Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина «Объектно-ориентированное программирование и проектирование»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры ПИКС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Листратенко |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Игровое программное средство “Карточная игра” для мобильных устройств»**

БГУИР КП 1-40 05 01-10

|  |  |
| --- | --- |
|  | Выполнил студент группы 114301  Тюленев Юрий-Франтишек Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2022

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 2](#_Toc121006277)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ 3](#_Toc121006278)

[1.1 Анализ исходных данных к курсовому проекту 3](#_Toc121006279)

[1.2 Обоснование и описание выбора языка программирования, средств разработки, используемых технологий и сторонних библиотек 3](#_Toc121006280)

[**2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА** 5](#_Toc121006281)

[**2.1 Постановка задачи** 5](#_Toc121006282)

[**2.2 Проектирование архитектуры и описание состояний программного средства** 6](#_Toc121006283)

# **ВВЕДЕНИЕ**

Сегодня мобильные устройства занимают не последнее место в нашей жизни. В частности мобильные игры. Благодаря мобильности и мощности современных телефонов, они дают возможность играть в игры, по возможностям не сильно уступающим играм на персональных компьютерах, причем играть в совершенно любых местах.

Люди играют пока едут в метро, чтобы скоротать время, в перерывах на работе, чтобы расслабится, на парах в университете, когда им скучно.

Много мобильных игр позволяет играть людям вместе, совместно или против друг друга, находясь на тысячи километров друг от друга.

В данной работе рассматривается разработка игрового программного средства “Карточная игра”.

Цель курсового проекта – дать возможность интересно и просто играть людям вместе, через интернет.

Для достижения цели курсового проекта необходимо решить следующие задачи:

– изучить предметную область;

– построить алгоритм программы;

– разработать интерфейс программы;

– построить архитектуру программы;

– наладить взаимодействие клиента и сервера

–протестировать программу, исправить обнаруженные ошибки.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

* 1. Анализ исходных данных к курсовому проекту

Тема курсовой работы: разработка игрового программного средства “Карточная игра” для пользователей под ОС Android.

Cредства разработки Unity

Язык программирования С#.

Парадигма программирования – объектно-ориентированная.

Способ организации данных – классы.

Способ хранения данных – Player Prefs

К защите курсовой работы представляются:

1. мобильное программное средство;
2. схема алгоритма (формат А2/А3);
3. UML диаграмма классов (плакат, формат А2/А3);
4. диаграмма состояний (плакат, формат А2/А3);
5. структура графического пользовательского интерфейса (плакат, формат А2/А3) и пояснительная записка.

Текст пояснительной записки оформляется в соответствии со стандартом предприятия СТП 01–2017.

1.2 Обоснование и описание выбора языка программирования, средств разработки, используемых технологий и сторонних библиотек

В качестве средства разработки я выбрал Unity. Это межплатформенная среда разработки компьютерных игр, которая позволяет создавать приложения, работающие под более чем 20 различными операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства и интернет-приложения.

Преимущества Unity:

* Кроссплатформенность
* Удобный и функциональный интерфейс
* Компонентно-ориентированный подход разработки
* Много встроенных библиотек, плагинов и поддерживаемых API
* Наличие отладчика
* Большое сообщество разработчиков

В Unity уже реализованы многие игровые модули: Физический,

звуковой, графический, менеджер ресурсов, пользовательский ввод и многие другие. Это позволяет быстро и качественно разрабатывать приложение, сосредотачиваться только на постройке игровой архитектуры и логике.

Язык программирования я выбрал исходя из средства разработки. В новых версиях Unity поддерживает только C#.

С# («Си Шарп») – один из наиболее быстро растущих, востребованных и при этом «удобных» языков программирования. Это модификация фундаментального языка С от компании Microsoft, призванная создать наиболее универсальное средство для разработки программного обеспечения для большого количества устройств и операционных систем.

Язык входит в семью С-подобных языков. Синтаксис приближен к Java и C++. Его особенности:

* статистическая типизация,
* поддерживается полиморфизм,
* поддерживается перегрузка операторов,
* доступна делегация, атрибуты, события, обобщенные типы и анонимные функции.

В рамках Unity язык C# отвечает за игровую логику. В Unity представлены базовые классы для взаимодействие с игровым окружением. В частности MonoBehavior, который имеет базовые события. Основные из них:

* Start() – Вызывается при старте сцены
* Update() – Вызывается при каждом обновлении кадра
* FixedUpdate() – Вызывается каждые N секунд, не зависит от частоты кадров

В качестве клиент-серверной части я использую библиотеку Photon Unity Networking. Она позволяет быстро и просто создавать клиент-серверное взаимодействие между игроками. Преимущества Photon Unity Networking:

* Простой и многофункциональный API
* Встроенный подбор оппонентов
* Понятная документация
* Условно бесплатный

# **2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

## **2.1 Постановка задачи**

Задачей курсового проекта является разработка программного средства, позволяющего пользователю играть в карточную игру “Дурак”.

Программное средство должно обладать следующими основными функциями, предоставляемые пользователю:

* создание игровой комнаты
* присоединение к игровой комнате
* смена игрового имени
* просмотр игровой статистики
* настройка звука
* смена языка приложения

**2.2 Проектирование архитектуры и описание состояний программного средства**

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки: классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

Основными элементами являются классы и связи между ними. Классы характеризуются при помощи атрибутов и операций.

Атрибуты описывают свойства объектов класса. Большинство объектов в классе получают свою индивидуальность из-за различий в их атрибутах и взаимосвязи с другими объектами. Однако, возможны объекты с идентичными значениями атрибутов и взаимосвязей.

Виды связей:

– ассоциация;

– агрегация;

– наследование.

Ассоциация (association) – представляет собой отношения между экземплярами классов. Каждый конец ассоциации обладает кратностью, которая показывает, сколько объектов может участвовать в данном отношении.

Агрегация (aggregation) – это ассоциация типа «целое-часть». Агрегация в UML представляется виде прямой с ромбом на конце.

Композиция (composition) – это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Композиция изображается так же, как ассоциация, только ромбик закрашен.

Наследование (inheritance) – это отношение типа «общее-частное». Позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой ряда других классов.

На рисунке 2.2.1 представлена диаграмма классов.

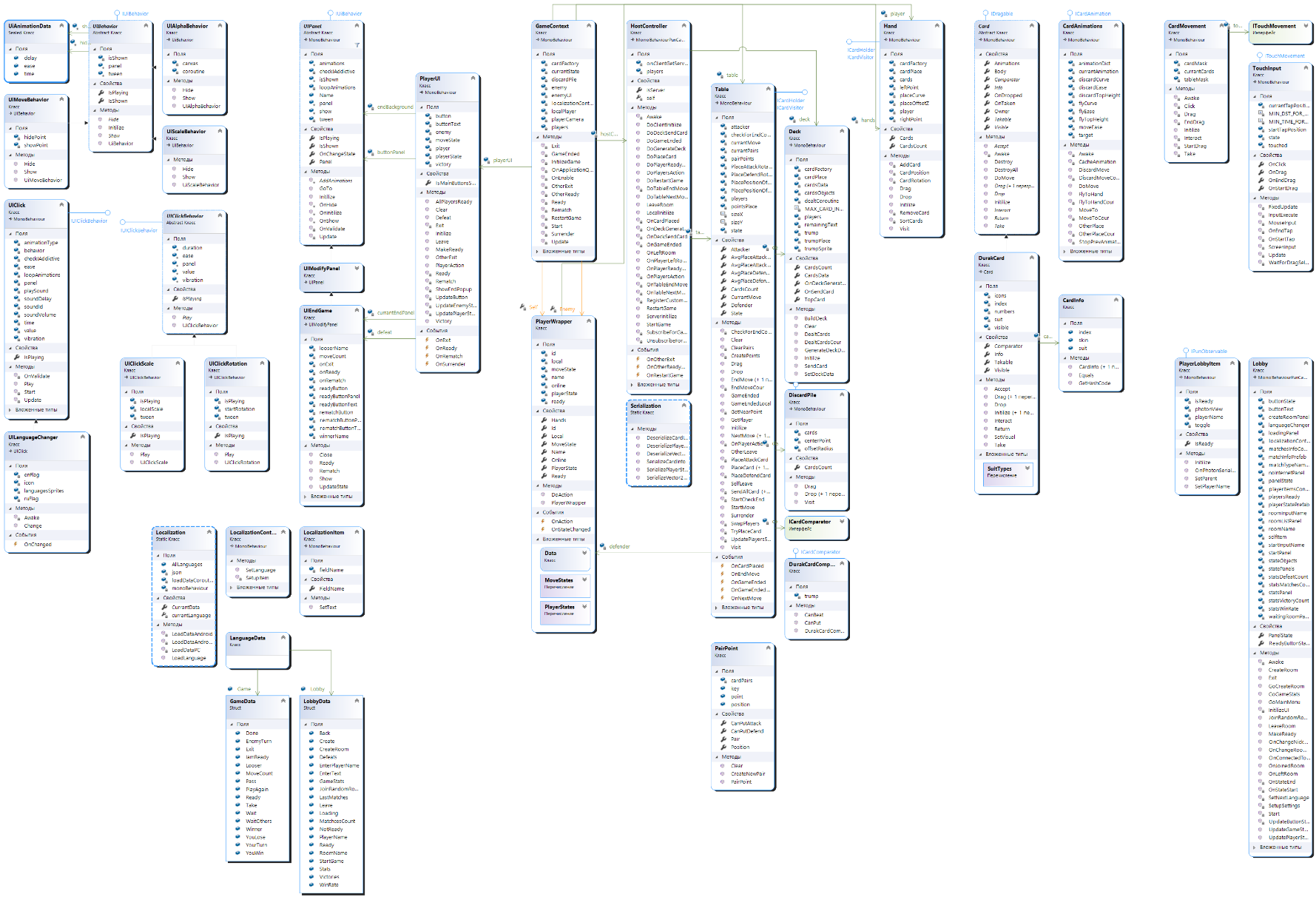


Рисунок 2.2.1 – Диаграмма классов

Диаграмма состояний – один из доступных видов диаграмм, поддерживаемых Flexberry Designer. Она показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события. Диаграмма состояний изображена на рисунке 2.2.2.

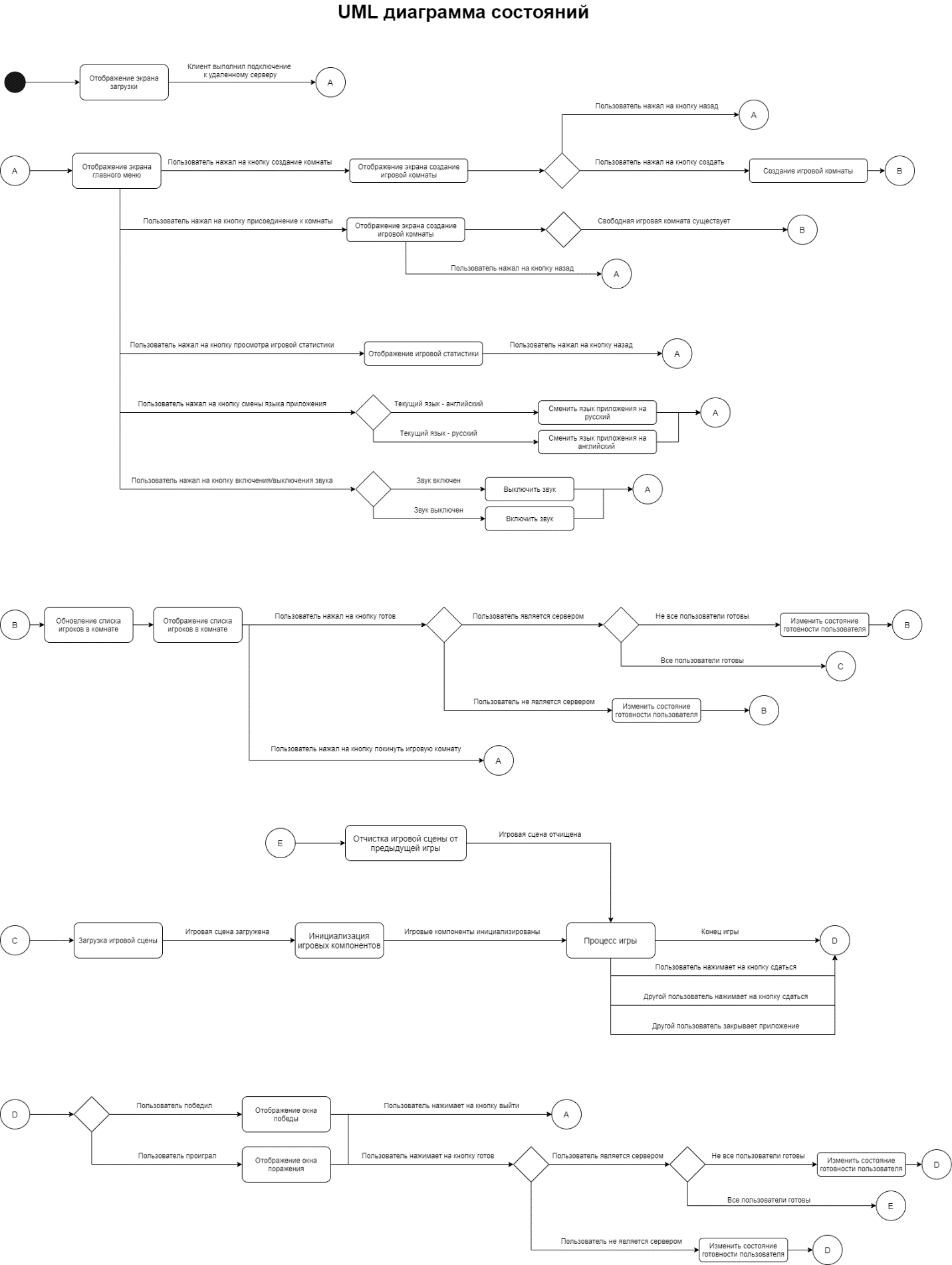


Рисунок 2.2.2 – UML диаграмма состояний

2.3 Проектирование и разработка графического интерфейса

GUI (Graphical User Interface, графический интерфейс пользователя) –

это форма пользовательского интерфейса , которая позволяет пользователям

взаимодействовать с приложением используя графические элементы.

Реализация загрузочного экрана на рисунке 2.3.1.



Рисунок 2.3.1 – загрузочный экран

Реализация главного меню на рисунке 2.3.2.

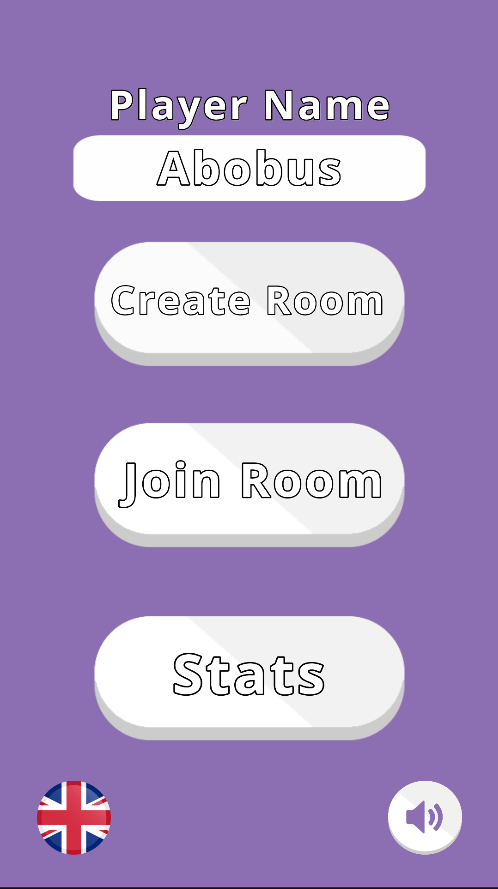


Рисунок 2.3.2 – главное меню

Реализация меню создания комнаты на рисунке 2.3.3.



Рисунок 2.3.3 – меню создания комнаты

Реализация меню присоединения к комнате на рисунке 2.3.4.



Рисунок 2.3.4 – меню присоединения к комнате

Реализация меню ожидания игроков на рисунке 2.3.5.

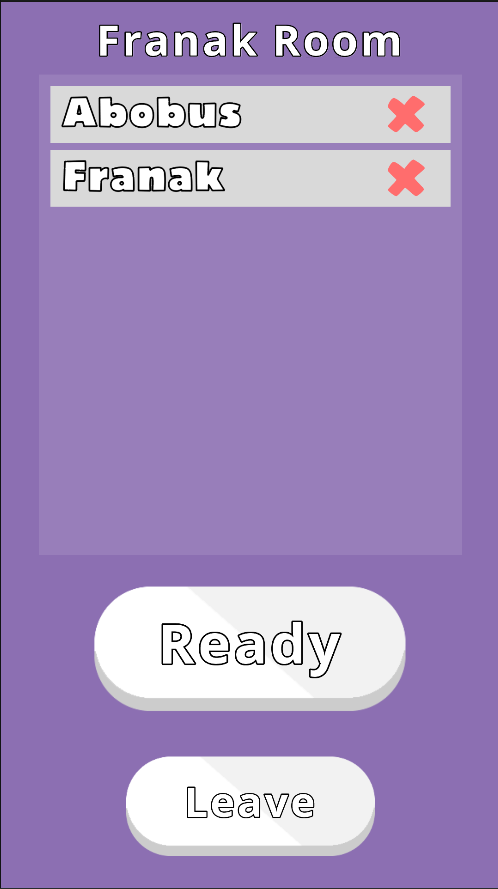


Рисунок 2.3.5 – меню ожидания игроков

Реализация меню игровой статичтики на рисунке 2.3.6.



Рисунок 2.3.6 – меню игровой статистики

Реализация игрового интерфейса на рисунке 2.3.7.

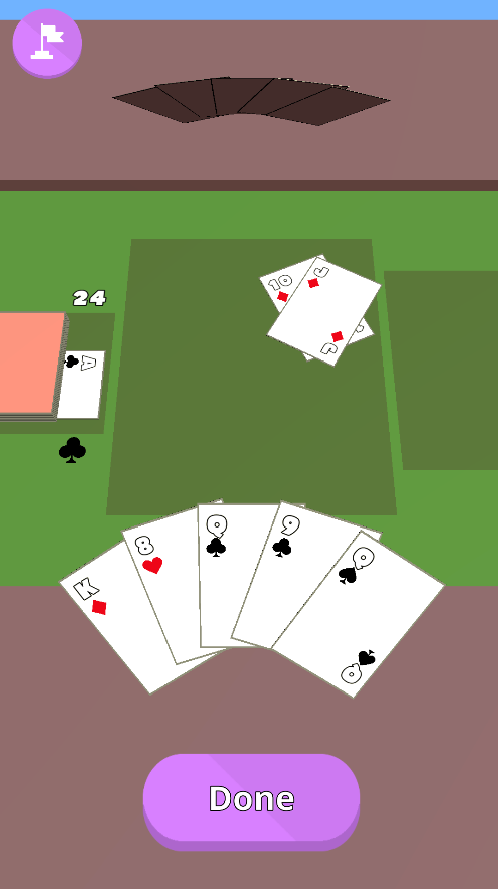


Рисунок 2.3.7 – игровой интерфейс

Реализация панели конца матча на рисунке 2.3.8.



Рисунок 2.3.8 – панель конца матча

**2.4 Описание и реализация используемых в программном средстве**

**алгоритмов.**

Схема алгоритма проверки конца матча показана на рисунке 2.4.1. Данный метод асинхронный, реализован с помощью Coroutine. Метод вызывается только на клиенте-сервере.

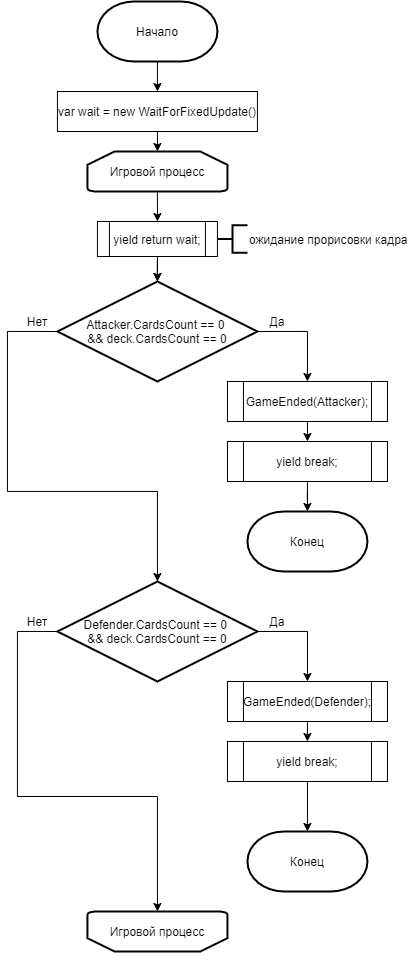


Рисунок 2.4.1 – блок-схема алгоритма проверки конца матча

Схема алгоритма раздачи кард показана на рисунке 2.4.2. Данный метод асинхронный, реализован с помощью Coroutine. Метод вызывается только на клиенте-сервере.

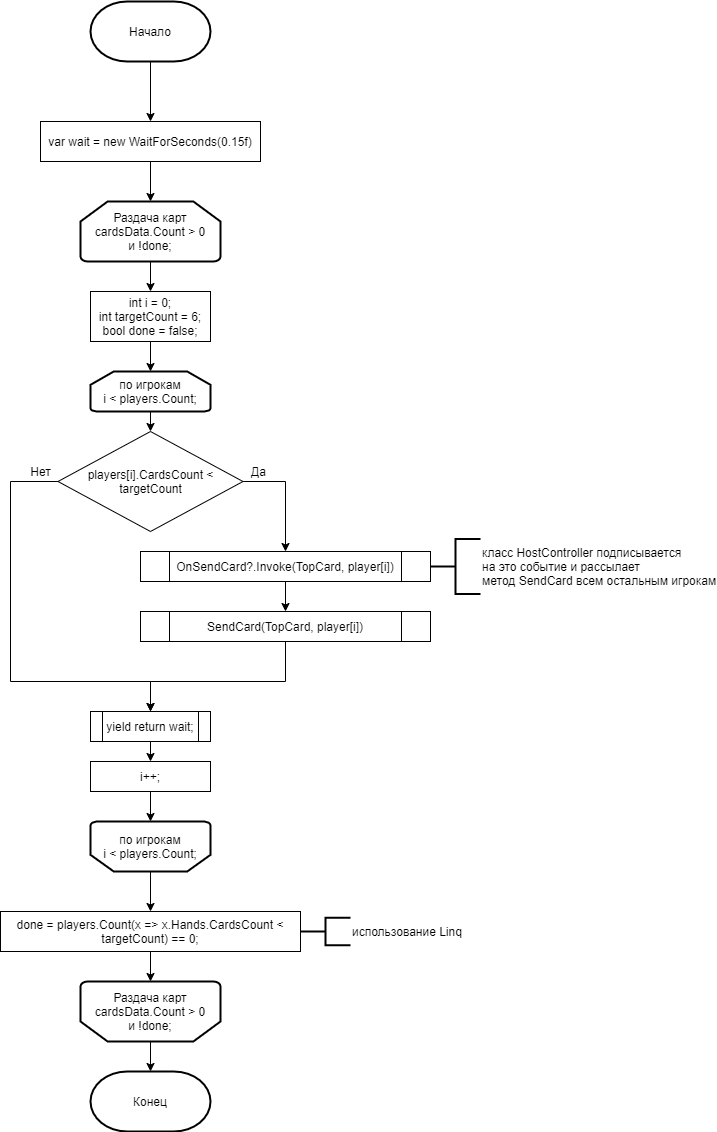


Рисунок 2.4.2 – блок-схема алгоритма проверки конца матча

3 ЭКСПЛУАТАЦИЯ ПРОГРАММНОГО СРЕДСТВА

**3.1 Ввод в эксплуатацию и обоснование минимальных технических требований к оборудованию**

Для распостранение игры среди пользователей, использующих мобильные устройства на операционной системе Android нужно собрать проект в .apk файл. При установке Unity, так же устанавливаются необходимые SDK, JDK и Gradle для сборки приложения под операционную систему Android.

При сборке проекта Unity сгенерирует дистрибутив приложения (.apk-файл) со всеми необходимыми библиотеками и сериализованными ассетами.

В операционной системе Android минимальные технические требования представлены двумя атрибутами: android:minSdkVersion и android:targetSdkVersion.

Android:minSdkVersion – это минимальное значение версии платформы Android, которое поддерживает приложение.

Android:targetSdkVersion – это значение версии платформы Android, для которого разработано приложение.

Минимальные требования к оборудовнию изображены в таблице 1.

Таблица 1 – минимальные требования к мобильному устройству

|  |  |
| --- | --- |
| **Требование** | **Значение** |
| Android:minSdkVersion | 19 |
| Android:targetSdkVersion | 29 |
| Свободная память | 51мб |

3.2 Ввод в эксплуатацию программного средства

После установки .apk файла, нужно запустить приложение “Card Game” после чего запуститься загзузочный экран представленный на рисунке 2.3.1.

Для работы приложение нужно подключение к интернету для взаимодействия с серверами Photon Networking.

После загрузки приложения игрок попадает в главное меню, представленное на рисунке 2.3.2, в котором может изменить свой игровой псевдоним, при первмо запуске будет сгенерировано случайное имя. Игрок может изменить язык приложения нажав на круг слева снизу, так же может включить/выключить звук приложения.

Для проведения матчка необходимо два устройства. Первый игрок нажимает на кнопку создания комнаты, затем указывает ее имя –индентификатор, по которому второй игрок сможет к ней присоединится (игрок создавший комнату является клинтом-сервером), после чего нажимает на кнопку создать комнату, изображенную на рисунке 2.3.3

Второй игрок нажимает на кнопку присоединения к комнате, вписывает нужное имя комнаты(имя комнаты сохраняется и при последующих присоединениях к комнате одно и тоже имя комнаты вписывать не придется) и нажимает на кнопку присоединиться, изображенной на рисунке 2.3.4

После того ка