Лабораторная работа № 1

по дисциплине комп. геометрия и графика

студента группы ИТ-32

Манукова Давида Альбертовича

Выполнение: \_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_

**Введение в GDI+**

Цель работы: Получение практических навыков построения и визуализации графических примитивов с помощью GDI+.

Содержание работы

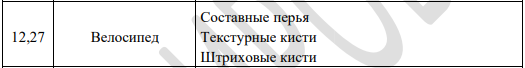
1. Изучить возможности графического вывода с помощью GDI+.
2. Разработать в Visual C++ оконное приложение Win32, на главном

окне которого:

* должен отображаться рисунок, состоящий из линий, кривых, многоугольников и закрашенных областей, в соответствии с вариантом задания;
* должно выводиться растровое изображение рисунка, указанного в варианте задания;
* должно выводиться название рисунка, указанного в варианте задания.

Обязательно использовать графические объекты, указанные в варианте задания. При необходимости можно также использовать и другие графические объекты.

1. Протестировать работу приложения, разработанного в п. 2. Результаты тестирования отразить в отчете. Включить в отчет исходный программный код и выводы о проделанной работе.



1. Включить в отчет исходный программный код и выводы о проделанной работе.

Ход работы

1. Разработал в Visual C++ оконное приложение, на окне которого выводится изображение велосипеда.

// Task1.cpp : Определяет точку входа для приложения.

//

#include "framework.h"

#include "Task1.h"

#include <windows.h>

#include <objidl.h>

#include <gdiplus.h>

#include "Resource.h"

using namespace Gdiplus;

#pragma comment (lib,"Gdiplus.lib")

#define MAX\_LOADSTRING 100

// Глобальные переменные:

GdiplusStartupInput gdiplusStartupInput;

ULONG\_PTR gdiplusToken;

Image\* textureimg;

Image\* bikeimg;

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

RECT clientRect;

HWND mhWnd;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

VOID Display(HDC);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_TASK1, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_TASK1));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

GdiplusShutdown(gdiplusToken);

return (int) msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_TASK1));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_TASK1);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

100, 100, 1280, 720, nullptr, nullptr, hInstance, nullptr);

textureimg = Image::FromFile(L"Road.png");

bikeimg = Image::FromFile(L"Bike.png");

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

mhWnd = hWnd;

return TRUE;

}

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

Display(hdc);

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

if (textureimg != NULL)

{

delete textureimg;

}

if (bikeimg != NULL)

{

delete bikeimg;

}

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

VOID Display(HDC hdc)

{

Graphics g(hdc);

g.Clear(Color::White);

g.SetSmoothingMode(SmoothingModeHighQuality);

// Текстурная кист

if (textureimg != NULL)

{

TextureBrush textureBrush(textureimg, WrapModeTile);

g.FillRectangle(&textureBrush, 0, 0, 1280, 640);

}

// Колеса

Pen compPen(Color::Black, 10.f);

float comp[6] = {

0.0f, 0.2f,

0.3f, 0.7f,

0.8f, 1.0f };

compPen.SetCompoundArray(comp, 6);

g.DrawEllipse(&compPen, 400, 400, 120, 120); //заднее колесо

g.DrawEllipse(&compPen, 620, 400, 120, 120); //переднее колесо

HatchBrush hatchBrush(HatchStyleOutlinedDiamond, Color::Gray, Color::White);

g.FillEllipse(&hatchBrush, 400, 400, 120, 120); //заднее колесо

g.FillEllipse(&hatchBrush, 620, 400, 120, 120); //переднее колесо

Pen redPen(Color::Red, 8.f);

redPen.SetStartCap(LineCapRound);

redPen.SetLineJoin(LineJoinRound);

redPen.SetEndCap(LineCapRound);

// Рама

Point framePoints[4] = {

Point(460, 460),

Point(550, 460),

Point(650, 360),

Point(490, 360)

};

g.DrawPolygon(&redPen, framePoints, 4);

// Руль

Point wheelPoints[3] = {

Point(680, 460),

Point(637, 320),

Point(610, 320)

};

g.DrawLines(&redPen, wheelPoints, 3);

// Сиденье

Point Wpt1(550, 460);

Point Wpt2(475, 340);

g.DrawLine(&redPen, Wpt1, Wpt2);

Pen blackPen(Color::Black, 8.f);

blackPen.SetStartCap(LineCapRound);

blackPen.SetEndCap(LineCapRound);

Point Qpt1(460, 340);

Point Qpt2(490, 340);

g.DrawLine(&blackPen, Qpt1, Qpt2);

// Вывод текста

Font font(L"Calibre", 20.f, FontStyleBoldItalic);

SolidBrush blackBrush(Color::Black);

StringFormat sf;

RectF rect(10.f, 10.f, 200.f, 80.f);

g.DrawString(L"Велосипед", -1, &font, rect, &sf, &blackBrush);

// Вывод изображения

if (bikeimg != NULL)

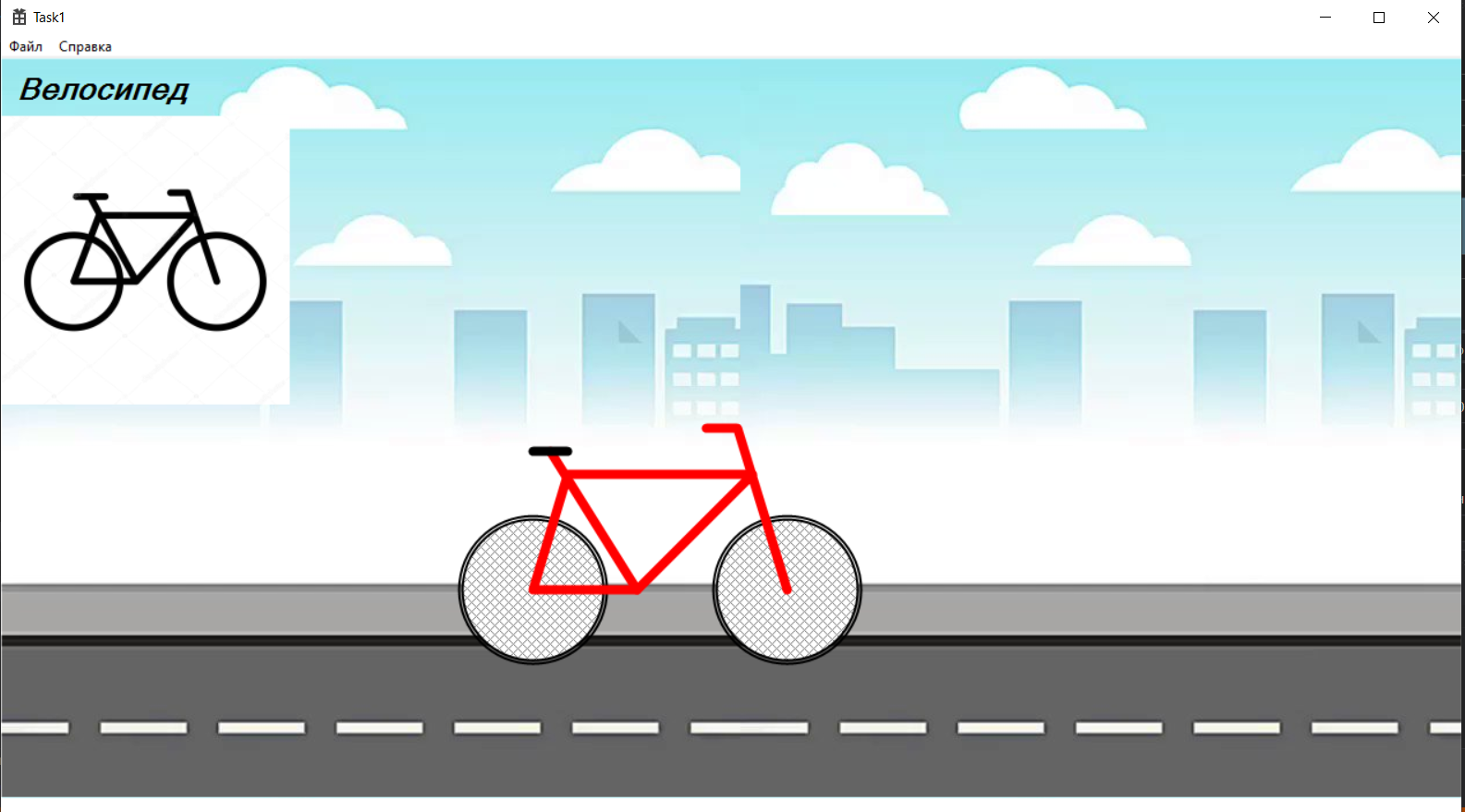
{

g.DrawImage(bikeimg, 0, 50, 250, 250);

}

}

1. Результаты работы



Вывод: получил практические навыки построения и визуализации графических примитивов с помощью GDI+.