Министерство образования и науки Российской Федерации

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**Разработка структур хранения для графических объектов**

**Выполнил**:студент группы 381606-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимакин Н.Е.

Подпись

**Проверил**: к.ф.-м.н., доц.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Баркалов К.А.

Подпись

Нижний Новгород 2017

**Введение**

Иногда командной строки недостаточно для визуализации интерфейса программы. И C++ как раз представлена возможность создания графических объектов, для этого используется Microsoft Windows Forms.

**Постановка задачи**

Разработать структуру, которая бы могла хранить простейшие геометрические фигуры и проводить с ними такие действия, как показать, скрыть, передвинуть.

**Описание структуры программы**

В программе представлена иерархия классов:

figure

FigureGroup

point

circle

line

rectangle

От базового класса фигура открыто наследуются классы группа фигур и точка, от неё – круг и отрезок, от него – прямоугольник.

figure.h – содержит поля и методы класса figure

Поля:

bool visible; - видимость фигуры

Методы:

virtual void show(Graphics ^gr) = 0; - показать фигуру

virtual void hide(Graphics ^gr) = 0; - скрыть фигуру

virtual void move(Graphics ^gr, int dx, int dy) = 0; - подвинуть фигуру

virtual int getXmax() = 0; - максимальное значение по координате x

virtual int getXmin() = 0; - минимальное значение по координате x

virtual int getYmax() = 0; - максимальное значение по координате y

virtual int getYmin() = 0; - минимальное значение по координате y

point.h – содержит поля и методы класса point

Поля:

int x1, y1; - координаты точки

Методы:

point(int \_x1 = 0, int \_y1 = 0); - конструктор по умолчанию и инициализатор

virtual void show(Graphics ^gr) ; - показать точку

virtual void hide(Graphics ^gr) ; - скрыть точку

virtual void move(Graphics ^gr, int dx, int dy) ; - подвинуть точку

virtual int getXmax(); - максимальное значение по координате x

virtual int getXmin(); - минимальное значение по координате x

virtual int getYmax(); - максимальное значение по координате y

virtual int getYmin(); - минимальное значение по координате y

circle.h – содержит поля и методы класса circle  
Поля:

int r; - радиус

Методы:

circle(int\_x1,int\_y1,int\_r);-конструктор-инициализатор

virtual void show(Graphics ^gr) ; - показать круг

virtual void hide(Graphics ^gr) ; - скрыть круг

virtual int getXmax(); - максимальное значение по координате x

virtual int getXmin(); - минимальное значение по координате x

virtual int getYmax(); - максимальное значение по координате y

virtual int getYmin(); - минимальное значение по координате y

Метод move не перекрывается и наследуется от класса point

line.h – содержит поля и методы класса line  
Поля:

int x2, y2; - координаты конца отрезка

Координаты начала отрезка наследуются от класса point

Методы:

line(int \_x1 = 0, int \_y1 = 0, int \_x2 = 0, int \_y2 = 0); - конструктор по умолчанию и инициализатор

virtual void show(Graphics ^gr) ; - показать отрезок

virtual void hide(Graphics ^gr) ; - скрыть отрезок

virtual void move(Graphics ^gr, int dx, int dy) ; - подвинуть отрезок

virtual int getXmax(); - максимальное значение по координате x

virtual int getXmin(); - минимальное значение по координате x

virtual int getYmax(); - максимальное значение по координате y

virtual int getYmin(); - минимальное значение по координате y

rectangle.h – содержит поля и методы класса rectangle

Поля:

Координаты верхнего левого угла наследуются от класса point, правого нижнего – от line

Методы:

rectangle(int \_x1 = 0, int \_y1 = 0, int \_x2 = 0, int \_y2 = 0); - конструктор по умолчанию и инициализатор

virtual void show(Graphics ^gr) ; - показать прямоугольник

virtual void hide(Graphics ^gr) ; - скрыть прямоугольник

virtual int getXmax(); - максимальное значение по координате x

virtual int getXmin(); - минимальное значение по координате x

virtual int getYmax(); - максимальное значение по координате y

virtual int getYmin(); - минимальное значение по координате y

Метод move не перекрывается и наследуется от класса line

FigureGroup.h – содержит поля и методы класса FigureGroup

Поля:

int count; - количество фигур

figure \*\*figures; - массив указателей на фигуры

Методы:

FigureGroup(int \_count = 1); - конструктор по умолчанию и инициализатор

~FigureGroup(); - деструктор

void setfigure(figure \*f, int pos); - вставить фигуру на заданную позицию

figure\* getfigure(int pos); - получить указатель на фигуру из данной позиции

virtual void show(Graphics ^gr) ; - показать группу фигур

virtual void hide(Graphics ^gr) ; - скрыть группу фигур

virtual void move(Graphics ^gr, int dx, int dy) ; - подвинуть группу фигур

virtual int getXmax(); - максимальное значение по координате x

virtual int getXmin(); - минимальное значение по координате x

virtual int getYmax(); - максимальное значение по координате y

virtual int getYmin(); - минимальное значение по координате y

MyForm.h и MyForm.cpp – содержат визуальную составляющую программы

**Описание алгоритмов**

figure.h

В этом файле содержатся абстрактный класс и его виртуальные методы, которые не требуют описания

point.h

point(int \_x1 = 0, int \_y1 = 0); - присвоить полям x1 и y1 соответствующие значения

virtual void show(Graphics ^gr) ; - нарисовать окружность радиуса 2 чёрного цвета

virtual void hide(Graphics ^gr) ; - нарисовать окружность радиуса 2 белого цвета

virtual void move(Graphics ^gr, int dx, int dy) ; - вызвать метод hide, прибавить к x1 dx, к y1 dy, вызвать метод show

virtual int getXmax(); - вернуть x1

virtual int getXmin(); - вернуть x1

virtual int getYmax(); - вернуть y1

virtual int getYmin(); - вернуть y1

circle.h

circle(int\_x1,int\_y1,int\_r); - вызвать конструктор для точки, присвоить радиусу соответствующее значение

virtual void show(Graphics ^gr) ; - нарисовать окружность радиуса r жёлтого цвета

virtual void hide(Graphics ^gr) ; - нарисовать окружность радиуса r белого цвета

virtual int getXmax(); - вернуть x1+r

virtual int getXmin(); - вернуть x1-r

virtual int getYmax(); - вернуть y1+r

virtual int getYmin(); - вернуть y1-r

line.h

line(int \_x1 = 0, int \_y1 = 0, int \_x2 = 0, int \_y2 = 0); - вызвать конструктор для точки, присвоить x2 и y2 соответствующие значения

virtual void show(Graphics ^gr) ; - нарисовать отрезок с координатами x1 y1 и x2 y2 чёрного цвета

virtual void hide(Graphics ^gr) ; - нарисовать отрезок с координатами x1 y1 и x2 y2 белого цвета

virtual void move(Graphics ^gr, int dx, int dy) ; - вызвать метод hide, прибавить к x1 dx, к y1 dy, к x2 dx, к y2 dy, вызвать метод show

virtual int getXmax(); - если x1>x2 вернуть x1, иначе вернуть x2

virtual int getXmin(); - если x1<x2 вернуть x1, иначе вернуть x2

virtual int getYmax(); - если y1>y2 вернуть y1, иначе вернуть y2

virtual int getYmin(); - если y1<y2 вернуть y1, иначе вернуть y2

rectangle.h

rectangle(int \_x1 = 0, int \_y1 = 0, int \_x2 = 0, int \_y2 = 0); - вызвать конструктор для отрезка

virtual void show(Graphics ^gr) ; - нарисовать прямоугольник с точкой (x1,y1), длиной x2-x1, высотой y2-y1 синего цвета

virtual void hide(Graphics ^gr) ; - нарисовать прямоугольник с точкой (x1,y1), длиной x2-x1, высотой y2-y1 белого цвета

virtual int getXmax(); - вернуть x2

virtual int getXmin(); - вернуть x1

virtual int getYmax(); - вернуть y2

virtual int getYmin(); - вернуть y1

FigureGroup.h

FigureGroup(int \_count = 1); - присвоить count соответствующее значение и выделить память под массив из указателей на фигуры размера count

~FigureGroup(); - поочерёдно освободить все figures[i], освободить figures

void setfigure(figure \*f, int pos); - figures[pos] присвоить f

figure\* getfigure(int pos); - вернуть figures[pos]

virtual void show(Graphics ^gr) ; - поочерёдно вызвать метод show для каждой фигуры

virtual void hide(Graphics ^gr) ; - поочерёдно вызвать метод hide для каждой фигуры

virtual void move(Graphics ^gr, int dx, int dy) ; - поочерёдно вызвать метод move для каждой фигуры

virtual int getXmax(); - выбрать максимум из максимальных x всех фигур и вернуть его

virtual int getXmin(); - выбрать минимум из минимальных x всех фигур и вернуть его

virtual int getYmax(); - выбрать максимум из максимальных y всех фигур и вернуть его

virtual int getYmin(); - выбрать минимум из минимальных y всех фигур и вернуть его

MyForm.h

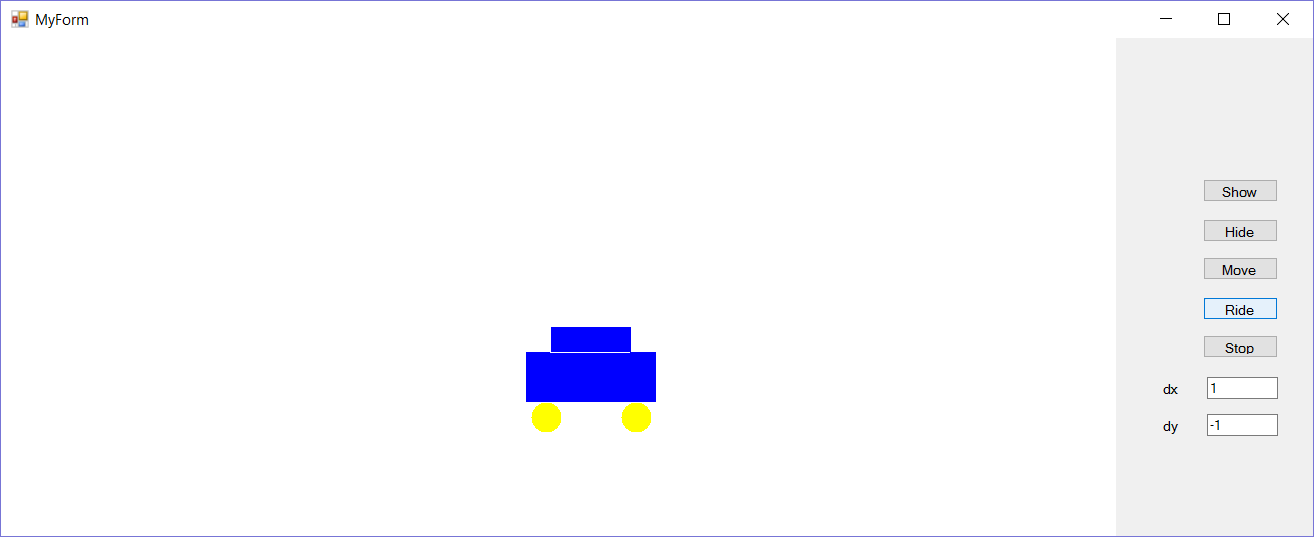
Полями формы сделать указатели на группу фигур и несколько других фигур, в конструкторе формы выделить память под группу фигур и другие фигуры, записать другие фигуры в группу. Создать 5 кнопок: 1 – show, 2 – hide, 3 – move, 4 – позволяет группе фигур двигаться бесконечно, отталкиваясь от стенок, 5 – остановка. Создать 2 textBox, в которые будут записываться dx и dy. По нажатию на кнопку должно производиться соответствующее действие. Изображение выводится на pictureBox

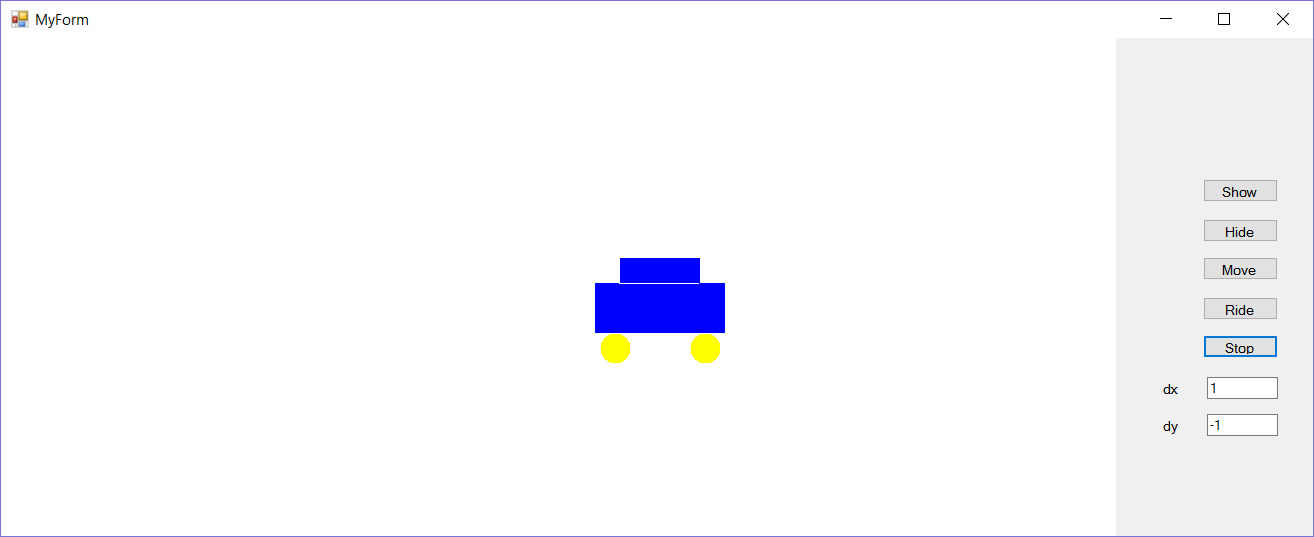
**Результаты**











**Вывод**

Как видно из результатов, все действия выполняются корректно, а значит, задача по разработке структур хранения графических объектов выполнена.

**Литература**

Брайан Керниган, Деннис Ритчи «Язык программирования Си»

Брюс Эккель «Философия С++. Введение в стандартный С++»

Стивен Прата «Язык программирования С++. Лекции и упражнения»

**Приложение**

figure.h

#pragma once

using namespace System::Drawing;

class figure {

protected:

bool visible;

public:

figure() {

visible = true;

}

virtual void show(Graphics ^gr) = 0;

virtual void hide(Graphics ^gr) = 0;

virtual void move(Graphics ^gr, int dx, int dy) = 0;

virtual int getXmax() = 0;

virtual int getXmin() = 0;

virtual int getYmax() = 0;

virtual int getYmin() = 0;

};

point.h

#pragma once

#include "figure.h"

class point : public figure {

protected:

int x1, y1;

public:

point(int \_x1 = 0, int \_y1 = 0) {

x1 = \_x1;

y1 = \_y1;

}

virtual void show(Graphics ^gr) {

//gr->DrawEllipse(Pens::Black, x1 - 1, y1 - 1, 2, 2);

gr->FillEllipse(Brushes::Black, x1 - 1, y1 - 1, 2, 2);

}

virtual void hide(Graphics ^gr) {

//gr->DrawEllipse(Pens::White, x1 - 1, y1 - 1, 2, 2);

gr->FillEllipse(Brushes::White, x1 - 1, y1 - 1, 2, 2);

}

virtual void move(Graphics ^gr, int dx, int dy) {

hide(gr);

x1 += dx;

y1 += dy;

show(gr);

}

virtual int getXmax() {

return x1;

}

virtual int getXmin() {

return x1;

}

virtual int getYmax() {

return y1;

}

virtual int getYmin() {

return y1;

}

};

circle.h

#pragma once

#include "point.h"

class circle : public point {

protected:

int r;

public:

circle(int \_x1, int \_y1, int \_r) : point(\_x1, \_y1) {

r = \_r;

}

virtual void show(Graphics ^gr) {

//gr->DrawEllipse(Pens::Black, x1-r, y1-r, 2\*r, 2\*r);

gr->FillEllipse(Brushes::Yellow, x1 - r, y1 - r, 2 \* r, 2 \* r);

}

virtual void hide(Graphics ^gr) {

//gr->DrawEllipse(Pens::White, x1-r, y1-r, 2 \* r, 2 \* r);

gr->FillEllipse(Brushes::White, x1 - r, y1 - r, 2 \* r, 2 \* r);

}

virtual int getXmax() {

return x1 + r;

}

virtual int getXmin() {

return x1 - r;

}

virtual int getYmax() {

return y1 + r;

}

virtual int getYmin() {

return y1 - r;

}

};

line.h

#pragma once

#include "point.h"

class line : public point {

protected:

int x2, y2;

public:

line(int \_x1 = 0, int \_y1 = 0, int \_x2 = 0, int \_y2 = 0) : point(\_x1, \_y1) {

x2 = \_x2;

y2 = \_y2;

}

virtual void show(Graphics ^gr) {

gr->DrawLine(Pens::Black, x1, y1, x2, y2);

}

virtual void hide(Graphics ^gr) {

gr->DrawLine(Pens::White, x1, y1, x2, y2);

}

virtual void move(Graphics ^gr, int dx, int dy) {

hide(gr);

x1 += dx;

x2 += dx;

y1 += dy;

y2 += dy;

show(gr);

}

virtual int getXmax() {

if (x1 > x2) return x1;

else return x2;

}

virtual int getXmin() {

if (x1 < x2) return x1;

else return x2;

}

virtual int getYmax() {

if (y1 > y2) return y1;

else return y2;

}

virtual int getYmin() {

if (y1 < y2) return y1;

else return y2;

}

};

rectangle.h

#pragma once

#include "line.h"

class rectangle : public line {

public:

rectangle(int \_x1 = 0, int \_y1 = 0, int \_x2 = 0, int \_y2 = 0) : line(\_x1, \_y1, \_x2, \_y2) {}

virtual void show(Graphics ^gr) {

//gr->DrawRectangle(Pens::Black, x1, y1, x2 - x1, y2 - y1);

gr->FillRectangle(Brushes::Blue, x1, y1, x2 - x1, y2 - y1);

}

virtual void hide(Graphics ^gr) {

//gr->DrawRectangle(Pens::White, x1, y1, x2 - x1, y2 - y1);

gr->FillRectangle(Brushes::White, x1, y1, x2 - x1, y2 - y1);

}

virtual int getXmax() {

return x2;

}

virtual int getXmin() {

return x1;

}

virtual int getYmax() {

return y2;

}

virtual int getYmin() {

return y1;

}

};

FigureGroup.h

#pragma once

#include "figure.h"

class FigureGroup : public figure {

protected:

int count;

figure \*\*figures;

public:

FigureGroup(int \_count = 1) {

count = \_count;

figures = new figure\*[count];

}

~FigureGroup() {

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

delete[] figures[i];

}

delete[] figures;

}

void setfigure(figure \*f, int pos) {

figures[pos] = f;

}

figure\* getfigure(int pos) {

return figures[pos];

}

virtual void show(Graphics ^gr) {

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

figures[i]->show(gr);

}

}

virtual void hide(Graphics ^gr) {

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

figures[i]->hide(gr);

}

}

virtual void move(Graphics ^gr, int dx, int dy) {

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

figures[i]->move(gr, dx, dy);

}

}

virtual int getXmax() {

int res = figures[0]->getXmax();

for (int i = 1; i < count; i++) {

if (figures[i]->getXmax() > res) res = figures[i]->getXmax();

}

return res;

}

virtual int getXmin() {

int res = figures[0]->getXmin();

for (int i = 1; i < count; i++) {

if (figures[i]->getXmin() < res) res = figures[i]->getXmin();

}

return res;

}

virtual int getYmax() {

int res = figures[0]->getYmax();

for (int i = 1; i < count; i++) {

if (figures[i]->getYmax() > res) res = figures[i]->getYmax();

}

return res;

}

virtual int getYmin() {

int res = figures[0]->getYmin();

for (int i = 1; i < count; i++) {

if (figures[i]->getYmin() < res) res = figures[i]->getYmin();

}

return res;

}

};

MyForm.h

#pragma once

#include "rectangle.h"

#include "circle.h"

#include "FigureGroup.h"

namespace Графика {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

Graphics ^gr;

FigureGroup \*fg;

//point \*p;

//line \*p;

rectangle \*r;

circle \*c;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::Timer^ timer1;

private: System::Windows::Forms::PictureBox^ pictureBox1;

private: System::Windows::Forms::Button^ button5;

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

gr = CreateGraphics();

fg = new FigureGroup(4);

//p = new point(100, 100);

//p = new line(50, 100, 100, 100);

r = new rectangle(100, 100, 230, 150);

fg->setfigure(r, 0);

r = new rectangle(125, 75, 205, 100);

fg->setfigure(r, 1);

c = new circle(120, 165, 15);

fg->setfigure(c, 2);

c = new circle(210, 165, 15);

fg->setfigure(c, 3);

}

private: System::Windows::Forms::TextBox^ textBox1;

public:

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Button^ button2;

public:

private: System::Windows::Forms::Button^ button3;

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ button1;

private: System::ComponentModel::IContainer^ components;

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

this->button1 = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->button2 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->button4 = (gcnew System::Windows::Forms::Button());

this->timer1 = (gcnew System::Windows::Forms::Timer(this->components));

this->button5 = (gcnew System::Windows::Forms::Button());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->SuspendLayout();

//

// button1

//

this->button1->Location = System::Drawing::Point(1202, 141);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(75, 23);

this->button1->TabIndex = 0;

this->button1->Text = L"Show";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// textBox1

//

this->textBox1->Location = System::Drawing::Point(1206, 339);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(71, 22);

this->textBox1->TabIndex = 1;

this->textBox1->Text = L"1";

//

// textBox2

//

this->textBox2->Location = System::Drawing::Point(1206, 376);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(71, 22);

this->textBox2->TabIndex = 2;

this->textBox2->Text = L"1";

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(1159, 342);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(22, 17);

this->label1->TabIndex = 3;

this->label1->Text = L"dx";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(1159, 379);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(23, 17);

this->label2->TabIndex = 4;

this->label2->Text = L"dy";

//

// button2

//

this->button2->Location = System::Drawing::Point(1202, 181);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(75, 23);

this->button2->TabIndex = 5;

this->button2->Text = L"Hide";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(1202, 219);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(75, 23);

this->button3->TabIndex = 6;

this->button3->Text = L"Move";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// button4

//

this->button4->Location = System::Drawing::Point(1202, 259);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(75, 23);

this->button4->TabIndex = 7;

this->button4->Text = L"Ride";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &MyForm::button4\_Click);

//

// timer1

//

this->timer1->Interval = 1;

this->timer1->Tick += gcnew System::EventHandler(this, &MyForm::timer1\_Tick);

//

// button5

//

this->button5->Location = System::Drawing::Point(1202, 297);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(75, 23);

this->button5->TabIndex = 8;

this->button5->Text = L"Stop";

this->button5->UseVisualStyleBackColor = true;

this->button5->Click += gcnew System::EventHandler(this, &MyForm::button5\_Click);

//

// pictureBox1

//

this->pictureBox1->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->pictureBox1->Location = System::Drawing::Point(0, -1);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(1115, 500);

this->pictureBox1->TabIndex = 9;

this->pictureBox1->TabStop = false;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::SystemColors::ButtonFace;

this->ClientSize = System::Drawing::Size(1312, 498);

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->button5);

this->Controls->Add(this->button4);

this->Controls->Add(this->button3);

this->Controls->Add(this->button2);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->button1);

this->Name = L"MyForm";

this->Text = L"MyForm";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

fg->show(gr);

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

fg->hide(gr);

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

int dx = Convert::ToInt32(textBox1->Text);

int dy = Convert::ToInt32(textBox2->Text);

fg->move(gr, dx, dy);

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

timer1->Enabled = true;

}

private: System::Void timer1\_Tick(System::Object^ sender, System::EventArgs^ e) {

int dx = Convert::ToInt32(textBox1->Text);

int dy = Convert::ToInt32(textBox2->Text);

fg->move(gr, dx, dy);

if (((fg->FigureGroup::getXmax()) >= (/\*this->Width\*/ pictureBox1->Width)) || ((fg->FigureGroup::getXmin() <= 0))) {

textBox1->Text = Convert::ToString((-1)\*dx);

}

if (((fg->FigureGroup::getYmin() <= 0)) || ((fg->getYmax()) >= (/\*this->Height\*/ pictureBox1->Height))) {

textBox2->Text = Convert::ToString((-1)\*dy);

}

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

timer1->Enabled = false;

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

gr = pictureBox1->CreateGraphics();

}

};

}

MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void Main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Графика::MyForm form;

Application::Run(%form);

}