Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему
ИГРА “Палитра”
БГУИР КР 1-40 02 01 323 ПЗ

Студент: Шандраков С. А.

Руководитель: Байдун Д. Р.

Минск 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

C++ – компилируемый, статически типизированный язык программирования. Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование и обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. **C++** широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие.

Язык ассемблера — машинно-ориентированный язык программирования низкого уровня. Представляет собой систему обозначений, используемую для представления в удобно читаемой форме программ, записанных в машинном коду. Его команды прямо соответствуют отдельным командам машины или их последовательностям. Является существенно платформо-зависимым: языки ассемблера для различных аппаратных платформ несовместимы, хотя могут быть в целом подобны.

Язык ассемблера позволяет программисту пользоваться алфавитными мнемоническими кодами операций, по своему усмотрению присваивать символические имена регистрам ЭВМ и памяти, а также задавать удобные для себя схемы адресации (например, индексную или косвенную).

1 Обзор методов и алгоритмов поставленной задачи

Для создания этого программного обеспечения был выбрана технология Qt Creator. Qt Creator (ранее известная под кодовым названием Greenhouse) – кроссплатформенная свободная IDE для разработки на C,C++, JavaScript и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML.

Qt creator позволяет подключать разные библиотеки для соответствующих целей. Для работы с игрой были выбраны следующие библиотеки: QApplication, QMainWindow, QWidget, QPushButton, QPixmap, QIcon, QTimer, QLabel, QGraphicsScene, QGraphicsView, QGraphicsPixmapItem, QKeyEvent.

- QApplication содержит основной цикл событий, в котором все события от оконной системы и других источников обрабатываются и координируются. Он также обрабатывает инициализацию и завершение приложения и обеспечивает управление сессиями. Кроме того, QApplication обрабатывает большинство общесистемных и общепрограммных настроек.
- Класс QMainWindow предоставляет главное окно приложения и структуру для создания пользовательского интерфейса приложения. Qt имеет класс QMainWindow и связанные с ним классы для управления главным окном.
- Класс QWidget является базовым для всех объектов пользовательского интерфейса. Виджет - это элементарный объект пользовательского интерфейса: он получает события мыши, клавиатуры и другие события от оконной системы и рисует свое изображение на экране. Каждый виджет имеет прямоугольную форму, и все они отсортированы в порядке наложения (Z-order). Виджет ограничен своим родителем и другими виджетами, расположенными перед ним.
- Виджет QPushButton представляет собой командную кнопку. Кнопка, или командная кнопка, является, по всей видимости, наиболее часто используемым виджетом в любом графическом пользовательском интерфейсе.
- Класс QPixmap – представление изображения вне экрана, которое можно использовать в качестве устройства рисования.
- QIcon – может генерировать меньшие, большие, активные и отключённые растровые изображения из заданного ему набора растровых изображений. Такие растровые изображения используются виджетами Qt для отображения значка, представляющего конкретное действие.

- Класс QTimer – предоставляет интерфейс программирования для таймеров.
- Виджет QLabel позволяет отображать текст или изображение. Не поддерживает никаких функций взаимодействия с пользователем.
- Класс QGraphicsScene предоставляет поверхность для управления большим числом графических 2D элементов. Этот класс служит как контейнер для QGraphicsItems. Он используется вместе с QGraphicsView для отображения графических объектов, таких как линии, прямоугольники, текст или даже собственные элементы на двухмерной поверхности.
- Класс QGraphicsView предоставляет виджет для отображения содержимого QGraphicsScene. QGraphicsView является частью каркаса графического представления.
- Класс QGraphicsPixmapItem предоставляет элемент растрового изображения, который вы можете добавить на QGraphicsScene.
- Класс QKeyEvent содержит описание ключевого события. Ключевые события отправляются виджету с фокусом ввода с клавиатуры при нажатии или отпускании клавиш.

С их помощью можно реализовать логику объектов в игре, использовать 2D текстуры, отображать объекты в окне, а также дать пользователю возможность взаимодействия с объектами через нажатие клавиш клавиатуры. В данном разделе будет рассмотрены основные функциональные составляющие проектируемой игры.

класс Palitra

Рассмотрим основной элемент данной программы. Данный класс занимается подготовкой к запуску игры и создаёт виджет по запросам пользователя: выбор карты, выбор персонажа и т. д. Предназначение полей класса:

QGraphicsScene *scene – класс, который отвечает за связывание всех игровых объектов в общую картину, которая позже будет отображаться в виджете.

QTimer *gameTime – класс, отвечающий за таймер, по истечению которого игра будет окончена.

QLabel *timer – виджет, отображающий оставшееся время до конца игры.

QLabel *player1result и *player2result – виджеты, которые по истечению таймера игры отображают количество очков, набранных игроками.

QPushButton *restartButton и *quitButton – командные кнопки, отвечающие за перезапуск игры с теми же параметрами и выхода в главное меню, отображаются по истечению игрового таймера.

Player* player1 – игровой объект, который осуществляет управление персонажей и следит за их местоположением на карте.

Конструктор класса:

Palitra(QWidget *parent = 0) – создаёт виджет фиксированного размера в котором будет отображаться сцена, создаёт сцену и добавляет в неё все игровые объекты, создаёт и настраивает таймер и командные кнопки.

Методы класса:

void clickedRestart() – метод, который при нажатии кнопки “Restart”, перезапускает игру с теми же параметрами.

void clickedQuit() – метод, который при нажатии кнопки “Quit” закрывает виджет и выходит в главное меню.

void GameTimer – метод, который отвечает за игровой таймер, по истечению таймера останавливает все процессы в игре, выводит на экран командные кнопки и счёт обоих игроков.

класс Mainwindow

Данный класс является основным в работе с пользовательским окном. Он реализован с целью взаимодействия пользователя с продуктом, обработки его запросов и передачи их классу Palitra. Данный класс содержит данные поля:

Palitra *palitra – объект вышеупомянутого класса.

Int map – переменная для создания карты под определённым номером

Int character1 – переменная для смены текстур первого игрока.

Int character2 – переменная для смены текстур второго игрока.

Конструктор:

MainWindow() – создаёт главное окно и фиксирует его размер, инициализирует командные кнопки и соединяет сигналы кнопок с методами.

Методы класса:

void clickedStartGame() – метод, который срабатывает при нажатии кнопки “Start Game”. Создаёт объект класса Palitra, после того как объект был создан и полностью готов к использованию, выводит виджет на экран.

void on_Desktop_clicked() – метод, которые позволяет получить

класс **Player**

Данный класс реализован для создания объектов сцены, их управления и для их взаимодействия между собой.

Поля класса:

QString player1_Img[8] – представляет собой массив текстур первого игрока.

Bool moveVertical и moveHorizontal – переменные, предназначенные для смены текстур при движении персонажей по карте.

Tile *** tiles – массив объектов класса Tile, представляющий собой игровую карту.

QString tile_Img[4] – представляет собой массив текстур для игрового поля.

Float i, j – предназначены для отслеживания координат первого игрока на игровом поле.

Player2 *player2 – игровой объект представляющий собой второго игрока.

QTimer *player1_TimerW … *player1_TimerD, *player2_timer8 … *player2_timer6 – таймеры, использующиеся для передвижения игроков по карте в определённом направлении.

Конструктор:

Player() – инициализирует таймеры предназначенные для передвижения игроков, заполняет массив QString player1_Img[8] текстурами.

Методы класса:

void createTiles() – метод, который инициализирует массив клеток tiles и заполняет массив QString tile_Img[4] текстурами.

void keyPressEvent(QKeyEvent *event) – метод, срабатывающий при нажатии клавиши с клавиатуры. Отвечает за управление персонажами, таймерами и передачи сигналов таймеров в другие методы класса.

void player1_ActionW() … player1_ActionD(), player2_Action8() … player2_Action6() – срабатывают после подачи сигнала от определённого таймера. Отвечают за перемещение персонажа по игровому полю и за его взаимодействие с клетками.

Класс **Player2**

Данный класс реализован для создания объекта сцены.

Поля класса:

QString player2_Img[8] – представляет собой массив текстур второго игрока.

Float i, j – предназначены для отслеживания координат второго игрока на игровом поле.

Конструктор:

Player2() – заполняет массив QString player2_Img[8] текстурами.

Класс **Tile**

Данный класс реализован для создания объекта сцены.

Поля класса:

Int status – показывает нынешний статус определённой клетки игровой карты.

Использование клавиш в приложении:
W – движение первого игрока вверх.
A – движение первого игрока влево.
S – движение первого игрока вниз.
D – движение первого игрока вправо.

8 – движение второго игрока вверх.
4 – движение второго игрока влево.
5 – движение второго игрока вниз.
6 – движение второго игрока вправо.

Главная функция main() создаёт объект класса MainWindow, который в свою очередь обеспечивает дальнейшую работу программы.

2 Обоснование выбранных методов и алгоритмов

Метод Palitra() класса Palitra является одним из основых методов этого программного обеспечения. Именно он инициализирует все объекты сцены, создаёт игровое поле и подготавливает игру к запуску.

Данный алгоритм был выбран из-за своей простоты в реализации программного кода, небольшого объема и низкой вычислительной сложности алгоритма. Простота реализации достигнута путем работы с библиотеками QGraphicsScene и QGraphicsView, которые позволяют без какого-либо труда объединить все имеющиеся игровые объекты в одну систему и спокойно манипулировать ими в процессе игры. Недостатком является тот факт, что в сцене только один объект может быть в фокусе (только один объект сцены принимает нажатие клавиш с клавиатуры).

Алгоритм приёма сигналов с клавиатуры. Этот алгоритм является одним из основополагающих алгоритмов этого программного обеспечения. Данный алгоритм написан с помощью библиотеки QKeyEvent, которая предоставляет возможность считывать нажатия с клавиатуры. Как только сигнал считан, запускается таймер для определённого направления движения. По истечению этого таймера срабатывает метод перемещения персонажа в определённом направлении. Данный алгоритм прост в реализации программного кода, имеет маленькую вычислительную сложность, но большой по объёму. Такой объём достигнут путем прохода по всем операторам ветвления до того момента, как будет найден нужный сигнал клавиши.

Алгоритм перемещения персонажей. Алгоритм срабатывает после приёма сигнала с таймера. Он проверяет возможность перемещения персонажа в данном направлении, и, если перемещение возможно, меняет его текстуру и меняет состояние клетки, совпадающей по координатам с координатами игрока.

Данный алгоритм простой в реализации поставленной задачи и небольшой по размеру программного кода.

Алгоритм истечения игрового таймера – данный алгоритм принимает сигнал с основного игрового таймера и постепенно уменьшает значение переменной, отвечающей за игровое время. По его истечению происходят следующие действия: останавливаются все процессы в игре, происходит подсчёт закрашенных клеток и результаты выводятся на экран, появляются кнопки “Restart” и “Quit”. Алгоритм простой в реализации поставленной задачи и объемный по размеру программного кода.

Алгоритм создания карты. Алгоритм инициализирует массив клеток и присваивает им некий статус и текстуру, в зависимости от того, какие стартовые настройки задал пользователь в главном окне. Алгоритм работает

в паре с классом QString. Простой в реализации поставленной задачи, но большой по размеру программного кода.

Все конструкторы классов служат для инициализации полей и соединения между собой кнопок и соответствующих методов. Все деструкторы классов служат для удаления ненужной информации. Все конструкторы и деструкторы простые в реализации поставленной задачи и небольшие по размеру программного кода.

3 Описание программы для программистов

При создании данного программного обеспечения использовались правила паттерна проектирования приложений MVC.

При создании сложных приложений существенно увеличивается трудоёмкость разработки кода. Основная концепция идеологии MVC – разделить части программы, отвечающие за хранение и доступ к данным, их отображение и взаимодействие с пользователем на три отдельных модуля, разработка которых относительно независима.

Такой подход:

- увеличивает степень повторного использования кода (например, одна модель может быть применена к нескольким различным между собой представлениям данных);
- облегчает модификацию программы (в идеале можно менять модель, вид или контроллер отдельно);
- становится выгоден при разработке приложений в архитектуре "клиент-сервер", приложений СУБД, использующих хранилища данных, или для перенаправления однопоточного GUI в многопоточные обработчики.

Итак, MVC включает в себя 3 типа объектов:

- Модель – осуществляет соединение с источником данных, служит их логической моделью, предоставляет интерфейс другим компонентам архитектуры. Модель представляет знания о моделируемом объекте, но не должна содержать информации о том, как эти знания визуализировать;
- Вид (представление) – обеспечивает конечное (например, экранное) представление данных для пользователя. Вид получает из модели модельные индексы, являющиеся ссылками на элементы данных. Сообщая модельные индексы модели, вид может получить элементы из источника данных. Существенно то, что к модели можно применить несколько видов, не изменяя её. Например, одни и те же данные можно показать в виде таблицы, графика или круговой диаграммы;
- Контроллер - отвечает за пользовательский интерфейс. Он обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Существуют различные реализации MVC, во многих из них базовый шаблон MVC модифицируется. Так, в QT применяется следующая разновидность архитектуры "модель-представление":

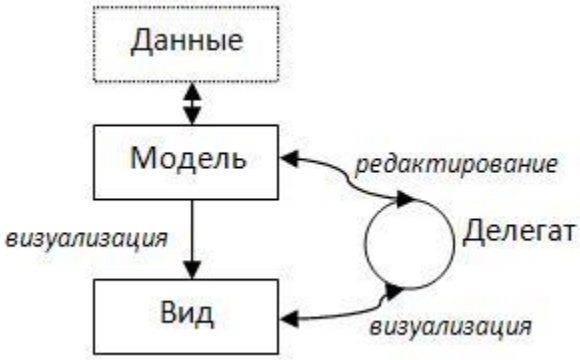

Рис. 3.1. MVC в QT

В стандартных представлениях, уже имеющихся в QT, делегат отображает элементы данных. Если элемент редактируемый, делегат связывается с моделью, непосредственно используя модельные индексы.

Файл MainWindow.ui отвечает за визуализацию и получение команд пользователя. Данные команды обрабатываются при помощи методов, описанных в файлах MainWindow.cpp, Palitra.cpp, Player.cpp соответственно, которые в свою очередь выполняют действия контроллера. В совокупности с файлами MainWindow.h, Palitra.h, Player1.h осуществляется работа модели.

Таким образом осуществляется обработка данных, вводимых пользователем, их визуализация, а также логика работы данного программного обеспечения.

Схема взаимодействия файлов представлена ниже:

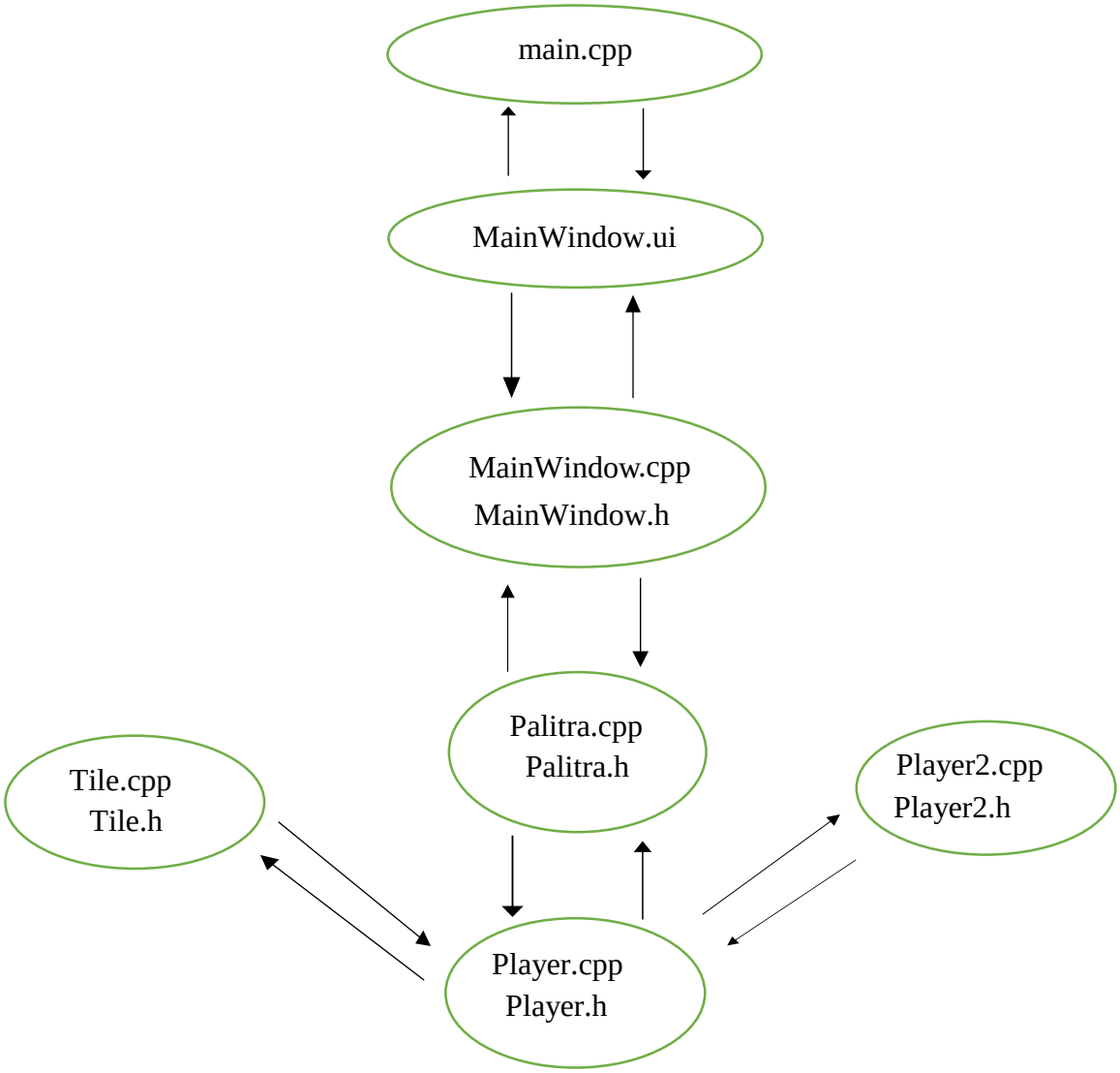

Рис 3.2. Структура файлов

4 Описание алгоритмов решения задач

В данном разделе будут рассмотрены алгоритмы, используемые в разработанном ПО.

Алгоритм создания карты

Данный алгоритм разработан в методе createMap класса Player. Существуют несколько карт, отличие создания которых лишь в проходе по клеткам в циклах и в отличающихся массивах текстур. Опишем его кратко по шагам.

Шаг 1. Заходим в цикл, исключая некоторые координаты.

Шаг 2. Если исключённая координата, переходим в шаг 3, если нет, переход в шаг 4.

Шаг 3. Присваиваем клетке status = 0, меняем текстуры на tile_Img[0]. Переход в шаг 5.

Шаг 4. Присваиваем клетке status = 3, меняем текстуры на tile_Img[3].

Шаг 5. Переход в шаг 1, пока цикл не закончится. Если закончился, переход в шаг 6.

Шаг 6. Завершение алгоритма.

Алгоритм приёма сигналов с клавиатуры

Данный алгоритм разработан в методе void keyPressEvent(QKeyEvent *event) класса Player.

Опишем алгоритм при отправки определённого одного сигнала. Для всех остальных сигналов алгоритм идентичный.

Шаг 1. Если поступил сигнал нужной клавиши, переход к шагу 2.

Шаг 2. Если таймер этого сигнала не активен и персонаж находится в центре клетки, переход к шагу 3, если наоборот – переход к шагу 8.

Шаг 3. Если таймер любого другого сигнала активен, переход к шагу 4, если нет, переход к шагу 5.

Шаг 4. Остановить таймеры других сигналов.

Шаг 5. Проинициализировать таймер нужного сигнала.

Шаг 6. Запустить таймер нужного сигнала с интервалом 150 мс.

Шаг 7. Соединить сигнал проинициализированного таймера с нужным методом Action.

Шаг 8. Завершение алгоритма.

Алгоритм перемещения персонажа

Следующий алгоритм – перемещение персонажа. Данный алгоритм описан в методе player1_ActionW класса Player.cpp . Для каждой кнопки движения прописан свой метод Action. Опишем данный метод для одного направления движения с помощью блок-схемы.

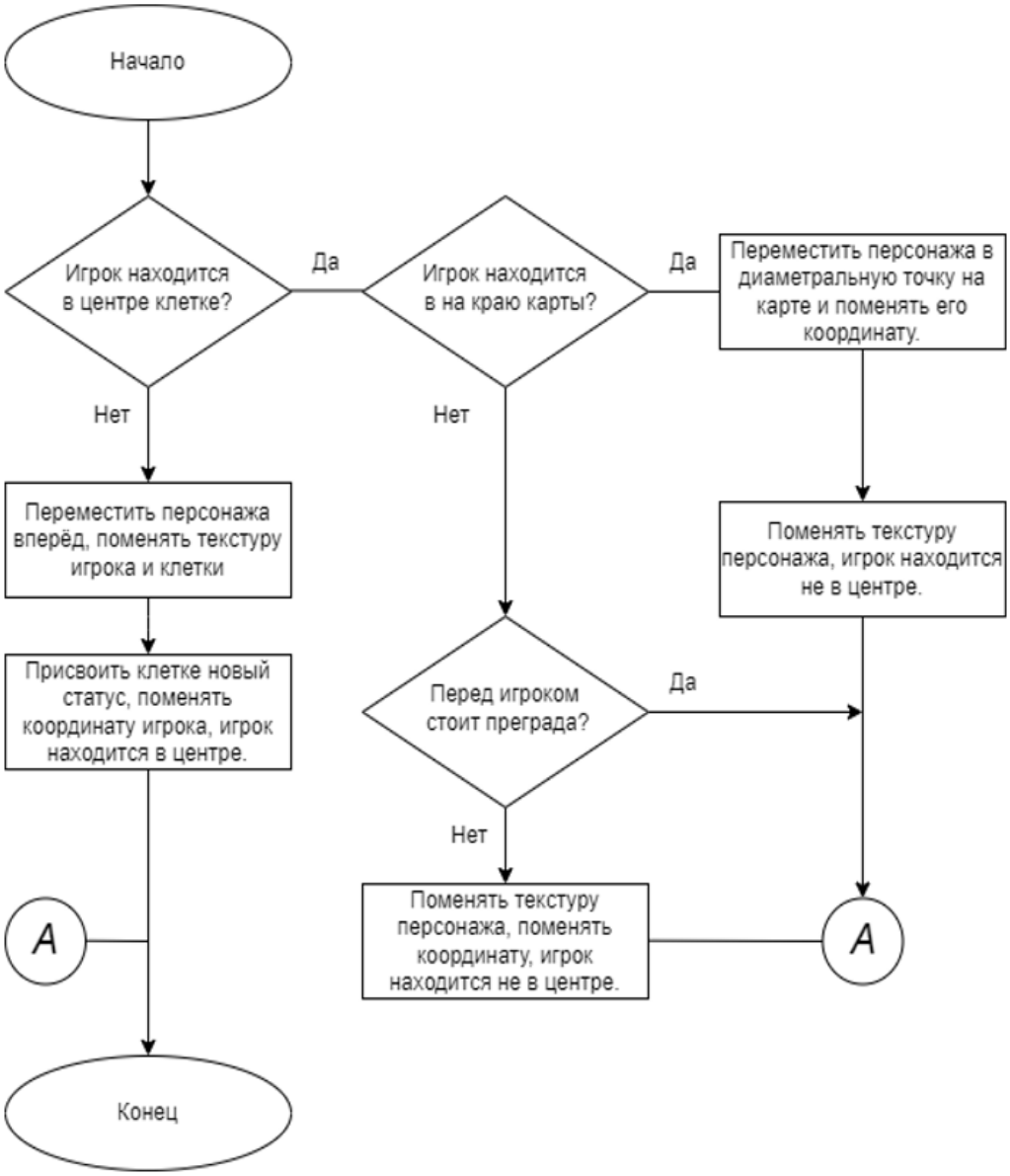

Рис. 4.1. Алгоритм удаления выбранной папки.

Алгоритм создания объекта класса Player

Следующий алгоритм – создание объекта класса Player. Данный алгоритм описан в методе Player() класса Player.

Рис. 4.2. Алгоритм создания новой папки.

Алгоритм игрового таймера и окончания игры

Данный алгоритм описан в методе GameTime класса Palitra. Данный метод описан с помощью блок-схемы в приложение Б.

Алгоритм создани объекта класса Palitra

Данный алгоритм описан в методе Palitra(int map, int character1, int character 2) класса Palitra.

Опишем данный алгоритм по шагам.

Шаг 1. Создание сцены, фиксирование размера окна в котором будет игра.

Шаг 2. Инициализация Player player1 и Player2 player2.

Шаг 3. Присваивание клеткам карты координат.

Шаг 4. Создание карты по запросу пользователя.

Шаг 5. Добавление player1 и player 2 в сцену, установка фокуса на объекте player1.

Шаг 6. Присваивание координат player1 и player2. Перемещение объектов на координаты.

Шаг 7. Создание кнопок “Quit” и “Restart” и их настройка.

Шаг 8. Создание виджетов player1result, player2result, timer.

Шаг 9. Добавление кнопок и виджетов в сцену.

Шаг 10. Запуск игрового таймера.

Шаг 11. Завершение алгоритма.

Данные алгоритмы являются достаточными, для организации логики игры, управления персонажами и отрисовки сцены. Так же стоит заметить, что благодаря организации взаимодействия с кнопками пользователю стали доступны и другие функции, которые не обязательны, но созданы для упрощённого пользования программой. Все эти алгоритмы были созданы для простого использования программы и понятной главной функции.

5 Руководство пользователя

В данном разделе приведена вся информация, которая должна публиковаться вместе с продуктом. В данной программе для корректного функционирования требуются следующие файлы: include для загрузки заголовочных файлов, src для загрузки .cpp файлов.

Далее будет представлены снимки экранов из приложения с пояснением каждых функций.

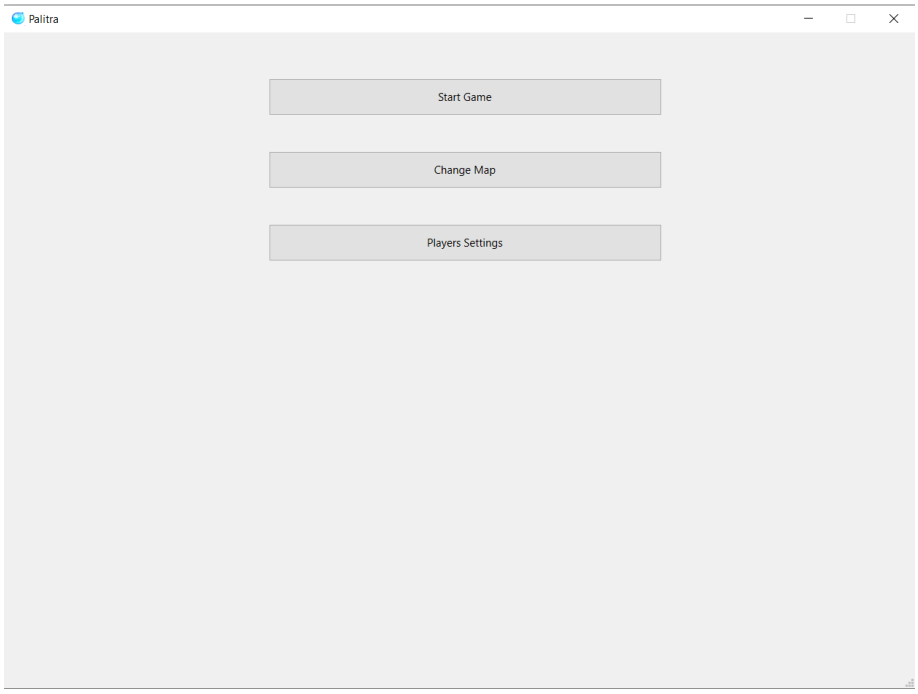

Рисунок 5.1 – главное окно приложения.

На рисунке выше показано главное окно приложения. Пользователь может взаимодействовать с программой при помощи 3-х кнопок: “Start Game” “Change Map” и “Players Settings”. Две последние кнопки используются для настройки игроков и карты. Кнопка “Start Game” запускает игру с заданными пользователем параметрами. Если же параметры не были установлены, игра запускается с параметрами по умолчанию.

Рисунок 5.2 – Виджет с запустившейся игрой.

На рисунке 5.2 показан виджет с запустившейся игрой. Здесь мы видим 2 игрока с закрашенным в их цвет стартовыми клетками. Для управления 1-ым игроком используются клавиши W, A, S, D. Для управления 2-ым: 8, 4, 5, 6. Также сверху можно наблюдать таймер, по истечению которого, игра закончится.

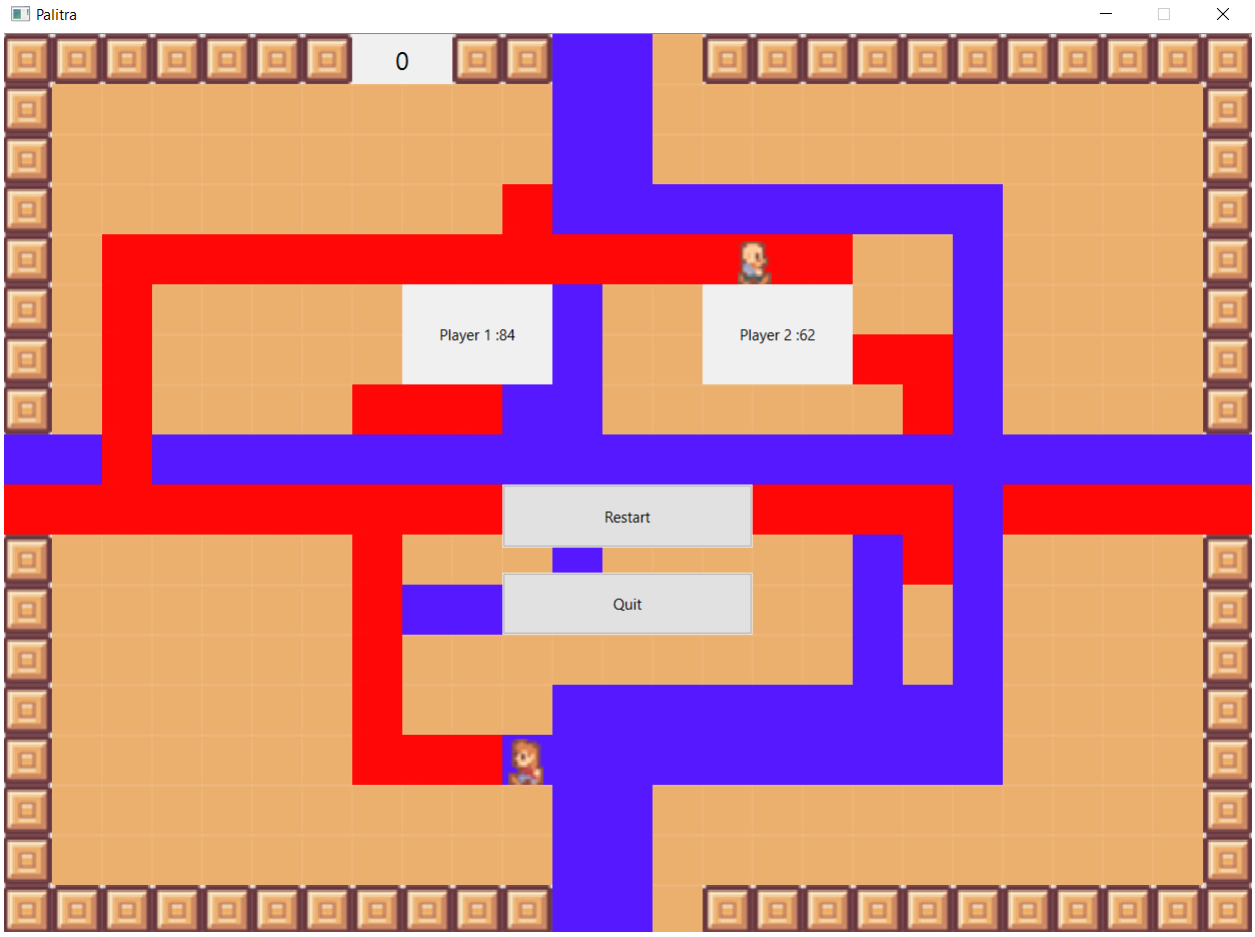

Рисунок 5.3 – окончание игры.

На рисунке 5.3 показан виджет после окончания игрового таймера. Появляются результаты 2-х игроков за игру и кнопки “Restart” и “Quit”. “Quit” вернёт пользователя в главное меню, где он снова сможет поменять настройки. Кнопка “Restart” перезапустит игру с теми же параметрами.

Заключение

В данном разделе будут проведены итоги разработки программного обеспечения.

В ходе разработки и тестирования данного ПО было проверено и практически обосновано, что оно выполняет все требуемые функции:

- Создание игрового окна по параметрам пользователя;
- Использование примитивной графики;
- Управление игроками и корректная работа игры;

По ходу разработки данного программного обеспечения были получены знания по работе с объектно-ориентированным программировании, с библиотеками Qt, с проектированием программного обеспечения, с пользовательским интерфейсом.

Таким образом данный курсовой проект помог понять, как устроены многие приложения, показать их недостатки и понять, как их можно усовершенствовать.

Литература

1) Лафоре Р. Объектно-ориентированное программирование с C++/ 4-е издание М.:Питер, 2004. – 923 с.
2) Луцик Ю.А. Объектно-ориентированное программирование на языке C++ / Ю.А.Луцик, А.М.Ковальчук, И.В.Лукьянова. – Мн.: БГУИР, 2003
3) Документация фреймворка Qt на русском языке. – [Электронный ресурс]. – Адрес ресурса: http://doc.crossplatform.ru/qt/4.5.0 - Дата доступа 10.12.2021
4) Применение паттерна проектирования MVC. – [Электронный ресурс]. – Адрес ресурса: https://ru.hexlet.io/blog/posts/chto-takoe-mvcrasskazyvaem-prostymi-slovami - Дата доступа 08.12.2021

ПРИЛОЖЕНИЕ А
(Обязательное)

Листинг кода

ПРИЛОЖЕНИЕ Б
(Обязательное)

Диаграмма классов приложения

ПРИЛОЖЕНИЕ В
(Обязательное)

Блок-схема алгоритма игрового таймера

ПРИЛОЖЕНИЕ Г
(Обязательное)

Диаграмма последовательностей приложения

ПРИЛОЖЕНИЕ
(Обязательное)