|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** | |
|  | Институт Кибернетики | |
|  | Кафедра программного обеспечения систем радиоэлектронной аппаратуры | |

|  |  |
| --- | --- |
| **КУРСОВОЙ ПРОЕКТ (РАБОТА)**  **по дисциплине** | |
| «Методы и стандарты программирования»  (название дисциплины)  **Тема курсового проекта (работы)**  «Игра Pacman»  (наименование темы) | |
| Студент группы КМБО-02-19 *Дугин И. А.*  *(учебная группа)* |  |
| Руководитель курсового проекта (работы) *Милонов Г. А.*  *должность, звание, ученая степень* |  |
| Рецензент  *должность, звание, ученая степень* |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_\_\_\_2020 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_\_\_\_2020 г. |  |

Москва 2020 г.

**Оглавление**

[Введение 3](#_Toc59992651)

[Описание задачи 3](#_Toc59992652)

[Решение задачи 3](#_Toc59992653)

[Руководство пользователя 3](#_Toc59992654)

[Запуск 3](#_Toc59992655)

[Управления 4](#_Toc59992656)

[Механика игры 4](#_Toc59992657)

[Пакман = Герой 4](#_Toc59992658)

[Враги = Боты = Приведения 4](#_Toc59992659)

[Очки 5](#_Toc59992660)

[Конец игры 5](#_Toc59992661)

[Документация 5](#_Toc59992662)

[Сборка программы 5](#_Toc59992663)

[Движок 6](#_Toc59992664)

[Инициализация 6](#_Toc59992665)

[Главный цикл 6](#_Toc59992666)

[Unit 7](#_Toc59992667)

[Графика 7](#_Toc59992668)

[Поля. 7](#_Toc59992669)

[Методы. 8](#_Toc59992670)

[Заключение 9](#_Toc59992671)

# Введение

Тема данной курсовой работы – создание игры в стиле “Pacman”. Для начала необходимо было изучить механику уже существующей игры и, выбрав технические средства для её реализации, приступить к разработке самой игры.

# Описание задачи

Задача – создать игру по образу и подобию игры «Pacman», реализующую механизм передвижения юнитов и их взаимодействия, механизм начисления очков и механизм изменения в процессе игры самих юнитов.   
Идея игры заключается в том, что герой (пакман) должен очистить всё поле от «очков» с наивысшим счётом. Таким образом пакман погибает при обычном столкновении с приведением, но может их съесть, находясь в режиме «силы». Также у героя имеется 3 жизни, т.е. попытки.

# Решение задачи

Для создания всей игры была использована среда разработки Qt 5.15.0. С++ стандарта C++11, а для корректирования графики библиотека «QPainter».  
Графика игры состоит частично из спрайтов, представленнных в виде картинок типа «.png», всё остальное место занимает фон программы, а управление происходит с помощью клавиатуры.

Для реализации управления через клавиатуру была выбрана библиотека «QKeyEvent», так как в игре принимают участие только 4 кнопки, соответствующие четырём сторонам: UP, DOWN, LEFT, RIGHT.

Экран обновляется с помощью переопределённого метода *void timerEvent(QTimerEvent \*e).*

# Руководство пользователя

Для того, чтобы начать игру, стоит ознакомиться с разделом руководство пользователя, в котором описано как запустить игру под Windows, как создать игровое поле и ботов.

## Запуск

Для запуска игры требуется запустить (нажать два раза) файл под названием Pacman.exe под Windows. При запуске игры открывается заставка. Для того, чтобы перейти к игре, необходимо нажать кнопку “НАЧАТЬ ИГРУ” (рис. 1). Чтобы выйти из игры необходимо закрыть окно программы.

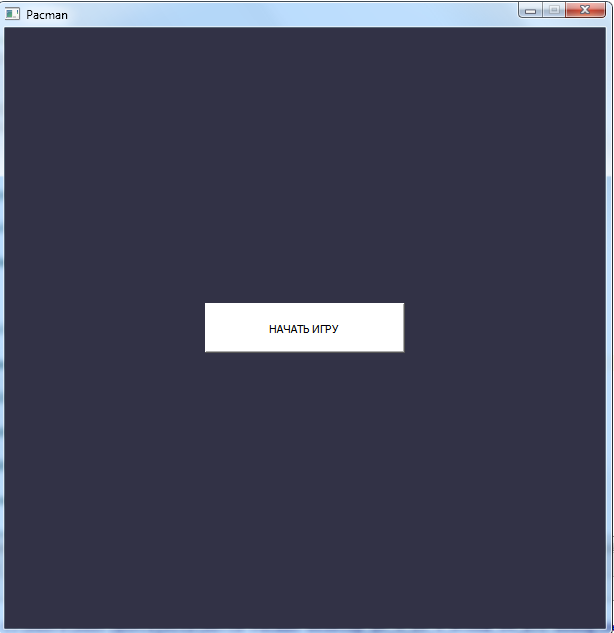


Рисунок 1 (главное меню игры)

## Управления

После перехода к игре действует следующее управление:

* UP/DOWN/LEFT/RIGHT – управление пакманом: вверх, вниз, влево, вправо.

# Механика игры

## Пакман = Герой

Игрок управляет главным героем. Герой:

* Собирает («ест») очки;
* Ускоряется, когда съедает «таблетку силы», дающее ему на 10 секунд возможность поедать ботов на удвоенной скорости;
* При соприкосновении с призраком погибает, теряя одну жизнь;
* Имеет 3 жизни.

## Враги = Боты = Приведения

На экране помимо главного героя присутствуют и враги, каждый из которых:

* Имеет неограниченный запас жизней;
* Имеет 5 секундное восстановление после съедения пакманом;
* Уничтожает пакмана при соприкосновении с ним в обычном режиме героя;
* Свободно перемещается по карте.

## Очки

Также присутствует механика очков.

* Игрок начинает игру с нулём очков;
* Далее за каждый собранный балл с каждой новой клетки поля, игрок получает +15 очков к общему счёту;
* В режиме «силы» балл прибавляет 30 очков к общему счёту пользователю»;
* При уничтожении приведения в режиме «силы», очки за каждого нового бота, уничтоженного до истечения данного режима соответствуют (2^n) \* 100, где n – порядок убитого бота. После того, как истечения режима «силы» n сбрасывается до n = 1.

## Конец игры

Игра заканчивается в двух случаях:

* Приведение съедает пакмана в 3ий раз;
* На поле не остается ни одной заполненной дополнительными баллами клетки.

После того, как главный герой соберёт все возможные очки на поле или погибнет 3 раза, пользователю выплывает QMessageBox с вопросом, желает ли пользователь сыграть ещё раз. В случае отрицательного ответа игра закрывает все окна и заканчивается (рис. 2), в случае положительного – игра открывает главное меню с кнопкой «НАЧАТЬ ИГРУ» (рис. 1).

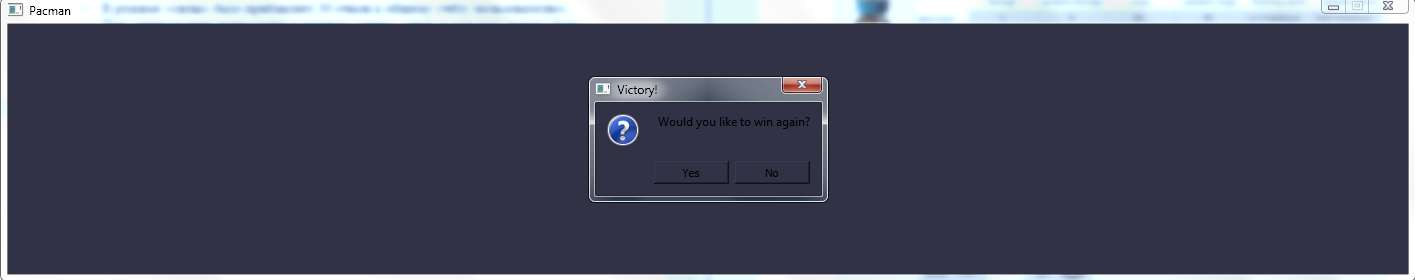


Рисунок 2 (конец игры)

# Документация

## Сборка программы

Для запуска игры на Windows требуется установить Qt 5.15.0.

Если игра запускается через Qt, то достаточно просто запустить программу, если же игра запускается не через Qt, то запуск происходит с помощью двойного клика по исполняемому файлу Pacman.exe.

## Движок

В данный момент движок реализует работу с двумерной графикой, таймерами и стеком состояний игры.

## Инициализация

Главный класс движка – class Game. Для его инициализации необходимо в главной функции программы, в методе, обрабатывающем нажатие на кнопку «Start Game» *void MainWindow::on\_button\_start\_game\_clicked()* создать объект этого класса, подсоединить сигнал открытия нового окна из MainWindow (окна меню) и вывести окно:

*sWindow = new Game();*

*connect(sWindow, &Game::firstWindow, this, &MainWindow::show);*

*sWindow->show();*

Таким образом, сразу после создания объекта класса инициализируется двумерный массив поля, считывая информацию о его (поля) структуре из файла «.txt», где:

* «-» - стены;
* «1» - клетки, на которых расположены очки;
* «p» - клетки, на которых расположены таблетки «силы»;
* «0» - пустые клетки.

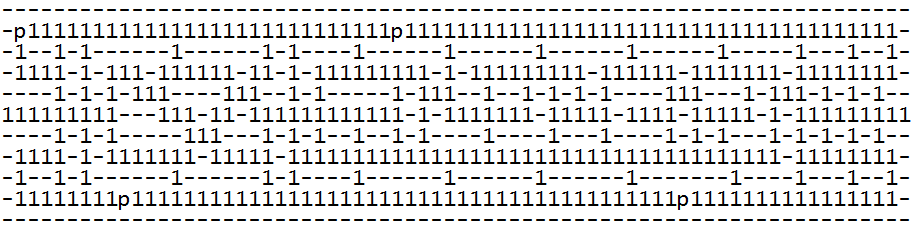
**

Рисунок 3 (текстовое поле в файле ".txt")

А также в самом конструкторе вызывается метод *void Game::initGame(),* отвечающий за инициализацию всех переменных игры и юнитов и запуск таймера (*void Game::timerEvent(QTimerEvent \*e)).*

## Главный цикл

Логика игры и отрисовки описана в классе Game. За отрисовку отвечает переопределённый метод *void Game::paintEvent(QPaintEvent \*event),* вызывающий метод *void Game::doDrawing(),* который и выполняет всю логику отрисовки поля и всех «юнитов» на нём.

Game содержит следующие переопределённые методы:

* *void timerEvent(QTimerEvent \*event)* – таймер, метод, который «толкает» программу, заставляя её обновляться один раз в определённое время и производить операции движения юнитов, их отрисовку на поле (с помощью *this->repaint()*), а так же проверяя завершение игры, вызывая в случае её завершения метод *void Game::gameOver(bool flag), который* выполняет обработку дальнейших действий ползователя;
* *void keyPressEvent(QKeyEvent \*event)* – обработка нажатий клавиатуры;
* *void paintEvent(QPaintEvent \*event)* – обновление экрана.

А так же имеет перечисление, отвечающее за удобное хранение направления движения каждого отдельного юнита:

*enum Directions {*

*left, right, up, down, sleep*

*};*

### Unit

Класс Unit – класс всех юнитов на поле. Он служит для хранения данных о каждом отдельном «движущимся» объектом игры со следующими переменными:

* *Directions \_direction, \_future\_direction, \_last\_direction* – переменные, определяющие текущее направление движения, предыдущее направление движения (часто: *\_direction* = *\_last\_direction*) и, соответственно, будущего направления. Такой механизм необходим для того, чтобы пользователь не пытался вовремя нажать на поворот, но мог нажать поворот заранее, для оптимизации управления. Таким образом при наличии блока в направлении только что нажатой кнопки, бот будет использовать *\_last\_direction*, продолжая идти в предыдущем направлении, пока не появится возможность свернуть в направлении *\_future\_direction*;
* *QString \_name* – имя юнита;
* *int \_speed* – скорость юнита;
* *int \_x, \_y* – координаты юнита;
* *bool \_alive* – переменная жив/мёртв;
* *int \_reabilitation\_time* – окончание времени реабилитации (т.е. когда *\_alive* станет true).

## Графика

Структура, используемая для передачи данных при отрисовке графики.

### Поля.

*«*

*static const int DOT\_WIDTH = 20;*

*static const int DOT\_HEIGHT = 20;*

*static const int DOT\_SIZE = 20;*

*static const int FIELD\_WIDTH = 70;*

*static const int FIELD\_HEIGHT = 11;*

*static const int TIMER\_EVENT = 50;*

*» -* константные статические переменные, характеризующие размеры поля, клеток, юнитов и окна игры, частоту обновления экрана.

*«  
QPixmap \*IMAGE\_WALL = new QPixmap(":/images/wall.png");*

*QPixmap \*IMAGE\_PACMAN = new QPixmap(":/images/pacman\_right\_2.png");*

*QPixmap \*IMAGE\_PACMAN\_POWER = new QPixmap(":/images/pacman\_power.png");  
» - переменные, описывающие блоки графического интерфейса, т.е. блоки стены, иконки юнитов.*

* *bool \_isGameOver   
  –* вызов метода *Game::gameOver(bool flag);*
* *bool \_inGame*– пересёкся ли пакман с ботом;
* *bool \_pacman\_power\_beans   
  –* действует ли сейчас «сила» пакмана;
* *int \_chet\_time  
   –* проверка чётности текущего времени;
* *int \_end\_pacman\_power   
  –* ожидаемое время завершения действия «силы» пакмана;
* *int \_time   
  –* переменная текущего времени, изменяющаяся каждый раз, прибавлением TIMER\_EVENT, для удобства отслеживания окончания времени реабилитации ботов и окончания действия «силы» пакмана;
* *int \_lives  
   –* количество жизней пакмана;
* *int \_cf\_new\_kill = 1*– коэффициент за новый убийства пакмана во время действия «силы»;
* *int \_point   
  –* счёт очков пакмана;
* *Unit \*\_pacman  
  –*  пакман;
* *Unit \*\_bot\_1, \*\_bot\_2, \*\_bot\_3, \*\_bot\_4*– боты;
* *char\*\* \_field = new char \*[FIELD\_HEIGHT]   
  –* динамический массив поля заполненный из файла;
* *QVector <QPoint> \_points\_wall   
  –* стены.

### Методы.

* *void initGame()*   
  - инициализация всех полей игры;
* *void gameOver(bool flag)*- завершение игры, закрытие окна Game и переход к окну меню;
* *void isInGame()*- на основе метода *isXYCrosses(Unit \*bot)* произведение операций с юнитами;
* *void redraw(Unit::Directions dir)*- отрисовка пакмана и бота на изначальных позициях, после смерти пакмана с задержкой в 1 секунду для удобства игры;
* *void doDrawing()*- перерисовка экрана;
* *void drawUnit(Unit \*unit)*- перерисовка юнитов, вызываемая из doDrawing();
* *void move\_bot(Unit \*unit)*- метод, определяющий движение каждого отдельного бота, выбор одной из 4-х сторон при наличии более 3-х доступных путей (*isNewWay()*), при наличии 2-ух бот идёт, как шёл раньше (т.е. не может просто так развернуться и пойти обратно);
* *void move(Unit \*unit)*- обработка *\_future\_direction* для каждого юнита, корректирующий движение всех юнитов, изменяющий координаты юнитов, их направления движения для дальнейшей отрисовки;
* *void reloadField()*- обновление поля и очков на нём поля;
* *void checkReabilitationBots()*- проверка окончания реабилитации ботов;
* *bool isXYCrosses(Unit \*bot)*- проверка пересечения пакмана с ботами;
* *bool isWall(Unit::Directions dir, Unit \*unit)*- проверка наличия доступного пути по определённому направлению для определённого юнита;
* *bool isPoint(Unit \*unit)*- проверка наличия на текущей клетке балла очков или «силы»;
* *bool isPointPacmanPower(Unit \*unit)*- действует ли сейчас сила пакмана;
* *bool isNewWay(Unit \*unit)*- есть ли 3 и более доступных направления движения.

# Заключение

В данной работе была выполнена поставленная задача – реализация игры Pacman. Также, в процессе выполнения работы были частично изучены библиотеки «QPainter», «QFile», «QMessageBox», «QWidget» и «QKeyEvent». Также был реализован некоторый “игровой движок”, позволяющий реализовывать игры, основанные на двумерной графике, таймерах и интерполяции, в том числе и игру Pacman.

**Ссылки**

* Исходные файлы: https://github.com/JohnDugin/Pacman.git
* Документация Qt: https://doc.qt.io/qt-5/
* C++ 11: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf