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

//

// Defines the classes to run a Windows Store game.

#include "pch.h"

#include <SimpleDirectXGame.h>

//Common namespaces

using namespace Windows::ApplicationModel;

using namespace Windows::ApplicationModel::Core;

using namespace Windows::ApplicationModel::Activation;

using namespace Windows::UI::ApplicationSettings;

using namespace Windows::UI::Core;

using namespace Windows::Foundation;

using namespace Microsoft::WRL;

using namespace DirectX;

using namespace Platform;

//Step1: Define the main method, the starting point for our app

[MTAThread]

int main(Array<String^>^ arguments)

{

auto direct3DApplicationSource = ref new SimpleDirectXGameViewSource();

CoreApplication::Run(direct3DApplicationSource);

return 0;

}

//Step2: Implement method for our IFrameworkViewSource class (SimpleDirectXGameViewSource )

IFrameworkView^ SimpleDirectXGameViewSource::CreateView()

{

//Create an instance of our IFrameworkView class and return it

return ref new SimpleDirectXGame();

}

//Step3: Implement methods for our IFrameworkView class (SimpleDirectXGame)

//Note that we use a C++ initialization list to initialize the member variables

SimpleDirectXGame::SimpleDirectXGame() :

windowClosed(false)

, windowVisible(true)

{

}

void SimpleDirectXGame::Initialize(CoreApplicationView^ applicationView)

{

//Set up Windows event handlers for the critical process lifetime events

applicationView->Activated += ref new TypedEventHandler<CoreApplicationView^, IActivatedEventArgs^>(this, &SimpleDirectXGame::OnActivated);

CoreApplication::Suspending += ref new EventHandler<SuspendingEventArgs^>(this, &SimpleDirectXGame::OnSuspending);

CoreApplication::Resuming += ref new EventHandler<Object^>(this, &SimpleDirectXGame::OnResuming);

}

void SimpleDirectXGame::SetWindow(CoreWindow^ ourWindow)

{

//Set up Windows event handlers for changes to the window

ourWindow->Closed += ref new TypedEventHandler<CoreWindow^, CoreWindowEventArgs^>(this, &SimpleDirectXGame::OnWindowClosed);

ourWindow->VisibilityChanged += ref new TypedEventHandler<CoreWindow^, VisibilityChangedEventArgs^>(this, &SimpleDirectXGame::OnVisibilityChanged);

//Initialize the game renderer, which draws to this window

this->theGame.Initialize(ourWindow);

}

void SimpleDirectXGame::Load(Platform::String^ entryPoint)

{

}

// This method is called after the window becomes active.

void SimpleDirectXGame::Run()

{

//Run a loop that handles Windows events and plays the game

while(!this->windowClosed)

{

if (this->windowVisible)

{

//Update the game timer

this->gameTimer.Update();

//Process all events in the queue ...

CoreWindow::GetForCurrentThread()->Dispatcher->ProcessEvents(CoreProcessEventsOption::ProcessAllIfPresent);

//... then run the game

this->theGame.Update(this->gameTimer.TickTotal(), this->gameTimer.TickDelta(), this->gameTimer.TimeElapsed(), this->gameTimer.TimeDelta());

this->theGame.Render();

//Finally, present the rendered frame to the monitor

this->theGame.Present();

}

else

{

//Wait for an event that makes the game visible again

CoreWindow::GetForCurrentThread()->Dispatcher->ProcessEvents(CoreProcessEventsOption::ProcessOneAndAllPending);

}

}

}

void SimpleDirectXGame::Uninitialize()

{

//Specialist clean-up tasks

}

//Implement the event handler methods

void SimpleDirectXGame::OnActivated(CoreApplicationView^ sender, IActivatedEventArgs^ arguments)

{

CoreWindow^ ourWindow = CoreWindow::GetForCurrentThread();

ourWindow->Activate();

}

void SimpleDirectXGame::OnSuspending(Object^ sender, SuspendingEventArgs^ arguments)

{

//Suspend any app resources that won't be automatically suspended; e.g. audio

}

void SimpleDirectXGame::OnResuming(Object^ sender, Object^ arguments)

{

//Resume any app resources you suspended manually in OnSuspending()

}

void SimpleDirectXGame::OnWindowClosed(CoreWindow^ sender, CoreWindowEventArgs^ arguments)

{

windowClosed = true;

}

void SimpleDirectXGame::OnVisibilityChanged(CoreWindow^ sender, VisibilityChangedEventArgs^ arguments)

{

windowVisible = arguments->Visible;

}