```
/*
    MODULE: winMaker.cpp

    PURPOSE: Manages the creation and control of generic windows.

    AUTHORS: Doug Penner
             Kyle Macdonald
             Steffen L. Norgren
             Max Wardell
             Eddie Zhang
 */

#include "s_winMaker.h"

// String Resource
ResString::ResString (HINSTANCE hInst, int resId) {
    if (!::LoadString(hInst, resId, _buf, MAX_RESSTRING + 1)) {
        ::MessageBox(NULL, TEXT("Load String failed"), TEXT(""), MB_OK);
    }
}

WinSimpleClass::WinSimpleClass(int resId, HINSTANCE hInst)
    : _hInstance(hInst) {

    ResString resStr(hInst, resId);
    _name = resStr;
}

WinClass::WinClass(char const * className, HINSTANCE hInst, WNDPROC wndProc)
    : WinSimpleClass(className, hInst) {

    _class.lpfnWndProc = wndProc;
    SetDefaults();
}

WinClass::WinClass(int resId, HINSTANCE hInst, WNDPROC wndProc)
    : WinSimpleClass(resId, hInst) {

    _class.lpfnWndProc = wndProc;
    SetDefaults();
}

void WinClass::SetDefaults () {
    // Provide reasonable default values
    _class.cbSize          = sizeof(WNDCLASSEX);
    _class.style           = 0;
    _class.lpszClassName   = GetName();
    _class.hInstance       = GetInstance();
    _class.hIcon           = 0;
```

```
    _class.hIconSm          = 0;
    _class.lpszMenuName     = 0;
    _class.cbClsExtra       = 0;
    _class.cbWndExtra       = 0;
    _class.hbrBackground    = reinterpret_cast<HBRUSH>(COLOR_WINDOW + 1);
    _class.hCursor          = ::LoadCursor(0, IDC_ARROW);
}

HWND WinSimpleClass::GetRunningWindow() {
    HWND hWnd = ::FindWindow(GetName(), 0);

    if (::IsWindow(hWnd)) {
        HWND hWndPopup = ::GetLastActivePopup(hWnd);
        if (::IsWindow(hWndPopup)) {
            hWnd = hWndPopup;
        }
    }
    else {
        hWnd = 0;
    }

    return hWnd;
}

void WinClass::Register() {
    if (::RegisterClassEx(&_class) == 0) {
        ::MessageBox(NULL, TEXT("Internal error: RegisterClassEx failed."), TEXT(""),
MB_OK);
    }
}

// Makes top window class with icons and menu
TopWinClass::TopWinClass(int resId, HINSTANCE hInst, WNDPROC wndProc)
    : WinClass(resId, hInst, wndProc) {

    _class.lpszMenuName = MAKEINTRESOURCE(resId);
}

// The maker of a window of a given class
WinMaker::WinMaker(WinClass & winClass)
  : _hWnd(0),
    _class(winClass),
    _exStyle(0),            // extended window style
    _windowName(0),         // pointer to window name
    _style(WS_OVERLAPPED),  // window style
    _x(CW_USEDEFAULT),      // horizontal position of window
    _y(0),                  // vertical position of window
    _width(X_SIZE),         // window width
    _height(Y_SIZE),            // window height
```

```
    _hWndParent(0),          // handle to parent or owner window
    _hMenu(0),               // handle to menu, or child-window identifier
    _data(0)                 // pointer to window-creation data
{ }

void WinMaker::Create() {
    _hWnd = ::CreateWindowEx(
            _exStyle,
            _class.GetName(),
            _windowName,
            _style,
            _x,
            _y,
            _width,
            _height,
            _hWndParent,
            _hMenu,
            _class.GetInstance (),
            _data);

    if (_hWnd == 0) {
        ::MessageBox(NULL, TEXT("Internal error: Window Creation Failed."), TEXT(""),
MB_OK);
    }
}

void WinMaker::Show(int nCmdShow) {
    ::ShowWindow(_hWnd, nCmdShow);
    ::UpdateWindow(_hWnd);
}

// Makes top overlapped window with caption
TopWinMaker::TopWinMaker(WinClass & winClass, char const * caption)
    : WinMaker(winClass) {

    _style = WS_OVERLAPPEDWINDOW | WS_VISIBLE;
    _windowName = caption;
}
```