Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

ЛАБОРАТОРНАЯ РАБОТА 2

по дисциплине «Операционные системы и оболочки»

## Обработка сообщений от элементов управления экранных форм

Студент группы 6ИСб-1 И. Нозимзода

Преподаватель В.Я Столяров

**Цель работы**: Разместить элементы формы и настроить обработчики событий на элементы.

**Задание**: Меню (двухуровневое), выбирает вид фигуры, кнопка – рисует эту фигуру на графическом окне. Другая кнопка – стирает эту фигуру. Спиннер выбирает толщину линии.

**Выполнение:**

Текст программы приведен в листинге 1.

Листинг 1 – Текст файла Prog2.cpp

|  |
| --- |
| #include <windows.h>  #include "resource.h"  #include <string>  #include <CommCtrl.H>  LRESULT CALLBACK PviewDlgProc(HWND hWnd, UINT wMsg, WPARAM wParam, LPARAM lParam);  int WINAPI WinMain  (  \_In\_ HINSTANCE hInstance, // дескриптор экземпляра приложения. Этот дескриптор содержит адрес начала кода программы в ее адресном пространстве. Дескриптор hInstance чаще всего требуется функциям, работающим с ресурсами программы  \_In\_opt\_ HINSTANCE, // дескриптор предыдущего экземпляра приложения. Этот дескриптор остался от старых версий Windows - скорее всего, вам он никогда не пригодится. Всегда равен 0  \_In\_ LPSTR, // указатель на начало командной строки, введенной при запуске программы  \_In\_ int // это значение содержит желаемый вид окна (например, свернутый или развернутый)  )  {  DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG1), 0, PviewDlgProc);  return 0;  }  // Процедура обработки сообщений диалогового окна  LRESULT CALLBACK PviewDlgProc  (  HWND hWnd, // уникальный идентификатор окна от которого пришло сообщение  UINT wMsg, // сообщение, получаемое от ОС  WPARAM wParam, // дополнительная информация сообщения  LPARAM lParam // дополнительная информация сообщения  )  {  switch (wMsg)  {  case WM\_INITDIALOG:  {  // Устанавливаем стандартные значения  SendDlgItemMessage(hWnd, IDC\_SPIN2, UDM\_SETRANGE32, 1, 50);  CheckRadioButton(hWnd, IDC\_RADIO1, IDC\_RADIO3, IDC\_RADIO1);  SendDlgItemMessage(hWnd, IDC\_SPIN2, UDM\_SETPOS32, 0, 1);  SetWindowText(GetDlgItem(hWnd, IDC\_EDIT1), "1");  }  break;  // Сообщение при закрытии диалогового окна  case WM\_CLOSE:  PostQuitMessage(0);  break;  // Сообщение от элементов диалогового окна  case WM\_COMMAND:  {  switch (LOWORD(wParam))  {  case ID\_BUTTON\_CIRCLE:  {  CheckRadioButton(hWnd, IDC\_RADIO1, IDC\_RADIO3, IDC\_RADIO1);  }  break;  case ID\_BUTTON\_RECTANGLE:  {  CheckRadioButton(hWnd, IDC\_RADIO1, IDC\_RADIO3, IDC\_RADIO2);  }  break;  case ID\_BUTTON\_ELLIPSE:  {  CheckRadioButton(hWnd, IDC\_RADIO1, IDC\_RADIO3, IDC\_RADIO3);  }  break;  case IDOK:  {  InvalidateRect(hWnd, NULL, TRUE);  UpdateWindow(hWnd);  HDC hdc = GetDC(hWnd);  // Установка пера  int widthPen = SendDlgItemMessage(hWnd, IDC\_SPIN2, UDM\_GETPOS32, 0, 0);  HPEN pen = CreatePen(PS\_SOLID, widthPen, RGB(0, 255, 0));  SelectObject(hdc, pen);  // Определение центра окна  RECT rect;  GetClientRect(hWnd, &rect);  LONG dlgCenterWidth = (rect.right - (float)rect.left) /1.9;  LONG dlgCenterHeight = (rect.bottom - (float)rect.top) / 1.4;  const size\_t RADIUS = 100;  if (IsDlgButtonChecked(hWnd, IDC\_RADIO1))  {  Ellipse  (  hdc,  dlgCenterWidth - RADIUS,  dlgCenterHeight + RADIUS,  dlgCenterWidth + RADIUS,  dlgCenterHeight - RADIUS  );  }  else if (IsDlgButtonChecked(hWnd, IDC\_RADIO2))  {  Rectangle  (  hdc,  dlgCenterWidth - RADIUS,  dlgCenterHeight + RADIUS / 2,  dlgCenterWidth + RADIUS,  dlgCenterHeight - RADIUS / 2  );  }  else if (IsDlgButtonChecked(hWnd, IDC\_RADIO3))  {  Ellipse  (  hdc,  dlgCenterWidth - RADIUS,  dlgCenterHeight + RADIUS / 2,  dlgCenterWidth + RADIUS,  dlgCenterHeight - RADIUS / 2  );  }  // Очистка ресурсов  DeleteObject(pen);  ReleaseDC(hWnd, hdc);  }  break;  case IDCANCEL:  {  InvalidateRect(hWnd, NULL, TRUE);  }  break;  default:  return FALSE;  }  break;  }  // Вертикальное изменение спина  case WM\_VSCROLL:  {  // Меняем поле информирующее пользователя о ширине линии кисти  int widthPen = SendDlgItemMessage(hWnd, IDC\_SPIN2, UDM\_GETPOS32, 0, 0);  SetWindowText(GetDlgItem(hWnd, IDC\_EDIT1), std::to\_string(widthPen).c\_str());  }  default:  return FALSE;  }  return TRUE;  } |

Результат работы программы приведены на рисунках 1 ‑ 4.

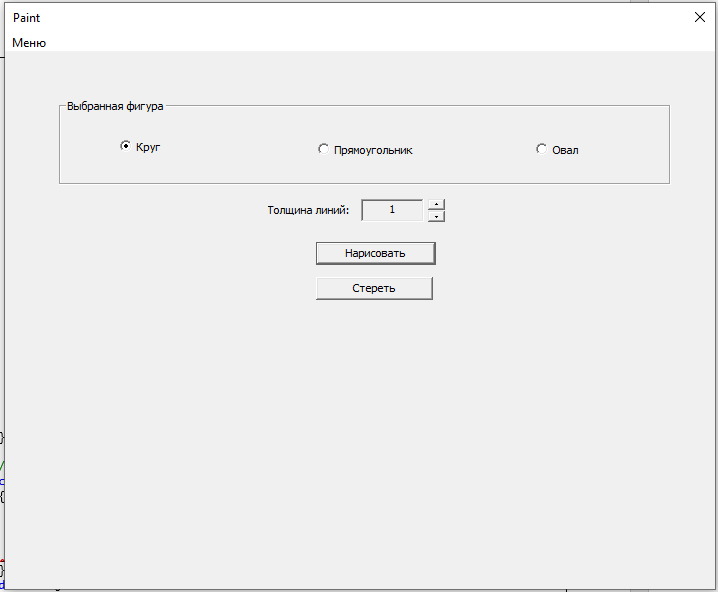


Рисунок 1 – Начальное окно программы

В двухуровневом меню происходит выбор типа фигуры (рисунок 2):

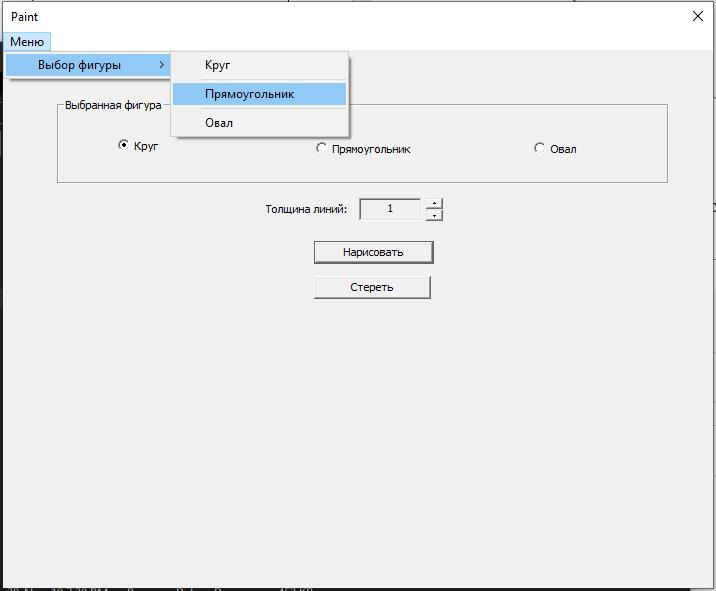


Рисунок 2 – Двухуровневое меню

Вертикальным спинером можно регулировать толщину линий фигур (рисунок 3):

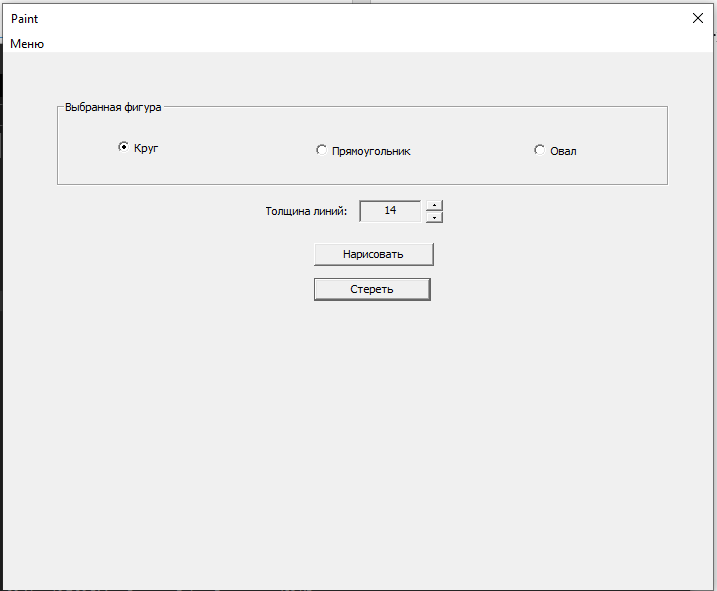


Рисунок 3 – Увеличение толщины линий вертикальным спинером

На кнопку «Нарисовать» происходит отрисовка фигуры выбранного типа, с выбранной толщиной линий (рисунок 4):

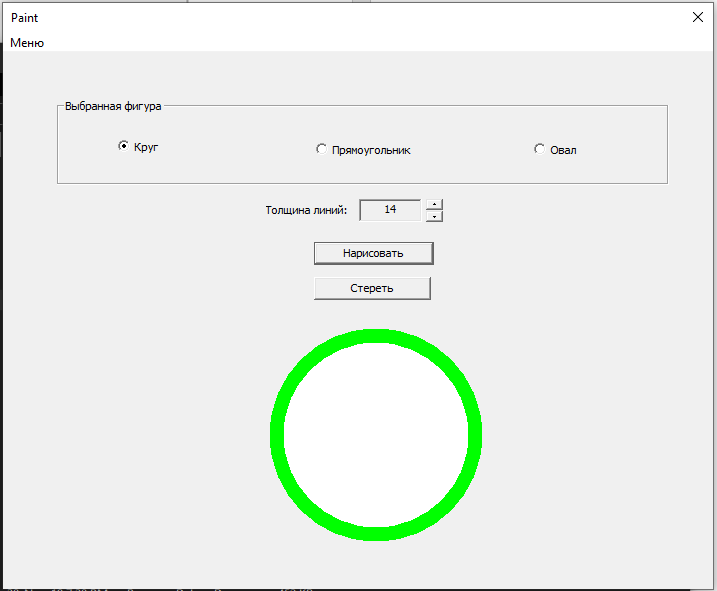


Рисунок 4 – Отрисовка фигуры

**Список использованных источников**

1. Саймон, Р. Windows 2000 API Энцеклопедия программиста / Р. Саймон. - М: DiaSoft, 2002