```
/*
    MODULE: winMaker.h

    PURPOSE: Window specific classes and templates

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

#ifndef _WINMAKER_H_
#define _WINMAKER_H_

#define X_SIZE      640
#define Y_SIZE      400
#define X_MIN_SIZE  429
#define Y_MIN_SIZE  346

#include <windows.h>
#include <string>

// Allows us to retrieve predefined strings from the resource file
class ResString {
    enum { MAX_RESSTRING = 255 };
public:
    ResString(HINSTANCE hInst, int resId);
    operator char const *() const { return _buf; }
private:
    char _buf[MAX_RESSTRING + 1];
};

// Getting and Setting WindowLong: default is GWL_USERDATA
template <class T>
inline T WinGetLong(HWND hWnd, int which = GWL_USERDATA) {
    return reinterpret_cast<T>(::GetWindowLong(hWnd, which));
}

template <class T>
inline void WinSetLong (HWND hWnd, T value, int which = GWL_USERDATA) {
    ::SetWindowLong(hWnd, which, reinterpret_cast<long>(value));
}

// Use for built-in classes
class WinSimpleClass {
public:
    WinSimpleClass (char const * name, HINSTANCE hInst) : _name (name), _hInstance
(hInst) {}
```

```
    WinSimpleClass(int resId, HINSTANCE hInst);
    char const * GetName() const { return _name.c_str(); }
    HINSTANCE GetInstance() const { return _hInstance; }
    HWND GetRunningWindow();
protected:
    HINSTANCE   _hInstance;
    std::string _name;
};

class WinClass: public WinSimpleClass {
public:
    WinClass(char const * className, HINSTANCE hInst, WNDPROC wndProc);
    WinClass(int resId, HINSTANCE hInst, WNDPROC wndProc);
    void SetBgSysColor(int sysColor) {
        _class.hbrBackground = reinterpret_cast<HBRUSH>(sysColor + 1);
    }

    void Register();
protected:
    void SetDefaults();
    WNDCLASSEX  _class;
};

class TopWinClass: public WinClass {
public:
    TopWinClass(int resId, HINSTANCE hInst, WNDPROC wndProc);
};

class WinMaker {
public:
    WinMaker(WinClass & winClass);
    operator HWND() { return _hWnd; }
    void AddCaption(char const * caption) {
        _windowName = caption;
    }
    void AddSysMenu()       { _style |= WS_SYSMENU; }
    void AddVScrollBar()    { _style |= WS_VSCROLL; }
    void AddHScrollBar()    { _style |= WS_HSCROLL; }
    void Create();
    void Show(int nCmdShow = SW_SHOWNORMAL);
protected:
    WinClass &   _class;
    HWND         _hWnd;

    DWORD        _exStyle;       // extended window style
    char const * _windowName;    // pointer to window name
    DWORD        _style;         // window style
    int          _x;             // horizontal position of window
    int          _y;             // vertical position of window
```

```
    int           _width;         // window width
    int           _height;        // window height
    HWND          _hWndParent;    // handle to parent or owner window
    HMENU         _hMenu;         // handle to menu, or child-window identifier
    void        * _data;          // pointer to window-creation data
};

class TopWinMaker: public WinMaker {
public:
    TopWinMaker (WinClass & winClass, char const * caption);
};

#endif
```