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

К ЗАЩИТЕ ДОПУСТИТЬ

_____ *Е.В. Богдан*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему
ИГРА «АРКАНОИД»

БГУИР КП 1–40 02 01 213 ПЗ

Студент: Каплич В.А.

Руководитель: Ассистент кафедры ЭВМ
Богдан Е.В.

Минск 2023

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ

_____

(подпись)

_____2023 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Каплич ВалерииАлександровне*

Тема проекта *«Игра «Арканоид»*

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.
   3.1. Обзор методов и алгоритмов решения поставленной задачи.
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 метода keyPressEvent(QKeyEvent *e).*

*3. Схема метода keyReleaseEvent(QKeyEvent *e).*

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

7. Дата выдачи задания *15.09.2023 г.*

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

1. Выбор задания. Разработка содержания пояснительной записки.

Перечень графического материала – 15 %;

разделы 2, 3 – 10 %;

разделы 4 к – 20 %;

разделы 5 к – 35 %;

раздел 6,7,8 – 5 %;

раздел 9 к – 5%;

оформление пояснительной записки и графического материала к 15.12.23 – 10 %

Защита курсового проекта с 21.12 по 28.12.23г.

РУКОВОДИТЕЛЬ Е.В. Богдан _____

(подпись)

Задание принял к исполнению _____ Каплич В.А.

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня игра «Аракноид» очень распространена, хотя пиком их популярностибыли девяностые и двухтысячные. Я всегда любил компьютерные игры и решил попробовать создать свою игру. Выбрал жанр платформер так как мне нравятся игры данного жанра, и они не слишком сложны в реализации. Потенциал для развития этого проекта я считаю почти бесконечным так как можно придумывать новые уровни, рассказывать истории с помощью игры и улучшать графическую реализацию.

Объектно-ориентированное программирование представляет собой технологию программирования, которая базируется на классификации и абстракции объектов. Одним из наиболее популярных средств объектно-ориентированного программирования, позволяющим разрабатывать программы, эффективные по объёму кода и скорости выполнения является C++.

C++ – компилируемый, статически типизируемый язык общего назначения. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое 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 в себе как возможности низкоуровневых языков программирования, так и возможности высокоуровневых.

Основными концепциями объектно-ориентированного программирования являются инкапсуляция, полиморфизм и наследование. Язык C++ предоставляет исчерпывающие возможности для реализаций этих концепций. В результате использования инкапсуляции, программа, написанные на C++, обладает повышенной защищённостью объектов от влияния на них кода других частей этой же программы. Наследование предоставляет важную возможность повторного использования кода, что может значительно уменьшить количество кода, однако требует предварительного проектирования архитектуры. Полиморфизм позволяет программе быть более гибкой. Такая система удобна в тестировании и позволяет легко заменять и модифицировать свои компоненты.

Универсальность и гибкость языка позволяют использовать его в различных целях. Область применения данного языка включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Исходя из этого можно считать, что данный язык достаточно удобен для написания выбранной курсовой работы.

1 ПОСТАНОВКА ЗАДАЧИ

Для реализации игры будем использовать фреймворк Qt.

Это очень удобный конструктор графической оболочки, который так же имеет свой встроенный методы для разработки кроссплатформенных приложений, отличное решение для Арканоида.

В программе будут реализованы функции:

- Пауза и продолжение игры.
- Разрушение блоков под действием удара
- Выпадение бонусов из блоков
- Счёт
- Основная система будет Windows.

Для реализации данного программного обеспечения используется объектно-ориентированный язык программирования C++.

Для начала надо определить одни из самых важных определений при создании игры. Это paddle, rect.

- Rect (от английского слова rectangle) – это прямоугольник вокруг какого-то объекта. Он нужен чтобы мы могли управлять этим объектом и создавать логику взаимодействий предметов.
- Paddle – наша ракета (платформа), на которую падает шарик.
- Ball – наш шарик.

2 ОБЗОР ЛИТЕРАТУРЫ

2.1 Обзор методов и алгоритмов решения поставленной задачи

Основной алгоритм множества игр – это цикл в котором мы обрабатываем разные действия пользователя (например, нажатие кнопок), обновляем состояния разных объектов (например, перемещение главного персонажа) и в конце после этого происходит отрисовка новых позиций объектов. Программа проходит по этому циклу очень большое количество раз, и мы многих итераций цикла просто не замечаем. Это сделано для более плавной картинки и удобства работы.

Для создания окна в котором будет отображаться игра мы используем класс MainWindow [2] в конструкторе, которого мы задаём размер окна в пикселях и название этого окна.

Листинг кода программы находится в приложении А.

Для уникальности объекта используется шаблон template < class T>. Этот шаблон имеет метод класса, возвращающий ссылку на единый экземпляр объекта типа T, реализующий создание и возвращение единственного экземпляра объекта. Также данный класс содержит оператор присваивания, чтобы предотвратить присваивание экземпляров.

Листинг кода программы находится в приложении А.

2.2 Анализ существующих аналогов

Тема курсового проекта была выбрана с целью освоения навыков разработки интернет-магазина с использованием фреймворка Qt и языка SQL для работы с базой данных.

Интернет-магазины являются актуальными и востребованными в современном мире, поскольку все больше людей предпочитают делать покупки онлайн. Для создания корректно работающего интернет-магазина с использованием Qt и SQL необходимо иметь представление о существующих аналогах.

Нужно также проанализировать существующие решения интернет-магазинов, реализованных с использованием Qt и SQL, чтобы определить, какие функциональности и особенности могут быть полезны в создаваемом приложении.

2.2.1 Приложение Brick Breaker

Brick Breaker – это игра по уничтожению блоков на фоне космоса. Любой человек может играть в игру, из-за её простого контроля и лёгких правил. Вы можете наслаждаться многочисленными уровнями совершенно бесплатно. Проходите миссии и получайте множество монет. Вы можете играть оффлайн. Вы можете играть в режиме самолёта. Игра также доступна

и на планшетах.

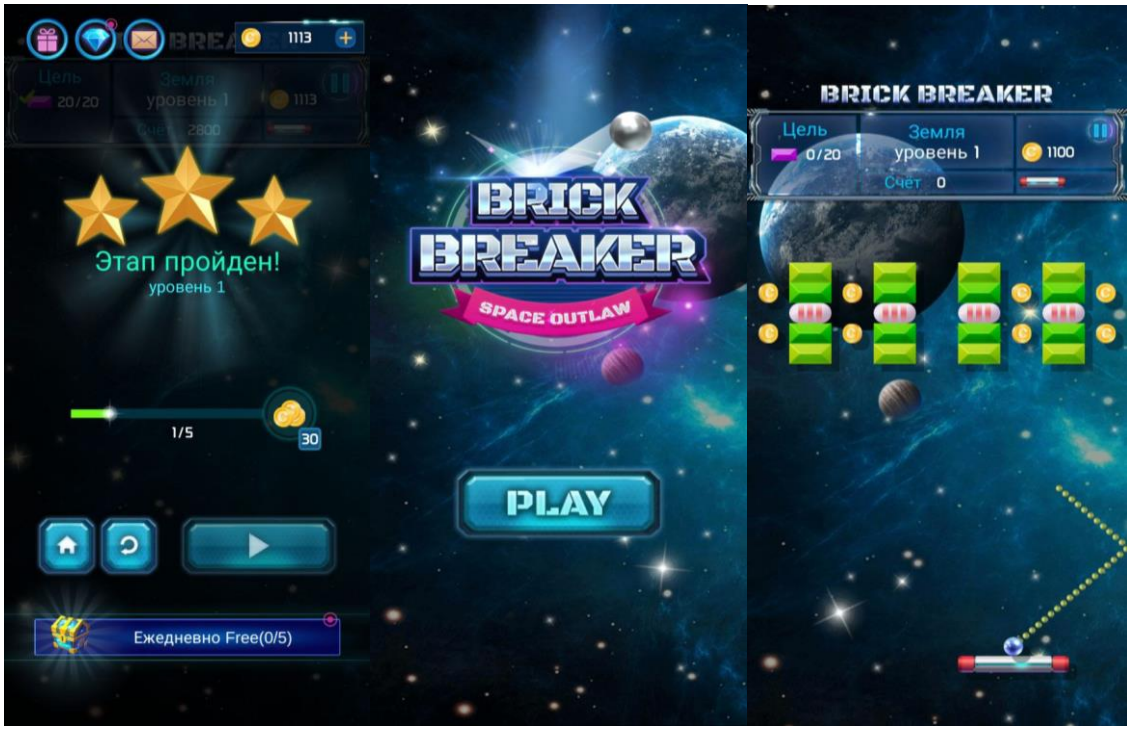

Рисунок 1.1 – Скриншоты Brick Breaker: Space Outlaw

2.2.2 Приложение Brickscapes: Bricks Breaker

Игра «Brickscapes: Bricks Breaker» - это бесплатная игра, оптимизирован для планшетов и больших экранов. Откройте для себя тысячи веселых и сложных игровых уровней. Удивительные уровни из кубиков и шариков с впечатляющими миссиями. Играйте в офлайн-игру, Wi-Fi не требуется. Она Подходит для семей и всех возрастов.

Эти игры с кубиками и шариками хорошо разработаны для мобильных телефонов. Вы можете загрузить Android-версию игры Brickscapes: Bricks Breaker с дробилкой мячей из Google Play. Они также предлагаем планшетную версию. Это удобный и чистый пользовательский интерфейс игры для всех возрастов.

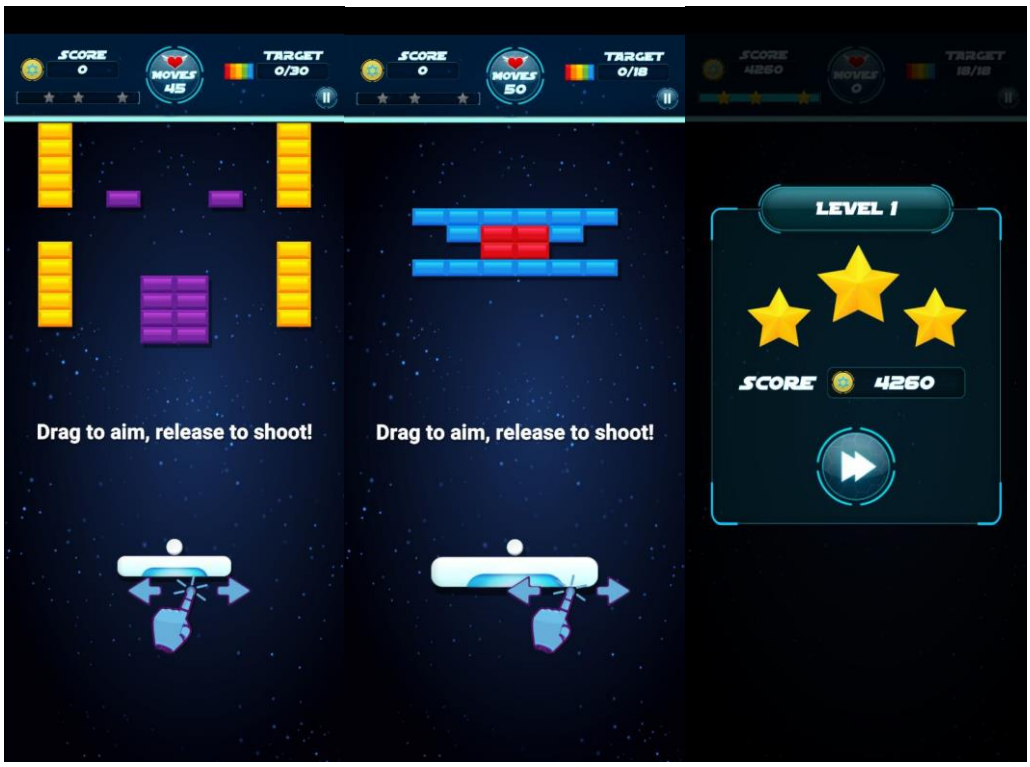

Рисунок 1.2 – Скриншоты Brickscapes: Bricks Breaker

2.2.3 Приложение Brick Mania

В последней версии Brick Mania есть четыре вселенные (режима), а именно Easy, Dense, Hard и Super Hard. Легкие уровни проходить легко и расслабляюще; сложные уровни имеют в 4 раза больше перерывов и чрезвычайно приятны для прохождения; сложные уровни сложны с «бомбами” и ловушками, сжимающими планку, и поэтому подходят только для тех, кто хорошо обучен разбивать кирпичи.

Пожалуйста, имейте в виду, что независимо от того, в какой вселенной вы находитесь, конечная цель состоит в том, чтобы поймать как можно больше бонусов и умножить количество шаров. Не пытайтесь поймать каждый падающий шар, потому что иначе вы не пройдете много уровней в плотном и сложном режимах. На каждом уровне вы можете собрать максимум 3 звезды, которые затем можно использовать для «покупки» бонусов.

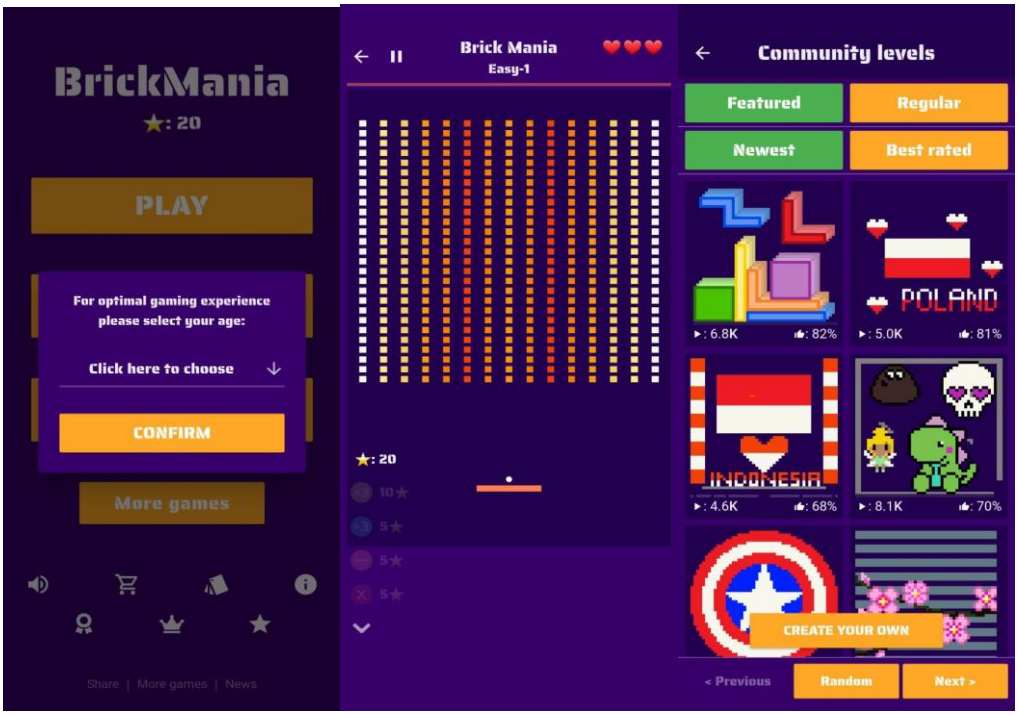

Рисунок 1.3 – Скриншоты Brick Mania

2.3 Требования к работе программы

Основной алгоритм множества игр – это цикл в котором мы обрабатываем разные действия пользователя (например, нажатие кнопок), обновляем состояния разных объектов (например, перемещение главного персонажа) и в конце после этого происходит отрисовка новых позиций объектов. Программа проходит по этому циклу очень большое количество раз, и мы многих итераций цикла просто не замечаем. Это сделано для более плавной картинки и удобства работы.

Для создания окна в котором будет отображаться игра мы используем класс MainWindow [2] в конструкторе, которого мы задаём размер окна в пикселях и название этого окна. В программе были созданы 14 классов: Breakout, Paddle, YellowBrick, RedBrick, BlueBrick, Ball, Bonus, FileNotFoundException, IOException, MainWindow, MyException, Ui_MainWindow, List, NoSuchElementException.

В игре используется три цвета кирпича: синий, красный и жёлтый. У самого первого 3 жизни, а у остальных на единицу меньше. Такое разнообразие реализовано за счёт наследование.

При каждом новом запуске игры, будет сгенерирована новая раскладка кирпичей. Это вызвано тем, что каждый раз, при создании кирпича, его конструктор определяется случайным образом. Если мяч ударяется о нижнюю часть Кирпича, то мы меняем направление у мяча – он идет вниз. В зависимости от вида Кирпича за разрушение начисляется определенное количество очков.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ

3.1 Структура входных и выходных данных

Программа состоит из 8 файлов с кодом:

- `main.cpp` – в этом файле располагается главная функция main;
- `breakout.cpp` – в этом файле располагается определение методов класса Breakout;
- `breakout.h` – в этом файле располагается объявление методов и переменных класса Breakout;
- `paddle.cpp` – в этом файле располагается определение методов класса Paddle;
- `paddle.h` – в этом файле располагается объявление методов и переменных класса Paddle;
- `yellowBrick.cpp` – в этом файле располагается определение методов класса YellowBrick;
- `yellowBrick.h` – в этом файле располагается объявление методов и переменных класса YellowBrick;
- `redBrick.cpp` – в этом файле располагается определение методов класса RedBrick;
- `redBrick.h` – в этом файле располагается объявление методов и переменных класса RedBrick;
- `blueBrick.cpp` – в этом файле располагается определение методов класса BlueBrick;
- `blueBrick.h` – в этом файле располагается объявление методов и переменных класса blueBrick;
- `ball.cpp` – в этом файле располагается определение методов класса Ball;
- `ball.h` – в этом файле располагается объявление методов и переменных класса Brick.
- `bonus.cpp` – в этом файле располагается определение методов класса Bonus;
- `bonus.h` – в этом файле располагается объявление методов и переменных класса Bonus;

Также в программе используются изображения в формате png. Они лежат в папке img.

Для подключения библиотеки Qt нужно скачать файл с официального сайта [2] библиотеку собрать её и добавить, как third-party library. Но более простой способ – это просто скачать Qt версии 6. Проект собирался с помощью утилиты CQtDeployer[3], также создан был deb пакет вместе с архивами.

Клавиши управления:

- Стрелочка влево или «A» – передвижение влево;
- Стрелочка вправо или «D» – передвижение вправо;
- Клавиша P – пауза игры;

- Escape – выход из игры.

3.2 Разработка диаграммы классов

Диаграмма классов представлена в приложении Б.

3.3 Описание классов

3.3.1 Ball

Класс Ball описывает наш мяч. В переменных xdir и ydir хранится направление движения Мяча. В начале игры Мяч движется в направлении вправо-вверх: xdir = -1, ydir = -1. Метод autoMove() вызывается в каждом игровом цикле для перемещения Мяча по экрану. Если Мяч достигает границ окна (за исключением нижней), то он меняет свое направление. Если же Мяч попадает в нижнюю границу окна, то назад он не отскакивает, а игра при этом считается завершенной.

Метод `autoMove()` перемещает мяч по игровому полю в соответствии с текущими значениями направления движения (xdir, ydir). Если мяч достигает границ поля, направление движения изменяется на противоположное.

Метод `resetState()` возвращает мяч в начальное положение на игровом поле.

Метод `setXDir()` устанавливает значение переменной xdir, которая определяет направление движения мяча по оси X.

Метод `setYDir()` устанавливает значение переменной ydir, которая определяет направление движения мяча по оси Y.

Метод `getXDir()` возвращает текущее значение переменной xdir.

Метод `getYDir()` возвращает текущее значение переменной ydir.

Метод `getRect()` возвращает прямоугольник (rect), который определяет положение и размеры мяча на игровом поле.

Метод `getImage()` возвращает изображение (image) мяча.

3.3.2 BlueBrick

Класс BlueBrick описывает блок. Класс BlueBrick наследуется от класса RedBrick, но добавляется одна булевская переменная activeBonus, для доступа к ней создан метод getActiveBonus, который показывает наличие бонуса в данном блоке или нет, в зависимости от этого, возвращает бонус. Конструктор класса BlueBrick загружает изображение Кирпича, инициализирует переменную-флаг activeBonus и устанавливает изображение в исходную позицию.

Метод `getBonus()` возвращает бонус, связанный с синим кирпичом.

Метод `getActiveBonus()` проверяет, был ли уничтожен синий кирпич. Если кирпич уничтожен, он возвращает, доступен ли активный бонус или нет.

Метод `destroy()` уничтожения изменяет состояние синего кирпича. В зависимости от стабильности кирпича метод снижает стабильность, обновляет изображение кирпича и, наконец, помечает кирпич как раз

Метод `deleteObject()` отвечает за удаление объекта синего кирпича.

3.3.3 Bonus

Класс Bonus описывает наш бонус. У класса Bonus такие же поля и геттеры/сеттеры, как и у Paddle, но добавляется одна булевская переменная active, для доступа к ней создан метод `isActive()`, который показывает активен бонус или нет, в зависимости от этого, отображает его в окне. Конструктор класса Bonus загружает изображение Бонуса, инициализирует переменную-флаг active и устанавливает изображение в исходную позицию.

Метод `deleteObject()` удаляет объект бонуса.

Метод `resetState()` сбрасывает состояние бонусного объекта, перемещая его прямоугольник в исходное положение (INITIALX, INITIALY).

Метод `isDestroyed()` возвращает логическое значение, указывающее, уничтожен ли бонусный объект.

Метод `getRect()` возвращает прямоугольник, представляющий положение и размер бонусного объекта.

Метод `getImage()` возвращает ссылку на изображение, связанное с бонусным объектом.

Метод `autoMove()` автоматически перемещает бонусный объект, перемещая его прямоугольник на основе его текущих направлений x и y (xdir и ydir).

Метод `getBonus(int Score)` определяет тип бонуса и возвращает обновленный балл в зависимости от типа бонуса. В этом примере балл удваивается, если тип бонуса равен 1.

3.3.4 Breakout

Класс Breakout хранит состояние нашей игры.

Метод `paintEvent(QPaintEvent event)` отвечает за рисование игровой сцены на основе состояния игры. Он вызывает метод `finishGame(QPainter Painter, QString message)`, если игра окончена или выиграна, а в противном случае вызывает метод `drawObjects(QPainter Painter)` для визуализации игровых объектов.

Метод `finishGame(QPainter Painter, QString message)` рисует экран окончания игры или победы на основе переданного ему сообщения, а также визуализирует фон и соответственно набирает очки.

Метод `drawObjects(QPainter Painter)` отвечает за рисование

игровых объектов, таких как мяч, ракетка, кирпичи, бонусы и счет, на игровом экране. Он проверяет, не активно ли игровое меню, а затем рисует каждый игровой объект в соответствующем положении и состоянии.

Метод события таймера `timerEvent(QTimerEvent *event)` отвечает за обработку события таймера в игре. Он вызывает метод `moveObjects()` для обновления положения игровых объектов, метод `checkCollision()` для обработки столкновений объектов.

Метод `moveObjects()` перемещает мяч, ракетку и бонусы, обновляя их позиции на основе их текущей скорости и направления.

Метод события выпуска ключа `keyReleaseEvent(QKeyEvent *event)` отвечает за обработку события выпуска ключа. Он проверяет, отпущены ли клавиши со стрелками влево или вправо, а также клавиши «A» или «D», и соответствующим образом обновляет горизонтальное движение весла.

Метод события нажатия клавиши `keyPressEvent(QKeyEvent *event)` обрабатывает события нажатия клавиши. Он проверяет, нажата ли клавиша со стрелкой влево или вправо, а также клавиши «A» или «D», и соответствующим образом обновляет горизонтальное движение весла. Кроме того, он обрабатывает события нажатия клавиш: «P» для приостановки игры, «Пробел» для запуска игры и «Escape» для выхода из игры. Если нажата любая другая клавиша, она вызывает метод базового класса для обработки события нажатия клавиши.

Метод `startGame()` инициализирует состояние игры при запуске игры. Он устанавливает фон игрового экрана, сбрасывает состояние мяча и ракетки, очищает кубики и бонусы, генерирует новые кубики в случайных позициях, устанавливает переменные состояния игры и запускает игровой таймер.

Метод `pauseGame()` управляет функцией паузы в игре. Если игра в настоящий момент приостановлена, она возобновляет игровой таймер и устанавливает для состояния паузы значение false. Если игра не поставлена на паузу, она приостанавливает игровой таймер и устанавливает для состояния паузы значение true.

Метод `stopGame()`, останавливает игру, убивая игровой таймер и устанавливая для состояния завершения игры значение true, а также отмечая игру как не запущенную.

Метод победы `victory()` вызывается, когда игра выиграна. Он останавливает игровой таймер и устанавливает для состояния «Выиграна игра» значение «истина», отмечая при этом игру как не начавшуюся.

Метод `drawScore(QPainter *painter, int x, int y, int size)` рисует счет на игровом экране в указанной позиции с заданным размером шрифта. Он создает сообщение о счете, устанавливает шрифт, а затем рисует текст на игровом экране.

Метод `checkCollision()` отвечает за обработку столкновений между игровыми объектами (такими как мяч, платформа и кирпичи), а также за проверку условий выигрыша/проигрыша игры.

3.3.5 FileNotFoundException

Класс FileNotFoundException представляет собой специальное исключение, которое можно использовать в программе C++ для обработки случаев, когда файл не найден.

Метод `show()` отвечает за отображение конкретной информации об ошибке при возникновении исключения FileNotFoundException. При вызове он выводит на консоль сообщение о том, что было создано исключение FileNotFoundException, а также код, связанный с этим исключением.

Метод `getCode()` используется для получения кода ошибки, связанной с FileNotFoundException.

3.3.6 IOException

Класс IOException – это особый тип исключения, производный от базового класса MyException. Он предоставляет функциональные возможности для обработки и передачи исключений, связанных с проблемами ввода/вывода.

Метод `show()` из базового класса для отображения конкретной информации об ошибке при возникновении исключения IOException. Он выводит на консоль сообщение, указывающее, что было создано исключение IOException, а также конкретный код, связанный с исключением.

Метод `getCode()` возвращает код ошибки, связанный с IOException.

3.3.7 MainWindow

MainWindow публично наследуется от QMainWindow и содержит несколько переменных-членов и частный слот. Ниже приведено подробное описание класса и его методов.

Метод `TimerSlot()` частный слот для обработки операций, связанных со временем, увеличения переменной времени и обновления метки в пользовательском интерфейсе новым значением времени.

3.3.8 MyException

Класс MyException служит основой для создания пользовательских классов исключений.

Метод `show()` выводит сообщение о том, что было сгенерировано базовое исключение.

3.3.9 Ui_MainWindow

Класс Ui_MainWindow – это класс пользовательского интерфейса,

который содержит макет и элементы главного окна приложения. Существует вложенное пространство имен Ui с классом MainWindow, который наследуется от Ui_MainWindow. Это позволяет использовать элементы пользовательского интерфейса в других частях кода.

Метод `setupUi(QMainWindow *MainWindow)` устанавливает макет и инициализирует элементы пользовательского интерфейса. Он создает и устанавливает свойства центрального виджета, метки, кнопки, строки меню и строки состояния. Он также подключает слоты к сигналам для обработки событий.

Метод `retranslateUi(QMainWindow *MainWindow)` используется для перевода элементов пользовательского интерфейса на разные языки. Он устанавливает заголовок окна, текст метки и текст кнопки с помощью функции QApplication::translate.

3.3.10 Paddle

Paddle описывает игрока (платформу). У класса Paddle есть поля dx это его скорость по координатам x, он может перемещаться в лево и в право, rect – это его положение на поле и его размер, image – это переменная, которая отвечает за картинку самой «ракетки».

Метод `setDx(int x)` устанавливает горизонтальную скорость (изменение положения) весла. Перегруженная версия метода перемещения, которая перемещает весло в указанную координату X, сохраняя при этом исходную координату Y.

Метод `move()` отвечает за горизонтальное перемещение весла в зависимости от его текущей скорости (dx).

Метод `resetState()` сбрасывает состояние весла, перемещая его в исходное положение.

Метод `getRect()` возвращает прямоугольник, представляющий положение и размер платформы.

Метод `getImage()` возвращает ссылку на изображение, связанное с веслом.

3.3.11 RedBrick

Класс RedBrick описывает блок. Класс RedBrick наследуется от класса YellowBrick, но добавляется одна переменная stability, которая определяет стойкость блока, от стойкости блока зависит с какого удара разрушится блок. Конструктор класса RedBrick загружает изображение Кирпича, инициализирует переменную stabillity и устанавливает изображение в исходную позицию.

Метод destroy() отвечает за изменение состояния объекта из красного кирпича при попадании в него игрового мяча. Если стабильность красного кирпича равна 1, метод обновляет изображение кирпича до изображения

желтого кирпича и устанавливает стабильность на 0. Если стабильность не равна 1, метод помечает кирпич как уничтоженный, устанавливая для флага «destroyed» значение true.

Метод `deleteObject()` обеспечивает удаление экземпляра класса RedBrick. В C++ используется ключевое слово delete для освобождения памяти, связанной с объектом из красного кирпича.

3.3.12 YellowBrick

Класс YellowBrick описывает блок. У класса YellowBrick такие же поля и геттеры/сеттеры, как и у Paddle, но добавляется одна булевская переменная destroyed, для доступа к ней создан метод isDestroyed(), который показывает разрушен ли данный блок или нет, в зависимости от этого, отображает его в окне. Конструктор класса Brick загружает изображение Кирпича, инициализирует переменную-флаг destroyed и устанавливает изображение в исходную позицию.

Метод `getRect()` возвращает прямоугольник, представляющий положение и размер желтого кирпича.

Метод `setRect(QRect rct)` устанавливает прямоугольник желтого кирпича в указанный QRect.

Метод `getImage()` возвращает ссылку на изображение, связанное с желтым кирпичом.

Метод `isDestroyed()` возвращает логическое значение, указывающее, был ли уничтожен желтый кирпич.

Метод `setDestroyed(bool destroy)` устанавливает «уничтоженное» состояние желтого кирпича на основе предоставленного логического значения.

Методы `getXDir()` и `getYDir()` возвращают направления x и y желтого кирпича.

Метод `getScore()` возвращает значение балла желтого кирпича. Если кирпич уничтожен, он возвращает очки; в противном случае возвращается 0.

Метод `destroy()` устанавливает для флага «уничтожено» значение true, что указывает на то, что желтый кирпич был уничтожен.

Метод `deleteObject()` удаляет экземпляр класса YellowBrick с помощью ключевого слова «delete» в C++.

3.3.13 List

Класс List представляет собой шаблонный контейнер. Используется для хранение объектов классов RedBrick, YellowBrick, BlueBrick, Bonus.

Метод `pushBack(T data)` добавляет в конец списка новый узел, содержащий заданные данные.

Метод `pushFront(T data)` добавляет новый узел, содержащий заданные данные, в начало списка.

Метод `getSize()` возвращает размер списка.

Метод стирания удаляет узел по указанному индексу из списка. Если индекс выходит за пределы, выдается исключение.

Метод `ClearList()` удаляет все узлы из списка.

Метод `popFront()` удаляет первый узел из списка.

Метод `popBack()` удаляет последний узел из списка.

Метод `peek(int num)` извлекает данные по указанному индексу в списке.Метод оператора позволяет получить доступ к данным по указанному индексу, используя запись массива.

3.3.14 NoSuchElementException

Класс NoSuchElementException представляет невозможность найти элемент в коллекции. Он включает методы для управления и получения информации об исключении.

Метод `getCode()` возвращает код ошибки, связанный с исключением NoSuchElementException.

Метод `show()` отображает конкретную информацию об ошибке при возникновении исключения NoSuchElementException. При вызове этого метода на консоль выводится сообщение, указывающее, что было создано исключение NoSuchElementException, включая конкретный код, связанный с исключением.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов

Схема метода `keyPressEvent(QKeyEvent *e)` приведена в приложении В. – метод обрабатывает события нажатия клавиши.

Схема метода `keyReleaseEvent(QKeyEvent *e)` приведена в приложении Г. – метод отвечает за обработку события выпуска ключа.

4.2 Разработка алгоритмов

4.2.1 Алгоритм перерисовки объектов.
Метод void Breakout::paintEvent(QPaintEvent *e)

Шаг 1. Создание объекта класса QPainter: QPainter painter (this)
Шаг 2. Если gameOver равно значению true, то Шаг 3, иначе Шаг 4
Шаг 3. Переходим в функцию отрисовски сообщения о поражении finishGame(&painter, "Game lost")
Шаг 4. Если gameWon равно значению true, то Шаг 6, иначе Шаг 7
Шаг 5. Переходим в функцию отрисовски сообщения о победе `finishGame(&painter, "Victory")`
Шаг 6. Продолжаем отрисовку объектов : `drawObjects(&painter)`
Шаг 7. Конец

4.2.2 Алгоритм рисования блоков.
Метод void Breakout::drawObjects(QPainter *painter))

Шаг 1. Вызываем метод `drawImage(ball->getRect(),ball->getImage())`
Шаг 2. Вызываем метод `drawImage(ball->getRect(),paddle->getImage())` у painter
Шаг 3. Цикл `for int i=0; i < bonuses.size(); i++`
Шаг 4. Если бонус активен: то вызываем метод у painter, `painter ->drawImage(bonuses[i]->getRect(), bonuses[i]->getImage())` Иначе Шаг 5
Шаг 5. Конец цикла с шага 3
Шаг 6. Вызываем метод отрисовки счета drawScore(painter) у painter
Шаг 7. Цикл `for int i=0; i < bricks.size(); i++`
Шаг 8. Если блок не разрушен: то вызываем метод у painter, `painter ->drawImage(bricks[i]->getRect(), bricks[i]->getImage())` Иначе Шаг 9
Шаг 9. Конец цикла с шага 7
Шаг 10. Конец

4.2.3 Алгоритм столкновений игрока с блоками. Метод void Breakout::checkCollision()

Шаг 1. Если `ball -> getRect().bottom()` больше BOTTOM_EDGE то вызываем метод void Breakout::stopGame()

Шаг 2. Если `ball -> getRect(). bottom()` меньше 0, то Шаг 3, иначе Шаг 7

Шаг 3. Присваиваем указателю типа QRandomGenerator значение `QRandomGenerator::global()`

Шаг 4. Создаём переменную e типа int и присваиваем ей значение `rg -> bounded(1, 10)`

Шаг 5. Если переменная e чётная `ball -> setXDir(-1)` – шарик оттолкнётся влево , иначе `ball -> setXDir(1)` – шарик оттолкнётся вправо

Шаг 6. `ball -> setYDir(+2)` – устанавливаем скорость шарика по оси Y вниз

Шаг 7. Цикл `for int i=0, j=0; i< bricks.size(); i++`

Шаг 8. Если `brick[i] ->isDestroyed()` вернул true, то j++

Шаг 9. Если переменная j равна числу блоков (N_OF_BRICKS) вызываем метод `void Breakout: victory ()`

Шаг 10. Конец цикла с шага 7

Шаг 11. Если мячик касается платформы: `(ball->getRect()).intersects(paddle->getRect())` равно true, то Шаг 12 иначе Шаг 33

Шаг 12. Создаем переменную paddleLPos типа int равную `paddle->getRect().left()`

Шаг 13. Создаем переменную ballLPos типа int равную `ball->getRect().left()`

Шаг 14. Создаем переменную first типа int равную paddleLPos + 3

Шаг 15. Создаем переменную second типа int равную paddleLPos + 25

Шаг 16. Создаем переменную third типа int равную paddleLPos + 50

Шаг 17. Создаем переменную fourth типа int равную paddleLPos + 75

Шаг 18. Если ballLPos меньше first Шаг 19 иначе Шаг 21

Шаг 19. Ставим направление шарика по иксам в лево `ball->setXDir(-1)`

Шаг 20. Ставим направление шарика по игрекам в верх `ball->setYDir(-2)`

Шаг 21. Если ballLPos больше либо равен first и ballLPos меньше second Шаг 22 иначе Шаг 24

Шаг 22. Ставим направление шарика по иксам в лево `ball->setXDir(-1)`

Шаг 23. Ставим направление шарика по игрекам в противоположную сторону относительно его движения `ball->setYDir(-1*ball->getYDir())`

Шаг 24. Если ballLPos больше либо равен second и ballLPos меньше third (средняя часть ракетки) Шаг 25 иначе Шаг 27

Шаг 25. Ставим направление шарика по иксам в 0, т.е он двигается только

по игрекам `ball->setXDir(0)`

Шаг 26. Ставим направление шарика по игрекам `ball->setYDir(-3)`

Шаг 27. Если ballLPos больше либо равен third и ballLPos меньше fourth Шаг 28 иначе Шаг 30

Шаг 28. Ставим направление шарика по иксам в право `ball->setXDir(1)`

Шаг 29. Ставим направление шарика по игрекам в противоположную сторону относительно его движения `ball->setYDir(-1*ball->getYDir())`

Шаг 30. Если ballLPos больше fourth Шаг 31, иначе Шаг 33

Шаг 31. Ставим направление шарика по иксам в право `ball->setXDir(1)`

Шаг 32. Ставим направление шарика по игрекам в верх `ball->setYDir(-2)`

Шаг 33. Цикл `for int i=0; i < bricks.size(); i++`

Шаг 34. Если (`ball->getRect()).intersects(bricks[i]->getRect()`) равна true Шаг 35, иначе Шаг 64

Шаг 35. Создаем переменную ballLeft типа int равную `ball->getRect().left()`

Шаг 36. Создаем переменную ballHeight типа int равную `ball->getRect().height()`

Шаг 37. Создаем переменную ballWidth типа int равную `ball->getRect().width()`

Шаг 38. Создаем переменную ballTop типа int равную ball->getRect().top()

Шаг 39. Создаем объект pointRight типа QPoint равную `pointRight(ballLeft + ballWidth + 1, ballTop)`

Шаг 40. Создаем объект pointLeft типа QPoint равную `pointLeft(ballLeft - 1, ballTop)`

Шаг 41. Создаем объект pointTop типа QPoint равную `pointTop(ballLeft, ballTop -1)`

Шаг 42. Создаем объект pointBottom типа QPoint равную `pointBottom(ballLeft, ballTop + ballHeight + 1)`

Шаг 43. Если i-тый блок не разрушен то Шаг 44, иначе Шаг 64

Шаг 44. Если i-тый блок содержит правую точку шарика, то ставим его направление в лево: `bricks[i]->getRect().contains(pointRight)` Шаг 45 иначе Шаг 46

Шаг 45. `ball->setXDir(-1)`

Шаг 46. Если i-тый блок содержит левую точку шарика, то ставим его направление в право: `bricks[i]->getRect().contains(pointLeft)` Шаг 47 иначе Шаг 48

Шаг 47. `ball->setXDir(1)`

Шаг 48. Если i-тый блок содержит верхнюю точку шарика то ставим его направление в низ: `bricks[i]->getRect().contains(pointTop)` Шаг 49 иначе Шаг 50

Шаг 49. ball->setYDir(2)

Шаг 50. Если i-тый блок содержит нижнюю точку шарика то ставим его направление в верх: `bricks[i]->getRect().contains(pointBottom)` Шаг 51 иначе Шаг 52

Шаг 51. `ball->setYDir(-2)`

Шаг 52. Вызываем метод destroy() i-того элемента массива

Шаг 53. Увеличиваем переменную счёта score на возращенное значение метода `bricks[i]->getScore()`

Шаг 54. Конец цикла с шага 33

Шаг 55. Конец

5 РЕЗУЛЬТАТ РАБОТЫ

5.1 Использование приложения

Для запуска программы необходим открыть файлы исходного кода в Qt Creator и собрать проект. Информация по созданию проекта находится в файле product.db. После этого откроется окно программы с главным меню (рисунок 5.1).

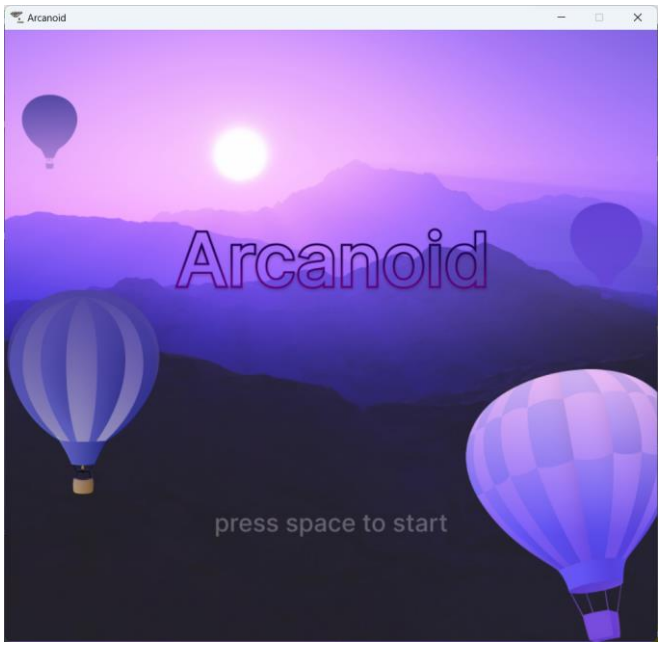

Рисунок 5.1 – Главное меню

При нажатии на кнопку откроется окно 800x750 с игровым полем. Игровое поле состоит из блоков трёх видов: красного, жёлтого и синего цветов, платформы и счётика очков (рисунок 5.2).

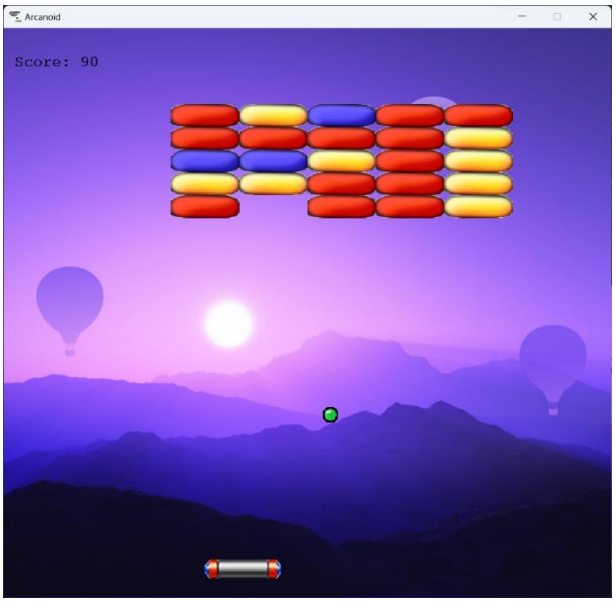

Рисунок 5.2 – Игра

После завершения игры открывается окно “Victory”, котрое содержит информацию о количестве очков, набранных во время игры. Это окно является конечным этапом игры (рисунок 5.3).

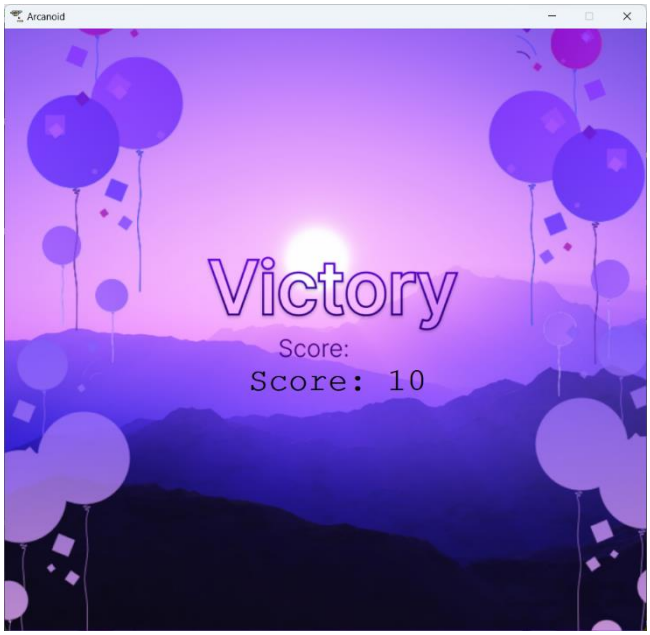

Рисунок 5.3 – Каталог менеджера

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта было изучено и реализована игра «Арканоид». Использовано большое количество текстовой и графической информации, в результате чего стало возможным проектирование программного продукта, его тестирование и устранение ошибок.

Для реализации данной игры был выбран язык программирования C++, так как он имеет следующие преимущества:

1. Высокая производительность: C++ позволяет создавать высокопроизводительные игры благодаря возможности оптимизации кода и использованию низкоуровневых операций.
2. Управление ресурсами: C++ предоставляет возможность более точного управления ресурсами, такими как память и процессорное время, что особенно важно для игр.
3. Широкие возможности: C++ обладает богатыми возможностями для создания сложных игровых механик, включая работу с графикой, физикой и звуком.
4. Поддержка библиотек: Существует множество библиотек и фреймворков на C++, специально разработанных для создания игр, что упрощает процесс разработки.
5. Переносимость: Игра, написанная на C++, может быть легко портирована на различные платформы, что позволяет достичь широкой аудитории игроков.

В рамках этого курсового проекта по игре «Арканоид» можно предложить несколько направлений для усовершенствования проекта. Во-первых, мы можем добавить новые механики в игру, такие как различные мини-игры и мутаторы для ракетки, чтобы сделать геймплей более разнообразным и интересным.

Также мы можем улучшить графические возможности игры, чтобы сделать её более привлекательной для игроков. Мы также можем добавить редактор уровней, чтобы игроки могли создавать свои собственные уровни и делиться ими с другими игроками.

Кроме того, мы можем добавить дополнительные бонусы в игру, чтобы игроки могли получать дополнительные возможности и улучшения во время игры. И наконец, мы можем добавить возможность смены скинов у шарика, чтобы игроки могли персонализировать свой игровой опыт.

Эти усовершенствования помогут сделать игру «Арканоид» более увлекательной и привлекательной для игроков, и мы надеемся, что они будут положительно восприняты нашими пользователями.

СПИСОК ЛИТЕРАТУРЫ

1. Страуструп, Б. Язык программирования C++/ Б.Страуструп .;
2. специальное издание. Пер. с англ. – Спб.: BHV, 2008. – 1098 с.
3. Официальный сайт документации Qt [Электронный ресурс].-2023-Режим доступа - https://doc.qt.io/qt-5/ - Дата доступа - 10.10.2023
4. Официальная документация CQtDeployer [Электронный ресурс].-2023-Режим доступа - https://github.com/QuasarApp/CQtDeployer - Дата доступа – 10.12.2023
5. Конструирование программ и языки программирования: метод. Указания по К65 курсовому проектированию для студ. I-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич , А. М. Ковальчук , И. В. Лукьянова. - Минск: БГУИР , 2009.
6. Объектно-ориентированное программирование на языке C++: учеб. Пособие /Ю. А. Луцик , В. Н. Комличенко. – Минск: БГУИР , 2008.
7. QT Documentation [Электронный ресурс].-2023-Режим доступа-https://doc.qt.io/all-topics.html - Дата доступа 01.10.2023

ПРИЛОЖЕНИЕ А

Листинг кода

ball.h:

```
#pragma once
#include <QImage>
#include <QRect>
class Ball {
public:
Ball();
~Ball();
void resetState();
void autoMove();
void setXDir(int);
void setYDir(int);
int getXDir();
int getYDir();
QRect getRect();
QImage & getImage();
private:
int xdir;
int ydir;
QImage image;
QRect rect;
static const int INITIAL_X = 375;
static const int INITIAL_Y = 680;
static const int RIGHT_EDGE = 800;
};
```

ball.cpp:

```
#include <iostream>
#include "./Headers/ball.h"
Ball::Ball() {
xdir = 1;
ydir = -2;
image.load(":/img/ball3.png");
rect = image.rect();
resetState();
}
Ball::~Ball() {
std::cout << "Ball deleted" << std::endl;
}
void Ball::autoMove() {
rect.translate(xdir, ydir);
if (rect.left() == 0) {
xdir = 1;
}
if (rect.right() == RIGHT_EDGE) {
xdir = -1;
}
```

```
if (rect.top() == 0) {
ydir = 1;
}
}
void Ball::resetState() {
rect.moveTo(INITIAL_X, INITIAL_Y);
}
void Ball::setXDir(int x) {
xdir = x;
}
void Ball::setYDir(int y) {
ydir = y;
}
int Ball::getXDir() {
return xdir;
}
int Ball::getYDir() {
return ydir;
}
QRect Ball::getRect() {
return rect;
}
QImage & Ball::getImage() {
return image;
}
```

paddle.h:

```
#pragma once
#include <QImage>
#include <QRect>
class Paddle {
public:
Paddle();
~Paddle();
void resetState();
void move();
void move(int);
void setDx(int);
QRect getRect();
QImage & getImage();
private:
QImage image;
QRect rect;
int dx;
static const int INITIAL_X = 335;
static const int INITIAL_Y = 700;
};
```

paddle.cpp:

```
#include <iostream>
```

```
#include "./Headers/paddle.h"
Paddle::Paddle() {
dx = 0;
image.load(":/img/paddle2.png");
rect = image.rect();
resetState();
}
Paddle::~Paddle() {
std::cout << ("Paddle deleted") << std::endl;
}
void Paddle::setDx(int x) {
dx = x;
}
void Paddle::move() {
int x = rect.x() + dx;
int y = rect.top();
rect.moveTo(x, y);
}
void Paddle::move(int x) {
rect.moveTo(x, INITIAL_Y);
}
void Paddle::resetState() {
rect.moveTo(INITIAL_X, INITIAL_Y);
}
QRect Paddle::getRect() {
return rect;
}
QImage & Paddle::getImage() {
return image;
}
```

bonus.h:

```
#pragma once
#include <QImage>
#include <QRect>
class Bonus
{
public:
Bonus(int, int, int);
~Bonus();
void autoMove();
void deleteObject();
void resetState();
bool isDestroyed();
int getBonus(int);
QRect getRect();
QImage & getImage();
private:
int xdir;
```

```
int ydir;
int typeBonus;
bool destroyed;
QImage image;
QRect rect;
int INITIAL_X;
int INITIAL_Y;
};
```

bonus.cpp:

```
#include <iostream>
#include "./Headers/Bonus.h"
Bonus::Bonus(int x, int y, int type) {
INITIAL_X = x + 35;
INITIAL_Y = y;
typeBonus = type;
xdir = 0;
ydir = +1;
destroyed = true;
image.load(":/img/bonus.png");
rect = image.rect();
resetState();
}
Bonus::~Bonus(){
std::cout<<"Bonus deleted"<<std::endl;
}
void Bonus::deleteObject(){
delete this;
}
void Bonus::resetState() {
rect.moveTo(INITIAL_X, INITIAL_Y);
}
bool Bonus::isDestroyed(){
return destroyed;
}
QRect Bonus::getRect() {
return rect;
}
QImage & Bonus::getImage() {
return image;
}
void Bonus::autoMove() {
rect.translate(xdir, ydir);
}
int Bonus::getBonus(int score){
switch (typeBonus) {
case 1:
score *=2;
break;
}
```

```
return score;
}
```
yellowBrick.h:
```
#pragma once
#include <QImage>
#include <QRect>
class YellowBrick
{
public:
YellowBrick(int, int);
~YellowBrick();
bool isDestroyed();
void setDestroyed(bool);
QRect getRect();
void setRect(QRect);
QImage & getImage();
int getXDir();
int getYDir();
int getScore();
virtual void destroy();
virtual void deleteObject();
protected:
QImage image;
QRect rect;
int score = 10;
bool destroyed;
int xdir;
int ydir;
};
```
yellowBrick.cpp:
```
#include <iostream>
#include "./Headers/YellowBrick.h"
YellowBrick::YellowBrick(int x, int y){
image.load(":/img/brick_yellow.png");
xdir = x;
ydir = y;
destroyed = false;
rect = image.rect();
rect.translate(x, y);
}
YellowBrick::~YellowBrick(){
std::cout << "YellowBrick deleted" << std::endl;
}
QRect YellowBrick::getRect() {
return rect;
}
void YellowBrick::setRect(QRect rct) {
rect = rct;
```

```
}
QImage & YellowBrick::getImage() {
return image;
}
bool YellowBrick::isDestroyed() {
return destroyed;
}
void YellowBrick::setDestroyed(bool destroy) {
destroyed = destroy;
}
int YellowBrick::getXDir(){
return xdir;
}
int YellowBrick::getYDir(){
return ydir;
}
int YellowBrick::getScore(){
if (destroyed)
return score;
else
return 0;
}
void YellowBrick::destroy(){
destroyed = true;
}
void YellowBrick::deleteObject(){
delete this;
}
```

redBrick.h:

```
#pragma once
#include <QRect>
#include "./Headers/YellowBrick.h"
class RedBrick : public YellowBrick
{
public:
RedBrick(int, int);
~RedBrick();
void destroy() override;
void deleteObject() override;
protected:
int stabillity = 1;
};
```

redBrick.cpp:

```
#include <iostream>
#include "./Headers/RedBrick.h"
RedBrick::RedBrick(int x, int y) : YellowBrick(x,y){
image.load(":/img/brick_red.png");
xdir = x;
```

```
ydir = y;
score = 20;
destroyed = false;
rect = image.rect();
rect.translate(x, y);
}
RedBrick::~RedBrick(){
std::cout << "RedBrick deleted" << std::endl;
}
void RedBrick::destroy(){
if(stabillity == 1){
image.load(":/img/brick_yellow.png");
stabillity = 0;
}
else
destroyed = true;
}
void RedBrick::deleteObject(){
delete this;
}
```

blueBrick.h:

```
#pragma once
#include <QRect>
#include <QRandomGenerator>
#include "./Headers/RedBrick.h"
#include "./Headers/Bonus.h"
class BlueBrick : public RedBrick
{
public:
BlueBrick(int, int);
~BlueBrick();
void destroy() override;
void deleteObject() override;
bool getActiveBonus();
Bonus getBonus();
protected:
int stabillity = 2;
Bonus *bonus;
bool activeBonus;
};
```

blueBrick.cpp:

```
#include <iostream>
#include "./Headers/BlueBrick.h"
BlueBrick::BlueBrick(int x, int y) : RedBrick(x,y){
image.load(":/img/brick_blue.png");
xdir = x;
ydir = y;
QRandomGenerator *rg = QRandomGenerator::global();
```

```
if (rg->bounded(1, 10) % 2 == 0){
bonus = new Bonus(x,y,1);
activeBonus = true;
}else
activeBonus = false;
score = 50;
destroyed = false;
rect = image.rect();
rect.translate(x, y);
}
BlueBrick::~BlueBrick(){
std::cout << "BlueBrick deleted" << std::endl;
}
Bonus BlueBrick::getBonus(){
return *bonus;
}
bool BlueBrick::getActiveBonus(){
if(destroyed)
return activeBonus;
else
return false;
}
void BlueBrick::destroy(){
switch (stabillity) {
case 2:
image.load(":/img/brick_blue1.png");
stabillity--;
break;
case 1:
image.load(":/img/brick_blue2.png");
stabillity--;
break;
case 0:
destroyed = true;
break;
}
}
void BlueBrick::deleteObject(){
delete this;
}
```

breakout.h:

```
#pragma once
#include <QWidget>
#include <QVector>
#include <QKeyEvent>
#include "ball.h"
#include "paddle.h"
#include "BlueBrick.h"
#include "YellowBrick.h"
```

```
#include "RedBrick.h"
#include "Bonus.h"
#include "list.h"
class Breakout : public QWidget {
public:
Breakout(QWidget *parent = 0);
~Breakout();
protected:
void paintEvent(QPaintEvent *);
void timerEvent(QTimerEvent *);
void keyPressEvent(QKeyEvent *);
void keyReleaseEvent(QKeyEvent *);
void drawObjects(QPainter *);
void finishGame(QPainter *, QString);
void moveObjects();
void drawScore(QPainter *,int,int,int);
void startGame();
void pauseGame();
void stopGame();
void victory();
void checkCollision();
private:
int x;
int score = 0;
int timerId;
static const int DELAY = 10;
static const int BOTTOM_EDGE = 750;
Ball *ball;
Paddle *paddle;
List<YellowBrick*> yellowBricks;
List<RedBrick*> redBricks;
List<BlueBrick*> blueBricks;
List<Bonus*> bonuses;
bool gameOver;
bool gameWon;
bool gameStarted;
bool paused;
bool gameMenu;
};
```

breakout.cpp:

```
#include <QPainter>
#include <QApplication>
#include "./Headers/breakout.h"
#include <QRandomGenerator>
#include <QDateTime>
#include <QCoreApplication>
#include <QDateTime>
#include <iostream>
Breakout::Breakout(QWidget *parent): QWidget(parent) {
```

```
QPixmap background(":/img/backgroundMenu.png");
QPalette palette;
palette.setBrush(QPalette::Window, background);
this->setPalette(palette);
x = 0;
gameOver = false;
gameWon = false;
paused = false;
gameMenu = true;
gameStarted = false;
ball = new Ball();
paddle = new Paddle();
}
Breakout::~Breakout() {
delete ball;
delete paddle;
yellowBricks.clearList();
}
void Breakout::paintEvent(QPaintEvent *event) {
Q_UNUSED(event);
QPainter painter(this);
if (gameOver) {
finishGame(&painter, "Game lost");
}
else if(gameWon) {
finishGame(&painter, "Victory");
}
else {
drawObjects(&painter);
}
}
void Breakout::finishGame(QPainter *painter, QString message) {
if(QString::compare(message,"Game lost") != 0)
{
QPixmap background(":/img/backgroundGameLost.png");
QPalette palette;
palette.setBrush(QPalette::Window, background);
this->setPalette(palette);
drawScore(painter,300,450, 30);
}
else
{
QPixmap background(":/img/backgroundVictory.png");
QPalette palette;
palette.setBrush(QPalette::Window, background);
this->setPalette(palette);
drawScore(painter,300,450,30);
}
}
```

```
void Breakout::drawObjects(QPainter *painter) {
if(!gameMenu)
{
painter->drawImage(ball->getRect(), ball->getImage());
painter->drawImage(paddle->getRect(), paddle->getImage());
for (int i = 0; i < bonuses.getSize(); ++i) {
if((bonuses.peek(i))->isDestroyed())
painter->drawImage((bonuses.peek(i))->getRect(),
(bonuses.peek(i))->getImage());
}
drawScore(painter,10,50,15);
for (int i=0; i<yellowBricks.getSize(); i++) {
if (!(yellowBricks.peek(i))->isDestroyed()) {
painter->drawImage((yellowBricks.peek(i))->getRect(),
(yellowBricks.peek(i))->getImage());
}
}
for (int i=0; i<redBricks.getSize(); i++) {
if (!(redBricks.peek(i))->isDestroyed()) {
painter->drawImage((redBricks.peek(i))->getRect(),
(redBricks.peek(i))->getImage());
}
}
for (int i=0; i<blueBricks.getSize(); i++) {
if (!(blueBricks.peek(i))->isDestroyed()) {
painter->drawImage((blueBricks.peek(i))->getRect(),
(blueBricks.peek(i))->getImage());
}
}
}
}
void Breakout::timerEvent(QTimerEvent *event) {
Q_UNUSED(event);
moveObjects();
checkCollision();
repaint();
}
void Breakout::moveObjects() {
ball->autoMove();
paddle->move();
for (int i = 0; i < bonuses.getSize(); ++i) {
(bonuses.peek(i))->autoMove();
}
}
void Breakout::keyReleaseEvent(QKeyEvent *event) {
if(event->key() == Qt::Key_Left || event->key() == Qt::Key_A)
{
if(paddle->getRect().left() <= 10)
paddle->move(10);
```

```
paddle->setDx(0);
}
if(event->key() == Qt::Key_Right || event->key() == Qt::Key_D)
{
if(paddle->getRect().right() >= 790)
paddle->move(690);
paddle->setDx(0);
}
}
void Breakout::keyPressEvent(QKeyEvent *event) {
if(event->key() == Qt::Key_Left || event->key() == Qt::Key_A)
{
if(paddle->getRect().left() <= 10) {
paddle->setDx(0);
paddle->move(10);
}
else
paddle->setDx(-2);
}
if(event->key() == Qt::Key_Right || event->key() == Qt::Key_D)
{
if(paddle->getRect().right() >= 790) {
paddle->setDx(0);
paddle->move(690);
}
else
paddle->setDx(2);
}
switch (event->key()) {
case Qt::Key_P:
pauseGame();
break;
case Qt::Key_Space:
startGame();
break;
case Qt::Key_Escape:
qApp->exit();
break;
default:
QWidget::keyPressEvent(event);
}
}
void Breakout::startGame() {
if (!gameStarted) {
QPixmap background(":/img/background.png");
QPalette palette;
palette.setBrush(QPalette::Window, background);
this->setPalette(palette);
ball->resetState();
```

```
paddle->resetState();
yellowBricks.clearList();
redBricks.clearList();
blueBricks.clearList();
bonuses.clearList();
QRandomGenerator *rg = QRandomGenerator::global();
for (int i=0; i<5; i++) {
for (int j=0; j<6; j++) {
if (rg->bounded(1, 10) % 2 == 0)
yellowBricks.pushBack(new YellowBrick(j * 90 + 130, i * 30 +
100));
else {
redBricks.pushBack(new RedBrick(j * 90 + 130, i * 30 + 100));
if (rg->bounded(1, 10) %2 == 0){
redBricks.popBack();
blueBricks.pushBack(new BlueBrick(j * 90 + 130, i * 30 + 100));
}
}
}
}
gameOver = false;
gameWon = false;
gameStarted = true;
gameMenu = false;
timerId = startTimer(DELAY);
}
}
void Breakout::pauseGame() {
if (paused) {
timerId = startTimer(DELAY);
paused = false;
}
else {
paused = true;
std::cout << "Pause" << std::endl;
killTimer(timerId);
}
}
void Breakout::stopGame() {
killTimer(timerId);
gameOver = true;
gameStarted = false;
}
void Breakout::victory() {
killTimer(timerId);
gameWon = true;
gameStarted = false;
}
```

```
void Breakout::drawScore(QPainter *painter, int x, int y, int
size){
QString stringNumber;
std::cout << score<<std::endl;
QString messageScore = "Score: " + stringNumber.setNum(score);
QFont font("Courier", size, QFont::DemiBold);
QFontMetrics fm(font);
painter->setFont(font);
painter->translate(QPoint(0, 0));
painter->drawText(messageScore.length()/2 + x, y, messageScore);
}
void Breakout::checkCollision() {
if (ball->getRect().bottom() > BOTTOM_EDGE) {
stopGame();
}
for (int i = 0; i < bonuses.getSize(); ++i)
if((bonuses.peek(i))->getRect().bottom() > BOTTOM_EDGE)
bonuses.erase(i);
for (int i = 0; i < bonuses.getSize(); ++i) {
if(((bonuses.peek(i))->getRect()).intersects(paddle-
>getRect())){
score = (bonuses.peek(i))->getBonus(score);
bonuses.erase(i);
}
}
if (ball->getRect().bottom() < 0) {
QRandomGenerator *rg = QRandomGenerator::global();
if (rg->bounded(1, 10) % 2 == 0) {
ball->setXDir(-1);
}
else {
ball->setXDir(1);
}
ball->setYDir(+2);
}
int numberOfBlocks = 0;
for (int i=0; i<yellowBricks.getSize(); i++)
if ((yellowBricks.peek(i))->isDestroyed())
numberOfBlocks++;
for (int i=0; i<redBricks.getSize(); i++)
if ((redBricks.peek(i))->isDestroyed())
numberOfBlocks++;
for (int i=0; i<blueBricks.getSize(); i++)
if ((blueBricks.peek(i))->isDestroyed())
numberOfBlocks++;
if (numberOfBlocks == 30) {
victory();
}
if ((ball->getRect()).intersects(paddle->getRect())) {
```

```
int paddleLPos = paddle->getRect().left();
int ballLPos = ball->getRect().left();
int first = paddleLPos + 3;
int second = paddleLPos + 25;
int third = paddleLPos + 50;
int fourth = paddleLPos + 75;
if (ballLPos < first) {
ball->setXDir(-1);
ball->setYDir(-2);
}
if (ballLPos >= first && ballLPos < second) {
ball->setXDir(-1);
ball->setYDir(-1*ball->getYDir());
}
if (ballLPos >= second && ballLPos < third) {
ball->setXDir(0);
ball->setYDir(-3);
}
if (ballLPos >= third && ballLPos < fourth) {
ball->setXDir(1);
ball->setYDir(-1*ball->getYDir());
}
if (ballLPos > fourth) {
ball->setXDir(1);
ball->setYDir(-2);
}
}
for (int i=0; i<yellowBricks.getSize(); i++) {
if ((ball->getRect()).intersects((yellowBricks.peek(i))-
>getRect())) {
int ballLeft = ball->getRect().left();
int ballHeight = ball->getRect().height();
int ballWidth = ball->getRect().width();
int ballTop = ball->getRect().top();
QPoint pointRight(ballLeft + ballWidth + 1, ballTop);
QPoint pointLeft(ballLeft - 1, ballTop);
QPoint pointTop(ballLeft, ballTop -1);
QPoint pointBottom(ballLeft, ballTop + ballHeight + 1);
if (!(yellowBricks.peek(i))->isDestroyed()) {
//x
if((yellowBricks.peek(i))->getRect().contains(pointRight)) {
ball->setXDir(-1);
}
else if((yellowBricks.peek(i))->getRect().contains(pointLeft)) {
ball->setXDir(1);
}
//y
if((yellowBricks.peek(i))->getRect().contains(pointTop)) {
ball->setYDir(2);
```

```
}
else if((yellowBricks.peek(i))->getRect().contains(pointBottom))
{
ball->setYDir(-2);
}
(yellowBricks.peek(i))->destroy();
score += (yellowBricks.peek(i))->getScore();
}
}
}
for (int i=0; i<redBricks.getSize(); i++) {
if ((ball->getRect()).intersects((redBricks.peek(i))-
>getRect())) {
int ballLeft = ball->getRect().left();
int ballHeight = ball->getRect().height();
int ballWidth = ball->getRect().width();
int ballTop = ball->getRect().top();
QPoint pointRight(ballLeft + ballWidth + 1, ballTop);
QPoint pointLeft(ballLeft - 1, ballTop);
QPoint pointTop(ballLeft, ballTop -1);
QPoint pointBottom(ballLeft, ballTop + ballHeight + 1);
if (!(redBricks.peek(i))->isDestroyed()) {
//x
if((redBricks.peek(i))->getRect().contains(pointRight)) {
ball->setXDir(-1);
}
else if((redBricks.peek(i))->getRect().contains(pointLeft)) {
ball->setXDir(1);
}
//y
if((redBricks.peek(i))->getRect().contains(pointTop)) {
ball->setYDir(2);
}
else if((redBricks.peek(i))->getRect().contains(pointBottom)) {
ball->setYDir(-2);
}
(redBricks.peek(i))->destroy();
score += (redBricks.peek(i))->getScore();
}
}
}
for (int i=0; i<blueBricks.getSize(); i++) {
if ((ball->getRect()).intersects((blueBricks.peek(i))-
>getRect())) {
int ballLeft = ball->getRect().left();
int ballHeight = ball->getRect().height();
int ballWidth = ball->getRect().width();
int ballTop = ball->getRect().top();
QPoint pointRight(ballLeft + ballWidth + 1, ballTop);
```

```
QPoint pointLeft(ballLeft - 1, ballTop);
QPoint pointTop(ballLeft, ballTop -1);
QPoint pointBottom(ballLeft, ballTop + ballHeight + 1);
if (!(blueBricks.peek(i))->isDestroyed()) {
//x
if((blueBricks.peek(i))->getRect().contains(pointRight)) {
ball->setXDir(-1);
}
else if((blueBricks.peek(i))->getRect().contains(pointLeft)) {
ball->setXDir(1);
}
//y
if((blueBricks.peek(i))->getRect().contains(pointTop)) {
ball->setYDir(2);
}
else if((blueBricks.peek(i))->getRect().contains(pointBottom)) {
ball->setYDir(-2);
}
(blueBricks.peek(i))->destroy();
score += (blueBricks.peek(i))->getScore();
if((blueBricks.peek(i))->getActiveBonus()){
bonuses.pushBack(new Bonus((blueBricks.peek(i))-
>getXDir(),(blueBricks.peek(i))->getYDir(),1));
//bonuses.push_back(blueBricks[i]->getBonus());
}
}
}
}
}
```

list.h:

```
#pragma once
#include <iostream>
#include <conio.h>
#include <iostream>
#include <iomanip>
#include "./Headers/NoSuchElementException.h"
template<typename T>
class List
{
public:
List();
~List();
List(const List&);
void pushBack(T data);
int getSize();
void erase(int num);
void clearList();
void pushFront(T data);
void popFront();
```

```
void popBack();
T& peek(int num);
private:
struct Node
{
T data;
Node* next, * prev;
Node(T data = T(), Node* next = nullptr, Node* prev = nullptr)
{
this->data = data;
this->next = next;
this->prev = prev;
}
};
Node* head;
int Size;
};
template<typename T>
List<T>::List()
{
Size = 0;
head = nullptr;
}
template<typename T>
List<T>::~List()
{
while (Size)
{
Node* temp = head;
head = head->next;
delete temp;
Size--;
}
}
template<typename T>
List<T>::List(const List<T>& copy)
{
try
{
Node* temp_cpy = copy.head;
head = new Node(temp_cpy->data);
temp_cpy = temp_cpy->next;
Node* temp = head;
for (int i = 0; i < copy.Size - 1; i++)
{
temp->next = new Node(temp_cpy->data, nullptr, temp);
temp = temp->next;
temp_cpy = temp_cpy->next;
}
```

```
}
catch (bad_alloc& e)
{
std::cout << e.what() << std::endl;
}
this->Size = copy.Size;
}
template<typename T>
void List<T>::pushBack(T data)
{
if (head == nullptr)
{
try
{
head = new Node(data);
}
catch (bad_alloc& e)
{
std::cout << e.what() << std::endl;
}
}
else
{
try
{
Node* temp = this->head;
while (temp->next != nullptr)
{
temp = temp->next;
}
temp->next = new Node(data, nullptr, temp);
}
catch (bad_alloc& e)
{
std::cout << e.what() << std::endl;
}
}
Size++;
}
template<typename T>
inline int List<T>::getSize()
{
return Size;
}
template<typename T>
void List<T>::erase(int num)//удаление по индексу
{
if (!Size)
return;
```

```
Node* temp = head;
try {
if (num > this->Size || num < 0)
throw NoSuchElementException();
}
catch (NoSuchElementException& exp)
{
exp.show();
return;
}
if (num == 0)
{
head = head->next;
head->prev = nullptr;
delete temp;
Size--;
return;
}
for (int i = 0; i < num; i++)
{
temp = temp->next;
}
temp->prev->next = temp->next;
temp->next = temp->prev;
Size--;
delete temp;
}
template<typename T>
void List<T>::clearList()//очистка очереди
{
if (!Size) return;
while (Size)
{
Node* temp = head;
head = head->next;
delete temp;
Size--;
}
}
template<typename T>
void List<T>::pushFront(T data)
{
try
{
if (head == nullptr)
{
head = new Node(data);
}
else
```

```
{
this->head->prev = new Node(data, head);
head = head->prev;
}
}
catch (bad_alloc& e)
{
std::cout << e.what() << std::endl;
}
Size++;
}
template<typename T>
void List<T>::popFront()
{
this->erase(0);
}
template<typename T>
void List<T>::popBack()
{
this->erase(this->Size - 1);
}
template<typename T>
T& List<T>::peek(int num)
{
if (this->Size == 0)
{
cout << "empty list" << endl;
exit(1);
}
Node* temp = this->head;
for (int i = 0; i < num; i++)temp = temp->next;
return temp->data;
}
```

MyException.h:

```
#pragma once
#include<cstring>
#include<iostream>
class MyException
{
protected:
char message[80];
int code;
public:
MyException(const char* message);
MyException();
virtual void show();
};
```

MyException.cpp:

```
#include "./Headers/MyException.h"
MyException::MyException(const char * message) {
strcpy_s(this->message, message);
this->code = code;
}
MyException::MyException() {
std::cout << "Exception was generated" << std::endl;
}
void MyException::show() {
std::cout << "Base exception was generated" << std::endl;
}
```

IOException.h:

```
#pragma once
#include "./Headers/MyException.h"
class IOException : public MyException
{
public :
IOException(int code,const char* message);
IOException();
void show() override;
int getCode();
};
```

IOException.cpp:

```
#include "./Headers/IOException.h"
IOException::IOException(int code,const char * message) :
MyException(message){
this->code = code;
}
IOException::IOException() {
this->code = 2;
}
int IOException::getCode() {
return code;
}
void IOException::show() {
std::cout << "IOException was generated" << std::endl;
std::cout << "Code: " << this->getCode() << std::endl;
}
```

FileNotFoundException.h:

```
#pragma once
#include "./Headers/IOException.h"
class FileNotFoundException : public IOException{
FileNotFoundException(const char* message,int code);
FileNotFoundException();
int getCode();
void show() override;
};
```

FileNotFoundException.cpp:

```
#include "./Headers/FileNotFoundException.h"
FileNotFoundException::FileNotFoundException(const char*
message, int code) : IOException(code, message) {
this->code = code;
}
FileNotFoundException::FileNotFoundException() {
this->code = 3;
}
void FileNotFoundException::show() {
std::cout << "FileNotFoundException was generated" << std::endl;
std::cout << "Code: " << this->getCode() << std::endl;
}
int FileNotFoundException::getCode() {
return code;
}
```

NoSuchElementException.h:

```
#pragma once
#include "./Headers/MyException.h"
class NoSuchElementException : public MyException
{
public:
NoSuchElementException(int code, const char* message);
NoSuchElementException();
void show() override;
int getCode();
};
```

NoSuchElementException.cpp:

```
#include "./Headers/NoSuchElementException.h"
NoSuchElementException::NoSuchElementException(int code, const
char* message) : MyException(message) {
this->code = code;
}
NoSuchElementException::NoSuchElementException() {
this->code = 1;
}
int NoSuchElementException::getCode() {
return code;}
void NoSuchElementException::show() {
std::cout << "NoSuchElementException was generated" <<
std::endl;
std::cout << "Code: " << this->getCode() << std::endl;}
```

MainWindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
```

```
class MainWindow : public QMainWindow
{
Q_OBJECT
public:
MainWindow(QWidget *parent = nullptr);
~MainWindow();
private:
Ui::MainWindow *ui;
int time;
QTimer *timer;
private slots:
void TimerSlot();
};
#endif // MAINWINDOW_H
```

MainWindow.cpp:

```
#include "./Headers/mainwindow.h"
#include "./Headers/ui_mainwindow.h"
#include <QTimer>
MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
ui->setupUi(this);
time = 0;
timer = new QTimer(this);
connect(timer, SIGNAL(timeout()),this,SLOT(TimerSlot()));
timer->start(1);
}
MainWindow::~MainWindow()
{
delete ui;
}
void MainWindow::TimerSlot(){
time++;
ui->label->setText(QString::number(time));}
```

Main.cpp:

```
#include <QIcon>
#include <QApplication>
#include "./Headers/breakout.h"
int main(int argc, char *argv[]) {
QApplication app(argc, argv);
app.setWindowIcon(QIcon(":/img/breakout.png"));
Breakout window;
window.setFixedSize(QSize(800, 750));//660 900
window.setWindowTitle("Arcanoid");
window.show();
return app.exec();
}
```

ПРИЛОЖЕНИЕ Б

Диаграмма классов

ПРИЛОЖЕНИЕ В

Схема метода keyPressEvent(QKeyEvent *e)

ПРИЛОЖЕНИЕ Г

Схема метода keyReleaseEvent(QKeyEvent *e)

ПРИЛОЖЕНИЕ Д

Ведомость документов