Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Игра Pacman

БГУИР КП 1-40 02 01 116 ПЗ

Студент: гр. 650501 Никанов И.В.

Руководитель: старший преподаватель Ковальчук А. М.

Минск 2017

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    *Никанову Ивану Владимировичу –––––––––––––––––––*

1. Тема проекта   Игра Pacman –

2. Срок сдачи студентом законченного проекта–––––*21 декабря 2017 г.––––*

3. Исходные данные к проекту *файлы с изображениями персонажей и карты; запись информации в файл; среда разработки – QT Creator(кросс-платформенная)–*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение. 1. Постановка задачи. 2. Структура входных и выходных данных. 3. Описание алгоритмов. 4.Разработка программного обеспечения. 5. Результаты работы программы. 6. Руководство пользователя. Заключение. Литература. Приложения.*

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

*1. Диаграмма классов 2. Схема алгоритмов (2) шт – чертёж*

6. Консультант по проекту (с обозначением разделов проекта)   *Ковальчук А. М.*

7. Дата выдачи задания –––––*15 сентября 2017 г.–––––––––––––––––––––––*

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

*разделы 1 к 10.10 – 15 %;–––––––––––––––––––––– –––––––––––––––––*

*раздел 2 к 01.11 – 15 %;–––––––––––––––––––––– –––––––––––––––––––*

*разделы 3,4 к 20.11 – 45 %;––––––––––––––––––––– –––––––––––––––––––*

*разделы 5, 6 к 01.12 – 10 %;––––––––––––––––––––– –––––––––––––––––––*

*оформление пояснительной записки и графического материала к 04.12 – 15 %*

*Защита курсового проекта с 21 по 29 декабря––––––––––––––––––––––––––*

РУКОВОДИТЕЛЬ*––––––––– А. М. Ковальчук*

(подпись)

Задание принял к исполнению *–––––– –\_\_ \_ \_\_\_\_\_\_\_–– И.В,Никанов*

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5

1 обзор методов и алгоритмов решения поставленной задачи 6

2 ОПИСАНИЕ ВХОДНЫХ И ВЫХОДНЫХ ФАЙЛОВ 9

3 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА 10

4 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 11

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 11

ЗАКЛЮЧЕНИЕ 14

СПИСОК ЛИТЕРАТУРЫ 15

ПРИЛОЖЕНИЕ А 16

ПРИЛОЖЕНИЕ Б 17

ПРИЛОЖЕНИЕ В 18

ПРИЛОЖЕНИЕ Г 21

ВВЕДЕНИЕ

В настоящее время C++ является одним из языков программирования, наиболее чётко представляющих собой основные парадигмы программирования. Он поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование.

Целью настоящей курсовой работы является создание прикладного программного обеспечения (далее ПО). Это вероятно самая огромная область программирования, так как каждый человек использует прикладное ПО при работе с компьютером (например, когда слушает музыку, смотрит фильмы, печатает документы или ищет информацию в интернете). Для разработки прикладного ПО разработано множество библиотек и фреймворков. Прежде всего, стандартная библиотека языка C++, но помимо её существует множество других: SFML, SDL, Silicon, Qt и другие. В ходе курсовой работы графический интерфейс пользователя осуществляется при помощи средств фреймворка Qt, предоставляющего широкий спектр возможностей для создания прикладного ПО.

1 обзор методов и алгоритмов решения поставленной задачи

1.1 Обзор языка программирования

С++ – это компилируемый, статически типизированный язык программирования общего назначения.

Как уже отмечалось во введении, C++ поддерживает следующие парадигмы программирования:

– процедурное программирование;

– объектно-ориентированное программирование;

– обобщённое программирование.

Принцип ООП, взятый за основу в настоящей работе, основан на трёх основных концепциях: инкапсуляции, наследовании, полиморфизме. Программа в таком случае представлена в виде совокупности объектов, каждый из которых является экземпляром отдельного класса, а классы в совокупности образуют иерархию наследования. Действия программы представляются как свойства какого-либо объекта, либо как результат взаимодействия нескольких объектов одного или разных типов.

Язык C++ имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков, что делает его одним из самых мощных средств программирования, однако тем самым повышая его сложность. Но именно в связи с этим язык C++ является отличной базой для изучения программирования, основных парадигм и алгоритмов. Также в дальнейшем это способствует быстрому изучению программистом других языков, в частности C-подобных (Java, C# и других).

Язык C++ является популярным средством программирования, даже не смотря на то, что в последнее время лидирующие позиции по популярности перед C++ заняли такие языки программирования, как JavaScript, Java, PHP, Python, C#. Лидерство этих языков легко объясняется: они проще. Однако это отнюдь не является главным критерием выбора языка программирования. При помощи языка C++ можно разрабатывать огромный спектр продуктов: начиная от простых прикладных программ или компьютерных игр, заканчивая полноценными операционными системами или системами управления серверами. И, как правило, при разработке более крупных и важных проектов, где необходима надёжность и защищённость (например, при разработке программного или системного обеспечения для космической техники) программисты оставляю однозначный выбор за языком C++.

1.2 Обзор среды разработки

Qt Creator – это кроссплатформенная свободная интегрированная среда разработки, работающая с языками C и C++. Разработана компанией Qt Development Frameworks для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика GDB и визуальные средства разработки интерфейса. Для Windows-версий комплектуется компилятором, заголовочными и объектными файлами MinGW.

Qt Creator также включает в себя Qt Assistant – справочную систему Qt, упрощающую работу с документацией по библиотеке, а также позволяющую создавать кроссплатформенную справку для разрабатываемого на основе Qt программного обеспечения.

Основная задача Qt Creator — упростить разработку приложения с помощью фреймворка Qt на разных платформах. Поэтому среди возможностей, присущих любой среде разработки, есть и специфичные, такие как отладка приложений на QML и отображение в отладчике данных из контейнеров Qt, встроенный дизайнер интерфейсов как на QML, так и на QtWidgets.

1.3 Обзор фреймворка Qt

Qt – это кроссплатформенный фреймворк для разработки ПО на языке программирования C++. Qt позволяет запускать написанное с его помощью ПО на большинстве современных операционных систем путём простой компиляции программы для каждой из систем без изменения исходного кода программы. Фреймворк включает в себя библиотеки со всеми основными классами, которые могут потребоваться при разработке прикладного ПО, в частности при разработке элементов графического интерфейса, именно поэтому данный фреймворк был выбран для выполнения курсовой работы.

Отличительной особенностью фреймворка является использование метаобъектной системы – предварительной системы обработки кода.

Метаобъектная система – часть ядра фреймворка Qt для поддержки в С++ таких возможностей, как сигналы и слоты для коммуникации между объектами в режиме реального времени и динамических свойств системы. Метаобъектная система содержит:

– класс *QObject*;

– макрос *Q\_OBJECT*;

– утилиту *moc* (метаобъектный компилятор).

*QObject* – это базовый класс для всех Qt-классов. Макрос Q\_*OBJECT* используется для включения метаобъектных функций в классах и на этапе компиляции работает как препроцессор, который преобразует применения макроса в исходный код C++.

Таким образом, использование метаобъектной системы упрощает реализацию цикла обработки сообщений всех объектов. Связывая между собой разные объекты через определенные методы, присущие этим объектам (сигналы и слоты).

Сигнал ­– это метод объекта класса *QObject*, который вызывается в реальном времени в результате какого-либо непосредственного внешнего воздействия на объект (например, нажатие пользователь на кнопку, изменение текста в строке поиска и т. д.). Сигналы представляют собой методы класса, но в отличие от методов их не нужно определять, и они не могут быть вызваны как обычная функция. Для того, чтобы сигнал вызвался, он должен быть "выработан" (это реализуется при помощи ключевого слова *emit*) во время выполнения определённой логически связанной с данным сигналом функции. Например, сигнал *void valueChanged(int)* логично "вырабатывать" при вызове метода *void setValue(int)*. Сигнал во время вызова может отправлять информацию (как аргументы функции) о текущем состоянии объекта, свойством которого он является. Этот аргумент в последующем будет передан вызванному данным сигналом слоту. Сигналы по умолчанию являются константными методами класса (указывать при объявлении это не требуется), так как лишь передают информацию о состоянии.

Слот – это метод объекта класса *QObject*, который производит непосредственное действие в реальном времени над самим объектом (например, закрытия окна, очистка содержимого, обработка информации и т. д.) при определённых условиях (как правило, при вызове сигнала). Слот представляет собой обычный метод класса: он может иметь любое количество параметров, может также быть напрямую вызван, но в отличие от обычного метода он ещё может быть вызван определённым сигналом. Для того чтобы слот был вызван неким сигналом, он должен быть "связан" с ним, в таком случае, когда будет вызван связанный с данным слотом сигнал, то тогда будет вызван и сам слот. Слоты, как правило, являются не константными методами, так как они обычно производят обработку данных.

2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ФАЙЛОВ

Входные файлы представлены файлами формата PNG. Данные файлы предназначены для визуализации карты, главного героя, его врагов и представлены на рисунке 2.1

.

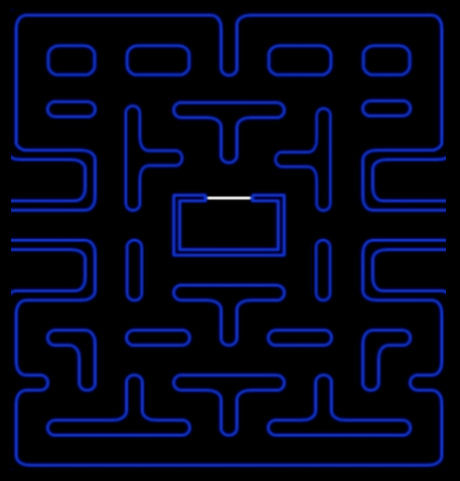


Рисунок 2.1

Выходной файл представлен текстовым файлом, в который заносится информация о предыдущих игроках. Содержимое файла users.txt представлено на рисунке 2.2.



Рисунок 2.2

Файл users.txt нельзя назвать лишь выходным, так как для отображения лучшего результата производится считывание данных из файла.

3 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА

Для построения диаграммы классов использовался язык UML.

UML— [язык](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [графического](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) описания для [объектного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) в области [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), [моделирования бизнес-процессов](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1), [системного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и отображения [организационных структур](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0).

UML является языком широкого профиля, это — [открытый стандарт](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82), использующий графические обозначения для создания [абстрактной модели](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) [системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна [генерация кода](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%BE%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F).

Разработанная диаграмма классов находится в Приложении А.

Описание алгоритма работы функции выводящей самый высокий результат (int get\_high\_score()) и функции обработки нажатия клавиши “Play Game” (void Options::on\_startButton\_clicked()) находятся в Приложении Б (чертёж 1 и чертёж 2 соответственно).

Описание алгоритмов по шагам находятся в Приложении В.

4 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

В результате работы программы, результат запишется в файл. Если данный результат окажется лучшим среди остальных, то он будет отображаться в меню игры.

Скриншоты работы программы представлены в следующем разделе.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Минимальные системные требования:

– операционная система: Windows 7 или Windows Vista;

– процессор: частота не менее 500 МГц;

– оперативная память: не менее 128 Мб;

– место на диске: не менее 32 Мб;

– дополнительно: размер приложений текста и других элементов не должен превышать 100%.

Для запуска программы необходимо открыть файл «Pacman.exe». После этого откроется окно программы, где необходимо ввести имя игрока и выбрать сложность (Рис. 5.1).



Рисунок 5.1 – Меню ввода данных

После успешного ввода данных откроется окно, где необходимо нажать определенную клавишу для старта игры. В нашем случае это клавиша Space (Рис 5.2). Также в данном окне можно узнать управление.



Рисунок 5.2 – Главное меню

Далее начинается сам игровой процесс: появляется игровое поле, главный герой (вы), а также враги (Рис. 5.3).

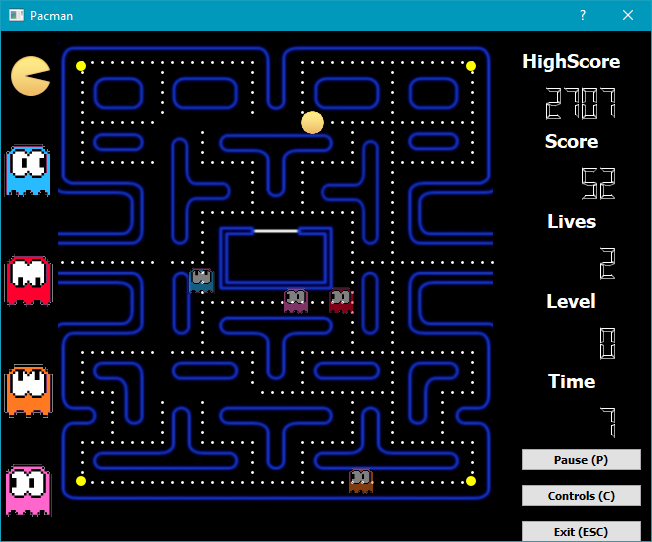


Рисунок 5.3 – Игровое поле

При поднятии Шаров Силы (Больших желтых шаров) вы становитесь неуязвимы для врагов и даже можете их сами убивать, после чего они будут возрождаться в центральной комнате (Рис. 5.4).

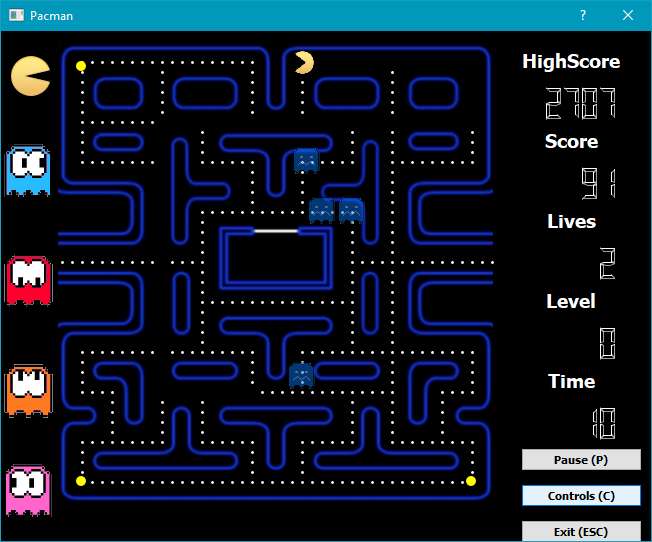


Рисунок 5.4 – Поднятие Шара Силы

После того, как вы потратите все жизни у вас будет выбор: либо начать игру заного, либо можно выйти нажатием клавиши Esc (Рис. 5.5).



Рисунок 5.5 – Меню после потери всех жизней

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было разработано приложение “Pacman”. Приложение не требует большого количества оперативной памяти и времени ожидания, имеет дружественный интерфейс и понятный функционал. При тестировании проекта была проведена корректная обработка ошибок, которые могут возникнуть в ходе эксплуатации приложения.

Для разработки использовалась кроссплатформенная среда Qt Creator.

Тематику приложения можно развивать во множестве направлений, одним из которых является поддержка сетевой игры. То игры со своими друзьями.

Хочется отметить, что данное приложение будет как детям, так и взрослым из-за своей простоты и наглядности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Луцик, Ю. А. Объектно-ориентированное программирование на языке С++ : учебное пособие / Ю. А. Луцик, А. М. Ковальчук, И. В. Лукьянова. – Мн. : БГУИР, 2003 – 203 с.
2. Шилдт, Г. Самоучитель C++ : Пер. с англ. – 3-е изд. – СПб. : БХВ-Петербург, 2006. – 688 с.
3. Standard C++ Library Reference [Электронный ресурс]. – Режим доступа : http://www.cplusplus.com/reference/.
4. C++ Reference [Электронный ресурс]. – Режим доступа : http://en.cppreference.com/w/.
5. Qt 5.10 Documentation [Электронный ресурс]. – Режим доступа : http://doc.qt.io/qt-5/
6. Дейтел, Х. М. Как программировать на С++ / Х. М. Дейтел, П.Д. Дейтел : пер. с англ. – Мск. : Бином, 2007.
7. Доманов, А. Т. Стандарт предприятия. Дипломные проекты (работы) : общие требования / А. Т. Доманов, Н. И. Сороко. – Мн. : БГУИР, 2013.

ПРИЛОЖЕНИЕ А

*(обязательное)*

Диаграмма классов

ПРИЛОЖЕНИЕ Б

*(обязательное)*

Описание алгоритмов (блок-схемы)

ПРИЛОЖЕНИЕ В

*(обязательное)*

Описание алгоритмов (по шагам)

Алгоритм функции**- void insert(const Iterator &past, const T &past\_data)** класса **List<T>.** Является функцией шаблонного класса, предназначена для вставки элемента с данными past\_data в любое место в списке.

Шаг 1: Начало.

Шаг 2: Объявляется переменная **flag** типа bool и инициализируется значением 1.

Шаг 3: Создаем объект **temp** класса Iterator при помощи конструктора по умолчанию и инициализируем результатом вызова функции this->begin().

Шаг 3: Цикл while по переменной temp c выходным условием temp!= this->end().

Шаг 4: Если временная переменная итератора temp равна переменной past, предполагаемому месту вставки элемента, то перейти к шагу.

Шаг 5: Конец цикла while по переменной temp.

Шаг 6: Если временная переменная итератора temp равна переменной past, предполагаемому месту вставки элемента, то перейти к шагу 16.

Шаг 7: Создаем указатель **temp** на объект класса Cell<T> и выделяем память под 1 элемент класса Cell<T>.

Шаг 8: Создаем объект **it** класса Iterator при помощи конструктора по умолчанию и инициализируем переменной past класса Iterator, указывающей на место добавления нового элемента.

Шаг 9: Инициализируем поле data объекта temp данными вставляемого элемента из переменной past\_data.

Шаг 10: Инициализируем указатель на следующий элемент объекта temp адресом переменной it.

Шаг 11: Инициализируем указатель на предыдущий элемент объекта temp адресом указателя на предыдущий элемент объекта it.

Шаг 12: Присваиваем указателю на предыдущий элемент объекта it адрес указателя на предыдущий элемент объекта temp.

Шаг 13: Если указатель на предыдущий элемент объекта temp не равен nullptr, то переход к шагу 17.

Шаг 14: Если адрес it равен адресу предыдущего элемента объекта this, то переход к шагу 18 .

Шаг 15: Если адрес it равен адресу первого элемента объекта this, то переход к шагу 19 .

Шаг 16: Присваиваем flag значение 0 и переходим к шагу 5.

Шаг 17: Присваиваем temp->previous->next адрес объекта temp.

Шаг 18: Присваиваем указателю на последний элемент объекта адрес указателя на следующий элемент объекта temp.

Шаг 19: Присваиваем указателю на первый элемент объекта адрес this адрес указателя на предыдущий элемент &it.

Шаг 20: Увеличиваем размер списка на единицу.

Шаг 20: Выход из функции.

- **void push\_back(T to\_add)**

Является функцией шаблонного класса **List<T>**, предназначена для добавление элемента в конец списка.

Шаг 1: Начало.

Шаг 2: Создаем указатель **temp** на объект класса Cell<T> и выделяем память под 1 элемент класса Cell<T>..

Шаг 3: Инициализируем поле data объекта переменной to\_add.

Шаг 4: Инициализируем указатель на следующий элемент объекта temp nullptr.

Шаг 5:Если размер списка равен 0, то переход к шагу 8.

Шаг 6: Присваиваем указателю на следующий элемент последнего элемента списка this адрес объекта temp. Присваиваем указателю на последний элемент this адрес объекта temp.

Шаг 7: Увеличиваем размер списка на единицу.

Шаг 8: Присваиваем указателям на первый и последний элементы списка this адрес объекта temp.

Шаг 9: Выход из функции.

ПРИЛОЖЕНИЕ Г

*(обязательное)*

Листинг кода

// ball.h

#ifndef BALL\_H

#define BALL\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

// Класс описывающий поведение точки

class Ball : public QGraphicsItem

{

public:

Ball();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **drawballs**(QPainter \*painter);

void **setpoints**(QVector<QPoint> points);

void **removepoint**(QPoint p);

void **generatePointPixmap**();

int ballx,bally;

int ballw,ballh;

QPixmap ballpix;

QRectF rec;

QVector<QPoint> points;

};

#endif // BALL\_H

// ghost.h

#ifndef GHOST\_H

#define GHOST\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

// Класс описывающий поведение Искуственного Интелекта(призраки)

class Ghost:public QGraphicsItem

{

public:

Ghost();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **advance**();

void **setgosx**(int);

void **setgosy**(int);

void **setDirection**(int dir);

void **setColor**(QString col);

void **changeGeometry**() { prepareGeometryChange(); }

QPixmap left1,left2;

QPixmap up1,up2;

QPixmap down1,down2;

QPixmap right1,right2;

QPixmap scareb,scareb1,scarew,scarew1;

int gosx,gosy;

int direction;

int animestate;

bool is\_Scared,whiteb;

};

#endif // GHOST\_H

// list.h

#ifndef LIST\_H

#define LIST\_H

#include <iostream>

using namespace std;

template <class T>

struct Cell

{

Cell \*previous;

Cell \*next;

T data;

};

// собсвенный контейнер

// двунаправленный список

template <class T>

class List

{

Cell<T> \*first; // указатель на начало

Cell<T> \*last; // конец

unsigned int size; // размер

public:

class Iterator;

List():first(nullptr), last(nullptr), size(0) {}

List(const List &object):first(nullptr),last(nullptr),size(0)

{

List<T>::Iterator current = object.begin(), ite = object.end();

while(current != ite)

this->push\_back(\*current++);

if (!object.empty()) this->push\_back(\*current++);

}

List **operator**=(const List &);

void **push\_back**(T);

void **push\_front**(T);

T **pop\_back**();

T **pop\_front**();

T **pop\_midle**(Iterator &);

void **insert**(const Iterator &, const T &);

void **sort**();

Iterator **search**(const T &);

void **clear**();

bool **empty**() const;

void **show**() const;

Iterator **begin**() const;

Iterator **end**() const;

~List()

{

this->clear();

}

};

// class Iterator

template <class T>

class List<T>::Iterator

{

Cell<T> \*current;

public:

**Iterator**(): current(nullptr) {}

**Iterator**(const Iterator &object)

{

this->current = object.current;

}

**Iterator**(Cell<T> \*temp)

{

this->current = temp;

}

Iterator **operator**=(const Iterator &object);

Iterator **operator**++(int);

Iterator **operator**++();

Iterator **operator**--(int);

Iterator **operator**--();

Cell<T> \***operator**&();

T &**operator**\*();

T \***operator**->();

bool **operator**==(const Iterator &);

bool **operator**!=(const Iterator &);

};

#endif // LIST\_H

// options.h

#ifndef OPTIONS\_H

#define OPTIONS\_H

#include <QWidget>

#include <QMessageBox>

namespace Ui {

class Options;

}

// Класс описывающий меню опций

class Options : public QWidget

{

Q\_OBJECT

public:

explicit Options(QWidget \*parent = 0);

~***Options***();

QString mes;

int difficulty;

private slots:

void **on\_startButton\_clicked**();

void **on\_easyButton\_toggled**(bool checked);

void **on\_mediumButton\_toggled**(bool checked);

void **on\_hardButton\_toggled**(bool checked);

private:

Ui::Options \*ui;

};

#endif // OPTIONS\_H

// pacman.h

#ifndef PACMAN\_H

#define PACMAN\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

// класс описывающий поведение пакмана

class Pacman : public QGraphicsItem

{

public:

Pacman();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget); // метод отрисовки

void **advance**();

void **setpacx**(int);

void **setpacy**(int);

void **setDirection**(int dir);

void **setDeath**(int);

bool deadBool;

void **changeGeometry**() { prepareGeometryChange(); }

void **deathseq**();

QPixmap left1,left2,left3,left4;

QPixmap dead1,dead2,dead3,dead4,dead5,dead6,dead7,dead8,dead9;

QPixmap up1,up2,up3,up4;

QPixmap down1,down2,down3,down4;

QPixmap right1,right2,right3,right4;

int pacx,pacy,x,y,height,width;

int direction;

int animestate;

int death;

};

#endif // PACMAN\_H

// pacmanboard.h

#ifndef PACMANBOARD\_H

#define PACMANBOARD\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

#include <QVector>

#include "ball.h"

#include "powerball.h"

// класс описывающий поведение игровой доски

class PacmanBoard : public QGraphicsItem

{

public:

PacmanBoard();

Ball \*ball;

PowerBall \*powerball;

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **advance**();

void **generateBalls**();

void **AddPathPoints**(int, int, int, int);

bool **canmove**(QPoint);

void **fillpacpoints**(int pacx,int pacy);

void **setballpoints**(QVector<QPoint> points);

void **setpowerballpoints**(QVector<QPoint> points);

void **changeGeometry**() { prepareGeometryChange(); }

QVector<QPoint> **getballpoints**();

QVector<QPoint> **getpowerballpoints**();

QPixmap mappic;

QVector<QPoint> pacpoints,ballpoints;

QVector<QPoint> powerballpoints;

};

#endif // PACMANBOARD\_H

// pacmanwindow.h

#ifndef PACMANWINDOW\_H

#define PACMANWINDOW\_H

#include <QDialog>

#include <QtCore>

#include <QtGui>

#include <QSound>

#include <QSoundEffect>

#include "pacman.h"

#include "pacmanboard.h"

#include "ball.h"

#include "ghost.h"

#include "powerball.h"

#include "textdrawing.h"

#include <QShortcut>

#include <fstream>

#include "list.h"

namespace Ui {

class Pacmanwindow;

}

// класс, описывающий поведение игрового процесса

class Pacmanwindow : public QDialog

{

Q\_OBJECT

public:

explicit Pacmanwindow(QString, QWidget \*parent = 0);

void **pacman\_move**();

void **ghostsmove**();

void **ghostsmove1**();

void **ghostsmove2**();

void **ghostsmove3**();

void **moveghostsinrect3**();

void **moveghostsinrect2**();

void **moveghostsinrect1**();

void **moveghostsinrect**();

void **checklost**();

void **delay**();

void **start\_Game**();

void **end\_Game**();

void **pause**();

void **retry**();

void **easy**();

void **medium**();

void **hard**();

void **parseMessage**(QString dataIn, int);

Pacman \*pacman;

Ghost \*ghost;

Ghost \*ghost1;

Ghost \*ghost2;

Ghost \*ghost3;

PacmanBoard \*pac\_map;

Ball \*ball;

PowerBall \*powerball;

TextDrawing \*text;

QString name, difficultyText; // текст, используемый для названия и сложности на экране

QString user;

// переменные отвечающие за расположение всех элементов игры

int pacx,pacy,direction,nextdirection;

int gosx,gosy,ghostdir,nextghostdir;

int gosx1,gosy1,ghostdir1,nextghostdir1;

int gosx2,gosy2,ghostdir2,nextghostdir2;

int gosx3,gosy3,ghostdir3,nextghostdir3;

// в этих переменыых лежат состояния игры

bool moving,ghostmoving,ghostmoving1,ghostmoving2,ghostmoving3;

bool scared,scared1,scared2,scared3;

int score,hiscore,state3,state2,state1,state,lives,level,difficulty;

int starting;

bool start;

bool ghoststart,ghoststart1,ghoststart2,ghoststart3;

bool playing;

bool isPaused;

QVector<QPoint> ballpoints; // вектор содержащий все шары, которые нужно собрать

QVector<QPoint> Powerballpoints; // вектор, содержащий, все усиляющие шары

QSound \*waka;

~***Pacmanwindow***();

public slots:

void **updater**();

void **ghostupdater**();

protected:

void ***keyPressEvent***(QKeyEvent \*event);

private slots:

int **timeoutTime**() { return 1000 / (2); }

void **on\_pauseButton\_clicked**();

void **on\_controlsButton\_clicked**();

private:

Ui::Pacmanwindow \*ui;

QGraphicsScene \*scene;

QTimer \*timer;

QTimer \*ghoststimer;

};

#endif // PACMANWINDOW\_H

// powerball.h

#ifndef POWERBALL\_H

#define POWERBALL\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

// Класс описывающий поведение усиливающего шара

class PowerBall : public QGraphicsItem

{

public:

PowerBall();

QRectF ***boundingRect***() const;

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

void **setpoints**(QVector<QPoint> points);

void **drawballs**(QPainter \*painter);

void **removepoint**(QPoint p);

void **generatePointPixmap**();

void **changeGeometry**() { prepareGeometryChange(); }

int ballx,bally;

int ballw,ballh;

QVector<QPoint> points;

QPixmap pBallPix;

};

#endif // POWERBALL\_H

//textdrawing.h

#ifndef TEXTDRAWING\_H

#define TEXTDRAWING\_H

#include <QPainter>

#include <QGraphicsItem>

#include <QGraphicsScene>

// класс, отвечащий за отрисовку статистики игры

class TextDrawing : public QGraphicsItem

{

public:

TextDrawing();

QRectF ***boundingRect***() const;

void **SetOver**(bool over);

void ***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget);

bool over,playing,paused;

int x,y,w,h,score,timeElapsed;

QString name, difficulty;

private:

QTimer \*timer;

};

#endif // TEXTDRAWING\_H

// ball.cpp

#include "ball.h"

Ball::**Ball**()

{

// установка геометрии шара

ballx=0;

bally=0;

ballw=5;

ballh=5;

generatePointPixmap();

}

QRectF Ball::***boundingRect***() const

{

return QRect(0, 0, 450, 550);

}

// метод создание модельки шара

void Ball::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

for(int i=0;i<points.size();i++){

painter->drawPixmap( points[i].x(), points[i].y(), ballpix );

}

}

// метод отрисовки шара

void Ball::**drawballs**(QPainter \*painter)

{

for(int i=0;i<points.size();i++){

painter->drawPixmap( points[i].x(), points[i].y(), ballpix );

}

}

void Ball::**setpoints**(QVector<QPoint> points)

{

this->points.clear();

this->points=points;

}

void Ball::**removepoint**(QPoint p)

{

}

// метод генерации расположения шаров

void Ball::**generatePointPixmap**()

{

QRect bounds = QRect(0, 0, 3, 3);

QPainter painter;

ballpix = QPixmap(bounds.size());

ballpix.fill(Qt::transparent);

painter.begin(&ballpix);

painter.setRenderHint(QPainter::Antialiasing);

painter.setPen(Qt::NoPen);

painter.setBrush(Qt::white);

painter.drawEllipse(0, 0, 3, 3);

}

// ghost.cpp

#include "ghost.h"

Ghost::**Ghost**()

{

animestate=0;

gosx=450/2;

gosy=480/2;

direction=1;

is\_Scared=false;

whiteb=false;

// Загрузка изображения для моделек.

right1.load(":/images/ghostright1.png");

right2.load(":/images/ghostright2.png");

up1.load(":/images/ghostup1.png");

up2.load(":/images/ghostup2.png");

down1.load(":/images/ghostdown1.png");

down2.load(":/images/ghostdown2.png");

left1.load(":/images/ghostleft1.png");

left2.load(":/images/ghostleft2.png");

scareb.load(":/images/ghostscaredblue1.png");

scareb1.load(":/images/ghostscaredblue2.png");

scarew.load(":/images/ghostscaredwhite1.png");

scarew1.load(":/images/ghostscaredwhite2.png");

}

QRectF Ghost::***boundingRect***() const

{

return QRect(gosx-15,gosy-15,20,20);

}

// метод отрисовки призрака

void Ghost::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

painter->setOpacity(0.5);

if(!is\_Scared){

switch(direction){

case 1:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,left1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,left2);

}

break;

case 4:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,right1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,right2);

}

break;

case 3:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,down1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,down2);

}

break;

case 2:

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,up1);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,up2);

}

break;

}

}else{

if(whiteb){

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,scareb);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,scarew1);

}

}else{

if(animestate==0){

painter->drawPixmap(gosx-15,gosy-15,30,30,scareb);

}else{

painter->drawPixmap(gosx-15,gosy-15,30,30,scareb1);

}

}

}

}

void Ghost::**advance**()

{

if(animestate>2){

animestate=0;

}else{

animestate++;

}

}

// метод установки координаты x

void Ghost::**setgosx**(int x)

{

gosx=x;

}

// метод установки координаты y

void Ghost::**setgosy**(int y)

{

gosy=y;

}

// метод установки направления

void Ghost::**setDirection**(int dir)

{

direction=dir;

}

// метод установки цвета

void Ghost::**setColor**(QString col)

{

if(col=="blue"){

right1.load(":/images/ghostrightblue1.png");

right2.load(":/images/ghostrightblue2.png");

up1.load(":/images/ghostupblue1.png");

up2.load(":/images/ghostupblue2.png");

down1.load(":/images/ghostdownblue1.png");

down2.load(":/images/ghostdownblue2.png");

left1.load(":/images/ghostleftblue1.png");

left2.load(":/images/ghostleftblue2.png");

}else if(col=="orange"){

right1.load(":/images/ghostrightorange1.png");

right2.load(":/images/ghostrightorange2.png");

up1.load(":/images/ghostuporange1.png");

up2.load(":/images/ghostuporange2.png");

down1.load(":/images/ghostdownorange1.png");

down2.load(":/images/ghostdownorange2.png");

left1.load(":/images/ghostleftorange1.png");

left2.load(":/images/ghostleftorange2.png");

}else if(col=="red"){

right1.load(":/images/ghostrightred1.png");

right2.load(":/images/ghostrightred2.png");

up1.load(":/images/ghostupred1.png");

up2.load(":/images/ghostupred2.png");

down1.load(":/images/ghostdownred1.png");

down2.load(":/images/ghostdownred2.png");

left1.load(":/images/ghostleftred1.png");

left2.load(":/images/ghostleftred2.png");

}

}

// list.cpp

#include "list.h"

template <class T>

bool List<T>::**empty**() const

{

return !static\_cast<bool>(this->size);

}

template <class T>

void List<T>::**sort**()

{

Cell<T> \*current, \*preend = this->last;

while(preend != this->first)

{

current = this->first;

while(current != preend)

{

if(current->data > current->next->data)

{

T temp = current->data;

current->data = current->next->data;

current->next->data = temp;

}

current = current->next;

}

preend= preend->previous;

}

}

template <class T>

void List<T>::**insert**(const Iterator &past, const T &past\_data)

{

bool flag = 1;

List<T>::Iterator temp = this->begin();

while(temp != this->end())

{

if(temp == past)

{

flag = 0;

break;

}

temp++;

}

if(temp == past) flag = 0;

else

{

Cell<T> \*temp = new Cell<T>;

List<T>::Iterator it = past;

temp->data = past\_data;

temp->next = &it;

temp->previous = (&it)->previous;

(&it)->previous = temp;

if(temp->previous != nullptr) temp->previous->next = temp;

if(&it == this->last) this->last = temp->next;

if(&it == this->first) this->first = (&it)->previous;

++this->size;

}

}

template<class T>

void List<T>::**clear**()

{

Cell<T> \*current = this->first;

while(this->first)

{

current = this->first;

this->first = this->first->next;

delete current;

}

this->last = this->first;

this->size = 0;

}

template <class T>

List<T> List<T>::**operator**=(const List<T> &object)

{

if(this != &object)

{

this->clear();

List<T>::Iterator itb = object.begin(), ite = object.end();

while(itb != ite)

this->push\_back(\*itb++);

if( !object.empty()) this->push\_back(\*itb);

}

return \*this;

}

template <class T>

typename List<T>::Iterator List<T>::**begin**() const

{

return Iterator(first);

}

template <class T>

typename List<T>::Iterator List<T>::**end**() const

{

return Iterator(last);

}

template <class T>

typename List<T>::Iterator List<T>::**search**(const T &data)

{

Iterator i = this->begin();

bool flag = false;

while(i != this->end())

if(data == \*i++)

{

flag = true;

break;

}

if(!flag)

if(\*i == data) flag = true;

return flag?i:Iterator();

}

template <class T>

void List<T>::**push\_back**(T to\_add)

{

Cell<T> \*temp = new Cell<T>;

temp->data = to\_add;

temp->next = nullptr;

temp->previous = this->last;

if(!size)

this->first = this->last = temp;

else

{

this->last->next = temp;

this->last = temp;

}

++size;

}

template <class T>

void List<T>::**show**() const

{

Cell<T> \*temp = this->first;

while(temp)

{

cout<< temp->data << endl;

temp = temp->next;

}

cout << endl;

}

template <class T>

void List<T>::**push\_front**(T to\_add)

{

Cell<T> \*temp = new Cell<T>;

temp->data = to\_add;

temp->previous = nullptr;

temp->next = this->first;

if(!size)

this->first = this->last = temp;

else

{

this->first->previous = temp;

this->first = temp;

}

++size;

}

template <class T>

T List<T>::**pop\_midle**(Iterator &current)

{

T data;

if(current == this->begin()) data = this->pop\_front();

else if(current == this->end()) data = this->pop\_back();

else

{

(&current)->previous->next = (&current)->next;

(&current)->next->previous = (&current)->previous;

data = \*current;

delete &current;

}

return data;

}

template <class T>

T List<T>::**pop\_back**()

{

T data;

if(size)

{

Cell<T> \*temp = this->last;

data = temp->data;

this->last = this->last->previous;

if(size != 1) this->last->next = nullptr;

delete temp;

--size;

if(!size) this->first= this->last;

}

return data;

}

template <class T>

T List<T>::**pop\_front**()

{

T data;

if(size)

{

Cell<T> \*temp = this->first;

data = temp->data;

this->first = this->first->next;

if (size != 1) this->first->previous = nullptr;

delete temp;

--size;

if(!size) this->last = this->first;

}

return data;

}

template <class T>

Cell<T> \*List<T>::Iterator::operator&()

{

return this->current;

}

template <class T>

bool List<T>::Iterator::operator==(const List<T>::Iterator &object)

{

return this->current == object.current;

}

template <class T>

bool List<T>::Iterator::operator!=(const List<T>::Iterator &object)

{

return this->current != object.current;

}

template <class T>

typename List<T>::Iterator List<T>::Iterator::operator=(const Iterator &object)

{

if(this != &object)

this->current = object.current;

return \*this;

}

template <class T>

typename List<T>::Iterator List<T>::Iterator::operator++(int)

{

List<T>::Iterator temp = \*this;

this->current = current->next;

return temp;

}

template <class T>

typename List<T>::Iterator List<T>::Iterator::operator++()

{

current = current->next;

return \*this;

}

template <class T>

typename List<T>::Iterator List<T>::Iterator::operator--(int)

{

List<T>::Iterator temp = \*this;

this->current = this->current->previous;

return temp;

}

template <class T>

typename List<T>::Iterator List<T>::Iterator::operator--()

{

current = current->previous;

return \*this;

}

template <class T>

T &List<T>::Iterator::operator\*()

{

return current->data;

}

template <class T>

T \*List<T>::Iterator::operator->()

{

return &current->data;

}

// main.cpp

#include <QApplication>

#include "options.h"

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Options o;

o.show();

return a.exec();

}

// options.cpp

#include "options.h"

#include "ui\_options.h"

#include "pacmanwindow.h"

#include <exception>

#include <QDebug>

Options::Options(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Options)

{

QLabel \*image = new QLabel();

image->setPixmap( QPixmap( "paclogo.jpg" ) );

image->show();

ui->setupUi(this);

difficulty = 0;

}

Options::~Options()

{

delete ui;

}

// обработка нажатия клавиши старт

void Options::on\_startButton\_clicked()

{

QString nickname = ui->nameText->toPlainText();

bool good\_start = true;

try

{

if(nickname.length() < 1 || nickname.length() > 6) throw QString("Ник не должен быть пустым, а также его длинна не должна превышать 6 символов");

if(difficulty == 0) throw QString("Вы не выбрали уровень сложности");

}

catch(QString &message)

{

QMessageBox::warning(this,"Warning", message);

good\_start = false;

}

if(good\_start)

{

Pacmanwindow \*w = new Pacmanwindow(ui->nameText->toPlainText());

QString nametemp = ui->nameText->toPlainText();

w->parseMessage(nametemp, difficulty);

w->show();

this->close();

}

}

void Options::on\_easyButton\_toggled(bool )

{

difficulty = 1;

}

void Options::on\_mediumButton\_toggled(bool )

{

difficulty = 2;

}

void Options::on\_hardButton\_toggled(bool )

{

difficulty = 3;

}

//pacman.cpp

#include "pacman.h"

Pacman::**Pacman**()

{

animestate=0;

pacx=400/2;

pacy=360;

x=10;

y=10;

height = 23;

width = 23;

// Загрузка изображений для отображения пакмана

direction=4;

right1.load(":/images/pacround.png");

right2.load(":/images/pacright1.png");

right3.load(":/images/pacright2.png");

up1.load(":/images/pacround.png");

up2.load(":/images/pacup1.png");

up3.load(":/images/pacup2.png");

down1.load(":/images/pacround.png");

down2.load(":/images/pacdown1.png");

down3.load(":/images/pacdown2.png");

left1.load(":/images/pacround.png");

left2.load(":/images/pacleft1.png");

left3.load(":/images/pacleft2.png");

dead1.load(":/images/die1.png");

dead2.load(":/images/die2.png");

dead3.load(":/images/die3.png");

dead4.load(":/images/die4.png");

dead5.load(":/images/die5.png");

dead6.load(":/images/die6.png");

dead7.load(":/images/die7.png");

dead8.load(":/images/die8.png");

dead8.load(":/images/die9.png");

}

QRectF Pacman::***boundingRect***() const

{

return QRect(pacx-x, pacy-y, height, width);

}

void Pacman::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

// установка изображений в зависимости от направления

switch(direction){

case 1:

if(animestate<2){

painter->drawPixmap(pacx-x,pacy-y,height,width,left1);

}else if(animestate<4){

painter->drawPixmap(pacx-x,pacy-y,height,width,left2);

}else if(animestate<6){

painter->drawPixmap(pacx-x,pacy-y,height,width,left3);

}

break;

case 4:

if(animestate<2){

painter->drawPixmap(pacx-x,pacy-y,height,width,right1);

}else if(animestate<4){

painter->drawPixmap(pacx-x,pacy-y,height,width,right2);

}else if(animestate<6){

painter->drawPixmap(pacx-x,pacy-y,height,width,right3);

}

break;

case 3:

if(animestate<2){

painter->drawPixmap(pacx-x,pacy-y,height,width,down1);

}else if(animestate<4){

painter->drawPixmap(pacx-x,pacy-y,height,width,down2);

}

else if(animestate<6){

painter->drawPixmap(pacx-x,pacy-y,height,width,down3);

}

break;

case 2:

if(animestate<2){

painter->drawPixmap(pacx-x,pacy-y,height,width,up1);

}else if(animestate<4){

painter->drawPixmap(pacx-x,pacy-y,height,width,up2);

}else if(animestate<6){

painter->drawPixmap(pacx-x,pacy-y,height,width,up3);

}

break;

}

}

void Pacman::**advance**()

{

if(animestate>3 ){

animestate=0;

}else{

animestate++;

}

}

void Pacman::**deathseq**()

{

}

// установка коодрдинаты x

void Pacman::**setpacx**(int x)

{

pacx=x;

}

// установка координаты y

void Pacman::**setpacy**(int y)

{

pacy=y;

}

// Установка направления

void Pacman::**setDirection**(int dir)

{

direction=dir;

}

// Установка колчиства смертрей

void Pacman::**setDeath**(int d)

{

death=d;

}

// pacmanboard.cpp

#include "pacmanboard.h"

PacmanBoard::**PacmanBoard**()

{

// Загрузка изображения карты

mappic.load(":/images/pac\_map\_old.png");

ball=new Ball;

powerball=new PowerBall;

// Добавления шаров на карту

AddPathPoints(30, 30, 200, 30);

AddPathPoints(250, 30, 420, 30);

AddPathPoints(30, 90, 420, 90);

AddPathPoints(30, 130, 110, 130);

AddPathPoints(150, 130, 200, 130);

AddPathPoints(250, 130, 300, 130);

AddPathPoints(340, 130, 420, 130);

AddPathPoints(150, 180, 300, 180);

AddPathPoints(0, 230, 150, 230);

AddPathPoints(300, 230, 450, 230);

AddPathPoints(150, 270, 300, 270);

AddPathPoints(30, 320, 200, 320);

AddPathPoints(250, 320, 420, 320);

AddPathPoints(30, 360, 60, 360);

AddPathPoints(110, 360, 340, 360);

AddPathPoints(390, 360, 420, 360);

AddPathPoints(30, 410, 110, 410);

AddPathPoints(150, 410, 200, 410);

AddPathPoints(250, 410, 300, 410);

AddPathPoints(340, 410, 420, 410);

AddPathPoints(30, 450, 420, 450);

AddPathPoints(30, 30, 30, 130);

AddPathPoints( 30, 320, 30, 360);

AddPathPoints(30, 410, 30, 450);

AddPathPoints( 60, 360, 60, 410);

AddPathPoints(110, 30, 110, 410);

AddPathPoints( 150, 90, 150, 130);

AddPathPoints(150, 180, 150, 320);

AddPathPoints(150, 360, 150, 410);

AddPathPoints( 200, 30, 200, 90);

AddPathPoints(200, 135, 200, 180);

AddPathPoints( 200, 320, 200, 360);

AddPathPoints( 200, 410, 200, 450);

AddPathPoints( 250, 30, 250, 90);

AddPathPoints( 250, 135, 250, 180);

AddPathPoints( 250, 320, 250, 360);

AddPathPoints( 250, 410, 250, 450);

AddPathPoints( 300, 90, 300, 130);

AddPathPoints( 300, 180, 300, 320);

AddPathPoints( 300, 360, 300, 410);

AddPathPoints( 340, 30, 340, 410);

AddPathPoints( 390, 360, 390, 410);

AddPathPoints( 420, 30, 420, 130);

AddPathPoints( 420, 320, 420, 360);

AddPathPoints( 420, 410, 420, 450);

QPoint p;

QPoint p1,p2,p3,p4;

p1.setX(30);

p1.setY(450);

p2.setX(30);

p2.setY(35);

p3.setX(420);

p3.setY(35);

p4.setX(420);

p4.setY(450);

powerballpoints.push\_front(p1);

powerballpoints.push\_front(p2);

powerballpoints.push\_front(p3);

powerballpoints.push\_front(p4);

// Занесения шаров в список

for (int i=0; i<450-1; i++)

{

for(int j=0;j<480-1;j++){

p.setX(i);

p.setY(j);

if(j%10==0 && i%10==0){

if (pacpoints.contains(p)){

if(p!=p1 && p!=p2 && p!=p3 && p!=p4){

if(p.x()>0){

ballpoints.push\_front(p);

}

}

}

}

}

}

generateBalls();

}

QRectF PacmanBoard::***boundingRect***() const

{

return QRect(0,0,450,480); // Установка размера карты

}

void PacmanBoard::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

painter->drawPixmap(0,0,450,480,mappic);

}

void PacmanBoard::**generateBalls**()

{

QPainter painter;

painter.begin(&mappic);

painter.setRenderHint(QPainter::Antialiasing);

painter.setPen(Qt::NoPen);

painter.setBrush(Qt::white);

for(int i=0;i<ballpoints.size();i++){

painter.drawEllipse(ballpoints[i].x(),ballpoints[i].y(),3,3);

}

for(int i=0;i<powerballpoints.size();i++){

painter.setBrush(Qt::yellow);

painter.drawEllipse(powerballpoints[i].x()-5,powerballpoints[i].y()-5,10,10);

}

}

// Алгоритм передвижения пакмана по карте

bool PacmanBoard::**canmove**(QPoint point)

{

for(int i=0;i<pacpoints.size();i++){

if(pacpoints[i]==point)

{

return true;

}

}

return false;

}

// алгоритм прорисовки пути пакмана

void PacmanBoard::**fillpacpoints**(int pacx, int pacy)

{

QPainter painter;

QRect rec(pacx,pacy,3,3);

QRect rec2(pacx-5,pacy-5,10,10);

painter.begin(&mappic);

painter.fillRect(rec,Qt::black);

if((pacx==powerballpoints[0].x() && pacy==powerballpoints[0].y())||

(pacx==powerballpoints[1].x() && pacy==powerballpoints[1].y())||

(pacx==powerballpoints[2].x() && pacy==powerballpoints[2].y())||

(pacx==powerballpoints[3].x() && pacy==powerballpoints[3].y())){

painter.fillRect(rec2,Qt::black);

}

painter.end();

}

void PacmanBoard::**setballpoints**(QVector<QPoint> points)

{

ball->setpoints(points);

}

void PacmanBoard::**setpowerballpoints**(QVector<QPoint> points)

{

powerball->setpoints(points);

}

void PacmanBoard::**AddPathPoints**(int x1, int y1, int x2, int y2)

{

QPoint p;

if (x1 == x2)

{

if (y1 < y2)

{

for (int y=y1; y<y2+1; y++)

{

p.setX(x1);

p.setY(y);

if (! pacpoints.contains(p)){pacpoints.push\_front(p);

}

}

}

else

{

for (int y=y1; y>y2-1; y--)

{

p.setX(x1);

p.setY(y);

if (! pacpoints.contains(p)){pacpoints.push\_front(p);

}

}

}

}

else

{

if (x1 < x2)

{

for (int x=x1; x<x2+1; x++)

{

p.setX(x);

p.setY(y1);//= new QPoint(x1, y);

if (! pacpoints.contains(p)){pacpoints.push\_front(p);

}

}

}

else

{

for (int x=x1; x>x2-1; x--)

{

p.setX(x);

p.setY(y1);//= new QPoint(x1, y);

if (! pacpoints.contains(p)){pacpoints.push\_front(p);

}

}

}

}

}

QVector<QPoint> PacmanBoard::**getballpoints**()

{

return ballpoints;

}

QVector<QPoint> PacmanBoard::**getpowerballpoints**()

{

return powerballpoints;

}

// pacmanwindow.cpp

#include "Pacmanwindow.h"

#include "ui\_Pacmanwindow.h"

#include <QThread>

#include <QtWidgets>

#include "options.h"

#include "list.h"

#include "list.cpp"

// метод, достающий самый выскоий результат из файла

int get\_high\_score()

{

List<int> scores; // используем собственный контейнер

// для хранения очков, считанных из файла

ifstream file;

file.open("users.txt", ios::in); // файл, в котором хранятся результаты

// формат файла <ник> <кол-во набранных очков>

if(file)

{

while(!file.eof())

{

string user;

int score;

if(!file.eof()) file >> user >> score;

scores.push\_front(score); // добавляем каждый результат в конец

}

file.close();

scores.sort(); // сортируем

return \*(scores.end()); // последний самый большой

}

else return 0;

}

// класс, отвечающий за отрисовку окна, в котром проходит игровой процесс

Pacmanwindow::Pacmanwindow(QString nickname, QWidget \*parent) :

QDialog(parent),

ui(new Ui::Pacmanwindow)

{

user = nickname; // имя юзера, нужно для записи статистики в файл

score = 0; // счётчик очков

hiscore = get\_high\_score(); // получаем лучший результат

level = 0; // уровень сложности

lives = 3; // количество жизней

starting = 0;

// координаты призраков

gosx=450/2;

gosy=480/2;

gosx1=450/2;

gosy1=480/2;

gosx2=450/2;

gosy2=480/2;

gosx3=450/2;

gosy3=480/2;

pacx=410/2;

pacy=410;

ghostmoving=false;

ghostmoving1=false;

ghostmoving2=false;

ghostmoving3=false;

scared=false;

scared1=false;

scared2=false;

scared3=false;

start=false;

direction=0;

moving=false;

playing=false;

isPaused = false;

// настройка графической формы

ui->setupUi(this);

scene = new QGraphicsScene(this);

ui->graphicsView->setScene(scene);

ui->graphicsView->setRenderHint(QPainter::Antialiasing);

ui->scoreLabel->setText(name);// adds the name tp score label

scene->setSceneRect(0,0,640,480);

ui->graphicsView->setSceneRect(scene->sceneRect());

// обработка нажатия esc для выхода из игры

connect(ui->exitButton,SIGNAL(clicked()),qApp,SLOT(closeAllWindows()));

ui->exitButton->setShortcut(QKeySequence("Escape"));

text=new TextDrawing;

text->over=false;

text->paused=false;

scene->addItem(text);

text->setZValue(7);

}

void Pacmanwindow::parseMessage(QString name, int temp){

this->name = name;

ui->scoreLabel->setText("Score");

difficulty = temp;

}

// метод, вызов которого начинает игры

void Pacmanwindow::start\_Game()

{

level = 0;

direction = 0;

playing = true;

// установка координат призраков

gosx=450/2;

gosy=480/2;

gosx1=450/2;

gosy1=480/2;

gosx2=450/2;

gosy2=480/2;

gosx3=450/2;

gosy3=480/2;

pacx=400/2;

pacy=360;

pacman=new Pacman;

pac\_map=new PacmanBoard;

ghost=new Ghost;

ghost1=new Ghost;

ghost2=new Ghost;

ghost3=new Ghost;

// Установка цвета призраков

ghost1->setColor("orange");

ghost2->setColor("red");

ghost3->setColor("blue");

ball=new Ball;

powerball=new PowerBall;

ballpoints=pac\_map->getballpoints();

Powerballpoints=pac\_map->getpowerballpoints();

// установка всех игровых элементов на форму

scene->removeItem(text);

scene->addItem(pac\_map);

scene->addItem(pacman);

scene->addItem(ghost);

scene->addItem(ghost1);

scene->addItem(ghost2);

scene->addItem(ghost3);

scene->addItem(text);

text->playing=true;

ghostmoving=false;

ghostmoving1=false;

ghostmoving2=false;

ghostmoving3=false;

moving=false;

ghoststart=false;

ghoststart1=false;

ghoststart2=false;

ghoststart3=false;

state=0;

state1=0;

state2=0;

state3=0;

// проигрывание мелодии при начале

waka->play(":/Sound/pacman\_beginning.wav");

// настройка сложности в зависимости от выбора в начальном меню

if (starting==0)

{

if (difficulty == 1){

easy();

difficultyText = "easy";

}

if (difficulty == 2){

medium();

difficultyText = "medium";

}

if (difficulty == 3){

hard();

difficultyText = "hard";

}

}

}

// настройка игры при выборе сложности easy

void Pacmanwindow::easy(){

// таймеры для плавной анимации

timer = new QTimer(this);

ghoststimer=new QTimer(this);

// каждое срабатывание таймера отвечает за передвижение пакмана и призраков по карте

connect(timer, SIGNAL(timeout()), this,SLOT(updater()));

connect(ghoststimer, SIGNAL(timeout()), this,SLOT(ghostupdater()));

this->show();

scene->update();

this->update();

lives = 3;

timer->start();

timer->start(35);

ghoststimer->start(45);

this->setFocus();

}

void Pacmanwindow::medium(){

lives =2;

timer = new QTimer(this);

ghoststimer=new QTimer(this);

connect(timer, SIGNAL(timeout()), this,SLOT(updater()));

connect(ghoststimer, SIGNAL(timeout()), this,SLOT(ghostupdater()));

this->show();

scene->update();

this->update();

timer->start();

timer->start(50);

ghoststimer->start(50);

this->setFocus();

}

void Pacmanwindow::hard(){

lives = 0;

timer = new QTimer(this);

ghoststimer=new QTimer(this);

connect(timer, SIGNAL(timeout()), this,SLOT(updater()));

connect(ghoststimer, SIGNAL(timeout()), this,SLOT(ghostupdater()));

this->show();

scene->update();

this->update();

timer->start();

timer->start(40);

ghoststimer->start(35);

this->setFocus();

}

// Данный метод вызывается в случае проийгрыша, когда число жизней станет меньше 0

void Pacmanwindow::end\_Game()

{

starting = 0;

// очистка графической сцены

scene->removeItem(pacman);

scene->removeItem(ghost);

scene->removeItem(ghost1);

scene->removeItem(ghost2);

scene->removeItem(ghost3);

scene->removeItem(pac\_map);

text->playing=false;

start = false;

playing=false;

scene->update();

timer->stop();

ghoststimer->stop();

// Запись текущего результата в файл

ofstream file;

file.open("users.txt", std::ofstream::out | std::ofstream::app); // открывается файл для до записи

file << this->user.toStdString() << " " << score << endl;

file.close();

score = 0; // обнуляем счетчик очков

}

// метод запуска повторной попытки

void Pacmanwindow::retry(){

gosx=450/2;

gosy=480/2;

gosx1=450/2;

gosy1=480/2;

gosx2=450/2;

gosy2=480/2;

gosx3=450/2;

gosy3=480/2;

pacx=400/2;

pacy=360;

direction=4;

start = true;

ghoststart=false;

ghoststart1=false;

ghoststart2=false;

ghoststart3=false;

ghostmoving=false;

ghostmoving1=false;

ghostmoving2=false;

ghostmoving3=false;

moving=false;

state=0;

state1=0;

state2=0;

state3=0;

}

// метод отвечающий за передвижения пакмана по карте

void Pacmanwindow::pacman\_move()

{

QPoint p;

if(nextdirection!=direction){

switch(nextdirection)

{

case 1:

p.setX(pacx-5);

p.setY(pacy);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

case 4:

p.setX(pacx+5);

p.setY(pacy);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

case 3:

p.setX(pacx);

p.setY(pacy+5);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

case 2:

p.setX(pacx);

p.setY(pacy-5);

if(pac\_map->canmove(p)){

direction=nextdirection;

nextdirection=0;

}

break;

}

}

switch(direction)

{

case 1:

p.setX(pacx-5);

p.setY(pacy);

pacman->setDirection(direction);

if(pac\_map->canmove(p)){

pacx-=5;

moving=true;

}else{

moving=false;

}

break;

case 4:

pacman->setDirection(direction);

p.setX(pacx+5);

p.setY(pacy);

if(pac\_map->canmove(p)){

pacx+=5;

moving=true;

}else{

moving=false;

}

break;

case 3:

pacman->setDirection(direction);

p.setX(pacx);

p.setY(pacy+5);

if(pac\_map->canmove(p)){

pacy+=5;

moving=true;

}else{

moving=false;

}

break;

case 2:

pacman->setDirection(direction);

p.setX(pacx);

p.setY(pacy-5);

if(pac\_map->canmove(p)){

pacy-=5;

moving=true;

}else{

moving=false;

}

break;

}

// Телепортирование пакмана через боковые границы

if(pacx<=0){

pacx=450;

pacy=230;

}else if(pacx>=450){

pacx=0;

pacy=230;

}

pacman->setpacx(pacx);

pacman->setpacy(pacy);

}

// передвижение призраков

void Pacmanwindow::ghostsmove()

{

QPoint p;

if(!ghostmoving){

ghostdir=(qrand()%4)+1;

}else{

if(ghostdir==4 && gosy<pacy || ghostdir==1 && gosy>pacy){

if(ghostdir==1 && gosy>pacy){

nextghostdir=2;

}else if(ghostdir==4 &&gosy<pacy){

nextghostdir=3;

}

}else if(ghostdir==3 && gosx<pacx || ghostdir==2 && gosx>pacx){

if(ghostdir==2 && gosx>pacx){

nextghostdir=1;

}else if(ghostdir==3 && gosx<pacx){

nextghostdir=4;

}

}

}

if(nextghostdir!=ghostdir){

switch(nextghostdir)

{

case 1:

p.setX(gosx-5);

p.setY(gosy);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

case 4:

p.setX(gosx+5);

p.setY(gosy);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

case 3:

p.setX(gosx);

p.setY(gosy+5);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

case 2:

p.setX(gosx);

p.setY(gosy-5);

if(pac\_map->canmove(p)){

ghostdir=nextghostdir;

nextghostdir=0;

}

break;

}

}

switch(ghostdir)

{

case 1:

p.setX(gosx-5);

p.setY(gosy);

ghost->setDirection(ghostdir);

if(pac\_map->canmove(p)){

gosx-=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

case 4:

ghost->setDirection(ghostdir);

p.setX(gosx+5);

p.setY(gosy);

if(pac\_map->canmove(p)){

gosx+=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

case 3:

ghost->setDirection(ghostdir);

p.setX(gosx);

p.setY(gosy+5);

if(pac\_map->canmove(p)){

gosy+=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

case 2:

ghost->setDirection(ghostdir);

p.setX(gosx);

p.setY(gosy-5);

if(pac\_map->canmove(p)){

gosy-=5;

ghostmoving=true;

}else{

ghostmoving=false;

}

break;

}

if(gosx<=0){

gosx=450;

gosy=230;

}else if(gosx>=450){

gosx=0;

gosy=230;

}

ghost->setgosx(gosx);

ghost->setgosy(gosy);

}

void Pacmanwindow::ghostsmove1()

{

QPoint p;

if(!ghostmoving1){

ghostdir1=(qrand()%4)+1;

}else{

if(ghostdir1==4 && gosy1<pacy || ghostdir1==1 && gosy1>pacy){

if(ghostdir1==1 && gosy>pacy){

nextghostdir1=2;

}else if(ghostdir1==4 &&gosy1<pacy){

nextghostdir1=3;

}

}else if(ghostdir1==3 && gosx1<pacx || ghostdir1==2 && gosx1>pacx){

if(ghostdir1==2 && gosx1>pacx){

nextghostdir1=1;

}else if(ghostdir1==3 && gosx1<pacx){

nextghostdir1=4;

}

}

}

if(nextghostdir1!=ghostdir1){

switch(nextghostdir1)

{

case 1:

p.setX(gosx1-5);

p.setY(gosy1);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

case 4:

p.setX(gosx1+5);

p.setY(gosy1);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

case 3:

p.setX(gosx1);

p.setY(gosy1+5);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

case 2:

p.setX(gosx1);

p.setY(gosy1-5);

if(pac\_map->canmove(p)){

ghostdir1=nextghostdir1;

nextghostdir1=0;

}

break;

}

}

switch(ghostdir1)

{

case 1:

p.setX(gosx1-5);

p.setY(gosy1);

ghost1->setDirection(ghostdir1);

if(pac\_map->canmove(p)){

gosx1-=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

case 4:

ghost1->setDirection(ghostdir1);

p.setX(gosx1+5);

p.setY(gosy1);

if(pac\_map->canmove(p)){

gosx1+=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

case 3:

ghost1->setDirection(ghostdir1);

p.setX(gosx1);

p.setY(gosy1+5);

if(pac\_map->canmove(p)){

gosy1+=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

case 2:

ghost1->setDirection(ghostdir1);

p.setX(gosx1);

p.setY(gosy1-5);

if(pac\_map->canmove(p)){

gosy1-=5;

ghostmoving1=true;

}else{

ghostmoving1=false;

}

break;

}

if(gosx1<=0){

gosx1=450;

gosy1=230;

}else if(gosx1>=450){

gosx1=0;

gosy1=230;

}

ghost1->setgosx(gosx1);

ghost1->setgosy(gosy1);

}

void Pacmanwindow::ghostsmove2()

{

QPoint p;

if(!ghostmoving2){

ghostdir2=(qrand()%4)+1;

}else{

if(ghostdir2==4 &&gosy2<pacy || ghostdir2==1 && gosy2>pacy){

if(ghostdir2==1 && gosy2>pacy){

nextghostdir2=2;

}else if(ghostdir2==4 &&gosy2<pacy){

nextghostdir2=3;

}

}else if(ghostdir2==3 && gosx2<pacx || ghostdir2==2 && gosx2>pacx){

if(ghostdir2==2 && gosx2>pacx){

nextghostdir2=1;

}else if(ghostdir2==3 && gosx2<pacx){

nextghostdir2=4;

}

}

}

if(nextghostdir2!=ghostdir2){

switch(nextghostdir2)

{

case 1:

p.setX(gosx2-5);

p.setY(gosy2);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

case 4:

p.setX(gosx2+5);

p.setY(gosy2);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

case 3:

p.setX(gosx2);

p.setY(gosy2+5);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

case 2:

p.setX(gosx2);

p.setY(gosy2-5);

if(pac\_map->canmove(p)){

ghostdir2=nextghostdir2;

nextghostdir2=0;

}

break;

}

}

switch(ghostdir2)

{

case 1:

p.setX(gosx2-5);

p.setY(gosy2);

ghost2->setDirection(ghostdir2);

if(pac\_map->canmove(p)){

gosx2-=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

case 4:

ghost2->setDirection(ghostdir2);

p.setX(gosx2+5);

p.setY(gosy2);

if(pac\_map->canmove(p)){

gosx2+=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

case 3:

ghost2->setDirection(ghostdir2);

p.setX(gosx2);

p.setY(gosy2+5);

if(pac\_map->canmove(p)){

gosy2+=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

case 2:

ghost2->setDirection(ghostdir2);

p.setX(gosx2);

p.setY(gosy2-5);

if(pac\_map->canmove(p)){

gosy2-=5;

ghostmoving2=true;

}else{

ghostmoving2=false;

}

break;

}

if(gosx2<=0){

gosx2=450;

gosy2=230;

}else if(gosx2>=450){

gosx2=0;

gosy2=230;

}

ghost2->setgosx(gosx2);

ghost2->setgosy(gosy2);

}

void Pacmanwindow::ghostsmove3()

{

QPoint p,g;

if(!ghostmoving3){

ghostdir3=(qrand()%4)+1;

}else{

if(ghostdir3==4 &&gosy3<pacy || ghostdir3==1 && gosy3>pacy){

if(ghostdir3==1 && gosy3>pacy){

nextghostdir3=2;

}else if(ghostdir3==4 &&gosy3<pacy){

nextghostdir3=3;

}

}else if(ghostdir3==3 && gosx3<pacx || ghostdir3==2 && gosx3>pacx){

if(ghostdir3==2 && gosx3>pacx){

nextghostdir3=1;

}else if(ghostdir3==3 && gosx3<pacx){

nextghostdir3=4;

}

}

}

if(nextghostdir3!=ghostdir3){

switch(nextghostdir3)

{

case 1:

p.setX(gosx3-5);

p.setY(gosy3);

if(pac\_map->canmove(p)){

ghostdir3=nextghostdir3;

nextghostdir3=0;

}

break;

case 4:

p.setX(gosx3+5);

p.setY(gosy3);

if(pac\_map->canmove(p)){

ghostdir3=nextghostdir3;

nextghostdir3=0;

}

break;

case 3:

p.setX(gosx3);

p.setY(gosy3+5);

if(pac\_map->canmove(p)){

ghostdir3=nextghostdir3;

nextghostdir3=0;

}

break;

case 2:

p.setX(gosx3);

p.setY(gosy3-5);

if(pac\_map->canmove(p)){

ghostdir3=nextghostdir3;

nextghostdir3=0;

}

break;

}

}

switch(ghostdir3)

{

case 1:

p.setX(gosx3-5);

p.setY(gosy3);

ghost3->setDirection(ghostdir3);

if(pac\_map->canmove(p)){

gosx3-=5;

ghostmoving3=true;

}else{

ghostmoving3=false;

}

break;

case 4:

ghost3->setDirection(ghostdir3);

p.setX(gosx3+5);

p.setY(gosy3);

if(pac\_map->canmove(p)){

gosx3+=5;

ghostmoving3=true;

}else{

ghostmoving3=false;

}

break;

case 3:

ghost3->setDirection(ghostdir3);

p.setX(gosx3);

p.setY(gosy3+5);

if(pac\_map->canmove(p)){

gosy3+=5;

ghostmoving3=true;

}else{

ghostmoving3=false;

}

break;

case 2:

ghost3->setDirection(ghostdir3);

p.setX(gosx3);

p.setY(gosy3-5);

if(pac\_map->canmove(p)){

gosy3-=5;

ghostmoving3=true;

}else{

ghostmoving3=false;

}

break;

}

if(gosx3<=0){

gosx3=450;

gosy3=230;

}else if(gosx3>=450){

gosx3=0;

gosy3=230;

}

ghost3->setgosx(gosx3);

ghost3->setgosy(gosy3);

}

///// Keep ghosts in rectangles

void Pacmanwindow::moveghostsinrect3()

{

if(gosx3==450/2+40 || gosx3==450/2-40){

if(ghostdir3==4){

ghostdir3=1;

}else{

ghostdir3=4;

}

}

if(ghostdir3==4){

gosx3-=2;

}else{

gosx3+=2;

}

ghost3->setgosx(gosx3);

ghost3->setgosy(gosy3);

}

void Pacmanwindow::moveghostsinrect2()

{

if(gosx2==450/2+40 || gosx2==450/2-40){

if(ghostdir2==4){

ghostdir2=1;

}else{

ghostdir2=4;

}

}

if(ghostdir2==4){

gosx2+=5;

}else{

gosx2-=5;

}

ghost2->setgosx(gosx2);

ghost2->setgosy(gosy2);

}

void Pacmanwindow::moveghostsinrect1()

{

if(gosx1==450/2+40 || gosx1==450/2-40){

if(ghostdir1==4){

ghostdir1=1;

}else{

ghostdir1=4;

}

}

if(ghostdir1==4){

gosx1+=5;

}else{

gosx1-=5;

}

ghost1->setgosx(gosx1);

ghost1->setgosy(gosy1);

}

void Pacmanwindow::moveghostsinrect()

{

if(gosx==450/2+40 || gosx==450/2-40){

if(ghostdir==4){

ghostdir=1;

}else{

ghostdir=4;

}

}

if(ghostdir==4){

gosx+=5;

}else{

gosx-=5;

}

ghost->setgosx(gosx);

ghost->setgosy(gosy);

}

void Pacmanwindow::checklost()

{

/// When all the points have been eaten start new game

if(ballpoints.isEmpty() ){

level += 1;

starting += 1;

start\_Game();

int temp = hiscore;

hiscore = temp;

}

if(pacman->collidesWithItem(ghost) ||

pacman->collidesWithItem(ghost1) ||

pacman->collidesWithItem(ghost2) ||

pacman->collidesWithItem(ghost3)){

if(scared){

// за каждого съеденого призрака по сотке

score+=100;

if(pacman->collidesWithItem(ghost)){

gosx=450/2;

gosy=450/2;

ghoststart=false;

}

if(pacman->collidesWithItem(ghost1)){

gosx1=450/2;

gosy1=450/2;

ghoststart1=false;

}

if(pacman->collidesWithItem(ghost2)){

gosx2=450/2;

gosy2=450/2;

ghoststart2=false;

}

if(pacman->collidesWithItem(ghost3)){

gosx3=450/2;

gosy3=450/2;

ghoststart3=false;

}

waka->play(":/Sound/pacman\_eatghost.wav");

}else{

if (lives > 0){

waka->play(":/Sound/pacman\_death.wav");

lives -= 1;

retry();}

else{

scene->addItem(text);

end\_Game();

}

}

}

}

// обработка нажатия клавиши

void Pacmanwindow::keyPressEvent(QKeyEvent \*event)

{

// если пробел то начинаем игру

if(event->key()==Qt::Key\_Space)

{

if(!playing){

start\_Game();

playing=true;

}else{

}

}

if(gosx3==450/2){

state++;

}

if(state==4){

start=true;

state=0;

}

if(!start){

} else {

switch(event->key())

{

case Qt::Key\_Left:

if(!moving){

direction=1;

}else{

nextdirection=1;

}

break;

case Qt::Key\_Right:

if(!moving){

direction=4;

}else{

nextdirection=4;

}

break;

case Qt::Key\_Down:

if(!moving){

direction=3;

}else{

nextdirection=3;

}

break;

case Qt::Key\_Up:

if(!moving){

direction=2;

}else{

nextdirection=2;

}

break;

case Qt::Key\_A:

if(!moving){

direction=1;

}else{

nextdirection=1;

}

break;

case Qt::Key\_D:

if(!moving){

direction=4;

}else{

nextdirection=4;

}

break;

case Qt::Key\_S:

if(!moving){

direction=3;

}else{

nextdirection=3;

}

break;

case Qt::Key\_W:

if(!moving){

direction=2;

}else{

nextdirection=2;

}

break;

break;

case Qt::Key\_P:

pause();

break;

case Qt::Key\_C:

on\_controlsButton\_clicked();

break;

case Qt::Key\_Escape:

break;

default:

break;

}

}

}

int scarestate=0;

int soundstate=0;

int timeGame;

// метод обновления карты и табло

void Pacmanwindow::updater()

{

ui->livesLcdNumber\_2->display(lives);

ui->levelLcdNumber\_2->display(level);

checklost();

pacman\_move();

for(int i=0;i<ballpoints.size();i++){

if(pacman->pacx==ballpoints[i].x() && pacman->pacy==ballpoints[i].y()){

ballpoints.remove(i);

score++;

if(score > hiscore)

hiscore = score;

text->score = score;

ui->highScroreLcdNumber\_2->display(hiscore);

ui->scoreLcdNumber\_2->display(score);

if(soundstate==0){waka->play(":/Sound/pacman\_chomp.wav");soundstate++;}

else{soundstate++;if(soundstate==6){soundstate=0;}}

}

}

for(int i=0;i<Powerballpoints.size();i++){

if(pacman->pacx==Powerballpoints[i].x() && pacman->pacy==Powerballpoints[i].y()){

Powerballpoints.remove(i);

if(scared){scarestate=0;}

scared=true;

waka->play(":/Sound/pacman\_extrapac.wav");

}

}

if(scared){

ghost->is\_Scared=true;

ghost1->is\_Scared=true;

ghost2->is\_Scared=true;

ghost3->is\_Scared=true;

scarestate+=1;

if(scarestate==1){ghoststimer->setInterval(150);}

if(scarestate==100){

ghost->whiteb=true;

ghost1->whiteb=true;

ghost2->whiteb=true;

ghost3->whiteb=true;

}

if(scarestate==150){

scared=false; scarestate=0; ghoststimer->setInterval(75);

ghost->whiteb=false;

ghost1->whiteb=false;

ghost2->whiteb=false;

ghost3->whiteb=false;

}

}else{

ghost->is\_Scared=false;

ghost1->is\_Scared=false;

ghost2->is\_Scared=false;

ghost3->is\_Scared=false;

}

if (timeGame > -1)

{

timeGame++;

ui->timeLcdNumber\_2->display(timeGame/30);

}

text->timeElapsed = timeGame/30;

text->name = this->name;

text->difficulty = difficultyText;

ball->setpoints(ballpoints);

powerball->setpoints(Powerballpoints);

pac\_map->setballpoints(ballpoints);

pac\_map->setpowerballpoints(Powerballpoints);

pac\_map->fillpacpoints(pacx,pacy);

ghost->advance();

ghost1->advance();

ghost2->advance();

ghost3->advance();

this->setFocus();

scene->update(pac\_map->boundingRect());

pacman->advance();

}

void Pacmanwindow::pause(){

if (!playing)

return;

isPaused = !isPaused;

if (isPaused) {

timer->stop();

ghoststimer->stop();

text->paused=true;

scene->addItem(text);

text->setZValue(6);

} else {

scene->removeItem(text);

timer->start();

ghoststimer->start();

}

update();

}

void Pacmanwindow::ghostupdater()

{

if(ghoststart || ghoststart1 || ghoststart2 || ghoststart3){

if(ghoststart){ghostsmove();}

if(ghoststart1){ghostsmove1();}

if(ghoststart2){ghostsmove2();}

if(ghoststart3){ghostsmove3();}

}

if(gosx3==450/2){

state++;

}

if(state==4 || 0){

start=true;

state=0;

}

if(!start){

moveghostsinrect3();

moveghostsinrect2();

moveghostsinrect1();

moveghostsinrect();

}

else if(!ghoststart3 || !ghoststart2 || !ghoststart1 || !ghoststart){

QPoint p,p1,p2,p3;

if(!ghoststart3){

gosy3-=5;

ghost3->setgosx(gosx3);

ghost3->setgosy(gosy3);

p3.setX(gosx3);

p3.setY(gosy3);

if(pac\_map->pacpoints.contains(p3)){ghoststart3=true;}

}

if(!ghoststart2){

gosy2-=5;

ghost2->setgosx(gosx2);

ghost2->setgosy(gosy2);

p2.setX(gosx2);

p2.setY(gosy2);

if(pac\_map->pacpoints.contains(p2)){ghoststart2=true;}

}

if(!ghoststart1){

gosy1-=5;

ghost1->setgosx(gosx1);

ghost1->setgosy(gosy1);

p1.setX(gosx1);

p1.setY(gosy1);

if(pac\_map->pacpoints.contains(p1)){ghoststart1=true;}

}

if(!ghoststart){

gosy-=5;

ghost->setgosx(gosx);

ghost->setgosy(gosy);

p.setX(gosx);

p.setY(gosy);

if(pac\_map->pacpoints.contains(p)){ghoststart=true;}

}

}

}

Pacmanwindow::~Pacmanwindow()

{

delete ui;

}

void Pacmanwindow::on\_pauseButton\_clicked()

{

pause();

}

void Pacmanwindow::on\_controlsButton\_clicked()

{

if(isPaused == false){

pause();

QPalette palette;

palette.setColor(QPalette::Background, Qt::black);

QMessageBox \*controls = new QMessageBox;

controls->setPalette(palette);

controls->about(this,

tr("Управление"),

tr("<p align='center'><font><h1>Управление</h1><br>Нажимайте стрелочки и WASD для перемещения Пакмана по экрану.<br><br><br>Нажмите Esk для выхода из приложения.<br><br><br>Нажмите P для паузы</font></p>"));

}

}

// powerball.cpp

#include "powerball.h"

PowerBall::**PowerBall**()

{

ballx=0;

bally=0;

ballw=5;

ballh=5;

generatePointPixmap();

}

QRectF PowerBall::***boundingRect***() const

{

return QRect(0, 0, 450, 550);

}

void PowerBall::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

for(int i=0;i<points.size();i++){

painter->drawPixmap(points[i].x()-5, points[i].y()-5, pBallPix);

}

}

void PowerBall::**setpoints**(QVector<QPoint> points)

{

this->points.clear();

this->points=points;

}

void PowerBall::**drawballs**(QPainter \*painter)

{

for(int i=0;i<points.size();i++){

painter->drawPixmap(points[i].x()-5, points[i].y()-5, pBallPix);

}

}

void PowerBall::**generatePointPixmap**()

{

QRect bounds = QRect(0, 0, 10, 10);

QPainter painter;

pBallPix = QPixmap(bounds.size());

pBallPix.fill(Qt::transparent);

painter.begin(&pBallPix);

QPen pen=QPen(Qt::yellow);

painter.setRenderHint(QPainter::Antialiasing);

painter.setPen(pen);

painter.setBrush(Qt::red);

painter.drawEllipse(0, 0, 10, 10);

}

// textdrawing.cpp

#include "textdrawing.h"

TextDrawing::**TextDrawing**()

{

over=false;

w=350;

h=50;

x=450/2-w/2;

y=480/2-h/2;

score=0;

playing=false;

paused=false;

}

QRectF TextDrawing::***boundingRect***() const

{

return QRect(x, y, w, h);

}

void TextDrawing::**SetOver**(bool over)

{

this->over=over;

}

void TextDrawing::***paint***(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

QPen pen(Qt::blue);

painter->setPen(pen);

QFont font=painter->font() ;

if(!playing){

font.setPointSize ( 18 );

painter->setFont(font);

if(over){

font.setPointSize ( 32 );

QPen pen(Qt::red);

painter->setPen(pen);

painter->setFont(font);

painter->drawText( x+80,y-180, "GAME OVER" );

font.setPointSize ( 18 );

QPen highPen(Qt::yellow);

painter->setPen(highPen);

painter->setFont(font);

painter->drawText( x+120,y-110, "HIGHSCORE" );

painter->drawText( x+0,y-70, "Tom" );

painter->drawText( x+80,y-70, QString::number(score+1000));

painter->drawText( x+140,y-70, "Ultra Hard" );

painter->drawText( x+250,y-70, "1000 Seconds" );

painter->drawText( x+0,y-50, name );

painter->drawText( x+80,y-50, QString::number(score));

painter->drawText( x+140,y-50, difficulty );

painter->drawText( x+250,y-50, QString::number(timeElapsed)+" Seconds" );

painter->drawText( x+0,y-30, "Jeff" );

painter->drawText( x+80,y-30, QString::number(score-10));

painter->drawText( x+140,y-30, difficulty );

painter->drawText( x+250,y-30, "53 Seconds" );

painter->drawText( x+0,y-10, "AAA" );

painter->drawText( x+80,y-10, QString::number(score-100));

painter->drawText( x+140,y-10, "medium" );

painter->drawText( x+250,y-10, "89 Seconds" );

painter->drawText( x+0,y+10, "Doug" );

painter->drawText( x+80,y+10, QString::number(score-200));

painter->drawText( x+140,y+10, "easy" );

painter->drawText( x+250,y+10, "100 Seconds" );

}else{

painter->drawText( *boundingRect*(),Qt::AlignCenter, "Нажмите пробел для старта" );

font.setPointSize ( 32 );

QPen pen(Qt::yellow);

painter->setPen(pen);

painter->setFont(font);

painter->drawText( x+100,y-180, "Pac Man" );

}

}

if(paused && !over)

{

font.setPointSize ( 18 );

painter->setFont(font);

QPen pen(Qt::red);

painter->setPen(pen);

painter->drawText(*boundingRect*(), Qt::AlignCenter, "PAUSE");

}

}