**1 Постановка задачи**

* 1. **Организационно-экономическая сущность задачи**

Наименование задачи: «Pocket Arcadas»;

Цель разработки: создание сборника аркадных мини-игр;

Назначение: данный продукт создаётся с развлекательной целью, для хорошего проведения времени.

Периодичность использования: в любое время, по желанию пользователя.

Источники и способы получения данных: правила игр.

Обзор существующих аналогичных ПП**:** каждая из мини-игр проекта имеет физический или цифровой аналог в виде игрового аркадного автомата или игрового приложения.

1. Cosmic Chaos – Space Invaders;

2. Hammer Punch – Бей бобра;

3. Run Explorer – Google dinosaur;

4. Increasing Worm – Snake;

5. Wild Hunt – Duck Hunt;

6. Take Two – Найди пару;

7. Pong-pong – Pin-pong;

8. Sea Battle – Морской бой;

9. Lunch Take – сбор фруктов.

Каждая из представленных мини-игр является пародией на представленные аналоги с изменённой графикой и реалезацией.

**1.2 Функциональные требования**

Take Two:

Игроку дают поле с 64 карточкой. Целью игры является нахождение пары картинок на карточках, за один ход игрок может перевернуть только 2 карточки. С начала игры игроку дается N очков, при неверном выборе (когда игрок выбирает 2 несовпадающие карточки) количество очков уменьшается на 50. Среди карточек, расположенных на поле имеются карточки-бонусы: дополнительные очки, открытие карты, открытие пары карт. Игра заканчивается когда все карты на поле открыты, конечным счетом игрока является оставшееся количество очков. Все набранные игроком очки конвертируются во внутриигровую валюту «талоны».

Hammer Punch:

Игроку дано поле с девятью отверстиями. С различной переодичностью из которых могут появиться разные звери, такие как: мышь, крот, бобр, окунь. Целью игрока является при появлении животного из отверстия ударить молотком по нему (нажать на экран). Игра заканчивается, когда у игрока заканчиваются три очка здоровья. Очки здоровья отнимаются, когда игрок не успевает ударить по животному и он залазит к себе в дыру. Очки начисляются, когда игрок попадает по животному, в зависимости от скорости игры (10 очков с начальной скоростью, 20 с увеличенной скоростью, и 50 очков с самой высокой скоростью). Все набранные игроком очки конвертируются во внутриигровую валюту «талоны».

Wild Hunt:

Геймплей заключается в убийстве случайным образом появляющихся птиц на игровом поле. У игрока имеется ограниченное количество выстрелов. Игра заканчивается при использовании всех выстрелов. Игрок может промахнуться, при этом количество доступных выстрелов уменьшится, а очки не будут зачислены. В зависимости от вида птицы начисляется соответствующее количество очков (воробей-5, утка-15, орел-30). Все набранные игроком очки конвертируются во внутриигровую валюту «талоны».

Пользователь:

Выбор игры из списка разблокированных;

Играть в выбранную игру;

Тратить заработанную игровую валюту для открытия новых игр;

Сохранение результата в таблицу рекордов;

Просмотр таблицы рекордов;

Настройки звука/изображения;

Приобретение трофеев или бонусов в лавке;

Размещение трофеев на стене достижений.

**1.3 Эксплуатационные требования**

Требования к применению: поможет весело провести время.

Требования к реализации: Для полной реализации будет использоваться язык C#, с игровым движком Unity.

Требования к надежности: alfa- и beta-тестирования перед выпуском новых обновлений.

Требования к интерфейсу: при запуске будет проигрываться анимация запуска игровых автоматов и мигающих ламп. В главном меню будет находиться несколько кнопок: «пуск» (позволит выбрать игру), «настройки» ( управление настройками звука и изображения), «информация» (информация о всех мини-играх, механиках, авторах и подобной информации). Цвета интерфейса будут представлять собой ретро-неоновую палитру. Весь интерфейс будет исполнен с использованием атрибутики 80-х годов, визуальных ретро-решений.

**1.4 Диаграмма использования**

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования представлена на рисунке 1

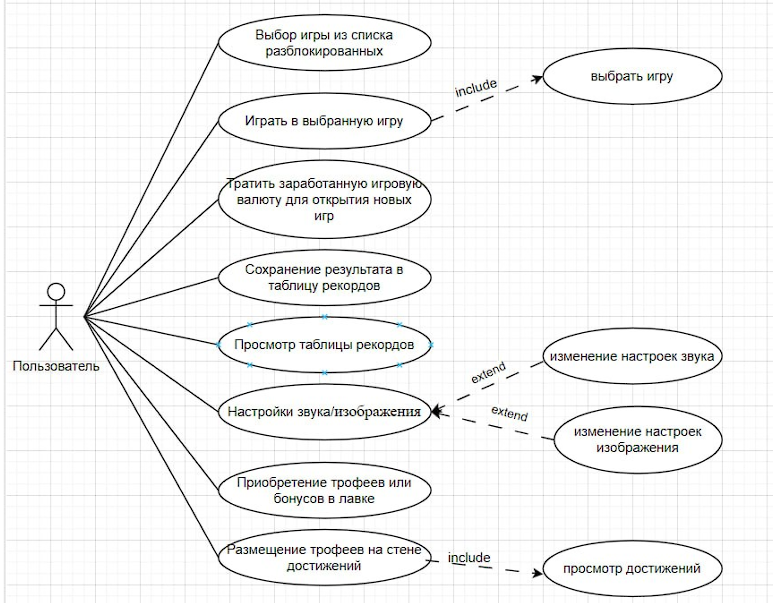


Рисунок 1 – Диаграмма вариантов использования

**1.5** **Выбор стратегии разработки и модели жизненного цикла**

Для разработки мобильной игры «Meme Cat» следует выбрать стратегию разработки и модель жизненного цикла. Осуществляем выбор посредством составления таблиц:

Таблица 1 – Выбор модели жизненного цикла на основе характеристик требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории требований | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да | Нет | Нет | Нет |
| 2. | Могут ли требования быть сформулированы в начале ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3. | Часто ли будут изменяться требования на протяжении ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |
| 4. | Нужно ли демонстрировать требования с целью их определения? | Нет | Нет | Да | Нет | Да | Да |
| 5. | Требуется ли проверка концепции программного средства или системы? | Нет | Нет | Да | Нет | Да | Да |
| 6. | Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Нужно ли реализовать основные требования на ранних этапах разработки? | Нет | Нет | Да | Да | Да | Да |

Вычисления: 2 за каскадную, за 2 V- образную, 5 за RAD, 3 за инкрементную, 5 за быстрого прототипирования и 5 за эволюционную.

Итог: На основе результатов заполнения табл. 3 подходящей является модель быстрого прототипирования и эволюционная.

Таблица 2 – Выбор модели жизненного цикла на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории команды разработчиков  проекта | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Являются ли проблемы предметной области проекта новыми для большинства разработчиков? | Нет | Нет | Нет | Нет | Да | Да |
| 2. | Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков? | Да | Да | Нет | Нет | Нет | Да |
| 3. | Изменяются ли роли участников проекта на протяжении ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5. | Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6. | Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да | Нет | Да | Да | Да |

Вычисления: 4 за каскадную, 4 за V-образную, 1 за RAD, 4 за инкрементную, 3 за быстрого прототипирования и 4 за эволюционную.

Итог: На основе результатов заполнения табл. 4 подходящими являются каскадная, V-образная, инкрементная и эволюционная модели.

Таблица 3 – Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории коллектива пользователей | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1. | Будет ли присутствие пользователей ограничено в ЖЦ разработки? | Да | Да | Нет | Да | Нет | Да |
| 2. | Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки? | Нет | Нет | Нет | Да | Да | Да |
| 3. | Будут ли пользователи вовлечены во все фазы ЖЦ разработки? | Нет | Нет | Да | Нет | Да | Нет |
| 4. | Будет ли заказчик отслеживать ход выполнения проекта? | Нет | Нет | Нет | Нет | Да | Да |

Вычисления: 1 за каскадную, 1 за V-образную, 1 за RAD, 2 за инкрементную, 3 за быстрого прототипирования и 3 за эволюционную.

Итог: На основе результатов заполнения табл. 5 подходящей является модель быстрого прототипирования и эволюционная.

Таблица 6 **–** Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

**2 Проектирование задачи**

**2.1** **Разработка UML-диаграмм**

**2.1.1 Диаграмма последовательности**

Диаграмма последовательности UML — такая диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления. Основные элементы диаграммы последовательности это: обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

Диаграмма последовательности для проекта показана на рисунке 2.

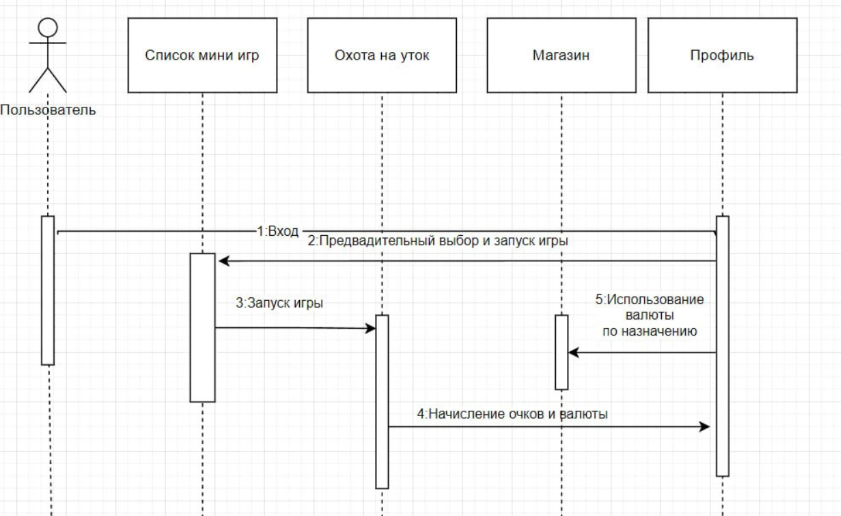
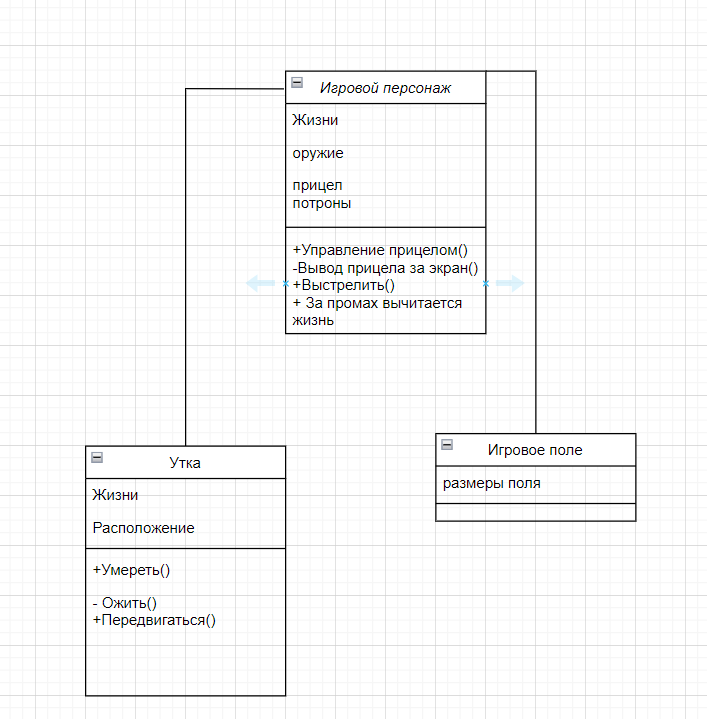


Рисунок 2 – Диаграмма последовательности

**2.1.2 Диаграмма классов**

Диаграмма классов – структурная [диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_(UML)) языка моделирования [UML](https://ru.wikipedia.org/wiki/UML), демонстрирующая общую структуру иерархии [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) системы, их коопераций, [атрибутов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0) (полей), [методов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов для проекта показана на рисунке 3.

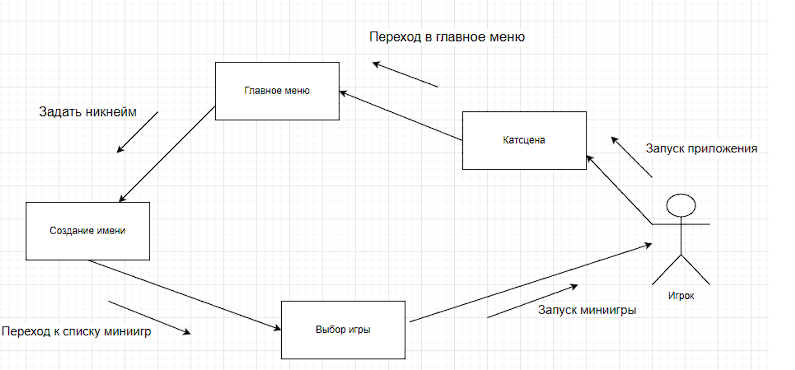
Рисунок 3 – Диаграмма классов

**2.1.3 Диаграмма объектов**

Диаграмма объектов в языке моделирования [UML](https://ru.wikipedia.org/wiki/UML) предназначена для демонстрации совокупности моделируемых объектов и связей между ними в фиксированный момент времени.

Диаграмма объектов описывает конкретные экземпляры объектов и напрямую соотносится с [диаграммой классов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2), которая даёт общее представление о конфигурации системы. Она используется для документирования структур данных и создания статических снимков состояний объектов принимая во внимание реальные экземпляры или прототипы. Динамику поведения объектов обычно изображают в виде последовательности таких диаграмм.

Диаграмма объектов для проекта показана на рисунке 4.

Рисунок 4 – Диаграмма объектов

**2.2 Проектирование ПО**

**2.2.1 Модель данных**

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных.

Модель данных для проекта показана на рисунке 5.

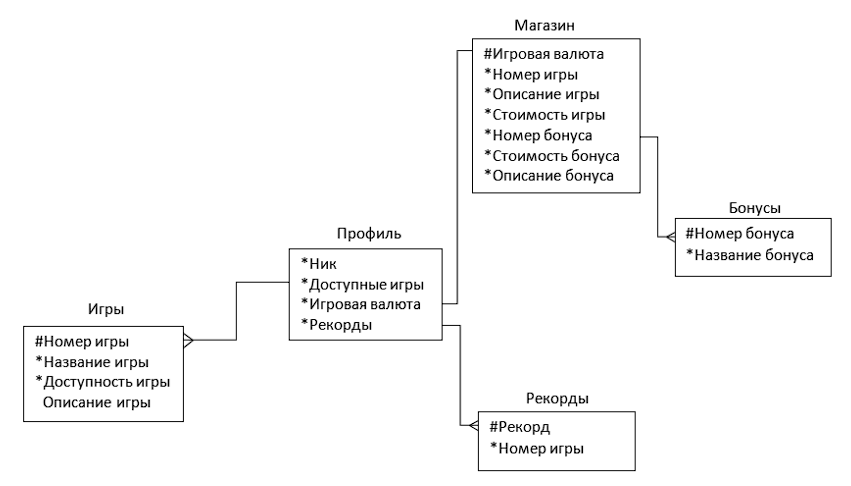
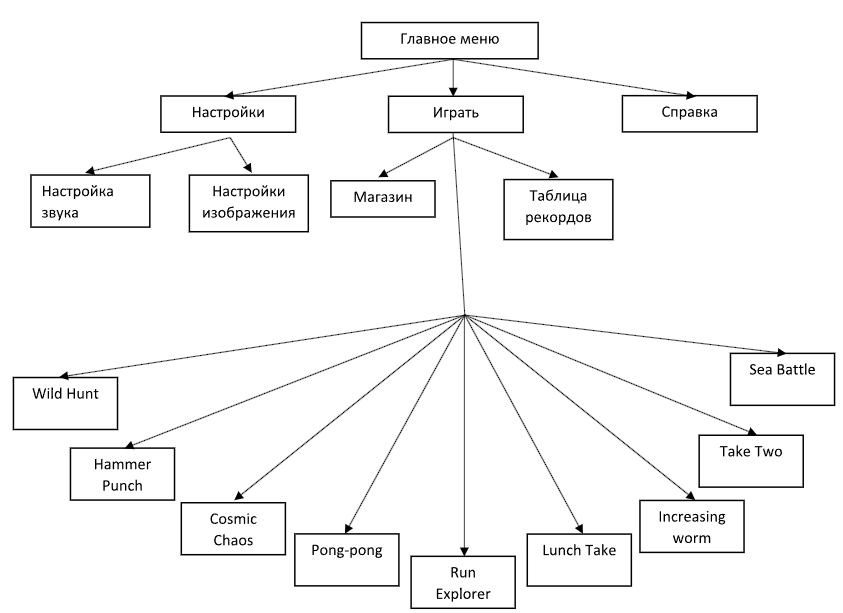


Рисунок 5 – Модель данных

**2.2.2 Структурное проектирование ПО**

Структурное проектирование ПО – это метод определения подсистем, компонентов и способов их соединения, задающий ограничения, при которых система должна функционировать, выбирающий наиболее эффективное сочетание людей, машин и программного обеспечения для реализации системы.

Главное меню для проекта показана на рисунке 6.

****Рисунок 6 – Главное меню

**2.2.3 Функциональная модель**

Функциональная модель представляет собой набор диаграмм потоков данных , которые описывают смысл операций и ограничений.

Функциональная модель для проекта показана на рисунке 7.

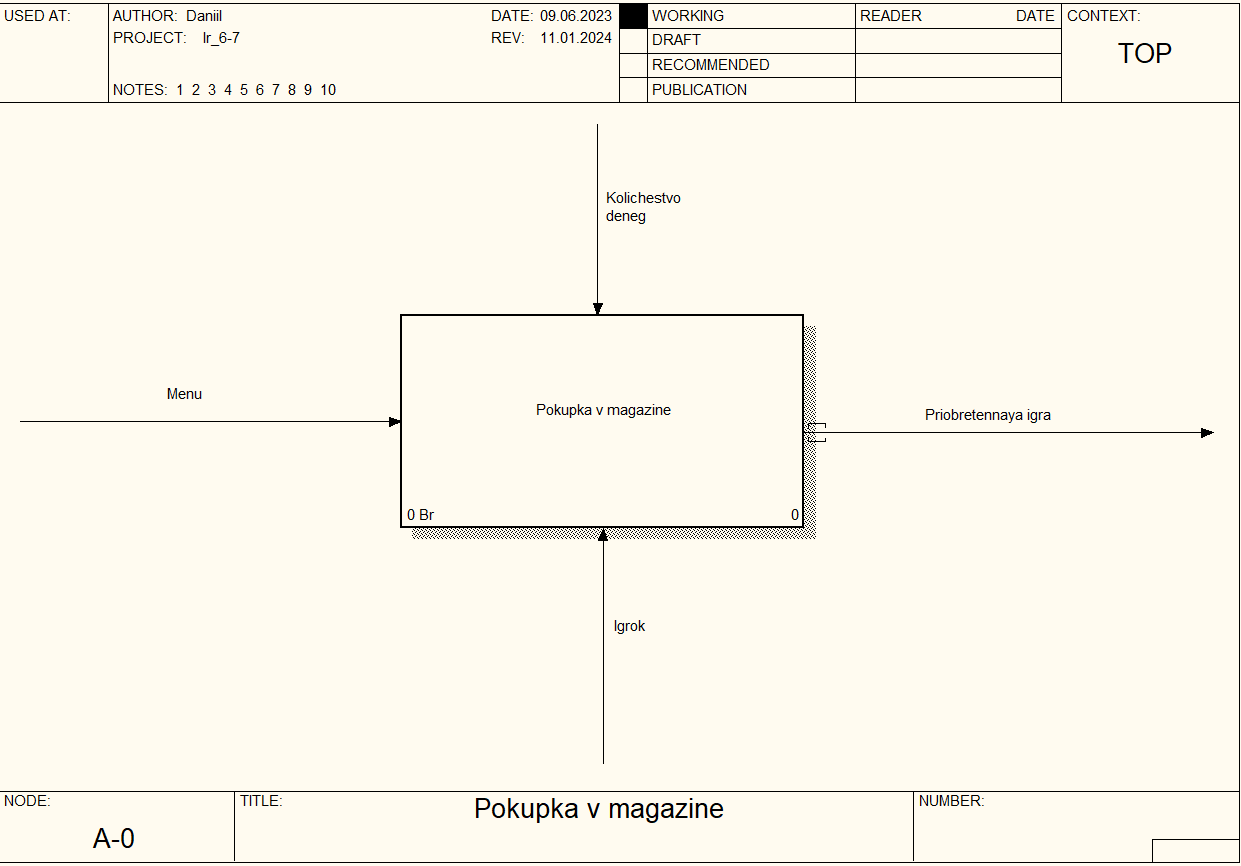


Рисунок 7 – Функциональная модель

**2.3** **Разработка пользовательского интерфейса**

Важным элементом проектирования данного программного продукта является описание внешнего интерфейса разрабатываемого мобильного игрового приложения.

Для разработки визуального дизайн использовались сдержанные, мягкие цвета для удобства использования программного продукта. Для удобного использования данного приложения, все кнопки расположены так что бы не мешать игровому процессу.

С UX/UI можно ознакомится по ссылке ниже. <https://www.figma.com/file/Tr8mxzERIy2CIkjGLZMnTt/Untitled?type=design&mode=design&t=KYU4BKY4AwawFFK8-1>

**3 Реализация**

Для начала нужно скачать и установить Unity и Unity Hub. Для этого переходим на сайт unity.com, регистрируемся, и на главной странице нажимаем кнопку загрузить версию для Windows. Скачав установщик, его нужно установить Unity Hub. Открыв его, вам нужно зайти в аккаунт и скачать новейшую версию Unity.

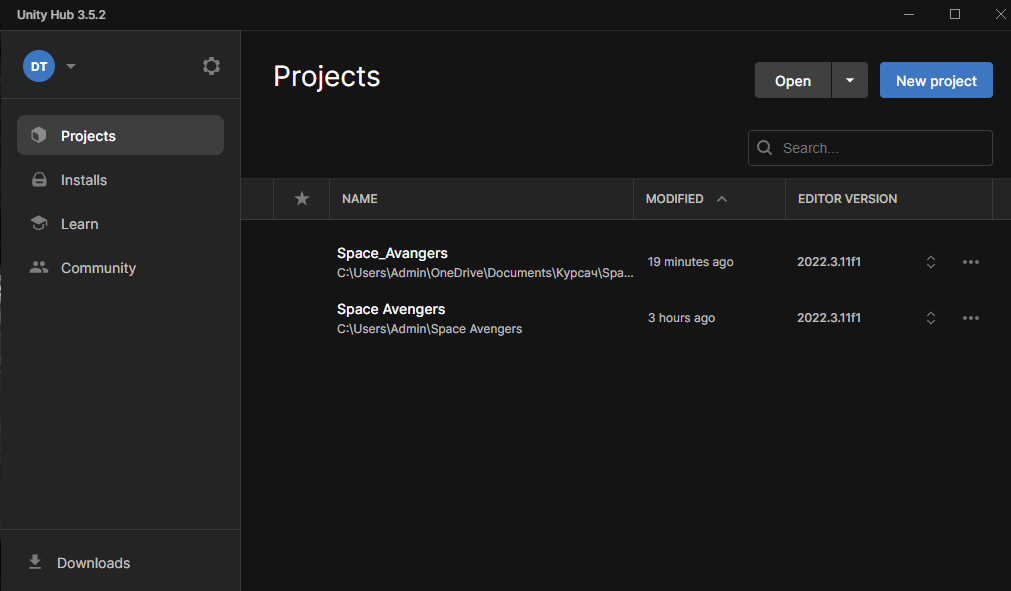
Затем в Unity Hub, открываем раздел Projects и нажимаем кнопку New Project представленную на рисунке 8.

Рисунок 8 – Кнопка New Project

Нажав на кнопку, выбираем 2D Mobile Core, выбираем имя и путь сохранения нового проекта, затем нажимаем Create Project, представлено на рисунке 9.

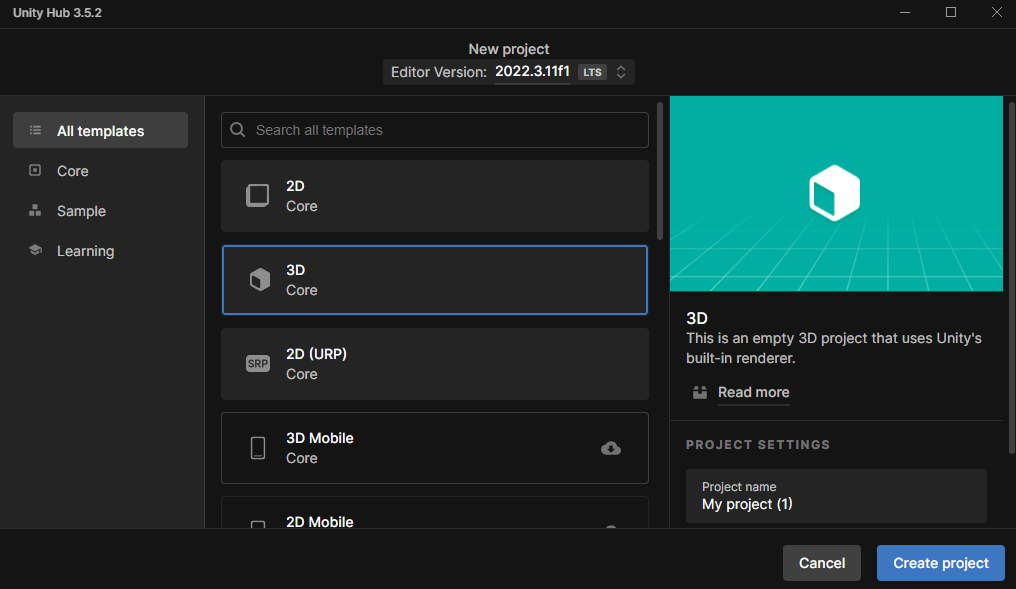


Рисунок 9 – Создание Проекта

Для создания игрового поля необходимо создать сцену и прикрепить к ней камеру. Далее добавить спрайты, поля для жизней и времени. Написать и добавить скрипты. У всех объектов (игрок, враги, выстрелы) есть BoxCollaider, который обозначает объект на поле, что позволяет вовремя обозначать попадание выстрела по объекту, не позволять выходить игроку за пределы поля, а так же удалять лишние объекты, выходящие за пределы поля (выстрелы).

В процессе игры также используются фоновые методы и карутины:

Метод Update – обновление поля с временем используется для движения объектов;

Игровое поле для проекта показана на рисунке 10.

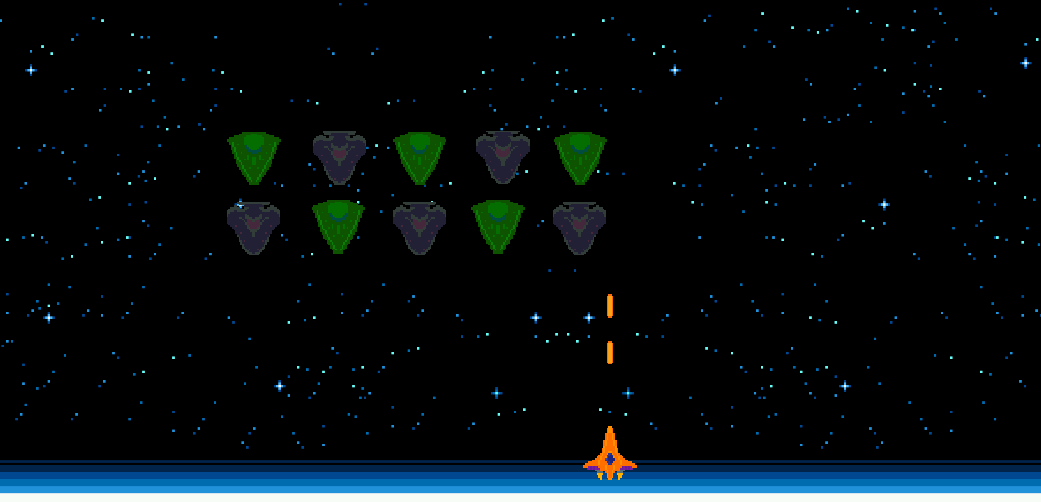


Рисунок 10 – Игровое поле

Настройки игрока для проекта показана на рисунке 11.

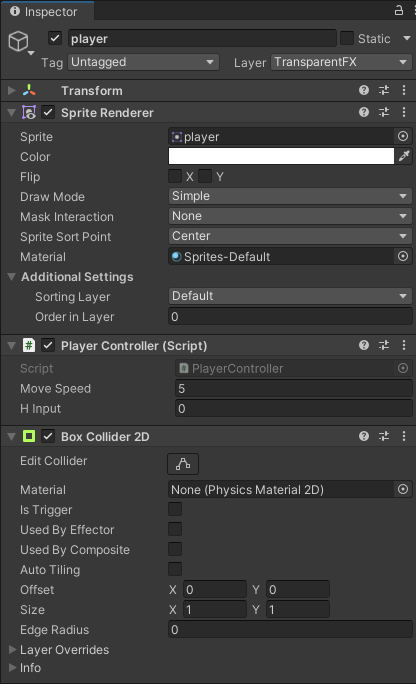
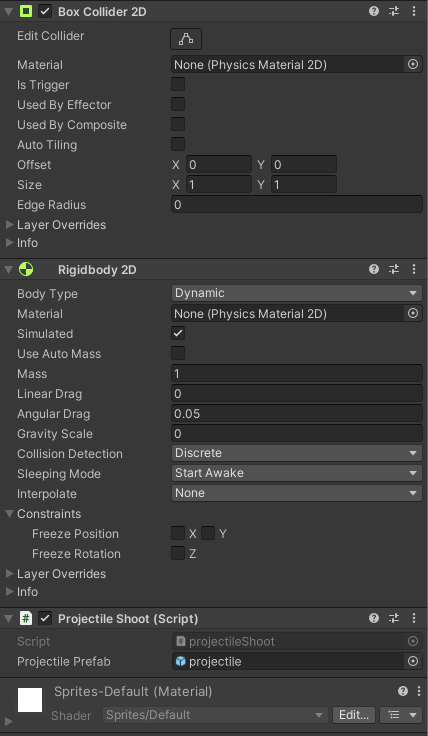


Рисунок 11 – Настройки игрока

**3.3 Руководство пользователя**

Цель игры: уничтожать поступающие вражеские волны в виде вражеских кораблей, набирая игровые очки.

Для начала необходимо запустить игру и нажать на экран. Первые волны врагов начнут наступать на Землю, а игрок должен уклоняясь от вражеских выстрелов, уничтожать вражеские корабли.

Для управления кораблем игрока нужно нажимать на экран телефона в правой стороне для поворота в право, на левую часть экрана - для поворота в левую.

**Для произведения выстрела необходимо нажать на кнопку выстрела.4 Тестирование**

При разработке мобильного игрового приложения «Pocket Arcadas» многие возникающие ошибки и недоработки были исправлены на этапе реализации программного продукта. После завершения испытания реализации мобильного игрового приложения было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программного продукта в автономном режиме. Тестирование ПП представлено в Таблице 4.

Таблица 4 – Отчёт о результатах тестирования функций для пользователя.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название теста | Действие | Ожидаемый результат | Фактический результат | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| Движение игрока вправо | Нажать на правую часть экрана | При нажатии корабль игрока движется вправо | Корабль движется вправо | Выполнено |
| Движение игрока влево | Нажать на левую часть экрана | При нажатии корабль игрока движется влево | Корабль движется влево | Выполнено |
| Выстрел | Нажать на кнопку выстрела | При нажатии кнопки, корабль игрока производит выстрел | Корабль произвел выстрел | Выполнено |
| Убийство врага | Попасть выстрелом во вражеский корабль | При попадании вражеский корабль взрывается | Корабль врага был повержен | Выполнено |
| Трата жизней | Подставиться под выстрел врага | Отнимется одна жизнь | Жизнь отнялась | Выполнено |
| Смерть игрока | Потратить все жизни | Появится экран поражения | Жизни потратились, появилось сообщение о проигрыше с рекордом | Выполнено |
| Счёт рекорда | Набрать большой рекорд | Рекорд отобразится полностью | Счёт полностью отобразился, текст адаптируется в размере | Выполнено |

При тестировании программного продукта, в первую очередь нужно обратить внимание на правильную работу всего ПП. Показываются ли все изображения, виден ли текст, нет ли лишних пробелов и больших отступов

В результате проведения тестирования выяснилось, что все ранее оговоренные функции и требования, были разработаны, а также протестированы. Тесты показали, что все функции работают правильно. Следовательно разработанный ПП можно опубликовать.

В ходе тестирования программного обеспечения продукта на разных устройствах не было выявлено каких-либо ошибок, так как адаптивность ПП была проведена на всех стадиях разработки.