**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Программирование

**Отчет по лабораторной работе № 5**

**Тема:** «Использование сторонних библиотек. Работа с готовыми классами в C++»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМИ-150 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Поташин А.В |  |  |  |
| Принял |  |  |  |  |

**Уфа 2018**

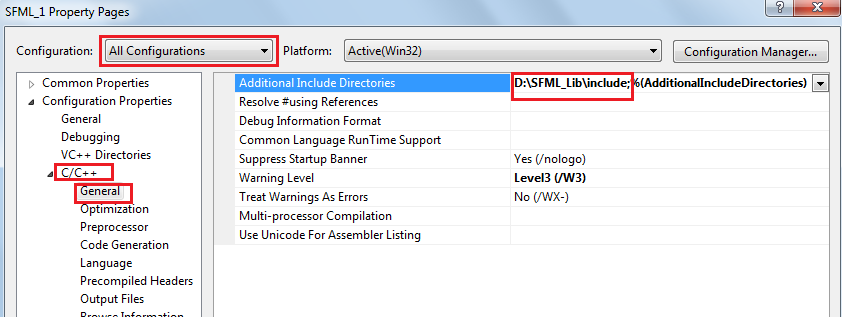
**Цель:** ознакомиться с основными приемами построения приложений с классами сторонней библиотеки.

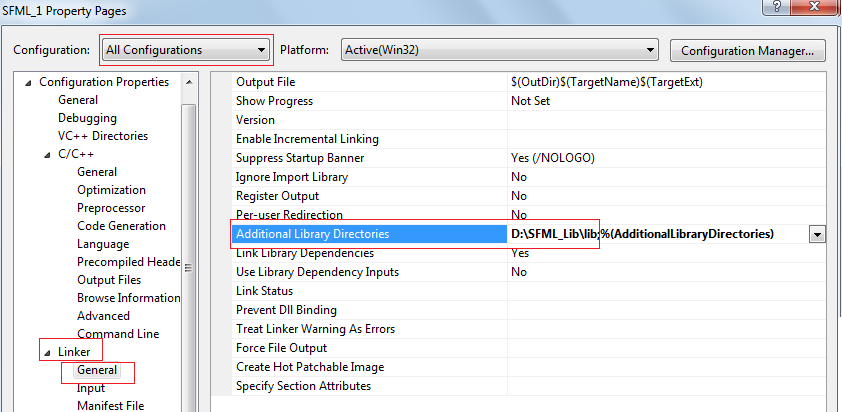
**Теоретический материал**

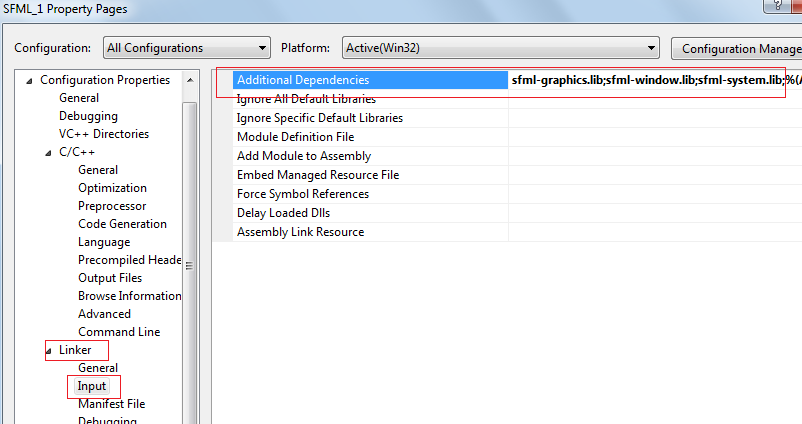
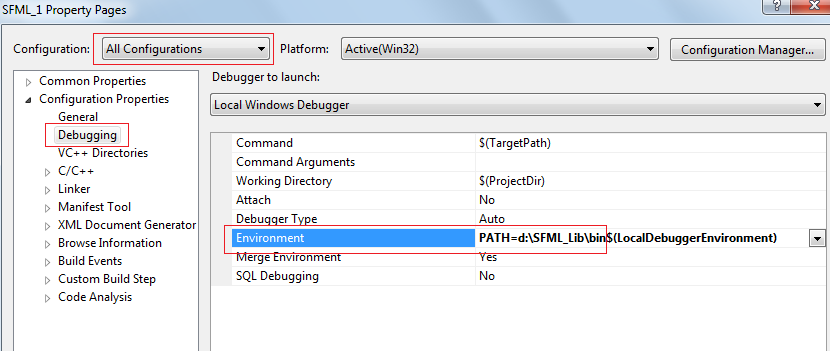
SFML— свободная кроссплатформенная мультимедийная библиотека. Написана на C++. Предоставляет методы работы с объектами (и их взаимодействием) двухмерной графики: многоугольники, эллипсы, векторы. Методы работы с графическим окном и реализации двухмерной анимации.  
  
Подключение:

1)Скачать на официальном сайте нужную версию.

2) Указать путь к файлам библиотеки в настройках проекта.





**Ответы на контрольные вопросы**

Реализация случайных событий:   
 Функция случайной генерации чисел в диапазоне путем подключения библиотек <random> И <ctime>

Реализация плавного прохождения через границу окна:

Рисование 8ми аналогов объекта, перемещение его на точку одного из копий при достижении края.

Реализация пересечений: расчет углов прямоугольных объектов внутри других прямоугольников.

Ввод/вывод: библиотека <iostream> и ее методы sin и cout

**Индивидуальное задание №1**

Задание:

Программа запрашивает у пользователя ширину и высоту графического окна (в пикселях), а также количество объектов в окне (от 4 до 10) и максимальное количество циклов отрисовки.

Все объекты в программе являются прямоугольниками, длины сторон и цвет заливки задаются случайно.

Объекты совершают следующие действия: объекты вращаются вокруг своей оси с заданной случайно угловой скоростью и перемещаются по всему полю в соответствии с случайно заданной линейной скоростью. В самом начале объекты не могут накладываться друг на друга.

Удара между объектами не происходит, они проходят друг свозь друга без какого-либо физического контакта, но пока они накладываются оба имеют определенный цвет. Объекты возвращают свой “родной” цвет при разъединении.

Объекты пролетают сквозь границу окна и появляются у противоположенной границы.

Описание программы**:**

**Основной файл:**

Блок ввода значений пользователя

Блок проверки введённых значений

Блок создания анимации (отчистка окна, рисование 9 копий каждого объекта, расчет коллизии)

**Файл класса:**

Метод: coordinate возвращает координаты точки.

Метод: collision возвращает true или false, находит координаты точки, проверяет коллизию объектов.

Метод: point\_in\_rect возвращает true или false, проверяет коллизию объектов, поворачивает оси координат.

Метод: center проверяет нахождения объекта, возвращает его внутрь окна.

Метод: random возвращает случайное значение в диапазоне.

Метод: move перемещает объект

Метод: render рисует объект

Метод: render\_shift рисует объект в 8 точках.

Исходный код программы

Основной файл:

#include <math.h>

#include <SFML\Graphics.hpp>

#include <iostream>

#include <random>

#include <ctime>

#include "Rectangles.h"

using namespace sf;

using namespace std;

int main() {

srand((unsigned int)time(NULL));

int X, Y, Frame, NumOfObj;

cout << "Please enter the width(first value) and height(second value) of the window: ";

cin >> X >> Y;

cout << "Please enter number of objects(1-30): ";

cin >> NumOfObj;

cout << "Please enter number of frames(1-60): ";

cin >> Frame;

if (X < 800) { X = 800; }

if (X > 3840) { X = 3840; }

if (Y < 600) { Y = 600; }

if (Y > 2160) { Y = 2160; }

if (Frame < 1) { Frame = 1; }

if (Frame > 60) { Frame = 60; }

if (NumOfObj > 30) { NumOfObj = 30; }

if (NumOfObj < 1) { NumOfObj = 1; }

RenderWindow window(VideoMode(X, Y), "SFML", Style::Close);

window.setFramerateLimit(Frame);

Rectangles \*arrayObj[30];

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

arrayObj[i] = new Rectangles(X, Y);

}

while (window.isOpen()) {

Event event;

while (window.pollEvent(event)) {

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

window.close();

}

}

window.clear();

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

arrayObj[i] -> collised = false;

}

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

arrayObj[i] -> move();

for (int j = i + 1; j < NumOfObj; j++) {

bool collision = arrayObj[i] -> collision(\*arrayObj[j], &window) || arrayObj[j]->collision(\*arrayObj[i], &window);

if (collision) {

arrayObj[i] -> collised = true;

arrayObj[j] -> collised = true;

}

}

arrayObj[i]->render(&window);

}

window.display();

}

delete arrayObj;

return 0;

}

Заголовочный файл класса:

#pragma once

#include <math.h>

#include <SFML\Graphics.hpp>

#include <iostream>

#include <random>

#include <ctime>

using namespace std;

using namespace sf;

class Rectangles

{

public:

float angle;

bool collised;

int random(int a, int b);

void set\_X\_coordinate(int X);

void set\_Y\_coordinate(int Y);

void move();

void render(RenderWindow \*window);

void render\_shift(RenderWindow \*window, int shiftX, int shiftY);

void center();

bool point\_in\_rect(Vector2f point, RenderWindow \*window);

bool collision(Rectangles rectan, RenderWindow \*window);

void corner(Vector2f point, Rectangles rectan, RenderWindow \*window);

void line(RenderWindow \*window, Vector2f start, Vector2f finish, Color startColor, Color finishColor);

Vector2f coordinate(float angle);

Rectangles(int X, int Y);

~Rectangles();

private:

int X\_coord, Y\_coord, width, heigth, sx, sy, r, g, b, X\_srart, Y\_start;

float sr, hypotenuse;

RectangleShape rect;

};

Файл класса:

#define \_USE\_MATH\_DEFINES

#include "Rectangles.h"

#include <math.h>

#include <SFML\Graphics.hpp>

#include <iostream>

#include <random>

#include <ctime>

using namespace sf;

using namespace std;

Rectangles::Rectangles(int X, int Y) {

set\_X\_coordinate(X);

set\_Y\_coordinate(Y);

sx = random(-2, 3);

sy = random(-2, 3);

width = random(20, 90);

heigth = random(20, 90);

sr = random(-9, 9);

r = random(50, 255);

g = random(50, 255);

b = random(50, 255);

hypotenuse = (float)sqrt(width \* width + heigth \* heigth);

angle = asinf(heigth / hypotenuse);

X\_srart = random(2 \* width, X - 2 \* width);

Y\_start = random(2 \* heigth, Y - 2 \* heigth);

rect.setSize(Vector2f((float)(2 \* width), (float)(2 \* heigth)));

rect.setOrigin((float)width, (float)heigth);

rect.setFillColor(Color::Transparent);

rect.setOutlineColor(Color(r,g,b));

rect.setOutlineThickness(1);

rect.setPosition((float)X\_srart, (float)Y\_start);

collised = false;

rect.setRotation(random(0, 90));

}

Vector2f Rectangles::coordinate(float angle) {

float x = hypotenuse \* cosf(angle + rect.getRotation() \* M\_PI / 180);

float y = hypotenuse \* sinf(angle + rect.getRotation() \* M\_PI / 180);

return Vector2f(x, y) + rect.getPosition();

}

bool Rectangles::collision(Rectangles rectan, RenderWindow \*window) {

Vector2f A = rectan.coordinate(rectan.angle);

Vector2f B = rectan.coordinate(-rectan.angle);

Vector2f C = rectan.coordinate(M\_PI - rectan.angle);

Vector2f D = rectan.coordinate(M\_PI + rectan.angle);

return point\_in\_rect(A, window) || point\_in\_rect(B, window) || point\_in\_rect(C, window) || point\_in\_rect(D, window);

}

void Rectangles::corner(Vector2f point, Rectangles rectan, RenderWindow \*window) {

line(window, rectan.rect.getPosition(), point, Color::Transparent, Color::Red);

}

bool Rectangles::point\_in\_rect(Vector2f point, RenderWindow \*window) {

float degree = rect.getRotation();

float rotation = degree \* M\_PI / 180;

rect.setRotation(0);

Vector2f relative = point - rect.getPosition();

float x = relative.x \* cosf(-rotation) - relative.y \* sinf(-rotation) + rect.getPosition().x;

float y = relative.x \* sinf(-rotation) + relative.y \* cosf(-rotation) + rect.getPosition().y;

FloatRect bounds = rect.getGlobalBounds();

bool result = bounds.contains(Vector2f(x, y));

rect.setRotation(degree);

return result;

}

void Rectangles::line(RenderWindow \*window, Vector2f start, Vector2f finish, Color startColor, Color finishColor = Color::Transparent) {

sf::Vertex line[] = {

sf::Vertex(start),

sf::Vertex(finish)

};

line[0].color = startColor;

line[1].color = finishColor;

window->draw(line, 2, sf::Lines);

}

void Rectangles::center() {

int xn = (int)rect.getPosition().x;

int yn = (int)rect.getPosition().y;

if (xn > X\_coord + hypotenuse) return rect.setPosition((float)(xn - X\_coord), (float)yn);

if (yn > Y\_coord + hypotenuse) return rect.setPosition((float)xn, (float)(yn - Y\_coord));

if (xn < -hypotenuse) return rect.setPosition((float)(xn + X\_coord), (float)yn);

if (yn < -hypotenuse) return rect.setPosition((float)xn, (float)(yn + Y\_coord));

}

int Rectangles::random(int a, int b) {

int minimum = min(a, b);

int maximum = max(a, b);

return rand() % (maximum - minimum) + minimum;

};

void Rectangles::set\_X\_coordinate(int X) {X\_coord = X;}

void Rectangles::set\_Y\_coordinate(int Y) {Y\_coord = Y;}

void Rectangles::move() {

rect.move((float)sx, (float)sy);

rect.setRotation(sr + rect.getRotation());

center();

}

void Rectangles::render(RenderWindow \*window) {

rect.setFillColor(collised ? Color(255, 255, 0, 50) : Color::Transparent);

render\_shift(window, 0, 0);

render\_shift(window, X\_coord, 0);

render\_shift(window, 0, Y\_coord);

render\_shift(window, -X\_coord, 0);

render\_shift(window, 0, -Y\_coord);

render\_shift(window, X\_coord, Y\_coord);

render\_shift(window, X\_coord, -Y\_coord);

render\_shift(window, -X\_coord, -Y\_coord);

render\_shift(window, -X\_coord, Y\_coord);

}

void Rectangles::render\_shift(RenderWindow \*window, int shiftX, int shiftY) {

rect.move((float)(sx + shiftX), (float)(sy + shiftY));

window->draw(rect);

rect.move((float)(sx - shiftX), (float)(sy - shiftY));

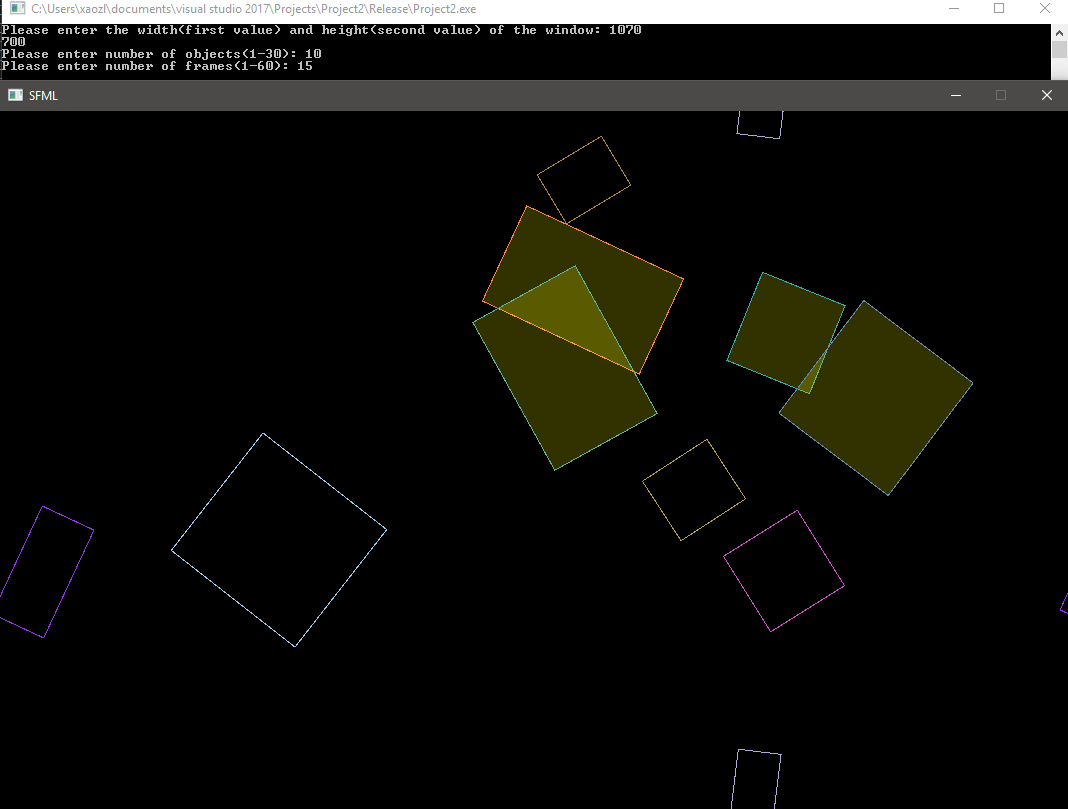
}

Rectangles::~Rectangles()

{

}

Пример выполнения программы



**Вывод**

Я научился подключать библиотеку SFML в Visual Studio. Были изучены возможности реализации двухмерной графики и анимации посредством библиотеки.

Реализованы взаимодействия (пересечения) объектов и их перенос в окне. Изучены основы ООП.

**Список использованной литературы**

В.В. Подбельский С.С. Фомин “Программирование на языке Си”.

Bikmeyev-at.ugatu.su.