

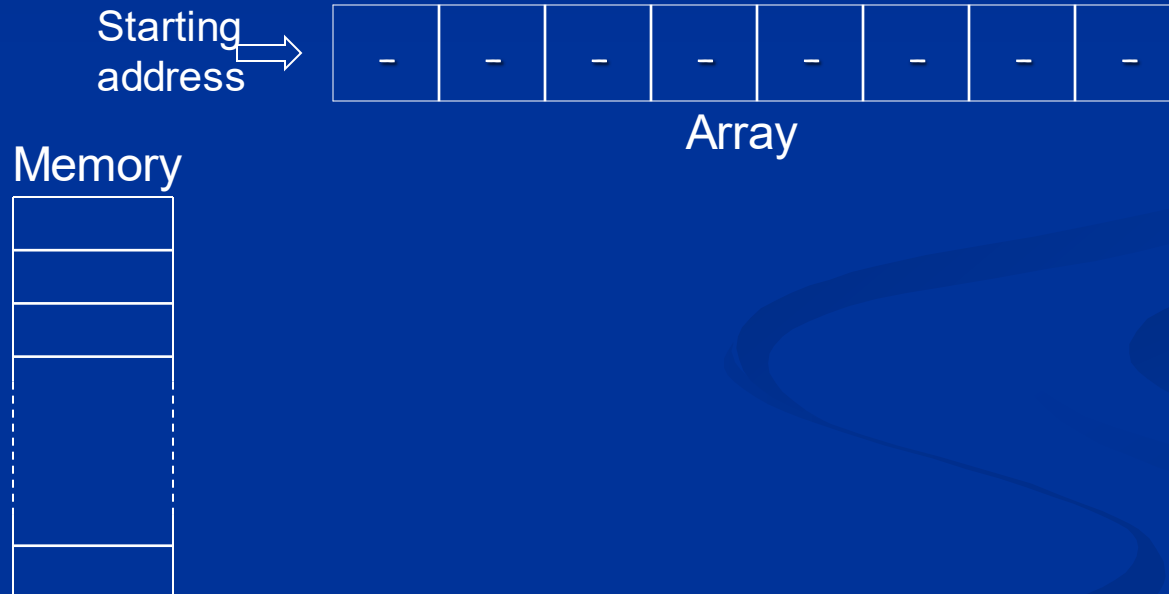
# L e c t u r e #



17

# Review of Today's Lecture

```
char *strings[] = {"One", "Two", "Three"};
```



```
printf("number=%d", 527);
```

|

Format string

Output

```
number=527
```

```
printf( [__address__] , 527) ;
```

|

"number=%d"

Format string

## Prototype of printf() in stdio.h

```
int printf( const char *format [, argument]... );
```

`char ch[] = "number=%d";` *format string stored  
in a character array*

`printf(ch, 527);`

|

Address of the format string passed to `printf`

## Map of an EXE file

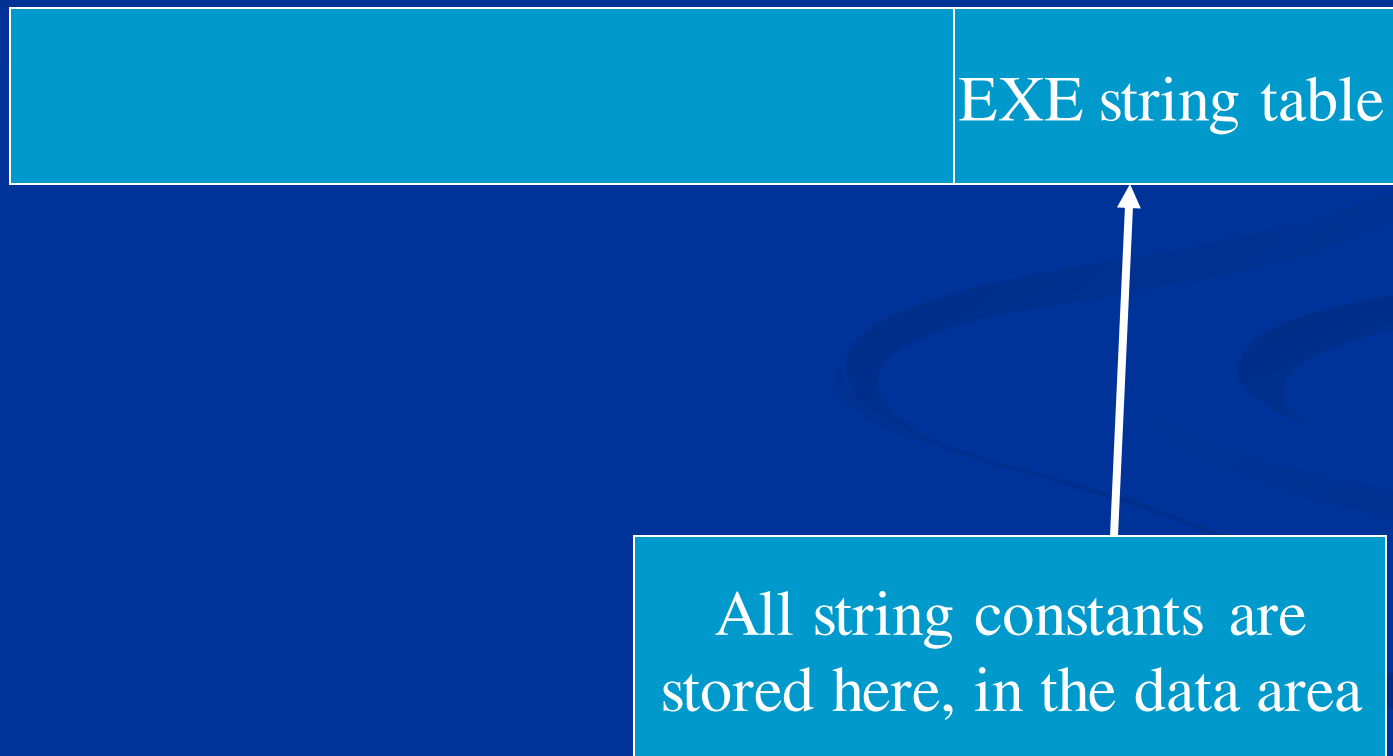
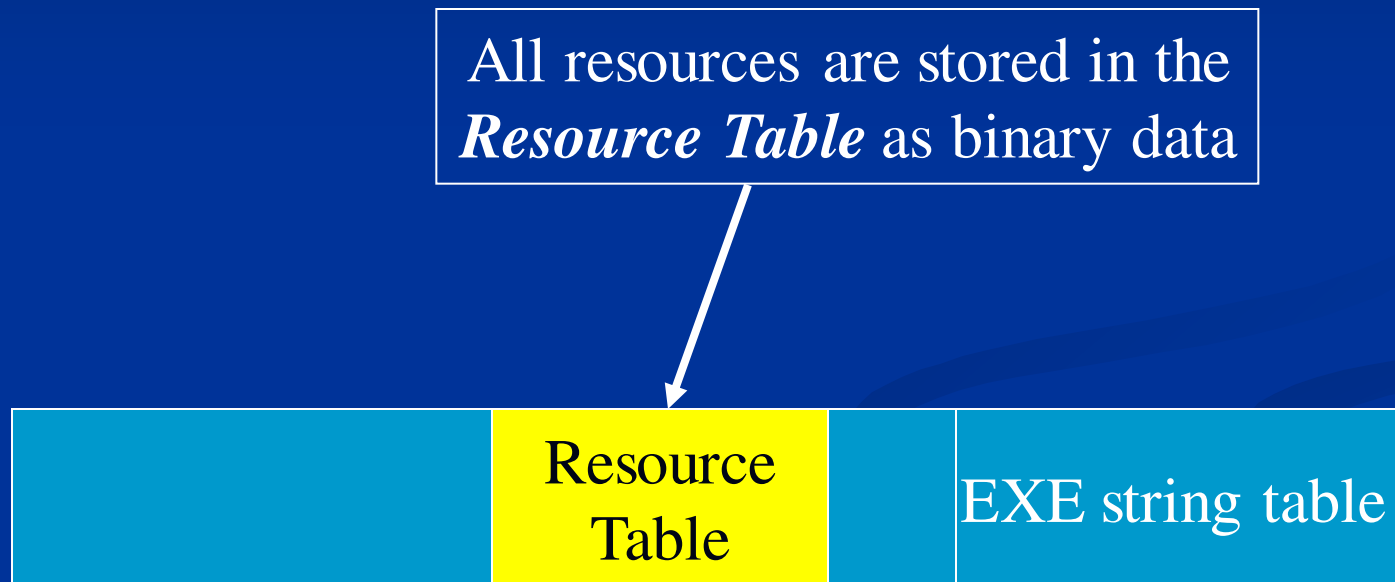
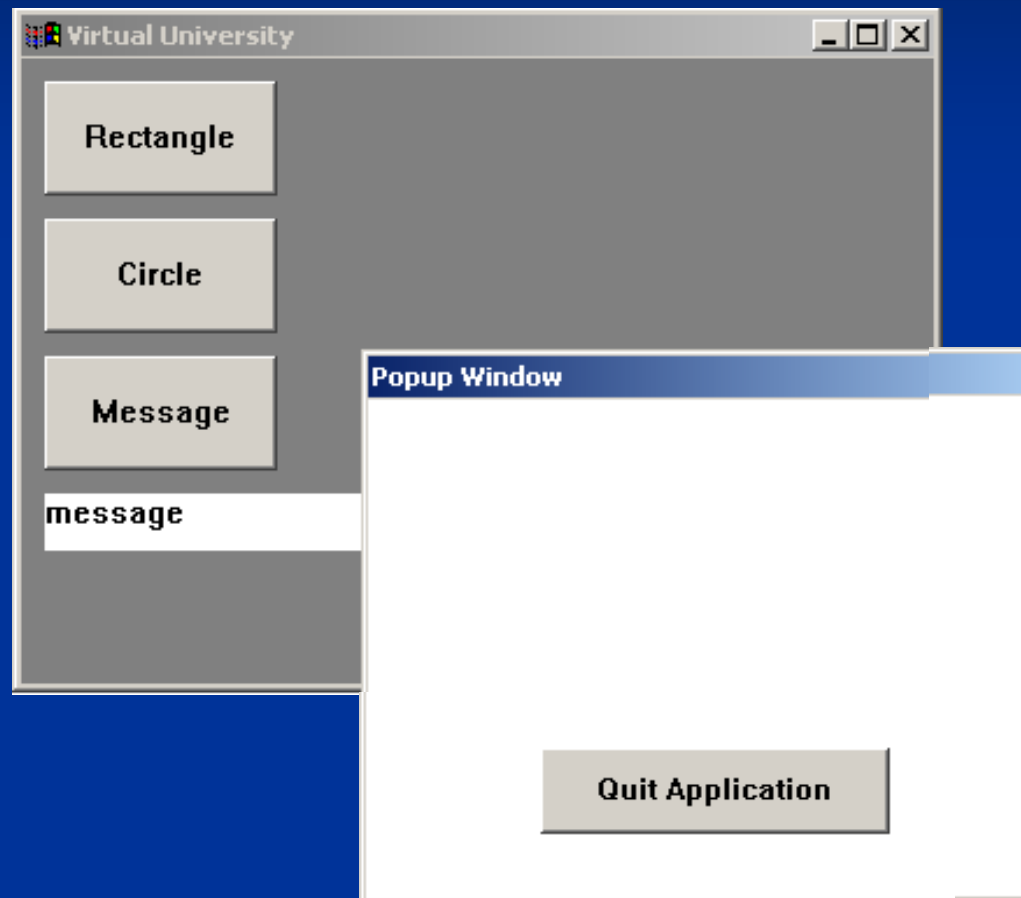


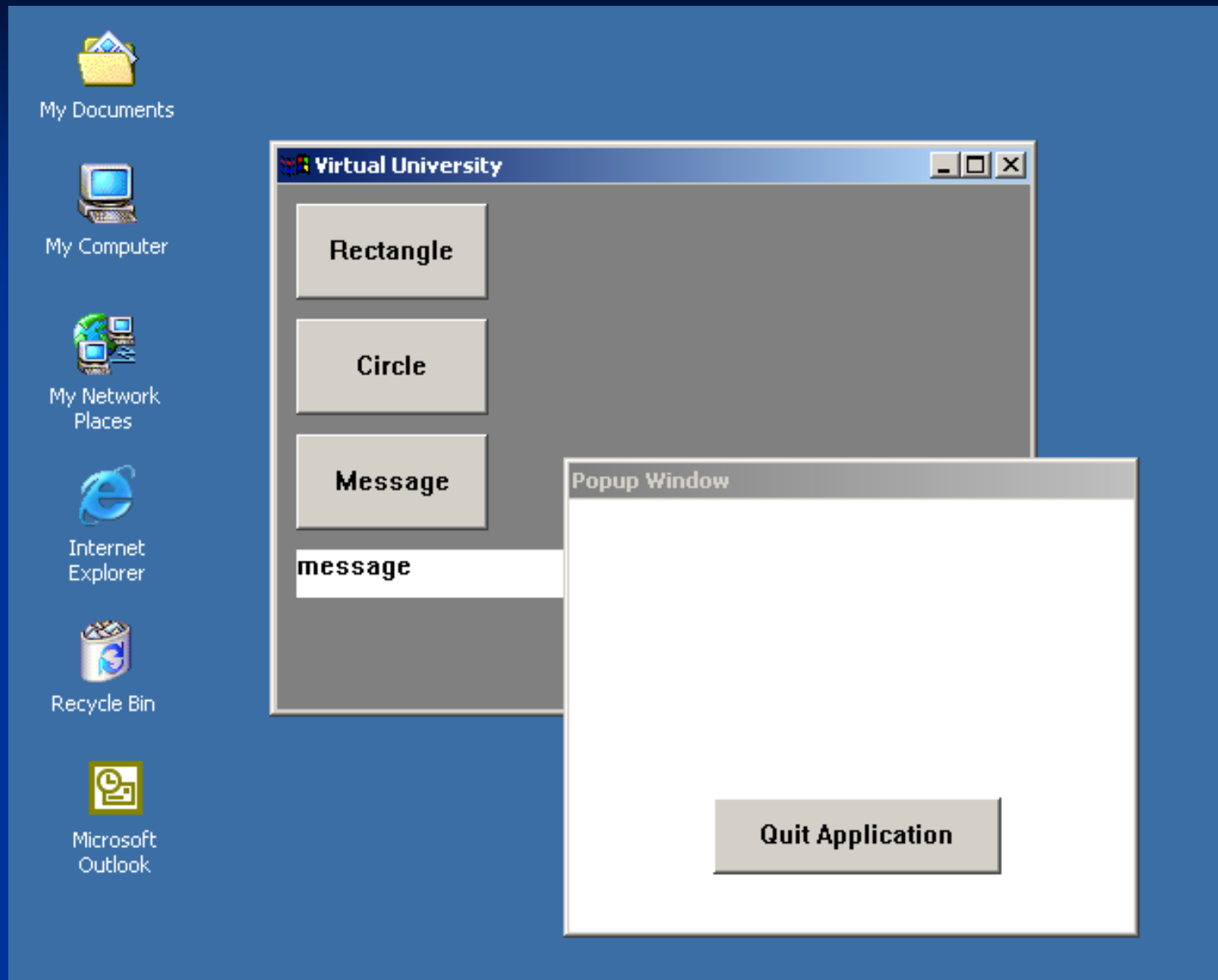


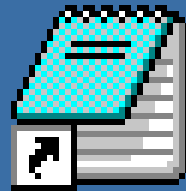
Figure of a map of EXE with all string literals stored in the data area as EXE string table



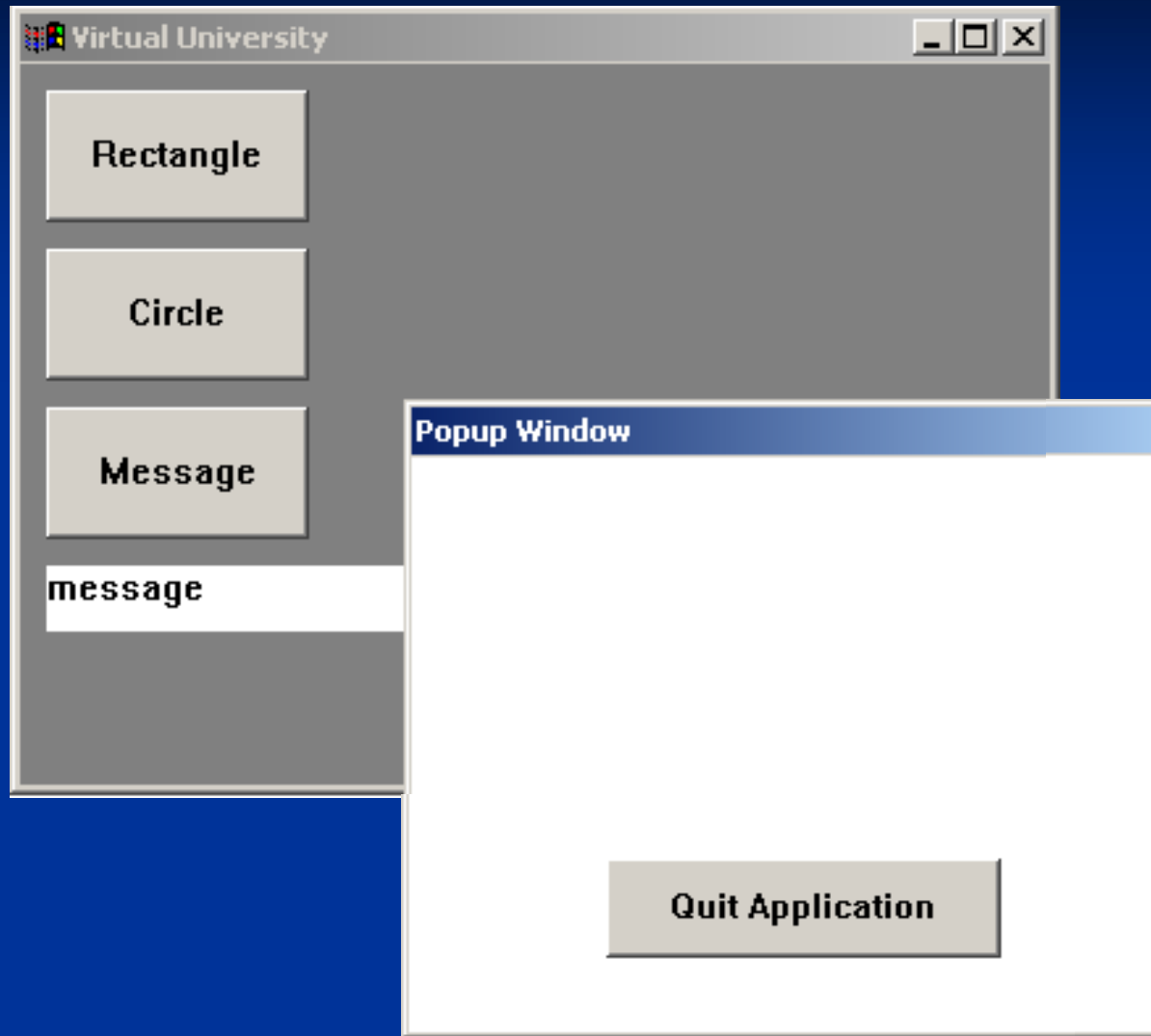
Resources are defined in a separate file and linked later







Notepad



# Accelerator

# Types of Windows Resources

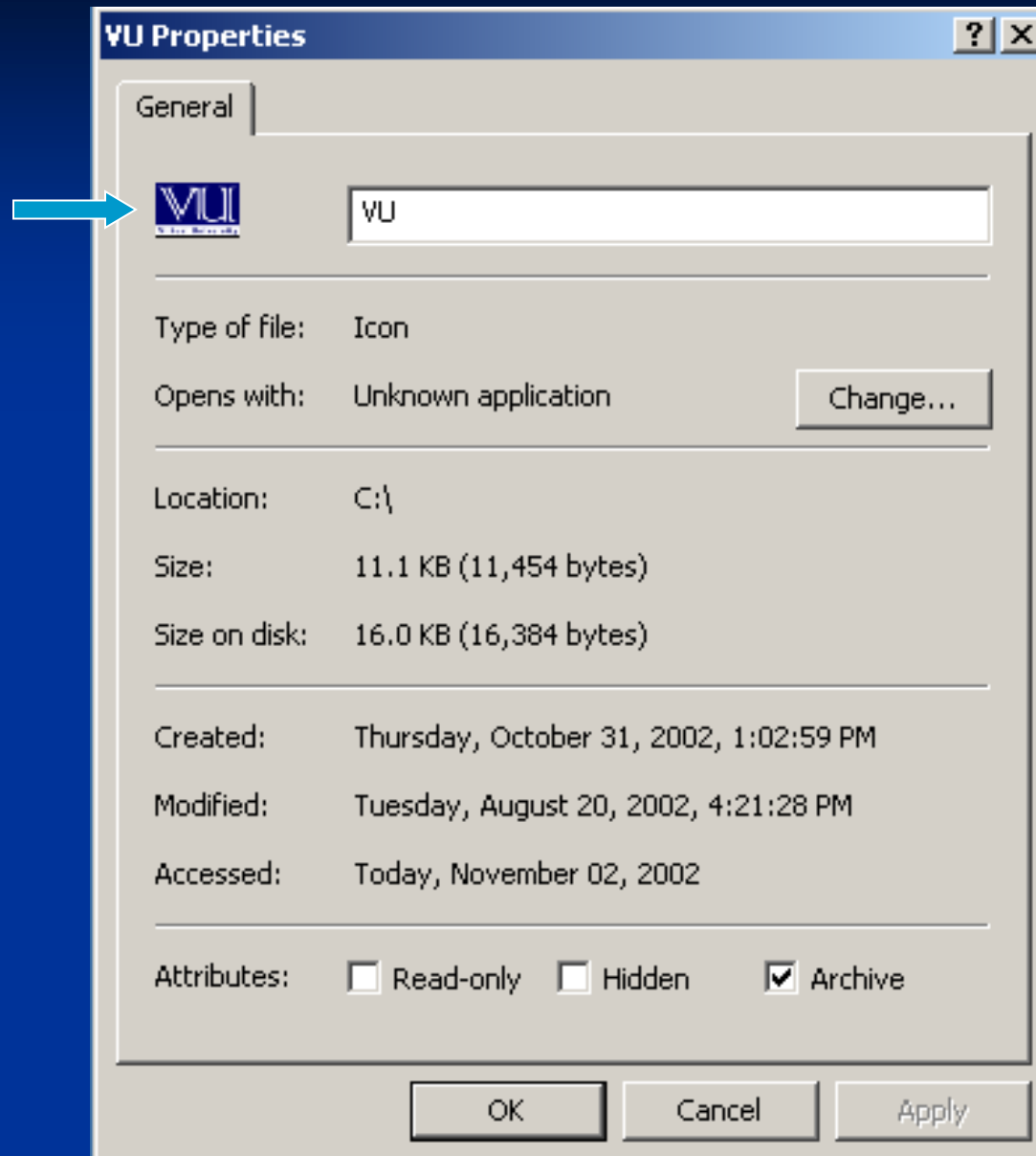
- Accelerator
- String Table
- Icon
- Bitmap
- Dialog
- Menu
- Cursor
- Version

# Usual Filename Extensions

.ico                  Icon files

.bmp                  Bitmap image files





.rc resource file (text file containing many resource statement)

|

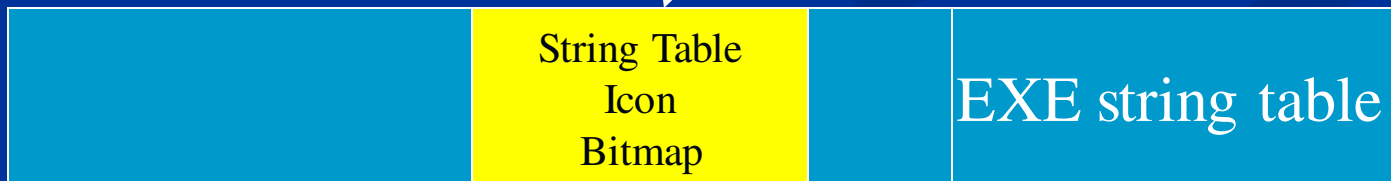
Compile to .res file (using Resource Compiler)

|

Link with other files to make final EXE (using linker)

Externally labelled as Resources : String Table, Icon, Bitmap, Dialog etc. etc.

All resources are stored in the  
*Resource Table* as binary data



## ICON resource statement in a resource file (.rc)

```
#define IDI_ICON 101
```

IDI_ICON	ICON	DISCARDABLE	"vu.ico"
Integer id	reserved word		icon filename
101	ICON	DISCARDABLE	"vu.ico"

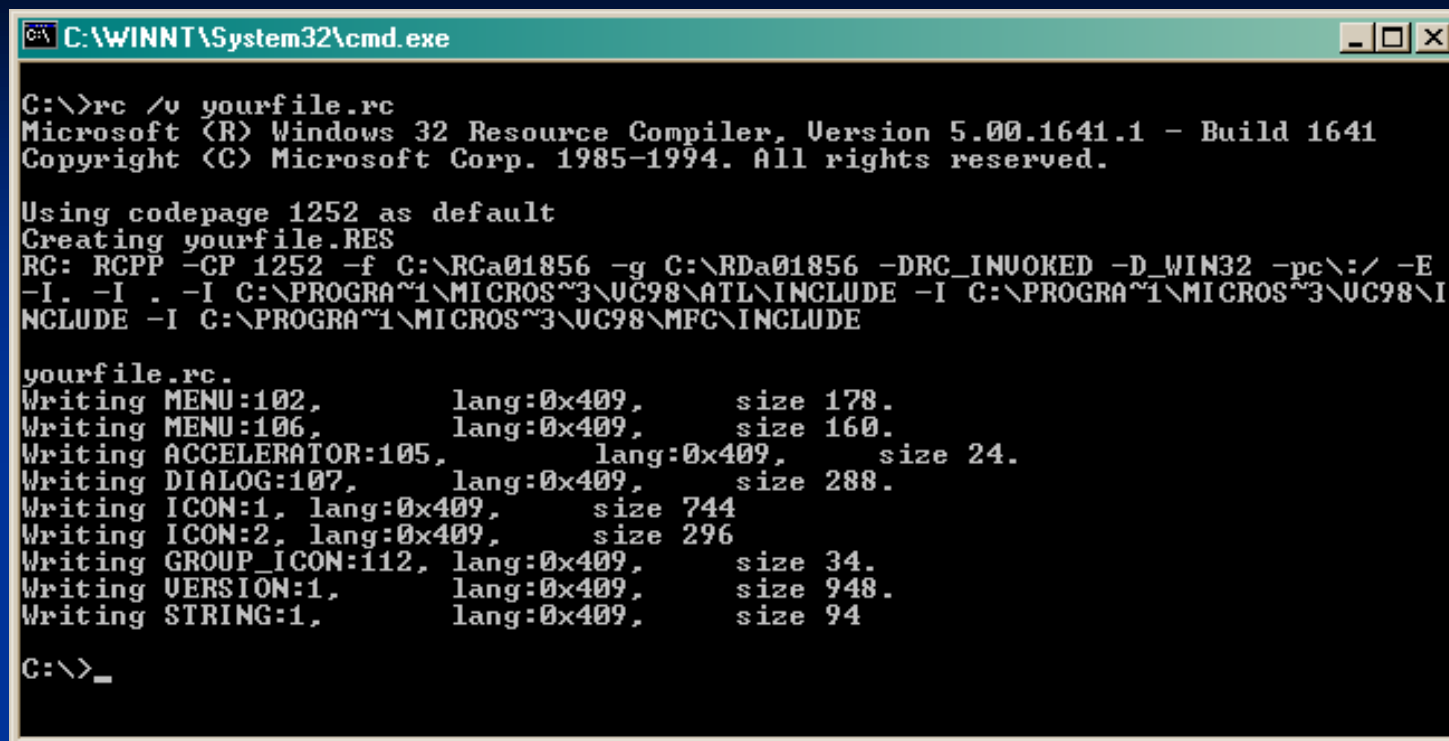
resource.h

```
#define IDI_ICON    101
```

yourfile.rc

```
#include "resource.h"
```

```
IDI_ICON    ICON    DISCARDABLE    "vu.ico"
```



```

C:\WINNT\System32\cmd.exe

C:\>rc /v yourfile.rc
Microsoft (R) Windows 32 Resource Compiler, Version 5.00.1641.1 - Build 1641
Copyright (C) Microsoft Corp. 1985-1994. All rights reserved.

Using codepage 1252 as default
Creating yourfile.RES
RC: RCPP -CP 1252 -f C:\RCa01856 -g C:\RDa01856 -DRC_INVOKED -D_WIN32 -pc\:/ -E
-I. -I . -I C:\PROGRA~1\MICROS~3\UC98\ATL\INCLUDE -I C:\PROGRA~1\MICROS~3\UC98\I
NCLUDE -I C:\PROGRA~1\MICROS~3\UC98\MFC\INCLUDE

yourfile.rc.
Writing MENU:102,      lang:0x409,      size 178.
Writing MENU:106,      lang:0x409,      size 160.
Writing ACCELERATOR:105,      lang:0x409,      size 24.
Writing DIALOG:107,      lang:0x409,      size 288.
Writing ICON:1, lang:0x409,      size 744
Writing ICON:2, lang:0x409,      size 296
Writing GROUP_ICON:112, lang:0x409,      size 34.
Writing VERSION:1,      lang:0x409,      size 948.
Writing STRING:1,      lang:0x409,      size 94

C:\>_
  
```

C:\>rc.exe yourfile.rc

It will generate **yourfile.res**

OR

give error messages if there are errors.

Icon read from Vu.ico

|

| (resource compiler)

|

Yourfile.res

other files

|

|

|-----|

| (linker)

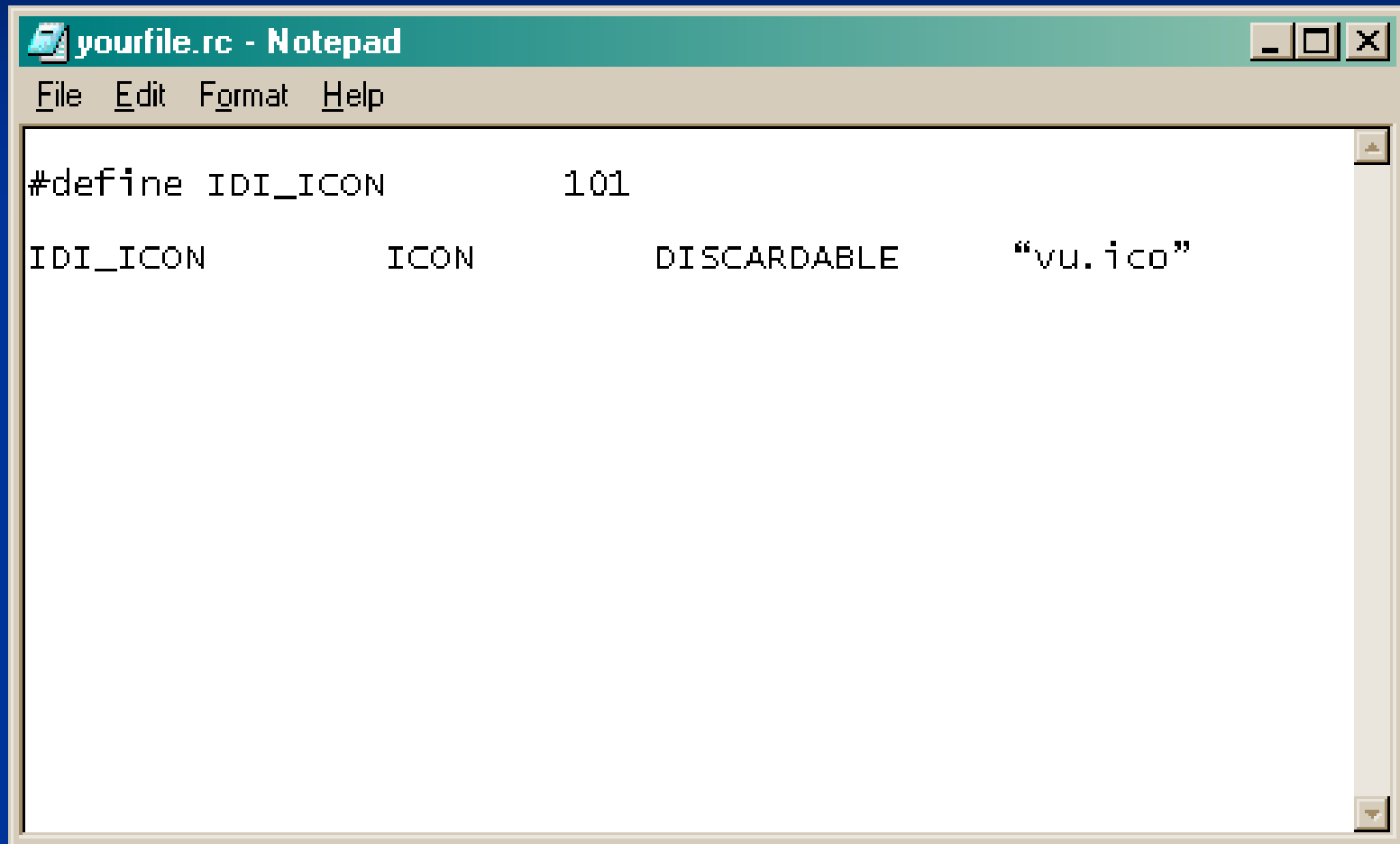
|

icon from vu.ico part of Resource Table

ABC.EXE (final executable)



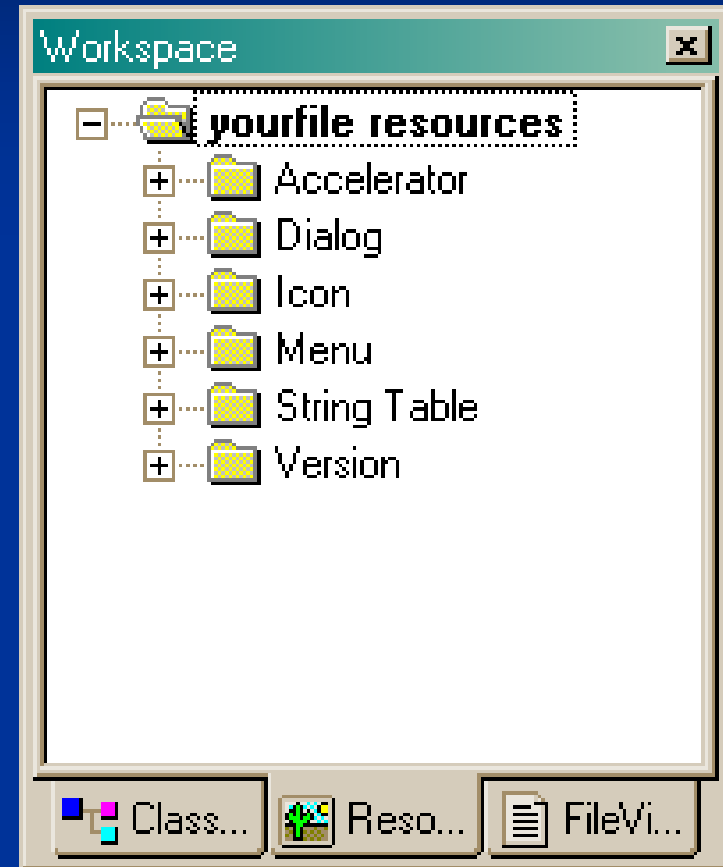
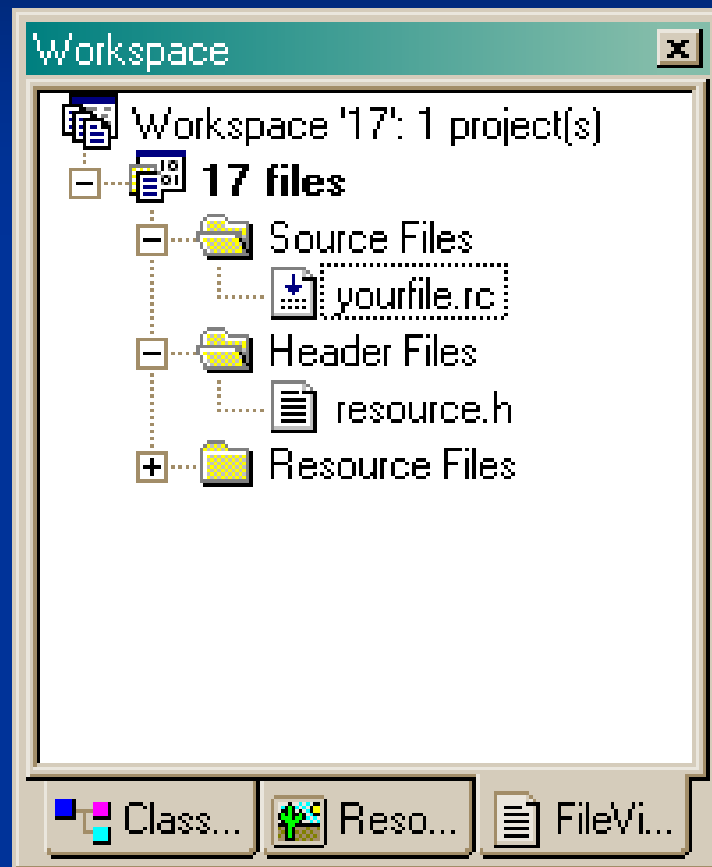
`yourfile.rc` typed in Notepad



```
#define IDI_ICON 101
IDI_ICON ICON DISCARDABLE "vu.ico"
```



# yourfile.rc added to Visual Studio project



# Loading an Icon from the resource table

```
HICON LoadIcon(  
    HINSTANCE hInstance, // handle to application instance  
    LPCTSTR lplconName  // resource identifier or name  
    string  
);
```

# Specifying a class icon

```
wc.cbClsExtra    = 0;  
wc.cbWndExtra    = 0;  
wc.hInstance     = hInstance;  
wc.hIcon         = LoadIcon(hInstance,  
MAKEINTRESOURCE(IDI_ICON));  
wc.hCursor       = LoadCursor(NULL,  
IDC_UPARROW);
```

```
#define IDI_ICON 101
```

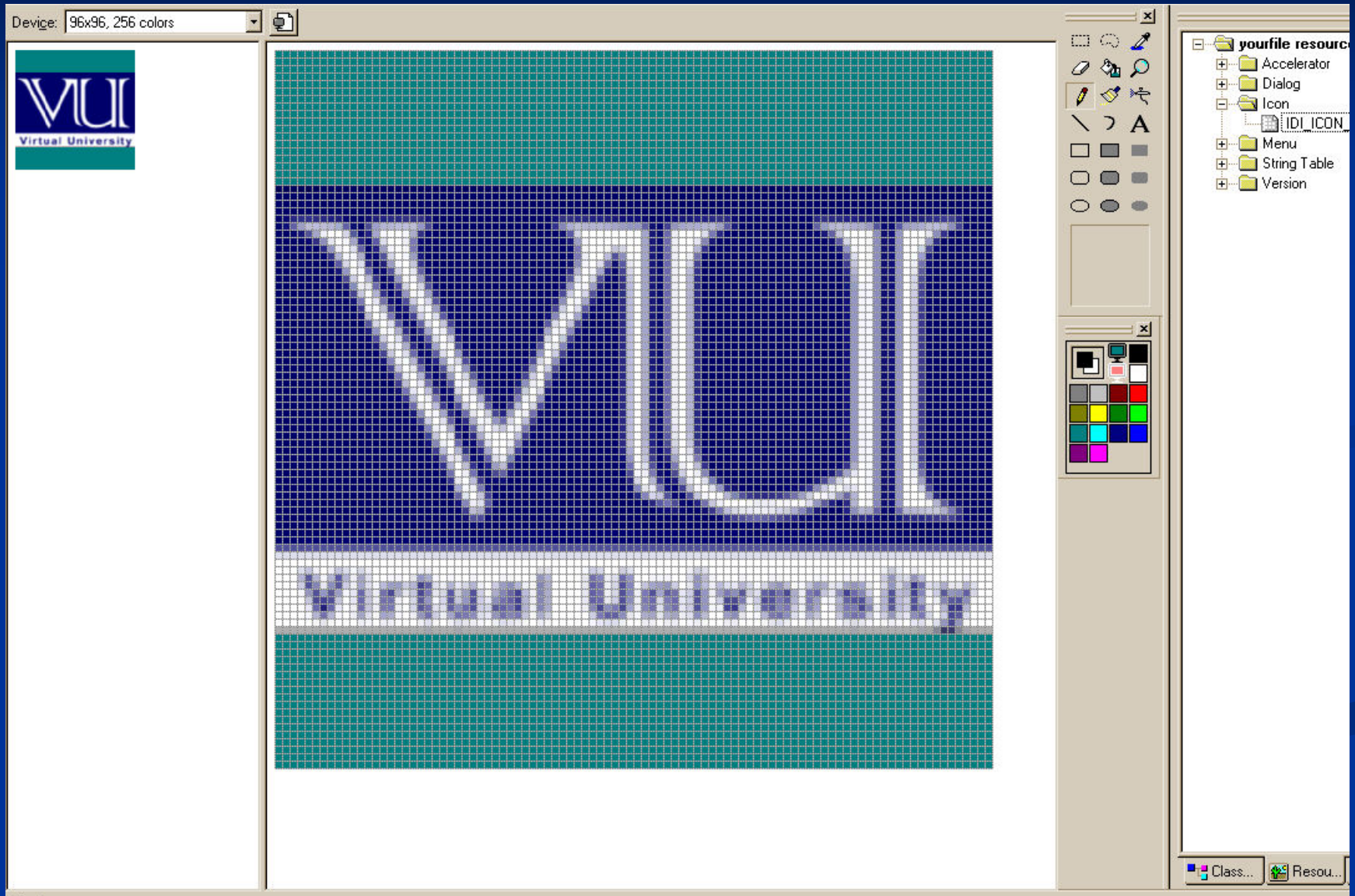
```
IDI_ICON    ICON          DISCARDABLE    "vu.ico"
```

Animation: change IDI\_ICON to some string like "myIcon"

```
HICON LoadIcon(hInstance, IDI_ICON);
```

```
HICON LoadIcon(hInstance, "myIcon");
```

# Icon editor of Visual Studio 6.0



# LPTSTR MAKEINTRESOURCE( WORD Integer);

If a resource id is given

```
HICON LoadIcon(hInstance, "myIcon");
```

If a resource name is given as a string

```
HICON LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON));
```

# String Table in a resource file

```
#include "resource.h"
```

```
STRINGTABLE DISCARDABLE  
BEGIN
```

```
    IDS_STRING1        "This is Virtual University"
```

```
    IDS_STRING2        "MyWindowClass"
```

```
    IDS_STRING3        "My Novel Programme"
```

```
END
```



# Loading a string from the string table

```
int LoadString(  
    HINSTANCE hInstance, // handle to resource module  
    UINT uID,           // resource identifier  
    LPTSTR lpBuffer,    // resource buffer  
    int nBufferMax      // size of buffer  
);
```



```
char msg[80];
```

```
LoadString(hInstance, IDS_STRING1, msg, 80);
```

```
MessageBox(NULL, msg, "Message", MB_OK);
```



# Keyboard Accelerators

# Defining an accelerator

```
#define ID_DO_BACK      1001
#define ID_ACC2         1002
#define ID_DRAWSTRING  1003
```

ACCELERATOR	ACCELERATORS	DISCARDABLE
BEGIN		
VK_BACK,	ID_DO_BACK,	VIRTKEY, ALT, NOINVERT
VK_DELETE,	ID_ACC2,	VIRTKEY, ALT, NOINVERT
.....		
"^S",	ID_DRAWSTRING,	ASCII, NOINVERT
END		

# Loading Accelerator Resource

```
HACCEL LoadAccelerators(  
    HINSTANCE hInstance, // handle to module  
    LPCTSTR lpTableName // accelerator table name  
);
```

```
int TranslateAccelerator(  
    HWND hWnd,      // handle to destination window  
    HACCEL hAccTable, // handle to accelerator table  
    LPMSG lpMsg      // message information  
);
```

# TranslateAccelerator() at work

VK\_BACK, ID\_DO\_BACK, VIRTKEY, ALT, NOINVERT

When ALT+BACKSPACE is pressed,

|  
TranslateAccelerator()

|  
| sends a

|  
WM\_COMMAND message

with wParam= low-word: ID\_DO\_BACK

HACCEL hAccel;

### *Load the accelerator table*

```
hAccel = LoadAccelerator(hInstance,
MAKEINTRESOURCE(ACCELERATOR))
```

```
While(GetMessage(&msg, . . . . . ))
{
```

### *Call translateAccelerator to test if accelerator is pressed*

```
    If(!TranslateAccelerator(msg.hwnd, hAccel, &msg))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
```

```
}
```

### *Windows Procedure*

```
Case WM_COMMAND:
```

```
    if(LOWORD(wParam) == ID_DO_BACK)
    {
        // accelerator is pressed
    }
```

# To Do ☹

Implement keyboard accelerators in the Lecture 15 application:

CTRL+S	String drawing
ALT+C	Circle drawing
“R”	Rectangle drawing



# Descriptive #define Constant Names

IDS\_ERROR\_MESSAGE

IDS\_ERROR\_MESSAGE\_FILE\_NOT\_FOUND