МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01 «Программное обеспечение

информационных технологий»

**КУРСОВОЙ ПРОЕКТ**

по предмету: «Конструирование программ и языки программирования»

**на тему: «Разработка игрового приложения «Minecraft Arkanoid»**

Пояснительная записка

Разработала обучающаяся

группы \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

Руководитель \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(дата) (подпись) (ФИО)

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отметка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2023

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc138772703)

[1. Описание методов и средств разработки приложения 5](#_Toc138772704)

[1.1 Обзор и анализ игрового процесса в жанре аркада 5](#_Toc138772705)

[1.2 Обоснование инструментария разработки приложения 6](#_Toc138772706)

[2. Проектирование программного продукта 9](#_Toc138772707)

[2.1 Функциональная модель программы 9](#_Toc138772708)

[2.2 Алгоритм работы программы 11](#_Toc138772709)

[3. Разработка программы 13](#_Toc138772710)

[3.1 Разработка графического интерфейса 13](#_Toc138772711)

[3.2 Компоненты приложения 15](#_Toc138772712)

[3.3 Описание программной реализации программы 16](#_Toc138772713)

[Заключение 22](#_Toc138772714)

[Список использованных источников 23](#_Toc138772715)

[Приложение А 24](#_Toc138772716)

[Приложение Б 30](#_Toc138772717)

[Приложение В 34](#_Toc138772718)

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 подразумевают 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.

«Minecraft Arcanoid» будет разработан как мобильная игра, запускаемая на устройствах с операционными системами «Android» и «iOS».

«iOS» — мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей, разрабатываемая и выпускаемая американской компанией «Apple».

«Android» — 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.

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 уникальных уровней и наборов уровней в игре;
* индикатор процесса загрузки
* сохранение игрового процесса;
* система энергии, восстанавливаемой со временем;
* физика игровых компонентов;
* управление игроком платформы;
* система попыток игры на уровне;
* индикатор прогресса прохождения уровня.

1 ОПИСАНИЕ МЕТОДОВ И СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

* 1. **Обзор и анализ игрового процесса в жанре аркада**

«Арканоид» изначально разрабатывался для специализированных автоматов. Официальная дата выхода — 1986 год. Продукт разрабатывался японской игровой корпорации «Taito», руководимой тогда советским гражданином Михаилом Коганом. Сегодня же классическая версия игры насчитывает более двухсот приложений.

В 1958 году физик Уильям Хигинботам из различных радиодеталей создатель атомной бомбы собрал некое устройство и подключил к осциллографу. На экране прибора появились ракетка и мячик.

А в 1976 году Стиву Джобсу, будущему соучредителю «Apple», поручили создать новую игру «Breakout». Интерфейс игры представлен на рисунке 1.1. Ее цель — бить мячиком по цветным блокам, пока не останется ни одного. Важно при этом, чтобы мяч не проходил мимо ракетки. Все эти игры можно считать прародителями будущего «Арканоида».

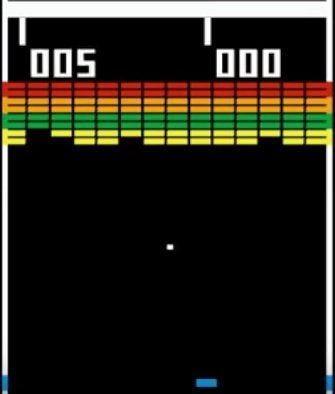


Рисунок 1.1 — Интерфейс игры «Breakout» 1976 года

Игры «Арканоид» разрабатываются в жанре аркады. Аркада — распространённый в индустрии компьютерных игр термин, обозначающий игры с нарочито примитивным игровым процессом.

Аркада как жанр понимается сегодня примерно как игра, в которой игроку приходится действовать быстро, полагаясь в первую очередь на свои рефлексы и реакцию. Аркада характеризуется развитой системой бонусов: начисление очков, постепенно открываемые элементы игры и т. д.

Приложение «Minecraft Arkanoid» — однопользовательская мобильная игра.

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

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

Ремейк компьютерной игры[9] — 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 процесс игры, 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 процесс игры в определённых случаях также могут быть изменены и адаптированы согласно современным трендам и стандартам.

Игровой процесс 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 их платформой.

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.

Создание игры «Minecraft Arkanoid» является современной адаптацией старых «Арканоидов», которая принесет свежий и привлекательный вид программы, добавит новые механики и адаптирует игру для современных устройств.

* 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»[8] — 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.

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» 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;
* 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.

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 программы**

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.

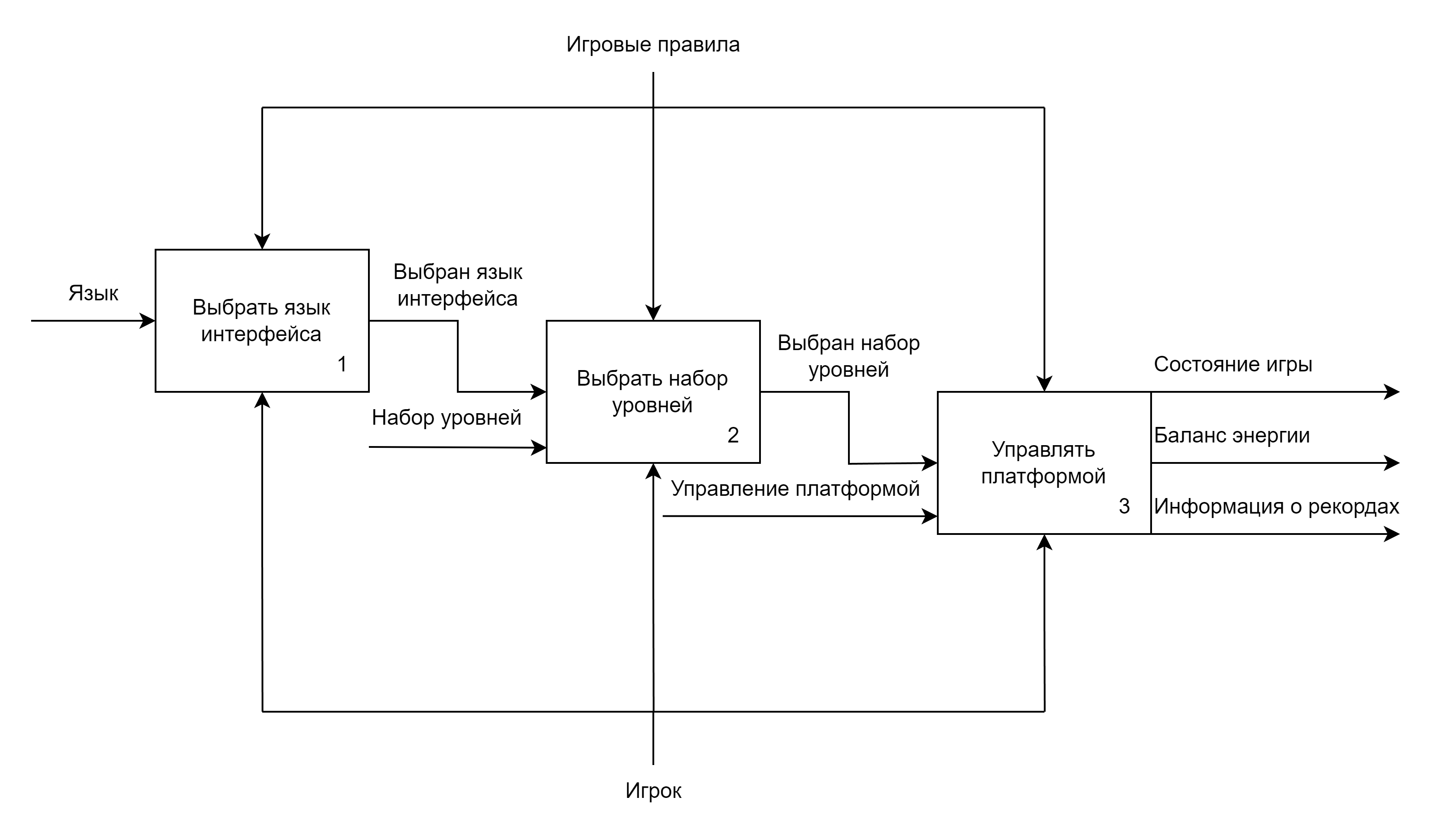
Контекстная диаграмма системы предоставлена на рисунке 2.1.



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»[1] — 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 программы**

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.

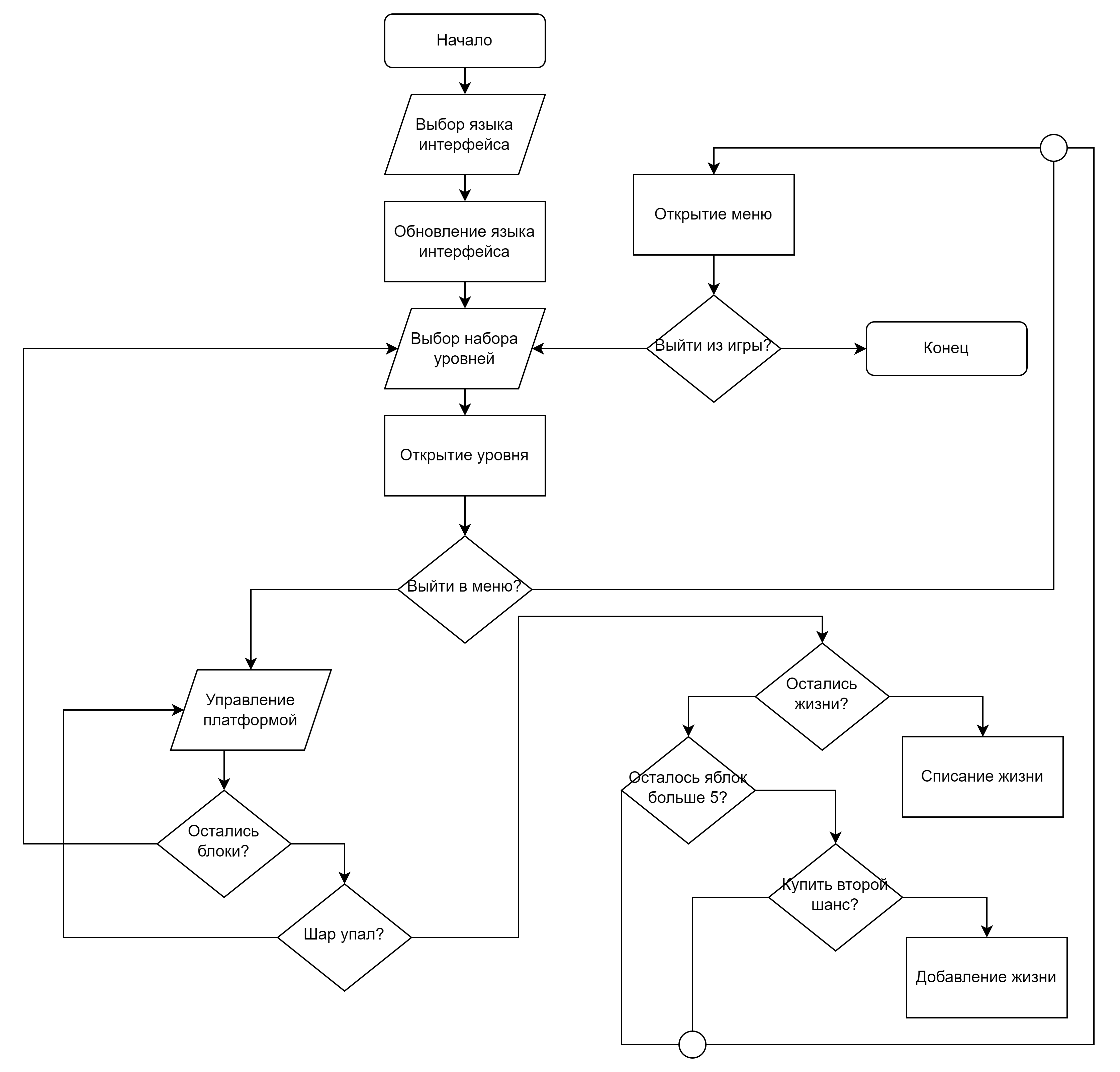


Рисунок 2.4 — Блок-схема приложения

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 РАЗРАБОТКА ПРОГРАММЫ**

**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.

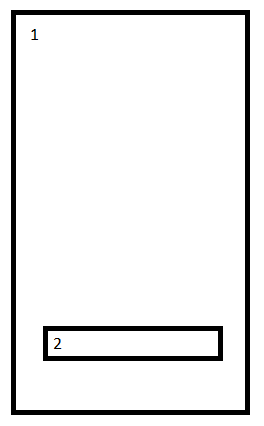


Рисунок 3.1 — Схема окна загрузки: 1 — окно загрузки; 2 — индикатор загрузки

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

На рисунке 3.2 представлена схема главного меню приложения.

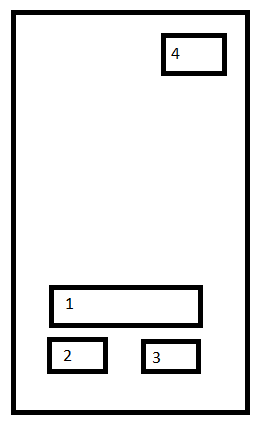


Рисунок 3.2 — Схема главного меню: 1 — кнопка перехода в меню выбора уровней; 2 — кнопка смены языка на английский; 3 — кнопка смены языка на русский; 4 — индикатор энергии

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

На рисунке 3.3 представлена схема меню выбора уровней.

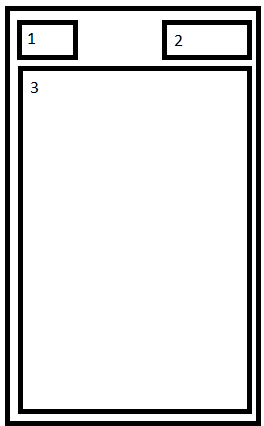


Рисунок 3.3 — Схема меню выбора уровней: 1 — кнопка для выхода в главное меню; 2 — индикатор энергии; 3 —список наборов уровней;

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

На рисунке 3.4 представлена схема игрового уровня.

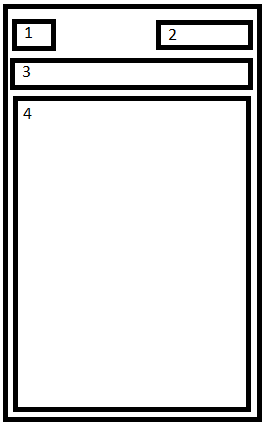


Рисунок 3.4 — Схема игрового уровня: 1 — кнопка выхода в меню уровня; 2 — индикаторы попыток; 3 — индикатор прогресса прохождения уровня; 4 — карта игрового уровня

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

На рисунке 3.5 представлена схема меню уровня.

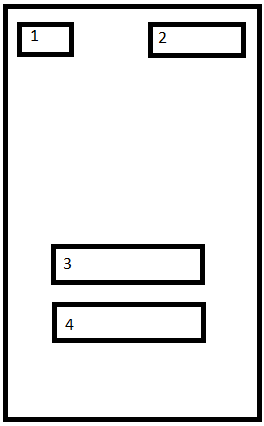


Рисунок 3.5 — Схема меню уровня: 1 — кнопка для выхода в меню выбора уровней; 2 — индикатор энергии; 3 — кнопка для покупки второго шанса игры; 4 — кнопка для прохождения уровня заново

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.

В ходе этого этапа был спроектирован 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 |
| 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 программы**

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

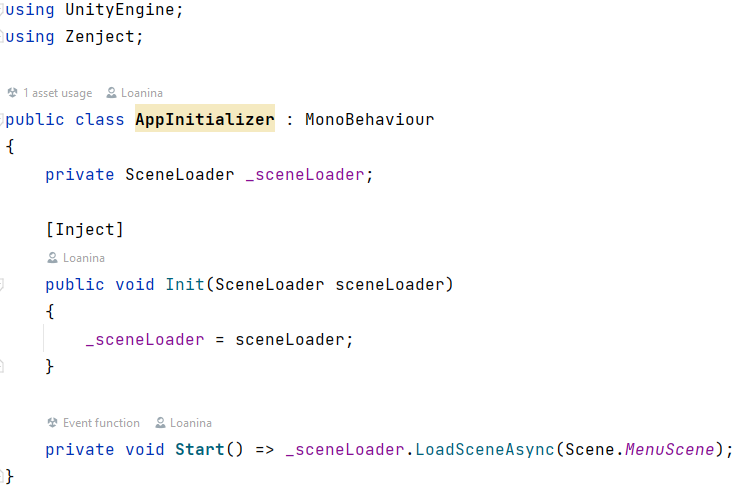


Рисунок 3.10 — Код загрузки сцены

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

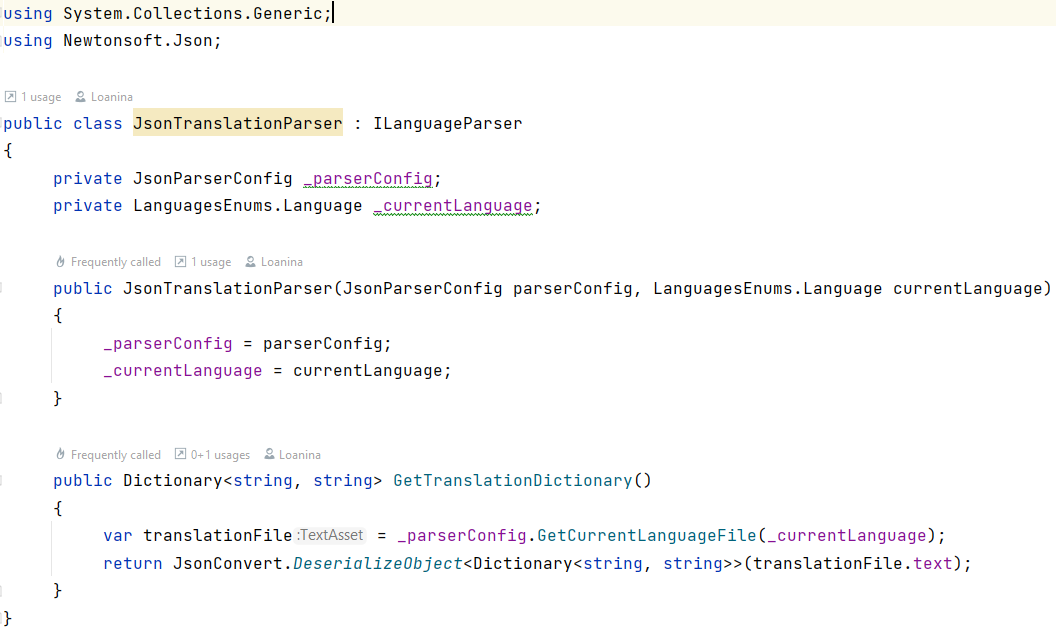


Рисунок 3.12 — Код осуществляющий перевод текста

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.

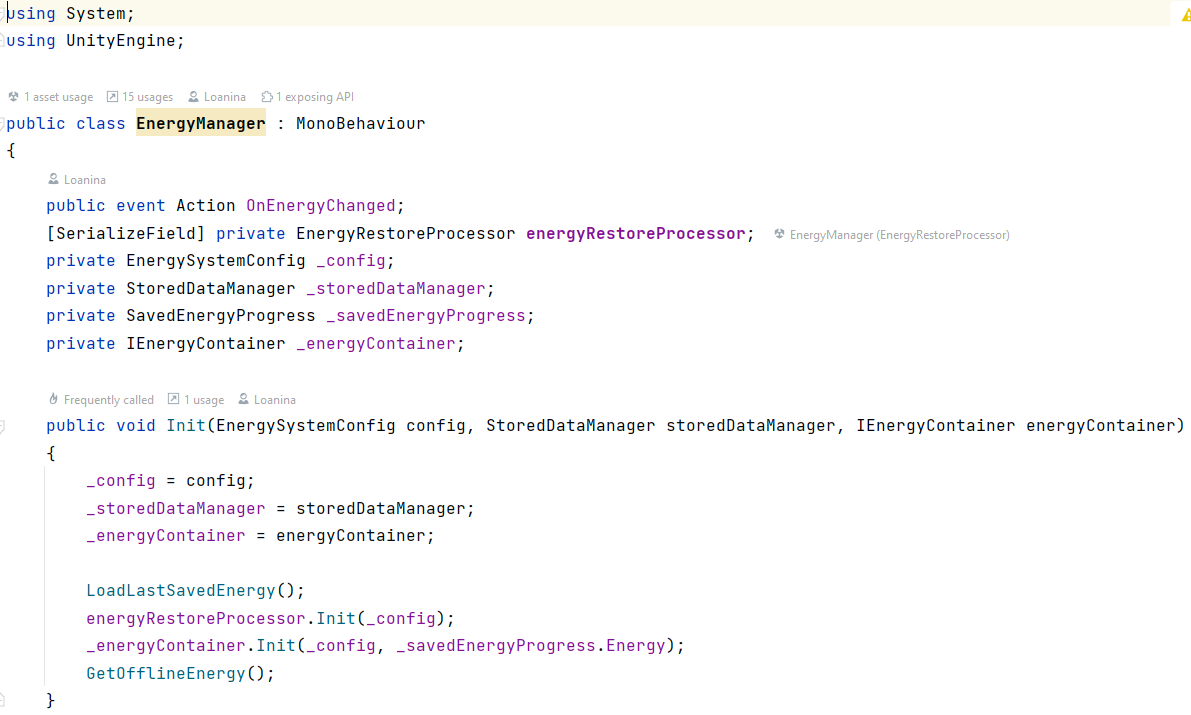


Рисунок 3.13 — Код системы энергии

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

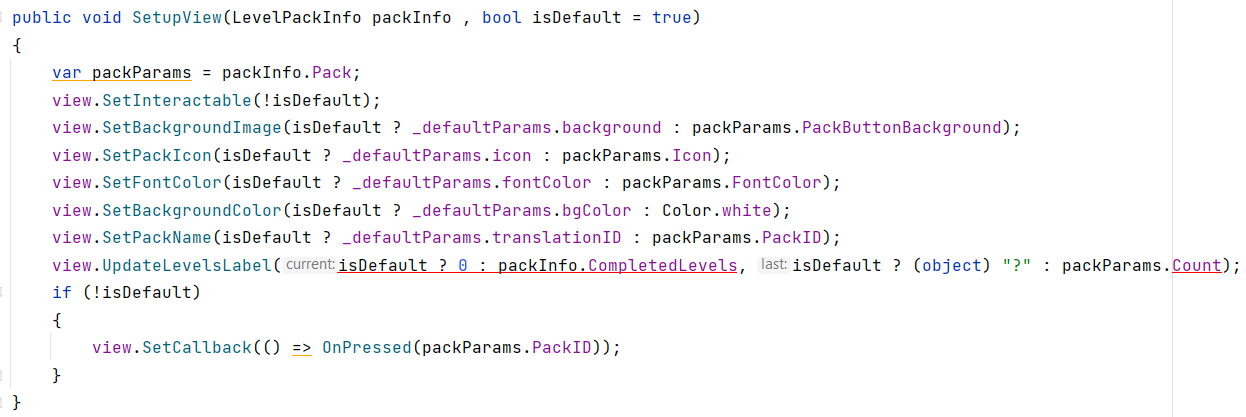


Рисунок 3.14 — Код обработки набора уровней

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.

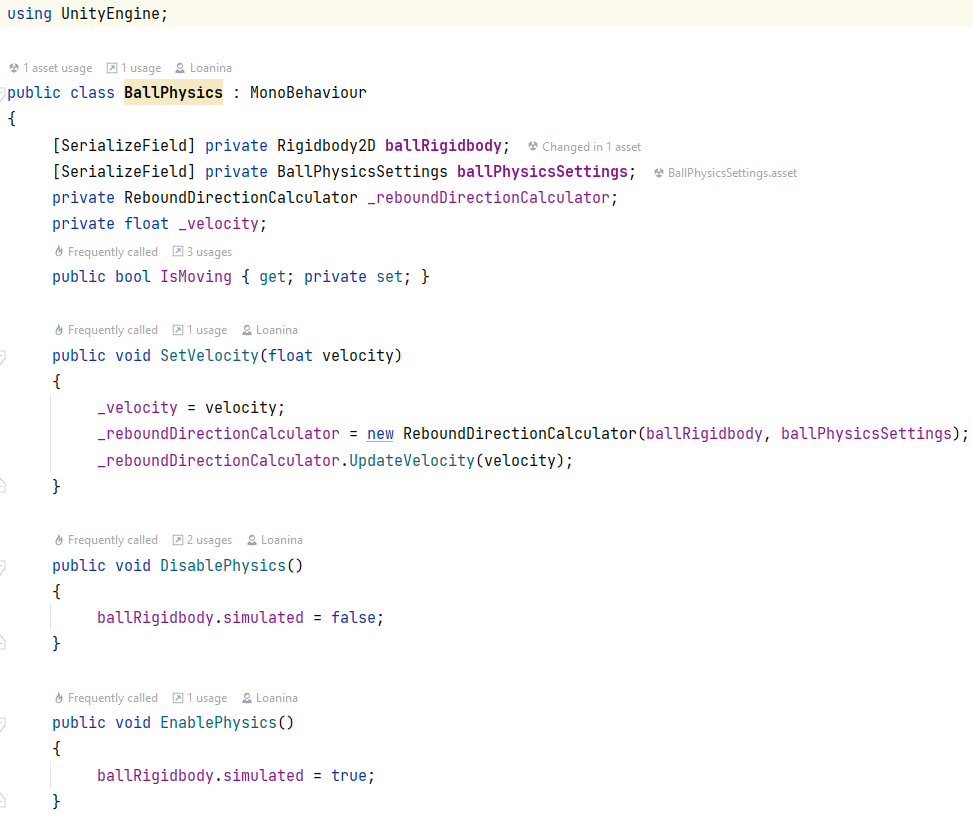


Рисунок 3.16 — Код обработки перемещения шара

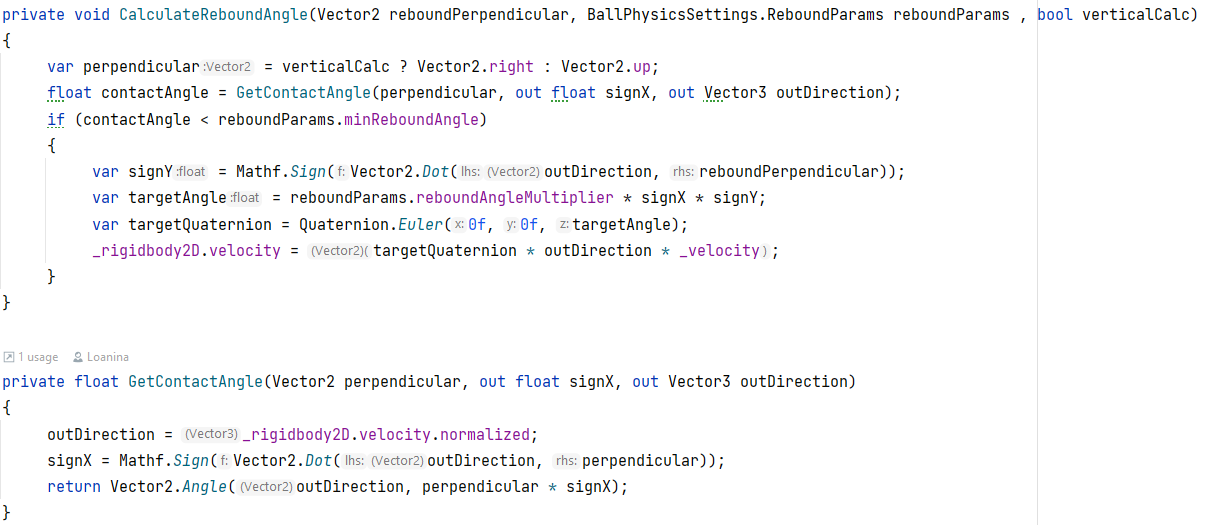


Рисунок 3.17 — Код вычисления углов при отталкивании

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.

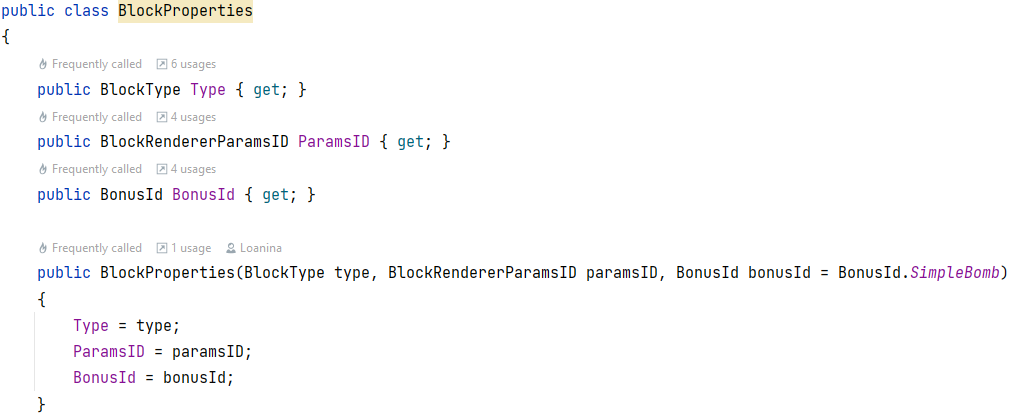


Рисунок 3.18 — Структура блоков

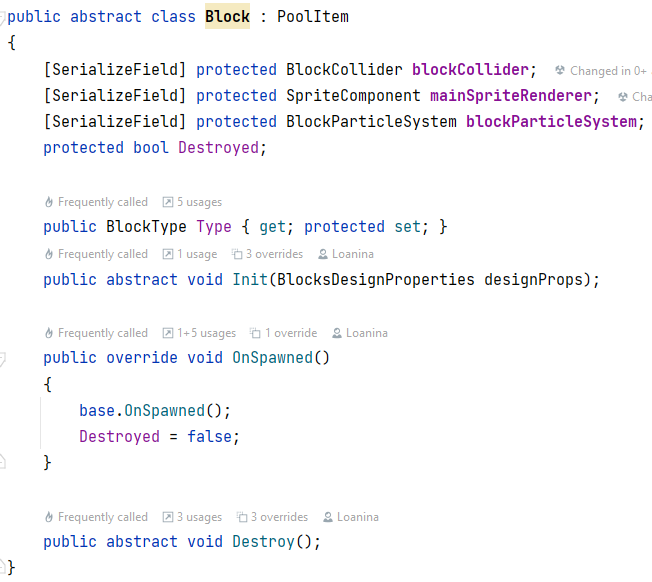


Рисунок 3.19 — Код обработки блоков

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

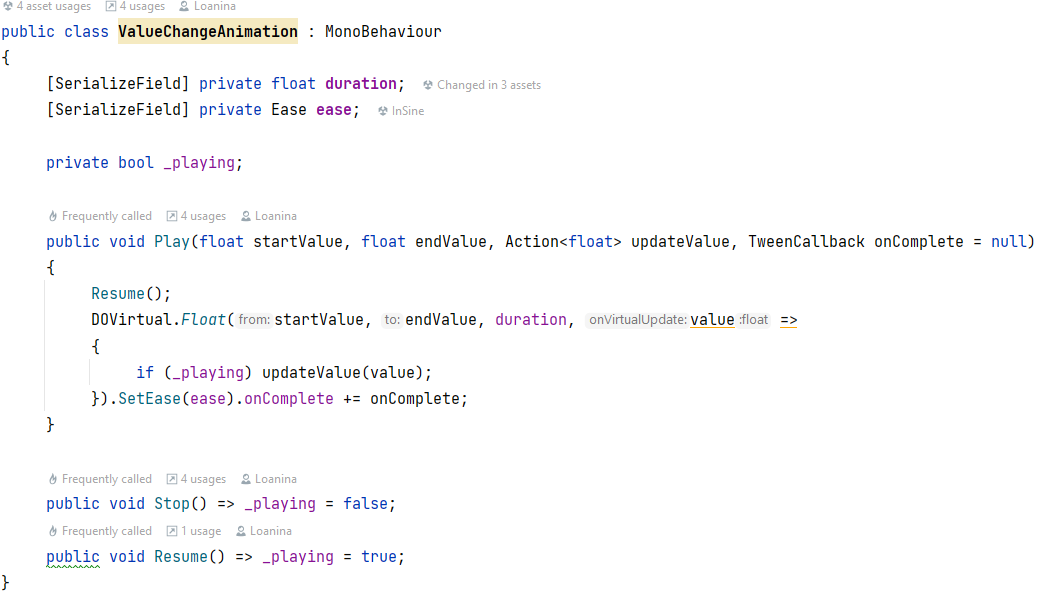


Рисунок 3.20 — Код обработки индикатора прогресса

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.

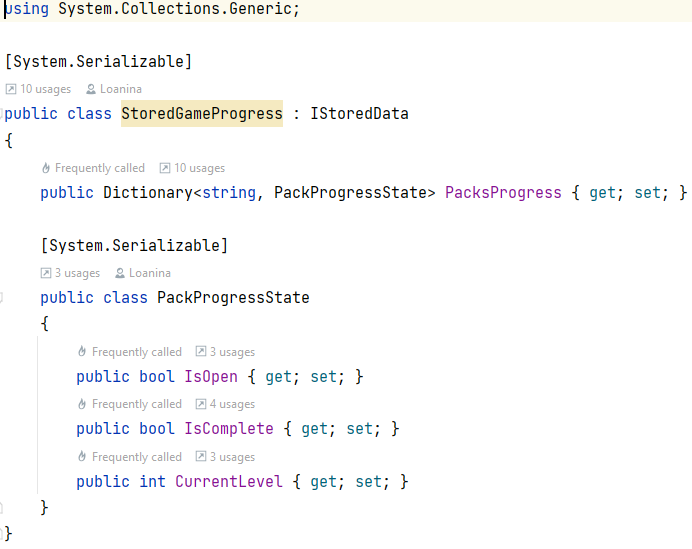


Рисунок 3.21 — Структура сохраняемого прогресса



Рисунок 3.22 — Код сохранения данных

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

## **ЗАКЛЮЧЕНИЕ**

В результате курсового проекта было разработано игровое приложение «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.

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 функциональная схема системы с её последующими декомпозициями, схема вариантов использования и схема алгоритма работы программного средства. Далее был разработан и реализован интерфейс приложения, написан и протестирован код приложения. 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 Б.1.



Рисунок Б.1 — Окно загрузки

В окне загрузки заполняется индикатор отображающий прогресс загрузки приложения. Заполнение индикатора загрузки приложения показано на рисунке Б.2.

Рисунок Б.2 — Заполнение индикатора загрузки приложения

В окне загрузки также присутствует анимация названия игры «Minecraft Arkanoid», показанная на рисунке Б.3.

Рисунок Б.3 — Анимация названия игры «Minecraft Arkanoid»

После загрузки откроется главное меню игры, показанное на рисунке Б.4.



Рисунок Б.4 — Главное меню игры

На главном меню изображен баланс энергии, за которую покупаются попытки прохождения уровней. Таймер восстановления энергии находится под балансом, он изображен на рисунке Б.5. Когда пользователь тратит энергию, одна энергия восстанавливается каждые 30 секунд в плоть до его максимального количества — 30 после чего таймер исчезает.

Рисунок Б.5 — Таймер восстановления энергии

Также в меню изображены кнопки «EN», «RU» при нажатии на которые интерфейс приложения меняет язык на английский и русский соответственно. Интерфейс с английским языком показан на рисунке Б.6, а с русским на рисунке Б.7.

Рисунок Б.6 — Интерфейс на английском языке

Рисунок Б.7 — Интерфейс на русском языке

При нажатии на кнопку «Играть» при русской локализации или «Play» при английской открывается меню выбора набора уровней. В промежутке открытия окна появляется изображение перехода между сценами, показанное на рисунке Б.8. Меню выбора набора уровней показано на рисунке Б.9.

Рисунок Б.8 — Изображение перехода между сценами



Рисунок Б.9 — Меню выбора набора уровней

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

В списке наборов уровней каждый экземпляр уникальный и имеет свое название, иконку, задний фон, количество пройденных и общее количество уровней.

Количество пройденных уровней увеличивается при каждом прохождении уровня в наборе, что продемонстрировано на рисунке Б.10.

Рисунок Б.10 — Количество пройденных и общее количество уровней в наборе

При первой загрузке игры открыт лишь один набор уровней, остальные имеют определенный вид, показанный на рисунке Б.11, и невозможность к открытию. Наборы открываются сразу же после прохождения всех уровней из предшествующего набора.

Рисунок Б.11 — Закрытый набор уровней

При прокрутке списка наборов уровней элементы перемещаются вверх и вниз, что показано на рисунке Б.12.

Рисунок Б.12 — Прокрутка списка наборов уровней

Пользователь в меню наборов уровней может выбрать любой открытый набор уровней чтобы его открыть, однако для игры ему нужно потратить 3 энергии. Если же у пользователя нету 3-х энергий для игры, то он не сможет открыть любой набор уровней до тех пор, пока она не восстановится. Когда у игрока менее 3-х энергий все наборы уровней закрываются, что показано на рисунке Б.13.

Рисунок Б.13 — Закрытие наборов уровней при недостатке энергии

При открытии набора уровней запускается первый не пройденный уровень в наборе. Если же все уровни в наборе пройдены, и пользователь решил переиграть его, то откроется первый уровень из этого набора. Один из игровых уровней показан на рисунке Б.14.

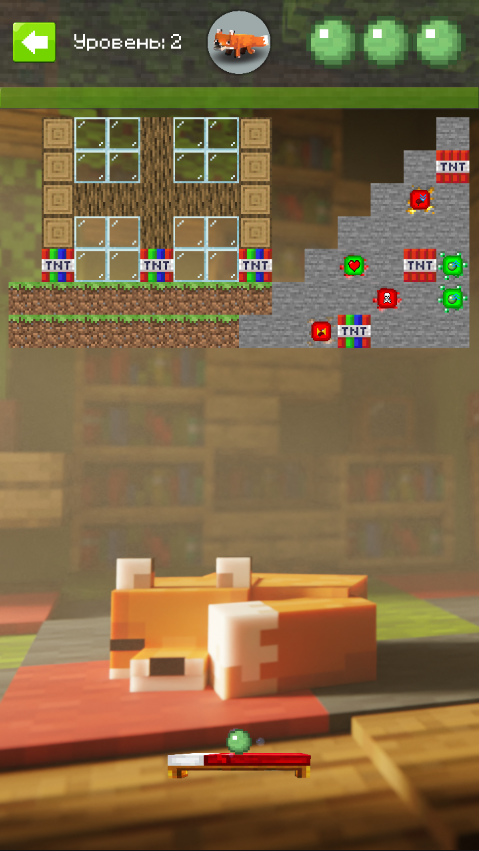


Рисунок Б.14 — Один из игровых уровней

Каждый уровень и набор уровней в игре уникальный. В каждом наборе уровней меняется задний фон игровой сцены и иконка набора, а в каждом уровне карта из блоков. На рисунке Б.15 показаны отличия уровней из двух разных наборов.

Рисунок Б.15 — Отличия уровней в разных наборах

В игровом уровне находятся следующие функциональные элементы:

* индикатор прогресса прохождения уровня;
* индикатор попыток;
* карта из блоков;
* платформа;
* шарик;
* надпись с текущим уровнем;
* кнопка перехода в меню уровня.

Пользователь может управлять игровой платформой, перемещая ее вправо и влево, что продемонстрировано на рисунке Б.16.

Рисунок Б.16 — Перемещение игровой платформы

С помощью платформы игрок может манипулировать шариком. Шар запускается после первого клика или перемещения платформы и перемещается вверх, отталкиваясь от всех объектов на игровой сцене. Если шар упадет ниже платформы, то он уничтожится и создастся новый, при наличии попыток игры.

Шар, столкнувшись с блоком повредит его, либо уничтожит в зависимости от прочности блока. Этот процесс продемонстрирован на рисунке Б.17.

Рисунок Б.17— Повреждение блока при столкновении с шариком

У каждого блока есть своя текстура и прочность. Весь набор блоков изображен на рисунке Б.18.

Рисунок Б.18 — Набор всех блоков

Степень прочности блока варьируется от 1 (уничтожается после первого удара шариком) до 5 (уничтожается после пятого удара шариком). При уничтожении блока появляется анимация распада блока, она продемонстрирована на рисунке Б.19.

Рисунок Б.19 — Анимация распада блока

После повреждения блока на него накладывается изображения трещины, которое зависит от количества ранее нанесенных ударов. Изображения трещин показаны на рисунке Б.20.

Рисунок Б.20 — Изображения трещин после повреждения блока

Некоторые блоки являются бонусными. После их уничтожения падает определенный бонус. На бонусных блоках есть изображение бонуса, эти изображения показаны на рисунке Б.21. Красным цветом отображаются отрицательные бонусы, которые будут препятствовать игре, а зеленым положительные, которые, наоборот, упрощают прохождение уровня.

Рисунок Б.21 — Изображения бонусов на блоках

В игре имеются 8 различных бонусов. Их вид при выпадении показан на рисунке Б.22.

Рисунок Б.22 — Вид бонусов при выпадении

Далее описан функционал, который несут бонусы.

Бонус добавление попытки игры: при поднятии такого бонуса игроку начисляется одна попытка игры, что продемонстрировано на рисунке Б.23.

Рисунок Б.23 — Результат поднятия бонуса на добавление попытки игры

Бонус на исключение попытки игры: при поднятии такого бонуса вычитается одна попытка игры. Результат поднятия такого бонуса показан на рисунке Б.24.

Рисунок Б.24 — Результат поднятия бонуса на исключение попытки игры

Бонус на усиление шара: при поднятии такого бонуса шарик меняет свой вид и уничтожает все блоки на пути вне зависимости от их прочности и не отталкивается от них. Эффект от такого бонуса изображен на рисунке Б.25.

Рисунок Б.25 — Эффект от поднятия бонуса на усиление шара

Бонус на расширение платформы: при поднятии такого бонуса игровая платформа расширяется, что продемонстрировано на рисунке Б.26.

Рисунок Б.26 — Результат поднятия бонуса на расширение платформы

Бонус на сужение платформы: при поднятии такого бонуса игровая платформа сужается;

* на замедление шарика: при поднятии такого бонуса шарик начинает двигаться медленнее;
* на ускорение шарика: при поднятии такого бонуса шарик начинает двигаться быстрее;
* на ускорение управления платформой: при поднятии такого бонуса игрок может быстрее перемещать платформу;
* на замедление управления платформой: при поднятии такого бонуса игрок будет перемещать платформу медленнее.

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 Б.5.



Рисунок Б.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 В.1 представлен вид файлов в системе.

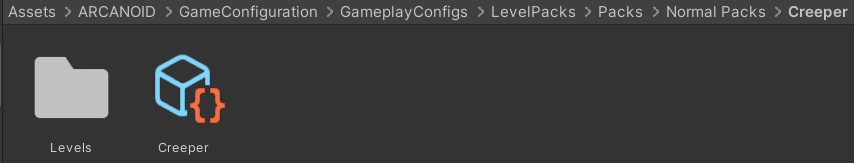


Рисунок В.1 — Вид файлов набора уровней

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

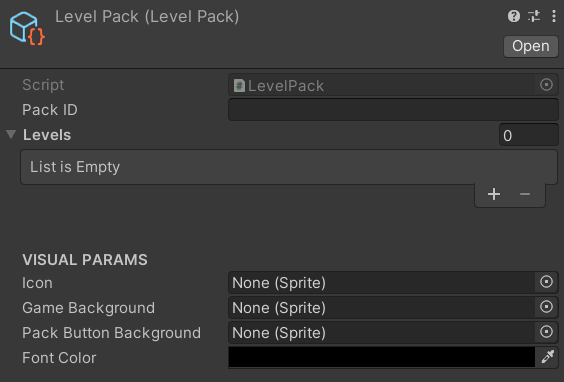


Рисунок В.2 — Вид компонента «LevelPack»

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 В.3 и добавить новое значение, присвоив ему созданный набор.

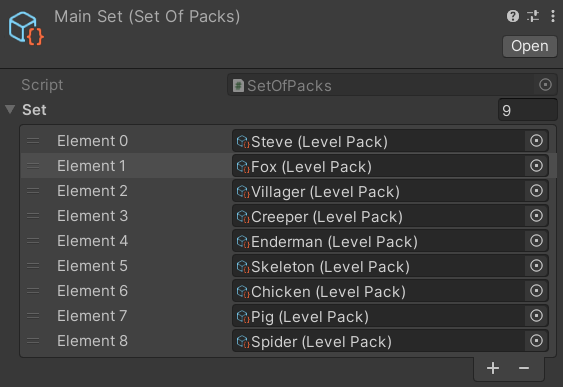


Рисунок В.3 — Вид компонента «MainSet»

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 В.4. Для русского языка все аналогично.



Рисунок В.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 В.5.



Рисунок В.5 — Сообщение о прохождении набора уровней

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