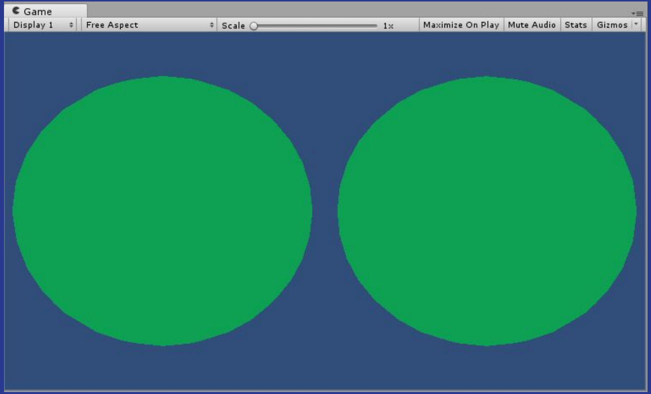
CS 114 Final - Unity3D Shaders

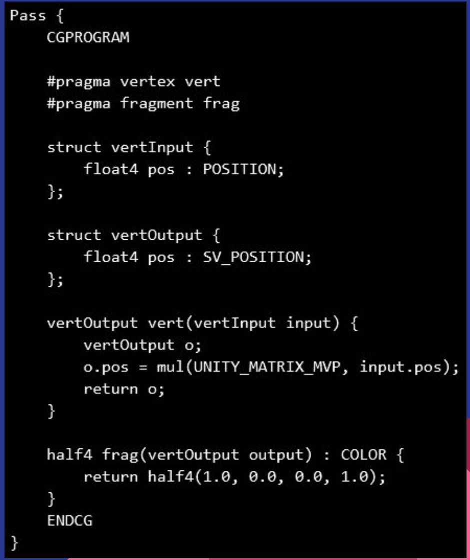
Christian Sydow

Starting out...

The most basic shaders start with a vertex and fragment shader

Flat shaders use the vertex shader to bring the model's vertex to screen space, and the fragment shader returns the color at that pixel location

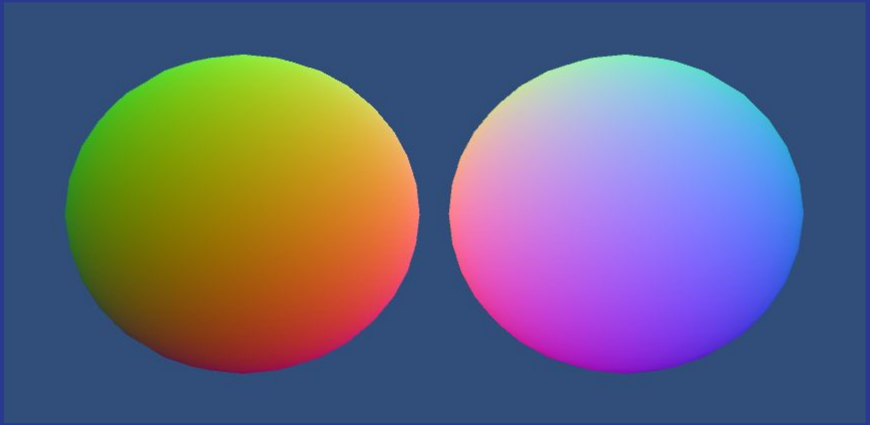


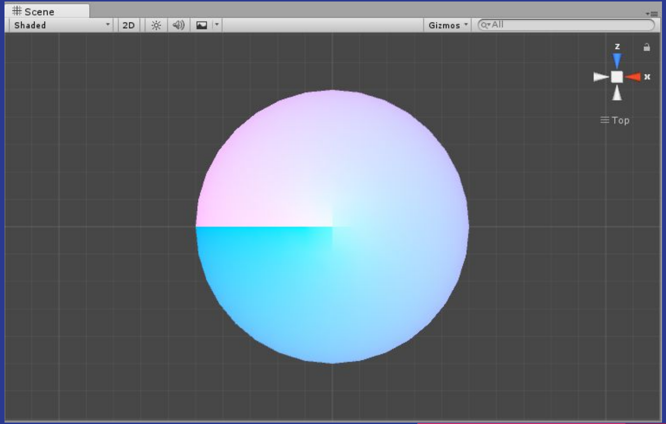


```
Pass {
    CGPROGRAM

    #pragma vertex vert
    #pragma fragment frag

    struct vertInput {
        float4 pos : POSITION;
    };

    struct vertOutput {
        float4 pos : SV_POSITION;
    };

    vertOutput vert(vertInput input) {
        vertOutput o;
        o.pos = mul(UNITY_MATRIX_MVP, input.pos);
        return o;
    }

    half4 frag(vertOutput output) : COLOR {
        return half4(1.0, 0.0, 0.0, 1.0);
    }
    ENDCG
}
```

Using vertex position

Next we can use vertex position to map textures

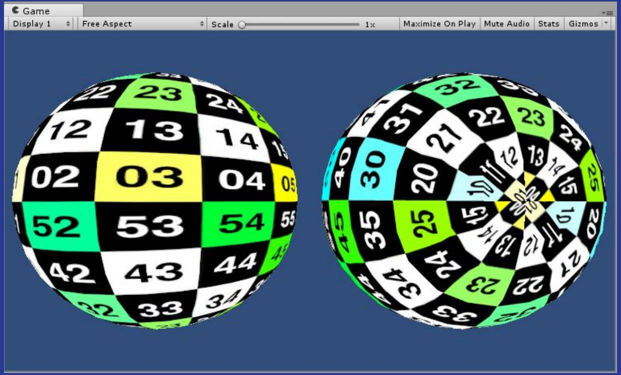


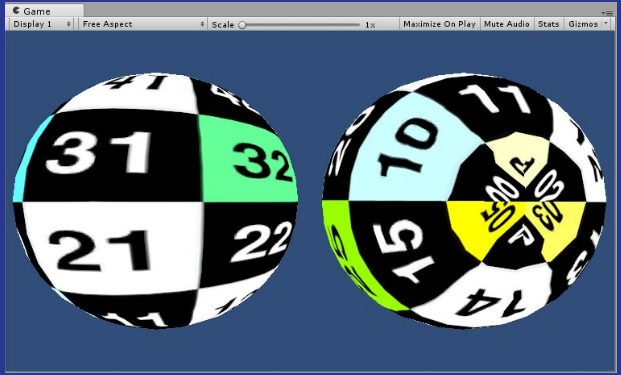


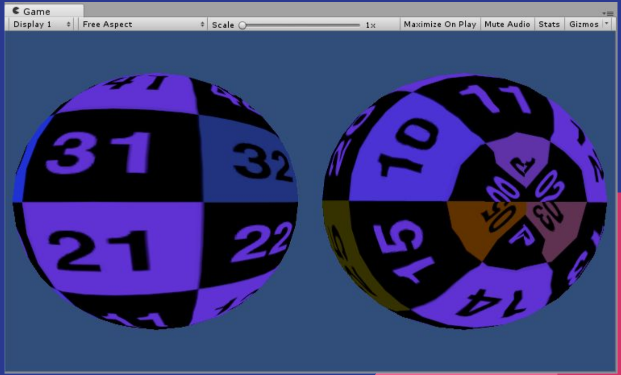
UV Mapping

We can multiply or add a texture in the vertex shader to offset it or tile it

We can also multiply it by an rgb value to alter the tint of the texture



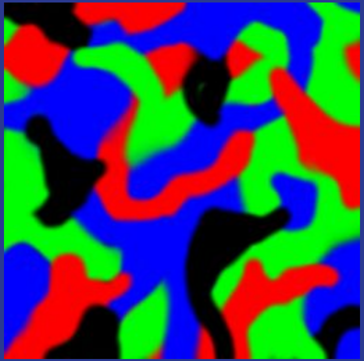


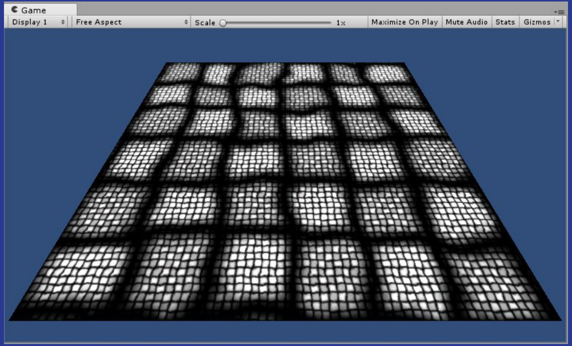


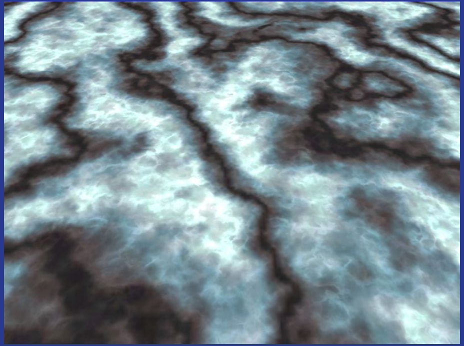
Combining Textures

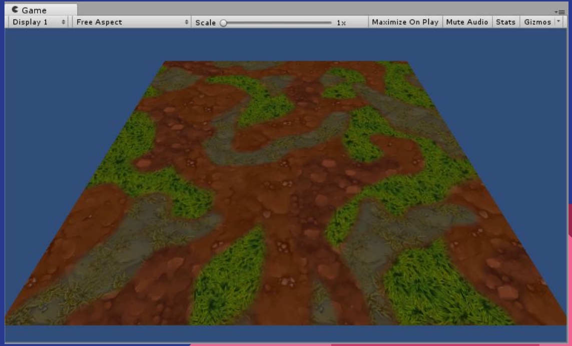
We can also multiply different textures together to add details

Using a splat map, we can map several textures to the same material and have them separated based on RGB channels





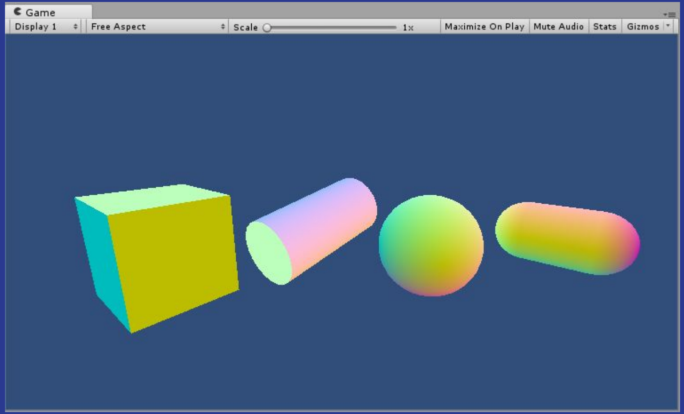




Adding Diffuse Lighting

To add lighting, we must first calculate the normals for the surface at each vertex

Calculate the diffuse using the dot product of the normal and light direction

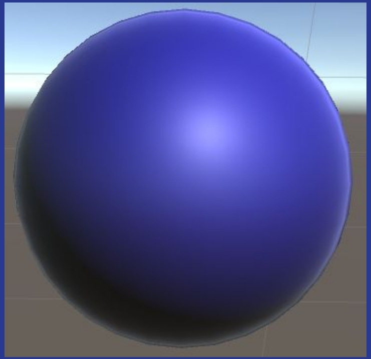
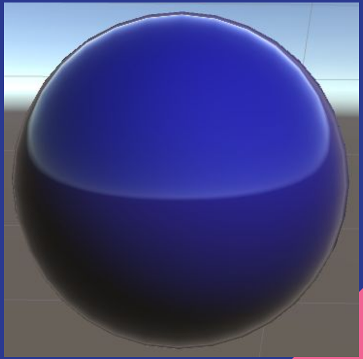
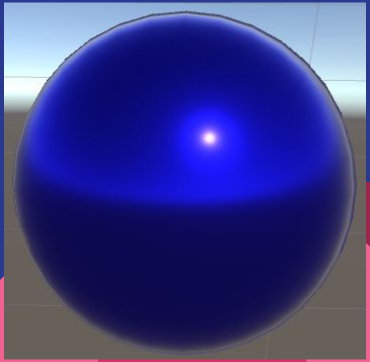


Adding Specular Lighting

For specular lighting, we need to find the view direction by subtracting the surface vertex position from the camera position in world space and normalizing it

We calculate the half-vector by normalizing the sum of the light direction and view direction, we then take the dot product of the normal and this and multiply it by a smoothness value between 0-1

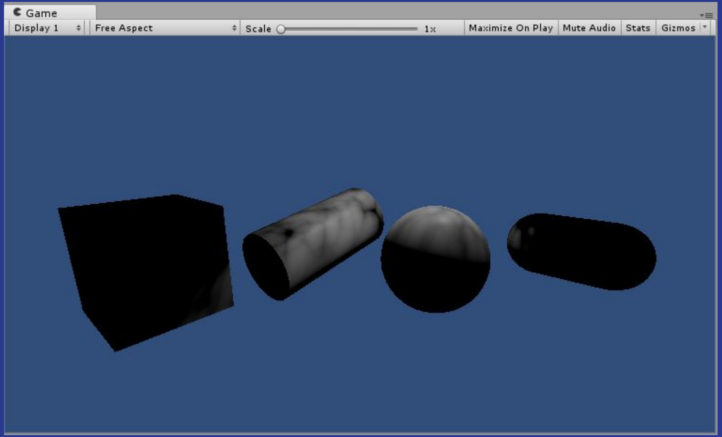
Add the diffuse for the complete lighting model

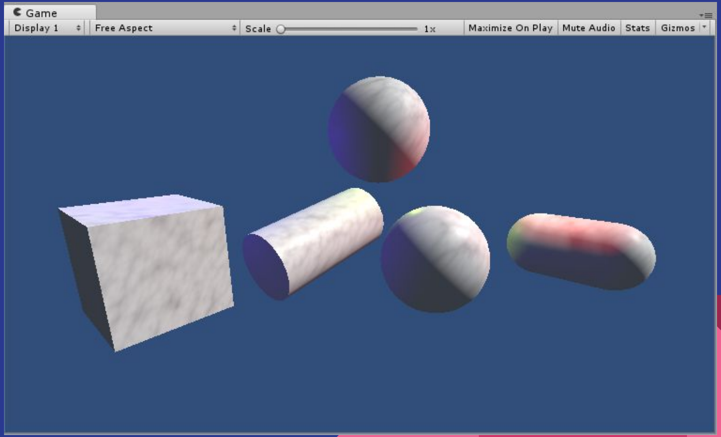




Multiple Light Sources

To enable interaction with multiple lights, we need to add another additive pass to the shader

Non-directional lights need to have their location taken into account since they aren't "infinite" like directional lights

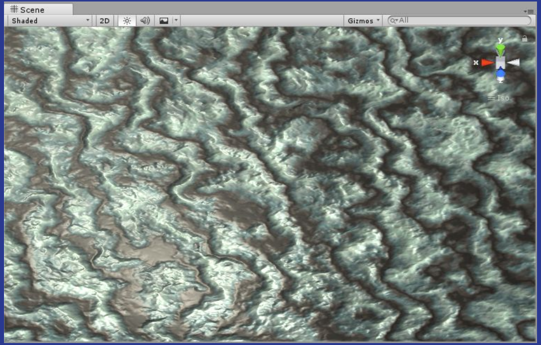


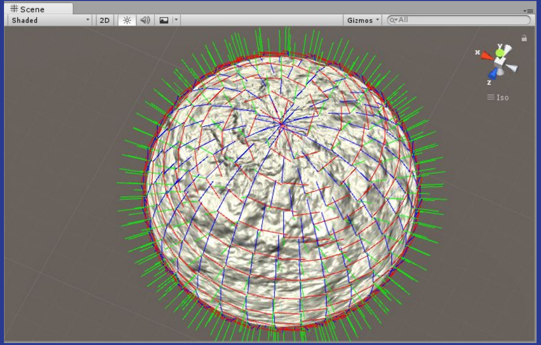


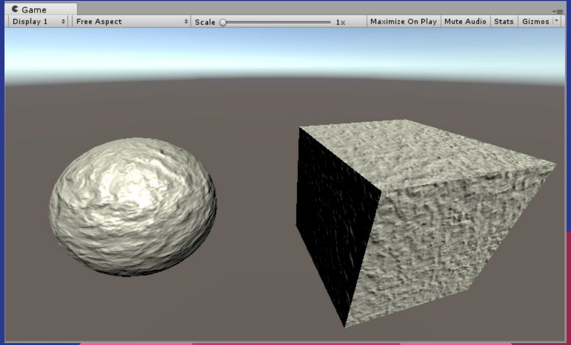
Bump/Normal Mapping

Calculate the height of the bump from the normal map, RGB = XYZ

Calculate the bivector with the tangent X normal to have the new adjusted normals for the bumps



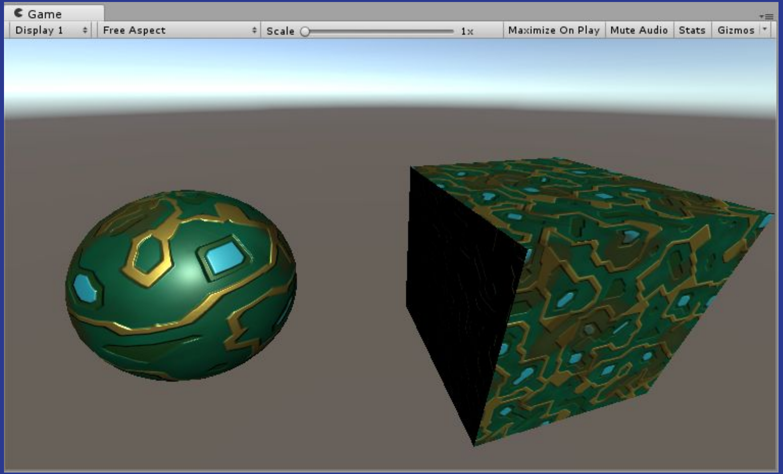


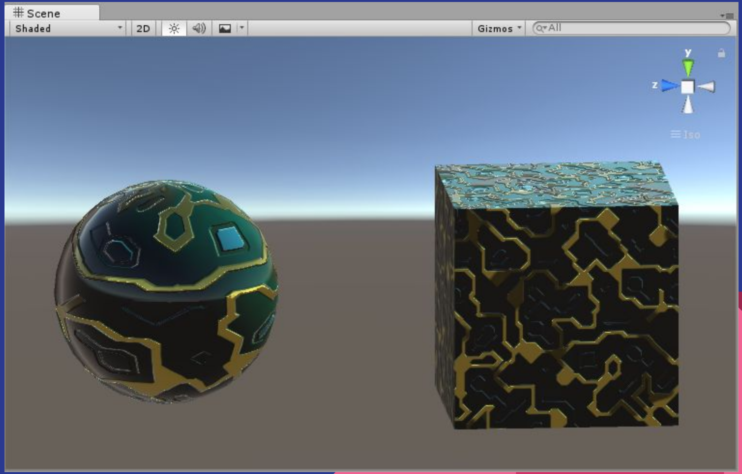


Complex Materials

Complex materials can combine maps for materials with higher detail

Store specular/metallic/smoothness maps to affect isolated parts of albedo texture

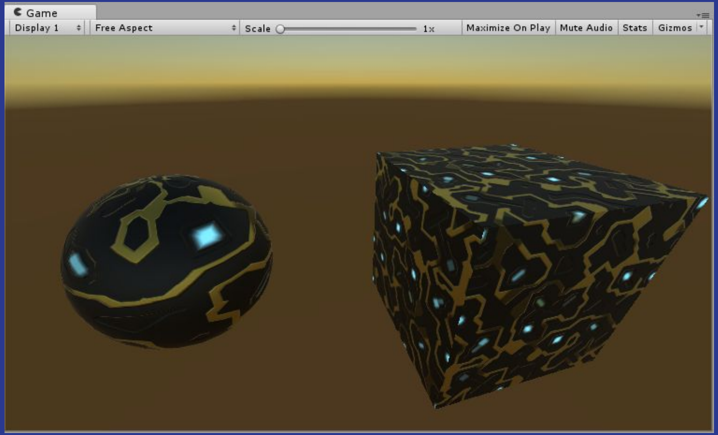


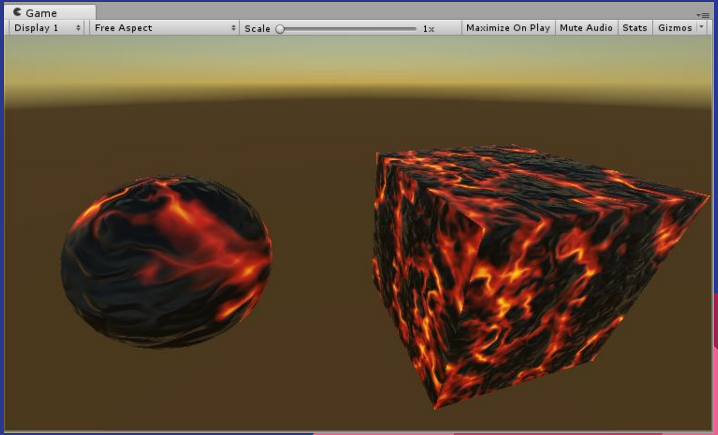


Complex Materials Continued

You can also store emissive maps to have parts of albedo remain lighted even in darkness

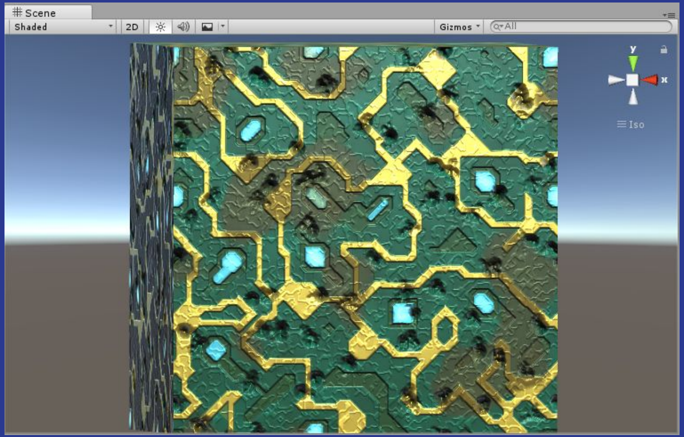
Emission is similar to flat shading, so the light doesn't affect the scene

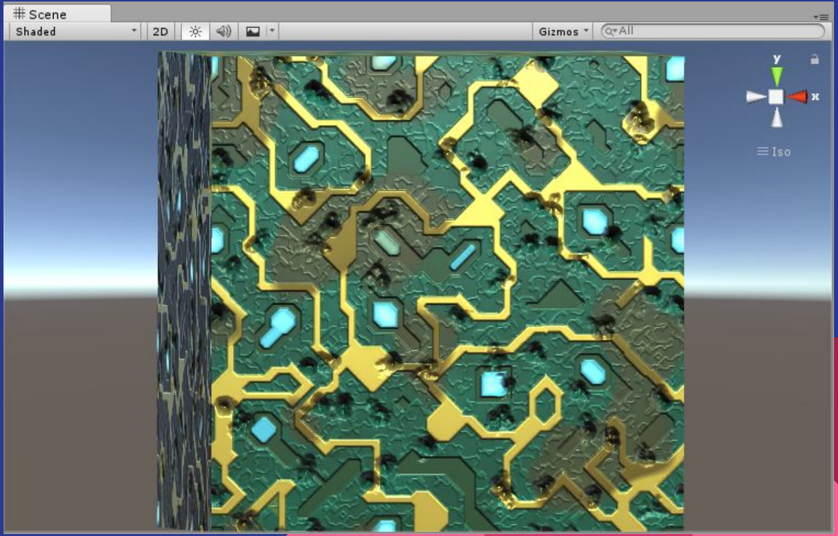




Complex Materials Continued

You can use masks, similar to the splat maps, to control where secondary detail albedo and normal maps affect, on the left is the material without a mask, and on the right the mask prevents the gold part and deep parts of the circuit from having the detail mapping


Scene
Shaded
2D
Gizmos
All
y
x
Iso


Scene
Shaded
2D
Gizmos
All
y
x
Iso