**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: .

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**

Lunch Take:

Геймплей заключается в сборе фруктов. Целью игрока заключается в сборе максимального количества фруктов и ничего лишнего на не ограниченное время. За каждый собранный фрукт начисляются небольшое количество очков, в размере его значимости и увеличивается скорость игры. Игроку предоставляются 3 жизни, если же игрок потеряет их, то он начинает заново. Если же игрок собрал не фрукт, а камень(бомбочку), то очки пропадают (пропадает жизнь), в последствии приведёт к проигрышу. У каждого фрукта есть своя значимость: легендарные фрукты – 50 очков; редкие фрукты – 20 очков; обычные фрукты – 10 очков; камень(бомбочка) – потеря очков (пропадает жизнь). Игра является бесконечной, а все набранные игроком очки являются показателем рекордного значения. Игра завершается, при проигрыше игрока. Все набранные игроком очки конвертируются во внутриигровую валюту «талоны».

Pong-pong:

Геймплей заключается в отбивания мячика во вражескою территорию. Целью игрока заключается в победе над соперником. За каждый забитый мяч, игроку начисляются 100 очков. Если же игрок пропустит мяч, то он потеряет жизнь или же это приведёт к проигрышу. За каждый проигрыш игрок теряет 250 очков. Игра завершается, когда игрок забьёт 5 мячей и победит или же, когда игрок пропустит 3 мяча и проиграет. Все набранные игроком очки конвертируются во внутриигровую валюту «талоны».

Increasing Worm:

Геймплей заключается в съедании фруктов, увеличивая длину своего тела. Цель игрока: съесть все необходимые фрукты и не врезаться. Победа засчитывается, когда игрок съест все фрукты и не во что не врежется. За каждое врезание в препятствия, игрок теряет жизнь и очки. У игрока 3 жизни. Если же игрок потеряет 3 жизни, он проиграет и начнёт сначала. За каждый фрукт игрок получает 50 очков, если же врезается, то теряет 100 очков.

Все набранные игроком очки конвертируются во внутриигровую валюту «талоны».

Магазин:

В игре реализована «торговая лавка», в которой игрок может приобрести за внутриигровую валюту трофеи каждой из мини-игр, при условии, что будут выполнены достижения определенного количества очков. В лавке так же можно приобрести ограниченные бонусы: увеличение получаемой внутриигровой валюты, возможность продолжить игру после поражения и некоторые визуальные эффекты изображения (цветовые схемы).

Пользователь:

1. Выбор игры из списка разблокированных;
2. Играть в выбранную игру;
3. Тратить заработанную игровую валюту для открытия новых игр;
4. Сохранение результата в таблицу рекордов;
5. Просмотр таблицы рекордов;
6. Настройки звука/изображения;
7. Приобретение трофеев или бонусов в лавке;
8. Размещение трофеев на стене достижений.
   1. **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.

**2 Проектирование**

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

Для разработки игрового приложения 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. 1 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. 2 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 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Продолжение таблицы 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 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. 3 подходящей является модель быстрого прототипирования и эволюционная.

Таблица 4 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории типов проекта и рисков | Каскадная | V-образная | RAD | Инкрементная | Быстрого прототипирования | Эволюционная |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1. | Разрабатывается ли в проекте продукт нового для организации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2. | Будет ли проект являться расширением существующей системы? | Да | Да | Да | Да | Нет | Нет |
| 3. | Будет ли проект крупно- или среднемасштабным? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Ожидается ли длительная эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |
| 5. | Необходим ли высокий уровень надежности продукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6. | Предполагается ли эволюция продукта проекта в течение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
|  | | | | | | | |
| Продолжение таблицы 4 | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 7. | Велика ли вероятность изменения системы (продукта) на этапе сопровождения? | Нет | Нет | Нет | Да | Да | Да |
| 8. | Является ли график сжатым? | Нет | Нет | Да | Да | Да | Да |
| 9. | Предполагается ли повторное использование компонентов? | Нет | Нет | Да | Да | Да | Да |
| 10. | Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |

Вычисления: 3 за каскадную, 2 за V-образную, 6 за RAD, 6 за инкрементную, 7 за быстрого прототипирования и 6 за эволюционную.

Итог: На основе результатов заполнения табл. 4 подходящей является модель быстрого прототипирования.

Общий итог: в итоге заполнения табл. 1 – 4 наиболее подходящей является модель быстрого прототипирования и эволюционная.

**2.2 Инструменты разработки**

Для разработки данного проекта будет выбрана среда Unity, так как это самое удобная и доступная среда разработки на данный момент. Unity – больше, чем движок, это среда для разработки компьютерных игр, в которой объединены различные программные средства, используемые при создании ПО – текстовый редактор, компилятор, отладчик и так далее. При этом, благодаря удобству использования, Unity делает создание игр максимально простым и комфортным, а мультиплатформенность движка позволяет игроделам охватить как можно большее количество игровых платформ и операционных систем. В первую очередь, как мы уже упоминали, движок Unity дает возможность разрабатывать игры, не требуя для этого каких-то особых знаний. Здесь используется компонентно-ориентированный подход, в рамках которого разработчик создает объекты (например, главного героя) и к ним добавляет различные компоненты (например, визуальное отображение персонажа и способы управления им). Благодаря удобному Drag & Drop интерфейсу и функциональному графическому редактору движок позволяет рисовать карты и расставлять объекты в реальном времени и сразу же тестировать получившийся результат.

Второе преимущество движка – наличие огромной библиотеки ассетов и плагинов, с помощью которых можно значительно ускорить процесс разработки игры. Их можно импортировать и экспортировать, добавлять в игру целые заготовки – уровни, врагов, паттерны поведения ИИ и так далее. Никакой возни с программированием.

Третья сильная сторона Unity – поддержка огромного количества платформ, технологий, API. Созданные на движке игры можно легко портировать между ОС Windows, Linux, OS X, Android, iOS, на консоли семейств PlayStation, Xbox, Nintendo, на VR- и AR-устройства.

Инструменты, используемые при разработке и написании сопутствующей документации:

Excel – для построения диаграммы Ганта

WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм.

Figma – будет использована для проектирования интерфейса игрового приложения.

Microsoft Office Word – для написания документации к программному продукту.

Microsoft Office PowerPoint – для создания презентации, которая будет использована на защите проекта.

Программа обладает удобным и интуитивно понятным интерфейсом, а также полным набором необходимых функций для создания профессиональных инсталляторов с минимальным размером, высокой степенью сжатия файлов и приятным интерфейсом.

Помимо стандартного минимума, Smart Install Maker позволяет редактировать системный реестр и INI-файлы, создавать программные ярлыки, запускать ассоциируемые и исполняемые файлы, регистрировать новые шрифты и ActiveX компоненты, отображать тексты информации и лицензионного соглашения. Также, с помощью этой утилиты, можно создать мульти языковые инсталляторы с поддержкой более 20-ти популярных языков мира.

Adobe IIIustrator - это популярная среди художников программа для рисования. Удобна своей простотой и в то же время большим функционалом. Программа очень сильно распространена в использовании во всем мире и предпочитается рисующими людьми как один из самых удобных инструментов для создания рисунков. В нем можно сохранить файлы картинок во множестве вариантов (.png, .jpg, .bmp и так далее).

Также есть возможность сохранять изображение как файл фотошопа, заносить все в последнюю указанную программу и работать со слоями проекта, ничего не потеряв.

Adobe Photoshop CS6 x64 – редактор для редактирования изображений. Необходим для оформления некоторых элементов интерфейса. Также использовался и для создания элементов программного продукта «Загадки Эйнштейна». Сложен, но удобен своей многофункциольнальностью.

Aseprite – это редактор изображений [с доступным исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), предназначенный в первую очередь для рисования и анимации в стиле [пиксель-арт](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D1%81%D0%B5%D0%BB%D1%8C-%D0%B0%D1%80%D1%82). Он работает на [Windows](https://ru.wikipedia.org/wiki/Windows), [macOS](https://ru.wikipedia.org/wiki/MacOS" \o "MacOS) и [Linux](https://ru.wikipedia.org/wiki/Linux) и содержит различные инструменты для редактирования изображений и анимации, такие как слои, кадры, поддержка тайловых карт, интерфейс командной строки и другие.

Разработка проекта ведется на ноутбуке Honor. У данного ноутбука следующие параметры:

- процессор AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz;

- объем ОЗУ 16 гб;

- объем места на SDD – 500 гб ;

- ОС – Windows 11.

Как видно разрабатываемое приложение не очень требовательно к аппаратным ресурсам, что, является большим плюсом.

**2.1 Разработка UML-диаграмм**

**2.1.1 Диаграмма вариантов использования**

Проектирование – один из важных шагов при разработке программы, который очень часто игнорируется начинающими разработчиками. Как результат, у них нет чёткого плана дальнейших действий, и проект может быть отложен.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег (рисунок 1).

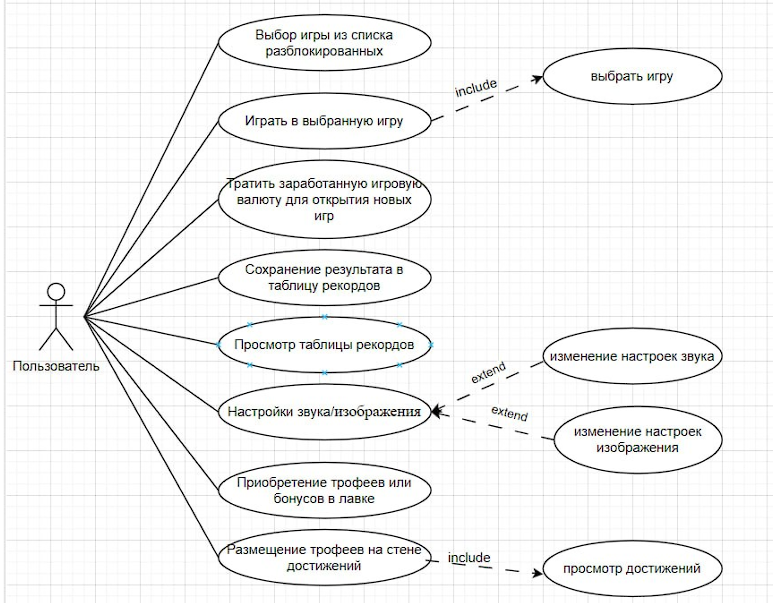
На диаграмме вариантов использования изображаются:

- акторы — группы лиц или систем, взаимодействующих с системой;

- варианты использования — сервисы, которые наша система предоставляет акторам;

- комментарии;

- отношения между элементами диаграммы.

Рисунок 1 – Диаграмма вариантов использования

**2.1.2 Главное меню**

Главное меню – один из важных шагов при разработке программы, который очень часто игнорируется начинающими разработчиками. Как результат, у них нет чёткого плана дальнейших действий, и проект может быть отложен.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег.

В главном меню реализована работа программы (рисунок 2).

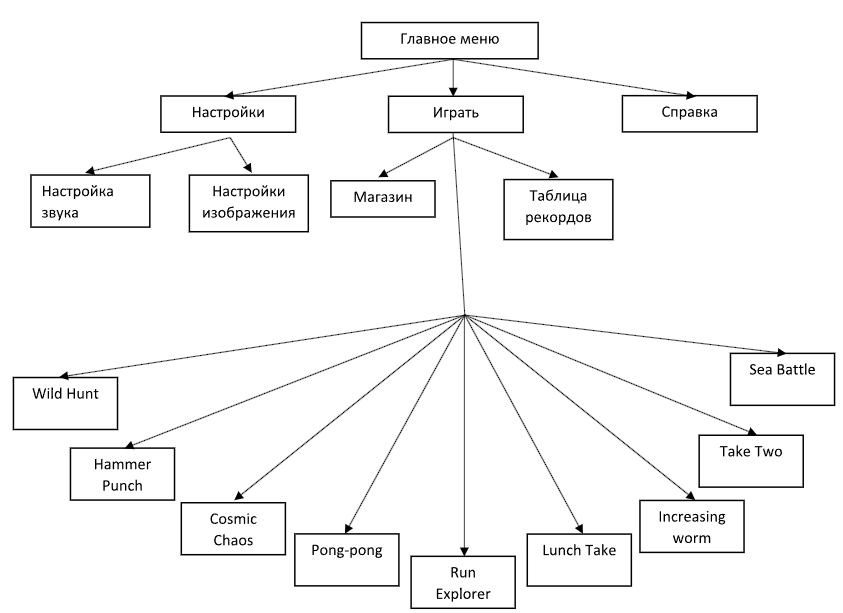
****

Рисунок 2 – Главное меню

**2.1.3 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.

Модели данных служат для проектирования структуры постоянных хранилищ данных, используемых системой. Профайл на языке UML для проектирования базы данных предоставляет разработчикам базы данных набор элементов моделирования, позволяющих разрабатывать подробный макет таблиц в базе данных и моделировать макет физической памяти базы данных. Профайл базы данных на языке UML также предоставляет конструкции для моделирования целостности по ссылкам (ограничений и триггеров), а также хранимых процедур, предназначенных для управления доступом к базе данных.

Модели данных могут создаваться на уровне предприятия, отдела или отдельного приложения. Модели данных на уровне предприятия или отдела могут использоваться для предоставления стандартных определений для ключевых бизнес-сущностей (таких как клиент и сотрудник), которые будут применяться всеми приложениями всего бизнес-процесса или его части. С помощью этих типов Моделей данных можно также определить, какая система предприятия будет "владельцем" данных для конкретной бизнес-сущности и какие другие системы будут пользователями данных.

С моделью данных созданную для игрового приложения «Pocket Arcados» можно ознакомиться в рисунке 3.

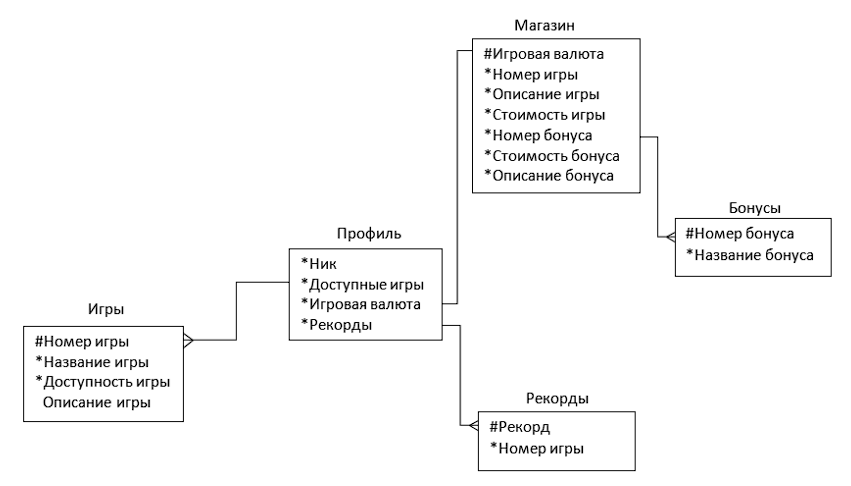


Рисунок 3 – Модель данных

**2.1.4 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.

При разработке программного продукта была поставлена задача разработать диаграмму объектов, которая описывает процесс связей между ними в фиксированный момент времени. С данной диаграммой можно ознакомиться в рисунке 4.

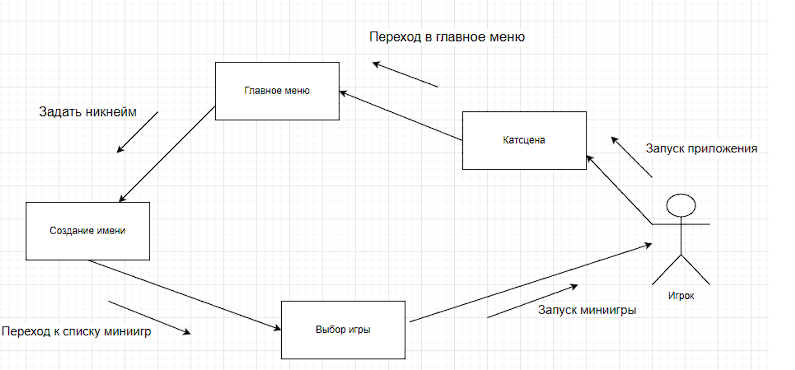
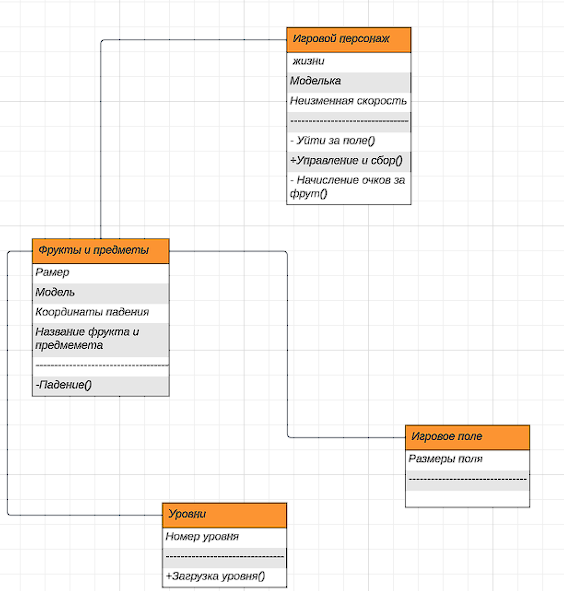


Рисунок 4 – Диаграмма объектов

**2.1.5 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.

При разработке программного продукта была поставлена задача разработать диаграмму классов, которая показывает 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. С данной диаграммой можно ознакомиться в рисунке 5.

Рисунок 5 – Диаграмма классов

**2.1.6 Модель данных**

Модель данных – один из важных шагов при разработке программы, который очень часто игнорируется начинающими разработчиками. Как результат, у них нет чёткого плана дальнейших действий, и проект может быть отложен.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег.

В модели данных схематично показана взаимодействие объектов программы (рисунок 6).

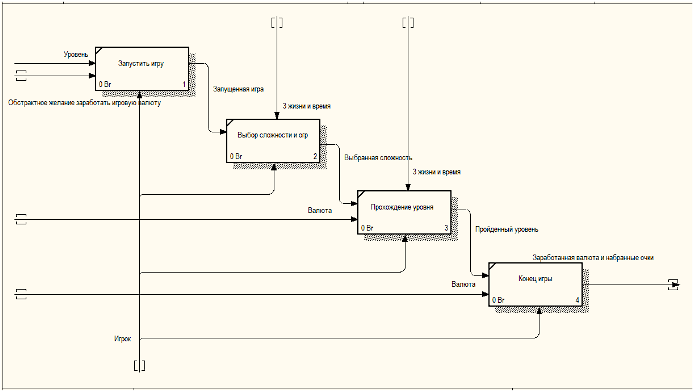
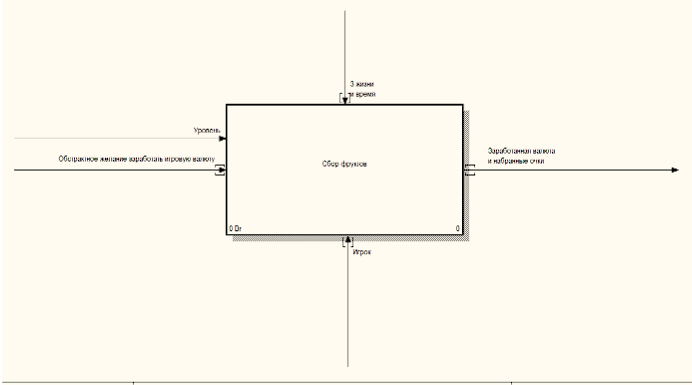


Рисунок 6 – Функциональная модель

**3 Пользовательский интерфейс**

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

Для разработки визуального дизайн использовались сдержанные, мягкие цвета для удобства использования программного продукта.

В ходе разработки был спроектирован дизайн главной страницы игрового приложения «Pocket Arcados».

Для организации эффективной работы пользователя нужно создать целостный программный продукт данной предметной области, в котором все компоненты будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя. Игровое приложение должно позволить пользователю решать задачи, затрачивая значительно меньше усилий, чем при работе с разрозненными объектами. Все исходные данные будут разделены на несколько групп.

Прототип – это наглядная модель пользовательского интерфейса. В сущности, это «черновик» созданный на основе представления разработчика о потребностях пользователя. Итоговое отображение программы может отличаться от прототипа.

Так же поставленной задачей на практику была реализация UX/UI интерфейсов. При создании UX/UI интерфейсов были использованы модульные сетки с целью создания пропорционального, понятного интерфейса. Использовались преимущественно оттенки красок 90х годов. Основные разделы доступны с первой страницы. Таким образом был реализован понятный пользовательский интерфейс, созданы макеты под такие устройства как: компьютер. Целью проекта была реализация интерфейса для фронтенд части.

https://www.figma.com/file/sxDi8oCXQFygKd87QrugSO/%D0%98%D0%B3%D1%80%D1%8B?type=design&node-id=0%3A1&mode=design&t=yFshMEROm6oZvzfi-1

**4 Прототип программного продукта**

Данный программный продукт был разработан с помощью игрового движка Unity с использованием языка программирования C#. Так как Unity прост в освоении для начинающих программистов, он имеет удобный и понятный интерфейс, много различных библиотек, ассетов и так далее. Данный движок даёт неограниченные возможности в создании программных продуктов, имея обширную базу настроек, библиотек, физики, спрайтов, что позволяет создавать достаточно качественные проекты.

**5 Документация на программный продукт**

**5.1Тестирование**

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения этапа написания программы было проведено тщательно функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме. Отчёт о результатах тестирования представлен в таблице 1.

Таблица 5 – Отчёт о результатах тестирования функций для пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название теста | Действие | Ожидаемый результат | Фактический результат | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| Запуск игры | Нажатие на кнопку «Start» | Запуститься игра | Запустилась игра | Выполнено |
| Выход в главное меню | Нажатие на кнопку «Exit» | Выйдет из текущей сцены в главное меню | Выход из текущей сцены в главное меню | Выполнено |
| При нажатии на кнопку «Settings», запуститься панель настройки | Нажатие на кнопку «Settings» | Откроется панель настройки | Открылась панель настройки | Выполнено |
| Смена направления движения вверх | Нажатие на кнопку «Up» | Червячок поползёт вверх | Червячок пополз вверх | Выполнено |
|  |  |  |  |  |
|  |  |  |  |  |
| Продолжение таблицы 5 | | | | |
| 1 | 2 | 3 | 4 | 5 |
| Смена направления движения вправо | Нажатие на кнопку «Right» | Червячок поползёт вправо | Червячок пополз вправо | Выполнено |
| Смена направления движения вниз | Нажатие на кнопку «Down» | Червячок поползёт вниз | Червячок пополз вниз | Выполнено |
| Смена направления движения влево | Нажатие на кнопку «Left» | Червячок поползёт влево | Червячок пополз влево | Выполнено |
| Запуск игры повторно | Нажатие на кнопку «Retry» | Игра повторно запуститься | Игра повторно запустилась | Выполнено |
| Выход в главное меню | Нажатие на кнопку «Exit» | Выйдет в главное меню | Вышло в главное меню | Выполнено |

Тест-кейсы

При запуске игры, нажав на кнопку «Start», включиться таймер, который считывает время проведённое в игры, начиная с нуля. После запуска, игрок может управлять червячком нажимая на кнопки «up», «left», «right», «down». Во время игры, её можно приостановить нажав, на кнопку «Settings».

При встречи червячка с фруктом, он в него вползает, имитируя съедание фрукта, при этом увеличивая длину тела. После того, как он его съел, начисляется 1 балл. Если очки превышают рекорд, то рекорд начисляется, равный очкам. Рекорд при первом запуске игры равен нулю.

При встречи червячка с препятствием или его телом, он погибает, т.е игрок проигрывает. Отнимается жизнь, игра приостанавливается, записывается рекорд по набранным очкам, открывается панель настройки, в которой можно запустить игру повторно и выйти в главное меню.

Отчёт о результатах тестирования

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 разработанное игровое приложение можно передать заказчику.

В ходе тестирования программного обеспечения продукта на разных устройствах не было выявлено каких-либо ошибок на всех стадиях разработки.

**5.2 Руководство пользователя**

**5.2.1 Общие сведения о программном продукте**

Данный программный продукт предназначен для всех пользователей, без ограничения в возрасте. Он предназначен для пользователей, заинтересованных в играх.

«Pocket Arcados» состоит из 9 мини-игр:

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.

Суть режима игры «Increasing Worm»: играя за червячка, нужно съедать фрукты и не врезаться в препятствия, съев фрукт начисляются балы, тем самым надо набрать большее количество очков.

Суть режима игры «Lunch Take»: играя за игрового персонажа, нужно собирать только фрукты, а не другие объекты, например бомбочка, за которые начисляются или уменьшаются балы, тем самым надо набрать большее количество очков.

Суть режима игры «Pin-pong»: управляя ракеткой, нужно отбивать мячик от другой ракетки, имитируя игру в пин-понг, и забивать на сторону противника. Если противник промахнётся, то начисляются балы, если игрок промахнётся, то игрок потеряет балы. То есть надо набрать большее количество очков.

После все очки конвертируются в валюту, с помощью которой мы сможем открывать другие мини игры.

Быстродействие любого игрового приложения во многом зависит от мощности персонального мобильного устройства. Несмотря на все реализованные задачи, игровое приложение легко запускается и функционирует на мобильных устройствах.

Тестирование проводилось на разных классах ЭВМ. Игровое приложение разработано на ПК со следующими характеристиками:

- процессор AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz;

- объем ОЗУ 16 гб;

- объем места на SDD – 500 гб ;

- ОС – Windows 11.

Данное игровое приложение предназначено для мобильных устройств на Android 4.0.0. Так как данное игровое приложение не требует большей производительности.

**5.2.2 Руководство программиста**

Данный программный продукт был разработан с помощью игрового движка Unity с использованием языка программирования C#. Так как Unity прост в освоении для начинающих программистов, он имеет удобный и понятный интерфейс, много различных библиотек, ассетов и так далее. Данный движок даёт неограниченные возможности в создании программных продуктов, имея обширную базу настроек, библиотек, физики, спрайтов, что позволяет создавать достаточно качественные проекты.

**5.2.3 Создание приложения**

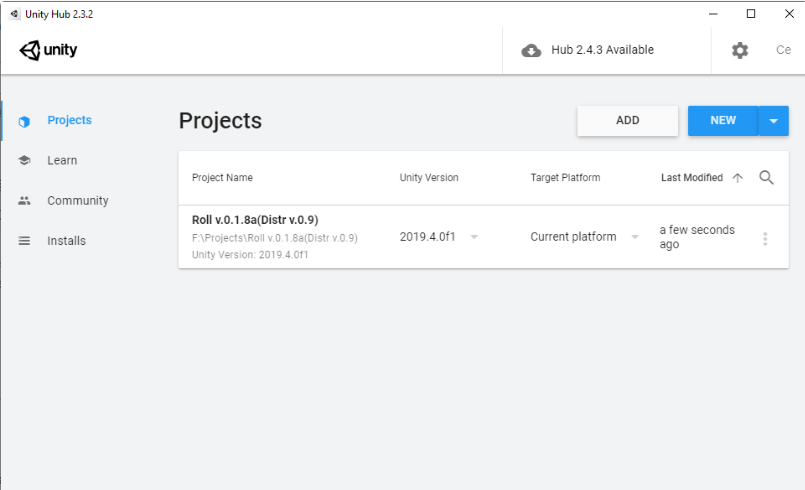
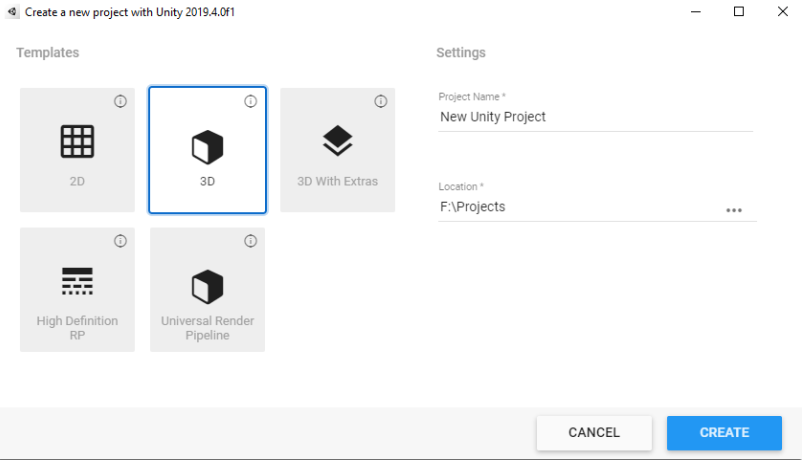
Для того, чтобы создать своей проект в Unity, необходимо сначала, открыть Unity Hub, предварительно его установив, и зарегистрироваться в нём (рисунок 7).

Рисунок 7 – Меню Unity Hub

В основной области находятся поле с проектами и еще правее кнопки «Add» и «New», с выпадающим меню.

Кнопка «Add» отвечает за добавление, уже имеющихся, проектов на вашем компьютере и при нажатии предложит выбрать папку с проектом. Нужная нам кнопка «New» отвечает за создание нового проекта, нажимаем на нее.

Открывается окно с предустановками проекта, здесь можно выбрать один из 5 подготовленных шаблонов для создания игры. Выбираете конечно тот, который подходит под вашу идею. Не забудьте указать имя проекта и папку, где будут храниться все его файлы (рисунок 8).

Рисунок 8 – Создание проекта

Далее нажимаем кнопку «Create» и дожидаемся окончания процесса инициализации.

После окончания загрузки вы можете начинать создавать все внутриигровые объекты (рисунок 9).

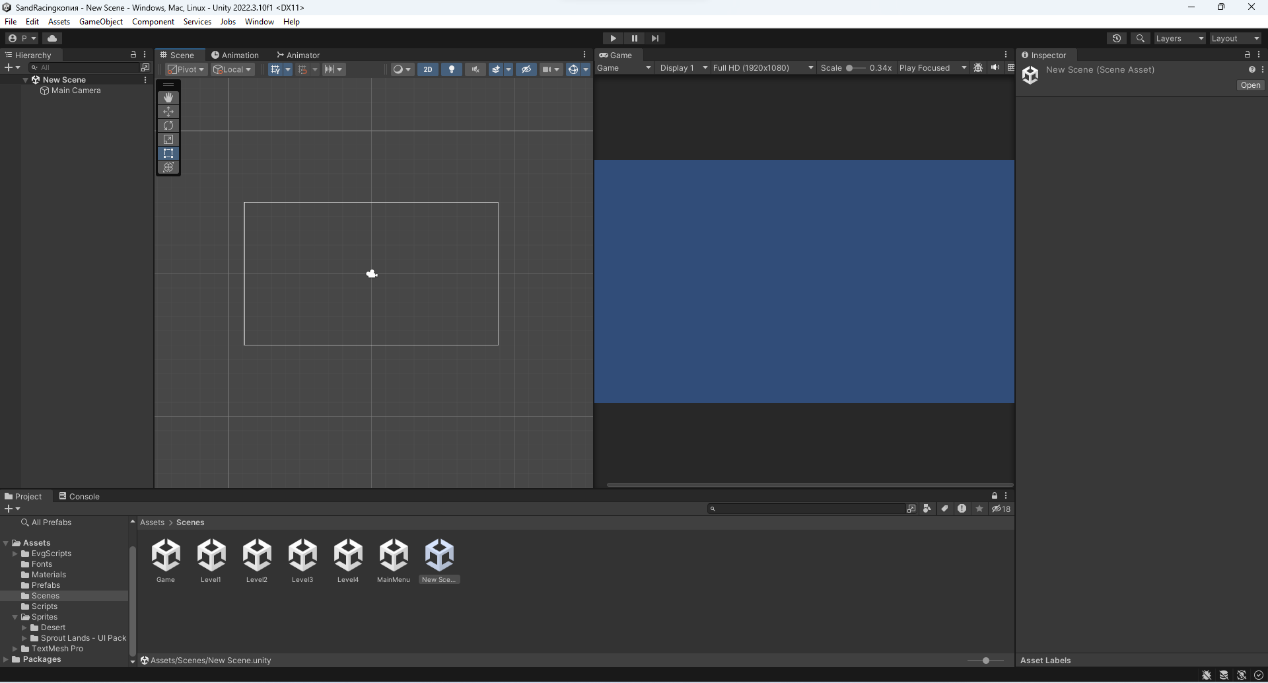


Рисунок 9 – Созданный проект Unity

**5.2.3.1 Настройка функциональности и дизайна**

В Unity проекта находиться множество панелей отвечающие за ту или иную функцию. Опишем нужные функции и задачи для создания нашего проекта:

- Hierarchy – иерархия объектов – здесь можно добавлять, копировать, создавать объекты;

- Build Settings – настройки сцен, и установщик;

- Build and Run – создание сборки приложения и запуск его;

- Scene – панель, на которой располагаются все созданные объекты, которые можно редактировать;

- Game – симулятор панели, на которой отображаться расположенные объекты в виде готового проекта;

- Inspector – панель, в которой можно добавлять, изменять или удалять объекты и скрипты;

- Project – иерархия файлов;

- Console - панель отображения запуска и сбоев компонентов проекта;

- Scripts – папка в которой создаются C# скрипты для самого проекта.

**5.2.4 Реализация 3 режимов игры**

Первым шагом создаём 3 режима игры.

При создании их надо создать все необходимые кнопки, компоненты, задний фон, к ним же прикрепить все необходимые скрипты (рисунок 10-12).

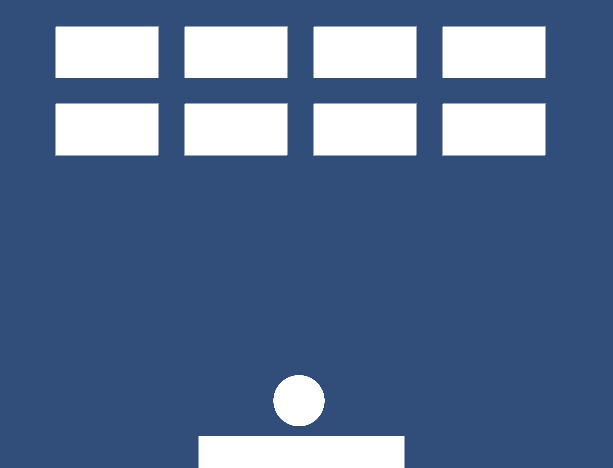


Рисунок 10 – «Pong-pong»

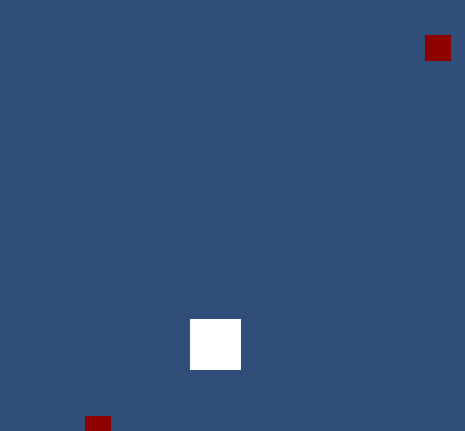
Рисунок 11 – «Lunch Take»



Рисунок 12 – «Increasing Worm»

Вторым шагом создаём панель при проигрыше, таймер, очки и рекорд.

Третьим шагом создадим уровни сложности для змейки, пин-понга, сбор фруктов.

Данные 3 режима игры ещё на стадии разработки и могут отличатся от существующих прототипах.

**5.2.5 Создание билдера игрового приложения**

Для того что бы создать билдер необходимо зайти: File > Build Settings… и выбрать платформу PC, потом добавить название приложения и его иконку. Далее нажимаем на кнопку Build и выбираем место куда будет создан ваш установщик.

**5.3 Руководство системного программиста**

Для того, чтобы установить программу необходимо запустить файл Setup.exe. Появится окно установки приложения. Затем достаточно следовать приведенной инструкции установки приложения.

**5.4 Руководство пользователя**

**5.4.1 Запуск программы**

Для запуска программы необходимо запустить APK.file «Pocket Arcadas».

**5.4.2 Инструкция по работе с программой**

Для запуска 3 режимов игры, надо нажать на их кнопки в меню с играми. Запустить игру.

Новые игры разблокируются путём приобретения их в магазине за игровою валюту.

При запуске «Increasing Worm» мы можем управлять червячком, нажимая на кнопки up, left, right, down; съедать фрукты, набирать рекорд. При смерти открывается панель настроек, где находятся две кнопки: «Restart», «Exit».

В меню игры находится панель с рекордом, панель с уровнями сложности и две кнопки: «Start», «Exit».

При запуске «Pong-pong» мы можем управлять ракеткой, нажимая на кнопки left и right, отбивать мячик и пропускать мячик теряя жизнь, набирать рекорд. При смерти открывается панель настроек, где находятся две кнопки: «Restart», «Exit».

В меню игры находится панель с рекордом, панель с уровнями сложности и две кнопки: «Start», «Exit».

При запуске «Lunch Take» мы можем управлять игроком, нажимая на кнопки left и right, собирая фрукты и другие предметы теряя жизнь, набирать рекорд. При смерти открывается панель настроек, где находятся две кнопки: «Restart», «Exit».

В меню игры находится панель с рекордом, панель с уровнями сложности и две кнопки: «Start», «Exit».

**5.4.3 Завершение работы с программой**

Для завершения работы с приложением, надо нажать на кнопку «Exit» в главном меню или выйти на рабочий стол через кнопку телефона.

**5.4.4 Использование системы справочной информации**

Для использования системы справочной информации необходимо нажать на кнопку «Info» в главном меню. Перед началом эксплуатации игрового приложения стоит прочитать и просмотреть справочную информацию.