МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01 «Программное обеспечение

информационных технологий»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: «Конструирование программ и языки программирования»

**на тему: Развлекательная игра «Minecraft Arkanoid»**

Пояснительная записка

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Разработал |  |  |  |  | Хололеенко Я. А. |
|  | (дата) |  | (подпись) |  | (ФИО) |
| Руководитель |  |  |  |  | Пилецкая С. А. |
|  | (дата) |  | (подпись) |  | (ФИО) |

Гомель, 2023

СОДЕРЖАНИЕы

[Введение 4](#_Toc138413162)

[1. Описание методов и средств разработки приложения 5](#_Toc138413163)

[1.1 Описание работы приложения 5](#_Toc138413164)

[1.2 Обоснование инструментария разработки приложения 5](#_Toc138413165)

[2 Проектирование программного продукта 7](#_Toc138413166)

[2.1 Функциональная модель 7](#_Toc138413167)

[2.2 Алгоритм работы 7](#_Toc138413168)

[3 Программная реализация продукта 7](#_Toc138413169)

[3.1 Разработка графического интерфейса 7](#_Toc138413170)

[3.2 Компоненты приложения 10](#_Toc138413171)

[3.3 Описание программной реализации функций и методов 10](#_Toc138413172)

[Заключение 10](#_Toc138413173)

[Список использованных источников 11](#_Toc138413174)

[Приложение А 12](#_Toc138413175)

[Приложение Б 13](#_Toc138413176)

[Приложение В 15](#_Toc138413177)

ВВЕДЕНИЕ

Аркадная игра[3] ⎯ игра с нарочно примитивным игровым процессом. Компьютерная игра называется «аркадной» в том случае, если она напрямую портатирована с автомата или же схожа по концепции с играми для автоматов. Аркадные игры подрузамевают простое, интуитивно-понятное управление игровыми обьектами. Пользователю нет необходимости настраивать и изучать управление игры, что позволяет практически сразу приступить к игровому процессу.

Целью данной работы является разработка приложения по мотивам игры «Arcanoid» в графическом оформлении игры «Minecraft».

«Arcanoid»[10] — видеоигра для игровых автоматов, разработанная компанией Taito в 1986 году. Игра основана на играх серии «Breakout» фирмы «Atari». Именно её название стало нарицательным для класса подобных игр.

Для разработки приложения «Minecraft Arcanoid» выбрана среда разработки «Unity».

Unity[7] — кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией «Unity Technologies». Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.

Задачами курсового проекта являются:

* исследование теоретической части проекта;
* разработка и проектирование алгоритмов решения поставленных задач;
* разработка ПО согласно заданию;
* отладка разработанного ПО;
* оформление материала пояснительной записки и графической части.

Программа создается с целью развлечения пользователей.

Для выполнения данного курсового проекта необходимо разработать алгоритм решения поставленного задания, правильно указав последовательное выполнение соответствующих команд для получения необходимых результатов.

Разрабатываемая игра должна содержать следующее:

* выбор языка игры (русский или английский);
* множество уровней в игре;
* сохранение игрового процесса;
* физика игровых компонентов;
* управление игроком платформы.

1. ОПИСАНИЕ МЕТОДОВ И СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

* 1. **Описание работы приложения**

Приложение «Minecraft Arkanoid» — однопользовательская мобильная игра. Приложение разрабатывается для мобильных устройств с операционной системой Android и IOS.

Android[9] — популярная операционная система, построенная на ядре Linux. Используется в смартфонах, планшетах, электронных книгах, цифровых проигрывателях, наручных часах, фитнес-браслетах, игровых приставках, ноутбуках, нетбуках, телевизорах и других устройствах.

IOS[8] — мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей, разрабатываемая и выпускаемая американской компанией Apple.

Однопользовательская игра или одиночная игра — режим игры, во время которого с ней через устройства ввода-вывода взаимодействует один человек.

Суть и геймплей игры в большей мере позаимствованы из игры «Arkanoid» 1986 года, поэтому ее можно назвать ремейком.

Ремейк компьютерной игры — это отдельная компьютерная игра, созданная на основе ранее созданной игры. Ремейк может как создаваться с нуля, так и использовать наработки оригинала, но никогда не копирует оригинал полностью. Обычно ремейк имеет практически то же название, сюжет и геймплей, что и оригинальная игра, но при этом часто содержит более современную графику и звуковые эффекты, изменённую внешность персонажей, интерфейс пользователя, дизайн и строение локаций. Сюжет и геймплей в определённых случаях также могут быть изменены и адаптированы согласно современным трендам и стандартам[1].

Геймплей игры прост и интуитивно понятен для пользователей, что позволяет начинать играть без обучения.

Игрок контролирует небольшую платформу-кровать, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз.

Удар шарика по блоку приводит к разрушению блока. После того как все блоки на данном уровне уничтожены, происходит переход на следующий уровень, с новым набором блоков. Есть и некоторое разнообразие: определённые блоки нужно ударять несколько раз, удар по некоторым блокам приводит к выпадению из них бонусов который активируется если поймать бонус кроватью.

Игрок может выбирать открытые наборы. Открыть наборы уровней можно пройдя предыдущий. Каждый набор уровней имеет определенную стилизацию: иконку, название и внешнее оформление и имеет свои уникальные уровни.

Также игроку доступно выбирать языковую адаптацию: английский или русский язык.

* 1. **Обоснование инструментария разработки приложения**

В качестве среды разработки приложения «Minecraft Arkanoid» была выбрана интегрированная среда разработки Unity. Игра использует версию Unity 2020.42f.

Unity — кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие.

Возможности Unity:

* создание игровых объектов;
* редактирование сцен;
* управление компонентами и системой скриптов;
* отладка и другие инструменты;
* работа с анимацией и графикой;
* поддержка многопользовательских игр;
* использование расширений.

Для создания игры был выбран язык программирования C#.

«C#»[2] — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании «Microsoft» под руководством Андерса Хейлсберга и Скотта Вильтаумота, как язык разработки приложений для платформы «Microsoft .NET Framework» и «.NET Core». Впоследствии был стандартизирован, как «ECMA-334» и «ISO/IEC» 23270.

Объектно-ориентированный язык программирования[4] — язык, построенный на принципах объектно-ориентированного программирования. В основе концепции объектно-ориентированного программирования лежит понятие объекта — некой сущности, которая объединяет в себе поля и методы.

«C#» относится к семье языков с C-подобным синтаксисом[5], синтаксис наиболее близок к «C++» и «Java». Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, «LINQ», исключения, комментарии в формате «XML».

Язык программирования C# считается одним из самых универсальных. Он применяется в самых разных сферах. Например, для создания продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android.

Основные преимущества языка:

* «С#» популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции;
* независимость от аппаратного функционала. Программу не нужно адаптировать под многочисленные платформы и операционные системы. Виртуальная машина .NET Framework[6] сама выполняет эту задачу;
* управление памятью. Разработчику не нужно контролировать расход памяти, устранять её утечки или удалять «мёртвые» куски кода так как это выполняется автоматически;
* строгая типизация данных;
* сохранение концепций ООП;
* функциональность;
* популярность языка — еще одно значимое преимущество. Большое количество поклонников C# способствуют его развитию. Также это благоприятно влияет на рост числа вакансий, связанных с разработкой на языке «Microsoft». Программисты, хорошо знакомые с «С#», востребованы в индустрии, несмотря на их большое и постоянно увеличивающееся количество;
* понятный синтаксис «C#» заметно упрощает не только разработку как таковую, но и другие важные аспекты совместной работы, например, чтение чужого кода. Это упрощает процесс рефакторинга и исправления ошибок при работе над приложениями в больших командах.

Данный язык получил широкое распространение среди разработчиков видеоигр. С# применяется для создания игр под Windows, macOS, Android и iOS. Объясняется это тем, что этот язык лучше всего подходит для работы с Unity. Именно по этой причине разработчики столь часто применяют комбинацию из Unity и C#.

Ниже приведены основные преимущества использования Unity и языка C# для разработки мобильных приложений:

* кроссплатформенность. Unity позволяет создавать приложения для множества платформ, таких как iOS, Android, Windows, macOS, Linux и др. Это означает, что вы можете создавать приложения для разных устройств и операционных систем, используя один и тот же код;
* мощный движок. Unity имеет мощный движок, который позволяет создавать высококачественные 2D и 3D приложения. Это включает в себя поддержку физики, графики, звука и других функций, которые позволяют создавать сложные приложения;
* язык программирования C#. C# современный и мощный язык программирования, который используется в Unity для разработки приложений. C# позволяет создавать высокопроизводительный код, который может быть легко поддерживаться и расширяться;
* большое сообщество разработчиков. Unity имеет огромное сообщество разработчиков, которое активно обменивается информацией и опытом. Это означает, что вы можете быстро найти ответы на свои вопросы и получить помощь от других разработчиков;
* легкость использования. Unity имеет простой и интуитивно понятный интерфейс, который позволяет быстро создавать приложения без необходимости знания сложных технологий и языков программирования;
* быстрое развертывание. Unity позволяет быстро развернуть приложение на разных платформах, что позволяет быстро запустить приложение на рынке;
* низкие затраты. Unity имеет бесплатную версию, которая позволяет создавать приложения без необходимости покупки дорогостоящей лицензии.

В целом, Unity и язык C# являются мощными инструментами для разработки мобильных приложений. Они позволяют быстро создавать приложения для разных устройств и операционных систем, а также обеспечивают высокую производительность и гибкость приложений, что является причиной выбора такой комбинации для создания «Minecraft Arkanoid».

**2** **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**

**2.1 Функциональная модель**

Методологию «IDEF0» можно считать следующим этапом развития графического языка описания функциональных систем «SADT» («Structured Analysis and Design Technique»).

Целью методологии является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы. Другими словами, в «IDEF0» моделируемая система представляется как совокупность взаимосвязанных работ (функций, активностей).

Основными элементами диаграммы в нотации IDEF0 являются:

* блоки, в виде которых представлены процессы, функции, операции, действия (в зависимости от степени детализации);
* стрелки, в виде которых на диаграмме отражают информационные и материальные ресурсы, связанные с функциями.

Контекстная диаграмма отражает интерфейс системы с внешним миром, а именно, информационные потоки между системой и внешними сущностями, с которыми она должна быть связана.

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции.

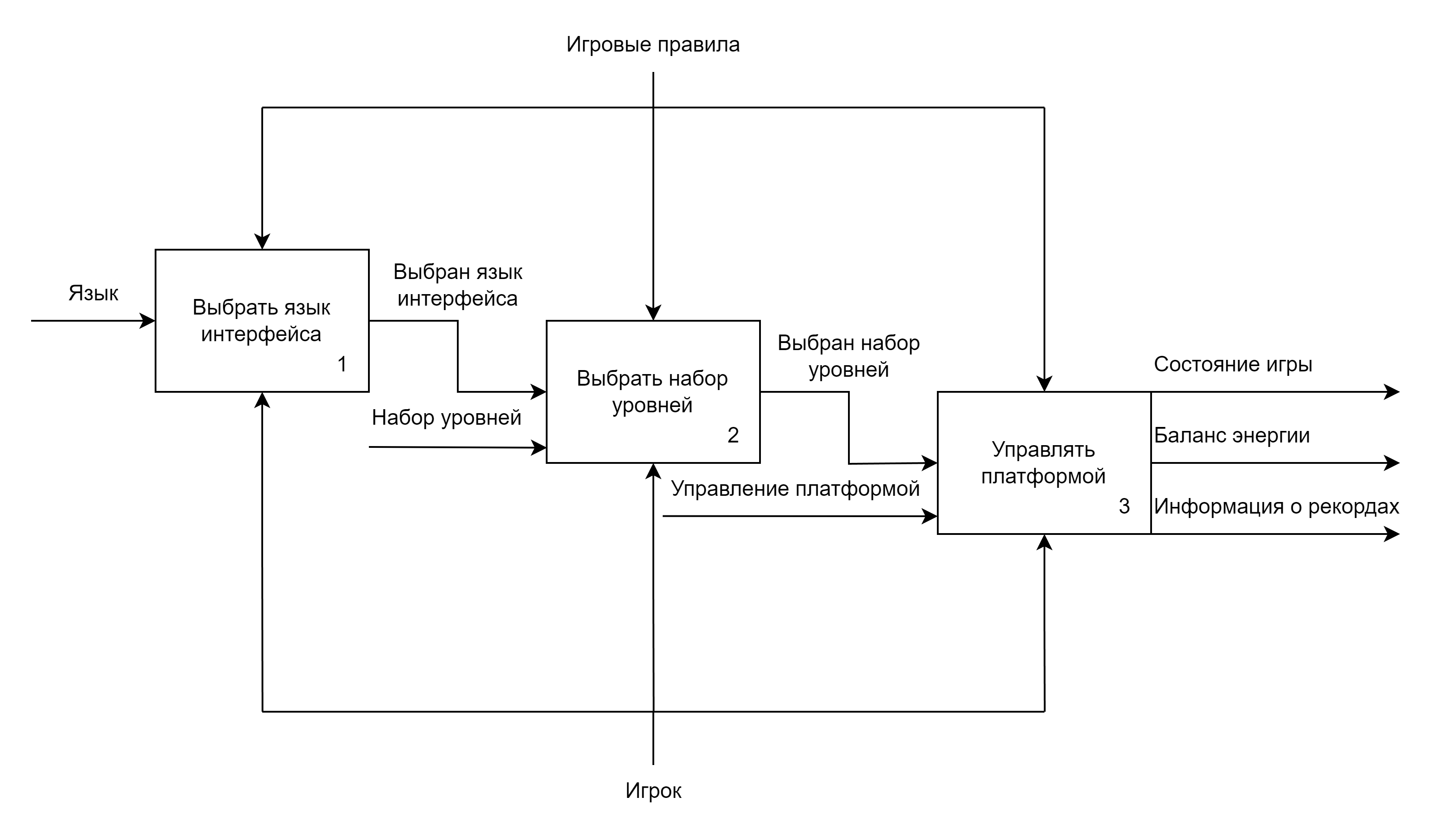
Контекстная диаграмма системы предоставлена на рисунке 2.1.



Рисунок 2.1 — Контекстная диаграмма системы

В контекстной диаграмме указаны такие входные данные как управление платформой, набор уровней, язык. Выходными данными являются состояние игры, баланс энергии и информация о рекордах. Управлением являются игровые правила, а механизмом — игрок.

Декомпозиция общей функциональной схемы на отдельные функции представлена на рисунке 2.2.







.

Рисунок 2.2 — Декомпозиция общей функциональной схемы

Первый блок «Выбрать язык интерфейса» диаграммы декомпозиции принимает входные данные о выбранном пользователем языке интерфейса. Выходными данными блока является выбранный язык интерфейса.

Второй блок «Выбрать набор уровней» принимает входные данные о выбранном пользователем наборе данных. Выходными данными блока является выбранный набор уровней.

Третий блок «Управлять платформой» принимает в качестве входных данных управление платформой. Выходными данными блока является состояние игры, баланс энергии и информация о рекордах.

Механизмом для всех блоков являются игровые правила.

Управлением для всех блоков является игрок.

Диаграмма прецедентов или диаграмма вариантов использования «Use Case» — диаграмма, отражающая отношения между участниками и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Компоненты «Use Case» диаграммы:

* участник — это любой объект или лицо, которое участвует в выполнении какого-то действия в системе. Участник не обязательно должен быть человеком, это может быть другая система или компьютер;
* вариант — это альтернативный сценарий, который может произойти в процессе выполнения «Use Case». Варианты могут быть полезны при моделировании ситуаций, когда происходят ошибки, система не работает, или когда пользователь делает что-то неправильно. Варианты могут быть связаны с основным потоком событий, а также друг с другом;
* направленная ассоциация — это связь между участником и «Use Case», которая указывает на направление взаимодействия между ними.

«Use Case» диаграмма приложения представлена на рисунке 2.3.





Рисунок 2.3 — «Use Case» диаграмма приложения

В созданной диаграмме «UseCase» участником является игрок. Существуют следующие варианты использования:

* войти в игру;
* выйти из игры;
* выбрать язык, что включает в себя выбор русского либо английского языка;
* выбрать набор уровней;
* управлять платформой;
* выйти в меню уровня что включает в себя такие варианты использования как: выйти в главное меню, начать уровень заново, купить второй шанс за внутриигровую валюту.

В ходе этого этапа было спроектировано приложение.

**2.2 Алгоритм работы**

Блок-схема — это схематичное представление процесса, системы или компьютерного алгоритма. Блок-схемы часто применяются в разных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых логичных диаграмм. Для построения блок-схем применяются прямоугольники, овалы, ромбы и некоторые другие фигуры (для обозначения конкретных операций), а также соединительные стрелки, которые указывают последовательность шагов или направление процесса.

Блок-схема приложения представлена на рисунке 2.4.

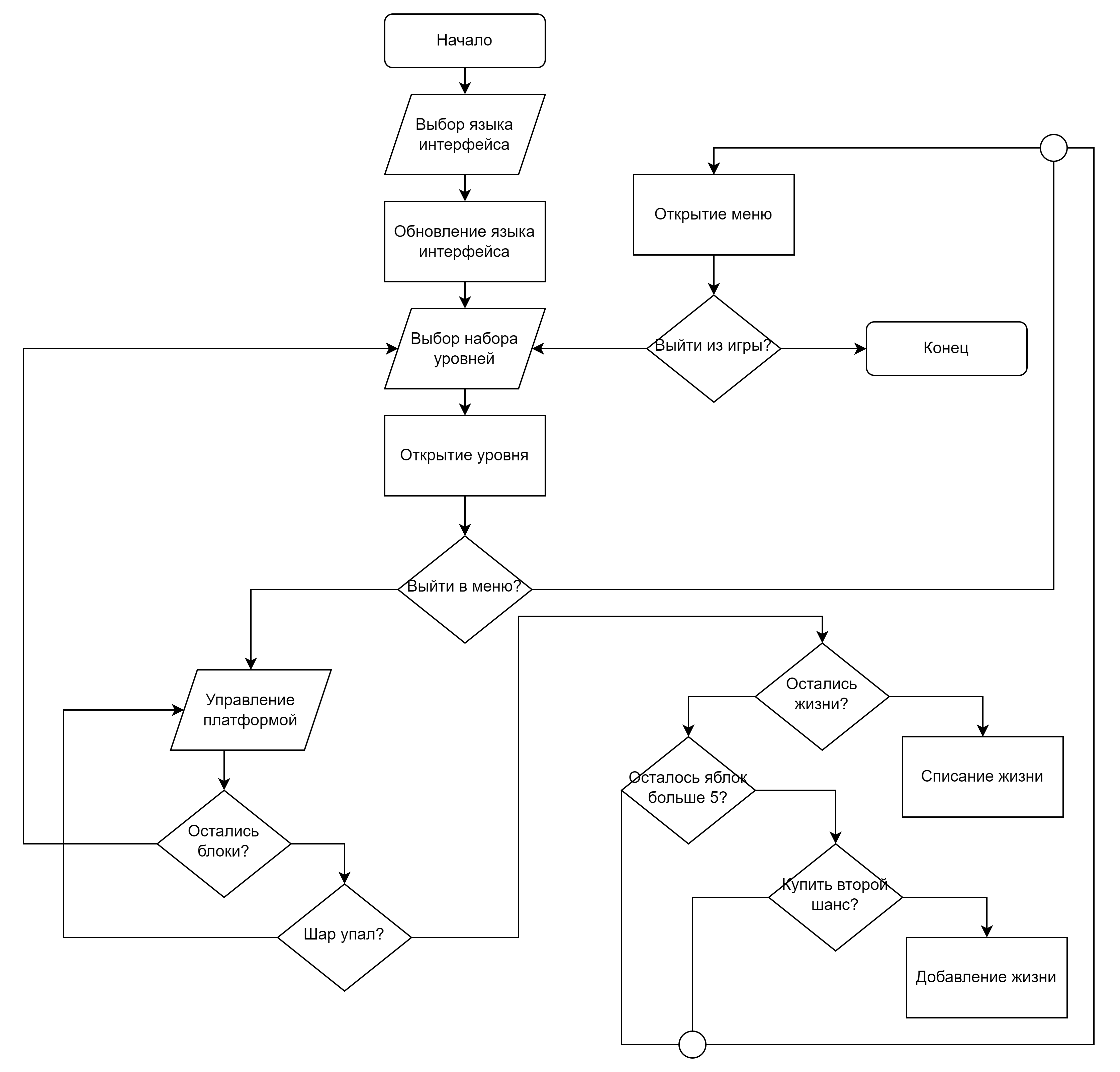


Рисунок 2.4 — Блок-схема приложения

Сперва пользователь выбирает язык интерфейса, после чего он загружается в приложении и открывается меню выбора уровней. В меню выбора уровней пользователь выбирает уровень. После открытия уровня пользователь может войти в меню либо продолжить играть.

Если пользователь играет, то приложение обрабатывает управление платформой пользователем. В случае если блоков на карте не осталось уровень завершается и пользователю следует выбрать набор уровней. Если блоки остались при падении шара обрабатывается количество оставшихся жизней. Если жизни остались происходит их списание, если не осталось и количество яблок больше 5, то пользователь может купить второй шанс для добавления жизней, в противном случае игра засчитывает проигрыш и переходит в меню. В меню можно выйти из игры, либо выбрать набор уровней.

В ходе этого этапа был разработан алгоритм работы приложения.

# **3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ**

**3.1 Разработка графического интерфейса**

Графический интерфейс пользователя — разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки), представленные пользователю на дисплее, исполнены в виде графических изображений.

При разработке интерфейса необходимо учесть ряд особенностей:

* интерфейс должен быть интуитивно понятен простому пользователю;
* интерфейс разрабатывается в едином стиле;
* интерфейс должен быть удобен.

На рисунке 3.1 представлена схема окна загрузки приложения.

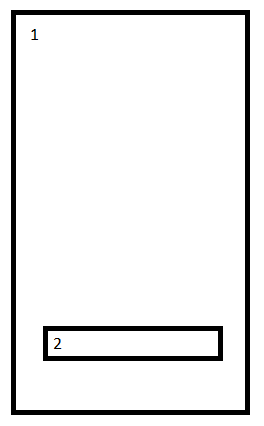


Рисунок 3.1 — Схема окна загрузки: 1 — окно загрузки; 2 — индикатор загрузки

На рисунке 3.2 представлен интерфейс окна загрузки приложения.



Рисунок 3.2 — Интерфейс окна загрузки

При запуске приложения и открытии уровней открывается форма загрузки необходимая для загрузки элементов. Прогресс загрузки элементов отображается в индикаторе загрузки.

На рисунке 3.3 представлена схема главного меню приложения.

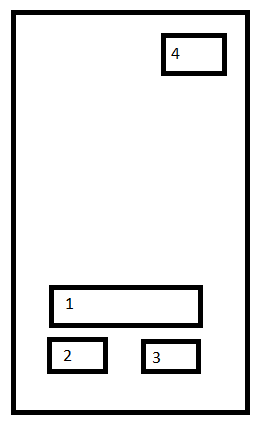


Рисунок 3.3 — Схема главного меню: 1 — кнопка перехода в меню выбора уровней; 2 — кнопка смены языка на английский; 3 — кнопка смены языка на русский; 4 — индикатор энергии

На рисунке 3.4 представлен интерфейс главного меню.



Рисунок 3.4 — Интерфейс главного меню

В главном меню пользователь может выбрать язык, посмотреть оставшуюся энергию и начать играть.

На рисунке 3.5 представлена схема меню выбора уровней.

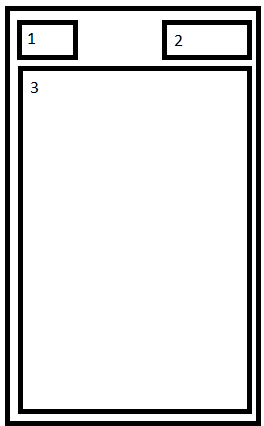


Рисунок 3.5 — Схема меню выбора уровней: 1 — кнопка для выхода в главное меню; 2 — индикатор энергии; 3 —список наборов уровней;

На рисунке 3.6 представлен интерфейс меню выбора уровней.



Рисунок 3.6 — Интерфейс меню выбора уровней

В меню выбора уровней пользователь имеет возможность вернуться в главное меню, посмотреть оставшуюся энергию и запустить уровень.

На рисунке 3.7 представлена схема игрового уровня.

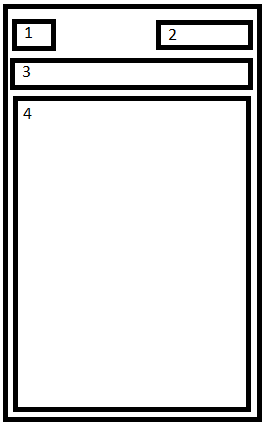


Рисунок 3.7 — Схема игрового уровня: 1 — кнопка выхода в меню уровня; 2 — индикаторы попыток; 3 — индикатор прогресса прохождения уровня; 4 — карта игрового уровня

На рисунке 3.8 представлен интерфейс игрового уровня.

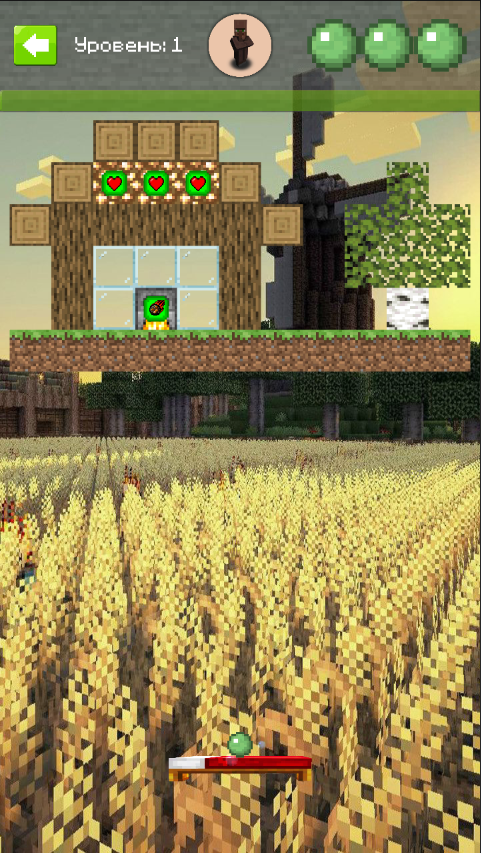


Рисунок 3.8 — Интерфейс игрового уровня

В игровом уровне пользователь имеет возможность перейти в меню уровня, просмотреть прогресс прохождения уровня, количество оставшихся попыток и проходить уровень.

На рисунке 3.9 представлена схема меню уровня.

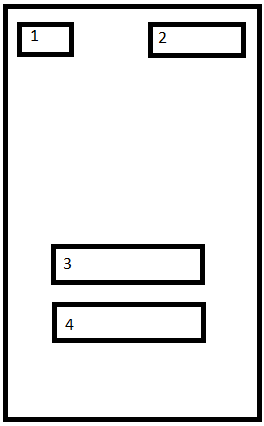


Рисунок 3.9 — Схема меню уровня: 1 — кнопка для выхода в меню выбора уровней; 2 — индикатор энергии; 3 — кнопка для покупки второго шанса игры; 4 — кнопка для прохождения уровня заново

На рисунке 3.10 представлен интерфейс меню уровня.



Рисунок 3.10 — Интерфейс меню уровня

В меню уровня игра ставиться на паузу и игроку предоставляется возможность начать его заново, продолжить игру, просмотреть оставшуюся энергию или вернуться в меню выбора уровней.

Для создания графического интерфейса игрового приложения в «Unity» были использованы различные UI-элементы(элементы графического интерфейса), такие как:

* «Canvas» — предназначен для объединения множества элементов интерфейса в один контейнер и последующего вывода их на экран;
* «Text» — предназначен для вывода информации на экран;
* «Slider» — предназначен для выбора числового значения на интервале;
* «Button» — предназначен для вызова события по нажатию;
* «Panel» — предназначен для группировки объектов;
* «Image» — предназначен для вывода графических элементов на экран;
* «Scroll View» — предназначен для прокрутки содержимого, которое не помещается на экране.

В ходе этого этапа был спроектирован и разработан графический интерфейс приложения.

**3.2 Компоненты приложения**

В соответствии с разработанным графическим интерфейсом для его программной реализации необходимо использование компонентов, которые приведены в таблице 3.1.

Таблица 3.1 — Использованные компоненты и их назначение

|  |  |
| --- | --- |
| Наименование компонента | Назначение |
| 1 | 2 |
| Transform | Диктует, где расположен игровой объект, как он поворачивается и масштабируется |
| Camera | Камеры являются устройствами, которые захватывают и отображают мир игроку |
| Box Collider 2d | Прямоугольный коллайдер, используемый при работе с 2D физикой |
| Rigidbody 2d | Подключает физическое поведение для объекта. С прикреплённым Rigidbody, объект немедленно начнёт реагировать на гравитацию |
| Sprite Rendered | Позволяет отображать изображения в виде спрайтов(Sprites), чтобы использовать их в сценах |
| Polygon Collider | Коллайдер произвольной настраиваемой формы, используемый при работе с физикой |
| Canvas | Область, внутри которой находятся все элементы UI |
| Rect Transform | Позволяет задать положение и размер игрового объекта, используя удобные визуальные контролы |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| Image | Позволяет задавать и настраивать изображение |
| Aspect Ratio Fitter | Контролирует размер своего компоновочного элемента |
| Canvas Scaler | Масштабирует компонент Canvas |
| Graphic Raycaster | Позволяет игроку взаимодействовать с кнопками и другими взаимодействующими элементами внутри холста UI |
| Horizontal Layout Group | Горизонтальная сортировка UI-элементов в группе |
| Text Mesh Pro | Позволяет редактировать текст |
| Event System | Включает компоненты визуального управления вызовом событий |
| Standalone Input Module | Позволяет управлять системой Input Manager. |

После создания графического интерфейса осуществляется переход в фазу программной реализации приложения.

## **3.3 Описание программной реализации функций и методов**

При открытии приложения идет загрузка игры, за это отвечает код, представленный на рисунке 3.10.

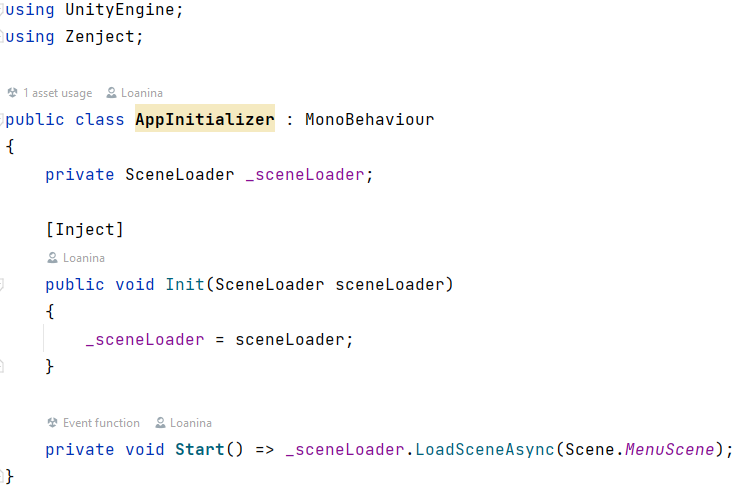


Рисунок 3.10 — Код загрузки сцены

После загрузки открывается сцена главного меню, в котором пользователь имеет возможность выбрать язык для приложения: английский или русский. Для реализации такой функции созданы словари английского и русского языка, показанные на рисунке 3.11. Перевод на языки осуществляется кодом, представленным на рисунке 3.12.

Рисунок 3.11 — Английский и русский словари

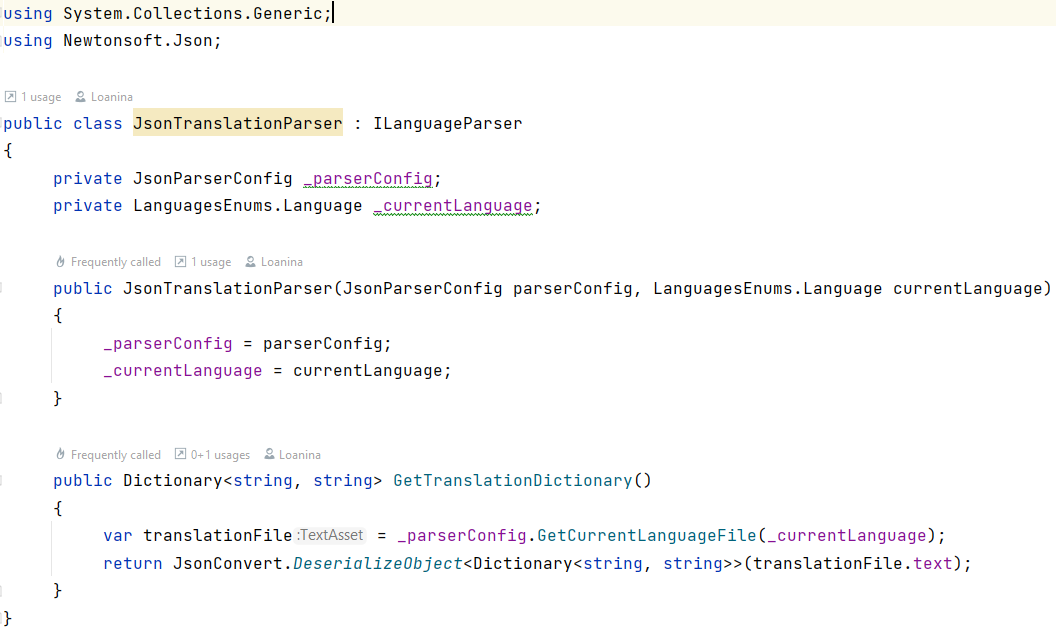


Рисунок 3.12 — Код осуществляющий перевод текста

Чтобы ограничить попытки игр пользователя создана система энергии. За энергию можно начать проходить уровень или покупать дополнительные разы игры. Энергия постоянно восстанавливается с течением времени. Реализация системы энергии показана на рисунке 3.13.

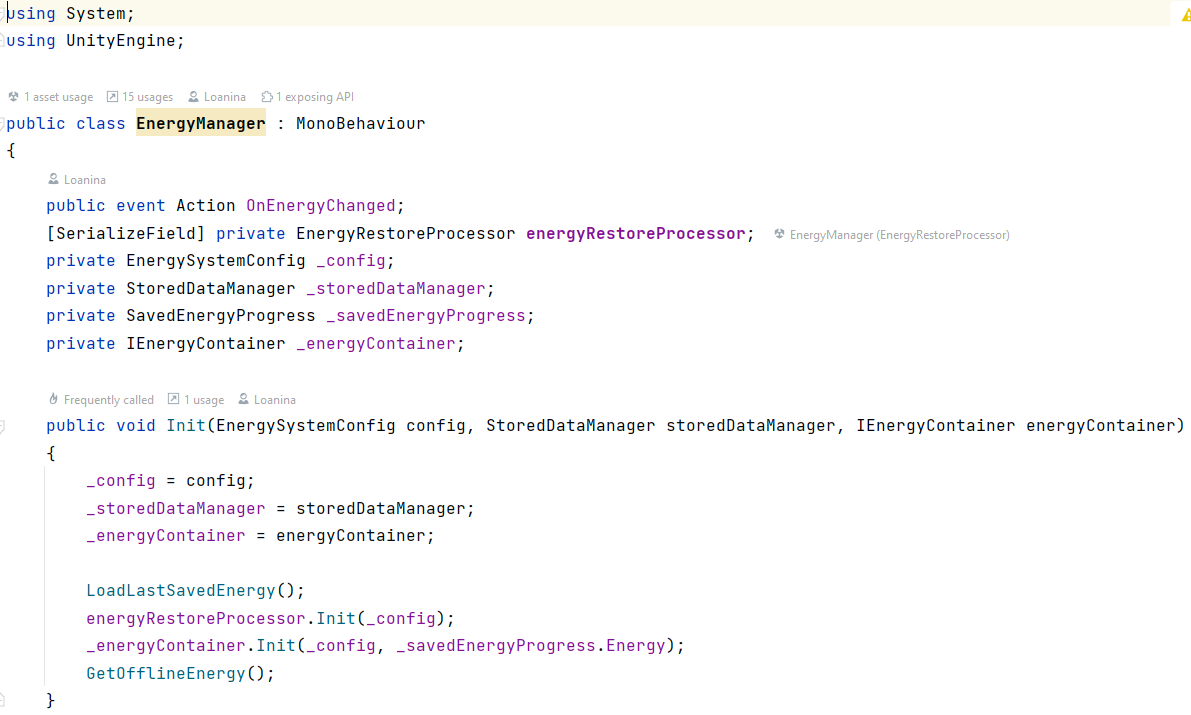


Рисунок 3.13 — Код системы энергии

В игре предусмотрены закрытые наборы уровни, которые открываются лишь после прохождения предыдущего набора уровней. Такие уровни пользователь открыть не может и изначально ему открыт лишь самый первый набор уровней. Код обработки набора уровней показан на рисунке 3.14

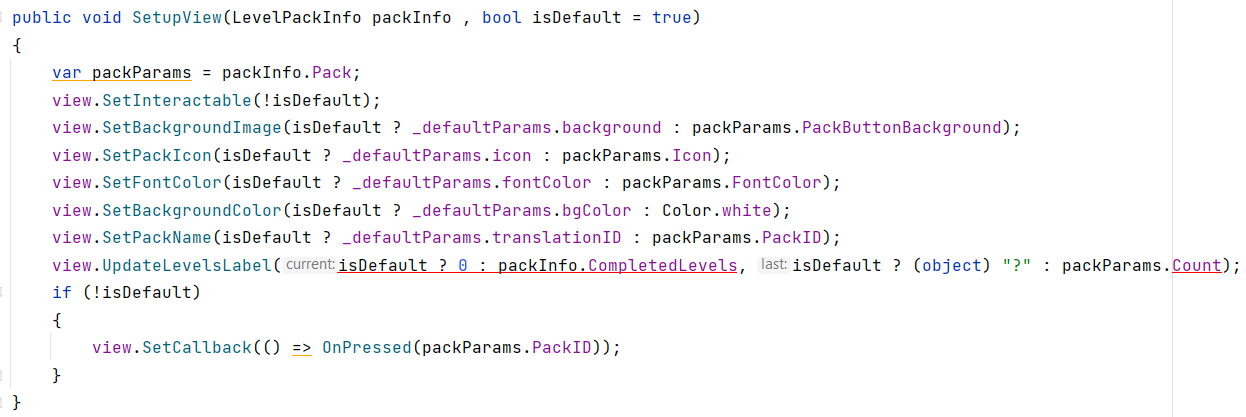


Рисунок 3.14 — Код обработки набора уровней

В игре пользователь имеет возможность перемещать платформу, которой можно манипулировать движениями шара. Код, обработки перемещения платформы показан на рисунке 3.15.



Рисунок 3.15 — Код обработки перемещения платформы

Шар перемещается без остановки, отталкиваясь от всех объектов и уничтожаясь после падения его ниже платформы. Код обработки перемещения шара представлен на рисунке 3.16. При отталкивании от объектов шар меняет свое направление, что реализовано с помощью математический функций в коде, показанных на рисунке 3.17.

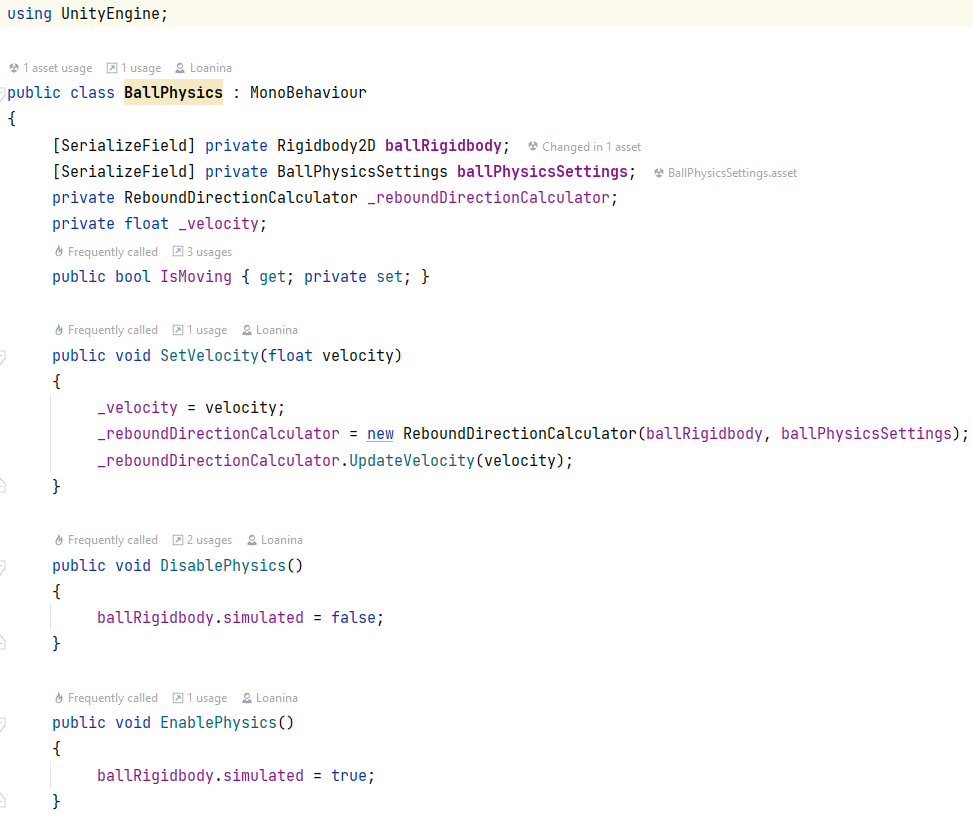


Рисунок 3.16 — Код обработки перемещения шара

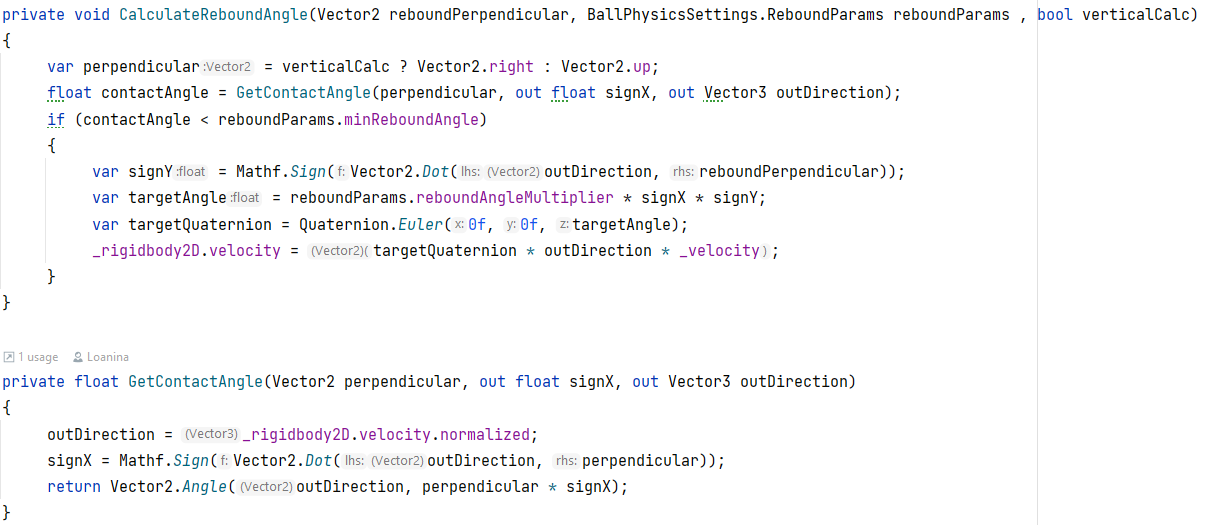


Рисунок 3.17 — Код вычисления углов при отталкивании

При сталкивании шара с блоком он повреждается или уничтожается, в зависимости от его параметров. Структура блоков показана на рисунке 3.18. Код обработки блоков показан на рисунке 3.19.

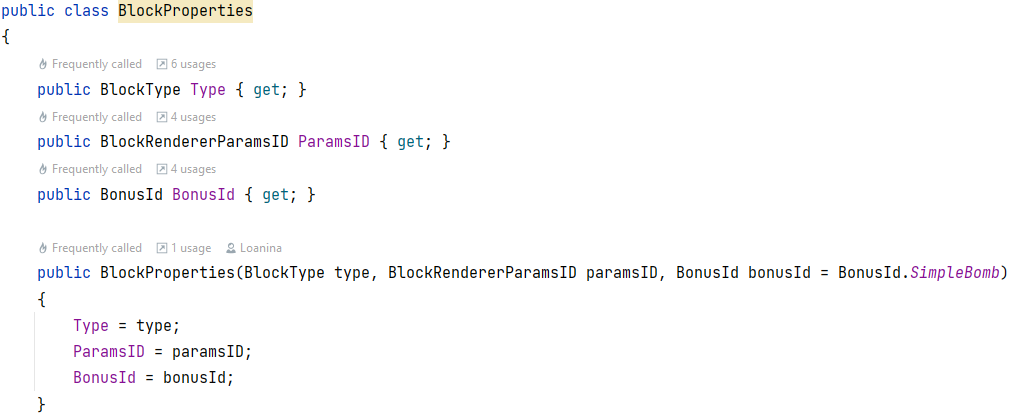


Рисунок 3.18 — Структура блоков

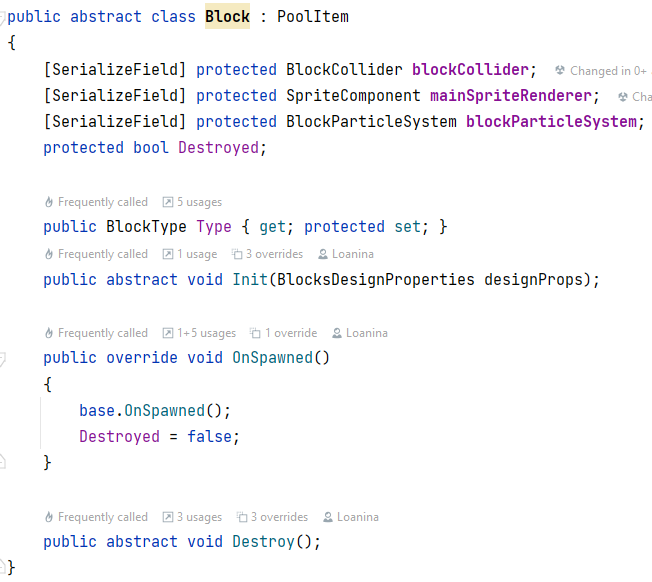


Рисунок 3.19 — Код обработки блоков

При каждом разбитом блоке заполняется индикатор прогресса прохождения уровней. Код обработки индикатора представлен на рисунке 3.20.

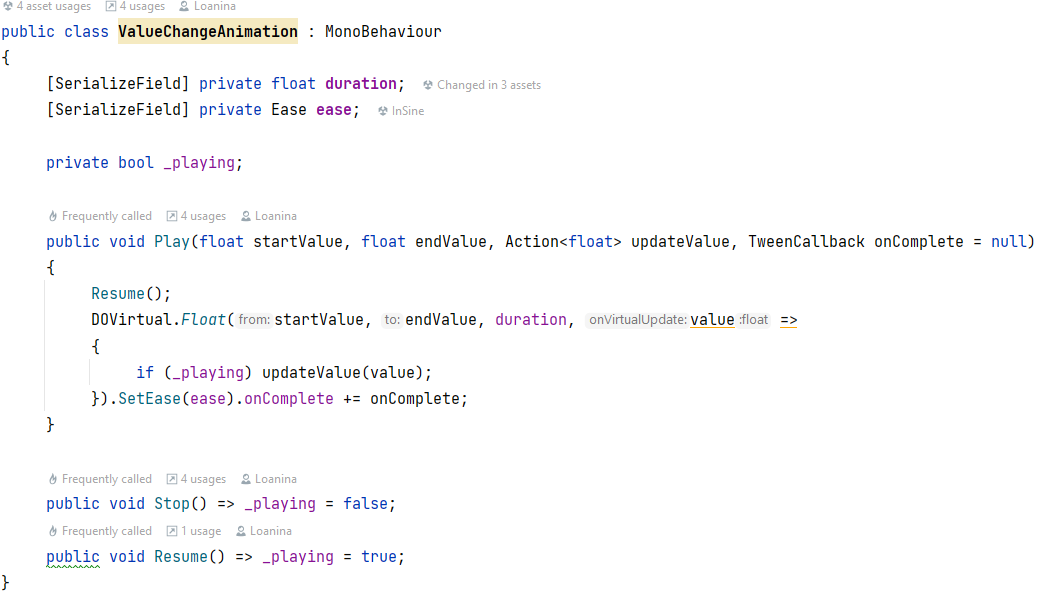


Рисунок 3.20 — Код обработки индикатора прогресса

Для сохранения прогресса игры данные приложения записываются в бинарный файл. Структура сохраняемого прогресса показана на рисунке 3.21. Код, осуществляющий обработку сохраняемого прогресса показан на рисунке 3.22.

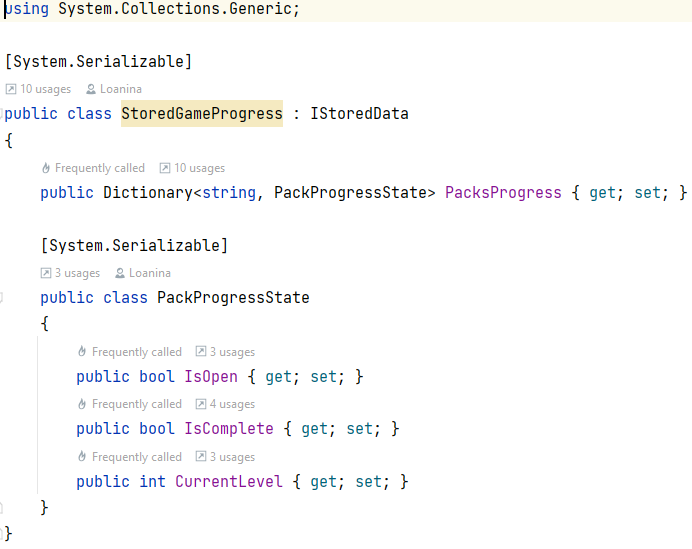


Рисунок 3.21 — Структура сохраняемого прогресса



Рисунок 3.22 — Код сохранения данных

В ходе этого этапа была продемонстрирована программная реализация игры «Minecraft Arkanoid».

## **ЗАКЛЮЧЕНИЕ**

В результате курсового проекта была разработана развлекательная игра «Minecraft Arkanoid» на языке «C#». Созданная программу можно запустить на устройстве с операционными системами Android, IOS или непосредственно в Unity.

В процессе её разработки была спроектирована и реализована система, соответствующая заявленным правилам, составленным к использованию.

Благодаря использованию выбранного языка программирования, а также среды разработки, значительно упростился процесс разработки, из-за наличия стандартных библиотек, инструментов и объектов класса.

Приложение является простым для понимания и не требует специальных знаний для её использования.

В процессе разработки приложения, была изучена предметная область. Также были исследованы используемые компоненты их свойства и принципы функционирования. Была разработана общая функциональная схема системы с её последующими декомпозициями, схема вариантов использования и схема алгоритма работы программного средства. Результатом выполнения данных действий является отлаженная и готовая к эксплуатации игра, сочетающая в себе все планируемые функции и возможности.

Разработанная программа позволила усовершенствовать навыки разработки ПО и углубить свои знания в этой области.

## **Список использованных источников**

1. Васильев, А.Ф. C#. Объектно-ориентированное программирование / А.Васильев. — М.: Питер, 2012. — 320 c.
2. Прайс. «C#» 7 и «.Net Core». Кросс-платфоменная разработка для профессионалов / Дж.Марк. СПб:Прайс, 3-е издание, 2017. — 146 с.
3. Walker M. Unity in Action: Multiplatform Game Development in C# / M. Walker. — Manning Publications, 2018. — 96 с.
4. Павловская, Т.А. Языки программирования C#: учебник / Т. А. Павловская, А. В. Отвагин. — СПб:Прайс, 2-е изд., перераб, и доп.: 2017. — 464 с.
5. Фаронов, В.В. Создание приложений с помощью C#: Руководство программиста / В.В. Фаронов. — М.: Эксмо, 2008. — 576 с.
6. Мюллер Д. П. C# для чайников / Д. П.Мюллер — СПб: Питер, 2012 — 519 с.
7. Harrison H. Unity 2018 Game Development in 24 Hours, Sams Teach Yourself / H. Harrison. — Sams Publishing, 2018. — 108 с.
8. Dic.academic.ru [Электронный ресурс] / «Графический интерфейс пользователя это…» — Режим доступа: http://dic.academic.ru/ dic.nsf/ruwiki/878908 — Дата доступа: 23.06.2023.
9. Professorweb.ru [Электронный ресурс] / EF — Режим доступа: <https://professorweb.ru> — Дата доступа: 01.05.2023.
10. Okita H. Learning Unity Android Game Development / H. Okita. — Packt Publishing, 2015 — 25 с.

## **ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

using System;  
using DG.Tweening;  
using UnityEngine;  
  
public class **PlatformMovement** : MonoBehaviour, IPointerPositionHandler  
{  
 [SerializeField] private Rigidbody2D **platformRigidbody**;  
 private BackToInitPositionSettings \_backToInitPositionSettings;  
 private Transform \_platformTransform;  
 private Vector3 \_startPosition;  
 private Vector2 \_targetPosition;  
 private Vector2 \_velocity;  
 private Vector2 \_direction;  
 private float \_platformHalfSize;  
 private float \_speed;  
 private float \_boundX;  
 private float \_gameBoundarySize;  
 private float \_targetPosAccuracy;  
 private double \_absDistanceToTarget;  
 private bool \_controlLock;  
 private bool \_isHolding;  
  
 public void Init(float targetPosAccuracy, float gameBoundarySize, BackToInitPositionSettings backToInitPositionSettings)  
 {  
 \_backToInitPositionSettings = backToInitPositionSettings;  
 \_platformTransform = transform;  
 \_startPosition = \_platformTransform.position;  
 \_targetPosition = \_startPosition;  
 \_targetPosAccuracy = targetPosAccuracy;  
 \_gameBoundarySize = gameBoundarySize;  
 \_boundX = \_gameBoundarySize / 2;  
 }  
   
 private void **OnEnable**() => MessageBus.*Subscribe*(this);  
 private void **OnDisable**() => MessageBus.*Unsubscribe*(this);  
  
 public void RefreshParameters()  
 {  
 transform.position = \_startPosition;  
 \_targetPosition = \_startPosition;  
 \_controlLock = false;  
 }  
   
 public void SetNewPhysicsSize(float size)  
 {  
 \_platformHalfSize = \_gameBoundarySize \* size / 2;  
 }  
   
 public void SetNewSpeed(float speed)  
 {  
 \_speed = speed;  
 }  
   
 public void OnUpdatePointerPosition(Vector3 position)  
 {  
 if (\_controlLock) return;  
  
 \_targetPosition = CalculateTargetMovePosition(position);  
 }  
  
 public void OnUpdateHoldingState(bool isHolding) => \_isHolding = isHolding;  
  
 private Vector3 CalculateTargetMovePosition(Vector3 pointerPos)  
 {  
 pointerPos.y = \_platformTransform.position.y;  
 float absPosX = Mathf.*Abs*(pointerPos.x) + \_platformHalfSize;  
  
 if (absPosX > \_boundX)  
 {  
 pointerPos.x = pointerPos.x > 0 ? \_boundX - \_platformHalfSize : \_platformHalfSize - \_boundX;  
 }  
 return pointerPos;  
 }  
  
 private void **FixedUpdate**()  
 {  
 if (\_controlLock) return;  
  
 if (!\_isHolding)  
 {  
 \_targetPosition = platformRigidbody.position;  
 }  
   
 \_absDistanceToTarget = Math.*Round*(Mathf.*Abs*(platformRigidbody.position.x - \_targetPosition.x), 2);  
  
 \_direction = \_targetPosition - platformRigidbody.position;  
 if (\_absDistanceToTarget > \_targetPosAccuracy)  
 {  
 \_velocity = \_direction.normalized \* \_speed;  
 }  
 else  
 {  
 \_velocity = \_direction.normalized \* \_speed \* (float)\_absDistanceToTarget;  
 }  
 platformRigidbody.velocity = \_velocity;  
 }  
   
 public void BackToInitialPosition(Action onComplete = null)  
 {  
 LockControl();  
 transform.DOMove(\_startPosition, \_backToInitPositionSettings.duration).SetEase(\_backToInitPositionSettings.ease).OnComplete(() =>  
 {  
 platformRigidbody.velocity = Vector2.zero;  
 onComplete?.Invoke();  
 });  
 }  
   
 public void LockControl() => \_controlLock = true;  
 public void UnlockControl() => \_controlLock = false;  
}

using System;  
using System.Collections.Generic;  
using System.Linq;  
  
public static class MessageBus  
{  
 private static readonly Dictionary<Type, SubscribersList<ISubscriber>> *Subscribers* = new Dictionary<Type, SubscribersList<ISubscriber>>();  
 private static readonly Dictionary<Type, List<Type>> *CachedSubscriberTypes* = new Dictionary<Type, List<Type>>();  
  
 public static void *Subscribe*(ISubscriber subscriber)  
 {  
 List<Type> subscriberTypes = *GetSubscriberTypes*(subscriber);  
 foreach (Type t in subscriberTypes)  
 {  
 if (!*Subscribers*.ContainsKey(t))  
 {  
 *Subscribers*[t] = new SubscribersList<ISubscriber>();  
 }  
 *Subscribers*[t].Add(subscriber);  
 }  
 }  
  
 public static void *Unsubscribe*(ISubscriber subscriber)  
 {  
 List<Type> subscriberTypes = *GetSubscriberTypes*(subscriber);  
 foreach (Type t in subscriberTypes)  
 {  
 if (*Subscribers*.ContainsKey(t))  
 *Subscribers*[t].Remove(subscriber);  
 }  
 }  
  
 public static void *RaiseEvent*<T>(Action<T> action) where T : class, ISubscriber  
 {  
 var type = typeof(T);  
   
 if (!*Subscribers*.ContainsKey(type)) return;  
   
 SubscribersList<ISubscriber> subscribersList = *Subscribers*[type];  
  
 subscribersList.IsExecuting = true;  
 foreach (ISubscriber subscriber in subscribersList.List)  
 {  
 action.Invoke(subscriber as T);  
 }  
 subscribersList.IsExecuting = false;  
 subscribersList.ClearNullSubs();  
 }  
   
 private static List<Type> *GetSubscriberTypes*(ISubscriber globalSubscriber)  
 {  
 Type type = globalSubscriber.GetType();  
 if (*CachedSubscriberTypes*.ContainsKey(type))  
 {  
 return *CachedSubscriberTypes*[type];  
 }  
 List<Type> subscriberTypes = *GetListOfSubTypes*(type);  
 *CachedSubscriberTypes*[type] = subscriberTypes;  
   
 return subscriberTypes;  
 }  
   
 private static List<Type> *GetListOfSubTypes*(Type type)  
 {  
 List<Type> subscriberTypes = new List<Type>();  
 foreach (var t in type.GetInterfaces())  
 {  
 if (t.GetInterfaces().Contains(typeof(ISubscriber)))  
 {  
 subscriberTypes.Add(t);  
 }  
 }  
 return subscriberTypes;  
 }  
}

using System;  
using UnityEngine;  
  
public class **LocalizationManager** : MonoBehaviour  
{  
 public event Action OnLanguageChanged;  
 private ITranslationsStorage \_translationsStorage;  
  
 public void Init(LanguageParserConfig parserConfig, StoredDataManager storedDataManager)  
 {  
 \_translationsStorage = new TranslationsStorage(parserConfig, storedDataManager);  
 }  
  
 private void **OnApplicationQuit**() => SaveCurrentLanguage();  
  
 private void **OnApplicationPause**(bool status) => SaveCurrentLanguage();  
  
 private void SaveCurrentLanguage()  
 {  
 \_translationsStorage?.SaveCurrentLanguage();  
 }  
  
 private void **Start**() => RaiseUpdateLanguageEvent();  
  
 public LanguagesEnums.Language GetCurrentLanguage()  
 {  
 return \_translationsStorage.GetCurrentLanguage();  
 }  
  
 public string GetTranslation(string itemID)  
 {  
 return \_translationsStorage.GetTranslation(itemID);  
 }  
  
 public void SetCurrentLanguage(LanguagesEnums.Language language)  
 {  
 \_translationsStorage.SetLanguage(language);  
 RaiseUpdateLanguageEvent();  
 }  
  
 private void RaiseUpdateLanguageEvent()  
 {  
 OnLanguageChanged?.Invoke();  
 }  
}

using System;  
using UnityEngine;  
  
public class **EnergyManager** : MonoBehaviour  
{  
 public event Action OnEnergyChanged;  
 [SerializeField] private EnergyRestoreProcessor **energyRestoreProcessor**;  
 private EnergySystemConfig \_config;  
 private StoredDataManager \_storedDataManager;  
 private SavedEnergyProgress \_savedEnergyProgress;  
 private IEnergyContainer \_energyContainer;  
  
 public void Init(EnergySystemConfig config, StoredDataManager storedDataManager, IEnergyContainer energyContainer)  
 {  
 \_config = config;  
 \_storedDataManager = storedDataManager;  
 \_energyContainer = energyContainer;  
  
 LoadLastSavedEnergy();  
 energyRestoreProcessor.Init(\_config);  
 \_energyContainer.Init(\_config, \_savedEnergyProgress.Energy);  
 GetOfflineEnergy();  
 }  
  
 private void **OnEnable**()  
 {  
 \_energyContainer.OnEnergyChanged += OnEnergyUpdated;  
 energyRestoreProcessor.OnRestoreComplete += AddEnergyAfterRestoring;  
 OnEnergyUpdated();  
 }  
  
 private void **OnDisable**()  
 {  
 \_energyContainer.OnEnergyChanged -= OnEnergyUpdated;  
 energyRestoreProcessor.OnRestoreComplete -= AddEnergyAfterRestoring;  
 }  
  
 private void LoadLastSavedEnergy()  
 {  
 var defaultEnergyProgress = new SavedEnergyProgress()  
 {  
 SaveTime = DateTime.Now,  
 Energy = \_config.MaxEnergy  
 };  
 \_savedEnergyProgress = \_storedDataManager.GetSavedData<SavedEnergyProgress>(defaultEnergyProgress);  
 }  
   
 private void GetOfflineEnergy()  
 {  
 if (\_energyContainer.IsFull()) return;  
  
 int offlineEnergy = energyRestoreProcessor.GetOfflineEnergy(\_savedEnergyProgress);  
 \_energyContainer.Add(offlineEnergy);  
 }  
   
 private void OnEnergyUpdated()  
 {  
 OnEnergyChanged?.Invoke();  
 if (\_energyContainer.IsFull())  
 {  
 energyRestoreProcessor.StopRestoring();  
 return;  
 }  
 energyRestoreProcessor.StartRestoring();  
 }  
   
 private void AddEnergyAfterRestoring()  
 {  
 \_energyContainer.Add(\_config.EnergyPerStep);  
 }  
  
 public void AddEnergyForAction(ActionWithEnergy action)  
 {  
 int energy = GetEnergyActionValue(action);  
 \_energyContainer.AddOverLimit(energy);  
 }  
  
 public void RemoveEnergy(ActionWithEnergy action)  
 {  
 int energy = GetEnergyActionValue(action);  
 if (IsEnoughEnergy(energy))  
 {  
 \_energyContainer.Remove(energy);  
 }  
 }  
  
 public EnergyState GetCurrentEnergyState() => \_energyContainer.GetEnergyState();  
  
 public bool IsEnoughEnergy(int energy) => \_energyContainer.IsEnough(energy);  
  
 public int GetEnergyActionValue(ActionWithEnergy action) => \_config.GetEnergyValue(action);  
  
 public TimeSpan GetCurrentRestoreInterval() => energyRestoreProcessor.GetCurrentRestoreInterval();  
  
 private void **OnApplicationPause**(bool pauseStatus) => SaveEnergyState();  
 private void **OnApplicationQuit**() => SaveEnergyState();  
  
 private void SaveEnergyState()  
 {  
 var currentEnergyState = GetCurrentEnergyState();  
 \_savedEnergyProgress.Energy = currentEnergyState.Energy;  
 \_savedEnergyProgress.RecoveryProgress = energyRestoreProcessor.GetCurrentRestoreStepProgress();  
 \_savedEnergyProgress.SaveTime = DateTime.Now;  
 \_storedDataManager.SaveEnergyProgress(\_savedEnergyProgress);  
 }  
   
 #if UNITY\_EDITOR  
   
 [EditorButton("Add 3 energy")]  
 public void AddEnergy()  
 {  
 \_energyContainer.AddOverLimit(3);  
 }  
   
 [EditorButton("Remove 3 energy")]  
 public void RemoveEnergy()  
 {  
 \_energyContainer.Remove(3);  
 }  
  
 #endif  
}

using System;  
using System.Collections;  
using UnityEngine;  
  
public class **EnergyRestoreProcessor** : MonoBehaviour  
{  
 public event Action OnRestoreComplete;  
 private EnergySystemConfig \_config;  
 private DateTime \_nextRestoreTime;  
 private bool \_hasOfflineProgress;  
 private bool \_isRestoreActive;  
  
 public void Init(EnergySystemConfig config)  
 {  
 \_config = config;  
 }  
  
 public void StartRestoring()  
 {  
 if (\_isRestoreActive) return;  
  
 StartCoroutine(RestoreProcess());  
 }  
  
 private IEnumerator RestoreProcess()  
 {  
 \_isRestoreActive = true;  
  
 if (\_nextRestoreTime != default && \_hasOfflineProgress)  
 {  
 \_nextRestoreTime = \_nextRestoreTime.AddSeconds(\_config.TimeToRestoreStep);  
 \_hasOfflineProgress = false;  
 }  
 else  
 {  
 \_nextRestoreTime = DateTime.Now.AddSeconds(\_config.TimeToRestoreStep);  
 }  
  
 yield return new WaitWhile(() => GetCurrentRestoreInterval().TotalSeconds > 0);  
   
 \_isRestoreActive = false;  
 OnRestoreComplete?.Invoke();  
 }  
  
  
  
 public TimeSpan GetCurrentRestoreInterval() => \_nextRestoreTime.Subtract(DateTime.Now);  
  
 public int GetOfflineEnergy(SavedEnergyProgress savedEnergyProgress)  
 {  
 var timeIntervalAfterSaving = DateTime.Now.Subtract(savedEnergyProgress.SaveTime);  
 var remainingTimerSeconds = savedEnergyProgress.RecoveryProgress \* \_config.TimeToRestoreStep;  
 var timerProgress = \_config.TimeToRestoreStep - remainingTimerSeconds;  
 var elapsedTimeAfterSaving = timeIntervalAfterSaving.TotalSeconds + timerProgress;  
 var energyValue = \_config.EnergyPerStep \* elapsedTimeAfterSaving / \_config.TimeToRestoreStep;  
 \_nextRestoreTime = DateTime.Now.AddSeconds(-energyValue % 1 \* \_config.TimeToRestoreStep);  
 \_hasOfflineProgress = true;  
 return (int)energyValue;  
 }  
}

# **ПРИЛОЖЕНИЕ Б**

(обязательное)

**Руководства пользователя**

1. **Введение.**

«Minecraft Arkanoid» — развлекательная игра, в которой управляя кроватью можно манипулировать мячом разбивая блоки. В игре есть системы энергии, отслеживания прогресса уровней, сохранения игровых данных и локализация на русский и английский язык.

Для работы с приложением потребуется минимальный уровень знаний работы с мобильными устройствами или планшетами.

1. **Назначение и условие использования.**

Игра «Minecraft Arkanoid» несет развлекательный характер. Целевая аудитория игры — пользователи стран СНГ возрастом 6 — 14 лет. Данная игра позволяет пользователю развлечься.

Работа приложения должна производиться на мобильном устройстве или планшете операционной системы Android или IOS, обладающим оперативной памятью не менее 256 Мб и сенсорным экраном.

1. **Подготовка к работе.**

Запуск производится на мобильном устройстве или планшете операционной системы Android и IOS, обладающим оперативной памятью не менее 256 Мб и сенсорным экраном. На устройстве должен быть файл MinecraftArkanoid.exe который следует запустить для установки приложения.

1. **Описание операций.**

При входе в приложение игроку открывается окно загрузки, изображенное на рисунке 1.



Рисунок 1 — Окно загрузки

В окне загрузки заполняется индикатор отображающий прогресс загрузки приложения. После загрузки откроется главное меню игры, показанное на рисунке 2.



Рисунок 2 — Главное меню программы.

На главном меню изображен баланс энергии, за которую покупаются попытки прохождения уровней. Таймер восстановления энергии показан под счетчиком. Также в меню изображены кнопки «EN», «RU» при нажатии на которые интерфейс приложения меняет язык на английский и русский соответственно. При нажатии на кнопку «Играть» при русской локализации или «Play» при английской открывается меню выбора набора уровней, показанное на рисунке 3.



Рисунок 3 — Меню выбора набора уровней

В меню выбора набора уровней расположена кнопка для перехода в главное меню, индикатор энергии и список наборов уровней.

При нажатии на набор уровней откроется первый уровень в этом наборе. Игровой уровень показан на рисунке 4.

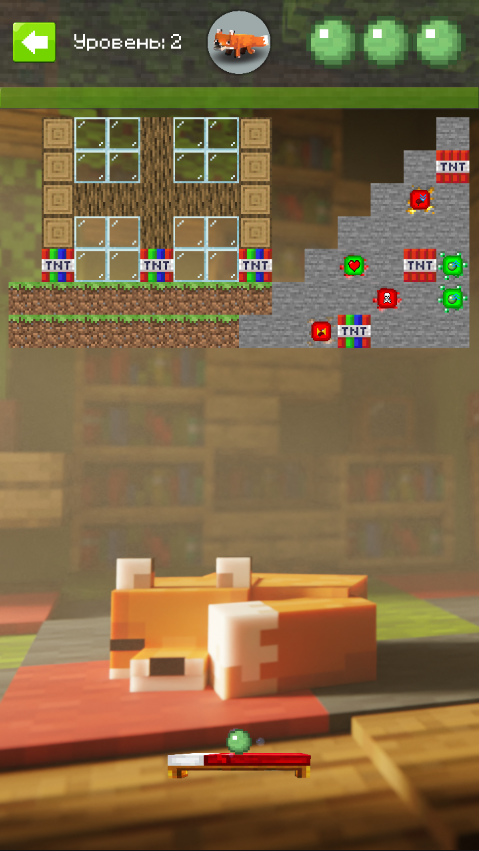


Рисунок 4 — Игровой уровень

После загрузки в игровом уровне появляется карта из блоков, платформа которой можно управлять, шарик манипулируя которым можно уничтожать блоки, индикатор попыток, кнопка перехода в меню уровня и индикатор прогресса прохождения уровня.

Пользователь может управлять платформой, перемещая её вправо и влево отталкивая шарик. Шарик непрерывно отталкивается от всех объектов. Если шарик оттолкнется от блока в карте, то он повредит его или уничтожит. Если же шарик упадет вниз и пропадет, то баланс попыток уменьшится на 1 и появится новый шар, либо при отсутствии попыток игрок попадет в меню уровня. При каждом уничтоженном блоке прогресс прохождения карты увеличивается. У каждого блока есть свое свойство здоровье, то есть того сколько раз его нужно ударить чтобы уничтожить. Также есть блоки с бонусами при уничтожении которых падает определенный бонус, использовать его можно коснувшись платформой. После прохождения уровня открывается следующий уровень в наборе. После прохождения набора уровней открывается следующий набор уровней.

При нажатии на кнопку перехода в меню уровня открывается меню уровня, показанное на рисунке 5.



Рисунок 5 — Меню уровня

В меню уровня есть кнопка перехода в меню выбора наборов уровня, при нажатии на который текущий уровень закроется. Также в меню есть индикатор энергии, кнопка покупки второго шанса за 5 энергий и кнопка для прохождения уровня заново за 3 энергии.

1. **Аварийные ситуации.**

При каких-либо технологических действиях, например, закрытия приложения и других аварийных ситуаций, приложение закроется и несохраненный прогресс будет утерян.

1. **Рекомендации по освоению.**

Перед началом использования рекомендуется ознакомится с руководством пользователя.

# **ПРИЛОЖЕНИЕ В**

(обязательное)

**Руководство программиста**

**1. Назначение и условия применения программ.**

Для запуска приложения потребуется мобильное устройство или планшет с операционной системой Android или IOS. Для работы с приложением потребуется минимальный уровень знаний работы с мобильными устройствами или планшетами.

**2. Характеристики программы.**

Работа с программой подразумевает один режим функционирования: в данном режиме доступны все функции управления и манипулирования объектами. Данная программа имеет легкий и удобный графический интерфейс, что позволяет быстро освоиться и начать работать с ней. Кроме того, программа имеет достаточно высокую скорость работы, что позволяет быстро обрабатывать данные и выполнять задачи.

**3. Обращение к программе.**

После запуска системы на экране отображается окно загрузки, после завершения которой запускается главное меню игры.

Выход из системы возможен при закрытии приложения.

**4. Входные и выходные данные.**

Чтобы добавить новые уровни в систему, необходимо перейти по следующему пути в файлах игры: «\Assets\ARCANOID\GameConfiguration\ GameplayConfigs\LevelPacks\Packs\Normal Packs» и выбрать папку, соответствующую набору. В этой папке выбрать папку с уровнями и положить в нее уровень. На рисунке 1 представлен вид файлов в системе.

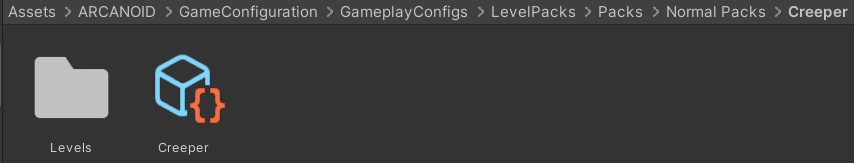


Рисунок 1 — Вид файлов набора уровней

Следующим шагом необходимо настроить компонент «LevelPack» добавив в список уровней новый уровень и присвоить ему тот уровень, который добавляли. На рисунке 2 представлен вид компонента.

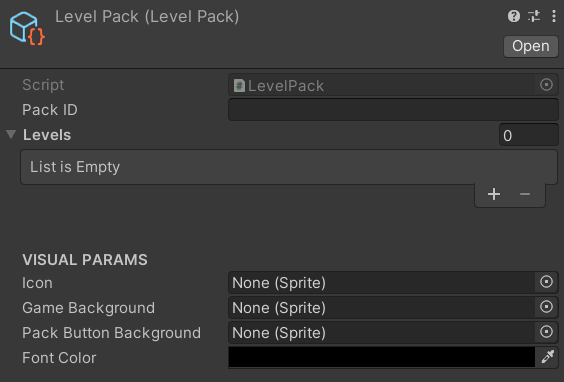


Рисунок 2 — Вид компонента «LevelPack»

Для того чтобы создать новый набор уровней необходимо перейти по следующему пути в файлах игры: «\Assets\ARCANOID\GameConfiguration\ GameplayConfigs\LevelPacks\Packs\Normal Packs». После чего создать папку с названием набора. В этой папке создать подпапку с уровнями, а также создать компонент «LevelPack» назвать его также как и набор уровней, он показан на рисунке 2. Далее необходимо настроить компонент установив ему значения уровни, иконку, задний фон, имя и цвет текста.

После настройки компонента необходимо перейти в папку и компоненту «Assets\ARCANOID\GameConfiguration\GameplayConfigs\ LevelPacks\PackSets\MainSet.asset», изображенный на рисунке 3 и добавить новое значение, присвоив ему созданный набор.

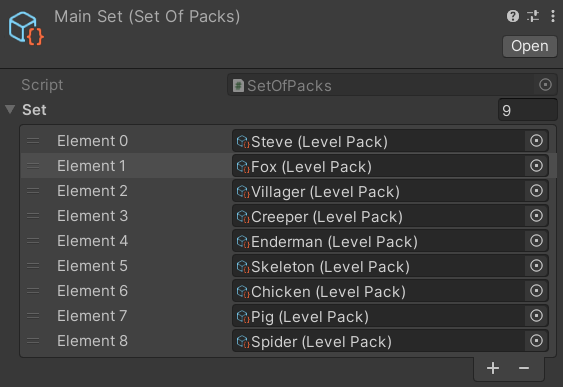


Рисунок 3 — Вид компонента «MainSet»

После настройки компонента «MainSet» необходимо настроить локализацию набора для английского и русского языка. Для этого необходимо перейти в папку «Assets\ARCANOID\Data\Localization» и изменить файлы «EnglishTranslation» и «RussianTranslation» добавив в них перевод. Пример файла с английской локализацией показан на рисунке 4. Для русского языка все аналогично.



Рисунок 4 — Пример файла с английской локализацией

Таким образом происходит добавление и изменение уровней в игре.

Входными данными в приложении являются наборы уровней и уровни.

Выходными данными является прогресс прохождения игры.

Кодирование входных и выходных данных не предусматривается.

**5. Сообщения.**

Данная программа предусматривает вывод сообщения, он представлен на рисунке 5.



Рисунок 5 — Сообщение о прохождении набора уровней

Данное сообщение выводится на экран при завершении всех уровней в наборе.