Render Targets

Rendering to off-screen buffers

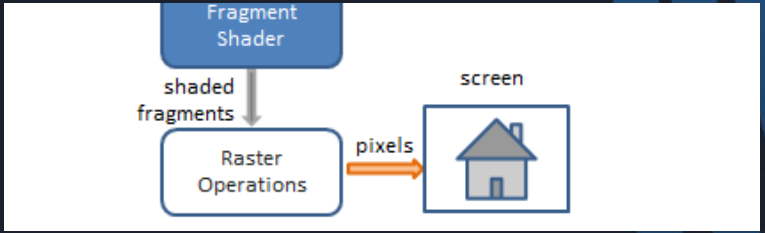
Programming – Computer Graphics

Contents

- Back Buffer
  - Double Buffering
- Off-screen Buffers
  - Effects
  - Benefits
- Reflections

Render Pipeline Output

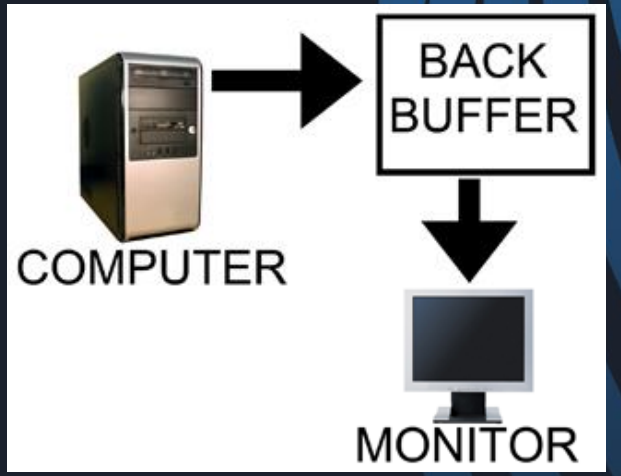
- Remembering back to the Render Pipeline…
  - Receives vertices during draw calls and rasterises an image



- The rasterised image has to go somewhere
- The pipeline outputs the rendered pixels into a buffer called a Frame Buffer
  - A type of Image Buffer, just like a Texture

Back Buffer

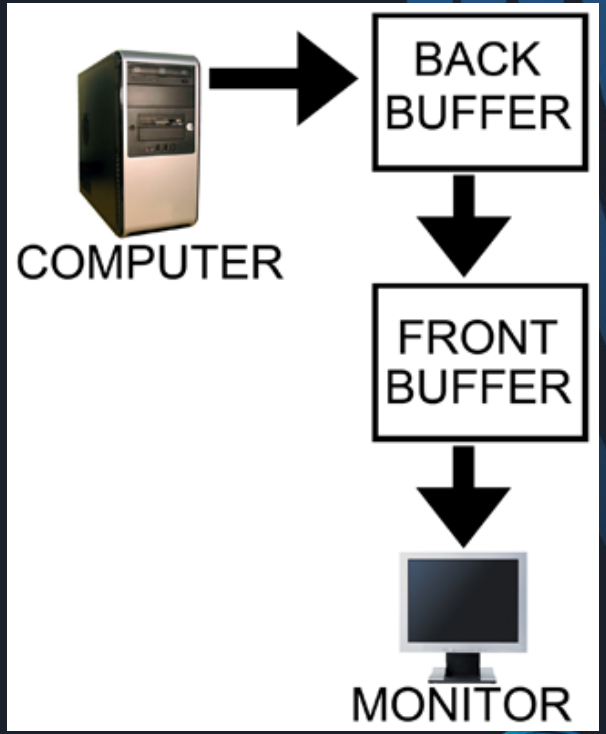
- The GPU uses a frame buffer to store the pixels that get displayed to the screen
  - Commonly called a Back Buffer
- At the end of each frame this buffer is presented to the screen for the user to see
  - While this is happening we can not render into the buffer
  - The screen's refresh rate limits the speed at which we can update this buffer
  - V-sync is the name of waiting for the screen to display before rendering
  - We can disable the V-sync which allows for higher frame rates, but frames are only partially displayed to the screen, which can cause a render artefact called tearing



Double Buffering

- We can also set it up to use more than 1 Back Buffer

- Doing this means we can render to the Back Buffer while the previous Back Buffer is being displayed to the user
  - When using an additional buffer this is called Double Buffering

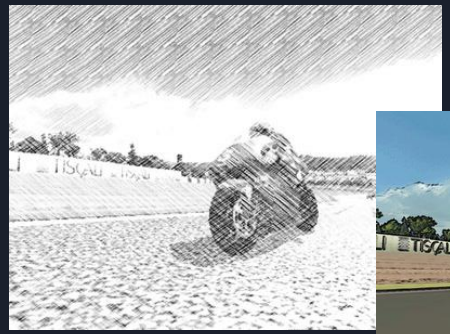


Off-Screen Buffers

- When we render a scene directly into the Back Buffer this is called Forward Rendering
- We aren't just limited to rendering to the Back Buffer though
- Rendering into any frame buffer is possible
  - These off-screen buffers are commonly referred to as Render Targets
  - We can even render to more than one at a time, referred to as Multiple Render Targets
- These buffers could then be used as input textures when rendering to any other buffer
  - This can give rise to many MANY new visual techniques!

Specifics? To name a few…

Depth of Field
Motion Blur



Non-Photorealistic Rendering (NPR)

Rendering to other surfaces

Bloom and High Dynamic Range (HDR) Lighting

Shadows

Specifics? To name a few...

Refractions…

…Reflections

…Lighting perfections

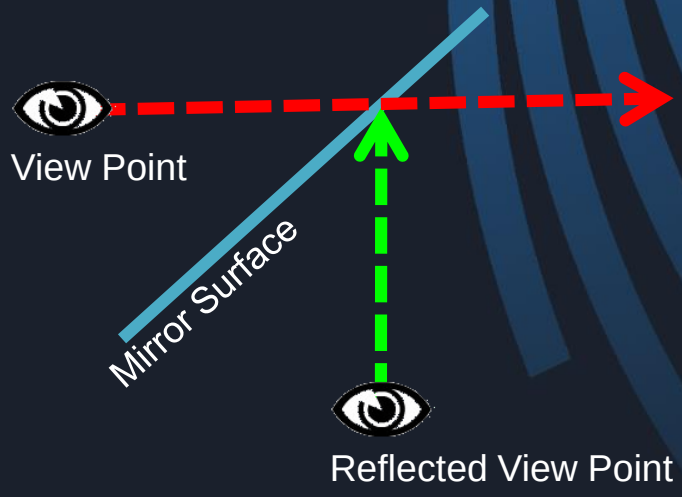
Benefits

- Currently when we render directly to the Back Buffer we don't have access to any of the pixel fragments around the one currently being rendered
- If instead we were to render the entire scene to a frame buffer, then render using that buffer as a texture stretched across the screen…
  - During the Fragment Shader we can sample the buffer at the matching pixel/texel location and see the original render
  - But we can also sample the pixels/texels around it!
  - If pixels have access to neighbouring pixels it can give rise to a great range of effects, such as Edge Detection

- We can also render from other angles or locations
  - For example, render from the view of a security camera then apply the frame buffer as a texture across a computer screen in front of the player

Reflections

- Reflections are a great example of render targets
- A scene is rendered to a Frame Buffer from a viewpoint that is reflected around a reflective surface, such as a mirror
- The scene is then rendered from the original viewpoint to the Back Buffer, and the target from the first pass is used as a texture across the reflective surface



Reflections

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Summary

- We don’t have to just render to the screen
- We can render to frame buffers and use them as textures during other draw calls
  - Can’t have a buffer as an output to the render pipeline and as an input in the same draw call!
- Many effects are possible!

Further Reading

- Wolff, D, 2013, *OpenGL 4 Shading Language Cookbook*, 2nd Edition, PACKT Publishing