#**pragma once**

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include "D3dx9math.h"

#include "d3d9.h"

#include <string>

#**pragma comment(lib, "d3d9.lib")**

#**pragma comment(lib, "D3dx9.lib")**

#define CUSTOMFVF (D3DFVF\_XYZRHW | D3DFVF\_DIFFUSE)

**struct** CUSTOMVERTEX //Flexible Vertex Format

{

FLOAT x, y, z, rhw; // from the D3DFVF\_XYZRHW flag

DWORD color; // from the D3DFVF\_DIFFUSE flag

};

**class** Direct3D

{

**public**:

Direct3D(HINSTANCE hInstance); // constructor

**virtual** ~Direct3D(); // destructor

// main app loop

**int** Run();

// framework

**virtual** **bool** Init();

**virtual** **void** Update(**float** dt) = 0; // = 0 = MUST be overridden, so no need to define

**virtual** **void** Render(**float** dt) = 0; // ditto.

**virtual** LRESULT MsgProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam); // handles all WINDOW messages

**void** EndScene(); // own-defined RENDER epilogue.

**void** ViewSetup(); // not currently used.

**void** BeginScene(); // own-defined RENDER prologue

**void** drawInfo(); // splash screen info

**void** drawMenu(); // splash screen menu

**void** statistics(**int** rotID); // draw the info currently relating to the transformation based on the trans-ID parameter

**protected**: // inheriting classes have access to these.

// win32 attributes

HWND m\_hwndAppWindow; // handle to our window

HINSTANCE m\_hInstance; // our instance to the window

UINT m\_nClientWidth; // window width

UINT m\_nClientHeight; // window height

**const** **char**\* m\_chAppTitle; // window title

DWORD m\_dwWindowStyle; // window style (redraw-on-resize)

//dx runtime variables

LPD3DXFONT g\_Font; // font object

IDirect3D9\* m\_pDirect3D; // direct3D-interface pointer

IDirect3DDevice9\* m\_pDevice; // pointer to the device

D3DPRESENT\_PARAMETERS m\_pD3dpp; // current parameters of the machine (struct)

D3DDISPLAYMODE m\_Mode; // display mode

D3DDEVTYPE m\_DevType; // developer options.

**protected**:

**bool** InitializeWindow(); // start the window procedure – this will initialize all the above variables

**bool** InitializeD3D(); // initialize direct3d from the window initialization.

// essentially. InitializeWindow() creates the frame for Direct3D to play with.

// however, Direct3D needs to be initialized first, via InitializeD3D.

};