Win API vs. GLUT

- OpenGL applications can be written using both Windows API or GLUT
- GLUT
 - Makes it possible to produce one source code for all operating systems supporting OpenGL
 - Has simple but enough window interface (mouse and keyboard input, windows, menus, etc.)
- Windows API
 - Offers so called WGL to port 3D OpenGL API into 2D Windows API
 - Gives all Windows features
- WE WILL BE USING WINDOWS API

Windows Programming – basic steps

- Create a new folder for project files. Prepare project.
- Create resources.
- Create source code managing resources.
- Compile, link and run application.

Windows Prog.- project creation (1)

- File -> New -> Project
- Choose Win32 Project



Windows Prog.– project creation (2)

- Press <<Next>> in Wizard
- Choose Empty Project Windows Application
- Press <<Finish>>

Windo

Win32 Application Wizard - w	in1	<u>?</u> ×
Application	on Settings	
Overview Application Settings Specify the project and	Application type:	Add common header files for:
Application		Empty Project

Windows Prog.- project creation (3)

- Project -> Add New Item -> C++ Source File
- Set File Name, check future location.

Choose

Add New Item - win1		? >
<u>C</u> ategories:	<u>T</u> emplates:	
✓ Visual C++ UI Code Data Resource Web Utility Property Sheets Creates a file containing C++	Visual Studio installed templates Windows Form C++ File (.cpp) HTML Page (.htm) Static Discovery File (.disco) Header File (.h) Midl File (.idl) Resource File (.rc) Server Response File (.srf) Module-Definition File (.def) Registration Script (.rgs)	
Name: <a>	ame>	
Logation: d:_ssam	olej\Dydaktyka\grafika_12mir\win32\my_src\win1\win1	<u>B</u> rowse
	Add	Cancel
Source	Put File Nar	me

A Simple Win23 Programme

#include <windows h> if(!RegisterClassEx(&wc)) const char g szClassName[] = "myWindowClass"; {MessageBox(NULL, "Window Registration Failed!", "Error!", MB ICONEXCLAMATION | MB OK); LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM IParam) return 0: } HDC HDCPaint; PAINTSTRUCT PaintStruct; RECT rect; int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, switch(msg) I PSTR // Step 2: Creating the Window case WM CLOSE: hwnd = CreateWindowEx(DestroyWindow(hwnd); break; case WM DESTROY: WS_EX_CLIENTEDGE, PostQuitMessage(0); break; g szClassName, case WM PAINT: "The title of my window". WS OVERLAPPEDWINDOW, HDCPaint=BeginPaint(hwnd,&PaintStruct); CW_USEDEFAULT, CW_USEDEFAULT, 240, 120, GetClientRect(hwnd,&rect); DrawText(HDCPaint,"Hello World!",-1,&rect, NULL, NULL, hInstance, NULL); DT SINGLELINE DT CENTERIDT VCENTER); if(hwnd == NULL)EndPaint(hwnd, &PaintStruct); break; default: MessageBox(NULL, "Window Creation Failed!", "Error!", MB ICONEXCLAMATION | MB OK); return DefWindowProc(hwnd, msg, wParam, IParam); return 0: return 0; ShowWindow(hwnd, nCmdShow); int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, UpdateWindow(hwnd); LPSTR lpCmdLine, int nCmdShow) // Step 3: The Message Loop WNDCLASSEX wc; while(GetMessage(&Msg, NULL, 0, 0) > 0) HWND hwnd: TranslateMessage(&Msg); MSG Msg; DispatchMessage(&Msg); //Step 1: Registering the Window Class wc.cbSize return Msg.wParam; = sizeof(WNDCLASSEX); The title of my window wc.style = CS_HREDRAW | CS_VREDRAW; wc.lpfnWndProc = WndProc: wc.cbClsExtra = 0; wc.cbWndExtra = 0: wc.hlnstance = hInstance; = LoadIcon(NULL, IDI APPLICATION); wc.hlcon wc.hCursor = LoadCursor(NULL, IDC_ARROW); Hello World! wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1); wc.lpszMenuName = NULL; wc.lpszClassName = g szClassName; wc hlconSm = LoadIcon(NULL, IDI_APPLICATION);

A Simple Win32 Programm Creation Steps

- 4 Steps:
 - Registering the Window Class
 - Creating the Window
 - The Message Loop
 - The Window Procedure.

Registering the Window Class

const char g_szClassName[] = "myWindowClass"; WNDCLASSEX wc;

//Step 1: Registering the Window Class

wc.cbSize	= sizeof(WNDCLASSEX); //size of the structure
wc.style	= CS_HREDRAW CS_VREDRAW; //class styles
wc.lpfnWndProc	= WndProc; //Pointer to the window procedure for this window class
wc.cbClsExtra	= 0; // extra data allocated for class
wc.cbWndExtra	= 0; // extra data allocated for window
wc.hInstance	= hInstance; // handle to application instance
wc.hlcon	= LoadIcon(NULL, IDI_APPLICATION); //The big icon
wc.hCursor	= LoadCursor(NULL, IDC_ARROW); // Coursor
wc.hbrBackground	= (HBRUSH)(COLOR_WINDOW+1); // Background Colour
wc.lpszMenuName	= NULL; // Name of a menu resource
wc.lpszClassName	= g_szClassName; // Name to identify the class with
wc.hlconSm	= LoadIcon(NULL, IDI_APPLICATION); // The small icon
// The class defined	must be registered in the system:
if(!RegisterClassEx	(&wc))
{	
MessageBox(N	NULL, "Window Registration Failed!", "Error!",
MB_ICON	NEXCLAMATION MB_OK);
return 0;	

}

Creating the Window

// Step 2: Creating the Window
HWND hwnd;

```
hwnd = CreateWindowEx(
    WS EX CLIENTEDGE, // set extended window style
    g_szClassName, // use registeret windows class
    "The title of my window", // Text on window's title bar
    WS OVERLAPPEDWINDOW, //set window style
    CW_USEDEFAULT, CW_USEDEFAULT, 240, 120, // location and window size
    NULL, NULL, hInstance, NULL);
    // parent handle, child ID, handle to instance, additional window data
// Check whether window has been created:
if(hwnd == NULL)
{
    MessageBox(NULL, "Window Creation Failed!", "Error!",
         MB ICONEXCLAMATION | MB OK);
    return 0:
}
```

ShowWindow(hwnd, nCmdShow); //paint window border, tiltle, menu... UpdateWindow(hwnd); // paint window working area

The Message Loop

// Step 3: The Message Loop

```
while(GetMessage(&Msg, NULL, 0, 0) > 0) // get message from application's queue
{
    TranslateMessage(&Msg); //translate nkeyboard messages
    DispatchMessage(&Msg); // send the message to the appropriate window
return Msg.wParam;
                               Create Window
   Other Windows
                                GetMessage
      programs
                                               Message Loop
                                    Return 0?
                                                             DispatchMessage
                                                   No
                                  Yes
                                                            WndProc processes
                                Exit WinMain
                                                                 messages
```

The Window Procedure

```
// Step 4: the Window Procedure
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM
    (IParam)
{ HDC HDCPaint; PAINTSTRUCT PaintStruct; RECT rect;
switch(msg)
{
    case WM CLOSE:
         DestroyWindow(hwnd);
                                   break:
    case WM DESTROY:
         PostQuitMessage(0);
                                    break;
    case WM PAINT:
         HDCPaint=BeginPaint(hwnd,&PaintStruct);
         GetClientRect(hwnd,&rect);
         DrawText(HDCPaint,"Hello World!",-1,&rect,
                           DT SINGLELINEIDT CENTERIDT VCENTER);
         EndPaint(hwnd, &PaintStruct); break;
    default:
         return DefWindowProc(hwnd, msg, wParam, IParam);
    }
```

return 0;

The Window Procedure - typical commands

```
LRESULT CALLBACK WndProc(WND hwnd, UINT iMsg,
                     WPARAM wParam, LPARAM IParam)
    switch(iMsg)
{
         case WM CREATE:
                  // first message ever serviced
         case WM SIZE:
                  // the size of window has been changed
         case WM COMMAND:
                  // a menu item has been chosen
         case WM CLOSE:
                  // SB, wants to close the window
         case WM DESTROY:
                  // clean before destroying application
         case WM PAINT:
                  // the window needs repainting
         case WM CHAR:
                  // the letter has been send to application
         case WM KEYDOWN: // A key has been pressed
         default:
                  return DefWindowProc(hwnd,iMsg,wParam,IParam);
```

Adding Menu (1)

- Project -> Add New Item... -> Resource File
- A created resource file will appear in the project files tree
- Open resource tree in the project
- Right click on it
- Choose -> Add resource
- Choose -> Menu

Resource Tree Minu Image: Section of the unique ID: Nenu has the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of the unique ID: Image: Section of th	 A menu 	ditor will appear:
Resource Tree Wenu has the unique ID: Nore False Nore False No		🗱 win1 - Microsoft Visual Studio
Resource Tree wini_res.rc (IDR_MENU) wini_res.rc (IDR_MENU) wini_res.rc (IDR_MENU) Nenu IDFILE_EXIT None File Sile Sile Sile Sile None Height Justify File Nile Nile Nile None Height Justify File Nile Nile Nile Menu has the unique ID: IDR_MENUI Name His Name IDR_MENUI IDR_MENUI Name His Name		<u>File Edit View Project Build Debug Tools Window Community Help</u>
Resource Tree ID_FILE_EXIT Ss. I None Risc None Risc Nisc None ID_FILE_EXIT Ss. I Ss. I ID_FILE_EXIT Show output from: ID_FILE_EXIT ID_FILE_EXIT		: [1] ▼ [2] ▼ [2] ▼ [2] ↓ [2] [2] ↓ [2] [2] [2] [2] [2] [2] [2] [2] [2] [2]
Resource Tree Menu Editor Menued ID_FILE_EXIT None Popup False None Break None Right Justify False False None Break None Show output Format ID_FILE_EXIT Menu has the unique ID: ID_FILE_EXIT ID_FILE_EXIT ID_BR_MENU1 ID_FILE_EXIT ID_FILE_EXIT		Resource View - win1 - + × win1_res.rc (IDR_MENU1 - Menu)* = × Properties
Resource Tree Ext Type Here False Behavior Break None Risc Misc Nisc Number Here Here Show output Show output from: None Show output Show output from: None Menu has the unique ID: IDR_MENU1 Name		Image: Win1 File Type Here Image: Win1 File Type Here
Resource Tree Image: Climatic prime result State State Image: Climatic prime Image: Climatic prime Im		Exit Type Here □ 2↓ □
Resource Tree Proput False Right Justify False Right Order False None Right Order False Output Prompt Show output from: Prompt Menu has the unique ID: IDR_MENU1		IDR_MENU1 Type Here Grayed False €
Resource Tree Image: Substrain the state unique ID: IDR_MENU1 Break None Right Justify False Right Order False ID ID ID ID ID ID ID <td></td> <td></td>		
Resource Tree Right Justify Right Justify False Right Justify False Nenu Nenu Menu False Nenu Menu False Nenu False Nenu False Nenu False IDR_MENU1 IDR IDR IDR		Break None
Image: Second control of the second	Desource Tr	Right Justify False
Image: Second state		
Image: Subscript of the second se		(Name) Menu Eduor
ID ID_ILE_EAT Prompt Show ou put from: ID I		Help False
Output Image: Separator False Show ou put from: Image: Separator False Menu has the unique ID: Image: Separator False IDR_MENU1 Image: Separator False		
Show ou put from: Image: The state of th		Output Version Separator False
Menu has the unique ID: IDR_MENU1	-	Show ou put from:
IDR_MENU1		Menu has the unique ID:

Menu Editor

Adding Menu (2)

• Modify Window Class:

return 0;}

```
#include "resource.h"
// ...
const char g szClassName[] = "myWindowClass";
WNDCLASSEX wc:
wc.cbSize
                            = sizeof(WNDCLASSEX); //size of the structure
                  = CS_HREDRAW | CS_VREDRAW; //class styles
wc.style
wc.lpfnWndProc
                  = WndProc; //Pointer to the window procedure for this window class
wc.cbClsExtra
                  = 0; // extra data allocated for class
wc.cbWndExtra
                  = 0; // extra data allocated for window
wc.hlnstance
                  = hInstance; // handle to application instance
                  = LoadIcon(NULL, IDI APPLICATION); //The big icon
wc.hlcon
                  = LoadCursor(NULL, IDC ARROW); // Coursor
wc.hCursor
wc.hbrBackground = (HBRUSH)(COLOR WINDOW+1); // Background Colour
wc.lpszMenuName = MAKEINTRESOURCE(IDR MENU1); // Name of a menu resource
wc.lpszClassName = g szClassName; // Name to identify the class with
                  = LoadIcon(NULL, IDI_APPLICATION); // The small icon
wc.hlconSm
// The class defined must be registered in the system:
if(!RegisterClassEx(&wc))
{
    MessageBox(NULL, "Window Registration Failed!", "Error!",
         MB ICONEXCLAMATION | MB OK);
```

Adding Menu (3)

• Modify the Window Procedure:

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM IParam)

{ HDC HDCPaint; PAINTSTRUCT PaintStruct; RECT rect; switch(msg)

```
case WM CLOSE:
         //...
case WM DESTROY:
         //...
case WM PAINT:
         //...
case WM COMMAND:
         switch(LOWORD(wParam))
         case ID FILE EXIT:
                  DestroyWindow(hwnd); break;
         break;
default:
         return DefWindowProc(hwnd, msg, wParam, IParam);
return 0;
```

ł

Adding Dialog (1)

Dialog Editor

- Open resource tree in the project
- Right click on it
- Choose -> Add resource
- Choose -> Dialog
- A dialog editor will appear:



Adding Dialog (2)

- A new menu item may be added: File -> Dialog
- The Window Procedure sholuld be updated:

BOOL CALLBACK AboutDlgProc(HWND, UINT, WPARAM, LPARAM);

```
LRESULT CALLBACK WndProc
                  (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM IParam)
{//...
static HINSTANCE hInstance; // The handle to Instance should be stored
switch(iMsg)
         case WM CREATE:
{
                  hInstance = ((LPCREATESTRUCT) IParam) -> hInstance;
                  return 0:
        case WM_COMMAND:
                  switch(LOWORD(wParam))
                  {case ID PLIK KONIEC:DestroyWindow(hwnd); return 0;
                  case ID PLIK DIALOG: // The Dialog Proccedure should be called
                           DialogBox(hInstance,
                                    MAKEINTRESOURCE(IDD DIALOG1),
                                    hwnd, AboutDlgProc); return 0;
                  }return 0;
```

Adding Dialog (3)

• The Dialog Procedure sholuld be created:

{

```
BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message,
WPARAM wParam, LPARAM IParam)
```

```
switch(message)
case WM INITDIALOG: // Assures that Dialog appears
         return TRUE;
case WM COMMAND: // Control buttons service
         switch(LOWORD (wParam))
         case IDOK:
         case IDCANCEL:
                  EndDialog(hDlg,0);
                  return TRUE;
         } break;
return FALSE;
```