**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

Алексеев С.В..

Проверил:

ассистент кафедры ПМ и К

Суходоева Н.Н.

Новосибирск – 2018

Оглавление

[Постановка задачи 3](#_Toc531290180)

[Технологии ООП](#_Toc531290181) 3

[Структура классов 4](#_Toc531290182)

[Алгоритм и результат работы 4](#_Toc531290183)

[Заключение 5](#_Toc531290184)

[Приложение. Листинг 6](#_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 *Graphics.h*, функциями библиотеки реализованы все графические объекты и функции движения.

За основу составного графического объекта были взяты простые фигуры:

1. Окружность
2. Прямоугольник
3. Треугольник

# 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. Чистые виртуальные функции
2. Абстрактный класс
3. Конструктор, перегрузка конструкторов
4. Списки инициализации
5. Массив указателей на объекты

### Структура классов

В программе реализован абстрактный класс (*shape*), который содержит в себе виртуальные и чистые виртуальные функции, для возвращения и записи координат, рисования фигур. Так же в нем есть защищенные поля, хранящие координаты фигур.

Shape является родительским для классов circleCls, triangleCls, triangLCls, triangRCls, rectangleCls, которые реализуют инициализацию полей с помощью конструткторов. Файл methods.cpp содержит в себе реализацию методов классов. В нём объявлены и реализованы методы установки и возвращения координат, методы рисования и движения фигур. В файле constants.h описаны константы для рисования фигур.

# Алгоритм и результат работы

Инициализируется окно. Создаются пять объектов фигур с заданными координатами и цветом. Создаётся массив указателей на объекты типа shape. Для каждого объекта поочерёдно вызываются функции рисования и движения(изменения его координат).

# Заключение

Были изучены и реализованы принципы ООП: полиморфизм, наследование, инкапсуляция.

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

Main.cpp

#include "constants.h"

#include "classes.h"

#include <stdio.h>

#include <stdlib.h>

#include <ctime>

#include <iostream>

#include <graphics.h>

main(){

srand(time(NULL));

initwindow(750, 750);

circleCls circleObj( 220 ,150 ,15 );

circleCls \*Circle = &circleObj;

rectangleCls rectanObj( 200, 110,11);

rectangleCls \*Rectangle = &rectanObj;

triangleCls trianObj( 220, 49, 4);

triangleCls \*Triangle = &trianObj;

triangLCls trianLObj( 241, 160, 1);

triangLCls \*LTriangle = &trianLObj;

trianRCls trianRObj(199, 160, 1);

trianRCls \*RTriangle = &trianRObj;

shape \*\*array = new shape\*[ARR\_SIZE];

array[0] = Rectangle;

array[1] = Circle;

array[2] = Triangle;

array[3] = LTriangle;

array[4] = RTriangle;

while(true){

cleardevice();

for(int i = 0; i < ARR\_SIZE; i++){

array[i] -> draw();

array[i] -> move();

}

delay(10);

}

closegraph();

}

Methods.cpp

#include "classes.h"

#include <math.h>

#include "constants.h"

#define N 0.09

float t = 0;

float r = 0;

float s = 0;

float g = 0;

float k = 0;

float shape::getXpos() const{

return x;

}

void shape::setXpos(const float x){

shape::x = x;

}

float shape::getYpos() const{

return y;

}

void shape::setYpos(const float y){

shape::y = y;

}

int shape::getClr() const{

return color;

}

void shape::setClr(const int color){

shape::color = color;

}

void circleCls::move(){

x = x + 15 \* cos(t);

y = y + 15 \* sin(t);

t+= N;

}

void circleCls::draw(){

setcolor(getClr());

circle(getXpos(),

getYpos(), CIRCLE\_RADIUS);

}

void rectangleCls::move(){

x = x + 15 \* cos(r);

y = y + 15 \* sin(r);

r+= N;

}

void rectangleCls::draw(){

setcolor(getClr());

rectangle(getXpos(), getYpos(),

getXpos() + RECTAN\_SIZE/ 2,

getYpos() + RECTAN\_SIZE );

}

void triangleCls::move(){

x = x + 15 \* cos(s);

y = y + 15 \* sin(s);

s+= N;

}

void triangleCls::draw(){

setcolor(color);

line(x, y, x + 17, y + 60);

line(x, y, x - 17, y + 60);

line(x - 19, y + 60,

x + 19, y + 60);

}

void triangLCls::move(){

x = x + 15 \* cos(k);

y = y + 15 \* sin(k);

k+= N;

}

void triangLCls::draw(){

setcolor(color);

line(x, y, x + 17, y + 60);

line(x, y, x , y + 60);

line(x - 1, y + 60,

x + 18, y + 60);

}

void trianRCls::move(){

x = x + 15 \* cos(g);

y = y + 15 \* sin(g);

g+= N;

}

void trianRCls::draw(){

setcolor(color);

line(x, y, x-17 , y + 60);

line(x, y, x , y + 60);

line(x - 18, y + 60,

x + 1, y + 60);

}

Classes.h

#ifndef CLASS\_H

#define CLASS\_H

#include <stdio.h>

#include <ctime>

#include <stdlib.h>

#include <iostream>

#include <graphics.h>

#include "constants.h"

class shape

{

public:

float getXpos() const;

void setXpos(const float x);

float getYpos() const;

void setYpos(const float y);

int getClr() const;

void setClr(const int color);

virtual void move() = 0;

virtual void draw() = 0;

shape(){}

protected:

float x;

float y;

int color;

};

class circleCls : public shape

{

public:

void move();

void draw();

circleCls(float x, float y, int color):shape(){

setXpos(x);

setYpos(y);

setClr(color);

}

};

class triangleCls : public shape

{

public:

void move();

void draw();

triangleCls(float x, float y, int color):shape(){

setXpos(x);

setYpos(y);

setClr(color);

}

};

class triangLCls : public shape

{

public:

void move();

void draw();

triangLCls(float x, float y, int color):shape(){

setXpos(x);

setYpos(y);

setClr(color);

}

};

class trianRCls : public shape

{

public:

void move();

void draw();

trianRCls(float x, float y, int color):shape(){

setXpos(x);

setYpos(y);

setClr(color);

}

};

class rectangleCls : public shape

{

public:

void move();

void draw();

int pl;

rectangleCls(float x, float y, int color):shape(){

setXpos(x);

setYpos(y);

setClr(color);

}

rectangleCls(void) : pl(300){

}

};

#endif

Constants.h

#ifndef CONST\_H

#define CONST\_H

#define CIRCLE\_RADIUS 12

#define RECTAN\_SIZE 75

#define TRIAN\_SIZE 40

#define ARR\_SIZE 5

#endif