// (c) Martyn Prigmore 2013 and beyond. All rights reserved.

//

// Declares classes we need to run a Windows Store app and provides member

// variables to hold our game and game timer objects.

//

// Notes

// 1) Declares a class that implements the IFrameworkView interface; the main

// class for a Windows Store App

// 2) Declares a class that implements the IFrameworkViewSource interface; the

// factory class that can create an IFrameworkView object

//

// Design simplifications for a teaching tool

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// This code is intended as a teaching tool for first year students of games programming.

// It provides a very simple starting point for a Windows Store DirectX game.

// The design makes some simplyfying assumptions, which are not realistic for a

// proper game, but are reasonable in a teaching tool.

// 1) The Windows Store app itself is only ever in two states:

// EITHER the window is visible and the game is running;

// OR the window is not visible and the game is not running.

// There is no "Game Paused" or "Loading resources" state.

#include <SpriteGame.h>

#include "Common\_DirectX/BasicTimer.h"

ref class SimpleDirectXGame sealed : public Windows::ApplicationModel::Core::IFrameworkView

{

public:

SimpleDirectXGame();

// IFrameworkView methods.

virtual void Initialize(Windows::ApplicationModel::Core::CoreApplicationView^ applicationView);

virtual void SetWindow(Windows::UI::Core::CoreWindow^ window);

virtual void Load(Platform::String^ entryPoint);

virtual void Run();

virtual void Uninitialize();

protected:

//Event handler methods ...

//... for critical process lifetime events

void OnActivated(Windows::ApplicationModel::Core::CoreApplicationView^ sender, Windows::ApplicationModel::Activation::IActivatedEventArgs^ arguments);

void OnSuspending(Platform::Object^ sender, Windows::ApplicationModel::SuspendingEventArgs^ arguments);

void OnResuming(Platform::Object^ sender, Platform::Object^ arguments);

//... for other Windows events

void OnWindowClosed(Windows::UI::Core::CoreWindow^ sender, Windows::UI::Core::CoreWindowEventArgs^ arguments);

void OnVisibilityChanged(Windows::UI::Core::CoreWindow^ sender, Windows::UI::Core::VisibilityChangedEventArgs^ arguments);

private:

bool windowClosed;

bool windowVisible;

//Provide the app with a game object and a timer object

SpriteGame theGame;

BasicTimer gameTimer;

};

ref class SimpleDirectXGameViewSource sealed : Windows::ApplicationModel::Core::IFrameworkViewSource

{

public:

virtual Windows::ApplicationModel::Core::IFrameworkView^ CreateView();

};