Charles Hancock

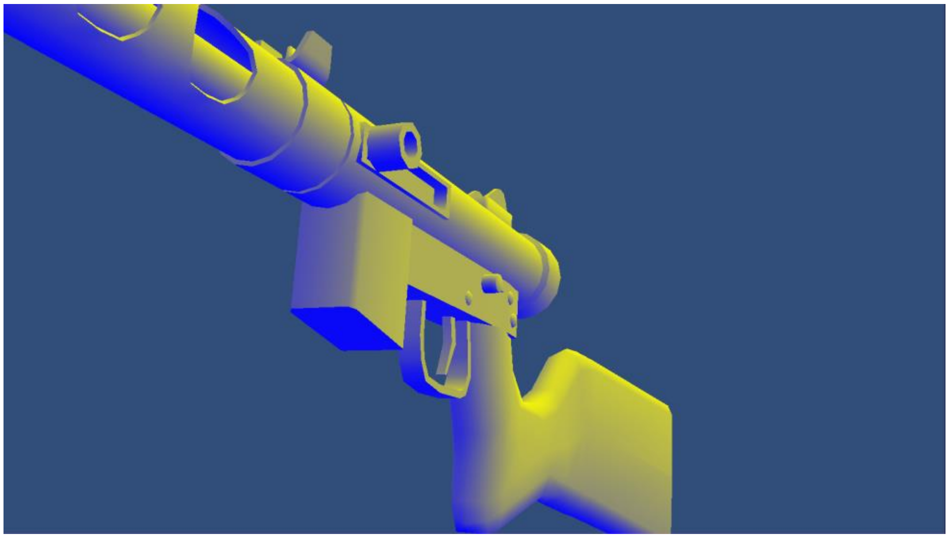
Homework 2

CS 4803

Aaron Lanterman

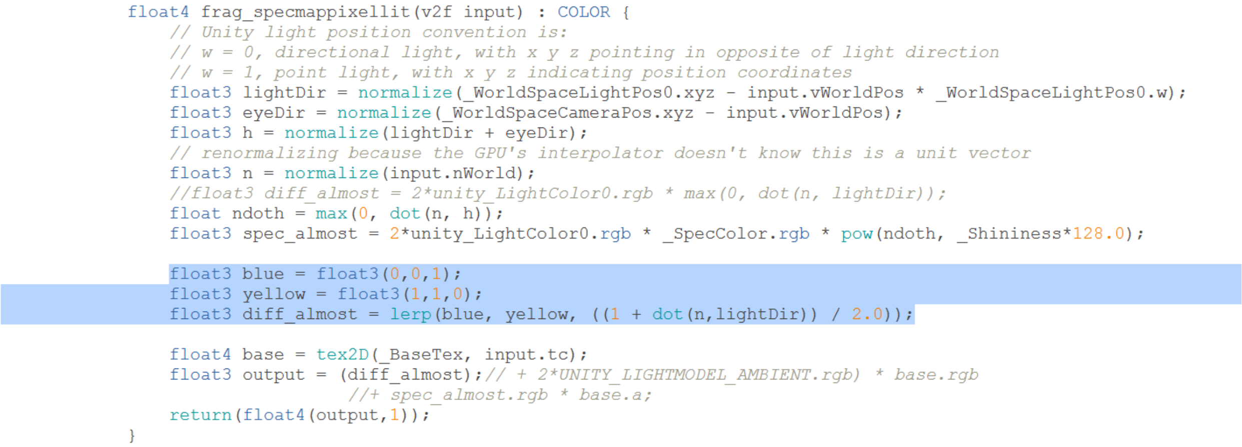
6/20/2014

PROBLEM 1



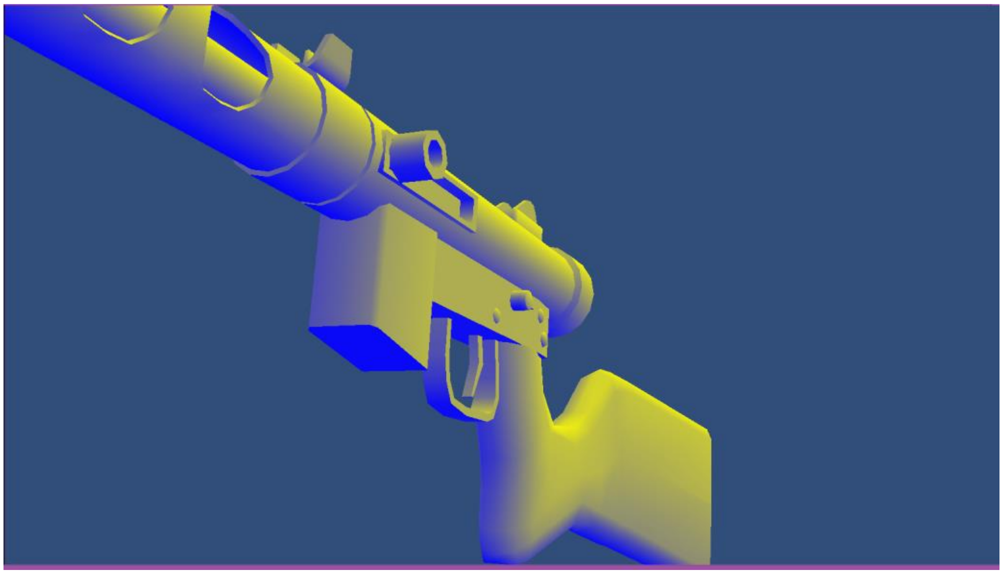
Pixel-lit

GPUXXSpecmapPixelLit.shader:



```
float4 frag_specmappixellit(v2f input) : COLOR {
    // Unity light position convention is:
    // w = 0, directional light, with x y z pointing in opposite of light direction
    // w = 1, point light, with x y z indicating position coordinates
    float3 lightDir = normalize(_WorldSpaceLightPos0.xyz - input.vWorldPos * _WorldSpaceLightPos0.w);
    float3 eyeDir = normalize(_WorldSpaceCameraPos.xyz - input.vWorldPos);
    float3 h = normalize(lightDir + eyeDir);
    // renormalizing because the GPU's interpolator doesn't know this is a unit vector
    float3 n = normalize(input.nWorld);
    //float3 diff_almost = 2*unity_LightColor0.rgb * max(0, dot(n, lightDir));
    float ndoth = max(0, dot(n, h));
    float3 spec_almost = 2*unity_LightColor0.rgb * _SpecColor.rgb * pow(ndoth, _Shininess*128.0);

    float3 blue = float3(0,0,1);
    float3 yellow = float3(1,1,0);
    float3 diff_almost = lerp(blue, yellow, ((1 + dot(n,lightDir)) / 2.0));

    float4 base = tex2D(_BaseTex, input.tc);
    float3 output = (diff_almost);// + 2*UNITY_LIGHTMODEL_AMBIENT.rgb) * base.rgb
                    //+ spec_almost.rgb * base.a;
    return(float4(output,1));
}
```

Note the commented out specular and ambient code at the bottom



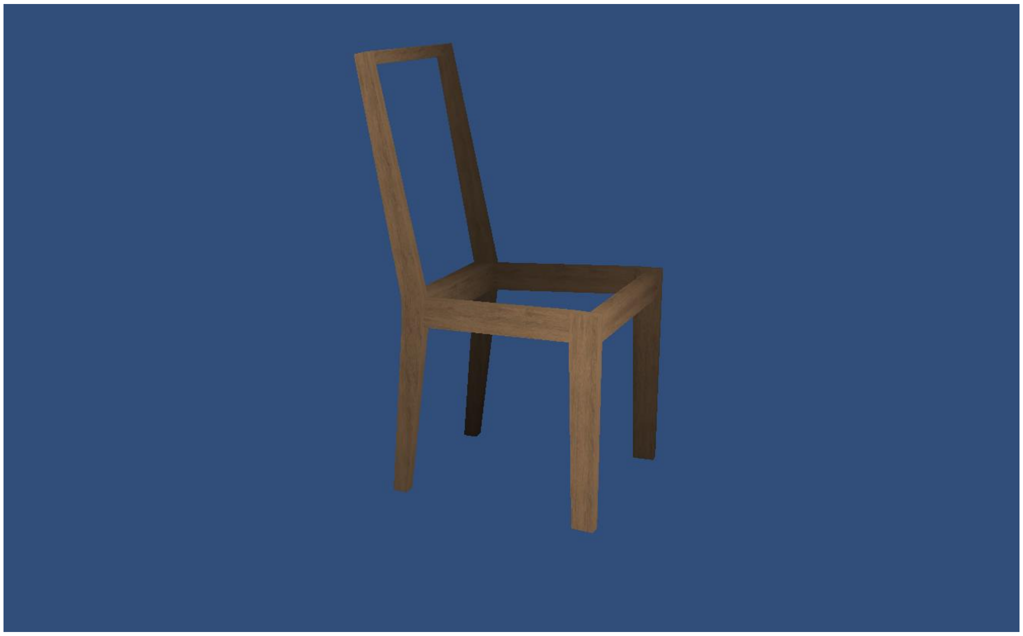
Vertex Lit (they are ever so slightly different)

GPUXXSpecmapVertexLit.shader:

```
v2f vert_specmapvertexlit(a2v input)  {
    v2f output;
    output.sv = mul (UNITY_MATRIX_MVP, input.v);

    float3 vWorldPos = mul (_Object2World, input.v).xyz;
    // To transform normals, we want to use the inverse transpose of upper left 3x3
    // Putting input.n in first argument is like doing trans((float3x3)_World2Object) * input.n;
    float3 nWorld = normalize(mul(input.n, (float3x3) _World2Object));

    // Unity light position convention is:
    // w = 0, directional light, with x y z pointing in opposite of light direction
    // w = 1, point light, with x y z indicating position coordinates
    float3 lightDir = normalize(_WorldSpaceLightPos0.xyz - vWorldPos * _WorldSpaceLightPos0.w);
    float3 eyeDir = normalize(_WorldSpaceCameraPos.xyz - vWorldPos);
    float3 h = normalize(lightDir + eyeDir);
    //output.diff_almost = 2*unity_LightColor0.rgb * max(0, dot(nWorld, lightDir));
    float3 blue = float3(0,0,1);
    float3 yellow = float3(1,1,0);
    output.diff_almost = lerp(blue, yellow, ((1 + dot(nWorld,lightDir)) / 2.0));

    float ndoth = max(0, dot(nWorld, h));
    //output.spec_almost = 2*unity_LightColor0.rgb * _SpecColor.rgb * pow(ndoth, _Shininess*128.0);

    output.tc = TRANSFORM_TEX(input.tc, _BaseTex);
    return output;
}

float4 frag_specmapvertexlit(v2f input) : COLOR {
    //float4 base = tex2D(_BaseTex, input.tc);
    float3 output = (input.diff_almost);// + 2*UNITY_LIGHTMODEL_AMBIENT.rgb) * base.rgb
                 //+ input.spec_almost.rgb * base.a;
    return(float4(output,1));
}
```

Also note the commented-out ambient/specular code at the bottom

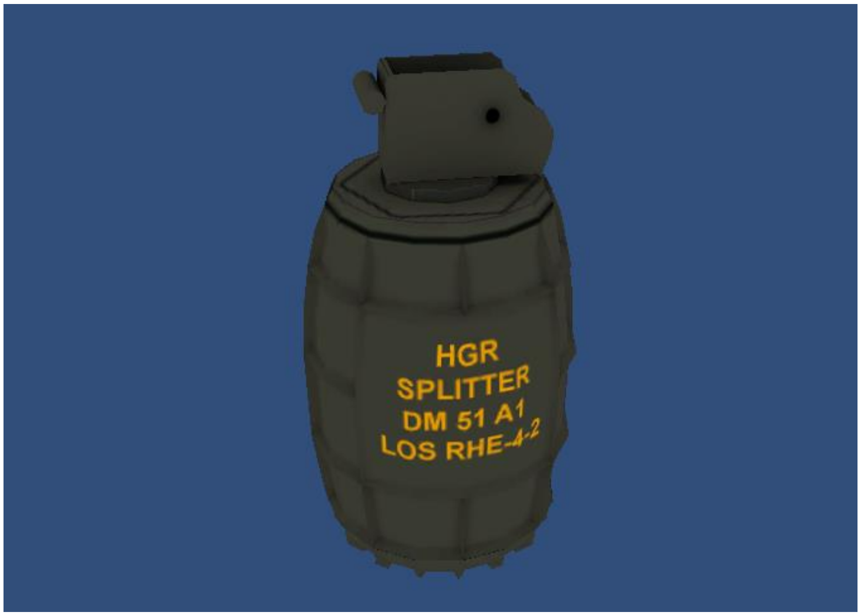
PROBLEM 2



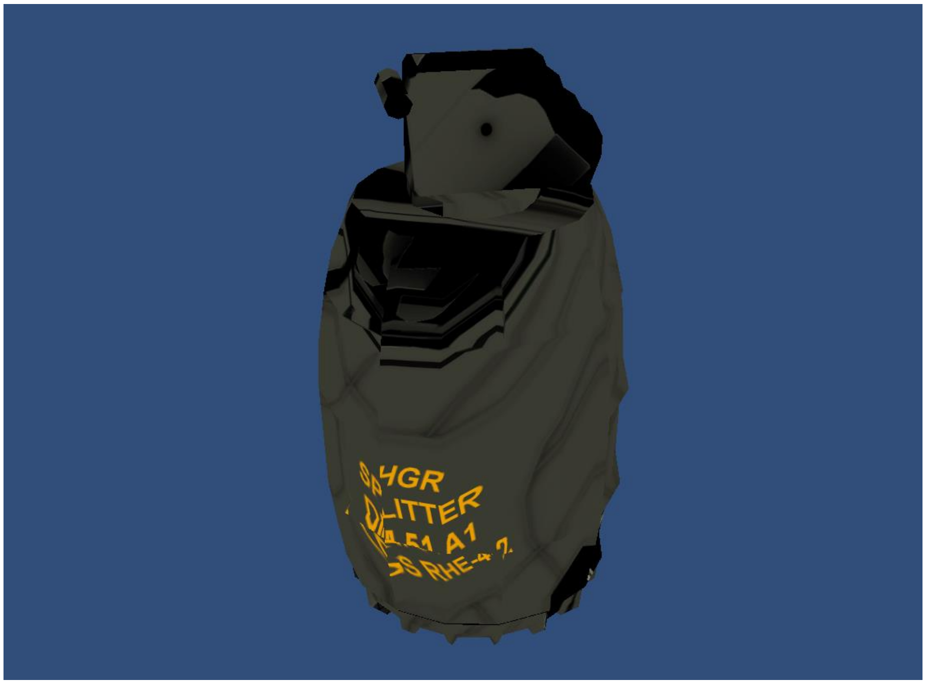
GPUXXTexturedTileCorrectly.shader:

```
struct v2f {                        // vertex to fragment
           float4 sv: SV_POSITION;
           float2 tc: TEXCOORD0;   // not same as TEXCOORD0 above
           float depthFactor: TEXCOORD1;
       };
v2f vert_texturedstruct(a2v input) {
     v2f output;
     output.sv = mul(UNITY_MATRIX_MVP, input.v);
     float e = 2.8;
     float s = 2;
     output.depthFactor = max(0,min(1,(e-output.sv.z)/(e-s)));
     // Make sure you TRANSFORM_TEX the vertex shader, not the
     // fragment shader!
     output.tc = TRANSFORM_TEX(input.tc, _BaseTex);
     return output;
}
```

PROBLEM 3



Before

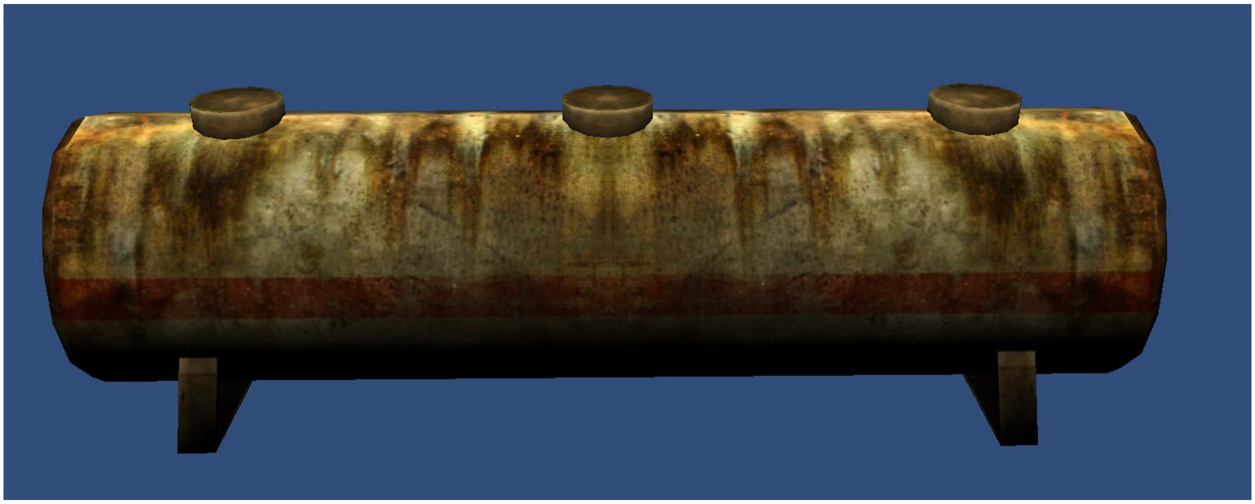


After

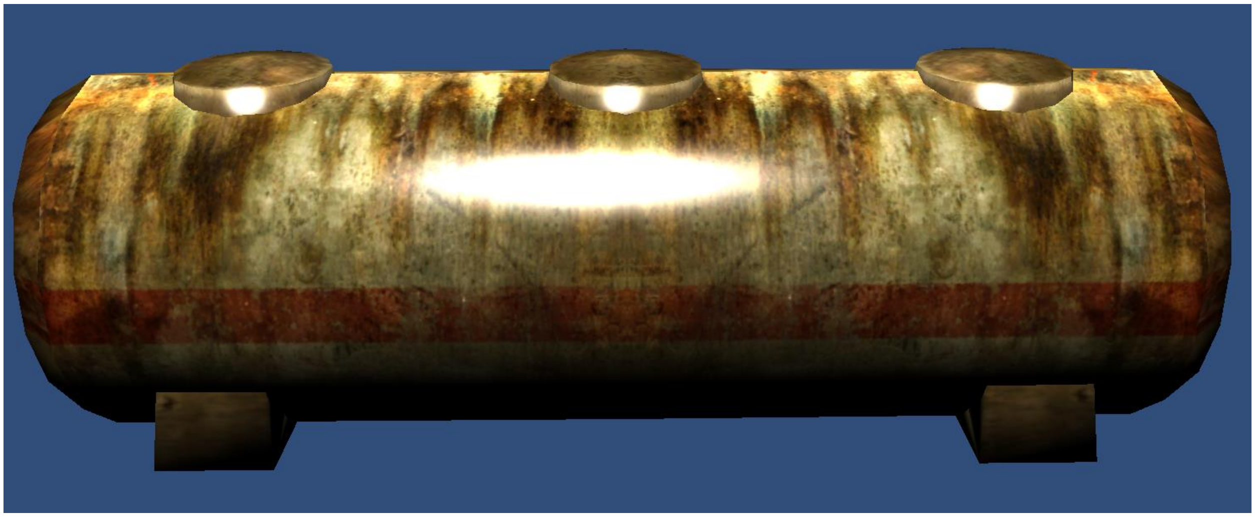
GPUXXTexturedStruct.shader:

```
v2f vert_texturedstruct(a2v input) {
                v2f output;
                output.sv = mul(UNITY_MATRIX_MVP, input.v);
                output.tc = input.tc;
                float AA = 1;
                float B = 10;
                float C = .5;
                float D = 1;
                float E = 10;
                float F = .5;
                output.tc.x += AA*sin(B*input.tc.y)*sin(C*_Time.x);
                output.tc.y += D*sin(E*input.tc.x)*sin(F*_Time.x);
                return output;
            }
```

PROBLEM 4



Before



After

GPUXXSpecmapPixelLit.shader:

```
v2f vert_specmappixellit(a2v input)  {
      v2f output;

      float AA = 1;
      float BB = 10;

      input.v += AA * float4(input.n.x, input.n.y, input.n.z, 0) *
      (1+sin(BB*_Time.x));

      output.sv = mul(UNITY_MATRIX_MVP, input.v);
      // To transform normals, we want to use the inverse
       transpose of upper left 3x3
      // Putting input.n in first argument is like doing
       trans((float3x3)_World2Object) * input.n;
      output.vWorldPos = mul(_Object2World, input.v).xyz;
      output.nWorld = normalize(mul(input.n, (float3x3)
       _World2Object));
      output.tc = TRANSFORM_TEX(input.tc, _BaseTex);
      return output;
   }
```