ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ МОЛДОВЫ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, ИНФОРМАТИКИ И МИКРОЭЛЕКТРОНИКИ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЕТ

По лабораторной работе №2
по Програмированию для Windows
Тема: Графический интерфейс(GDI)

Выполнила: ст. гр. TI-155 Зверкова К.

Проверил: Скроб С.

Кишинев 2017

1. Цель лабораторной работы

Изучение основ работы и концепций GDI, работа с примитивами.

2. Теоретические понятия

Типы функций

В основном, функции GDI могут быть разбиты на несколько крупных групп. Это группы не имеют четких границ и частично перекрываются

• Функции, которые получают (или создают) и освобождают (или уничтожают) контекст устройства. Как уже указывалось в главе 3, вам для рисования необходим описатель контекста устройства. Функции GetDC и ReleaseDC позволяют вам сделать это внутри обработчиков сообщений, отличных от WM_PAINT. Функции BeginPaint и EndPaint (хотя технически — это часть подсистемы USER в Windows) используются в теле обработчика сообщения WM_PAINT для рисования. Некоторые другие функции, относящиеся к работе с контекстом устройства, мы рассмотрим ниже.

• Функции рисования. Очевидно, из всех предварительно рассмотренных функций — это одна из самых важных. В главе 3 мы использовали функцию TextOut для отображения текста в рабочей области окна. Как мы увидим далее, другие функции GDI позволяют рисовать линии, залитые области, растровые образы.

• Функции, которые устанавливают и получают атрибуты контекста устройства. Атрибут контекста устройства определяет различные особенности работы функции рисования. Например, вы используете функцию SetTextColor для задания любого текста, выводимого с использованием функции TextOut (или любой другой функции вывода текста). В программах SYSMETS в главе 3 мы использовали функцию SetTextAlign для того, чтобы сообщить GDI, что начальное положение текстовой строки при вызове функции TextOut должно быть справа, по умолчанию — левое начальное положение. Все атрибуты контекста устройства имеют значение по умолчанию, которое устанавливается, при получении контекста устройства. Для всех функций Set есть функции Get, позволяющие получить текущее значение атрибута контекста устройства.

Получение описателя контекста устройства

Windows предоставляет несколько методов для получения описателя контекста устройства. Если вы получаете описатель контекста устройства в теле обработчика сообщения, вы должны освободить его (удалить, вернуть системе) перед выходом из оконной процедуры. После того, как вы освободите описатель контекста устройства, его значение теряет смысл.

Наиболее общий метод получения контекста устройства и его освобождения состоит в использовании функций BeginPaint и EndPaint при обработке сообщения WM_PAINT:

```
hdc = BeginPaint(hwnd, &ps);

[другие строки программы]

EndPaint(hwnd, &ps);
```

Переменная ps — это структура типа PAINTSTRUCT. Поле hdc этой структуры — это описатель контекста устройства, который возвращается функцией BeginPaint. Структура PAINTSTRUCT содержит также структуру типа RECT с именем rcPaint (прямоугольник), определяющую прямоугольную область, содержащую недействительный (требующий перерисовки) регион клиентской области окна. Получив описатель контекста устройства от функции BeginPaint, вы можете рисовать только в пределах этого региона. Функция BeginPaint делает этот регион действительным. Программы для Windows могут также получать описатель контекста устройства в теле обработчика сообщения, отличного от WM_PAINT:

```
hdc = GetDC(hwnd);

[другие строки программы]

ReleaseDC(hwnd, hdc);
```

Полученный контекст устройства с описателем hwnd относится к клиентской (рабочей) области окна. Основная разница между использованием этих функций и комбинации функций BeginPaint и EndPaint состоит в том, что вы можете рисовать в пределах всей рабочей области окна, используя описатель контекста устройства, возвращенный функцией GetDC. Кроме того, функции GetDC и ReleaseDC не делают действительным (не

требующим перерисовки) ни один недействительный регион клиентской области окна. Программы для Windows могут также получать описатель контекста устройства, относящийся ко всему окну программы, а не только к его клиентской области:

hdc = GetWindowDC(hwnd);

[другие строки программы]

ReleaseDC(hwnd. hdc);

3. Задание

Нарисовать произвольный рисунок, используя все примитивы GDI

4. Листинг программы

```
#include <windows.h>

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

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int
iCmdShow) {

   static char szAppName[] = "HelloWin";

   HWND hwnd;

   MSG msg;

   WNDCLASSEX wndclass;

   wndclass.cbSize = sizeof(wndclass);

   wndclass.style = CS_HREDRAW | CS_VREDRAW;

   wndclass.lpfnWndProc = WndProc;

   wndclass.cbClsExtra = 0;

   wndclass.cbWndExtra = 0;

   wndclass.hInstance = hInstance;

   wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);

   wndclass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);

   wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);

   wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);

   wndclass.lpszMenuName = NULL;

   wndclass.lpszClassName = szAppName;

   RegisterClassEx(&wndclass);
```

```
    hwnd = CreateWindow(szAppName,
        "Lab #2 Program",
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance,
        NULL);
    ShowWindow(hwnd, iCmdShow);
    UpdateWindow(hwnd);
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam) {
    HDC hdc;
    PAINTSTRUCT ps;
    RECT rect;
    HBRUSH hBrush;
    POINT pt;
    POINT points1[4] = {
        { 541, 213 },
        { 661, 474 },
        { 426, 474 },
        { 541, 213 }
    };
    POINT points2[4] = {
        { 426, 474 },
        { 373, 474 },
        { 373, 294 },
        { 426, 474 }
    };
```

```
POINT points3[4] = {
   { 534, 180 },
   { 550, 180 },
   { 542, 190 },
   { 534, 180 }
};

POINT points4[5] = {
   { 705, 406 },
   { 836, 406 },
   { 813, 463 },
   { 730, 463 },
   { 705, 406 }
};

switch (iMsg) {

case WM_PAINT:

   hdc = BeginPaint(hwnd, &ps);
   GetClientRect(hwnd, &rect);

   //bowl
   hBrush = CreateSolidBrush(RGB(77, 166, 255));
   SelectObject(hdc, hBrush);

   Polygon(hdc, points4, 5);

   SetBkColor(hdc, RGB(77, 166, 255));
   TextOut(hdc, 760, 425, "CAT", 3);

   //flower
   hBrush = CreateSolidBrush(RGB(255, 179, 217));
   SelectObject(hdc, hBrush);
   Pie(hdc, 730, 252, 808, 306, 750, 251, 790, 251);

   HPEN hPen0 = CreatePen(PS_SOLID, 1, RGB(41, 163, 41));
   SelectObject(hdc, hPen0);

   MoveToEx(hdc, 770, 306, &pt);
   LineTo(hdc, 770, 408);

   MoveToEx(hdc, 769, 278, &pt);
   LineTo(hdc, 769, 259);

   MoveToEx(hdc, 770, 360, &pt);
   LineTo(hdc, 750, 340);

   MoveToEx(hdc, 770, 360, &pt);
   LineTo(hdc, 790, 340);


   HPEN hPen01 = GetStockObject(BLACK_PEN);
   SelectObject(hdc, hPen01);

   hBrush = CreateSolidBrush(RGB(204, 0, 102));
   SelectObject(hdc, hBrush);
   Ellipse(hdc, 762, 250, 776, 260);

   //billboard
   hBrush = CreateSolidBrush(RGB(204, 230, 255));
   SelectObject(hdc, hBrush);

   RoundRect(hdc, 246, 23, 830, 118, 737, 18);
```

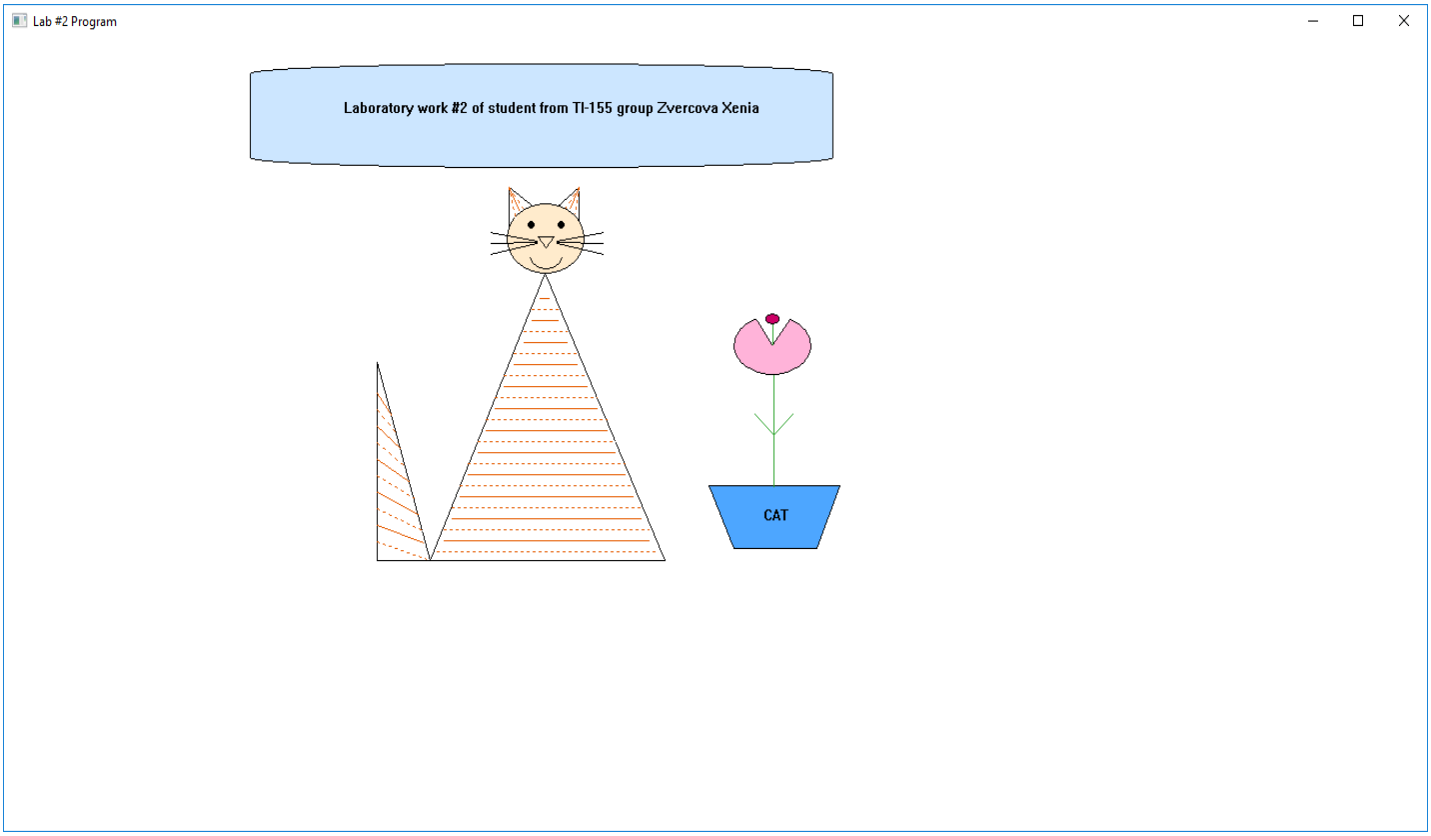
```
	SetBkColor(hdc, RGB(204, 230, 255));
	TextOut(hdc, 340, 55, "Laboratory work #2 of student from TI-155 group Zvercova Xenia",
62);

	//cat
	hBrush = CreateSolidBrush(RGB(255, 235, 204));
	SelectObject(hdc, hBrush);

	//face
	Ellipse(hdc, 503, 150, 581, 214);

	//eyes
	hBrush = CreateSolidBrush(RGB(0, 0, 0));
	SelectObject(hdc, hBrush);
	Ellipse(hdc, 524, 166, 531, 173);
	Ellipse(hdc, 554, 166, 561, 173);

	//mouth
	Arc(hdc, 525, 180, 560, 210, 520, 200, 570, 201);

	//trunk
	Polyline(hdc, points1, 4);

	//tail
	Polyline(hdc, points2, 4);

	//nose
	Polyline(hdc, points3, 4);

	//ears
	MoveToEx(hdc, 505, 170, &pt);
	LineTo(hdc, 505, 135);
	LineTo(hdc, 530, 153);

	MoveToEx(hdc, 575, 166, &pt);
	LineTo(hdc, 575, 135);
	LineTo(hdc, 553, 153);

	//mustaches
	MoveToEx(hdc, 553, 184, &pt);
	LineTo(hdc, 600, 176);

	MoveToEx(hdc, 553, 185, &pt);
	LineTo(hdc, 600, 186);

	MoveToEx(hdc, 553, 186, &pt);
	LineTo(hdc, 600, 196);

	MoveToEx(hdc, 533, 184, &pt);
	LineTo(hdc, 486, 176);

	MoveToEx(hdc, 533, 185, &pt);
	LineTo(hdc, 486, 186);

	MoveToEx(hdc, 533, 186, &pt);
	LineTo(hdc, 486, 196);

	//stripes - trunk
	SetBkColor(hdc, RGB(255, 255, 255));
	HPEN hPen1 = CreatePen(PS_DOT, 1, RGB(230, 92, 0));
	SelectObject(hdc, hPen1);

	MoveToEx(hdc, 528, 246, &pt);
	LineTo(hdc, 555, 246);
```

```
MoveToEx(hdc, 520, 266, &pt);
LineTo(hdc, 564, 266);

MoveToEx(hdc, 510, 286, &pt);
LineTo(hdc, 574, 286);

MoveToEx(hdc, 500, 306, &pt);
LineTo(hdc, 584, 306);

MoveToEx(hdc, 491, 326, &pt);
LineTo(hdc, 594, 326);

MoveToEx(hdc, 482, 346, &pt);
LineTo(hdc, 604, 346);

MoveToEx(hdc, 474, 366, &pt);
LineTo(hdc, 612, 366);

MoveToEx(hdc, 464, 386, &pt);
LineTo(hdc, 622, 386);

MoveToEx(hdc, 456, 406, &pt);
LineTo(hdc, 630, 406);

MoveToEx(hdc, 448, 426, &pt);
LineTo(hdc, 638, 426);

MoveToEx(hdc, 440, 446, &pt);
LineTo(hdc, 646, 446);

MoveToEx(hdc, 433, 466, &pt);
LineTo(hdc, 655, 466);

//stripes - tail
MoveToEx(hdc, 373, 337, &pt);
LineTo(hdc, 392, 358);

MoveToEx(hdc, 373, 367, &pt);
LineTo(hdc, 400, 388);

MoveToEx(hdc, 373, 397, &pt);
LineTo(hdc, 411, 418);

MoveToEx(hdc, 373, 427, &pt);
LineTo(hdc, 421, 448);

MoveToEx(hdc, 373, 457, &pt);
LineTo(hdc, 423, 473);

//stripes - ears
MoveToEx(hdc, 505, 135, &pt);
LineTo(hdc, 512, 165);

MoveToEx(hdc, 505, 135, &pt);
LineTo(hdc, 520, 155);

MoveToEx(hdc, 575, 135, &pt);
LineTo(hdc, 561, 155);

MoveToEx(hdc, 575, 135, &pt);
LineTo(hdc, 571, 160);


//stripes - trunk
HPEN hPen2 = CreatePen(PS_SOLID, 1, RGB(230, 92, 0));
```

```
SelectObject(hdc, hPen2);

MoveToEx(hdc, 536, 236, &pt);
LineTo(hdc, 546, 236);

MoveToEx(hdc, 528, 256, &pt);
LineTo(hdc, 555, 256);

MoveToEx(hdc, 520, 276, &pt);
LineTo(hdc, 564, 276);

MoveToEx(hdc, 510, 296, &pt);
LineTo(hdc, 574, 296);

MoveToEx(hdc, 500, 316, &pt);
LineTo(hdc, 584, 316);

MoveToEx(hdc, 491, 336, &pt);
LineTo(hdc, 594, 336);

MoveToEx(hdc, 482, 356, &pt);
LineTo(hdc, 604, 356);

MoveToEx(hdc, 474, 376, &pt);
LineTo(hdc, 612, 376);

MoveToEx(hdc, 464, 396, &pt);
LineTo(hdc, 622, 396);

MoveToEx(hdc, 456, 416, &pt);
LineTo(hdc, 630, 416);

MoveToEx(hdc, 448, 436, &pt);
LineTo(hdc, 638, 436);

MoveToEx(hdc, 440, 456, &pt);
LineTo(hdc, 646, 456);

//stripes - tail
MoveToEx(hdc, 373, 322, &pt);
LineTo(hdc, 388, 343);

MoveToEx(hdc, 373, 352, &pt);
LineTo(hdc, 397, 373);

MoveToEx(hdc, 373, 382, &pt);
LineTo(hdc, 406, 403);

MoveToEx(hdc, 373, 412, &pt);
LineTo(hdc, 415, 433);

MoveToEx(hdc, 373, 442, &pt);
LineTo(hdc, 420, 458);

//stripes - ears
MoveToEx(hdc, 505, 135, &pt);
LineTo(hdc, 516, 157);

MoveToEx(hdc, 575, 135, &pt);
LineTo(hdc, 566, 155);

EndPaint(hwnd, &ps);

break;
```

```
    case WM_DESTROY:

        PostQuitMessage(0);

        return 0;
    }

    return DefWindowProc(hwnd, iMsg, wParam, lParam);

}
```

5. Результат работы прогрммы



Вывод: В данной лабораторной работе были изучены основы работы и концепции GDI, была осуществлена работа с примитивами. Была реализована программа, результатом работы которой является рисунок, показанный выше. В работе применяялись такие примитивы, как Ellipse(), Polyline(), LineTo(), MoveToEx(),TextOut(), RoundRect(), Pie() и др.