Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Ім. Ігоря Сікорського

Факультет інформатики та обчислювальної техніки

**ЗВІТ**

з програмування № 3

на тему :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав студент** |  | *ІП-64 Вінницький В'ячеслав* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | Порєв В.М. |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2017

Ж = 2

Масив вказівників на фігури = Статичний

Гумовий слід = суцільна лінія червоного кольору

Прямокутник = Від центру до одного з кутів

Чорний контур з білим заповненням

Еліпс = По двом протилежним кутам охоплюючого прямокутника

Чорний контур еліпсу без заповнення

Позначка типу об’єкту в заголовці вікна

Editor.h

class Editor

{

public:

virtual void OnLBdown(HWND hwnd, int x, int y) = 0;

virtual void OnLBup(HWND hwnd, int x, int y) = 0;

virtual void OnMouseMove(HWND hwnd, int x, int y) = 0;

};

EllipsShape

#pragma once

#include "Shape.h"

class Ellips: public Shape

{

public:

void Draw(HDC hdc)

{

Arc(hdc, X1, Y1, X2, Y2, 0, 0, 0, 0);

}

};

EllipsEditor.cpp

#pragma once

#include "ShapeEditor.h"

#include "EllipsShape.h"

class EllipsEditor: public ShapeEditor

{

public:

void OnLBdown(HWND hwnd, int x, int y);

void OnLBup(HWND hwnd, int x, int y);

void OnMouseMove(HWND hwnd, int x, int y);

void NewShape()

{

shape = new Ellips();

}

void DrawShadow(HDC hdc);

};

LineShape.h

#pragma once

#include "Shape.h"

class Line : public Shape

{

public:

void Draw(HDC hdc)

{

MoveToEx(hdc, X1, Y1, nullptr);

LineTo(hdc, X2, Y2);

}

};

LineEditor.h

#pragma once

#include "ShapeEditor.h"

#include "LineShape.h"

class LineEditor: public ShapeEditor

{

public:

void OnLBdown(HWND hwnd, int x, int y);

void OnLBup(HWND hwnd, int x, int y);

void OnMouseMove(HWND hwnd, int x, int y);

void DrawShadow(HDC hdc);

void NewShape()

{

shape = new Line();

}

};

PointShape.h

#pragma once

#include "Shape.h"

class Point: public Shape

{

public:

void Draw(HDC hdc)

{

SetPixel(hdc, X1, Y1, RGB(255, 0, 0));

}

};

PointEditor.h

#pragma once

#include "Editor.h"

#include "ShapeEditor.h"

#include "PointShape.h"

class PointEditor: public ShapeEditor

{

public:

void OnLBdown(HWND hwnd, int x, int y) {}

void NewShape()

{

shape = new Point();

}

void OnLBup(HWND hwnd, int x, int y)

{

shape->SetX1Y1(x, y);

}

void OnMouseMove(HWND hwnd, int x, int y) {}

void DrawShadow(HDC hdc){}

};

RectangleEditor.h

#pragma once

#include "ShapeEditor.h"

#include "RectangleShape.h"

class RectangleEditor: public ShapeEditor

{

public:

void OnLBdown(HWND hwnd, int x, int y) override;

void OnLBup(HWND hwnd, int x, int y) override;

std::pair<int, int> SecondPointCoord(int x1, int y1, int x2, int y2)

{

return { 2 \* x1 - x2, 2 \* y1 - y2 };

}

void OnMouseMove(HWND hwnd, int x, int y) ;

void NewShape()

{

shape = new RectangleShape();

}

void DrawShadow(HDC hdc) ;

};

RectangleShape.h

#pragma once

#include "Shape.h"

class RectangleShape: public Shape

{

public:

void Draw(HDC hdc)

{

HBRUSH hBrush = HBRUSH(CreateSolidBrush(RGB(255,255,255)));

HBRUSH hBrushOld = HBRUSH(SelectObject(hdc, hBrush));

Rectangle(hdc, X1-(X2-X1),Y1-(Y2-Y1), X2, Y2);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

}

};

Shape.h

#pragma once

#include <utility>

class Shape

{

protected:

long X1 = 0, Y1 = 0, X2 = 0, Y2 = 0;

public:

virtual void Draw(HDC hdc) = 0;

void SetX1Y1(long x1, long y1)

{

X1 = x1;

Y1 = y1;

}

void SetX2Y2(long x2, long y2)

{

X2 = x2;

Y2 = y2;

}

long x1()

{

return X1;

}

long x2()

{

return X2;

}

long y1()

{

return Y1;

}

long y2()

{

return Y2;

}

};

ShapeEditor.h

#pragma once

#include "Editor.h"

#include "Shape.h"

class ShapeEditor: public Editor

{

protected:

Shape\* shape = nullptr;

bool lb = false;

public:

Shape\* ShapeR()

{

return shape;

}

virtual void NewShape() = 0;

virtual void DrawShadow(HDC hdc) = 0;

};

ShapeObjectEditor.h

#pragma once

#include "Editor.h"

#include <array>

#include "Shape.h"

#include "PointEditor.h"

#include "LineEditor.h"

#include "EllipsEditor.h"

#include "RectangleEditor.h"

class ShapeObjectsEditor

{

private:

ShapeEditor\* editor = nullptr;

Shape\* shapes[105];

int current\_shape = 0;

public:

void StartPointEditor()

{

if (editor)

{

delete editor;

}

editor = new PointEditor();

editor->NewShape();

}

void StartLineEditor()

{

if (editor)

{

delete editor;

}

editor = new LineEditor();

editor->NewShape();

}

void StartRectEditor()

{

if (editor)

{

delete editor;

}

editor = new RectangleEditor();

editor->NewShape();

}

void StartEllipseEditor()

{

if (editor)

{

delete editor;

}

editor = new EllipsEditor();

editor->NewShape();

}

void OnLBdown(HWND hwnd, int x, int y);

void OnLBup(HWND hwnd, int x, int y);

void OnMouseMove(HWND hwnd, int x, int y);

void OnPaint(HWND hwnd);

};

EllipsEditor.cpp

#include "stdafx.h"

#include "EllipsEditor.h"

void EllipsEditor::OnLBdown(HWND hwnd, int x, int y)

{

InvalidateRect(hwnd, nullptr, true);

shape->SetX1Y1(x, y);

shape->SetX2Y2(x, y);

lb = true;

}

void EllipsEditor::OnLBup(HWND hwnd, int x, int y)

{

shape->SetX2Y2(x, y);

lb = false;

}

void EllipsEditor::OnMouseMove(HWND hwnd, int x, int y)

{

if (lb)

{//черный контур без заполнения

//помаранчеве заповнення

HDC hdc = GetDC(hwnd);

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

HPEN hPenOld = HPEN(SelectObject(hdc, hPen));

SetROP2(hdc, R2\_NOTXORPEN);

DrawShadow(hdc);

shape->SetX2Y2(x, y);

DrawShadow(hdc);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hwnd, hdc);

}

}

void EllipsEditor::DrawShadow(HDC hdc)

{

shape->Draw(hdc);

}

LineEditor.cpp

#include "stdafx.h"

#include "LineEditor.h"

void LineEditor::OnLBdown(HWND hwnd, int x, int y)

{

InvalidateRect(hwnd, nullptr, true);

shape->SetX1Y1(x, y);

shape->SetX2Y2(x, y);

lb = true;

}

void LineEditor::OnLBup(HWND hwnd, int x, int y)

{

shape->SetX2Y2(x, y);

lb = false;

}

void LineEditor::OnMouseMove(HWND hwnd, int x, int y)

{

if (lb)

{

HDC hdc = GetDC(hwnd);

SetROP2(hdc, R2\_NOTXORPEN);

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

HPEN hPenOld = HPEN(SelectObject(hdc, hPen));

DrawShadow(hdc);

shape->SetX2Y2(x, y);

DrawShadow(hdc);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hwnd, hdc);

}

}

void LineEditor::DrawShadow(HDC hdc)

{

shape->Draw(hdc);

}

RectangleEditor.cpp

#include "stdafx.h"

#include "RectangleEditor.h"

void RectangleEditor::OnLBdown(HWND hwnd, int x, int y)

{

InvalidateRect(hwnd, nullptr, true);

shape->SetX1Y1(x, y);

shape->SetX2Y2(x, y);

lb = true;

}

void RectangleEditor::OnLBup(HWND hwnd, int x, int y)

{

const auto second\_point = SecondPointCoord(shape->x1(), shape->y1(), x, y);

shape->SetX2Y2(x, y);

lb = false;

}

void RectangleEditor::DrawShadow(HDC hdc)

{//черный цвет белое заполнение

// серое заполнение

MoveToEx(hdc, shape->x1() - (shape->x2() - shape->x1()), shape->y1() + (shape->y2() - shape->y1()), nullptr);

LineTo(hdc, shape->x2(), shape->y2());

MoveToEx(hdc, shape->x2(), shape->y2(), nullptr);

LineTo(hdc, shape->x1() + (shape->x2() - shape->x1()), shape->y1() - (shape->y2() - shape->y1()));

MoveToEx(hdc, shape->x1() - (shape->x2() - shape->x1()), shape->y1() - (shape->y2() - shape->y1()), nullptr);

LineTo(hdc, shape->x1() + (shape->x2() - shape->x1()), shape->y1() - (shape->y2() - shape->y1()));

MoveToEx(hdc, shape->x1() - (shape->x2() - shape->x1()), shape->y1() - (shape->y2() - shape->y1()), nullptr);

LineTo(hdc, shape->x1() - (shape->x2() - shape->x1()), shape->y1() + (shape->y2() - shape->y1()));

}

void RectangleEditor::OnMouseMove(HWND hwnd, int x, int y)

{

if (lb)

{

HDC hdc = GetDC(hwnd);

SetROP2(hdc, R2\_NOTXORPEN);

HPEN hPen = CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

HPEN hPenOld = HPEN(SelectObject(hdc, hPen));

DrawShadow(hdc);

shape->SetX2Y2(x, y);

DrawShadow(hdc);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hwnd, hdc);

}

}

ShapeObjectsEditor.cpp

#include "stdafx.h"

#include "ShapeObjectsEditor.h"

void ShapeObjectsEditor::OnLBdown(HWND hwnd, int x ,int y)

{

if (editor != nullptr)

{

editor->OnLBdown(hwnd, x, y);

}

}

void ShapeObjectsEditor::OnLBup(HWND hwnd, int x, int y)

{

if (editor != nullptr)

{

editor->OnLBup(hwnd, x, y);

if (shapes[current\_shape] != nullptr)

{

delete shapes[current\_shape];

}

shapes[current\_shape] = editor->ShapeR();

current\_shape++;

if (current\_shape > 105)

{

current\_shape = 0;

}

editor->NewShape();

InvalidateRect(hwnd, nullptr, true);

}

}

void ShapeObjectsEditor::OnMouseMove(HWND hwnd, int x, int y)

{

if (editor != nullptr)

{

editor->OnMouseMove(hwnd, x, y);

}

}

void ShapeObjectsEditor::OnPaint(HWND hwnd)

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

for(Shape\* shape: shapes)

{

if (shape)

{

shape->Draw(hdc);

}

}

EndPaint(hwnd, &ps);

}

Win32Project2.cpp

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Parse the menu selections:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case ID\_DOT:

choose\_shape.StartPointEditor();

SetWindowText(hWnd, L"DOT");

break;

case ID\_LINE:

choose\_shape.StartLineEditor();

SetWindowText(hWnd, L"LINE");

break;

case ID\_RECTANGLE:

choose\_shape.StartRectEditor();

SetWindowText(hWnd, L"RECTANGLE");

break;

case ID\_ELLIPSE:

choose\_shape.StartEllipseEditor();

SetWindowText(hWnd, L"ELLIPSE");

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_LBUTTONDOWN:

choose\_shape.OnLBdown(hWnd, GET\_X\_LPARAM(lParam), GET\_Y\_LPARAM(lParam));

break;

case WM\_LBUTTONUP:

choose\_shape.OnLBup(hWnd, GET\_X\_LPARAM(lParam), GET\_Y\_LPARAM(lParam));

break;

case WM\_MOUSEMOVE:

choose\_shape.OnMouseMove(hWnd, GET\_X\_LPARAM(lParam), GET\_Y\_LPARAM(lParam));

break;

case WM\_PAINT:

choose\_shape.OnPaint(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

Win32Project.h

#pragma once

#include "resource.h"

#include "stdafx.h"

Editor

Shape

EllipsShape

RectangleShape

PointShape.

LineShape

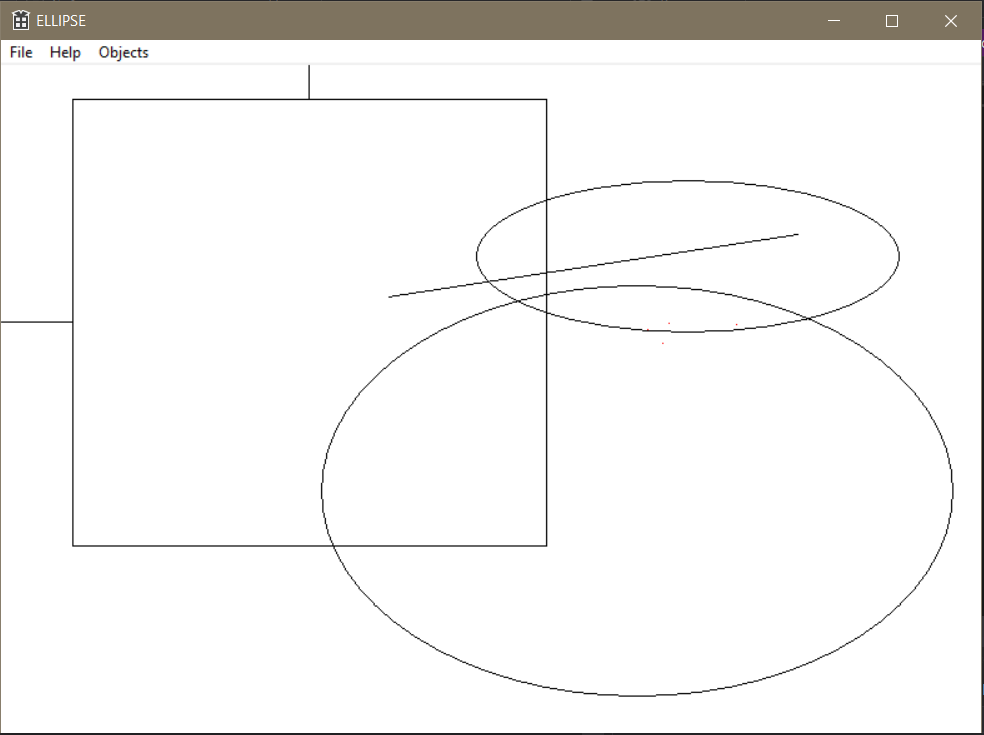
ShapeEditor

EllipsEditor

RectangleEditor

PointEditor

LinetEditor



Висновок  
за допомогою Win32API і функцій класів можно ефективно працювати з однодипними функціями для різних обьєктів.