Оглавление

[Введение 3](#_Toc104959169)

[Словарь терминов 4](#_Toc104959170)

[1 Аналитическая часть 6](#_Toc104959171)

[1.1 Описание предметной области 6](#_Toc104959172)

[1.2 Анализ требований к приложению пользователей 15](#_Toc104959173)

[1.3 Проектирование поведения игровых объектов 15](#_Toc104959174)

[1.4 Проектирование логики приложения 19](#_Toc104959175)

[1.5 Формулировка целей и требований проекта 27](#_Toc104959176)

[1.6 Описание логики игры 28](#_Toc104959177)

[1.7 Определение средств разработки 30](#_Toc104959178)

[2 Технологическая часть 32](#_Toc104959179)

[2.1 Разработка игровых объектов 32](#_Toc104959180)

[2.4 Создание сценария в Unity 35](#_Toc104959181)

[2.5 Разработка меню игры 40](#_Toc104959182)

[2.6 Описание функций приложения 42](#_Toc104959183)

[2.7 Организация тестирования приложения 46](#_Toc104959184)

[3 Экономическая часть 47](#_Toc104959185)

[3.1 Описание программного продукта 47](#_Toc104959186)

[3.2 Описание качества программного продукта 48](#_Toc104959187)

[3.3 Расчет технико-экономической эффективности разработки ПС 49](#_Toc104959188)

[3.4 Расчет затрат на разработку проекта 50](#_Toc104959189)

[Заключение 52](#_Toc104959190)

[Используемые источники информации 53](#_Toc104959191)

### Введение

Целью дипломного проекта является разработка мобильной игры «Lucky Day» (Удачный день).

Наша игра представляет собой уникальную историю со своими персонажами, диалогами и сюжетными линиями. С «Lucky Day» мы способны полностью продемонстрировать наши идеи, видение и качество. Игра сама по себе открывает в нас творческие стороны, которые являются очень важными в наше время.

Целью создания данной игры является оттачивание навыков разработки мобильных игр, что послужит неотъемлемой частью нашего портфолио и открывает двери в разработку игр. Этим проектом мы расширяем свои навыки и тем самым завоевывается новая аудитория. Проект «Lucky Day» содержит в себе неповторимый дизайн локаций и персонажей, которые были созданы под вдохновением референсов из различных источников, таких как: Pinterest, Behinds, Art Station.

Данный проект будет доступен в Play Market’e, а в последующем и в App Store. Со временем проект будет развиваться, будут выходить обновления, в которых добавиться новая сюжетная линия. В будущем планируется внедрение рекламных интеграций.

Основанием для разработки данного дипломного проекта является приказ №88-ОД директора Бюджетное профессиональное образовательное учреждение Омской области «Омский авиационный колледж имени Н.Е. Жуковского» от 4 апреля 2022 года.

### Словарь терминов

**Программное обеспечение (ПО)** - программа или множество программ, используемых для управления компьютером;

**C#** - объектно-ориентированный язык программирования. Разработан в 1998 — 2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core

**Unity** — межплатформенная среда разработки компьютерных 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.

**GameDev** – это аббревиатура Game Development, то есть разработка компьютерных игр.

**Игровой движок** — базовое программное обеспечение компьютерной игры.

**Игровые механики** (Game Mechanics) – это своды правил и петель обратной связи, которые предназначены для того, чтобы создавать доставляющий удовольствие геймплей. Они являются строительными блоками, которые могут применяться и комбинироваться с игровым и неигровым контекстом.

**Префаб** (англ. Prefab) – то один из типов ресурсов, предназначений для многократного использования и хранящийся в Project View. Префаб может быть вставлен в любое количество сцен и многократно в одну сцену. Когда префаб добавляется в сцену, создаётся его экземпляр. Все экземпляры являются ссылками на оригинальный префаб и фактически его клонами. Независимо от того, как много экземпляров в проекте, при изменении префаба изменяются соответственно и все его экземпляры.

**Пресет** (англ. Preset) – сохраненный набор настроек.

**Казуальщина** – успрощенный геймплей, не нуждающийся в особой мозговой активности, все построено на понятных и легких принципах.

### 1 Аналитическая часть

### 1.1 Описание предметной области

Компьютерная игры служат для организации игрового процесса, связи с партнёрами по игре, или сама выступающая в качестве партнёра. В настоящее время, в ряде случаев, вместо термина компьютерная игра может использоваться видеоигра, то есть данные термины могут употребляться как синонимы и быть взаимозаменяемыми. В компьютерных играх, как правило, игровая ситуация воспроизводится на экране дисплея или обычного телевизора (в этом случае компьютерные игры одновременно являются и видеоиграми).

Консольные игры же предназначены для работы на игровой консоли (игровой приставке), а не на каком-либо другом устройстве типа персонального компьютера или аркадного автомата. Версия мультиплатформенной компьютерной игры для консоли называется консольной версией игры.

В свою очередь, если рассмотреть мобильные игры, то мы увидим видеоигры, в которые обычно играют на мобильном телефоне. Этот термин также относится ко всем играм, в которые можно играть на любом портативном устройстве , в том числе с мобильного телефона ( многофункционального телефона или смартфона ), планшета , КПК и портативной игровой консоли , портативного медиаплеера или графического калькулятора , с доступом к сети или без него . Самой ранней известной игрой для мобильных телефонов был тетрис.

Для своего дипломного проекта я выбрал мобильную игру исходя из трендов на сегодняшний день. С каждым годом мобильные игры становятся все более и более популярными, ведь для них не нужен ПК, который занимает много места и даже ноутбук, сегодня у каждого с собой есть мобильный телефон, на котором можно играть в игры и не от чего не завесить. Так же можно отметить, что позволить купить себе телефон может куда большее количество людей нежели купить ПК или ноутбук. В следствии вытекает фактор что мобильных пользователей на рынке куда больше, а значит и возможностей для реализаций своих проектов.

Моя мобильная игра будет основана на нескольких жанрах, таких как: приключения, головоломка, однопользовательская игра. Теперь можем разобрать их поподробнее.

Приключенческие игры представляют собой жанр игр, в котором основное внимание уделяется сюжету, обычно игрок в роли протагониста должен достичь определенной цели. Для жанра характерны большое количество диалогов, хорошо продуманный сюжет и детально раскрытые характеры персонажей, также в приключенческих играх важное значение уделяется инвентарю при решении загадок.

Так же интересным жанром игр является головоломка, целью которых является решение логических задач, требующих от игрока задействования логики, стратегии и интуиции.

Выбранным режимом игры стала однопользовательская, во время которого с ней через устройства ввода-вывода взаимодействует один человек.

Описание игры: Главный герой Макс просыпается от запаха дыма в своей комнате, он узнает, что это пожар. Ему предстоит не только потушить его, но и узнать причину его возникновения, постепенно вспоминая каждый пройденный этап прошлого дня. Почти весь игровой процесс – это прошлое, но это прошлое способно удивить.

Жанры: Приключение, Головоломки, Драма.

Сеттинг: «Человечество 21-века, наше время, самые обычные школьные будни»

Целевая аудитория: от 12 лет (12+).

**Сценарий**

ПРОЛОГ

Камера постепенно опускается, делим этот пролет на пару этапов:

1. Лого и названия компании

2. Автор идеи

3. Разработчики, дизайнеры и сценаристы

4. Экран становится белым

5. Название игры

6. Постепенное пропадание белого экрана с названием

7. Комната с Максом

Монолог Макса

МАКС: (Просыпается) Сколько время??? Кажется, я опаздываю в школу…

МАКС: Стоп!

МАКС: почему на всю комнату дым? У меня что пожар?

(когда игрок выходит на кухню)

МАКС: нужно срочно потушить пожар, на столе есть стакан с водой, возможно это поможет!

(Уведомление: Такой пожар, невозможно потушить стаканом воды)

(Выливает стакан воды)

МАКС: кажется, толку от этого нет, попробую открыть окно! (Уведомление: при пожаре не стоит открывать окна, от этого пожар только усилится!)

(Макс открывает окно и огонь усиливается)

МАКС: точно у меня же есть огнетушитель!

(Потушил огонь огнетушителем)

МАКС: фух, повезло, успел потушить)

МАКС: уже пора выходить!

(игрок отрывает дверь)

МАКС: Хмм… Странно, в подъезде почему-то нет света, нужно починить щиток.

(Кликает на щиток)

Если игрок заранее не подобрал перчатки, то ему вылазит уведомление о том, что без резиновых перчаток не стоит прикасаться к электричеству! Суть данной мини игры заключается в том, что игроку необходимо последовательно соединить провода по цветам.

(Чинит щиток и нажимает на лестницу или лифт и перемещается на локацию дороги)

Локация дорога 1

Максу по пути в школу предстоит перейти несколько пешеходных переходов, как регулируемых, так и нерегулируемых. Не стоит забывать о правилах предосторожности! Макс пытается пройти нерегулируемый пешеходный переход, но если он не смотрит по сторонам, то приходит уведомление «При переходе дороги необходимо всегда смотреть по сторонам!»)

(Макс переходит дорогу)

Монолог Макса

МАКС: одну дорогу я уже преодолел, осталась последняя.

Локация дорога 2

Макс пытается пройти регулируемый пешеходный переход, необходимо убедиться, что горит зеленый сигнал светофора и посмотреть по сторонам иначе пройти не получится и мы услышим уведомление о том, как необходимо переходить пешеходный переход.)

Монолог Макса

МАКС: последняя преграда на пути к школе пройдена, но нужно поторопиться!

Локация школьный двор:

Макс встречает одноклассника, с которым у них начинается диалог.

АЛЕКС: привет Макс!

МАКС: привет!

АЛЕКС: ты чего опаздываешь?

МАКС: сегодня точно не мой день, причин для опоздания было много!

АЛЕКС: поторопись, у тебя осталось 5 минут чтобы сдать контрольную.

МАКС: до встречи!

(макс заходит в школу)

Локация школа

Монолог Макса

МАКС: нужно быстрее взять тетрадь из шкафчика!

МАКС: осталось вспомнить пароль.

Задание на проверку памяти, изначально у нас открывается воспоминание, где Макс вводит свой пароль от шкафчика из шестизначной комбинации. Игроку необходимо запомнить последовательность цифр, а после повторить её на кодовом замке. В случае если у игрока не получается открыть шкафчик у нас заново воспроизводится видео с кодом. При верном вводе шкафчик открывается

(макс открывает шкафчик и берет тетрадь)

МАКС: нужно быстрее идти в кабинет.

(макс заходит в кабинет)

УЧИТЕЛЬНИЦА: Макс, почему ты опаздываешь?

МАКС: у меня случился пожар, потом отключился свет, еще…

УЧИТЕЛЬНИЦА: хватит выдумывать отговорки, у тебя есть 2 минуты чтобы решить контрольную!

УЧИТЕЛЬНИЦА: садись за парту и решай!

(Макс садится за парту и начинает решать контрольную)

Суть данного квеста в том, чтобы решить 4 примера за ограниченное количество времени, к каждому примеру у нас будут 4 варианта ответа, один из которых будет верным, при нажатии на неверный вариант у нас появляется крестик справа от примера. После выбора неверного ответа у нас появляются новые варианты ответа, один из которых верный, при выборе верного ответа слева от примера появляется галочка. Для прохождения уровня необходимо за 2 минуты решить 4 примера.

(Макс решил контрольную)

УЧИТЕЛЬНИЦА: удивительно, но тут все верно, хоть ты и опоздал.

УЧИТЕЛЬНИЦА: видно, что готовился, получай свою заслуженную 5!

МАКС: спасибо!

ЭПИЗОД 1 ПРОЙДЕН!

### 1.2 Анализ требований к приложению пользователей

Требования к приложению пользователя:

* Пользователь должен иметь навыки управления персонажем, а так же уметь взаимодействовать с игровыми локациями.
* Возможность начинать новую игру.
* Возможность продолжить ранее начатую игру.
* Возможность выбора эпизода с которого пользователь хочет начать прохождение.

### 1.3 Проектирование поведения игровых объектов

Диаграммы состояний отражают модель поведения различных игровых объектов. На рисунке 1 отражена модель поведения персонажа, которым управляет игрок.

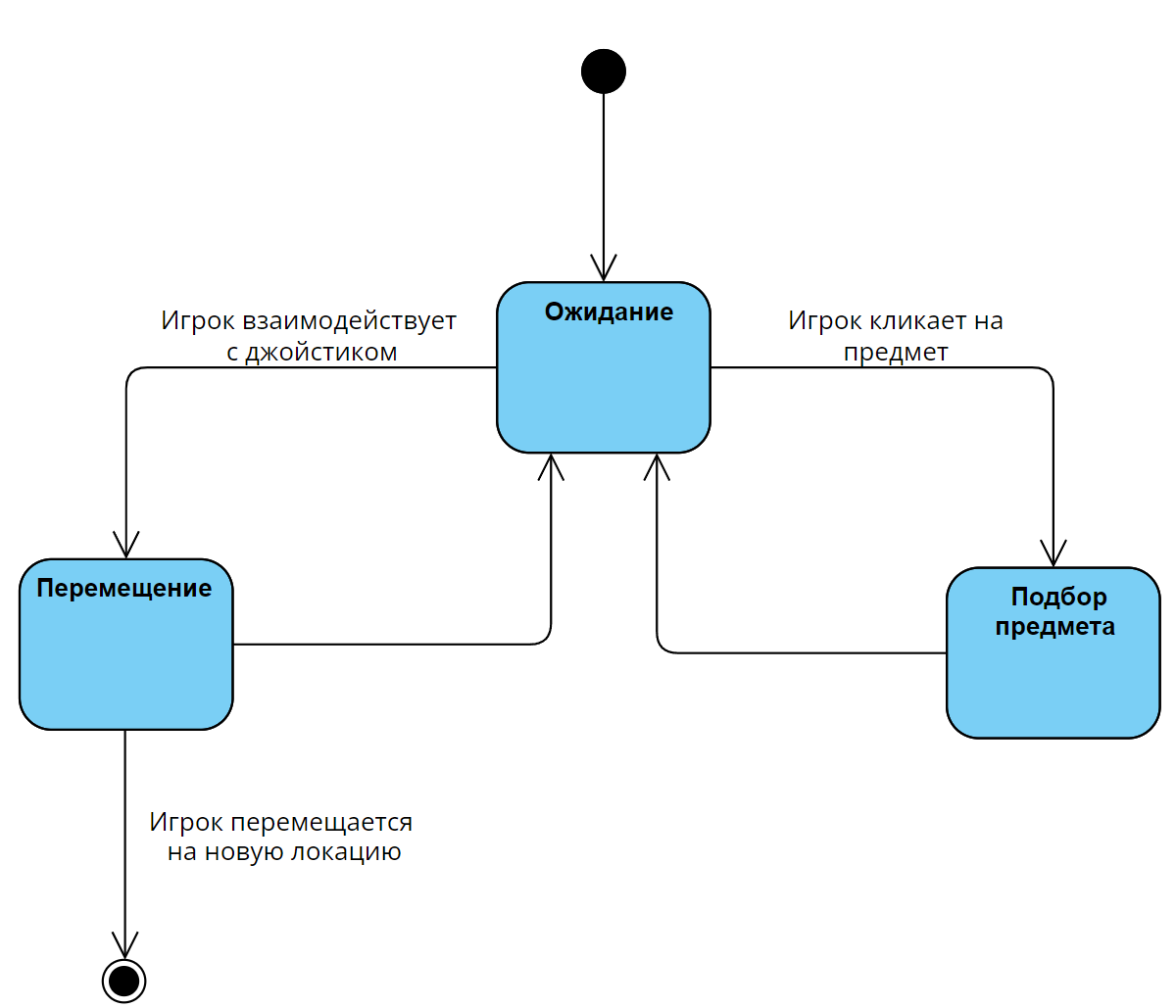


Рисунок 1 – Модель поведения игрока

Диаграмма помогает уточнить поведение игрока с момента начала игры и до любого из вариантов ее конца. На ней видны основные принципы, которым подчиняется игрок, во время прохождения.

На рисунке 2 изображена модель поведения NPC. Их реакция на обнаружение игрока. И сам жизненный цикл. На ней видено какие команды делает неигровой персонаж, перемещается, начинает диалог и т.д.

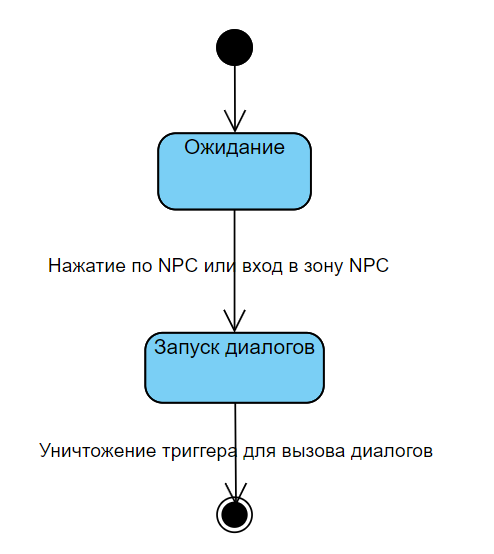


Рисунок 2 – Модель поведения NPC

Далее изображены модели поведений игровых объектов, которые каким-либо способом могут реагировать на внешние воздействия. На рисунке 3 – алгоритм работы дверей между комнатами локации.

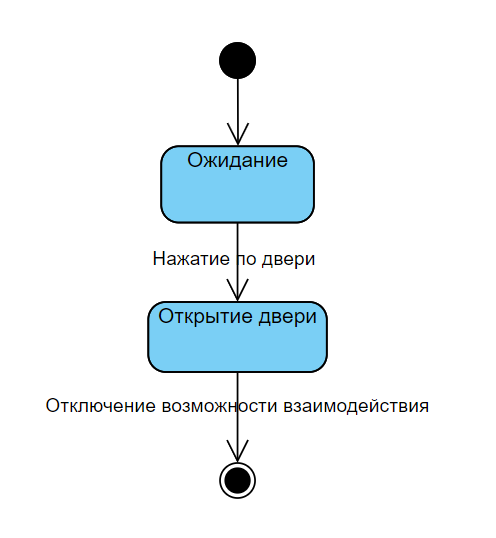


Рисунок 3 – Модель поведения двери

Модель поведения пожара изображена на рисунке 4. Для тушения пожара игроку необходимо выбрать верный вариант, чтобы его потушить.

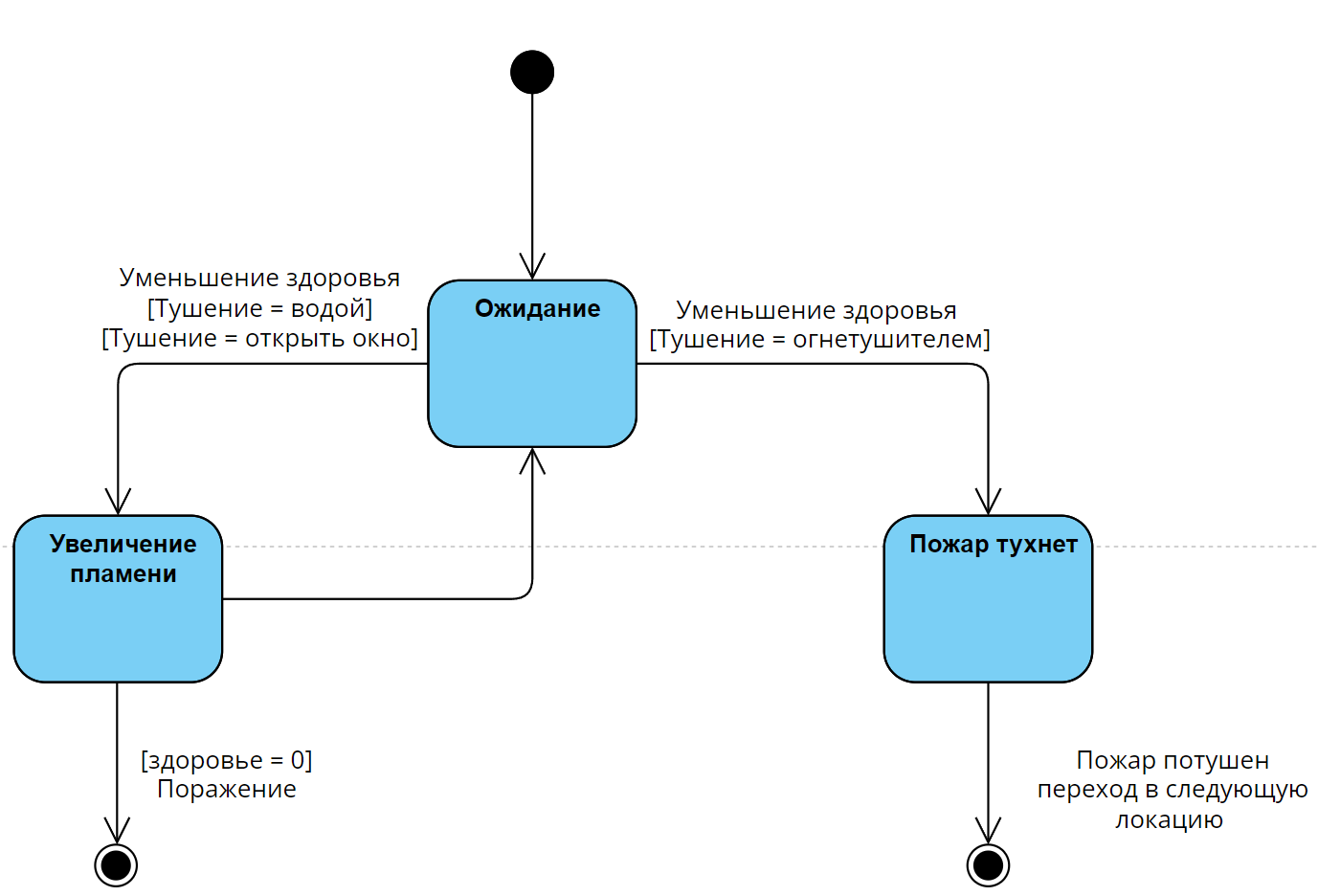


Рисунок 4 – Модель поведения пожара

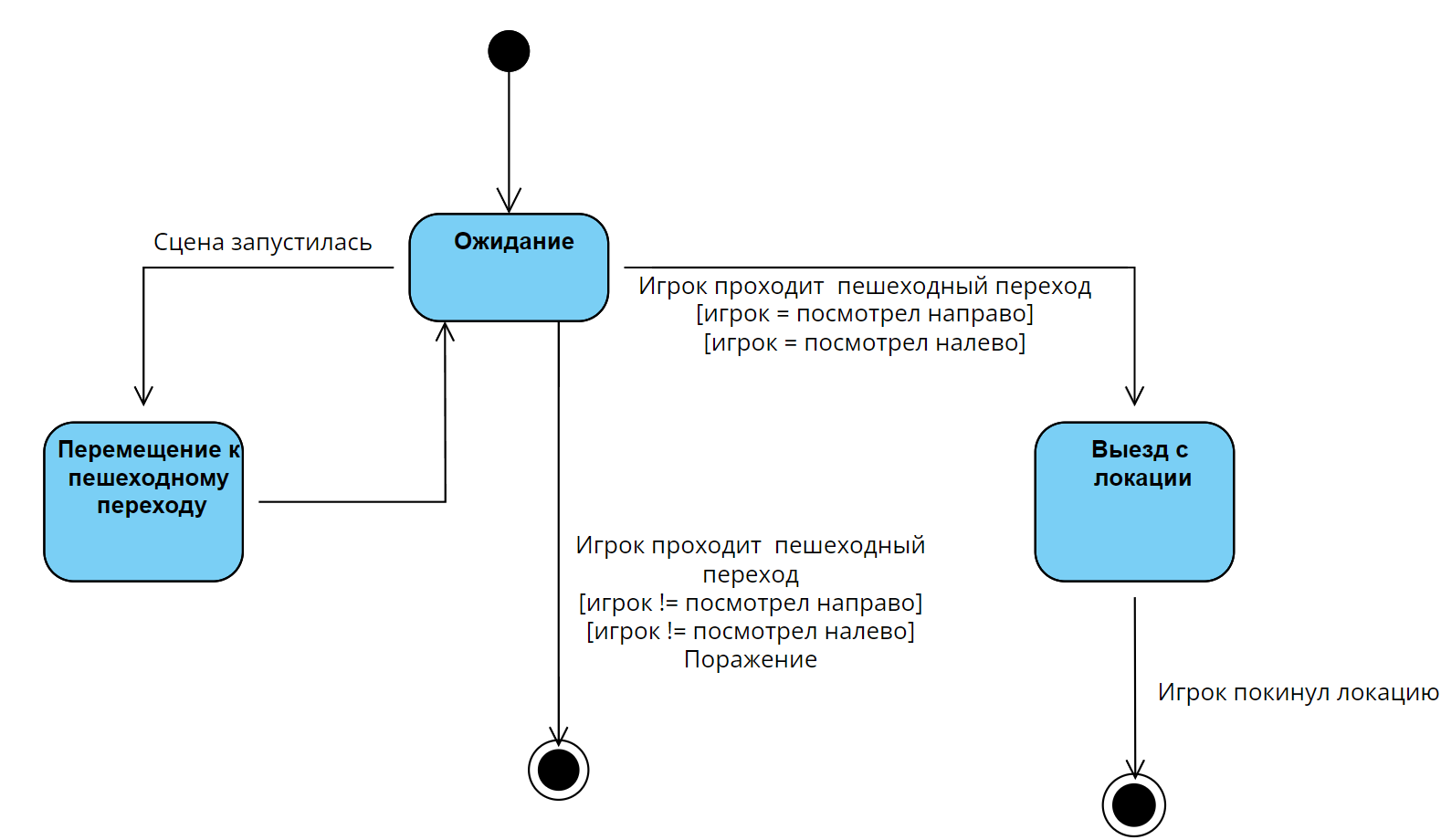
Модель поведения автомобиля изображена на рисунке 5. При запуске сцены автомобиль перемещается до пешеходного перехода, игроку необходимо осмотреться, нажав на кнопки посмотреть налево, посмотреть направо, после чего можно будет пройти и авто уедет за пределы локации.

Рисунок 5 – Модель поведения автомобиля

На рисунке 6 изображена модель поведения щитка на действия игрока. У щитка есть зона починки, в которую необходимо зайти игроку, а также в последующем соединить все провода.

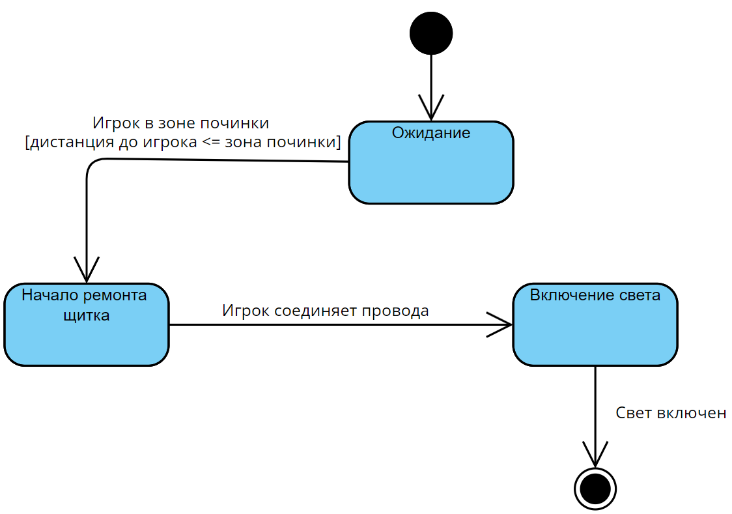


Рисунок 6 – Модель поведения щитка

### 1.4 Проектирование логики приложения

Игровые объекты данного приложения спроектированы в виде иерархии наследованных классов бибилиотеки Unity и библиотек VisualStudio 2019.

На этапе проектирования выделены следующие ключевые классы приложения. Классы содержат различные атрибуты и методы.

На рисунке 6 изображена модель проектирования элементов меню.

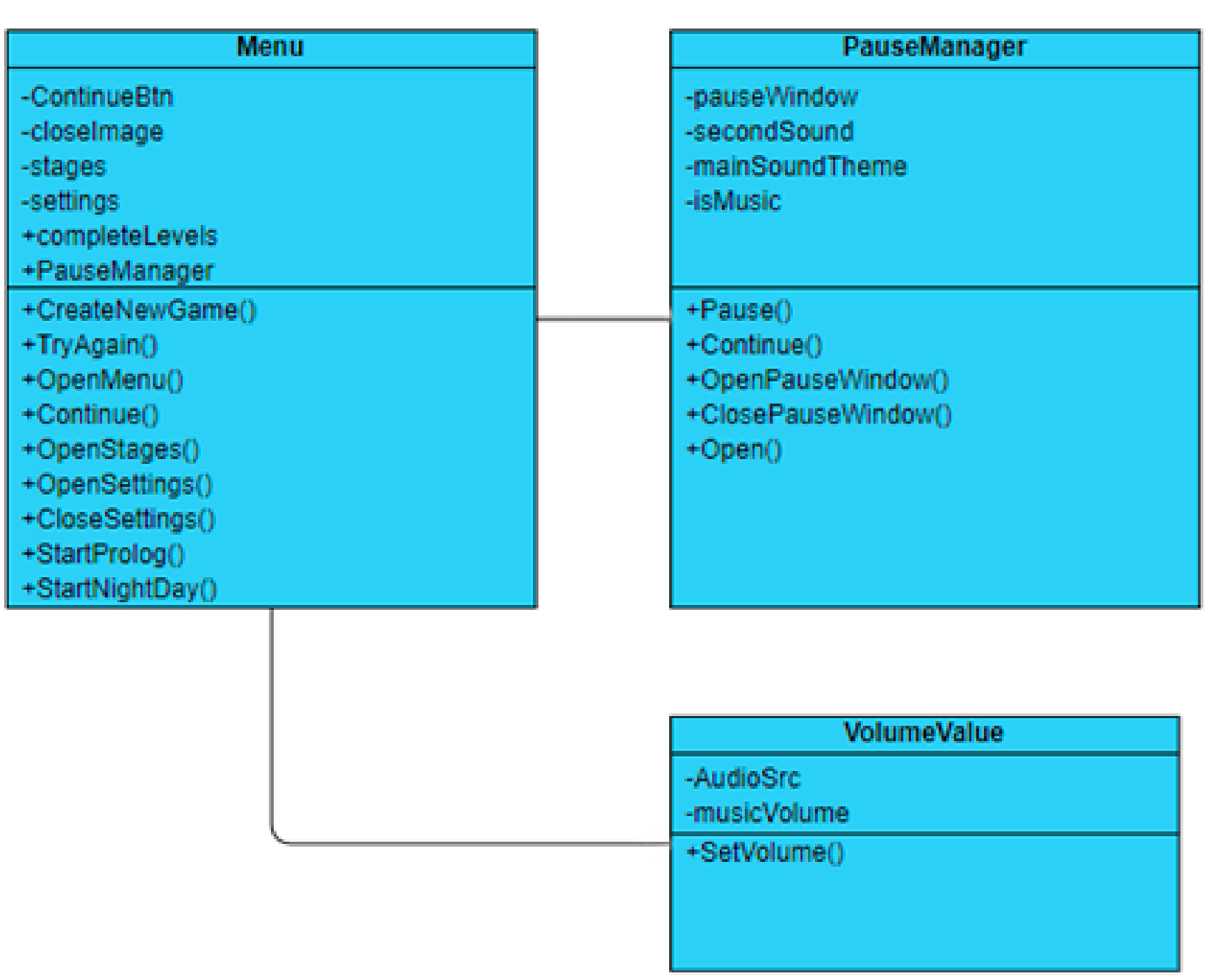


Рисунок 6 – Модель проектирования меню

Класс Menu отвечает за главное меню игры. Его поведение в зависимости от воздействий игрока.

**Атрибуты класса MainMenu**:

* ContinueBtn – кнопка «Продолжить»;
* CloseImage – элемент меню, который скрывается в зависимости от кол-ва пройденных уровней.
* Stages – окно выбора этапа;
* Settings – окно настроек;
* VolumeValue – объект, реализующий настройку звука.
* comleteLevels – число пройденных уровней.
* PauseManager – менеджер, отвечающий за паузу в игре;

**Методы класса MainMenu:**

* СreateNewGame () – создание новой игры;
* TryAgain () – начать заново уровень;
* OpenMenu () – открыть меню;
* Continue () – продолжить игру с последней контрольной точки;
* OpenStages () – открытие окна «Выбор этапа»;
* CloseStages () – закрыте окна «Выбор этапа»;
* OpenSettings () – открытие окна «Настройки»;
* CloseSettings () – закрытие окна «Настройки»;
* StartProlog () – запуск первой локации;
* StartNightDay () – запуска локации «Ночной день»;

**Класс VolumeValue имеет атрибуты:**

* AudioSrc – аудиодорожка;
* musicVolume – значение громкости звука;

**Класс VolumeValue имеет методы:**

* SetVolume () – установка уровня громкости;

**Атрибуты класса PauseManager**:

* pauseWindow – окно меню паузы;
* secondSound – вторая аудиодорожка;
* mainSoundTheme – первая аудиодорожка;
* isMusic – флаг для проверки включена ли пауза;
* VideoSet() – настройка графики;

**Методы класса PauseManager:**

* Pause () – остановка всех процессов игры;
* Continue () – запустить все процессы игры;
* OpenPauseWindow () – открытие окна паузы;
* ClosePauseWindow () – закрытие окна паузы;
* OpenMenu () – открытие меню;

Методы класса PauseManager:

На рисунке 7 изображена модель проектирования роботов-противников.

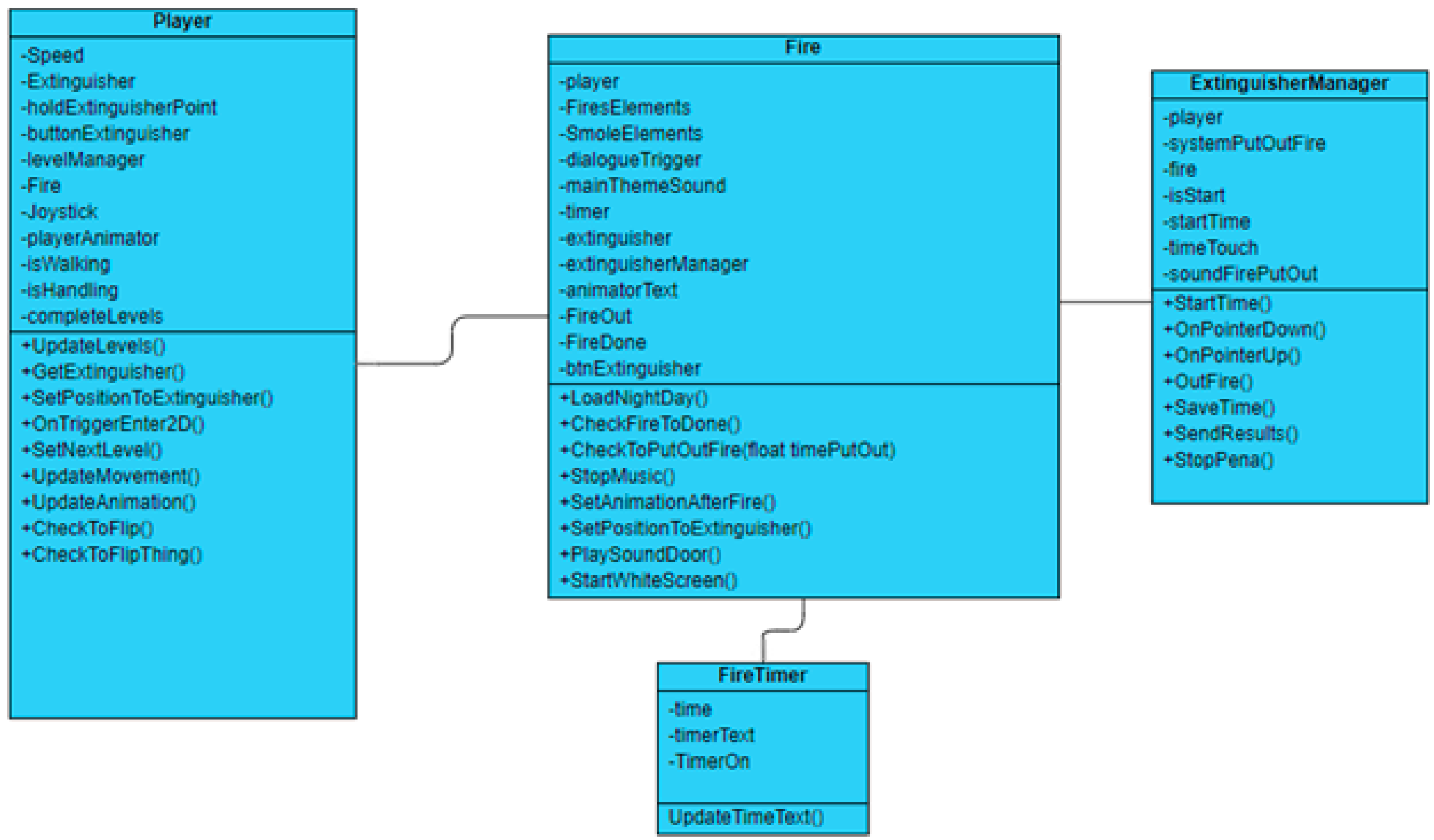


Рисунок 7 - Модель проектирования игрока и тушения пожара

Класс Player содержит основные характеристики главного-персонажа:

**Атрибуты класса**:

* Speed – скорость героя;
* Extinguisher – огнетушитель;
* holdExtinguisher – точка появления огнетушителя после подбора;
* buttonExtinguisher –кнопка для тушения пожара;
* levelManager – левел менеджер для загрузки уровня;
* Fire – огонь;
* Joystick – джойстик для управления героем;
* playerAnimator – аниматор для главного героя;
* isWalking – идет или не идет игрок?;
* isHandling – есть ли в руках что-то?;
* completeLevels – выполненные уровни;

**Методы класса Player:**

* UpdateLevels () – обновление уровней;
* GetExtinguisher () – взять огнетушитель;
* SetPositionToExtinguisher () – устновка позиции при взятии;
* OnTriggerEnter2d () – вход в облась для подбора огнетушителя;
* SetNextLevel () – установка следующего уровня;
* OnTriggerExit2D () – выход из области подбора;
* UpdateMovement () – обновление движений персонажа;
* UpdateAnimation () – обновление анимации персонажа;
* CheckToFlip () – проверка для поворота персонажа;
* CheckToFlipThing () - проверка для поворота персонажа;

Класс Fire отвечает за систему тушения и сам огонь.

**Класс Fire имеет атрибуты:**

* FiresElements – все элементы огня;
* SmokeElements – все элементы дыма;
* dialogueTrigger – триггер на запуск диалога;
* mainThemeSound – основная музыкальная тема;
* sirenaSound – звук сирены;
* Timer – объект класса таймер;
* textTimer – текст таймера;
* sirena – объект сирены;
* extinguisher – огнетушитель;
* holdPoint – точка появления огнетушителя после тушения пожара;
* player – игрок;
* btnExtinguisher – кнопка тушения пожара;

**Класс Fire имеет методы:**

* audioAfterFireDone – аудио после тушения пожара;
* whiteFinal – белый экран;
* extinguisherManager – объект менеджера по тушению пожара;

**Класс Fire имеет методы:**

* CheckToPutFire () – удаление объектов, в зависимости от времени тушения пожара;
* CheckFireToDone () – установка финальных данных по окончанию тушения пожара;
* LoadNightDay () – загрузка локации «Ночной день»;
* PlaySoundDoor () – запуск звука двери;
* StartWhiteScreen () – запуск белого экрана для перехода в следующую локацию;
* StopMusic () – остановка музыки;
* SetAnimationAfterFire () – установка анимации после тушения пожара;
* SetPositionToExtinguisher () – установка позиции огнетушителя после пожара;
* OnTriggerEnter2d () – вход в область тушения;
* OnTriggerExit2d () – выход из области тушения;

Класс FireTimer отвечает за время тушения пожара

**Класс FireTimer имеет атрибуты:**

* Time – значение времени;
* timerText – текст для отображения времени;
* TimerOn – свойство для запуска времени;

**Класс FireTimer имеет методы:**

* UpdateTimeText () – установка нового текста;

Класс ExtinguisherManager отвечает за работу огнетушителя

**Класс ExtinguisherManager имеет атрибуты:**

* Player – игрок
* systemPutOutFire – система частиц пены огнетушителя;
* soundFirePutOut – звук пены;
* fire – огонь;
* isStart – нажата ли кнопка?;
* startTime – время тушения пожара;
* timeTouch – время зажатия;

**Класс ExtinguisherManager имеет методы:**

* StartTime () – запуск времени тушения;
* OnPointerDown () – нажатие кнопки;
* OnPointerUp () – отпустить зажатие кнопки;
* SaveTime () – сохранение времени;
* OutFire () – запуск систем частиц и звука;
* SendResults () – отправка результатов к fire;
* StopPena () – остановка тушения;

На рисунке 8 изображена модель проектирования игрока.

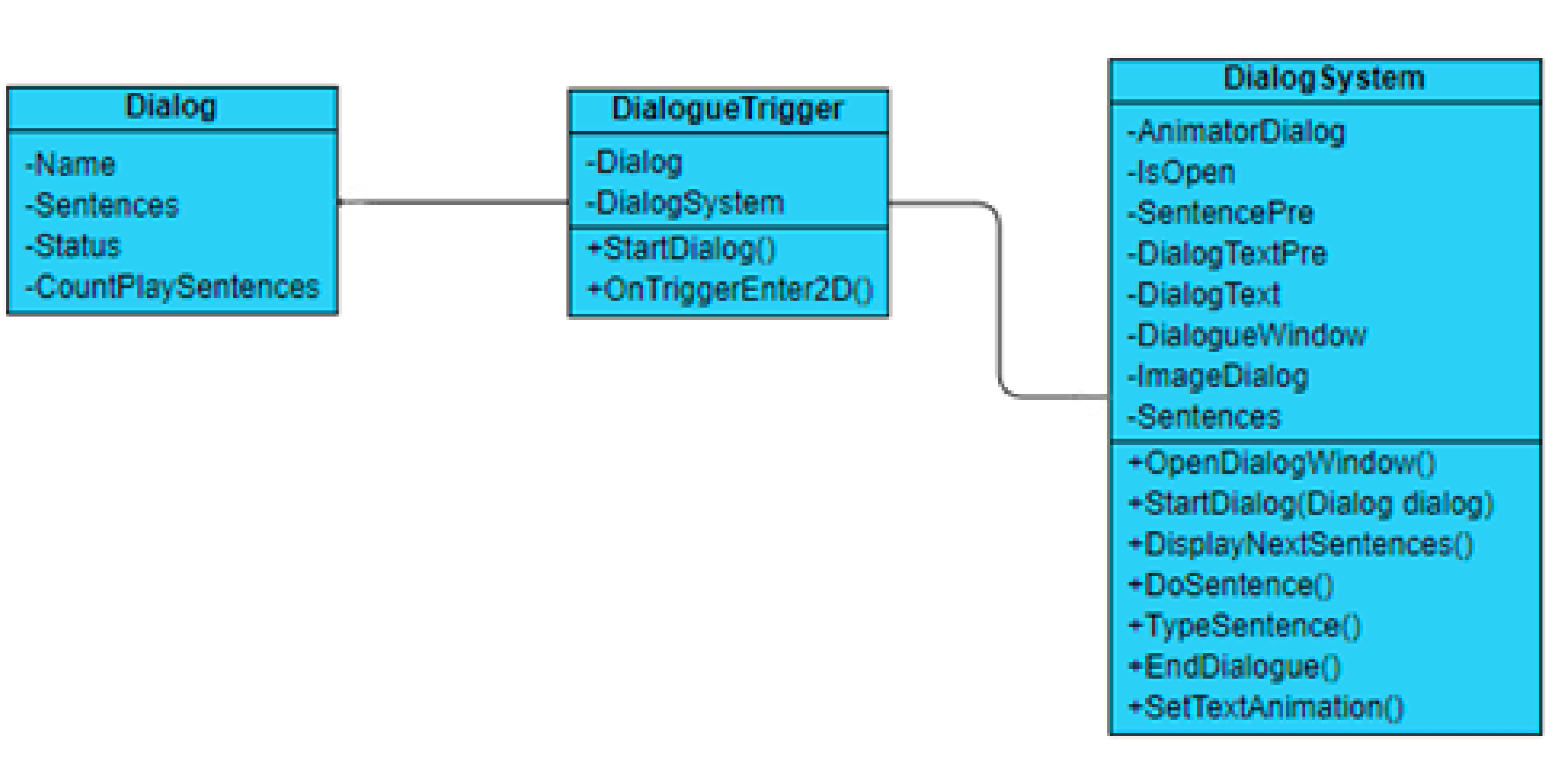


Рисунок 8 – Модель проектирования диалоговой системы

Класс DialogSystem содержит основные характеристики диалоговой системы, а также отвечает за правильное отображение всех диалогов.

**Атрибуты класса DialogSystem**:

* AnimatorDialog – аниматор для диалогового окна;
* IsOpen – открыто ли диалоговое окно;
* DialogTextPre – показатель прошлого сообщения;
* DialogText – показатель нового сообщения;
* DialogueWindow – диалоговое окно;
* ImageDialog – иконка того, кто говорит;
* Sentences – все сообщения;

**Методы класса DialogSystem:**

* OpenDialogWindow () – открытие диалогового окна;
* StartDialog () – запуска диалога;
* DisplayNextSentences () – отображение следующего сообщения;
* DoSentence () – карутина для отображения следующего сообщения с задержкой в 4 секунды;
* TypeSentences () – запись диалога в текст;
* EndDialogue () – окончание диалога;
* SetTextAnimation () – установка анимации для текста;

Класс DialogTrigger отвечает за вызов диалога.

**Атрибуты класса DialogTrigger**:

* Dialog – одно сообщение;
* DialogSystem – диалоговая система;

**Методы класса DialogTrigger:**

* StartDialog () – запуск диалога;
* OnTriggerExit2D () – вызов диалога при входе в нужную область;

Класс Dialog отвечает за описание диалога.

**Атрибуты класса Dialog**:

* Name – имя;
* Sentence – текст сообщения;
* Status – статус отображения;
* CountPlaySentences – число сообщения в одном диалоге;

### 1.5 Формулировка целей и требований проекта

**Главная цель:** Разработка игрового приложения на базе 2d-приключения с элементами головоломок по мотивам самостоятельно созданного сюжета на выбранном языке программирования.

**Дополнительные цели:**

* система вывода диалогов;
* оптимизация программного кода;
* создание компьютерного игрока;
* создание локаций;
* создание моделей персонажей;
* создание моделей предметов;
* создание интерактивных элементов локации.

Требования к проекту:

* пользователь должен иметь возможность взаимодействовать с локацией и персонажем;
* корректная работа всех логических задач;
* проигрыш определяется истечением времени или невыполнением нужных задач;
* выигрыш наступает в случае выполнения всех поставленных задач в локации.

**Определение стандартов качества:**

* простота в обращении;
* простота в обслуживании;
* отсутствие ошибок и сбоев, приводящих к остановке программы;
* небольшое время реакции;
* высокое качество графики;
* соблюдение корректности и согласованности данных.

Маркетинговые и технические спецификации:

Минимальные системные требования:

• Версия Android 5 и выше;

• Процессор - Qualcomm Snapdragon 636;

• Видеопроцессор - Adreno 509;

• Оперативная память – 6 гб;

• Свободного места – 1 гб;

Рекомендуемые системные требования:

• Версия Android 9 и выше;

• Процессор - Qualcomm Snapdragon 865;

• Видеопроцессор - Adreno 650;

• Оперативная память – 8 гб;

• Свободного места – 1 гб;

Существующие технические стандарты и соглашения о стиле программирования:

* использование информативных имен для стандартных и пользовательских объектов;
* комментарии в тексте программы;
* форматирование кодов программы в соответствии с алгоритмической структурой и принципами подчинения;
* объектно-ориентированный подход при реализации приложения;
* выпуск к сроку сдачи дипломного проекта.

### 1.6 Описание логики игры

Игра начинается в комнате Макса, которому предстоит потушить пожар с помощью огнетушителя. Огнетушитель находится непосредственно в самой комнате. После того как игрок тушит пожар, происходит звонок в дверь и нас переносит на день назад. Это нужно, чтобы понять из-за чего случился этот пожар и что стало причиной такой оплошности. И была ли это вообще оплошность со стороны Макса?

Дальше игрок продвигается по каждой локации, сначала это все так же его квартира, затем это школа, в которую нужно успеть добежать. В самой школе придется решить контрольную на время, нажить врага и получить деньги на кино.

Сюжет переносит нас в кино, где мы просто хорошо проводим время, но затем на обратном пути нас встречает враг. Происходит череда событий, из-за которых наш игрок отрубается, а его противник падает замертво. Дома же Макс хочет просто покушать, но, нормально не придя в себя, снова отрубается.

В игре основа – это логические задачки, которые должны быть идеально отработаны. Именно они являются главной движущей силой для прохождения локаций. Иногда нас ждем казульщина, по типу боевых сцен. Кстати, боевые системы тоже будут прописаны, это необходимо для разнообразия геймплея.

Почти в каждой локации будут персонажи, с которыми можно будет поговорить. Эти разговоры могут раскрыть сюжетные моменты всей игры. С некоторыми персонажами можно будет выбрать один из вариантов ответа, что повлечет за собой новые сюжетные линии. Данный метод очень хорош, потому что заставляет игрока проходить игру не один раз, а пройти ее несколько раз, чтобы понять ситуацию с каждой стороны.

### 1.7 Определение средств разработки

Для разработки был выбран игровой движок – Unity3d, который полностью удовлетворяет всем базовым требованиям разработки. На данном движке есть возможность полностью реализовать весь функционал, причем еще и на языке C#, который, как раз, преподавался в нашем учебном заведении. В качестве альтернатив движков были представлены Source 2, Unreal Engine, языков программирования С++, Java.

Source 2 представляет игровой движок является следующим движком компании Valve после Source (движок, использованный в Half-Life 2 2004 года, позднее задействован в ряде игр компании, в том числе Portal 2, Dota 2; благодаря своей модульной структуре данный движок постоянно дорабатывается, поддерживаясь на актуальном уровне, и имеет множество внутренних версий)

Движок, разрабатываемый и поддерживаемый компанией Epic Games, - Unreal Engine. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх.

Unity3d это ничто иное как, межплатформенная среда разработки компьютерных 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.

Основными языками для разработки игр является, конечно же С++, C# и Java.

Первый отличается многообразием и функциональностью, позволяет проводить операции по объектно-ориентированному, процедурному и обобщенному программированию. Основное преимущество **С++** — обширная стандартная библиотека, содержащая такие функции, как ввод/вывод и многопоточность и обеспечивающая возможность удобной алгоритмизации и контейнирования. C++ одинаково эффективно применяется на самых различных платформах и успешно комбинируется с другими средствами создания игровых программ. Изучение С++ — процесс достаточно сложный и требующий от начинающих программистов больших усилий.

**С#,** в свою очередь, пользуется популярностью в геймдеве благодаря своей полной объектной ориентированности. Применяется для разработки игровых продуктов на ПК (в частности, на платформе .NET Framework). Хорошо работает с движками и программами для создания графического и звукового оформления. Обеспечивает поддержку полиморфизма, перезагрузки операторов и позволяет разработать все необходимые элементы, такие, как архитектура и логика, требуемые для создания полноценной игры.

Тогда как **Java** используется для проработки логики и механики мобильных игр, особенно для продуктов, предназначенных для Android. Популярность этого языка обусловлена его многопоточностью и возможностью беспрепятственного взаимодействия с памятью мобильных устройств. Отличается хорошим взаимодействием с движками и программами графического и звукового оформления. На Java часто реализуют серверную структуру для многопользовательских игр.

Для разработки был выбран игровой движок – **Unity3d**, который полностью удовлетворяет всем базовым требованиям разработки. На данном движке есть возможность полностью реализовать весь функционал, причем еще и на языке C#, который, как раз, преподавался в нашем учебном заведении.

### 2 Технологическая часть

### 2.1 Разработка игровых объектов

Основными игровыми объектами являются игровое пространство, игрок и не игровые персонажи. Локации создаются линейно, согласно разработанному макету, они отличаются размером и наполнением, все предметы взаимодействия находятся на заранее выбранных местах случайным образом из набора.

Игроку доступны следующие взаимодействия:

* подбор предметов, находящихся на локации;
* при подборе огнетушителя игроку становится доступна возможность его использования. (рисунок 12);

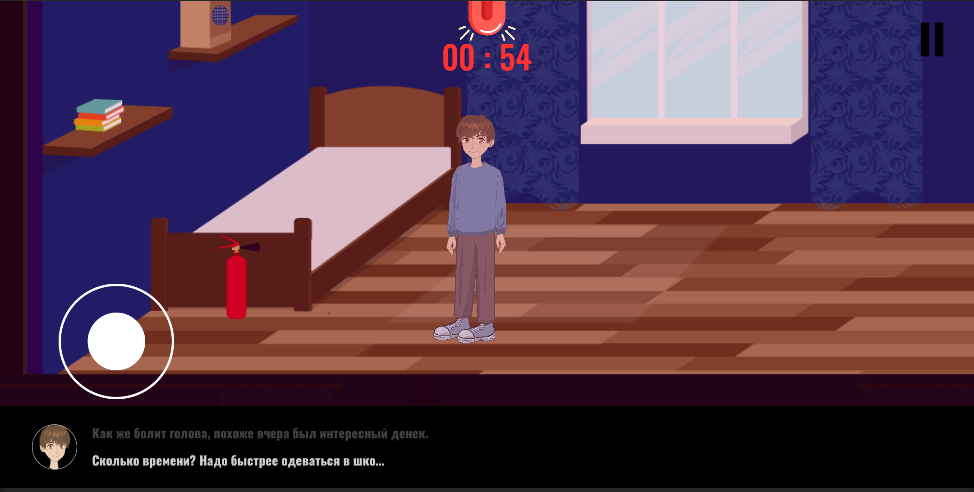


Рисунок 12 – Комната с огнетушителем

* пользователь может взаимодействовать с огнетушителем, чтобы потушить пожар, возникший на кухне (рисунок 13);



Рисунок 13 – Огнетушитель поднят

* тушение пожара, которое возможно после поднятия огнетушителя и похода к области тушения



Рисунок 14 – тушение пожара огнетушителем

Практически в каждой локации расположенно определенное мини задание, которое необходимо выполнить игроку. В случае если игрок долго не будет находить что от него требуется, предмет с которым необходимо взаимодействовать, либо же персонаж будут подсвечиваться синим цветом. Для большего понимания что же требуется от игрока, существует система диалогов, в которой игроку будут намекать что же надо будет сделать на следующем этапе игры.

Для тушения пожара используется система частиц (рисунок 15);



Рисунок 15 – Система частиц для пены

Игра будет наполенена второстипенными персонажами, с которыми игрок будет взаимодействовать. В некоторых случаях персонажи сами будут подходить к игроку и начинать с ним диалог. Так же игрок и сам может начинать диалоги с персонажами, для этого ему будет достаточно подойти в зону активации диалога и кажать на нужного персонажа.

### 2.4 Создание сценария в Unity

Проект в Unity делится на сцены (рисунок 31) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек.

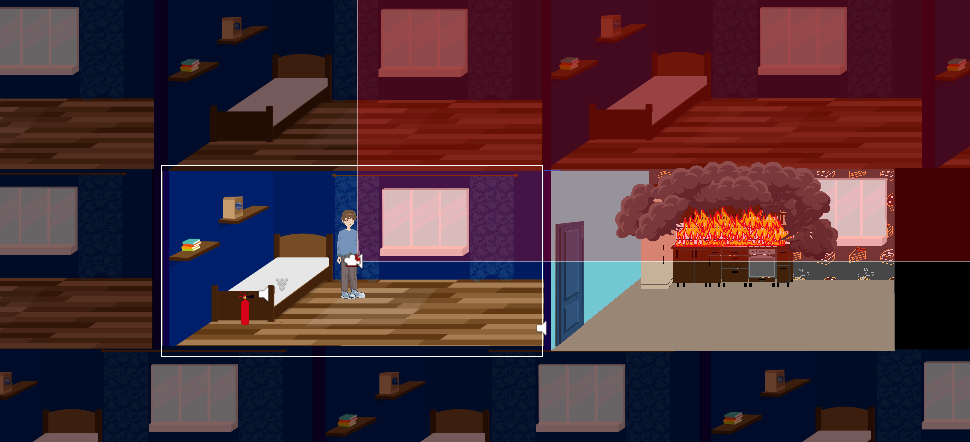


Рисунок 19 – Пример сцены

Unity включает в себя мощный инструмент для управления анимациями (рисунок 32). В нем можно создавать сценарий перехода между анимациями, в зависисмости от значения переменной. Там же можно установить анимацию по умолчанию (в инспекторе она будет выделена оранжевым). А так же настроить плавный переход между двумя различными анимациями.

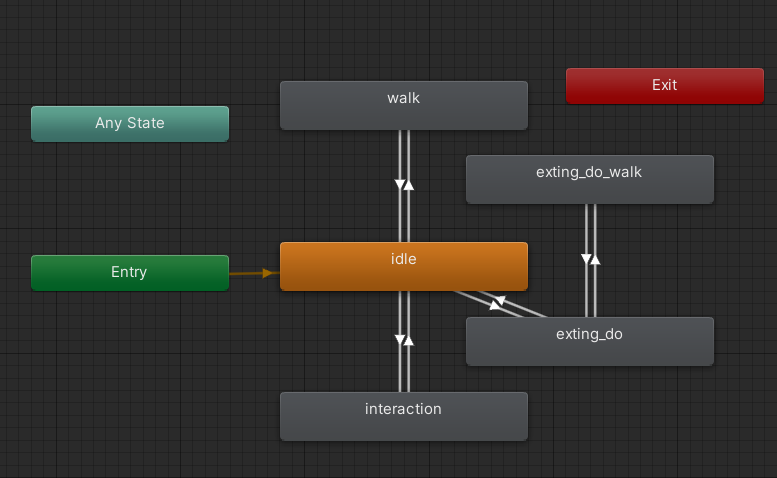




Рисунок 20 – Создание сценария смены анимаций

Редактор Unity имеет простой Drag&Drop интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Движок поддерживает два скриптовых языка: C#, JavaScript (модификация). Ранее была поддержка Boo (диалект Python), но его убрали в 5-ой версии. Расчёты физики производит физический движок PhysX от NVIDIA.

Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором он должен отображаться

У любого объекта на сцене обязательно присутствует компонент Transform (рисунок 21) — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

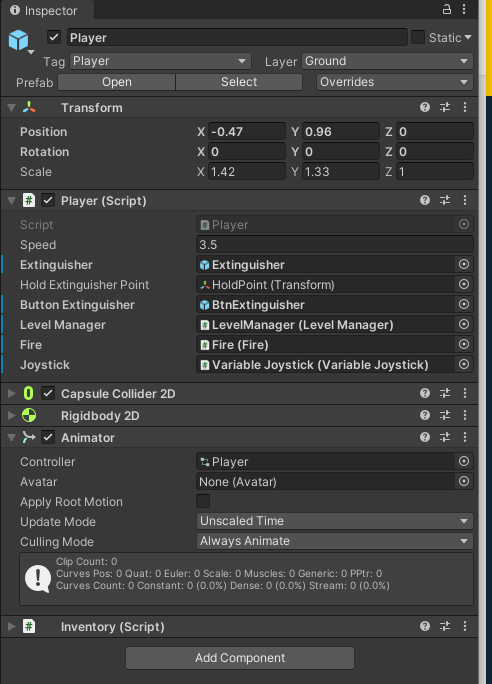


Рисунок 21 – Параметры объекта Player

К объектам можно применять коллизии (в Unity т. н. коллайдеры — collider), которых существует несколько типов.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll (тряпичная кукла). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта (рисунок 22). Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

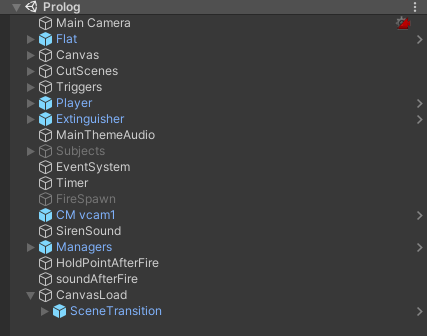


Рисунок 22 – Иерархическа система сцены

При импорте текстуры в Unity можно сгенерировать alpha-канал, mip-уровни, normal-map, light-map, карту отражений, однако непосредственно на модель текстуру прикрепить нельзя — будет создан материал, которому будет назначен шейдер, и затем материал прикрепится к модели (рисунок 23). Редактор Unity поддерживает написание и редактирование шейдеров. Редактор Unity имеет компонент для создания анимации, но также анимацию можно создать предварительно в 3D-редакторе или 2D-редакторе и импортировать вместе с моделью, а затем разбить на файлы.

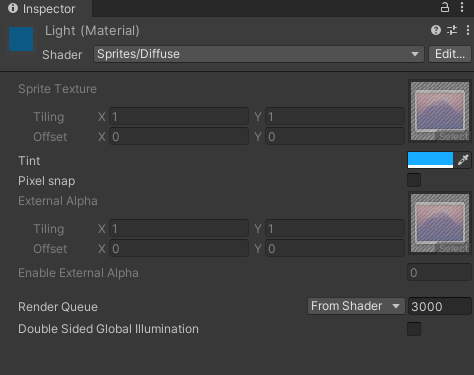


Рисунок 23 – Создание материалов

Движок поддерживает множество популярных форматов. Модели, звуки, текстуры, материалы, скрипты можно запаковывать в формат .unityassets и передавать другим разработчикам, или выкладывать в свободный доступ. Этот же формат используется во внутреннем магазине Unity Asset Store, в котором разработчики могут бесплатно и за деньги выкладывать в общий доступ различные элементы, нужные при создании игр.

### 2.5 Разработка меню игры

1. Игра разработана в виде Android-приложения. При запуске приложения открывается главное меню (рисунок 24), в котором пользователю открывается меню игры.

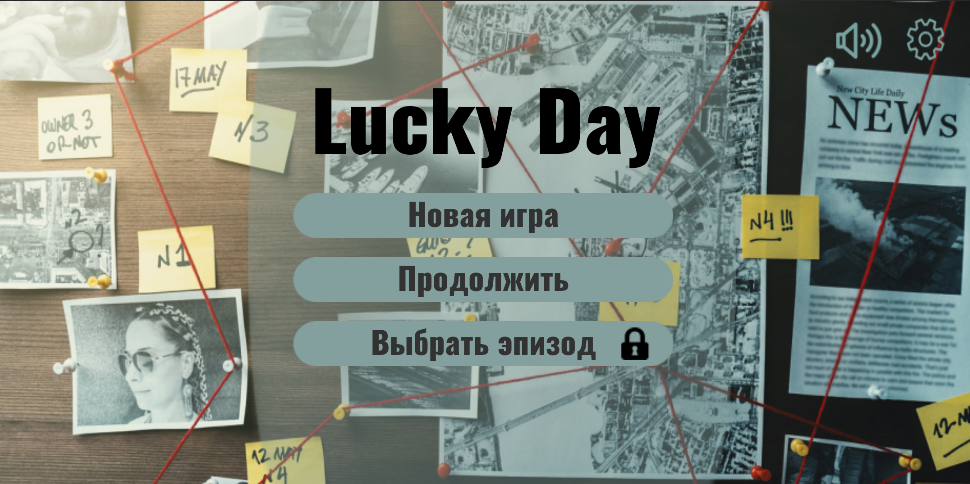


Рисунок 24 – Главгное меню

При переходе в меню настроек (рисунок 25) можно изменить параметры звука, регулировать громкость звука игровых взаимодействий, а также громкость музыки.

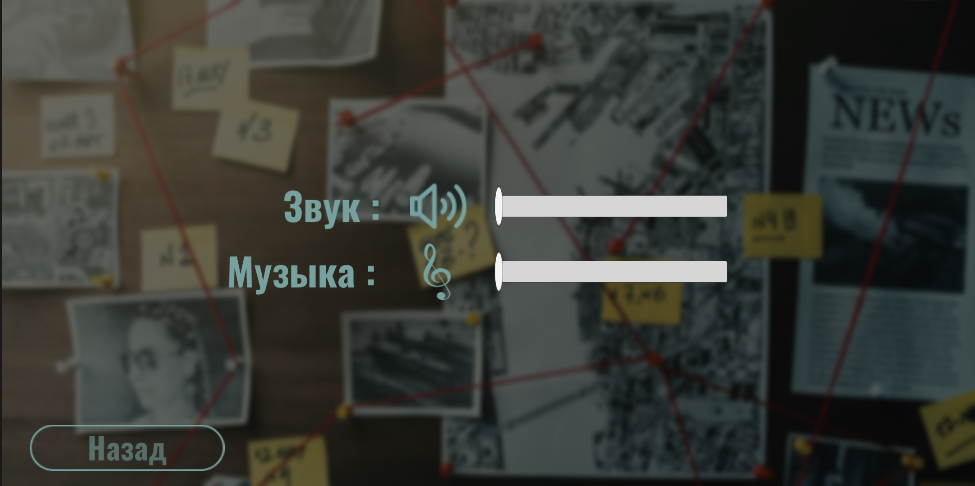


Рисунок 25 – Меню настройки уровня громкости

По мере прохождения игры игроку будут открываться возможности выбора эпизода (Рисунок 26). Тут игрок может выбрать с какой именно контрольной точки он хочет начат

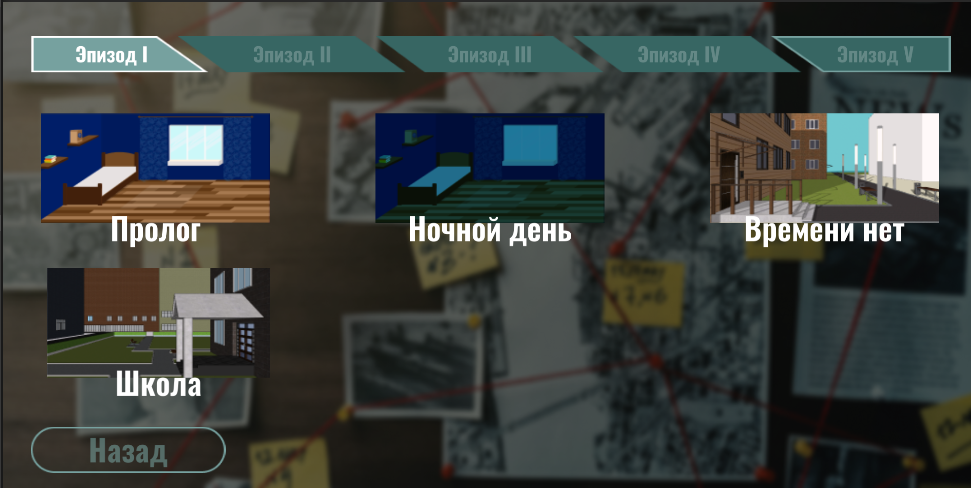


Рисунок 26 –Выбор этапа игры

### 2.6 Описание функций приложения

Основной функцией приложения является игра. Методы, отвечающие за реализацию логики игры, распределены во множестве префабов, в зависимости от целей. На противниках расположены скрипты, отвечающие за их поведение. На итерактивных объектах, таких как двери, статические персонажи, предметы инвентаря – реакция на взаимодействие с игроком. На локации – точки появления противников, интерактивных объектов и реакция на появление игрока.

Но основным классом можно назвать Player, который отвечает за всю работоспособность главного персонажа и его взаимодействие с некоторыми объектами.

**Пример кода класса Player**

Листинг 1 – Базовые Методы для класса Player

|  |
| --- |
| //Скорость движения игрока  [SerializeField] private float \_speed = 1.5f;  //Объект огнетушителя  [SerializeField] private GameObject extinguisher;  //Точка появления огнетушителя  [SerializeField] private Transform holdExtinguisherPoint;  //Кнопка тушения  [SerializeField] private GameObject buttonExtinguisher;  //Менеджер уровней  [SerializeField] private LevelManager levelManager;  //Объект класса fire для работы с огнем  [SerializeField] private Fire fire;  //обработка движений через joystick  [SerializeField] Joystick joystick;  //Аниматор, который отвечает за все анимации персонажа  private Animator \_playerAnimator;  //проверки на движение и наличие чего-либо в руках  private bool \_isWalking = false;  private bool \_isHandling = false;  //выполненные уровни  private int completeLevels;  //свойство на установку значения булевой переменной - наличие чего-либо в руках  public bool Handling  {  set  {  \_isHandling = value;  }  }  //проверка на возможность подобрать предмет (огнетушитель)  private bool \_isCanToHandThing = false;    private void Start()  {  //получаем Аниматор с нашего объекта, на котором висит этот скрипт  \_playerAnimator = GetComponent<Animator>();  //получаем пройденные уровни  completeLevels = levelManager.CompleteLevels;  }  private void Update()  {  //обновление движения  UpdateMovement();  //обновление анимаций  UpdateAnimation();  //обновление уровней  UpdateLevels();  //отобразить кнопку тушения, если взят огнетушитель  if (\_isHandling)  {  SetPositionToExtinguisher();  buttonExtinguisher.SetActive(true);  }  } |

Листинг 2 – Обработка движений и анимаций класса Player

|  |
| --- |
| private void UpdateMovement()  {  //установка значений с joystick'a  float horizontal = joystick.Horizontal;  float vertical = joystick.Vertical;  //проверка на поворот по горизонтали  CheckToFlip(horizontal);  //если в руках предмет, то проверяем и его на поворот  if (\_isHandling)  {  CheckToFlipThing(horizontal, extinguisher);  }  //движение персонажа  \_isWalking = horizontal > 0 || horizontal < 0 || vertical > 0 || vertical < 0;  Vector3 direction = new Vector3(horizontal, vertical, 0f);  transform.Translate(direction \* \_speed \* Time.deltaTime);  }  //обновление анимации  private void UpdateAnimation()  {  //если мы идем и предмет не взят  if (\_isWalking && !\_isHandling)  {  \_playerAnimator.SetBool("isExtinguisher", false);  \_playerAnimator.SetBool("isWalkingWithThing", false);  \_playerAnimator.SetBool("isWalking", true);  }  //если мы не идем и предмет не взят  else if (!\_isWalking && !\_isHandling)  {  \_playerAnimator.SetBool("isWalkingWithThing", false);  \_playerAnimator.SetBool("isExtinguisher", false);  \_playerAnimator.SetBool("isWalking", false);  }  //если мы идем и предмет взят  else if (\_isHandling && \_isWalking)  {  \_playerAnimator.SetBool("isWalking", false);  \_playerAnimator.SetBool("isExtinguisher", true);  \_playerAnimator.SetBool("isWalkingWithThing", true);  }  //если мы взяли предмет и не идем  else if (\_isHandling && !\_isWalking)  {  \_playerAnimator.SetBool("isWalkingWithThing", false);  }  } |

Листинг 3 – Проверки на поворот объектов класса Player

|  |
| --- |
| //проверка на поворот персонажа  public void CheckToFlip(float horizontal)  {  Vector3 localScale = transform.localScale;  if (horizontal < 0 && localScale.x > 0) localScale.x \*= -1;  if (horizontal > 0 && localScale.x < 0) localScale.x \*= -1;  transform.localScale = localScale;  }  //проверка на поворот вещи  private void CheckToFlipThing(float horizontal, GameObject thing)  {  Vector3 localScale = thing.transform.localScale;  if (horizontal < 0 && localScale.x > 0)  localScale.x \*= -1;  if (horizontal > 0 && localScale.x < 0)  localScale.x \*= -1;  thing.transform.localScale = localScale;  } |

### 2.7 Организация тестирования приложения

В процессе разработки необходимо тестирование приложения, для выявления ошибок и повышения производительности.

Тестирование осуществлялось на нескольких этапах. На этапе разработки производилась проверка работоспособности внутренних игровых объектов:

* правильное взаимодействие классов друг с другом;
* отсутствие синтаксических ошибок;

На этапе реализации производилось тестирование правильного отображения объектов:

* отображение материалов;
* корректное сохранение игры с возможностью продолжения;
* корректное работа диалоговой системы;
* сохранение пользовательских настроек игры;
* расположение элементов графического интерфейса игры;
* отсутствие логических ошибок.

Были найдены и устранены незначительные ошибки на различных этапах тестирования. Среди них:

* отсутсвие масштабируемости у графического интерфейса;
* некорректное отображение прозрачных материалов;
* некорректная генерация локации;
* отсутствие сохранения пользовательских настроек игры.

На этапе Beta-тестирования, была задействована группа человек в составе 4 активных пользователей. Изменения, которые внесены на этапе Beta-тестирования:

* исправление поведения персонажей;
* отображение меню;
* генерация локации;
* Работа логических задач.

### 3 Экономическая часть

### 3.1 Описание программного продукта

Програмный продукт представляет собой мобильную игру «Будь осторожен» для Android устройств.

Возможности программного продукта, доступные конечному пользователю:

Данное приложение несет исключительно обучающий характер и предназначена для сюжетного прохождения игрока.

Основные черты игры:

* удобный интерфейс;
* управление главным героем за счет джойстика;
* создание интересных мини игр внутри самой игры;
* звуковое сопровождение;
* наличие интересных диалогов;
* анимированные персонажи и объекты;
* возможность сохранения игрового прогресса.

Минимальные системные требования:

• Версия Android 5 и выше;

• Процессор - Qualcomm Snapdragon 636;

• Видеопроцессор - Adreno 509;

• Оперативная память – 6 гб;

• Свободного места – 1 гб;

Рекомендуемые системные требования:

• Версия Android 9 и выше;

• Процессор - Qualcomm Snapdragon 865;

• Видеопроцессор - Adreno 650;

• Оперативная память – 8 гб;

• Свободного места – 1 гб;

Предоставляется поддержка и сопровождение игры.

### 3.2 Описание качества программного продукта

**Функциональные возможности:**

* функциональная полнота;
* функциональная согласованность.

Пример важных функций данного приложения:

* возможность осуществлять управление главным героем посредством клавиатуры и мыши;
* возможность сохранять игровой прогресс с целью дальнейшего прохождения.

Данные функции выполняют критерий правильности функций в соответствии назначения продукта.

**Надежность** – приводится описание набора программных и организационных средств, обеспечивающих возможность сохранять качество функционирования программного продукта при установленных условиях за установленный период времени:

* средства проверки достоверности исходных данных (пользователь не вводит свои данные);
* средства зашиты от ошибок пользователя (пользователь имеет заранее выбранные клавиши управления и функционала приложения).

**Практичность** – приводится описание свойств, способствующих быстрому освоению, применению и эксплуатации программного продукта с минимальными трудозатратами с учетом характера решаемых задач и требований к квалификации обслуживающего персонала:

* интерфейса пользователя имеет простую структуру для понимания и не меняется в ходе всего прохождения.

**Эффективность** – приводится информация о характере поведения продукта, его степени удовлетворения потребности пользователя в обработке данных с учетом экономических, вычислительных и людских ресурсов:

* время отклика системы на запрос пользователя имеет первоначальный приоритет, так как данное приложения основано на умении игрока производить быстрые перемещения и действия.

**Мобильность** - означает способность продукта быть системой, переносимой из одной аппаратно-программной среды в другую:

* Данный продукт имеет возможность адаптации к различным условиям эксплуатации без применения дополнительных сервисов;

### 3.3 Расчет технико-экономической эффективности разработки ПС

**Оценка транзакционных функциональных типов для формы**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Транзакция** | **Тип** | **FTR** | **DET** | **Сложность** | **Коэффициент** |
| Сохранение прохождения | *EI* | *1* | *1* | Low | *7* |
| Открытие этапов на экране **«Этапы»** | EI | 1 | 1 | Low | *7* |
| Запуск этапа по нажатию на экране **«Этапы»** | EI | 1 | 1 | Low | 7 |
| Кнопка **«Продолжить»** загружает соответственный этап | EI | 1 | 1 | Low | 7 |
| Подбор предметов | EI | 1 | 1 | Low | 7 |

# 

**Расчет количества функциональных точек**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Категория функциональных типов** | **Простые** | **Средние** | **Сложные** | **Кол-во точек** |
| Количество внутренних логических файлов | *17* | *5* | *3* | *25* |
| Количество входных элементов | *4* | *8* | *6* | *18* |
| **Количество функциональных точек (UFP)** | | | | 43 |

Таблица - Факторы выравнивания

|  |  |  |
| --- | --- | --- |
| **№** | **Наименование фактора** | **Значение** |
| 1 | Обмен данными | 1 |
| 2 | Распределенная обработка данных | 5 |
| 3 | Производительность | 5 |
| 4 | Эксплуатационные ограничения | 3 |
| 5 | Частота транзакций | 5 |
| 6 | Ввод данных в режиме «онлайн» | 3 |
| 7 | Эффективность работы конечных пользователей | 5 |
| 8 | Онлайновое обновление | 5 |
| 9 | Сложная обработка | 5 |
| 10 | Повторное использование | 3 |
| 11 | Простота установки | 4 |
| 12 | Простота эксплуатации | 5 |
| 13 | Количество возможных установок на различных платформах | 5 |
| **Суммарное значение коэффициентов (М)** | | 54 |

UFP = 43

sum M = 54

VAF = (0,65 + (sum М \* 0,01)) = (0,65 + (43 \* 0,01)) = 1,08

AFP = UFP \* VAF = 43 \* 1,08= 46,44

Оценка дипломного проекта методом PERT.

Входом для данного метода оценки служит список элементарных пакетов работ.

Перечень элементарных пакетов работ данного мобильного приложения для проведения оценки:

* разработка игровых локаций (9 шт.);
* разработка новых методов (16 шт.);
* разработка дизайна приложения в Figma (4 шт);

Диапазон неопределенности достаточно охарактеризовать тремя оценками:

* Mi – Наиболее вероятная оценка трудозатрат;
* Oi - Минимально возможные трудозатраты на реализацию пакета работ;
* Pi – Максимально возможные трудозатраты.

Проведем оценку для лучших, худших и наиболее вероятных случаев.

Оценку средней трудоемкости по каждому элементарному пакету можно определить по формуле:

Ei = (Pi+4\*Mi + Oi)/6, (1)

где Pi – Максимально возможные трудозатраты,

Mi – Наиболее вероятная оценка трудозатрат,

Oi – Минимально возможные трудозатраты на реализацию пакета работ.

Разработка игровых локаций

Mi = 8

Oi =6

Pi =13

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (13+4\*8+6)/6= 8,5 чел. час

Разработка новых методов

Mi = 14

Oi = 11

Pi = 18

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (18+4\*14+11)/6=14,2 чел. час

Разработка дизайна приложения в Figma

Mi = 2

Oi = 1

Pi = 3

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (3+4\*2+1)/6= 2 чел. час

В таблице 1 показаны оценки наиболее вероятного и ожидаемого случаев.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Компоненты | Наиболее вероятный | Минимально возможный | Худший случай | Ожидаемый случай |
| Разработка элементов интерфейса | 8 | 6 | 13 | 8,5 |
| Разработка новых методов | 14 | 11 | 18 | 14,2 |
| Разработка дизайна приложения | 2 | 1 | 3 | 2 |
|  |  |  |  |  |
| Итого | 24 | 18 | 34 | 24.7 |

Для расчета среднеквадратичного отклонения используется формула:

CKOi = (Pi - Oi)/6, (2)

где Pi – Максимально возможные трудозатраты

Oi - Минимально возможные трудозатраты на реализацию пакета работ.

Для разработки игровых локаций, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (13-6)/6=1,16 чел. час

Для разработки новых методов, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (18-11)/6=1,16 чел. час

Для разработки дизайна приложения в Figma, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (3-1)/6= 0,33 чел. час

Суммарная трудоемкость проекта может быть рассчитана по формуле:

Е = Σ Ei

Е = Σ Ei = 10\*8,5+3\*14,2+10\*2=127,5

Cреднеквадратичное отклонение для оценки суммарной трудоемкости будет составлять:

CKO = √ (∑CKOi)^2

где CKO – среднеквадратичное отклонение

Cреднеквадратичное отклонение для оценки суммарной трудоемкости рассчитана по формуле (4)

CKO = √ (∑CKOi)^2 =

=√(█(10\*〖1,16〗^2+10\*〖1,16〗^2+6\*〖0,33〗^2@))= 4,3 чел. час

Оценка суммарной трудоемкости проекта, которая с вероятностью 95% точности расчета, можно применить формулу:

∑〖95%=E+2\*CKO〗

где CKO = Cреднеквадратичное отклонение

E = Ei

∑〖95%=E+2\*CKO〗 = (127,5+2\*4,3) = 136,1 чел. чаc

Полученную оценку трудоемкости кодирования необходимо умножить на четыре, поскольку помним, что кодирование составляет только 25% общих трудозатрат проекта. Поэтому суммарная трудоемкость нашего проекта составит, приблизительно = 544,5 чел.\*час.

В месяц сотрудник будет работать по проекту, примерно 144 чел.\*час/мес.

Следовательно, трудоемкость проекта в человеко-месяцах составит, приблизительно 544,5 /144 = 3,8

Тогда согласно формуле Б.Боэма оптимальная продолжительность проекта составит:

T = 2.5 \* (3,8) ^ 1/3=3,2 месяцев.

Средняя численность команды –3,8/3,2 = 1,2 человек.

### 3.4 Расчет затрат на разработку проекта

Дополнительные расчета будут произведены для проекта на языке C#.

Расчет количества функциональных точек произведен (f =43)

Размерность программного обеспечения для конкретного языка программирования определим с учетом нормативов, представленных в таблице по формуле:

R = f ×LOC = 43×53 =2279.

Оценка трудозатрат проводится с помощью степенной функции базовой модели COCOMO.

Значения параметров A и E определяются из таблицы коэффициентов математической модели оценки трудозатрат в зависимости от типа программной системы.

A = 3,6

R = 2279

E = 1,2

T = ((A\*R^E) / 12) = ((3,6 \* 22,79^1,2) / 12) = 0,68 (чел. – месяцев)

D = 12

N = T/D = 0,68 / 12 = 0,204

Таким образом, метод функциональных точек определил следующие основные технико-экономические показатели:

- трудозатраты на разработку системы за 12 месяцев составят 0,68 чел.- месяцев;

- необходимые людские ресурсы при реализации системы за 12 месяцев –0,204 чел.

### Заключение

В результате проектирования созданы следующие компоненты:

* игра «Будь осторожен»;
* пояснительная записка объемом 53 листа;
* презентация.

Для разработки игры использованы средства и среды разработки: компилятор Framework .NET и VisualStudioC# 2019, Unity. NET Framework - это программная платформа, в среде которой выполняется построение приложений. Visual Studio 2019 - линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Unity – среда для разработки игр на большинство существующих платформ.

Данными ресурсами создано игровое мультимедийное приложение с возможностью дальнейшего улучшения графики, системы улучшений персонажа, добавление новых локаций.

В дальнейшем планируется реализация и других локаций. Данные локации являются лишь частью всей истории. Ведь она так же будет иметь еще и продолжение в виде второй и третьей части. Там раскроется сюжет с других сторон, будут опробованы новые игровые механики.

### Используемые источники информации

1. ГОСТ Р ИСО/МЭК 12207-99 «Процессы жизненного цикла программных средств».
2. ГОСТ Р ИСО 9000-2008 «Системы менеджмента качества».
3. Голощапов Алексей Леонидович. MICROSOFT VISUAL STUDIO 2020 - 2-е изд.,перераб.и доп. - СПб.: БХВ-Петербург, 2020. - 448с.
4. Информация о всех видах игровых движков, url: https://blackcaviar.games/obzor\_igrovyh\_dvizhkov (16.02.2022);
5. Еще о движках, url: https://ru.wikipedia.org/wiki (18.02.2022);
6. Информация о Unity, url: https://docs.unity3d.com/Manual/index.html (21.02.2022);
7. Больше про Unity, url: https://docs.unity.com/ (15.02.2022);
8. Структура мобильной игры, url: https://gopractice.ru/market/top\_grossing\_appstore/ (23.02.2022);
9. Примеры описания различных структурных элементов, url: https://koloro.ua/blog/brending-i-marketing/razrabotka-pribylnoy-i-prostoy-mobilnoy-igry.html (25.02.2022);
10. Пример архитектуры игры, url: https://www.cyberforum.ru/game-dev/thread932889.html (03.03.2022);
11. Архитектура UI, url: https://dtf.ru/gamedev/14464-sozdanie-arhitektury-ui-na-unity (07.03.2022);
12. Про жанры игр, какие бывают, url: https://cubiq.ru/zhanry-kompyuternyh-igr/ (14.03.2022);
13. Классификация компьютерных игр, url: https://ru.wikipedia.org/wiki/Классификация\_компьютерных\_игр (18.03.2022);
14. Виды и типы компьютерных игр, url: http://batona.net/51048-vidy-tipy-i-raznovidnosti-sovremennyh-kompyuternyh-igr-11-foto.html (21.03.2022);
15. Особенности жанров игр, url: https://www.playground.ru/misc/news/kompyuternye\_igry\_klassifikatsiya\_i\_osobennosti-323484 (25.03.2022);
16. Как делать сюжет игры, url: https://dtf.ru/u/306047-matvey-aniskin/294634-kak-pridumat-syuzhet-igre-moy-opyt-napisanie-igrovogo-scenariya (27.03.2022);
17. Про роль сценариста, важно для написания сценария, url: https://habr.com/ru/company/vk/blog/197152/ (30.03.2022);
18. Примеры создания локаций для игр, url: https://dtf.ru/gamedev/779829-kak-sozdat-stilizovannoe-3d-okruzhenie (02.04.2022);
19. Этапы создания локаций, url: https://render.ru/ru/XYZ/post/17070 (10.04.2022);
20. Про создание игрового окружения в AAA-проектах, url: https://www.playground.ru/misc/news/kak\_sozdaetsya\_igrovoe\_okruzhenie\_v\_aaa\_igrah-217950 (13.04.2022);
21. Какие языки программирования используются для создания игр, url: https://gb.ru/blog/yazyki-programmirovaniya-dlya-sozdaniya-igr/ (15.02.2022);
22. Какой язык лучше выбрать для создания игр, url: https://coop-land.ru/helpguides/blogs/20594-ya-hochu-sdelat-igru-kakoy-yazyk-programmirovaniya-mne-vyuchit.html (15.02.2022);
23. Каталог игр, для понимания примеров проектов, url: https://unity3dschool.com/game-catalog/ (17.02.2022);