Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.04– «Программная инженерия»

**Отчёт по учебной практике**

**на тему игра “Змейка”**

Выполнил студент гр. РИС-23-3Б

Мельников А.Д

Проверил:

Доцент кафедры ИТАС

Петренко А.А

(оценка) (подпись)

(дата)

Пермь, 2024

ВВЕДЕНИЕ

Разработка игр является увлекательным и творческим процессом создания развлечения, которое могут полюбить многие пользователи. Сама разработка игры начинается с идеи п предпочтений, которые в дальнейшем превращаются в дизайн игры, сюжетную линию и механику игры. Каждый этап разработки игры требует креативности и профессионализма. Одним из примеров игр, которая многим нравилась и возможно нравится и сейчас, это змейка. Сама по себе змейка представляет собой простую и захватывающую аркаду, в которой игрок управляет змейкой, увеличивая ее размеры путем поедания еды. Существует множество вариаций и модификаций, но сама суть игры остается неизменной.

Цель индивидуального задания – разработать игру “Змейка” с реализацией графического интерфейса, позволяющего пользователю взаимодействовать с игрой.

Для достижения этой цели поставлены следующие задачи:

1. Анализ реализации игры “Змейка”.

2. Разработка игры с реализацией механики данной игры и графического интерфейса пользователя.

1. **Анализ реализации игры “Змейка”**

Главная суть игры “Змейка” заключается в том, чтобы путем движения змейка поедала еду и от этого увеличивалась, и так до того момента, пока змейка не займет все поле и не съест себя.

Для представления поля вводятся его размеры через переменные, данное поле делится на клетки, чтобы в дальнейшем можно было вывести их на экран. Поле будет отображаться в виде картинки, чтобы игра была визуализированной, а для этого подключаем заранее скачанную картинку к игре через библиотеку SFML.

Для змейки используется структура, в которой обозначены координаты, и освобождено место для массива, чтобы змейка могла увеличиваться в размерах. Сама змейка будет выводиться из блоков, а также тело и голова будут различаться, так как движение будет зависеть от положения головы. Сам блок змейки будет заранее загружен и подключен через библиотеку SFML.

Вводится функция, которая будет выполнять условия игры. В ней прописаны начальные координаты, движения змейки, поведение змейки, когда она выходит за поля, поедание еды и остановка игры при поедании самой себя. Если змейка выходит за пределы поля, то она прорисуется с противоположной стороны, поэтому игра будет окончена только тогда, когда змейка съест саму себя. В данной функции ещё будет прописано, как змейка будет двигаться, для этого вводится переменная, которая будет отличать право, лево, вверх и вниз.

Еда представлена в виде структуры с координатами. Изначальные координаты будут прописаны вручную, но в дальнейшем при поедании еда будет рандомно менять свое значение.

При нажатии клавиш змейка меняет свое направление с учетом прописанных значений в функции, которые отвечают за движение. А для того, чтобы змейка начала двигаться, вводится временное значения, означающее скорость передвижения змейки.

Когда игра будет окончена, то есть змейка врежется в саму себя, нужно будет перезапустить игру. Для этого нужно будет закрыть окно и заново запустить программу.

**Вывод**

Реализация игры “Змейка” может быть отличной и интересной задачей для разработчиков, так как в реализации используются различные методы и технологии для создания геймплея.

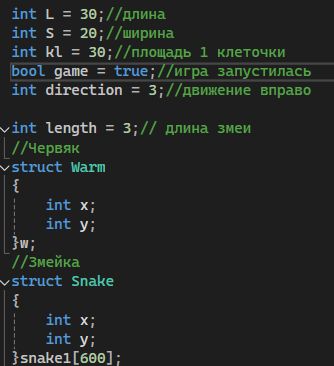
1. **Разработка программы**

Для реализации алгоритмов и логики игры был выбран язык программирования С++.

Для разработки и откладки кода программы используется программная среда SFML.

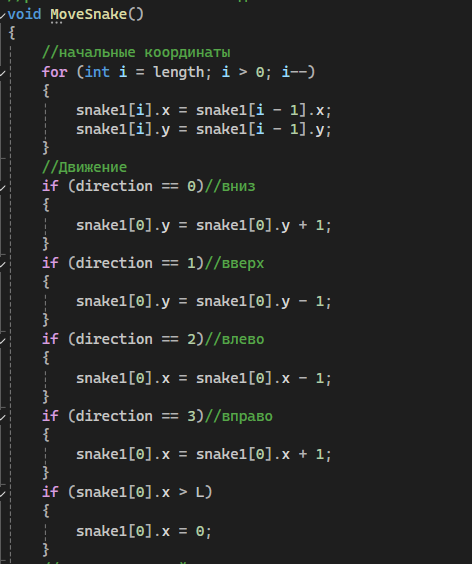
Графический интерфейс реализован средствами визуализации SFML.

Сначала прописаны значения и структуры.

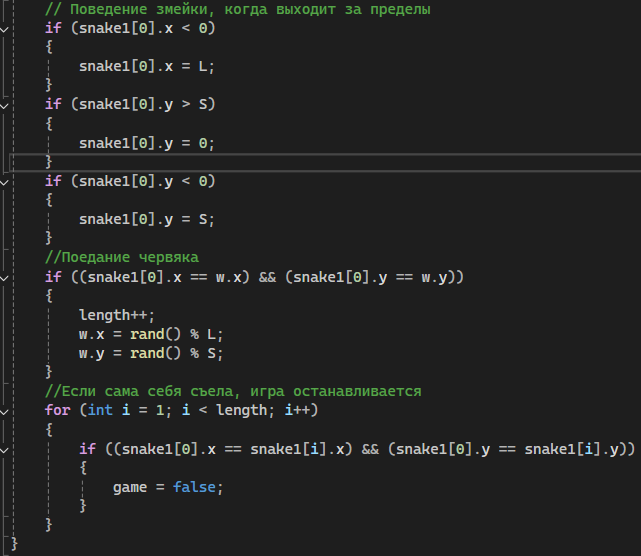


*Рисунок 1. Значения и структуры*

Метод MoveSnake() задает начальные координаты змейки. Указывает начальные координаты змейки. Указывает движения змейки через direction. Если direction = 0, то движение вниз. Если direction = 1, то движение вверх. Если direction = 2, то движение влево. Если direction = 3, то движение вправо. Указывает поведение змейки при попадании за пределы поля, то есть, если змейка уходит до правой стороны поля и заходит за пределы, то змейка выходит с левой стороны поля, аналогично работает с верхней стороной. Указывает увеличение змейки при поедании червяка. Увеличивается длина змейки и положение червяка рандомно меняется. Если змейка съела себя, то игра останавливается.



*Рисунок 2.1. Метод MoveSnake*

**

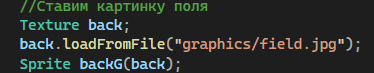
*Рисунок 2.2. Продолжение метода MoveSnake*

Установка изображений объектов:

Для визуализации объектов используется библиотека SFML.

Подключаются файлы с расширением “.png” и “.jpg” для объектов:

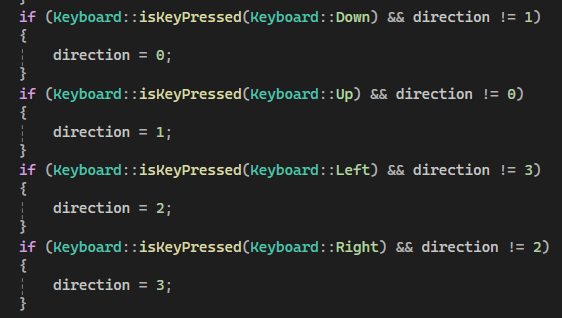
поля, змейки и червяка.



*Рисунок 3. Установка изображений объектов*

Реализация управления змейкой:

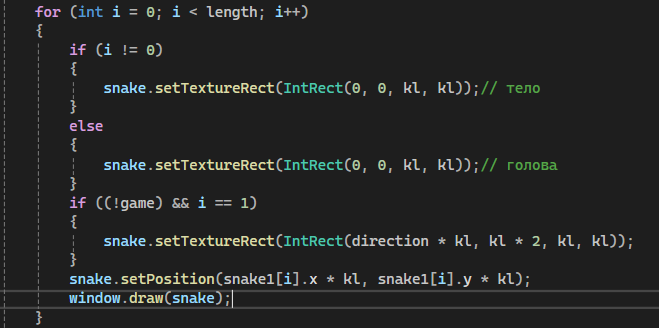
При нажатии кнопки Up змейка меняет свое направление и движется вверх, при нажатии кнопки Down змейка меняет свое направление и движется вниз, при нажатии кнопки Left змейка меняет свое направление и движется влево и при нажатии кнопки Right змейка меняет свое направление и движется вправо.



*Рисунок 4. Реализация управления змейкой.*

Вывод изображений объектов:

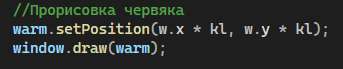
Вывод змейки, поля и червяка происходит благодаря функции draw(). У змейки выводится сначала тело, а затем голова, и задается начальное местоположение.



*Рисунок 5.1. Прорисовка змейки*

**

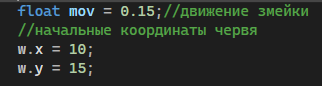
*Рисунок 5.2. Прорисовка поля*

**

*Рисунок 5.3. Прорисовка червяка*

Изначальное местоположение червяка и скорость змейки:

Координаты червяка прописаны до main(). Скорость змейки прописывается заранее.



*Рисунок 6. Изначальное местоположение червяка и скорость змейки*

Были использованы временные переменные из-за которых змейка смогла двигаться.

**Вывод:**

В ходе разработки игры “Змейка” с использованием SFML были реализованы сама идея игры в коде и визуализация игровых объектов таких, как поле, змейка и червяк. Змейка и червяк создаются с помощью структур, которые хранят координаты.

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы был выполнен анализ способа реализации программы для игры “Змейка”. “Змейка” представляет собой интересное задания для разработчиков, так как потребуется создать качественный дизайн и графику, реализовать суть игры, адаптировать игру под SFML, а также требует тестирование работы игры.

Была выполнена идея и механики игры с визуализацией через библиотеку SFML. Сама программа была написана на языке программирования C++. Можно доработать игру, добавив звуки и функцию подсчёта очков.

**Список литературы**

[1]Знакомство с SFML. Компьютерная графика/ [Электронный ресурс] // ps-group: [сайт]. – URL: https://ps-group.github.io/ps1/lesson\_sfml\_1

[2] SFML и Visual Studio / [Электронный ресурс] // devtribe: [сайт]. — URL: <https://devtribe.ru/p/programming/sfml-in-visual-studio>

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

#include <SFML/Graphics.hpp>

using namespace sf;

int L = 30;//длина

int S = 20;//ширина

int kl = 30;//площадь 1 клеточки

bool game = true;//игра запустилась

int direction = 3;//движение вправо

int length = 3;// длина змеи

//Червяк

struct Warm

{

int x;

int y;

}w;

//Змейка

struct Snake

{

int x;

int y;

}snake1[600];

//расположение змейки и ее движения

void MoveSnake()

{

//начальные координаты

for (int i = length; i > 0; i--)

{

snake1[i].x = snake1[i - 1].x;

snake1[i].y = snake1[i - 1].y;

}

//Движение

if (direction == 0)//вниз

{

snake1[0].y = snake1[0].y + 1;

}

if (direction == 1)//вверх

{

snake1[0].y = snake1[0].y - 1;

}

if (direction == 2)//влево

{

snake1[0].x = snake1[0].x - 1;

}

if (direction == 3)//вправо

{

snake1[0].x = snake1[0].x + 1;

}

if (snake1[0].x > L)

{

snake1[0].x = 0;

}

// Поведение змейки, когда выходит за пределы

if (snake1[0].x < 0)

{

snake1[0].x = L;

}

if (snake1[0].y > S)

{

snake1[0].y = 0;

}

if (snake1[0].y < 0)

{

snake1[0].y = S;

}

//Поедание червяка

if ((snake1[0].x == w.x) && (snake1[0].y == w.y))

{

length++;

w.x = rand() % L;

w.y = rand() % S;

}

//Если сама себя съела, игра останавливается

for (int i = 1; i < length; i++)

{

if ((snake1[0].x == snake1[i].x) && (snake1[0].y == snake1[i].y))

{

game = false;

}

}

}

int main()

{

RenderWindow window(VideoMode(L \* kl, S \* kl), L"Змейка");

//Ставим картинку поля

Texture back;

back.loadFromFile("graphics/field.jpg");

Sprite backG(back);

//Ставим иконку приложения

Image Icon;

Icon.loadFromFile("graphics/snake.png");

window.setIcon(32, 32, Icon.getPixelsPtr());

//Ставим картинку змейки

Texture sn;

sn.loadFromFile("graphics/block.png");

Sprite snake(sn);

//Картинка червяка

Texture warmm;

warmm.loadFromFile("graphics/food.png");

Sprite warm(warmm);

Clock clock;

float tim = 0;

float mov = 0.15;//движение змейки

//начальные координаты червя

w.x = 10;

w.y = 15;

while (window.isOpen())

{

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

clock.restart();

tim += time;

Event event;

while (window.pollEvent(event))

{

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

window.close();

}

if (Keyboard::isKeyPressed(Keyboard::Down) && direction != 1)

{

direction = 0;

}

if (Keyboard::isKeyPressed(Keyboard::Up) && direction != 0)

{

direction = 1;

}

if (Keyboard::isKeyPressed(Keyboard::Left) && direction != 3)

{

direction = 2;

}

if (Keyboard::isKeyPressed(Keyboard::Right) && direction != 2)

{

direction = 3;

}

// чтобы змейка могла двигаться

if ((tim > mov) && game)

{

tim = 0;

MoveSnake();

}

window.clear();

//прорисовка поля

window.draw(backG);

//прорисовка змейки

for (int i = 0; i < length; i++)

{

if (i != 0)

{

snake.setTextureRect(IntRect(0, 0, kl, kl));// тело

}

else

{

snake.setTextureRect(IntRect(0, 0, kl, kl));// голова

}

if ((!game) && i == 1)

{

snake.setTextureRect(IntRect(direction \* kl, kl \* 2, kl, kl));

}

snake.setPosition(snake1[i].x \* kl, snake1[i].y \* kl);

window.draw(snake);

}

//Прорисовка червяка

warm.setPosition(w.x \* kl, w.y \* kl);

window.draw(warm);

window.display();

}

return 0;

}

**ПРИЛОЖЕНИЕ Б**



*Рисунок Б.1. Блок змейки*



*Рисунок Б.2. Поле*



*Рисунок Б.3. Червяк*



*Рисунок Б.4. Иконка приложения*