

Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Ю.Е. Гагарин, А.В. Козина

БИБЛИОТЕКА SFML

Методические указания к выполнению домашней работы
по дисциплине «Высокоуровневое программирование»

Калуга – 2019

УДК 004.62
ББК 32.972.1
Г12

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационные технологии».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/9 от «10» апреля 2019 г.

Зав. кафедрой ИУ4-КФ  к.т.н., доцент Ю.Е. Гагарин


- Методической комиссией факультета ИУ-КФ протокол № Н от «29» апреля 2019 г.

Председатель методической комиссии факультета ИУ-КФ  к.т.н., доцент М.Ю. Адкин

- Методической комиссией КФ МГТУ им.Н.Э. Баумана протокол № 7 от «7» сентября 2019 г.

Председатель методической комиссии КФ МГТУ им.Н.Э. Баумана  д.э.н., профессор О.Л. Перерова

Рецензент: к.т.н., доцент кафедры ИУ3-КФ  А.В. Фиошин

Авторы к.т.н., доцент кафедры ИУ4-КФ ассистент кафедры ИУ4-КФ  Ю.Е. Гагарин
 А.В. Козина

Аннотация

Методические указания к выполнению домашней работы по курсу «Высокоуровневое программирование» содержат описание принципов работы с мультимедийной библиотекой SFML. Предназначены для студентов 1-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

© Калужский филиал МГТУ им. Н.Э. Баумана, 2019 г.
© Ю.Е. Гагарин, А.В. Козина, 2019 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ	6
ПОДКЛЮЧЕНИЕ БИБЛИОТЕКИ SFML К СРЕДЕ VISUAL STUDIO	8
ОСНОВНОЙ ПРИНЦИП РАБОТЫ SFML, ОБЯЗАТЕЛЬНЫЕ ФУНКЦИИ SFML.....	13
ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ	23
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ	28
ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ.....	29
ОСНОВНАЯ ЛИТЕРАТУРА.....	30
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	30

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Высокоуровневое программирование» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 1-го курса направления подготовки 09.03.04 «Программная инженерия», содержат описание принципов работы с мультимедийной библиотекой SFML.

Методические указания составлены для ознакомления студентов с основополагающими понятиями и принципами разработки программ с использованием мультимедийной библиотеки SFML на языке C++. Для выполнения домашней работы студенту необходимы минимальные навыки программирования.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения домашней работы является формирование практических навыков реализации графических программ с использованием библиотеки SFML.

Основными задачами выполнения домашней работы являются:

1. Познакомиться с разработкой графических программ на языке программирования C++;
2. Изучить основные процедуры и функции библиотеки SFML;

Результатами работы являются:

1. Разработанные алгоритмы программ индивидуального задания в виде блок-схем;
2. Реализация разработанных алгоритмов на языке программирования C++;
3. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

SFML (англ. Simple and Fast Multimedia Library — простая и быстрая мультимедийная библиотека) — свободная кроссплатформенная мультимедийная библиотека. Написана на C++, но доступна также для C, C#, .Net, D, Java, Python, Ruby, OCaml, Go и Rust.

SFML содержит ряд модулей для простого программирования игр и мультимедиа приложений. Исходный код библиотеки предоставляется под лицензией zlib/png license.

Модули

В настоящее время доступны следующие модули:

- System — управление временем и потоками, он является обязательным, так как все модули зависят от него.
- Window — управление окнами и взаимодействием с пользователем.
- Graphics — делает простым отображение графических примитивов и изображений.
- Audio — предоставляет интерфейс для управления звуком.
- Network — для сетевых приложений.

Простейшая программа

Следующий код на языке C++ демонстрирует простейшее приложение на SFML (отображение окна и заливка его синим цветом):

```
// Подключение заголовка модуля Graphics, который
автоматически подключает заголовок модуля Window
#include <SFML/Graphics.hpp>

int main()
{
    // создаём окно
```

```

    sf::RenderWindow app(sf::VideoMode(800, 600, 32),
        "Hello World - SFML");

    // основной цикл
    while (app.isOpen())
    {
        // проверка события (нажатие кнопки, закрытие
        окна и т.д.)
        sf::Event event;
        while (app.pollEvent(event))
        {
            // если событие "закрытие окна":
            if (event.type == sf::Event::Closed)
                // закрываем окно
                app.close();
        }

        // очистка экрана и заливка его синим цветом
        app.clear(sf::Color(0,0,255));

        // отображаем на экран
        app.display();
    }

    return 0;
}

```

ПОДКЛЮЧЕНИЕ БИБЛИОТЕКИ SFML К СРЕДЕ VISUAL STUDIO

Сначала необходимо скачать библиотеку по ссылке <http://www.sfml-dev.org/download/sfml/2.2/>. Далее нужно распаковать файлы, например D:/SFML-2.2/ (старайтесь избегать русских букв в пути к папке).

Запускаем Visual Studio, создаем проект. Добавляем исходный код – для этого слева в обозревателе решений находим вкладку «Файлы исходного кода» → правый клик → добавить → создать элемент → файл C++ (имя main.cpp) → добавить. (имя может быть любым, но опять же лучше всего английские буквы и главный файл называть "main").

Далее вставляем следующий код в файл main.cpp:

```
#include <SFML/Graphics.hpp>

int main()
{
    sf::RenderWindow window(sf::VideoMode(200, 200),
"sfm1test");
    sf::CircleShape shape(100.f);
    shape.setFillColor(sf::Color::Green);

    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        window.clear();
        window.draw(shape);
        window.display();
    }
}
```



```
    return 0;
}
```

Для того, чтобы всё это работало – необходимо подключить библиотеку [SFML](#).

Заходим вверху "проект" → свойства "имя проекта" → свойства конфигурации → C++ → общие.

В пункте «дополнительные каталоги включаемых файлов» прописываем путь до ранее распакованного архива с библиотекой (D:\SFML-2.2\include).

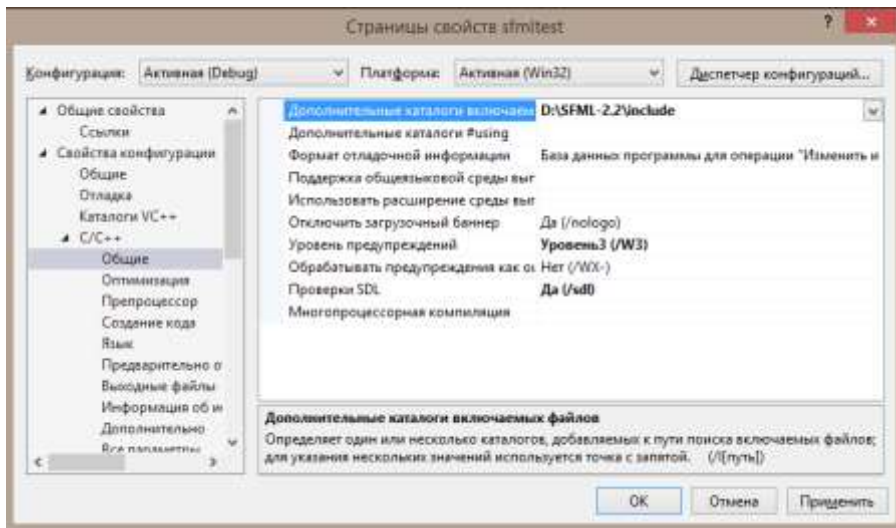


Рис. 1. Параметр «Дополнительные каталоги включаемых файлов»

Затем идем в компоновщик → общие и в пункте «дополнительные каталоги библиотек» прописываем путь к папке lib (D:\SFML-2.2\lib).

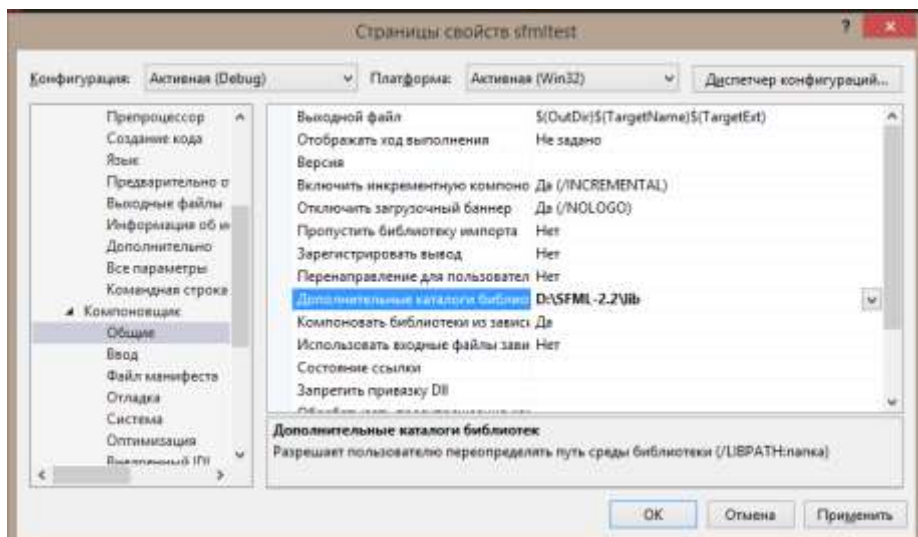


Рис. 2. Параметр «Дополнительные каталоги библиотек»

Далее идем в компоновщике во вкладку ввод и в первой строке «дополнительные зависимости» вписываем перед всеми библиотеками следующую строку:

sfml-graphics-d.lib;sfml-window-d.lib;sfml-system-d.lib;sfml-audio-d.lib;

Запустим и получим ошибку. Зато соберется проект и появится папка debug.

Теперь идем в папку "bin", которую скачали и распаковали с библиотекой (D:\SFML-2.2\bin\) и копируем всё содержимое этой папки (dll файлы, например sfml-graphics-2.dll) в папку с вашим проектом, в папку debug (после сборки появилась эта папка), но не в тот debug, где лежит файл с исходным кодом "main.cpp", а тот, где появляется exe файл. (C:\...\sfmltest\debug\)

В эту же папку копируем и файлы: msvcp100d.dll; msvcp110.dll; msvcp110d.dll; msucr100d.dll; msvcr110.dll; msvcr110d.dll (при необходимости скачать).

Теперь проект можно запустить и если вы всё сделали правильно, то увидите зелёный круг как на рисунке ниже:



Рис. 5. Результат выполнения программы

ОСНОВНОЙ ПРИНЦИП РАБОТЫ SFML, ОБЯЗАТЕЛЬНЫЕ ФУНКЦИИ SFML

Первой строкой кода необходимо подключить заголовочный файл, отвечающий за работу с графикой:

```
#include <SFML/Graphics.hpp>
```

Например, когда необходимо использовать звуки в игре — подключаем Audio.hpp

Затем идёт главная функция языка C++ , которая должна присутствовать в каждой программе, написанной на C++. Выглядит в нашем случае она следующим образом:

```
int main()
{
    return 0;
}
```

Уже в теле этой функции происходят вызовы других функций. У SFML тоже есть обязательные функции и объекты. Например будем использовать объект, отвечающий за появление окна нашей игры. Выглядит он так:

```
sf::RenderWindow window(sf::VideoMode(200, 200),
    "SFMLtest");
```

Разберем по порядку эту строчку. sf:: это пространство имён библиотеки SFML. Чтобы использовать функции и создавать объекты, относящиеся к этой библиотеке, необходимо перед началом каждой строки писать sf:: как в этом случае. Чтобы этого не делать каждый раз — можно написать такую строку перед int main()

```
using namespace sf;
```

RenderWindow window создаёт как бы переменную (объект) окна с названием "window". Затем в скобках идёт конструктор с параметрами. Первый параметр VideoMode(200, 200) — размер окна, далее идёт строка "SFML works!", являющаяся заголовком окна. После этого можно использовать необязательные параметры, например sf::Style::Fullscreen, что сделает окно развернутым на весь экран, как полноценную игру.

```
sf::RenderWindow window(sf::VideoMode(200, 200),
"SFML works!",sf::Style::Fullscreen);
```

После того, как создали окно, нужно что-нибудь нарисовать.

```
sf::CircleShape shape(100.f);
shape.setFillColor(sf::Color::Red);
```

Первая строка создали объект shape класса CircleShape. В скобках указан размер. Вторая строка — залили круг красным цветом.

Далее идёт обязательный цикл для SFML, который можете переводить для понимания как: "Пока окно открыто – выполняй то, что внутри цикла":

```
while (window.isOpen())
{
    sf::Event event;
    while (window.pollEvent(event))
    {
        if (event.type == sf::Event::Closed)
            window.close();
    }
    window.clear();
    window.draw(shape);
    window.display();
}
```

По сути этот цикл бесконечен и выход из него только один – закрыть окно программы.

Чтобы окно могло закрыться ниже идет 6 строк кода. Итак, есть некое "SFML-евское" событие event, строка — `sf::Event event`;

Это событие принимает состояние закрытого, когда закрываем окно и попросту говоря условие спрашивает: если событие event приняло значение Closed, то программа закрывает окно. Такие вот необходимые условности.

После этого идут функции:

`window.clear()`; – очищает экран.

`window.draw(shape)`; – рисует объект.

`window.display()`; – всё это показывает.

Имеем [бесконечный цикл](#), в котором очень быстро первым делом очищается экран, рисуется объект, затем всё это отображается и так происходит, пока не будет закрыто окно. Можно привести аналогию с блокнотом и анимацией из детства. Вы открываете блокнот (окно игры), он уже очищен для рисования первого слайда, вы рисуете картинку, когда нарисовали можете её кому то показать. потом всё это повторяется, когда вы перелистываете лист блокнота и рисуете другой слайд для анимации – ваш лист опять чист, вы опять что-то рисуете, опять можете показать кому-то. изрисовав весь блокнот, вы можете его пролистывать и наблюдать анимацию вашего рисунка. Представьте, что вы пришли к другу и пролистываете этот блокнот бесконечно, пока он не скажет вам хватит.

Общие свойства фигур

Цвет

Одно из главнейших свойств фигур — это их цвет, который можно менять с помощью метода `setFillColor()`.

```
sf::CircleShape shape(50);  
// задаём фигуре зелёный цвет  
shape.setFillColor(sf::Color(100, 250, 50));
```

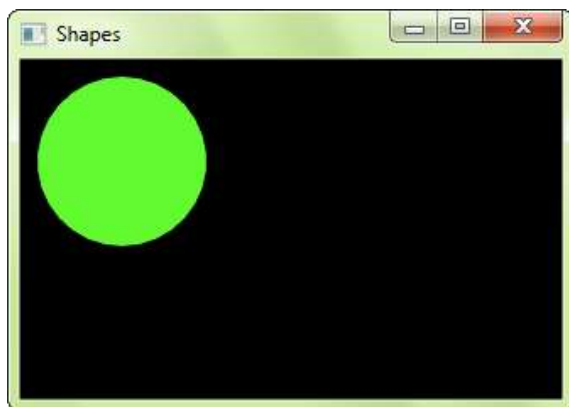


Рис. 6. Цвет

Контур

Фигуры могут иметь контур. Вы можете задать толщину и цвет контура методами `setOutlineThickness()` и `setOutlineColor()`.

```
sf::CircleShape shape(50);  
shape.setFillColor(sf::Color(150, 50, 250));  
// задаём контур толщиной 10 пикселей оранжевого цвета  
shape.setOutlineThickness(10);  
shape.setOutlineColor(sf::Color(250, 150, 100));
```

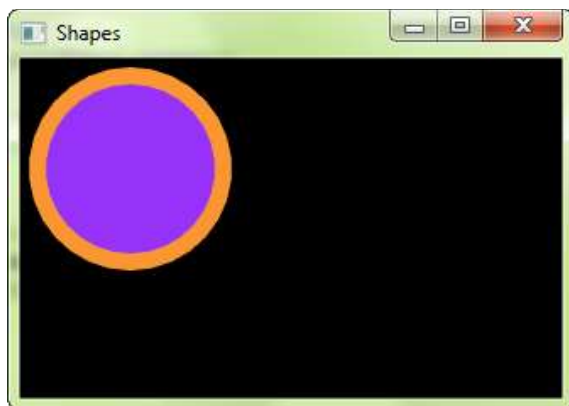


Рис. 7. Контур

По умолчанию контур вырисовывается снаружи фигуры (если у вас есть круг с радиусом 10 и контур толщиной 5, то вы получите круг с радиусом 15). Контур можно применять и по направлению к центру фигуры, для этого необходимо задавать отрицательные значения толщины.

Для того что бы убрать контур, необходимо задать нулевую толщину. Если же вам нужен только контур, то в функции `setFillColor()` следует задать в качестве аргумента цвет `sf::Color::Transparent`.

Рисование фигуры

Рисование фигур такое же простое как и рисование других объектов SFML:

```
window.draw(shape);
```

Типы форм

Прямоугольник

Для рисования прямоугольников необходимо использовать класс `sf::RectangleShape`. Он имеет только одно свойство: размер прямоугольника.

```
// определяем прямоугольник размером 120x50
sf::RectangleShape rectangle(sf::Vector2f(120, 50));

// меняем размер 100x100
rectangle.setSize(sf::Vector2f(100, 100));
```

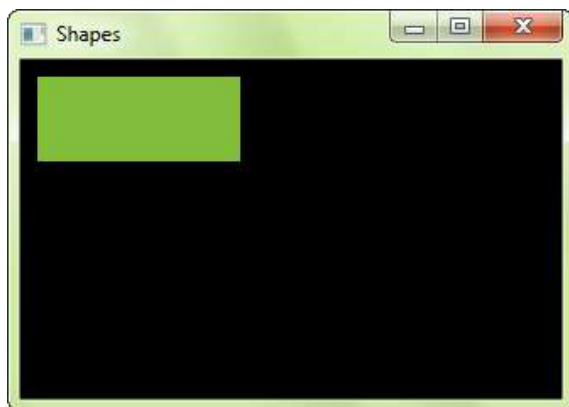


Рис. 8. Прямоугольник

Круг

Круг представлен классом `sf::CircleShape`. Класс имеет два свойства: радиус и количество граней. Количество граней является дополнительным свойством, позволяющим настроить качество отображения круга: круги имитируются многоугольниками с большим числом граней (видеокарты просто не способны рисовать идеальные круги), в общем это свойство устанавливает число граней. Если вы рисуете маленькие круги, то вам нужно совсем немного граней что бы они отображались достаточно гладкими. Обратное тоже верно — чем больше круг, тем больше нужно граней.

```
// определяем круг с радиусом = 200
sf::CircleShape circle(200);

// меняем радиус на 40
circle.setRadius(40);

// меняем число граней на 100
circle.setPointCount(100);
```

Правильные многоугольники

На самом деле для них нет отдельного класса, да он и не нужен. Воспользовавшись классом [sf::CircleShape](#) и установив число граней равным 3, мы получим треугольник, если взять 4 грани, то получится квадрат и т.д.

```
// определяем треугольник
sf::CircleShape triangle(80, 3);

// определяем квадрат
sf::CircleShape square(80, 4);

// определяем восьмиугольник
sf::CircleShape octagon(80, 8);
```

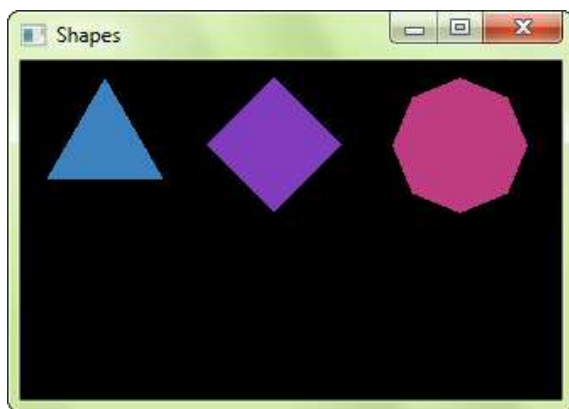


Рис. 9. Правильные многоугольники

Выпуклые фигуры

Класс `sf::ConvexShape` является последним из классов фигур: он позволяет определить фигуру, до тех пор пока она является выпуклой. Действительно, SFML не способен нарисовать вогнутые фигуры; если такие нарисовать необходимо, то их всегда можно разбить на несколько выпуклых.

Для определения выпуклой фигуры для начала необходимо указать общее число точек-вершин, и только потом определить эти точки.

```
// создаём пустую фигуру
sf::ConvexShape convex;

// изменяем размер на 5 вершин
convex.setPointCount(5);

// определяем вершины
convex.setPoint(0, sf::Vector2f(0, 0));
convex.setPoint(1, sf::Vector2f(150, 10));
convex.setPoint(2, sf::Vector2f(120, 90));
convex.setPoint(3, sf::Vector2f(30, 100));
convex.setPoint(4, sf::Vector2f(0, 50));
```

Очень важно определять вершины по часовой или против часовой стрелки, если определять их в произвольном порядке, то результат может быть непредсказуем.

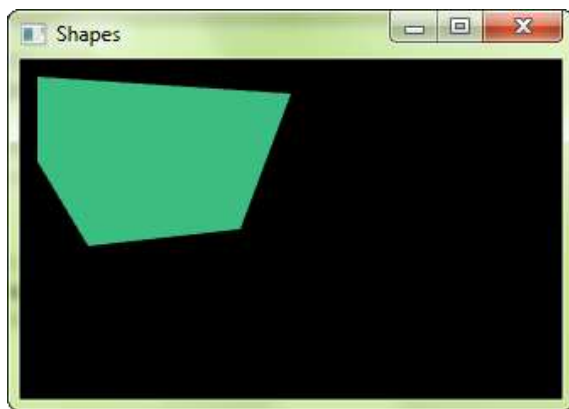


Рис. 10. Выпуклая фигура

Формально, `sf::ConvexShape` позволяет создавать только выпуклые фигуры. На деле же требования немножко мягче. На самом деле ваша

форма должна соответствовать следующему критерию: любая из линий проведённых от центра тяжести фигуры до вершины, не должна пересекать границу фигуры. Следуя этому правилу можно, например, рисовать звёзды.

Линии

Для линий нет отдельного класса. Причина очень простая: если у линии есть толщина, используется фигура [прямоугольник](#), если толщины нет, используется примитив линия.

Линия с толщиной:

```
sf::RectangleShape line(sf::Vector2f(150, 5));  
line.rotate(45);
```

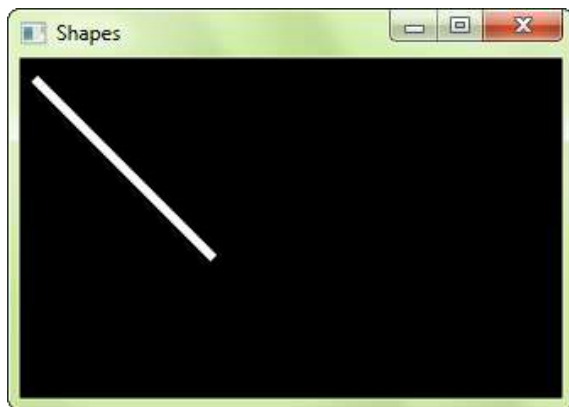


Рис. 11. Линия

Линия без толщины:

```
sf::Vertex line[] =  
{  
    sf::Vertex(sf::Vector2f(10, 10)),  
    sf::Vertex(sf::Vector2f(150, 150))  
};
```

```
window.draw(line, 2, sf::Lines);
```

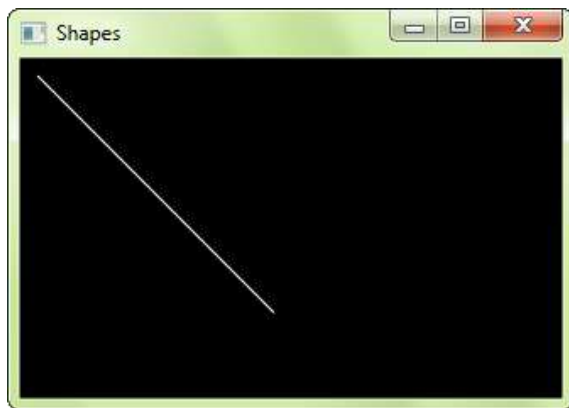


Рис. 12. Линия без толщины

ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ

Вариант 1

Задание 1. Напишите программу вычерчивания дуги окружности по заданному радиусу и координатам центра окружности. Параметры дуги могут вводиться в форме координат начальной и конечной точек или углов, соответствующих началу и концу дуги.

Задание 2. Преобразовать дугу в сектор и закрасить. Сектор должен вращаться по часовой стрелке и перемещаться вдоль экрана.

Вариант 2

Задание 1. Напишите программу, которая строит по периметру экрана семейство разноцветных квадратов, а в середине – закрашенную окружность.

Задание 2. Окружность должен исчезать и появляться в различных местах экрана.

Вариант 3

Задание 1. Построить семейство одинаковых окружностей, центры которых лежат на окружности большего диаметра.

Задание 2. Окружность должна пульсировать, т. е. периодически стягиваться в точку и расширяться до внешней.

Вариант 4

Задание 1. Разработайте программу для вычерчивания различных осей координат и графика синусоиды $Y = A \sin(fX - d)$, где A – амплитуда, f – частота, X – интервал, d – смещение. Программа должна работать при любых введенных значениях A , f , d . Воспользуйтесь масштабированием и постройте три периода синусоиды в интервале от

$$X = -\frac{d}{f} \text{ до } X = \frac{(6\pi - d)}{f}.$$

Задание 2. Коэффициент A меняется по закону $A = I + \cos(t)$, где t – время. Закрасьте площадь между осью X и линией, обозначающей значение функции.

Вариант 5

Задание 1. Разработайте программу для отображения набора данных в виде небольших окружностей, координаты центров которых равны значениям данных из набора. Внутренние области окружностей закрасьте разными цветами. При этом должны высвечиваться размеченные оси координат.

Задание 2. Окружности должны пульсировать, т. е. периодически стягиваться в точку и расширяться до максимального значения. Период пульсаций у различных окружностей должен быть различен.

Вариант 6

Задание 1. Напишите программу для построения изображения машины (вид сбоку). Нанесите штриховку или раскрасьте рисунок.

Задание 2. Измените программу, чтобы машина перемещалась в заданную область экрана. Скорость перемещения и координаты области задайте с клавиатуры.

Вариант 7

Задание 1. Напишите программу для построения изображения машины (вид сбоку). Нанесите штриховку или раскрасьте рисунок.

Задание 2. Поместите машину на дорогу с штриховой осевой линией. Имитируйте передвижение автомобиля с помощью перемещения штриховой осевой линии дороги.

Вариант 8

Задание 1. Разработайте программу для построения эллипса. Нанесите меридианы. Промежутки закрасьте различными цветами.

Задание 2. Эллипс должен пульсировать, т. е. должно меняться отношение осей эллипса во времени.

Вариант 9

Задание 1. Напишите программу для вычерчивания игрального кубика. Невидимые ребра кубика рисовать пунктиром. Грани закрасьте различными цветами.

Задание 2. Показать вращение кубика вокруг оси, параллельной оси Z.

Вариант 10

Задание 1. Напишите программу для высвечивания циферблата часов. Начертите стрелки часов: секундную, минутную, часовую.

Задание 2. Показать движение стрелок часов: при повороте секундной стрелки на 360° , минутная поворачивается на 6° . Часовая стрелка поворачивается на 30° за полный оборот минутной стрелки.

Вариант 11

Задание 1. Напишите программу для высвечивания части солнечной системы: Солнце, Венера, Земля, Луна, Марс. Нанесите траектории движения. Элементы схемы раскрасьте разными цветами.

Задание 2. Приведите в движение схему солнечной системы.

Вариант 12

Задание 1. Нарисуйте круговую диаграмму успеваемости студентов группы. Входные данные: число студентов получивших – «отлично», - «хорошо», - «удовлетворительно», - «неудовлетворительно». Секторы раскрасить разными цветами.

Задание 2. Покажите изменение успеваемости по 5 предметам. Изменение размеров секторов должно быть непрерывным.

Вариант 13

Задание 1. Напишите программу для высвечивания баскетбольного поля (вид сверху). Разместите на нем произвольным образом по 5 игроков каждой команды.

Задание 2. Покажите движение мяча. Игроки каждой команды передают мяч своему партнеру, последний игрок забрасывает мяч в корзину.

Вариант 14

Задание 1. Напишите программу для высвечивания велосипеда на наклонной плоскости. Колеса разделите на 4 сектора. Сектора закрасьте разными цветами.

Задание 2. Покажите движение велосипеда по наклонной плоскости.

Вариант 15

Задание 1. Напишите программу для высвечивания ветряной мельницы. Элементы схемы закрасьте разными цветами.

Задание 2. Покажите движение крыльев ветряной мельницы.

Вариант 16

Задание 1. Напишите программу для высвечивания прямоугольника и Г-образной фигуры, состоящих из 4-х квадратов (10x10 мм). Квадраты раскрасьте разными цветами.

Задание 2. Измените программу, чтобы командами с клавиатуры можно было поворачивать фигуры на 90° и перемещать их в любой место экрана.

Вариант 17

Задание 1. Напишите программу для высвечивания шахматной доски. Нанесите цифры и буквы. Клетки раскрасьте.

Задание 2. Нанесите на доску изображение коня и короля. Имитируйте перемещение фигур. Конь должен оказаться в выигрышном положении.

Вариант 18

Задание 1. Напишите программу для высвечивания дорожного знака «Пешеходный переход».

Задание 2. Имитировать движение пешехода.

Вариант 19

Задание 1. Напишите программу для высвечивания поля бильярда с лузами. На поле в случайном порядке расположите 4 шара. Поле закрасьте зеленым, а шары – желтым цветом.

Задание 2. Командами с клавиатуры задавать номер шара и направление движения. При попадании шара в лузу - начисляются очки, и шар выставляется возле борта, при попадании в другой шар – все исчезает.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение SFML.
2. Перечислите модули, содержащиеся в библиотеке SFML.
3. Сформулируйте порядок установки и добавления библиотеки SFML.
4. Назовите заголовочный файл, отвечающий за работу с графикой:
5. Опишите структуру SFML программы.
6. Приведите пример кода для рисования круга.
7. Приведите пример кода для рисования прямоугольника.
8. Приведите пример кода для рисования линии.
9. Приведите пример кода для рисования многоугольника.
10. Приведите пример кода для замены цвета фигуры.

ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ

На выполнение домашней работы отводится 8 академических часов: 7 часов на выполнение и сдачу домашней работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Составить блок-схему(ы) к задаче(ам) и реализовать её(их) в среде Microsoft Visio.
4. Разработать программу(ы) согласно варианту.
5. Выполнить тестирование программ(ы).
6. Продемонстрировать работу программ(ы) преподавателю.
7. Оформить отчет.
8. Защитить выполненную работу у преподавателя.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Васильев А.Н. Самоучитель С++ с примерами и задачами [Электронный ресурс]/ Васильев А.Н.— Электрон. текстовые данные. СПб.: Наука и Техника, 2017. 480 с. Режим доступа: <http://www.iprbookshop.ru/73049.html>
2. Зоткин С.П. Информатика. Программирование на языке высокого уровня С/С++. Конспект лекций [Электронный ресурс]: конспект лекций/ Зоткин С.П.— Электрон. текстовые данные. — М.: МИСИ-МГСУ, ЭБС АСВ, 2018. 140 с. Режим доступа: <http://www.iprbookshop.ru/76390.html>

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

3. Белева Л.Ф. Программирование на языке С++ [Электронный ресурс]: учебное пособие/ Белева Л.Ф.— Электрон. текстовые данные.— Саратов: Ай Пи Эр Медиа, 2018.— 81 с.— Режим доступа: <http://www.iprbookshop.ru/72466.html>.— ЭБС «IPRbooks»
4. Фридман, А.Л. Язык программирования Си++ [Электронный ресурс] : учебное пособие / А.Л. Фридман. — Электрон. дан. — Москва : , 2016. — 218 с. — Режим доступа: <https://e.lanbook.com/book/100541>. — Загл. с экрана.

Электронные ресурсы:

5. Электронно-библиотечная система <http://www.iprbookshop.ru>
6. Электронно-библиотечная система <http://e.lanbook.com>
7. Видеокурс
https://www.youtube.com/playlist?list=PL0lO_mIqDDFXNfqIL9PHQ_M7Wg_kOtDZsW
8. Руководство по языку программирования С++
<https://metanit.com/cpp/tutorial/>