HW3_pronlem1

Code:

```
Properties {
    _BaseTex ("Base (RGB) diffuse", 2D) = "white" {}
   _BaseTex2 ("Base (RGB) Gloss (A)", 2D) = "white" {}
   _Cube ("Reflection Cubemap", CUBE) = "white" {}

}
```

```
            sampler2D _BaseTex;
           float4 _BaseTex_ST;

sampler2D _BaseTex2;
float4 _BaseTex2_ST;
```

```
struct v2f {             // vertex to fragment
   float4 sv: SV_POSITION;
   float2 uv:TEXCOORD0;
   float2 uv2:TEXCOORD1;
   float3 positionWS: TEXCOORD2;
   float3 normalWS: TEXCOORD3;

};
```

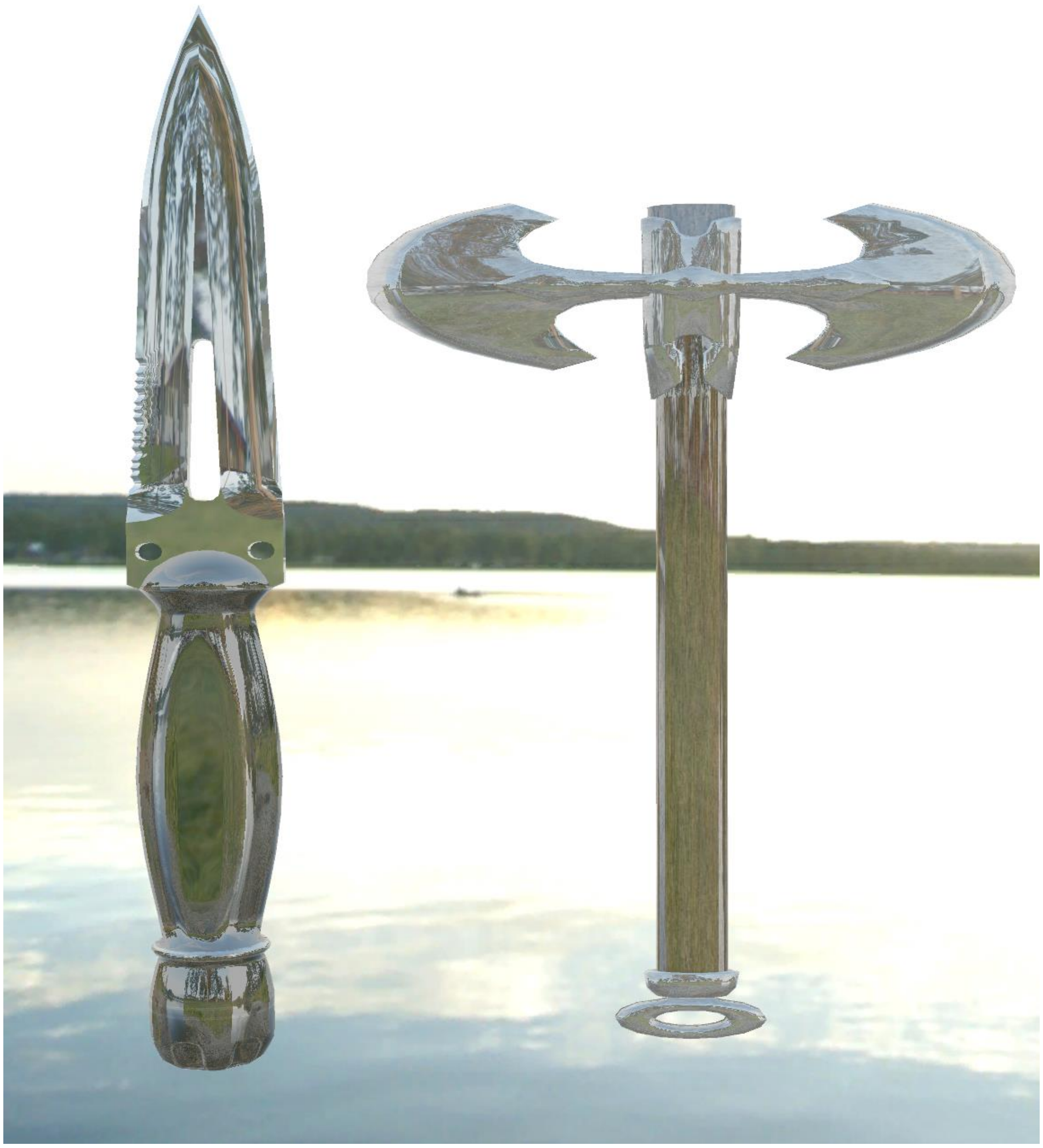
```
output.normalWS = normalize(mul(input.normalOS, (float3x3)
unity_WorldToObject));
output.uv = TRANSFORM_TEX(input.uv, _BaseTex);
output.uv2 = TRANSFORM_TEX(input.uv, _BaseTex2);
return output;
```

```
float4 FragEnvMapperPixel(v2f input) : COLOR {
          // incident is opposite the direction of eyeDir in our other
programs
float3 incidentWS = normalize(input.positionWS - _WorldSpaceCameraPos.xyz);
float3 reflectWS = reflect(incidentWS, input.normalWS);

float4 reflectColor = texCUBE(_Cube, reflectWS);
float4 base = tex2D(_BaseTex, input.uv);
float4 base2 = tex2D(_BaseTex2, input.uv2);
return(lerp(base, reflectColor, base2.a));

}
```

Demo:



As we can see they all have the glossy look on them which is refectiing the cube map.

HW3_problem2

Code:

```
Properties {
   _Cube ("Reflection Cubemap", CUBE) = "white" {}
       _Alpha ("Alpha", Range(0,1)) = 0.5
   _etaRatioRed ("Eta Ratio Red", Range(0.01,3)) = 1.5
       _etaRatioGreen ("Eta Ratio Green", Range(0.01,3)) = 1.5
       _etaRatioBlue ("Eta Ratio Blue", Range(0.01,3)) = 1.5
   _crossfade ("Crossfade", Range(0,1)) = 0
   _fresnelBias ("Fresnel Bias", Range(0,1)) = 0.5
   _fresnelScale ("Fresnel Scale", Range(0,1)) = 0.5
   _fresnelPower ("Fresnel Power", Range(0,10)) = 0.5
   [Toggle(BLUE_YELLOW)] _BlueYellow ("Blue Yellow", Float) = 0
       [KeywordEnum(Crossfade, Fresnel)] _Blend ("Blend Mode", Float) = 0
}
```

```
samplerCUBE _Cube;
float _etaRatioRed;
float _etaRatioBlue;
float _etaRatioGreen;
float _crossfade;
float _fresnelBias;
float _fresnelScale;
float _fresnelPower;
float _Alpha;
```

```
float4 FragEnvMapperPixel(v2f input) : COLOR {
          // incident is opposite the direction of eyeDir in our other
programs
float4 refractColor;
float3 incidentWS = normalize(input.positionWS - _WorldSpaceCameraPos.xyz);
float3 reflectWS = reflect(incidentWS, input.normalWS);
float3 refractWSRed = refract(incidentWS, input.normalWS, _etaRatioRed);
float3 refractWSGreen = refract(incidentWS, input.normalWS, _etaRatioGreen);
float3 refractWSBlue = refract(incidentWS, input.normalWS, _etaRatioBlue);
float4 reflectColor = texCUBE(_Cube, reflectWS);

refractColor.r = texCUBE(_Cube, refractWSRed).r;
refractColor.g = texCUBE(_Cube, refractWSGreen).g;
refractColor.b = texCUBE(_Cube, refractWSBlue).b;
refractColor.a = _Alpha;
float reflectFactor = saturate(_fresnelBias +fresnelScale * pow(1 +
dot(incidentWS, input.normalWS),_fresnelPower));

#ifdef BLUE_YELLOW  // for visualization
refractColor = float4(1,1,0,1);
reflectColor = float4(0,0,1,1);
#endif

#if defined(_BLEND_CROSSFADE)
return(lerp(reflectColor, refractColor, _crossfade));
#elif defined( _BLEND_FRESNEL)
return(lerp(refractColor, reflectColor, reflectFactor));
#endif
         }
```

Demo

HW3_problem3:

Code;

```
Properties {
   _BaseTex ("Base (RGB) Gloss (A)", 2D) = "white" {}
   _ProjTex ("Projected (RGB)", 2D) = "white" {}
   _NormalMap("Normalmap", 2D) = "bump" {}
    _SpotPower ("Spotlightiness", Range(0.01,1)) = 0.7
}
```

```
sampler2D _BaseTex;
float4 _BaseTex_ST;
sampler2D _ProjTex;

sampler2D _NormalMap;
float4 _NormalMap_ST;

float _SpotPower;

float4x4 _myProjectorMatrixVP;
float3 _spotlightDir;

float4 _LightColor0;
```

```
struct a2v {
float4 positionOS: POSITION;
float3 normalOS: NORMAL;
float4 tangentOS: TANGENT;
float2 uv: TEXCOORD0;
};

struct v2f {
float4 sv: SV_POSITION;
float2 bmap_uv:TEXCOORD0;
float2 nmap_uv: TEXCOORD1;
float3 positionWS: TEXCOORD2;
float3 normalWS: TEXCOORD3;
float4 positionProjected: TEXCOORD4;
float3 tangentWS: TEXCOORD5;
float3 bitangentWS: TEXCOORD6;

};
```

```
output.bmap_uv = TRANSFORM_TEX(input.uv, _BaseTex);
output.nmap_uv = TRANSFORM_TEX(input.uv, _NormalMap);
return output;
```

```
float4 FragProjectTexture(v2f input) : COLOR {
float2 nMapXY = 2 * tex2D(_NormalMap, input.nmap_uv).ag - 1;
float nMapRecreatedZ = sqrt(1 - saturate(dot(nMapXY,nMapXY)));

// we are renormalizing because the GPU's interpolator doesn't know these are
unit vectors
float3 nWS = normalize(input.normalWS);
float3 tWS = normalize(input.tangentWS);
float3 btWS = normalize(input.bitangentWS);

float3 newNormal = tWS * nMapXY.x + btWS * nMapXY.y + nWS * nMapRecreatedZ;
newNormal = normalize(newNormal);

// 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



         // Only use this shader with a point light
float3 lightDir = normalize(_WorldSpaceLightPos0.xyz - input.positionWS *
_WorldSpaceLightPos0.w);
         // Renormalizing because the GPU's interpolator doesn't know this is
a unit vector

float3 diffAlmost = _LightColor0.rgb * max(0, dot(newNormal, lightDir));
float spotlightEffect = pow(dot(normalize(_spotlightDir), -
lightDir),_SpotPower * 128.0);
diffAlmost *= spotlightEffect;
diffAlmost *= tex2Dproj(_ProjTex,input.positionProjected);

float4 base = tex2D(_BaseTex, input.bmap_uv);
float3 output = diffAlmost * base.rgb;
return(float4(output,1));
}
```

Demo: