Shading and Shadows

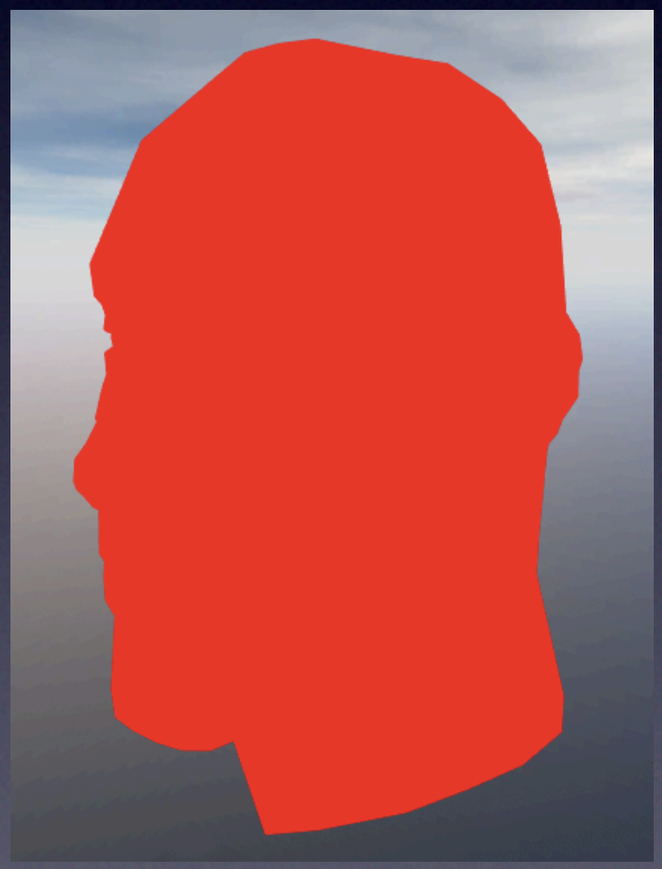
Oh my!

Shader what?

- What is that shader thing anyway?
- What does it do?
- Is it healthy for me?

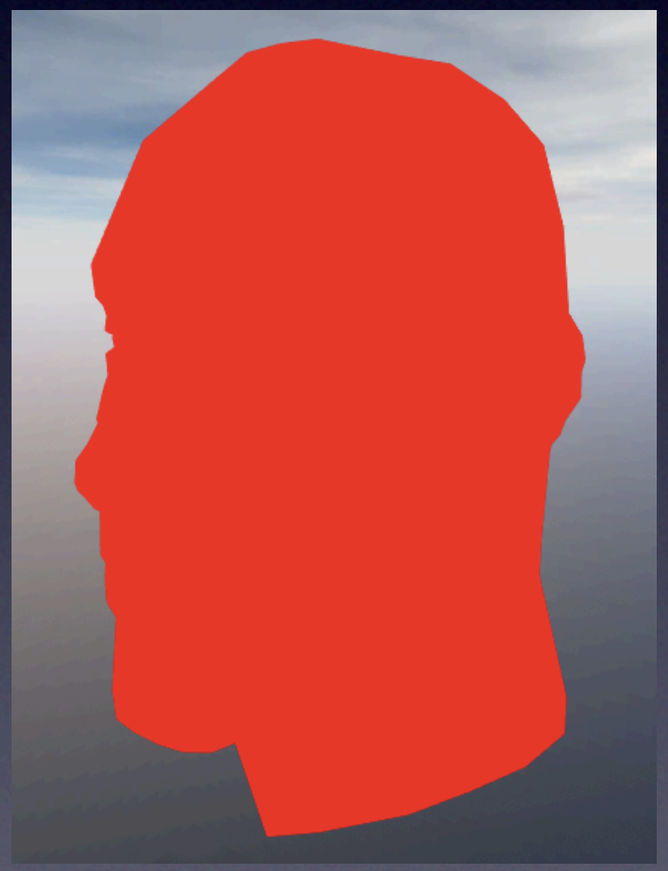
Are these shaders?

- Which one of those uses shaders?



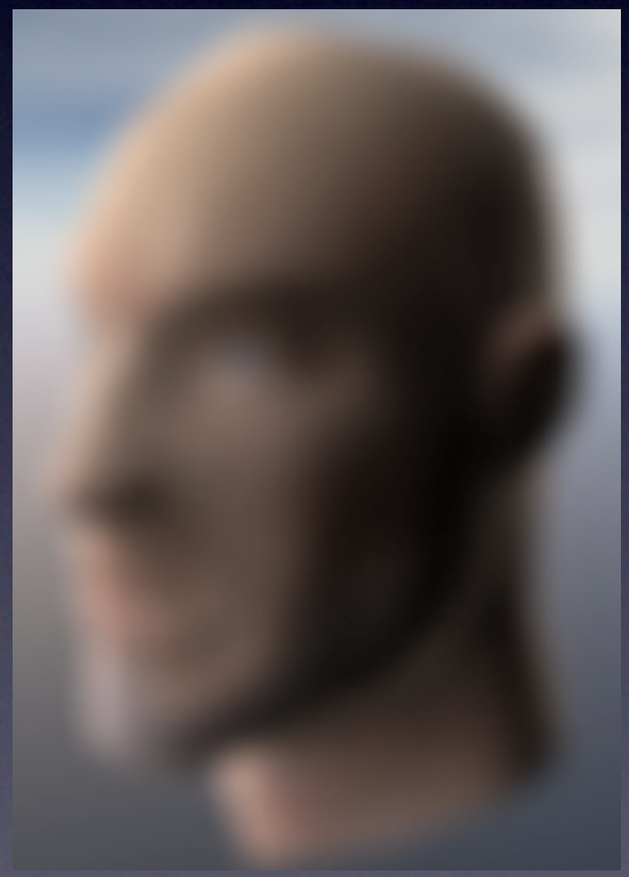
Are these shaders?

- Both! A next-gen red shader and a skin shader.



Are these shaders?

- A blur shader? Nope.



Shaders are nothing

- A shader does not actually do fancy things
- Vertex shader: vertex data in, vertex data out
- Pixel shader: pixel data & textures in, color out
- A shader knows nothing else!

Shaders are nothing

- Shader operates on a single vertex or single pixel only
- It can multiply two textures together
- It can’t blur the screen
  - Blur does not happen on single pixel!

Who does blur?

- Interesting effects are more than shaders
- Shaders + data + scripts + lights

Feeding the shader

- Often needs custom data in textures
  - Bumpmaps, glossmaps, AO, whatnot
- Data in vertices (tangents, colors, ...)
- Scripts
  - material.SetTexture, SetMatrix, ...
- Render Textures

Blur

- Render scene into Render Texture
- Blur a bit into another Render Texture
  - For each pixel: take average of couple neighbors
- Repeat

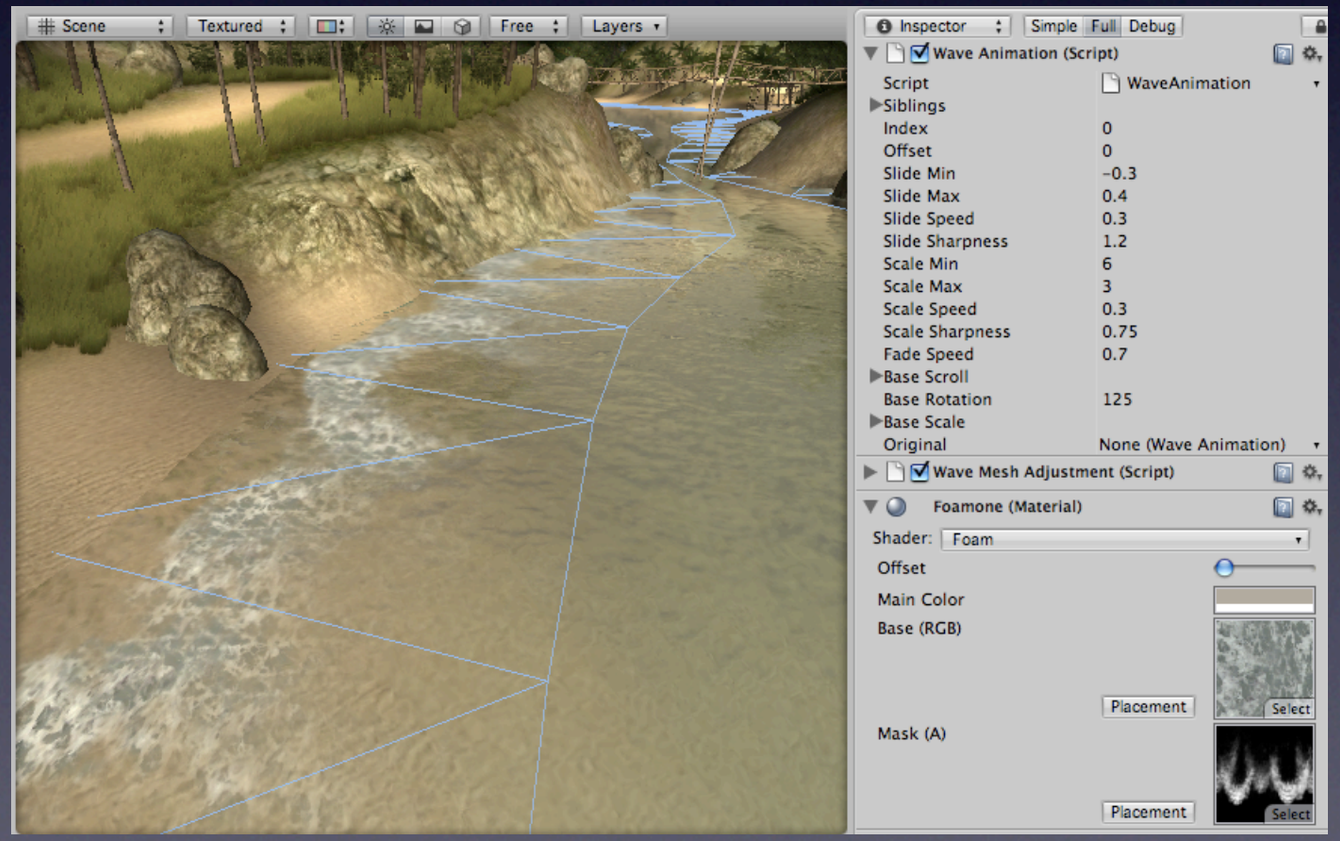
Examples

Water in island demo

- How did they do that?

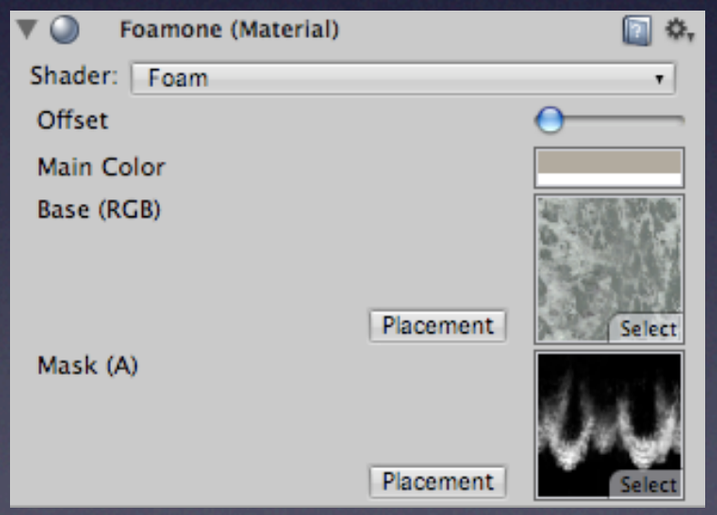
Water shorelines

- No funky shaders
- Just careful feeding



Water shorelines

- Shader combines wave mask & texture
- Script does some voodoo texture scrolling
- All about feeding the shader




```
function Update ()
{
    CheckHWSupport();

    slideInertia = Mathf.Lerp(slideInertia, Mathf.PingPong((Time.time * scaleSpeed) + offset, 1), slideSharpness * Time.deltaTime);
    slide = Mathf.Lerp(slide, slideInertia, slideSharpness * Time.deltaTime);
    theMaterial.SetTextureOffset("_MainTex", Vector3(index * 0.35, Mathf.Lerp(slideMin, slideMax, slide) * 2, 0));
    theMaterial.SetTextureOffset("_Cutout", Vector3(index * 0.79, Mathf.Lerp(slideMin, slideMax, slide) / 2, 0));

    fade = Mathf.Lerp(fade, slide - lastSlide > 0 ? 1 : 0, Time.deltaTime * fadeSpeed);
    lastSlide = slide;
    theMaterial.SetColor("_Color", Color.Lerp(fadeColor, color, fade));

    scaleInertia = Mathf.Lerp(scaleInertia, Mathf.PingPong((Time.time * scaleSpeed) + offset, 1), scaleSharpness * Time.deltaTime);
    scale = Mathf.Lerp(scale, scaleInertia, scaleSharpness * Time.deltaTime);
    theMaterial.SetTextureScale("_MainTex", Vector3(texScale.x, Mathf.Lerp(scaleMin,scaleMax, scale), texScale.z));

    basePos += baseScroll * Time.deltaTime;
    var inverseScale = Vector3 (1 / baseScale.x, 1 / baseScale.y, 1 / baseScale.z);
    var uvMat = Matrix4x4.TRS (basePos, Quaternion.Euler (baseRotation,90,90), inverseScale);
    theMaterial.SetMatrix ("_WavesBaseMatrix", uvMat);
}
```

Cubemaps FTW

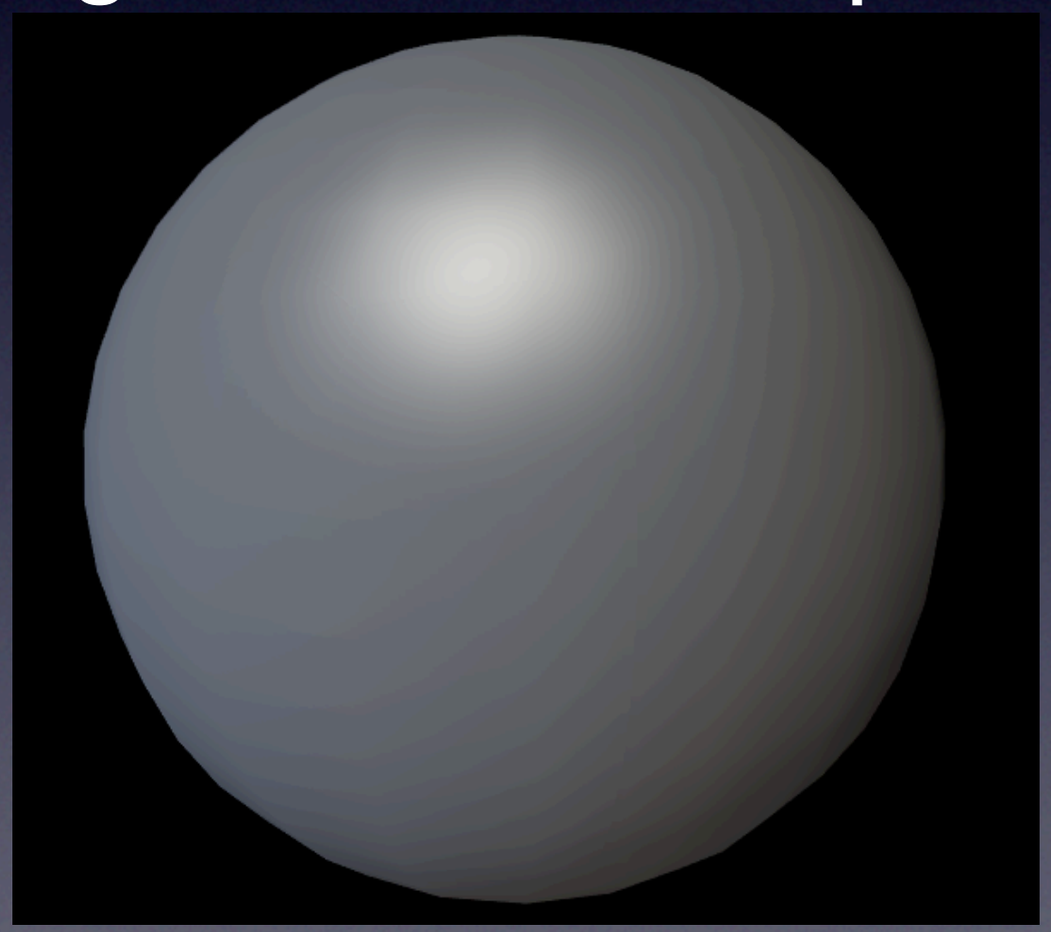
- Cubemap is like a textured cube
  - Much like skybox faces
- Reflections!
- Lighting!

Reflections

- Reflective shaders were in Unity for ages
- But how you make the cubemap?
- In 2.0 we have camera.RenderToCubemap
- Demotime

Cubemap lighting

- This uses no lights at all!
- All lighting is in the cubemap

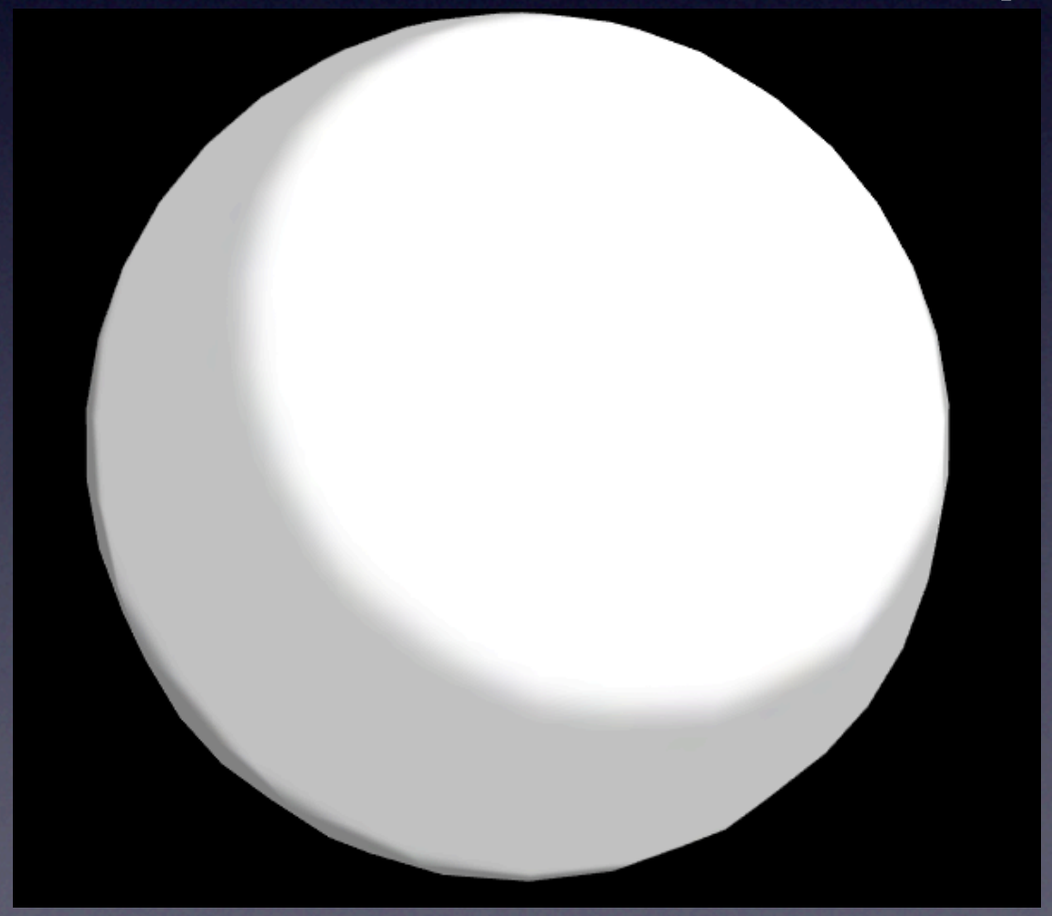


Cubemap lighting

- Encodes lots of lights in one cubemap
- Looks better
- Works a lot faster
- GC: Palestine used to great effect

Cubemap lighting

- Hey, it's toon lighting!
- Same shader. Different cubemap.



cubemap.SetPixel

- Compute your lighting into a cubemap

```
@MenuItem("Custom/Gen Cubemap _g")
static function GenCubemap() {
    var cubemap : Cubemap = EditorUtility.FindAsset("New Cubemap.cubemap", Cubemap);
    var size = cubemap.width;
    var fsize : float = size;
    for( var face = 0; face < 6; ++face )
    {
        for( var y = 0; y < size; ++y )
        {
            for( var x = 0; x < size; ++x )
            {
                var dir : Vector3;
                switch( face ) {
                case 0: dir = Vector3(          1.0 , -(y/fsize*2-1), -(x/fsize*2-1)); break;
                case 1: dir = Vector3(         -1.0 , -(y/fsize*2-1),   x/fsize*2-1) ; break;
                case 2: dir = Vector3(  x/fsize*2-1 ,           1.0 ,   y/fsize*2-1) ; break;
                case 3: dir = Vector3(  x/fsize*2-1 ,          -1.0 , -(y/fsize*2-1)); break;
                case 4: dir = Vector3(  x/fsize*2-1 , -(y/fsize*2-1),           1.0) ; break;
                case 5: dir = Vector3(-(x/fsize*2-1), -(y/fsize*2-1),          -1.0) ; break;
                }
                dir = dir.normalized; // this is direction to pixel now
                var color : Color;


                var lightDir = Vector3(1,1,0);
                var viewDir = Vector3(0,0,1);
                var diffuse = Vector3.Dot( lightDir, dir );
                color.g = diffuse;
                color.b = -diffuse;
                var edge = Vector3.Dot( viewDir, dir );
                if( edge < 0.6 )
                    color *= 0.5;

                cubemap.SetPixel( face, x, y, color );
            }
        }
    }
    cubemap.Apply();
}
```

Skin shading

- Diffuse vs. skin

Example project

- Will provide example project after conference

Performance

Performance

- Fillrate
- Geometry, draw calls
- Shadows

Fillrate

- Drawing pixels
  - VRAM bandwidth & shader computations
- Change resolution: does FPS change?

Fillrate

- GeForce 8800U
  - 40 billion pixels/sec, 100 GB/sec
- Intel GMA 950
  - 1.5 billion pixels/sec, 10 GB/sec (shared)
- Resolution & pixel light count

Shadows

- Use wisely
- Point light shadows = evil
  - Draws scene ~6 times into a cubemap

Custom shaders

- How long a shader should be?
- Is this expensive?

```
half4 frag(v2f i) : COLOR
{
    half3 n = i.normal;
    half4 col = texCUBE( _Cube, n );
    return col;
}
```

Custom shaders

- Inspector shows assembly
- This is 1 instruction

```
SubProgram "opengl " {
Keywords { }
SetTexture [_Cube] {CUBE}
"!!ARBfp1.0
# 1 instructions, 1 texture reads
TEX result.color, fragment.texcoord[0], texture[0], CUBE;
END
# 1 instructions, 0 R-regs
"
}
```

Custom shaders

- 50 instruction shader can run 5x slower than a 10 instruction one
- Could draw 5x more pixels!
- Don’t compute if you don’t have to
- “Bake” math into textures / cubemaps

Bake math into texture

- XRay shader on the wiki
- Run dot(normal, viewdir) through texture
- Very cheap to render

Bake math into texture

- Artist friendly
- This is just different ramp textures



Questions?