Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМ и К

Курсовая работа

По дисциплине: «Объектно-ориентированное программирование»

Тема: «Круговое движение с вращением составного графического объекта»

Выполнил: студент II курса

ИВТ, гр. ИП-713

Михеев Н.А.

Проверил: ассистент

Суходоева Н.Н.

6.12.18 отлично

Новосибирск 2018

Содержание

1. Постановка задачи

Круговое движение с вращением составного графического объекта.

2. Иерархия классов

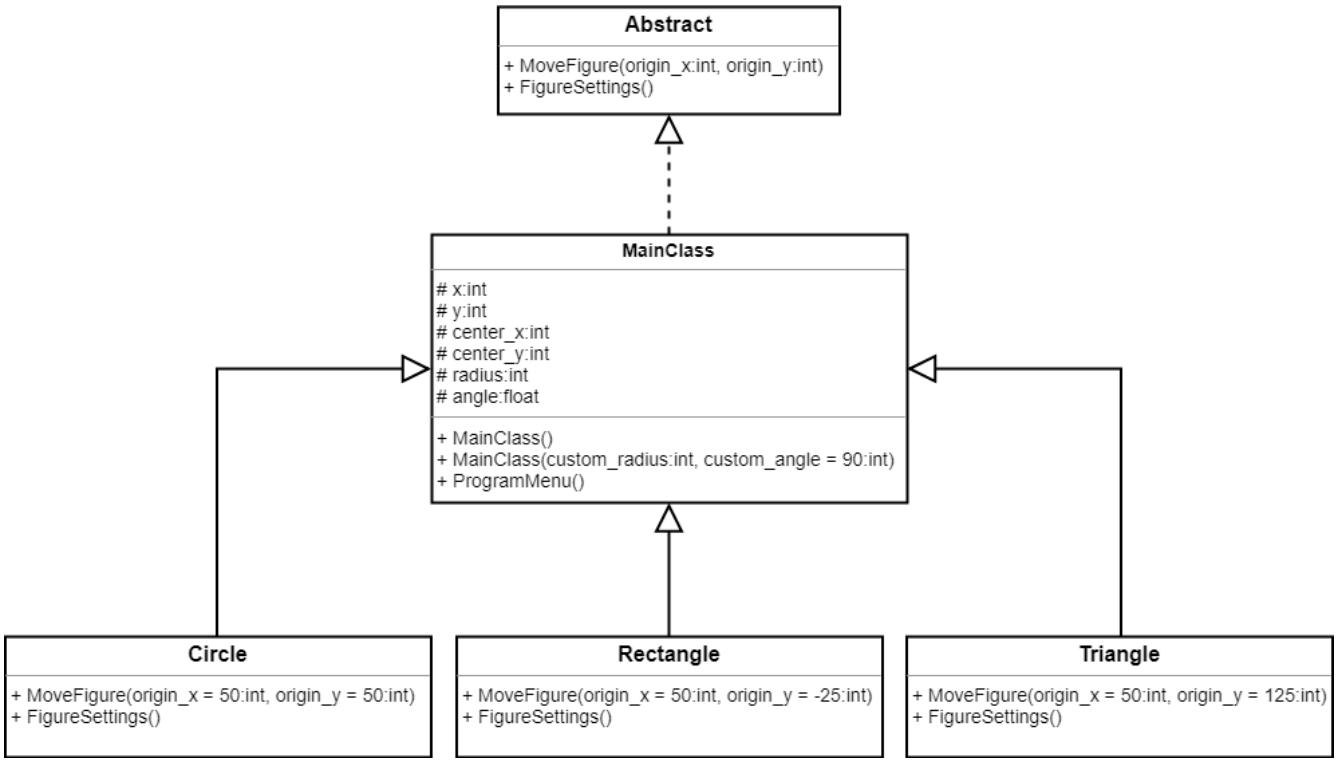

Рис. 1 – UML- диаграмма классов

Класс Abstract – абстрактный класс, в нем находятся два чистых виртуальных метода MoveFigure() и FigureSettings(). Этот класс служит основой для всех производных классов.

Класс MainClass наследует Abstract и содержит:

- Переменные, необходимые для задания начального положения графического объекта (x, y), значение радиуса движения объекта (radius), центр графического окна (center_x, center_y), угол начального движения объекта (angle)
- Перегрузку конструктора MainClass()
- Дружественную функцию ProgramMenu()

Класс Circle наследует MainClass и содержит:

- Функцию движения части графического объекта, в данном случае круга (MoveFigure), у которой заданы параметры по умолчанию (int origin_x = 50, int origin_y = 50)

- Функцию определения цветов и обводки части графического объекта

Класс Rectangle наследует MainClass и содержит:

- Функцию движения части графического объекта, в данном случае прямоугольника (MoveFigure), у которой заданы параметры по умолчанию (int origin_x = 50, int origin_y = -25)
- Функцию определения цветов и обводки части графического объекта

Класс Triangle наследует MainClass и содержит:

- Функцию движения части графического объекта, в данном случае труегольника (MoveFigure), у которой заданы параметры по умолчанию (int origin_x = 50, int origin_y = 125)
- Функцию определения цветов и обводки части графического объекта

3. Описание алгоритма основной программы

Основная програма подключает необходимые для ее функционирования библиотеки (SFML, iostream и модуль с описанием классов).

Затем, в функции main() вызывается функция ProgramMenu() в которой пользователю предлагается запустить графическую часть программы или закрыть ее. Если выбран вариант с запуском, то создаются объекты классов для фигуры_A и фигуры_B, создается объект окна с разрешением 800x800 пикселей и настройками сглаживания AntiAliasing x8, происходит отрисовка этих объектов и их движения.

4. Результат работы

На рисунке 2 изображено стартовое меню, где пользователю предлагается запустить графическую часть программы или закрыть все. Реализованно с помощью дружественной функции.

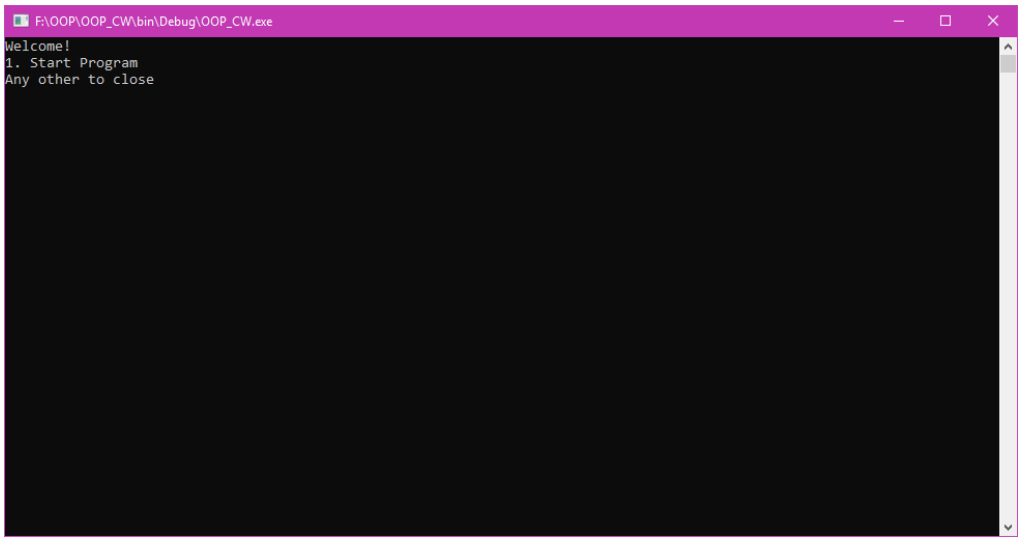

Рис.2 – меню программы, реализованное с помощью дружественной функции

На 3 рисунке работа графической части программы.

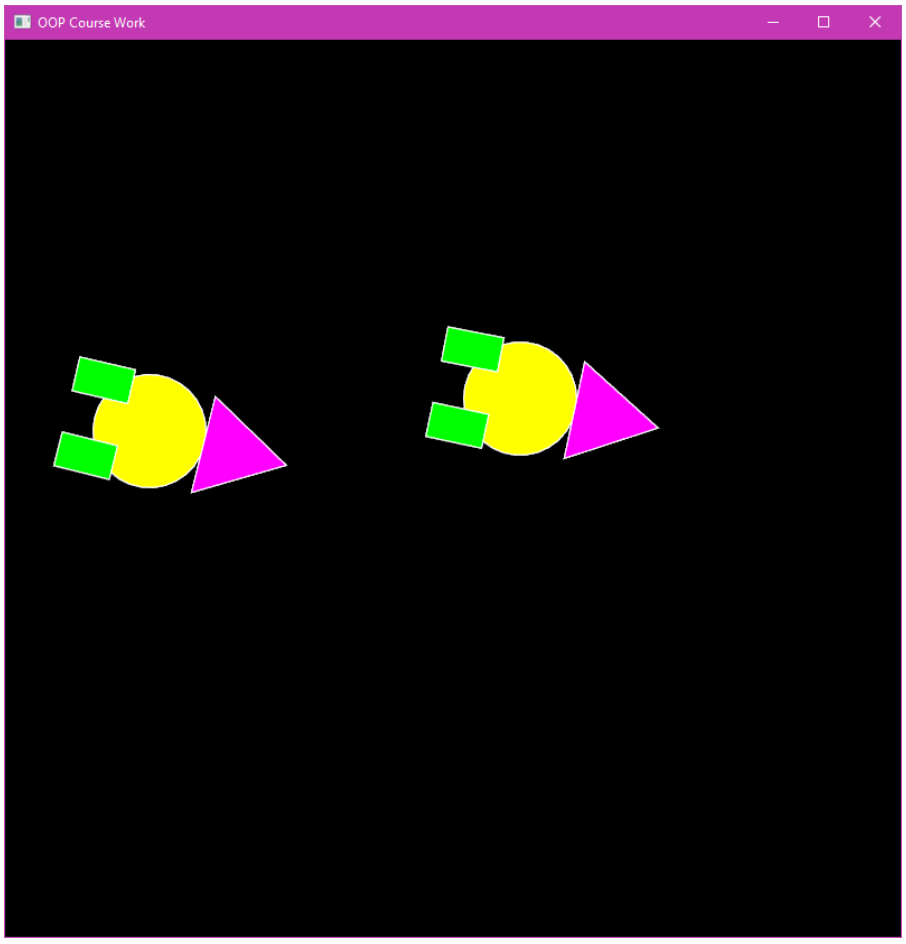

Рис.3 – графическая часть программы

5. Заключение

При написании курсовой работы использовались такие принципы объектно ориентированного программирования, как *наследование, полиморфизим, инкапсуляция.*

Наследование – конструирование новые, более сложных производных классов-потомков, из уже имеющихся базовых классов-родителей с помощью добавления новых полей и/или методов. *Полиморфизим* – механизм, обеспечивающий возможность определения различных описаний некоторого единого метода (единого по названию) для классов различного уровня иерархии. *Инкапсуляция* – механизм скрытия данных внутри объекта. Для создание объектов в классе существуют *конструкторы*, которые были использованы для установки центра графического окна, для обозначения радиуса движения фигур и начального угла, была реализована их *перегрузка*.

Так же использовались *чистые виртуальные методы,* которые послужили основой для функций в наследуемых классах и *абстрактный класс*. *Чистые виртуальные методы* – методы, которые объявляются в базовом классе как виртуальные и не содержащие описания выполняемых действий. *Абстрактный класс* – класс, в котором есть хотя бы один чистый (обнуленный) виртуальный метод. *Дружественная функция* – функция, которая не является компонентом класса, но имеют доступ ко всем его компонентам. *Параметр по умолчанию* - параметр функции, который имеет определенное значение по умолчанию. Если пользователь не передает в функцию значение для параметра по умолчанию, то будет использоваться значение по умолчанию.

6. Программная реализация

Файл main.cpp

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include "couse_lib.h"

int main()
{
    ProgramMenu();

    int custom_radius = 100; //Change it for A-Figure movement
    Circle ACircle(custom_radius); //A-Figure initialization
    Rectangle ARectangle_1(custom_radius);
    Rectangle ARectangle_2(custom_radius);
    Triangle ATriangle(custom_radius);

    Circle BCircle; //B-Figure initialization
    Rectangle BRectangle_1;
    Rectangle BRectangle_2;
    Triangle BTriangle;

    sf::ContextSettings settings;
    settings.antialiasingLevel = 8; //AntiAliasing for figures
    sf::RenderWindow window(sf::VideoMode(800, 800), "OOP Course Work", sf::Style::Default, settings); //window initialization

    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }
        sf::sleep(sf::milliseconds(10));

        ACircle.FigureSettings(); //A-Figure draw
        ACircle.MoveFigure();
        window.draw(circle);

        ARectangle_1.FigureSettings();
        ARectangle_1.MoveFigure(50);
        window.draw(rectangle);

        ARectangle_2.FigureSettings();
        ARectangle_2.MoveFigure(-20);
        window.draw(rectangle);

        ATriangle.FigureSettings();
        ATriangle.MoveFigure();
        window.draw(triangle);
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
        BCircle.FigureSettings(); //B-Figure draw
        BCircle.MoveFigure();
        window.draw(circle);

        BRectangle_1.FigureSettings();
        BRectangle_1.MoveFigure(50);
        window.draw(rectangle);

        BRectangle_2.FigureSettings();
        BRectangle_2.MoveFigure(-20);
        window.draw(rectangle);

        BTriangle.FigureSettings();
```

```
        BTriangle.MoveFigure();
        window.draw(triangle);

        window.display();
        window.clear();
    }
    return 0;
}
```

Файл course_lib.h

```
#ifndef COURSE_LIB_H
#define COUSE_LIB_H

using namespace std;

sf::CircleShape circle(50);
sf::RectangleShape rectangle(sf::Vector2f(30, 50));
sf::CircleShape triangle (50, 3);

class Abstract
{
public:
    virtual void MoveFigure(int origin_x, int origin_y) = 0; //pure virtual function
    virtual void FigureSettings() = 0;
};

class MainClass: public Abstract
{
protected:
    int x; //x axis coordinate
    int y; //y axis coordinate
    int center_x; //window center x
    int center_y; //window center y
    int radius; //movement radius
    float angle; //angle
public:
    explicit MainClass() : center_x(400), center_y(400)//basic data constructor
    {
        angle = 0;
        radius = 275;
    }
    explicit MainClass(int custom_radius, int custom_angle = 90) //basic data constructor overloading
    {
        center_x = 400;
        center_y = 400;
        angle = custom_angle;
        radius = custom_radius;
    }
    friend void ProgramMenu(); //friendly function
};

class Circle: public MainClass
{
    using MainClass::MainClass;
public:
    void MoveFigure(int origin_x = 50, int origin_y = 50)
    {
        circle.setPosition(x, y);
        x = center_x + radius * cos(angle);
        y = center_y + radius * sin(angle);
        angle = angle - 0.01;
        circle.setOrigin(origin_x, origin_y);
        circle.rotate(2);
    }
    void FigureSettings()
    {
        circle.setFillColor(sf::Color::Yellow);
        circle.setOutlineThickness(1);
```

```
        circle.setOutlineColor(sf::Color::White);
    }
};

class Rectangle: public MainClass
{
    using MainClass::MainClass;
public:
    void MoveFigure(int origin_x = 50, int origin_y = -25)
    {
        rectangle.setPosition(x, y);
        x = center_x + radius * cos(angle);
        y = center_y + radius * sin(angle);
        angle = angle - 0.01;
        rectangle.setOrigin(origin_x, origin_y);
        rectangle.rotate(1);
    }
    void FigureSettings()
    {
        rectangle.setFillColor(sf::Color::Green);
        rectangle.setOutlineThickness(1);
        rectangle.setOutlineColor(sf::Color::White);
    }
};

class Triangle: public MainClass
{
    using MainClass::MainClass;
public:
    void MoveFigure(int origin_x = 50, int origin_y = 125)
    {
        triangle.setPosition(x, y);
        x = center_x + radius * cos(angle);
        y = center_y + radius * sin(angle);
        angle = angle - 0.01;
        triangle.setOrigin(origin_x, origin_y);
        triangle.rotate(2);
    }
    void FigureSettings()
    {
        triangle.setFillColor(sf::Color::Magenta);
        triangle.setOutlineThickness(1);
        triangle.setOutlineColor(sf::Color::White);
    }
};

void ProgramMenu()
{
    int button;
    cout << "Welcome!" << endl;
    cout << "1. Start Program" << endl;
    cout << "Any other to close" << endl;
    cin >> button;
    switch(button)
    {
    case 1:
        break;
    default:
        terminate();
        break;
    }
}
#endif // COURSE_LIB_H
```