Федеральное агентство связи

Сибирский Государственный Университет Телекоммуникаций и Информатики

СибГУТИ

Кафедра прикладной математики и кибернетики

Курсовая работа

по дисциплине «Программирование».

Вариант 23. Тема “Пятнашки”

Выполнила: студентка 1 курса группы ИП-013

Иванов Леонид Дмитриевич

Преподаватель: Перцев Игорь Владимирович

Содержание:

1. Введение
2. Постановка задачи
3. Реализация программы
4. Код программы
5. Результат работы программы

**Введение**

Целью курсового проекта являлось составление игры «Пятнашки».

Игровой процесс завязан на решении головоломки. В квадратной коробке размещено nxn количество квадратов с размещенными на них случайными цифрами от 1 до n-1, в одном ячейке отсутствует. Цель игрока - перемещая пазл по заданной области с помощью мыши, добиться упорядочивания их по номерам, желательно сделав как можно меньше перемещений.

Программа написана на языке C++. Реализация интерфейса происходит с помощью графической библиотеки SFML.

**Постановка задачи**

Моей задачей было реализовать интерфейс меню игры, с выбором сложности – сложность 3х3 и 4х4.Так же сделать рандом(или тасовку) элементов. А так же при победе автоматическое завершение игры.

**Реализация программы**

Программа состоит из одного файла.

**tag.cpp.**

В нем я расписал программу на 4 функции не считая функцию **main.**

**Функция menu**

Включает в себя меню игры.   
При запуске программы, размер окна составляет (1280x720) и остаётся без изменений до запуска самой игры.

**Функция game**

В ней находится сама игра.

С начало с помощью переменных s и s2 мы задаём размер окна, в зависимости от уровня сложности, которое мы выбираем в функции **menu.**

Дальше мы создаём динамический массив Sprite и загружаем туда картинку.

Дальше мы через цикл for и другие массивы, разделяем картинку на элементы. У нас получается отсортированная игра “Пятнашки”.

Чтобы при запуске игры элементы были в рандомном порядке мы подключаем **функцию shuffle.** Дальше нам уметь переставлять элементы в пустую ячейку. Для этого мы создаём координаты **x,y** которые равны **позиция мышки / размер ячейки + 1.** Потом создаём коэффициенты **dx,dy** которые равны изначально **0**. Ну и пропишем условия для перестановки ячейки на пустое место зависимости от нахождения элемента.

**Функция shuffle**

В этой функции мы через **цикл for** рандомно выбираем элемент массива и просто через временную переменную тасуем массив.

**Функция checkwin**

Функция проверяет, упорядочен ли массив **fin**.Если “да”, - пятнашки собраны.  
Функция возвращает логическое **true**.Если финал сборки – пятнашек не наступил – возвращает ложное значение **false.**

**Код программы**

**Tag.cpp**

#include <SFML/Audio.hpp>

#include <SFML/Graphics.hpp>

#include <array>

#include <ctime>

#include <iostream>

#include <math.h>

#include <string.h>

using namespace sf;

using namespace std;

bool checkwin(int s2, int\*\* fin)

{

bool exit;

int i = 0, j = 0;

int k = 1;

exit = true;

for (int i = 0; i < s2; i++)

{

for (int j = 0; j < s2; j++) {

if (fin[i + 1][j + 1] == k)

k++;

else

{

exit = false;

break;

}

if (!exit)

{

break;

}

}

}

return exit;

}

void shuffle(int\* arr, int size)

{

srand(time(NULL));

for (int i = size - 1; i >= 1; i--) {

int j = rand() % (i + 1);

int tmp = arr[j];

arr[j] = arr[i];

arr[i] = tmp;

}

}

void game(const int Sprite\_size)

{

const int s = Sprite\_size;

int s2 = sqrt(s);

RenderWindow window(sf::VideoMode(s2 \* 128, s2 \* 128), "GAME");

Texture tagimage;

if (s == 9) {

tagimage.loadFromFile("media/image/tag\_game/9.png");

}

else {

tagimage.loadFromFile("media/image/tag\_game/tag\_image.png");

}

Sprite\* tag = new Sprite[s + 1];

int size = 128;

int\*\* fin = new int\* [s2 + 3];

for (int count = 0; count < s2 + 3; count++)

fin[count] = new int[s2 + 3];

int n = 0;

int\* arr = new int[s + 1];

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

{

arr[i] = i + 1;

}

shuffle(arr, s);

int\*\* arr2 = new int\* [s2 + 1];

for (int count = 0; count < s2 + 1; count++)

arr2[count] = new int[s2 + 1];

int count = 0;

for (int i = 0; i < s2; i++)

for (int j = 0; j < s2; j++) {

arr2[i][j] = arr[count];

count++;

}

for (int i = 0; i < s2; i++) {

for (int j = 0; j < s2; j++) {

n++;

tag[n].setTexture(tagimage);

tag[n].setTextureRect(IntRect(i \* size, j \* size, size, size));

fin[i + 1][j + 1] = arr2[i][j];

}

}

while (window.isOpen()) {

Vector2i mousePoz = Mouse::getPosition(window);

sf::Event event;

while (window.pollEvent(event)) {

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

window.close();

}

if (checkwin(s2, fin) == true)

{

window.close();

}

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

if (event.key.code == Mouse::Left) {

Vector2i Position = Mouse::getPosition(window);

int x = Position.x / size + 1;

int y = Position.y / size + 1;

int dx = 0;

int dy = 0;

if (fin[x + 1][y] == s) {

dx = 1;

dy = 0;

}

if (fin[x - 1][y] == s) {

dx = -1;

dy = 0;

}

if (fin[x][y + 1] == s) {

dx = 0;

dy = 1;

}

if (fin[x][y - 1] == s) {

dx = 0;

dy = -1;

}

n = fin[x][y];

fin[x][y] = s;

fin[x + dx][y + dy] = n;

}

}

for (int i = 0; i < s2; i++) {

for (int j = 0; j < s2; j++) {

n = fin[i + 1][j + 1];

tag[n].setPosition(i \* size, j \* size);

window.draw(tag[n]);

}

}

window.display();

}

delete[] tag;

delete[] arr;

for (int i = 0; i < s2 + 1; i++) {

delete[] arr2[i];

}

delete[] arr2;

for (int i = 0; i < s2 + 3; i++) {

delete[] fin[i];

}

delete[] fin;

}

void menu(RenderWindow& window, int menuType)

{

array<Sprite, 5> button;

array<RectangleShape, 5> coord;

SoundBuffer knopkasound;

knopkasound.loadFromFile("media/sound/knopka.ogg");

Sound knopkamusic(knopkasound);

Texture fontexture, gametexture, exittexture, difficultytext, difficulty3x3,

difficulty4x4, back;

fontexture.loadFromFile("media/image/fon.png");

gametexture.loadFromFile("media/image/game.png");

exittexture.loadFromFile("media/image/exit.png");

difficultytext.loadFromFile("media/image/Difficulty.png");

difficulty3x3.loadFromFile("media/image/Difficulty3x3.png");

difficulty4x4.loadFromFile("media/image/Difficulty4x4.png");

back.loadFromFile("media/image/back.png");

Sprite fon(fontexture), text1(difficultytext);

button[0].setTexture(gametexture);

button[1].setTexture(exittexture);

button[2].setTexture(difficulty3x3);

button[3].setTexture(difficulty4x4);

button[4].setTexture(back);

bool isMenu = 1;

int menuNum = 0;

button[0].setPosition(350, 80);

button[1].setPosition(350, 200);

text1.setPosition(350, 10);

coord[0].setSize(Vector2f(230, 50));

coord[0].setPosition(540, 250);

coord[1].setSize(Vector2f(120, 50));

coord[1].setPosition(580, 360);

if (menuType == 2) {

button[2].setPosition(350, 80);

button[3].setPosition(380, 160);

button[4].setPosition(350, 240);

coord[2].setSize(Vector2f(230, 30));

coord[2].setPosition(540, 250);

coord[3].setSize(Vector2f(230, 30));

coord[3].setPosition(540, 350);

coord[4].setSize(Vector2f(120, 30));

coord[4].setPosition(590, 410);

}

while (isMenu) {

Vector2i mousePos = Mouse::getPosition(window);

Vector2f pos = window.mapPixelToCoords(mousePos);

sf::Event event;

while (window.pollEvent(event)) {

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

window.close();

}

for (int i = 0; i < 5; i++) {

button[i].setColor(Color::White);

}

menuNum = 0;

for (int i = 0; i < 5; i++) {

if (coord[i].getGlobalBounds().contains(pos.x, pos.y)) {

button[i].setColor(Color::Blue);

menuNum = i + 1;

}

}

if (Mouse::isButtonPressed(Mouse::Left)) {

if (menuNum == 1) {

knopkamusic.play();

menu(window, 2);

}

if (menuNum == 2) {

window.close();

isMenu = false;

}

if (menuNum == 3) {

knopkamusic.play();

game(9);

}

if (menuNum == 4) {

knopkamusic.play();

game(16);

}

if (menuNum == 5) {

knopkamusic.play();

menu(window, 1);

}

}

window.draw(fon);

if (menuType == 1) {

window.draw(button[0]);

window.draw(button[1]);

}

if (menuType == 2) {

window.draw(button[2]);

window.draw(button[3]);

window.draw(button[4]);

}

window.display();

}

}

int main()

{

RenderWindow window(sf::VideoMode(1280, 720), "Tag game");

menu(window, 1);

while (window.isOpen()) {

Vector2i mousePoz = Mouse::getPosition(window);

sf::Event event;

while (window.pollEvent(event)) {

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

window.close();

}

window.clear();

window.display();

}

return false;

}

**Скриншоты**

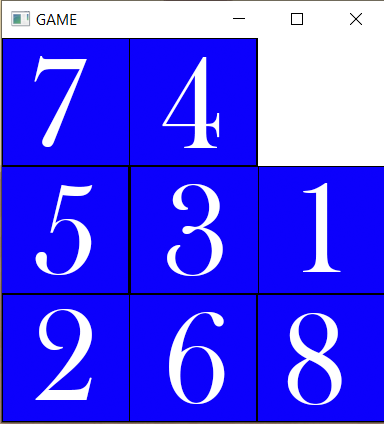
**Меню игры**



**Выбор сложности:**



**Сложность 3х3:**



**Сложность 4х4:**

