds", sliderValue);

}

public void MusicVolume(float slidervalue)///Изменение музыки

{

Music.SetFloat("Music", slidervalue);

}

public class start ///Класс для загрузки значения звука

{

public float Music;

public float Sounds;

}

}

**Приложение 19.**

**Листинг 7.18. Включение меню**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using System;

using System.IO;

public class EscMenu : MonoBehaviour {

public static bool isPaused = false;

public bool islocked = false;

public lengths sav = new lengths();

GameObject pauseMenu;

public GameObject pauseMenuRUS;

public GameObject pauseMenuENG;

public GameObject Player;

private void Awake()///Включение и выключение меню паузы

{

if (PlayerPrefs.HasKey("LENGSV"))

{

sav = JsonUtility.FromJson<lengths>(PlayerPrefs.GetString("LENGSV"));

Debug.Log(sav.Language);

if (sav.Language == 1)

pauseMenu = pauseMenuRUS;

else

pauseMenu = pauseMenuENG;

}

else

pauseMenu = pauseMenuENG;

}

void Update () {///Вывод паузы на выбранном языке

if (Input.GetKeyDown(KeyCode.Escape))

{

if (isPaused)

{

if (GameObject.Find("QuitWarRU") != null)

GameObject.Find("QuitWarRU").SetActive(false);

if (GameObject.Find("QuitWarENG") != null)

GameObject.Find("QuitWarENG").SetActive(false);

if (GameObject.Find("MAINWarRU") != null)

GameObject.Find("MAINWarRU").SetActive(false);

if (GameObject.Find("MAINWarENG") != null) GameObject.Find("MAINWarENG").SetActive(false);

Resume();

}

else

{

Pause();

}

}

}

public void Resume()///Продолжение игры

{

pauseMenu.SetActive(false);

SetCursorLocked();

Time.timeScale = 1;

isPaused = false;

Player.GetComponent<MainMove>().enabled = true;

}

public void Pause()///Пауза

{

Player.GetComponent<MainMove>().enabled = false;

pauseMenu.SetActive(true);

SetCursorLocked();

Time.timeScale = 0;

isPaused = true;

}

public void MainMenu()///Выход в главное меню

{

Resume();

SetCursorLocked();

SceneManager.LoadScene("SceneMainMenu");

}

public void Quit()///Выход из игры

{

Application.Quit();

}

public void SetCursorLocked()///Включение/Выключение курсора

{

Screen.lockCursor = islocked;

Cursor.visible = !islocked;

islocked = !islocked;

}

[Serializable]

public class lengths///Класс выбранного языка

{

public int Language;

}

}