|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное автономное образовательное учреждение высшего образования  «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» |

Кафедра информационных систем и технологий (№42)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ОТЧЁТ ПО ПРАКТИКЕ  ЗАЩИЩЁН С ОЦЕНКОЙ  Руководитель |  | | | |
| старший преподаватель |  |  |  | Гуков С.Ю. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | учебная | |
| тип практики |  | |
| на тему индивидуального задания | |  |
| Реализация компьютерной игры «Pacman» | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен | Матвеевым Дмитрием Сергеевичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.02 |  | Информационные системы и технологии |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 09.03.02 |  | Информационные технологии в дизайне |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4427 |  | 07.06.2025 |  | Матвеев Д. С. |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2025

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

09.03.02 Информационные системы и технологии

на прохождение *учебной* практики обучающегося направления подготовки/ специальности

Матвеев Дмитрий Сергеевич

1. Фамилия, имя, отчество обучающегося:

4427

1. Группа:
2. Тема индивидуального задания:

Реализация компьютерной игры «Pacman»

Visual Studio

1. Исходные данные:

1. Содержание отчетной документации:
   1. индивидуальное задание;
   2. отчёт, включающий в себя:
   * титульный лист;
   * материалы о выполнении индивидуального задания (содержание определяется кафедрой);
   * выводы по результатам практики;
   * список использованных источников.
   1. отзыв руководителя от профильной организации (при прохождении практики в профильной организации).
2. Срок представления отчета на кафедру: «07» июня 2025 г.

старший преподаватель

07.06.2025

Гуков С.Ю.

Руководитель практики

должность, уч. степень, звание подпись, дата инициалы, фамилия

СОГЛАСОВАНО

Руководитель практики от профильной организации

должность подпись, дата инициалы, фамилия

Задание принял к исполнению:

Матвеев Д. С.

10.02.2025

Обучающийся

дата подпись инициалы, фамилия

Санкт–Петербург 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc136043254)

[1 Основные положения и теоретические аспекты разработки 6](#_Toc136043255)

[1.1 Описание игры 6](#_Toc136043256)

[1.2 Пользовательская документация 11](#_Toc136043257)

[1.3 Техническая документация 13](#_Toc136043258)

[2 Практические аспекты разработки консольной игры 18](#_Toc136043259)

[2.1 Загрузка на Github 18](#_Toc136043260)

[2.2 Пробный запуск, проверка на выявление ошибок кода 21](#_Toc136043261)

[ЗАКЛЮЧЕНИЕ 24](#_Toc136043262)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc136043263)

[ПРИЛОЖЕНИЕ 27](#_Toc136043264)

# ВВЕДЕНИЕ

Учебная практика является неотъемлемой частью образовательного процесса студентов высших учебных заведений. Она представляет собой специально организованное обучение, которое проводится в условиях реального профессионального окружения. Цель учебной практики заключается в том, чтобы помочь студентам получить практические навыки и опыт работы в своей будущей профессии.

В рамках учебной практики студенты могут применять теоретические знания, полученные на лекциях и семинарах, на практике. Они могут ознакомиться с основными процессами и методиками работы в своей сфере деятельности, а также научиться работать в команде и эффективно взаимодействовать с коллегами.

Реализуя консольную игру в рамках учебной практики у студентов есть возможность применить свои знания и навыки программирования на практике. Во время создания своей игры, были изучены новые библиотеки, такие как SFML и time.h, и смогла применить полученные в ходе обучения навыки для реализации цели. Было изучено создание и использование текстуры и анимации в своем коде. Это позволило мне реализовать визуальную составляющую игры, что благополучно сказывается на вовлечение игрока в игровой процесс.

В игре пользователь с помощью клавиатуры управляет персонажем – Пакменом. Его задача – перемещаться по лабиринту, собирая по пути куриные ножки. Успешным завершением игры считается сбор всех куриных ножек.

Однако в лабиринте встречаются враги-призраки, столкнувшись с которыми Пакмен сам превращается в призрака, а игрок проигрывает.

Также в игре со всех четырех сторон карты присутствуют коридоры-телепорты. Попадая в них, игрок перемещается на противоположную сторону карты, что облегчает процесс сбора кексов и дает возможность сбежать от врагов.

Данные условия позволяют создать интересную и увлекательную игру, в которой игрок, чтобы собрать все куриные ножки и одержать победу, и при этом не попасться врагам, должен сосредоточиться, применить свои навыки и внимательность.

Для реализации консольной игры был выбран язык программирования C++, так как он изучался в течение всего семестра и появилось предположение, что можно воплотить свою идею на нем. В итоге программирование на C++ позволило мне полностью реализовать свою задумку и создать интересную консольную игру с приятным дизайном.

# 1 Основные положения и теоретические аспекты разработки

## 1.1 Описание игры

Основной геймплей игры заключается в том, чтобы перемещаться по лабиринту и собирать пирожные. При этом игроку следует избегать врагов, иначе при столкновении с ними игра закончится поражением.

В игре присутствуют следующие объекты:

Блоки стен – это объекты, создающие границы лабиринта. Через них ни игрок, ни враги передвигаться не могут. При столкновении со стеной Пакмен будет идти на месте, пока игрок не задаст другое направление. (Рисунок 1).

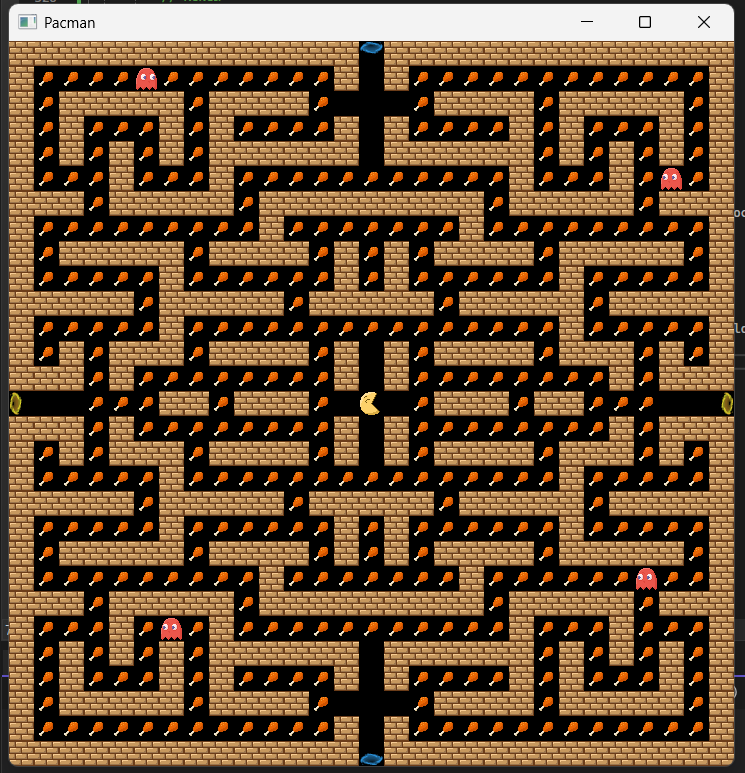


Рисунок 1 — Стены лабиринта

Для создания стен была использована следующая текстура. (Рисунок 2).



Рисунок 2 — Текстура стены

Среди стен лабиринта игроку могут встретиться враги – призраки, движущиеся в рандомном направлении. Игроку следует их избегать. (Рисунок 3).

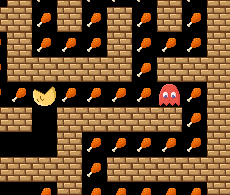


Рисунок 3 —Враги – призраки

Для реализации внешнего вида и анимации призраков, были использованы следующие текстуры. (Рисунок 4).

Enemy

Рисунок 4 — Текстуры призраков

Сам игрок управляет персонажем - Пакменом. Он движется по лабиринту и собирает по пути куриные ножки. (Рисунок 5).

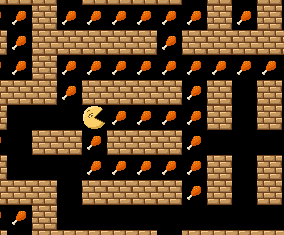


Рисунок 5 — Персонаж – Пакмен собирает куриные ножки

Для реализации внешнего вида и анимации Пакмена, были использованы следующие текстуры. (Рисунок 6).



Рисунок 6 — Текстуры Пакмена

Для реализации внешнего вида куриных ножек, были использованы следующие текстуры. (Рисунок 7).



Рисунок 7 — Текстуры куриных ножек

Также в игре со всех четырех сторон карты присутствуют коридоры-телепорты. Попадая в них, игрок перемещается на противоположную сторону карты, что облегчает процесс сбора куриных ножек и дает возможность сбежать от врагов. (Рисунок 8).

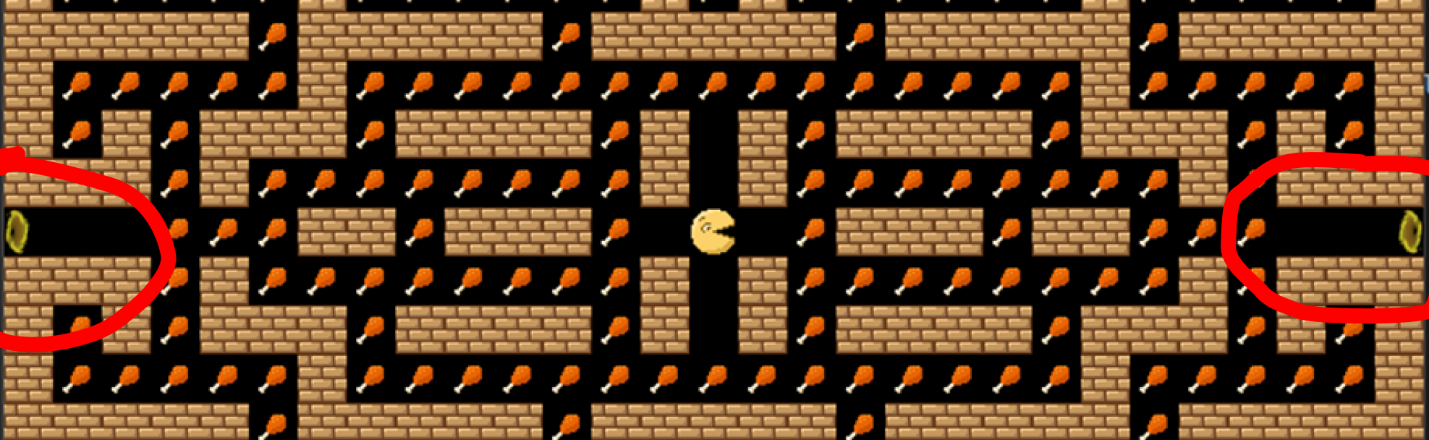


Рисунок 8 — Коридоры-телепорты

Когда игрок соберет все куриные ножки, игра завершится и появится надпись, символизирующая победу. (Рисунок 9).

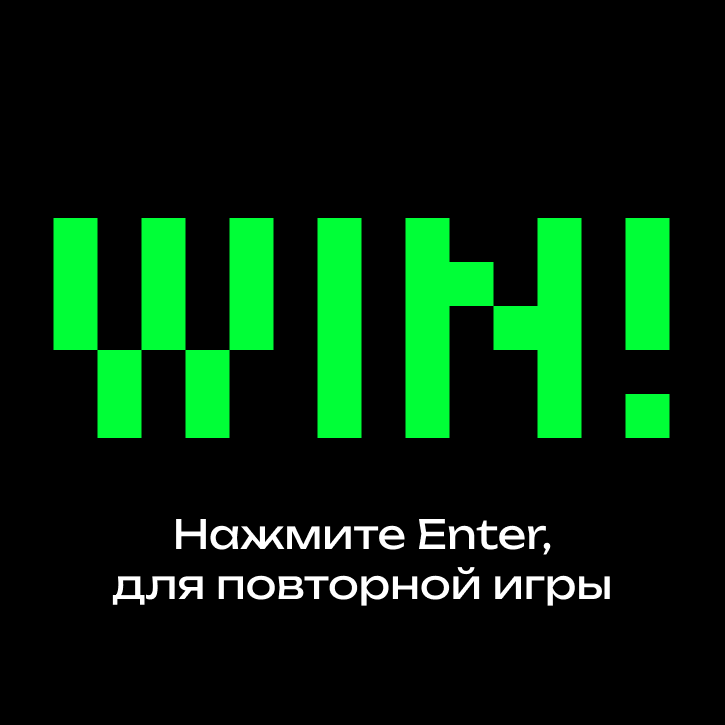


Рисунок 9 — Победа

Если игрок сталкивается с врагом, Пакмен превращается в призрака, игра заканчивается поражением и появляется соответствующая надпись. (Рисунок 10)

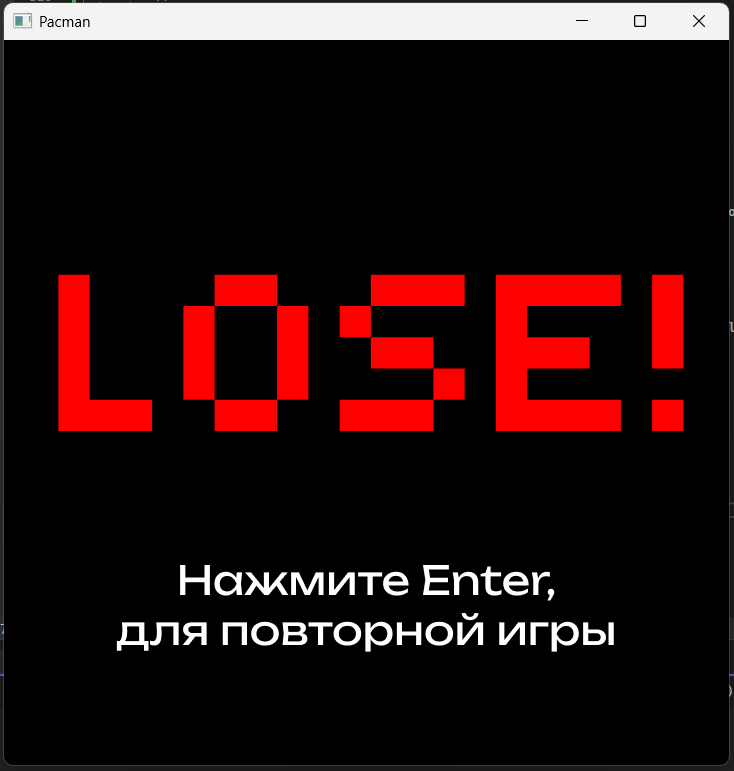


Рисунок 10 — Поражение

В целом, игра представляет собой аркаду, которая требует от игрока внимательности и сосредоточенности.

## 1.2 Пользовательская документация

Чтобы запустить игру, вам необходимо открыть проект в Visual Studio и запустить его, нажав на соответствующую клавишу или сочетание клавиш Ctrl + F5. После запуска игры появляется окно с лабиринтом. Для начала игры вы должны нажать на любую кнопку, приводящую персонажа в движение по направлению отсутствия препятствий (стен).

Управление персонажем осуществляется двумя способами:

1. Перемещение вашего персонажа осуществляется с помощью клавиш WASD, которые можно рассматривать как аналог крестовины на геймпаде. Клавиши W, A, S и D - передвижение вверх, влево, вниз и вправо соответственно.
2. Также вы можете использовать клавиши стрелок на вашей клавиатуре: «↑», «←», «↓» и «→» для передвижения вверх, влево, вниз и вправо соответственно. Используйте их для перемещения внутри лабиринта.
3. Клавиша ESC служит для выхода из игры. Если вам нужно быстро покинуть игру, просто нажмите клавишу ESC.

Это основные команды и управление в игре.

Ваша задача – перемещаться по лабиринту и собирать куриные ножки. При этом старайтесь избегать призраков. Для более удобного передвижения вы можете пользоваться телепортами со всех четырех сторон карты. Вы одержите победу, если сможете собрать все куриные ножки и не попасться призракам. В ином случае игра будет проиграна. Будьте внимательны и наслаждайтесь игровым процессом. Удачи!

## 1.3 Техническая документация

Разработка игры осуществлялась на основе ниже представленных блок-схем, рисунок 11 соответственно. Описание структур и функций представлено таблицах 1 и 2, соответственно.

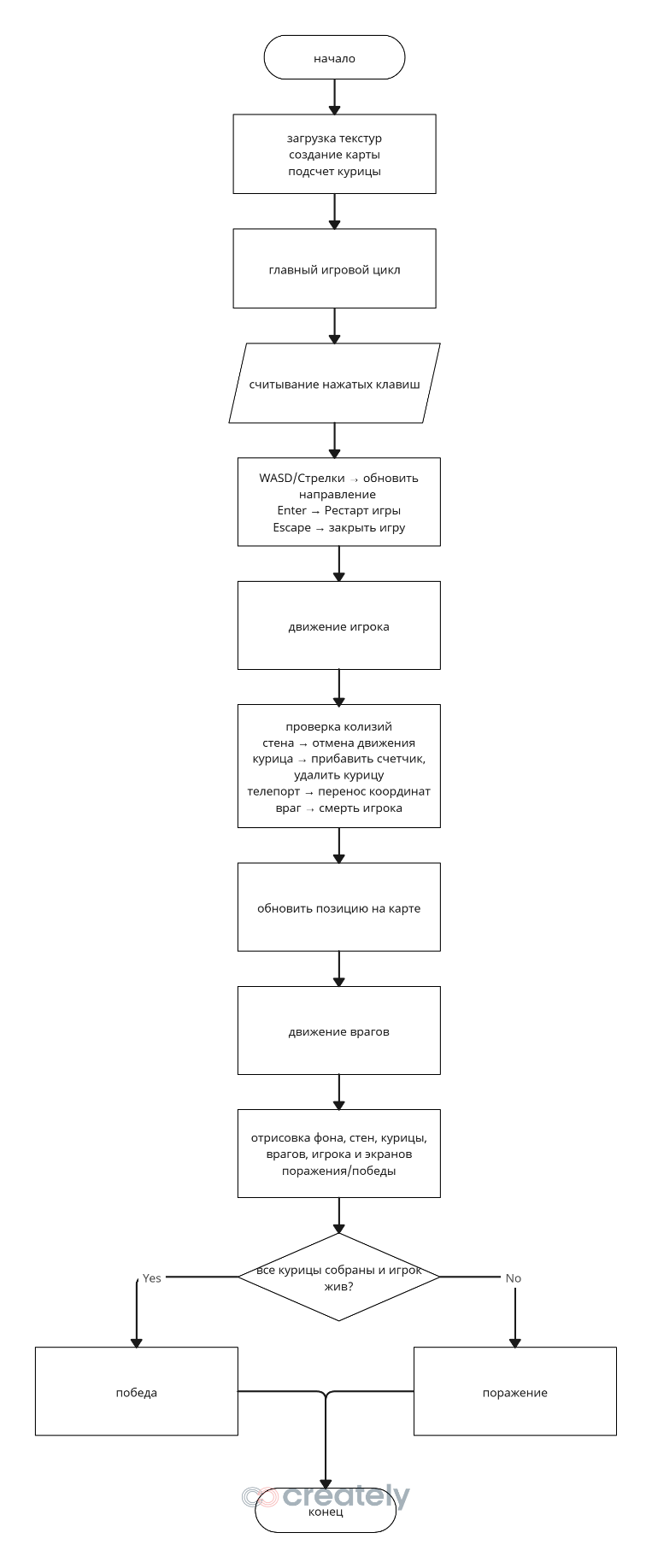


Рисунок 11 — Блок-схема основной функции

Таблица 1 — Описание переменных

|  |  |
| --- | --- |
| Тип хранимых данных | Компонент и его описание |
| const int | higth  Высота игрового поля (карты) |
| const int | width  Ширина игрового поля (карты) |
| const int | block  Размер одного блока (объекта) в пикселях |
| int | chicken\_num  Количество собранных кусочков курицы |
| int | total\_chicken  Общее количество кусочков курицы на карте |
| bool | life  Состояние жизни игрока (true — жив, false — проиграл) |
| bool | game\_over  Флаг окончания игры (поражение) |
| bool | Win  Флаг победы в игре |
| String[] | Map  Массив строк, представляющих карту лабиринта |
| float | frame  Кадры анимации Пакмена |
| int | x  Координаты игрока по x |
| int | y  Координаты игрока по y |
| int | changex  Измененные координаты игрока по x |
| int | changey  Измененные координаты игрока по y |
| int | rotate  Поворот игрока |
| int | del  Задержка |
| int | x[4]  Координаты врагов по x |
| int | y[4]  Координаты врагов по y |
| int | changex[4]  Измененные координаты врагов по x |
| int | changey[4]  Измененные координаты врагов по y |
| int | rotate[4]  Поворот врагов |
| Texture | w, pac, c, e  Текстуры для стен, игрока, курицы и врагов соответственно |
| Texture | win\_t, lose\_t  Текстуры для экранов победы и поражения |
| Texture | teleport\_\*  Текстуры для телепортов (левого, правого, верхнего, нижнего) |
| Sprite | wall, pacman  Спрайты для стен и игрока |
| Sprite | chicken, enemy  Спрайты для курицы и врагов |
| Sprite | win, lose  Спрайты для экранов победы и поражения |
| Sprite | \*\_portal  Спрайты для телепортов (левого, правого, верхнего, нижнего) |

Таблица 2 — Описание функций

|  |  |
| --- | --- |
| Название метода и тип возвращаемого значения | Описание |
| void Player::reset() | Сбрасывает состояние игрока в начальное положение (координаты, анимацию, направление). |
| void Player::changeP() | Обновляет состояние игрока: перемещение, анимацию, проверку столкновений, сбор курицы, телепортацию. |
| void Enemy::reset() | Сбрасывает состояние всех врагов в начальные позиции. |
| void Enemy::changeE() | Обновляет состояние врагов: случайное перемещение, проверку столкновений, телепортацию. |
| void countTotalChicken() | Подсчитывает общее количество кусочков курицы на карте. |
| void resetGame() | Полностью перезагружает игру: сбрасывает карту, счётчик курицы, состояние игрока и врагов. |
| int main() | Главная функция игры: инициализация окна, загрузка текстур, обработка событий, игровой цикл. |
| bool Texture::loadFromFile(...) | Загружает текстуру из файла |
| void Sprite::setPosition(float x, float y) | Устанавливает позицию спрайта на экране. |
| bool RenderWindow::isOpen() | Проверяет, открыто ли окно. |
| bool RenderWindow::pollEvent(Event &event) | Обрабатывает события (нажатия клавиш, закрытие окна и т. д.). |
| void RenderWindow::close() | Закрывает окно. |
| void RenderWindow::clear(const Color &color) | Очищает окно заданным цветом (по умолчанию — чёрный). |
| void Sprite::setTextureRect(const IntRect &rect) | Устанавливает область текстуры для отображения (кадры анимации). |
| void RenderWindow::draw(const Drawable &drawable) | Отрисовывает объект (спрайт, текст и т. д.) в окне. |
| void RenderWindow::display() | Отображает всё нарисованное в окне. |

# 2 Практические аспекты разработки консольной игры

## 2.1 Загрузка на Github

После завершения разработки игры в рамках учебной практики, было принятло решение загрузить её на платформу GitHub - это веб-сервис для хранения, управления и совместной работы над проектами с использованием системы контроля версий Git. Он предоставляет возможность разработчикам загружать свой код на хостинг-платформу, где он может быть сохранен, отслеживаться и управляться через веб-интерфейс.

GitHub является одним из самых популярных сервисов для хостинга проектов с открытым исходным кодом. Он также используется для хранения и управления закрытыми проектами, которые разрабатываются командами разработчиков и других IT-специалистов.

GitHub предоставляет множество инструментов для совместной работы над проектами, таких как возможность создания задач, обсуждение кода, управление проектами, ревью кода, автоматические тесты, контроль версий и многое другое. Это делает GitHub одним из самых эффективных инструментов для разработки программного обеспечения в команде, а также для открытых проектов с большим количеством участников.

Чтобы выполнить эту задачу, был создан новый репозиторий на GitHub, который стал центральным хранилищем для моего проекта. В файл code был загружен код игры. (Рисунок 12).

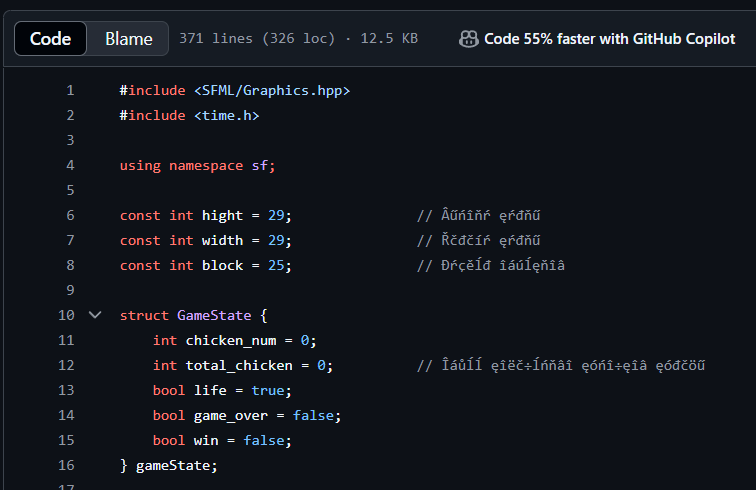


Рисунок 12 – Код программы

Кроме того, был создан файл README, который содержал информацию о игре, ее описании, инструкции по установке, запуску и управлению. (Рисунок 14). README файл является важным компонентом проекта, так как он предоставляет важную информацию для других разработчиков или заинтересованных лиц, которые посещают репозиторий на GitHub. Файл был оформлен аккуратно и информативно, чтобы обеспечить понятность и удобство использования для всех, кто заинтересован в проекте.

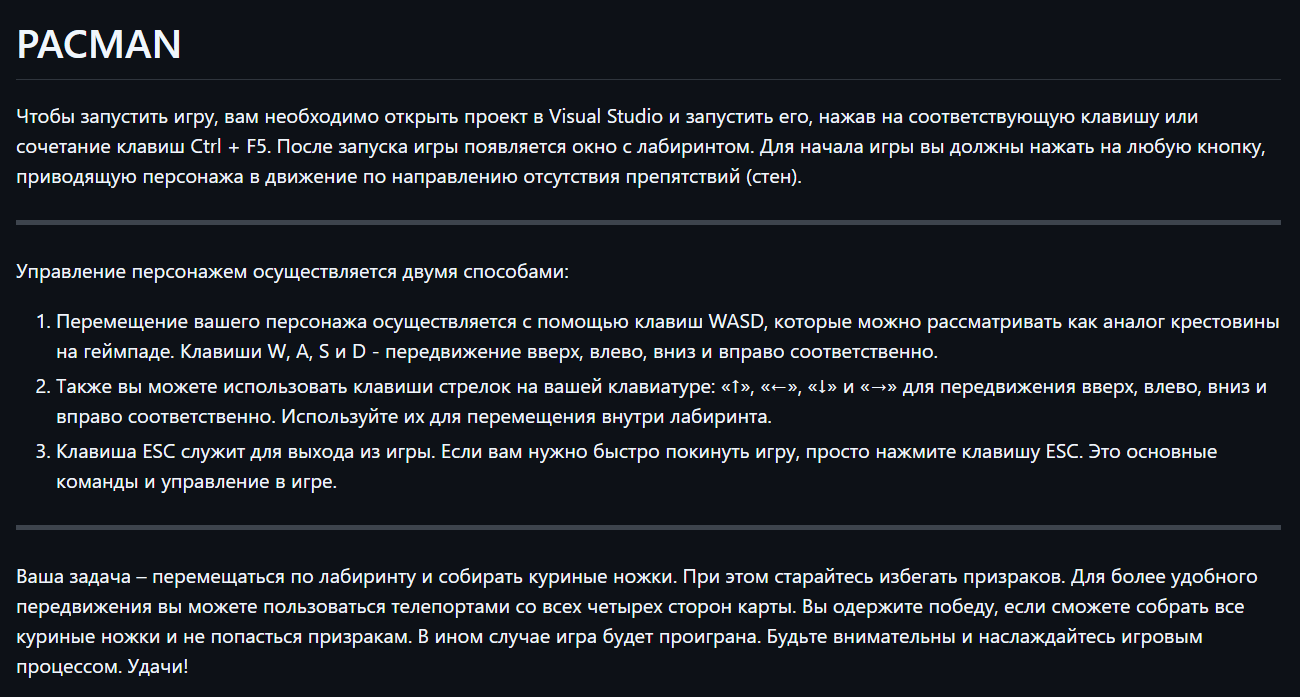


Рисунок 13 – Файл README

Также в файле представлены скриншоты игры. (Рисунок 14).



Рисунок 14 – Скриншоты игры в файле

В результате, проект был успешно загружен на GitHub и стал доступен на моей странице GitHub. Теперь люди могут просмотреть его, изучить код и, при желании, внести свои предложения и улучшения.

Ссылка на репозиторий: <https://github.com/techiedi/Pac.git>

Загрузка учебного проекта на GitHub способствовала более эффективной разработке игры и в будущем облегчит сопровождение и развитие проекта. Она предоставила возможность централизованного хранения и совместного ведения разработки, а также упростила процесс общения и обратной связи для заинтересованных лиц.

Также было принято решение загрузить видео с геймплеем игры на платформу Rutube. Rutube является популярным сервисом для хостинга видеороликов.

Сслыка на видео: <https://rutube.ru/video/private/2884a1097bd12299b136cce74b74499a/?p=VReDGAyTZjK6Lk4juVE9jw>

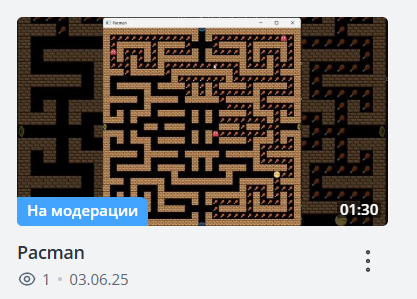


Рисунок 15 - Видео

## 2.2 Пробный запуск, проверка на выявление ошибок кода

Тестирование – это процесс проверки программного обеспечения с целью выявления ошибок, дефектов и недостатков, а также оценки его соответствия заявленным требованиям и ожиданиям пользователей. Оно является важной частью жизненного цикла разработки программного обеспечения и выполняется для обеспечения качества и надежности программы.

Главная цель тестирования - обнаружение ошибок и проблем в программном обеспечении до его выпуска в эксплуатацию. Тестирование помогает выявить дефекты, которые могут привести к некорректной работе программы, потере данных или негативному пользовательскому опыту. Кроме того, тестирование позволяет убедиться в соответствии программы заявленным требованиям и ожиданиям пользователей, а также проверить ее функциональность, производительность, безопасность и другие аспекты качества.

В ходе тестирования мной было обнаружено и исправлено несколько ошибок. Тестирование игры проводилось визуально и с помощью инструментов среды разработки Visual Studio.

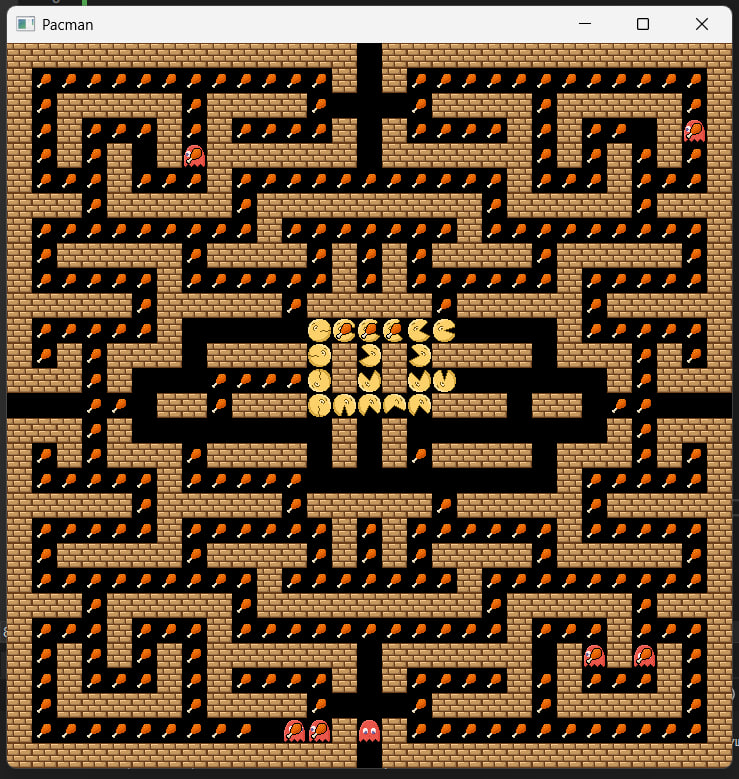


Рисунок 16 — Выявление бага

После нахождения предполагаемого решения, код программы редактируется и решение снова тестируется сначала визуально, потом, в случае повторения неисправности или появления новой, процесс тестирования повторяется снова, пока решение не будет найдено.

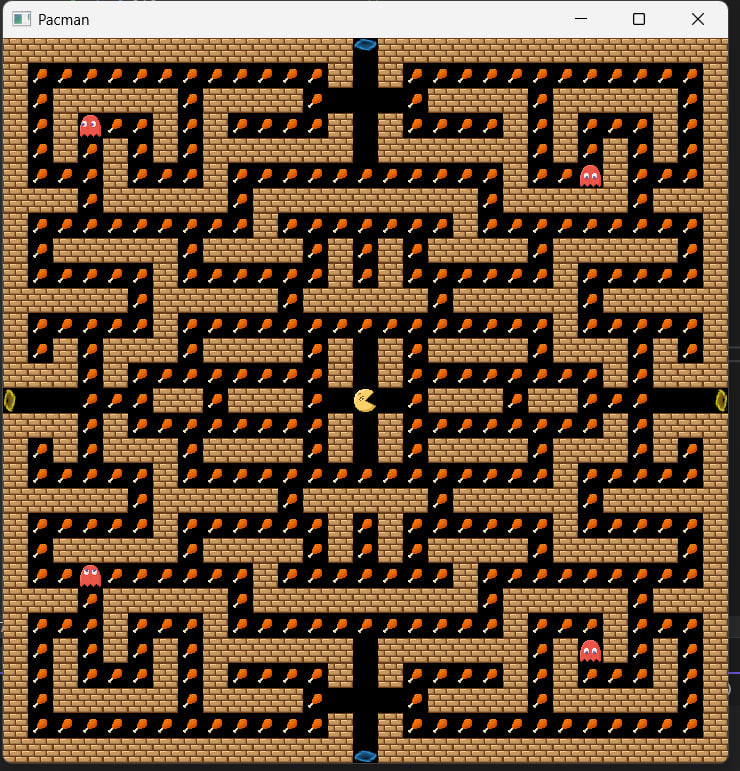


Рисунок 17 — Исправление бага

# ЗАКЛЮЧЕНИЕ

В рамках проведения учебной практики мной была успешно разработана консольная игра, использующая язык программирования C++. Процесс работы над данным проектом позволил мне углубить свои знания в области структур языка C++, в том числе в работе со строками, массивами, условиями и циклами.

Во время создания своей игры, были изучены новые библиотеки, такими как SFML и time.h, и смогла применить полученные в ходе обучения навыки для реализации цели. Было изучено создание и использование текстуры и анимации в своем коде. Благодаря этому опыту, значительно улучшились мои навыки программирования и разработки программного обеспечения.

Однако, данный проект лишь первый шаг на пути к дальнейшей разработке и улучшению. В будущем планируется внедрить новые механики и функции, которые сделают игру еще более увлекательной и интересной для пользователей. Например, возможно добавление новых уровней с различными картами, новые типы врагов, а также таймер и возможность записывать рекордное время игры. Это позволит повысить вовлеченность в игру.

В целом, учебная практика по разработке консольной игры с использованием языка программирования C++ была очень полезным и продуктивным опытом для меня. Были закреплены и расширены знания в программировании, а также они применялись на практике, создав интересный игровой проект. Дальнейшее развитие и улучшение данной игры позволит мне продолжать совершенствоваться в области программирования и разработки программного обеспечения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Введение в SFML — URL:

<https://habr.com/ru/articles/149071/> (дата посещения 05.04.2025)

1. Бесплатный курс по C++ для начинающих – URL:

<https://code-basics.com/ru/languages/cpp> (дата посещения 08.03.2025)

1. Как написать игру на C++ — URL:

<https://habr.com/ru/articles/537682/> (дата посещения 15.03.2025)

1. Documentation of SFML 2.5.1 – URL:

<https://www.sfml-dev.org/documentation/2.5.1/classsf_1_1Sprite.php#a3fefec419a4e6a90c0fd54c793d82ec2> (дата посещения 17.04.2025)

1. Documentation of SFML 2.4.2 – URL:

<https://www.sfml-dev.org/documentation/2.4.2/classsf_1_1RenderTarget.php#a6bb6f0ba348f2b1e2f46114aeaf60f26> (дата посещения 18.04.2025)

1. Видеоуроки по созданию игры C++ – URL:

<https://tproger.ru/video/how-to-program-a-game-in-cpp/> (дата посещения 23.04.2025)

1. Разработка игр на C++ – URL:

<https://learn.microsoft.com/ru-ru/cpp/overview/game-development-cpp?view=msvc-170> (дата посещения 19.04.2025)

# ПРИЛОЖЕНИЕ

Исходный код программы

#include <SFML/Graphics.hpp>

#include <time.h>

**using** **namespace** sf;

**const** **int** hight = **29**; // Высота карты

**const** **int** width = **29**; // Ширина карты

**const** **int** block = **25**; // Размер объектов

**struct** GameState {

**int** chicken\_num = **0**;

**int** total\_chicken = **0**; // Общее количество кусочков курицы

**bool** life = true;

**bool** game\_over = false;

**bool** win = false;

} gameState;

String Map[hight] = {

"WWWWWWWWWWWWWWVWWWWWWWWWWWWWW",

"W WVW W",

"W WWWWW WWWW VVV WWWW WWWWW W",

"W W W W WVW W W W W",

"W W W W WWWWWWVWWWWWW W W W W",

"W W W W W W",

"WWW WWWWW WWWWWWWWW WWWWW WWW",

"W W W W",

"W WWWWW WWWW W W WWWW WWWWW W",

"W W W W W W",

"WWWWW WWWWW WWWWW WWWWW WWWWW",

"W W W W",

"W W WWW WWWW WVW WWWW WWW W W",

"WWW W WVW W WWW",

"VVV WW WWW VPV WWW WW VVV",

"WWW W WVW W WWW",

"W W WWW WWWW WVW WWWW WWW W W",

"W W W W",

"WWWWW WWWWW WWWWW WWWWW WWWWW",

"W W W W W W",

"W WWWWW WWWW W W WWWW WWWWW W",

"W W W W",

"WWW WWWWW WWWWWWWWW WWWWW WWW",

"W W W W W W",

"W W W W WWWWWWVWWWWWW W W W W",

"W W W W WVW W W W W",

"W WWWWW WWWW VVV WWWW WWWWW W",

"W WVW W",

"WWWWWWWWWWWWWWVWWWWWWWWWWWWWW"

};

**class** **Player** {

**public:**

**float** frame = **0**;

**int** x = **14**, y = **14**;

**int** changex = **0**, changey = **0**;

**int** rotate = **0**, del = **0**;

**void** **reset**() {

frame = **0**;

x = **14**; y = **14**;

changex = **0**; changey = **0**;

rotate = **0**; del = **0**;

}

**void** **changeP**() {

frame += **0.03**;

**if** (frame > **6**) frame -= **6**;

del++;

**if** (del >= **100**) {

**switch** (rotate) {

**case** **0**: **if** (Map[y][changex + **1**] != 'W') changex += **1**; **break**;

**case** **1**: **if** (Map[y][changex - **1**] != 'W') changex -= **1**; **break**;

**case** **2**: **if** (Map[changey - **1**][x] != 'W') changey -= **1**; **break**;

**case** **3**: **if** (Map[changey + **1**][x] != 'W') changey += **1**; **break**;

}

del = **0**;

}

**if** (Map[changey][changex] == ' ' || Map[changey][changex] == 'V') {

**if** (Map[changey][changex] == ' ') {

gameState.chicken\_num++;

**if** (gameState.chicken\_num >= gameState.total\_chicken) {

gameState.win = true;

**return**; // Добавлено для немедленного выхода при победе

}

}

**if** (Map[changey][changex] == '1' || Map[changey][changex] == '2' ||

Map[changey][changex] == '3' || Map[changey][changex] == '4') {

gameState.life = false;

gameState.game\_over = true;

**return**;

}

Map[y][x] = 'V';

Map[changey][changex] = 'P';

x = changex;

y = changey;

}

// Горизонтальный телепорт

**if** (changey == **14** && (changex == **0** || changex == **28**)) {

changex = (changex == **0**) ? **27** : **1**;

Map[y][x] = 'V';

Map[changey][changex] = 'P';

x = changex;

y = changey;

}

// Вертикальный телепорт

**if** (changex == **14** && (changey == **0** || changey == **28**)) {

changey = (changey == **0**) ? **27** : **1**;

Map[y][x] = 'V';

Map[changey][changex] = 'P';

x = changex;

y = changey;

}

}

};

**class** **Enemy** {

**public:**

**int** x[**4**] = { **5**, **23** , **5**, **23** }, y[**4**] = { **4**, **4**, **24**, **24** };

**int** changex[**4**] = { **0** , **0** , **0**, **0** }, changey[**4**] = { **0**, **0**, **0**, **0** };

**int** rotate[**4**] = { **1**, **1**, **1**, **1** }, del = **0**;

**void** **reset**() {

x[**0**] = **5**; x[**1**] = **23**; x[**2**] = **5**; x[**3**] = **23**;

y[**0**] = **4**; y[**1**] = **4**; y[**2**] = **24**; y[**3**] = **24**;

**for** (**int** i = **0**; i < **4**; i++) {

changex[i] = **0**;

changey[i] = **0**;

rotate[i] = **1**;

}

del = **0**;

}

**void** **changeE**() {

del++;

**if** (del >= **100**) {

**for** (**int** i = **0**; i < **4**; i++) {

rotate[i] = rand() % **4** + **1**;

changex[i] = x[i];

changey[i] = y[i];

**switch** (rotate[i]) {

**case** **1**: **if** (Map[y[i]][changex[i] + **1**] != 'W') changex[i] += **1**; **break**;

**case** **2**: **if** (Map[y[i]][changex[i] - **1**] != 'W') changex[i] -= **1**; **break**;

**case** **3**: **if** (Map[changey[i] - **1**][x[i]] != 'W') changey[i] -= **1**; **break**;

**case** **4**: **if** (Map[changey[i] + **1**][x[i]] != 'W') changey[i] += **1**; **break**;

}

}

del = **0**;

}

**for** (**int** i = **0**; i < **4**; i++) {

**if** (Map[changey[i]][changex[i]] == ' ' || Map[changey[i]][changex[i]] == 'V' ||

Map[changey[i]][changex[i]] == 'P') {

**if** (Map[changey[i]][changex[i]] == 'V') Map[y[i]][x[i]] = 'V';

**else** **if** (Map[changey[i]][changex[i]] == ' ') Map[y[i]][x[i]] = ' ';

**else** **if** (Map[changey[i]][changex[i]] == 'P') {

gameState.life = false;

gameState.game\_over = true;

}

Map[changey[i]][changex[i]] = '1' + i;

x[i] = changex[i];

y[i] = changey[i];

}

// Телепорты для врагов

**if** (changey[i] == **14** && (changex[i] == **0** || changex[i] == **28**)) {

changex[i] = (changex[i] == **0**) ? **27** : **1**;

Map[y[i]][x[i]] = 'V';

Map[changey[i]][changex[i]] = '1' + i;

x[i] = changex[i];

y[i] = changey[i];

}

**if** (changex[i] == **14** && (changey[i] == **0** || changey[i] == **28**)) {

changey[i] = (changey[i] == **0**) ? **27** : **1**;

Map[y[i]][x[i]] = 'V';

Map[changey[i]][changex[i]] = '1' + i;

x[i] = changex[i];

y[i] = changey[i];

}

}

}

};

**void** **countTotalChicken**() {

gameState.total\_chicken = **0**;

**for** (**int** i = **0**; i < hight; i++) {

**for** (**int** j = **0**; j < width; j++) {

**if** (Map[i][j] == ' ') {

gameState.total\_chicken++;

}

}

}

}

**void** **resetGame**() {

String newMap[hight] = {

"WWWWWWWWWWWWWWVWWWWWWWWWWWWWW",

"W WVW W",

"W WWWWW WWWW VVV WWWW WWWWW W",

"W W W W WVW W W W W",

"W W W W WWWWWWVWWWWWW W W W W",

"W W W W W W",

"WWW WWWWW WWWWWWWWW WWWWW WWW",

"W W W W",

"W WWWWW WWWW W W WWWW WWWWW W",

"W W W W W W",

"WWWWW WWWWW WWWWW WWWWW WWWWW",

"W W W W",

"W W WWW WWWW WVW WWWW WWW W W",

"WWW W WVW W WWW",

"VVV WW WWW VPV WWW WW VVV",

"WWW W WVW W WWW",

"W W WWW WWWW WVW WWWW WWW W W",

"W W W W",

"WWWWW WWWWW WWWWW WWWWW WWWWW",

"W W W W W W",

"W WWWWW WWWW W W WWWW WWWWW W",

"W W W W",

"WWW WWWWW WWWWWWWWW WWWWW WWW",

"W W W W W W",

"W W W W WWWWWWVWWWWWW W W W W",

"W W W W WVW W W W W",

"W WWWWW WWWW VVV WWWW WWWWW W",

"W WVW W",

"WWWWWWWWWWWWWWVWWWWWWWWWWWWWW"

};

**for** (**int** i = **0**; i < hight; i++) {

Map[i] = newMap[i];

}

gameState.chicken\_num = **0**;

countTotalChicken();

gameState.life = true;

gameState.game\_over = false;

gameState.win = false;

}

**int** **main**() {

srand(time(**0**));

RenderWindow window(VideoMode(width \* block, hight \* block), "Pacman");

// Загрузка текстур

Texture w, pac, c, e, **win\_t**, **lose\_t**, teleport\_left, teleport\_right, teleport\_top, teleport\_bottom;

w.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/BrickWall2.png");

pac.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/pacman.png");

c.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/Chicken.png");

e.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/Enemy.png");

**win\_t**.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/Win1.png");

**lose\_t**.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/Lose1.png");

teleport\_left.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/YellowPortal1.png");

teleport\_right.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/YellowPortal2.png");

teleport\_top.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/BluePortal1.png");

teleport\_bottom.loadFromFile("C:/Users/matve/source/repos/Pac/Paint/BluePortal2.png");

Sprite wall(w), pacman(pac), chicken(c), enemy(e), win(**win\_t**), lose(**lose\_t**);

Sprite left\_portal(teleport\_left), right\_portal(teleport\_right),

top\_portal(teleport\_top), bottom\_portal(teleport\_bottom);

// Установка размеров для спрайтов победы и поражения

win.setScale((**float**)(width \* block) / win.getLocalBounds().width,

(**float**)(hight \* block) / win.getLocalBounds().height);

lose.setScale((**float**)(width \* block) / lose.getLocalBounds().width,

(**float**)(hight \* block) / lose.getLocalBounds().height);

left\_portal.setPosition(**0**, **14** \* block);

right\_portal.setPosition(**28** \* block, **14** \* block);

top\_portal.setPosition(**14** \* block, **0**);

bottom\_portal.setPosition(**14** \* block, **28** \* block);

Player pl;

Enemy en;

countTotalChicken(); // Подсчет кусочков курицы при запуске

**while** (window.isOpen()) {

Event event;

**while** (window.pollEvent(event)) {

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

window.close();

**if** (event.type == Event::KeyPressed) {

**if** (event.key.code == Keyboard::Escape)

window.close();

// Перезапуск игры при нажатии Enter

**if** ((gameState.game\_over || gameState.win) && event.key.code == Keyboard::Enter) {

resetGame();

pl.reset();

en.reset();

}

// Управление игроком

**if** (gameState.life && !gameState.game\_over && !gameState.win) {

pl.changex = pl.x;

pl.changey = pl.y;

**if** (event.key.code == Keyboard::D || event.key.code == Keyboard::Right) pl.rotate = **0**;

**if** (event.key.code == Keyboard::A || event.key.code == Keyboard::Left) pl.rotate = **1**;

**if** (event.key.code == Keyboard::W || event.key.code == Keyboard::Up) pl.rotate = **2**;

**if** (event.key.code == Keyboard::S || event.key.code == Keyboard::Down) pl.rotate = **3**;

}

}

}

// Игровая логика

**if** (gameState.life && !gameState.game\_over && !gameState.win) {

pl.changeP();

en.changeE();

}

// Отрисовка

window.clear(Color::Black);

// Фон (телепорты)

window.draw(left\_portal);

window.draw(right\_portal);

window.draw(top\_portal);

window.draw(bottom\_portal);

// Стены

**for** (**int** i = **0**; i < hight; i++)

**for** (**int** j = **0**; j < width; j++)

**if** (Map[i][j] == 'W') {

wall.setPosition(j \* block, i \* block);

window.draw(wall);

}

// Курица

**for** (**int** i = **0**; i < hight; i++)

**for** (**int** j = **0**; j < width; j++)

**if** (Map[i][j] == ' ') {

chicken.setPosition(j \* block, i \* block);

window.draw(chicken);

}

// Враги

**for** (**int** i = **0**; i < hight; i++)

**for** (**int** j = **0**; j < width; j++)

**if** (Map[i][j] >= '1' && Map[i][j] <= '4') {

**int** idx = Map[i][j] - '1';

enemy.setTextureRect(IntRect(block \* en.rotate[idx], **0**, block, block));

enemy.setPosition(j \* block, i \* block);

window.draw(enemy);

}

// Игрок - пакман

**if** (Map[pl.y][pl.x] == 'P') {

pacman.setTextureRect(IntRect(block \* **int**(pl.frame), block \* pl.rotate, block, block));

pacman.setPosition(pl.x \* block, pl.y \* block);

window.draw(pacman);

}

// Сообщения о победе/поражении

**if** (gameState.win) {

window.draw(win);

}

**else** **if** (gameState.game\_over) {

window.draw(lose);

}

window.display();

}

**return** **0**;

}