Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

По объектно-ориентированному программированию

«Разработка графического редактора на языке C++ с применением механизмов ООП»

Студент \_\_\_\_\_\_\_\_\_ Несмеянова Е.С.

Группа ПМ-21-1

Руководитель

к.т.н., доцент каф. АСУ \_\_\_\_\_\_\_\_\_ Кургасов В. В.

Липецк 2022 г

**Цель работы:** закрепить навыки использования механизмов ООП на примере реализации графического редактора.

**Задание кафедры:** реализовать на языке C++ редактор графической схемы. В ходе выполненной работы обязательно применение объектно-ориентированных возможностей языка C++: наследования и динамического полиморфизма. Каждый тип элемента схемы должен быть представлен в программе в виде отдельного класса, который наследован от базового класса «графический элемент» (имеющего чисто виртуальную функцию прорисовки). Также необходим один класс «поле рисования», который содержит все графические элементы и отвечает за вызов функций прорисовки. Хранение графических элементов осуществляется с использованием контейнеров стандартной библиотеки C++.

**Вариант 12:** Редактор позиций на футбольном поле.

**Исходный код**

DrawingField.cpp

#include "DrawingField.h"

void DrawingField::addForm\_for\_drawing(Form\_for\_drawing^ form)

{

forms.push\_front(form);

}

void DrawingField::drawForm\_for\_drawing(Graphics^ g)

{

if (!forms.empty()) {

for each (Form\_for\_drawing ^ form in forms) {

form->draw(g);

}

}

}

Form\_for\_drawing^ DrawingField::findForm\_for\_drawing(int x, int y)

{

Form\_for\_drawing^ result = nullptr;

for each (Form\_for\_drawing ^ form in forms)

if (form->find(x, y) != nullptr)

{

result = form;

forms.remove(form);

break;

}

return result;

}

void DrawingField::clear() {

forms.clear();

}

void DrawingField::deleteForm\_for\_drawing(Form\_for\_drawing^ form)

{

forms.remove(form);

}

DrawingField.h

#pragma once

#include <cliext/list>

#include "Form\_for\_drawing.h"

ref class DrawingField

{

private:

cliext::list<Form\_for\_drawing^> forms;

public:

void addForm\_for\_drawing(Form\_for\_drawing^ form);

void drawForm\_for\_drawing(Graphics^ g);

Form\_for\_drawing^ findForm\_for\_drawing(int x, int y);

void deleteForm\_for\_drawing(Form\_for\_drawing^ form);

void clear();

};

Football.cpp

#include "Football.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThreadAttribute]

void main(array<String^>^ args) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

IDZ\_oop::Football\_board form;

Application::Run(% form);

}

Football.h

#pragma once

#include "DrawingField.h"

#include "Form\_for\_drawing.h"

#include "Fut.h"

#include "Pole.h"

#include "Mach.h"

namespace IDZ\_oop {

constexpr int \_Cursor = 1;

constexpr int \_Fut = 3;

constexpr int \_Ladya = 5;

constexpr int \_Pole = 7;

constexpr int \_Mach = 8;

constexpr int \_Delete = 9;

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>

/// Сводка для Football\_board

/// </summary>

public ref class Football\_board : public System::Windows::Forms::Form

{

public:

Football\_board(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

maxpictureBox1Height = pictureBox1->Height;

maxpictureBox1Width = pictureBox1->Width;

isDown = false;

Bitmap^ bitmap = gcnew Bitmap(pictureBox1->Size.Width, pictureBox1->Size.Height);

Graphics^ g = Graphics::FromImage(bitmap);

g->FillRectangle(gcnew SolidBrush(Color::White), 0, 0, pictureBox1->Width, pictureBox1->Height);

pictureBox1->Image = bitmap;

draw();

cursorType = \_Cursor;

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~Football\_board()

{

if (components)

{

delete components;

}

}

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

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

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

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

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

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

private: System::Windows::Forms::Button^ button7;

private: System::Windows::Forms::Button^ button8;

private: System::Windows::Forms::Button^ button9;

private: System::Windows::Forms::Button^ button10;

private: System::Windows::Forms::Button^ button11;

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

private: System::Windows::Forms::TextBox^ textBox3;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

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

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

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

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

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

this->button7 = (gcnew System::Windows::Forms::Button());

this->button8 = (gcnew System::Windows::Forms::Button());

this->button9 = (gcnew System::Windows::Forms::Button());

this->button10 = (gcnew System::Windows::Forms::Button());

this->button11 = (gcnew System::Windows::Forms::Button());

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

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

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

this->SuspendLayout();

//

// pictureBox1

//

this->pictureBox1->Location = System::Drawing::Point(232, 7);

this->pictureBox1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->pictureBox1->Size = System::Drawing::Size(742, 847);

this->pictureBox1->TabIndex = 0;

this->pictureBox1->TabStop = false;

this->pictureBox1->Click += gcnew System::EventHandler(this, &Football\_board::pictureBox1\_Click\_1);

this->pictureBox1->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &Football\_board::pictureBox1\_MouseDown);

this->pictureBox1->MouseMove += gcnew System::Windows::Forms::MouseEventHandler(this, &Football\_board::pictureBox1\_MouseMove);

this->pictureBox1->MouseUp += gcnew System::Windows::Forms::MouseEventHandler(this, &Football\_board::pictureBox1\_MouseUp);

//

// button1

//

this->button1->Location = System::Drawing::Point(50, 155);

this->button1->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->button1->Size = System::Drawing::Size(140, 77);

this->button1->TabIndex = 1;

this->button1->Text = L"Курсор";

this->button1->UseVisualStyleBackColor = true;

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

//

// button2

//

this->button2->Location = System::Drawing::Point(1086, 508);

this->button2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->button2->Size = System::Drawing::Size(140, 77);

this->button2->TabIndex = 2;

this->button2->Text = L"Сохранить";

this->button2->UseVisualStyleBackColor = true;

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

//

// button3

//

this->button3->Location = System::Drawing::Point(50, 55);

this->button3->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->button3->Size = System::Drawing::Size(140, 77);

this->button3->TabIndex = 3;

this->button3->Text = L"Удалить";

this->button3->UseVisualStyleBackColor = true;

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

//

// button4

//

this->button4->Location = System::Drawing::Point(50, 278);

this->button4->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->button4->Size = System::Drawing::Size(140, 77);

this->button4->TabIndex = 4;

this->button4->Text = L"Удалить все";

this->button4->UseVisualStyleBackColor = true;

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

//

// button5

//

this->button5->Location = System::Drawing::Point(982, 62);

this->button5->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->button5->Size = System::Drawing::Size(244, 63);

this->button5->TabIndex = 5;

this->button5->Text = L"Футбольное поле";

this->button5->UseVisualStyleBackColor = true;

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

//

// button7

//

this->button7->Location = System::Drawing::Point(1929, 354);

this->button7->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->button7->Name = L"button7";

this->button7->Size = System::Drawing::Size(117, 68);

this->button7->TabIndex = 7;

this->button7->Text = L"Ладья";

this->button7->UseVisualStyleBackColor = true;

this->button7->Click += gcnew System::EventHandler(this, &Football\_board::button7\_Click);

//

// button8

//

this->button8->Location = System::Drawing::Point(1050, 287);

this->button8->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->button8->Name = L"button8";

this->button8->Size = System::Drawing::Size(118, 68);

this->button8->TabIndex = 8;

this->button8->Text = L"мяч";

this->button8->UseVisualStyleBackColor = true;

this->button8->Click += gcnew System::EventHandler(this, &Football\_board::button8\_Click);

//

// button9

//

//

this->button10->Location = System::Drawing::Point(1050, 159);

this->button10->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->button10->Name = L"button10";

this->button10->Size = System::Drawing::Size(118, 68);

this->button10->TabIndex = 10;

this->button10->Text = L"футболист";

this->button10->UseVisualStyleBackColor = true;

this->button10->Click += gcnew System::EventHandler(this, &Football\_board::button10\_Click);

//

// button11

//

this->textBox2->Location = System::Drawing::Point(982, 14);

this->textBox2->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

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

this->textBox2->Size = System::Drawing::Size(241, 26);

this->textBox2->TabIndex = 13;

this->textBox2->Text = L"Что добавить на поле\?";

this->textBox2->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// textBox3

//

this->textBox3->Location = System::Drawing::Point(13, 7);

this->textBox3->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(211, 26);

this->textBox3->TabIndex = 14;

this->textBox3->Text = L"Кнопки управления";

this->textBox3->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// Football\_board

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);

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

this->ClientSize = System::Drawing::Size(1370, 599);

this->Controls->Add(this->textBox3);

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

this->Controls->Add(this->button11);

this->Controls->Add(this->button10);

this->Controls->Add(this->button9);

this->Controls->Add(this->button8);

this->Controls->Add(this->button7);

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

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

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

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

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

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

this->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);

this->Name = L"Football\_board";

this->Text = L"Football";

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

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

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: Point DownPosition;

private: Point CurrentPosition;

private: Point PrevPosition;

private: DrawingField drawingField;

private: int maxpictureBox1Height;

private: int maxpictureBox1Width;

private: int cursorType;

private: Fut^ newFut;

private: Pole^ newPole;

private: Mach^ newMach;

private: Form\_for\_drawing^ movingForm\_for\_drawing;

private: bool isDown;

private: Form\_for\_drawing^ deletingForm\_for\_drawing;

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

}

private: System::Void pictureBox1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button10\_Click(System::Object^ sender, System::EventArgs^ e) {

cursorType = \_Fut;

}

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

cursorType = \_Delete;

}

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

cursorType = \_Cursor;

}

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

SaveFileDialog^ save = gcnew SaveFileDialog();

save->Title = "Сохранить";

save->Filter = "png files (\*.png)|\*.png|All files (\*.\*)|\*.\*";

if (save->ShowDialog() == Windows::Forms::DialogResult::OK)

{

pictureBox1->Image->Save(save->FileName);

}

}

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

drawingField.clear();

draw();

}

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

cursorType = \_Pole;

}

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) {

cursorType = \_Ladya;

}

private: System::Void button8\_Click(System::Object^ sender, System::EventArgs^ e) {

cursorType = \_Mach;

}

private: System::Void draw() {

Graphics^ g = Graphics::FromImage(pictureBox1->Image);

g->FillRectangle(gcnew SolidBrush(Color::White), 0, 0, pictureBox1->Width, pictureBox1->Height);

drawingField.drawForm\_for\_drawing(g);

pictureBox1->Invalidate();

}

private: System::Void pictureBox1\_MouseDown(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

isDown = true;

DownPosition = Point(e->X, e->Y);

PrevPosition = Point(e->X, e->Y);

draw();

switch (cursorType)

{

case \_Cursor:

movingForm\_for\_drawing = drawingField.findForm\_for\_drawing(e->X, e->Y);

if (movingForm\_for\_drawing != nullptr)

{

switch (movingForm\_for\_drawing->type)

{

case \_Fut:

movingForm\_for\_drawing = (Fut^)movingForm\_for\_drawing;

break;

case \_Pole:

movingForm\_for\_drawing = (Pole^)movingForm\_for\_drawing;

break;

case \_Mach:

movingForm\_for\_drawing = (Mach^)movingForm\_for\_drawing;

break;

}

}

break;

case \_Delete:

deletingForm\_for\_drawing = drawingField.findForm\_for\_drawing(e->X, e->Y);

if (deletingForm\_for\_drawing != nullptr)

drawingField.deleteForm\_for\_drawing(deletingForm\_for\_drawing);

deletingForm\_for\_drawing = nullptr;

break;

case \_Fut:

newFut = gcnew Fut(DownPosition);

newFut->draw(Graphics::FromImage(pictureBox1->Image));

break;

case \_Pole:

newPole = gcnew Pole(DownPosition);

newPole->draw(Graphics::FromImage(pictureBox1->Image));

break;

case \_Mach:

newMach = gcnew Mach(DownPosition);

newMach->draw(Graphics::FromImage(pictureBox1->Image));

break;

}

pictureBox1->Invalidate();

}

private: System::Void pictureBox1\_MouseMove(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

if (isDown)

{

CurrentPosition = Point(e->X, e->Y);

if (movingForm\_for\_drawing == nullptr) {

draw();

switch (cursorType)

{

case \_Cursor:

break;

case \_Fut:

newFut = gcnew Fut(DownPosition);

newFut->draw(Graphics::FromImage(pictureBox1->Image));

break;

case \_Pole:

newPole = gcnew Pole(DownPosition);

newPole->draw(Graphics::FromImage(pictureBox1->Image));

break;

case \_Mach:

newMach = gcnew Mach(DownPosition);

newMach->draw(Graphics::FromImage(pictureBox1->Image));

break;

}

}

else

{

changemovingForm\_for\_drawingPosition();

draw();

movingForm\_for\_drawing->draw(Graphics::FromImage(pictureBox1->Image));

PrevPosition = CurrentPosition;

}

pictureBox1->Invalidate();

}

}

private: System::Void pictureBox1\_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

isDown = false;

switch (cursorType)

{

case \_Cursor:

if (movingForm\_for\_drawing != nullptr) {

drawingField.addForm\_for\_drawing(movingForm\_for\_drawing);

movingForm\_for\_drawing = nullptr;

}

break;

case \_Fut:

drawingField.addForm\_for\_drawing(newFut);

break;

case \_Pole:

drawingField.addForm\_for\_drawing(newPole);

break;

case \_Mach:

drawingField.addForm\_for\_drawing(newMach);

break;

}

draw();

}

private: System::Void changemovingForm\_for\_drawingPosition()

{

Point p1;

Point p2;

int dX = CurrentPosition.X - PrevPosition.X;

int dY = CurrentPosition.Y - PrevPosition.Y;

switch (movingForm\_for\_drawing->type)

{

case \_Fut:

p1 = ((Fut^)movingForm\_for\_drawing)->getFut();

if (p1.X + dX < 0 || p1.X + dX > pictureBox1->Width || p1.Y + dY < 0 || p1.Y + dY > pictureBox1->Height)

return;

p1.X = p1.X + dX;

p1.Y = p1.Y + dY;

((Fut^)movingForm\_for\_drawing)->setFut(p1);

break;

case \_Pole:

p1 = ((Pole^)movingForm\_for\_drawing)->getPole();

if (p1.X + dX < 0 || p1.X + dX > pictureBox1->Width || p1.Y + dY < 0 || p1.Y + dY > pictureBox1->Height)

return;

p1.X = p1.X + dX;

p1.Y = p1.Y + dY;

((Pole^)movingForm\_for\_drawing)->setPole(p1);

break;

case \_Mach:

p1 = ((Mach^)movingForm\_for\_drawing)->getMach();

if (p1.X + dX < 0 || p1.X + dX > pictureBox1->Width || p1.Y + dY < 0 || p1.Y + dY > pictureBox1->Height)

return;

p1.X = p1.X + dX;

p1.Y = p1.Y + dY;

((Mach^)movingForm\_for\_drawing)->setMach(p1);

break;

break;

}

}

private: System::Void pictureBox1\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

}

};

}

Form\_for\_drawing.h

#pragma once

using namespace System::Drawing;

ref class Form\_for\_drawing abstract

{

public:

int type;

virtual void draw(Graphics^ g) = 0;

virtual Form\_for\_drawing^ find(int x, int y) = 0;

};

Fut.cpp

#include "Fut.h"

using namespace System::Drawing;

Fut::Fut(Point point) {

this->p = point;

type = 3;

}

Point Fut::getFut() {

return p;

}

void Fut::setFut(Point point) {

this->p = point;

}

void Fut::draw(Graphics^ g) {

Image^ img = Image::FromFile("футболист.png");

width = img->Width;

height = img->Height;

g->DrawImage(img, p);

}

Form\_for\_drawing^ Fut::find(int x, int y)

{

if (x >= p.X && x <= p.X + width) {

if (y >= p.Y && y <= p.Y + height)

return this;

}

return nullptr;

}

Fut.h

#pragma once

#include "Form\_for\_drawing.h"

ref class Fut :

public Form\_for\_drawing

{

private:

Point p;

int width, height;

public:

Fut(Point point);

Point getFut();

void setFut(Point point);

virtual void draw(Graphics^ g) override;

virtual Form\_for\_drawing^ find(int x, int y) override;

};

Mach.cpp

#include "Mach.h"

using namespace System::Drawing;

Mach::Mach(Point point) {

this->p = point;

type = 8;

}

Point Mach::getMach() {

return p;

}

void Mach::setMach(Point point) {

this->p = point;

}

void Mach::draw(Graphics^ g) {

Image^ img = Image::FromFile("мяч.png");

width = img->Width;

height = img->Height;

g->DrawImage(img, p);

}

Form\_for\_drawing^ Mach::find(int x, int y)

{

if (x >= p.X && x <= p.X + width) {

if (y >= p.Y && y <= p.Y + height)

return this;

}

return nullptr;

}

Mach.h

#pragma once

#include "Form\_for\_drawing.h"

ref class Mach :

public Form\_for\_drawing

{

private:

Point p;

int width, height;

public:

Mach(Point point);

Point getMach();

void setMach(Point point);

virtual void draw(Graphics^ g) override;

virtual Form\_for\_drawing^ find(int x, int y) override;

};

Pole.cpp

#include "Pole.h"

using namespace System::Drawing;

Pole::Pole(Point point) {

this->p = point;

type = 7;

}

Point Pole::getPole() {

return p;

}

void Pole::setPole(Point point) {

this->p = point;

}

void Pole::draw(Graphics^ g) {

Image^ img = Image::FromFile("поле.png");

width = img->Width;

height = img->Height;

g->DrawImage(img, p);

}

Form\_for\_drawing^ Pole::find(int x, int y)

{

if (x >= p.X && x <= p.X + width) {

if (y >= p.Y && y <= p.Y + height)

return this;

}

return nullptr;

}

Pole.h

#pragma once

#include "Form\_for\_drawing.h"

ref class Pole :

public Form\_for\_drawing

{

private:

Point p;

int width, height;

public:

Pole(Point point);

Point getPole();

void setPole(Point point);

virtual void draw(Graphics^ g) override;

virtual Form\_for\_drawing^ find(int x, int y) override;

};

**Результаты работы алгоритма**

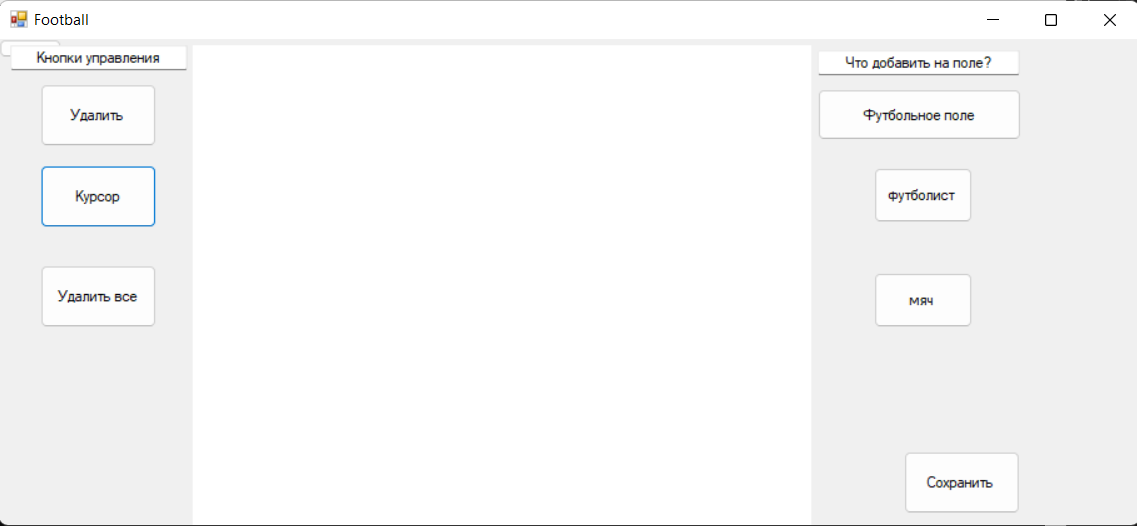


Рисунок 1. Начальное окно программы.

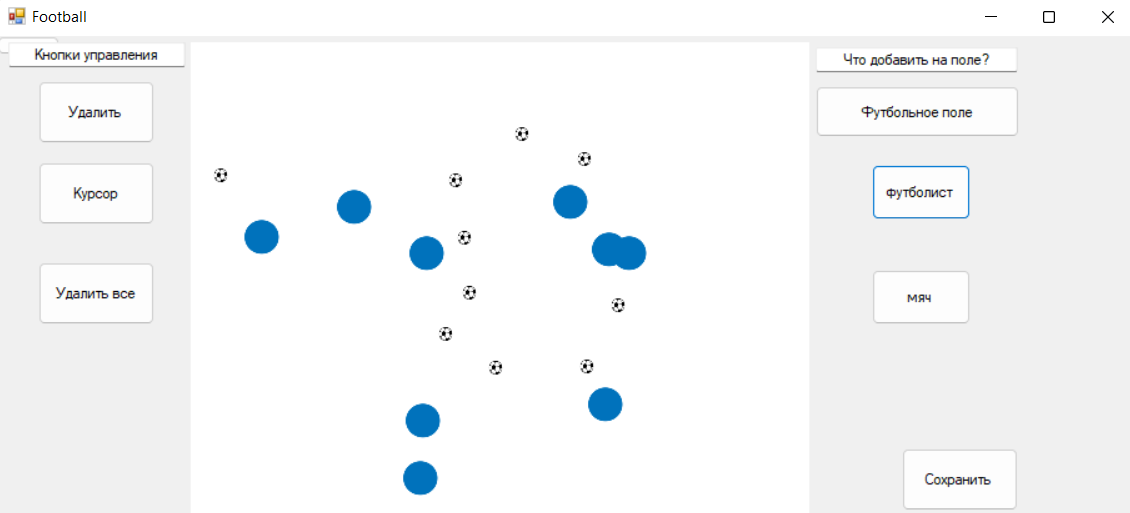


Рисунок 2. Пример расстановки футболистов.

Для того чтобы поставить игрока или мяч на поле нужно выбрать её и кликнуть ЛКМ на поле

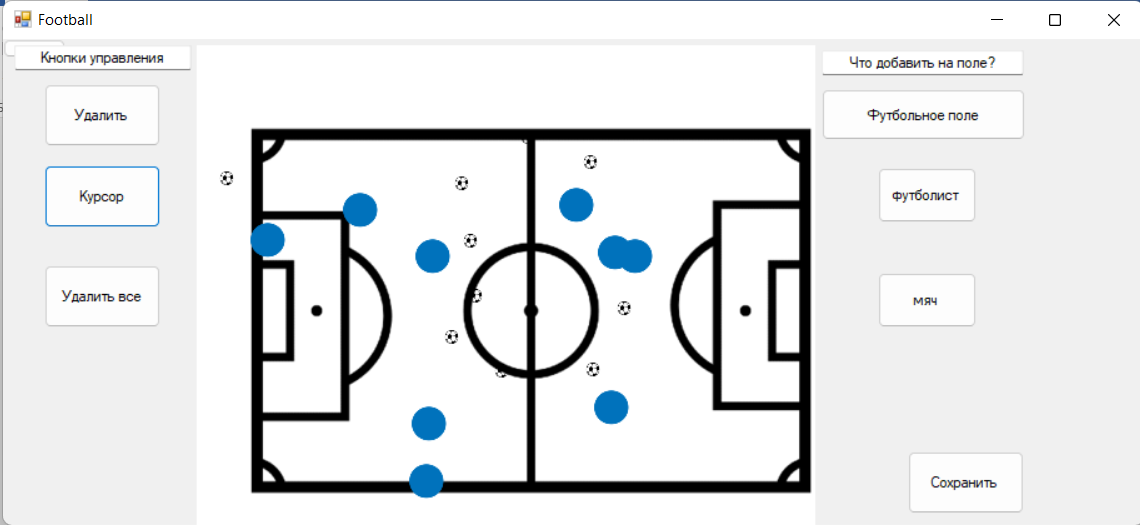


Рисунок 3. Пример расстановки с футбольным полем.

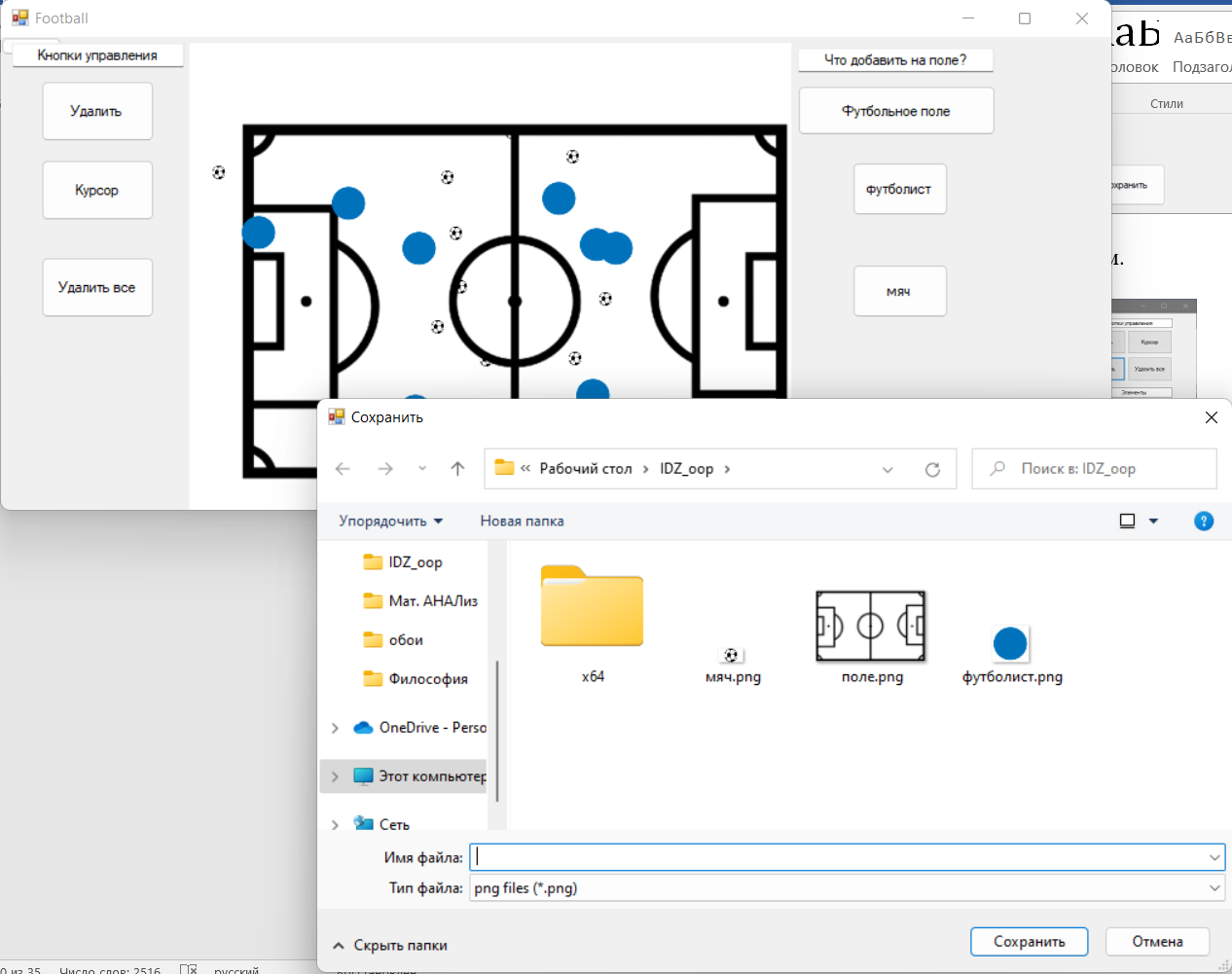


Рисунок 4. Пример сохранения изображения.

После того, как пользователь расставит фигуры, он может сохранить изображение в удобную для него папку.

**Ответы на контрольные вопросы**

**1. Зачем нужен перегруженный оператор присваивания?**

Перегруженный оператор присваивания используется для присваивания объектов одного конкретного класса друг другу, если объекты класса не являются обычными типами данных.

**2. Зачем нужен механизм наследования?**

Наследование – это механизм создания нового класса на основе уже существующего. Основное назначение механизма наследования — повторное использование кодов, так как большинство используемых типов данных являются вариантами друг друга, и писать для каждого свой класс нецелесообразно.

Объекты разных классов и сами классы могут находиться в отношении наследования, при котором формируется иерархия объектов, соответствующая заранее предусмотренной иерархии классов.

**3. Зачем используются модификаторы при наследовании классов? Какие это модификаторы?**

Существуют модификаторы доступа и наследования - public, private, protected.

В модификаторах доступа:

Public – доступ открыт всем, кто видит определение данного класса.

Protected – доступ открыт классам, производным от данного. То есть, производные классы получают свободный доступ к таким свойствам или метода. Все другие классы такого доступа не имеют.

Private – доступ открыт самому классу (т.е. функциям-членам данного класса) и друзьям (friend) данного класса – как функциям, так и классам. Однако производные классы не получают доступа к этим данным совсем. И все другие классы такого доступа не имеют.

В C++ существует public-наследование, private-наследование и protected-наследование. В зависимости от того, какой тип используется, изменяется доступ к членам базового класса для клиентов производного.

**4. Зачем нужен механизм полиморфизма?**

Полиморфизм – это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных. Тип данных, который используется при вызове функции, определяет, какая конкретная версия функции действительно выполняется. В С++ можно использовать одно имя функции для множества различных действий. Это называется перегрузкой функций. В более общем смысле, концепцией полиморфизма является идея «один интерфейс, множество методов».

**5. Что понимается под динамическим полиморфизме?**

Динамический полиморфизм предстает перед нами в форме классов с виртуальными функциями и объектов, работа с которыми осуществляется косвенно – через указатели или ссылки, в то время как статический полиморфизм включает шаблоны классов и функций.

**6. Что такое интерфейс класса?**

Интерфейс – это класс, который не имеет переменных-членов, и все методы которого являются чистыми виртуальными функциями.

**7. Зачем нужен чисто виртуальный метод?**

Чисто виртуальный метод является методом, который объявляется в базовом классе, но не имеет в нем определения. Поскольку она не имеет определения, то есть тела в этом базовом классе, то всякий производный класс обязан иметь свою собственную версию определения данного метода. При введении чисто виртуальной функции в производном классе обязательно необходимо определить свою собственную реализацию этой функции. Если класс не будет содержать определения этой функции, то компилятор выдаст ошибку.

**8. Какой класс называется абстрактным?**

Если класс имеет хотя бы одну чисто виртуальную функцию, то такой класс называется абстрактным. Важной особенностью абстрактных классов является то, что он не имеет объектов. Вместо этого абстрактный класс служит в качестве базового для других производных классов. Причина, по которой абстрактный класс не может быть использован для объявления объекта, заключается в том, что одна или несколько его функций-членов не имеют определения. Тем не менее, даже если базовый класс является абстрактным, все равно можно объявлять указатели или ссылки на него, с помощью которых затем поддерживается полиморфизм времени исполнения.

**Вывод:** в данном индивидуальном задании я закрепила навыки ООП и создала графический редактор позиций футболистов на футбольном поле.