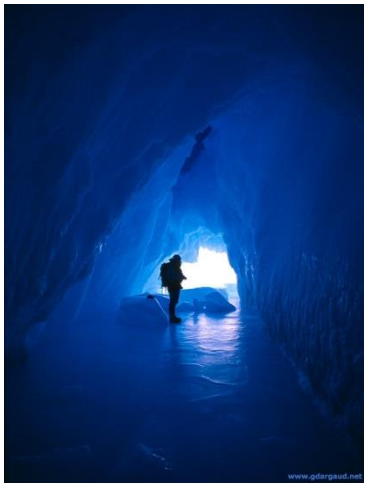
Ice Shader R&D in UDK
I3 DLC

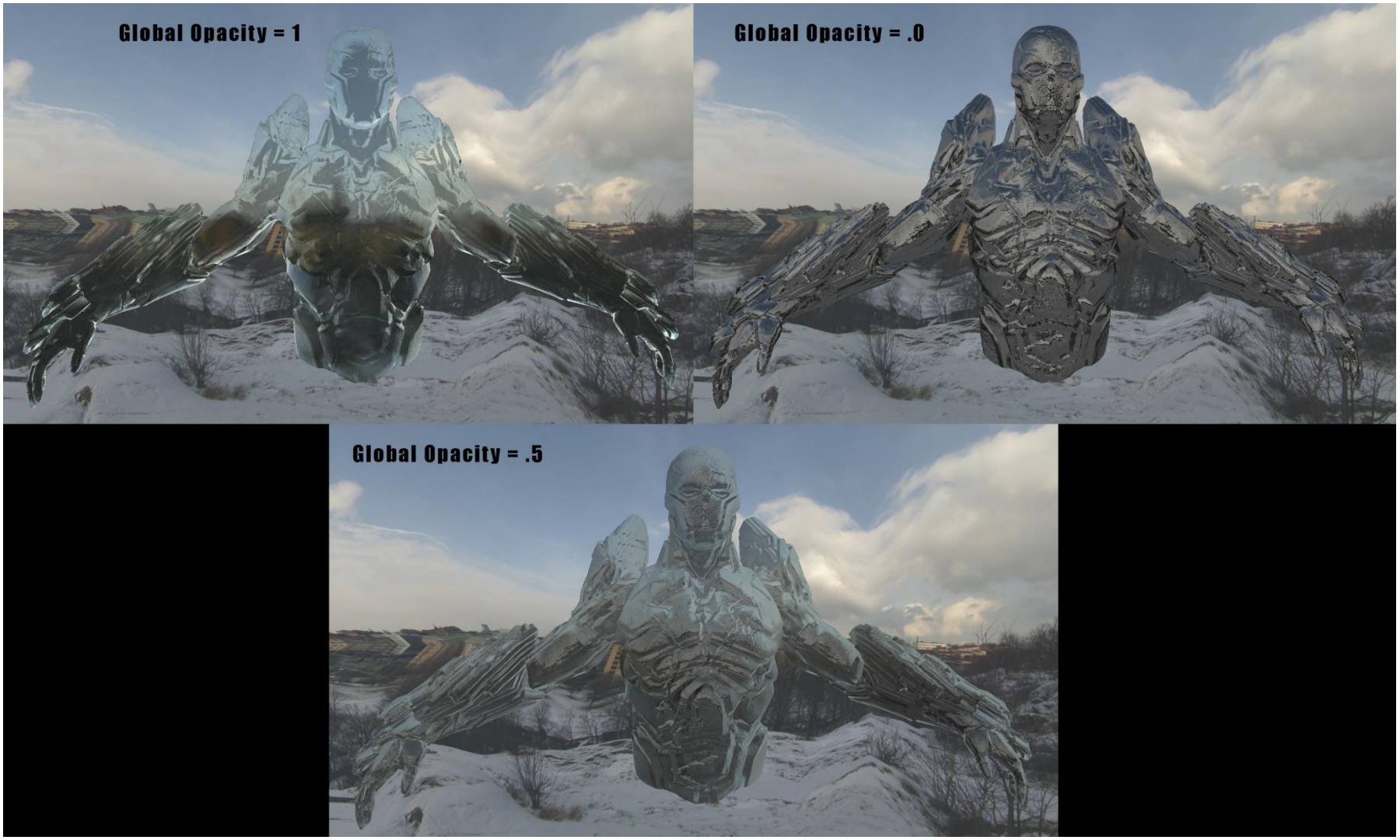
Has to fit many different uses for ice



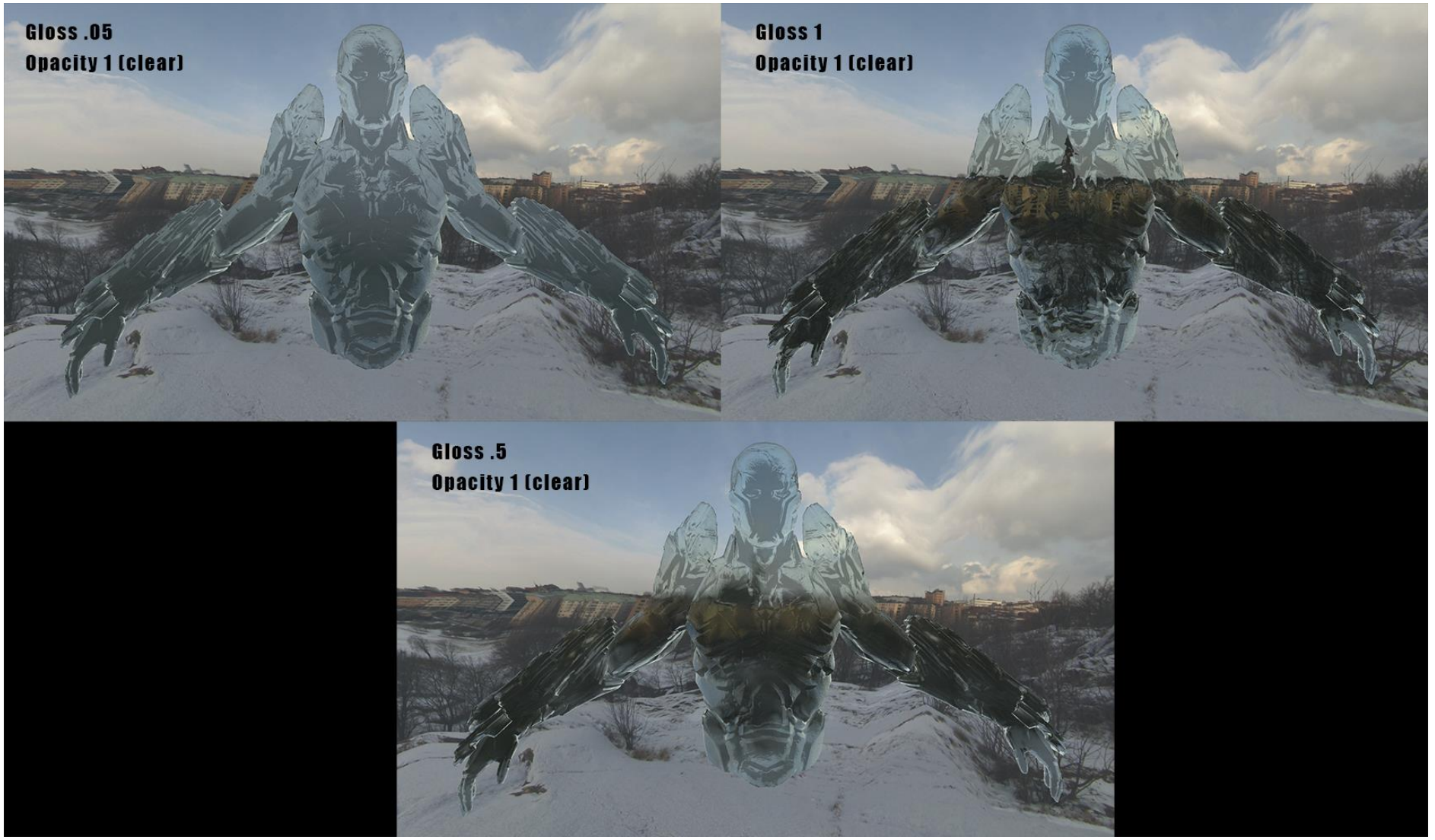
Key Features/Requirements

- All done with an Opaque shader (all translucency faked)
- Parallax support for extra depth
- Ice cracks that bleed light “into” the surface
- Fake refractions built into the fake translucency
- Fake “coreness” to give “translucent” ice depth
- All encompassing solution for Ice (walls, frozen water, surfaces, etc)
- Fake SSS (Dice approximate translucency solution for SSS)

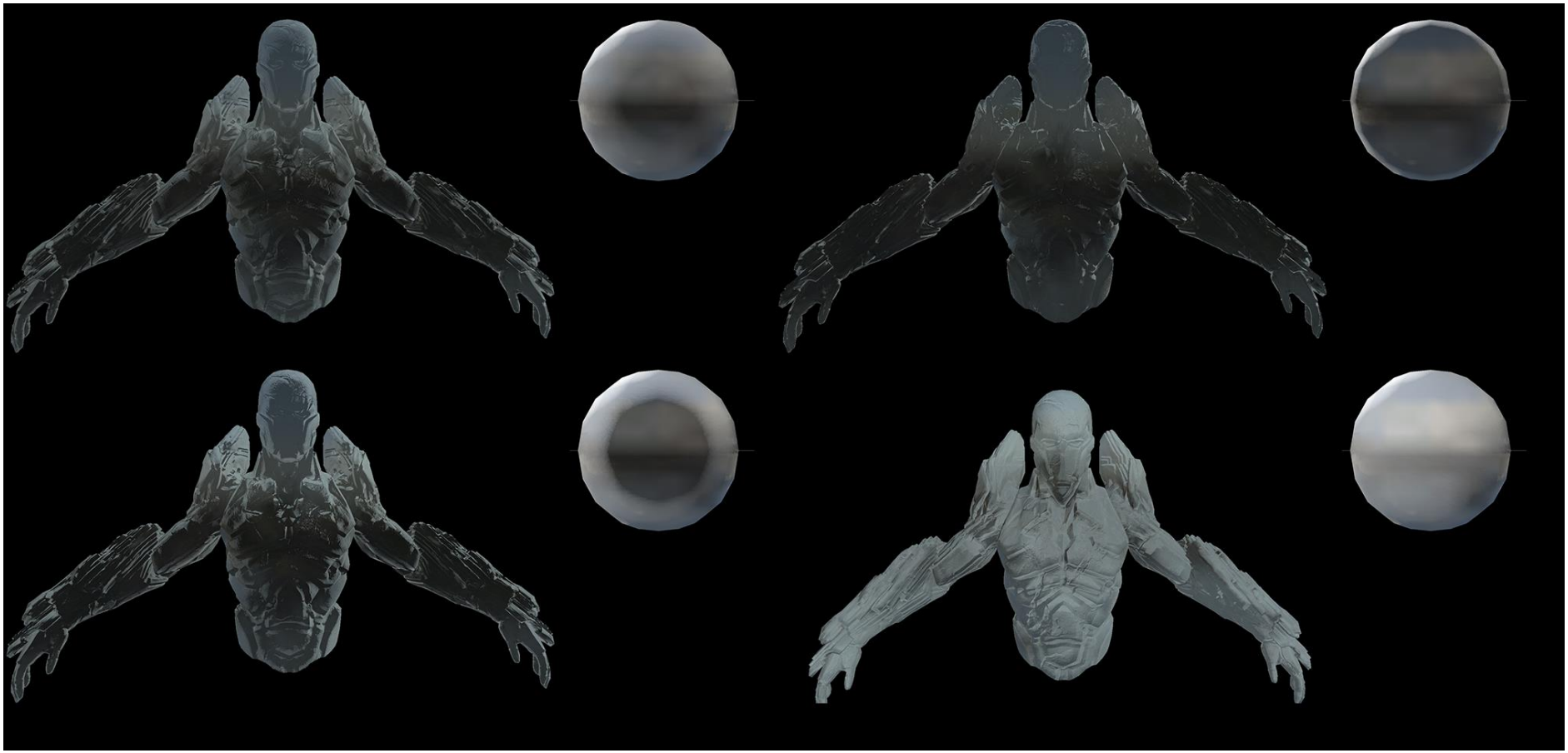
Fake Translucency



- Why do it?
  - Performance
  - True translucency is rarely required
  - Allows for more accurate ice rendering without the cost of translucency
- What did I do?
  - I used a control variable (opacity) to lerp between different features that are obtained when surfaces become see through.
  - As the opacity value shifts to 1 (see through) Light absorbed and reflected (environment mapping and diffuse/specular) gets replaced with light “behind” the object (refractions, environment map with reflection vector replaced with inverted camera vector)
  - Refractions were achieved with the use of the refract function, accepting realistic indexes of refraction
  - The refractions and “see through” or “normal view” environment map were lerped between each other based off of a Fresnel using Schlick's approximation
  - Gloss blurs the “normal view” image by mipping down the cubemap.
- What should/can we do/takeaway?
  - With our spec probe system, there could be really large potential for the “faked translucency” to be accurate
  - Per shader opacity controls lets artists tweak for different types of ice, helping the “one size fits all” problem
  - If we really have no use for translucent ice (icicles, etc) this entire aspect and the SSS could be cut
  - Refraction/opacity global controls should be relatively easy to hook up to a physically plausible BRDF system
  - Having that opacity control is quite important. As the last slide showed, having a balance of translucency and appropriate reflections really matters on the selling of the “ice” material
  - Having the fake “normal view” through the object be a cubemap means we can mip it down for a “frosted glass look” (see images on next slide)


Gloss .05
Opacity 1 (clear)
Gloss 1
Opacity 1 (clear)
Gloss .5
Opacity 1 (clear)

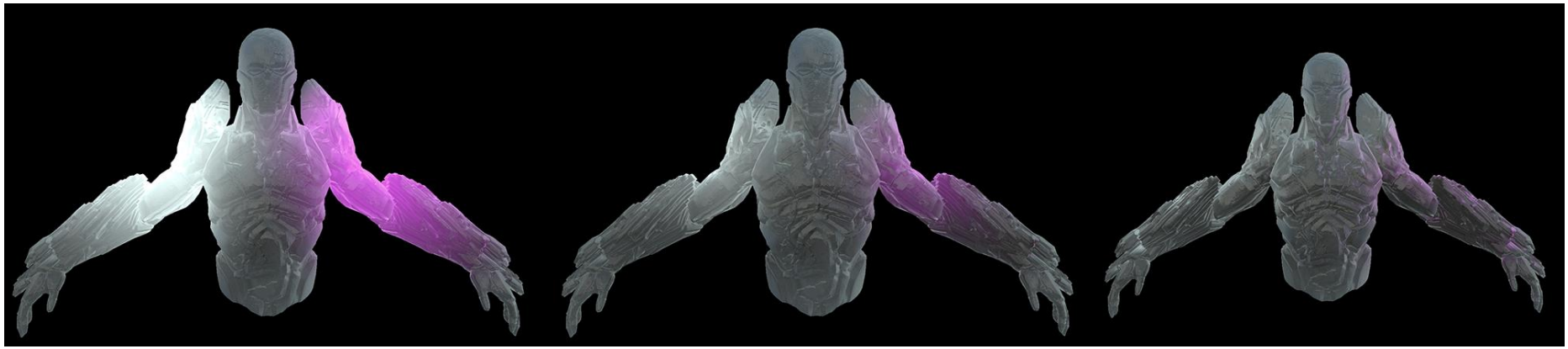
Fake “Coreness”



- Why do it?
  - Much cheaper than true calculations (ray traces, etc)
  - Helps give volume and depth to the ice
  - Easily artist controlled
- What did I do?
  - I took the dot product of the camera vector and an adjusted normal vector and used that to darken certain parts of the shader
  - Originally I had artist control over how much it affected every single aspect (refraction, reflection, diffuse, etc) but eventually locked all the values down and only left exponent and strength controls available.
  - The variables for control are still present for tuning, just not in the final material controls
- What should/can we do/takeaway?
  - Very powerful addition to the final image
  - Not necessary for truly opaque surfaces, could be removed as the object becomes more “translucent”
  - Could save on instruction cost by inverting the fresnel calculated for the schlicks approximation with the refractions

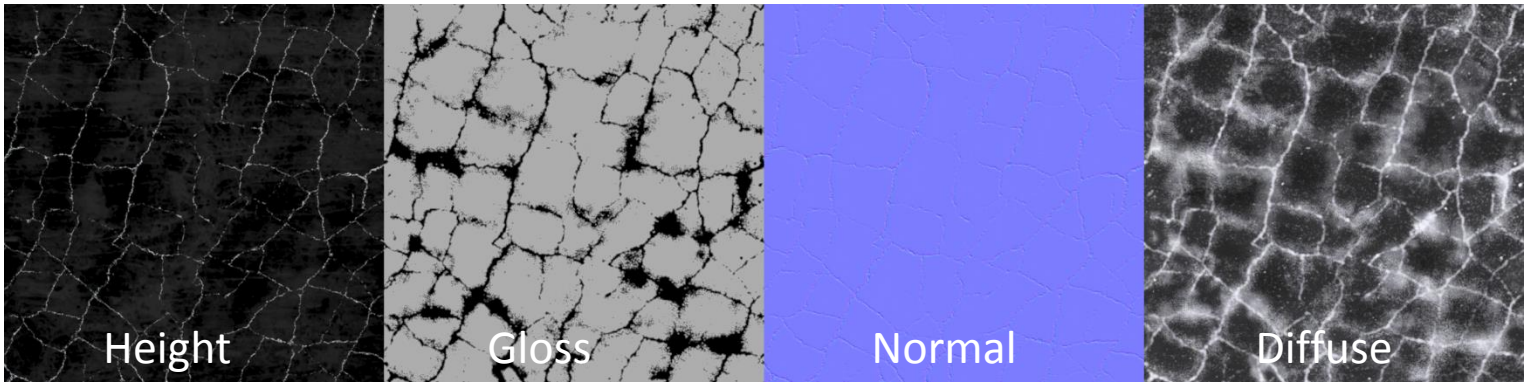
Fake “Sub surface Scattering”

- Why do it?
  - Looks great for solid volumes of ice
  - Not insanely expensive
- What did I do?
  - Implemented and optimized Dice's approximate sub surface scattering Link here
  - Only left the subsurface Scale and exponent control available to the artists. Those could potentially be removed
  - Removed the Thickness mask. Having the fake "coreness" affect the SSS compensates for this nicely, saves texture memory, and works well for our environment texture style
- What should/can we do/takeaway?
  - For selling cool translucent materials, this is killer, could even be potentially used elsewhere (skin shaders, etc)
  - Consider removing all artist control and link the scale and power to the opacity controls making the shader even simpler.
  - On the flip side, with no thickness map, having artist control of the strength would help sell individual materials
  - Since the extra light is additive on top of the spec/diffuse model I would imagine it wouldn't be too difficult to add to our current system.
  - Unsure how light transmitted fits into energy conserving models (will look it up)



Cracks with Light Bleeding

- Why do it?
  - Cracks in ice may be needed for gameplay (can confirm if required)
  - One of the best ways to break up ice (which it sounds like there will be lots of it
  - Looks really good, potentially not a huge extra cost
- What did I do?
  - Sampled an Ice height texture down by offsetting the UV's by the camera vector, mipping down at each step, and lowering the intensity of each step
  - Used the diffuse texture to lerp between showing the normal diffuse/norm/spec/gloss material, and what was "underneath" the ice
  - Ended up getting good results between 12-20 samples. I used a for loop for iterative testing, but if done by hand, we could probably get better results for less samples
- What should/can we do/takeaway?
  - I recommend custom passes for each sample. That way we can focus more on optimizing by mipping each sample by hand
  - I found that more intense crack textures (solid, softer lines) make for a clearer read very high fidelity noisy masks. This means lower resolution crack textures, and potentially thicker lines mean less samples
  - The lerp using the diffuse was super important. This makes sure the light bleed from the cracks appears at the right location
  - A cool accidental feature, was adding cloudy bits to the crack height texture, but not to the diffuse for the cracks. This allowed for some cool 3d blob shapes to be formed within the ice to add to the milky thickness



Parallaxed background texture

- Why do it?
  - Adds depth to solid ice
  - Can give the illusion of Ice “on top” of another material (rock, snow, etc)
- What did I do?
  - Offset an Ice height texture down by offsetting the UV’s by the camera vector
  - Gave artists control of the texture, the height, UV ration, UV scale, and tinting (for texture re-use)
  - I colored the texture based off of the core Ice texture before using it as a base to build up the rest of the image
- What should/can we do/takeaway?
  - Artist control of the texture/height is the most important aspect
  - Some variation control could be useful (separate tiling clouds, etc)
  - The order in which this part of the shader fits into the rest made a big difference. It was important that this texture is multiplied by the base core Ice color (float3) then has cracks, etc added up on top of it. I would expect the base Ice texture to be pretty dark (more on that in the coming slides

Artist Inputs:

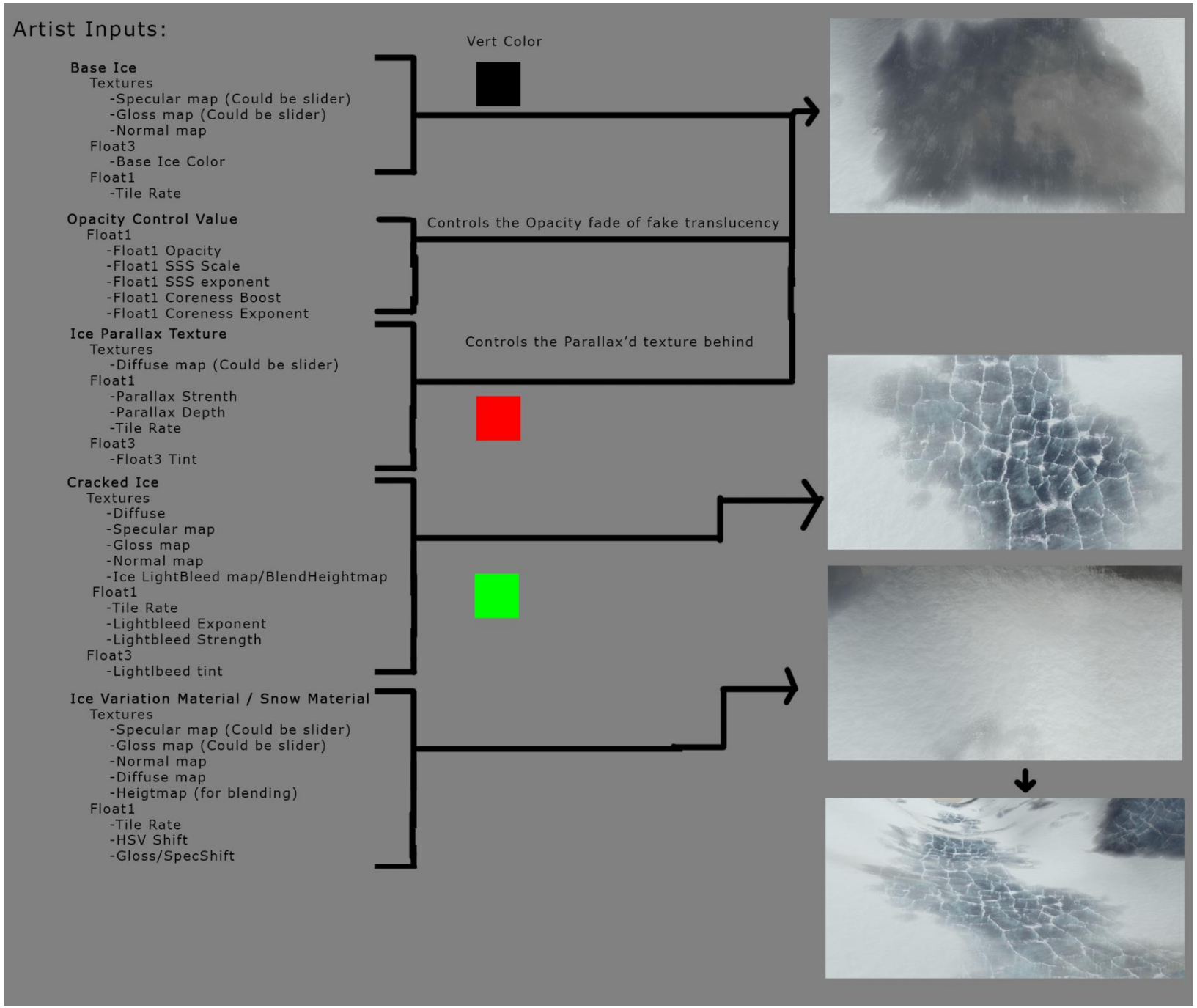
Base Ice
- Textures
  - -Specular map (Could be slider)
  - -Gloss map (Could be slider)
  - -Normal map
- Float3
  - -Base Ice Color
- Float1
  - -Tile Rate

Opacity Control Value
- Float1
  - -Float1 Opacity
  - -Float1 SSS Scale
  - -Float1 SSS exponent
  - -Float1 Coreness Boost
  - -Float1 Coreness Exponent

Ice Parallax Texture
- Textures
  - -Diffuse map (Could be slider)
- Float1
  - -Parallax Strenth
  - -Parallax Depth
  - -Tile Rate
- Float3
  - -Float3 Tint

Cracked Ice
- Textures
  - -Diffuse
  - -Specular map
  - -Gloss map
  - -Normal map
  - -Ice LightBleed map/BlendHeightmap
- Float1
  - -Tile Rate
  - -Lightbleed Exponent
  - -Lightbleed Strength
- Float3
  - -Lightlbeed tint

Ice Variation Material / Snow Material
- Textures
  - -Specular map (Could be slider)
  - -Gloss map (Could be slider)
  - -Normal map
  - -Diffuse map
  - -Heigtmap (for blending)
- Float1
  - -Tile Rate
  - -HSV Shift
  - -Gloss/SpecShift



Final Thoughts:

- The opacity controls and how they would fit into the entire blend shader are a bit of a complex thing. I think from a usability standpoint, unless we know we are going to have lots of translucency, icicles, frozen waterfalls, that most of that functionality could be put in a completely separate shader. The parallax stuff, the cracks, etc all would still work in a blend shader though
- In my outline, I had the red and green taken for different blending. Having the blue channel for an extra one while sacrificing wetness, would mean that we can have overall less shader in an environment because the ice shader supports one more material, making it way easier to blend, re-use, etc.
- The overall layering of the shader proved to be the hardest part. Using additive blending for the “inner Ice” materials was pretty important, as was using the lerp of the crack diffuse to cover up what was underneath. That to me, was when things truly felt like a 3D material.
- Run the UDK in dx11 for my shader and map to work
- There are 2 instances in the test scene group within my UPK. The first contains the material tweaked for the floor, the second is tweaked for the ice statue