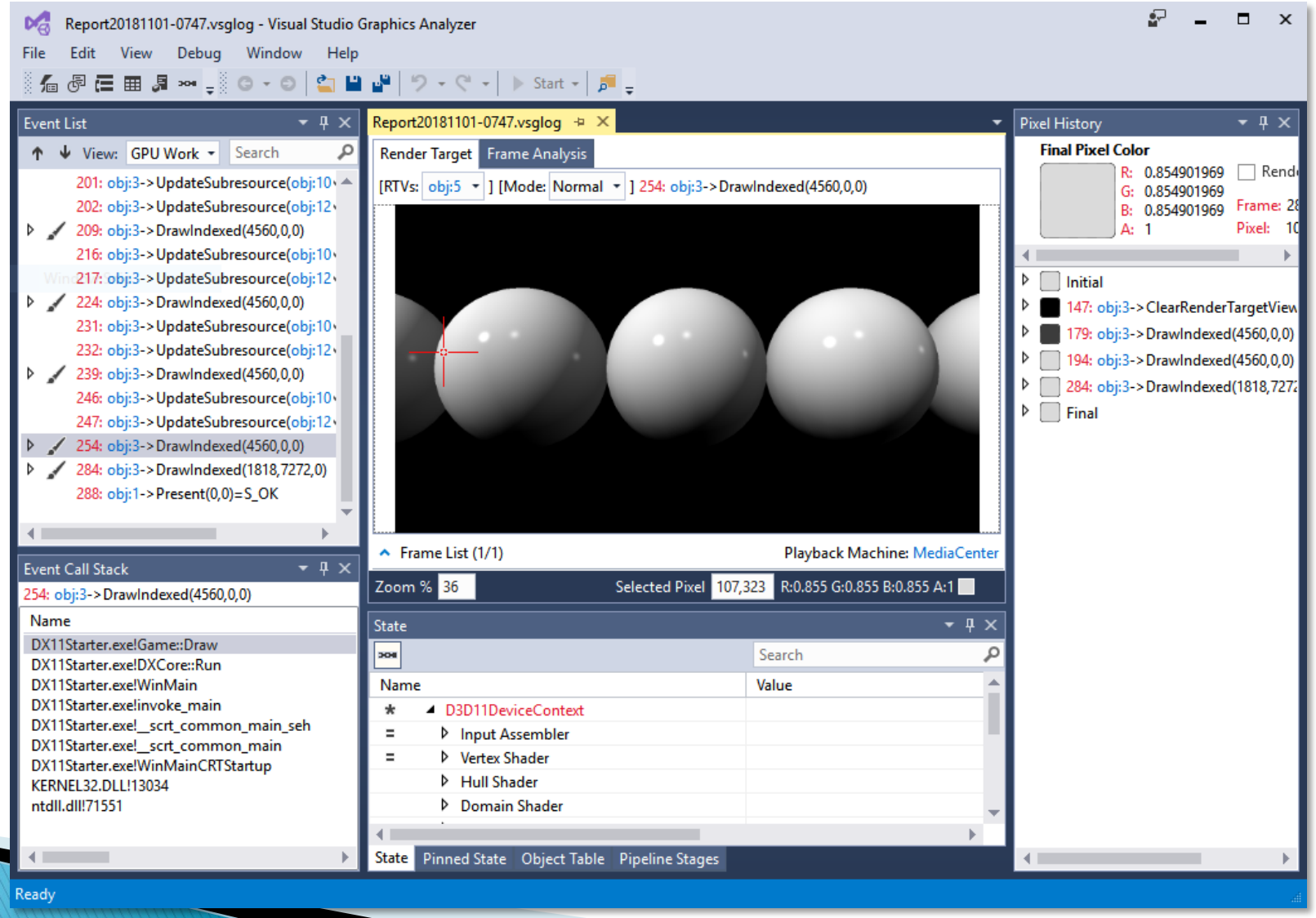
Visual Studio's Graphics Analyzer

AKA The Graphics Debugger

VS Graphics Analyzer

- Allows the capture and analysis (debugging) of individual frames of a DirectX application
- Including actual shader debugging!
  - Full step-over/into/out
  - Locals window with variables' contents
- Extremely useful for debugging DX drawing
  - Especially if objects aren't drawing at all

Graphics Analyzer overview



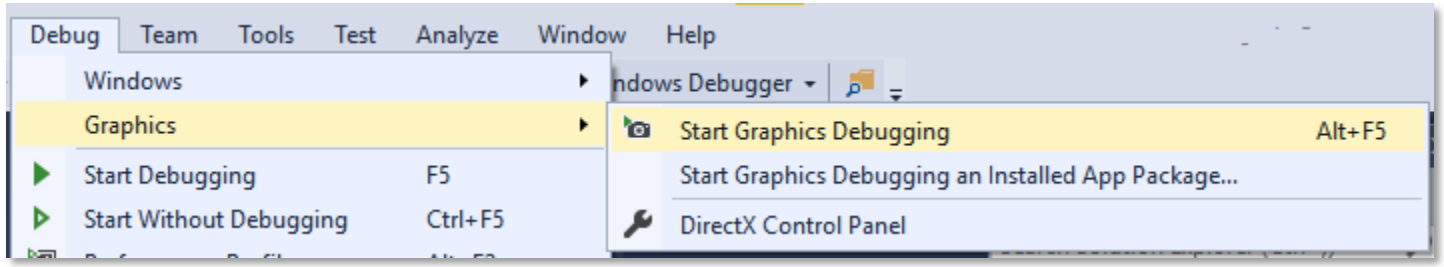
Graphics Analyzer features

- **Event List** – lists all DirectX API calls made
- **Event Call Stack** – shows *where* DX calls made
- **State** – Lists entire DX state for current event
- **Object Table** – Lists every live DX resource
- **Render Target** – Shows current output
- **Pixel History** – Lists changes to selected pixel
- **Pipeline Stages** – Visualizes each pipeline step

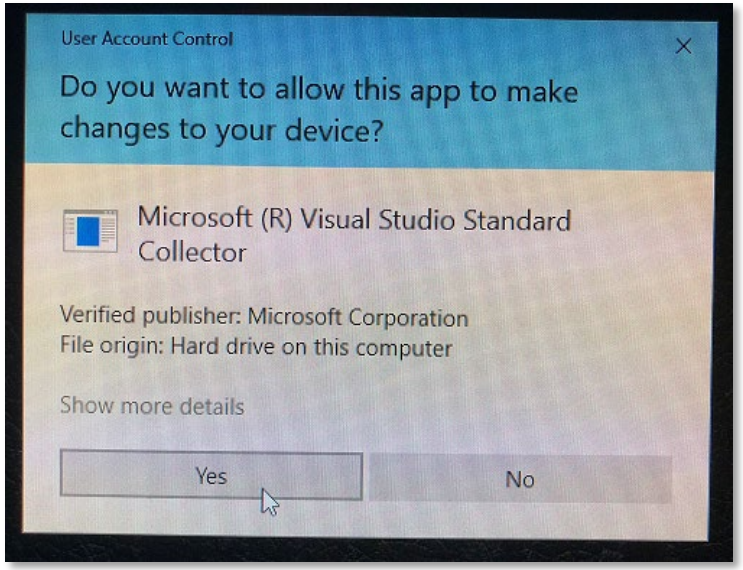
- **Shader debugging**
  - Can select individual verts/pixels to “debug”

Step 1: Start Graphics Debugging

- Shortcut is Alt+F5

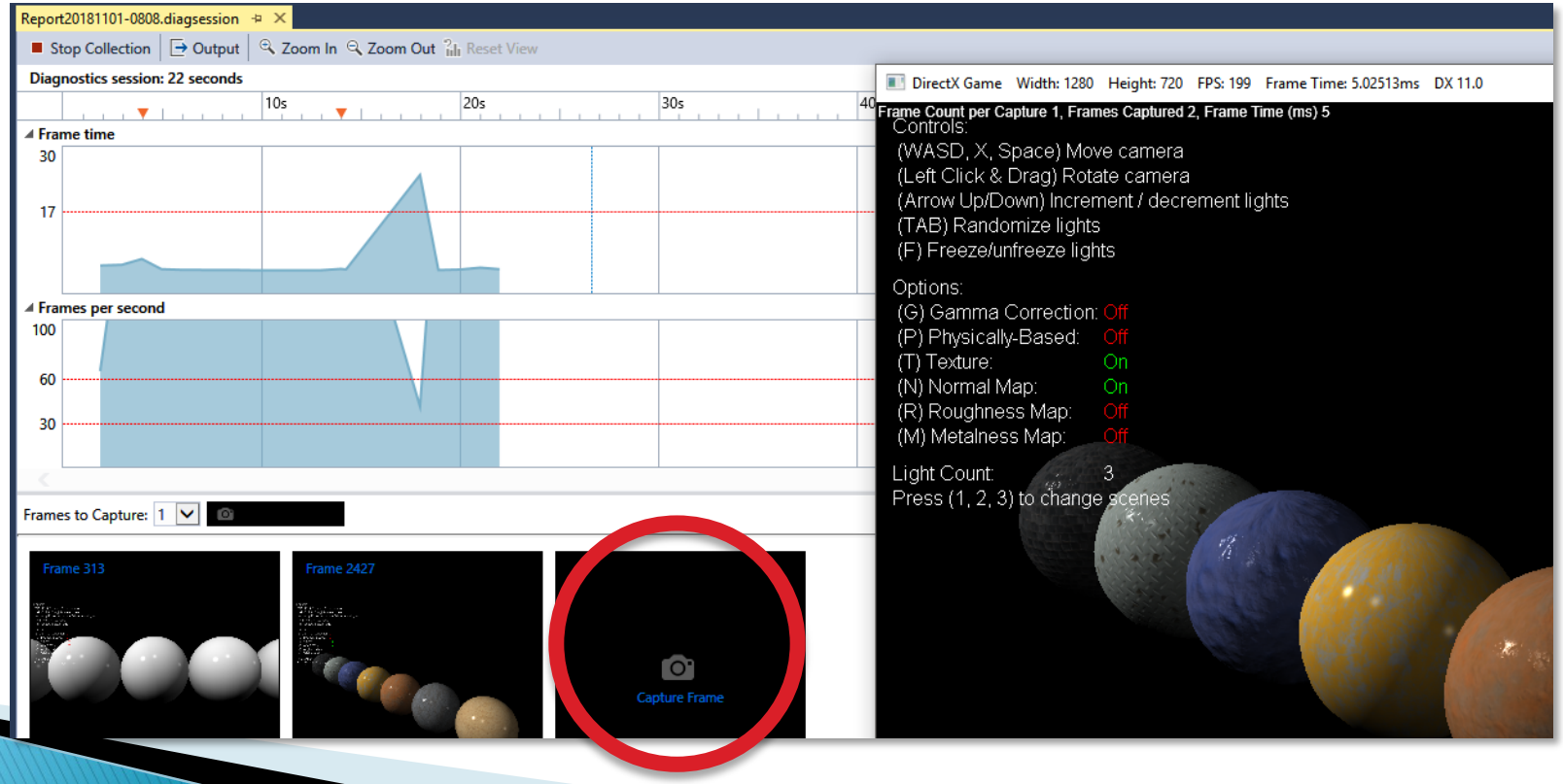


- Allow VS to capture data (every time!)



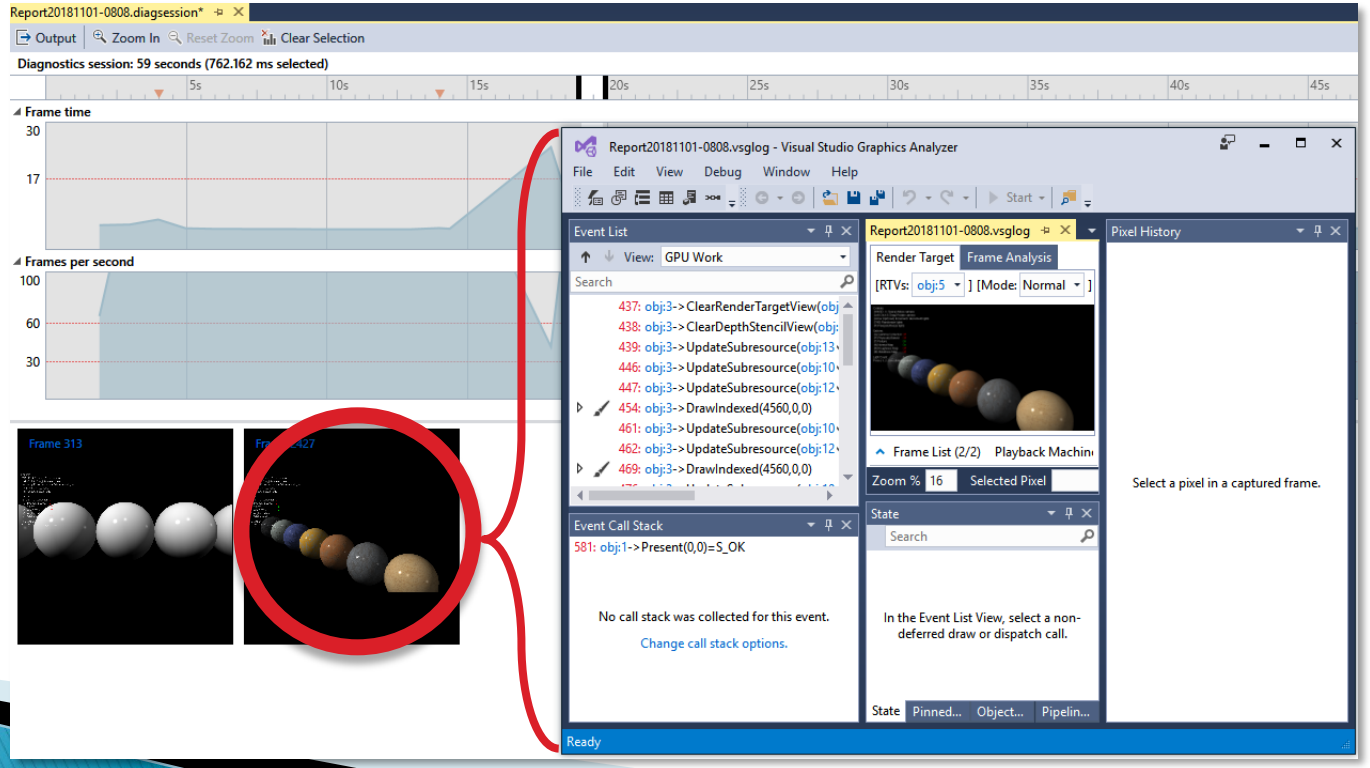
Step 2: Capture frame(s)

- Press “Print Screen” key
  - Or “Capture Frame” button in Visual Studio

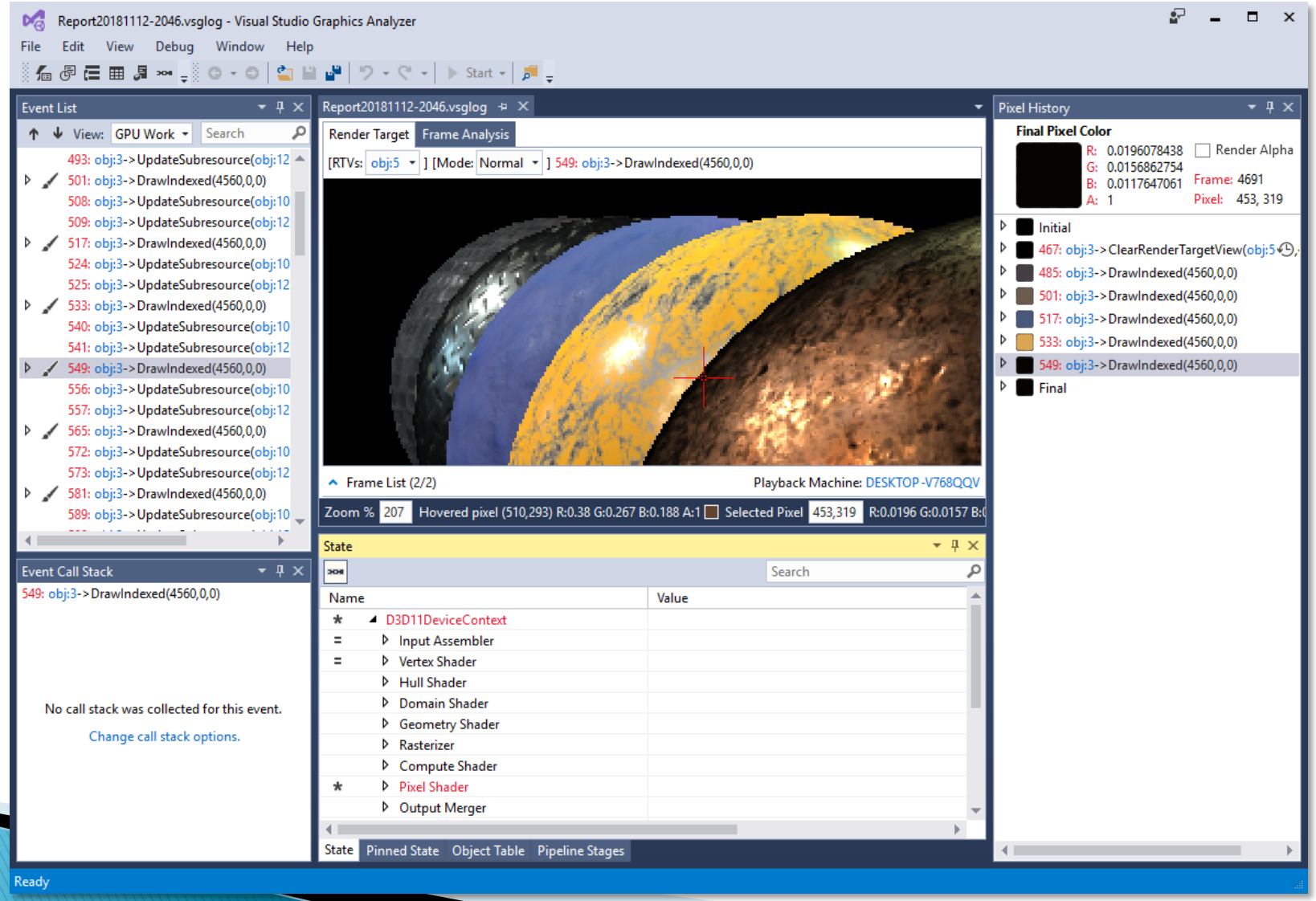


Step 3: Pick a frame to analyze

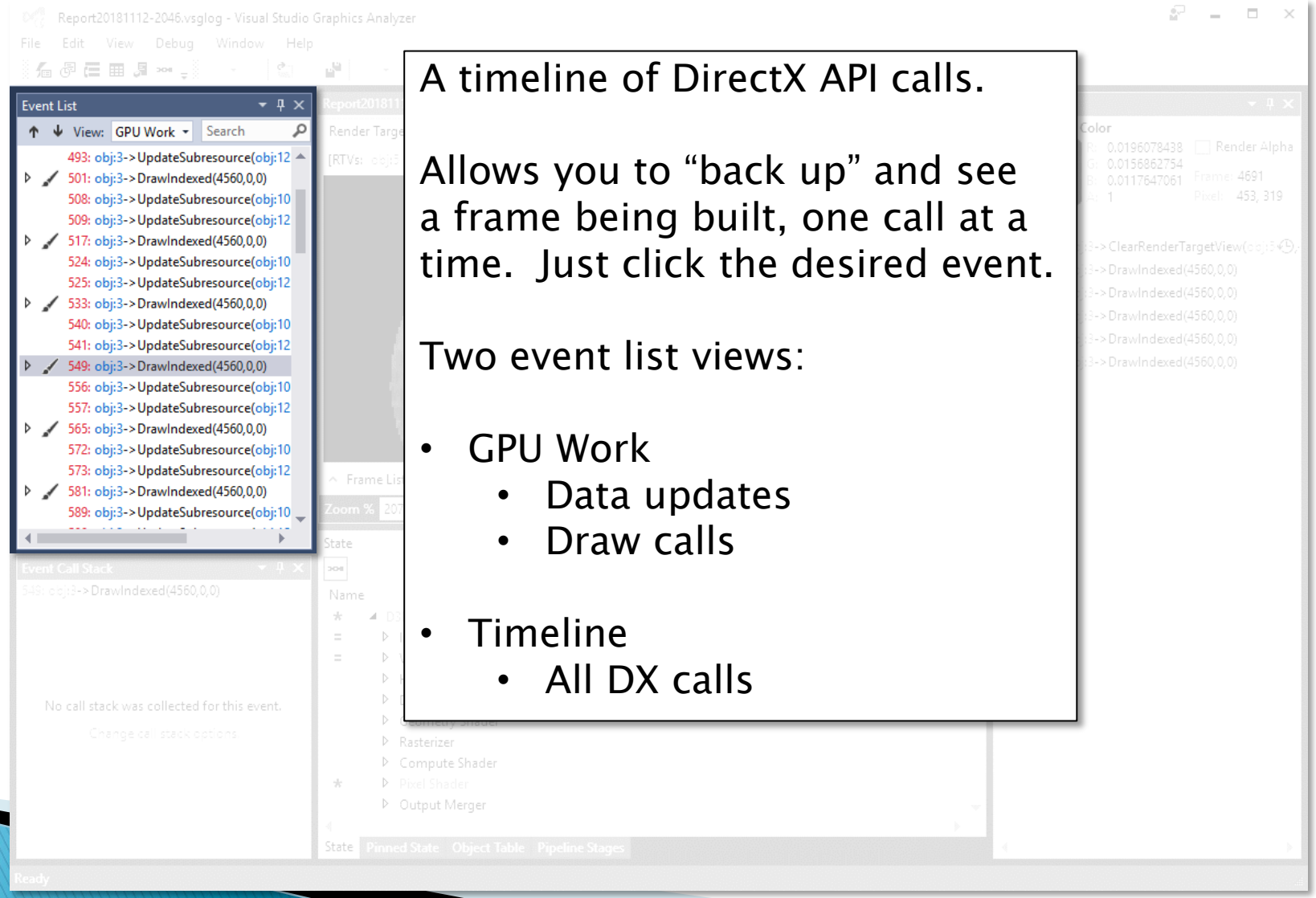
- 3a: Close your program
- 3b: Double click on a captured frame
  - This opens the Graphics Analyzer window



Step 4: Dig through your frame



Event List



A timeline of DirectX API calls.

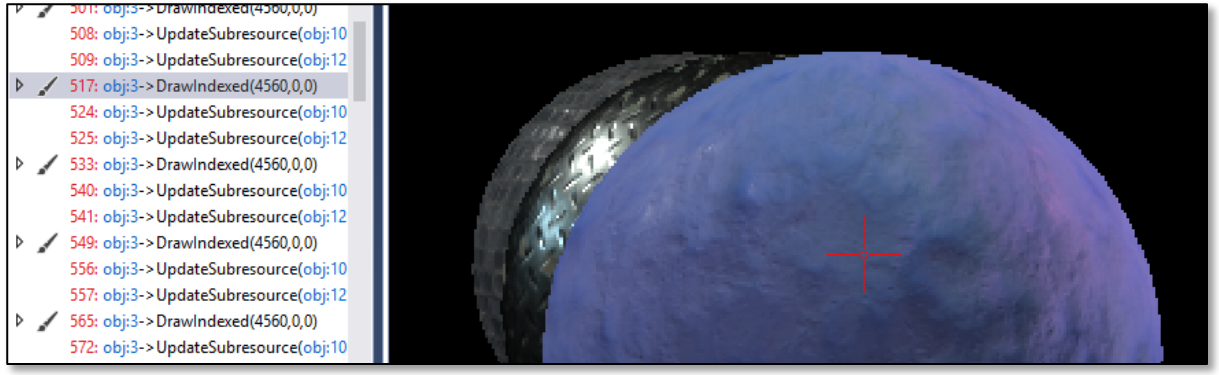
Allows you to “back up” and see a frame being built, one call at a time. Just click the desired event.

Two event list views:

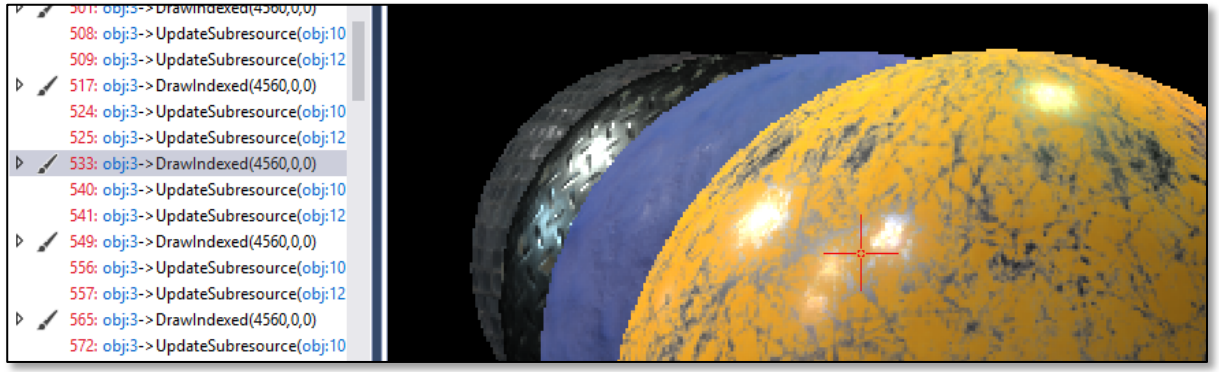
- GPU Work
  - Data updates
  - Draw calls
- Timeline
  - All DX calls

Event List – Timeline traversal

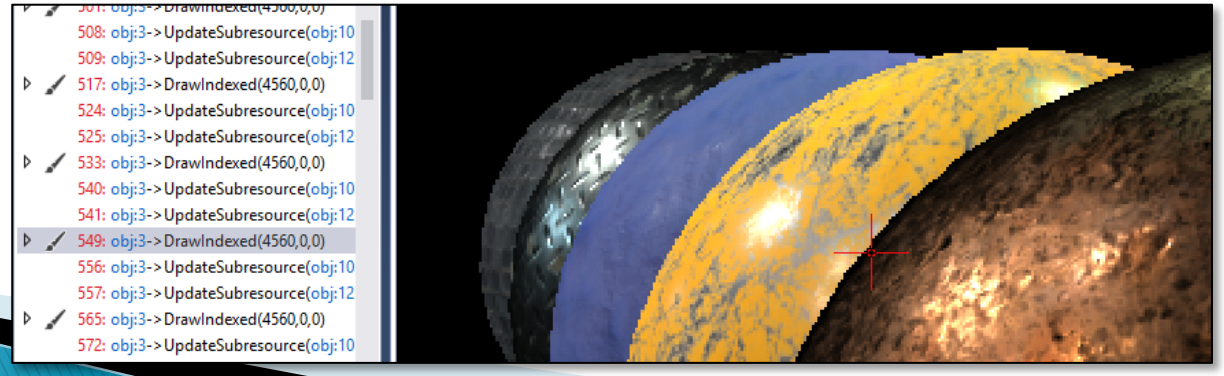
Event 517



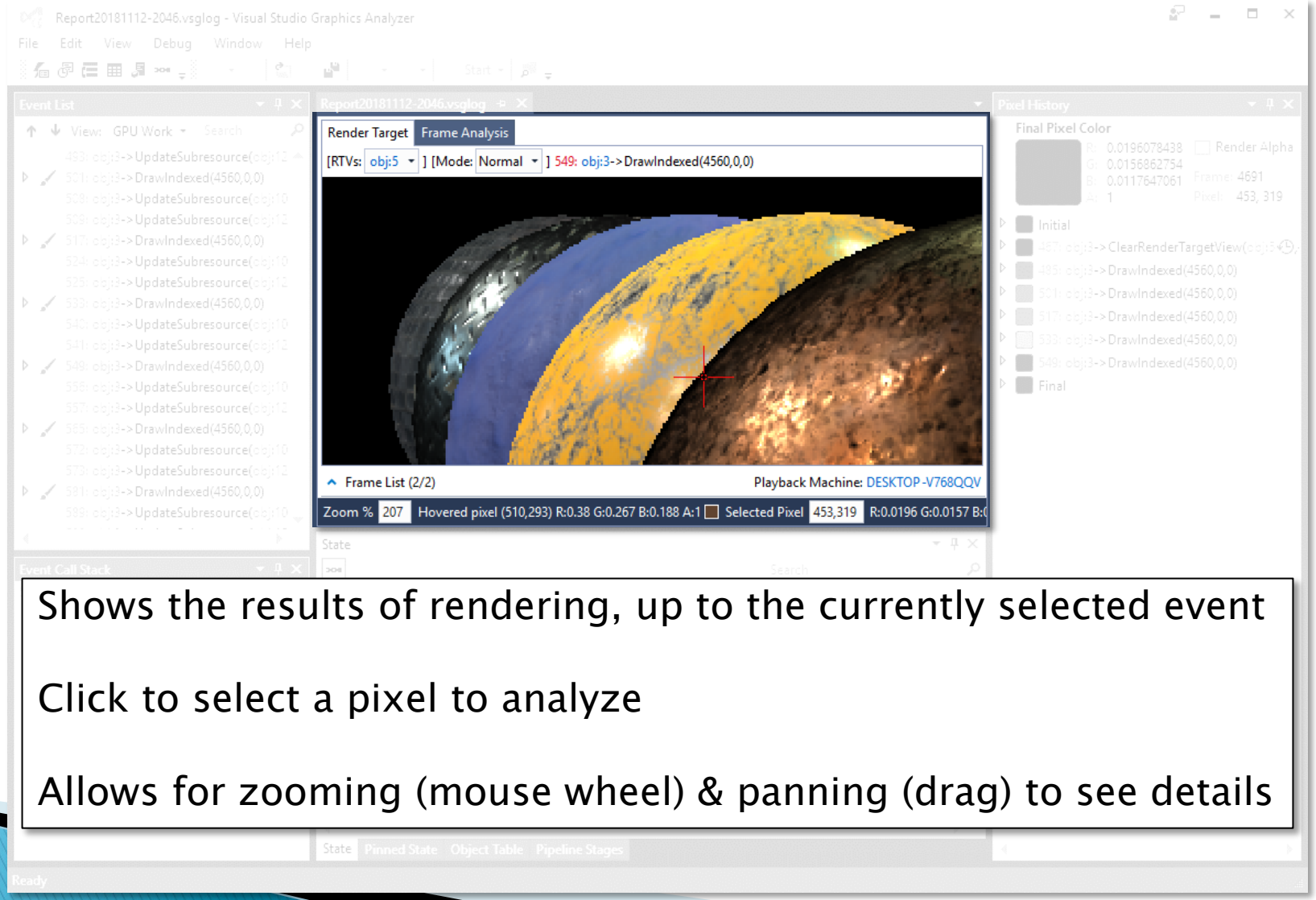
Event 533



Event 549



Render Target

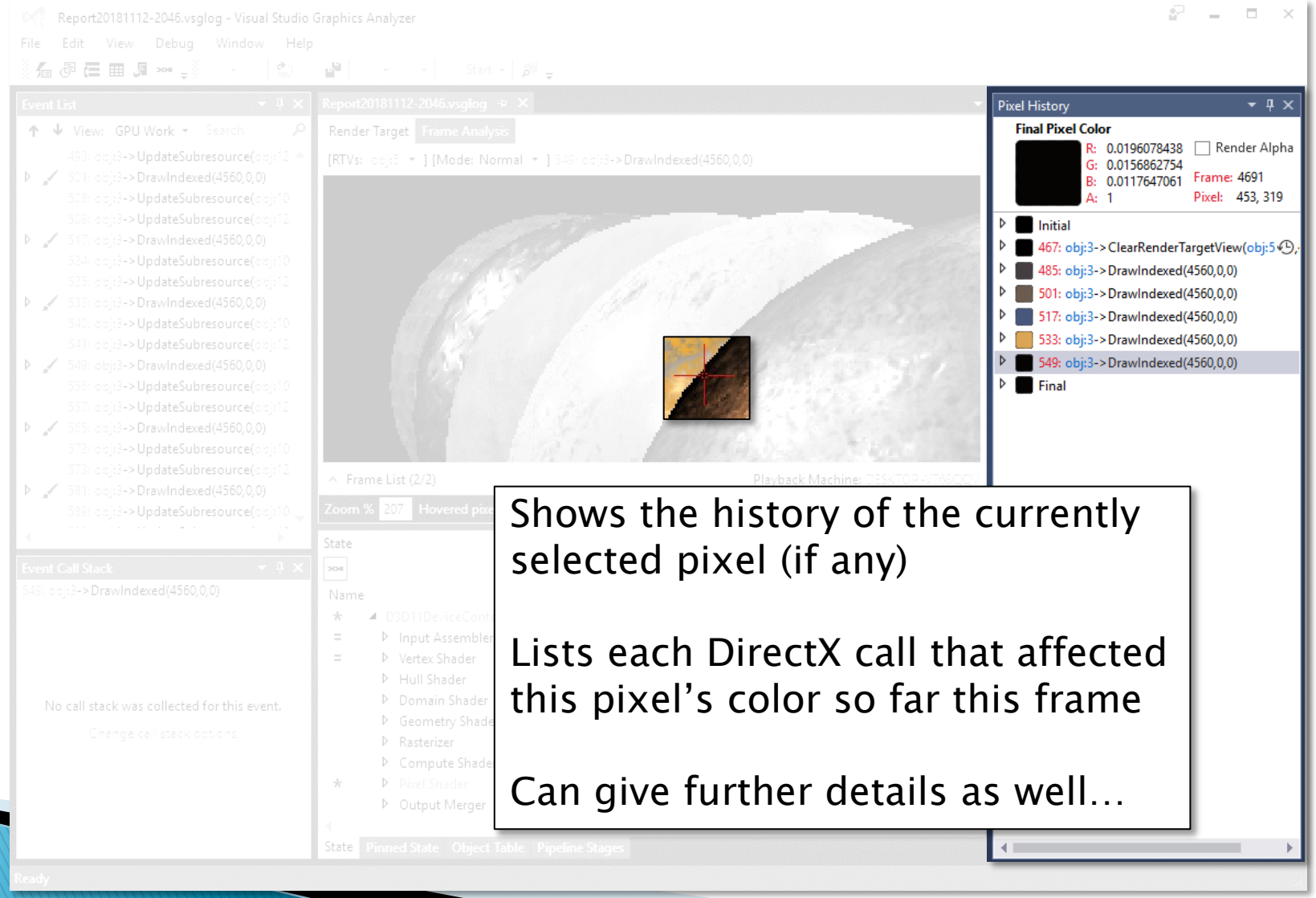


Shows the results of rendering, up to the currently selected event

Click to select a pixel to analyze

Allows for zooming (mouse wheel) & panning (drag) to see details

Pixel History



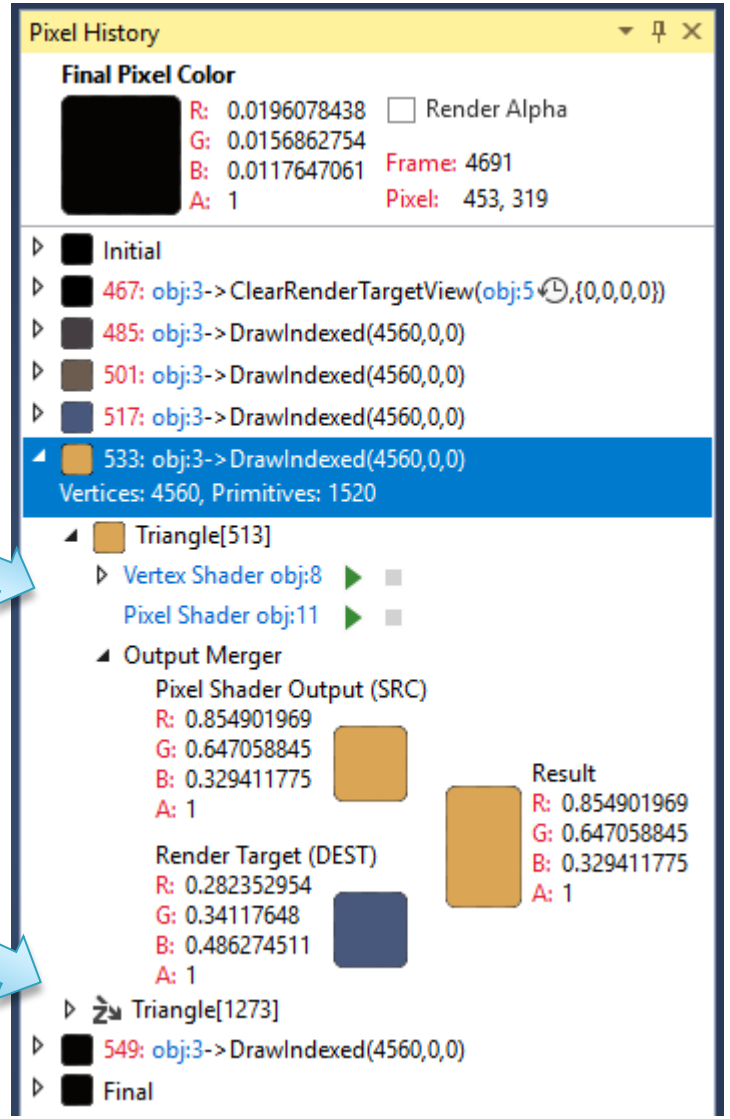
Shows the history of the currently selected pixel (if any)

Lists each DirectX call that affected this pixel's color so far this frame

Can give further details as well…

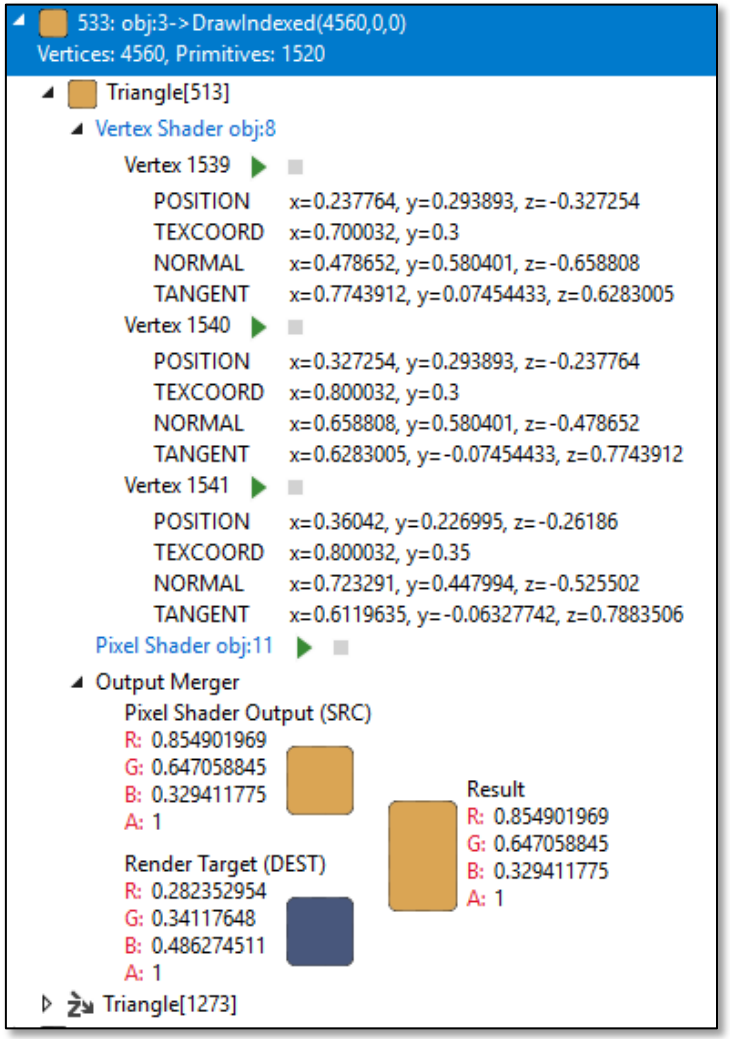
Pixel History – Details

- Expand a call for details
- Triangles rasterized here
  - Output of pixel shader
  - Result of O.M. stage
- Triangles that were culled
  - Not drawn for some reason



Pixel History – More details

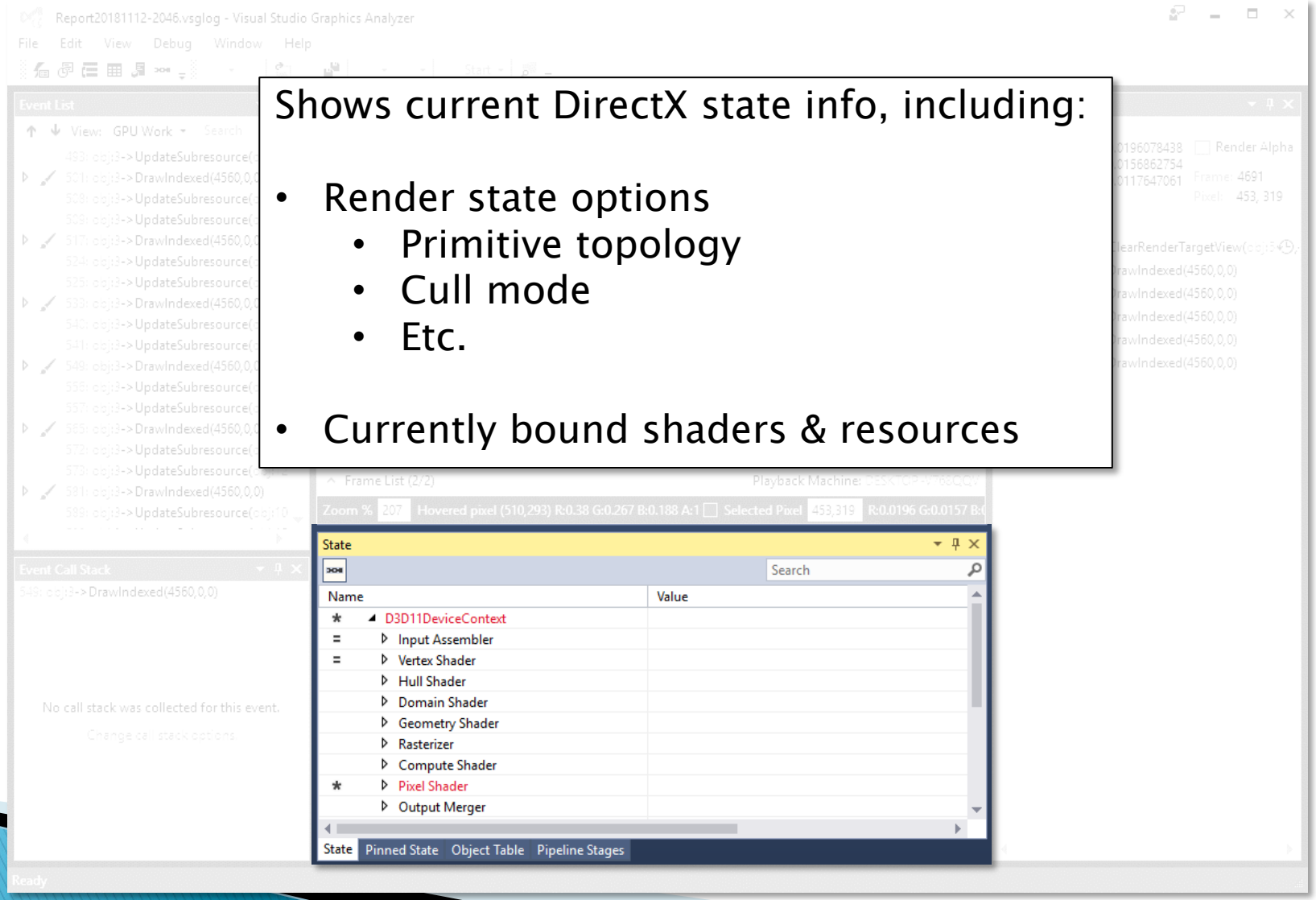
- Expand “Vertex Shader” for individual vertices
  - And their data
- Click green “play button” to **debug a shader**
  - Line by line debugging
  - For a particular vertex
  - Or this very pixel



State

Shows current DirectX state info, including:

- Render state options
  - Primitive topology
  - Cull mode
  - Etc.
- Currently bound shaders & resources

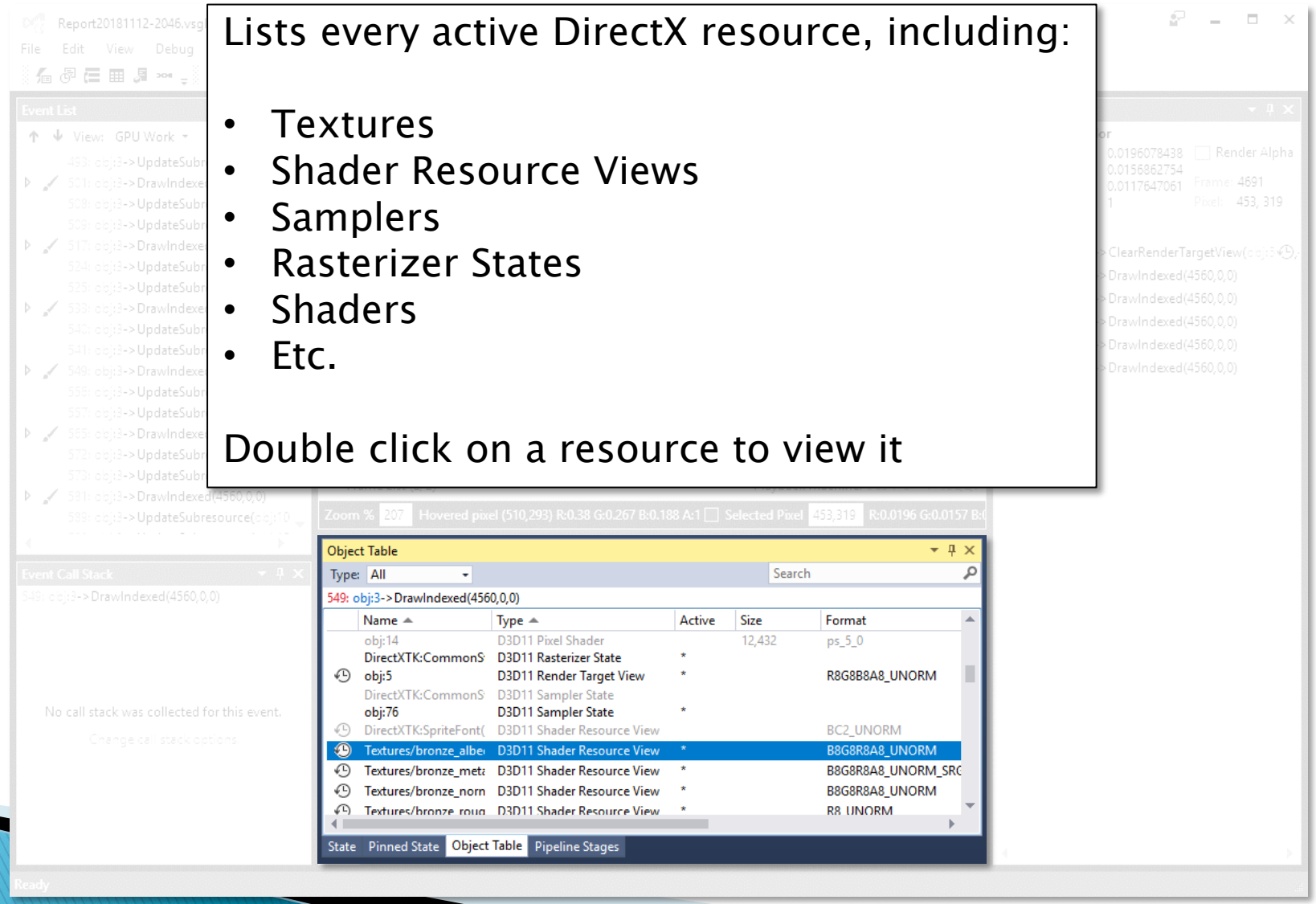


Object Table

Lists every active DirectX resource, including:

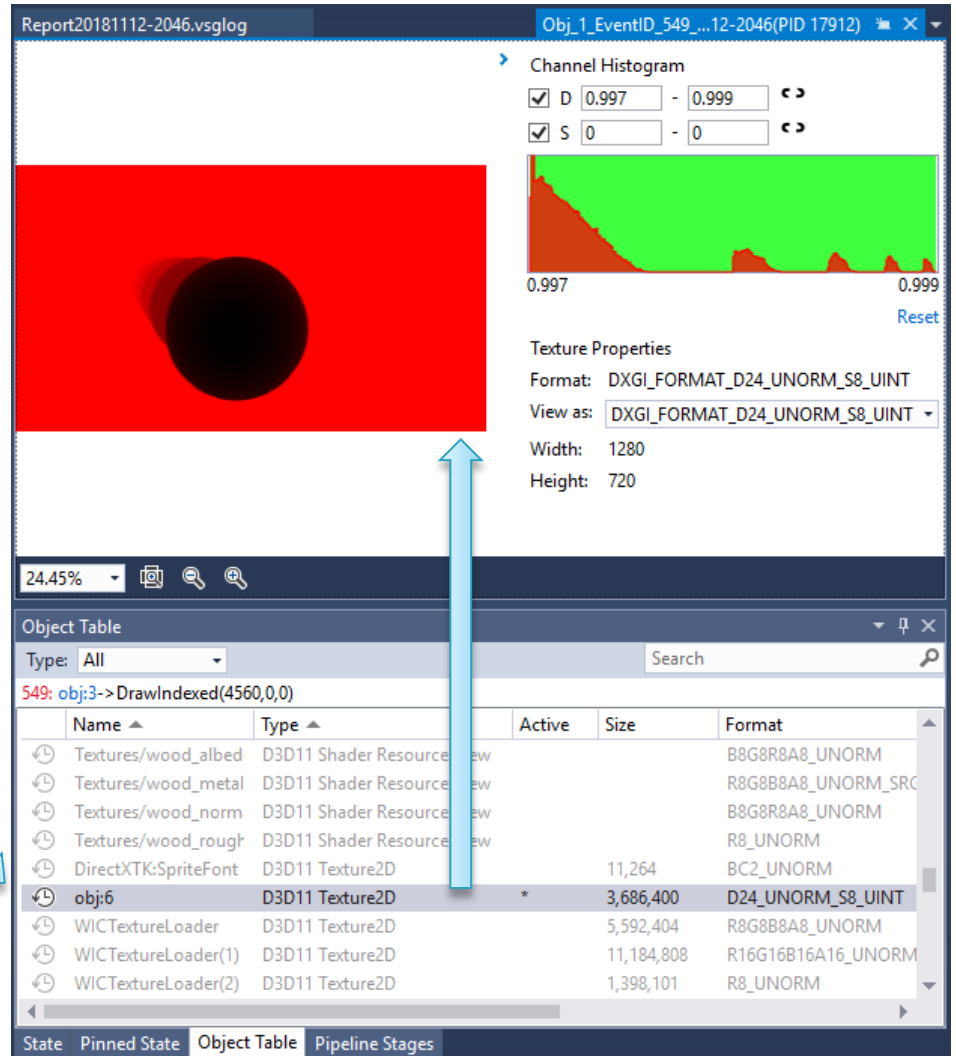
- Textures
- Shader Resource Views
- Samplers
- Rasterizer States
- Shaders
- Etc.

Double click on a resource to view it

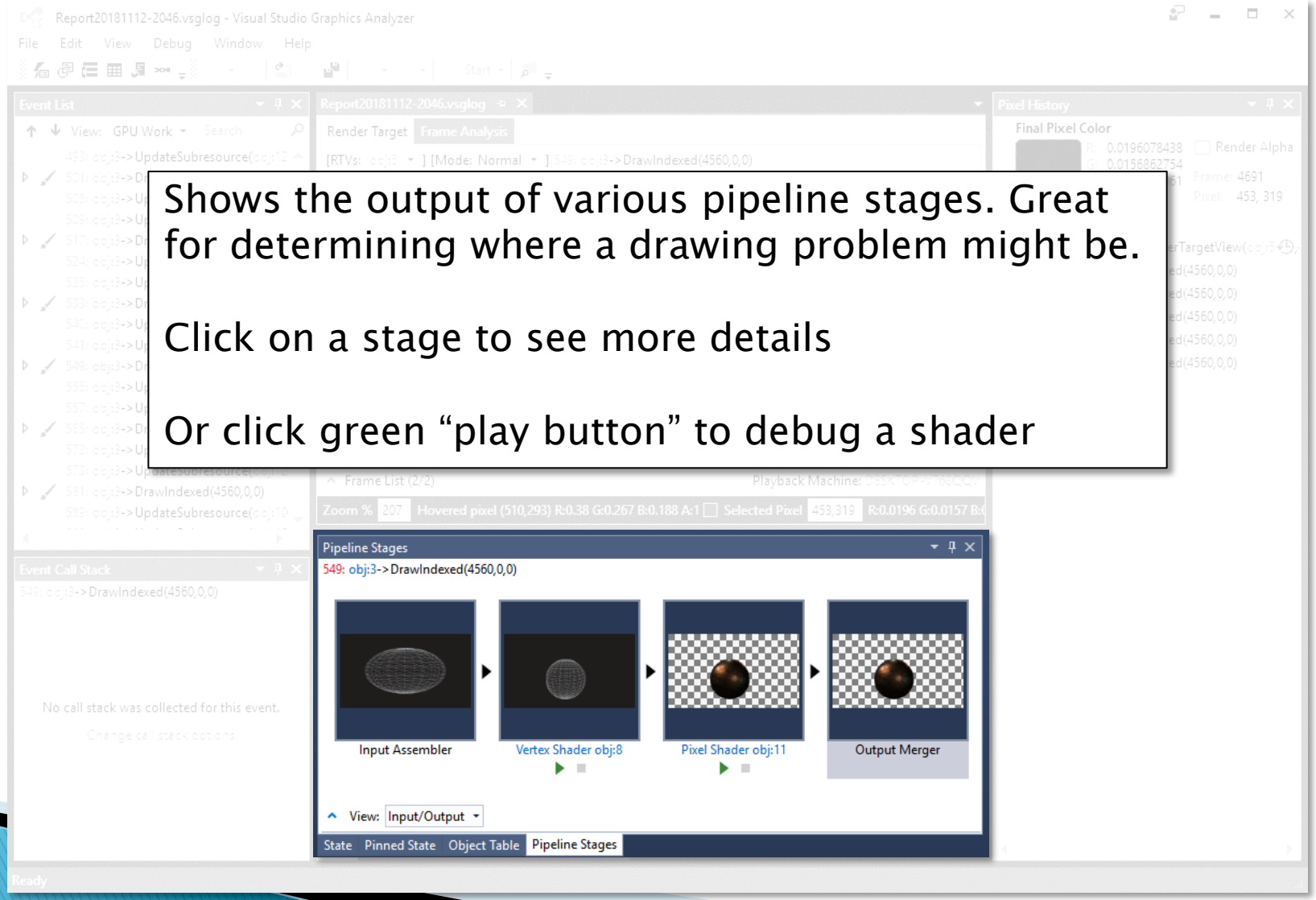


Object Table – Viewing a resource

- Double click the depth buffer to actually see it
  - Format is D24…
  - D for Depth



Pipeline Stages



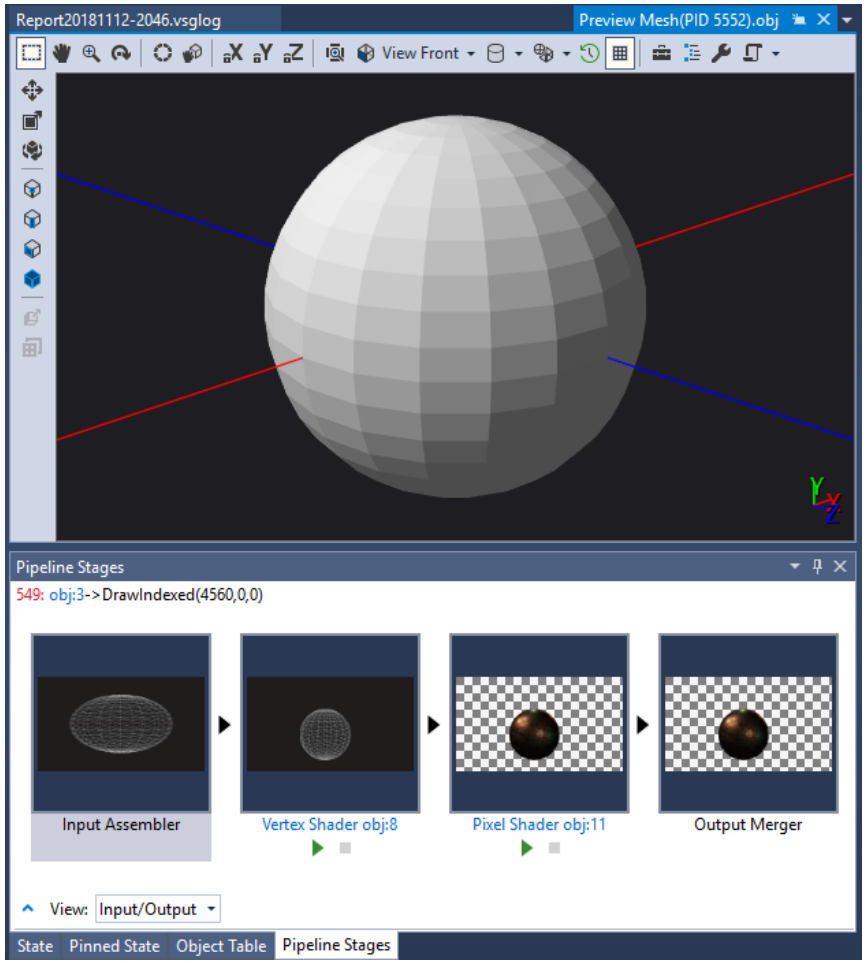
Shows the output of various pipeline stages. Great for determining where a drawing problem might be.

Click on a stage to see more details

Or click green “play button” to debug a shader

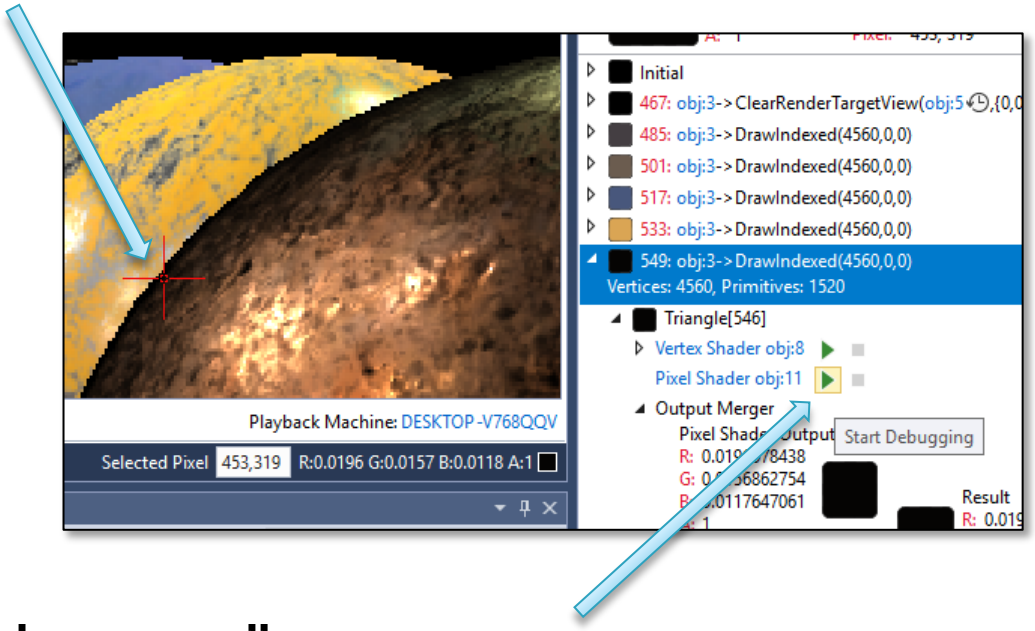
Pipeline Stage example: Input Assembler

- Click I.A. stage for a mesh preview
  - Right in Visual Studio
- Helps ensure you're drawing the proper geometry



Shader debugging

- Need to debug a shader?
  - First choose a pixel
  - And a draw call



- Then click the green “play button”