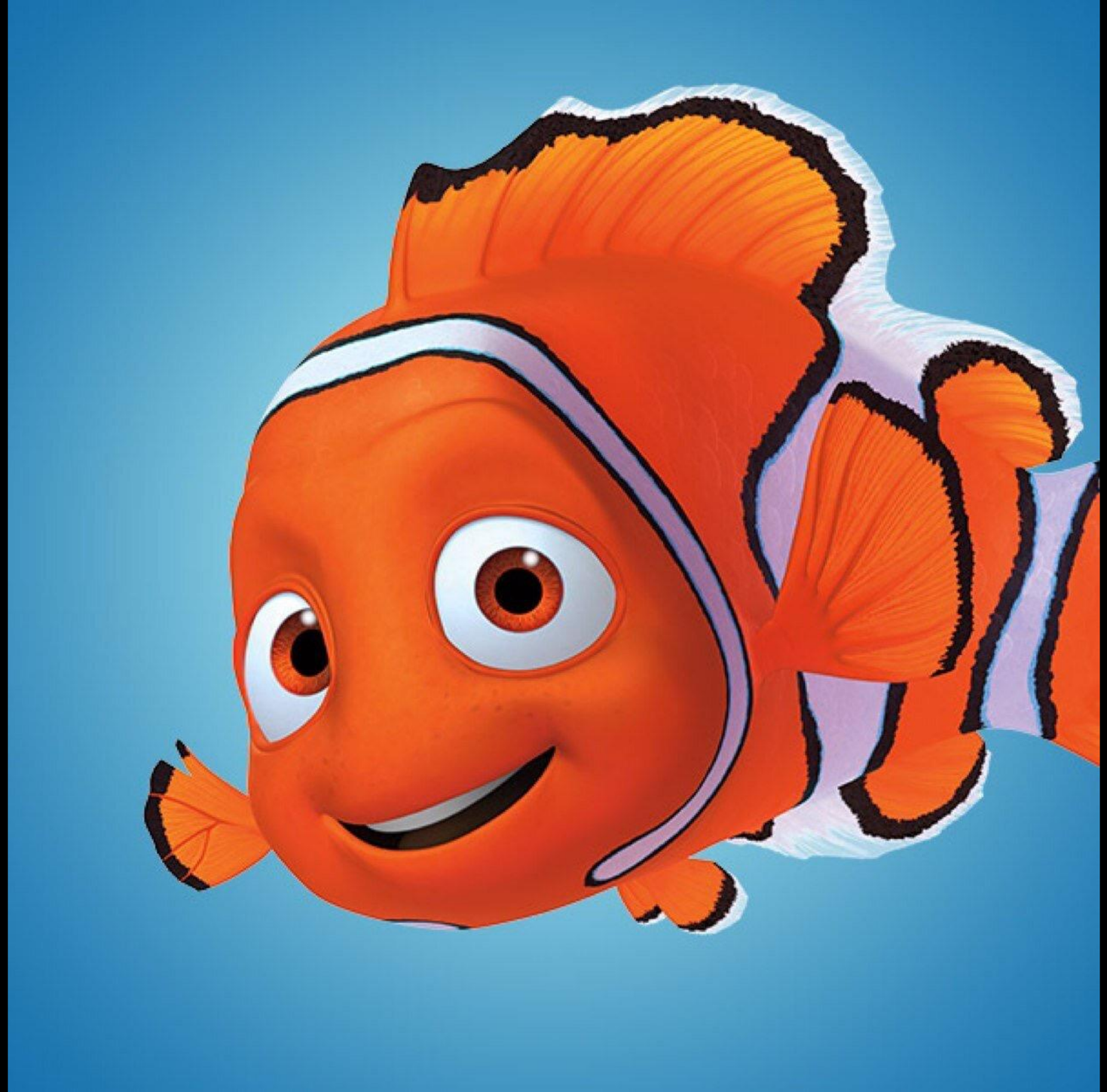
Subsurface Scattering in the Unreal Forward Renderer

David Wilkinson / AMD
Tom Sanocki / Limitless Ltd


diffuse
subsurface + specular
combined illumination




A4351
A4351

Moving Forward

An Optimized Path For VR Rendering

David Wilkinson

Developer Technology Engineer, AMD

Introduction

- Overview of Forward Rendering
- Pros and Cons
- Feature Support
- Example: Subsurface Rendering

Forward Rendering Overview

- Scenes are rendered with multiple lights for each prim
  - Lighting and shading happen together during rasterization
  - Legacy forward renderers were inefficient with large # lights
- Only a subset of lights are selected
  - Using Compute shaders
  - Lights and reflections are tiled and culled into frustum-spaced grid
  - Only lights influencing a local grid tile are considered
  - Only top ‘m’ contributors considered during rendering
- 1000s of lights per scene no problem

Forward Rendering Overview

- Rendering performance is significantly increased
  - Around 20% increase for base pass (UE4)
  - Average 25-30% increase for total frame time (on UE4)
- Overdraw is avoided by using a depth pre-pass
- Hardware Anti-aliasing (MSAA) can be enabled
- Translucency just works

Forward Rendering – Pros and Cons

Pros

- MSAA
- Complex Materials
- Bandwidth friendly
- Translucency just works
- Features can be enabled per-material

Cons

- No screen-space operations
  - SSR, SSAO, Contact Shadows, IES, Subsurface Profiles.
- GPU Occupancy suffers
  - Tiny triangles
  - VGPR Usage

Why Are Some Features Unsupported?

- Maturity of Implementation
  - Not a trivial amount of work, porting each feature takes time
- Technical Complexity
  - Some features may not have a viable equivalent forward implementation
    - May require hybrid solution (forward + mini-gbuffer)
    - Or, looking forward, a Texel Shader / Object Space approach

Working Around Unsupported Features

- Screen Space Alternatives
  - SSR -> Planar Reflections and Reflection Captures
  - SSAO -> AO using depth pre-pass buffer
  - SSSS -> Texture Space Diffusion Or Diffuse Wrap w/ textures
- Use Hybrid Rendering
  - Use the full screen depth-pass to perform deferred rendering passes
  - Keep Gbuffers small – 1-2 packed 64-bit params
  - Do not adopt deferred bottlenecks!

Working Around Unsupported Features

- Look for forward based substitutes
  - Revisit older algorithms that may not have been previously feasible for RTR
- See what other studios are doing
  - ‘RoboRecall’ by Epic is a great example of what can be done in the Forward path
  - Check the forums, others are likely tackling similar issues
  - Definitely approach IHVs and engine developers, there may be a solution in the works…
  - Plus, it assists them in evaluating and prioritizing in-demand features

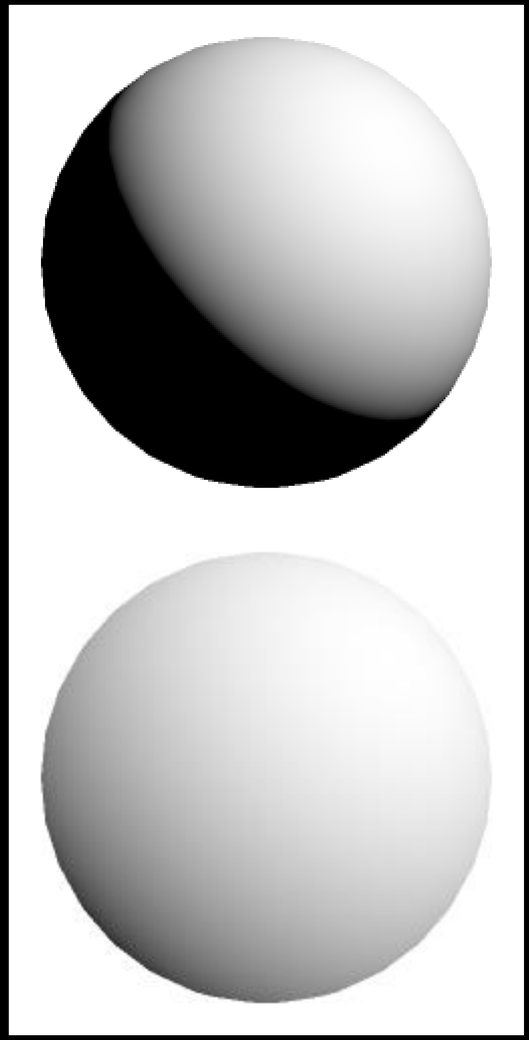
Example: Subsurface Rendering

Skin Rendering Example

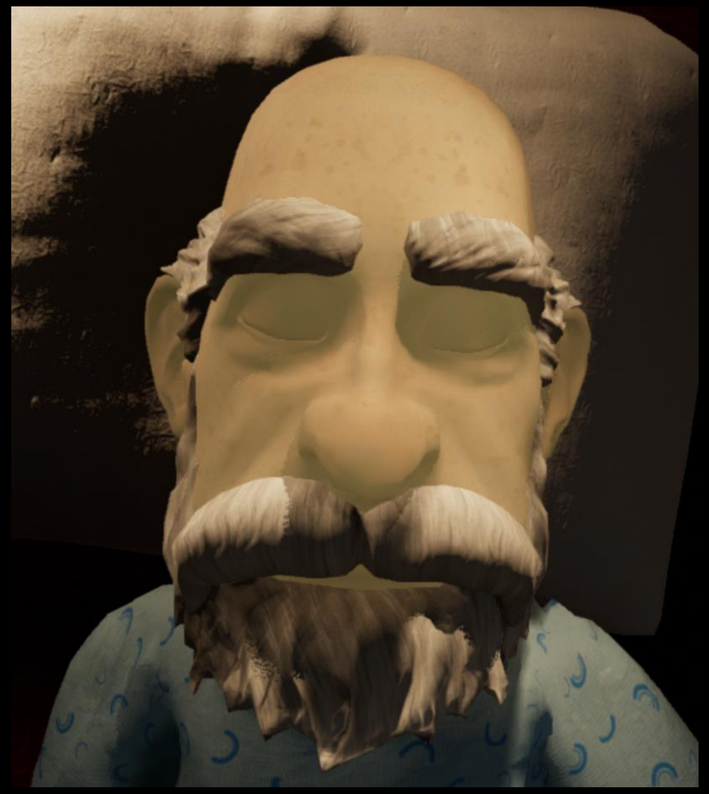
- ‘Reaping Rewards’ contains many assets that feature skin, cloth, hair
  - In UE4, these could be implemented with the skin subsurface profile
  - However, this is a screen space technique – not available in the forward path
- Limitless is currently experimenting with two alternative techniques
  - Diffuse wrap shaders w/ textures
  - Hybrid mini-Gbuffer pass for subsurface scattering
- The wrap shader will be shown here
- Can be implemented in UE4 material editor, no engine changes

Diffuse Wrap

- Simulates subsurface scattering
  - A key part of skin rendering
  - Models the translucent nature of skin
  - Light enters, and bounces numerous times
  - Then exits at a different place
- Extends diffuse lighting to wrap around an object
  - Gives a translucent appearance
  - Forms the base layer of the skin surface
- Can be implemented as a Material shader in UE4
  - Create a material and set shader model to:
  - Two sided foliage
  - Subsurface
  - Pre-integrated skin

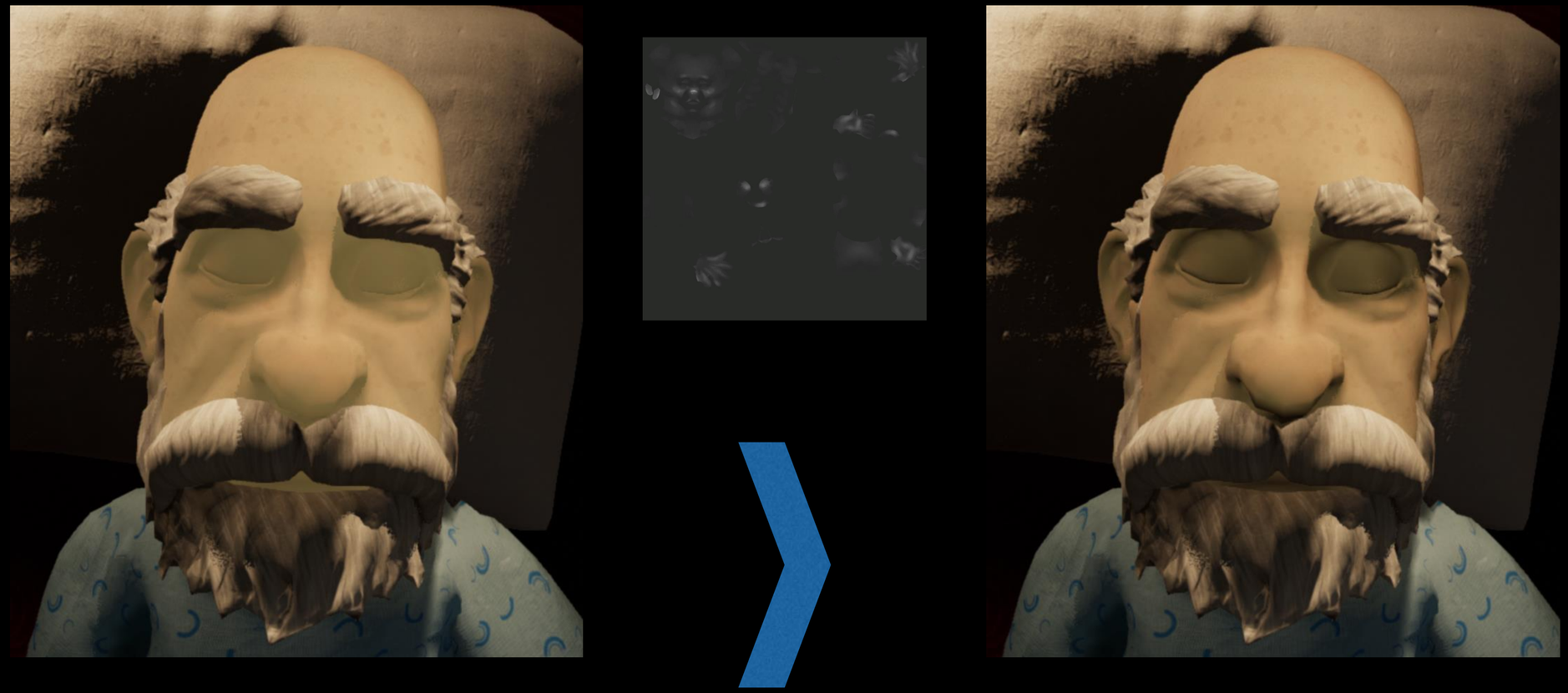


Diffuse Wrap

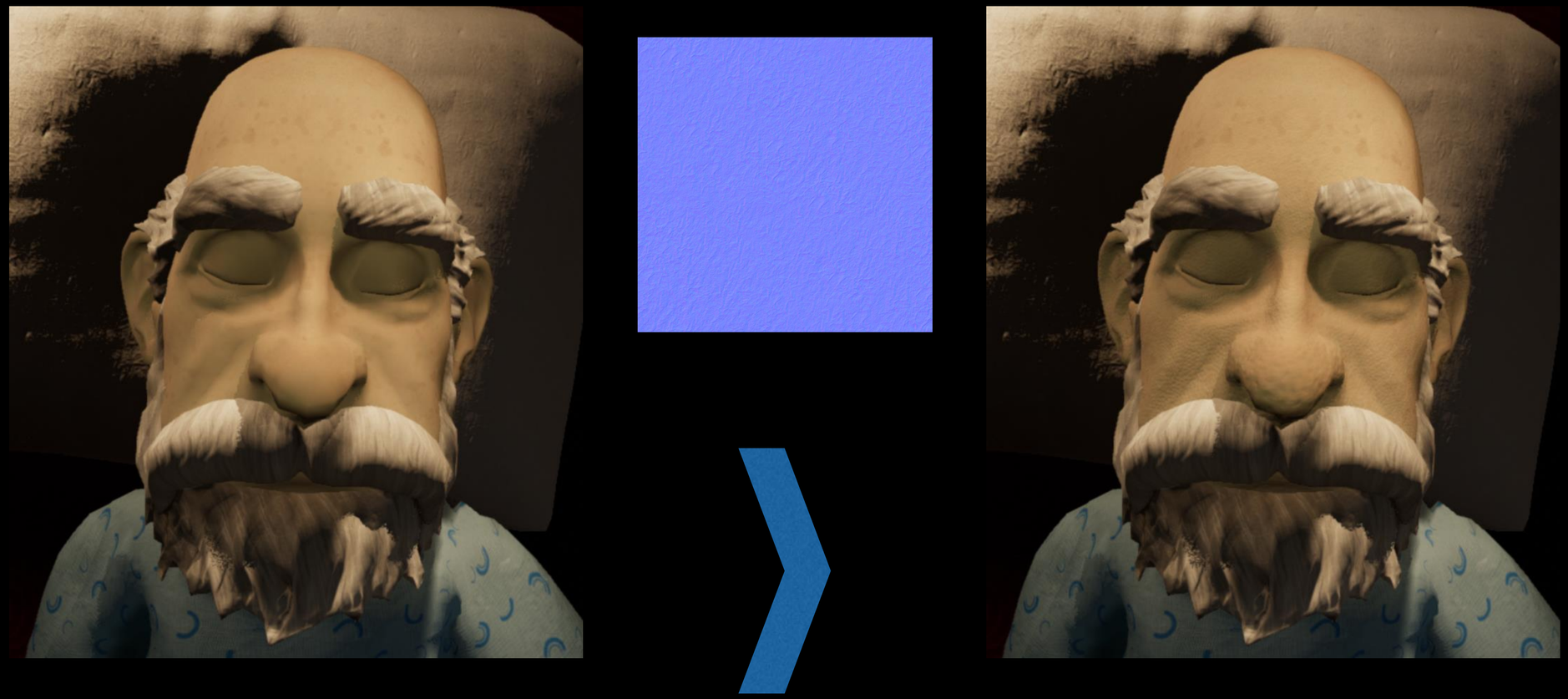


Hm. Needs less cowbell.
Let’s texture some layers…

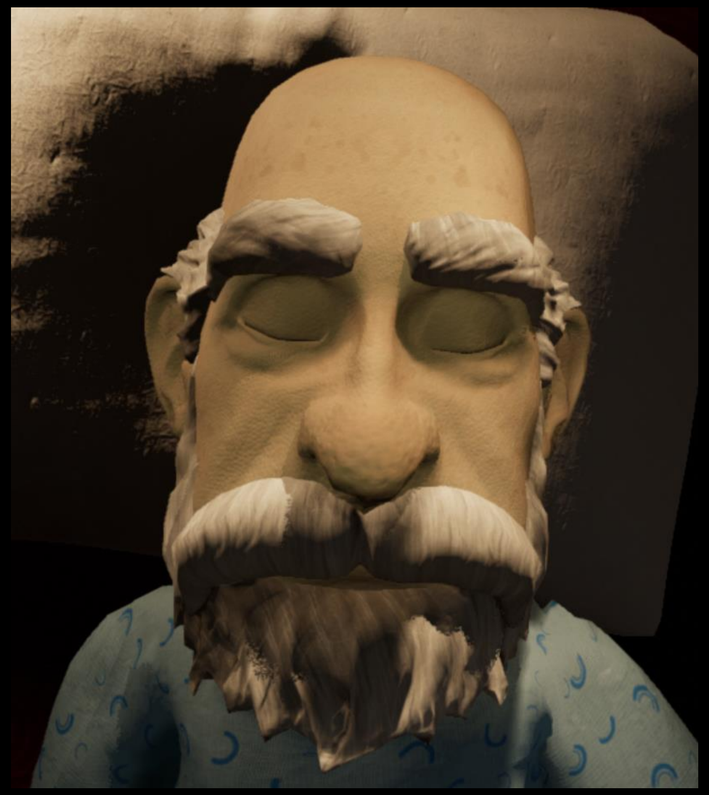
ScatterMap

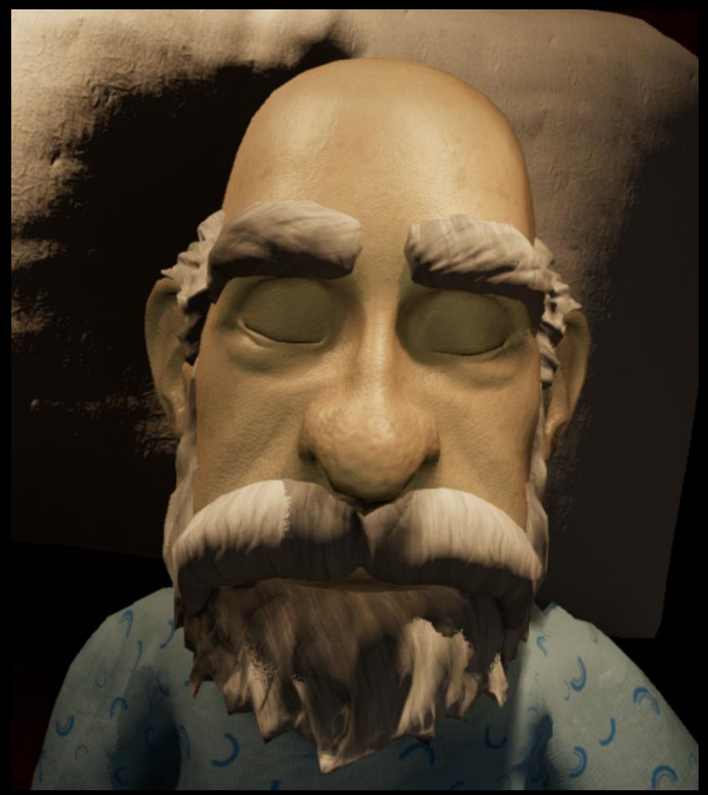


Roughness

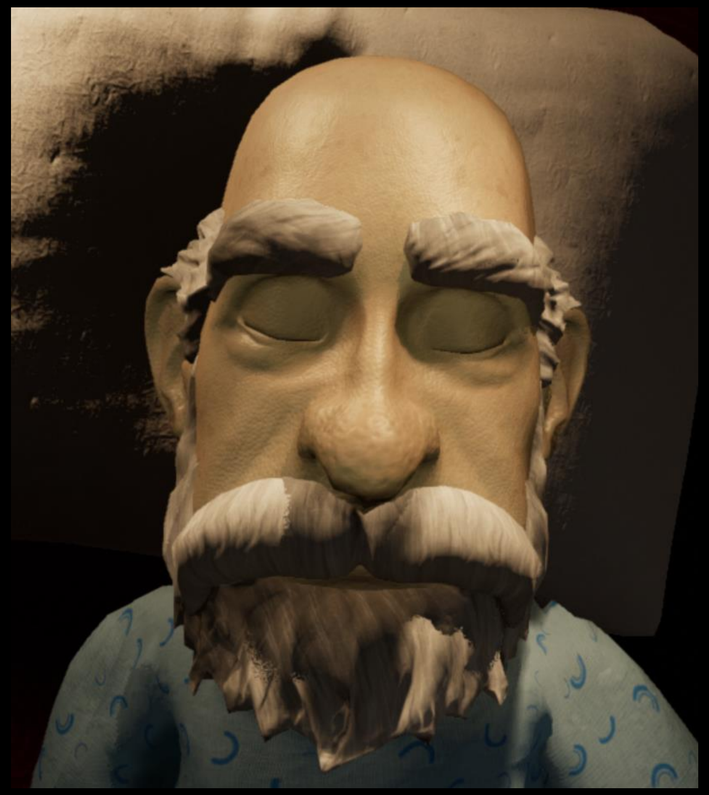


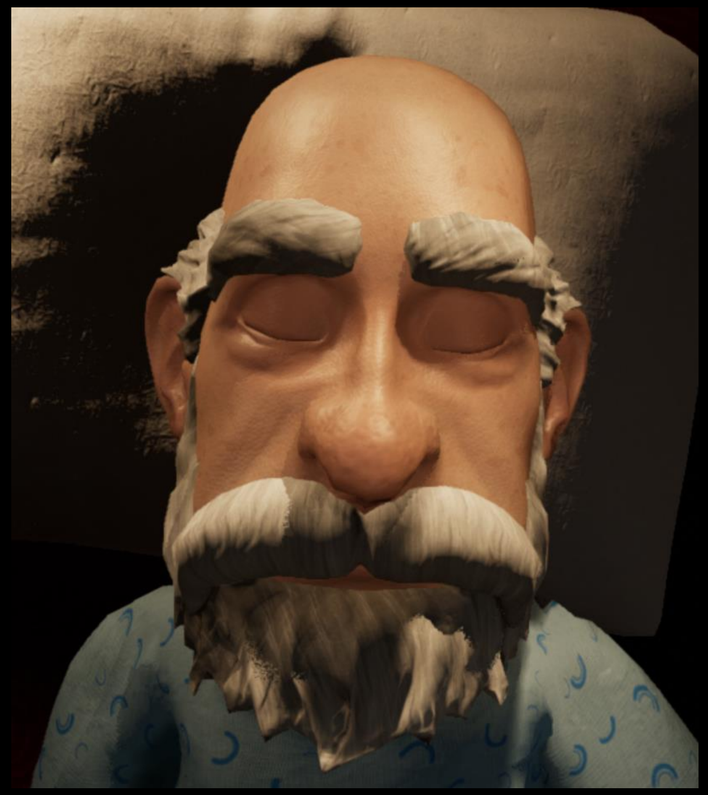
Sheen



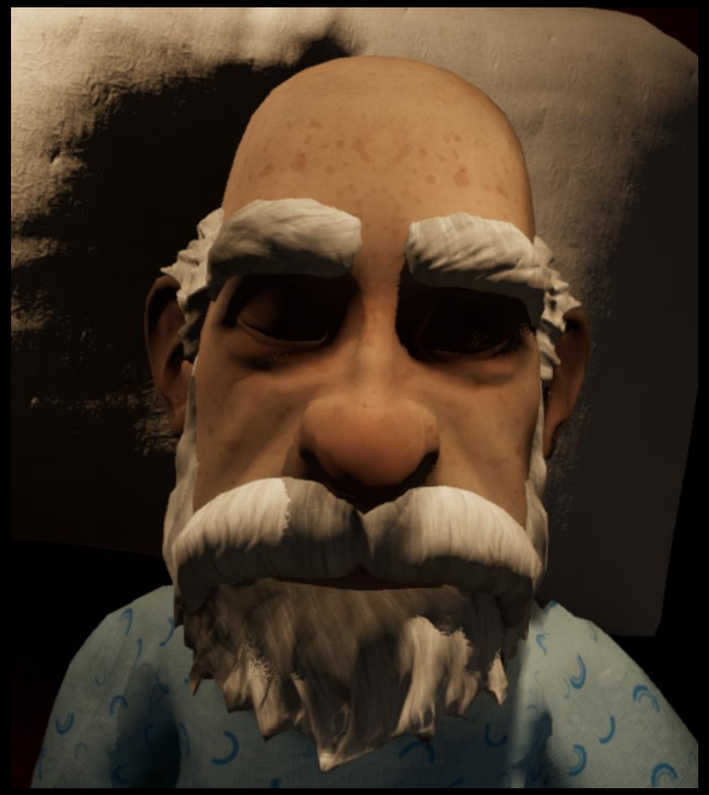


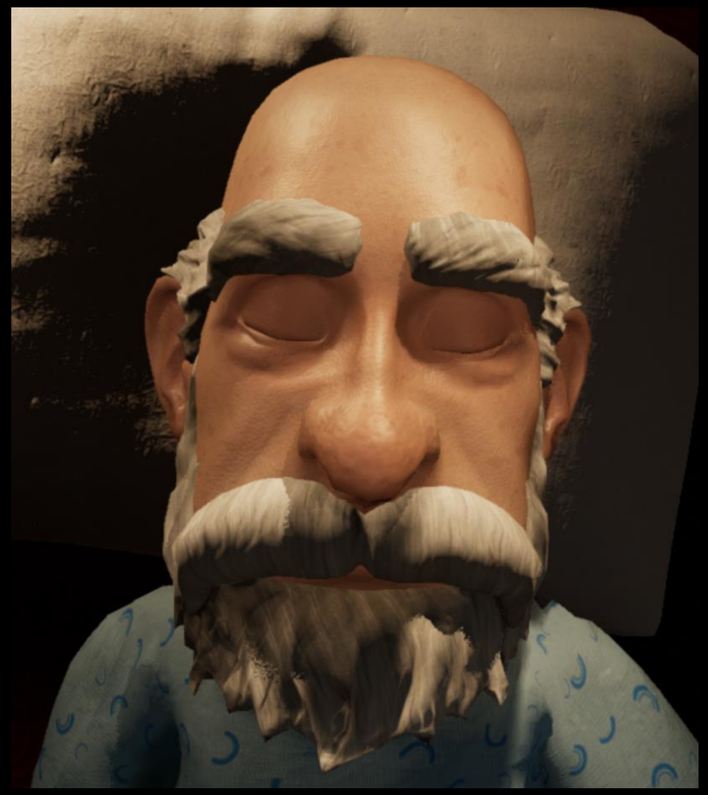
Subsurface Color





Final Result





Thank You