|  |  |  |  |
| --- | --- | --- | --- |
| ПРАВИТЕЛЬСТВО САНКТ-ПЕТЕРБУРГА  КОМИТЕТ ПО НАУКЕ И ВЫСШЕЙ ШКОЛЕ | | | |
|  | | | |
| Санкт-Петербургское государственное бюджетное  профессиональное образовательное учреждение  «ПЕТРОВСКИЙ КОЛЛЕДЖ»  (СПб ГБПОУ «Петровский колледж») | | | |
|  | | | |
| Отделение информационно-промышленных технологий и судостроения | | | |
|  | | | |
| ДИПЛОМНЫЙ ПРОЕКТ | | | |
|  | | | |
| **Разработка модулей управления уровнями игрового приложения в жанре Cyberpunk** | | | |
|  | | | |
|  | Выполнил: | | |
| Балев Александр Эдуардович | | |
| Студент | 3602 | группы |
| Специальность | 09.02.03 Программирование в компьютерных системах | |
| Руководитель: | | |
| *(Личная подпись)* | М.А. Ерина | |
| Консультант: | | |
| *(Личная подпись)* |  | |
| Нормоконтролер: | | |
| *(Личная подпись)* | Г.Н. Фёдорова | |
|  | | | |
|  | | | |
| Санкт-Петербург, 2020 | | | |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc42595417)

[1 Аналитическая часть 5](#_Toc42595418)

[1.1 Постановка задачи 5](#_Toc42595419)

[1.2 Техническое задание на разработку 12](#_Toc42595423)

[2 Проектная часть 13](#_Toc42595424)

[2.1 Проектирование программного продукта 13](#_Toc42595425)

[2.2 Проектирование игрового интерфейса 23](#_Toc42595430)

[2.3 Реализация основных функций программного продукта 25](#_Toc42595431)

[2.4 Тестирование программного продукта 35](#_Toc42595432)

[3 Экономическая часть 40](#_Toc42595433)

[3.1 Цели, задачи и алгоритм выполнения экономического части дипломного проекта 40](#_Toc42595434)

[3.2 Расчёт показателей себестоимости разработки программного или иного информационного продукта и формирование сметы затрат 41](#_Toc42595435)

[3.3 Расчёт сметы затрат на разработку мобильного игрового приложения. 46](#_Toc42595445)

[ЗАКЛЮЧЕНИЕ 59](#_Toc42595457)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 60](#_Toc42595458)

[ПРИЛОЖЕНИЕ А – Концепт документ 62](#_Toc42595459)

[ПРИЛОЖЕНИЕ Б – Дизайн-документ 62](#_Toc42595461)

[ПРИЛОЖЕНИЕ В – Листинги 90](#_Toc42595463)

ВВЕДЕНИЕ

Благодаря широкой доступности персональных компьютеров, они стали появляться буквально в каждом доме, офисе и производстве, изменяя устойчивые социально-культурные взгляды на работу и отдых человека. С момента создания первого компьютера в 1941 году технологии революционно продвинулись вперёд, и сейчас, в цифровую эпоху, практически все процессы зависят от них. Современные технологии позволяют не только хранить отчетности, но и обрабатывать огромные пакеты данных, помогая людям решать сложнейшие задачи. Вместе с необходимыми для человечества процессами, компьютерный мир также перебрались и игры. Люди с древнейших времен придумывали способы, как развлечь себя, соревнуясь друг с другом на Олимпе, сражаясь в гладиаторских поединках, а также играя в более миролюбивые «Казаки-разбойники», «салочки», шахматы и т.д. Это был лишь вопрос времени, как быстро человечество, чья жизнь становилась всё более компьютеризированной и автоматизированной, спроецирует на процессоре всем полюбившиеся виды спортивно-развлекательных игр.

Первые игры, которые появились на компьютерах, были довольно просты: вся их суть заключалась в достижении максимального количества очков. Но и этого было достаточно, чтобы собирать у игровых автоматов толпы людей и фанатов, соревнующихся между собой, ставя новые рекорды. С развитием новых технологий и возможностей игровой механики, компьютерные развлечения приобретали идею, сюжет и цель, которую разработчики хотели донести до игрока или выработать у него определённые навыки: развитие внимания, реакции, логики. Благодаря своей продуманности, слаженности сюжета и картинки, пользе и удовольствию, которую получали потребители, игры стали приносить больше прибыли, команды разработчиков стали расти и объединяться в большие компании, но оставались и энтузиасты, создававшие игры сами, инди-разработчики.

Компьютерная игра, как и любая другая программа, это программный код, написанный программистами. Написания качественного продукта требует много ресурсов, времени и знаний, это довольно сложная и интересная задача, которая должна принести не мало опыта в данной сфере разработки. Это и послужило основанием для выбора данной темы дипломного проекта.

Целью дипломного проекта является разработка модулей генерации и управления уровнями игрового приложения.

Для достижения этой цели были поставлены следующие задачи:

* Разработать модули генерации и управления уровнями
* Изучить проектирование программного продукта для компьютерной игры
* Получить опыт проектирования компьютерной игры
* Изучить современные средства разработки
* Показать важность компьютерных игр в современной жизни
* Получить опыт разработки компьютерных игр
* Получить опыт совместной разработки

# Аналитическая часть

## Постановка задачи

### Описание предметной области

С каждым годом выручка от продаж видеоигр растет быстрыми темпами. Ресурс gameindustry.biz каждый год выкладывают статистику, которая наглядно это демонстрирует (Рисунки 1-3). Доля компьютерных видеоигр в них тоже увеличивается с каждым годом. Российский рынок же занимает только малую часть от него. По статистике ресурса pwc.ru на 2018 год общий объем продаж составлял $134,9 млрд и российский рынок в нем состоял из $1,9 млрд. К сожалению, игровая индустрия в России очень мала, не так много компаний занята разработкой игр, в основном идет разработка мобильных игровых приложений. Но эти показатели не сказываются на продаже покупок игр среди россиян. Наибольшей популярностью в России пользуются массовые многопользовательские ролевые игры, но также игроки готовы покупать и одиночные видеоигры.

Исходя из этого была поставлена задача о написании модулей генерации и управления уровнями для компьютерной игры.

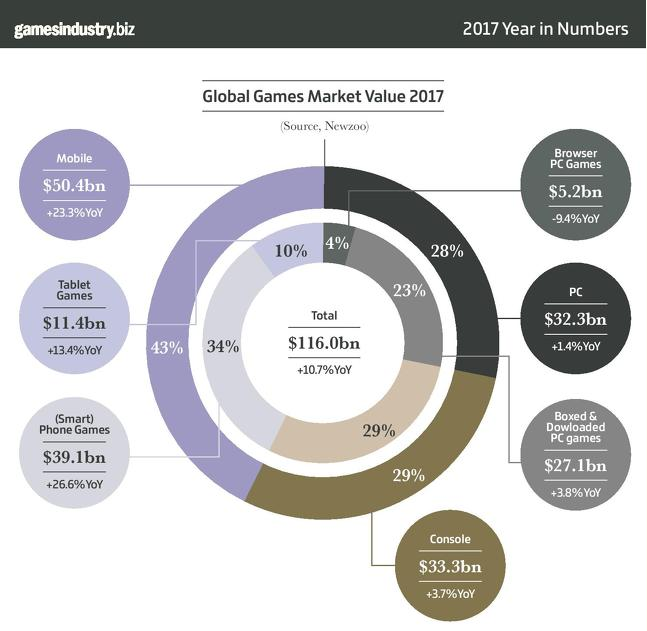


Рисунок 1 − Продажи игр за 2017 год

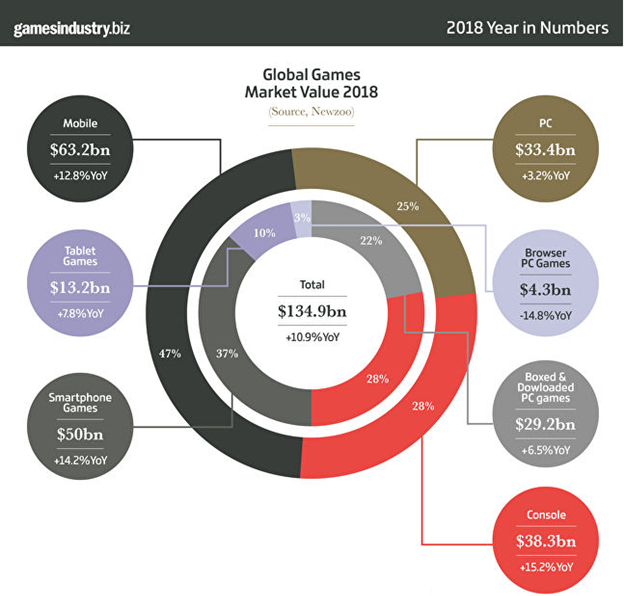


Рисунок 2 − Продажи за 2018 год

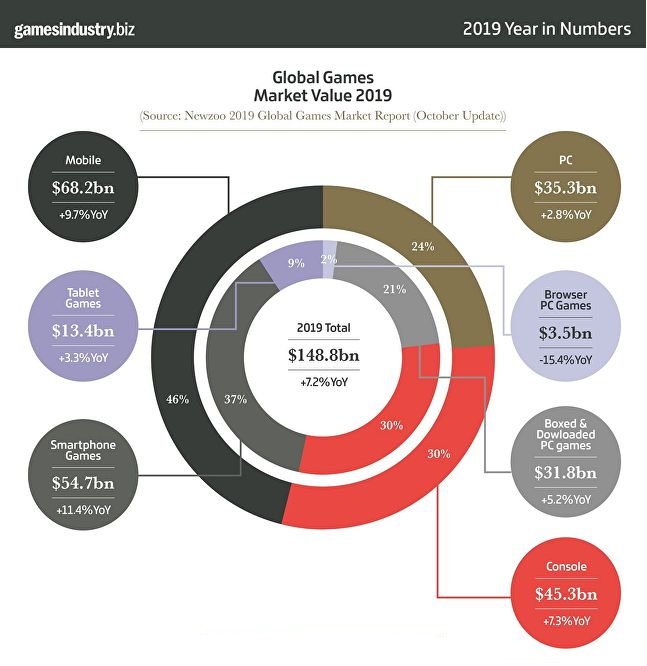


Рисунок 3 − Продажи за 2019 год

В мире видеоигровой индустрии существует множество игровых жанров и их комбинаций. Выбор жанра разрабатываемой игры должен складываться из нескольких параметров, таких как: популярность игрового жанра на рынке, целевая аудитория будущей игры, необходимость в будущей поддержке и дополнительном контенте. Также, немаловажен сегмент рынка, в котором будет вестись разработка игрового приложения.

Сегмент рынка, в котором будет вестись разработка – инди-игры. Инди-игра − компьютерная игра, созданная отдельным разработчиком или небольшим коллективом без финансовой поддержки издателя компьютерных игр.

Для верного выбора наиболее подходящего жанра, были проанализированы некоторые популярные жанры, которые будут приведены далее.

**Action** – жанр компьютерных игр, в которых успех игрока в большой степени зависит от его скорости реакции и способности быстро принимать тактические решения. Действие таких игр развивается очень динамично и требует напряжения внимания и быстрой реакции на происходящие в игре события. При этом в качестве основного средства прогресса в игре, как правило, используется какое-либо оружие.

Яркими представителями жанра являются:

* Серия игр Assassin’s creed
* Серия игр GTA
* Серия игр Call of Duty
* Серия игр BioShock

**Fighting −** жанр компьютерных игр, имитирующих рукопашный бой малого числа персонажей в пределах ограниченного пространства, называемого ареной. Файтинги близки к играм жанра «Избей их всех», однако между ними существуют различия. Так, в большинстве файтингов игроку не требуется перемещаться по длинному уровню и нельзя выйти за границы арены, а бой состоит из нечётного числа отдельных раундов и не является непрерывным. Менее значительными и необязательно присутствующими признаками жанра являются использование многочисленных шкал для изображения жизненно важных показателей персонажей и прорисовка бойцов на арене в профиль.

Яркими представителями жанра являются:

* Серия игр Street Fighter
* Серия игр Mortal Combat
* Серия игр Tekken

**Platform** − жанр компьютерных игр, в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня. Некоторые предметы, называемые пауэр-апами ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *power-up*), наделяют управляемого игроком персонажа особой силой, которая обычно иссякает со временем (к примеру: силовое поле, ускорение, увеличение высоты прыжков). Коллекционные предметы, оружие и «пауэер-ап» собираются обычно простым прикосновением персонажа и для применения не требуют специальных действий со стороны игрока. Реже предметы собираются в «инвентарь» героя и применяются специальной командой.

Яркими представителями жанра являются:

* Super Meat Boy
* Cuphead
* Sonic the Hedgehog

**Role-playing или RPG** − жанр компьютерных игр, основанный на элементах игрового процесса традиционных настольных ролевых игр. В ролевой игре игрок управляет одним или несколькими персонажами, каждый из которых описан набором численных характеристик, списком способностей и умений; примерами таких характеристик могут быть хит-пойнты ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) hit points, HP), показатели силы, ловкости, интеллекта, защиты, уклонения, уровень развития того или иного навыка и т. п..

Яркими представителями жанра являются:

* Серия игр Fallout
* Серия игр Ultima
* Серия игр The Elder Scrolls

**Strategy** − жанр компьютерных игр; игры данного жанра характеризуются тем, что игроку для достижения цели необходимо применять стратегическое мышление, и оно противопоставляется быстрым действиям и реакцией, которые, как правило, не обязательны для успеха в таких играх. Стратегические игры бывают абстрактные (Ataxx), настольные (шахматы, шашки, «Монополия»), с симуляцией менеджмента (M.U.L.E., Spaceward Ho!) и другие.

Яркими представителями жанра являются:

* Stronghold
* Серия игр StarCraft
* Серия игр Heroes of Might and Magic

**Roguelike** − жанр компьютерных игр, поджанр компьютерных ролевых игр. Характерными особенностями классического roguelike являются генерируемые случайным образом уровни, пошаговость и необратимость смерти персонажа — в случае его гибели игрок не может загрузить игру и должен начать её заново. Многие roguelike выполнены в декорациях эпического фэнтези под сильным влиянием настольных ролевых игр наподобие Dungeons & Dragons.

**Action RPG** − поджанр компьютерных ролевых игр, в котором ролевая составляющая сочетается с элементами action. Другими словами, если в классической ролевой игре успех действий персонажа во время боевой ситуации зависит от персональных характеристик героя, то в Action/RPG не меньшую роль играет и скорость реакции самого игрока.

Рассмотрев все эти жанры, было принято решение разрабатывать игровое приложение в жанре Action RPG, жанр, появившийся благодаря смешиванию Action и RPG, с элементами Roguelike, поджанром RPG.

### Анализ существующих разработок

Как таковых аналогов проектируемого игрового приложения не присутствуют на рынке, но есть приложения, которые имеют схожие механики.

Одна из таких игр − **Enter The Gungeon**. Она совпадает с данным проектом тем, что в ней также используется генерация случайных уровней, на них построена вся игра. Игрок продвигается по новым уровням сражаясь с различными противниками, отличающихся между собой по количеству здоровья, внешнему виду и умениями. Данная игра распространяется на ПК и игровых консолях. Скриншот игры показан на рисунке 4.



Рисунок 4 − Геймплей игры Enter The Gungeon

**The Binding of Isaac** − данная игра также похожа своей механикой случайной генерации уровней, на чем и был поставлен упор разработчиками. Имея более простой дизайн, игра смогла покорить сердца многих игроков. Также распространяется на ПК и игровые консоли. Пример геймплея можно увидеть на рисунке 5.



Рисунок 5 − Геймплей игры The Binding of Isaac

### Выбор методики, технологии и инструментальных средств проектирования и разработки

Для реализации игрового приложения были рассмотрены следующие среды разработки: Unity, GameMaker Studio 2 и Unreal Engine. Проанализировав их, была выбрана Unity, как наиболее подходящая среда для разработки данных модулей. Для разработки приложений в Unity используется язык C# − простой, современный объектно-ориентированный язык программирования. Язык C# подходит для создания проектов разной сложности: от простеньких системных утилит до веб-порталов с миллионной аудиторией. Он охватывает 45% рынка игровых движков. Unreal Engine же использует язык C++ и является более тяжелой для компьютера программой, что не позволяет вести разработку продуктов на некоторых компьютерах. GameMaker Studio работает с Game Maker Language − языком специально разработанным для этой среды. Также GameMaker Studio − платное ПО, что отличает ее от остальных сред.

Благодаря платформе .NET Framework игры, разработанные на Unity, имеют кроссплатформенность.

Microsoft Visual Studio встроено в дистрибутив Unity как средство для написания скриптов, оно и было выбрано как IDE для разработки, как рекомендуемая разработчиками среда. Из ее возможностей можно выделить:

Подсветка синтаксиса

Авто дополнение кода

Встроенный отладчик

Сворачивание кода

Для проектирования модулей использовался язык моделирования UML, и среда для построения диаграмм StarUML.

Для работы в команде использовалась система контроля версий Git и бесплатный репзиторий GitHub.

## Техническое задание на разработку

Техническое задание на разработку продукта состоит из двух частей – концепт документа и дизайн документа. Они соответственно расположены в приложении А и приложении Б.

# Проектная часть

## Проектирование программного продукта

### Диаграмма прецедентов и спецификации

При проектировании модуля игрового приложения были созданы диаграммы вариантов использования. Данные диаграмма описывают основные функции, с которыми может взаимодействовать пользователь: функции противников, игровые условности, функции игровых объектов.

Диаграмма вариантов использования главного меню демонстрирует то, как пользователь может взаимодействовать с интерфейсом. На рисунке 6 представлена данная диаграмма.

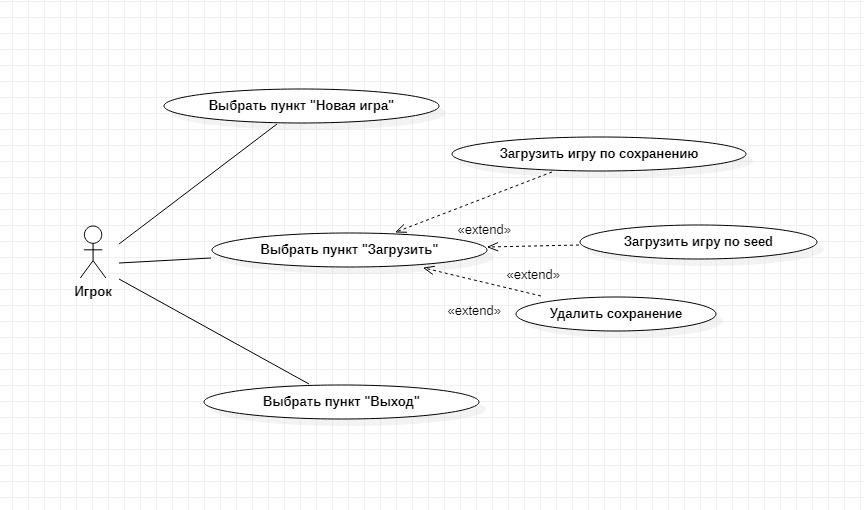


Рисунок 6 – Диаграмма вариантов использования для главного меню

Диаграмма вариантов использования для основного игрового цикла показывает то, какие возможности есть у игрока по мере прохождения игры. Диаграмма представлена на рисунке 7.

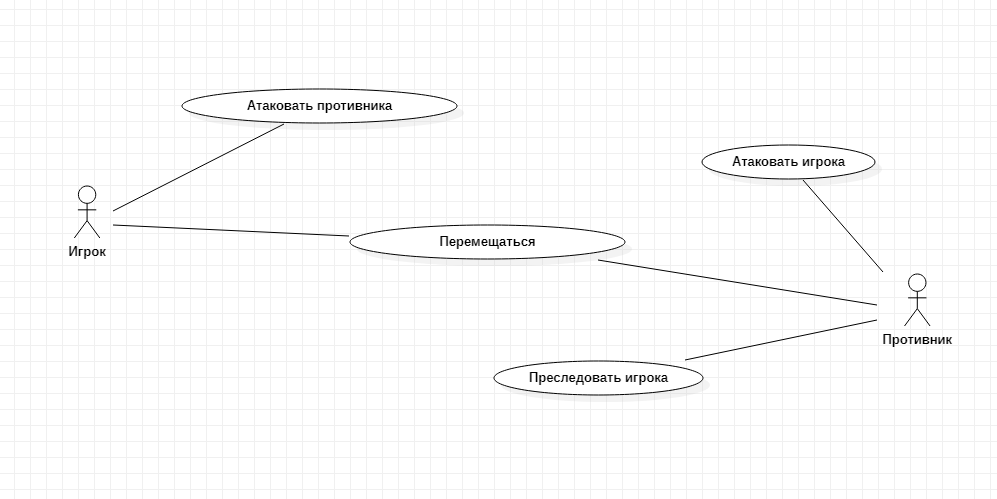


Рисунок 7 – Диаграмма вариантов использования для основного игрового цикла

### Алгоритмы генерации уровней

Для генерации случайных уровней были рассмотрены следующие алгоритмы: с использованием BSP-деревьев, туннелированние и на основе клеточных автоматов.

#### BSP-деревья

BSP-деревья – метод рекурсивного разбиения евклидова пространства в выпуклые множества и гиперплоскости. В результате объекты получают представление в виде структуры данных.

Реализация самого алгоритма представляет собой разделение первоначальной области на множество частей, выделения пространства под комнаты в этих частях, и последующее их соединение между собой (Рисунки 8-9).

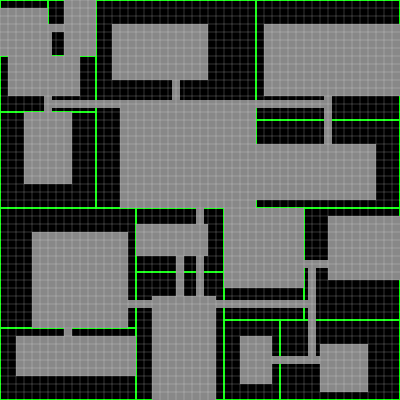


Рисунок 8 − Пример генерации уровня

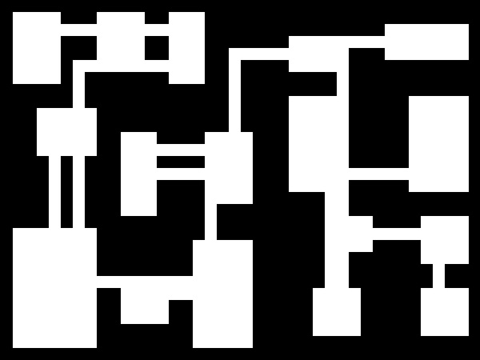


Рисунок 9 − Пример генерации уровня

#### Туннелированние

Алгоритмы туннелирования сами как-бы «прокапывают» себе коридоры и комнаты, как это бы делал человек. После определения начальной точки, по заранее определенному полю, начинаются создаваться тунели и комнаты, создавая множество маршрутов (Рисунки 10-11).

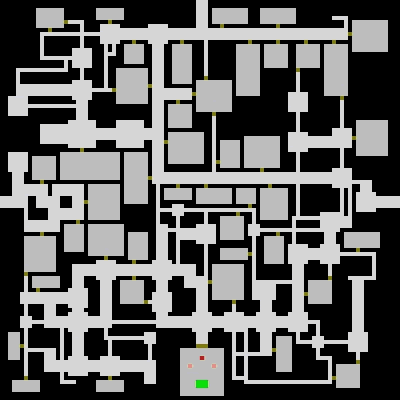


Рисунок 10 − Пример генерации уровня

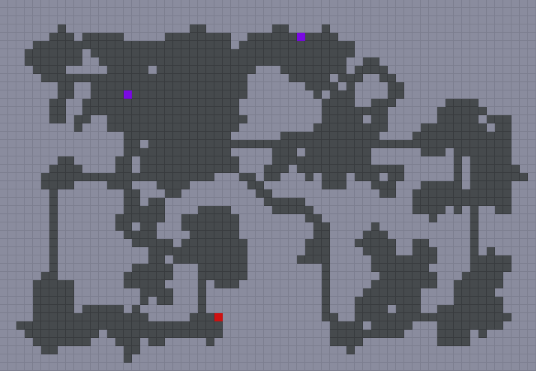


Рисунок 11 − Пример генерации уровня

#### Клеточные автоматы

Клеточный автомат — дискретная модель, изучаемая в математике, теории вычислимости, физике, теоретической биологии и микромеханике. Включает регулярную решётку ячеек, каждая из которых может находиться в одном из конечного множества состояний, таких как 1 и 0. Решетка может быть любой размерности. Для каждой ячейки определено множество ячеек, называемых окрестностью. К примеру, окрестность может быть определена как все ячейки на расстоянии не более 2 от текущей (окрестность фон Неймана ранга 2). Для работы клеточного автомата требуется задание начального состояния всех ячеек и правил перехода ячеек из одного состояния в другое. На каждой итерации, используя правила перехода и состояния соседних ячеек, определяется новое состояние каждой ячейки. Обычно правила перехода одинаковы для всех ячеек и применяются сразу ко всей решётке.

Как пример реализации клеточных автоматов, на локации могут расставляться множество точек со своими окрестностями и после изменения состояния клеток в окрестности, выходит локация. В отличие от других способов в этом разработчику после генерирования карты необходимо самостоятельно обеспечить соединения, потому что у некоторых алгоритмов велика вероятность создания разделённых областей (Рисунки 12-13).

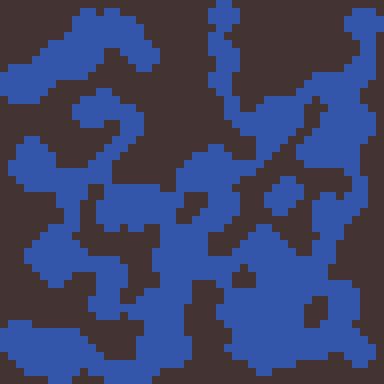


Рисунок 12 − Пример генерации уровня



Рисунок 13 − Пример генерации уровня

После анализа указанных алгоритмов, для разработки был выбран алгоритм с использованием туннелирования, так как алгоритм на основе клеточных автоматов может создавать замкнутые пространства, не связанные между собой. Уровни, созданные на алгоритмах с использованием разбиения пространства с помощью деревьев, показались же менее привлекательными для исследования игроком.

Разработанный алгоритм, основанный на туннелированнии, работает следующим образом. В начале выбирается максимальное количество квадратов, которое может быть создано на одном пути и задается текущее количество квадратов созданного на маршруте (приравнивается нулю). От размера квадрата зависит чем она будет являться, размер же задается радиусом вокруг центральной клетки квадрата. Коридором будет являться квадрат с радиусом 1, а комнатой от 3ех клеток. После этого создается начальная комната с радиусом 1 вокруг центральной клетки. Сразу над ней же создается еще одна, для задания направления. На этой комнате создается новый маршрут, который выбирает в каком направлении будет создаваться комната: спереди, слева или справа. На каждое направление есть шанс 15% и есть шанс, что выпадет каждый из этих маршрутов. Но также есть и шанс, что не один из них не выпадет, в таком случае он продолжит продвигаться в том направлении в котором и двигался до этого. Если при построении маршрута он наткнется сам на себя или на другие пути, то создастся новый путь. Размер каждого пути рассчитывается из разницы максимальной длины пути и созданного количества комнат на текущем маршруте (если это первоначальный маршрут, то между максимальной длиной и нулем). При создании квадрата есть шанс 15% что создастся комната, в ином случае создается коридор. В коридоре могут появиться 1 противник и 1 препятствие, где шанс появления противника равен 30% а препятствия 45%. В комнатах же количество препятствий может быть от 3 до Х + 2, а количество противников от 1 до Х – 2, где Х – радиус созданной комнаты. По окончанию построения маршрутов, в последнем построенном создается плита перехода на следующий уровень. После этого, вокруг созданных комнат создаются стены (Рисунок 14).



Рисунок 14 − Пример уровня созданного по алгоритму

### Диаграмма классов

Была разработана диаграмма классов и пакетов, для представления того, какие классы существуют в приложении. Классы были объединены в пакеты, которые группируют их по функционалу. Данная диаграмма представлена на рисунке 15.

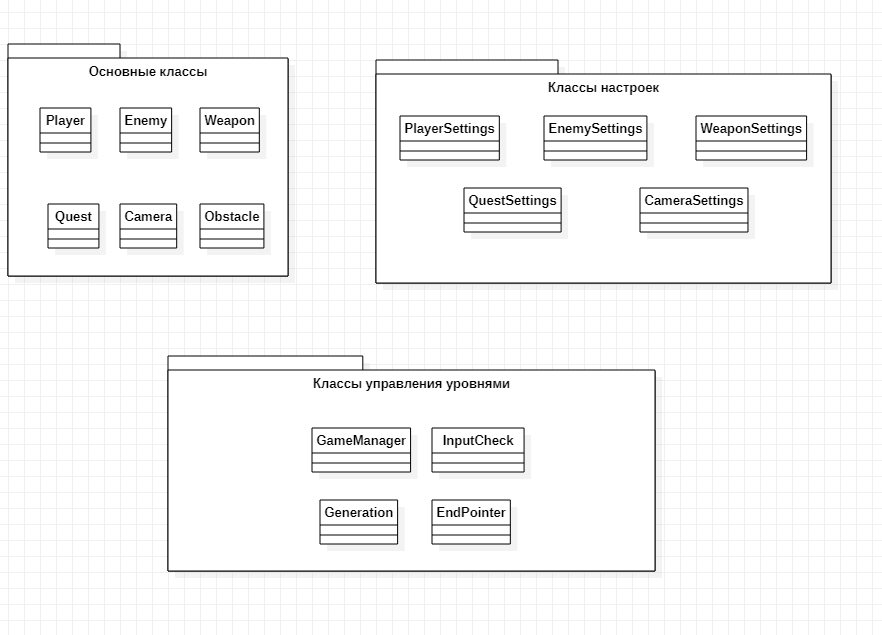


Рисунок 15 − Диаграмма классов

### Диаграммы деятельности и последовательности

Для демонстрации работы генерации мира была разработана диаграмма последовательности генерации мира (Рисунок 16). Она упрощённо демонстрирует работу класса Generation, который и создает новые уровни. Его код представлен в листинге 7 (Приложение В).

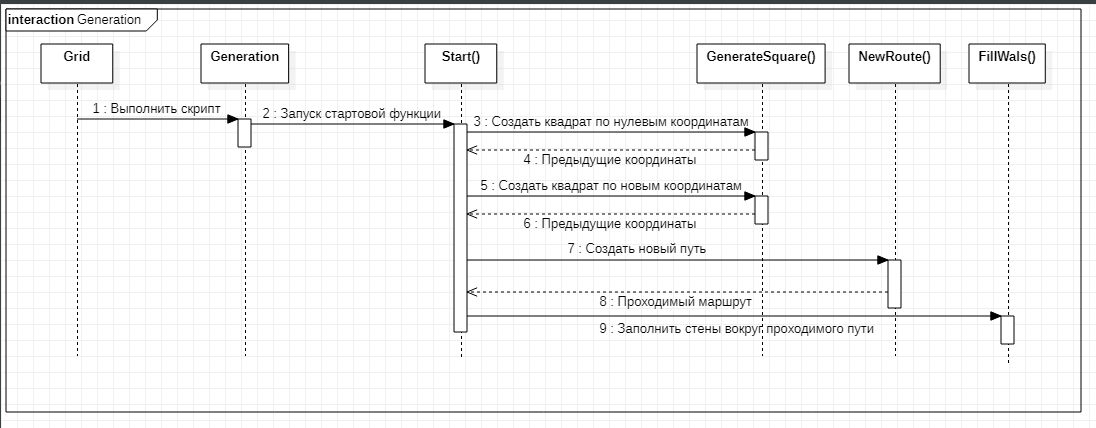


Рисунок 16 – Диаграмма последовательности класса Generation

На рисунке 17 продемонстрирована диаграмма последовательности класса InputCheck. Этот класс необходим для передвижения персонажа и изменения сцен, например, остановки игры на паузу. Его код представлен в листинге 7 (Приложение В).

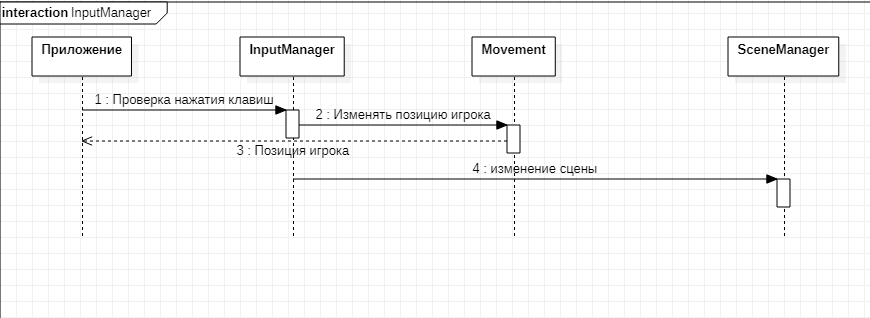


Рисунок 17 – Диаграмма последовательности класса InputManager

В игре существует компас, который должен помогать игроку искать выход. Скрипт получает положение игрока и выхода, после сего высчитывает поворот стрелки компаса (Рисунок 18). Код скрипта предоставлен в листинге 6 (Пункт 2.3).

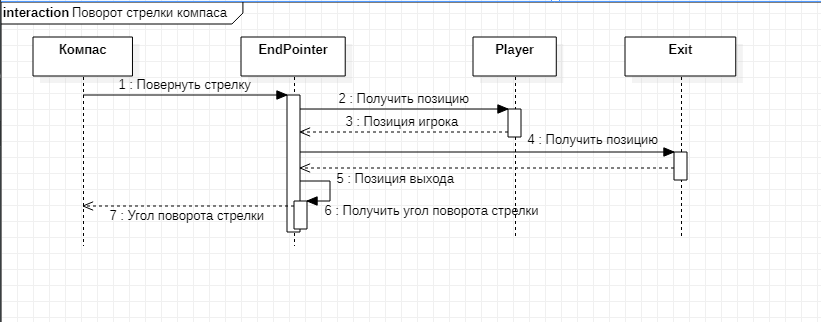


Рисунок 18 – Диаграмма последовательности прецедента поворота стрелки компаса

На рисунке 19 представлена диаграмма последовательности для прецедента передвижение. Данная диаграмма демонстрирует то, с помощью каких объектов и классов персонаж имеет возможность передвигаться. Реализация представлена в листинге 9, приложение В.

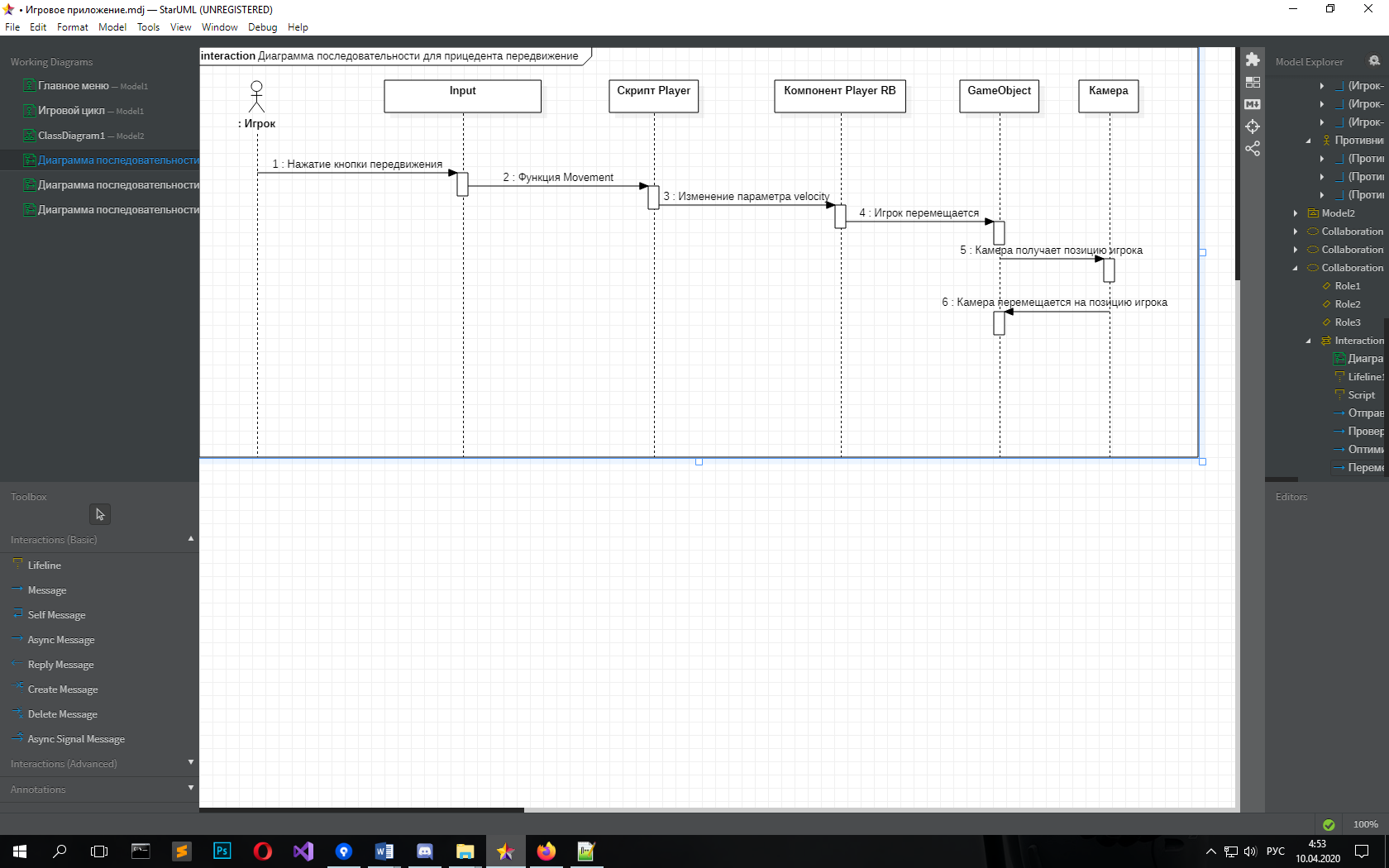


Рисунок 19 – Диаграмма последовательности для прецедента передвижение

На рисунке 20 представлена диаграмма последовательности для прецедента атака противника. Данная диаграмма демонстрирует, каким образом игрок сможет атаковать противника. Реализация представлена в листинге 10, приложение В.

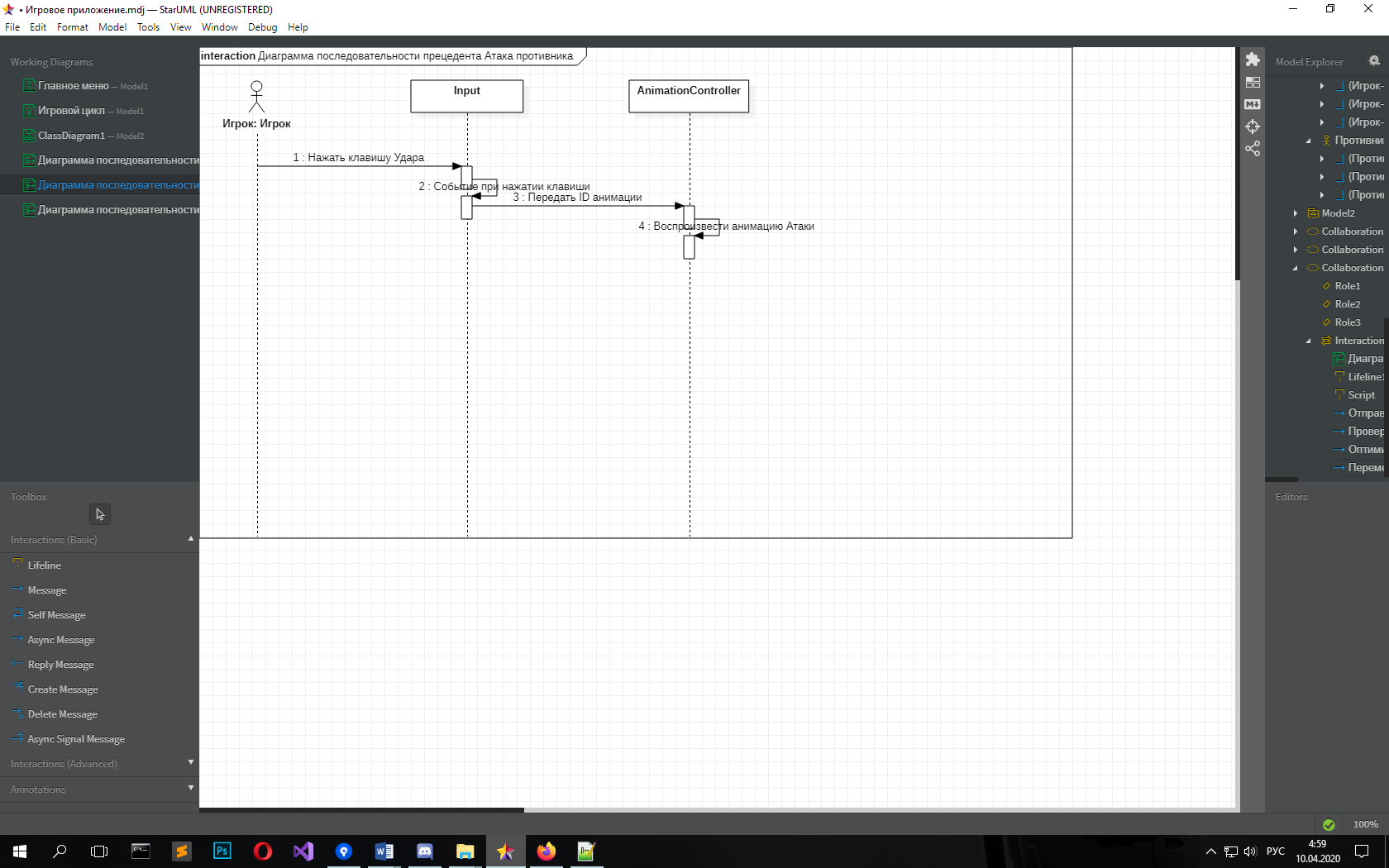


Рисунок 20 – Диаграмма последовательности для прецедента атака противника

Также, для помощи игроку в ориентировании на уровне, в игре существует миникарта. Она отображает небольшую область вокруг игрока, противников и препятствия. На рисунке 21 предоставлена диаграмма последовательности прецедента отображение миникарты. Сам код представлен в листинге 11, в приложении В.

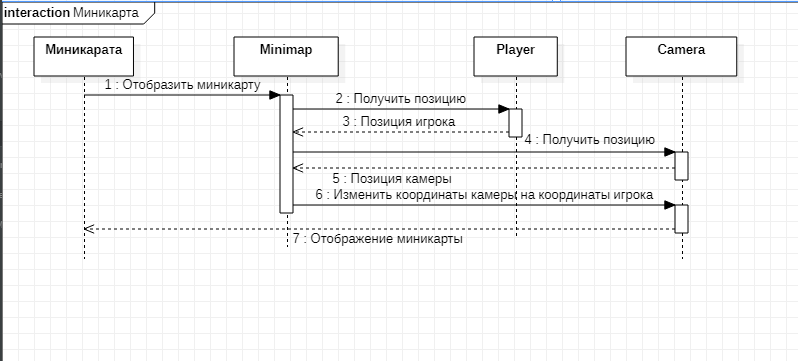


Рисунок 21 – Диаграмма последовательности прецедента отображение миникарты

## Проектирование игрового интерфейса

Макет пользовательского интерфейса был разработан в программе Microsoft Paint и был спроектирован с целью создать наиболее удобный интерфейс для пользования, где пользователю будет предоставлена только необходимые ему данные.

На рисунке 22 предоставлен макет главного меню игрового приложения. На нем располагаются кнопки начала игры, загрузки, настройки и выхода из игры. Так же в правом нижнем углу предполагается расположение контактов разработчиков.



Рисунок 22 – Эскиз главного меню приложения

На рисунке 23 предоставлен макет внутреигрового интерфейса пользователя. На нем отображены все необходимые пользователю показатели: иконка и имя героя, количество его здоровья, миникарта, слоты быстрого доступа, текущее оружие и количество патронов к нему.

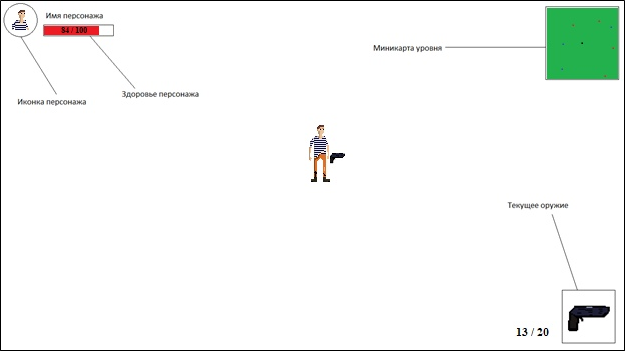


Рисунок 23 – Эскиз интерфейса игрового процесса

## Реализация основных функций программного продукта

В ходе разработки игрового приложения были разработаны два основных модуля управления уровнями: модуль генерации случайного уровня и игровой менеджер.

Игровой менеджер необходим для:

* Начала новой игры
* Загрузки уровней
* Сохранения игры
* Загрузки игры
* Удаления сохранения
* Загрузки игры по seed
* Остановки игры на паузу
* Продолжения игры с паузы
* Перезагрузки уровня
* Выхода из игры

Реализация игрового менеджера представлен в листинге 1.

Листинг 1 – Игровой менеджер

public class GameManager : MonoBehaviour

{

public Grid grid;

public Canvas GUI;

public Canvas Loss;

public Canvas Paused;

public Canvas Menu;

public InputField input;

public Text warningSave;

public Text warningDelete;

private bool paused = false;

private bool seedAcceptable = false;

private void Start()

{

if (SceneManager.GetActiveScene().name == "Dungeon")

{

GUI.enabled = true;

Loss.enabled = false;

Paused.enabled = false;

}

}

private void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

if (paused)

{

Resume();

}

else

{

Pause();

}

}

}

public void NewGame()

{

PlayerPrefs.SetInt("seed", 0);

PlayerPrefs.SetInt("savedSeed", 0);

SceneManager.LoadScene("Hub");

}

public void SaveGame()

{

PlayerPrefs.SetInt("savedSeed", grid.GetComponent<Generation>().Seed);

}

public void LoadGame()

{

int seed;

if (PlayerPrefs.GetInt("loadedSeed") > 0)

{

seed = PlayerPrefs.GetInt("loadedSeed");

PlayerPrefs.SetInt("seed", seed);

SceneManager.LoadScene("Dungeon");

}

else

{

if (PlayerPrefs.GetInt("savedSeed") > 0)

{

seed = PlayerPrefs.GetInt("savedSeed");

PlayerPrefs.SetInt("seed", seed);

SceneManager.LoadScene("Dungeon");

}

if (PlayerPrefs.GetInt("savedSeed") == 0)

{

warningDelete.enabled = true;

}

}

}

public void ToMenu()

{

SceneManager.LoadScene("MainMenu");

Time.timeScale = 1f;

}

public void DeleteSave()

{

PlayerPrefs.SetInt("savedSeed", 0);

}

public void LoadBySeed()

{

if (seedAcceptable)

{

PlayerPrefs.SetInt("loadedSeed", Int32.Parse(input.text));

LoadGame();

}

}

public void InputChecker()

{

if (Int32.Parse(input.text) > 999999999 || Int32.Parse(input.text) < 10000)

{

warningSave.enabled = true;

seedAcceptable = false;

}

else

{

warningSave.enabled = false;

seedAcceptable = true;

}

}

public void Pause()

{

paused = true;

Time.timeScale = 0f;

Paused.enabled = true;

GUI.enabled = false;

}

public void Resume()

{

paused = false;

Time.timeScale = 1f;

GUI.enabled = true;

Loss.enabled = false;

Paused.enabled = false;

}

public void Restart()

{

Time.timeScale = 1f;

GUI.enabled = true;

Loss.enabled = false;

Paused.enabled = false;

SceneManager.LoadScene(Application.loadedLevel);

}

public void Exit()

{

Application.Quit();

}

}

Модуль генерации уровней необходим для создания уровней. Его алгоритм был описан ранее в пункте 2.1.2. Он состоит из нескольких методов, которые будут рассмотрены далее, весь же код класса расположен в приложении В.

Создание комнат реализовано в методе GenerateSquare(). Он получает координаты и размер комнаты и создает по ним квадраты. Данный метод представлен в листинге 2.

Листинг 2 – Метод GenerateSquare()

private void GenerateSquare(int x, int y, int radius)

{

lastRoomSize = radius;

Vector3 obstaclePos;

Vector3 enemyPos;

lastX = x;

lastY = y;

for (int tileX = x - radius; tileX <= x + radius; tileX++)

{

for (int tileY = y - radius; tileY <= y + radius; tileY++)

{

Vector3Int tilePos = new Vector3Int(tileX, tileY, 0);

groundMap.SetTile(tilePos, groundTile);

}

}

switch(radius)

{

case 1:

if (Random.Range(0, 100) <= obstacleRate)

{

obstaclePos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

Instantiate(obstacleTiles[Random.Range(0, obstacleTiles.Length)], obstaclePos, Quaternion.identity);

obstacleList.Add(obstaclePos);

}

if (Random.Range(0, 100) <= enemyRate)

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

while (obstacleList.Contains(enemyPos) || enemyList.Contains(enemyPos))

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

}

enemyList.Add(enemyPos);

var Enemy = Instantiate(enemy, enemyPos, Quaternion.identity);

Enemy.GetComponent<Enemy>().target = player.transform;

}

break;

default:

for (int i = 0; i < Random.Range(3, radius); i++)

{

obstaclePos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

while (obstacleList.Contains(obstaclePos) || enemyList.Contains(obstaclePos))

{

obstaclePos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

}

var obstacle = Instantiate(obstacleTiles[Random.Range(0, obstacleTiles.Length)], obstaclePos, Quaternion.identity);

obstacle.GetComponent<ObstacleScript>().grid = grid;

obstacleList.Add(obstaclePos);

}

for (int i = 0; i < Random.Range(1, radius - 1); i++)

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

while (obstacleList.Contains(enemyPos) || enemyList.Contains(enemyPos))

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

}

enemyList.Add(enemyPos);

var Enemy = Instantiate(enemy, enemyPos, Quaternion.identity);

Enemy.GetComponent<Enemy>().target = player.transform;

}

break;

}

}

Метод FillWalls() необходим для расставления стенок вокруг сгенерированной сцены. Он представлен в листинге 3.

Листинг 3 – Метод FillWalls()

private void FillWalls()

{

BoundsInt bounds = groundMap.cellBounds;

for (int xMap = bounds.xMin - 11; xMap <= bounds.xMax + 10; xMap++)

{

for (int yMap = bounds.yMin - 11; yMap <= bounds.yMax + 10; yMap++)

{

Vector3Int pos = new Vector3Int(xMap, yMap, 0);

if (startPos == Vector3.zero)

{

startPos = pos;

}

Vector3Int posAbove = new Vector3Int(xMap, yMap + 1, 0);

Vector3Int posBelow = new Vector3Int(xMap, yMap - 1, 0);

Vector3Int posBefore = new Vector3Int(xMap - 1, yMap, 0);

Vector3Int posAfter = new Vector3Int(xMap + 1, yMap, 0);

Vector3Int dotTL = new Vector3Int(xMap + 1, yMap - 1, 0);

Vector3Int dotTR = new Vector3Int(xMap - 1, yMap - 1, 0);

Vector3Int dotBL = new Vector3Int(xMap + 1, yMap + 1, 0);

Vector3Int dotBR = new Vector3Int(xMap - 1, yMap + 1, 0);

TileBase tile = groundMap.GetTile(pos);

TileBase tileBelow = groundMap.GetTile(posBelow);

TileBase tileAbove = groundMap.GetTile(posAbove);

TileBase tileBefore = groundMap.GetTile(posBefore);

TileBase tileAfter = groundMap.GetTile(posAfter);

TileBase tileTL = groundMap.GetTile(dotTL);

TileBase tileTR = groundMap.GetTile(dotTR);

TileBase tileBR = groundMap.GetTile(dotBR);

TileBase tileBL = groundMap.GetTile(dotBL);

if (tile == null)

{

pitMap.SetTile(pos, pitTile);

propPositionsList.Add(pos);

if (tileTL != null)

{

wallMap.SetTile(pos, dotTLTile);

}

if (tileTR != null)

{

wallMap.SetTile(pos, dotTRTile);

}

if (tileBR != null)

{

wallMap.SetTile(pos, dotBRTile);

}

if (tileBL != null)

{

wallMap.SetTile(pos, dotBLTile);

}

if (tileBefore != null)

{

wallMap.SetTile(pos, leftWallTile);

}

if (tileAfter != null)

{

wallMap.SetTile(pos, rightWallTile);

}

if (tileAbove != null)

{

wallMap.SetTile(pos, bottomWallTile);

}

if (tileAbove != null && tileBefore != null)

{

wallMap.SetTile(pos, rightBtTile);

}

if (tileAbove != null && tileAfter != null)

{

wallMap.SetTile(pos, leftBtTile);

}

if (tileBelow != null && tileBefore != null)

{

wallMap.SetTile(pos, leftTpTile);

}

if (tileBelow != null && tileAfter != null)

{

wallMap.SetTile(pos, rightTpTile);

}

if ((tileBefore != null && tileAfter != null) || (tileAbove != null && tileBelow != null))

{

wallMap.SetTile(pos, WallTile);

}

if (tileBelow != null)

{

wallMap.SetTile(pos, topWallTile[Random.Range(0, topWallTile.Length)]);

}

}

}

}

}

Метод CreateSeed() необходим для создания seed’а генерируемого уровня (Листинг 4).

Листинг 4 – Метод CreateSeed()

private void CreateSeed()

{

string datetime = System.DateTime.Now.ToString("MM/dd") + System.DateTime.Now.ToString("hh:mm:ss");

string resultString = "";

for (int i = 0; i < datetime.Length; i++)

{

if (datetime[i] >= '0' && datetime[i] <= '9')

resultString += datetime[i];

}

Random.InitState(int.Parse(resultString));

seed = Random.Range(10000, 999999999);

}

Метод NewRoute() создает маршруты, по которым создаются сами комнаты. Он представлен в листинге 5.

Листинг 5 – Метод NewRoute()

private void NewRoute(int x, int y, int routeLength, Vector2Int previousPos)

{

if (routeCount < maxRoutes)

{

routeCount++;

while (++routeLength < maxRouteLength)

{

//Initialize

bool routeUsed = false;

int xOffset = x - previousPos.x;

int yOffset = y - previousPos.y;

int roomSize = 1;

if (Random.Range(1, 100) <= roomRate)

roomSize = Random.Range(3, 6);

previousPos = new Vector2Int(x, y);

//Go Straight

if (Random.Range(1, 100) <= deviationRate)

{

if (routeUsed)

{

GenerateSquare(previousPos.x + xOffset, previousPos.y + yOffset, roomSize);

NewRoute(previousPos.x + xOffset, previousPos.y + yOffset, Random.Range(routeLength, maxRouteLength), previousPos);

}

else

{

x = previousPos.x + xOffset;

y = previousPos.y + yOffset;

GenerateSquare(x, y, roomSize);

routeUsed = true;

}

}

//Go left

if (Random.Range(1, 100) <= deviationRate)

{

if (routeUsed)

{

GenerateSquare(previousPos.x - yOffset, previousPos.y + xOffset, roomSize);

NewRoute(previousPos.x - yOffset, previousPos.y + xOffset, Random.Range(routeLength, maxRouteLength), previousPos);

}

else

{

y = previousPos.y + xOffset;

x = previousPos.x - yOffset;

GenerateSquare(x, y, roomSize);

routeUsed = true;

}

}

//Go right

if (Random.Range(1, 100) <= deviationRate)

{

if (routeUsed)

{

GenerateSquare(previousPos.x + yOffset, previousPos.y - xOffset, roomSize);

NewRoute(previousPos.x + yOffset, previousPos.y - xOffset, Random.Range(routeLength, maxRouteLength), previousPos);

}

else

{

y = previousPos.y - xOffset;

x = previousPos.x + yOffset;

GenerateSquare(x, y, roomSize);

routeUsed = true;

}

}

if (!routeUsed)

{

x = previousPos.x + xOffset;

y = previousPos.y + yOffset;

GenerateSquare(x, y, roomSize);

}

}

}

}

Помимо главных модулей был реализован скрипт, который определяет положение выхода на локации и рассчитывает поворот стрелки компаса к нему. Данный скрипт находится в листинге 6.

Листинг 6 – EndPointer

public class EndPointer : MonoBehaviour

{

public Transform target;

public Transform player;

Rigidbody2D rb;

private void Start()

{

rb = GetComponent<Rigidbody2D>();

}

public void Update()

{

Vector3 lookDir = target.position - player.position;

float angle = Mathf.Atan2(lookDir.y, lookDir.x) \* Mathf.Rad2Deg - 180;

rb.rotation = angle;

}

}

## Тестирование программного продукта

Для того чтобы протестировать правильное функционирование модулей программы, были разработаны тест-кейсы. Каждый из них был протестирован, и весь функционал работает (Таблицы 1-6).

Таблица 1 – Тест логики генерации локации

|  |  |  |  |
| --- | --- | --- | --- |
| Название: | Тест логики генерации локации | | |
| Функция: | Случайная генерация мира | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден  провален  заблокирован |
| Предусловие: | |  | |
| Начать игру | | Загрузилась сцена | Пройден |
| Перейти на верхнюю плитку | | Сцена изменилась на новую | Пройден |
| Шаги теста: | |  | |
| Исследовать локацию | | Предметы не должны преграждать проход к серой плите с зеленой окантовкой перехода | Пройден |
| Найти серую плиту с зеленой окантовкой на локации и зайти на нее | | Появилась новая локация, отличающаяся от предыдущей | Пройден |
| Повторить шаги 1-2 несколько раз | | Генерирующаяся сцена не повторяется при загрузке | Пройден |

Таблица 2 − Тест логики противника

|  |  |  |  |
| --- | --- | --- | --- |
| Название: | Тест логики противника | | |
| Функция: | Логика противников | | |
| Действие | | **Ожидаемый результат** | **Результат теста:**  пройден  провален  заблокирован |
| Предусловие: | |  | |
| Начать игру | | Загрузилась сцена | Пройден |
| Перейти на верхнюю плитку | | Сцена изменилась на новую | Пройден |
|  | |  |  |
| Шаги теста: | |  | |
| Исследовать локацию | | На локации есть противники | Пройден |
| Двигаться к противнику | | Противник замечает игрока и двигается в его сторону | Пройден |

Таблица 3 − Тест спавна противников и препятствий

|  |  |  |  |
| --- | --- | --- | --- |
| Название: | Тест подбора предметов | | |
| Функция: | Инвентарь | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден  провален  заблокирован |
| Предусловие: | |  | |
| Начать игру | | Загрузилась сцена | Пройден |
| Перейти на верхнюю плитку | | Сцена изменилась на новую | Пройден |
| Шаги теста: | |  | |
| Исследовать локацию | | На локации есть противники и препятствия. Они не налегают друг на друга | Пройден |
| Найти серую плиту с зеленой окантовкой на локации и зайти на нее | | Появилась новая локация, отличающаяся от предыдущей | Пройден |
| Повторить шаги 1-2 несколько раз | | Сгенерированные объекты не налегают друг на друга | Пройден |

Таблица 4 − Тест логики противника

|  |  |  |  |
| --- | --- | --- | --- |
| Название: | Тест работы оружия | | |
| Функция: | Работа оружия | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден  провален  заблокирован |
| Предусловие: | |  | |
| Начать игру | | Загрузилась сцена | Пройден |
| Перейти на верхнюю плитку | | Сцена изменилась на новую | Пройден |
| Шаги теста: | |  | |
| Нажать на ЛКМ | | Из пистолета вылетела пуля по направлению прицела | Пройден |
| Перевести курсор мыши на другое место | | Прицел изменил позицию | Пройден |
| Нажать на ЛКМ | | Из пистолета вылетела пуля по направлению прицела | Пройден |
| Повторить шаги 2 и 3 несколько раз | | Из пистолета вылетела пуля по направлению прицела | Пройден |

Таблица 5 − Тест сохранения игры

|  |  |  |  |
| --- | --- | --- | --- |
| Название: | Тест сохранения игры | | |
| Функция: | Сохранение | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден  провален  заблокирован |
| Предусловие: | |  | |
| Начать игру | | Загрузилась сцена | Пройден |
| Перейти на верхнюю плитку | | Сцена изменилась на новую | Пройден |
| Шаги теста: | |  | |
| Нажать клавишу паузы (по умолчанию «ESC») | | Появился черный экран с надписью «Пауза» | Пройден |
| Нажать на кнопку «Сохранить» | | Появилось окно подтверждения сохранения | Пройден |
| Нажать кнопку «Сохранить» | | Игра вернулась к окну паузы | Пройден |
| Нажать кнопку «Выйти в главное меню» | | Экран сменился на новый с подписью NoNameProject | Пройден |
| Нажать кнопку «Загрузить» | | Экран сменился на новый с подписью Загрузить | Пройден |
| Нажать кнопку «Продолжить игру» | | Экран сменился на новый, появилась сцена, которая была при сохранении | Пройден |

Таблица 6 − Тест загрузки по номеру уровня

|  |  |  |  |
| --- | --- | --- | --- |
| Название: | Тест подбора предметов | | |
| Функция: | Инвентарь | | |
| Действие | | Ожидаемый результат | Результат теста:  пройден  провален  заблокирован |
| Предусловие: | |  | |
| Начать игру | | Загрузилась сцена | Пройден |
| Перейти на верхнюю плитку | | Сцена изменилась на новую | Пройден |
| Шаги теста: | |  | |
| Нажать кнопку «Загрузить» | | Экран сменился на новый с подписью Загрузить | Пройден |
| Ввести в поле ”Seed” число от 10000 до 999999999 | | В поле исчезла надпись “Seed” и отображается число | Пройден |
| Нажать кнопку «Загрузить» | | Экран сменился на новый, появилась игровая сцена | Пройден |
| Найти надпись “Seed: “ в левом нижнем углу экрана | | Надпись есть на экране | Пройден |
| Сравнить число после двоеточия с введенным числом | | Числа совпадают | Пройден |

# Экономическая часть

## Цели, задачи и алгоритм выполнения экономического части дипломного проекта

Основными целями экономической части дипломного проекта являются:

* Формирование и закрепление профессиональных и общих компетенций, необходимых выпускнику для реализации видов профессиональной деятельности по специальности 09.02.03 «Программирование в компьютерных системах».
* Закрепление и развитие экономических знаний и умений, приобретённых студентами специальности при освоении профессиональной образовательной программы для специальности.
* Разработка технически и экономически эффективного программного или иного информационного продукта в рамках выполнения задания по дипломному проектированию.

Задачи экономической части дипломного проекта включали в себя:

* Расчёт трудоёмкости разработки программного или иного информационного продукта.
* Формирование сметы затрат на разработку программного или иного информационного продукта и расчёт необходимых показателей.
* Расчёт рентабельности от реализации разработанного программного или иного информационного продукта.

Алгоритм выполнения заданий по экономическому разделу дипломного проекта:

* Расчёт и выбор показателей трудоёмкости разработки программного или иного информационного продукта
* Расчёт показателей, необходимых для формирования сметы затрат на разработку и реализацию программного или иного информационного продукта по статьям:
  1. Затраты на материалы и комплектующие
  2. Заработная плата (основная и дополнительная)
  3. Отчисления на социальные нужды
  4. Эксплуатационные затраты
  5. Накладные расходы
  6. Себестоимость разработки продукта
  7. Цена реализации продукта
* Расчёт рентабельности от реализации программного или иного информационного продукта.

## Расчёт показателей себестоимости разработки программного или иного информационного продукта и формирование сметы затрат

### Расчёт трудоёмкости разработки мобильного игрового приложения

Она складывается из шести составляющих, связанных с соответствующими операциями разработки программного обеспечения (ПО) или иного информационного продукта

t∑ = toп + tпс + tап + tпр + tотл. + tд (1)

где

t∑ − Общая трудоёмкость (чел/час)

toп − Затраты труда на подготовку и описание задачи (чел/час)

tпс − Затраты труда на исследование алгоритма решения задачи (чел/час)

tап − Затраты труда на разработку Блок−схемы алгоритма (чел/час)

tпр− Затраты труда на программирование по блок−схеме (чел/час)

tотл. − Затраты труда на отладку и тестирование (чел/час)

tд − Затраты труда на оформление документации (чел/час)

### Базовый показатель [Q] для определения составляющих затрат труда.

Объём исходных текстов программы, прежде всего отражает трудоёмкость и длительность их разработки и позволяет оценивать относительные характеристики производительности труда специалистов – разработчиков ПО. Основной труд специалистов вкладывается в разработку текста программы, поэтому важно, чтобы выбранная единица измерения трудоёмкости была наглядной и просто измеряемой. Поэтому за базовый показатель для определения составляющих затрат труда можно принять условное число операторов (команд) в разрабатываемой программе. Разные литературные источники рекомендуют считать за число операторов в программе разные величины:

1. Число команд на языке программирования.
2. Число логических операторов (математических, перехода, других) в исходном тексте программы.
3. число строк в программе (для языков высокого уровня)

Базовый показатель Q для вычисления затрат труда при разработке ПО можно определить по формуле (2):

Q = q \* c \* (1+p) (2)

где

Q − условное число команд.

q – коэффициент, учитывающий условное число команд в зависимости от типа задачи.

Значение коэффициента q = 400, так как разработку игрового приложения можно отнести к задачам оперативного управления.

С – коэффициент, учитывающий новизну и сложность разрабатываемой программы.

Программные продукты по степени новизны могут быть отнесены к одной из 4 групп:

1. Разработка принципиально новых программ.
2. Разработка оригинальных программ.
3. Разработка программ с использованием типовых решений.
4. Разовая типовая задача.

Программные продукты по степени сложности могут быть отнесены к 1 из 3 групп:

1. Алгоритмы оптимизации и моделирования систем.
2. Задачи учёта и статистики.
3. Стандартные алгоритмы.

Язык программирования высокого уровня, степень новизны - относится к 3 группе. В итоге получаем С = 0.60

p – Коэффициент коррекции программы в ходе её разработки. Рекомендуется: [p = 0,05…0,10]. Выберем p = 0,05.

Выбранные коэффициенты подставляются в исходную формулу (2) и определяется значение Q.

Q = 400 \* 0,60 \* (1 + 0,05) = 252

### Затраты труда на подготовку и описание задачи [tоп].

Этот показатель может быть рассчитан по формуле (3):

tоп= (чел/час) (3)

где

Q – Условное число команд. Q = 252

В – коэффициент учёта изменений задачи в зависимости от сложности задачи и числа изменений. Коэффициент В может быть: В=1,2…1,5.

К – коэффициент, учитывающий квалификацию программиста.

Выберем В = 1,2.

Так как, программист является ещё студентом и стаж работы не имеет, выберем самый наименьший коэффициент K = 0,8.

Выбранные коэффициенты подставляются в исходную формулу (3) и определяется значение tоп.

tоп = (252\*1,2) / (50\*0,8) = 7,56 (чел/час.)

### Затраты труда на исследование алгоритма решения задачи.

Этот показатель может быть рассчитан по формуле (4):

tпс = (чел/час) (4)

где

Q – базовый показатель составляющих затрат.

В – коэффициент недостаточности описания задачи. Рекомендуется [В=1,2...1,5].

K – коэффициент квалификации программиста.

Выберем B = 1,2, Q = 252, K = 0,8.

Выбранные коэффициенты подставляются в исходную формулу (4) и определяется значение tпс.

tпс = (252\*1,2) / (75\*0,8) = 5,04 (чел/час.)

### Затраты труда на разработку блок-схемы алгоритма.

Этот показатель может быть рассчитан по формуле (5):

tал = Q / (20…25) \* K (чел/час) (5)

где

Q – базовый показатель составляющих затрат. Q = 252.

K – коэффициент квалификации программиста. K = 0,8.

Выбранные коэффициенты подставляются в исходную формулу (5) и определяется значение:

tал = 252 / 25 \* 0,8 = 12,6 (чел/час) .

### Затраты труда на программирование по блок-схеме алгоритма.

Этот показатель может быть рассчитан по формуле (6):

tпр = Q / (20…25) \* K (чел/час) (6)

Q = 252

K = 0,8

Выбранные коэффициенты подставляются в исходную формулу (6) и определяется значение tпр:

tпр = 252 / 25 \* 0,8 = 12,6 (чел/час).

### Затраты труда на отладку программы.

Этот показатель может быть рассчитан по формуле (7):

tотл = Q / (4…5) \* K (чел/час) (7)

Q = 252

K = 0,8

Выбранные коэффициенты подставляются в исходную формулу (7) и определяется значение tотл.:

tотл = 252 / 4 \* 0,8 = 78,75 (чел/час).

### Затраты труда на подготовку документации по задаче.

Этот показатель может быть рассчитан по формуле (8):

tд = tпр.д + tоф. (чел/час) (8)

где

tпр.д − затраты труда на подготовку(программирование) материала по задаче. Этот показатель может быть рассчитан по формуле (9).

tоф − Затраты труда на оформление документации. Этот показатель может быть рассчитан по формуле (10).

tпр.д = Q / (15…20) \* K (9)

Q = 252

K = 0,8

Выбранные коэффициенты подставляются в исходную формулу (9) и определяется значение tпр.д:

tпр.д = 252 / 20 ∗ 0,8 = 15,75 (чел/час).

tоф = 0,75 \* tпр (чел/час) (10)

tпр = 12,6 (чел/час)

Выбранный коэффициент подставляют в исходную формулу (10) и определяется значение tоф:

tоф = 0,75 \* 12,6 = 9,45 (чел/час)

Полученные коэффициенты подставляются в исходную формулу (8) и определяется значение tд:

tд = 15,75 + 9,45 = 25,2 (чел/час).

### Суммарные затраты труда на разработку мобильного игрового приложения.

Подставляем все полученные значения в формулу (1) и получаем искомый результат трудоёмкости.

tоп = 7,56 (чел/час.)

tпс = 5,04 (чел/час.)

tал = 12,6 (чел/час.)

tпр = 12,6 (чел/час.)

tотл = 78,75 (чел/час.)

tд = 25,2 (чел/час.)

t∑ = 7,56 + 5,04 + 12,6 + 12,6 + 78,75 + 25,2 = 141,75 (чел/час)

## Расчёт сметы затрат на разработку мобильного игрового приложения.

Себестоимость продукции (работ, услуг) отражает либо текущие затраты на весь объём продукции (работ, услуг), либо текущие затраты на единицу продукции. В первом случае составляется смета затрат на производство объема продукции (работ, услуг), а во втором случае формируется плановая калькуляция.

В этой работе целесообразно составить смету затрат на весь объём работ по разработке программного обеспечения автоматизированной системы. Выбираем типовую структуру сметы затрат, которая представлена в таблице 7.

Таблица 7 – Смета затрат на разработку мобильного игрового приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование статей расходов  (затрат) | Обозначения | Сумма, % |
| 1 | Основные, вспомогательные материалы и комплектующие | Зм | Согласно расчёта |
| 2 | Основная заработная плата | 3По | “ - ” - ” |
| 3 | Дополнительная заработная плата | 3Пд | 20…30% 3По |
| 4 | Отчисления стразовых взносов | ОСВ | 30%(3По+3Пд) |
| 5 | Эксплуатационные затраты, ВСЕГО: | Зэк | Согласно расчёта |
| 5.1. | Затраты на электроэнергию | Зэ.э. | “ - ” - ” |
| 5.2. | Затраты на амортизацию основных фондов | За | “ - ” - ” |
| 5.3. | Затраты на техническое обслуживание и текущий ремонт оборудования и оснастки | Зт.о.р. | “ - ” - ” |
| 6 | Накладные расходы (затраты) | Зн | 40…50% 3По |
| 7 | Себестоимость объёма продукции  (работ) | С | Сумма расходов по статьям с 1 по 6 |
| 8 | Плановая прибыль | П | 15…20% С |
| 9 | Оптовая цена объема продукции  (работ) | Цопт | Сумма расходов по статьям 7 и 8 |
| 10 | Налог на добавленную стоимость | НДС | 18% Цоит |
| 11 | Отпускная цена объёма продукции  (работ) | Ц | Сумма расходов по статьям 9 и 10 |

Все расчёты проводятся далее постатейно.

### Расчёт затрат на материалы и комплектующие.

В этой статье рассматриваются все виды материальных затрат, которые необходимы для выполнения задач по разработке программного или иного информационного продукта. Этот показатель может быть рассчитан по формуле (11):

Зм = Зм1 + Зм2 + .... + Змi (11)

где

Зм – общие затраты на материалы.

Зм1,Зм2 и т.д. – затраты на конкретные материалы.

Так как не было никаких затрат на материалы, то показатель Зм = 0 (руб.).

### Расчёт основной заработной платы [3По]

Специалист работает на должностном окладе.

Месячный оклад специалиста (Ом) составляет 25000 рублей в месяц, значит Ом = 25000 (руб.)

Условный оклад специалиста за день работы ОД рассчитывается по формуле (12):

ОД = Ом / Тм (12)

где

ОД – оклад за день работы (руб.)

Ом – оклад в месяц (руб.)

Тм – число рабочих дней в месяц (среднее за год)

ОД = 25000/22 = 1136,36 (руб.)

Продолжительность работы специалиста в день:

Тg = 8 часов

где Тg – продолжительность рабочего дня.

Стоимость 1 нормо/часа работы специалиста и определяется через условную часовую тарифную ставку (ЧТСусл.) по формуле (13) :

ЧТСусл = Од / Tg (13)

ЧТСусл = 1136,36 / 8 = 142,045 (руб.)

Общие трудозатраты (трудоёмкость) ti на выполнение всего объёма работ, выполненных этим специалистом составили 141,75 часов.

ti = ∑tн оп = 160 часов

Основная заработная плата конкретного специалиста за выполненный объём работы (ЗПор. i) будет равна условной сдельной расценке, то есть произведению условной часовой тарифной ставки (ЧТСусл) на трудоёмкость всего объёма работ, выполненных этим работником (ti)

Исходя из условия СРi усл= ЧТСусл \* tн оп и принимая ti = ∑tн оп получаем формулу (14):

ЗПор.i = СРi усл = ЧТСусл\*ti (14)

ЗПор.i = 142,045 \*190 = 22727,27 (руб.)

Весь объём работы выполнял один специалист, то есть в нашем случае: t∑ = ti

тогда получаем формулу (15):

ЗПо = ЗПор.i = ЧТСусл \* t∑ (15)

ЗПо = 22727,27 (руб.)

### Расчёт дополнительной заработной платы [ЗПд].

Дополнительная заработная плата представляет собой вознаграждение работника за сложность и напряженность, проявленную в своей работе, а также за эффективность трудовой деятельности. По статическим данным в сметах затрат на разработку программного или иного информационного продукта ряда организаций Санкт-Петербурга дополнительная заработная плата специалистов (ЗПд) составляет 20 - 30% от основной заработной платы(3По)

Принимаем % и подставляем в формулу (16):

3Пд = 20% 3По = (20/100) 3По(16)

ЗПо = 22727,27 (руб.)

Выбранные коэффициенты подставляются в исходную формулу (16) и определяется значение 3Пд:

3Пд = 20% \* 22727,27 = 4545,45 (руб.).

### Расчёты отчислений страховых взносов на социальные нужды

В 2020 году суммарная ставка страховых взносов, которые оплачивает работодатель, составит от 30% от суммы основной и дополнительной заработной платы работника.

Этот показатель может быть рассчитан по формуле (17):

ОСВ = 30% (3По + 3Пд) = 0.30 (3По + 3Пд) (17)

ЗПо = 22727,27 (руб.)

3Пд= 4545,45 (руб.)

Выбранные коэффициенты подставляются в исходную формулу (17) и определяется значение ОСВ:

ОСВ =30%(22727,27 +4545,45) = 8181,82 (руб.).

Причём страховые взносы распределяются следующим образом:

Страховые взносы в пенсионный фонд (СВПФ) составляет 20% от суммы основной и дополнительной заработной платы работника.

Этот показатель может быть рассчитан по формуле (18):

СВПФ = 22 % (3По + 3Пд) (18)

ЗПо = 22727,27 (руб.)

3Пд= 4545,45 (руб.)

Выбранные коэффициенты подставляются в исходную формулу (18) и определяется значение СВПФ:

СВПФ = 22% \* (22727,27 + 4545,45) = 5999,98 (руб.).

Страховые взносы в фонд социального страхования (СВФСС) составляет 2,9% от суммы основной и дополнительной заработной платы работника.

Этот показатель может быть рассчитан по формуле (19):

СВФСС = 2.9% (3По + 3Пд) (19)

ЗПо = 22727,27 (руб.)

3Пд= 4545,45 (руб.)

Выбранные коэффициенты подставляются в исходную формулу (19) и определяется значение СВФСС:

СВФСС = 2.9% \* (22727,27 + 4545,45) = 790,908 (руб.).

Страховые взносы в фонд обязательного медицинского страхования (СВФОМС) составляет 5.1% от суммы основной и дополнительной заработной платы работника. Этот показатель может быть рассчитан по формуле (20):

СВФОМС = 5.1% (3По + 3Пд) (20)

ЗПо = 22727,27 (руб.)

3Пд= 4545,45 (руб.)

Выбранные коэффициенты подставляются в исходную формулу (20) и определяется значение СВФОМС:

СВФОМС = 5.1% \* (22727,27 + 4545,45) = 1390,908 (руб.).

В смету затрат включаются суммарные отчисления страховых взносов (ОСВ).

### Расчёт эксплуатационных затрат

Статья сметы «эксплуатационные затраты» включает в себя:

1. затраты на электроэнергию [3ээ],
2. затраты на амортизацию основных фондов [3А],
3. затраты на техническое обслуживание и текущий ремонт оборудования [3тор].

Произведём расчёт этих затрат.

#### Расчёт затрат на электроэнергию [3ээ].

Затраты на электроэнергию [Зээ] складываются из затрат на силовую энергию (Зсээ) и затрат на освещение (Зос) по формуле (21):

Зээ = Зсээ + Зос (21)

Стоимость силовой электроэнергии (Зс.э.э.) необходимой для функционирования оборудования (ПЭВМ и переферийных устройств) при выполнении задачи, можно определить по формуле (22):

3сээ = Сквт.ч \* М \* К3 \* Fэф (22)

где

Сквт.ч – стоимость 1 киловатт-часа электроэнергии, руб. На 01.01.2020 года Сквт.ч. = 4,65 руб. − в квартире.

М – потребляемая мощность оборудования (ПЭВМ современного уровня и т.п.), квт. Рекомендующиеся: М=0.3…0.5 (квт.) Выберем М = 0.3.

К3 – коэффициент загрузки оборудования. Рекомендующиеся: К3=0.8…0.9

Fэф – эффективный фонт времени работы оборудования, необходимый для выполнения задачи по формуле (23), (час)

Fэф = t \*(1-f/100) (23)

где

t − количество времени, затраченное на выполнение проектного задания (создание программного или иного информационного продукта), час В нашем случае допустимо приравнять t= t∑ *f -* % времени, затраченного на текущий ремонт оборудования (ПЭВМ и т.п.) рекомендуется: f=2…3%. Выбираем: f =2%

Выбранные коэффициенты подставляются в формулу (23), и определяется значение Fэф.

Fэф = 160 \* (1-2 / 100) = 156,8 (час).

Выбранные коэффициенты подставляются в формулу (22), и определяется значение 3сээ:

3сээ = 4,65 \* 0,3 \* 0,8 \* 156,8= 174,988 (руб.)

Затраты на освещение можно определить по формуле (24):

Зос= Сквт.ч \* Fэф \* Мос (24)

Где

Fэф − эффективный фонт времени работы оборудования, необходимый для выполнения задачи, (час) Fэф = 156,8 (час).

Мос – Суммарная мощность световых приборов для освещения рабочего места.

Мос = 50 ватт = 0.05 квт

Далее определяем суммарные затраты на освещение по формуле (24)

Зос = 4,65 \* 156,8 \* 0,05 = 36,46 (руб.)

Полученные результаты подставляем в основную формулу и определяем Зэ.э. по формуле (21):

Зээ = 36,46 + 174,99 = 393,73 (руб.)

#### Расчёт затрат на амортизацию основных фондов.

Полные амортизированные отчисления на ПЭВМ и периферийные устройства рассчитываются по формуле (25):

3А= 3А1 + 3А2 + ЗА3 + ... + ЗАi (25)

Где:

3А − полные (суммарные) затраты на амортизацию всего оборудования

Для конкретной единицы оборудования данной модели расчёты проводятся по формуле (26):

3Аi = Собор.i\*НАi **(**26**)**

где

ЗАi– затраты на амортизацию ПЭВМ и периферийных устройств

Собор.i –стоимость нового оборудования данной модели (ПВЭМ и т.п.), руб.

НАi – норма амортизации находится по формуле (27):

НАi = (Собор.i – Cликв.i) / ( Тнорм.i \* Собор.i) \* 100% (27)

где

Собор.i − первоначальная стоимость оборудования I – го вида

Далее рассматривается весь комплекс оборудования и специальных устройств и при этом определяются НА и 3А.

[Собор.1] = 59000 (руб.).

[Cликв.1] − рекомендуемая ликвидационная стоимость данного оборудования I – го вида

[Cликв.1] − рекомендуется от 3 до 5% от первоначальной стоимости [Собор]. Выбираем Сликв = 5% от 59000 (руб) = 2950 (руб).

[Тнорм1] − Рекомендуемый нормативный срок службы данного оборудования [Тнорм1] составляет 3…5 лет из-за быстрого морального износа.

Принимаем Тнорм.1 = 5 лет.

Подставив значения в формулу (26) находим значение НА1 .

НА1 = (59000 – 2950) / (5 \* 59000) \* 100% = 19%

Подставив значения в формулу (27) находим значение ЗА1 .

3А1 =Собор.1\*(НА1 /100%) = 59000 \* (19% / 100%) = 11210 (руб.)

[Собор.2]= 14000 (руб.).

[Cликв.2] = 700 (руб.).

[Тнорм2] = 5 лет.

Подставив значения в формулу (26) находим значение НА2 .

НА2 = (14000 – 700) / (5 \* 14000) \* 100% = 19%

Подставив значения в формулу (27) находим значение ЗА2.

3А2 =Собор.1\*(НА1 /100%) = 14000 \* (19% / 100%) = 2660 (руб.)

После этого определяются общие затраты на амортизацию по формуле (25).

3А = 11210 + 2660 = 13870 (руб.)

#### Расчёт затрат на техническое обслуживание и текущий ремонт всего оборудования [3тор].

Рекомендуемая величина затрат на текущий ремонт [3тор] составляет 3…5% от первоначальной стоимости всего оборудования (ПЭВМ и т.п.) [Собор]

Выберем 3тор = 5% Собор

Исходя из выбранного значения Зтор равняется:

3тор = 5 % \* 59000 = 2950(руб.)

После определения искомых величин в пунктах 2.2.5.1. – 2.2.5.3 все значения подставляем в основную формулу и определяем общую величину эксплуатационных

затрат [3эк]

3эк = 3э.э. + 3А + 3то

Зээ = 393,73 (руб.)

3А = 13870 (руб.)

Зтор = 2950 (руб.)

3эк = 393,73 + 13870 + 2950 = 17218,73 (руб.)

Стоимость 1 часа машинного времени работы ПЭВМ определяется по формуле (28):

Счмв = 3эк / Fэф (руб) (28)

Где:

Счмв – единица стоимости часа машинного времени работы ПЭВМ

3эк – общие эксплуатационные затраты на создание ПО

3эк= 17218,73 (руб.)

Fэф – эффективный фонд рабочего времени Fэф = 156,8 (час).

Данные подставляем в формулу (28):

Счмв= 17218,73 / 156,8 = 109,78 (руб.)

### Расчёт накладных расходов [Зн].

Накладные расходы − это постоянные издержки, которые остаются неизменными и осуществляются вне зависимости от объёма производства. К таким расходам можно отнести величину арендной платы, оплату административно- управленческого персонала, страхование имущества, командировки.

Величина накладных расходов [3н] рекомендуется 40…50% от величины заработной платы [3По]. Принимаем конкретное значение и подставляем в формулу (29):

3н=…% 3По= … / 100 \* 3По=…. (29)

Выберем 3н = 40%

ЗПо = 22727,27 (руб.)

Подставим выбраное значение в формулу (29):

3н = 40% \* 22727,27 = 9090,91 (руб.)

### Расчёт себестоимости разработки модулей игрового приложения.

Величина себестоимости [С] разработки программного или иного информационного продукта составит сумму значений статей с 1 по 6 сметы расходов по формуле (30):

С = 3м + 3По + 3Пд + 3с.н. + 3эк + 3н (30)

Подставив значения, получим

С = 22727,27 + 4545,45 + 8181,82 + 17218,73 + 9090,91 = 61764,18 (руб.)

3м = 0 (руб.)

3По = 22727,27 (руб.)

3Пд = 4545,45 (руб.)

ОСВ = 8181,82 (руб.)

3эк = 17218,73 (руб.)

Зн = 9090,91 (руб.)

### Расчёт плановой прибыли от реализации мобильного игрового приложения.

Прибыль (убыток) от реализации продукции определяется как разница между выручкой от реализации продукции без налога на добавленную стоимость и затратами на производство и реализацию продукцию(себестоимостью). Прибыль может быть валовой и чистой.

Валовая прибыль – это разница доходов и расходов до уплаты всех налогов.

Чистая прибыль – это денежная сумма, которая остаётся после уплаты валовой прибыли и всех установленных законом налогов.

Величина плановой прибыли (П) может быть рекомендована в сумме равной 15…20% от величины себестоимости (С) и зависит от качества и конкурентоспособности разработанного продукта.

Принимаем:

П = 15% \* 61764,18 = 9264,63 (руб.)

### Расчёт налога на добавленную стоимость [НДС].

С 01.01.2019 года ставка НДС составляет 20% от оптовой цены продукта [Цопт] и находится по формуле (31):

НДС=20% Ц опт (31)

где

Цопт  находится по формуле (32):

Цопт = С + П (32)

Подставляем значения в формулу (31):

Цопт = С + П = 9264,63 + 61764,18 = 71028,81 (руб.)

Подставляем найденное значение в формулу (32):

НДС = 0.20 \* 71028,81 = 14205,76 (руб.)

### Расчёт отпускной цены мобильного игрового приложения [Цотп].

Определяется как сумма Цопт и НДС по формуле (33):

Цотп = Цопт + НДС (33)

Подстовляем значения в формулу (33):

Цотп = Цопт + НДС = 71028,81 + 14205,76 = 85234,57 (руб.)

### Итоговая смета затрат на разработку мобильного игрового приложения

Полученные результаты во время выполнения всех расчётов подставляем в таблицу 8 смета затрат.

Таблица 8 – Смета затрат на разработку игрового мобильного приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование статей расходов  (затрат) | Обозначения | Сумма, руб. |
| 1 | Основные, вспомогательные материалы и комплектация | Зм | 0,00 |
| 2 | Основная заработная плата | 3По | 22727,27 |
| 3 | Дополнительная заработная плата | 3Пд | 4545,450 |
| 4 | Отчисления страховых взносов | ОСВ | 8181,82 |
| 5 | Эксплуатационные затраты. ВСЕГО: | Зэк | 17218,73 |
| 5.1. | Затраты на электроэнергию | Зэ.э. | 393,73 |
| 5.2. | Затраты на амортизацию основных фондов | За | 13870 |
| 5.3. | Затраты на техническое обслуживание и текущий ремонт оборудования и оснастки | Зтор. | 2950 |
| 6 | Накладные расходы (затраты) | Зн | 9090,91 |
| 7 | Себестоимость объёма продукции  (работ) | С | 61764,18 |
| 8 | Плановая прибыль | П | 9264,63 |
| 9 | Оптовая цена объема продукции  (работ) | Цопт | 71028,81 |
| 10 | Налог на добавленную стоимость | НДС | 14205,76 |
| 11 | Отпускная цена объёма продукции  (работ) | Ц | 85234,57 |

Расчёт рентабельности от реализации разработанного продукта.

Она определяется, как отношение плановой прибыли к величине себестоимости по формуле (34):

Р = П / С (34)

где

П – Плановая прибыль

С – себестоимость изготовления продукта

Подставляем значения в формулу (34):

Р = 9264,63 / 61764,18= 0,15 %

ЗАКЛЮЧЕНИЕ

В ходе создания проекта были разработаны модули для игрового приложения. Была достигнута цель проекта и выполнены поставленные задачи.

При разработке игрового приложения были разработаны дизайн и концепт документы, отвечающие на все пункты для создания приложений, спроектированы и реализованы интерфейсы главного меню и внутреигрового цикла.

Был разработан алгоритм для модуля генерации случайных уровней. Сам модуль так же был реализован. Каждый уровень генерируется по своему сииду и уникален сам по себе. На каждом уровне присутствуют препятствия и противники, переход между уровнями происходит после нахождения игроком плиты выхода.

В помощь игроку был реализован скрипт для компаса, который указывает на выход. Так же была разработана миникарта, отображающая небольшую область вокруг персонажа. Для самого же игрока был реализован скрипт передвижения.

Были найдены показатели и коэффициенты для расчета единовременных затрат на проектирование и разработку игрового приложения. Для определения производственной программы в стоимостном выражении произведен расчет себестоимости изделия и фонд заработной платы, который составил 25000 руб. Отпускная цена продукции составила 85234,57 руб.

По результатам расчетов, разработанное приложение является рентабельным и экономически обоснованным, а отпускная цена продукта обеспечит конкурентоспособность продукции на внутреннем рынке товаров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шампандар А.Дж. Искусственный интеллект в компьютерных играх. Как обучить виртуальные персонажи реагировать на внешние воздействия [текст]. – М.: Вильямс, 2017. – 768 с.
2. Грегори Дж. Game Engine Architecture [электронный ресурс] / Дж. Грегори. – Режим доступа: https://geekbrains.ru/posts/gamedev\_books\_part1
3. Яннакакис Г.Н. [Искусственный интеллект и игры](http://gameaibook.org/) [электронный ресурс] / Г.Н. Яннакакис, Ю. Тогелиус. – Режим доступа: http://gameaibook.org/book.pdf
4. Куик Дж. Learn to implement games with code [электронный ресурс]. – Дж.Куик Дж. – Режим доступа: https://habr.com/ru/company/plarium/blog/384229/
5. Галёнкин С. Маркетинг игр [электронный ресурс] / С. Галёнкин. – Режим доступа: https://galyonkin.com/wp-content/book/Games-Marketing-by-Galyonkin-designed.pdf
6. Рудаков А.В., Федорова Г.Н. Технология разработки программных продуктов: Учебник. СПО. - 8-е изд., стер. - М.: ИЦ "Академия", 2015. - 208 с. - ЭБС АКАДЕМИЯ
7. Гагарина Л.Г., Федоров А.Р., Федоров П.А. Введение в архитектуру проектирования программного обеспечения: учебное пособие [электронный ресурс] / Л.Г. Гагарина, А.Р. Федоров. – М.: ИД «ФОРУМ» – ИНФРА-М, 2017. – 320 с. – (Высшее образование). – Режим доступа: – <http://znanium.com/bookread2.php?book=615207>
8. Бубнов А.А. Введение в программную инженерию: Учебник [электронный ресурс] / В.А. Антипов, А.А. Бубнов, А.Н. Пылькин, В.К. Столчнев. — М.: КУРС: ИНФРА-М.: 2018. — 336 с. – Режим доступа: – <http://znanium.com/bookread2.php?book=944151>
9. Хокинг Дж. Х70 Unity в действии. Мультиплатформенная разработка на C# / Пер. с англ. И. Рузмайкиной. — СПб.: Питер, 2016. — 336 с.
10. [Разработка игр на Unity](https://www.intuit.ru/studies/courses/3487/729/info) Национальный открытый Университет ИНТУИТ [Электронный ресурс]; URL – <https://www.intuit.ru/studies/courses/3487/729/info>
11. Джозеф Х. Unity в действии. Мультиплатформенная разработка на C#. 2-е издание / Д. Хокинг – М: 2016 г. – 336 стр.
12. Гейг М. Unity 2018 Game Development in 24 Hours - 3rd Edition / М. Гейг – М: 2018. – 574 с.
13. Котляров В.П. Основы тестирования программного обеспечения: курс лекций / Котляров В.П.- М.: БИНОМ. ИНТУИТ, 2012.- 288 с. –ЭБС «IPRbooks
14. Хокинг Д. Unity в действии. Мультиплатформенная разработка на C#. 1-е издание / Д.Хогинг – М: 2017. – 564 с.
15. Unity – Manual: Unity User Manual [Электронный ресурс] / Режим доступа: https://docs.unity3d.com/Manual/index.html .
16. Unity3DSchool [Электронный ресурс] / Режим доступа: <https://unity3dschool.ru>
17. «Руководство Unity» [Электронный ресурс] Режим доступа: <https://docs.unity3d.com/ru/current/Manual/index.html>
18. «Создание аркады на движке Unity» [Электронный ресурс] Режим доступа: <http://vcnmo.ru/upload/activities/kolmogor_2016/357/3578c.pdf>
19. Unity3D для начинающих [Электронный ресурс] / Режим доступа: <https://habr.com/ru/post/112287/>
20. Забытое секретное оружие Unity - UnityEvents [Электронный ресурс] / Режим доступа: <https://habr.com/ru/post/308570/>
21. Процедурная генерация подземелий в roguelike [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/354826/
22. Процедурная генерация уровней [Электронный ресурс] – / Режим доступа: <https://habr.com/ru/post/418685/>
23. Руководство по программированию на C# [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/ruru/dotnet/csharp/programming-guide
24. Работа с Корутинами в Unity [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/216185/
25. Процедурная генерация уровней [Электронный ресурс] / Режим доступа: https://habr.com/ru/post/418685/

ПРИЛОЖЕНИЕ А

Концепт-документ

|  |
| --- |
| центрифуга |
|  |
| **NoName** |
| **Концепт-документ** |

## Введение

В одной из наших параллельных вселенных, на нашей планете Земля проживали такие же маленькие люди, как и мы сами. Но жизнь у них была не так проста, мир разделенный на две большие империи враждует между собой. Технический прогресс не стоял на месте и развивался с космическими скоростями, государствам приходится вкладывать все ресурсы в научную и военную промышленности. Кроме того, в мире было найдено новое полезное ископаемое – рофланиум, которое позволит создавать передовые образцы вооружения. Единственное мест добычи рофланиума – континент, который у нас называется Австралией, %странанейм%, единственное государство, которое не входит ни в одну из империй и находится в нейтралитете между ними.

Началась охота за рофланиумом и две империи столкнулись друг с другом, что и заставило холодную войну перерости в мировую, которой еще не бывало до этого. В одной из этих империй – %империянейм% и проживает наш главный герой – Игнат, обычный работяга на заводе. В его государстве начинают происходить теракты, правительство обвиняет вражеских агентов, желающих пошатнуть и без того слабую экономическую ситуацию в империи. Не смотря на огромное количество патрулей на улицах, теракты продолжались, и в одном из них и погибла семья Игната. Эта трагедия повлияла на его решение вступить в ряды народного ополчения, для защиты своего государства.

## Жанр и аудитория

Игра «NoName» относится к жанрам: Souls-like, Top-Down, Action, Adrenaline-Action, Cyberpunk. Также игра вмещает в себя элементы RPG и Rogue-Like игр.

Игра более ориентирована на взрослую аудиторию, так как в ней содержатся сцены насилия, табакокурения и распития спиртных напитков, но все это можно отключить в настройках. Минимальный рекомендуемый возраст игрока – 16 лет. Дополнительную привлекательность игра имеет для владельцев не самой современной конфигурации PC и людей, ищущих новые, немногочисленные на данный момент, проекты в жанре Cyberpunk.

Игра не использует торговые марки или другу собственность, подлежащую лицензированию.

## Основные особенности игры

Ключевые особенности игры (USP):

* Souls-like составляющая игры не даст игроку пробегать уровни «сломя голову», любая ошибка игрока вынудит проходить уровень сначала, продумывая стратегию для успешного выполнения задания.
* Параллельная вселенная, в которой проходит альтернативное развитие нашего мира
* Cyberpunk стилистика игры, показывающая нам мир, где бедность простых людей сплетается с быстроразвивающимся прогрессом.
* Множество задач и миссий, требующих тактических решений.
* Невысокие требования к технике при отличной яркой графике. Игра пригодна для издания на западном рынке.

Сюжет рассчитан на 5~10 часов прохождения.

## Описание игры

Главный герой проживает в государстве, форма правления в которой – тирания. В государстве сильно развит культ личности, жители буквально молятся на своего правителя. Хоть с виду жизнь людей идет довольно неплохо, большинство людей беднеет, война и технический прогресс отбирает большое количество ресурсов у государства. Правительство заменило людей на алгоритмичных работах роботами. Там, где требовался творческий подход и людской разум, все еще оставались люди. Люди начали терять рабочие места, начала расцветать воровство и появилось несколько крупных преступных группировок.

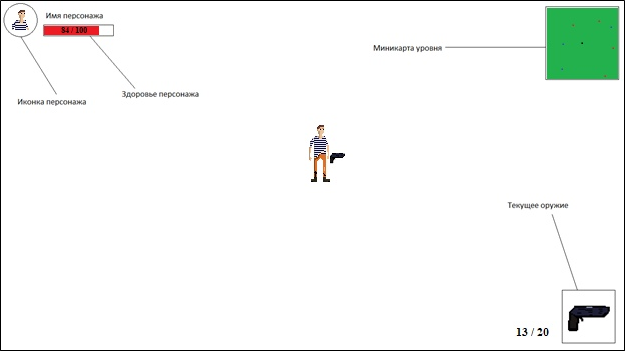
Государство начало предлагать людям роботизированные имплантаты, которые каким-либо образом улучшали заменяемые части тела либо просто заменяли их в случае утраты. На улицах начали появляться киборги – люди с этими модификациями. Эти имплантаты не самое дешевое удовольствие и лишь не многие могут позволить их себе приобрести, поэтому начали появляться черные рынки, где люди могли купить и установить себе подобные модификации по более низкой цене, но они не получали никакой гарантии, что они переживут их установку в подпольных условиях.

Время от времени безработные поднимают бунты, которые успешно подавляются полицией, это сильно предается огласке. Иногда их людно казнили для уменьшения недовольства среди людей, но чаще отправляли на «общественные работы». На самом же деле, провинившихся насильно чипируют и изменяют кибернетически в интересах государства, так измененные становятся лояльными правительству и пропагандируют это в своем окружении. Измененные не сильно отличаются от обычных людей, они ведут себя, как и прежде, но на их мысли могут влиять правительство. Так же есть шанс того, что измененный сойдет с ума и чем дольше он в этом состоянии, тем выше шансы. Первое появление измененных произошло после восстания на заводе «Зеленый квадрат», которое было успешно подавлено, а восставшие рабочие были прилюдно казнены.

Спонтанно начавшиеся теракты начали беспокоить народ. Так как слухи очень быстро расползаются среди народа и довольно поздно доходили до полиции, было решено сформировать народное ополчение для предотвращения терактов. Людей беспокоило то, что иностранные агенты в принципе были способны проникнуть в их государство и устроить подобную подрывную деятельность, так как народ считал свою империю – непреступной крепостью. Многие даже и не догадываются о ведущейся войне, власти умалчивают это и не дают распространению этих новостей среди народа, держа в страхе людей, находящихся в непосредственной близости к области боевых действий. Эти мысли наталкивают некоторых о возможности существования повстанческого движения.

Устав от повседневности своих дней, главный герой не находит ничего лучше, чем присоединится к народному ополчению, с целью защиты от возможных будущих атак вражеских агентов. Продвигаясь по сюжету, выполняя задания своих руководителей, Игнат успешно борется с вражескими террористами. В одной из операций, оставшись последним из своего отряда, Игнат начинает разговор с раненным врагом, от которого и узнает и о войне, и о повстанцах, и о причине их появления. Повстанцы считали, что государство, как и рыба, начинает гнить с головы, и текущее правительство необходимо свергнуть, для успешного процветания государства. Своими терактами они устраняли важных персон в государстве, постепенно достигая своей цели. После долгих раздумий, герой спасает раненного повстанца и перемыкает к повстанцам, восстав против правительства.

## Пример геймплея



Вид игры – Top-Down, плоский интерфейс пользователя, управление героем производится клавиатурой и мышью.

Интерфейс функционально привычен и стандартен для RPG и Action.

Стиль юнитов и построек выполнен в стиле cyberpunk, мир наполнен темными мрачными краками.

Общий стиль графики пиксельный, вписывающийся в сеттинг и подчеркивающий атмосферу игры.

## Сравнение и особенности позиционирования

Игра «Noname» имеет, с одной стороны, некоторые оригинальные решения в жанре, но в то же время концепция игры использует следующие лучшие свойства выбранных образцов:

* «Enter the Gungeon» – исследование карты, локации со случайной генерацией уровней, большое количество оружия разнообразного типа.
* «The Binding of Isaac» - сложность прохождения игры, одна ошибка вынуждает проходить весь уровень заново.
* «Hotline Miami» – сложность прохождение игры, жесткость сюжета, не позволяющая сильно отклоняться от него по мере прохождения игры.

Данный подбор свойств игры ориентировался на такие современные тенденции в Souls-like, как: зависимость от прокачки персонажа по его классу, правильному подбору обмундирования и разработки тактики для прохождения игры.

## Платформа

Разработка игры для иных платформ кроме PC Windows не планируется.

|  |  |  |
| --- | --- | --- |
| Требования | Минимальные | Рекомендуемые |
| Процессор | Celeron 266 MHz | Pentium 3 400MHz |
| ОЗУ | 128MB | 256MB |
| Видео карта | DirectX 5 совместимая, класса ATI RAGE | |

## Контакты

|  |  |
| --- | --- |
| Контактные лица: | Балев Александр  Межецкий Иван |
|  |  |
| Телефоны: | +7 (921) 372 25 89  +7 (800) 555 35 35 |
|  |  |
| E-mail: | [contact@centrifuge.com](mailto:contact@centrifuge.com) |
|  |  |
| Адрес: | Санкт-Петербург |
|  |  |
| Сайт: | <https://centrifuge.games> |

ПРИЛОЖЕНИЕ Б

Дизайн-документ

|  |
| --- |
| центрифуга |
|  |
| **NoName** |
| **Дизайн-документ** |

## Введение

Дальнейшее содержание этого документа организовано по разделам, в которых приводится следующая информация:

* Раздел 2. Концепция. Данный раздел содержит основные сведение об игре – общее описание, жанр, предпосылки создания игры и ее особенности, целевая аудитория и платформа. Раздел предназначен для всех заинтересованных лиц.
* Раздел 3. Функциональная спецификация. Описание игры с точки зрения пользователя и предлагаемых ею возможностей. Содержит данные про сюжет и принципы игры; интерфейс; графику и звуковое наполнение; состав игровых уровней. Этот раздел предназначен для всех заинтересованных лиц.
* Раздел 4. **Ошибка! Источник ссылки не найден.**. Техническое описание реализации функциональной спецификации. Содержит сведения про программную структуру игры; структур хранения данных; организация работы с графическими и звуковыми ресурсами игры. Раздел предназначен для технических специалистов.
* Раздел 5. **Ошибка! Источник ссылки не найден.**. План организации разработки проекта. Содержит информацию о существующих и требуемых ресурсах; финансовые вопросы и календарный план разработки. Раздел предназначен для лиц, принимающих решения.
* Раздел 6. Контакты. Содержит контактную информацию.

# Концепция

## Введение

Город N. 2069 год. В городе царит безработица и люди вынуждены перебиваться случайными заработками. Совершенно обычный гражданин решает, что его семья заслуживает лучшей жизни и решает примкнуть к группировке с не лучшей репутацией.

Гражданин показывает отличную работоспособность в данной группировке, что не спасает его от череды событий, которые повлекли за собой очень необычные события…

## Платформа, жанр и аудитория

Игра «NoName» относится к жанрам: Souls-like, Top-Down, Action, Adrenaline-Action, Cyberpunk. Также игра вмещает в себя элементы RPG и Rogue-Like игр.

Игра более ориентирована на взрослую аудиторию, так как в ней содержатся сцены насилия, употребления табачной и алкогольной продукции, но все это можно отключить в настройках. Минимальный рекомендуемый возраст игрока – 16 лет. Дополнительную привлекательность игра имеет для владельцев не самой современной конфигурации PC и людей, ищущих новые, немногочисленные на данный момент, проекты в сеттинге киберпанк.

Игра не использует торговые марки или другу собственность, подлежащую лицензированию.

## Основные особенности игры

Ключевые особенности игры (USP):

* Souls-like составляющая игры не даст игроку пробегать уровни «сломя голову», любая ошибка игрока вынудит проходить уровень сначала, продумывая стратегию для успешного выполнения задания.
* Параллельная вселенная, в которой проходит альтернативное развитие нашего мира
* Cyberpunk стилистика игры, показывающая нам мир, где бедность простых людей сплетается с быстроразвивающимся прогрессом.
* Множество задач и миссий, требующих тактических решений.
* Невысокие требования к технике при отличной яркой графике. Игра пригодна для издания на западном рынке.

Сюжет рассчитан на 5~10 часов прохождения.

## Описание игры

Главный герой проживает в государстве, форма правления в которой – тирания. В государстве сильно развит культ личности, жители буквально молятся на своего правителя. Хоть с виду жизнь людей идет довольно неплохо, большинство людей беднеет, война и технический прогресс отбирает большое количество ресурсов у государства. Правительство заменило людей на алгоритмичных работах роботами. Там, где требовался творческий подход и людской разум, все еще оставались люди. Люди начали терять рабочие места, начало расцветать воровство и появилось несколько крупных преступных группировок.

Государство начало предлагать людям роботизированные имплантаты, которые каким-либо образом улучшали заменяемые части тела либо просто заменяли их в случае утраты. На улицах начали появляться киборги – люди с этими модификациями. Эти имплантаты не самое дешевое удовольствие и лишь не многие могут позволить их себе приобрести, поэтому начали появляться черные рынки, где люди могли купить и установить себе подобные модификации по более низкой цене, но они не получали никакой гарантии, что они переживут их установку в подпольных условиях.

Время от времени безработные поднимают бунты, которые успешно подавляются полицией, это сильно предается огласке. Иногда их людно казнили для уменьшения недовольства среди людей, но чаще отправляли на «общественные работы». На самом же деле, провинившихся насильно изменяют кибернетически в интересах государства, так измененные становятся лояльными правительству и пропагандируют это в своем окружении. Измененные не сильно отличаются от обычных людей, они ведут себя, как и прежде, но на их мысли могут влиять правительство. Так же есть шанс того, что измененный сойдет с ума и чем дольше он в этом состоянии, тем выше шансы. Первое появление измененных произошло после восстания на заводе «Зеленый квадрат», которое было успешно подавлено, а восставшие рабочие были прилюдно казнены.

Спонтанно начавшиеся теракты начали беспокоить народ. Так как слухи очень быстро расползаются среди народа и довольно поздно доходили до полиции, было решено сформировать народное ополчение для предотвращения терактов. Людей беспокоило то, что иностранные агенты в принципе были способны проникнуть в их государство и устроить подобную подрывную деятельность, так как народ считал свою империю – непреступной крепостью. Многие даже и не догадываются о ведущейся войне, власти умалчивают это и не дают распространению этих новостей среди народа, держа в страхе людей, находящихся в непосредственной близости к области боевых действий. Эти мысли наталкивают некоторых о возможности существования повстанческого движения.

Устав от повседневности своих дней, главный герой не находит ничего лучше, чем присоединится к народному ополчению, с целью защиты от возможных будущих атак вражеских агентов. Продвигаясь по сюжету, выполняя задания своих руководителей, Игнат успешно борется с вражескими террористами. В одной из операций, оставшись последним из своего отряда, Игнат начинает разговор с раненным врагом, от которого и узнает и о войне, и о повстанцах, и о причине их появления. Повстанцы считали, что государство, как и рыба, начинает гнить с головы, и текущее правительство необходимо свергнуть, для успешного процветания государства. Своими терактами они устраняли важных персон в государстве, постепенно достигая своей цели. После долгих раздумий, герой спасает раненного повстанца и перемыкает к повстанцам, восстав против правительства.

## Сравнение и предпосылки создания

Игра «Noname» имеет, с одной стороны, некоторые оригинальные решения в жанре, но в то же время концепция игры использует следующие лучшие свойства выбранных образцов:

* «Enter the Gungeon» – исследование карты, локации со случайной генерацией уровней, большое количество оружия разнообразного типа.
* «The Binding of Isaac» - сложность прохождения игры, одна ошибка вынуждает проходить весь уровень заново.
* «Hotline Miami» – сложность прохождение игры, жесткость сюжета, не позволяющая сильно отклоняться от него по мере прохождения игры.

Данный подбор свойств игры ориентировался на такие современные тенденции в Souls-like, как: зависимость от прокачки персонажа по его классу, правильному подбору обмундирования и разработки тактики для прохождения игры.

Создание данной игры, прежде всего обосновано интересным сеттингом.

## Системные требования

Как уже было сказано, игра принципиально ориентирована на конфигурацию 3х-летней давности:

|  |  |  |
| --- | --- | --- |
| Требования | Минимальные | Рекомендуемые |
| Операционная система | Windows 98/ME (поддерживается 2000/XP) | |
| Процессор | Celeron 266 MHz | Pentium 3 400MHz |
| ОЗУ | 128MB | 256MB |
| CD-ROM привод | 8X cкоростной | 24X cкоростной |
| Свободное место на HDD | 50MB (музыка на CD) | 650 MB (музыка на HDD) |
| Видео карта | DirectX 5 совместимая, класса ATI RAGE | |
| Звуковая карта | Любая DirectX 11 совместимая звуковая карта | |
| Управление | Мышь, клавиатура | |

# Функциональная спецификация

## Ход игры и сюжет

При запуске игры игрока встречает главное меню.

Если пользователь впервые запускает игру и у него не имеется файлов сохранений игрового процесса, то элемент главного меню «Загрузить игру» ему будет не доступен. Игрок сможет лишь войти в настройки и загрузить новую игру.

При загрузке новой игры, пользователю демонстрируются вступительная кат-сцена (все кат-сцены в игре выполнены на игровом движке). Сперва игрока обучают основам игрового процесса, после которого игроку будет доступен главное задание. Выполнив его игроку открывается свободное перемещение по миру, в котором у него есть возможность выполнять побочные задания и разговаривать с персонажами.

По ходу прохождения заданий игрок сможет сражаться с противниками, разговаривать с персонажами и находить предметы которые у него будет возможность применять или продать в магазине.

По ходу продвижения по сюжету игроку будет постепенно открываться доступ к другим локациям.  
Так же во время прохождения заданий игроку может предлагаться сюжетный выбор, который будет влиять на состояние мира вокруг него.

У игрока есть возможность приступить к главному заданию даже через 10-15 часов.  
Перемещение игрока может быть ограничено сюжетно обоснованными игровыми проблемами и состоянием мира на данный момент.

Ход сюжета зависит от выбора игрока в определенных диалогах или от его действий.

## Физическая модель

### Игровой мир и время.

Действия игры проходят в параллельном мире, в нашем недалеком будущем. Главное отличие этого мира в том, что привычных нам государств там нет, как и нас самих. В ней Земля населена совершенно другими людьми и людьми-киборгами, производство которых стало возможным благодаря растущему технологическому процессу. Все государства объединены в 2 большие империи, враждующие между собой.

Время в мире игры не имеет существенного значения, оно имеет декоративный вид для изменения внешнего вида сцен.

### Жизненная и боевая модель

Каждый юнит (и некоторые специальные объекты) обладают определенным уровнем здоровья (HPстат), которое может уменьшаться под действием атаки противника и действий предметов из окружающей среды, а также повышаться за счет условных аптечек. Текущее состояние здоровья (HPтек) отображается в левом верхнем углу.

Каждая сторона владеет умением атаковать с помощью различного оружия. Каждое оружие имеет свой урон (Dmgстат), боезапас (Ammoстат) и дальность поражения (Rangeстат). Каждое использование оружия уменьшает его боезапас (Ammoтек), а на изменение урона (Dmgтек) могут влиять другие предметы героя, так и умения самого героя (Dmgбнс). Так же, урон по юниту варьируется от его сопротивления к урону (Resistстат)

### Ресурсы

Существование игрока в мире игры поддерживается предметами: аптечки лечат героя, предметы дают защиту и эффекты оружию, с помощью оружия игрок защищает себя. И все это можно приобрести за внутреигровой эквивалент денег – монеты.

Получить монеты можно, продавая предметы и выполняя задания.

Каждый предмет стоит по-разному, чем лучше его показатели, тем дороже он и стоит, соответственно.

### Формулы

Принято, что единицы нормированы, т.е. изменяются в пределах от 0 (минимальное значение) до 100 (максимальное значение).

Таким образом, расчет в изменениях уровня HPтек производится по формулам:

* Расчет урона оружия

Dmgтек = Dmgстат \* ( 1 + (Dmgбон(слот предмета 1) + Dmgбон(слот предмета 2) + Dmgбон(слот предмета 3) + Dmgбон(герой)) / 100) \* (1 - Resistстат / 100)

* Повреждения в атаке

HPтек = HPтек – Dmgтек

## Персонажи игрока

При запуске новой игры игроку будет предложено создать своего персонажа из пресетов, а также выбрать класс. Разнообразные предметы одежды и оружия игрок волен изменять в любое время на то, которое ему понравится. То же самое относится и к внешности противника.

### Персонаж 1

Игнат – главный герой. Обычный гражданин своей страны, который решает заработать себе на жизнь не совсем честным трудом и законным трудом.

**Общая характеристика**: Игнат является главным героем, он новичок на его новой работе, но он быстро схватывает. В зависимости от класса, одежды и оружия у Игнат могут повышаться или понижаться те или иные параметры.

## Элементы игры

### Главный враг

Главный враг – антагонист в данном мире. В империи поддерживается культ Главного врага, жители верят в его всемогущество и молятся на него.

Сам Главный враг будет появляться в игре на экранах телевизоров, билбордах и новостных сводках, а с самим игроком повстречается в самом конце игры, где игрок будет решать с кем расправиться главой повстанцев или Главным врагом. В зависимости от выбора главного героя будет зависеть и концовка игры.

**Общая характеристика**: Главный босс, с большим показателем здоровья сильным сопротивлением к урону и сильными, но медленными атаками. Имеет супер-атаку.

Характеристики:

|  |  |
| --- | --- |
| Параметр | Значение |
| Здоровье (HPстат) | Очень высокое |
| Атака физическая (Dmgстат) | Высокая |
| Бонус героя к атаке (Dmgбнс) | Очень высокая |
| Бонус героя к защите (Resistстат) | Полная (специальный объект) |

У главного антагониста будет реализован особенный паттерн поведения.

### Бандит

Бандит – наиболее частый юнит, которого игрок будет встречать в основном на улице или некоторых локациях. Изначально будет враждебен к главному герою, но продвигаясь далее по сюжету, становится ему врагом или союзником. При убийстве данного юнита, есть шанс выпадения огнестрельного оружия, первого для Персонаж 1. Противник для всех, кроме юнитов того же типа.

**Общая характеристика**: Слабый враг, у которого полностью отсутствует бонус к урону.

Характеристики:

|  |  |
| --- | --- |
| Параметр | Значение |
| Здоровье (HPстат) | Низкое |
| Атака физическая (Dmgстат) | Низкая |
| Бонус героя к атаке (Dmgбнс) | - (отсутсвует) |
| Бонус героя к защите (Resistстат) | Низкий |

Патрулирующий может быть в следующих состояниях:

* Стоит
* Преследует
* Атакует
* Умирает

### Патрульный

Патрульный – стандартный юнит главного антагониста так же будут часто встречаться главному герою с самого начала игры. Изначально нейтральны к главному герою, пока тот не нарушит закон или не атакует данного юнита или других юнитов этого типа.

При убийстве данного юнита есть возможность сделать враждебными остальных юнитов этого типа к игроку, а также получить достаточно сильное оружие.

При победе над игроком есть шанс того, что игрок попадет под стражу.

**Общая характеристика**: Юнит средней силы, у которого полностью отсутствует бонус к урону.

Характеристики:

|  |  |
| --- | --- |
| Параметр | Значение |
| Здоровье (HPстат) | Среднее |
| Атака физическая (Dmgстат) | Среднее |
| Бонус героя к атаке (Dmgбнс) | - (отсутсвует) |
| Бонус героя к защите (Resistстат) | Низкий |

Повстанец может быть в следующих состояниях:

* Патрулирует
* Стоит
* Преследует
* Атакует
* Умирает
* Побеждает

### Солдат

Солдат – самый сильный среди обычных врагов главного героя. Будет встречаться главному герою, когда тот будет приближаться к боссу.

**Общая характеристика**: Юнит со средним показателем здоровья, но сильной атакой и сопротивлением

Характеристики:

|  |  |
| --- | --- |
| Параметр | Значение |
| Здоровье (HPстат) | Средний |
| Атака физическая (Dmgстат) | Высокая |
| Бонус героя к атаке (Dmgбнс) | Средний |
| Бонус героя к защите (Resistстат) | Высокий |

Объект может находиться в следующих состояниях:

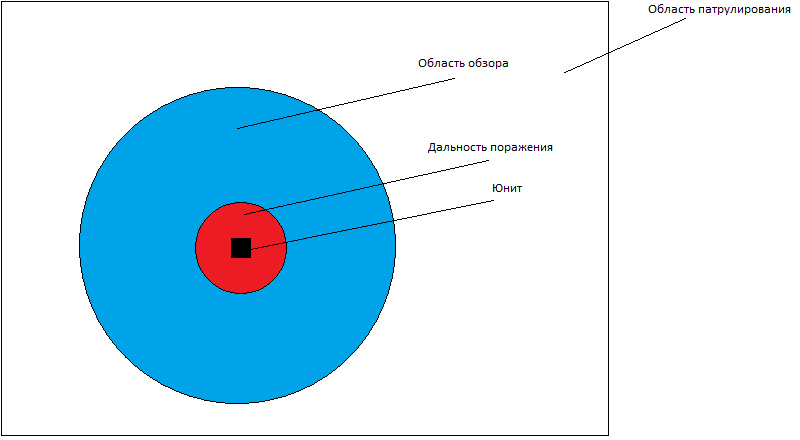
* Патрулирует
* Стоит
* Преследует
* Атакует
* Умирает

## «Искусственный интеллект»

Задачами «искусственного интеллекта» (далее ИИ) является:

* Управление юнитами противника для защиты и отстаивания периметра, указанному юниту.
* Изменять состояния противника в зависимости от условий.
* Поиск оптимального пути до цели и обход препятствий.
* Поиск, обнаружение и преследования главного героя, с целью уничтожить, если тот попадает в его поле и дальность зрения.
* Атака главного героя, если тот попадает в его радиус поражения, либо преследовать, пока тот не пропадет.

При реализации алгоритмов поведения юнитов следует учитывать различные зоны юнитов:



Реализация ИИ для управления юнитами противника должна предусматривать:

* Патрулирование выделенной территории
* Обнаружение главного героя, если тот попадает в поле обзора
* Атаку главного героя, если тот попадает в радиус поражения оружием
* Если главный герой не попадает в радиус поражения, то преследовать его, пока тот не попадет в нее и не будет убит
* Нахождение кратчайшего пути к главному герою
* Если главный герой пропадает из радиуса обзора, то прийти на последнюю точку где он был виден, подождать там 2 секунды и начать поиск.
* Если при поиске противник не был найден, вернуться к патрулированию.

## Многопользовательский режим

Игра построена на прохождение одиночной компании и не предполагает использование многопользовательского режима.

## Интерфейс пользователя

### Блок-схема

Интерфейс пользователя организован по следующим схемам:

**Save / Load**

**New game menu**

Новая игра

Сохранить игру

Загрузить игру

Выход

Сохранить / Загрузить

Отмена

**Начало игры**

**Выход**

Игровое меню в режиме новой игры (new-game menu)

**Save / Load**

**In-game menu**

Вернуться к игре

Сохранить игру

Загрузить игру

Конец игры

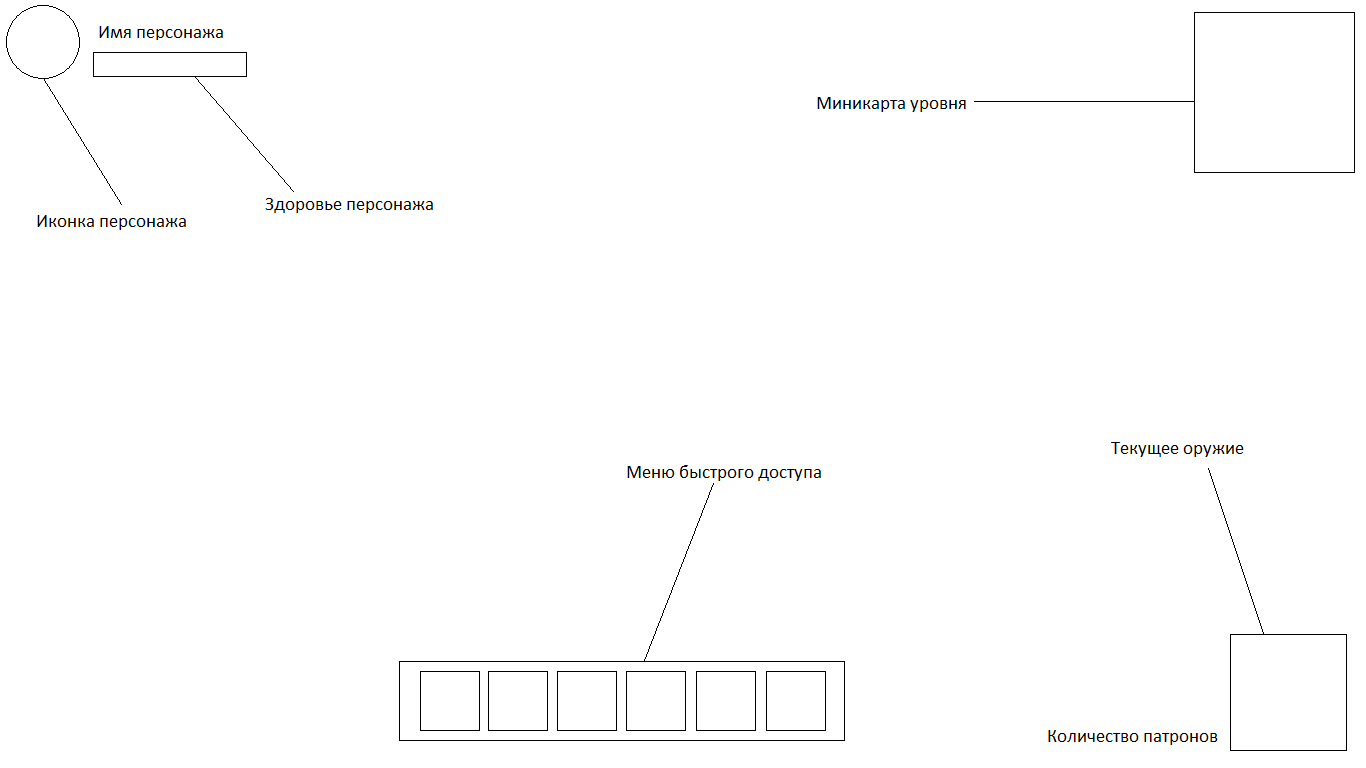
Сохранить / Загрузить

Отмена

**Игра**

**New game menu**

Игровое меню в режиме игры (in-game menu)



Пользовательский интерфейс

### Функциональное описание и управление

Функционирование new game menu и in-game menu не имеет каких-либо особенностей и не требует дополнительно к приведенным схемам описания.

Доступ к элементам интерфейса кроме использования мыши выполняется и по клавиатурным командам:

|  |  |
| --- | --- |
| Клавиша | Результат |
| ESC | Из new game menu – выход (с подтверждением)  Из in-game menu – возврат к игре  Из окон Save/Load, Help, Quest log – закрытие окна и возврат к предыдущему экрану  Из игры – вызов in-game menu |
| Клавиши управления курсором | В меню – выбор активной команды |
| Enter | В меню – ввод выбранной команды |
| I | Из игры – открытие окна инвентаря  Из окна инвентаря – закрытие окна инвентаря |

Управление главным героем проходит с помощью клавиатуры и мыши, мышкой игрок наводит прицелом на цель и атакует, клавиатурой же он двигается персонажем. Управление происходит за счет следующих клавиш:

|  |  |
| --- | --- |
| Клавиша | Результат |
| W | Движение персонажем вверх по карте |
| A | Движение персонажем влево по карте |
| S | Движение персонажем вниз по карте |
| D | Движение персонажем вправо по карте |
| Перемещение мышки | Перемещение прицела персонажа |
| LCL | Атака |
| RCL | Атака метательным оружием (если имеется) |
| MWheelDown | Переключение на предыдущее оружие в слоте быстрого доступа |
| MWheelUp | Переключение на следующее оружие в слоте быстрого доступа |
| 1 | Переключить на / Воспользоваться предметом в слоте быстрого доступа 1 |
| 2 | Переключить на / Воспользоваться предметом в слоте быстрого доступа 2 |
| 3 | Переключить на / Воспользоваться предметом в слоте быстрого доступа 3 |
| 4 | Переключить на / Воспользоваться предметом в слоте быстрого доступа 4 |
| 5 | Переключить на / Воспользоваться предметом в слоте быстрого доступа 5 |
| 6 | Переключить на / Воспользоваться предметом в слоте быстрого доступа 6 |
| Е | Взаимодействие с предметами / юнитами |

### Объекты интерфейса пользователя

Объектами интерфейса пользователя являются:

* Окно – графическая область прямоугольной формы с изображением и расположенными на ней другими объектами. Окнами являются menu, Quest log и другие.
* Кнопка – графическая область прямоугольной или овальной формы с изображением и ассоциированной командой. Может использовать эффект вдавливания при нажатии, а может и нет.
* Зона - графическая область прямоугольной или овальной формы с изображением и ассоциированной командой. Зонами являются игровая область и мини-карта.
* Поле текста – прямоугольная область в рамках окна, содержащая форматированный текст и имеющая кнопки прокрутки текста по вертикали.
* Слот игры – область в меню Save/Load, в которую может быть помещена игра для сохранения и из которой она может быть доступна
* Нефункциональная область – любая область на экране, взаимодействие с которой не дает никакого эффекта (например, информация о состоянии здоровья главного героя).

## Графика и видео

### Общее описание

Общий вид игры должен соответствовать жанру киберпанка – темные мрачные комнаты освещенные яркими неоновыми цветами.

### Двумерная графика и анимация

Для создания игры требуется разработка следующих основных графических частей:

* Интерфейс
* Персонажи, строения и юниты
* Ландшафты
* Статические объекты
* Маркетинговые материалы

## Интерфейс.

К графическим элементам интерфейса относятся:

|  |  |
| --- | --- |
| Элемент | Комментарии |
| Шрифты игры | Основные шрифта игры, включающие латиницу и кириллицу |
| Фон новой игры | После запуска игры на этом фоне отображается Game Menu и другие окна (Save/Load, Credits) |

Продолжение траблицы

|  |  |
| --- | --- |
| Фон игры после неудачного финала | По сути, то же, что и предыдущее, но отражающее результат предыдущей игры – mission failed. Темный фон с крупным белым текстом «You died». |
| Прямоугольные кнопки | Изображения для реализации кнопок меню. Содержат изображения для 4х состояний – normal, down, mouse over, disabled |
| Кнопки «вверх» и «вниз», используемые для прокрутки текста | То же, что и прямоугольные кнопки, но квадратной формы |
| Овальные кнопки игровой панели | Кнопки «Меню», «Ряба» и «Квест». То же, что и прямоугольные кнопки, но овальной формы |
| Курсор | Курсор в виде прицела, по которому игрок будет |
| Игровая панель | Основная игровая панель |

### Трехмерная графика и анимация

В игре не предполагается использование 3D графики. Анимации построены на основе быстрого переключения спрайтов.

### Анимационные вставки и видеосъемки

В игре присутствуют следующие анимационные вставки:

* Плохой финал игры.
* Положительный финал игры.
* Начальная кат-сцена.
* Диалоги.

## Звуки и музыка

### Общее описание

Подход к звуковому и музыкальному оформлению игры опирается на основной принцип создания напряженной атмосферы.

Музыкальное сопровождение основано на использовании современной обработки музыки советской эпохи в жанры synthwave и sovietwave, а также использование бесплатной музыки в данном стиле.

### Звук и звуковые эффекты

Озвучиванию подлежат следующие части:

* Начальное меню
* Тренировочная комната
* Хаб
* Диалог
* Звуки, издаваемые игровыми объектами в течение их жизненного цикла: перемещение, сражение, получение команд и другие

### Музыка

В игре предполагаются музыкальные темы, ограниченные выполнением миссий и состоящие из фрагментов. Фрагменты зациклены и не могут объединяться, чтобы не создавать смешанный эмбиент. Средняя продолжительность каждого фрагмента – 3 минуты.

# Контакты

|  |  |
| --- | --- |
| Контактные лица: | Балев Александр  Межецкий Иван |
|  |  |
| Телефоны: | +7 (921) 37-25-439  +7 (981) 45-92-371 |
|  |  |
| E-mail: | games@centrifuge.com |
|  |  |
| Адрес: | Санкт-Петербург |
|  |  |
| Сайт: | <https://centrifuge.games> |

|  |  |
| --- | --- |
|  |  |

ПРИЛОЖЕНИЕ В

Листинги программного кода приложения

Листинг 7 – Генерация уровня

﻿using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Tilemaps;

using UnityEngine.UI;

public class Generation : MonoBehaviour

{

[SerializeField]

private Grid grid;

[SerializeField]

private Tile groundTile;

[SerializeField]

private Tile pitTile;

[SerializeField]

private Tile[] topWallTile;

[SerializeField]

private Tile bottomWallTile;

[SerializeField]

private Tile leftWallTile;

[SerializeField]

private Tile rightWallTile;

[SerializeField]

private Tile WallTile;

[SerializeField]

private Tile rightBtTile;

[SerializeField]

private Tile rightTpTile;

[SerializeField]

private Tile leftBtTile;

[SerializeField]

private Tile leftTpTile;

[SerializeField]

private Tile dotTRTile;

[SerializeField]

private Tile dotTLTile;

[SerializeField]

private Tile dotBRTile;

[SerializeField]

private Tile dotBLTile;

[SerializeField]

private GameObject[] obstacleTiles;

[SerializeField]

private Tilemap groundMap;

[SerializeField]

private Tilemap pitMap;

[SerializeField]

private Tilemap wallMap;

[SerializeField]

private GameObject player;

[SerializeField]

private GameObject enemy;

[SerializeField]

private GameObject exit;

[SerializeField]

private int deviationRate = 10;

[SerializeField]

private int roomRate = 15;

[SerializeField]

private int obstacleRate = 45;

[SerializeField]

private int enemyRate = 30;

[SerializeField]

private int maxRouteLength;

[SerializeField]

private int maxRoutes = 20;

[SerializeField]

private Text text;

private int seed;

private int lastX;

private int lastY;

private int lastRoomObstacles = 0;

private int lastRoomSize = 0;

private List<Vector2> gridPositions = new List<Vector2>();

private Vector2 exitPos;

public List<Vector2> enemyList = new List<Vector2>();

private int level = 0;

// PathFinding

[SerializeField]

private GameObject notWalk;

private int countTiles = 0;

private Vector3 startPos = Vector3.zero;

private PathfindingSystem pathfinding;

private List<Vector3Int> propPositionsList = new List<Vector3Int>();

public List<Vector2> obstacleList = new List<Vector2>();

// PathFindig

public int routeCount = 0;

public int Seed { get => seed;}

private void Start()

{

CreateSeed();

if(PlayerPrefs.HasKey("seed"))

{

if (PlayerPrefs.GetInt("seed") > 0)

seed = PlayerPrefs.GetInt("seed");

}

Random.InitState(seed);

maxRoutes = Random.Range(30, 50);

int x = 0;

int y = 0;

int routeLength = 0;

GenerateSquare(x, y, 1);

Vector2Int previousPos = new Vector2Int(x, y);

y += 3;

GenerateSquare(x, y, 1);

NewRoute(x, y, routeLength, previousPos);

FillWalls();

PathFinding();

exitPos = new Vector2(lastX + 0.5f, lastY + 0.5f);

while (obstacleList.Contains(exitPos))

{

exitPos = new Vector2(Random.Range(lastX - lastRoomSize, lastX + lastRoomSize + 1) + 0.5f, Random.Range(lastY - lastRoomSize, lastY + lastRoomSize + 1) + 0.5f);

}

exit.transform.position = exitPos;

player.transform.position = new Vector2(0.5f, 1f);

text.text = "seed: " + seed + " maxRoutes = " + maxRoutes + " " + level;

PlayerPrefs.SetInt("seed", 0);

PlayerPrefs.SetInt("loadedSeed", 0);

}

private void CreateSeed()

{

string datetime = System.DateTime.Now.ToString("MM/dd") + System.DateTime.Now.ToString("hh:mm:ss");

string resultString = "";

for (int i = 0; i < datetime.Length; i++)

{

if (datetime[i] >= '0' && datetime[i] <= '9')

resultString += datetime[i];

}

Random.InitState(int.Parse(resultString));

seed = Random.Range(10000, 999999999);

}

private void PathFinding()

{

pathfinding = new PathfindingSystem(pitMap.size.x, pitMap.size.y, 1f, startPos, false);

foreach (Vector3Int pos in propPositionsList)

{

pathfinding.Grid.GetCellIndex(pos, out int xp, out int yp);

pathfinding.GetNode(xp, yp).IsWalkable = false;

}

foreach (Vector2 pos in obstacleList)

{

pathfinding.Grid.GetCellIndex(pos, out int xo, out int yo);

pathfinding.GetNode(xo, yo).IsWalkable = false;

}

}

private void FillWalls()

{

BoundsInt bounds = groundMap.cellBounds;

for (int xMap = bounds.xMin - 11; xMap <= bounds.xMax + 10; xMap++)

{

for (int yMap = bounds.yMin - 11; yMap <= bounds.yMax + 10; yMap++)

{

Vector3Int pos = new Vector3Int(xMap, yMap, 0);

if (startPos == Vector3.zero)

{

startPos = pos;

}

Vector3Int posAbove = new Vector3Int(xMap, yMap + 1, 0);

Vector3Int posBelow = new Vector3Int(xMap, yMap - 1, 0);

Vector3Int posBefore = new Vector3Int(xMap - 1, yMap, 0);

Vector3Int posAfter = new Vector3Int(xMap + 1, yMap, 0);

Vector3Int dotTL = new Vector3Int(xMap + 1, yMap - 1, 0);

Vector3Int dotTR = new Vector3Int(xMap - 1, yMap - 1, 0);

Vector3Int dotBL = new Vector3Int(xMap + 1, yMap + 1, 0);

Vector3Int dotBR = new Vector3Int(xMap - 1, yMap + 1, 0);

TileBase tile = groundMap.GetTile(pos);

TileBase tileBelow = groundMap.GetTile(posBelow);

TileBase tileAbove = groundMap.GetTile(posAbove);

TileBase tileBefore = groundMap.GetTile(posBefore);

TileBase tileAfter = groundMap.GetTile(posAfter);

TileBase tileTL = groundMap.GetTile(dotTL);

TileBase tileTR = groundMap.GetTile(dotTR);

TileBase tileBR = groundMap.GetTile(dotBR);

TileBase tileBL = groundMap.GetTile(dotBL);

if (tile == null)

{

pitMap.SetTile(pos, pitTile);

propPositionsList.Add(pos);

if (tileTL != null)

{

wallMap.SetTile(pos, dotTLTile);

}

if (tileTR != null)

{

wallMap.SetTile(pos, dotTRTile);

}

if (tileBR != null)

{

wallMap.SetTile(pos, dotBRTile);

}

if (tileBL != null)

{

wallMap.SetTile(pos, dotBLTile);

}

if (tileBefore != null)

{

wallMap.SetTile(pos, leftWallTile);

}

if (tileAfter != null)

{

wallMap.SetTile(pos, rightWallTile);

}

if (tileAbove != null)

{

wallMap.SetTile(pos, bottomWallTile);

}

if (tileAbove != null && tileBefore != null)

{

wallMap.SetTile(pos, rightBtTile);

}

if (tileAbove != null && tileAfter != null)

{

wallMap.SetTile(pos, leftBtTile);

}

if (tileBelow != null && tileBefore != null)

{

wallMap.SetTile(pos, leftTpTile);

}

if (tileBelow != null && tileAfter != null)

{

wallMap.SetTile(pos, rightTpTile);

}

if ((tileBefore != null && tileAfter != null) || (tileAbove != null && tileBelow != null))

{

wallMap.SetTile(pos, WallTile);

}

if (tileBelow != null)

{

wallMap.SetTile(pos, topWallTile[Random.Range(0, topWallTile.Length)]);

}

}

}

}

}

private void NewRoute(int x, int y, int routeLength, Vector2Int previousPos)

{

if (routeCount < maxRoutes)

{

routeCount++;

while (++routeLength < maxRouteLength)

{

//Initialize

bool routeUsed = false;

int xOffset = x - previousPos.x;

int yOffset = y - previousPos.y;

int roomSize = 1;

if (Random.Range(1, 100) <= roomRate)

roomSize = Random.Range(3, 6);

previousPos = new Vector2Int(x, y);

//Go Straight

if (Random.Range(1, 100) <= deviationRate)

{

if (routeUsed)

{

GenerateSquare(previousPos.x + xOffset, previousPos.y + yOffset, roomSize);

NewRoute(previousPos.x + xOffset, previousPos.y + yOffset, Random.Range(routeLength, maxRouteLength), previousPos);

}

else

{

x = previousPos.x + xOffset;

y = previousPos.y + yOffset;

GenerateSquare(x, y, roomSize);

routeUsed = true;

}

}

//Go left

if (Random.Range(1, 100) <= deviationRate)

{

if (routeUsed)

{

GenerateSquare(previousPos.x - yOffset, previousPos.y + xOffset, roomSize);

NewRoute(previousPos.x - yOffset, previousPos.y + xOffset, Random.Range(routeLength, maxRouteLength), previousPos);

}

else

{

y = previousPos.y + xOffset;

x = previousPos.x - yOffset;

GenerateSquare(x, y, roomSize);

routeUsed = true;

}

}

//Go right

if (Random.Range(1, 100) <= deviationRate)

{

if (routeUsed)

{

GenerateSquare(previousPos.x + yOffset, previousPos.y - xOffset, roomSize);

NewRoute(previousPos.x + yOffset, previousPos.y - xOffset, Random.Range(routeLength, maxRouteLength), previousPos);

}

else

{

y = previousPos.y - xOffset;

x = previousPos.x + yOffset;

GenerateSquare(x, y, roomSize);

routeUsed = true;

}

}

if (!routeUsed)

{

x = previousPos.x + xOffset;

y = previousPos.y + yOffset;

GenerateSquare(x, y, roomSize);

}

}

}

}

public void Save()

{

PlayerPrefs.SetInt("savedSeed", seed);

Debug.Log("Saved, seed: " + PlayerPrefs.GetInt("savedSeed"));

}

private void GenerateSquare(int x, int y, int radius)

{

lastRoomSize = radius;

Vector3 obstaclePos;

Vector3 enemyPos;

lastX = x;

lastY = y;

for (int tileX = x - radius; tileX <= x + radius; tileX++)

{

for (int tileY = y - radius; tileY <= y + radius; tileY++)

{

Vector3Int tilePos = new Vector3Int(tileX, tileY, 0);

groundMap.SetTile(tilePos, groundTile);

}

}

switch(radius)

{

case 1:

if (Random.Range(0, 100) <= obstacleRate)

{

obstaclePos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

Instantiate(obstacleTiles[Random.Range(0, obstacleTiles.Length)], obstaclePos, Quaternion.identity);

obstacleList.Add(obstaclePos);

}

if (Random.Range(0, 100) <= enemyRate)

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

while (obstacleList.Contains(enemyPos) || enemyList.Contains(enemyPos))

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

}

enemyList.Add(enemyPos);

var Enemy = Instantiate(enemy, enemyPos, Quaternion.identity);

Enemy.GetComponent<Enemy>().target = player.transform;

}

break;

default:

for (int i = 0; i < Random.Range(3, radius); i++)

{

obstaclePos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

while (obstacleList.Contains(obstaclePos) || enemyList.Contains(obstaclePos))

{

obstaclePos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

}

var obstacle = Instantiate(obstacleTiles[Random.Range(0, obstacleTiles.Length)], obstaclePos, Quaternion.identity);

obstacle.GetComponent<ObstacleScript>().grid = grid;

obstacleList.Add(obstaclePos);

}

for (int i = 0; i < Random.Range(1, radius - 1); i++)

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

while (obstacleList.Contains(enemyPos) || enemyList.Contains(enemyPos))

{

enemyPos = new Vector2(Random.Range(x - radius, x + radius + 1) + 0.5f, Random.Range(y - radius, y + radius + 1) + 0.5f);

}

enemyList.Add(enemyPos);

var Enemy = Instantiate(enemy, enemyPos, Quaternion.identity);

Enemy.GetComponent<Enemy>().target = player.transform;

}

break;

}

}

}

Листинг 8 – Логика препятствий

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ObstacleScript : MonoBehaviour

{

[SerializeField]

private int hp = 5;

[SerializeField]

private bool breakable = false;

private Vector2 obstaclePosition;

[SerializeField]

public Grid grid;

private void Start()

{

obstaclePosition = new Vector2(transform.position.x, transform.position.y);

}

private void FixedUpdate()

{

if (hp <= 0 && breakable)

{

Destroy(gameObject);

grid.GetComponent<Generation>().obstacleList.Remove(obstaclePosition);

}

}

public void HpLoss()

{

if (breakable)

{

hp--;

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

switch (collision.gameObject.layer)

{

case 14:

HpLoss();

break;

}

}

}

Листинг 9 – Класс игрока

public class Player : MonoBehaviour

{

public float speed = 3f;

[SerializeField] internal Animator playerAnimator;

private Rigidbody2D rb;

private int hp = 10;

public Canvas GUI;

public Canvas Loss;

public Canvas Pause;

public Image hpBar;

private Vector2 movement;

private Vector2 mousePos;

[SerializeField] private Camera cam;

void Start()

{

rb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

private void FixedUpdate()

{

rb.velocity = Movement();

if(hp == 0)

{

PlayerDead();

}

}

public void PlayerDead()

{

Time.timeScale = 0f;

GUI.enabled = false;

Loss.enabled = true;

Pause.enabled = false;

}

private Vector2 Movement()

{

movement.x = Input.GetAxis("Horizontal");

movement.y = Input.GetAxis("Vertical");

if(movement.x < 0)

{

}

return new Vector2(movement.x \* speed, movement.y \* speed);

}

private float Rotation()

{

mousePos = cam.ScreenToWorldPoint(Input.mousePosition);

Vector2 lookDir = mousePos - rb.position;

float angle = Mathf.Atan2(lookDir.y, lookDir.x) \* Mathf.Rad2Deg;

return angle;

}

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.CompareTag("Exit"))

Application.LoadLevel(Application.loadedLevel);

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.gameObject.layer == 13)

{

hpLoss();

hpBar.fillAmount = hp / 10f;

}

}

private void hpLoss()

{

hp--;

}

}

Листинг 10 – Класс оружия

public class Weapon : MonoBehaviour

{

[SerializeField] private Sprite weaponSprite;

[SerializeField] private Camera cam;

[SerializeField] private GameObject projectTile;

[SerializeField] private float projectTileForce = 20f;

private SpriteRenderer weaponRenderer;

private Vector2 mousePos;

private Rigidbody2D rb;

private Transform firePoint;

void Start()

{

firePoint = GetComponentInChildren<Transform>();

weaponRenderer = gameObject.GetComponent<SpriteRenderer>();

rb = gameObject.GetComponent<Rigidbody2D>();

weaponRenderer.sprite = weaponSprite;

gameObject.transform.localScale = new Vector2(0.3f, 0.3f);

}

void Update()

{

mousePos = cam.ScreenToWorldPoint(Input.mousePosition);

Vector2 lookDir = mousePos - rb.position;

float angle = Mathf.Atan2(lookDir.y, lookDir.x) \* Mathf.Rad2Deg;

this.transform.localRotation = Quaternion.Euler(0f, 0f, angle);

if (Input.GetMouseButtonDown(0))

{

Shooting();

}

}

private void Shooting()

{

GameObject pt = Instantiate(projectTile, firePoint.position, firePoint.rotation);

pt.GetComponent<Rigidbody2D>().AddForce(firePoint.right \* projectTileForce, ForceMode2D.Impulse);

}

}

Листинг 11 – Класс миникарты

public class CameraMovement : MonoBehaviour

{

[SerializeField] private Transform player;

[SerializeField]internal bool isCameraMove = true;

internal CameraSettings settings;

private Vector3 toPosition;

private new Transform camera;

// Start is called before the first frame update

void Start()

{

camera = gameObject.transform;

camera.position = new Vector3(player.position.x, player.position.y, camera.position.z);

settings = this.GetComponent<CameraSettings>();

Cursor.SetCursor(settings.cursorImage,Vector2.zero, CursorMode.Auto);

}

// Update is called once per frame

void FixedUpdate()

{

if (isCameraMove)

{

toPosition = player.position;

toPosition.z = camera.position.z;

camera.position = Vector3.Lerp(camera.position, toPosition, settings.cameraSpeed \* Time.deltaTime);

}

}

}