Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Программирование на языках
высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему
«Игра Тривиадор»

БГУИР КП 1-40 02 01 301 ПЗ

Студент И. Я. Николаев

Руководитель Е. В. Богдан

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____
_____
(подпись)
2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Николаеву Ивану Ярославовичу*

Тема проекта *«Игра Тривиадор»*

2. Срок сдачи студентом законченного проекта *11 декабря 2023 г.*
3. Исходные данные к проекту: *Questions.json (текстовый файл сериализации структуры вопросов), res.qrc (файл ресурсов проекта с использованными изображениями).*
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)
1. Лист задания.
2. Введение.
3. Обзор литературы.
3.1. Обзор методов и алгоритмов решения поставленной задачи.
3.2. Разработка требований к функционалу.
4. Функциональное проектирование.
4.1. Структура входных и выходных данных.
4.2. Разработка диаграммы классов.
4.3. Описание классов.
5. Разработка программных модулей.
5.1. Разработка схем алгоритмов(два наиболее важных метода).
5.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух методов).
6. Результаты работы.
7. Заключение
8. Литература
9. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)
1. Диаграмма классов.
2. Схема алгоритма startCompetiton.
3. Схема алгоритма loadRandomQuestion.

6. Консультант по проекту (с обозначением разделов проекта) Е.В. Богдан

7. Дата выдачи задания *15 сентября 2023 г.*
8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):
1. Выбор задания. Разработка содержания пояснительной записки.
Перечень графического материала – 15 %;
разделы 2, 3 – 10 %;
разделы 4 к –20 %;
разделы 5 к – 35 %;
раздел 6,7,8 – 5 %;
раздел 9 к – 5%;
оформление пояснительной записки и графического материала к 11.12.23 - 10 %
Защита курсового проекта с 21.12 по 28.12.23г.

РУКОВОДИТЕЛЬ Е.В. Богдан

_____
(подпись)

Задание принял к исполнению_____Николаев И.Я.
(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В рамках моей курсовой работы был реализован проект игры «Тривиадор», основанный на языке программирования C++ и фреймворке Qt, с использованием JSON-файла для хранения ресурсов. Выбор этих технологий обусловлен стремлением к созданию эффективного и масштабируемого приложения.

В ходе разработки было решено воспользоваться языком программирования C++, что обусловлено его многочисленными преимуществами. C++ предоставляет высокоуровневые абстракции и одновременно обеспечивает близкую к металлу производительность, что является критическим фактором для эффективной работы приложений. Кроме того, C++ поддерживает парадигму объектно-ориентированного программирования (ООП), что позволяет создавать модульный и гибкий код.

Применение ООП в моем проекте обеспечило легкость сопровождения и масштабирования приложения. Используя принципы инкапсуляции, наследования и полиморфизма, я создал структуру, облегчающую добавление нового функционала и поддержание кодовой базы. Это способствовало улучшению общей архитектуры приложения и повышению его устойчивости.

Qt представляет собой мощный и гибкий инструментарий, спроектированный для создания кросс-платформенных приложений. Важными достоинствами Qt являются его обширная библиотека виджетов, интегрированная система управления событиями и высокий уровень абстракции, что позволяет разработчикам сосредоточиться на функциональности приложения, минимизируя заботы о различиях в операционных системах.

Таким образом, интеграция фреймворка Qt в разработке игры "Тривиадор" не только существенно улучшила пользовательский интерфейс, но и обеспечила надежные средства для управления ресурсами, что в целом способствовало созданию качественной и привлекательной игровой среды.

Для хранения текстовых данных в проекте я воспользовался форматом JSON, что предоставило мне удобство и легкость в управлении ресурсами. JSON файлы обеспечили структурированное хранение данных, что упростило процесс чтения и записи информации. Кроме того, формат JSON легко читаем и понятен человеку, что упрощает взаимодействие с данными в процессе разработки и отладки.

В итоге, сочетание языка программирования C++, фреймворка Qt и использование JSON файлов позволило мне создать эффективное, масштабируемое и легко поддерживаемое приложение, соответствующее поставленным требованиям.

Игра "Тривиадор" – увлекательная настольная стратегия, сочетающая в себе географический квест и вопросы на общеобразовательные темы. Ее преимущества включают образовательный компонент, способствующий расширению знаний о странах мира, географии и культурных фактах. В то же

время, игра предлагает стратегический геймплей, где игроки принимают решения о перемещении и захвате территорий, развивая свое стратегическое мышление. Таким образом, "Тривиадор" представляет собой многогранное развивающее развлечение, сочетая в себе образование и стратегию для увлекательного игрового опыта.

Выбор в качестве курсового проекта разработку игры обусловлен, возможность погружения в создание программного продукта, требующего реализации различных аспектов, таких как логика игры, пользовательский интерфейс и обработка данных.

Кроме того, выбор "Тривиадора" был обусловлен возможностью в полной мере продемонстрировать ранее изученный материал и освоить несколько новых для меня технологий.

1. ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Целью данной работы является овладение практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++. В рамках этой работы предлагается разработать игру «Тривиадор» с использованием среды разработки Qt. Поэтому можно определить следующие основные задачи: изучение основ C++ и изучение Qt и его компонентов.

Программа должна обеспечивать корректное выполнение игровой логики, взаимодействие игровых объектов и управление игровым процессом. Реализовать пользовательский интерфейс с необходимыми элементами управления, включая меню, экраны начала и завершения игры, а также систему уровней и достижений. Создать игровой контент, включая игровые элементы и графические ресурсы, соответствующие концепции игры "Тривиадор". Разработать систему взаимодействия игрока с игровым миром, включая обработку пользовательского ввода и реакцию на действия игрока. Реализовать механизм обработки исключительных ситуаций для обеспечения стабильной работы игры.

Игра "Тривиадор", имеет целый ряд положительных аспектов и применений. Во-первых, она способствует развитию интеллектуальных способностей игроков, поскольку требует решения различных логических задач и вопросов. Во-вторых, она может служить прекрасным средством для обучения и обогащения знаний в различных областях. Таким образом, данная игра объединяет образовательные и развлекательные аспекты, что делает ее полезной и интересной для широкой аудитории.

Овладение принципами объектно-ориентированного программирования (ООП) является неотъемлемой частью процесса разработки приложений, направленных на более гибкое и эффективное написание кода. Это обеспечивает не только улучшенную организацию кода, но и более глубокое понимание взаимодействия его компонентов.

Особое внимание уделяется принципу наследования, который играет ключевую роль в выбранной среде разработки. Надежное владение принципами наследования необходимо для свободного расширения классов и их изменения, что существенно способствует более эффективной работе в среде Qt. Глубокое понимание этого принципа обеспечивает уверенность в создании структурированного и легко модифицируемого кода.

Создание графического пользовательского интерфейса в среде Qt представляет собой важный этап, и овладение данной средой позволяет эффективно использовать широкий спектр компонентов для разработки привлекательных и функциональных приложений. Освоение Qt не только упрощает процесс создания графического интерфейса, но и значительно повышает производительность и качество разрабатываемых приложений.

Использование JSON-формата для структурирования данных при разработке игры "Тривиадор" представляет собой неотъемлемую часть эффективного управления ресурсами и обеспечивает легкость в обработке текстовых данных. Гибкость и простота JSON делают его отличным выбором для хранения информации о вопросах, ответах и других игровых элементах.

Таким образом, овладение принципами работы с JSON-форматом обеспечивает не только структурированность данных, но и удобство в поддержке и обновлении игровых ресурсов в процессе разработки игры.

Для реализации программы используется объектно-ориентированный язык программирования C++, интегрированная среда разработки Clion и QtCreator. Приложение написано на ОС Windows 11.

2. ОБЗОР ЛИТЕРАТУРЫ

2.1 Обзор методов и алгоритмов решения поставленной задачи

Выбор языка программирования C++ и фреймворка Qt для создания приложения отслеживания задач обусловлен стремлением использовать основные принципы объектно-ориентированного программирования (ООП) и обеспечить совместимость между кодом и графическим интерфейсом. Для выполнения данной задачи была изучена доступная о игре информация.

2.1.1 Использование языка C++

Выбор языка программирования C++ для написания кода приложения отслеживания задач обоснован рядом фундаментальных аспектов:

- Обладает полной поддержкой принципов ООП, таких как инкапсуляция, наследование и полиморфизм. Это позволяет создавать код, ориентированный на объекты, что в свою очередь способствует легкости в создании модульного и структурированного программного кода;
- Позволяет разрабатывать приложение в виде отдельных модулей, что обеспечивает легкость в поддержке и модификации кода. Модульность позволяет разделить приложение на небольшие, самодостаточные блоки, что упрощает понимание и обеспечивает структурированность кодовой базы;
- Применение принципов ООП и модульной структуры кода в C++ способствует повышению надежности приложения. Четкая структура кода упрощает процесс сопровождения, а изменения в коде одного модуля могут быть внесены с минимальными воздействиями на остальные части приложения;
- Подробная документация, множество онлайн-ресурсов, литературы предоставляют необходимые ресурсы и информацию в процессе разработки [1][2][3].

2.1.2 Использование фреймворка Qt

Qt представляет собой мощный фреймворк для разработки приложений на C++, предоставляя обширный набор инструментов и библиотек для создания графических интерфейсов, обработки событий, работы с сетью, базами данных и других задач. Использование фреймворка Qt на C++ при создании пользовательского графического интерфейса обеспечивает не только высокую производительность, но и удобство в использовании Qt Creator — интегрированной среды разработки, представляющей инструменты для эффективной работы с Qt и ускорения процесса создания приложений. При

разработке была использована официальная документация Qt [4] и мануал по использованию Qt creator [5].

Использование фреймворка Qt на C++ для создания пользовательского графического интерфейса имеет ряд преимуществ:

- Кросс-платформенность: Qt обеспечивает возможность создания кросс-платформенных приложений, что позволяет запускать один и тот же код на различных операционных системах без значительных изменений;
- Обширный функционал: Фреймворк предоставляет богатый набор инструментов и библиотек для создания графических интерфейсов, обработки событий, работы с сетью, базами данных и других распространенных задач, упрощая разработку сложных приложений;
- Qt Creator: Интегрированная среда разработки Qt Creator предоставляет удобный интерфейс и множество инструментов для эффективной работы с Qt, упрощая процесс разработки и отладки;
- Сигналы и слоты: Механизм сигналов и слотов в Qt обеспечивает эффективное взаимодействие между различными частями приложения. Это делает код более модульным и облегчает его поддержку и модификацию;
- Активное сообщество: На просторах интернете можно найти огромное множество проектов написанных с использованием фреймворка Qt, что может эффективно помочь в решении конкретных технических задач и улучшении навыков программирования.

2.1.3 Правила игры «Тривиадор»

Этот раздел представляет описание правил игры. Важно отметить, что на момент данной работы конкретные правила игры «Тривиадор» не представлены в доступных источниках или документации. Отсутствие формального описания правил оставляет место для предположений и интерпретации механик игры. Несмотря на это, в анализе данного проекта возможно предоставление обзора основных компонентов, предполагаемых механик и элементов игры или же их адаптация под данную задачу:

- Цель игры: Игрок должен захватить как можно больше полей противника, чтобы достичь нужного количества очков для победы;
- Компоненты игры: Территории игроков, которые представляют набор полей разного размера и количества вопросов, которые в них содержатся;
- Начало игры: У игроков одинаковое количество полей всех размеров, территории отличаются цветом. Игрок, который ходит первым, определяется случайным образом. Игроки выбирают категории вопросов для игры;
- Ход игры: Игроки поочередно захватывают поля противника путем ответа на вопросы из выбранных категорий. За каждый правильный

ответ или череду правильных ответов игрок получает очки. В случае захвата поля, его цвет изменяется на цвет полей игрока, захватившего это поле;

- Вопросы и ответы: При захвате поле, игрокам нужно ответить на случайный вопрос из выбранной категории. В случае, если поле содержит более одного вопроса и игрок не смог ответь на все из них, то очки за правильные ответы к предыдущим вопросам этого поля сгорают;
- Завершение игры: Достигается, если один из игроков набрал нужное количество очков.

При изучении возможных правил была использована информация, полученная из электронного ресурса [6].

2.2 Разработка требований к функционалу

При разработке игры «Тривиадор» уделено внимание разнообразному функционалу. Это включает игровой интерфейс, правила и разнообразные вопросы, гарантируя увлекательный опыт. Таким образом можно составить основные требования к функционалу приложения:

1. Графический дизайн: Разработка пользовательского интерфейса, соответствующего тематике игры «Тривиадор»;
2. Простота использования: Обеспечение удобства взаимодействия с приложением для пользователей, включая интуитивное управление и навигацию;
3. Вопросы и ответы: Реализация разнообразных и интересных вопросов с ответами, соответствующих теме игры. Выбор категории вопросов;
4. Логика игры: Разработка четких правил и механик игры, учитывающих соревновательный характер и поддерживающих взаимодействие между игроками.

3 ФУНКЦИАНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В этом разделе представлены входные и выходные данные программы, диаграмма классов, а также предоставлено описание классов и их методов, используемых в проекте.

3.1 Структура входных и выходных данных

Приложение работает с двумя основными типами входных данных:

- Данные пользователя: включают информацию, которую пользователь вводит через графический интерфейс (GUI). Эти данные охватывают выбор игроком начало либо завершение соревнования, выбор категории вопросов;
- Информация о вопросах: это данные, которые приложение получает из файла Questions.json. Включают в себя информацию о категории вопросов, содержании вопроса, вариантов ответа, информация о правильном ответе;

Приложение генерирует следующие выходные данные:

- Сигналы: приложение также генерирует сигналы как часть механизма сигналов и слотов Qt. Сигналы используются для обработки событий, таких как нажатие кнопок пользователем или захвата игроком полей противника;
- Данные GUI: это данные, которые отображаются пользователю через GUI. Они включают в себя информацию о корректности ответа.

3.2 Разработка диаграммы классов

Диаграмма классов – это инструмент в языке моделирования UML, предназначенный для визуализации структуры классов в системе, их атрибутов, методов, интерфейсов и взаимосвязей между ними. Часто используется при проектировании архитектуры, документировании системы, уточнении требований и поддержке системы.

Она наглядно отображает статическое представление системы, включая различные аспекты приложения. Диаграмма состоит из блоков, представляющих классы, с тремя основными компонентами:

- В верхней части указывается имя класса, выравнивается по центру и использует полужирный шрифт;
- В средней части перечисляются атрибуты (поля) класса;
- В нижней части перечисляются методы класса.

Такая диаграмма включает в себя не только классы, но и отношения между ними, такие как наследование, агрегация, ассоциация, композиция и другие, что делает ее мощным инструментом для понимания структуры и взаимодействия в системе.

Диаграмма классов данной работы показана в приложении А

3.3 Описание классов

В данном разделе описаны основные классы используемые в ходе работы приложения. Приведены атрибуты, методы каждого класса и их описание.

3.3.1 Класс `MainWindow`

Является основным классом приложения из которого идет обращение ко всем остальным виджетам. В нем описаны основные методы и переменные используемые в ходе работы приложения:

Поля класса `MainWindow`:

`ui` – Это интерфейс пользователя, созданный с использованием Qt Designer. Он представляет собой графический интерфейс, который пользователь видит и с которым взаимодействует.

`btnPlayerSmall[4]` – Это массив указателей на виджет самого маленького поля игрока.

`btnPlayerMedium` – Это указатель на виджет среднего поля игрока.

`btnPlayerLarge` – Это указатель на виджет самого большого поля игрока.

`player` – Это объект класса `Player`, инициализирующийся в рассматриваемом классе.

`competition` – Это объект класса `Competition`, инициализирующийся в рассматриваемом классе.

`actionGeography` – Это объект запроса к полю выбора категории `Geography`.

`actionHistory` – Это объект запроса к полю выбора категории `History`.

`actionMath` – Это объект запроса к полю выбора категории `Math`.

Методы:

`startGame()` – Это обработчик события нажатия кнопки `Start`, находящейся в `MenuBar`. Она начинает соревнование.

`overGame()` – Это обработчик события нажатия кнопки `Start`, находящейся в `MenuBar`. Она завершает соревнование и закрывает программу.

`toggleCategoryGeography(bool checked)` – Это обработчик нажатия на `CheckBox` внутри `MenuBar`. При включенном `CheckBox` добавляется категория `Geography`.

`toggleCategoryHistory(bool checked)` – Это обработчик нажатия на `CheckBox` внутри `MenuBar`. При включенном `CheckBox` добавляется категория `History`.

`toggleCategoryMath(bool checked)` – Это обработчик нажатия на `CheckBox` внутри `MenuBar`. При включенном `CheckBox` добавляется категория `Math`.

3.3.2 Класс `Competition`

Класс `Competition` представляет собой ключевой компонент, ответственный за реализацию игровой логики. Этот класс играет решающую роль в управлении и координации основных элементов соревнования. В его атрибутах и методах отражены основные аспекты взаимодействия игроков, обработка вопросов, расчет результатов и поддержание состояния игры.

Поля класса `Competition`:

`startingPlayer` – Это объект класса `Player`, отвечает за игрока, которой начал соревнование.

`opponentPlayer` – Это объект класса `Player`, отвечает за противника.

`categories` – Это вектор, содержащий информацию о включенных категориях.

`currentPlayerIndex` – Это переменная, содержащая информацию о том какой игрок ходит.

Методы:

`startCompetition()` – Этот метод начинает соревнование.

`overCompetition()` – Этот метод завершает соревнование.

`switchPlayer()` – Этот метод передает ход другому игроку.

`checkGameOver()` – Этот метод проверяет состояние игры, в случае завершения возвращает `true`.

3.3.3 Класс `QuestionLoader`

Класс `QuestionLoader` отвечает за инициализацию вопроса, путем загрузки данных из файла `Question.json`.

Поля класса:

`category` – Это строка, которая содержит в себе название категории вопроса.

`text` – Это строка, которая содержит в себе текст вопроса.

`answers` – Это контейнер, в котором содержатся строки хранящие в себе текст вариантов ответа.

`correctAnswerIndex` – Это переменная которая хранит в себе индекс правильного ответа.

Методы:

`readJsonFile()` – Этот метод загружает данные из JSON файла передает их программе.

`parseQuestions(const QJsonObject& jsonObject)` – Этот метод принимает данные из JSON файла и возвращает вектор со всеми вопросами содержащимися в файле.

`loadRandomQuestion(const QVector<int>& categories)` – Этот метод возвращает случайный вопрос исходя из выбранных игроком категорий.

3.3.4 Класс `Player`

Класс `Player` отвечает за поведение игрока.

Поля класса:

`score` – Это переменная, которая хранит в себе счет игрока.

`fields` – Это вектор, который содержит в себе поля принадлежащие игроку.

`fieldClicked` – Это переменная, которая содержит информацию о состоянии поля игрока.

Методы:

`onFieldClicked()` – Это метод, который посылает сигнал, если нажато на поле игрока.

`connectFieldSignals()` – Этот метод соединяет кнопки из интерфейса приложения с полями игрока.

`disconnectFieldSignals()` – Этот метод соединяет кнопки из интерфейса приложения с полями игрока.

`addScore(int points)` – Этот метод отвечает за добавление очков к конечному счету игрока. Принимает в себя количество очков, которыми нужно пополнить счет игрока.

`captureField()` – Этот метод отвечает за логику захвата игроком поля противника.

3.3.5 Класс `Field`

Класс `Field` отвечает за логику полей игроков.

Поля класса:

`button` – Это переменная, в которой храниться указатель на кнопку к которой привязано поле.

`questionCount` – Это переменная, которая хранит в себе количество вопросов, которые будут предложены игроку при нажатии.

Методы:

`setButton()` – Этот метод привязывает кнопку из интерфейса к полю.

`getQuestionCount()` – Этот метод возвращает количество вопросов, которые будут предложены игроку после нажатия на поле.

`changeColor()` – Этот метод меняет цвет поля на цвет соответствующий тому игроку, который его захватил.

`onButtonClicked()` – Это обработчик события нажатия на кнопку в интерфейсе.

Класс `Field` является базовым для таких классов как `SmallField`, `MediumField` и `LargeField`.

3.3.6 Класс `QuestionWindow`

Класс `QuestionWindow` отвечает за окно вопроса, демонстрирует пользователю текст вопроса, варианты ответов, дает возможность ответить на вопрос.

Поля класса:

`ui` – Это интерфейс пользователя, созданный с использованием Qt Designer. Он представляет собой графический интерфейс, который пользователь видит и с которым взаимодействует.

`currentPlayer` – Это переменная, которая содержит указатель на текущего игрока.

`correctAnswerIndex` – Это переменная, которая содержит индекс правильного варианта ответа.

Методы:

`setQuestion(const QString& question, const QVector<QString>& answers, int correctAnswerIndex, Player* currentPlayer)` – Этот метод отвечает за загрузку данных полученных из файла в окно вопроса. Он принимает в себя данные о вопросе и указатель на текущего игрока.

`onConfirmButtonClicked()` – Этот метод сигнализирует о том, что была нажата кнопка подтверждения ответа.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов

Метод `startCompetition` отвечает за начало соревнования и реализует главный игровой цикл. Схема метода `startCompetition` представлена в приложении Б.

Метод `loadRandomQuestion` отвечает за загрузку случайного вопроса исходя из выбранных игроком категорий. Схема метода `loadRandomQuestion` представлена в приложении В.

4.2 Разработка алгоритмов

Разработка алгоритмов – это процесс создания последовательности шагов или набора инструкций для решения конкретной задачи или выполнения определенной операции. Она представляет собой ключевой этап в программировании и информатике, где алгоритмы играют важную роль в оптимизации и эффективной реализации программных решений.

Приведенные выше методы могут быть систематизированы в виде последовательного алгоритма, который представляет основные этапы выполнения соответствующих функций в программе.

4.2.1 Алгоритм метода `startCompetition`

Шаг 1. Вызывается метод Qt `emit competitionStarted()`, который отправляет сигнал остальной части приложения о начале соревнования. Это нужно для того, чтобы пользователю можно было ограничить или расширить функционал приложения при вызове метода.

Шаг 2. Вызывается обработчик исключительной ситуации. Если до исполнения метода не были инициализированы игроки, то мы прекращаем его исполнение.

Шаг 3. Инициируем главный игровой цикл. В нем заключена основная логика соревнования.

Шаг 4. Генерируем случайный индекс `categoryIndex`, который представляет из себя номер категории из выбранных игроком.

Шаг 5. Инициализируем начального игрока, который является объектом класса Player. Присваиваем ему указатель на одно из игроков.

Шаг 6. Проверяем нажал ли игрок на поле противника. В случае нажатия, создаем окно вопроса.

Шаг 7. В случае успешного создания окна вопроса, при помощи метода `setQuestion()` загружаем случайный вопрос из выбранных категорий.

Шаг 8. Добавляем вопрос в окно вопроса и показываем его игроку.

Шаг 9. После ответа игрока, закрываем окно вопроса и вызываем диалоговое окно, в котором содержится информация о корректности выбора игрока и текущем счете.

Шаг 10. В случае если игрок ответил неправильно вызываем метод `switchPlayer()`, который передает ход противнику.

Шаг 11. Вызываем метод `checkCompetitionCondition()`, который проверяет условие победы и возвращает true, если один из игроков победил. В таком случае завершаем цикл.

Шаг 12. Вызываем метод `overCompetition()`, который в свою очередь вызывает метод Qt `emit competitionOver()`, что сигнализирует остальной части программы о завершении соревнования.

Шаг 13. Вызываем диалоговое окно, которое информирует игрока о победе.

4.2.2 Алгоритм метода `loadRandomQuestion`

Шаг 1. Открываем файл `Questions.json` в режиме чтения.

Шаг 2. Если не удается открыть файл возвращаем пустой `QJsonObject`.

Шаг 3. Преобразуем содержимое файла в объект `QJsonDocument`.

Шаг 4. Извлекаем массив вопросов из объекта `JSON`.

Шаг 5. Для каждой категории в массиве: проверяем наличие ключей `category` и `questions`.

Шаг 6. Для каждого вопроса проверяем наличие ключей `text`, `correct_answer` и `answers`.

Шаг 7. Для валидных вопросов: извлекаем текст вопроса, массив ответов и номер правильного ответа.

Шаг 8. Создаем объект типа `Question` и добавляем его в вектор вопросов.

Шаг 9. Создаем пустой вектор `validQuestions` для хранения вопросов, соответствующих выбранным категориям.

Шаг 10. Для каждого вопроса из вектора вопросов проверяем принадлежит ли он к выбранным категориям.

Шаг 11. Добавляем валидные вопросы в вектор `validQuestions`.

Шаг 12. В пределах размера вектора `validQuestions` генерируем случайный индекс.

Шаг 13. Возвращаем выбранный случайным образом вопрос.

Шаг 14. Если вектор `validQuestions` оказывается пустым, выводим предупреждение.

5. РЕЗУЛЬТАТ РАБОТЫ

При запуске программы игроков встречает основное окно. На основном окне располагаются игровое поле и виджет MenuBar. Игровое поле представляет из себя множество виджетов, которые в свою очередь представляют из себя территории(поля) игроков. Соответствующее по значимости поле обозначено уникальным изображением. Виджет Menu или MenuBar расположен в верхнем левом углу окна. На рисунке 5.1 показано основное окно игры.

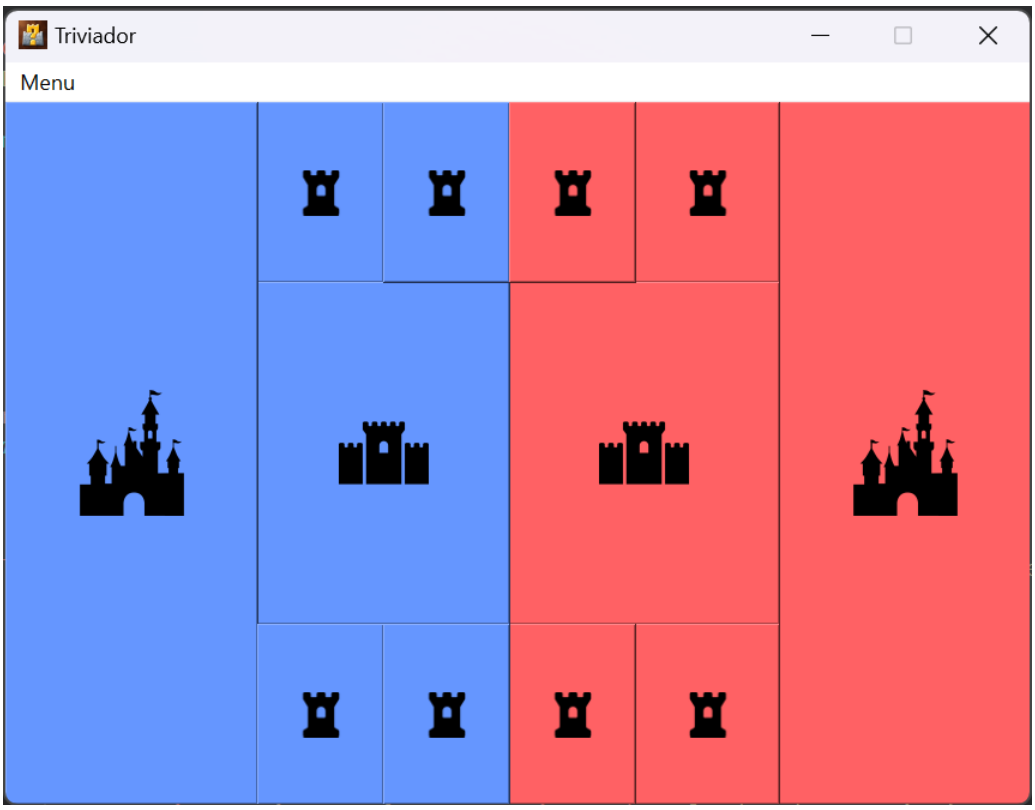

Рисунок 5.1. – Основное окно игры.

Остановимся подробнее на MenuBar. В меню есть две основные вкладки: Game и Category. В Game есть две опции: Start и Over. Опция Start отвечает за начало игры. После выбора данной опции функционал игроков расширяется, т.е. они могут захватывать территории противника. Опция Over отвечает за завершение игры. После выбора данной опции игра завершается и происходит выход из программы. Во вкладке Category есть три опции History, Geography, Math. Эти опции представляют из себя CheckBox, при нажатии на который включается соответствующая категория. Структура Menubar представлена на рисунках 5.2 и 5.3.

При захвате игроком территории противника ему нужно ответить на вопрос из выбранной категории. Взаимодействия игрока с вопросом представлено в окне вопроса. Окно вопроса представляет содержит в себе

текст вопроса, кнопки опций ответа и кнопку подтверждения. Окно вопроса показано на рисунке 5.4.

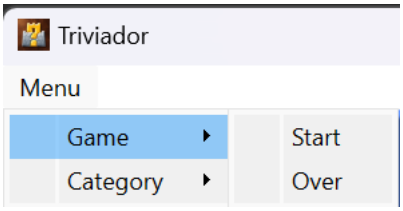

Рисунок 5.2. – Вкладка Game в MenuBar.

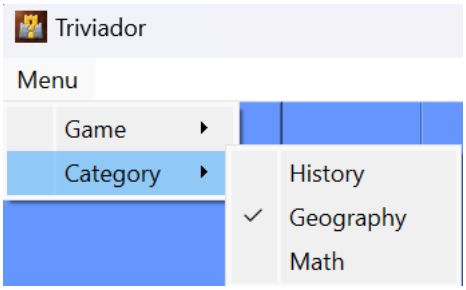

Рисунок 5.3. - Вкладка Category в Menubar.

В случае правильного или неправильного ответа игроку выводится диалоговое окно, которое информирует о корректности ответа и текущем счете игрока. Диалоговое окно информации представлено на рисунке 5.5.

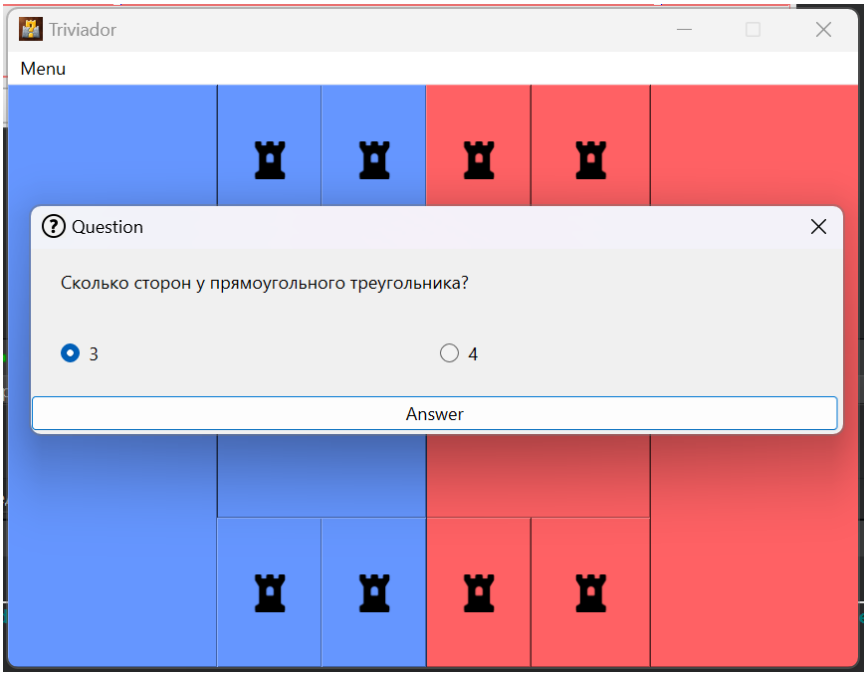

Рисунок 5.4. – Окно вопроса.

Если игрок ответил правильно ответил на все вопросы предложенные полем противника, то он захватывает это поле. В графическом интерфейсе это представлено как закрашивание данного поля цветом игрока. Демонстрация захвата поля игроком приведена на рисунке 5.6.

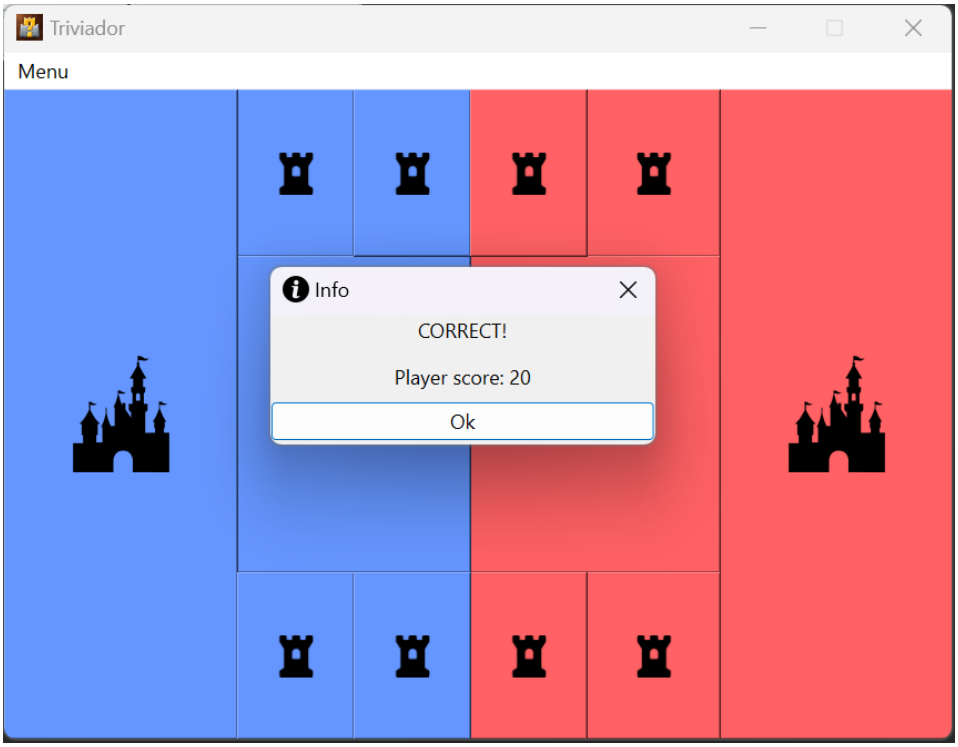

Рисунок 5.5. – Диалоговое окно информации.

При захвате определенного количества полей противника, т.е. при достижении определенного количества очков игра завершается и выводится окно в котором отображается информация о том какой игрок выиграл, о конечном результате игроков. Демонстрация окна информирования игрока о результатах игры представлена на рисунке 5.7.

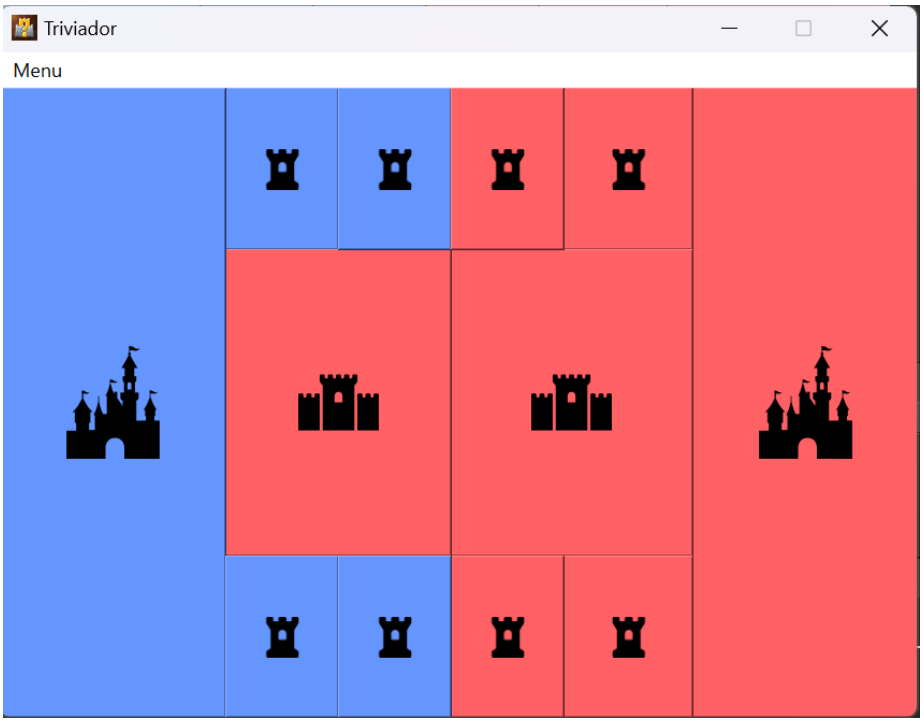

Рисунок 5.6 – Демонстрация захвата поля.

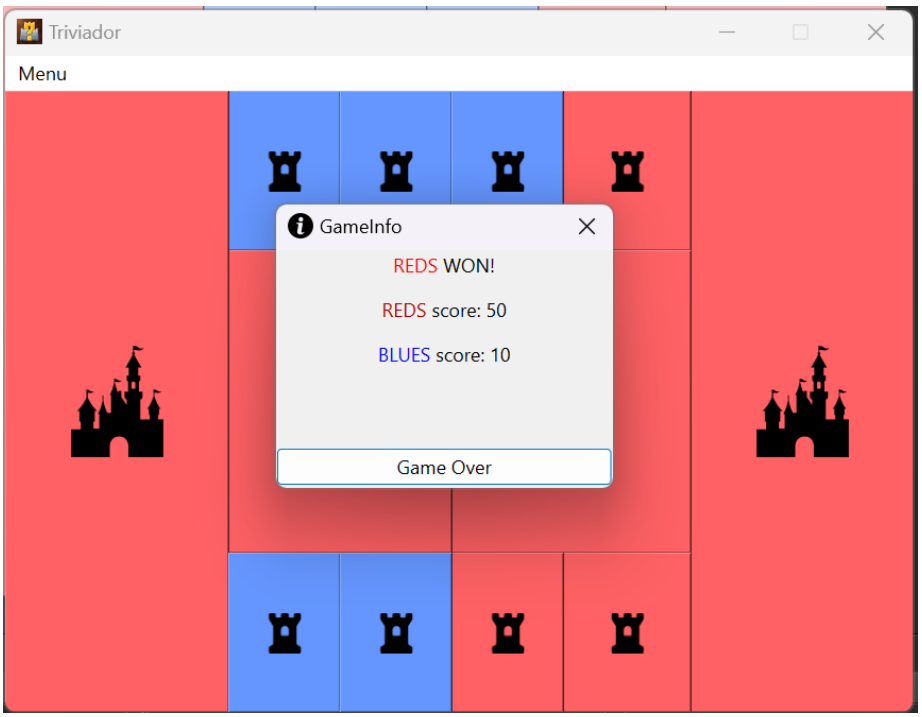

Рисунок 5.7 – Окно информирования игрока о результатах финала.

Стоит уделить внимание результатам разработки ресурсного файла Questions.json. Файл разделен на категории в которых содержатся вопросы с соответствующими им вариантами ответа. При расширении проекта, не составит труда добавить новые категории вопросы и т.д. без изменения исходного кода. Файл легко понятен как человеку, так и программе. Структура файла Questions.json приведена на рисунке 5.8.

```
{
  "questions": [
    {
      "category": "Название категории",
      "questions": [
        {
          "text": "Текст вопроса",
          "answers": ["Вариант1", "Вариант2"],
          "correct_answer": "Индекс правильного ответа"
        }
      ]
    }
  ]
}
```

Рисунок 5.8 – Структура файла Questions.json.

В заключении раздела можно сказать, что был продемонстрирован основной функционал игры, обеспечивающий интерактивное и увлекательное взаимодействие с пользователем. Демонстрация базовых функций игры не только позволила продемонстрировать применение изученных концепций программирования, но и подчеркнула эффективность выбранных подходов в создании интерактивного программного продукта. Выявлены потенциальные направления улучшения и расширения функционала игры, что включает в себя возможности улучшения пользовательского добавления дополнительного игрового контента.

Помимо вышеперечисленного, стоит отметить, что была проведена отладка игры, были выявлены и устранены возможные исключения.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана игра «Тривиадор» с применением фреймворка Qt. Игра предоставляет игрокам уникальный и образовательный опыт, удобный и интуитивно понятный интерфейс.

Парадигма объектно-ориентированного программирования, которую предоставляет язык C++, была использована в разработке игры «Тривиадор» существенно улучшив ее архитектуру и поддерживаемость. Реализовав основные принципы ООП, такие как инкапсуляция, наследование и полиморфизм, я создал модульную и гибкую структуру кода. Таким образом, применение ООП не только повысило читаемость кода, но и способствовало созданию масштабируемого и структурированного программного продукта, реализации функциональных требований.

Для разработки GUI был выбран фреймворк Qt, который существенно облегчил разработку функциональной части игры.

Освоение работы с форматом JSON в проекте демонстрирует гибкость подхода к управлению ресурсами и структурированию данных. Это также обеспечивает легкость в реализации будущих дополнений, что важно для дальнейшего развития игры.

Игра, разработанная в ходе курсового проектирования предоставляет пользователю не только способ развлечения, но образовательный элемент. Возможности улучшения, такие как расширение категорий вопросов и улучшение интерфейса, открывают перспективы для дальнейшего развития проекта и увеличения его привлекательности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Страуструп, Б. Язык программирования C++ / Б. Страуструп. –М. : БИНОМ, 2004.- 1098 с.

[2] Справочник по языку C++ с обширной документацией и примерами кода [Электронный ресурс]. –Режим доступа: https://en.cppreference.com/w/. – Дата доступа: 15.10.2023

[3] Object-Oriented Programming in C++: A Comprehensive Guide [Электронный ресурс]. –Режим доступа: https://wiingy.com/learn/cpp/cpp-object-oriented-programming-guide/. –Дата доступа: 18.10.2023

[4] Официальная документация Qt [Электронный ресурс]. –Режим доступа: https://doc.qt.io/. –Дата доступа: 18.10.2023

[5] Официальный мануал Qt Creator [Электронный ресурс]. –Режим доступа: https://doc.qt.io/qtcreator/. –Дата доступа: 17.10.2023

[6] Правила игры «Тривиадор» [Электронный ресурс]. –Режим доступа: https://wisegeek.ru/aae/triviador. –Дата доступа: 22.10.2023

ПРИЛОЖЕНИЕ А
(обязательное)
Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема метода `startCompetition`

ПРИЛОЖЕНИЕ В

(обязательное)

Схема метода `loadRandomQuestion`

ПРИЛОЖЕНИЕ Г
(обязательное)
Код Программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов