|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по домашнему заданию №** | 3 |

**Название:**

Программирование на С++ с использованием классов

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22Б |  |  | Н.О.Марков |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.А.Веселовский |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Задача**:

*Часть 3.1. Композиция*

Разработать и реализовать диаграмму классов для описанных объектов предметной области, используя механизм композиции. Протестировать все методы каждого класса. Все поля классов должны быть скрытыми (private) или защищенными (protected). Методы не должны содержать операций ввода/вывода, за исключением процедуры, единственной задачей которой является вывод информации об объекте на экран.

**Объект – туристическая услуга**. Поля: название, стоимость за человека, признак «обязательная». Методы: процедура инициализации объекта, процедура вывода на экран информации об объекте и функции, возвращающие значения полей по запросу.

**Объект – тур.** Поля: название, число туристов, массив услуг. Методы объекта должны позволять: инициализировать объект, выводить на экран содержимое объекта, определять полную стоимость тура и определять стоимость базового пакета тура, состоящего из обязательных услуг.

В отчете привести диаграмму разработанных классов и объектную декомпозицию.

*Часть 3.2. Qt. Полиморфное наследование*

Разработать программу, содержащую описание трех графических объектов:

окружность, астроида, окружность с вписанной астроидой.

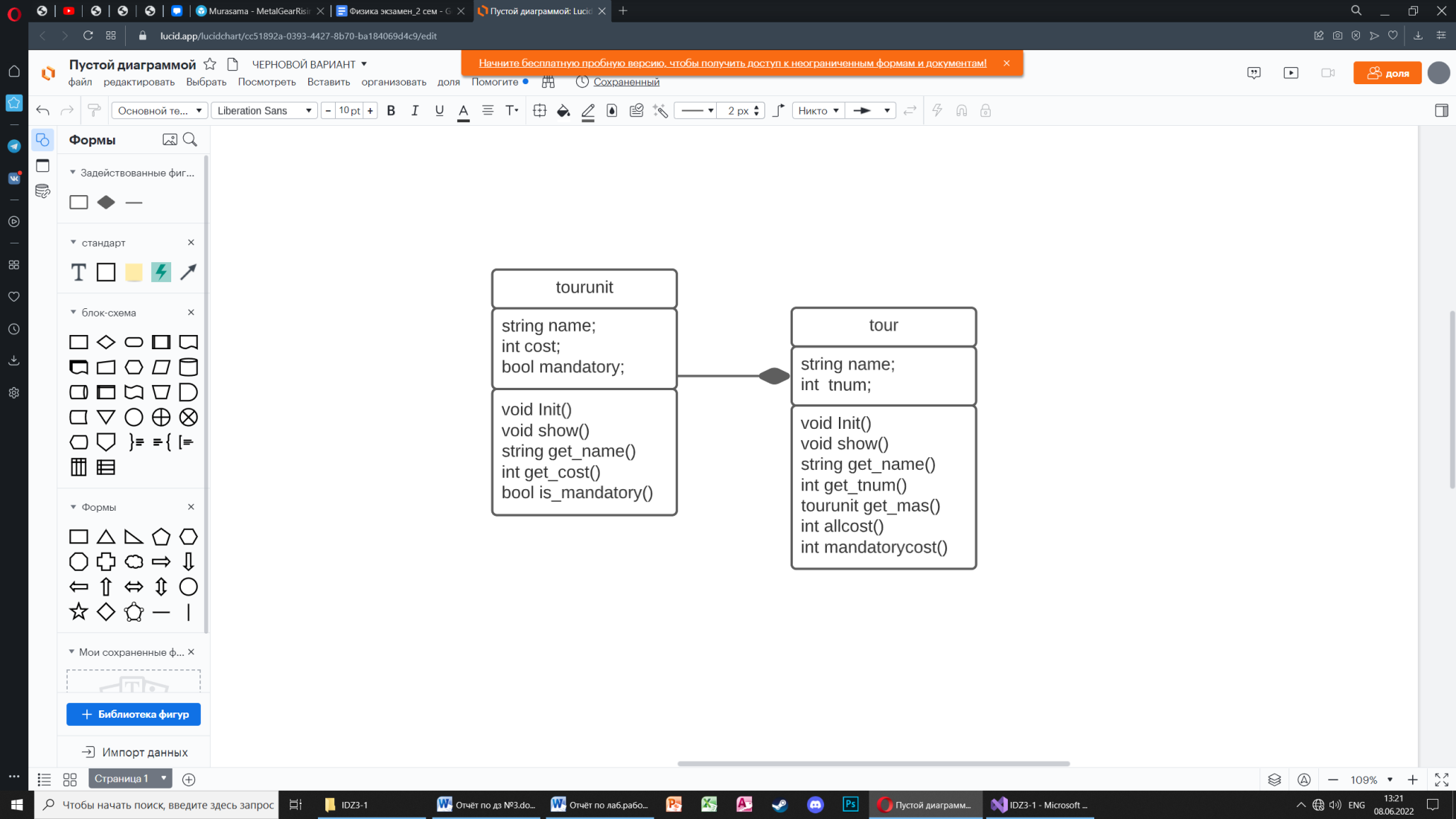
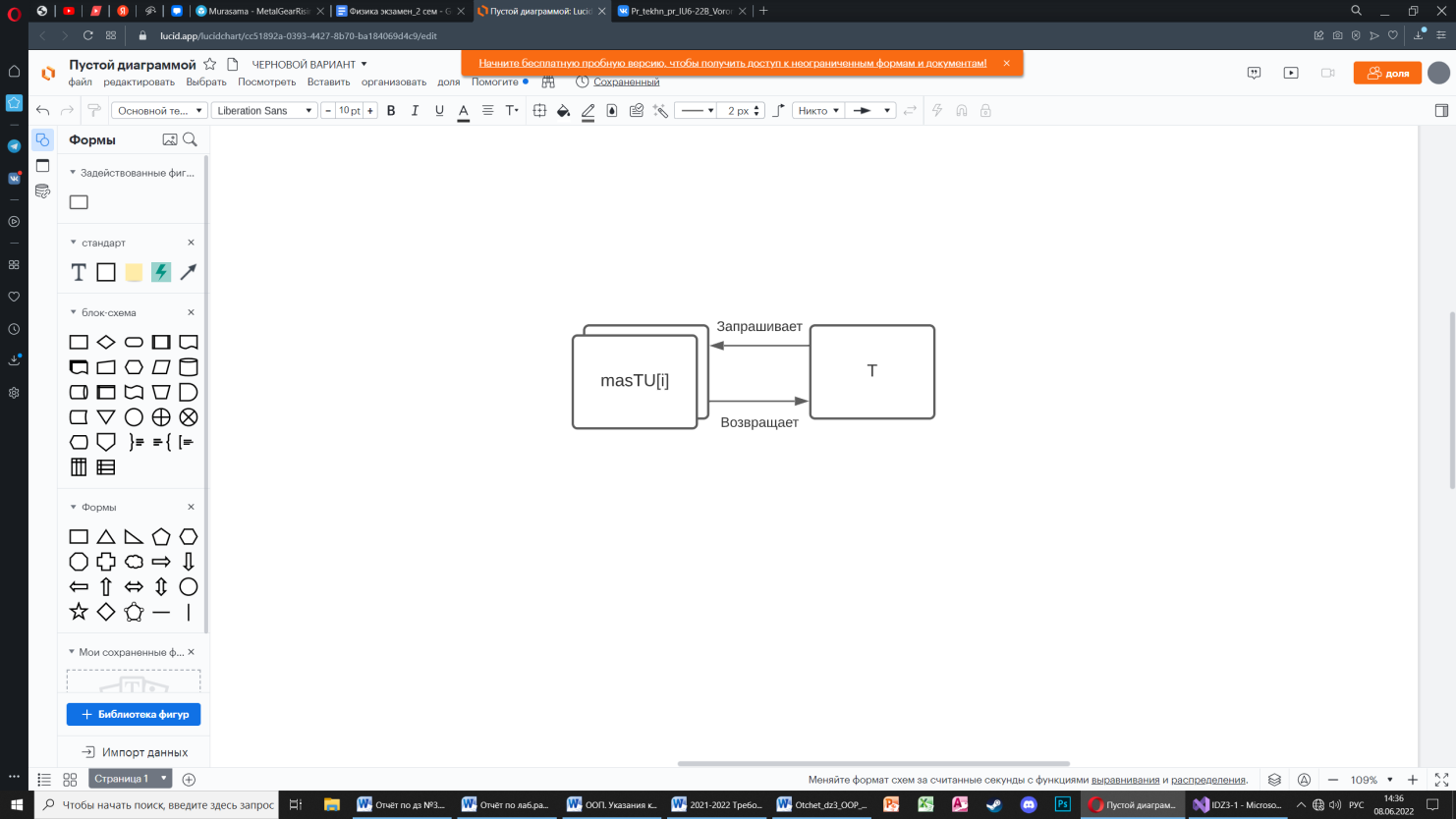
Реализуя механизм полиморфизма, привести объекты в одновременное колебательное движение вокруг верхних точек с разными амплитудами и периодами колебаний.

В отчете привести диаграмму используемых классов Qt и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозицию.

**Решение:**

*Часть 3.1. Композиция*

**Диаграмма классов: Диаграмма объектов:**



#include <iostream>

#include <stdio.h>

#include <stdio.h>

#include <string>

#include <locale>

#include <windows.h>

using namespace::std;

int main()

{

const int maxunitnum = 3;

setlocale(0, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

class tourunit

{

protected:

string name;

int cost;

bool mandatory;

public:

void Init(string n, int c, bool m) { name = n; cost = c; mandatory = m; };

void show() {

cout << "Название услуги: " << get\_name() << "\nЦена услуги: " << get\_cost()<< "\n";

if (is\_mandatory()) cout << "Услуга является обязательной";

else cout << "Услуга не является обязательной";

};

string get\_name() { return name; };

int get\_cost() { return cost; };

bool is\_mandatory() { return mandatory; };

};

class tour

{

protected:

string name;

int tnum;

tourunit mas[maxunitnum];

public:

void Init(string n, int tn, tourunit m[maxunitnum]) { name = n; tnum=tn; for(int i=0;i< maxunitnum;i++) mas[i] = m[i]; };

void show() {

cout << "Название тура: " << get\_name() << "\nЧисло туристов: " << get\_tnum() << "\nВключенные услуги: "<< "\n";

for (int i = 0; i < maxunitnum; i++) cout << " " << get\_mas(i).get\_name() << "\n";

cout << "Стоимость полного пакета: " << allcost() << "\n";

cout << "Стоимость базового пакета: " << mandatorycost() << "\n";

};

string get\_name() { return name; };

int get\_tnum() { return tnum; };

tourunit get\_mas(int i) { return mas[i]; }

int allcost() {

int sum = 0;

for (int i = 0; i < maxunitnum; i++) sum += get\_mas(i).get\_cost();

return sum;

};

int mandatorycost() {

int sum = 0;

for (int i = 0; i < maxunitnum; i++) if(get\_mas(i).is\_mandatory()) sum += get\_mas(i).get\_cost();

return sum;

};

};

tourunit masTU[maxunitnum];

tour T;

string Tname, TUname;

int TUcost, Tnum;

bool m;

cout << "Введите название тура: ";

cin >> Tname;

cin.ignore(2, '\n');

cout << "Введите количество туристов: ";

cin >> Tnum;

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

cout << "\nВведите название услуги: ";

cin >> TUname;

cout << "Введите цену услуги: ";

cin >> TUcost;

cout << "Обязательна ли услуга? (1-обязательна, 0-не обязательна): ";

cin >> m;

masTU[i].Init(TUname, TUcost, m);

}

T.Init(Tname, Tnum, masTU);

cout << "\n\n";

T.show();

}

*Часть 3.2. Qt. Полиморфное наследование*

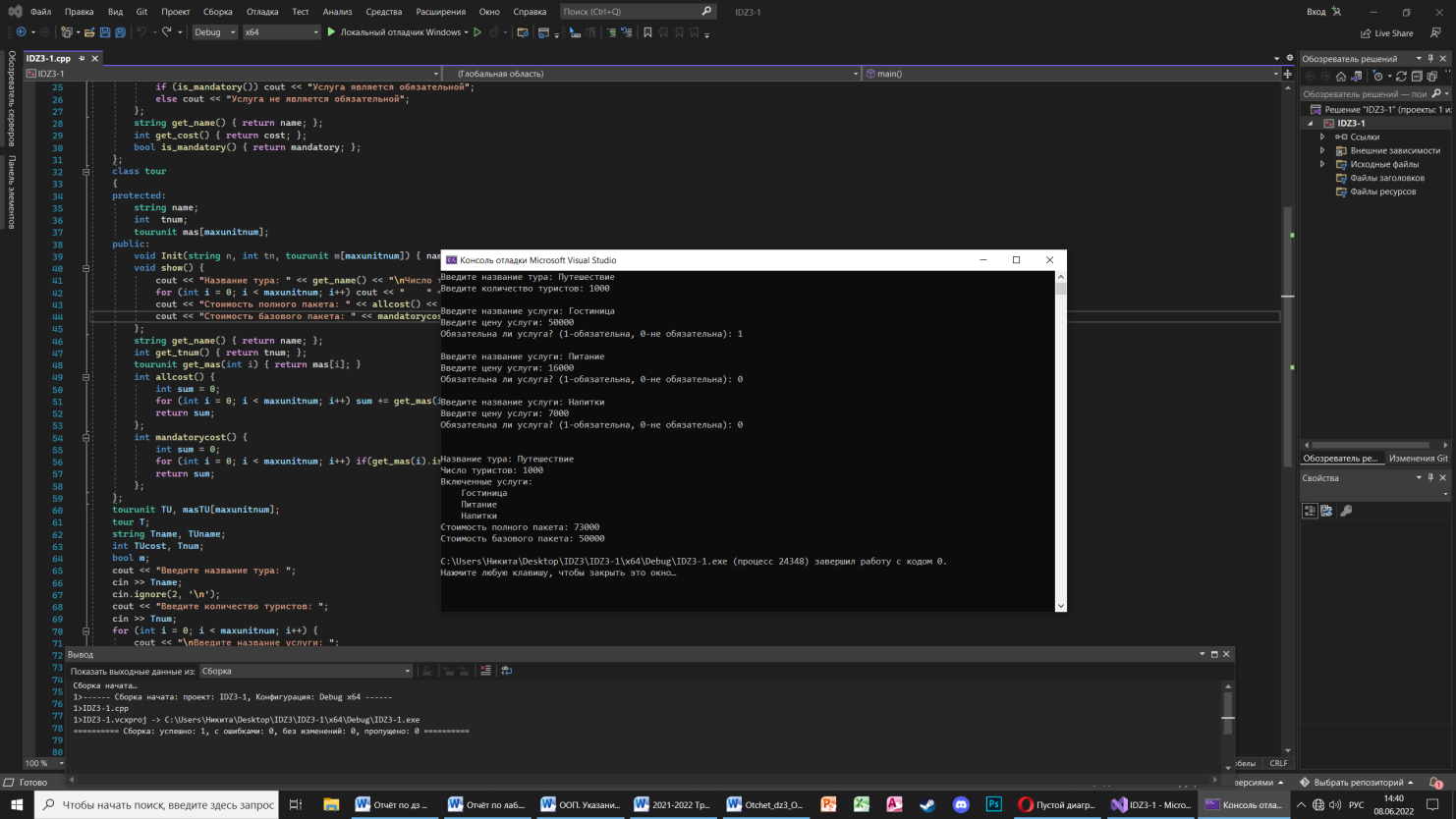
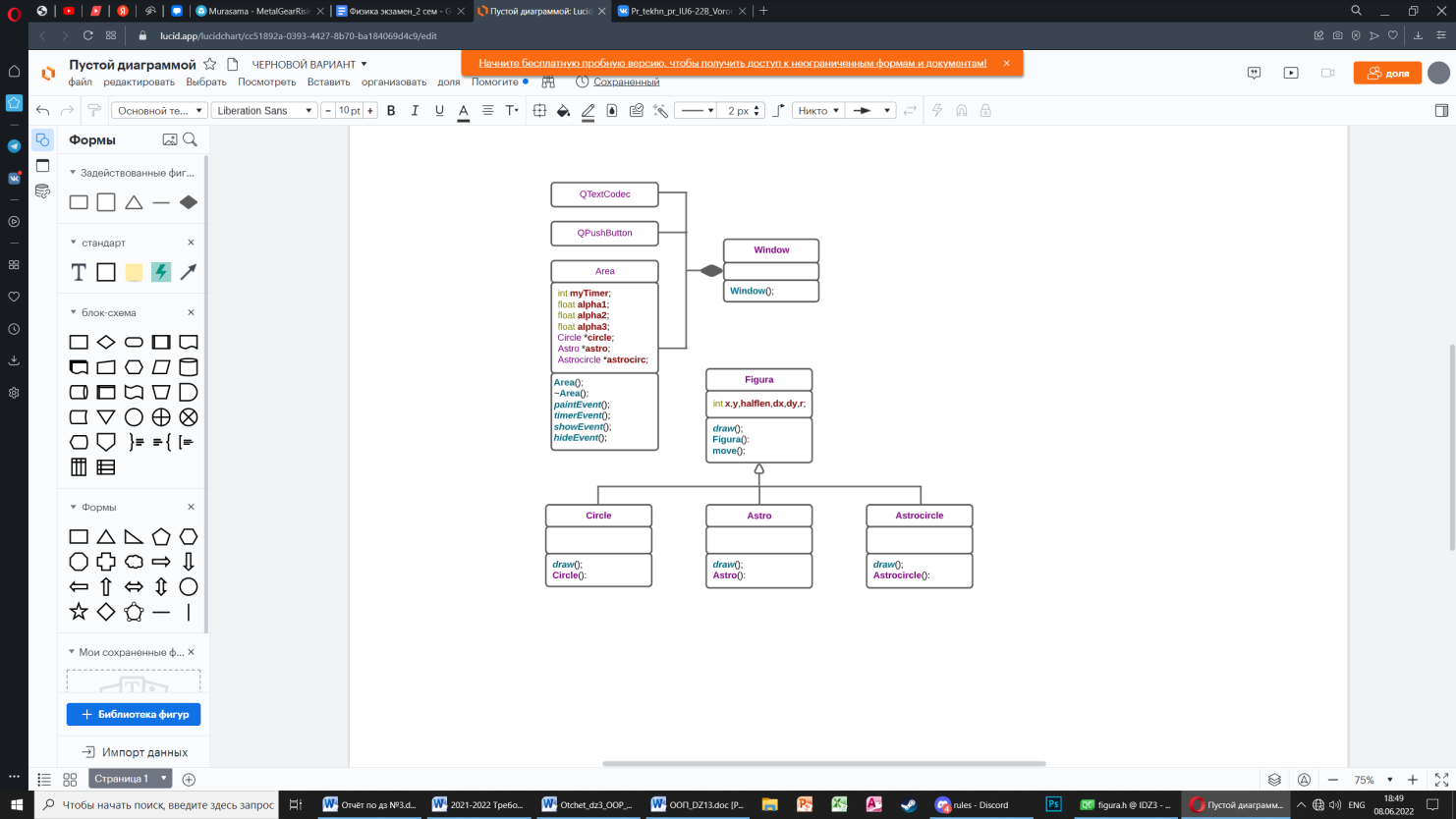
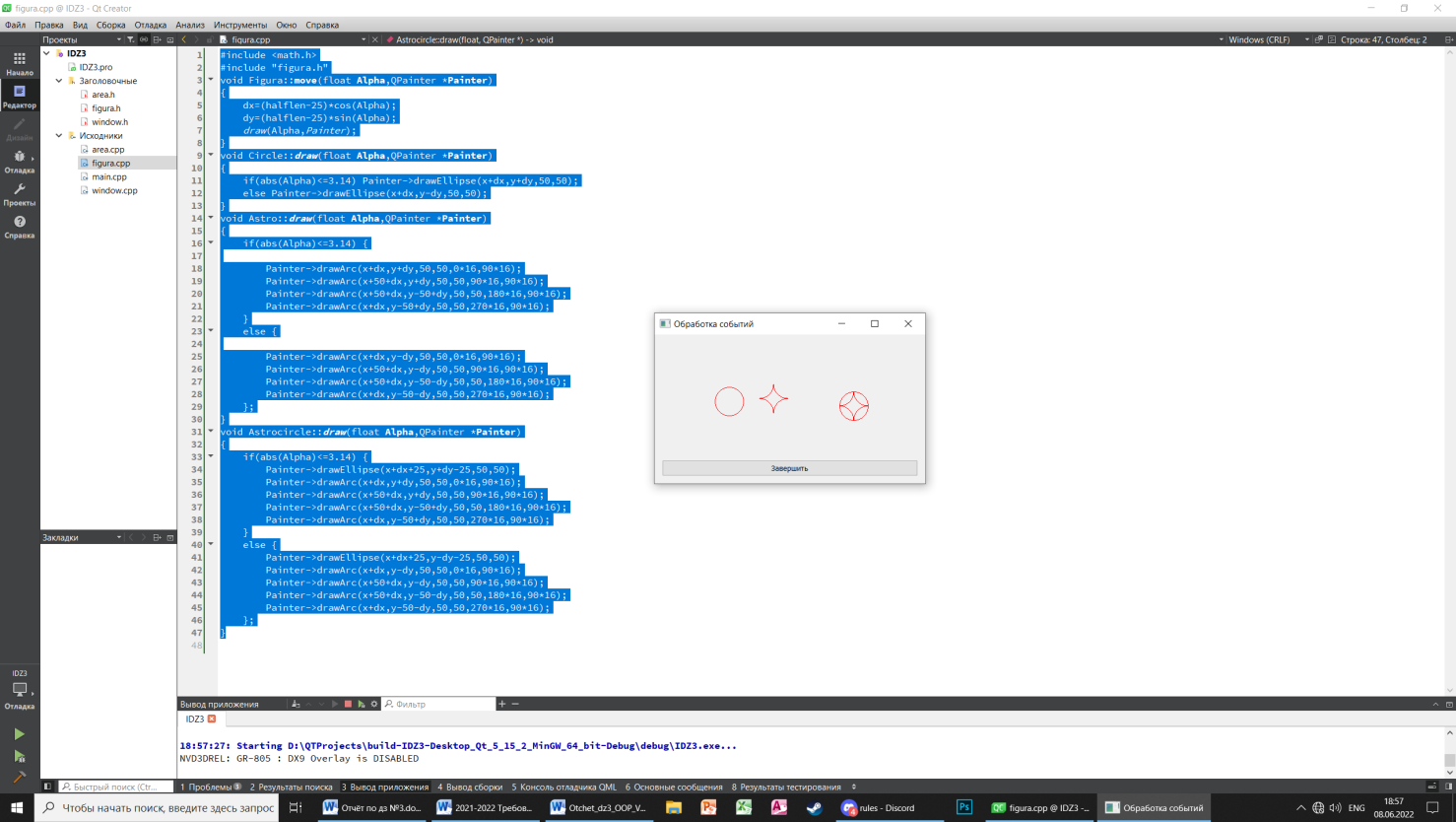


Диаграмма классов:



**Main.cpp**

#include "qapplication.h"

#include "window.h"

int main(int **argc**, char \***argv**[])

{

QApplication **appl**(*argc*, *argv*);

Window **win**;

win.show();

return appl.exec();

}

**Window.h**

#ifndef WINDOW\_H

#define **WINDOW\_H**

#include <QWidget>

#include <QtGui>

#include "area.h"

#include <QPushButton>

#include <QVBoxLayout>

class **Window** : public QWidget

{

Q\_OBJECT

protected:

QTextCodec \***codec**;

Area \* **area**; // область отображения рисунка

QPushButton \* **btn**;

public:

**Window**();

};

#endif // WINDOW\_H

**Window.cpp**

#include "window.h"

Window::**Window**()

{

codec = QTextCodec::codecForName("UTF-8");

this->setWindowTitle(codec->toUnicode("Обработка событий"));

area = new Area( this );

btn = new QPushButton(codec->toUnicode("Завершить"),this );

QVBoxLayout \***layout** = new QVBoxLayout(this);

layout->addWidget(*area*);

layout->addWidget(*btn*);

connect(btn, SIGNAL(clicked(bool)),this,SLOT(close()));

};

**Area.h**

#ifndef AREA\_H

#define **AREA\_H**

#include "figura.h"

#include "qwidget.h"

class **Area** : public QWidget

{

int **myTimer**; // идентификатор таймера

float **alpha1**;

float **alpha2**;

float **alpha3**;// угол поворота

Circle \***circle**;

Astro \***astro**;

Astrocircle \***astrocirc**;

public:

**Area**(QWidget \***parent** = 0);

~**Area**();

protected:

// обработчики событий

void ***paintEvent***(QPaintEvent \***event**);

void ***timerEvent***(QTimerEvent \***event**);

void ***showEvent***(QShowEvent \***event**);

void ***hideEvent***(QHideEvent \***event**);

};

#endif // AREA\_H

**Area.cpp**

#include "area.h"

Area::**Area**(QWidget \***parent**):QWidget(*parent*)

{

setFixedSize(QSize(450,200));

circle=new Circle(70,70,50);

astro=new Astro(170,95,50);

astrocirc=new Astrocircle(270,95,50);

alpha1=0;

alpha2=0;

alpha3=0;

}

void Area::***showEvent***(QShowEvent \*)

{

myTimer=startTimer(50); // создать таймер

}

void Area::***paintEvent***(QPaintEvent \*)

{

QPainter **painter**(this);

painter.setPen(Qt::red);

circle->move(alpha1,&*painter*);

astro->move(alpha2,&*painter*);

astrocirc->move(alpha3,&*painter*);

}

void Area::***timerEvent***(QTimerEvent \***event**)

{

if (event->timerId() == myTimer) // если наш таймер

{

alpha1=alpha1+0.2;

alpha2=alpha2+0.1;

alpha3=alpha3+0.4;

if(alpha1>6.28) alpha1-=6.28;

if(alpha2>6.28) alpha2-=6.28;

if(alpha3>6.28) alpha3-=6.28;

update(); // обновить внешний вид

}

else

QWidget::*timerEvent*(*event*); // иначе передать для стандартной

// обработки

}

void Area::***hideEvent***(QHideEvent \*)

{

killTimer(myTimer); // уничтожить таймер

}

Area::~**Area**()

{

delete circle;

delete astro;

delete astrocirc;

}

**Figura.h**

#ifndef FIGURA\_H

#define **FIGURA\_H**

#include <QtGui>

class **Figura**

{

protected:

int **x**,**y**,**halflen**,**dx**,**dy**,**r**;

virtual void ***draw***(float **Alpha**,QPainter \***Painter**)=0;

public:

**Figura**(int **X**,int **Y**,int **Halflen**):

x(X),y(Y),halflen(Halflen){}

void **move**(float **Alpha**,QPainter \***Painter**);

};

class **Circle**:public Figura

{

protected:

void ***draw***(float **Alpha**,QPainter \***Painter**);

public:

**Circle**(int **x**,int **y**,int **halflen**):Figura(x,y,halflen){}

};

class **Astro**:public Figura

{

protected:

void ***draw***(float **Alpha**,QPainter \***Painter**);

public:

**Astro**(int **x**,int **y**,int **halflen**):Figura(x,y,halflen){}

};

class **Astrocircle**:public Figura

{

protected:

void ***draw***(float **Alpha**,QPainter \***Painter**);

public:

**Astrocircle**(int **x**,int **y**,int **halflen**):Figura(x,y,halflen){}

};

#endif // FIGURA\_H

**Figura.cpp**

#include <math.h>

#include "figura.h"

void Figura::**move**(float **Alpha**,QPainter \***Painter**)

{

dx=(halflen-25)\*cos(Alpha);

dy=(halflen-25)\*sin(Alpha);

*draw*(Alpha,*Painter*);

}

void Circle::***draw***(float **Alpha**,QPainter \***Painter**)

{

if(abs(Alpha)<=3.14) Painter->drawEllipse(x+dx,y+dy,50,50);

else Painter->drawEllipse(x+dx,y-dy,50,50);

}

void Astro::***draw***(float **Alpha**,QPainter \***Painter**)

{

if(abs(Alpha)<=3.14) {

Painter->drawArc(x+dx,y+dy,50,50,0\*16,90\*16);

Painter->drawArc(x+50+dx,y+dy,50,50,90\*16,90\*16);

Painter->drawArc(x+50+dx,y-50+dy,50,50,180\*16,90\*16);

Painter->drawArc(x+dx,y-50+dy,50,50,270\*16,90\*16);

}

else {

Painter->drawArc(x+dx,y-dy,50,50,0\*16,90\*16);

Painter->drawArc(x+50+dx,y-dy,50,50,90\*16,90\*16);

Painter->drawArc(x+50+dx,y-50-dy,50,50,180\*16,90\*16);

Painter->drawArc(x+dx,y-50-dy,50,50,270\*16,90\*16);

};

}

void Astrocircle::***draw***(float **Alpha**,QPainter \***Painter**)

{

if(abs(Alpha)<=3.14) {

Painter->drawEllipse(x+dx+25,y+dy-25,50,50);

Painter->drawArc(x+dx,y+dy,50,50,0\*16,90\*16);

Painter->drawArc(x+50+dx,y+dy,50,50,90\*16,90\*16);

Painter->drawArc(x+50+dx,y-50+dy,50,50,180\*16,90\*16);

Painter->drawArc(x+dx,y-50+dy,50,50,270\*16,90\*16);

}

else {

Painter->drawEllipse(x+dx+25,y-dy-25,50,50);

Painter->drawArc(x+dx,y-dy,50,50,0\*16,90\*16);

Painter->drawArc(x+50+dx,y-dy,50,50,90\*16,90\*16);

Painter->drawArc(x+50+dx,y-50-dy,50,50,180\*16,90\*16);

Painter->drawArc(x+dx,y-50-dy,50,50,270\*16,90\*16);

};

}