**Министерство образования и науки Российской федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)». Университетский колледж информационных технологий**

Специальность 09.02.03 «Программирование в компьютерных системах»

КУРСОВОЙ ПРОЕКТ

Модуль ПМ.01 Разработка программных модулей программного обеспечения

для компьютерных систем

МДК.01.02 Прикладное программирование

на тему «Игра Pacman»

**Пояснительная записка**

**УКИТ 09.02.03.2016.304.06ПЗ**

|  |  |  |
| --- | --- | --- |
| Группа | П-304 |  |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Головешкин Р. И. |
| Руководитель проекта | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Глускер А. И. |

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МГУТУ им. К. Г. Разумовского (ПКУ)»  
Университетский колледж информационных технологий

|  |  |
| --- | --- |
| Рассмотрено предметной (цикловой) комиссией специальности «Программирование в компьютерных системах»  Протокол №1 от «31» августа 2016 года  Председатель \_\_\_\_\_\_\_\_\_\_\_\_ А. И. Кириллов | **Утверждаю**  Заместитель директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. В. Лындина  «\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ года. |
|  |  |

ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студент Головешкин Роман Игоревич группы П-304  
Специальность 09.02.03 «Программирование в компьютерных системах»

1. Наименование проекта: Разработка программы «Pacman»

2. Основание для разработки: учебный план специальности 09.02.03 «Программирование в компьютерных системах»

3. Срок сдачи проекта: 30 декабря 2016 года

4. Этапы разработки   
(a) Разработка спецификации программного продукта (до 7.10.2016)   
(b) Разработка программы и методики испытаний (8.10.2016-14.10.2016)   
(c) Проектирование программного продукта (15.10.2016-28.10.2016)  
(d) Реализация программного продукта на языке программирования (29.10.2016-09.12.2016)   
(e) Тестирование программного продукта   
(f) Разработка пользовательской документации (10.12.2016-16.12.2016)   
(g) Подготовка доклада и презентации (17.12.2016-23.12.2016)   
(h) Оформление пояснительной записки (24.12.2016-30.12.2016)

5. Содержание пояснительной записки   
(a) Титульная страница  
(b) Задание на курсовое проектирование  
(c) Содержание   
(d) Введение   
(e) Основная часть

i. Введение в предметную область   
ii. Спецификация   
iii. Программа и методика испытаний   
iv. Технический проект программного изделия   
v. Реализация программного изделия на языке программирования   
vi. Тестирование программного продукта

(f) Заключение

(g) Список использованных источников

(h) Приложения: код программы; техническая документация (автоматически генерируемая); пользовательская документация; текст доклада; слайды презентации; протокол системы контроля версий; отчёт системы учёта ошибок;

6. Порядок приемки работы: исполнитель предъявляет пояснительную записку и электронный носитель, содержащий исполнительный код, исходный код, презентацию, пояснительную записку, репозиторий системы контроля версий

Руководитель проекта \_\_\_\_\_\_\_\_ , «30» сентября 2016 г. А. И. Глускер

Задание принял к исполнению \_\_\_\_\_\_\_\_\_, «\_\_» 201\_г. Р. И. Головешкин

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc501627506)

[ОСНОВНАЯ ЧАСТЬ 7](#_Toc501627507)

[1 Введение 7](#_Toc501627508)

[2 Основание для разработки 7](#_Toc501627509)

[3 Назначение разработки 7](#_Toc501627510)

[4 Требования к программе или программному изделию 7](#_Toc501627511)

[8 Цель испытаний 10](#_Toc501627512)

[9 Требования к программе 10](#_Toc501627513)

[10 Требования программной документации 11](#_Toc501627519)

[11 Методы испытаний 12](#_Toc501627520)

[12 Технический проект 18](#_Toc501627521)

[16 Реализация программного изделия на языке программирования 19](#_Toc501627522)

[17 Тестирование программного продукта 22](#_Toc501627523)

[ЗАКЛЮЧЕНИЕ 25](#_Toc501627524)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc501627525)

[ПРИЛОЖЕНИЕ А Руководство оператора 27](#_Toc501627526)

[ПРИЛОЖЕНИЕ Б Исходный код 28](#_Toc501627527)

[ПРИЛОЖЕНИЕ В Протокол системы контроля версий 35](#_Toc501627528)

[ПРИЛОЖЕНИЕ Г Руководство программиста 36](#_Toc501627529)

[ПРИЛОЖЕНИЕ Д Доклад 38](#_Toc501627530)

# ВВЕДЕНИЕ

Благодоря компьютерам стало возможно улучшать и обучать человека не только в стандартных формах, но и в игровой.

На данный момент существует множество игр, которые предназначены для развития логики, реакции, быстрого принятия решений, концетрации, внимания.

Цель курсового проекта – разработать игру на примере Pacman, которая сможет тренировать, такие части мозга, которые отвечают за логику, внимания и быстрого принятия решений в целом, также разработать документацию к ней.

Основные задачи, которые были поставлены при разработке игры Pacman:

* Анализ предметной области;
* Выбор средств реализации;
* Прорисовка персонажа;
* Реализация движения игрока;
* Вывод статистики игрока;
* Создание программной документации к игре Pacman.

Используемые методы при разработке игры Pacman:

* Анализ;
* Изучение дополнительной литературы;
* Изучение необходимых программ;
* Экспиримент, как проведение опытов.

Структура курсового проекта:

Пояснительная записка составлена таким образом, чтобы была возможность отследить программный продукт на всем жизненном цикле. А точнее от появления идеи о создании и до предоставления результатов тестирования и выводах о готовности программного обеспечения к выпуску.

1. Основная часть:

Раздел описывает предметную область, для которой разрабатывается игра Pacman.

1. Спецификация:

Раздел спецификации описывает курсовой проект игры «Pacman», определяет его предметную область, основной функционал программы и условия эксплуатации, в которых данная программа будет стабильно работать. Еще в данном разделе описаны требования к программной документации, которая определяет другие характеристики программы курсового проекта.

1. Программа и методика испытаний:

Раздел описывает применяемые методы тестирования и тестовые примеры к разрабатываемой игре Pacman.

1. Технический проект:

Раздел описывает архитектурные решения, который применяются при разработке.

1. Реализация программного продукта:

Раздел описывает процесс разработки игры Pacman.

1. Тестирование программного продукта:

Раздел описывает результаты тестирования игры Pacman по программе и методике испытаний.

1. Заключение:

Раздел подводит итог по проделанной работе, о навыках, которые были получены, о достоинствах и недостатке программы игры Pacman.

1. Список используемых источников:

Раздел приводиться список используемых источников, которые использовались при разработке игры Pacman.

1. Приложения:

Раздел приводятся дополнительная информация, которая включает в себя руководство оператора, исходный код приложения, протокол системы контроля версий, руководство программиста, доклад.

# Основная часть

## Введение

### Наименование программы

Игра Pacman

### Область применения

Игровая деятельность.

Для людей, которые хотят потренировать свои зоны мозга, отвечающие за логику, внимания и быстрого принятия решений в целом.

### Объект, в котором используется программа

Игра может быть использована на ПК пользователя с характеристиками, описанными в пункте 4.2 Основной.

## Основание для разработки



### Документ, на основании которого ведётся разработка

Техническое задание на курсовой проект.

### Наименование и (или) условное обозначение темы для разработки

Pacman.



## Назначение разработки



### Функциональное назначение

Тренировать зоны мозга, отвечающие за логику, внимания и быстрого принятия решений в целом, также может использоваться как, расслабляющий элемент.

### Эксплуатационное назначение

Игра Pacman предназначена для пользователей ПК, которые хотят потренироватать зоны мозга, отвечающие за логику, внимания и быстрого принятия решений в целом.

## Требования к программе или программному изделию



### Требования к информационному обеспечению



### Требования к составу и параметрам технических средств

В состав технических средств должен входить компьютер клиента, включающий:

* Процессор Pentium 4 или выше;
* Клавиатура и мышь;
* Видеокарта;
* Монитор;
* HDD объёмом свободного пространства не менее 1 ГБ;
* Windows XP и выше.

### Требования к информационной и программной совместимости

### Требования к информационным структурам на входе и выходе не предъявляются

### Требования к методам решения

Игра должна быть разработана с использованием следующий технологий:

* С++;
* Библиотека SFML;
* Графический редактор Paint и Photoshop;
* Visual Studio 2015.

### Требования к программной документации

### Состав программной документации

* Состав программной документации должен включать:
* Техническое задание;
* Пояснительная записка;
* Текст приложения;
* Программа и методика испытания;
* Руководство пользователя;
* Руководство программиста.

### Специальные требования к пояснительной записке

Специальные требования не предъявляются.

### Стадии и этапы разработки

### Стадии разработки

Разработка осуществляется в три стадии:

* Техническое задание;
* Технический проект;
* Рабочий проект.

### Этапы разработки

На стадии техническое задание осуществляется разработка, согласование и утверждение технического задания.

На стадии технический проект осуществляется разработка, согласование и утверждение пояснительной записки.

На стадии рабочий проект осуществляется разработка текста программы.

### Порядок контроля и приёмки

### Приёмосдаточные испытания должно проводиться в соответствии с программой и методикой испытаний, разработанной, согласованной и утверждённой не позднее 31 декабря 2016 года

### Вывод

В данном разделе были изложены все основные особенности программного продукта и требования к нему.

### Программа и методика испытания



### Введение

Данный раздел описывает основные методы тестирования для проверки соответствия программного продукта функциональным требованиям.

### Программа и методика испытания

### Объект испытания

### Наименование

Игра Pacman



### Область применения

Для людей, которые хотят потренировать свои зоны мозга, отвечающие за логику, внимания и быстрого принятия решений в целом.

### Обозначение программы

Pacman

## Цель испытаний

Проверка соответствия программного продукта требованиям технического задания.

## Требования к программе

### Требования к функциональным характеристикам

### Требования к составу выполняемых функций

Программа должна обеспечивать выполнение следующих функций – пункт 4.2.4 Технического задания.

### Требования к информационному обеспечению

Пункт 4.1.1 Технического задания.

### Требования к надёжности

Пункт 4.3 Технического задания.

### Требования к информационной и программной совместимости

Пункт 4.6 Технического задания



## Требования программной документации



### Состав программной документации

Состав программной документации должен включать:

* Техническое задание;
* Пояснительная записка;
* Текст приложения;
* Программа и методика испытания;
* Руководство пользователя;
* Руководство программиста.

Пункт 5.1 Технического задания

### Специальные требования к пояснительной записке

Специальные требования не предъявляются пункт 5.2 Технического задания

### Требования к исходным кодам изложены в документе:

А. И. Глускер «Сборник задач по курсу “Основы алгоритмизации и программирования” [Электронный ресурс] – 2011 /раздел 3.1/ (п. 4.5.3 технического задания)

### Программа должна быть написана на языке С++ и компилироваться транслятором Средства

### Технические средства, используемые при проведении испытаний

В состав технических средств должен входить IBM-совместимый компьютер, включающий:

* Процессор;
* Видеокарту;
* Клавиатуру и мышь;
* Монитор;
* Жёсткий диск.

### Программные средства, используемые при проведении испытаний

В состав программных средств должно входить:

* Операционная система семейства Windows (Начиная с Windows XP);

### Порядок проведения испытаний

### Подготовка к проведению испытаний заключается в обеспечении наличия компьютера в п. 5.1, и программных средств, указанных в п. 5.2, установленных на этом компьютере

### Ход проведения испытаний документируется в протоколе, где указывается перечень проводимых испытаний, результат каждого испытания и возможно замечания.

### Состав испытания

### Проверка состава программной документации в соответствии с методом, описанном в п. 6.2

### Проверка требований к программе

### Проверка обеспечений требования к программе (п. 3) в соответствии с методом, описанным в п. 6.1

## Методы испытаний



### Метод проверки требований к программе

Проверка осуществляется путём запуска программы и сравнения ожидаемых результатов привидённых в (Таблица 1).

### Метод проверки требований к составу программной документации

Проверка состава программной документации осуществляется визуально путем сравнения набора предъявленных документов (в форме распечатки или в рукописной форме) списку, приведенному в п. 4.1. При этом исходные тексты программ должны быть предоставлены так же и в электронной форме.

В случае если набор предъявленных документов соответствует списку, а исходные тексты предоставлены также в электронной форме, то в протокол заносится запись: «Состав программной документации» – соответствует; в противном случае: «Состав программной документации» – не соответствует.

### Метод проверки требований к пояснительной записке

Проверка состоит из следующих этапов:

* проверка наличия блок-схемы (блок-схем) в пояснительной записке;
* проверка соблюдения требований ГОСТ 19.701-90 для каждой блок-схемы;
* проверка соблюдения локальных стандартов для блок-схем;
* проверка соответствия каждой блок-схемы алгоритму, закодированному в программе.

Проверка соблюдения требований ГОСТ 19.701-90 состоит из следующих работ:

* проверка использования только тех символов, которые указаны как применимые к схемам программ в п. 5 ГОСТ 19.701-90;
* проверка соответствия символов их назначению (экспертная оценка лица, проводящего испытания);
* проверка правильности выполнения соединения линий (п. 4.2.3 ГОСТ 19.701-90);
* проверка того, что линии потока управления, выходящие из символа «решение» подписана (п. 4.3.1.2 ГОСТ 19.701-90);

Проверка соблюдения локальных стандартов для блок-схем состоит из следующих работ:

* проверка того, что все символы (кроме терминаторов, соединителей, линий и комментариев) имеют одинаковые размеры;
* проверка того, что терминаторы имеют ту же ширину, что и другие символы;
* проверка того, что отношение ширины к высоте составляет 2 к 1 для каждого символа, кроме терминаторов, комментариев и линий;
* проверка того, что отношение ширины к высоте составляет 4 к 1 для терминаторов;
* проверка того, что высота соединителей совпадает с высотой терминаторов;
* проверка того, что линии потока управления входят в символ слева или сверху, а выходят снизу или справа;
* проверка того, что подписи к линиям не находятся на самих линиях.

Проверка соответствия каждой блок-схемы алгоритму, закодированному в программе, осуществляется путем экспертной оценки лицом, осуществляющим проведение испытаний. В случае, если все вышеприведенные проверки прошли успешно, в протокол заносится запись: «Специальные требования к пояснительной записке» – соответствует; в противном случае «Специальные требования к пояснительной записке» – не соответствует.

### Метод проверки к исходным кодам

Изложенный ниже метод применяется ко всем файлам, содержащим исходный текст, и входящим в состав программной документации по отдельности. Для каждого файла вносится в протокол запись: «Требования к исходным кодам для файла ##### – соответствует/не соответствует (где вместо #### указывается название файла).

Проверка состоит из следующих этапов:

* Проверка исходных кодов, написанных на C++ с использование библиотеки SFML C++;
* Наличие комментария в начале файла, содержащего автора работы, номера задания и варианта, краткой формулировки задания (или его части);
* Наличие комментариев к неочевидным действиям (проверяется методом экспертной оценки лицом, осуществляющим испытания);
* Для каждой подпрограммы наличие комментария, содержащего полное описание ее работы, описание всех аргументов и результатов. Достаточность этого комментария для возможности использовать подпрограмму в других программах (без изучения собственно текста подпрограммы);
* Для каждой глобальной переменной указание ее назначения;
* Для всех переменных, кроме переменных цикла, использование «говорящих» названий;
* Для всех подпрограмм использование говорящих названий;
* Использование одного оператора на одной строке программы;
* Количество табулирований перед строкой программы должно соответствовать уровню вложенности;
* Операторы { и } , соответствующие друг другу, располагаются строго с одной и той же позиции по вертикали;
* Количество строк в подпрограмме и в самой программе (между { и }) – не более 50 строк;
* Использование модулей для трех и более сходных по назначению подпрограмм;
* Отсутствие в подпрограммах использования глобальных переменных (напрямую);
* Разделение подпрограмм на предназначенные для вычислений (в них не должно быть ввода-вывода) и на предназначенные для ввода-вывода (в них вычисления должны быть только такие, что нужны для ввода-вывода);
* Отсутствие операторов goto, break, continue; процедур halt и exit;
* Проверка того, что вместо явно указанных значений чисел, в тексте программы используются константы.

В случае, если все перечисленные этапы пройдены, то в протокол о соответствии файла требованиям, в противном случае – о несоответствии.

Таблица – Тестовые примеры

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 1 | После захода в игру | При нажатии клавиши «Начать игру» | Должна загрузиться карта, персонажи |
| 2 | После загрузки игры | После нажатия пользователем кнопки движение стрелками, либо (W,A,S,D) для персонажа | Персонаж должен среагировать с учетом нажатия клавишей отклик не более 1 сек |
| 3 | При появлении стены перед персонажем | Персонаж приближается к стене, и пользователь нажимает клавишу перемещения, которая ведет на эту стену | Персонаж не должен перемещаться, если пользователь направляет персонажа к стене, при том что персонаж уже находится у стены и персонаж будет ожидать другой клавиши управления от пользоватея им, которая сменит направлене двжения персонажем |
| 4 | Персонаж должен находится около точки | При прохождении персонажа через точку | Точки будут пропадать, а счетчик очков будет увеличиваться прямопропорционально съеденным точкам |
| 5 | Персонаж должен находится около привидения | При столкновении персонажа с полным здоровьем с привидением | У персонажа отнимится здоровье |
| 6 | Персонаж должен потерять все здоровье | При столкновении с привидение с последним здоровьем у персонажа | Игра заканичвается, это означает, что персонаж не будет реагировать на кнопки управление нажимаемым пользователем |
| 7 | Подьзователь должен нажать «Tab» при начатой игре | При нажатии пользователем клавиши «Tab» | Будет открыто окно во время игры со статистикой |
| 8 | При нахождении персонажа у последней точки | Пользователь должен съесть последнюю точку | Игра будет закончена в пользу пользователя |

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** |
| 9 | Нажата кнопка «Tab» | Пользователь должен отпустить кнопку «Tab» | Окно со статистикой будет закрыто |
| 10 | Пользователь должен приблизиться к привидению | Привидение должно столкнуться с персонажем | Персонаж потеряет часть здоровья, а привидение исчезнет |
| 11 | Пользователь удерживает кнопку управления (W,A,S,D) | Пользователль отпускает клавишу (W, A, S, D) | После отпусканиея клавиши движения пользователем, персонаж должен остановиться на месте и ждать дальнейшего нажатие клавиши движение от пользователя |

### Вывод

В этом разделе были изложены методы, которые будут применяться к игре Pacman для проверки соответствия требованиям технического задания.

## Технический проект

### Введение

В данном разделе изложены и алгоритмы, используемые в игре Pacman

### Наименование программы

Игра Pacman

### Условное обозначение темы разработки

Pacman.

### Документ, на основании которого ведётся разработка

Техническое задание на курсовой проект.

### Назначение и область применения

### Назначение программы

Тренировать свои зоны мозга, отвечающие за логику, внимания и быстрого принятия решений в целом, также может использоваться как, расслабляющий элемент.

### Краткая характеристика области применения

Игра Pacman предназначен для использования пользователями ПК. Для тренировки в игровом стиле и отдыха.

### Технические характеристики

### Постановка задачи на разработку

Создание игры Pacman позволит пользователям:

* Улучшить внимание;
* Улучшить быстрое мышление;
* Позволит тренироваться в игровом стиле, следовательно, и отдохнуть.

### Аппаратные средства

ПК пользователя должен обладать следующими техническими характеристиками:

* Процессор: Pentium и выше;
* ОЗУ: 2GB и выше;
* HDD: 1GB и выше.

### Программные средства

Для создания приложение использовалась Visual Studio 2015 года, это обусловлено тем, что библиотека SFML C++ работает более корректно, нежели с новыми версиями Visual Studio.

### Описание используемых технологий

Для разработки приложения использовалась библиотека SFML C++.

SFML C++ обладает следующими особенностями:

1. Анимирование персонажей с помощью спрайтов;
2. Более удобная библиотка для разработки такого типа приложений;
3. Поддерживает различные темы оформления.

### Описание алгоритма

Ниже представлен алгоритм движения привидения, цель которых – не дать пользователю выйграть, следовательно, довести здоровье персонажа до 0.

### Вывод

Раздел описывает проектировочную часть разрабатываемого игры Pacman.

## Реализация программного изделия на языке программирования



### Введение

В этом разделе описывается разработка игры Pacman.

### Процесс реализации

Разработка игры Pacman началась в январе 2016 года и был затянут из-за недостатка навыков использования библиотеки SFML C++.

Срок сдачи курсового проекта был 31.12.2016, но из-за того, что разработка игры велась одним человеком и у этого человека не было навыков в Photoshop, Paint, sfml c++, то разработка была приотовлена для получения необходимых знаний для реализации данной игры.

Разработка игры Pacman была начата со сбора информации о проекте, о методах реализации некоторых функций и т.д. После сбора информации было принято решение нарисовать тайлики, то есть нарисовать персонажей при этом с несколькими позах для анимирования в будущем, карту, была выбрана темное оформелние карты для хорошего восприятия для глаз, точик (очки), вообщем была выполена вся графическая часть курсового проекта.

Следующим этапом было создание программной части приложение в файле main.cpp, была подключена библиотека SFML C++ (#include <SFML/Graphics.hpp>), как только все было подключено, были прописаны классы, как привидения (class Entity), так и персонажа (class Player), следовательно, были определены их общие свойства, которыми являются здоровье(health), движение, точнее его статус (enum { left, right, up, down} state; и т.д), так же и индивидуальные, как у персонажа счетчик очков (playscore), так и других. Более подробно предствленов исходном коде. После этого была создана карта в файле map.h и подключена в main.cpp (#include "map.h"), карта была созданна в виде массива знаков, каждый знак означал свое действие,свое имя («0» - элементы стены, «b» - пустота, «’ ’»- очки , при подборе которых счетчик «playscore» будет увеличивать прямопропорцианально сбору очков), где каждому знаку соответвала прорисованная текстурка и другие параметры. Как только это было прописано, то при запуске программы, на экран была выведена карта с персонажем, но у персонажа не было реализовано движение и самого персонажа не было видно, и я занялся этим, как только все было прописано, а точнее изображение для персонажа (image.loadFromFile("images/" + File);) бралось из тайликов, вырезку картинок я поместил в функцию (void control()), пример одной из кнопок движения в функции (void control()), выглядит она следующим образом (if ((Keyboard::isKeyPressed(Keyboard::Left)||(Keyboard::isKeyPressed(Keyboard::A))))

{state = left; p.dir = 1; p.speed = 0.07 ; CurrentFrame += 0.005\*time;if CurrentFrame > 3) CurrentFrame -= 3;p.sprite.setTextureRect(IntRect(34\*int(CurrentFrame), 33, 32, 32));}),

В этой части кода было реализовано, не только движение персонажа, но и анимация при движении персонажа. Следовательно, после запуска игры персонаж мог счасливо двигаться по карте во все стороны, кроме диагонали, что есть хорошо, а также двигаться с анимацией, что не только приятно для глаз, но и придает игре свежий взгляд. Дальше была реализована статистика при нажатии «TAB», счетчик очков, который обновлялся при поедании точек, как раз данный код показывает как это реализовано (if ((event.key.code == Keyboard::Tab)), что значит, что при нажатии клавиши «TAB» будет переключаься логичекая модель switch (showMissionText) при значении true будет выполняться следующий текст программы std::ostringstream playerHealthString;playerHealthString << p.playerScore;std::ostringstream task; task << getTextMission(getCurrentMission(p.getplayercoordinateX()));,а уже эта функция принимает в качестве аргумента функцию p.getplayercoordinateX() text.setString("Здоровье: "+playerHealthString.str() + "\n" + task.str()); showMissionText = false;break; что означает появление окна статистики, при значении логической модели false, будет выполняться следующий текст программы text.setString(""); showMissionText = true; break;,что означает скрытие окна статистики, после этого было добавлено завершение игры, как в пользу игрока, засчет сбора всех очков (if (p.playerScore == 100) {std::cout << "You Win!"<<‘\n'; system("pause"); window.close();}), так и в пользу привидений засчет потери всего здоровья у персонажа (if (p.health <= 0) {std::cout << "You Die,Maannn! It's not Funny, guy" << '\n'; system("pause");window.close();}). После всей реализации программе не хватало только привидений, которые не должны буду позволять пользователю выйграть. И я приступил к выполнению этого задания, так как привидение уже было нарисовано, осталось только его добавить в игру и заставить его взаимодествовать с картой и персонажем.

### Вывод

В этом разделе был описан процесс разработки игры Pacman.

## Тестирование программного продукта



### Введение

Тестирование исходного кода путём написания юнит-тестов не производилось.

Сдача игры Pacman происходит путём приёмо-сдаточных испытаний.

### Протоколы прохождения тестов

Все тестовые примеры были пройдены успешно. Ниже на Таблице 2 приведены результаты некоторых тестов.

Таблица 2 – Тестовые примеры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** | **Результат** |
| 1 | После захода в игру | При нажатии клавиши «Начать игру» | Должна загрузиться карта, персонажи | Положительный |
| 2 | После загрузки игры | После нажатия пользователем кнопки движение стрелками, либо (W,A,S,D) для персонажа | Персонаж должен среагировать с учетом нажатия клавишей отклик не более 1 сек | Положительный |
| 3 | При появлении стены перед персонажем | Персонаж приближается к стене, и пользователь нажимает клавишу перемещения, которая ведет на эту стену | Персонаж не должен перемещаться, если пользователь направляет персонажа к стене, при том что персонаж уже находится у стены и персонаж будет ожидать другой клавиши управления от пользоватея им, которая сменит направлене двжения персонажем | Положительный |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Предусловия** | **Действие** | **Ожидаемые результаты** | **Результат** |
| 4 | Персонаж должен находится около точки(бонусного очка) | При прохождении персонажа через точку | Точки будут пропадать, а счетчик очков будет увеличиваться прямопропорционально съеденным точкам | Положительный |
| 5 | Персонаж должен находится около привидения | При столкновении персонажа с полным здоровьем с привидением | У персонажа отнимится здоровье | Положительный |
| 6 | Персонаж должен потерять все здоровье | При столкновении с привидение с последним здоровьем у персонажа | Игра заканичвается, это означает, что персонаж не будет реагировать на кнопки управление нажимаемым пользователем | Положительный |
| 7 | Подьзователь должен нажать «Tab» при начатой игре | При нажатии пользователем клавиши «Tab» | Будет открыто окно во время игры со статистикой | Положительный |
| 8 | При нахождении персонажа у последней точки | Пользователь должен съесть последнюю точку | Игра будет закончена в пользу пользователя | Положительный |
| 9 | Нажата кнопка «Tab» | Пользователь должен отпустить кнопку «Tab» | Окно со статистикой будет закрыто | Положительный |
| 10 | Пользователь должен приблизиться к привидению | Привидение должно столкнуться с персонажем | Персонаж потеряет часть здоровья, а привидение исчезнет | Положительный |
| 11 | Пользователь удерживает кнопку управления (W,A,S,D) | Пользователль отпускает клавишу (W, A, S, D) | После отпусканиея клавиши движения пользователем, персонаж должен остановиться на месте и ждать дальнейшего нажатие клавиши движение от пользователя | Положительный |

### Вывод

В данном разделе были описаны методы тестирования, которые использовались для проверки соответствия требованиям технического задания.

# ЗАКЛЮЧЕНИЕ

Результатом работы оказался разработанная игра Pacman, которая включает в себя следующее:

Достоинства игры Pacman:

1. Доступность, так как является некомерческим проектом и имеет открытый исходный код на сайте Github;
2. Позволяет тренировать части мозга.

Недостатки разработанного игры Pacman:

1. Присутствуют баги движения Pacman и привидений;
2. Не лучшая прорисовка персонажей и уровня в целом.

В итоге, было сделано заключение о том, что разработка подобной игры Pacman должна происходит в команде из нескольких людей включающе в своем составе не только разработчиков(программистов), но и дизайнеров и наставника, который будет разбивать огромные задачи на более мелкие, для улучшения производительности, так как разрабатывать одному долго и неэффективно. Подход к разработке подобных систем давно выработан и заключается в написании модулей, интерфейсы которых унифицированы для совместимости друг с другом и непосредственного взаимодействия между ними.

Так же были закреплены навыки:

* Работа с библиотекой SFML C++ и самим языком C++;
* Работа с графическим редактором Paint и Photoshop CS5.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Уроки по sfml, форум, вики, чат, командные проекты. [Электронный ресурс]. – Режим доступа: http://kychka-pc.ru (15.07.2017г.)
2. Цикл переводов официальных уроков по графическому 2D движку SFML 2.1. [Электронный ресурс]. Режим доступа: https://progressor-blog.ru(18.09.2017г.)
3. Работа с текстурами и изображениями [Электронный ресурс].

Режим доступа: https://habrahabr.ru/post/180893/(20.02.2017г.)

# ПРИЛОЖЕНИЕ А

**Руководство оператора**

1. Назначение программы

### Наименование

Игра Pacman

### Назначение

Сделать тренировку частей мозга

### Характеристика области применения

Пользовтелям с желанием улучшить свои умественные способности.

1. Условия выполнения программы

### Аппаратные средства

В состав аппаратных средств пользователя должен входит компьютер, который включает в себя:

* Процессор;
* Видеокарта;
* Жёсткий диск;
* Сетевая карта;
* Клавиатура;
* Мышь.

### Программные средства

В состав программных средств пользователя должен входит интернет браузер Visual Studio 2015.

1. Выполнение программы

### Запуск программы

Для запуска программы необходимо открыть pacman.exe файл, на котором расположена игра Pacman.

### Руководство пользователя

Интерфейс интуитивно понятный так же, как и управление пероснажем

# ПРИЛОЖЕНИЕ Б

**Исходный код**

Исходный код игры Pacman в файле main.cpp

#include <iostream>

#include <SFML/Graphics.hpp>

#include <sstream>

#include "view.h"

#include "map.h" //подключили код с картой

#include "mission.h"

//#include "Audio.hpp"

using namespace sf;

class Player { // класс Игрока

public:

int dir,playerScore, health; //новая переменная, хранящая жизни игрока

bool life;//логическая переменная жизнь

float x, y, w, h, dx, dy,speed = 0; //координаты игрока х и у, высота ширина, ускорение (по х и по у), сама скорость

String File; //файл с расширением

Image image;//сфмл изображение

Texture texture;//сфмл текстура

Sprite sprite;//сфмл спрайт

Player(String F, float X, float Y, float W, float H) { //Конструктор с параметрами(формальными) для класса Player. При создании объекта класса мы будем задавать имя файла, координату Х и У, ширину и высоту

dir = 0; speed = 0; playerScore = 0; health = 100; // инициализировали переменную жизни в конструкторе

life = true;//инициализировали логическую переменную жизни

File = F;//имя файла+расширение

w = W; h = H;//высота и ширина

image.loadFromFile("images/" + File);//запихиваем в image наше изображение вместо File мы передадим то, что пропишем при создании объекта. В нашем случае "hero.png" и получится запись идентичная image.loadFromFile("images/hero/png");

image.createMaskFromColor(Color(255, 255, 255));//убираем ненужный темно-синий цвет, эта тень мне показалась не красивой.

texture.loadFromImage(image);//закидываем наше изображение в текстуру

sprite.setTexture(texture);//заливаем спрайт текстурой

x = X; y = Y;//координата появления спрайта

sprite.setTextureRect(IntRect(0, 0, w, h)); //Задаем спрайту один прямоугольник для вывода одного льва, а не кучи львов сразу. IntRect - приведение типов

}

void update(float time) //функция "оживления" объекта класса. update - обновление. принимает в себя время SFML , вследствие чего работает бесконечно, давая персонажу движение.

{

switch (dir)//реализуем поведение в зависимости от направления. (каждая цифра соответствует направлению)

{

case 0: dx = speed; dy = 0; break;//по иксу задаем положительную скорость, по игреку зануляем. получаем, что персонаж идет только вправо

case 1: dx = -speed; dy = 0; break;//по иксу задаем отрицательную скорость, по игреку зануляем. получается, что персонаж идет только влево

case 2: dx = 0; dy = speed; break;//по иксу задаем нулевое значение, по игреку положительное. получается, что персонаж идет только вниз

case 3: dx = 0; dy = -speed; break;//по иксу задаем нулевое значение, по игреку отрицательное. получается, что персонаж идет только вверх

}

x += dx\*time;//то движение. наше ускорение на время получаем смещение координат и как следствие движение

y += dy\*time;//аналогично по игреку

speed = 0;//зануляем скорость, чтобы персонаж остановился.

sprite.setPosition(x, y); //выводим спрайт в позицию x y , посередине. бесконечно выводим в этой функции, иначе бы наш спрайт стоял на месте.

interactionWithMap();//вызываем функцию, отвечающую за взаимодействие с картой

if (health <= 0) { life = false; speed = 0; }//если жизней меньше либо равно 0, то умираем и исключаем движение игрока после смерти

}

void interactionWithMap()//ф-ция взаимодействия с картой

{

for (int i = y / 32; i < (y + h) / 32; i++)//проходимся по тайликам, контактирующим с игроком,, то есть по всем квадратикам размера 32\*32, которые мы окрашивали в 9 уроке. про условия читайте ниже.

for (int j = x / 32; j<(x + w) / 32; j++)//икс делим на 32, тем самым получаем левый квадратик, с которым персонаж соприкасается. (он ведь больше размера 32\*32, поэтому может одновременно стоять на нескольких квадратах). А j<(x + w) / 32 - условие ограничения координат по иксу. то есть координата самого правого квадрата, который соприкасается с персонажем. таким образом идем в цикле слева направо по иксу, проходя по от левого квадрата (соприкасающегося с героем), до правого квадрата (соприкасающегося с героем)

{

if (TileMap[i][j] == '0')//если наш квадратик соответствует символу 0 (стена), то проверяем "направление скорости" персонажа:

{

if (dy>0)//если мы шли вниз,

{

y = i \* 32 - h;//то стопорим координату игрек персонажа. сначала получаем координату нашего квадратика на карте(стены) и затем вычитаем из высоты спрайта персонажа.

}

if (dy<0)

{

y = i \* 32 + 32;//аналогично с ходьбой вверх. dy<0, значит мы идем вверх (вспоминаем координаты паинта)

}

if (dx>0)

{

x = j \* 32 - w;//если идем вправо, то координата Х равна стена (символ 0) минус ширина персонажа

}

if (dx < 0)

{

x = j \* 32 + 32;//аналогично идем влево

}

}

/\*

if (TileMap[i][j] == ' ') { //если символ равен 's' (камень)

playerScore++;

TileMap[i][j] = 'b';//убираем камень, типа взяли бонус. можем и не убирать, кстати.

}\*/

if (TileMap[i][j] == ' ') {

health-=20;//если взяли ядовитый цветок,то переменная health=health-90;

TileMap[i][j] = 'b';//убрали цветок

}

if (TileMap[i][j] == 'h') {

health += 20;//если взяли сердечко,то переменная health=health+20;

TileMap[i][j] = ' ';//убрали сердечко

}

}

}

float getplayercoordinateX() {

return x;

}

float getplayercoordinateY() {

return y;

}

};

int main()

{

RenderWindow window(sf::VideoMode(640, 480), "Pacman:Game"); //принимает несколько параметров 1- размер экрана 2-title 3- стиль окна например [sf::Style::None]

view.reset(sf::FloatRect(0, 0, 640, 480));

Font font;//шрифт

font.loadFromFile("DejaVuSans.ttf");//передаем нашему шрифту файл шрифта

Text text("", font, 25);//создаем объект текст. закидываем в объект текст строку, шрифт, размер шрифта(в пикселях);//сам объект текст (не строка)

text.setStyle(sf::Text::Bold | sf::Text::Underlined);//жирный и подчеркнутый текст. по умолчанию он "худой":)) и не подчеркнутый

Image map\_image;//объект изображения для карты

map\_image.loadFromFile("images/map.png");//загружаем файл для карты

Texture map;//текстура карты

map.loadFromImage(map\_image);//заряжаем текстуру картинкой

Sprite s\_map;//создаём спрайт для карты

s\_map.setTexture(map);//заливаем текстуру спрайтом

Image quest\_image;

quest\_image.loadFromFile("images/missionbg.jpg");

quest\_image.createMaskFromColor(Color(0, 0, 0));

Texture quest\_texture;

quest\_texture.loadFromImage(quest\_image);

Sprite s\_quest;

s\_quest.setTexture(quest\_texture);

s\_quest.setTextureRect(IntRect(0, 0, 340, 510)); //приведение типов, размеры картинки исходные

s\_quest.setScale(0.6f, 0.6f);//чуть уменьшили картинку, => размер стал меньше

float CurrentFrame = 0;//хранит текущий кадр

Clock clock;

Player p("PacmanPhoto(new).png", 285, 320, 25, 25);//создаем объект p класса player,задаем "hero.png" как имя файла+расширение, далее координата Х,У, ширина, высота.

bool showMissionText = true;//логическая переменная, отвечающая за появление текста миссии на экране

while (window.isOpen())

{

if (p.health <= 0) { //условие завершения игры в пользу игрока

return EXIT\_SUCCESS; //завершение игры

}

float time = clock.getElapsedTime().asMicroseconds();

clock.restart();

time = time / 800;

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

if (event.type == Event::KeyPressed)//событие нажатия клавиши

if ((event.key.code == Keyboard::Tab)) {//если клавиша ТАБ

switch (showMissionText) {//переключатель, реагирующий на логическую переменную showMissionText

case true: {

std::ostringstream playerHealthString;//строка здоровья игрока

playerHealthString << p.health; //заносим в строку здоровье

std::ostringstream task;//строка текста миссии

task << getTextMission(getCurrentMission(p.getplayercoordinateX()));//вызывается функция getTextMission (она возвращает текст миссии), которая принимает в качестве аргумента функцию getCurrentMission(возвращающую номер миссии), а уже эта ф-ция принимает в качестве аргумента функцию p.getplayercoordinateX() (эта ф-ция возвращает Икс координату игрока)

text.setString("Здоровье: " + playerHealthString.str() + "\n" + task.str());//задаем

showMissionText = false;//эта строка позволяет убрать все что мы вывели на экране

break;//выходим , чтобы не выполнить условие "false" (которое ниже)

}

case false: {

text.setString("");//если не нажата клавиша таб, то весь этот текст пустой

showMissionText = true;// а эта строка позволяет снова нажать клавишу таб и получить вывод на экран

break;

}

}

}

}

///////////////////////////////////////////Управление персонажем с анимацией////////////////////////////////////////////////////////////////////////

if (p.life) {

if ((Keyboard::isKeyPressed(Keyboard::Left) || (Keyboard::isKeyPressed(Keyboard::A)))) {

p.dir = 1; p.speed = 0.07;//dir =1 - направление вверх, speed =0.1 - скорость движения. Заметьте - время мы уже здесь ни на что не умножаем и нигде не используем каждый раз

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

p.sprite.setTextureRect(IntRect(34 \* int(CurrentFrame), 33, 32, 32)); //через объект p класса player меняем спрайт, делая анимацию (используя оператор точку)

}

if ((Keyboard::isKeyPressed(Keyboard::Right) || (Keyboard::isKeyPressed(Keyboard::D)))) {

p.dir = 0; p.speed = 0.07;//направление вправо, см выше

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

p.sprite.setTextureRect(IntRect(34 \* int(CurrentFrame), 1, 32, 32)); //через объект p класса player меняем спрайт, делая анимацию (используя оператор точку)

}

if ((Keyboard::isKeyPressed(Keyboard::Up) || (Keyboard::isKeyPressed(Keyboard::W)))) {

p.dir = 3; p.speed = 0.07;//направление вниз, см выше

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

p.sprite.setTextureRect(IntRect(34 \* int(CurrentFrame), 66, 32, 32)); //через объект p класса player меняем спрайт, делая анимацию (используя оператор точку)

}

if ((Keyboard::isKeyPressed(Keyboard::Down) || (Keyboard::isKeyPressed(Keyboard::S)))) { //если нажата клавиша стрелка влево или англ буква А

p.dir = 2; p.speed = 0.07;//направление вверх, см выше

CurrentFrame += 0.005\*time; //служит для прохождения по "кадрам". переменная доходит до трех суммируя произведение времени и скорости. изменив 0.005 можно изменить скорость анимации

if (CurrentFrame > 3) CurrentFrame -= 3; //проходимся по кадрам с первого по третий включительно. если пришли к третьему кадру - откидываемся назад.

p.sprite.setTextureRect(IntRect(34 \* int(CurrentFrame), 100, 32, 32)); //проходимся по координатам Х. получается 96,96\*2,96\*3 и опять 96

}

}

p.update(time);//оживляем объект p класса Player с помощью времени sfml, передавая время в качестве параметра функции update. благодаря этому персонаж может двигаться

//viewmap(time);//функция скроллинга карты, передаем ей время sfml

//changeview();//прикалываемся с камерой

//window.setView(view);//"оживляем" камеру в окне sfml (не очень корректно работает(-;-))

window.clear();

/////////////////////////////Рисуем карту/////////////////////

for (int i = 0; i < HEIGHT\_MAP; i++)

for (int j = 0; j < WIDTH\_MAP; j++)

{

if (TileMap[i][j] == ' ') s\_map.setTextureRect(IntRect(0, 0, 32, 32)); //если встретили символ пробел, то рисуем 1й квадратик

if (TileMap[i][j] == 's') s\_map.setTextureRect(IntRect(32, 0, 32, 32));//если встретили символ s, то рисуем 2й квадратик

if ((TileMap[i][j] == '0')) s\_map.setTextureRect(IntRect(64, 0,32, 32));//если встретили символ 0, то рисуем 3й квадратик

if (TileMap[i][j] == 'b') s\_map.setTextureRect(IntRect(97, 0, 32, 32)); //если встретили символ пробел, то рисуем 1й квадратик

s\_map.setPosition(j \* 32, i \* 32);//по сути раскидывает квадратики, превращая в карту. то есть задает каждому из них позицию. если убрать, то вся карта нарисуется в одном квадрате 32\*32 и мы увидим один квадрат

window.draw(s\_map);//рисуем квадратики на экран

}

/\*std::ostringstream playerScoreString; // объявили переменную

playerScoreString << p.playerScore; //занесли в нее число очков, то есть формируем строку

text.setString("Score:" + playerScoreString.str());//задаем строку тексту и вызываем сформированную выше строку методом .str()

text.setPosition(265, 32);//задаем позицию текста, отступая от центра камеры

\*/

if (!showMissionText) {

text.setPosition(view.getCenter().x + -70, view.getCenter().y - 150);//позиция всего этого текстового блока

s\_quest.setPosition(view.getCenter().x + -105, view.getCenter().y - 200);//позиция фона для блока

window.draw(s\_quest); window.draw(text); //рисуем спрайт свитка (фон для текста миссии). а затем и текст. все это завязано на логическую переменную, которая меняет свое состояние от нажатия клавиши ТАБ

}

window.draw(p.sprite); //рисуем спрайт объекта p класса player

window.display();

/\*

std::ostringstream playerHealthString; // объявили переменную

playerHealthString << p.health; //занесли в нее число здоровья, то есть формируем строку

text.setString("Здоровье:" + playerHealthString.str());//задаем строку тексту и вызываем сформированную выше строку методом .str()

text.setPosition(240,32);//задаем позицию текста, отступая от центра камеры

window.draw(text);//рисую этот текст

\*/

}

return 0;

}

Исходный код игры Pacman в файле map.h

#pragma once

#include <SFML\Graphics.hpp>

const int HEIGHT\_MAP = 15;//размер карты высота

const int WIDTH\_MAP = 20;//размер карты ширина

//0- элементы стены

//b- пустота

//все остальное монеты

sf::String TileMap[HEIGHT\_MAP] = {

"00000000000000000000",

"0 0bbbb0 0",

"0 0000 0bbbb0 0000 0",

"0 0bbbb0 0",

"00 000 000000 000 00",

"b0 0 0 0b",

"b0 0 0 00bb00 0 0 0b",

"b0 0 0 0bbbb0 0 0 0b",

"b0 0 0bbbb0 0 0b",

"00 000 000000 000 00",

"0 bbbbbb 0",

"0 000 00000000 000 0",

"0 00 0 0 00 0",

"00 0000 00",

"00000000000000000000",

};

Исходный код игры Pacman в файле view.h

#pragma once

#include <SFML/Graphics.hpp>

using namespace sf;

sf::View view;//объ¤вили sfml объект "вид", который и ¤вл¤етс¤ камерой

void viewmap(float time) { //функци¤ дл¤ перемещени¤ камеры по карте. принимает врем¤ sfml

if (Keyboard::isKeyPressed(Keyboard::D)) {

view.move(0.1\*time, 0);//скроллим карту вправо (см урок, когда мы двигали геро¤ - всЄ тоже самое)

}

if (Keyboard::isKeyPressed(Keyboard::S)) {

view.move(0, 0.1\*time);//скроллим карту вниз (см урок, когда мы двигали геро¤ - всЄ тоже самое)

}

if (Keyboard::isKeyPressed(Keyboard::A)) {

view.move(-0.1\*time, 0);//скроллим карту влево (см урок, когда мы двигали геро¤ - всЄ тоже самое)

}

if (Keyboard::isKeyPressed(Keyboard::W)) {

view.move(0, -0.1\*time);//скроллим карту вправо (см урок, когда мы двигали геро¤ - всЄ тоже самое)

}

}

# ПРИЛОЖЕНИЕ В

**Протокол системы контроля версий**

Протокол системы контроля версий

|  |  |
| --- | --- |
| **Summary** | **Description** |
| **Commits on Dec 16, 2016** | |
| Создание папок |  |
| Структурирование данных |  |
| [Анализ](https://github.com/pmswga/EDUKIT/commit/d4eb141e29f2f4f9acdb5ef78a527707f099c111) пертметной области |  |
| Docum |  |
| **Commits on Jan 17, 2017** | |
| Programm |  |
| **Commits on Sep 20, 2017** | |
| Fix programm |  |
| Smth things |  |
| [image](https://github.com/pmswga/EDUKIT/commit/62326a771caa52359d3d0979b5bd04f877e64ca9) complete |  |
| **Commits on Sep 17, 2017** | |
| small fix |  |
| fix |  |
| [write documents](https://github.com/pmswga/EDUKIT/commit/e2bff22526e13604c0c918749b7de5d95a1be452) |  |
| **Commits on Sep 16, 2017** | |
| Something |  |
| [smth](https://github.com/pmswga/EDUKIT/commit/916e3e2dd8477ebc5f4d03294e0af111b6f96c2e) |  |
| **Commits on Dec 10, 2017** | |
| documentation |  |
| **Commits on Dec 20, 2017** | |
| Doc fix |  |
| [doc](https://github.com/pmswga/EDUKIT/commit/c7a8a0beb93b5e6a1e388febe18e03e41eae6034) v5 complete may be yeah |  |

# ПРИЛОЖЕНИЕ Г

**Руководство программиста**

Руководство содержит следующие основные разделы:

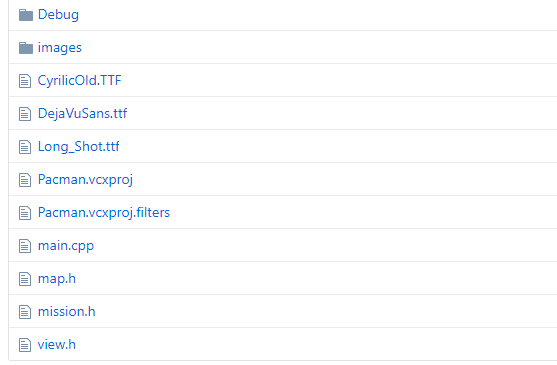


Рисунок 1 – Файлы

В данном разделе приводиться иерархия файлов.

# ПРИЛОЖЕНИЕ Д

**Доклад**

1. Доклад

### Введение

Для людей, которые хотят потренировать свои зоны мозга, отвечающие за логику, внимания и быстрого принятия решений в целом.

### Основная часть

Основные задачи, решаемые игрой Pacman:

* Тренировка частей мозга, отвечающие за логику, внимания и быстрого принятия решений в целом.

Основные достоинства:

1. Бесплатная;
2. Не ресурсозатратный;

Основные недостатки:

1. Не имеет кроссплатформенность
2. Любительская разработка.

### Заключение

Результаты моей работы заключается в изучении разработки игры Pacman, предназначенных для тренировки частей мозга, отвечающие за логику, внимания и быстрого принятия решений в целом.

Разработанная система предназначена только для внутреннего использования пользователем ПК и не претендует на дальнейшее развитие как универсальной игры, которая была бы способна конкурировать с аналогами, такими как другие учебно-развлекательные игры.