**Федеральное государственное бюджетное образовательное**

**учреждение «Сибирский государственный университет**

**телекоммуникаций и информатики»**

**кафедра ПМ и К**

**КУРСОВАЯ РАБОТА**

**Тема: «Круговое движение с вращением произвольного составного графического объекта»**

Выполнил:

студент 2 курса факультета ИВТ группы ИП-712

Рещиков А.Е.

Проверил:

ассистент кафедры ПМ и К

Суходоева Н.Н.

Новосибирск – 2018

Оглавление

[Постановка задачи 3](#_Toc531290180)

[Технологии ООП 4](#_Toc531290181)

[Структура классов 5](#_Toc531290182)

[Алгоритм и результат работы 6](#_Toc531290183)

[Заключение 8](#_Toc531290184)

[Приложение. Листинг 9](#_Toc531290185)

## Постановка задачи

Необходимо написать программу, используя объектно-ориентированный подход, которая будет реализовать круговое движение с вращением произвольного составного графического объекта.

Программа написана при помощи графической библиотеки *SFML*, функциями библиотеки реализованы все графические объекты и функции движения.

За основу составного графического объекта были взяты простые фигуры, а именно:

1. Окружность
2. Эллипс
3. Прямоугольник
4. Треугольник

Из этих геометрических объектов был составлен графический объект, похожий на дом, рядом с которым есть облако ([рис.1](#Рисунок1)).

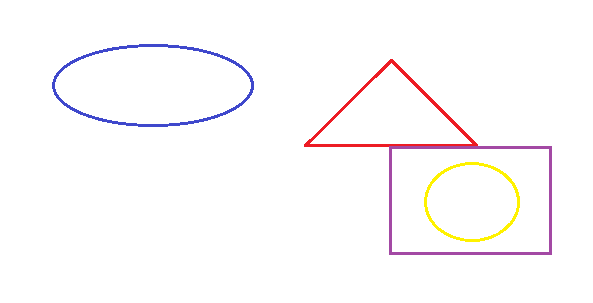


Рисунок 1

# Технологии ООП

В курсовой работе были применены такие принципы объектно-ориентированного программирования, как:

1. Инкапсуляция
2. Полиморфизм
3. Наследование

Из технологий ООП были применены:

1. Дружественные функции
2. Чистые виртуальные функции
3. Полиморфный класс
4. Абстрактный класс
5. Конструктор, перегрузка конструкторов
6. Списки инициализации
7. Массив указателей на объекты

### Структура классов

Основная структура классов ([рис.2](#Рисунок2)).

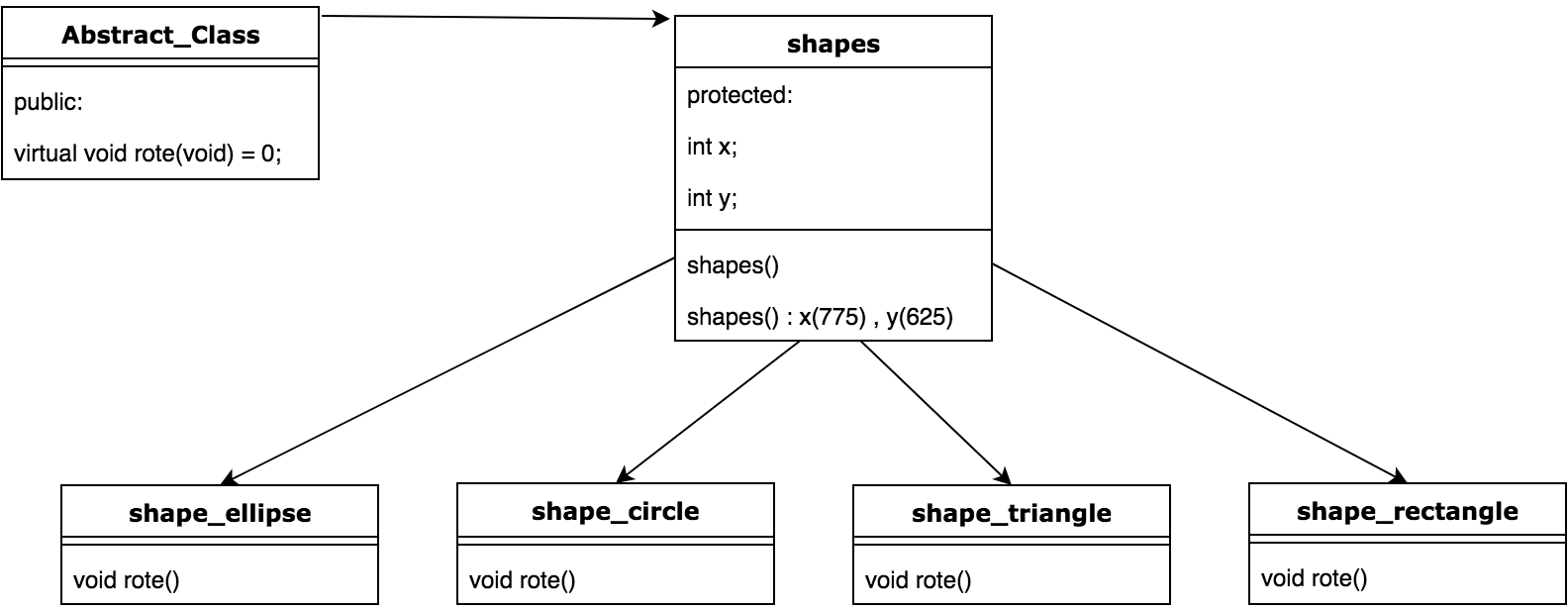
**[](https://www.draw.io/?page=4&scale=2" \l "G1pvqtEFRZxu9l8G2hiOD057-xfM1tQSLy)**

Рисунок 2

В программе реализован абстрактный класс (*Abstract\_Class*), который содержит в себе чистую виртуальную функцию, созданную для движения составной фигуры.

Этот класс является родительским для класса *shapes*, который содержит в себе инкапсулированные переменные, имеющие тип доступа *protected*, олицетворяющие координаты *x, y,* необходимые для реализации движения составной фигуры. Также этот класс содержит в себе перегрузку конструкторов, один из которых имеет список инициализации.

От класса *shapes* берут происхождение берут классы со схожими названиями, описанные в структуре ([рис. 2](#Рисунок2)).

Каждый из классов, наследников *shapes* реализует движение, использую полиморфный метод (функцию *rote*), в котором создаётся движение по окружности.

# Алгоритм и результат работы

В программе имеется 6 классов. Классами наследниками полиморфно реализуется функция движения. В главной части программы при помощи массивов объектов инициализируются фигуры, входящие в состав основной. Базовые фигуры и метод движения реализуется функциями библиотеки *SFML*.

Для начала создается абстрактный класс (*Abstract\_Class*), содержащий чистый виртуальный метод (*rote*), в котором потом будет реализовано движение по окружности.

Далее создаётся наследник от абстрактного класса, класс *shapes*, содержащий в себе инкапсулированные переменные (*protected*), необходимые для хранения координат. Также в этом классе реализуется перегрузка конструкторов, в которых создается центр, вокруг которого будет вращаться составная фигура. Один из конструкторов содержит списки инициализации.

От класса *shapes* происходят в свою очередь потомки, каждый из которых отвечает за движение каждой фигуры, входящей в состав основной, а именно:

* + - 1. *shape\_ellipse*, отвечающая за движение эллипса.
      2. *shape\_circle*, отвечающая за движение окружности.
      3. *shape\_triangle*, отвечающая за движение треугольника.
      4. *shape\_rectangle*, отвечающая за движение квадрата.

Каждый из классов полиморфно реализует метод абстрактного класса для движения по окружности, задавая разный интервал для гладкого движения простых фигур, создавая из них составную. Движение задаётся функциями графической библиотеки *SFML*.

В дружественной функции *Cord*, дружественной для *shapes\_ellipse* реализуется вывод сообщения о том, как прервать работу программы.

Открыв и скомпилировав программу, на экране появляется окно, в котором мы видим составную фигуру ([рис. 3](#Рисунок3)).

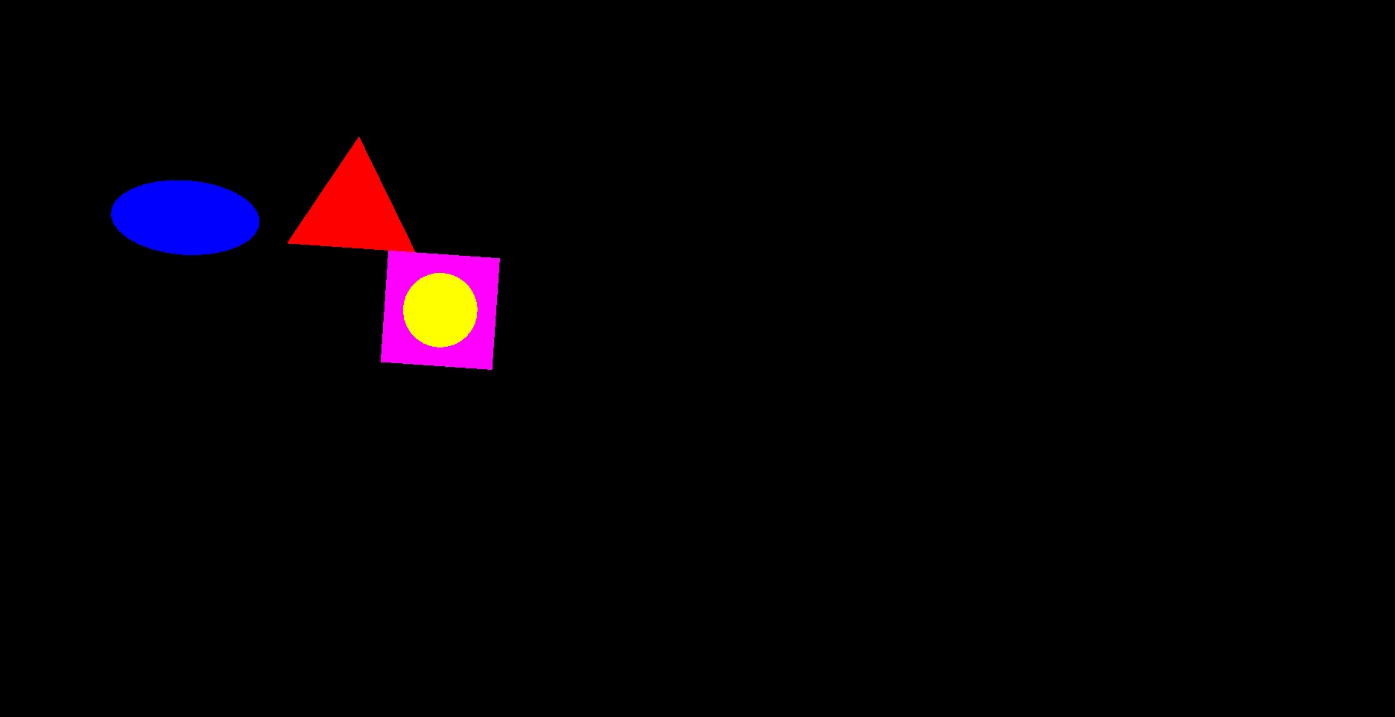


Рисунок 3

Через несколько секунд фигура находится уже совершенно в другом положении, о чем свидетельствует следующее изображение ([рис. 4](#Рисунок4)).

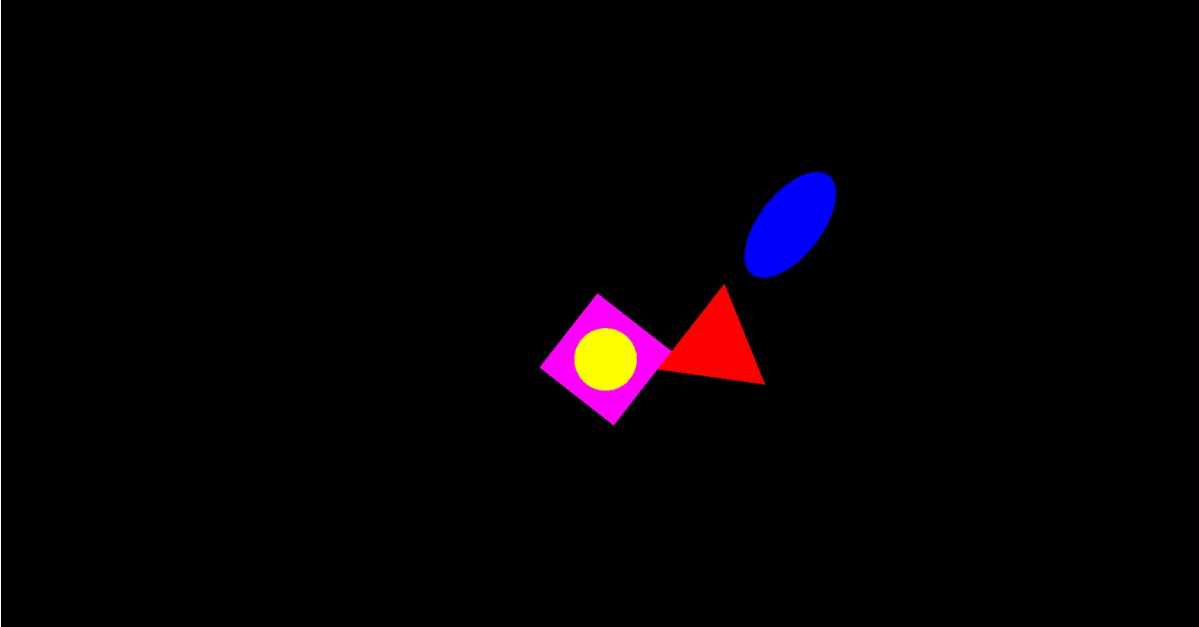


Рисунок 4

# Заключение

При работе использовались такие принципы ООП, как наследование, полиморфизм и инкапсуляция. Наследование – конструирование новых, более сложных производных классов-потомков, из уже имеющихся базовых классов-родителей, с помощью добавления новых полей и/или методов. Полиморфизм - механизм, обеспечивающий возможность определения различных описаний некоторого единого метода (единого по названию) для классов различного уровня иерархии. Инкапсуляция – механизм скрытия данных внутри объекта.

Для создания объектов классов в программе существуют конструкторы, была реализована их перегрузка. Также использовалась дружественная функция, чистый виртуальный метод и абстрактный класс. Дружественные функции – функции, которые не являются компонентом некоторого класса, имеющие доступ ко всем его компонентам. Абстрактный класс – класс, в котором есть хотя бы один чистый (обнуленный) виртуальный метод. Он служит основой для производных классов. Чистые виртуальные функции – функции, которые объявляются в базовом классе как виртуальные и не содержащие описания выполняемых действий.

В главный части программного кода создан массив объектов типа shapes, который является родительским для классов фигур. При помощи этого массива и вызываются методы, созданные для движения.

Таким образом было реализовано круговое движение с вращением произвольного составного графического объекта с применением технологий объектно-ориентированного программирования.

# Приложение. Листинг

Листинг кода из файла main.cpp:

#include <SFML/Graphics.hpp>

#include <time.h>

#include <conio.h>

#include "header.hpp"

sf::RenderWindow window(sf::VideoMode(1980, 1200), "Consisted figure");

int main(void)

{

shape\_ellipse a;

shapes \*array[4];

array[0] = new shape\_triangle;

array[1] = new shape\_ellipse;

array[2] = new shape\_rectangle;

array[3] = new shape\_circle;

ellipse.setFillColor(sf::Color::Blue);

circle.setFillColor(sf::Color::Yellow);

rectangle.setFillColor(sf::Color::Magenta);

triangle.setFillColor(sf::Color::Red);

Cord(a);

while (window.isOpen() and (!kbhit ()))

{

sf::Event event;

while (window.pollEvent(event))

{

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

window.close();

}

sf::sleep(sf::milliseconds(10));

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

array[i]->rote();

window.draw(triangle);

window.draw(ellipse);

window.draw(rectangle);

window.draw(circle);

window.display();

window.clear();

}

return 0;

}

Листинг кода из файла header.hpp:

#include <iostream>

#ifndef HEADER\_HPP

#define HEADER\_HPP

sf::CircleShape ellipse(50);

sf::CircleShape circle(50);

sf::RectangleShape rectangle(sf::Vector2f(150, 150));

sf::CircleShape triangle(100, 3);

class Abstract\_Class

{

public:

virtual void rote(void) = 0;

};

class shapes: public Abstract\_Class

{

protected:

int x;

int y;

public:

explicit shapes(int cx, int cy)

{

x = cx;

y = cy;

}

explicit shapes() : x(775), y(625)

{

x = 775;

y = 625;

}

};

class shape\_ellipse: public shapes

{

using shapes::shapes;

void rote(void)

{

ellipse.setScale(2, 1);

ellipse.setOrigin(300, 300);

ellipse.setPosition(x, y);

ellipse.rotate(2);

}

};

class shape\_circle: public shapes

{

using shapes::shapes;

void rote(void)

{

circle.setOrigin(200, 200);

circle.setPosition(x, y);

circle.rotate(2);

}

};

class shape\_rectangle: public shapes

{

using shapes::shapes;

void rote(void)

{

rectangle.setOrigin(225, 225);

rectangle.setPosition(x, y);

rectangle.rotate(2);

}

};

class shape\_triangle: public shapes

{

using shapes::shapes;

void rote(void)

{

triangle.setOrigin(375, 375);

triangle.setPosition(x, y);

triangle.rotate(2);

}

};

void Cord(shape\_ellipse a)

{

std::cout << "Press any key to break the program.\n";

}

#endif