Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Е. Н. Скуратенко
подпись
«____» _____ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03 Прикладная информатика

Разработка игры для изучения языка программирования Python

Руководитель _____ доцент, канд. техн.наук Е. Н. Скуратенко
подпись, дата

Выпускник _____ Р. А. Раводин
подпись, дата

Консультанты
по разделам:

Экономический _____ Е. Н. Скуратенко
подпись, дата

Нормоконтролер _____ В. И. Кокова
подпись, дата

Абакан 2021

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Е. Н. Скуратенко
подпись
«____» _____ 2021 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Раводину Родиону Алексеевичу

Группа ХБ 17-03

Направление 09.03.03 Прикладная информатика

Тема выпускной квалификационной работы: Разработка игры для изучения языка программирования Python

Утверждена приказом по институту № 222 от 08.04.2021 г.

Руководитель ВКР: Е.Н. Скуратенко, доцент, канд. техн. наук, ХТИ – филиал СФУ

Исходные данные для ВКР: заказ ХТИ – филиала СФУ.

Перечень разделов ВКР:

1. Исследование предметной области.
2. Описание разработки игры.
3. Расчёт затрат и оценка экономической эффективности реализации игры.

Перечень графического материала: нет

Руководитель ВКР _____ Е. Н. Скуратенко
подпись

Задание принял к исполнению _____ Р. А. Раводин
подпись

«08» апреля 2021 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме: «Разработка игры для изучения языка программирования Python», содержит 65 страниц , 56 рисунков, 5 таблиц, 4 формулы, 9 использованных источников.

ПРОЕКТ, UNITY, РАЗРАБОТКА, СИСТЕМА, ИЗУЧЕНИЕ, C#, 3D, ИГРА, ЭФФЕКТИВНОСТЬ, ЭКСПЛУАТАЦИЯ.

Объект ВКР: обучение студентов языку программирования Python.

Предмет ВКР: разработка игры для изучения языка программирования Python.

Цель работы: создание игры для изучения языка программирования Python.

Основные задачи ВКР:

– Проанализировать деятельность ХТИ – филиала СФУ.

– Определить требования заказчика.

– Выполнить сравнительный анализ игр, работающих в похожей сфере деятельности.

– Разработать игру согласно требованиям заказчика.

– Рассчитать затраты и экономическую выгоду проекта, определить риски, которым подвержен проект.

В результате реализации цели и задач ВКР разработана и создана игра для изучения языка программирования Python.

Разработанная игра будет способствовать повышению интереса студентов к языку Python.

Работа выполнена по заданию ХТИ — филиала СФУ.

SUMMARY

The theme of the graduation thesis is «Development of “Python” Language-learning Software Package Game». It contains 65 pages, 56 figures, 5 tables, 4 formulae, 9 reference items.

PROJECT, UNITY, DEVELOPMENT, SYSTEM, STUDY, C #, 3D, GAME, EFFICIENCY, EXPLOITATION.

The object of the graduation thesis: teaching students the Python programming language.

The subject of the graduation thesis: development of a game for learning the Python programming language.

The purpose of the graduation thesis: creating a game for learning the Python programming language.

Objectives of the graduation thesis:

- to analyze the workflow activity of KhTI - branch of Siberian Federal University;
- to determine customer requirements;
- to make comparative analysis of games in a similar field of activity;
- to develop a game according to customer requirements;
- to calculate costs and economic benefits of the project, to determine risks to which the project is exposed.

As a result of implementation of the purpose and objectives of the graduation thesis, a game for learning the Python programming language has been developed and designed.

The developed game will help to increase students' interest in the Python language.

The work has been carried out by order of KhTI - branch of Siberian Federal University.

English language supervisor _____ N.V. Chezybaeva signature,
date (full name)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Исходя из статистики, в мире треть населения являются пользователями игровых приложений. Из них 95% являются пользователями мобильных игр. В то же время в России большая часть населения использует компьютеры как платформу для игр. Увлечения, предлагаемые компьютерными играми, становятся все разнообразнее и затрагивают все большую аудиторию, в тоже время как компьютеры становятся все чаще элементом быта современного человека. Тема игр в обучении представляет теоретический и практический интересы, потому что с развитием и расширением многих индустрий в сфере развлечений и досуга, игры стали очень популярны и с помощью игр можно привлечь внимание не только школьников и студентов, но также взрослых.

Игры оказывают все большее влияние на быт пользователя и становятся проблемой для некоторых людей, вызывая зависимость. Часто такая зависимость оказывает негативное влияние на студентов и школьников, мешая их обучению. Поэтому некоторые разработчики игр делают такие игры, которые могут обучить пользователя тому или иному навыку.

Предпосылками для создания данной игры является усиление интереса учащихся к изучению языка Python.

С каждым годом все сложнее становится привлечь внимание учащихся к учебе. Игры занимают сейчас немалое место в жизни студентов, поэтому чтобы привлечь внимание студентов к обучению, будет создана игра для обучения студентов основам языка Python.

Цель выпускной квалификационной работы разработать игру, обучающую студента языку программирования Python.

Задачи:

- Проанализировать деятельность ХТИ – филиала СФУ.
- Определить требования заказчика.

- Выполнить сравнительный анализ игр, работающих в похожей сфере деятельности.
- Разработать игру согласно требованиям заказчика.
- Рассчитать затраты и экономическую выгоду проекта, определить риски, которым подвержен проект.

1 Исследование предметной области

1.1 Краткое описание деятельности ХТИ – филиала СФУ

Хакасский технический институт – филиал федерального государственного автономного образовательного учреждения высшего образования «Сибирский федеральный университет».

Миссией ХТИ — филиала СФУ является создание передовой образовательной, научно-исследовательской и инновационной инфраструктуры, продвижение новых знаний и технологий для решения задач социально-экономического развития Сибирского федерального округа, а также формирование кадрового потенциала — конкурентоспособных специалистов по приоритетным направлениям развития Сибири и Российской Федерации, соответствующих современным интеллектуальным требованиям и отвечающих мировым стандартам.

Институт осуществляет подготовку по очной, очно-заочной и заочной формам по 1 направлению специалитета и 6 направлениям бакалавриата, 2 направлению магистратуры. Ведется подготовка специалистов и бакалавров в следующих областях: электроэнергетика; машиностроение и материалообработка; строительство; экономика; транспорт; информатика.[1]

Направление подготовки 09.03.03 «Прикладная информатика» готовит специалистов для развивающейся отрасли информационных технологий. Важно внедрять новые знания и способы их усвоения для успешной подготовки специалистов данной сферы.

Одной из особо важных дисциплин является “Языки и системы программирования”. Студенты изучают несколько языков программирования.

Разработка игры для изучения языка позволит повысить интерес к изучению предмета.

1.2 Бизнес-процессы предметной области. Обоснование для разработки проекта

Современные курсы по обучению программированию на сегодняшний день немного устарели и требуют нововведений в целях повышения заинтересованности студентов в обучении программированию.

Студент выполняет задания на платформе Е-курсов, выполняет поставленные ему задания и изучает лекционный материал.

IDEF0 – это методология функционального моделирования, которая позволяет графически рассмотреть работу системы и выглядит как взаимосвязь функций. На рисунке 1 представлена диаграмма бизнес-процесса ХТИ–филиала СФУ.

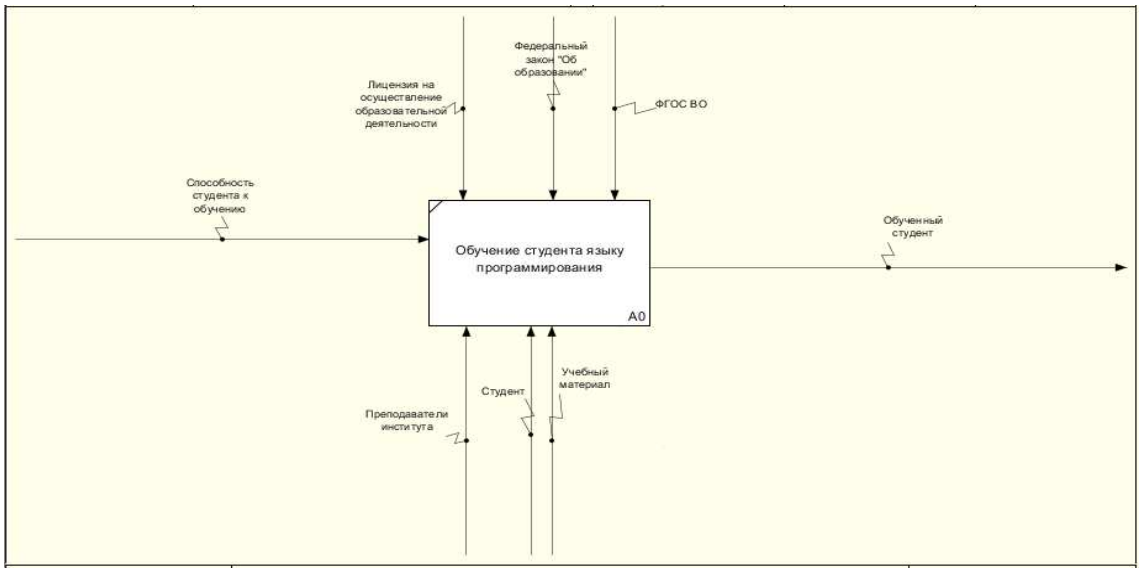


Рисунок 1 – IDEF0 работы ХТИ – филиала СФУ

Вход: Поступление студента на курс.

Выход: Студент, получивший знания.

Управление: Федеральный закон «Об образовании», ФГОС ВО, Лицензия на осуществление образовательной деятельности.

Механизмы: Преподаватели института, Кафедра “ПИМиЕД”, Учебный материал.

После необходимо создать декомпозицию функционального блока «Обучение студента», она представлена на рисунке 2.

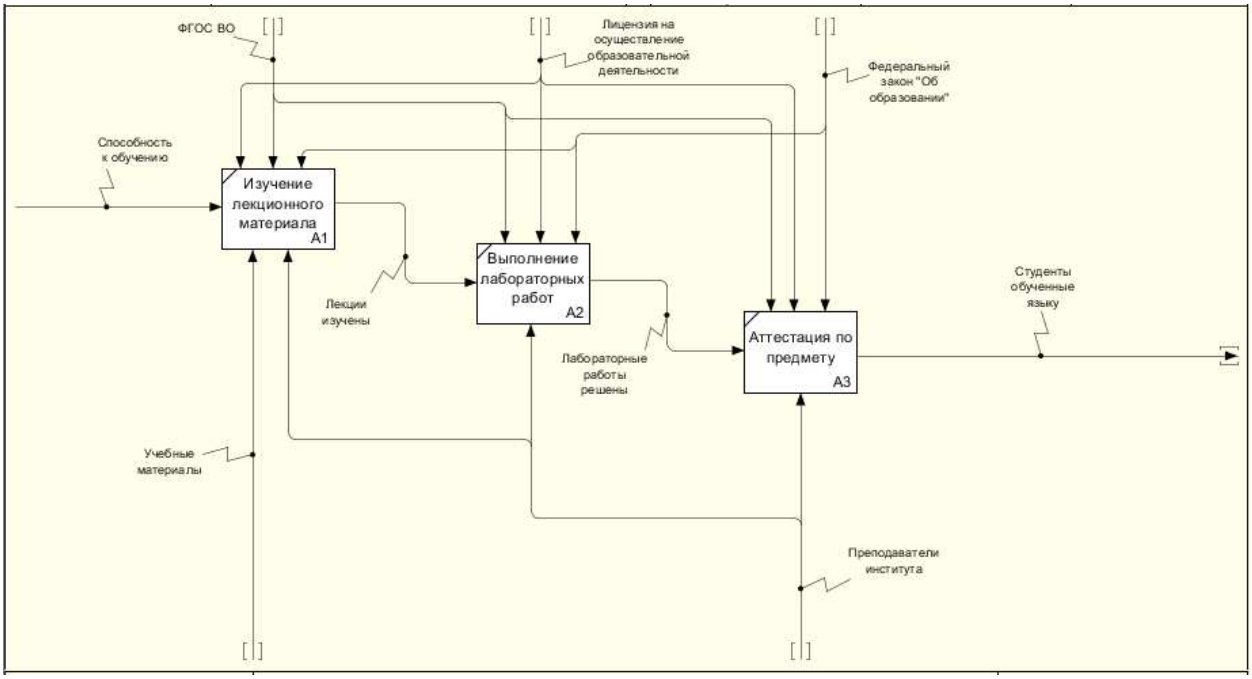


Рисунок 2 –IDEF0 декомпозиция работы ХТИ– филиала СФУ

Часто усвоение информации для студента сложная задача. Очень сложно сконцентрировать внимание на учебе во время студенческой жизни. Чтобы частично решить эту проблему преподаватели стараются разнообразить учебный материал и сделать его более понятным. Одной из форм подачиучебного материала являются обучающие игры.

1.3 Принципы функционирования реализуемой информационной системы

Разрабатываемая игра должна обучать студента основам языка программирования Python. Новому пользователю не должны быть доступны уровни кроме первого. Игра должна быть не требовательной к компьютерам ХТИ – филиала СФУ. Приемка программы должна осуществляться после периода опытной эксплуатации программы. Если игра работает корректно и устойчиво в течение 7-и календарных дней, период опытной эксплуатации считается завершенным. Исполнитель обязуется сопровождать программный продукт в течение 1-го месяца с начала периода эксплуатации, устранять все недоработки.

Применяемые в игре задания и методический материал должны соответствовать программе дисциплины «Языки и системы программирования», преподаваемой для студентов направления «Прикладная информатика».

Вышеперечисленные принципы — это основные принципы для работы игры, которые позволят привлечь интерес студентов к изучению языка программирования Python.

1.4 Анализ предметной области обучающих игр

В современном мире существует огромное количество игр различных жанров. Однако сегмент обучающих игр очень мал. Это часто происходит из-за незаинтересованности разработчиков в этом жанре игр. Основная масса игр этого жанра сделана для школьников и дошкольников. Однако обучающие игры по программированию все же есть.

Ниже приведены аналоги обучающих игр по программированию:

– Python Challenge. Все уровни проходятся с помощью простых и очень коротких скриптов. Большинство задач можно решить на любом языке программирования, но для некоторых из них всё-таки требуется знание Python. Вид игры представлен на рисунке 3.

Рисунок 3 – Обучающая игра Python Challenge

– JSdares. Экспериментальная образовательная среда программирования, где можно создавать собственные игры на JavaScript. Долгосрочная цель проекта - расширить возможности человечества, серьезно усовершенствовав программирование. Краткосрочная - сделать процесс

реализации и тестирования новых идей простым и удобным. Игра представлена на рисунке 4.

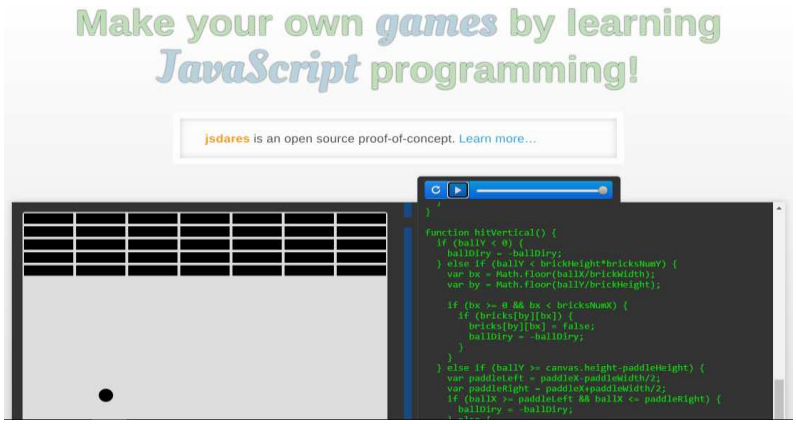


Рисунок 4 – Игра JSdares

– Robozzle. Игра поможет развить стратегическое мышление, а также умение искать наиболее эффективный способ решения задачи. Правила просты: в каждом уровне нужно запрограммировать робота, чтобы собрать звезды, используя такие команды, как "двигаться вперед", "повернуть направо", и "повторить". С каждым уровнем задачи становятся всё сложнее. Игра представлена на рисунке 5.

Рисунок 5 – Игра Robozzle

– CodeCombat. Эта браузерная RPG игра поможет в изучении JavaScript. Она состоит из блоков, которые, в свою очередь, разбиты на уровни. Результатом прохождения 37-ми уровней первого блока будет знакомство с синтаксисом, методами и прочими базовыми понятиями. С каждым уровнем сложность и время прохождения плавно возрастают. Игра представлена на рисунке 6.

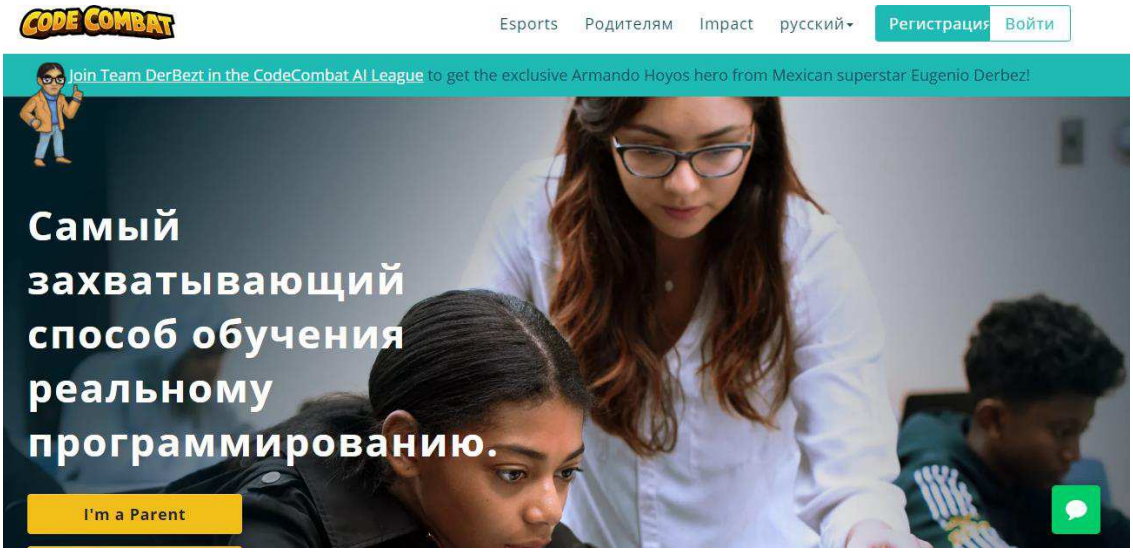


Рисунок 6 – Игра Codecombat

После рассмотрения аналогов обучающих игр был выбран метод визуальной демонстрации вариантов и выборе сделанных программ на языке Python в специальной консоли. Игра показывает лекционный материал через размещенные на уровнях записки. Также есть специальный компьютер показывающий при взаимодействии консоль и сейф который открывается при вводе правильных ответов. Внутри сейфа находится ключ, открывающий дверь, ведущую на следующий уровень или в главное меню в случае, если это последний уровень. Игрок, появившись на уровне должен решить задачу, представленную в правой части его экрана. Он может воспользоваться записками, изучить лекционный материал, решить задачу уровня, зайдя в консоль подсвеченного монитора компьютера и, после решения задачи, взять в

открывшемся сейфе ключ и открыть дверь, ведущую на следующий уровень или в главное меню. Нахождение записок не обязательно для прохождения уровня.

1.5 Требования к системе. Постановка цели и задач

В системе должен быть предусмотрен следующий функционал:

1. Выбор уровня.
2. Инструкция к управлению.
3. Выход из игры.
4. Начало новой игры.

При создании проекта необходимо опираться на следующие нормативные документы:

- Нотация описания бизнес-процессов IDEF0.
- ISO / IEC 12207: 1995-08-01.
- ГОСТ Р ИСО/МЭК 12207-99 «Информационная технология. Процессы жизненного цикла программных средств».

Гост 19.201-78 Единая система программной документации (ЕСПД). Техническое задание. Требования к содержанию и оформлению. Цель выпускной квалификационной работы

Цель: разработать игру, обучающую основам языка Python.

Задачи:

- Проанализировать деятельность ХТИ - филиала СФУ.
- Определить требования заказчика.
- Выполнить сравнительный анализ игр, работающих в похожей сфере деятельности.
- Разработать игру согласно требованиям заказчика.

- Рассчитать затраты и экономическую выгоду проекта, определить риски, которым подвержен проект.

1.6 Выбор среды разработки

В качестве среды среди рассмотренных вариантов среды разработки был выбран игровой движок Unity. В ходе сравнения с другими движками он был выбран как самый подходящий для проекта. В качестве сравнения были приведены такие игровые движки как:

- Unity.
- Unreal Engine 4
- GameMakerStudio 2
- Hero Engine

Сравнение данных игровых движков представлено в таблице 1.

Таблица 1 – Таблица сравнения игровых движков

	Unity	Unreal Engine 4	GameMakerStudio 2	Hero Engine
Доступность движка	Бесплатный	Платный, но недорогой	Бесплатный в создании, но требует денег для реализации	Платный, очень дорогой
Набор инструментов	Большой	Огромный	Большой	Огромный
Порог вхождения	Низкий	Низкий	Низкий	Высокий

Продолжение таблицы 1

Сообщество (доступность обучения движку)	Огромное	Огромное	Среднее	Небольшое
Язык программирования (среда)	C#	C++	GML	C++,C#,HSL

1.7 Выводы по разделу «Исследование предметной области»

В данном разделе была проанализирована основная деятельность Хакасского технического института – филиала СФУ. Были определены предпосылки для создания игры, определены основные требования к системе.

Был проведен анализ предметной области и, исходя из анализа, было определено, что обучающие игры — это очень редкий жанр игр. Также были определены общие принципы построения игры.

2 Описание разработки игры

2.1 Планирование инфраструктуры. Выбор оборудования и программного обеспечения

Для разработки игры потребуется компьютер и игровой движок Unity. Выбор комплектующих компьютера представлен в экономическом разделе.

Unity — межплатформеннаясреда разработкикомпьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями[3].

2.2 Планирование и определение функций. Декомпозиция на подсистемы

Диаграмма вариантов использования.

Вариант использования описывает, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату.

Варианты использования являются описаниями типичных взаимодействий между пользователями системы и самой системой. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать [4].

Модель новой организации процесса TO-BE

Для наглядной демонстрация работы игры была создана диаграмма IDEF0, которая представлена на рисунке 7.

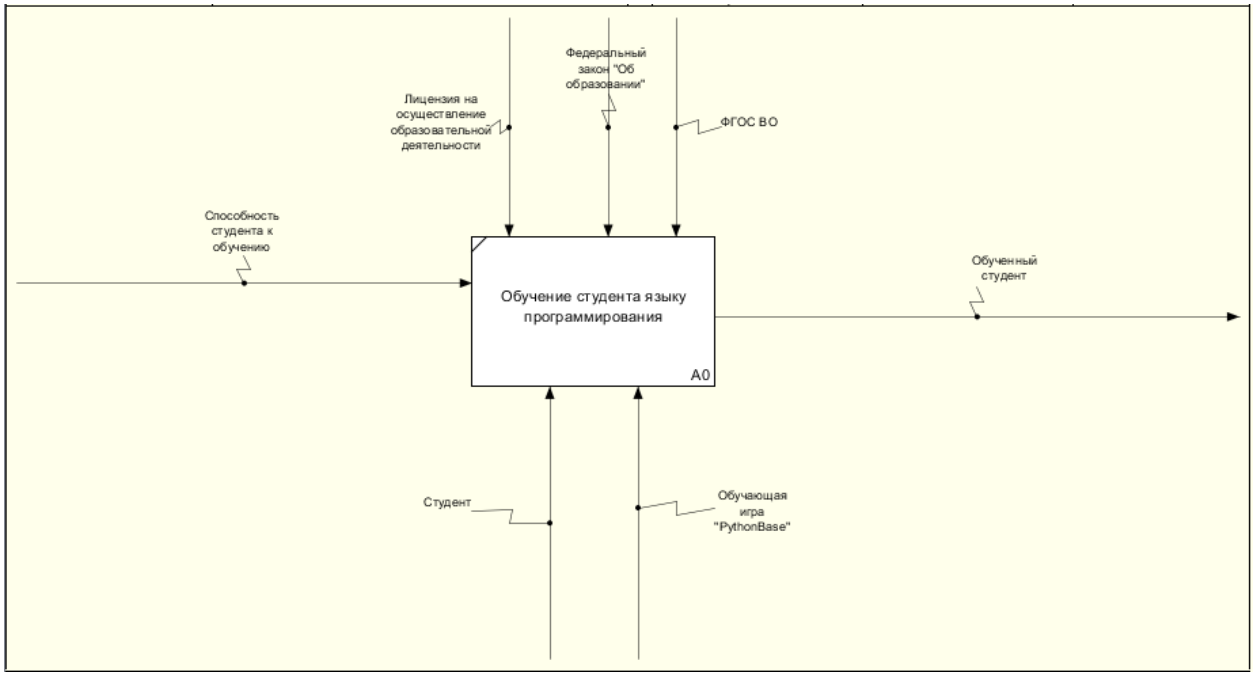


Рисунок 7 – IDEF0 работы игры после внедрения

Вход: Способность студента к обучению.

Выход: Студент, получивший знания.

Управление: Федеральный закон «Об образовании», ФГОС ВО, Лицензия на осуществление образовательной деятельности.

Механизмы: Обучающая игра “PythonBase”, Студент.

После этого необходимо создать декомпозицию функционального блока “Обучение студента языку программирования”. Представлена на рисунке 8.

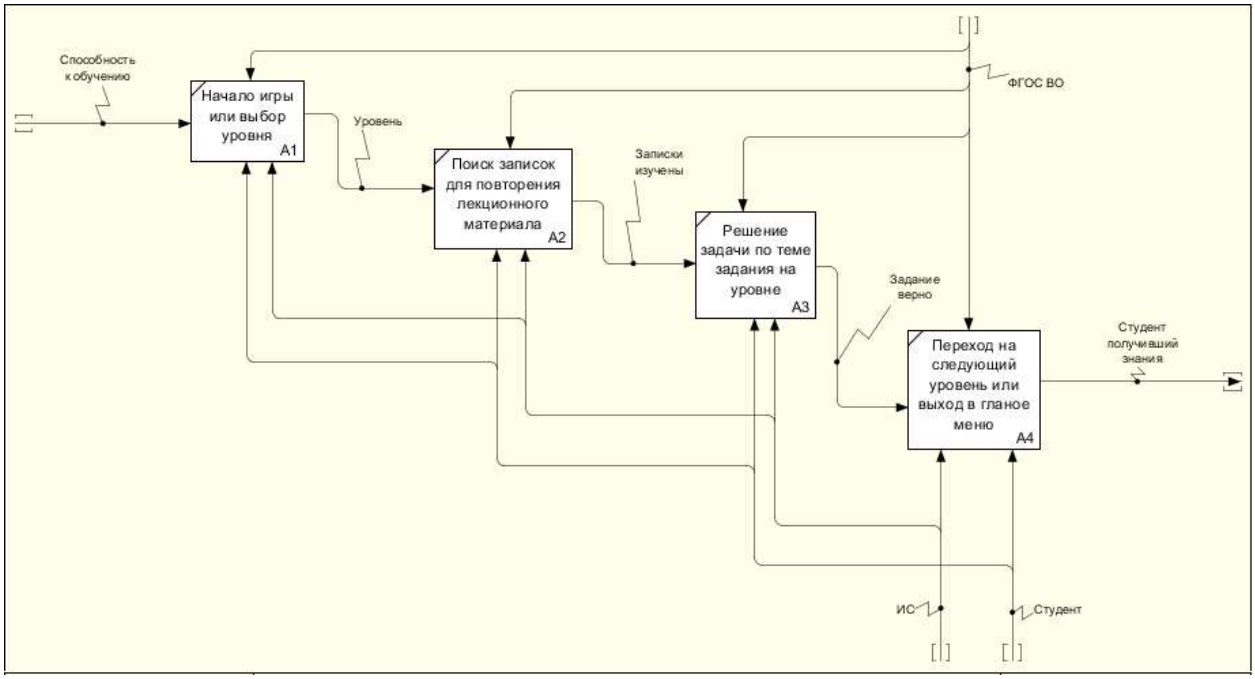


Рисунок 8 – IDEF0 декомпозиция работы игры после внедрения

Диаграмма потоков данных.

DFD — общепринятое сокращение от англ. data flow diagrams — диаграммы потоков данных. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ [5].

Диаграмма потока данных понадобится для наглядного показа, как данные будут двигаться внутри игры.

В диаграммы потока данных входят следующие элементы:

- внешняя сущность;
- процесс;
- хранилище данных;
- поток данных.

Внешняя сущность – это любые объекты, которые не входят в саму систему, но являются для нее источником информации либо получателями какой-либо информации из системы после обработки данных. Это может быть

человек, внешняя система, какие-либо носители информации и хранилища данных.

Процесс – это функция или последовательность действий, которые нужно предпринять, чтобы данные были обработаны [5].

Хранилище данных – это различные базы данных, предназначенные для хранения данных.

Поток данных – это стрелки, которые показывают, какая информация входит, а какая исходит из того или иного блока на диаграмме.

На рисунке 9 представлена диаграмма потока данных сайта для ХТИ - филиала СФУ.

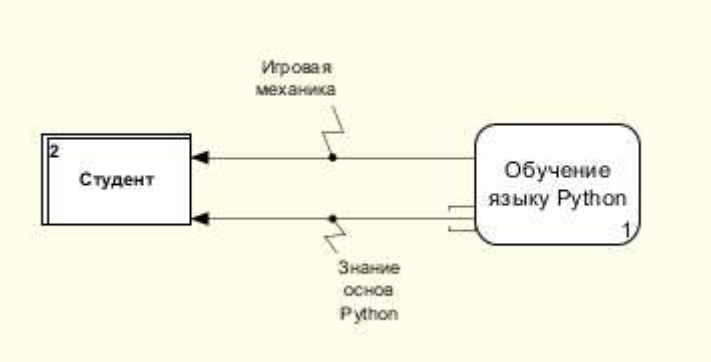


Рисунок 9 – DFD обучающей игры для ХТИ - филиала СФУ

На рисунке 10 показана декомпозиция диаграммы потока данных разрабатываемой игры.

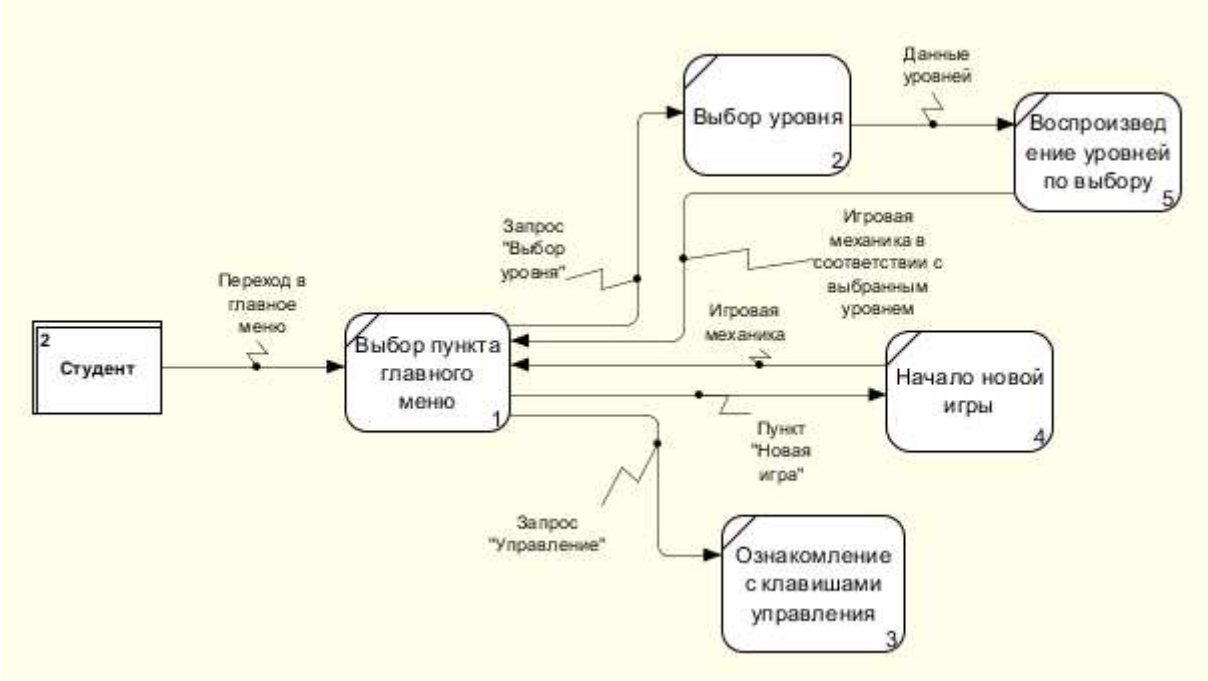


Рисунок 10 – Декомпозиция DFD игры для ХТИ – филиала СФУ

2.3 Построение модели работы пользователя с информационной системой в нотации IDEF3

IDEF3 (англ. Integrated DEFinition for Process Description Capture Method) — методология моделирования и стандарт документирования процессов, происходящих в системе. Метод документирования технологических процессов представляет собой механизм документирования и сбора информации о процессах [6].

Методология IDEF3 является одним из стандартов семейства IDEF и довольно широко используется при декомпозиции моделей IDEF0 для моделирования процессов более низкого уровня, поскольку с его помощью можно смоделировать технологические процессы, происходящие на предприятии, то есть описать возможные сценарии реализации процессов, в рамках которых происходит последовательное изменение свойств объекта. Данная методология позволяет показывать возможные разветвления в процессе [7].

На рисунке 11 представлен процесс использования игры, когда пользователь только включил ее.

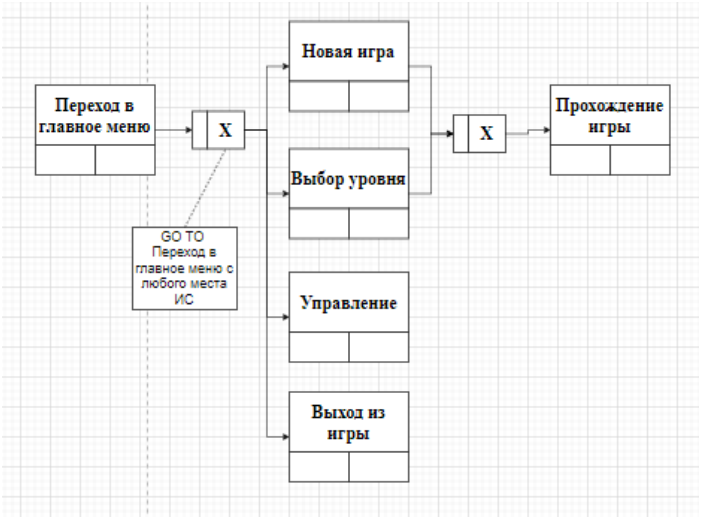


Рисунок 11 – IDF3 процесса использования игры

2.4 Описание разработки игры

В первую очередь для разработки необходимо подготовить все программное обеспечение, требуемое для корректной работы системы. Для разработки игры потребуется игровой движок Unity и любой браузер. За основной браузер будет взят Chrome. Его установка не потребуется, т.к. он уже установлен на компьютере разработчика. Помимо этого, потребуется скачать и установить Unity.

После инсталляции необходимо произвести первый запуск программы, чтобы продолжить дальнейшую настройку, а также просто убедиться в том, что она функционирует нормально. Для этого потребуется всего лишь выбрать или создать новый проект (рисунок 12).

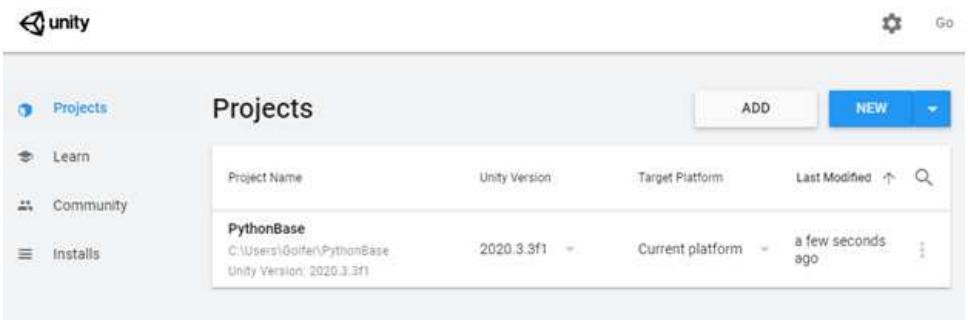


Рисунок 12– Окно создания или выбора проекта Unity

Дальше появится панель инструментов игрового движка (рисунок 13).

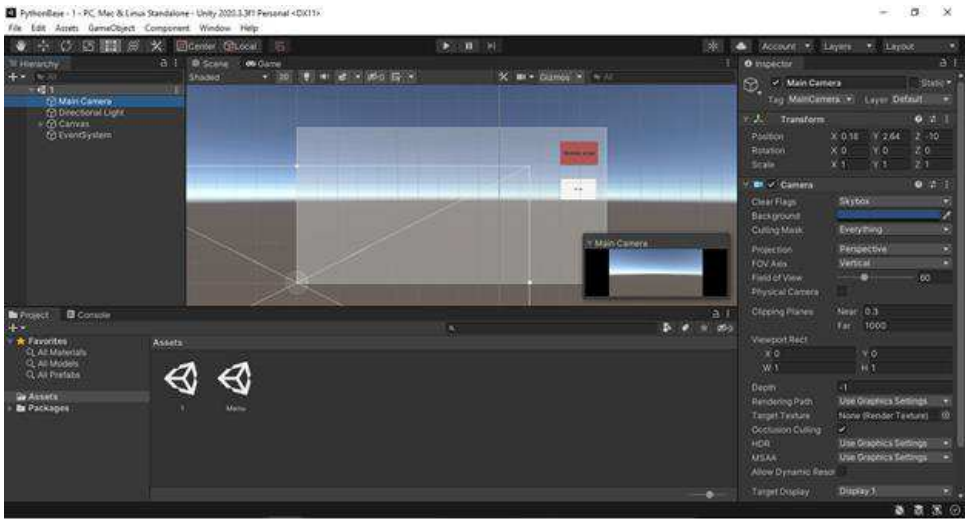


Рисунок 13– Панель инструментов движка

На этом установка программного обеспечения завершена. После установки создано главного меню игры. В нем были сделаны вкладки:

- Новая игра.
- Выбор уровня.
- Управление.

- Выход из игры.

Визуальное представление главного меню показано на рисунке 14.

Рисунок 14–Главное меню

Дальше были созданы вкладки “Выбор уровня” и “Настройки”.

В меню выбора уровня должны быть вкладки доступных уровней и вкладка выхода в главное меню. Вкладка представлена на рисунке 15.

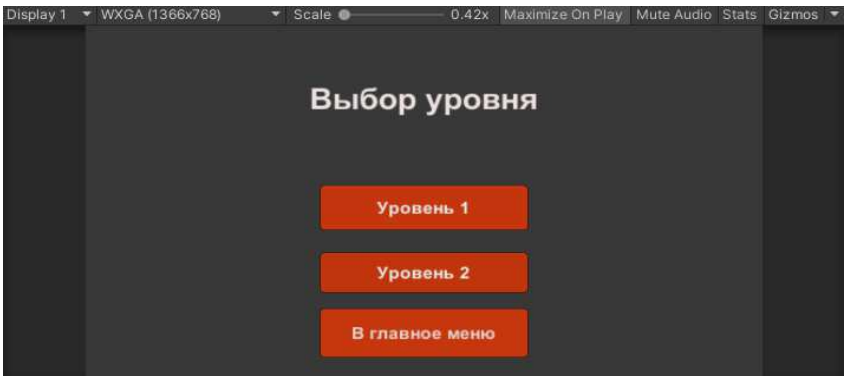


Рисунок 15– Вкладка “Выбор уровня”

Во вкладке “Управление” должны быть указаны все заданные клавиши управления персонажем игры. Также должна быть вкладка для выхода в главное меню. Она будет называться “Назад”. Общий вид вкладки настроек представлен на рисунке 16.

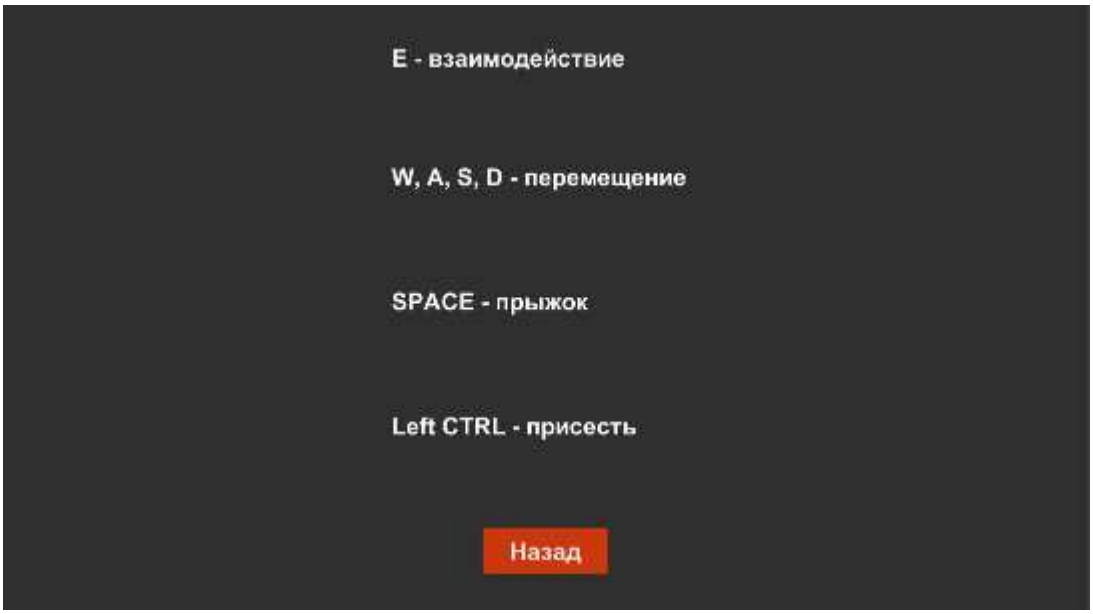


Рисунок 16– Вкладка “Управление”

Класс или скрипт — это программный код, который содержит в себе описание некоего игрового объекта.

Таким образом был создан скрипт в виде класса с названием “Menu”. Класс отвечает за работу вкладок “Новая игра” и “Выход из игры”. Представлен на рисунке 17.

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

public class Menu : MonoBehaviour
{

 public void PlayPressed()
  {
     SceneManager.LoadScene("Game");
  }
 public void ExitPressed()
  {
     Application.Quit();
  }
 Debug.Log("Exit pressed!")
}
```

Рисунок 17– Класс “Menu”

Для других вкладок код необязателен, для переключения между вкладками достаточно создать их сцены и указать нужную вкладку в списке действий кнопки. Для кнопки “Управление” в панели ее взаимодействия была указана функция запуска сцены “Управление” GameObject.SetActive. Как это выглядит показано на рисунке 18.

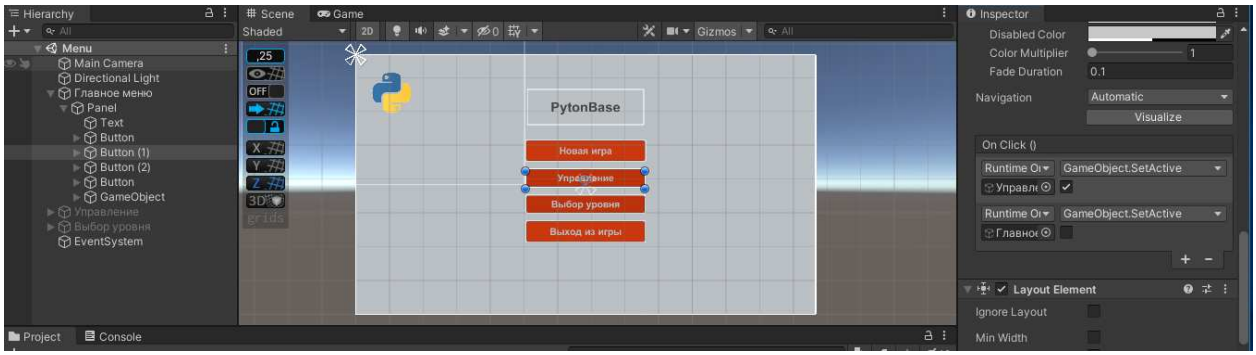


Рисунок 18– Работа вкладки “Управление”

Далее был создан префаб комнаты. Префаб — это объект или группа объектов, объединенных в одну модель для упрощения создания копий и

передвижения. Для создания префаба была использована библиотека ассетов Unity (Unity Asset Store). В создании комнаты были выбраны ассеты офиса, а именно стены, двери и рабочие столы с компьютерами. Как выглядит страница ассетов Unity показано на рисунке 19.

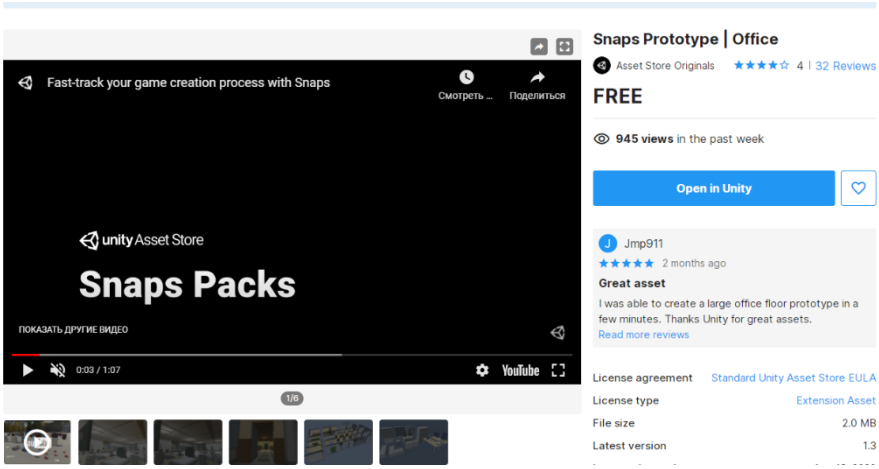


Рисунок 19–Страница ассета “Office”

Была спроектирована комната на основе префабов стен, дверных проемов, дверей, стен с окнами, плитки пола и потолка. Конечный вид комнаты представлен на рисунках 20 и 21.

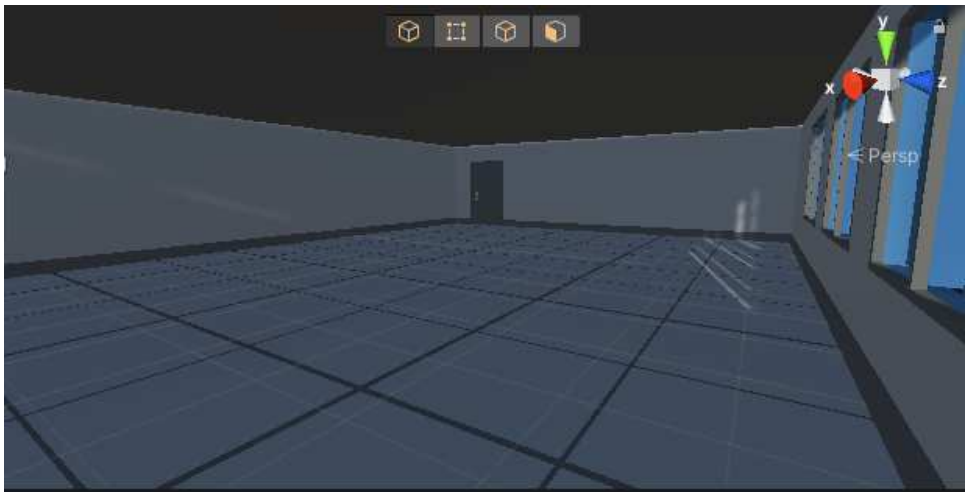


Рисунок 20–Вид комнаты (первый ракурс)

Рисунок 21–Вид комнаты (второй ракурс)

Также был использован префаб персонажа с видом от первого лица. Он был взят из стандартных ассетов Unity. Персонаж был размещен внутри комнаты. Модель для персонажа использовалась в виде прозрачной, вытянутой по вертикали, сферы с прикрепленной к верхней вершине камерой. Вид от лица персонажа, а также взгляд на модель представлены на рисунках 22 и 23.

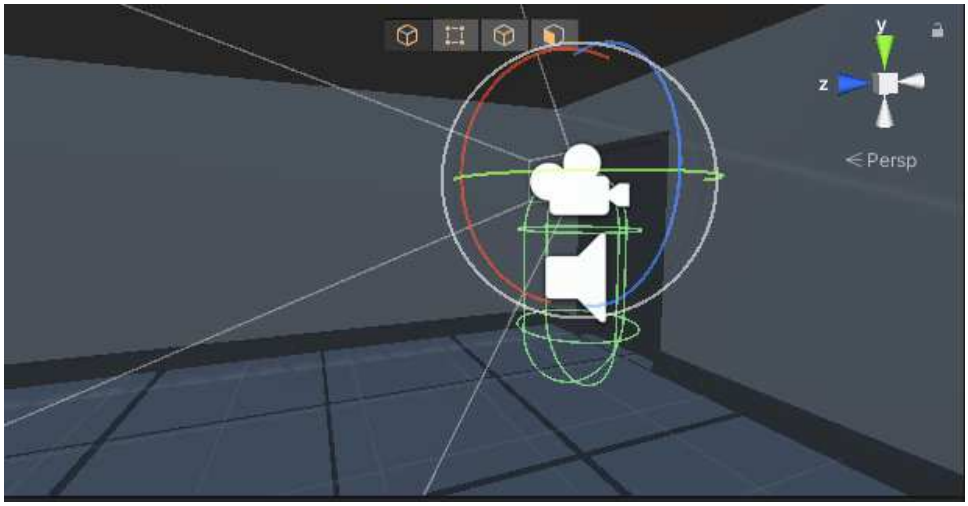


Рисунок 22–Взгляд на модель персонажа

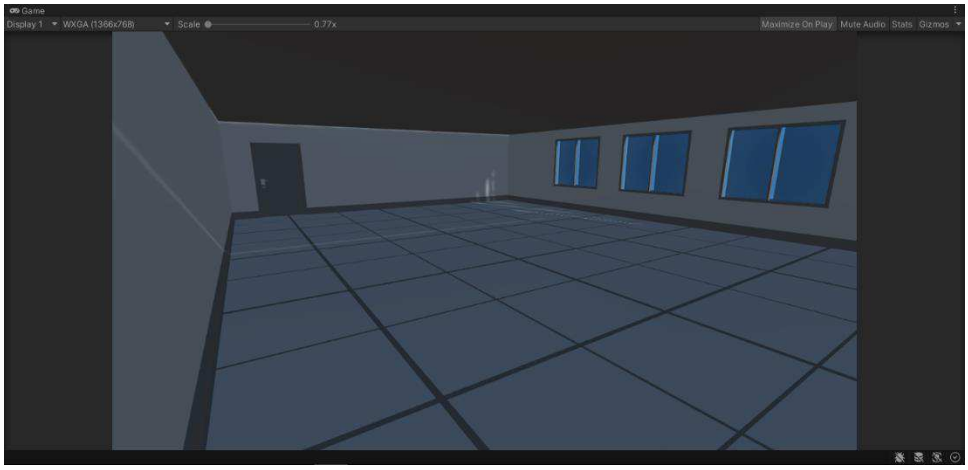


Рисунок 23–Взгляд на комнату от лица “Персонажа”

Для создания обстановки комнаты были использованы префабы компьютерного стола, стола преподавателя, а также модели сейфа, доски, потолочной лампы, а также комнатного растения. Все модели представлены на рисунках 24-29.

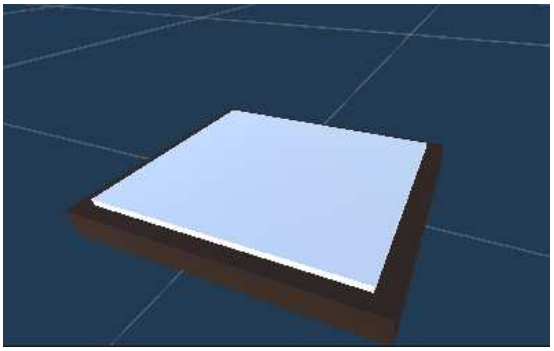


Рисунок 24– Модель доски

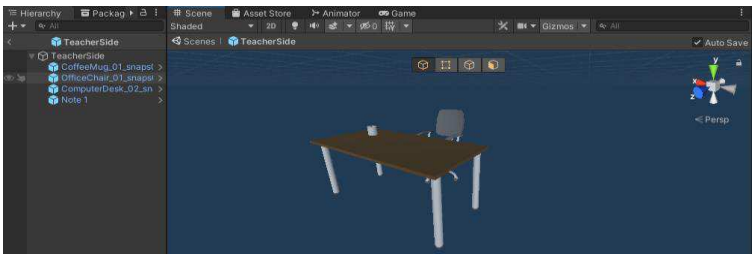


Рисунок 25–Префаб стола преподавателя

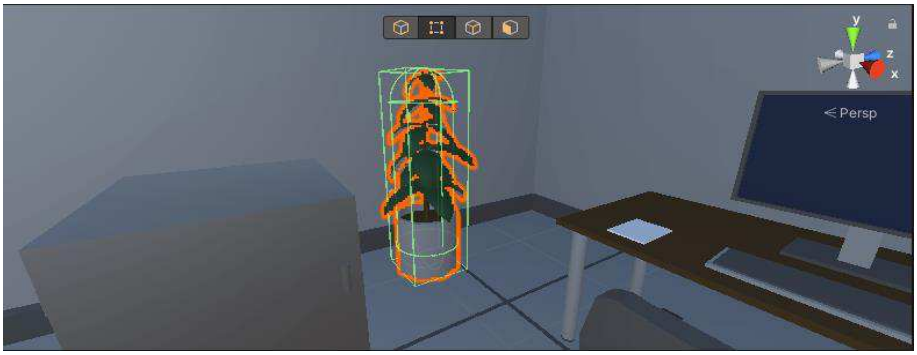


Рисунок 26–Модель растения

Рисунок 27–Префаб компьютерного стола

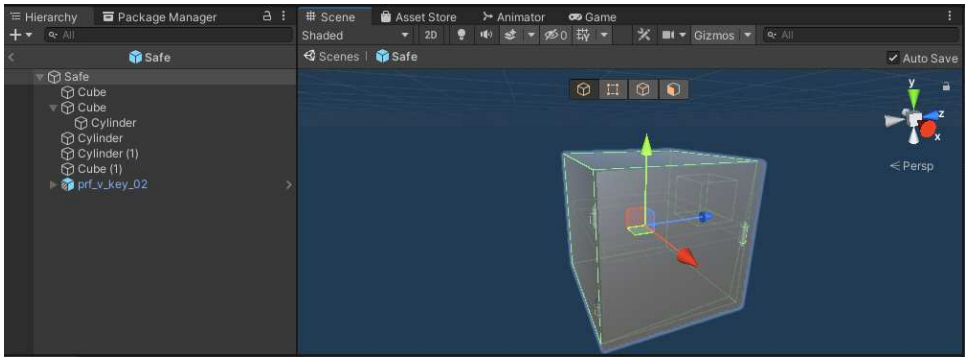


Рисунок 28–Модель сейфа

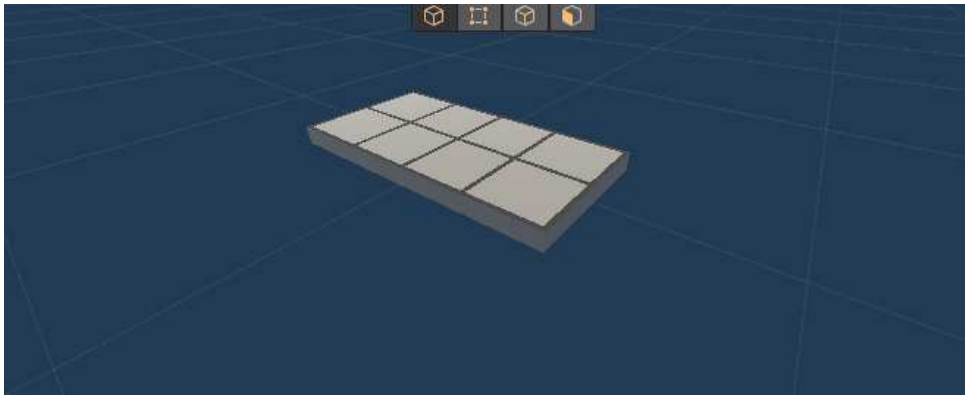


Рисунок 29–Модель лампы

Далее все модели и префабы были расставлены по комнате. Конечный результат представлен на рисунке 30.

Рисунок 30–Префаб комнаты

Далее была создана и продублирована модель записки. Записки были расставлены по комнате в различных местах. Для их расстановки некоторые модели были перемещены. Вид комнаты с записками, модель записки, а также пример расположения представлены на рисунках 31, 32 и 33.

Рисунок 31–Комната с записками

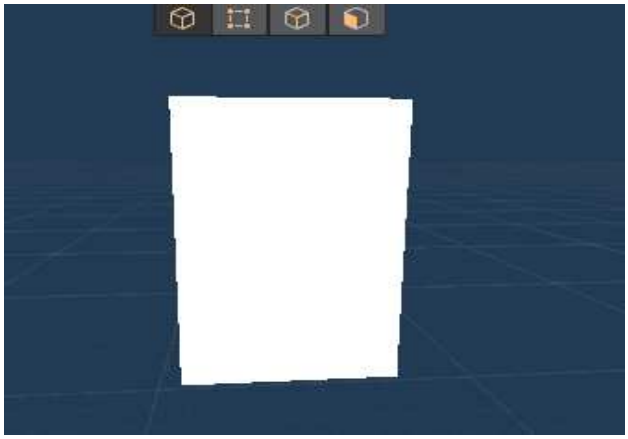


Рисунок 32–Модель записки

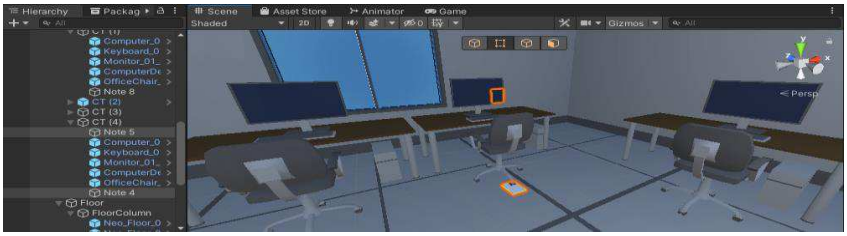


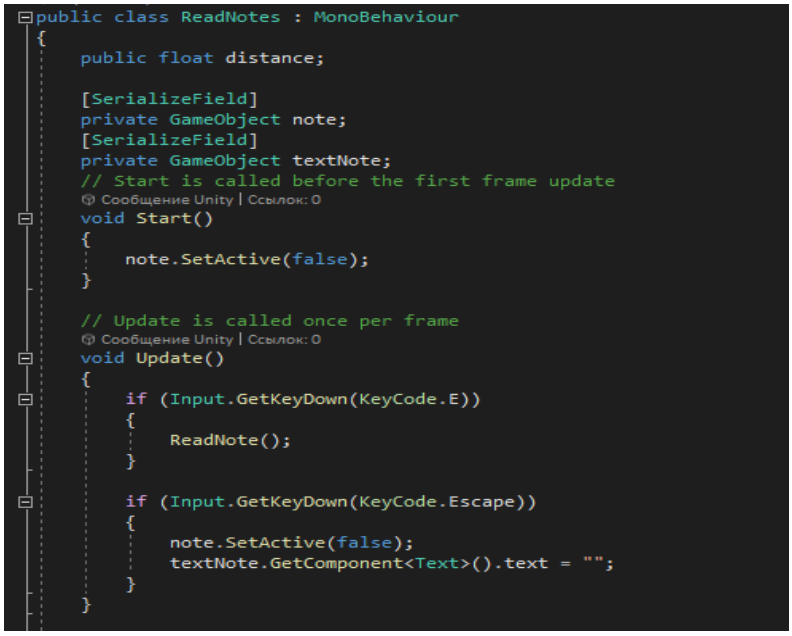
Рисунок 33–Пример расположения записок

Были созданы два скрипта для записок. Один отвечает за редактирование текстовой информации внутри записки под названием “Note”. Второй отвечает за отображение записки при нажатии на определенную клавишу. Его название - “ReadNotes”. Код обоих скриптов представлен на рисунках 34 и 35.

```
public class Note : MonoBehaviour {
    [TextArea(3, 5)]
    public string TextNote;

}
```

Рисунок 34–Скрипт “Note”



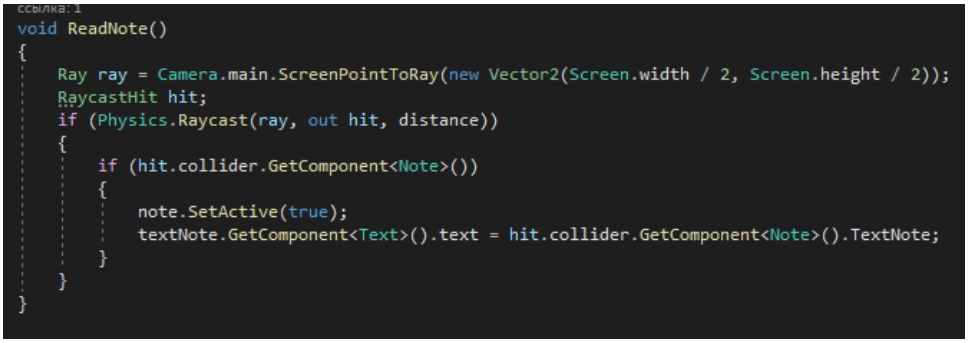
```
public class ReadNotes : MonoBehaviour
{
    public float distance;

    [SerializeField]
    private GameObject note;
    [SerializeField]
    private GameObject textNote;
    // Start is called before the first frame update
    Сообщение Unity | Ссылок: 0
    void Start()
    {
        note.SetActive(false);
    }

    // Update is called once per frame
    Сообщение Unity | Ссылок: 0
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            ReadNote();
        }

        if (Input.GetKeyDown(KeyCode.Escape))
        {
            note.SetActive(false);
            textNote.GetComponent<Text>().text = "";
        }
    }
```

Рисунок 35 – Скрипт “ReadNotes”, лист 1



```
ссылка: 1
void ReadNote()
{
    Ray ray = Camera.main.ScreenPointToRay(new Vector2(Screen.width / 2, Screen.height / 2));
    RaycastHit hit;
    if (Physics.Raycast(ray, out hit, distance))
    {
        if (hit.collider.GetComponent<Note>())
        {
            note.SetActive(true);
            textNote.GetComponent<Text>().text = hit.collider.GetComponent<Note>().TextNote;
        }
    }
}
```

Рисунок 35, лист 2

Чтобы код скриптов работал правильно, а также чтобы редактировать информацию внутри Unity, в начале у обоих кодов были созданы специальные поля, отображающиеся для разработчика в Unity. Их вид представлен на рисунках 36 и 37.

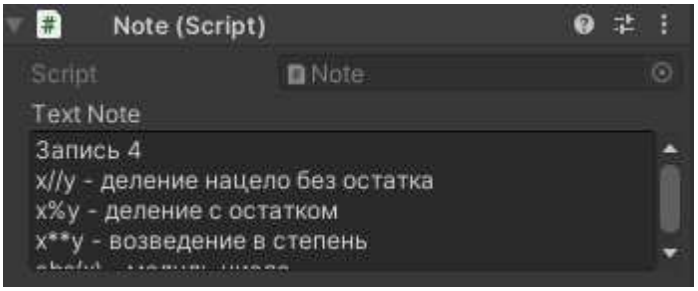


Рисунок 36–Панель редактирования текста записки

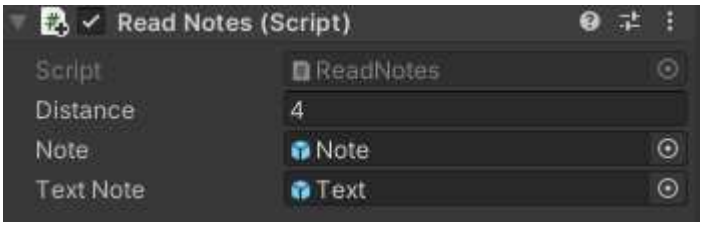


Рисунок 37 – Панель указания объектов для скрипта

Чтобы записка отображалась был создан объект “Canvas”, а внутри объекта было создано изображение с помещенным внутрь его пустым текстом. Текст пустой в виду изменения его значения через скрипт записки. Общий вид созданной записки представлен на рисунке 38.

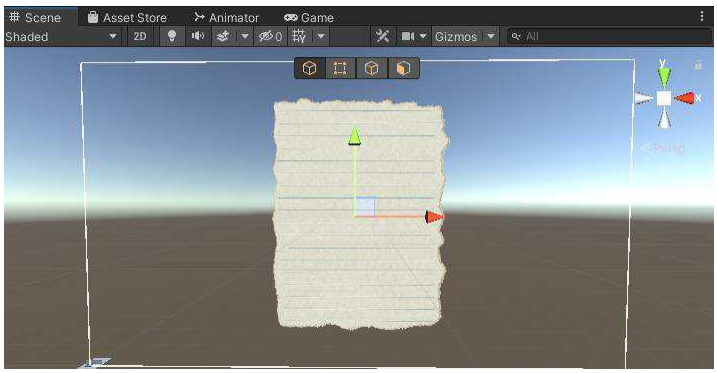


Рисунок 38– Окно записки

Для открытия двери в комнате для нее была создана анимация, а также два скрипта “Door” и “DoorController” для управления ей. Модель двери представлена на рисунке 39.

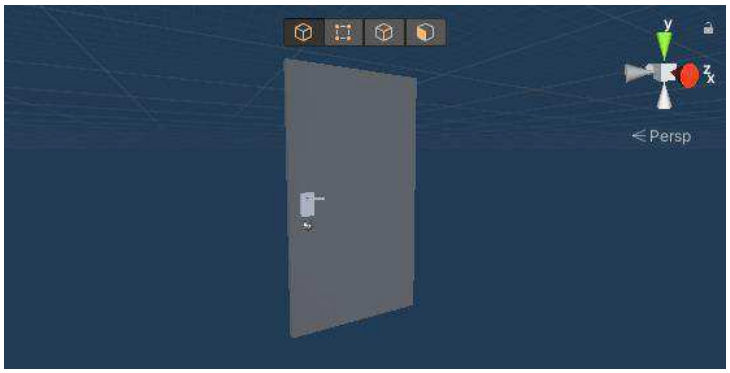


Рисунок 39–Модель двери

Анимация двери сделана из двух ее состояний - открытого и закрытого. Также для анимации использовался поворот от ее правой стороны. Точка поворота сделана через Pivot. Примерный вид окна анимации двери показан на рисунке 40.

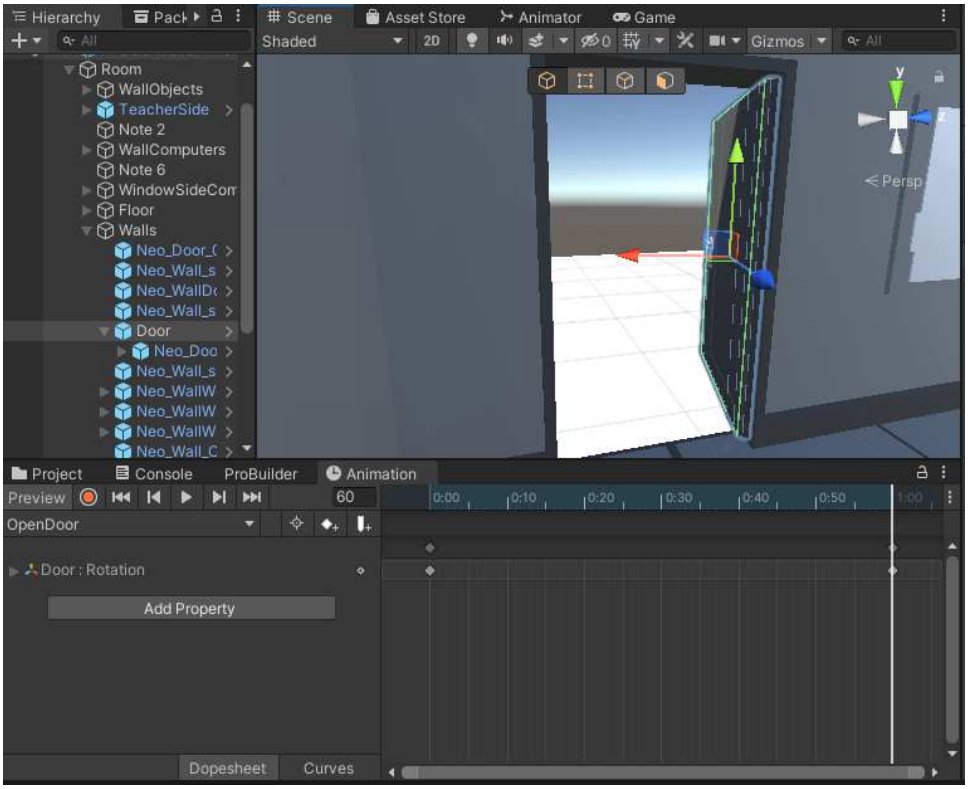


Рисунок 40–Окно анимации двери

Дверь открывается при взаимодействии персонажа с ней через определенную кнопку. Для того чтобы персонаж мог отличать дверь от остальных объектов комнаты для скрипта “DoorController” был написан метод “Raycast”.

Метод “Raycast создает луч задаваемой длины от камеры персонажа. Когда луч сталкивается с объектами, он меняет свой цвет в зависимости от типа объекта, с которым столкнулся луч.

Для двери был написан скрипт “Door”. Он определяет тип для двери. Оба скрипта представлены на рисунках 41 и 42.

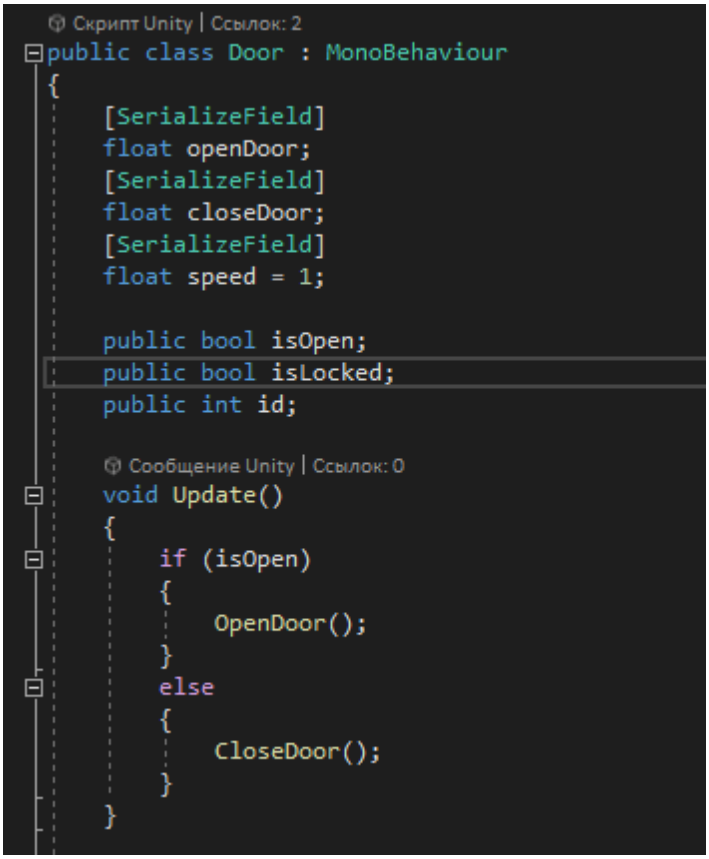


Рисунок 41– Скрипт ”Door”, лист 1

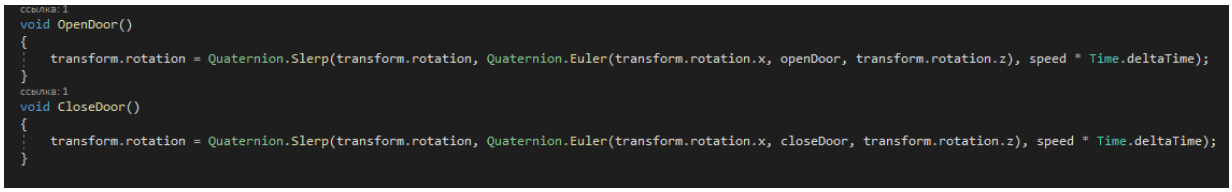


Рисунок 41, лист 2

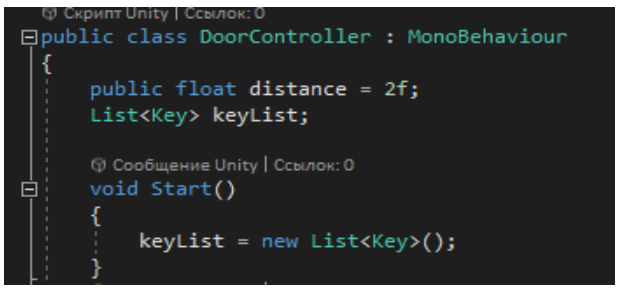


Рисунок 42– Скрипт “DoorController”, лист 1

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.E))
    {
        Ray ray = Camera.main.ScreenPointToRay(new Vector2(Screen.width / 2, Screen.height / 2));
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit, distance))
        {
            if (hit.collider.tag == "Door")
            {
                Door door = hit.collider.GetComponent<Door>();
                if (door.isLocked)
                {
                    for (int i = 0; i < keyList.Count; i++)
                    {
                        if (keyList[i].id == door.id)
                        {
                            door.isLocked = false;
                            door.isOpen = !door.isOpen;
                        }
                        else
                        {
                            Debug.Log("Нет ключа!");
                        }
                    }
                }
                else
                {
                    door.isOpen = !door.isOpen;
                }

            }
```

Рисунок 42, лист 2

Для обоих скриптов также были созданы их панели. Для скрипта “Door” панель определяет градус и скорость поворота двери, id, а также статус двери. Панель “DoorController” определяет дистанцию луча для функции “Raycast”. Скрипты показаны на рисунках 43 и 44.

Рисунок 43–Панель скрипта “DoorController”

Рисунок 44–Панель скрипта “Door”

Чтобы дверь была закрыта и ее нельзя было открыть без решения задачи был создан ключ. Модель ключа представлена на рисунке 45.

Рисунок 45– Модель ключа

Для работы ключа был создан скрипт “Key”. Его панель дает ему id. Код скрипта представлен на рисунке 46.

```
if (hit.collider.GetComponent<Key>())
{
    Key key = hit.collider.GetComponent<Key>();
    keyList.Add(key);
    Debug.Log(keyList.Count);
    Destroy(key.gameObject);
}
```

Рисунок 46–Код скрипта “Key”

Чтобы спрятать ключ заранее был создан сейф. Для открытия сейфа была создана анимация, подобная анимации двери, а также написан скрипт “Door0” для работы этой анимации. Код представлен на рисунке 47.

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

Скрипт Unity | Ссылок: 3
public class Door0 : MonoBehaviour
{
    private bool safeOpen;
    [SerializeField] private Animator _animator;
    public bool isLocked;

    Сообщение Unity | Ссылок: 0
    private void Start()
    {

    }

    Ссылок: 2
    public void Open()
    {
        _animator.SetBool("SafeOpened", safeOpen);
        safeOpen = !safeOpen;
    }
}
```

Рисунок 47–Скрипт Door0

Чтобы игрок решил задачу, в игре была создана консоль. Она показана на рисунке 48.

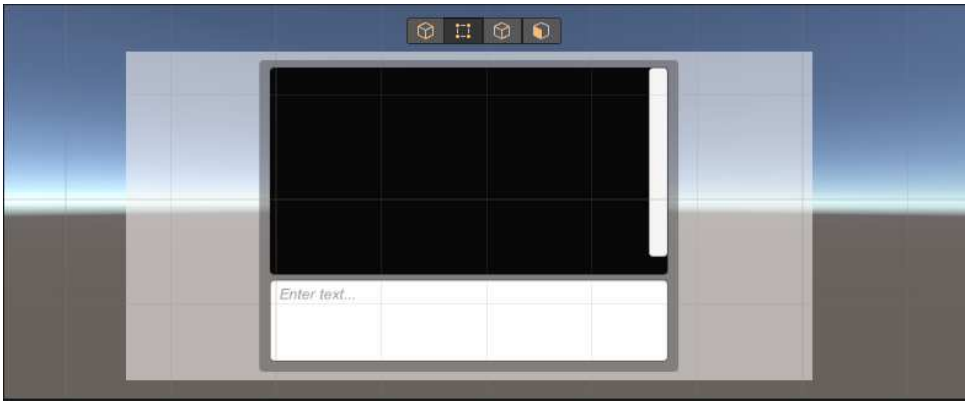


Рисунок 48– Консоль

Чтобы отобразить варианты ответов решения задачи были созданы 3 панели с находящимися внутри них текстами. Как они выглядят представлено на рисунке 49.

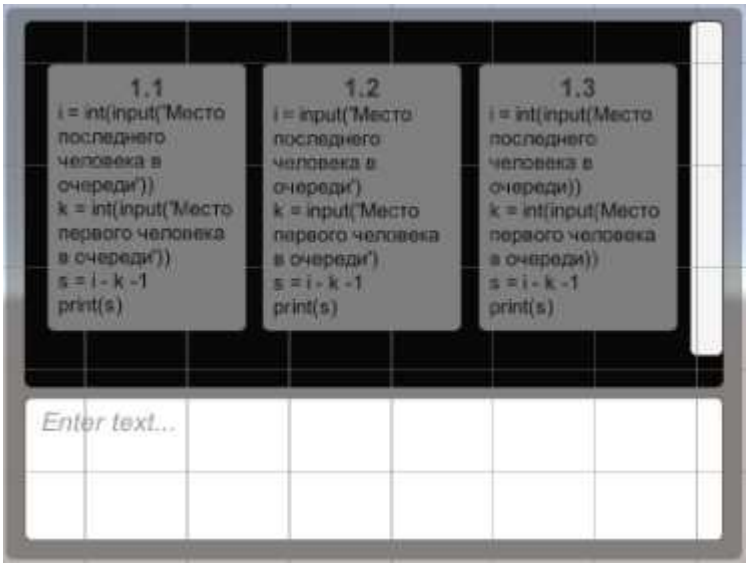
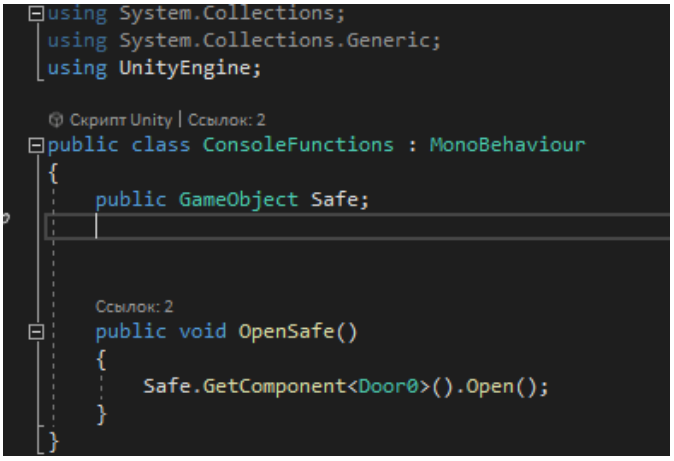


Рисунок 49 – Панели с вариантами ответов

Для работы консоли созданы скрипты “ConsoleManger” и “ConsoleFuctions”. Первая отвечает за создание и редактирование вариантов решения задачи в консоли, а вторая за их функционирование. Код для консоли первого уровня и панели скрипта представлены на рисунках 50-52.



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

Скрипт Unity | Ссылок: 2
public class ConsoleFunctions : MonoBehaviour
{
    public GameObject Safe;


    Ссылок: 2
    public void OpenSafe()
    {
        Safe.GetComponent<Door0>().Open();
    }
}
```

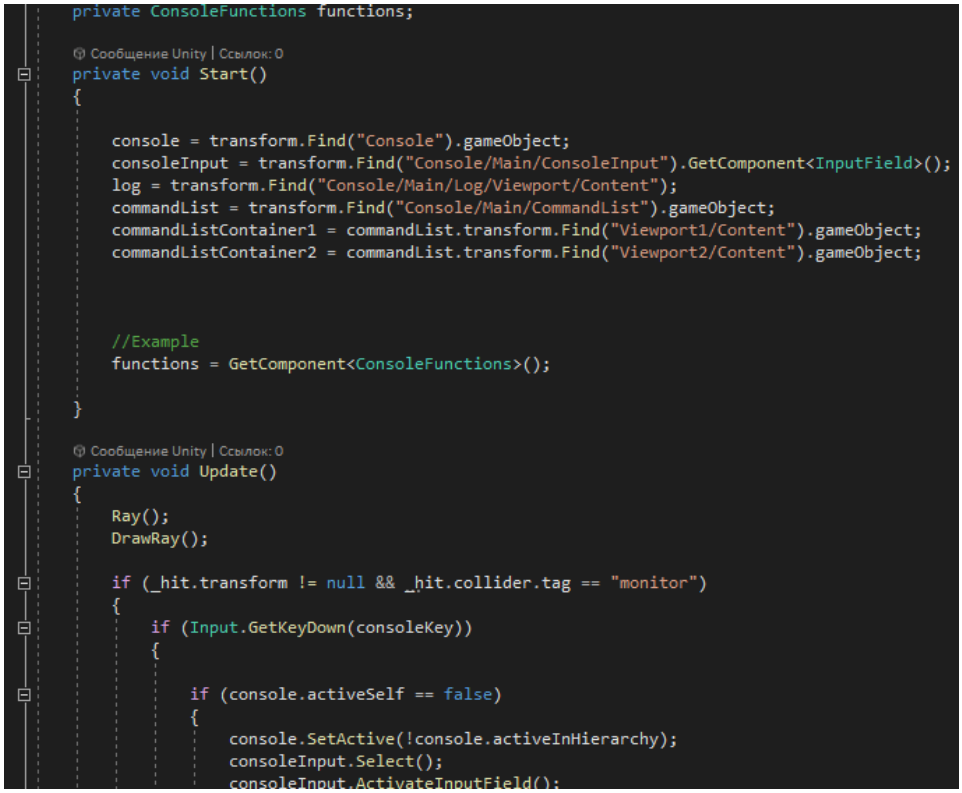
Рисунок 50– Скрипт “ConsoleFuctions”

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using UnityEngine;
using UnityEngine.UI;
using UnityStandardAssets.Characters.FirstPerson;


Скрипт Unity | Ссылок: 0
public class ConsoleManager : MonoBehaviour
{
    [TextArea(3, 5)]
    public List<string> commands = new List<string>();
    public bool canDie;
    public float delay;
    public KeyCode consoleKey;
    public KeyCode PconsoleKey;
    public KeyCode consoleKeyExit;
    public GameObject logPrefab;
    public GameObject cmdListPrefab;
    public FirstPersonController Fpc;

    [SerializeField]private Camera _fpcCamera;
    private Ray _ray;
    private RaycastHit _hit;
    [SerializeField] private float _maxDistanceRay;
    private InputField consoleInput;
    private Transform log;
    private GameObject console;
    private GameObject commandList;
    private GameObject commandListContainer1;
    private GameObject commandListContainer2;
```

Рисунок 51– Скрипт “ConsoleManager”, лист 1



```
    private ConsoleFunctions functions;

    Сообщение Unity | Ссылок: 0
    private void Start()
    {

        console = transform.Find("Console").gameObject;
        consoleInput = transform.Find("Console/Main/ConsoleInput").GetComponent<InputField>();
        log = transform.Find("Console/Main/Log/Viewport/Content");
        commandList = transform.Find("Console/Main/CommandList").gameObject;
        commandListContainer1 = commandList.transform.Find("Viewport1/Content").gameObject;
        commandListContainer2 = commandList.transform.Find("Viewport2/Content").gameObject;



        //Example
        functions = GetComponent<ConsoleFunctions>();

    }

    Сообщение Unity | Ссылок: 0
    private void Update()
    {
        Ray();
        DrawRay();

        if (_hit.transform != null && _hit.collider.tag == "monitor")
        {
            if (Input.GetKeyDown(consoleKey))
            {

                if (console.activeSelf == false)
                {
                    console.SetActive(!console.activeInHierarchy);
                    consoleInput.Select();
                    consoleInput.ActivateInputField();
```

Рисунок 51, лист 2

```
            }
            if (Input.GetKeyDown(consoleKeyExit))
            {
                if (console.activeSelf == true)
                {

                    console.SetActive(false);
                    Fpc.enabled = true;
                    Cursor.visible = false;
                    Cursor.lockState = CursorLockMode.Locked;

                }
            }
            if (Input.GetKeyDown(PconsoleKey))
            {
                run_cmd();
            }
        }
    }

    ссылка: 1
    private void Ray()
    {
        _ray = _fpcCamera.ScreenPointToRay(new Vector2(Screen.width / 2, Screen.height / 2));
    }

    ссылка: 1
    private void DrawRay()
    {
        if (Physics.Raycast(_ray, out _hit, _maxDistanceRay))
        {
            UnityEngine.Debug.DrawRay(_ray.origin, _ray.direction * _maxDistanceRay, Color.blue);
        }

        if (_hit.transform == null)
```

Рисунок 51, лист 3

```
        {
            UnityEngine.Debug.DrawRay(_ray.origin, _ray.direction * _maxDistanceRay, Color.red);
        }
    }
    ссылка: 1
    void Command(string cmd)
    {
        if (cmd == "1.1")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Правильно";
            if (canDie) Destroy(entry, delay);

            functions.OpenSafe();
            commandListContainer1.SetActive(false);
            commandListContainer2.SetActive(true);
        }
        if (cmd == "1.2")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Неправильно";
            if (canDie) Destroy(entry, delay);
        }
        if (cmd == "1.3")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Неправильно";
            if (canDie) Destroy(entry, delay);
        }
        if (cmd == "2.2")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Правильно, сейф открыт";
            if (canDie) Destroy(entry, delay);
```

Рисунок 51, лист 4

```
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Неправильно";
            if (canDie) Destroy(entry, delay);
        }
        if (cmd == "2.3")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Неправильно";
            if (canDie) Destroy(entry, delay);
        }
        if (cmd == "1")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Введите номер варианта в точности как в шапке";
            if (canDie) Destroy(entry, delay);
        }
        if (cmd == "2")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Введите номер варианта в точности как в шапке";
            if (canDie) Destroy(entry, delay);
        }
```

Рисунок 51, лист 5

```
        if (cmd == "3")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "Введите номер варианта в точности как в шапке";
            if (canDie) Destroy(entry, delay);
        }
        if (cmd == "111")
        {
            var entry = Instantiate(logPrefab, log);
            entry.GetComponentInChildren<Text>().text = "111";
        }
    }
```

Рисунок 51, лист 6

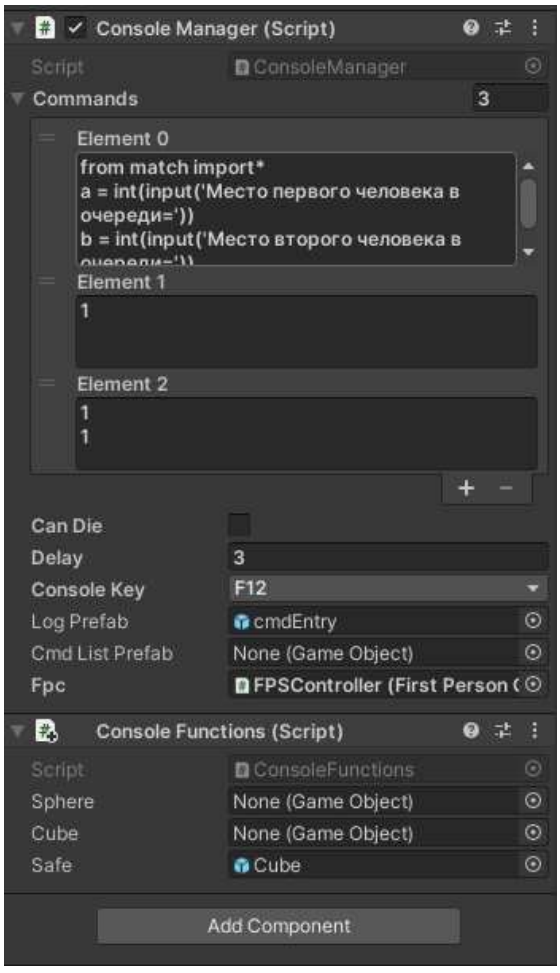


Рисунок 52–Панели скриптов

После этого был создан рабочий билд. Окно создания билда показано на рисунке 53.

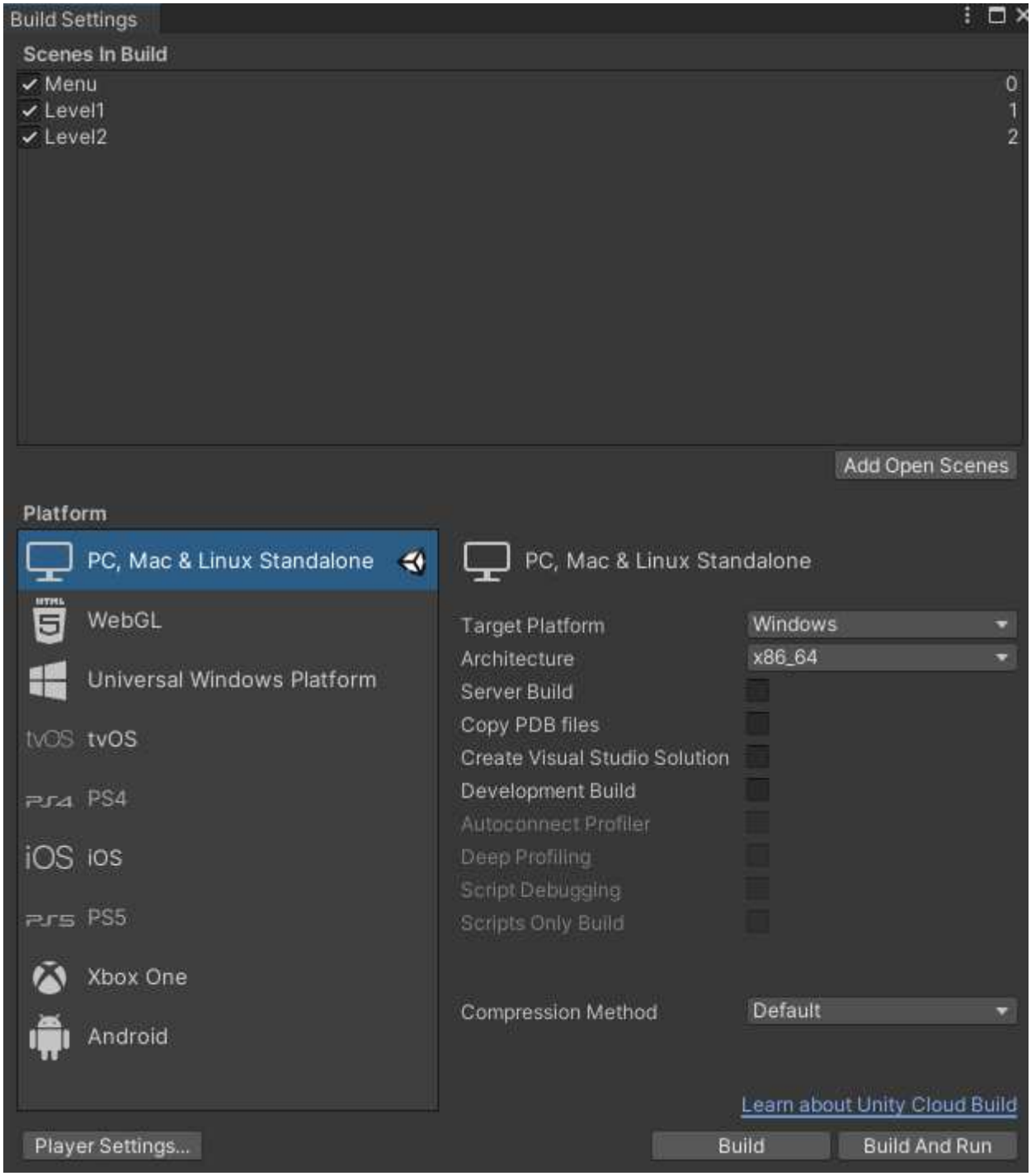


Рисунок 53 – Создание билда игры

После создания билда, функциональные элементы игры были протестированы. Были протестированы меню, записки, сейф и консоль.

Записки должны выводить текст с лекционным материалом. Одна из записок представлена на рисунках 54 и 55.

Рисунок 54 – Записка (вид на уровне)

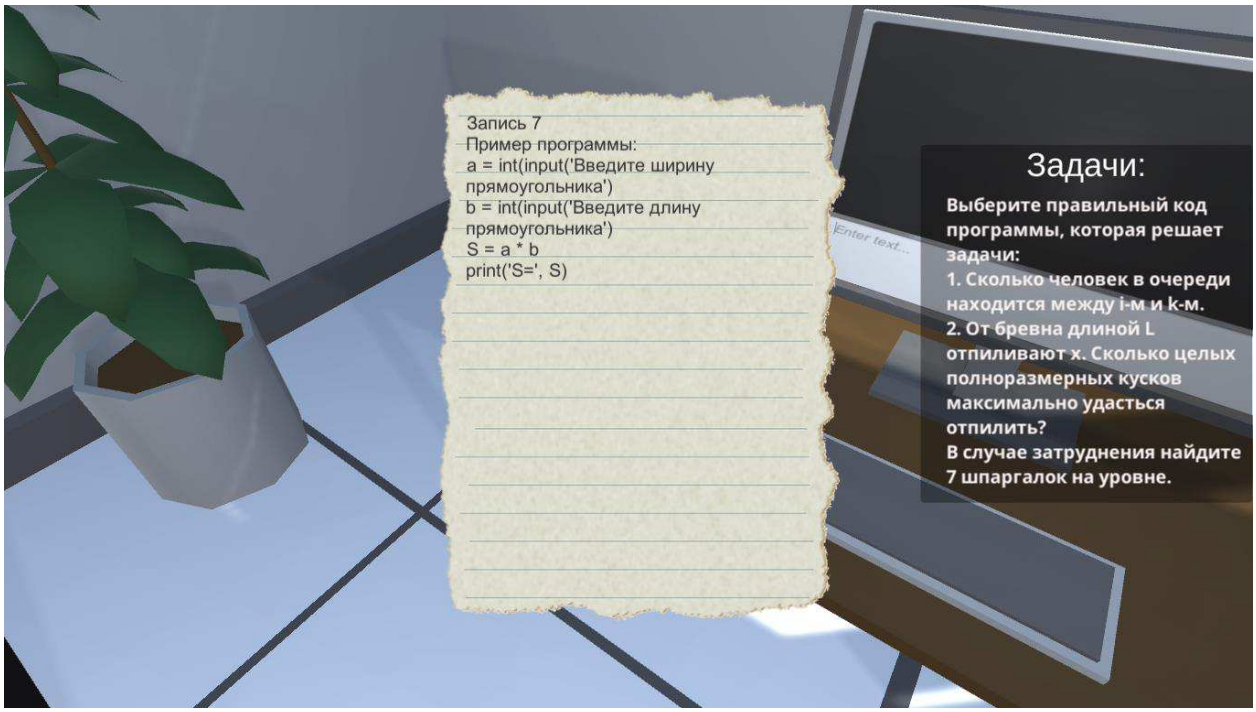


Рисунок 55 – Записка (вид при активации)

После этогобыла проверена работоспособность консоли. Представлена на рисунках 56 и 57.

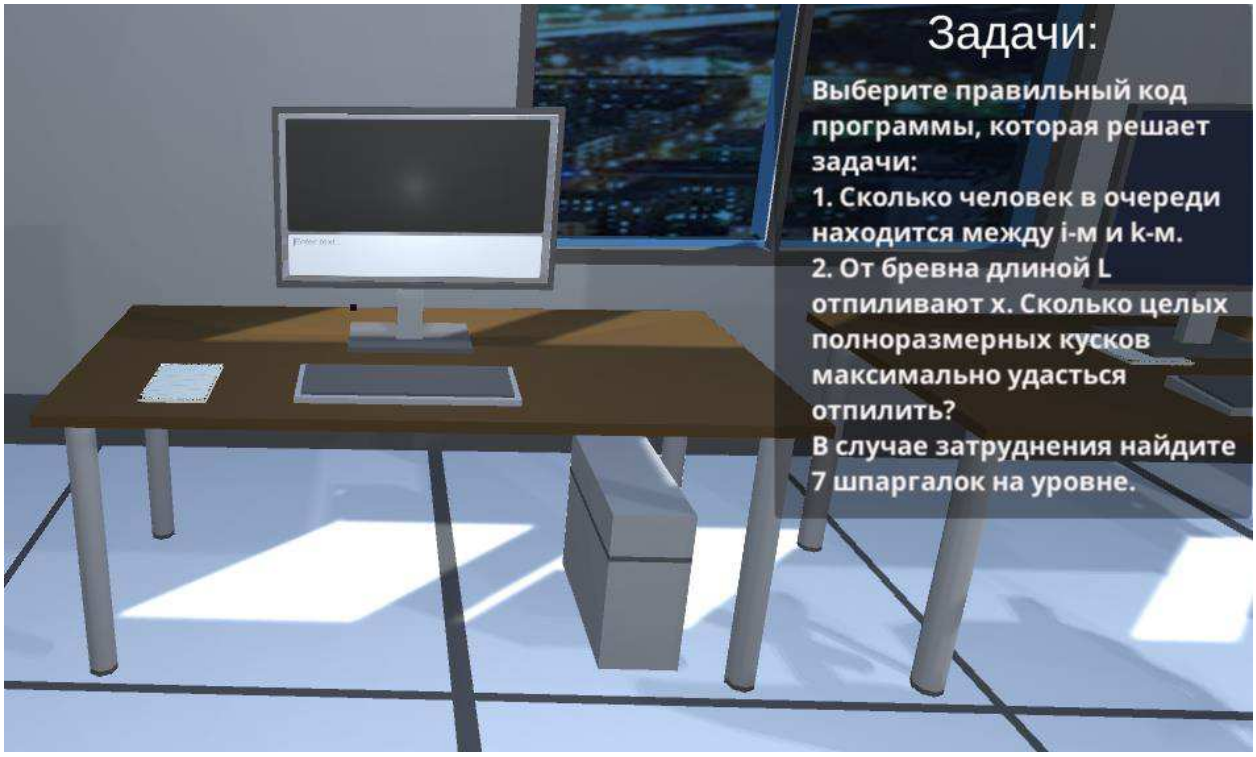


Рисунок 56 – Консоль (вид на уровне)

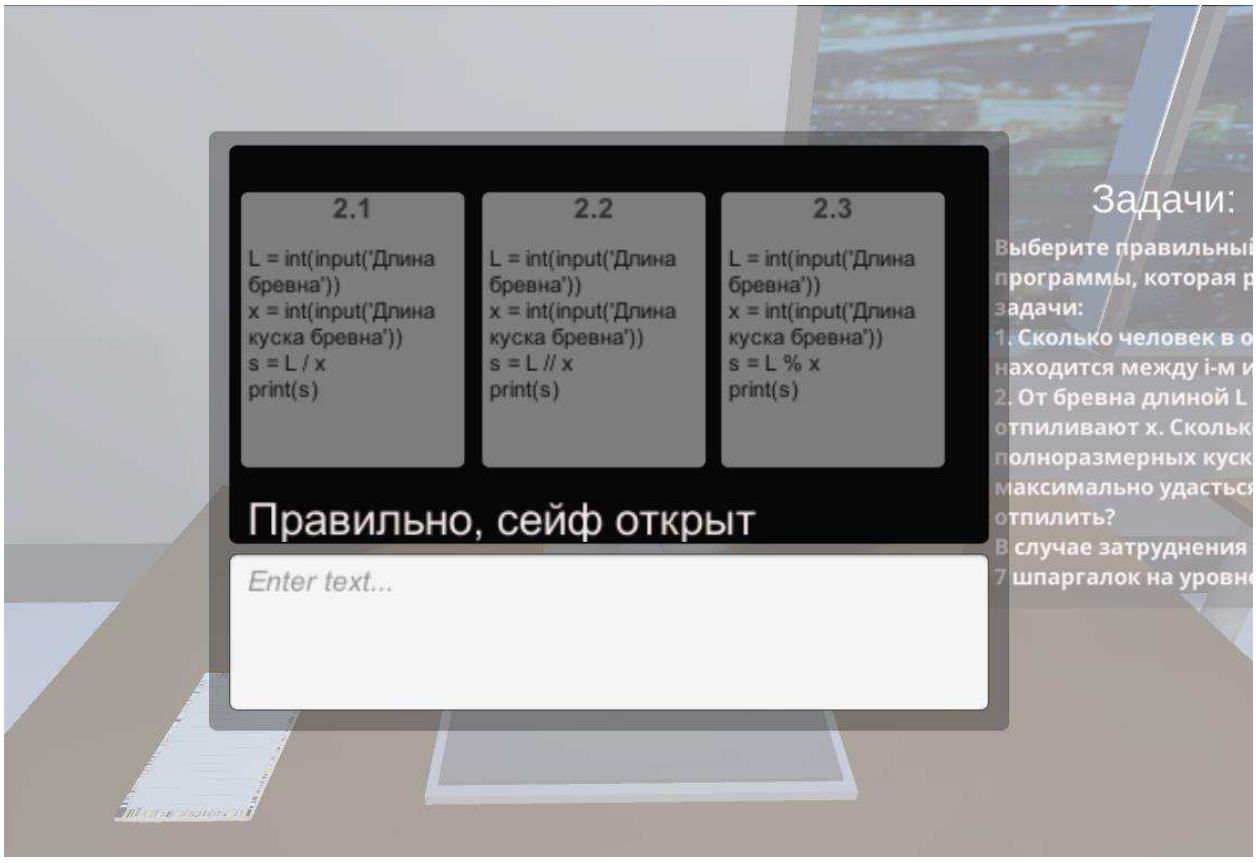


Рисунок 57 – Консоль (вид активного окна)

Затембыли проверены скрипт и анимация сейфа. Открытый сейф представлен на рисунке 58.

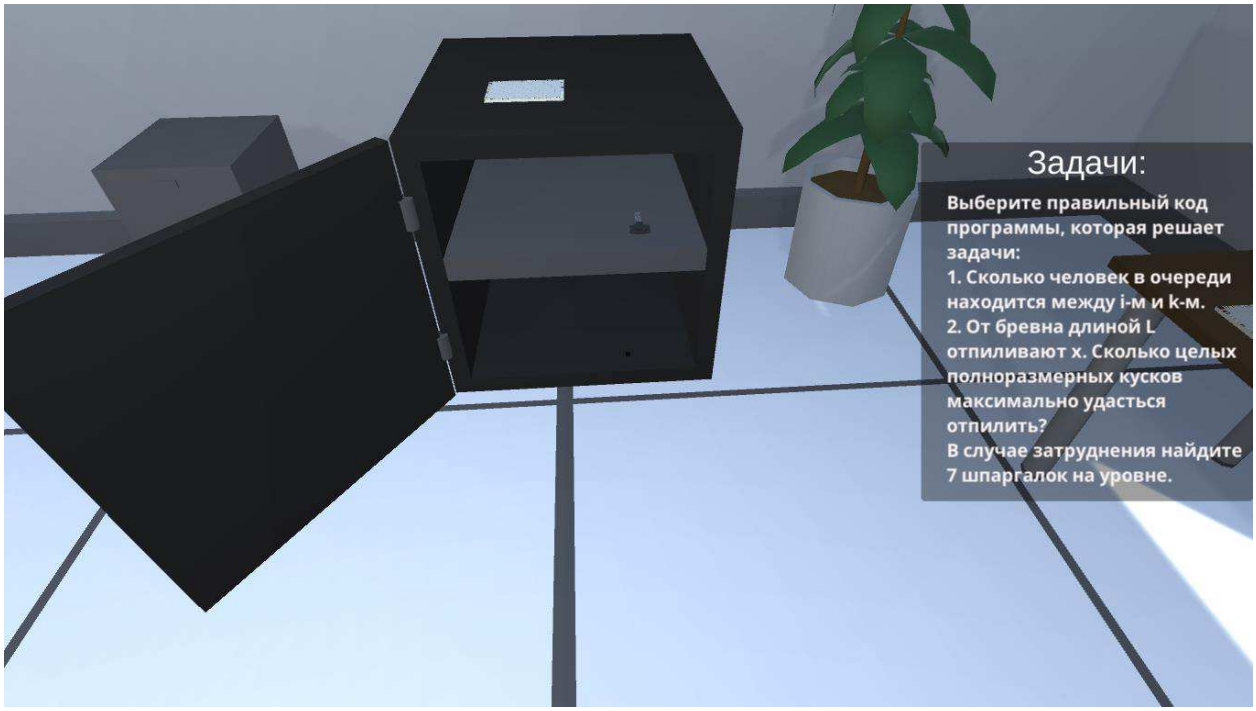


Рисунок 58 – Открытый сейф

2.5 Выводы по разделу планирования и разработки игры

В этом разделе были спроектированы разные модели для правильного понимания системы, а также лучшей разработки в будущем. К ним относятся диаграмма IDEF3, для лучшего понимания взаимодействия пользователей с системой. IDEF0 диаграмма использовалась для демонстрации поэтапной работы сайта. DFD диаграмма использовалась для демонстрации преобразования информации внутри системы. Также был продемонстрирован скелет интерфейс будущей игры.

Также была разработана игра и протестированы ее функциональные элементы.

3 Оценка экономической эффективности создания игры

3.1 Капитальные затраты

Капитальные затраты – единовременные затраты, которые носят разовый характер.

Капитальные затраты рассчитываются по формуле

$$К = К_{пр} + К_{тс} + К_{лс} + К_{по} + К_{ио} + К_{об} + К_{оэ}, \quad (1)$$

где $К_{пр}$ – затраты на проектирование;

$К_{тс}$ – затраты на технические средства;

$К_{лс}$ – затраты на создание линий связи;

$К_{по}$ – затраты на программные средства;

$К_{ио}$ – затраты на формирование Информационной Базы;

$К_{об}$ – затраты на обучение персонала;

$К_{оэ}$ – затраты на опытную эксплуатацию.

Затраты на проектирование.

Разработчику игры будет выплачиваться минимальная оплата труда в Республике Хакасия, составляющая 20467 рублей в месяц.

Также нужно платить обязательные платежи во внебюджетные фонды, которые составляют 30,2% от зарплаты разработчика.

Итого $К_{зп}$=26648 руб.

Затраты на оборудование.

Для создания игры потребуется компьютер. Стоимость его составляющих отдельно показана в таблице 2.

Таблица 2 – Аппаратное обеспечение при проектировании системы

№	Наименование оборудования	Количество единиц оборудования	Стоимость, руб.	Время работы оборудования (годы)
1	Видеокарта Nvidia GTX 1060 3gb	1	10000	4
2	Процессор Intel Xeon E5 2620v3	1	3000	4
3	Оперативная память 8 ГБ Память REG ECC DDR4 2666 МГц	2	2500	10
4	Мышь Bloody V7 game mouse Black USB	1	1790	4
5	Материнская плата X99	1	4000	4
6	Жесткий диск 1Tb	1	3000	4
7	20.7 Монитор HP V214a 60Гц	1	7000	4
8	Клавиатура Defender Next HB-440	1	300	3
9	Куллер Aigo	1	1800	4
10	Блок питания Aerocool ECO 600W	1	2400	4

Стоимость оборудования составляет:

10 000 + 3000 + 1790 + 4000 + 3000 + 2500*2 + 7000 + 300 + 1800 + 2400 = 38290 рублей

Вычисление амортизации оборудования:

((10 000+3 000+1 790+4 000+3 000+7 000+1 800+2 400)/4) + (300/3) + (2 500*2/10)) =8847,5р. – амортизация за 1 год

8847,5/12≈737,3р. – амортизация за 1 месяц

$К_{свт}$=737,3 рубля

Также для разработки потребуется программное обеспечение. Список используемого программного обеспечения представлен в таблице 3.

Таблица 3 – Список используемого программного обеспечения

№	Наименование ПО	Количество единиц ПО	Стоимость	Время работы ПО (годы)
1	Windows 10 Home	1	12,665	5
2	Unity	1	Бесплатное ПО	5
3	Google Chrome	1	Бесплатное ПО	5

Расчет амортизации ПО:

12 665/5=2533р. – амортизация за 1 год

2533/12≈211,08р. – амортизация за 1 месяц

$К_{ипс}$=211,08 рублей

Нормой на прочие затраты (электроэнергия, ком. платежи и т.д.) берем 3%.

$К_{проч}$=(26648+737,3+211,08)*0,03=827,89 р.

$К_{пр}$=26648+737,3+211,08+827,89=28424,27р.

Затраты на технические средства управления.

Затраты на технические средства управления не учитываются, т.к. работа игры не зависит от компьютера пользователя. $К_{тс}$= 0 рублей.

Затраты на создание линий связи.

Так как разработка игры будет проходить в институте, то затраты на линии связи не потребуются и будут равны $К_{лс}$=0 рублей.

Затраты на формирование информационной базы.

Затраты на формирование информационной базы учитываются в расчете заработной платы программиста, следовательно, её можно не учитывать, значит $К_{ио}$=0.

Затраты на опытную эксплуатацию.

На опытную эксплуатацию будут привлечены пять человек, которые протестируют приложение и выскажут свое мнение, замечания и пожелания. Каждому тестируемому планируется заплатить 500 рублей. Затраты на опытную эксплуатацию будут составлять: $К_{оэ}$= 500*4=2 000 рублей.

Капитальные затраты

Капитальные затраты, которые рассчитываются по формуле (1).

К=28424,27+0+116,5+0+0+2000=30540,77 рублей.

3.2 Эксплуатационные затраты

Расчет эксплуатационных затрат происходит по формуле:

$$C = C_{зп} + C_{ао} + C_{то} + C_{гс} + C_{ни} + C_{проч}, \quad (2)$$

где $C_{зп}$ - зарплата персонала, работающего с информационной системой;

$C_{ао}$ - амортизационные отчисления;

$C_{то}$ - затрата на техническое обслуживание;

$C_{лс}$ - затраты на использование глобальных сетей;

$C_{ни}$- затраты на носитель информации;

$C_{проч}$ - прочие затраты.

Затраты на заработную плату персонала.

Эксплуатация игры потребуется в виде последующей поддержке игры патчами и исправлениями. Она будет осуществляться институтом, следовательно, Сзп=0 рублей.

Затраты на амортизационные отчисления.

Для применения игры будут использоваться компьютеры института, следовательно, расчет амортизационных затрат на эксплуатацию не требуется.

$C_{ао}$ =0 рублей.

Затраты на техническое обслуживание.

Затраты на обслуживание и ремонт не учитываются, так как ремонт требуется только для компьютера, на котором установлена игра, следовательно, эксплуатационные затраты на техническое обслуживание не потребуется.

$C_{то}$=0 рублей.

Затраты на использование глобальных сетей.

Затраты на глобальные сети не потребуются, т.к. работа игры не зависит от них.

$C_{лс}$=0 рублей.

Затраты на носители информации.

Затраты на носители информации не потребуются.

Прочие затраты.

Прочие затраты равны 3% от всех затрат

$C_{проч}$=0

Расчет эксплуатационных затрат.

Расчет эксплуатационных затрат происходит по формуле (2).

В связи с тем, что последующей эксплуатацией игры будет заниматься институт, то это значит, что все эксплуатационные затраты будут равны нулю.

C=0 рублей.

3.3 Расчет совокупной стоимости владения информационной системой по методике TCO

Расчет по методике TCO проходит по формуле

$$TCO=DE+IC_1+IC_2, \quad (3)$$

где DE (directexpenses) – прямыерасходы;

IC_1 (indirectcosts) – косвенные расходы первой группы;

IC_2 (indirectcosts) – косвенные расходы второй группы.

$$DE=DE_1+DE_2+DE_3+DE_4+DE_5+DE_6+DE_7+DE_8, \quad (4)$$

где DE_1 – капитальные затраты;

DE_2 – расходы на управление информационными технологиями;

DE_3 – расходы на техническую поддержку автоматизированного обеспечения и программного обеспечения;

DE_4 – расходы на разработку прикладного программного обеспечения внутренними силами;

DE_5 – расходы на аутсорсинг;

DE_6 – командировочные расходы;

DE_7 – расходы на услуги связи;

DE_8 – другие группы расходов.

DE_1=K=31517 рублей;

DE_2=0 рублей, т.к. для работы игры управление не потребуется;

DE3=0 рублей, т.к. для технической эксплуатации наем работников со стороны разработчика не потребуется;

DE_4=0 рублей, т.к. расходы на разработку программного обеспечения учтены в капитальных затратах;

DE_5=0 рублей, т.к. услуги аутсорсинга не потребуются;

DE_6=0 рублей, т.к. командировка не потребуется;

DE_7=0 рублей, т.к. разработка будет проходить в институте;

DE_8=0 рублей;

DE=30540,77+0+0+0+0+0+0+0=30540,77 рублей.

TCO = 30540,77 + 0 + 0 = 30540,77 рублей.

Косвенные затраты реализации проекта равны нулю в связи с тем, что мероприятия по преодолению рисков не требуют финансовых вложений, проект не имеет высокого риска и возможно предусмотреть все затраты по проекту в группе прямых.

3.4 Расчет рисков от реализации проекта

В связи с тем, что игра разрабатывается для ХТИ - филиала СФУ и с тем, что после разработки институт возьмет на себя техническую эксплуатацию проекта, риски при реализации проекта очень малы. Однако есть несколько рисков, которым подвержен проект.

Технологический риск присутствует в связи с тем, что жанр обучающих игр только начал развиваться и редко используется в учебных заведениях как метод обучения студентов. Это может означать,что система будет недостаточно проработана в связи с тем, что аналогов проекта мало.

Риски, связанные с субъективностью оценок и предвзятостью могут появиться из-за недостаточно четкого понимания разработчиком своего проекта.

В таблице 4 перечислены риски, которым подвержен разрабатываемый проект.

Таблица 4 – Перечень рисков проекта

№	Группа рисков	Перечень риска проекта	Уровень влияния риска	Вероятность риска	Возможность снижения или предотвращения риска
1	Технологический риск	Отсутствие знаний о выбранном решении и проработанности технологий	Средний	Низкий	Знание выбранного решения и проработка технологий

Продолжение таблицы 4

2	Риски необъективности оценок и предвзятости	Отсутствие четкого понимания проекта со стороны разработчика	Высокий	Низкий	Тщательное изучение среды разработки и предметной области

3	Риск, что игра будет легкой или не окажет должного влияния	Задачи игры слишком просты и неинтересны	Высокий	Средний	Тщательное обсуждение заданий с преподавателем

3.5 Анализ рынка продуктов-аналогов. Установление стоимости программного продукта

На современном рынке есть множество игр. Однако игр такого жанра как обучающие очень редки. При этом привлечение интереса к игредовольно непростая задача.

Характеристикой разрабатываемого проекта является обучение студентов языку программирования Python.

Так как обучающие игры очень редки, поэтому найти стоимость разработки подобных игр очень тяжело, но можно сравнить зарплату разработчика со средней зарплатой в сфере.

Таблица 5 – Список средней цены разработчика игр

Название	Цена
PayScale.com	5 040$.
Glassdoor.com	7 180$.
Russia.trud.com	71 000 руб.
Себестоимость собственной разработки	31517 руб.

Исходя из полученных данных, можно сделать вывод, что собственная разработка имеет преимущество перед конкурентами в плане цены.

3.6 Выводы по разделу оценки экономической эффективности создания игры

Выбрана методика расчета затрат и оценки экономической эффективности проекта ТСО. Рассчитаны капитальные и эксплуатационные затраты. Капитальные затраты равны 30540,77рублей, а эксплуатационные отсутствуют. Прямой экономической выгоды проект не имеет и косвенных доходов в виде: снижения затрат, увеличения производительности или чего-то подобного тоже. Предполагается, что применение игры позволит заинтересовать студентов к изучению языка и стремлению создать подобные игры, что в свою очередь повышает престижность обучения на направлении подготовки “Прикладная информатика” в ХТИ – филиала СФУ.

ЗАКЛЮЧЕНИЕ

В ходе выполнения ВКР была достигнута поставленная цель и решены задачи:

1. Проанализирована деятельность ХТИ - филиала СФУ.
2. Определены требования заказчика.
3. Выполнен сравнительный анализ игр, работающих в образовательной сфере деятельности.
4. Разработана игра согласно требованиям заказчика.
5. Рассчитаны затраты и экономическая выгода проекта, определены риски, которым подвержен проект.

В первой части была проанализирована деятельность ХТИ - филиала СФУ, определены основные требования к системе, проанализирована предметная область, построена модель IDEF0 до внедрения системы, разработано техническое задание.

Во второй части была определена лучшая средадля разработки данной системы, построена модель IDEF0 после внедрения системы и построена модель работы пользователя в виде диаграммы IDEF3. Также была разработана игра.

В третьей части работы были рассчитаны капитальные затраты, эксплуатационные затраты, определена стоимость по методике ТСО, определены риски проекта, выполнен анализ рынка предоставляющие похожие услуги.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия свободная энциклопедия [Электронный ресурс]: HeroEngine. – Режим доступа: https://en.wikipedia.org/wiki/HeroEngine
2. Википедия свободная энциклопедия [Электронный ресурс]: IDEF3. – Режим доступа: https://ru.wikipedia.org/wiki/IDEF3
3. Википедия свободная энциклопедия [Электронный ресурс]: Python. – Режим доступа: https://ru.wikipedia.org/wiki/Python
4. Википедия свободная энциклопедия [Электронный ресурс]: Unity. – Режим доступа: https://ru.wikipedia.org/wiki/Unity_(игровой_движок)
5. Документация ХТИ – филиала СФУ [Электронный ресурс]: Документация института. – Режим доступа: http://khti.sfu-kras.ru/dokumentatsiya/dokumentatsiya-institut.php
6. Сообщество IT-специалистов [Электронный ресурс]: UML — диаграмма вариантов использования (use case diagram). – Режим доступа: https://habr.com/ru/post/47940/
7. Сообщество IT-специалистов [Электронный ресурс]: Что такое DFD (диаграммы потоков данных). – Режим доступа: https://habr.com/ru/company/trinion/blog/340064/
8. Учебные материалы для студентов [Электронный ресурс]:Методология IDEF3. – Режим доступа: https://studme.org/87186/ekonomika/metodologiya_idef3
9. App2top [Электронный ресурс]: Топ-10 игровых движков. – Режим доступа: https://app2top.ru/game_development/top-10-igrovy-h-dvizhkov-vy-beri-svoj-45170.html

Выпускная квалификационная работа выполнена мной самостоятельно. Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Отпечатано в одном экземпляре.

Библиография 9 наименований.

Один экземпляр сдан на кафедру.

«____» _____ 2021 г.

_____ Раводин Родион Алексеевич
подпись

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Е. Н. Скуратенко
подпись
«21» июня 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03 Прикладная информатика

Разработка игры для изучения языка программирования Python

Руководитель _____ 21.06.21 доцент, канд. техн. наук Е. Н. Скуратенко
подпись, дата

Выпускник _____ 21.06.21 Р. А. Раводин
подпись, дата

Консультанты
по разделам:

Экономический _____ 21.06.21 Е. Н. Скуратенко
подпись, дата

Нормоконтролер _____ 21.06.21 В. И. Кокова
подпись, дата

Абакан 2021

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»

Кафедра прикладной информатики, математики и естественно-научных дисциплин

УТВЕРЖДАЮ
Заведующий кафедрой
_____Е.Н. Скуратенко
подпись
«08» апреля 2021 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Раводину Родиону Алексеевичу

Группа ХБ 17-03

Направление 09.03.03 Прикладная информатика

Тема выпускной квалификационной работы: Разработка игры для изучения языка программирования Python

Утверждена приказом по институту № 222 от 08.04.2021 г.

Руководитель ВКР: Е.Н. Скуратенко, доцент, канд. техн. наук, ХТИ – филиал СФУ

Исходные данные для ВКР: заказ ХТИ – филиала СФУ.

Перечень разделов ВКР:

1. Исследование предметной области.
2. Описание разработки игры.
3. Расчёт затрат и оценка экономической эффективности реализации игры.

Перечень графического материала: нет

Руководитель ВКР _____ Е. Н. Скуратенко
подпись

Задание принял к исполнению _____ Р. А. Раводин
подпись

«08» апреля 2021 г.