Deferred Rendering

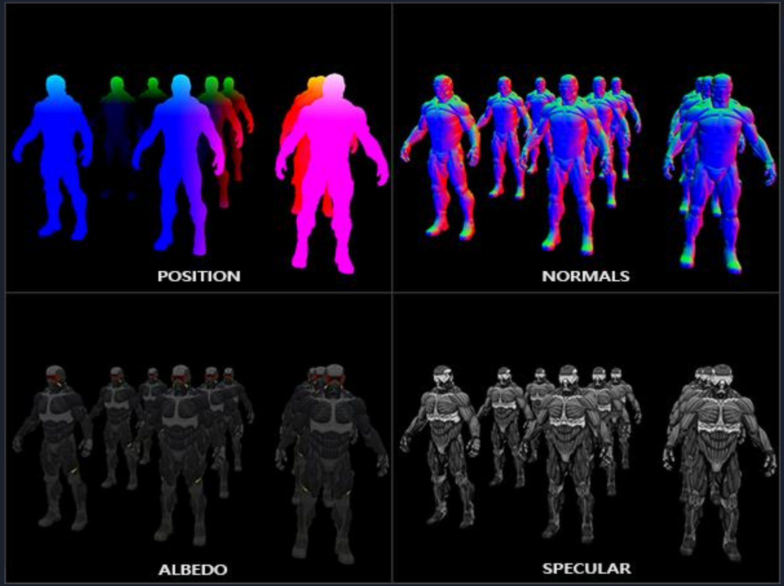
Introduction to screen-space lighting

Programming – Computer Graphics

aie
SPECIALIST EDUCATORS IN
GAMES, ANIMATION & FILM VFX

Contents

- The Problem of Forward Rendering
- Deferred Rendering
  - Steps
  - G-Pass
  - Light Pass
  - Composite Pass
  - Benefits
  - Optimisations



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

The Problem of Forward Rendering

- Forward Rendering is when we draw a mesh to the back-buffer, then the next mesh, and the next, each separate from the other
  - Pixels can be written to multiple times (mesh B renders over the pixels of mesh A, etc)
  - Overdraw
- This means our complex lighting calculations and shaders get wasted
  - Most PC/Console games are Fragment Shader heavy, rather than vertex heavy, so wasting calculations can be costly!
- We also only have access to the pixel currently being rendered
  - We'd have to use Post-Processing to sample nearby fragments, but this would be after lighting has been calculated

Deferred Rendering

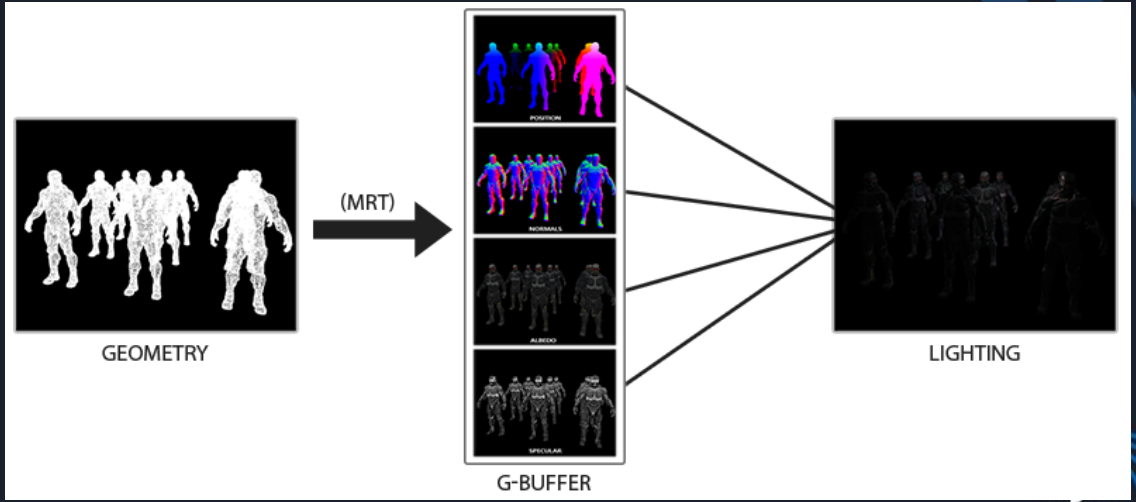
- A popular method of rendering that exists in almost all major PC/Console engines is Deferred Rendering
  - Also called Deferred Shading

- A scene is rendered to off-screen buffers without applying lighting/shading, then the lighting is applied as a Post-Processing step
  - This reduces the lighting calculations by not having to perform them for pixels that would get written over with overdraw
  - Only visible pixels are lit

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Deferred Rendering Steps

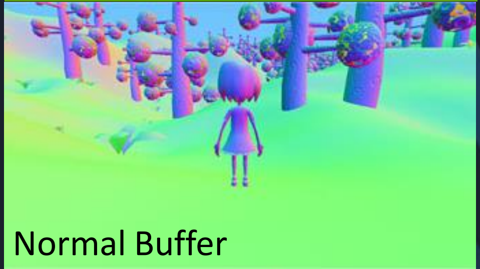
- There are 3 stages to deferred rendering
  - The Geometry Pass or G-Pass
  - The Light Pass
  - The Composite Pass

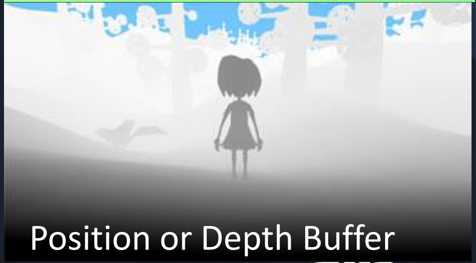


The G-Pass

- This stage is done as a forward pass
  – Each mesh is rendered into off-screen buffers rather than the Back Buffer, called the G-Buffers
- Traditionally renders into buffers that hold
  – Material information, such as unlit diffuse colour of geometry, called the Albedo
  – Positional information of the geometry at each pixel, or depth as distance from the camera
  – Normal information of the geometry at each pixel
- Requires the use of Multiple Render Targets


Albedo Buffer


Normal Buffer


Position or Depth Buffer

The G-Pass

- Once we have rendered the entire scene in to the G-Buffers we can use them as input textures to the Light Pass and Composite Pass

- We can store various data within the G-Buffers but there are a few considerations
  - How many Components / Elements?
  - Data type and size

- Some examples:
  - Using 3 buffers with RGB 32F format (3x 32-bit float values)

Albedo
Red
Green
Blue

Position
X
Y
Z

Normal
X
Y
Z

The Light Pass

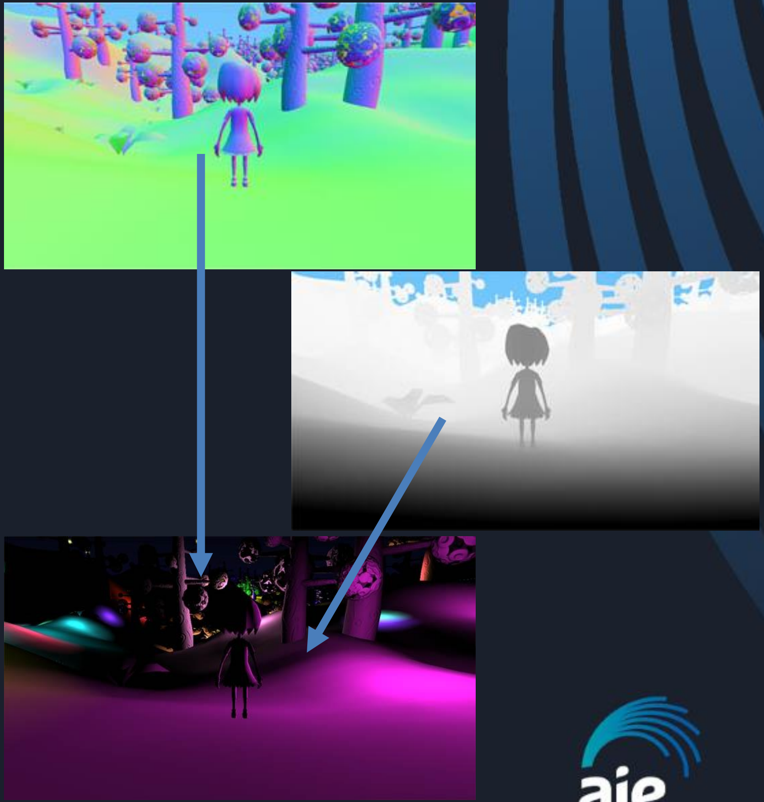
- The next pass in Deferred Rendering is the Light Pass
  - We render in to a new render target, the Light Buffer
  - We use 2 of the render targets from the previous pass as textures; the Position Buffer and Normal Buffer

- We then render all lights as geometry, with additive blending enabled
  - We use geometry that would cover the area of the screen that the light would illuminate
    - Directional lights can be rendered as a full-screen quad
    - Point lights can be rendered as spheres or boxes
    - Spot lights can be render as cones or pyramids
  - The geometry is rendered with a standard Vertex Shader
  - The Fragments that get processed are then fragments which may be illuminated by the light

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

The Light Pass

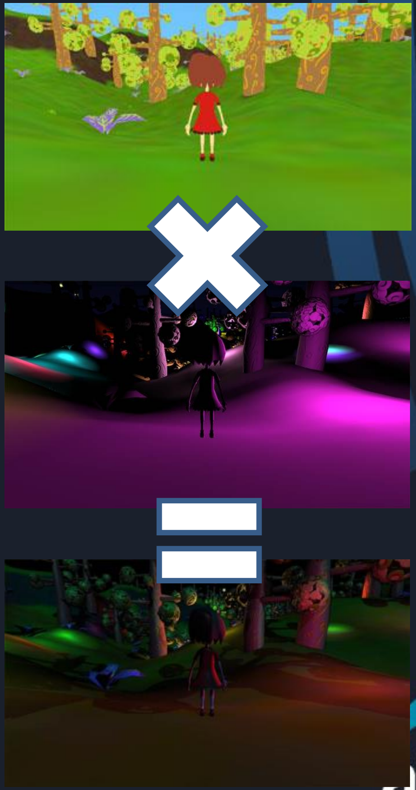
- Within the Fragment Shader for the light geometry we calculate the light for the pixel
  - At each pixel we know its Position and Normal thanks to the G-Pass
    - We simply sample the textures for that pixel
  - We then calculate lighting as usual
    - The Normal comes from the Normal Buffer
    - Position from the Position Buffer can be used for calculating specular and view vectors
- We don’t use the surface colour in the calculation, only the light colour
- We then output the colour with additive blending
  - The more lights shining on a single pixel the brighter it will be
- We end up with a buffer that contains all light for each pixel



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

The Composite Pass

- Once all lights have been rendered we can move on to the Composite Pass
  - This is the final pass and renders typically in to the Back Buffer
  - We use the Albedo Buffer and the Light Buffer from the earlier passes
- This pass is a simple Post-Process pass
  - We render a full-screen quad with the Albedo and Light Buffers applied as textures
  - The Fragment Shader simply multiplies the Albedo with the Light Buffer



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

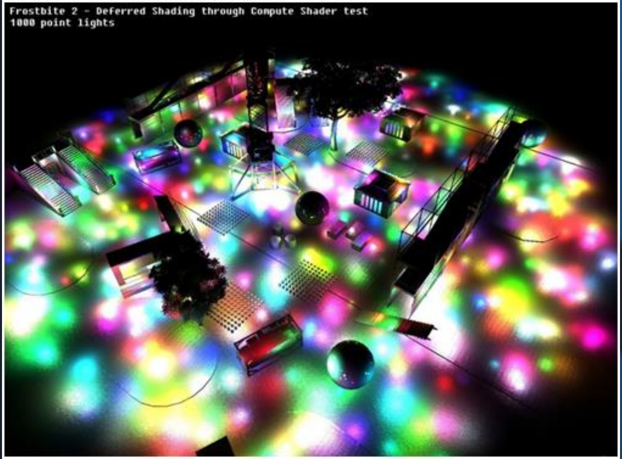
Deferred Rendering Benefits

- The main benefit of Deferred Rendering is the reduction in lighting complexity and ability to have many lights
  - To be able to light a mesh with many lights in Forward Rendering we would have to render the mesh multiple times, or have shaders available for all the different light counts

$$O(meshCount \times lightCount)$$

- With Deferred we can now have many lights as they simply render as geometry
  - Each mesh is rendered, then each light

$$O(meshCount + lightCount)$$



Deferred Rendering Benefits

- This means we can have environments with hundreds or thousands of lights!
  - We also only need one set of lighting shaders, one for each type of light, rather than different shaders depending on the number of active lights in a scene

Deferred Rendering Benefits

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Deferred Rendering Pitfalls

- Transparency is extremely hard to include!
  - Typically transparent objects are rendered afterwards in a forward rendering manner, back to front
  - Modern techniques allow forms of “layers” or pixel linked-lists in deferred rendering for transparent objects, but these can be slow and require high-end hardware

- Memory Footprint Issues
  - Not all systems like Multiple Render Targets
  - High Definition displays require large G-Buffers
    - Around 8mb for a single buffer using 4 bytes, closer to 32mb for a vec4-based buffer!
    - Video card memory can disappear fast

- All meshes use the same material!

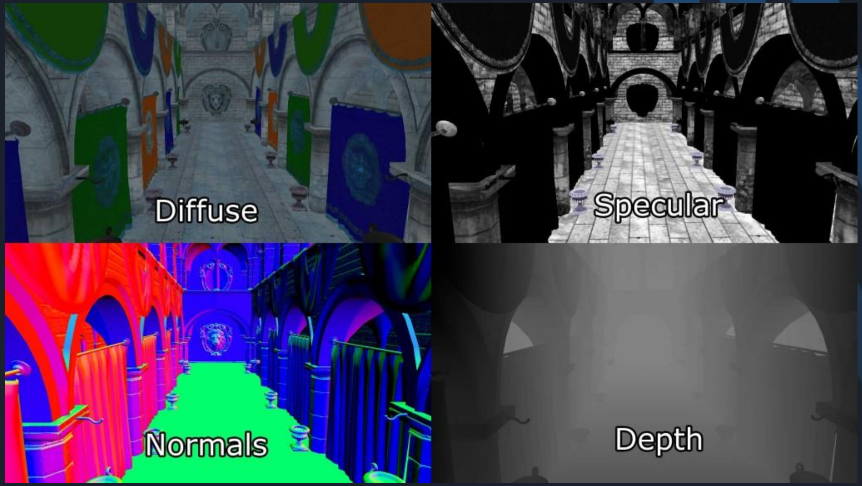
Deferred Rendering Optimisations

- We can reduce the size of the data stored in the g-buffer
  - Normals usually take 3 elements, XYZ, and are best stored as floats for accuracy
  - Position also would typically use 3 elements for XYZ

- Rather than position we could store depth as a single float
  - Using depth we could recreate the position by projecting the 2D screen-coordinate back into 3D at the required depth
- Rather than XYZ for the normal we could store 2 floats!
  - Early implementations ignored Z with the assumption that in view-space it always faced the camera and so was positive, and could then be recreated
  - However this is incorrect!
  - Newer implementations convert the normal to other formats, such as 2 polar angles, or to sphere-map coordinates with Azimuthal Projections

Summary

- Deferred Rendering allows for many lights in a scene
  - Thousands!
- Performance issues must be taken into account, but optimisations can reduce them substantially
- Majority of AAA developers use deferred techniques, as do majority of commercial game engines
  - Unreal, Unity, Source, Frostbite, PhyreEngine, CryEngine



Further Reading

- Wolff, D, 2013, *OpenGL 4 Shading Language Cookbook*, 2nd Edition, Chapter 5, PACKT Publishing

- Haemel, N, Sellers, G & Wright, R, 2014, *OpenGL SuperBible*, 6th Edition, Chapter 12, Addison Wesley