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 курсовому проекту
на тему

ИГРА “2048”

БГУИР КП 1-40 02 01 228 ПЗ

Студент: гр. 250502 Хилько И. А.

Руководитель: Богдан Е. В.

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 *Хилько Ивану Александровичу*
Тема проекта *Игра “2048”*
2. Срок сдачи студентом законченного проекта *15 декабря 2023 г.*
3. Исходные данные к проекту *Язык программирования – C++, среда разработки – Qt-Creator*
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)
1. Лист задания.
2. Введение.
3. Обзор литературы.
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 метода isFull ()
*3. Схема алгоритма метода play ()*_____

6. Консультант по проекту (с обозначением разделов проекта) Богдан Е. В.
7. Дата выдачи задания ———*15.09.2023г.*———————————— –
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 15.12.23 – 10 %
Защита курсового проекта с 21.12 по 28.12.23г.———————————

РУКОВОДИТЕЛЬ Богдан Е. В.

(подпись)

Задание принял к исполнению _____ _____ __ ____

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Для данного курсового проекта был выбран язык программирования C++ и фреймворке Qt. C++ - компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ добавляет к C объектно-ориентированные возможности. Он вводит классы, которые обеспечивают три самых важных свойства ООП: инкапсуляцию, наследование и полиморфизм.

В C++ при наследовании одного класса от другого наследуется реализация класса, плюс класс-наследник может добавлять свои поля и функции или переопределять функции базового класса. Множественное наследование разрешено.

Конструктор наследника вызывает конструкторы базовых классов, а затем конструкторы нестатических членов-данных, являющихся экземплярами классов. Деструктор работает в обратном порядке.

Наследование бывает публичным, защищённым и закрытым.

Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных.

Преимуществом полиморфизма является то, что он помогает снижать сложность программ, разрешая использование того же интерфейса для задания единого класса действий. Выбор же конкретного действия, в зависимости от ситуации, возлагается на компилятор. Полиморфизм может применяться также и к операторам.

Основным способом организации информации в C++ являются классы. В отличие от структуры (struct) языка C, которая может состоять только из полей и вложенных типов, класс (class) C++ может состоять из полей, вложенных типов и функций-членов. Инкапсуляция в C++ реализуется через указание уровня доступа к членам класса: они бывают публичными (public), защищёнными (protected) и закрытыми (private). В C++ структуры отличаются от классов тем, что по умолчанию члены и базовые классы у структуры публичные, а у класса — собственные.

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 развлекательных приложений.

Для разработки графического интерфейса и создания кроссплатформенных приложений решающим фактором стал выбор фреймворка Qt.

Qt предоставляет обширный набор инструментов для построения современных и интуитивно-понятных пользовательских интерфейсов. Его кроссплатформенность позволяет создавать приложения, работающие на различных операционных системах без необходимости переписывать код.

Система сигналов и слотов в Qt обеспечивает эффективную обработку событий и взаимодействие между компонентами приложения, а богатая стандартная библиотека Qt дополняет функциональность C++, предоставляя готовые решения для ряда типичных задач.

Таким образом, выбор C++ и Qt в данной курсовой работе обусловлен их выдающейся функциональностью, универсальностью и возможностью эффективного взаимодействия, что в совокупности делает их оптимальным инструментарием для реализации поставленных задач.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Анализ существующих аналогов

“2048” является достаточно популярной логической игрой.

Существуют множество подобных логических и головоломных игр существует как на мобильных устройствах, так и на компьютере. Вот несколько аналогов, которые также предоставляют увлекательный геймплей и требуют стратегического мышления.

1.1.1 Игра "Threes!"

- Суть игры: Объединение плиток с одинаковыми числами, начиная с 1 и 2, чтобы создавать более высокие числа.
- Структура: Квадратное игровое поле с плитками, представляющими числа.
- Цель игры: Создать плитку с числом 6144.
- Правила: Двигайте все плитки в одном направлении за один ход, объединяйте одинаковые числа.
- Как играть: Сдвигайте плитки, стремясь создать более высокие числа и освободить место на поле.
- Как победить: Побеждаете, создав плитку с числом 6144.
- Как проиграть: Игра заканчивается, когда поле полностью заполнено, и нет возможности сделать ходов.

Пример игры "Threes!" показан на рисунке 1.1.

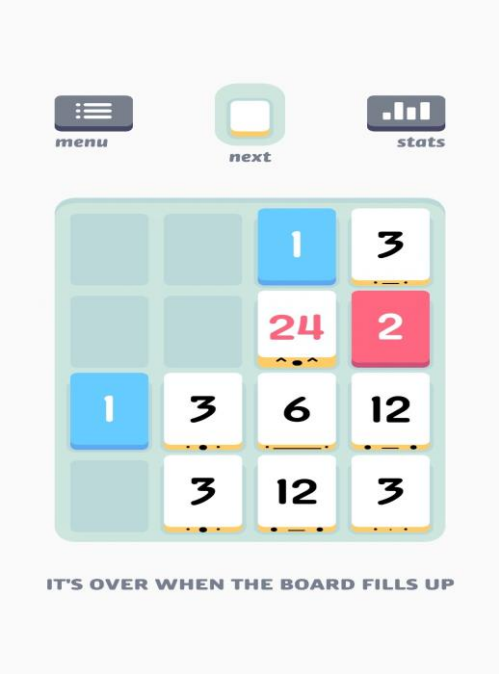

Рисунок 1.1 – Пример игры "Threes!"

1.1.2 Игра "1024!"

- Суть игры: Объединение плиток, чтобы создавать числа и достичь 1024.
- Структура: Квадратное игровое поле с плитками, представляющими числа.
- Цель игры: Создать плитку с числом 1024.
- Правила: Двигайте все плитки в одном направлении за один ход, объединяйте одинаковые числа.
- Как играть: Сдвигайте плитки, стремясь объединять числа и создавать плитку с числом 1024.
- Как победить: Побеждаете, создав плитку с числом 1024.
- Как проиграть: Игра завершается, когда поле полностью заполнено, и нет возможности объединить плитки.

Пример игры "1024!" показан на рисунке 1.2.

Рисунок 1.2 – Пример игры "1024!"

1.1.3 Игра "Fibonacci"

- Суть игры: Объединение чисел Фибоначчи.

- Структура: Квадратное игровое поле с плитками, представляющими числа Фибоначчи.
- Цель игры: Создать следующее число Фибоначчи в последовательности.
- Правила: Двигайте все плитки в одном направлении за один ход, объединяйте одинаковые числа.
- Как играть: Сдвигайте плитки, следуя последовательности чисел Фибоначчи.
- Как победить: Побеждаете, создав плитку, соответствующую следующему числу в последовательности Фибоначчи.
- Как проиграть: Игра заканчивается, когда поле полностью заполнено, и нет возможности сделать ходов.

Пример игры "Fibonacci" показан на рисунке 1.3.

Рисунок 1.3 – Пример игры "Fibonacci"

1.1.4 Игра "2048 Multiplayer"

- Суть игры: Состязание с другими игроками в реальном времени, объединяя плитки с одинаковыми числами.
- Структура: Квадратное игровое поле с плитками, представляющими числа.

- Цель игры: Победить, создав плитку с числом 2048 или заработав больше очков за ограниченное время.
- Правила: Двигайте все плитки в одном направлении за один ход, объединяйте одинаковые числа.
- Как играть: Сдвигайте плитки, соревнуясь с другими игроками.
- Как победить: Выигрываете, создав плитку с числом 2048 или заработав больше очков за отведенное время.
- Как проиграть: Игра завершается, когда поле полностью заполнено, и нет возможности сделать ходов.

Пример игры "2048 Multiplayer" показан на рисунке 1.4.

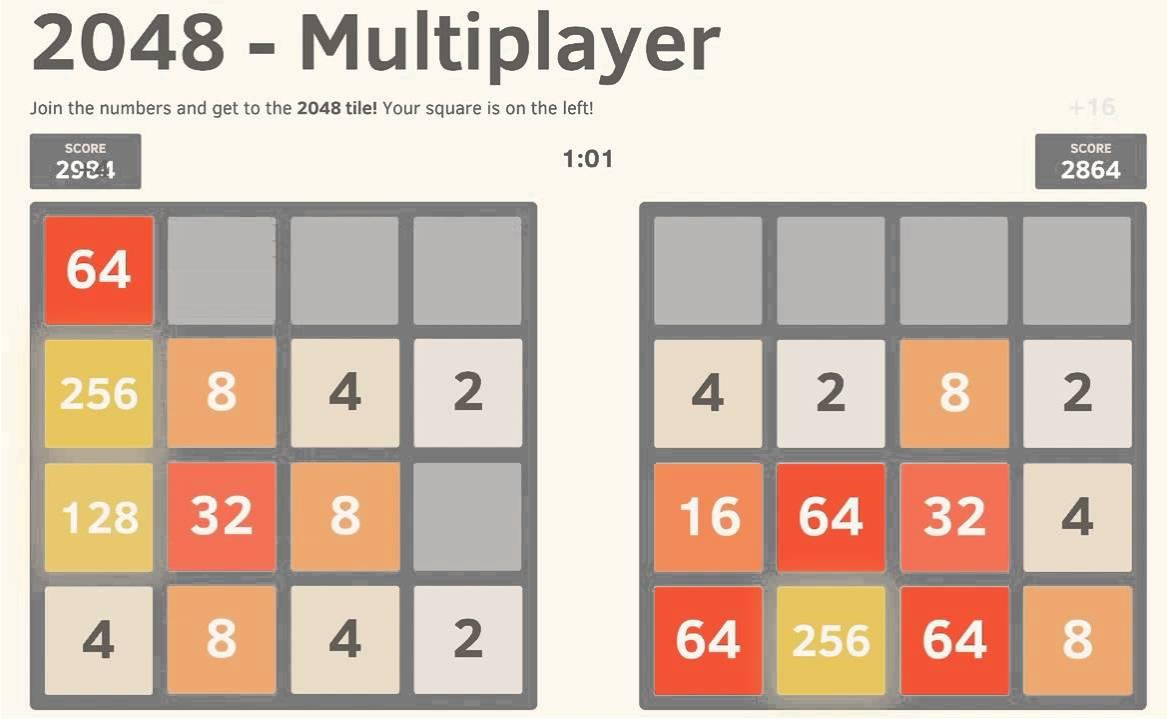

Рисунок 1.4 – Пример игры "2048 Multiplayer"

1.1.5 Игра "Doge 2048"

- Суть игры: Объединение изображений Doge, аналогично 2048.
- Структура: Квадратное игровое поле с плитками, представляющими изображения Doge.
- Цель игры: Создать плитку с числом 2048, представляющую изображение Doge.
- Правила: Двигайте все плитки в одном направлении за один ход, объединяйте одинаковые изображения Doge.
- Как играть: Сдвигайте плитки, стремясь создать плитку с числом 2048 и изображением Doge.
- Как победить: Побеждаете, создав плитку с числом 2048 и изображением Doge.

- Как проиграть: Игра заканчивается, когда поле полностью заполнено, и нет возможности сделать ходов.

Пример игры "Doge 2048" показан на рисунке 1.5.

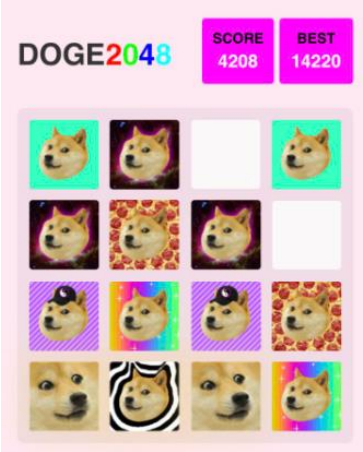

Рисунок 1.5 – Пример игры "Doge 2048"

1.1.6 Игра "2048 Hex"

- Суть игры: Объединение плиток на шестиугольной сетке, аналогично 2048.
- Структура: Шестиугольная игровая сетка с плитками, представляющими числа.
- Цель игры: Создать плитку с числом 2048.
- Правила: Двигайте шестиугольные плитки в одном направлении за один ход, объединяйте одинаковые числа.
- Как играть: Сдвигайте плитки, стремясь создать плитку с числом 2048 на шестиугольной сетке.
- Как победить: Побеждаете, создав плитку с числом 2048.
- Как проиграть: Игра завершается, когда поле полностью заполнено, и нет возможности сделать ходов.

Пример игры "2048 Hex" показан на рисунке 1.6.

Рисунок 1.6 – Пример игры "2048 Hex"

1.1.7 Игра "Super 2048"

- Суть игры: Расширенная версия 2048 с более крупным полем и дополнительными числами.
- Структура: Квадратное игровое поле с плитками, представляющими числа.
- Цель игры: Создать плитку с максимальным числом.
- Правила: Двигайте все плитки в одном направлении за один ход, объединяйте одинаковые числа.
- Как играть: Сдвигайте плитки, стараясь создать как можно более высокие числа.
- Как победить: Побеждаете, создав плитку с максимальным числом.

Как проиграть: Игра заканчивается, когда поле полностью заполнено, и нет возможности сделать ходов.
Пример игры "Super 2048" показан на рисунке 1.7.

Рисунок 1.7 – Пример игры "Super 2048"

Сейчас мы более подробно рассмотрим классическую игру “2048”.
Суть игры: 2048 - это головоломка, в которой игрок перемещает плитки по игровому полю, объединяя одинаковые числа и стремясь создать плитку с числом 2048. Игра основана на логике и стратегии, и требует от игрока принятия решений о том, в каком направлении двигать плитки, чтобы достичь максимального возможного числа.
Структура:

- Игровое поле представляет собой квадрат, поделенный на ячейки.
- В начале игры на поле располагаются две плитки с числами 2 или 4.

Цель игры: Создать плитку с числом 2048, объединяя плитки с одинаковыми числами.

Правила:

1. Игрок может сдвигать все плитки на игровом поле в одном из четырех направлений: вверх, вниз, влево или вправо.
2. Плитки с одинаковыми числами объединяются при сдвиге в одном направлении.
3. После каждого хода на пустой ячейке появляется новая плитка с числом 2 или 4.
4. Цифры на плитках всегда являются степенями двойки (2, 4, 8, 16, и так далее).

Как играть:

1. Используйте клавиши управления (вверх, вниз, влево, вправо) или свайпы (на мобильных устройствах) для перемещения плиток.
2. Плитки с одинаковыми числами объединяются, если они соприкасаются при сдвиге.
3. Старайтесь строить числа, удваивая их, пока не создадите плитку с числом 2048.

Как победить: Побеждаете, когда создаете плитку с числом 2048 на игровом поле.

Как проиграть: Игра завершается, когда поле полностью заполнено, и нет возможности сделать новый ход. Проигрываете, если больше нет доступных ходов.

Пример игры "2048" показан на рисунке 1.8.

Рисунок 1.8 – Пример игры "2048"

1.2 Обзор методов и алгоритмов решения поставленной задачи

Для создания игры “2048” был использован фреймворк Qt. Он содержит стандартные библиотеки C++, а также набор инструментов для создания удобного графического интерфейса.

Система сигналов и слотов в Qt обеспечивает не только эффективную обработку событий, но и быстрое взаимодействие между компонентами приложения. Дополняя функциональность C++, богатая стандартная библиотека Qt предоставляет готовые решения для различных типичных задач, что делает этот фреймворк незаменимым инструментарием для разработки.

Мой выбор C++ и Qt в данной курсовой работе обусловлен не только их выдающейся функциональностью, универсальностью и эффективностью, но и способностью этих технологий эффективно взаимодействовать друг с другом.

Игрок может совершать действия, нажимая на различные кнопки. В моем проекте реализованы три окна: главное окно, окно для выборов параметра игры, окно игры.

В моей версии игры разработаны разные методы , например:
представление игрового поля, генерация новых чисел, логика перемещения плиток, проверка завершения игры, графический интерфейс и обработка пользовательских действий.

2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

2.1 Структура входных и выходных данных

В начале игры пользователь сразу находится в окне игры, где он видит игровое поле , свой счет игры и лучший результат игры.

Это окно показано на рисунке 2.1.

Рисунок 2.1 – Окно игры

Начальные данные у игроков такие: счет очков равен 0 , лучшего результата нет, а на игровом поле находятся две плитки номиналом 2 или 4.

2.2 Разработка диаграммы классов

Диаграмма классов данной курсовой работы приведена в приложении А.

2.3 Описание классов

2.3.1 Класс игры

Поля класса `Game`:

`Board *board` - Объект класса `Board`

`bool gameOver` - Флаг, указывающий, завершена ли игра (true, если игра завершена, false в противном случае).

`string line` - Поле строкового типа.

`int finish` - Числовое значения Цели игры.

`int score` - Текущий счет игры.

`int best` - Лучший результат (набранные очки) игры.

Методы класса `Game`:

`Game(int dimension, int target)` - Конструктор класса.

`Board* getGameBoard() const` - Метод, возвращающий указатель на объект класса `Board`, представляющий игровое поле.

`void restart()` - Метод для перезапуска игры.

`void playerMove(DIRECTION dir)` - Метод для обработки движения игрока в указанном направлении (`dir`).

`int getScore() const` - Метод для получения текущего счета игры.

`int getBest() const` - Метод для получения лучшего результата (максимального счета) из предыдущих игр.

`bool isGameOver() const` - Метод для проверки, завершена ли игра.

`bool isFinished() const` - Метод для проверки, достигнута ли цель игры.

2.3.2 Класс игрового поля

Описание полей класса `Board`:

`QVector< QVector<Tile*> > board` - Поле, представляющее собой двумерный вектор (матрицу) плиток.

`int dimension` - Поле, содержащее размерность игрового поля.

`int lastPoints` - Поле, хранящее количество очков, заработанных в последнем ходе.

`bool lastCollision` - Поле, указывающее, произошла ли последняя коллизия при движении плиток

Методы класса `Board`:

`Board(int dimension)` - Конструктор класса.

`Board(const Board &brd)` - Конструктор копирования.

`~Board()` - Деструктор класса.

`Tile* getTile(int i, int j)` - Метод, возвращающий указатель на плитку по указанным координатам (`i, j`) на игровом поле.

`void move(DIRECTION direction)` - Метод, осуществляющий движение плиток на игровом поле в указанном направлении.

`void reset()` - Метод, сбрасывающий состояние игрового поля.int `getDimension() const` - Метод, возвращающий размерность игрового поля.

`getPoints() const` - Метод, возвращающий количество очков на игровом поле.

`bool isFull() const` - Метод, проверяющий, заполнено ли игровое поле.

`bool isCollision() const` - Метод, проверяющий, произошла ли коллизия при последнем движении плиток.

`bool isMovePossible() const` - Метод, проверяющий, возможно ли сделать допустимый ход на текущем состоянии игрового поля.

`QVector<int> freePosition()` - Метод, возвращающий вектор свободных позиций на игровом поле.

`void initialize()` - Метод, инициализирующий игровое поле.

`void moveHorizontal(int i, int j, DIRECTION dir)` - Метод, реализующий горизонтальное движение плиток.

`void moveVertical(int i, int j, DIRECTION dir)` - Метод, реализующий вертикальное движение плиток.

`bool isChanged(Board &brd) const` - Метод, проверяющий, изменилось ли состояние игрового поля.

`bool isInBounds(int i, int j)` - Метод, проверяющий, находятся ли координаты в пределах игрового поля.

2.3.3 Классы Observer

Описание полей и методов класса `Observer`:

`Observer()` - Конструктор класса

`virtual void notify()` = 0 - Виртуальная функция `(Observer)`.

2.3.4 Класс Лучшего результата игры

Описание полей класса `QBest`:

`Game *game` - Поле, представляющее указатель на объект игры (`Game`).

`QVBoxLayout *mainLayout` - Поле, представляющее вертикальное расположение виджетов.

`QLabel *title` - Поле, представляющее заголовок виджета.

`QLabel *count` - Поле, представляющее счетчик лучшего результата.

`string line` - Строковое поле.

`int curr` - Поле, представляющее текущее значение.

2.3.5 Класс Интерфейса игрового поля

Описание полей класса `QBoard`:

`Game *game` - Поле, представляющее указатель на объект игры (`Game`).

`QVector< QVector<QTile*> > playerBoard` - Матрица игровых плиток, представляющая игровое поле в пользовательском интерфейсе.

`QGridLayout *mainLayout` - Основной макет для размещения элементов интерфейса виджета.

`QGridLayout *board` - Макет для размещения игровых плиток.

`QLabel *gameTitle` - Элемент интерфейса, отображающий заголовок игры.

`QHead *head` - Элемент интерфейса, представляющий заголовок (верхнюю часть) игрового поля.

`QHint *hint` - Элемент интерфейса, представляющий подсказку или подсказывающий элемент.

`QScore *score` - Элемент интерфейса, представляющий счет игры.

`QBest *best` - Элемент интерфейса, представляющий лучший результат.

`QNewButton *reset` - Кнопка для начала новой игры.

`QGameOver gameOver` - Элемент интерфейса, представляющий сообщение о завершении игры.

`QWinning winning` - Элемент интерфейса, представляющий сообщение о победе в игре.

`int currVersion` - Текущая версия игры.

`int currDimension` - Текущая размерность игрового поля.

`bool finished` - Флаг, указывающий, завершена ли игра.

Методы класса `QBoard`:

`QBoard(int version, int dimension, int target, QWidget *parent = 0)` - Конструктор класса `QBoard`.

`void notify()` - Метод, реализующий функциональность интерфейса `Observer`.

`bool getFinished()` - Метод, возвращающий флаг, указывающий, завершена ли игра.

`void drawBoard(int version, int dimension)` - Метод, отрисовывающий игровое поле с заданной версией и размерностью.

`void keyPressEvent(QKeyEvent *event)` - Метод обработки событий клавиш.

`void newGame() (слот)` - Слот, вызываемый при начале новой игры.

`void resetGame() (слот)` - Слот, вызываемый при сбросе игры.

Методы класса `QBest`:

`QBest(QWidget *parent = 0)` - Конструктор класса `QBest`.

`QString getBest()` - Метод, возвращающий лучший результат в виде строки.

`void setBest(QString count)` - Метод, устанавливающий лучший результат.

2.3.6 Класс Интерфейса кнопок

Описание полей класса `QButton`:

`QGraphicsTextItem *text` - Поле, представляющее текстовую часть кнопки. Методы класса `QButton`:

`QButton(QString name, QGraphicsItem *parent = 0)` - Конструктор класса `QButton`.

`void mousePressEvent(QGraphicsSceneMouseEvent *event)` - Метод, обрабатывающий событие нажатия кнопки мыши на кнопке.

`void hoverEnterEvent(QGraphicsSceneHoverEvent *event)` - Метод, обрабатывающий событие наведения курсора мыши на кнопку.

`void hoverLeaveEvent(QGraphicsSceneHoverEvent *event)` - Метод, обрабатывающий событие ухода курсора мыши с кнопки.

2.3.7 Класс Интерфейса игры

Описание полей класса `QGame`:

`QGraphicsScene *scene` - Сцена, представляющая графическую область для отображения объектов игры.

`Game *game` - Объект игры.

`QComboBox *combobox1, *combobox2, *combobox3` - Выбор параметров игры, таких как версия, размерность и цель.

`QGameOver gameOver` - Объект, представляющий сообщение о завершении игры.

`QBoard *board` - Объект, представляющий игровое поле.

`QScore *score` - Объект, представляющий счет в игре.

`QBest *best` - Объект, представляющий лучший результат в игре.

`int version, dimension, target` - Параметры игры, такие как версия, размерность и цель.

Методы класса `QGame`:

`QGame(QWidget *parent = 0)` - Конструктор класса `QGame`.

`void drawMenu()` - Метод для отрисовки меню игры.

`void drawOption()` - Метод для отрисовки опций игры.

`void drawGameOver()` - Метод для отрисовки сообщения о завершении игры.

`void setFinished()` - Метод для установки флага, указывающего на завершение игры.

`bool getFinished()` - Метод для получения флага, указывающего на завершение игры.

2.3.8 Класс Интерфейса конца игры

Описание полей класса `QGameOver`:

`QResetButton *reset` - Элемент интерфейса, представляющий кнопку сброса

Методы класса `QGameOver`:

`QGameOver(QWidget *parent = 0)` - Конструктор класса `QGameOver`. `QResetButton* getResetBtn() const` - Метод, возвращающий указатель на кнопку сброса.

2.3.9 Класс Интерфейса заголовка игры

Описание полей класса `QHead`:

`Game *game` - Объект игры.

`QHBoxLayout *mainLayout` - Основной макет для размещения элементов интерфейса виджета в горизонтальной компоновке.

`QLabel *gameTitle` - Элемент интерфейса, представляющий заголовок игры. `string head` - Строковое поле.

Методы класса `QHead`:

`QHead(int version, int target, QWidget *parent = 0)` - Конструктор класса `QHead`.

2.3.10 Класс Интерфейса текстовой подсказки в игре

Описание полей класса `QHint`:

`QHBoxLayout *mainLayout` - Основной макет для размещения элементов интерфейса виджета в горизонтальной компоновке.

`QLabel *hint` - Элемент интерфейса, представляющий текстовую подсказку в игре.

Методы класса `QHint`:

`QHint(int target, QWidget *parent = 0)` - Конструктор класса `QHint`.

2.3.11 Класс Интерфейса кнопки начала новый игры

Методы класса `QNewButton`:

`QNewButton(QWidget *parent = 0)` - Конструктор класса `QNewButton`.

`void mousePressEvent(QMouseEvent *event)` - Метод, обрабатывающий событие нажатия кнопки мыши на виджете.

`void hoverEnterEvent(QHoverEvent *event)` - Метод, обрабатывающий событие наведения курсора мыши на виджет.

`void hoverLeaveEvent(QHoverEvent *event)` - Метод, обрабатывающий событие ухода курсора мыши с виджета.

2.3.12 Класс Интерфейса кнопки сброса игры

Методы класса `QResetButton`:

`QResetButton(QWidget *parent = 0)` - Конструктор класса `QResetButton`.

`void mousePressEvent(QMouseEvent *event)` - Метод, обрабатывающий событие нажатия кнопки мыши на виджете.

2.3.13 Класс Подсчета очков в игре

Описание полей класса `QScore`:

`QVBoxLayout *mainLayout` - Основной макет для размещения элементов интерфейса виджета в вертикальной компоновке.

`QLabel *title, *count` - Элемент интерфейса, представляющий заголовок счета и сам счет.

Методы класса `QScore`:

`QScore(QWidget *parent = 0)` - Конструктор класса `QScore`.

`QString getScore()` - Метод для получения текущего значения счета в виде строки.

`void setScore(QString count)` - Метод для установки значения счета.

2.3.14 Класс Интерфейса плитки в игре

Описание полей класса `QTile`:

`Tile *tile` - Объект типа `Tile`.

`qreal scale` - Переменная для хранения коэффициента масштабирования. Методы класса `QTile`:

`QTile(const QString &text)` - Конструктор класса `QTile`.

`QTile(Tile *tile)` - Конструктор класса `QTile`.

`void drawTile(int version, int dimension)` - Метод для отрисовки плитки с учетом версии и размерности игры.

2.3.15 Класс Интерфейса в случае победы

Описание полей класса `QWinning`:

`QResetButton *reset` - Элемент интерфейса, представляющий кнопку сброса.

`QBest *best` - Объект, представляющий лучший результат в игре.

`QScore * score` - Объект, представляющий счет в игре.

Методы класса `QWinning`:

`QWinning(QWidget *parent = 0)` - Конструктор класса `QWinning`.

`QResetButton* getResetBtn() const` - Метод, возвращающий указатель на кнопку сброса.

2.3.16 Класс Subject

Описание полей класса `Subject`:

`vector<Observer*> observers` - Вектор для хранения объектов-наблюдателей

Методы класса `Subject` - Конструктор класса `Subject`.

`void notifyObs()` - Метод для оповещения всех подписанных наблюдателей.

`void registerObs(Observer *observer)` - Метод для регистрации нового наблюдателя.

2.3.17 Класс Плитки

Описание полей класса `Tile`:

`struct VALUE value` - Поле, представляющее числовое и текстовое значение тайла.

Структура `VALUE` содержит следующие поля:

`int number` - Числовое значение тайла.

`string word` - Текстовое представление тайла.

Методы класса `Tile`:

`Tile()` - Создает объект `Tile` с неопределенными значениями.

Tile(const Tile &tile) - Создает копию объекта `Tile` на основе переданного объекта.

`Tile(int number)` - Создает объект `Tile` с заданным числовым значением и текстовым значением, которое представляется соответствующим числом.

`int getNumber()` - Возвращает числовое значение текущего тайла.

`string getWord(int number)` - Возвращает слово, соответствующее заданному числовому значению.

`string match(int number)` - Возвращает слово, соответствующее заданному числовому значению.

`void upgrade()` - Повышает уровень тайла или выполняет другие операции обновления.

3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

3.1 Разработка схем алгоритмов

Метод `isFull()` в классе `Board` проверяет, полностью ли заполнена игровая доска плитками. Он возвращает логическое значение true, если все ячейки доски заняты, и false в противном случае.

Метод `match()` в классе `Tile` возвращает текстовое представление числового значения, переданного в качестве аргумента. Для этого метод использует оператор switch, чтобы сопоставить переданное число с одним из заранее определенных случаев. Затем метод присваивает числовое значение и соответствующее текстовое представление объекту value и возвращает это текстовое представление.

3.2 Разработка алгоритмов

3.2.1 Разработка алгоритма метода `Board::moveVertical()`

Метод `Board::moveVertical` взаимодействует с игровой доской в контексте игры 2048, обеспечивая вертикальное движение плиток. При вызове метода указывается направление движения: вверх (UP) или вниз (DOWN). Начиная с указанной позиции (`i, j`) на доске, метод перемещает соответствующую плитку в выбранном направлении.

В процессе выполнения метода проверяется, не является ли исходная позиция (`i, j`) пустой, то есть не содержит ли плитку. Если на этой позиции есть плитка, то выполняются следующие шаги.

Определение Новой Позиции: Вычисляется новая позиция (`newi`) в зависимости от выбранного направления. Метод продолжает перемещаться в указанном направлении до тех пор, пока не достигнет границы доски или не столкнется с другой плиткой.

Обработка Границы Доски: Если новая позиция находится за пределами границы доски, то плитка перемещается на противоположную сторону. Например, если движение вверх и новая позиция `newi` меньше 0, то плитка перемещается в нижнюю часть столбца.

Обработка Столкновений и Объединений: Если новая позиция не превышает границы доски, проверяется, есть ли на новой позиции другая плитка. Если плитки имеют одинаковые значения, они объединяются, и значение объединенной плитки увеличивается. Игрок также получает очки. Если значения плиток различны, то текущая плитка перемещается на пустое место перед или после другой плитки в зависимости от направления.

Обновление Игровой Доски: В зависимости от результата перемещения и возможного объединения, метод обновляет элементы игровой доски. Если

столкновение произошло, устанавливается флаг `lastCollision`, и доска обновляется новой плиткой.

Опишем данный алгоритм по шагам:

Шаг 1. Проверка наличия блока в текущей позиции.
Шаг 2. Инициализация переменных и определение новой позиции.
Шаг 3. Поиск новой позиции для блока.
Шаг 4. Обработка случая, когда новая позиция находится за пределами доски.
Шаг 5. Обработка случая слияния блоков.
Шаг 6. Удаление блока из текущей позиции, если была коллизия или произошло перемещение.
Шаг 7. Установка флага коллизии, если она произошла.

Метод `Board::moveVertical` вносит изменения в игровую доску, отражая действия игрока по вертикали и обрабатывая возможные столкновения и объединения плиток в игре 2048.

3.2.2 Разработка алгоритма метода `Board::moveHorizontal()`

Метод `Board::moveHorizontal` предназначен для управления горизонтальным движением плиток в игре 2048 на игровой доске. При вызове метода указывается направление движения, которое может быть либо вправо (RIGHT), либо влево (LEFT).
В ходе выполнения метода осуществляется итерация по каждой строке доски. Для каждой плитки в текущей строке проверяется возможность её перемещения в указанном направлении. При этом метод учитывает наличие препятствий (неподвижных плиток или границы доски) и определяет новое положение плитки после перемещения.
Если в результате движения две плитки с одинаковыми значениями оказываются рядом, происходит их объединение. В этом случае игрок получает определённое количество очков, соответствующее значению объединившейся плитки. Метод также обновляет флаг `lastCollision`, указывая, произошло ли столкновение плиток при последнем выполнении метода.

После завершения перемещения и объединения плиток метод проверяет, изменилась ли игровая доска. Если да, то генерируется новая плитка и устанавливается на свободную позицию. Затем вызывается метод `notifyObs()`, который уведомляет наблюдателей (например, графический интерфейс), что состояние доски изменилось.

Опишем данный алгоритм по шагам:

Шаг 1. Проверка наличия блока в текущей позиции.
Шаг 2. Инициализация переменных и определение новой позиции.
Шаг 3. Поиск новой позиции для блока.
Шаг 4. Обработка случая, когда новая позиция находится за пределами доски.
Шаг 5. Обработка случая слияния блоков.
Шаг 6. Удаление блока из текущей позиции , если была коллизия или произошло перемещение.
Шаг 7. Установка флага коллизии, если она произошла.

В итоге, метод `Board::moveHorizontal` обеспечивает выполнение всей логики горизонтального движения плиток в игре 2048, включая их объединение, начисление очков и генерацию новых плиток после каждого хода.

4 РЕЗУЛЬТАТЫ РАБОТЫ

В самом начале, пользователь сразу попадает в главное окно, где кратко объясняется как играть , а так же есть кнопки "Начать" и "Выход". Пример этого окна показан на рисунке 4.1.

При нажатии на кнопку "Выход" приложении закрывается.

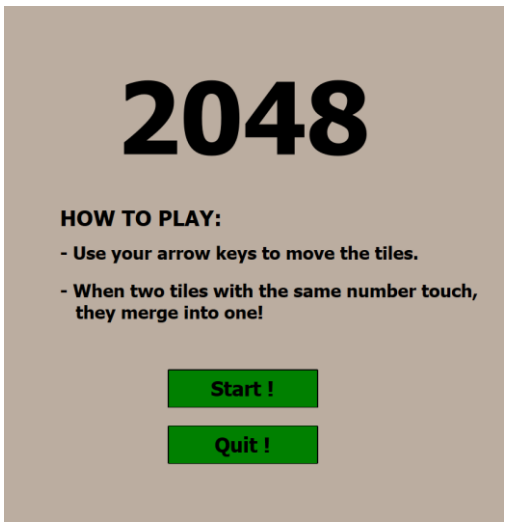

Рисунок 4.1 – Окно главного меню игры

При нажатии на кнопку "Начать" открывается новое окно, где пользователь может выбрать параметры игра , а именно:

1. Версию игры (цифровую или буквенную).
2. Размер поля (3 на 3, 4 на 4, 5 на 5, 6 на 6, 7 на 7, 8 на 8).
3. Цель игры (набрать плитку номиналом 2048 или 4096).

Пример этого окна показан на рисунке 4.2.

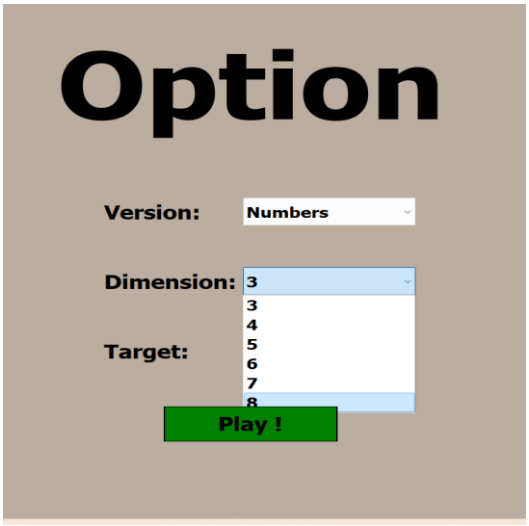

Рисунок 4.2 – Окно для выборов параметров игры

После выбора параметров пользователь может нажать на кнопку "Играть" и откроется окно игры.

Пример этого окна показан на рисунке 4.3.

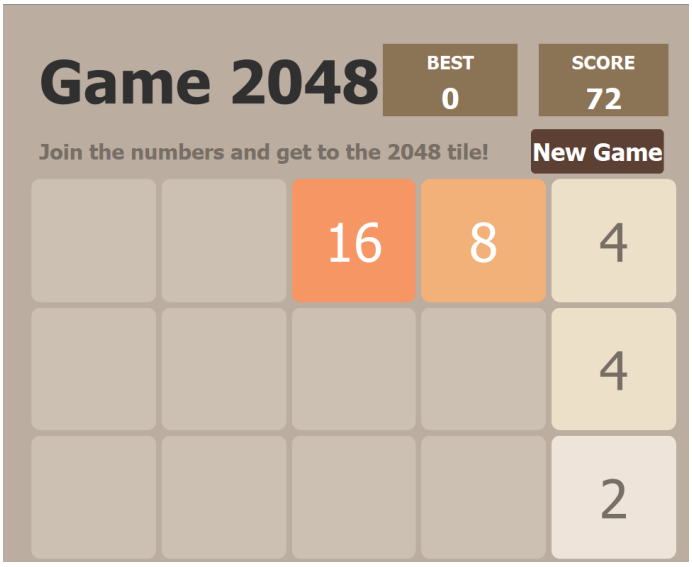

Рисунок 4.3 – Окно игры

В игровом окне пользователь может нажать кнопу "Новая игра" , в этом случае игра начнется сначала с теми же параметрами . Так же пользователь может играть в игру нажимая стрелки , тем самым сдвигая плитки на игровом поле. В случае достижения цели , а именно получения плитки 2048 или 4096, в зависимости от параметра игры , открывается окно выигрыша.

Пример этого окна показан на рисунке 4.4.

В нем можно нажать кнопку "Новая игра" или продолжить эту игру закрыв окно.

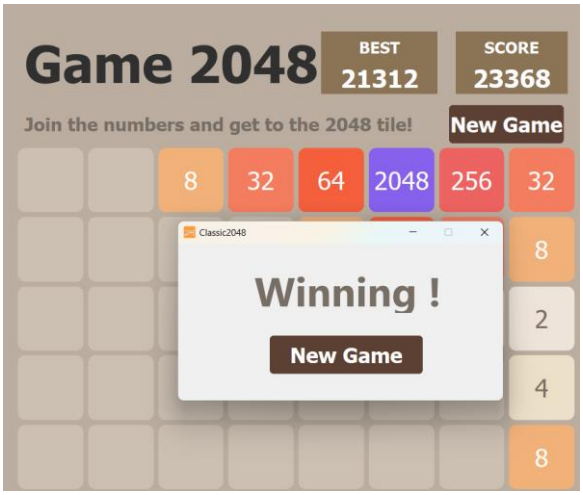

Рисунок 4.3 – Окно выигрыша

В случае если у пользователя не остается ходов , а все игровые плитки заполнены , открывается окно проигрыша.

Пример этого окна показан на рисунке 4.5.

В этом окне пользователь может нажать кнопку "Новая игра" для начала новый игры.

Рисунок 4.3 – Окно проигрыша

В игровом окне так же есть подсчет очков текущей игры и лучший результат по набранным очкам. Очки даются в свою очередь следующим образом .
В случае слияния плиток 2 и 2 дается 4 очка , плиток 4 и 4 - 8 очков и т. д.

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы был проведен анализ и исследование игры 2048, разработанной с использованием языка программирования C++ и фреймворкера Qt Creator. Задачей работы было изучение основных концепций, алгоритмов и структур данных, лежащих в основе игры, а также создание собственной реализации игры с использованием указанных технологий.

В процессе выполнения курсовой работы были рассмотрены основные принципы разработки игр, использование классов и объектно-ориентированного программирования для создания структуры игры 2048. Применение фреймворкера Qt Creator позволило легко реализовать графический интерфейс пользователя, обеспечивая удобство и привлекательность взаимодействия с игрой.

Процесс создания игры 2048 на языке C++ с применением Qt Creator предоставил возможность углубленного изучения многих аспектов программирования, включая работу с пользовательским интерфейсом, обработку событий, логику игры и оптимизацию кода.

В итоге, курсовая работа не только позволила получить глубокие знания о процессе разработки игр на практике, но и предоставила основу для дальнейших исследований и улучшений в области программирования и геймдизайна.

Однако игра открыта для модернизации. Например, можно добавить механику отмены ходов, добавить подсчетов времени прохождения игры.

Игра оказалась нетрудной в реализации, имеет удобный интерфейс.

На основе полученных знаний при создании игры можно будет создавать более проработанные и интересные игры, например стратегии или викторины.

Создание игры было выполнено на ОС Windows 11, используя среду разработки QtCreator.

Код программы приведен в приложении Г.

СПИСОК ЛИТЕРАТУРЫ

[1] Шилдт Г. C++ базовый курс / Г. Шилдт - 3-е изд. М. : Вильямс , 2007 – 624с.
[2] Конструирование программ и языки программирования: метод. указания по курсовому проектированию – А. В. Бушкевич, А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2009.
[3] Qt Documentation [Электронный ресурс]. -Электронные данные. -Режим доступа: https://doc.qt.io/ Дата доступа: 24.10.2023
[4] Объектно-ориентированное программирование на языке C++; учеб. Пособие /Ю. А. Луцик , В. Н. Комличенко. – Минск: БГУИР , 2008.
[5] СТП 01–2017. Дипломные проекты (работы): общие требования. – Введ. 2017–01–01. – [Электронный ресурс].– 2017– Режим доступа: http://library.bsuir.by/online/showpage.jsp?PageID=86151 – Дата доступа: 23.11.2023

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема метода isFull()

ПРИЛОЖЕНИЕ В

(обязательное)

Схема метода play ()

ПРИЛОЖЕНИЕ Г

(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д

(обязательное)

Ведомость документов