Министерство образования Республики
Беларусь

Учреждение образования
«Белорусский государственный
университетинформатики 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 “Змейка”»

Студент В. В. Приловский

Руководитель Е. В. Богдан

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

_____
(подпись)
_____2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Приловскому Виктору Витальевичу*

1. Тема проекта игра “Змейка”
2. Срок сдачи студентом законченного проекта *15 декабря 2023 г.*
3. Исходные данные к проекту: Программа представляет собой игру “Змейка”, в которой игрок управляет змеей, которая движется по экрану и пытается съесть яблоки. Цель игры - сделать змею как можно длиннее, избегая столкновений с границами экрана и собственным хвостом. Программа написана на языке C++ с использованием фреймворком QT.
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Лист задания.
Введение.
1. Обзор источников.
1.1. Обоснование разработки.
1.2. Обзор аналогов.
1.3. Обзор методов и алгоритмов решения поставленной задачи.
2. Функциональное проектирование.
2.1. Структура входных и выходных данных.
2.2. Разработка диаграммы классов.
2.3. Описание классов.
3. Разработка программных модулей.
3.1. Разработка схем алгоритмов.
3.2. Разработка алгоритмов.
4. Результаты работы.
5. Литература
Заключение

Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Диаграмма классов.

2. Схема алгоритма `CreateBonus(Snake* m_snake, int m_fieldSize, SnakeFood* m_food, SnakeBonus *&m_bonus, QTimer *m_bonusTimer).`

3. Схема алгоритма `keyPressEvent(QKeyEvent *e).`

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

7. Дата выдачи задания *15 сентября 2023*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

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 15.12.23 – 10 %

Защита курсового проекта с 21.12 по 28.12.23г.

РУКОВОДИТЕЛЬ Е.В. Богдан

_____

(подпись)

Задание принял к исполнению _____ В.В. Приловский

(дата и подпись студен

СОДЕРЖАНИЕ

ВВЕДЕНИЕ……5
1 ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ ……6
2 ОБЗОР ЛИТЕРАТУРЫ .. ……7
2.1 Обзор методов и алгоритмов решения поставленной задачи……..7
2.2 Разработка требований к функционалу……..8
2.3 Обзор аналогов……..8
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ……12
3.1 Структура входных и выходных данных..……12
3.2 Разработка диаграммы классов. ……12
3.3 Описание классов ……13
4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ……..19
4.1 Разработка схем алгоритмов ……..19
4.2 Разработка алгоритмов ……...19
5 РЕЗУЛЬТАТ РАБОТЫ……..21
ЗАКЛЮЧЕНИЕ……28
ПРИЛОЖЕНИЕ А ……30
ПРИЛОЖЕНИЕ Б……31
ПРИЛОЖЕНИЕ В ……32
ПРИЛОЖЕНИЕ Г……33

ВВЕДЕНИЕ

В течение всей истории информационных технологий, люди постоянно преобразуют свои идеи в код, создавая программы для различных сфер жизни, включая развлекательную индустрию. Одним из ярких примеров такого влияния является компьютерная игра "Змейка". Эта игра, несмотря на свою простоту, завоевала мировую популярность и стала классикой жанра.

Змейка, впервые появившаяся в конце 1970-х годов, была одной из первых игр, которую можно было сыграть на персональных компьютерах. Она представляет собой аркадную игру, где игрок управляет змейкой, собирая пищу и стараясь не столкнуться со стенами или с собственным хвостом. С каждым съеденным кусочком пищи змейка становится длиннее, что усложняет управление и требует стратегического мышления.

Змейка быстро стала популярной благодаря своей простоте и захватывающей геймплею. Она была включена во множество компьютерных платформ и мобильных устройств, что позволило миллионам людей наслаждаться этой игрой в любое время и в любом месте. Змейка стала символом ранних компьютерных игр и оказала значительное влияние на развитие игровой индустрии.

С появлением новых технологий и развитием графики, змейка получила новые интерпретации и вариации. Она стала более визуально привлекательной и разнообразной, сохраняя при этом свою простоту и веселье. В настоящее время существуют многочисленные версии змейки с улучшенной графикой, разнообразными игровыми режимами и возможностью соревноваться с другими игроками онлайн.

Таким образом, игра "Змейка" является примером того, как простая идея может стать классикой игровой индустрии и оставить свой след на протяжении десятилетий. Ее популярность и влияние демонстрируют важность творчества и инноваций в создании игр, которые могут вдохновить и развлечь людей по всему миру.

1. ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Цель данного проекта заключается в разработке игры "Змейка" с использованием языка программирования C++ и фреймворка Qt. Игра представляет собой увлекательную аркаду, где игрок управляет змейкой с целью съесть как можно больше еды, избегая столкновения с собственным хвостом.

В рамках проекта были реализованы следующие функциональные возможности:

1. Регистрация новых пользователей: Пользователи могут создавать учетные записи для сохранения своих результатов и достижений.
2. Авторизация пользователя: Зарегистрированные пользователи могут войти в свои учетные записи для доступа к сохраненным данным и результатам.
3. Выход из аккаунта: Пользователи могут безопасно выйти из своих учетных записей.
4. Просмотр максимального результата всех зарегистрированных пользователей в порядке убывания: Игроки могут просматривать лучшие результаты других игроков и соревноваться с ними.
5. Настройка цвета "Змейки": Пользователи могут выбрать цвет своей змейки по своему вкусу.
6. Регулировка громкости музыки: Игроки могут изменять уровень громкости музыки в игре или полностью отключить его.
7. Включение и отключение звука кнопок: Игроки могут включать или отключать звуки кнопок во время игры.
8. Сбор еды для увеличения длины "Змейки" и увеличения счета: Змейка может съесть еду для увеличения своей длины и получения большего количества очков.
9. Сбор бонусов для ускорения и замедления "Змейки" и увеличения счета: Игроки могут собирать бонусы, которые влияют на скорость змейки и позволяют набирать больше очков.
10. Пауза игры: Игроки могут ставить игру на паузу в любой момент для временного приостановления игрового процесса.

Для хранения данных о игроках и их результатах используются текстовые файлы с расширением .txt. В дальнейшем планируется расширение функциональности игры, включая добавление препятствий и различных уровней сложности.

Разработка данной игры является актуальной задачей, которая приносит новые идеи в классическую игру "Змейка" и предоставляет пользователям интересное и увлекательное времяпрепровождение.

2. ОБЗОР ЛИТЕРАТУРЫ

2.1 Обзор методов и алгоритмов решения поставленной задачи

2.1.1 Объектно-ориентированное программирование

Объектно-ориентированное программирование – это методология программирования, которая использует объекты для моделирования и организации кода. В C++ ООП реализуется с помощью классов и объектов [7].

Классы в C++ являются шаблонами для создания объектов. Они определяют, какие данные и функции будут содержать объекты этого класса.

Объекты в C++ – это экземпляры класса. Каждый объект имеет свой собственный набор данных (атрибутов) и функций (методов), определенных в его классе.

ООП в C++ поддерживает четыре основных принципа:

1. Инкапсуляция: Инкапсуляция означает сокрытие деталей реализации и объединение данных и методов в одном объекте. В приложении, например, данные пользователя инкапсулированы в объектах класса GameField;
2. Наследование: Наследование позволяет создавать новые классы на основе существующих, переиспользуя их код и добавляя новые функции. В приложении классы Snake, SnakeFood и SnakeBonus наследуются от класса ItemElements. Это позволяет им использовать переменные m_x и m_y из базового класса;
3. Полиморфизм: Полиморфизм позволяет использовать один и тот же интерфейс для различных типов данных;
4. Абстракция: Абстракция означает представление сложных систем через более простые интерфейсы.

2.1.2 Фреймворк QT

Для создания графического интерфейса также используется Qt. Qt – это мощный инструмент для создания GUI в C++ [4]. Qt SQL. Qt – полностью объектно-ориентированная библиотека. Новая концепция ведения меж объектных коммуникаций, именуемая "сигналы и слоты", полностью заменяет былую, не вполне надежную модель обратных вызовов [6]. Этот модуль включает классы QWidget, QPainter и другие, которые обеспечивают удобный объектно-ориентированный интерфейс для работы с базами данных. Также имеется возможность обработки событий, например нажатия клавиш клавиатуры, нажатия элементов интерфейса [5].

2.2 Разработка требований к функционалу

“Игра «Змейка»” — это приложение, которое представляет собой интерактивную игру, позволяющую пользователям управлять змейкой на игровом поле. Функционал может зависеть от нужд пользователя:

1. Игра: Все пользователи могут играть в игру “Змейка”. Змейка движется по игровому полю, и цель игры - съесть как можно больше еды, избегая столкновения с собственным хвостом;
2. Использование бонусов: Пользователи могут подбирать бонусы во время игры, которые влияют на скорость змейки;
3. Просмотр счета: Пользователи могут просматривать свой текущий счет во время игры;
4. Пауза/продолжение игры: Пользователи могут ставить игру на паузу и продолжать игру после паузы;
5. Начать новую игру: Пользователи могут начать новую игру в любое время;
6. Регистрация и авторизация пользователей: Приложение поддерживает систему регистрации и авторизации пользователей. Пользователи должны ввести свои учетные данные для входа в систему;
7. Рекорды: Пользователи могут просматривать таблицу рекордов, которая отображает лучшие результаты всех зарегистрированных пользователей.

Дополнительные требования к приложению включают в себя:

1. Удобство использования: Интерфейс пользователя должен быть интуитивно понятным и легким в использовании;
2. Привлекательный дизайн: Дизайн игры должен быть привлекательным и интересным для пользователей;
3. Музыкальное сопровождение: Игра может включать в себя музыкальное сопровождение для улучшения игрового опыта;
4. Настройки: Пользователи могут настраивать различные параметры игры, такие как цвет змейки или громкость музыки.

2.3 Обзор аналогов

Snake (1976) - Это одна из первых змеек, созданных для игровых автоматов и домашних компьютеров. Игра также была известна под названием Blockade. Snake была разработана и выпущена компанией Gremlin Industries. Геймплей заключался в управлении змейкой, которая перемещалась по игровому полю, собирая пищу и становясь длиннее. Игрок должен был избегать столкновения со своим хвостом и стенами, поскольку это приводило к проигрышу. Эта игра стала популярной аркадной игрой и заложила основы жанра "Змейка".

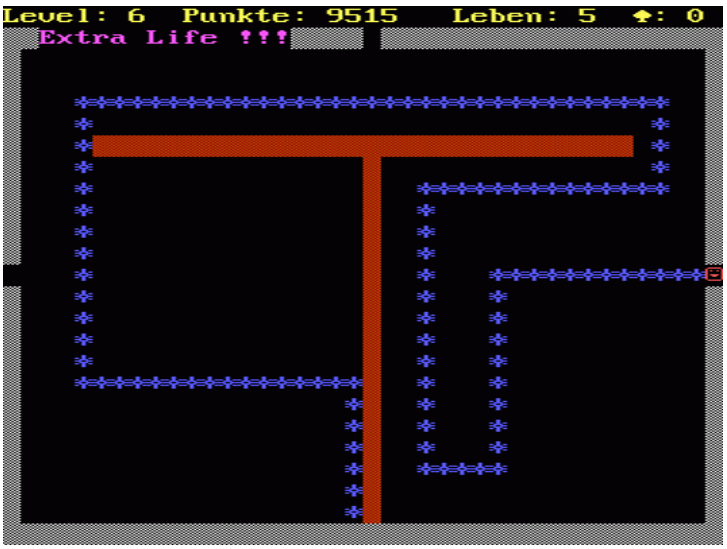

Рисунок 2.1 – игра Snake (*Blockade)* 1976 года

Nibbler (1982) - Эта змейка была выпущена компанией Rock-Ola. Она отличалась от других змеек тем, что у игрока было ограниченное количество времени на прохождение уровней. Целью игрока было собирать пищу, чтобы змейка становилась длиннее и двигалась быстрее. Каждый съеденный предмет приносил игроку очки. Если змейка сталкивалась с препятствием или своим хвостом, игра заканчивалась. Nibbler была успешной игрой в свое время и получила признание игровых фанатов.

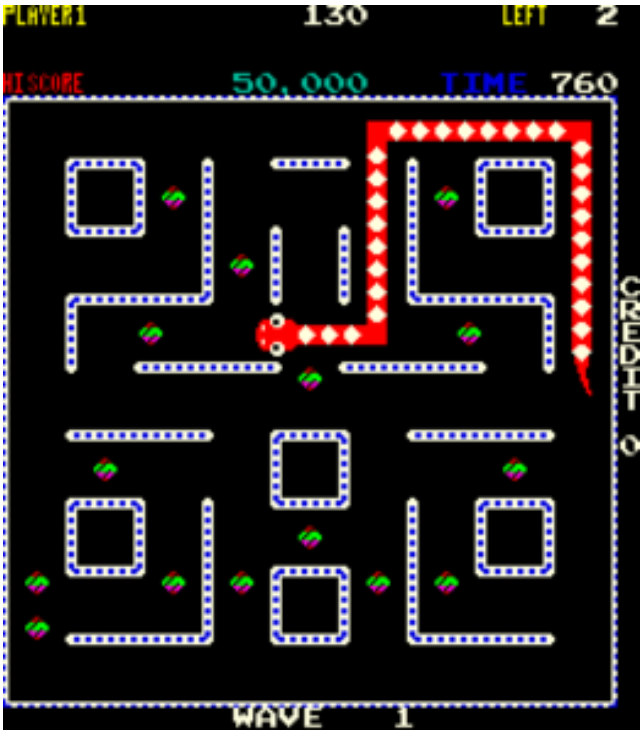

Рисунок 2.2 – игра Nibbler 1982 года

Snake II (1998) - Эта версия змейки была включена в мобильные телефоны Nokia и стала одной из самых популярных игр на мобильных устройствах. Игроки могли управлять змейкой, передвигая ее по игровому полю и собирая пищу. Целью было избегать столкновения со своим хвостом и стенами. Snake II стала известна своей простой и привлекательной механикой игры, которая привлекла множество игроков.

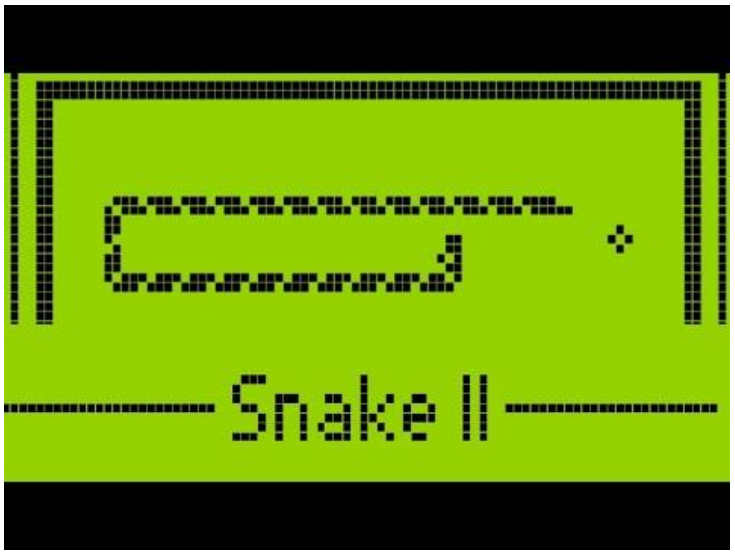

Рисунок 2.3 – игра Snake II 1998 года

Slither.io (2016) - Это многопользовательская онлайн-игра, где игроки управляют змейками и соревнуются между собой. Цель игры - стать самой длинной змейкой, съев пищу и других игроков. Slither.io получила огромную популярность благодаря своей простой и привлекательной механике игры, а также возможности играть с другими игроками по всему миру. Игроки могут настроить свои змейки и стратегии, чтобы выжить и стать лидером на игровом поле.

Рисунок 2.4 – игра Slither.io 2016 года

Каждая из этих игр имеет свои особенности и вклад в историю жанра "Змейка". Они продемонстрировали различные этапы развития этого жанра, начиная с ранних аркадных автоматов до мобильных и онлайн-игр.

Ранние аркадные автоматы, такие как "Blockade" и "Nibbler", представляли собой простые и увлекательные варианты игры "Змейка". Они были ограничены в графике и функциональности, но стали популярными благодаря своей простоте и привлекательности для широкой аудитории.

С появлением персональных компьютеров и консолей, змейки получили новые возможности и улучшения. Игры, такие как "Snake Byte" для персональных компьютеров Apple II и "Nibbles" для операционной системы MS-DOS, предложили игрокам более сложные уровни, улучшенную графику и новые игровые механики.

С развитием мобильных устройств змейки стали одними из самых популярных игр для мобильных телефонов. Игры "Snake" для телефонов Nokia стали культовыми и покорили сердца миллионов игроков. Мобильные змейки обычно предлагали различные уровни сложности, разнообразные локации и дополнительные элементы геймплея, такие как специальные фрукты, бонусы и препятствия.

С развитием онлайн-игр змейки стали доступны для массового многопользовательского взаимодействия. Игроки могут соревноваться друг с другом в режиме реального времени, сражаться за лидерство в рейтингах и создавать собственные команды. Онлайн-змейки обычно предлагают различные режимы игры, разнообразные варианты внешнего оформления и возможность настраивать правила игры под свои предпочтения.

Эти игры вдохновили множество разработчиков создавать свои собственные версии змеек с различными нововведениями и улучшениями. Некоторые разработчики добавляют разнообразные игровые режимы, специальные эффекты и уникальные графические стили. Другие фокусируются на инновационных механиках геймплея, введении новых элементов или даже объединении змейки с другими жанрами игр.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Структура входных и выходных данных

Приложение работает с двумя основными типами входных данных:

1. Данные пользователя: Это данные, которые пользователь вводит через GUI. Они включают в себя информацию для регистрации нового пользователя, данные для входа в систему существующего пользователя, а также действия, которые пользователь хочет выполнить в игре, такие как управление змейкой перевод режима игры в паузу;

2. Данные игры: Это данные, которые приложение генерирует во время игры, такие как положение змейки, еды и бонусов на игровом поле.

Приложение генерирует следующие выходные данные:

1. Данные GUI: это данные, которые отображаются пользователю через GUI. Они могут включать в себя сообщения об ошибках, подтверждения успешных действий и информацию из файлов с расширением типа .txt;

2. Сигналы: приложение также генерирует сигналы как часть механизма сигналов и слотов Qt. Сигналы используются для обработки событий, таких как нажатие кнопок пользователем;

3. Текстовые файлы: приложение способно открывать текстовые файлы, содержащие всю информацию о призывниках.

3.2 Разработка диаграммы классов

Диаграмма классов – это структурная диаграмма языка моделирования UML, которая демонстрирует общую структуру иерархии классов системы, их кооперaций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Она широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На диаграмме классы представлены в рамках, содержащих три компонента:

1. В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом;

2. В средней части перечислены атрибуты (поля) класса;

3. В нижней части перечислены методы класса.

Целью создания диаграммы классов является графическое представление статической структуры классов.

Диаграмма классов для игры «Змейка» приведена в приложении А.

3.3 Описание классов

3.3.1 Класс MainWindow

Класс `MainWindow` представляет собой главное окно приложения и является базовым классом для остальных классов. Вот более подробное описание его методов.

Поля класса MainWindow:

1. `QString SNAKE_COLOR` - поле, которое хранит цвет змейки, выбранный пользователем;
2. `QMediaPlayer* player` - поле, которое хранит указатель на объект класса `QMediaPlayer`, который используется для воспроизведения музыки в приложении;
3. `int snakeColor` - поле, которое хранит код цвета змейки;
4. `GameField* m_gameField` - поле, которое хранит указатель на объект класса `GameField`, который представляет собой игровое поле для змейки.

Методы класса MainWindow:

1. Конструктор `(MainWindow::MainWindow(QWidget *parent))`: Это конструктор класса `MainWindow`. Он настраивает пользовательский интерфейс для окна, устанавливает размеры окна и стиль. Он также инициализирует медиаплеер для воспроизведения музыки во время игры;
2. Деструктор `(MainWindow::~MainWindow())`: Это деструктор класса `MainWindow`, который удаляет пользовательский интерфейс при уничтожении объекта;
3. `playGame()`: Этот метод вызывается при нажатии кнопки “Play”. Он создает новое окно игры и скрывает главное окно;
4. `enterName()`: Этот метод вызывается при нажатии кнопки “Enter nickname”. Он создает диалоговое окно для ввода имени пользователя.
5. `openSettings(QString&)`: Этот метод вызывается при нажатии кнопки “Settings”. Он создает диалоговое окно для настройки параметров игры, таких как цвет змейки и громкость музыки;
6. `createRegistrationDialog()`: Этот метод создает диалоговое окно для регистрации нового пользователя. Он проверяет валидность введенных данных и сохраняет их в файле, если они допустимы;
7. `createAuthorizationDialog()`: Этот метод создает диалоговое окно для авторизации пользователя. Он проверяет валидность введенных данных и загружает данные пользователя из файла, если они допустимы;
8. `openImageWindow()`: Этот метод вызывается при нажатии на изображение в главном окне. Он создает новое окно с информацией об авторизованном пользователе и таблицей рекордов;
9. `showMainWindow()`: Этот метод вызывается при нажатии кнопки “Back” в окне игры. Он закрывает окно игры и отображает главное окно.

3.3.2 Класс CustomTitleBar

Класс `CustomTitleBar` представляет собой пользовательскую панель заголовка окна.

Поля класса CustomTitleBar:

`QPoint m_dragPosition` - поле, которое хранит позицию курсора мыши при перетаскивании окна.

Методы класса CustomTitleBar:

1. Конструктор `(CustomTitleBar::CustomTitleBar (QWidget *parent))`: Это конструктор класса `CustomTitleBar`. Он настраивает пользовательский интерфейс для панели заголовка и устанавливает фиксированную высоту для панели;
2. `mousePressEvent(QMouseEvent *event)`: Этот метод вызывается при нажатии кнопки мыши на панели заголовка. Он сохраняет текущую позицию курсора;
3. `mouseMoveEvent(QMouseEvent *event)`: Этот метод вызывается при перемещении мыши по панели заголовка. Если кнопка мыши зажата, он перемещает окно в новую позицию курсора.

3.3.3 Класс GameWindow

Класс `GameWindow` представляет собой окно игры.

Поля класса GameWindow:

1. `MainWindow *m_mainWindow` - поле, которое хранит указатель на объект класса `MainWindow`, который представляет собой главное окно приложения;
2. `GameField *m_gameField` - поле, которое хранит указатель на объект класса `GameField`, который представляет собой игровое поле для змейки;
3. `HelpField *m_helpField` - поле, которое хранит указатель на объект класса `HelpField`, который представляет собой поле с подсказками для игры;
4. `QWidget *m_centralWidget` - поле, которое хранит указатель на объект класса `QWidget`, который используется как центральный виджет окна игры;
5. `QGridLayout *m_gridLayout` - поле, которое хранит указатель на объект класса `QGridLayout`, который используется для размещения виджетов в окне игры.

Методы класса GameWindow:

1. Конструктор `(GameWindow::GameWindow(MainWindow *mainWindow, QWidget *parent))`: Это конструктор класса GameWindow. Он принимает указатель на MainWindow и настраивает пользовательский интерфейс для окна игры;
2. `handleAltPressed()`: Этот метод вызывается при нажатии клавиши Alt. Он закрывает окно игры и открывает главное окно.

3.3.4 Класс SoundButton

Класс `SoundButton` представляет собой настраиваемую кнопку `QPushButton`, которая может воспроизводить звук при наведении курсора.

Поля класса SoundButton:

`static bool soundEnabled` - статическое поле, которое хранит состояние включения или выключения звука.

Методы класса SoundButton:

1. Конструктор `(SoundButton::SoundButton(const QString &text, QWidget *parent))`: Это конструктор класса SoundButton. Он принимает текст кнопки и указатель на родительский виджет в качестве параметров;
2. `enterEvent(QEvent *event)`: Этот метод вызывается при наведении курсора на кнопку. Если звук включен, он воспроизводит звук.

3.3.5 Класс HelpField

Класс `HelpField` представляет собой виджет, который отображает текстовую информацию.

Поля класса HelpField:

`QString m_text` - поле, которое хранит текст, который выводится на поле.

Методы класса HelpField:

1. Конструктор `(HelpField::HelpField())`: Это конструктор класса `HelpField`. Он инициализирует текстовое поле;
2. `paintEvent(QPaintEvent *e)`: Этот метод вызывается при обновлении виджета. Он перерисовывает текстовое поле;
3. `ChangeTextSlot(QString text)`: Этот слот вызывается при получении сигнала `ChangeTextSignal` от другого объекта. Он изменяет текст в текстовом поле.

3.3.6 Класс ItemElements

Класс `ItemElements` представляет собой базовый класс для элементов игры.

Поля класса ItemElements:

1. `int m_x` - поле, которое хранит координату `x` элемента на игровом поле;
2. `int m_y` - поле, которое хранит координату `y` элемента на игровом поле.

Методы класса ItemElements:

Конструктор `(ItemElements::ItemElements(int x, int y))`: Это конструктор класса `ItemElements`. Он принимает координаты x и y в качестве параметров и инициализирует положение элемента.

3.3.7 Класс Snake

Класс `Snake` представляет собой змейку в игре.

Поля класса Snake:

1. `QList<ItemElements*> m_SnakeBody` - поле, которое хранит список указателей на объекты класса ItemElements, которые составляют тело змейки;
2. `int m_SnakeBeginSize` - поле, которое хранит начальный размер змейки в элементах;
3. `enum SnakeDirection` - перечисление, которое определяет возможные направления движения змейки: `up`, `left`, `down`, `right`;
4. `SnakeDirection m_snakeDirection` - поле, которое хранит текущее направление движения змейки.

Методы класса Snake:

1. Конструктор `(Snake::Snake())`: Это конструктор класса `Snake`. Он инициализирует змейку;
2. `getBody()`: Этот метод возвращает тело змейки в виде списка элементов.

3.3.8 Класс SnakeFood

Класс `SnakeFood` представляет собой еду для змейки.

Поля класса SnakeFood:

1. `int m_x` - поле, которое хранит координату x еды на игровом поле. Наследуется от класса `ItemElements`;
2. `int m_y` - поле, которое хранит координату y еды на игровом поле. Наследуется от класса `ItemElements`;

Методы класса SnakeFood:

1. Конструктор `(SnakeFood::SnakeFood(int x, int y))`: Это конструктор класса `SnakeFood`. Он принимает координаты x и y в качестве параметров и инициализирует положение еды;
2. `CreateFood(Snake* m_snake, int m_fieldSize)`: Этот метод создает новую еду для змейки в случайном месте на игровом поле, которое не занято змейкой.

3.3.9 Класс SnakeBonus

Класс `SnakeBonus` представляет собой бонус для змейки.

Поля класса SnakeBonus:

1. `int m_x` - поле, которое хранит координату x бонуса на игровом поле. Наследуется от класса `ItemElements`;
2. `int m_y` - поле, которое хранит координату y бонуса на игровом поле. Наследуется от класса `ItemElements`;

3. `bool m_isSpeedUp` - поле, которое хранит тип бонуса: `true` - ускорение, `false` - замедление.

Методы класса SnakeBonus:

1. Конструктор `(SnakeBonus::SnakeBonus(int x, int y, bool isSpeedUp))`: Это конструктор класса `SnakeBonus`. Он принимает координаты x и y и булево значение `isSpeedUp` в качестве параметров и инициализирует положение и тип бонуса;

2. `CreateBonus(Snake* m_snake, int m_fieldSize, SnakeFood* m_food, SnakeBonus *&m_bonus, QTimer *m_bonusTimer)`: Этот метод создает новый бонус для змейки в случайном месте на игровом поле, которое не занято змейкой или едой. Бонус может быть либо ускорением, либо замедлением. Бонус исчезает после определенного времени, если его не съели.

3.3.10 Класс GameField

Класс `GameField` представляет собой игровое поле.

Поля класса GameField:

1. `QString color_snake` - поле, которое хранит цвет змейки, выбранный пользователем;

2. `Snake *m_snake` - поле, которое хранит указатель на объект класса `Snake`, который представляет собой змейку;

3. `SnakeFood *m_food` - поле, которое хранит указатель на объект класса `SnakeFood`, который представляет собой еду для змейки;

4. `SnakeBonus *m_bonus` - поле, которое хранит указатель на объект класса `SnakeBonus`, который представляет собой бонус для змейки;

5. `int m_sizeItem` - поле, которое хранит размер одного элемента змейки или еды в пикселях;

6. `int m_fieldSize` - поле, которое хранит размер игрового поля в элементах;

7. `int m_score` - поле, которое хранит текущий счет игры;

8. `int m_bonusTimeLeft` - поле, которое хранит оставшееся время действия бонуса в миллисекундах;

9. `int snakeColor` - поле, которое хранит код цвета змейки;

10. `QTimer *m_moveSnakeTimer` - поле, которое хранит указатель на объект класса `QTimer`, который используется для управления скоростью движения змейки;

11. `QTimer *m_bonusTimer` - поле, которое хранит указатель на объект класса `QTimer`, который используется для управления появлением и исчезновением бонуса;

12. `bool m_isPause` - поле, которое хранит состояние паузы игры;

13. `bool m_isMoveBlocked` - поле, которое хранит состояние блокировки движения змейки;

14. `bool m_isGameOver` - поле, которое хранит состояние окончания игры.

Методы класса GameField:

1. Конструктор `(GameField::GameField())`: Это конструктор класса `GameField`. Он инициализирует игровое поле и создает змейку, еду и бонусы;

2. `paintEvent(QPaintEvent *e)`: Этот метод вызывается при обновлении виджета. Он перерисовывает игровое поле;

3. `keyPressEvent(QKeyEvent *e)`: Этот метод вызывается при нажатии клавиши на клавиатуре. Он обрабатывает управление змейкой;

4. `MoveSnakeSlot()`: Этот слот вызывается при каждом шаге таймера. Он перемещает змейку на одну клетку в текущем направлении;

5. `RemoveBonus()`: Этот слот вызывается при истечении времени действия бонуса. Он удаляет бонус с игрового поля;

6. `ExitToMenu()`: Этот метод вызывается при завершении игры. Он закрывает окно игры и открывает главное меню;

7. `SetGameStatus()`: Этот метод вызывается для установки статуса игры (пауза, игра, конец игры);

8. `GameOver()`: Этот метод вызывается при проигрыше в игре. Он отображает сообщение о проигрыше и предлагает начать новую игру или выйти в меню;

9. `StartNewGame()`: Этот метод вызывается для начала новой игры. Он сбрасывает игровое поле и создает новую змейку, еду и бонусы;

10. `GameOverAndGoInMenu`(): Этот метод вызывается при проигрыше в игре. Он отображает сообщение о проигрыше и возвращает пользователя в главное меню;

11. `updateScore(int gameScore)`: Этот метод вызывается для обновления счета игры. Он принимает новый счет в качестве параметра и обновляет отображаемый счет;

12. `changeSnakeColor(int color)`: Этот слот вызывается для изменения цвета змейки. Он принимает новый цвет в качестве параметра и изменяет цвет змейки.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов

Метод `CreateBonus(Snake* m_snake, int m_fieldSize, SnakeFood* m_food, SnakeBonus *&m_bonus, QTimer *m_bonusTimer)` отвечает за перемещение создание бонусов для змейки. Схема метода `CreateBonus(Snake* m_snake, int m_fieldSize, SnakeFood* m_food, SnakeBonus *&m_bonus, QTimer *m_bonusTimer)` показана в приложении Б.

Метод `keyPressEvent(QKeyEvent *e)` обрабатывает нажатия клавиш для управления змейкой и паузы в игре. Схема метода `keyPressEvent(QKeyEvent *e)` показана в приложении В.

4.2 Разработка алгоритмов

4.2.1 Алгоритм для начала игры “Змейка”

1. Запуск программы: При запуске программы отображается главное окно приложения MainWindow, которое содержит кнопки “PLAY”, “ ENTER NICKNAME ”, “SETTINGS” и “QUIT THE GAME”.

2. Ввод имени пользователя: Пользователь может ввести свое имя пользователя, нажав на кнопку “ENTER NICKNAME”. Открывается диалоговое окно для ввода имени пользователя. После ввода имени пользователя и нажатия кнопки “OK”, имя пользователя сохраняется.

3. Настройка игры: Пользователь может выбрать цвет змейки, нажав на кнопку “SETTINGS”. Открывается диалоговое окно, где можно выбрать цвет змейки из предложенных вариантов.

4. Начало игры: Пользователь нажимает кнопку “PLAY”, чтобы начать игру. Создается новое окно игры GameWindow, главное окно скрывается. На игровом поле появляется змейка и еда.

5. Игровой процесс: Змейка автоматически движется в текущем направлении. Пользователь может изменить направление змейки, нажав клавиши со стрелками на клавиатуре. Если змейка сталкивается со стеной или собственным телом, игра заканчивается, и отображается сообщение о проигрыше.

6. Еда: На игровом поле случайным образом появляется еда. Если змейка съедает еду, она увеличивается в размере, и пользователь получает очки. После того как еда съедена, на игровом поле появляется новая еда.

7. Бонусы: На игровом поле случайным образом появляются бонусы. Если змейка съедает бонус, она может временно ускориться или замедлиться. После определенного времени бонус исчезает.

8. Конец игры: Если змейка сталкивается со собственным телом, игра заканчивается. Отображается сообщение о проигрыше и предлагается начать новую игру или выйти в меню.

9. Возврат в меню: Пользователь может вернуться в главное меню в любое время, нажав клавишу Alt. Текущая игра прерывается, и отображается главное окно.

4.2.2 Алгоритм регистрации пользователя

1. Пользователь нажимает кнопку “Enter nickname”, чтобы ввести свое имя пользователя. Открывается диалоговое окно для ввода имени пользователя.

2. Пользователь вводит свое имя пользователя и его пароль в текстовые поля после чего нажимает кнопку “Save”.

3. Если имя пользователя допустимо (не пустое и не содержит недопустимых символов), оно сохраняется в файле.

4. Если имя пользователя недопустимо, отображается сообщение об ошибке, и пользователь может попробовать ввести имя пользователя снова.

5. После успешной регистрации имя пользователя отображается в главном окне. Пользователь теперь может начать игру с этим именем пользователя.

6. Если пользователь нажал кнопку “Back” в диалоговом окне регистрации, диалоговое окно закрывается, и имя пользователя не сохраняется. Пользователь возвращается в главное меню.

7. Если пользователь закрыл диалоговое окно регистрации, не вводя имя пользователя, отображается сообщение об ошибке, и пользователь может попробовать ввести имя пользователя снова.

8. Имя пользователя используется для отображения в таблице рекордов и для сохранения прогресса игры.

5 РЕЗУЛЬТАТ РАБОТЫ

При запуске игры “Змейка” пользователю откроется окно главного меню.

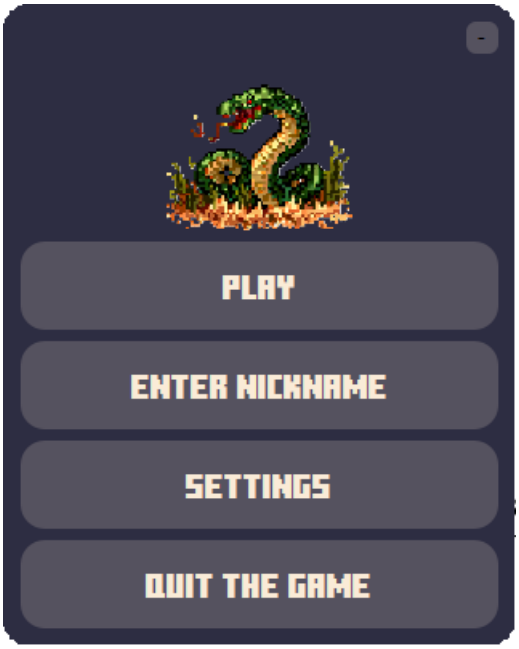

Рисунок 5.1 – Окно главного меню программы

В окне главного меню можно нажать на изображение и четыре кнопки, а именно: "PLAY", "ENTER NICKNAME", "SETTINGS" и "QUIT THE GAME".

При нажатии на изображение пользователю откроется окно с плашкой, в которой отображается авторизированный пользователь и его счёт, а также кнопка "LOGOUT", по нажатию на которую пользователь выйдет из своего аккаунта. Также в этом окне будет отображаться "scrollable window", в котором можно просмотреть всех зарегистрированных пользователей и их максимальные результаты. В нижней части окна есть кнопка "Back", по нажатию на которую пользователя вернёт в окно главного меню.

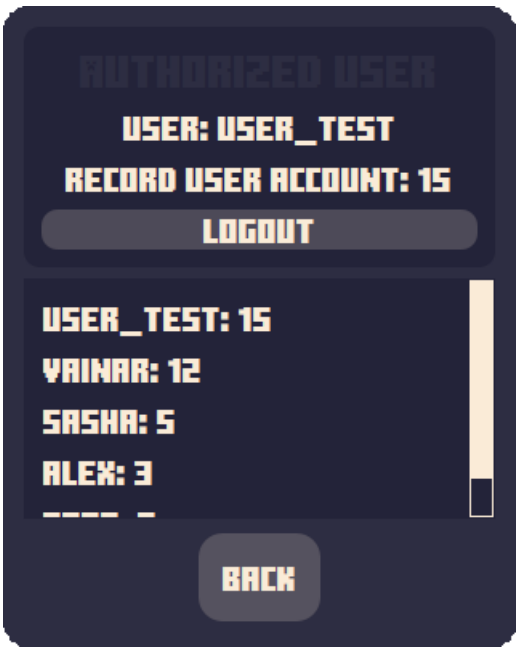

Рисунок 5.2 – Окно, открывающееся при нажатии на изображение

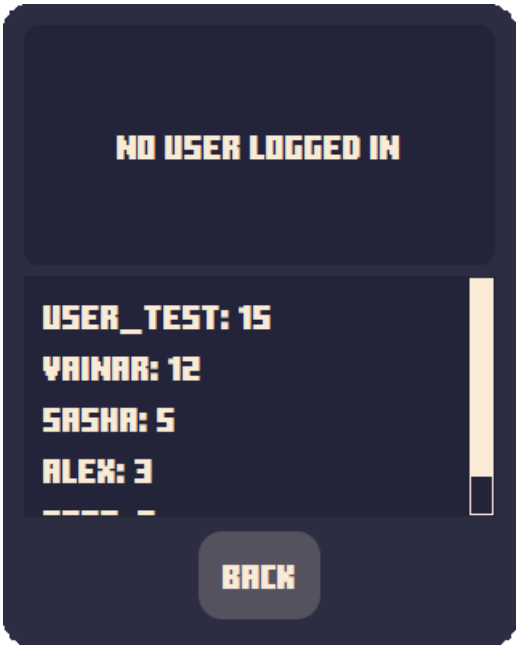

Рисунок 5.3 – Изменение окна с отображением информации пользователях, при нажатии на кнопку “LOGOUT”

При нажатии на кнопку "PLAY" пользователю откроется окно с игрой. В окне с игрой, по нажатию на клавиши "стрелок", а именно: ↑ , ↓ , ← , →, пользователь может управлять "змейкой". Также, по нажатию клавиши SPACE, пользователь может поставить игру на паузу и по такому же принципу вернуться к игре. По нажатии клавиши Alt пользователь закончит игру, и его вернёт в окно главного меню. Если пользователь авторизован, то при выходе из игры его счёт будет перезаписан как новый, только если текущий счёт

превышает его максимальный счёт за всё время существования аккаунта. Если в игре никто не авторизовался, счёт игры никуда не будет записан.

Рисунок 5.4 – окно, с игровым полем

Рисунок 5.5 – окно, с игровым полем в режиме паузы

Рисунок 5.6 – окно, с игровым полем после того как пользователь проиграл

При нажатии на кнопку "ENTER NICKNAME" появится окно с тремя кнопками: "REGISTRATION", "AUTHORIZATION" и "BACK".

При нажатии на кнопку "BACK" пользователя вернёт в окно главного меню.

При нажатии на кнопку "REGISTRATION" пользователю откроется окно с двумя полями для имени и пароля, кнопками "SAVE" и "BACK". При нажатии на кнопку "BACK" пользователя вернет к окну "ENTER NICKNAME". При нажатии кнопки "SAVE" введенные пользователем данные в полях имени и пароля сохраняются в качестве нового пользователя. Однако, кнопка "SAVE" может не сработать, если пользователь с таким именем уже существует или если в поле имени или пароля присутствует символ пробела. В таких случаях пользователю будет сообщена ошибка.

При нажатии кнопки "AUTHORIZATION" пользователю откроется окно с двумя полями для имени и пароля, кнопками "SAVE" и "BACK". При нажатии на кнопку "BACK" пользователя вернет к окну "ENTER NICKNAME". При нажатии кнопки "SAVE" данные, введённые пользователем в полях имени и пароля, сохранятся как авторизованный пользователь. Если в поле имени или пароля присутствует символ пробела или одно из полей (или оба поля) остаются незаполненными, кнопка "SAVE" может не сработать. В таких случаях пользователю будет сообщено об ошибке.

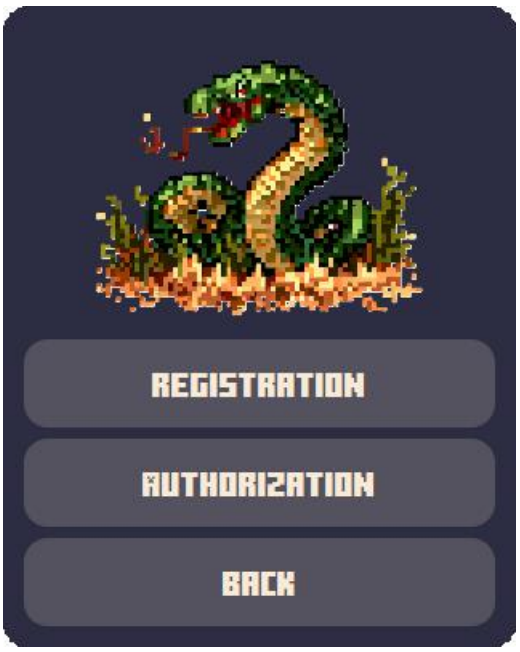

Рисунок 5.7 – окно, появляющееся при нажатии на кнопку “ENTER NICKNAME”

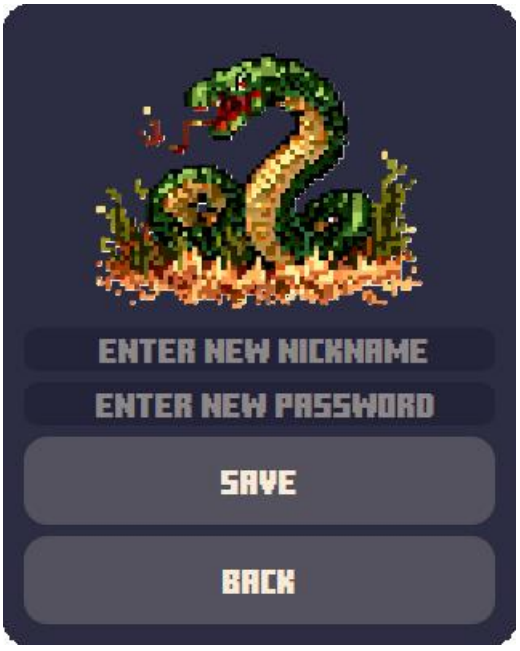

Рисунок 5.8 – окно, появляющееся при нажатии в окне “ENTER NICKNAME” на кнопку “REGISTRATION”

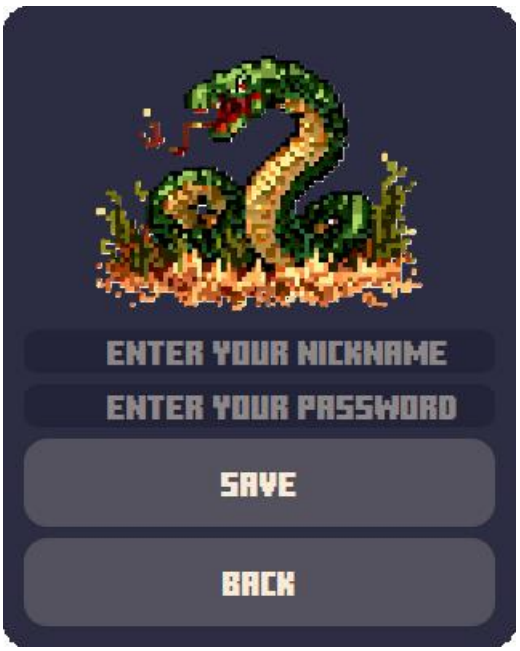

Рисунок 5.9 – окно, появляющееся при нажатии в окне “ENTER NICKNAME” на кнопку “AUTHORIZATION”

При нажатии в главном меню кнопки "SETTINGS" откроется окно настроек, в котором можно отрегулировать громкость музыки, цвет змейки и вкл/выкл звука от кнопок при наведении на них курсора.

Громкость музыки регулируется ползунком, который пользователь может двигать в двух направлениях: вправо – сделать громче, влево – сделать тише. Цвет змейки меняется через выпадающее окно. Звук от кнопок включается и выключается с помощью кнопки.

Также в окне настроек есть кнопки "BACK" и "SAVE". По кнопке "BACK" пользователя вернёт к окну главного меню без сохранения настроек. По кнопке "SAVE" настройки, изменённые пользователем, сохранятся, и пользователя также вернёт к окну главного меню.

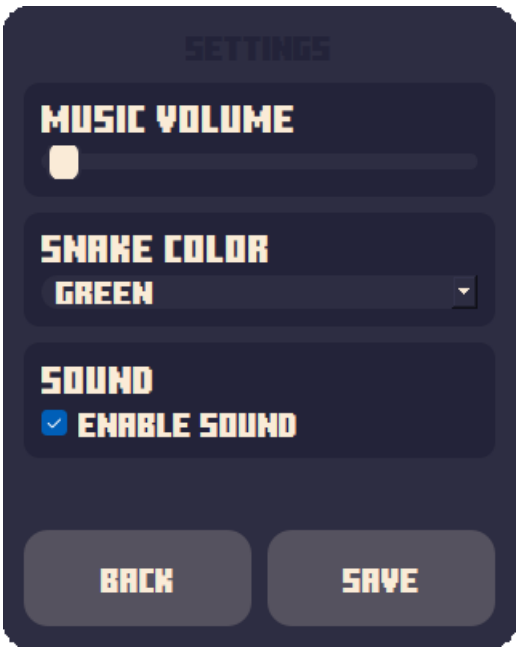

Рисунок 5.10 – Окно настроек

ЗАКЛЮЧЕНИЕ

В рамках данной работы было успешно разработана игра "Змейка" с использованием фреймворка Qt. Это приложение предлагает пользователям возможность играть в классическую игру "Змейка" с дополнительными функциями, такими как выбор цвета змейки и регистрация пользователя.

Для обеспечения удобства взаимодействия с пользователем были использованы различные виджеты Qt для отображения игрового поля и интерфейса. Была также реализована функциональность по работе с файлами для сохранения данных пользователя.

В ходе работы были применены различные методы для достижения поставленных целей. Вначале был проведен анализ требований к функционалу и характеристикам приложения. Затем были разработаны алгоритмы на языке программирования C++, которые управляют игровым процессом и обрабатывают данные.

В итоге разработанное приложение успешно реализует задуманный функционал и предоставляет интересный и увлекательный игровой процесс. Благодаря использованию фреймворка Qt, приложение обладает приятным и интуитивно понятным интерфейсом, что делает его доступным для широкого круга пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Рожнова, Н. Г. Вычислительные машины, системы и сети. Дипломное проектирование : учебно-метод.пособие / Н. Г. Рожнова, Н. А. Искра, И. И. Глецевич. – Минск : БГУИР, 2014. – 96 с. : ил.

[2] СТП 01–2013. Дипломные проекты (работы): общие требования. – Введ. [Электронный ресурс]. – 2013. – Режим доступа: http://www.bsuir.by/online/showpage.jsp.

[3] Единая система конструкторской документации (ЕСКД): справ. пособие / С. С. Борушек [и др.]. – М. : Изд-во стандартов, 1989. – 352 с

[4] Шлее М. - Qt4. Профессиональное программирование на C+/ Шлее М. - Л.:Наука, 2013. - 770 с.

[5] Бланшет Ж., Саммерфилд М. - QT 4: программирование GUI на C+/ Бланшет Ж., Саммерфилд М. М. — М. : ALT Linux, 2015. — 948 с. : ил.

[6] А. В.Чеботарев Библиотека Qt 4. Программирование прикладных приложений в среде Linux / А. В.Чеботарев - Л.: Наука, 2013. - 821 с.

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

[8] Дейтел, Х. Как программировать на C++ / Х. Дейтел, П. Дейтел. - М. : БИНОМ, 2001. - 1152 с.

ПРИЛОЖЕНИЕ А
(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б
(обязательное)

Блок-схема метода

ПРИЛОЖЕНИЕ В
(обязательное)

Блок-схема метода

ПРИЛОЖЕНИЕ Г
(обязательное)

Код программы