Государственное автономное профессиональное образовательное учреждение Свердловской области «Уральский радиотехнический колледж им. А. С. Попова»

**ЛАБОРАТОРНАЯ РАБОТА № 4**

По дисциплине «Информатика»

На тему «Тетрис»

|  |  |
| --- | --- |
| Студент: | Катаев Е. А., Веремчук М. Р. |
| Группа: | Р-153 |
| Преподаватель: | Зырянов А. В. |

Екатеринбург

2023

**Оглавление**

[1 Постановка задачи 3](#_Toc128516652)

[2 Правила игры тетрис 4](#_Toc128516653)

[2.1 Текстуры 5](#_Toc128516654)

[3 Код 6](#_Toc128516655)

[4 Результат работы 9](#_Toc128516656)

[5 Вывод 12](#_Toc128516657)

# **1 Постановка задачи**

Изучение библиотеки SFML – Simple and Fast Multimedia Library, написание программы с использованием данной библиотеки, а именно: реализация видео-игры "Тетрис" на языке программирования C++, который является объектно-ориентированным языком программирования.

# **2 Правила игры тетрис**

Правила игры знают, пожалуй, все: сверху в прямоугольный стакан шириной 10 и высотой 20 клеток в произвольном порядке падают фигурки, которые игрок во время полёта может поворачивать и смещать по горизонтали. Также можно «сбрасывать» фигурку, то есть ускорять её падение, когда уже продумано, куда этот элемент должен встать. При заполнении без пустот целого ряда высотой в один квадратик этот ряд исчезает и всё, что выше него, опускается ниже на одну клетку. Темп игры постепенно увеличивается. Задача игрока — заполнять полностью ряды, ускорять их исчезновение, получать за них очки. Как только всё игровое поле заполнится рядами с пустотами, игра закончится. Начисление очков в разных версиях тетриса не совпадает. Очки могут начисляться за убранные ряды.

## **2.1 Текстуры**

Ниже представлены текстуры, использованные для реализации объектов (спрайтов) в игре:

**C:\Users\koteg\Desktop\C++shka\Tetris True\texture.png**

Рисунок 1 - Текстура

****

Рисунок 2 - Иконка

# **3 Код**

#include <SFML/Audio.hpp>

#include <SFML/Graphics.hpp>

#include <sstream>

struct coordinates {

int x;

int y;

} oneFigure[4], phantom[4], nextFigure[4];

const int width = 10;

const int height = 20;

int figures[7][4] = {

4, 14, 24, 34, //I

4, 14, 15, 24, //T

4, 14, 24, 25, //L

5, 15, 24, 25, //J

4, 5, 14, 15, //O

4, 14, 15, 25, //S

5, 14, 15, 24, //Z

};

int nextFigures[7][4] = {

38, 64, 90, 116, //I

38, 64, 65, 90, //T

38, 64, 90, 91, //L

39, 65, 90, 91, //J

38, 39, 64, 65, //O

38, 64, 65, 91, //S

39, 64, 65, 90, //Z

};

int nextFigureRandom = rand() % 7;

int field[width][height] = { 0 };

bool gameOver = false;

bool win = false;

int figureRandom = rand() % 7;

int color = rand() % 8 + 1;

int newColor = rand() % 8 + 1;

bool check() {

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

if ((oneFigure[i].x < 0) || (oneFigure[i].x > (width - 1)) || (oneFigure[i].y > (height - 1)))

return false;

if (field[oneFigure[i].x][oneFigure[i].y])

return false;

}

}

void revers() {

if (!check())

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

oneFigure[i] = phantom[i];

}

void nextSpawn() {

srand(time(0));

if ((!win) && (!gameOver)) {

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

nextFigure[i].x = nextFigures[nextFigureRandom][i] % (width \* 2 + 6);

nextFigure[i].y = nextFigures[nextFigureRandom][i] / (width \* 2 + 6);

}

}

}

void spawn() {

srand(time(0));

if ((!win) && (!gameOver))

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

oneFigure[i].x = figures[figureRandom][i] % width;

oneFigure[i].y = figures[figureRandom][i] / width;

}

}

int main() {

srand(time(0));

const int size = 34;

sf::RenderWindow window(sf::VideoMode((width + 6) \* size, height \* size), L"Тетрис Р-153", sf::Style::Close);

sf::Texture textureBlock;

textureBlock.loadFromFile("texture.png");

sf::Sprite block(textureBlock);

sf::Image icon;

icon.loadFromFile("icon.png");

window.setIcon(512, 512, icon.getPixelsPtr());

sf::Font font;

font.loadFromFile("font.ttf");

sf::Text scoreText("", font, size);

sf::Text gameOverText("", font, size \* 2);

sf::Text winText("", font, size \* 2);

scoreText.setFillColor(sf::Color::Color(192, 192, 192, 255));

gameOverText.setFillColor(sf::Color::Color(192, 192, 192, 255));

gameOverText.setStyle(sf::Text::Bold);

winText.setFillColor(sf::Color::Color(192, 192, 192, 255));

winText.setStyle(sf::Text::Bold);

int blockX = 0;

int score = 0;

bool rotate = false;

float delay = 0.5;

float time = 0;

sf::Clock watch;

while (window.isOpen()) {

sf::Time timer = watch.getElapsedTime();

time += timer.asSeconds();

sf::Event press;

while (window.pollEvent(press)) {

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

window.close();

if (press.type == sf::Event::KeyPressed) {

if ((press.key.code == sf::Keyboard::A) || (press.key.code == sf::Keyboard::Left))

blockX--;

else if ((press.key.code == sf::Keyboard::D) || (press.key.code == sf::Keyboard::Right))

blockX++;

else if ((press.key.code == sf::Keyboard::Space) || (press.key.code == sf::Keyboard::Up))

rotate = true;

else if (press.key.code == sf::Keyboard::LShift)

delay = 0;

}

}

if ((sf::Keyboard::isKeyPressed(sf::Keyboard::S)) || ((sf::Keyboard::isKeyPressed(sf::Keyboard::Down))))

delay = 0.2;

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

phantom[i] = oneFigure[i];

oneFigure[i].x += blockX;

}

revers();

if ((time >= delay) && (!win) && (!gameOver)) {

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

phantom[i] = oneFigure[i];

oneFigure[i].y++;

}

if (!check()) {

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

field[phantom[i].x][phantom[i].y] = color;

figureRandom = nextFigureRandom;

color = newColor;

spawn();

nextFigureRandom = rand() % 7;

newColor = rand() % 8 + 1;

nextSpawn();

}

time = 0;

}

if (rotate) {

coordinates centerBlock = oneFigure[1];

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

int newX = oneFigure[i].y - centerBlock.y;

int newY = oneFigure[i].x - centerBlock.x;

oneFigure[i].x = centerBlock.x - newX;

oneFigure[i].y = centerBlock.y + newY;

}

}

revers();

if (oneFigure[0].y == 0) {

spawn();

nextSpawn();

}

int line = height - 1;

for (int i = line; i > 0; i--) {

int blockCount = 0;

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

if (field[j][i] != 0)

blockCount++;

field[j][line] = field[j][i];

}

if (blockCount != width)

line--;

else

score += 100;

}

if (score >= 100)

win = true;

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

if ((field[phantom[i].x][phantom[i].y] == color) && (field[oneFigure[i].x][oneFigure[i].y]))

gameOver = true;

watch.restart();

rotate = false;

blockX = 0;

delay = 0.5;

window.clear();

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

for (int j = 0; j < height; j++) {

block.setTextureRect(sf::IntRect(field[i][j] \* size, 0, size, size));

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

window.draw(block);

}

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

block.setTextureRect(sf::IntRect(color \* size, 0, size, size));

block.setPosition(oneFigure[i].x \* size, oneFigure[i].y \* size);

window.draw(block);

}

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

block.setTextureRect(sf::IntRect(newColor \* size, 0, size, size));

block.setPosition(nextFigure[i].x \* size, nextFigure[i].y \* size);

window.draw(block);

}

if (gameOver) {

gameOverText.setString("!YOU LOSE!");

gameOverText.setPosition((width / 4) \* size - 40, height);

window.draw(gameOverText);

}

if (win) {

winText.setString("!YOU WIN!");

winText.setPosition((width / 4) \* size - 10, height);

window.draw(winText);

}

std::ostringstream scoreStream;

scoreStream << score;

scoreText.setString("Score:\n" + scoreStream.str());

scoreText.setPosition((width + 1) \* size, 6 \* size);

window.draw(scoreText);

window.display();

}

}

# **4 Результат работы**

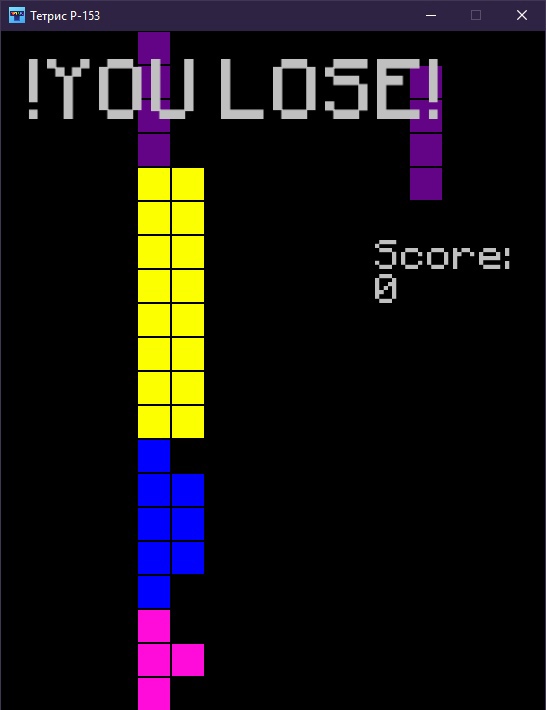


Рисунок 6 - Проигрыш



Рисунок 7 - Победа

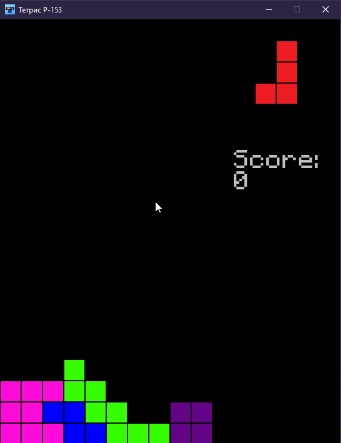


Рисунок 8 - удаление линии (До)

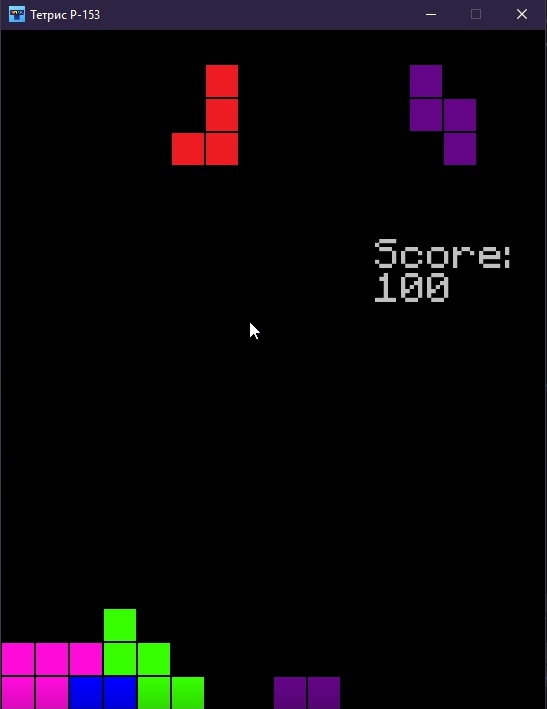


Рисунок 9 - удаление линий (После)

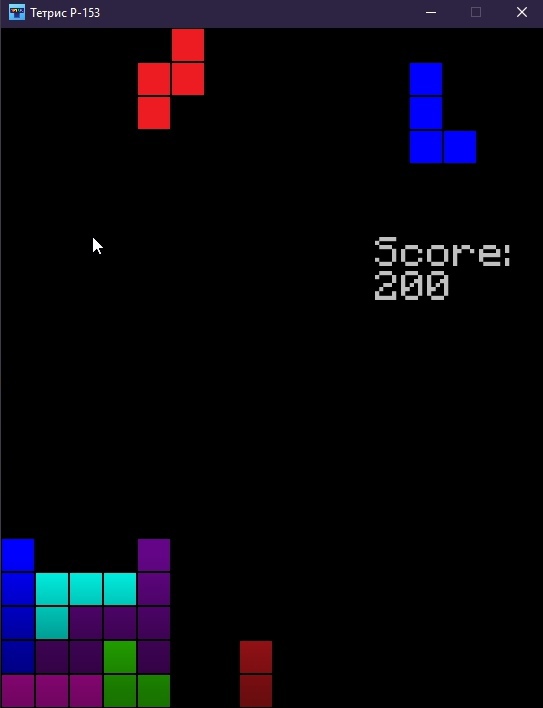


Рисунок 10 - Появление новой фигуры

# **5 Вывод**

В ходе лабораторной работы мы научились пользоваться библиотекой SFML, и реализовали игру "Тетрис", на основе данной библиотеки в объектно-ориентированном программировании, в среде разработки "Visual studio" версии "community" 2021 года, на языке "C++".