# Sequence diagram

*Data members and member functions are described after the table.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MainWindow::** | **Sample3DSceneRenderer::** | **DeviceResources::** | **Cube::** | **SampleTextRenderer::** |
| m\_sample3DSceneRenderer  -> | m\_deviceResources  -> | DeviceResources  -> DeviceIndependentSetup |  |  |
| -> | Sample3DSceneRenderer  -> |  | Cube |  |
|  | -> |  |  | SampleTextRenderer |
| MainWindow  -> OnSwapChainPanelLoaded  -> OnXamlRootChanged  -> OnDpiChanged  -> | OnDpiChanged |  |  |  |
| -> | SetWindowAndSwapChainPanel  -> | SetWindowAndSwapChainPanel  -> OutputSize |  |  |
| -> | StartRenderLoop  <- (starts task, then returns) |  |  |  |
|  | -> SetupAsync  -> ShaderSetupAsync  -> WindowIndependentSetup  -> | WindowIndependentSetup |  |  |
|  | -> |  |  | WindowIndependentSetup |
|  | -> CreateBuffers  -> |  | CreateBuffers |  |
|  | -> | WaitForGpu |  |  |
|  | -> |  | ReleaseUploadBuffers |  |
|  | -> WindowDependentSetup  -> | WindowDependentSetup  -> WaitForGpu  -> CreateSwapChain |  |  |
|  | -> | SetSwapChainOnSwapChainPanelAsync |  |  |
|  | -> UpdateViewMatrix |  |  |  |
|  | -> UpdateAndRender (loop) |  |  |  |
|  | if (dpi change queued) {  -> | DpiAndOutputSize  -> OutputSize |  |  |
|  | -> WindowDependentSetup  (etc.)  } |  |  |  |
|  | if (size change queued) {  -> | OutputSize |  |  |
|  | -> WindowDependentSetup  (etc.)  } |  |  |  |
|  | -> |  | Rotation |  |
|  | -> |  | Render  <- |  |
|  | WorldTransform |  |  |  |
|  |  |  | -> SetIAState |  |
|  | -> | AcquireWrappedRenderTarget |  |  |
|  | -> |  |  | UpdateAndRender |
|  | -> | ReleaseWrappedRenderTargetAndPresent  -> Present  -> MoveToNextFrame |  |  |
| -> OnSizeChanged  -> | OnSizeChanged |  |  |  |
| **MainWindow** | **Sample3DSceneRenderer** | **DeviceResources** | **Cube** | **SampleTextRenderer** |

**wWinMain**

**wWinMain** is defined in the generated file `App.xaml.g.hpp`. That entry points calls [Microsoft::UI::Xaml::Application::Start](https://docs.microsoft.com/windows/winui/api/microsoft.ui.xaml.application.start), passing a lambda as the [ApplicationInitializationCallback](https://docs.microsoft.com/windows/winui/api/microsoft.ui.xaml.applicationinitializationcallback). The lambda creates a new instance of the **App** class.

**App**

Represents the application. The **App** class has a constructor, and **OnLaunched** (which creates and activates a new **MainWindow**).

**MainWindow**

Represents the main window. Owns an automatic instance of **Sample3DSceneRenderer**. In its XAML, **MainWindow** has a **SwapChainPanel** (named swapChainPanel) and a **Button** (the button starts animating the cube).

**HWND m\_hWnd**

The window's HWND.

**Sample3DSceneRenderer m\_sample3DSceneRenderer**

An automatic instance of **Sample3DSceneRenderer**.

**MainWindow**

Calls **InitializeComponent**.

Registers **OnSwapChainPanelLoaded** to handle swapChainPanel().**Loaded**.

Via **IWindowNative**, stores the window's HWND in m\_hWnd.

Passes the window, HWND, and swapChainPanel() to m\_sample3DSceneRenderer.**SetWindowAndSwapChainPanel**.

Calls m\_sample3DSceneRenderer.**StartRenderLoop**.

Calls **SizeChanged**.

**OnAnimateButtonClick**

Calls m\_sample3DSceneRenderer.**Animate**, which starts animating the cube.

**OnDpiChanged**

Calls m\_sample3DSceneRenderer.**OnDpiChanged**.

**OnSizeChanged**

Calls m\_sample3DSceneRenderer.**OnSizeChanged**.

**OnSwapChainPanelLoaded**

Registers **OnXamlRootChanged** to handle swapChainPanel().XamlRoot().**Changed**.

**OnXamlRootChanged**

Calls **OnDpiChanged**.

**Sample3DSceneRenderer**

Represents the 3D scene. Owns an automatic instance of **DeviceResources**, a smart-pointer instance of **Cube**, a smart-pointer instance of **SampleTextRenderer**, and an automatic instance of **WorldViewProjectionConstantBuffer**.

**DeviceResources m\_deviceResources**

An automatic instance of **DeviceResources**.

**std::unique\_ptr<Cube> m\_pCube**

A smart-pointer instance of **Cube**.

**std::unique\_ptr<SampleTextRenderer> m\_pSampleTextRenderer**

A smart-pointer instance of **SampleTextRenderer**.

**WorldViewProjectionConstantBuffer m\_wvpConstantBufferData**

An automatic instance of **WorldViewProjectionConstantBuffer**.

**CreateBuffers**

Calls m\_pCube->**CreateBuffers**.

**ReleaseBuffers**

Calls m\_pCube->**ReleaseBuffers**.

**OnDpiChanged**

Queues a dpi change.

**OnSizeChanged**

Queues a size change.

**Reset**

Stops the render loop.

Calls **WindowDependentReset**.

Calls **WindowIndependentReset**.

**SetupAsync**

Main setup entry point.

Calls **ShaderSetupAsync**, and blocks.

Calls **WindowIndependentSetup**.

Calls **WindowDependentSetup**.

**SetWindowAndSwapChainPanel**

Calls m\_deviceResources.**SetWindowAndSwapChainPanel**.

**ShaderSetupAsync**

Asynchronously loads shaders.

**StartRenderLoop**

Creates a high-priority background task, if one's not already running.

If we're setting up, then the task calls **SetupAsync** if necessary.

The task then calls **UpdateAndRender** in a loop until canceled.

**UpdateAndRender**

If a dpi change is queued, calls m\_deviceResources.**DpiAndOutputSize**, and then **WindowDependentSetup**.

If a size change is queued, calls m\_deviceResources.**OutputSize**, and then **WindowDependentSetup**.

Calls m\_pCube->**Rotation**.

Calls m\_pCube->**Render**.

Calls m\_deviceResources.**AcquireWrappedRenderTarget**.

Calls m\_pSampleTextRenderer->**UpdateAndRender**.

Calls m\_deviceResources.**ReleaseWrappedRenderTargetAndPresent**.

If the device is lost, calls **Reset**, and then **StartRenderLoop**.

**UpdateViewMatrix**

Stores the view transform matrix in m\_wvpConstantBufferData.view.

**WindowDependentReset**

Calls m\_deviceResources.**WindowDependentReset**.

**WindowDependentSetup**

Calls m\_deviceResources.**WindowDependentSetup**.

Calls m\_deviceResources.**SetSwapChainOnSwapChainPanelAsync** (which is fire and forget).

Calls m\_deviceResources.**GetOutputSizeInDIPs**, and stores the world-view-projection matrix in m\_wvpConstantBufferData.projection.

Calls **UpdateViewMatrix**.

**WindowIndependentReset**

Calls **ReleaseBuffers**.

Calls m\_pSampleTextRenderer->**WindowIndependentReset**.

Calls m\_deviceResources.**WindowIndependentReset**.

**WindowIndependentSetup**

Calls m\_deviceResources.**WindowIndependentSetup**.

Calls m\_pSampleTextRenderer->**WindowIndependentSetup**.

Creates a root signature with a single constant buffer slot (accessible to the IA stage).

Creates a pipeline state object.

Creates a command list.

Calls **CreateBuffers**.

Closes and executes the command list.

Waits for the GPU, and calls m\_pCube->**ReleaseUploadBuffers**.

**WorldTransform**

Sets the value of m\_wvpConstantBufferData.world.

**DeviceResources**

Represents the Direct3D device, swap chain, and other necessary bits and pieces.

**DeviceResources**

Calls **DeviceIndependentSetup**.

**CreateSwapChain**

Creates (or resizes) the swap chain.

Creates render target views for the back buffers.

Creates wrapped 11On12 render targets for the back buffers.

Creates a depth stencil and a depth stencil view.

Sets the viewport and scissor rect to the entire window.

**DeviceIndependentSetup**

Creates factories for Direct2D, DirectWrite, and WIC.

**DpiAndOutputSize**

Sets the current dpi on the Direct2D context.

Calls **OutputSize**.

**MoveToNextFrame**

Queues the GPU to update the fence to the current frame's value.

Waits for the fence to reach the oldest frame's value.

**OutputSize**

Takes and stores an output size in either DIPs or raw pixels.

**Present**

Calls **Present** on the swap chain.

Calls **MoveToNextFrame**.

Returns *false* if device lost, otherwise *true*.

**ReleaseSwapChain**

Releases the swap chain, depth stencil, render targets, and wrapped render targets.

**ReleaseWrappedRenderTargetAndPresent**

Releases the 11On12 wrapped render target resource for the current back buffer.

Calls **Flush** on the Direct3D 11 device context.

Calls **Present**.

Returns *false* if device lost, otherwise *true*.

**SetSwapChainOnSwapChainPanelAsync**

Asynchronously switches to the window object's dispatcher queue.

Via **ISwapChainPanelNative**, sets the swap chain on the swap chain panel.

**SetWindowAndSwapChainPanel**

Takes and stores the window object, its HWND, and the swap chain panel.

Calls **OutputSize**.

**Trim**

Calls **Trim** on the DXGI device.

**AcquireWrappedRenderTarget**

Acquires the 11On12 wrapped render target resource for the current back buffer.

Sets the Direct2D target bitmap as the current target.

**WaitForGpu**

Queues the GPU to update the fence to the current frame's value.

Waits for the fence to reach that value.

**WindowDependentReset**

Calls **ReleaseSwapChain**.

Sets the Direct2D target bitmap to nullptr.

**WindowDependentSetup**

Clears the previous window-size-dependent content.

Calls **Flush** on the Direct3D 11 device context.

Calls **WaitForGpu**.

Sets all the fence values to ones that haven't yet been signaled.

Stores the render target size (the output size in raw pixels) that'll be used when the swap chain is created.

Calls **CreateSwapChain**.

Returns *false* if device lost, otherwise *true*.

**WindowIndependentReset**

Calls **Trim**.

Releases all the window-independent Direct2D, Direct3D, and DXGI resources.

**WindowIndependentSetup**

Create a DXGI factory.

Enumerates adapters until one creates a suitable Direct3D 12 device.

Creates a Direct3D 12 command queue.

Creates an 11On12 device and a Direct3D 11 device context.

Creates a Direct2D device and context from the 11On12 device.

Creates descriptor heaps for the render target view and depth stencil view.

Creates a command allocator for each frame buffer.

Creates a fence and a fence event.

**Cube**

Represents a 3D model.

**Cube**

Creates and stores the cube's vertex and index data.

**~Cube**

Unmaps the world-view-projection matrix resource.

Calls **ReleaseBuffers**.

**CreateBuffers**

Creates a resource for the world-view-projection constant buffers, and maps it.

Creates a descriptor heap for the constant buffers.

Creates constant buffer views for accessing the upload buffer.

Creates vertex and index buffer resources on the GPU, and copies data into them.

Creates views for the vertices and indices.

**ReleaseBuffers**

Calls **ReleaseUploadBuffers**.

Releases other resources.

**ReleaseUploadBuffers**

Releases the upload buffers.

**Render**

Calls **Reset** on the current Direct3D 12 command allocator.

Calls **Reset** on the Direct3D 12 graphics command list.

Sets the root signature and descriptor heaps to be used by this frame.

Sets the viewport, and scissor rectangle.

Clears the render target and depth stencil.

Sets the render target.

Calls m\_sample3DSceneRenderer.**WorldTransform**.

Updates the world-view-projection constant buffer resource.

Binds the current frame's constant buffer to the pipeline.

Sets input assembler primitive topology.

Calls **SetIAState**.

Calls **DrawIndexedInstanced** on the Direct3D 12 graphics command list.

Calls **Close** on the Direct3D graphics command list.

Calls **ExecuteCommandLists** on the Direct3D 12 command queue.

**Rotation**

Updates the world transform with a rotation.

**SetIAState**

Calls **IASetVertexBuffers** and **IASetIndexBuffer** on the Direct3D 12 graphics command list.

**SampleTextRenderer**

Represents some text just to show how you can combine Direct 2D with Direct 3D and XAML.

**UpdateAndRender**

Creates and draws a text layout, using the text format and brush.

**WindowIndependentSetup**

Creates a text format and a brush.

**WindowIndependentReset**

Releases all the resources.