***Pacman***

**Разработкой занимался**:

- Наливайко Александр - Главный разработчик

**Дизайн документа**

**Цель проекта**:

Разработка 2D-игры "Maze!" с использованием графической библиотеки SFML. Игрок управляет персонажем, собирая монеты, избегая врагов и перемещаясь по лабиринту.

**Задачи проекта**:

- Создание графического интерфейса с использованием платформы Windows Forms.

- Разработка логики игры на языке программирования на основе правил игры Пакман.

- Интеграция Windows Forms с языком программирования C++.

- Использование технологии Github для хранения версий программы.

**Концепция**

Pac-Man — аркадная видеоигра, разработанная японской компанией Namco и вышедшая в 1980 году. Задача игрока — управляя колобком Пакманом, съесть все точки в лабиринте, избегая встречи с привидениями, которые гоняются за героем. С повышением уровня игры изменяется только её сложность, архитектура лабиринта остаётся идентичной.

В конце 1970-х из-за успеха игры Space Invaders рынок видеоигр сосредоточился на создании аркадных шутемапов и привлекал к себе сугубо мужскую аудиторию. Сотрудник Namco Тору Иватани хотел сделать игру, которая понравилась бы всем, в особенности девушкам, и в качестве основной темы своей работы выбрал еду. В Японии созданная игра получила название Puck-Man, но при локализации в США компания Midway изменила название на Pac-Man, считая, что при оригинальном названии дети могли бы закрасить середину буквы «P», превратив её в «F», и сделать слово обсценным.

После выхода в Японии игра была принята хорошо, но не стала популярной. В Америке же аудитория была впечатлена отсутствием в аркаде насильственного мотива, что привлекло в том числе женскую аудиторию и помогло заработать лояльность родителей к видеоигре. Всё это обеспечило популярность Pac-Man среди людей разных возрастов и профессий, породив повышенный интерес к игре и сделав Пакмана первой звездой видеоигр. Такой успех вдохновил разработчиков на создание более разнообразных игр, в том числе с отсутствием насилия.

**Игровой процесс**

- Игрок управляет персонажем (Pac-Man), перемещая его по лабиринту.

- Основная цель — собрать все монеты на карте (171 шт.), избегая врагов.

- Враги (4 персонажа) перемещаются по лабиринту случайным образом.

- Игра заканчивается победой, если все монеты собраны, или поражением, если игрок сталкивается с врагом.

**Персонажи**

- Главный герой в виде колобка.

- Враги в виде призраков.

**Техническая документация**

**Карта игры**

* Карта представлена сеткой размером 19x21.
* Элементы игры:

‘A’ - стены.

‘ ’ - проходы.

‘B’ - пройденные игроком клетки.

‘C’ - текущая позиция игрока.

‘1’,‘2’,‘3’,‘4’ - враги.

* Края карты соединены — персонажи могут переходить с одного края на другой.

**Основные компоненты кода**

**[Ссылка на Github с игрой(нажмите)](https://github.com/RadioDemon1219/Packman.git)**

**Карта игры (tileMap):**

Массив строк, представляющий игровое поле. Символы определяют объекты на карте.

**Класс Player**

Управляет логикой игрока:

1. Движение в направлениях: вправо, влево, вверх, вниз.
2. Сбор монет: увеличение счётчика coins.
3. Реакция на столкновение с врагами: окончание игры.
4. Переходы между краями карты.

Основные поля:

1. x, y : текущие координаты игрока.
2. newx, newy: координаты новой позиции после движения.
3. rotate: направление движения (1 — вправо, 2 — влево, 3 — вверх, 4 — вниз).
4. delayOfStep: задержка между шагами

**Класс Enemy**

Управляет логикой врагов:

1. Случайное движение.
2. Обработка столкновений с игроком.
3. Переходы между краями карты.

Основные поля:

1.x[4], y[4]: текущие координаты каждого врага.

2.newx[4],newy[4]: новые координаты врагов после движения.

3.rotate[4]: направления движения.

4.delayOfStep: задержка между шагами.

Метод update: Обновляет позиции врагов и проверяет столкновения с игроком.

Главная функция (main):

1. Инициализирует окно игры и текстуры.
2. Реализует игровой цикл:

* Обработка событий (закрытие окна, управление игроком).
* Обновление состояния игрока и врагов.
* Отрисовка карты и элементов интерфейса.
* Проверка условий завершения игры.

**Используемые технологии:**

Библиотека SFML:

1. Управление окном, графикой и событиями.
2. Классы: RenderWindow, Texture, Sprite,Event.

Язык программирования: C++.

**Архитектура программы:**

Уровень данных:

1. Массив tileMap хранит карту игры.
2. Переменные coins, определяют состояние игры.

Уровень логики:

1. Классы Player и Enemy реализуют поведение персонажей.
2. Методы update обновляют состояние объектов.

Уровень представления:

1. Графические текстуры отображают элементы игры.
2. Интерфейс обновляется в игровом цикле.

**Пространство имен**

using namespace sf; - способ использования SFML в качестве мультимедии, чтобы облегчить разработку игры.

Пространство имен SFML используется для упрощения доступа к мультимедийным функциям библиотеки, что облегчает разработку игры.

**Константы**

const int heightOfMap = 21;

Высота игрового поля в ячейках.

const int widthOfMap = 19;

Ширина игрового поля в ячейках.

const int sizeOfPlate = 25.

Размер одной ячейки игрового поля в пикселях.

**Элементы управления**

Игрок может управлять движением персонажа с помощью следующих клавиш:

1. Стрелки:Keyboard::Right, Keyboard::Left, Keyboard::Up, Keyboard::Down.
2. WASD: Keyboard::D, Keyboard::A,Keyboard::W, Keyboard::S.

Каждая из этих клавиш позволяет игроку перемещаться по карте в соответствующем направлении.

**Классы**

**Player**

Класс, представляющий игрока. Содержит:

1. Координаты игрока (x, y).
2. Методы управления движением и взаимодействием с объектами карты.
3. Метод void update() для обновления состояния игрока, включая обработку сбора монет и столкновений.

**Enemy**

Класс, представляющий врагов. Содержит:

1. Координаты и направления движения для каждого врага.
2. Метод void update() для управления движением врагов и обработки столкновений.

**Настройка элементов**

**Основные текстуры и объекты:**

1. Победа:

* Texture pictureYouWin;
* Загрузка текстуры и размещение на экране.

1. Проигрыш:

* Texture pictureYouLoose;
* Загрузка текстуры и размещение на экране.

1. Карта игры:

* Texture mapGame;
* Загрузка текстуры для игрового поля.

**Окно игры**

Создание игрового окна:

RenderWindow window(VideoMode(widthOfMap \* sizeOfPlate, heightOfMap \* sizeOfPlate), "Maze!");

**Обработчики событий**

В игре предусмотрены следующие обработчики событий:

1. Движение игрока: if (coins < 171 && life) { player.update(); enemy.update(); }
2. Отображение победы: if(coins==171)window.draw(youwin);
3. Отображение поражения: if (!life) window.draw(youlose);

**Метод обновления карты**

Каждый кадр игра обновляет карту, отрисовывая все текстуры, состояния игрока и врагов: window.clear(Color::Black); for (int i = 0; i < heightOfMap; i++) for (int j=0; j < widthOfMap; j++) { plat.setPosition(j \* sizeOfPlate, i \* sizeOfPlate); window.draw(plat); } window.display();

**Документации по тестированию**

**Тестирование**

Успешные сценарии:

1. Игрок корректно перемещается и собирает монеты.
2. Враги передвигаются случайным образом.
3. Карта отображается корректно.

Неудачные сценарии:

1.Не обнаружено.

**Возможные улучшения.**

1. Добавить уровни сложности.
2. Реализовать бонусные предметы.
3. Улучшить управление и анимацию.
4. Добавить аудиоэффекты.