# Министерство науки и высшего образования Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной информатики

### Лабораторная работа № 1 по дисциплине «Объектно-ориентированное программирование»

**Программирование графики**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |  |  |
| Группа: | ПМИ-81 |  |  |
| Студенты: | Демидович Е.Ю.,  Муравьев М.И. |  |  |
| Вариант: | 7 |  |  |
| Преподаватель: | Еланцева И.Л., Лисицин Д.В. |  |  |

Новосибирск

2019

1. **Постановка задачи**

Для фигуры-треугольника разработать следующие функции:

* рисование незакрашенной фигуры (фигуры-контура);
* рисование закрашенной фигуры;
* рисование двух, вложенных одна в другую, фигур, где внешняя должна быть закрашена, за исключением пространства второй.

Ввод параметров фигур (координат и др.), параметров рисуемых линий и закраски осуществлять из файлов (отдельно для каждого теста). Включить в программу проверки корректности данных (принадлежность всех точек одной прямой), включая проверки нахождения фигуры в пределах окна и вложенности двух фигур.

1. **Анализ задачи**

**Чтение из файла:**

Файл с исходной информацией имеет следующую структуру:

mode режим (0 – закрашенная фигура, 1 – контур, 2 – фигура с вырезом)

x[0] y[0]

x[1] y[1] координаты первого треугольника

x[2] y[2]

style стиль заливки (0-6)

r1 g1 b1 цвет заливки (три значения, каждое <256)

border bstyle ширина обводки (>=0), стиль кисти(0-6)

r2 g2 b2 цвет кисти(контура) (три значения, каждое <256)

x[0] y1[0]

x[1] y[1] координаты второго треугольника(выреза)

x[2] y[2] (не читается если m != 2)

**Проверка корректности данных:**

Если выполняется следующее равенство, то заданные точки лежат на одной прямой:

(x[2] – x[0]) \* (y[1] - y[0]) == (y[2] - [0].y) \* (x[1] - x[0]);

Если для каждой точки не выполняется следующее условие, то треугольник находится в пределах окна: i = [0,2]

x[i] < 0 || x[i] > rtx || y[i] < 0 || y[i] > rty , где rtx и rty – соответствующие размеры окна;

Если для каждой точки треугольника для выреза выполняется следующее условие, то треугольник не выходит за рамки первого треугольника:

a = (x[0] - x[i]) \* (y[1] - y[0]) - (x[1] - x[0]) \* (y[0] - y[i]);

b = (x[1] - x[i]) \* (y[2] - y[1]) - (x[2] - x[1]) \* (y[1] - y[i]);

c = (x[2] - x[i]) \* (y[0] - y[2]) - (x[0] - x[2]) \* (y[2] – y[i]);

((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <= 0))

x[i],y[i] – координаты точек второго треугольника, i = [0,2];

**Рисование фигур:**

Закрашенной:

Построение фигуры по заданным координатам, стилям и цветам заливки и обводки.

Контура:

Построение замкнутой линии по заданным точкам, стилю и цвету обводки.

С вырезом:

Сначала строится закрашенная фигура, затем внутри неё фигура с чёрным (так как фон контекста отображения консоли – чёрный) цветом обводки и заливки.

**Стили рисования фигур:**

Стиль кисти:

Определяется по следующим параметрам:

|  |  |  |
| --- | --- | --- |
| № Стиля | Описание | Пример |
| 0 | Сплошная |  |
| 1 | Горизонтальные линии | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-47-758.jpg |
| 2 | Вертикальные линии | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-48-988.jpg |
| 3 | Диагональная, с наклоном в 45 градусов вниз | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-53-272.jpg |
| 4 | Диагональная, с наклоном в 45 градусов вверх | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-54-689.jpg |
| 5 | Клеточная | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-56-097.jpg |
| 6 | Клеточная диагональная | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 09-49-57-425.jpg |

Стиль пера:

Определяется по следующим параметрам:

|  |  |  |
| --- | --- | --- |
| № Стиля | Описание | Пример |
| 0 | Сплошное | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-23-887.jpg |
| 1 | Пунктир | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-25-285.jpg |
| 2 | Точка | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-27-795.jpg |
| 3 | Пунктир-точка | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-29-289.jpg |
| 4 | Пунктир-точка-точка | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-31-027.jpg |
| 5 | Прозрачное | C:\Users\dom2\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bandicam 2019-09-20 10-06-33-768.jpg |

1. **Алгоритм**

**Структура Triangle:**

Массив точек points[3] – координаты точек треугольника

Целое число style – стиль заливки

Цветовой тип color – цвет заливки

Целое число border, bstyle – ширина обводки, стиль обводки

Цветовой тип bcolor – цвет обводки

**checkTriangle(\*triangle – указатель на треугольник):**

вернуть !((triangle->points[2].x - triangle->points[0].x) \* (triangle->points[1].y - triangle->points[0].y) == (triangle->points[2].y - triangle->points[0].y) \* (triangle->points[1].x - triangle->points[0].x));

**checkSizeT(\*triangle, rtx – размер окна по горизонтали, rty – размер по вертикали):**

Для (i = 0; i < 3; i++)

Если (triangle->points[i].x < 0 || triangle->points[i].x > rtx || triangle->points[i].y < 0 || triangle->points[i].y > rty)

Вернуть 0;

Вернуть 1;

**checkSizeTC(\*triangle, \*triangle2Cut – указатели на треугольники):**

Для (int i = 0; i < 3; i++) {

a = (triangle->points[0].x - triangle2Cut->points[i].x) \* (triangle->points[1].y - triangle->points[0].y) - (triangle->points[1].x - triangle->points[0].x) \* (triangle->points[0].y - triangle2Cut->points[i].y);

b = (triangle->points[1].x - triangle2Cut->points[i].x) \* (triangle->points[2].y - triangle->points[1].y) - (triangle->points[2].x - triangle->points[1].x) \* (triangle->points[1].y - triangle2Cut->points[i].y);

c = (triangle->points[2].x - triangle2Cut->points[i].x) \* (triangle->points[0].y - triangle->points[2].y) - (triangle->points[0].x - triangle->points[2].x) \* (triangle->points[2].y - triangle2Cut->points[i].y);

Если !((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <= 0)) Вернуть 0;

Вернуть 1;

**readInf(\*triangle, \*triangle2Cut – указатели на треугольники):**

Открытие файла input.txt

Если не открыт, вывод «Can not open input file»; вернуть -1

Чтение mod

Для(i = 0; i < 3; i++) чтение (triangle->points[i].x, triangle->points[i].y)

Чтение style

Для(i = 0; i < 3; i++) чтение (triangle->color[i])

Чтение border

Чтение bstyle

Для(i = 0; i < 3; i++) чтение (triangle->bcolor[i])

triangle->bcolor = RGB(bcolor[0], bcolor[1], bcolor[2])

triangle->color = RGB(color[0], color[1], color[2])

Если mode != 2 вернуть mode

Для(i = 0; i < 3; i++) чтение (triangle2Cut->points[i].x, triangle2Cut->points[i].y)

Закрыть файл

Вернуть mode

**hidecursor() – скрытие приглашения к вводу в консоли**

**drawTriangle(hdc – контекст отображения, \*triangle – указатель на треугольник):**

hMyPen = Создание пера(triangle->bstyle, triangle->border, triangle->bcolor)

hMyBrush = Создание кисти(triangle->style, triangle->style-1, triangle->color)

Выбор пера(hdc, hMyPen)

Выбор кисти(hdc, hMyBrush)

Построение фигуры(hdc, triangle->points, 3);

Удаление пера(hMyPen);

Удаление кисти(hMyBrush);

**drawCounter(hdc – контекст отображения, \*triangle – указатель на треугольник):**

hMyPen = Создание пера(triangle->bstyle, triangle->border, triangle->bcolor)

Массив точек ppt[4] = {triangle->points[0], triangle->points[1], triangle->points[2], triangle->points[0]}

Построение линии(hdc, ppt, 4);

Удаление пера(hMyPen);

**drawCut(hdc – контекст отображения, \*triangle2Cut – указатель на треугольник для выреза):**

hMyPen = Создание пера(0, 0, RGB(0,0,0))

hMyBrush = Создание кисти(RGB(0,0,0))

Выбор пера(hdc, hMyPen)

Выбор кисти(hdc, hMyBrush)

Построение фигуры(hdc, triangle2Cut->points, 3);

Удаление пера(hMyPen);

Удаление кисти(hMyBrush);

**main():**

Задание контекста отображения(hdc)

RECT rt;

hOldPen = Взятие стандартного пера(BLACK\_PEN);

hOldBrush = Взятие стандартной кисти(WHITE\_BRUSH);

oldx = 0; oldy = 0;

rtx = 0; rty = 0;

Triangle\* triangle = new Triangle;

Triangle\* triangle2put = new Triangle;

mode = readInf(triangle, triangle2put);

Если (!checkTriangle(triangle))

Вывод «Your triangle is wrong»

Ожидание ввода символа

Вернуть

Если (mode == 2 && !checkTriangle(triangle2put))

Вывод «Your 2nd triangle is wrong»

Ожидание ввода символа

Вернуть

Если (mode == 2 && !checkSizeTC(triangle, triangle2put))

Вывод «Your 2nd triangle doesn't fit in 1st one»

Ожидание ввода символа

Вернуть

c = 0;

Делать {

Получение размеров окна(hwnd, &rt);

rtx = rt.right;

rty = rt.bottom;

Если (oldx != rtx || oldy != rty)

system("CLS");

hidecursor();

oldx = rtx;

oldy = rty;

Если (checkSizeT(triangle, rtx, rty))

Sleep(50);

Если (mode)

0:

drawTriangle(hdc, triangle);

break;

1:

drawCounter(hdc, triangle);

break;

2:

drawTriangle(hdc, triangle);

drawСut(hdc, triangle2put);

break;

else

Вывод ("Increase window size(current is %d : %d)", rtx, rty);

Если (\_kbhit())

c = \_getch();

Sleep(50);

Пока (c != 27);

system("CLS");

Выбрать перо(hdc, hOldPen);

Выбрать кисть (hdc, hOldBrush);

Освободить контекст отображения(hwnd, hdc);

1. **Описание функций**

Булева функция checkTriangle(Triangle\* triangle) - проверка на принадлежность трех точек одной прямой;

Булева функция checkSizeT(Triangle\* triangle, int rtx, int rty) - проверка вместимости треугольника в окно текущего размера;

Булева функция checkSizeTC(Triangle\* triangle, Triangle\* triangle2Cut) - проверка вместимости треугольника для выреза в основной треугольник;

Целочисленная функция readInf(Triangle\* triangle, Triangle\* triangle2Cut) - чтение входных данных из файла;

hidecursor() – функция скрытия приглашения ко вводу в консоли;

drawTriangle(HDC hdc, Triangle\* triangle) – рисование закрашенного треугольника;

drawCounter(HDC hdc, Triangle\* triangle) – рисование контура;

drawСut(HDC hdc, Triangle\* triangle2Сut) – рисование треугольника с вырезом.

1. **Текст программы**

**oop1lab.cpp**

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

#include "triangle.h"

void main()

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

RECT rt;

HPEN hOldPen = GetStockPen(BLACK\_PEN);

HBRUSH hOldBrush = GetStockBrush(WHITE\_BRUSH);

int oldx = 0, oldy = 0;

int rtx = 0, rty = 0;

Triangle\* triangle = new Triangle;

Triangle\* triangle2put = new Triangle;

int mode = readInf(triangle, triangle2put);

if (!checkTriangle(triangle)) {

printf("\nYour triangle is wrong");

\_getch();

return;

}

if (mode == 2 && !checkTriangle(triangle2put)) {

printf("\nYour 2nd triangle is wrong");

\_getch();

return;

}

if (mode == 2 && !checkSizeTC(triangle, triangle2put)) {

printf("\nYour 2nd triangle doesn't fit in 1st one");

\_getch();

return;

}

char c = 0;

do {

GetClientRect(hwnd, &rt);

rtx = rt.right;

rty = rt.bottom;

if (oldx != rtx || oldy != rty)

{

system("CLS");

hidecursor();

oldx = rtx;

oldy = rty;

if (checkSizeT(triangle, rtx, rty)) {

Sleep(50);

switch (mode)

{

case 0:

drawTriangle(hdc, triangle);

break;

case 1:

drawCounter(hdc, triangle);

break;

case 2:

drawTriangle(hdc, triangle);

drawСut(hdc, triangle2put);

break;

}

}

else

printf\_s("Increase window size(current is %d : %d)", rtx, rty);

}

if (\_kbhit()) {

c = \_getch();

}

Sleep(50);

} while (c != 27);

system("CLS");

SelectPen(hdc, hOldPen);

SelectBrush(hdc, hOldBrush);

ReleaseDC(hwnd, hdc);

}

**triangle.h**

#pragma once

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

class Triangle {

public:

POINT points[3];

int style;

COLORREF color;

int border, bstyle;

COLORREF bcolor;

};

bool checkTriangle(Triangle\* triangle);

bool checkSizeT(Triangle\* triangle, int rtx, int rty);

bool checkSizeTC(Triangle\* triangle, Triangle\* triangle2Cut);

int readInf(Triangle\* triangle, Triangle\* triangle2Cut);

void hidecursor();

void drawTriangle(HDC hdc, Triangle\* triangle);

void drawCounter(HDC hdc, Triangle\* triangle);

void drawСut(HDC hdc, Triangle\* triangle2Сut);

**triangle.cpp**

#pragma once

#include "triangle.h"

bool checkTriangle(Triangle\* triangle) {

return !((triangle->points[2].x - triangle->points[0].x) \* (triangle->points[1].y - triangle->points[0].y) == (triangle->points[2].y - triangle->points[0].y) \* (triangle->points[1].x - triangle->points[0].x));

}

bool checkSizeT(Triangle\* triangle, int rtx, int rty) {

for (int i = 0; i < 3; i++) {

if (triangle->points[i].x < 0 || triangle->points[i].x > rtx || triangle->points[i].y < 0 || triangle->points[i].y > rty)

return 0;

}

return 1;

}

bool checkSizeTC(Triangle\* triangle, Triangle\* triangle2Cut) {

int a, b, c;

for (int i = 0; i < 3; i++) {

a = (triangle->points[0].x - triangle2Cut->points[i].x) \* (triangle->points[1].y - triangle->points[0].y) - (triangle->points[1].x - triangle->points[0].x) \* (triangle->points[0].y - triangle2Cut->points[i].y);

b = (triangle->points[1].x - triangle2Cut->points[i].x) \* (triangle->points[2].y - triangle->points[1].y) - (triangle->points[2].x - triangle->points[1].x) \* (triangle->points[1].y - triangle2Cut->points[i].y);

c = (triangle->points[2].x - triangle2Cut->points[i].x) \* (triangle->points[0].y - triangle->points[2].y) - (triangle->points[0].x - triangle->points[2].x) \* (triangle->points[2].y - triangle2Cut->points[i].y);

if (!((a >= 0 && b >= 0 && c >= 0) || (a <= 0 && b <= 0 && c <= 0))) return false;

}

return true;

}

int readInf(Triangle\* triangle, Triangle\* triangle2Cut) {

FILE\* f;

fopen\_s(&f, "input.txt", "r");

if (f == NULL) { printf\_s("Can not open input file"); return -1; };

int mode, bcolor[3], color[3];

fscanf\_s(f, "%d", &mode);

for (int i = 0; i < 3; i++)

fscanf\_s(f, "%d%d", &(triangle->points[i].x), &(triangle->points[i].y));

fscanf\_s(f, "%d", &(triangle->style));

for (int i = 0; i < 3; i++)

fscanf\_s(f, "%d", &(color[i]));

fscanf\_s(f, "%d", &(triangle->border));

fscanf\_s(f, "%d", &(triangle->bstyle));

for (int i = 0; i < 3; i++)

fscanf\_s(f, "%d", &(bcolor[i]));

triangle->bcolor = RGB(bcolor[0], bcolor[1], bcolor[2]);

triangle->color = RGB(color[0], color[1], color[2]);

if (mode != 2) return mode;

for (int i = 0; i < 3; i++)

fscanf\_s(f, "%d%d", &(triangle2Cut->points[i].x), &(triangle2Cut->points[i].y));

fclose(f);

return mode;

}

void hidecursor()

{

HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 25;

info.bVisible = FALSE;

SetConsoleCursorInfo(consoleHandle, &info);

}

void drawTriangle(HDC hdc, Triangle\* triangle) {

COLORREF buffcolor = triangle->bcolor;

if (triangle->border == 0)

buffcolor = triangle->color;

HPEN hMyPen = CreatePen(triangle->bstyle, triangle->border, buffcolor);

LOGBRUSH\* logBrush = new LOGBRUSH;

logBrush->lbStyle = (triangle->style == 0) ? BS\_SOLID : BS\_HATCHED;

logBrush->lbHatch = triangle->style - 1;

logBrush->lbColor = triangle->color;

HBRUSH hMyBrush = CreateBrushIndirect(logBrush);

SelectPen(hdc, hMyPen);

SelectBrush(hdc, hMyBrush);

Polygon(hdc, triangle->points, 3);

DeletePen(hMyPen);

delete logBrush;

DeleteBrush(hMyBrush);

}

void drawCounter(HDC hdc, Triangle\* triangle) {

HPEN hMyPen = CreatePen(triangle->bstyle, triangle->border, triangle->bcolor);

SelectPen(hdc, hMyPen);

POINT ppt[4] = { triangle->points[0],triangle->points[1],triangle->points[2],triangle->points[0] };

Polyline(hdc, ppt, 4);

DeletePen(hMyPen);

}

void drawСut(HDC hdc, Triangle\* triangle2Сut) {

HPEN hMyPen = CreatePen(0, 0, RGB(0, 0, 0));

HBRUSH hMyBrush = CreateSolidBrush(RGB(0, 0, 0));

SelectPen(hdc, hMyPen);

SelectBrush(hdc, hMyBrush);

Polygon(hdc, triangle2Сut->points, 3);

DeletePen(hMyPen);

DeleteBrush(hMyBrush);

}

1. **Тесты**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Пояснение** | **Результат** |
| 0  100 200  300 500  600 500  0  255 0 0  1 0  255 255 0 | Корректные входные данные.  Рисование закрашенного треугольника.  Изображение появляется после расширения окна. |  |
| 1  100 100  300 250  600 280  0  255 0 0  1 0  255 255 0 | Корректные входные данные.  Рисование контура треугольника. |  |
| 2  100 200  300 500  600 500  0  0 128 192  1 0  255 255 0  300 440  250 350  500 485 | Корректные входные данные.  Рисование треугольника с вырезом. |  |
| 0  100 200  100 500  100 400  0  0 128 192  1 0  255 255 0 | Некорректные входные данные.  Вершины треугольника лежат на одной прямой. |  |
| 2  100 200  300 500  600 500  0  255 0 0  1 0  255 255 0  50 440  250 40  500 485 | Некорректные входные данные.  Треугольник для выреза не находится внутри основного. |  |