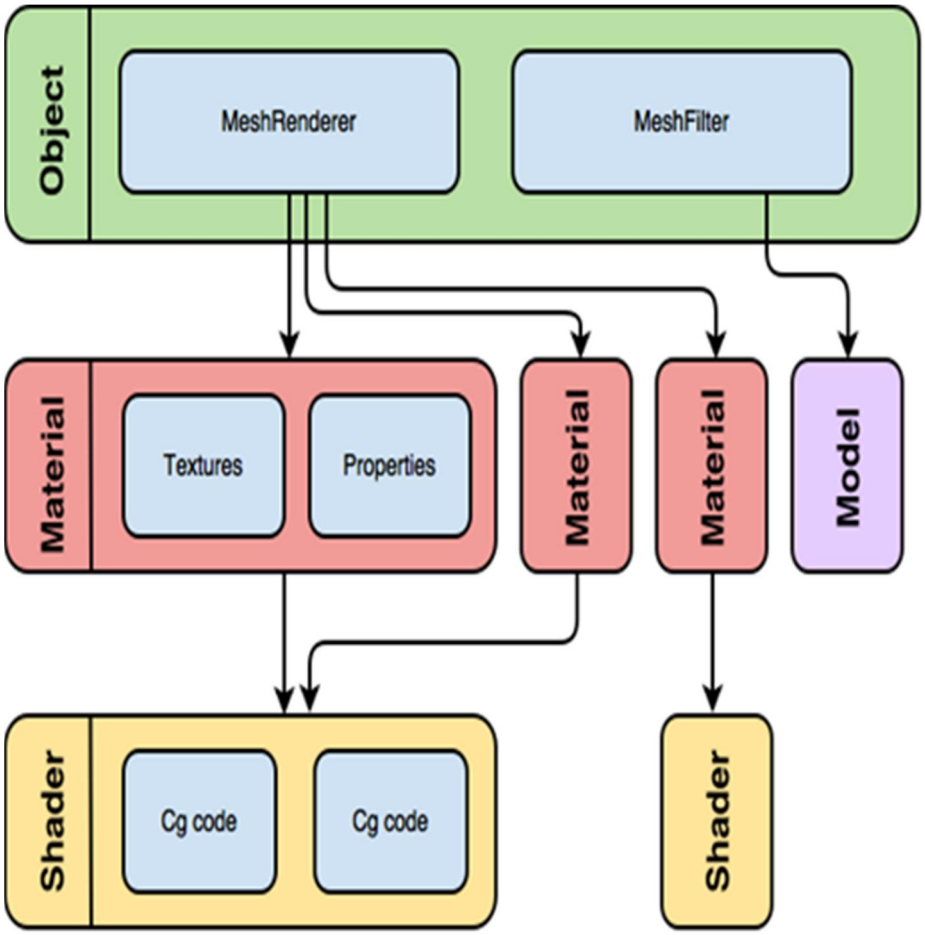
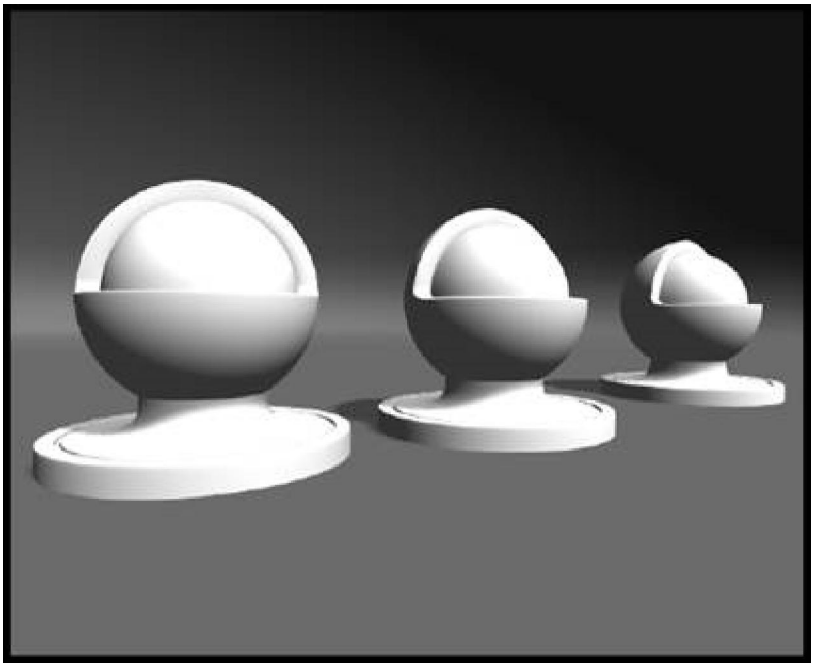
Chapter 1: Shaders and its properties





UNITY DOWNLOAD ARCHIVE

From this page you can download the previous versions of Unity for both Unity Personal and Professional Editions (if you have a Pro license, enter in your key when prompted after installation). Please note that there is no backwards compatibility from Unity 5; projects made in 5.x will not open in 4.x. However, Unity 5.x will import and convert 4.x projects. We advise you to back up your project before converting and check the console log for any errors or warnings after importing.

5.x 4.x 3.x

UNITY 5.1.2
16 Jul, 2015

Downloads (Win) Downloads (Mac) RELEASE NOTES

- Unity Installer
- Unity Editor 64-bit
- Unity Editor 32-bit
- Cache Server
- Built-in shaders
- Standard Assets
- Example Project
- Samsung TV Support installer

UNITY 5.1.1
16 Jun, 2015

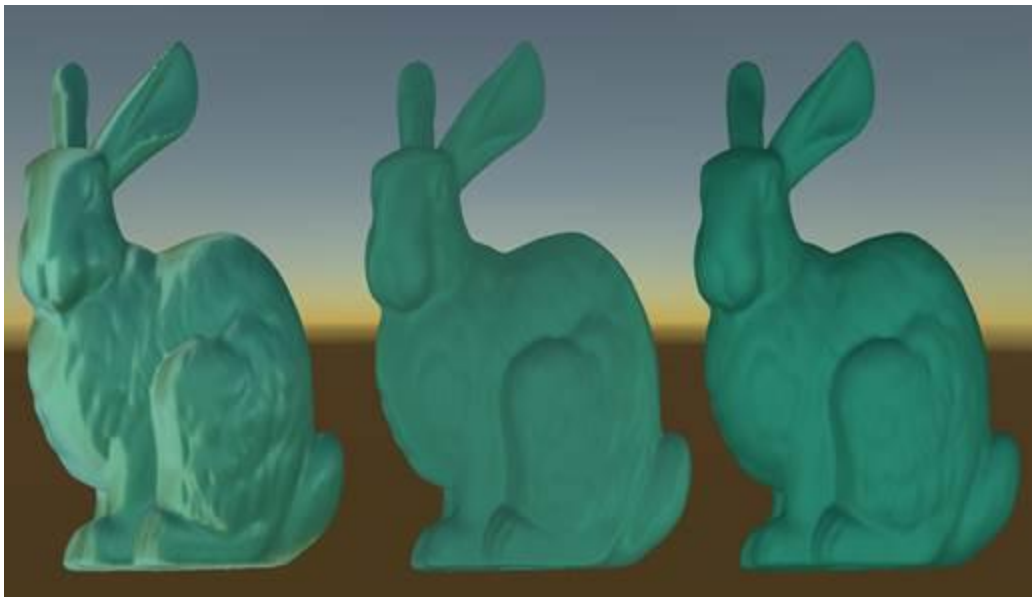
RELEASE NOTES

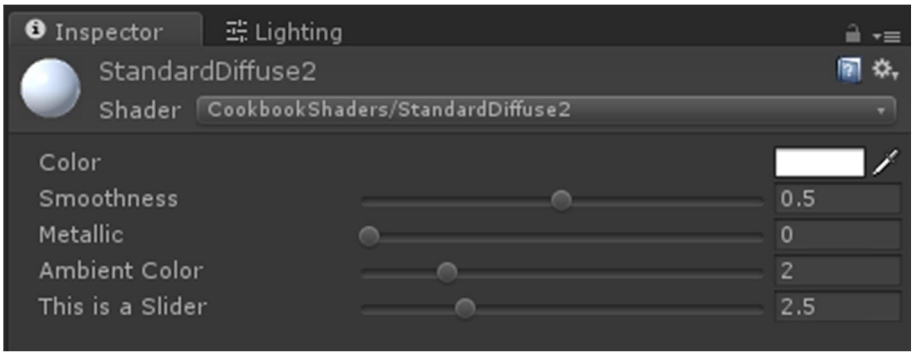
UNITY 5.1.0
9 Jun, 2015

RELEASE NOTES

UNITY 5.0.4
8 Jul, 2015

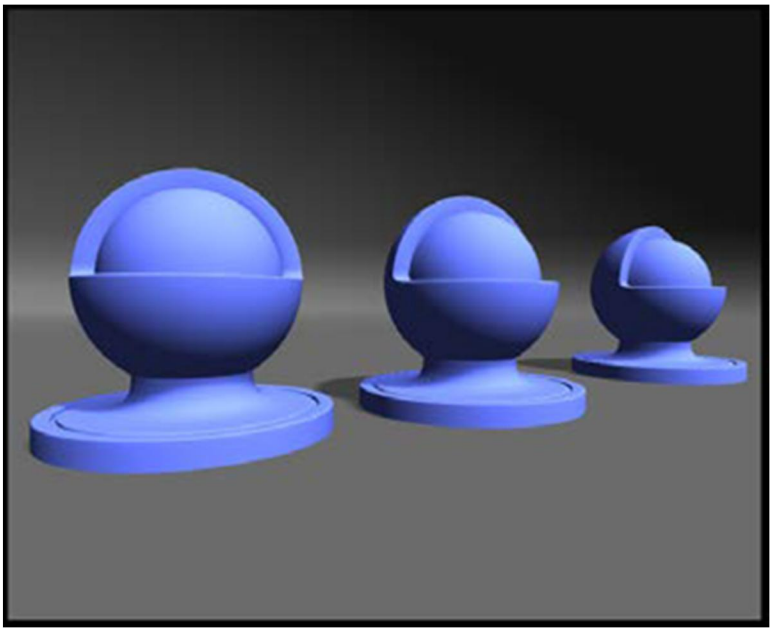
RELEASE NOTES

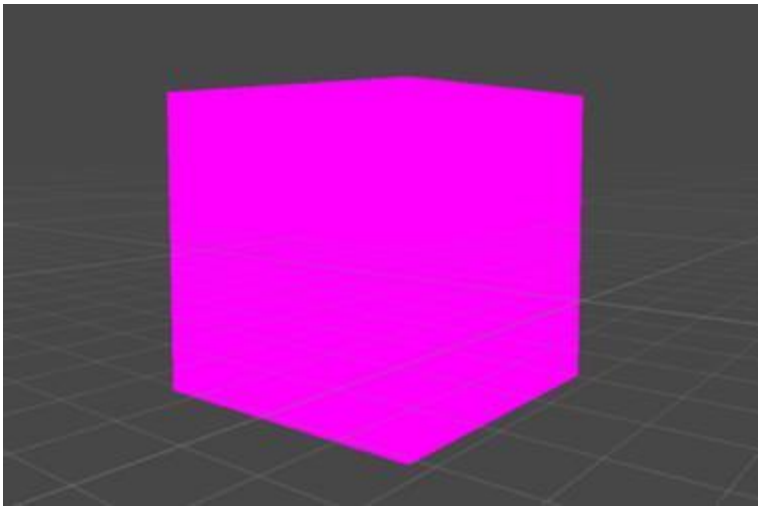


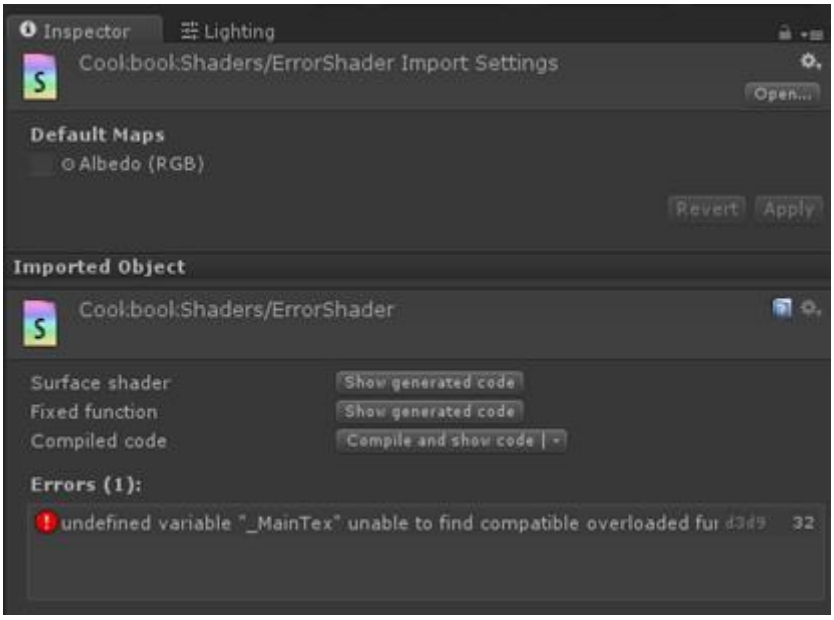



```
Properties
{
    _AmbientColor ("Ambient Color", Color) = (1,1,1,1)
}
```

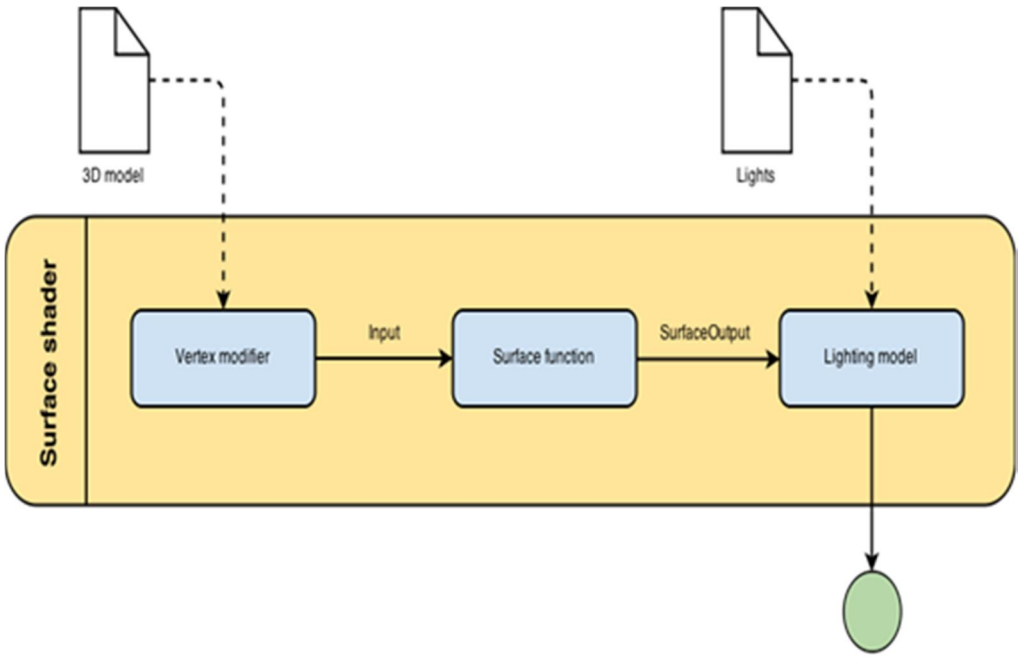
Variable Name | Inspector GUI Name | Type | Default Value

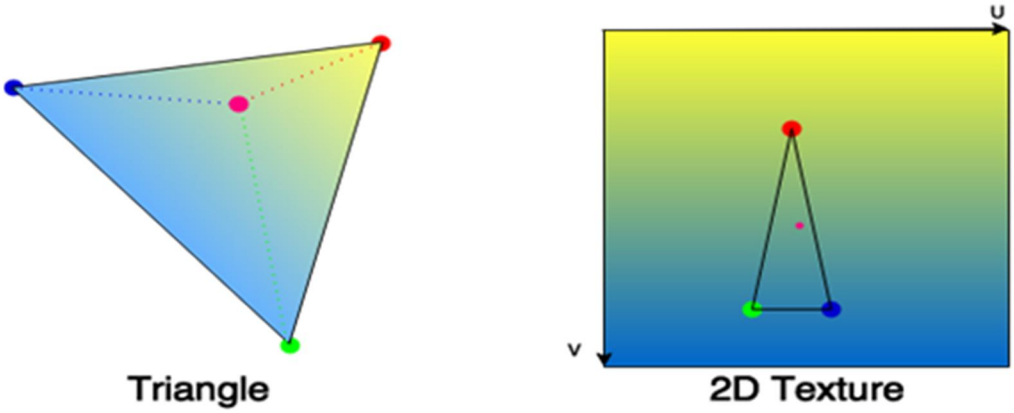




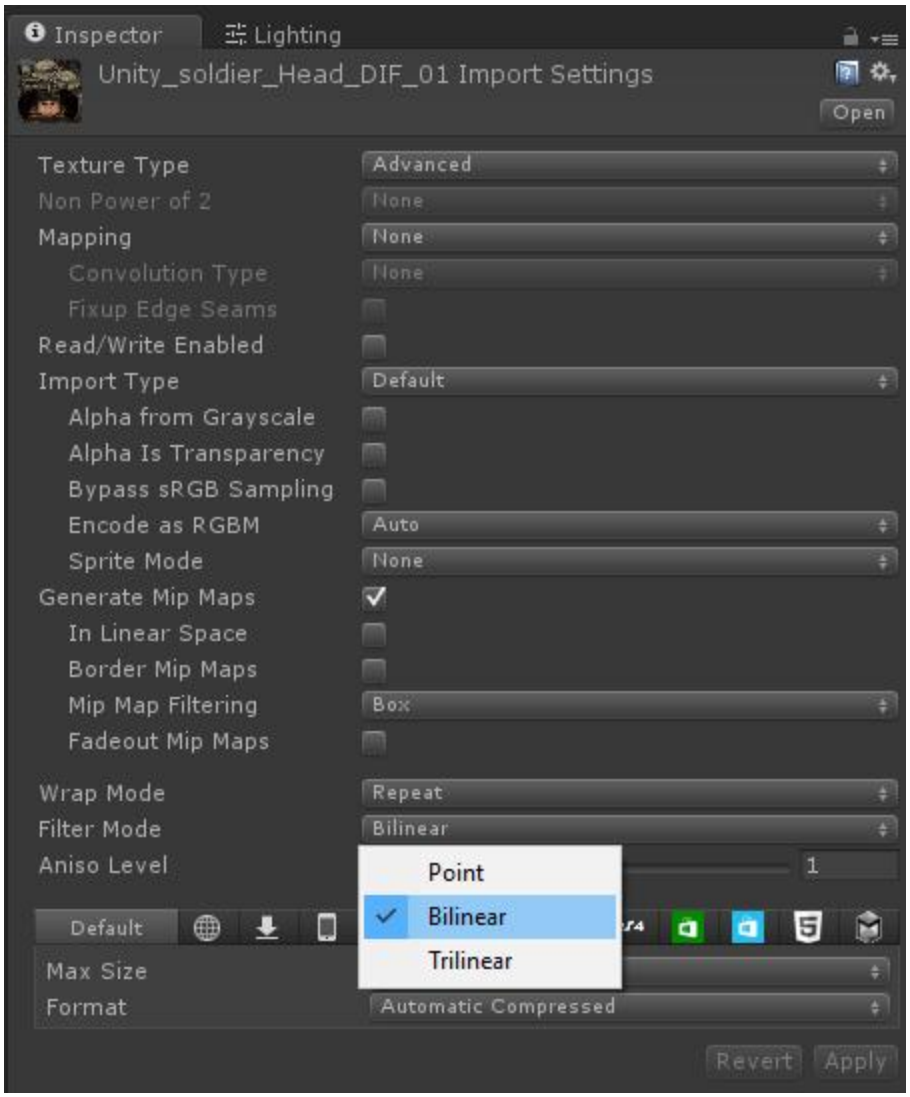


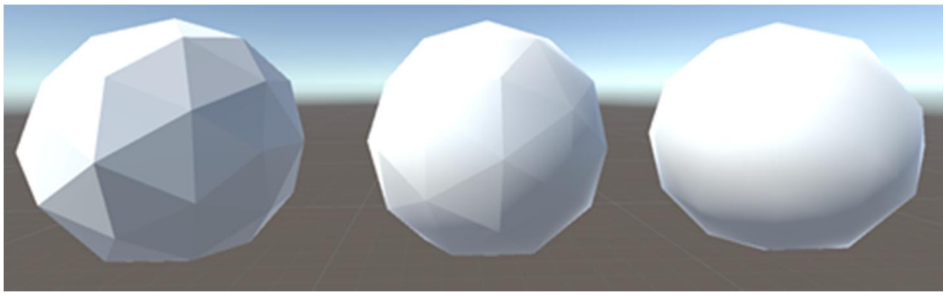
Chapter 2: Surface shaders and texture mapping

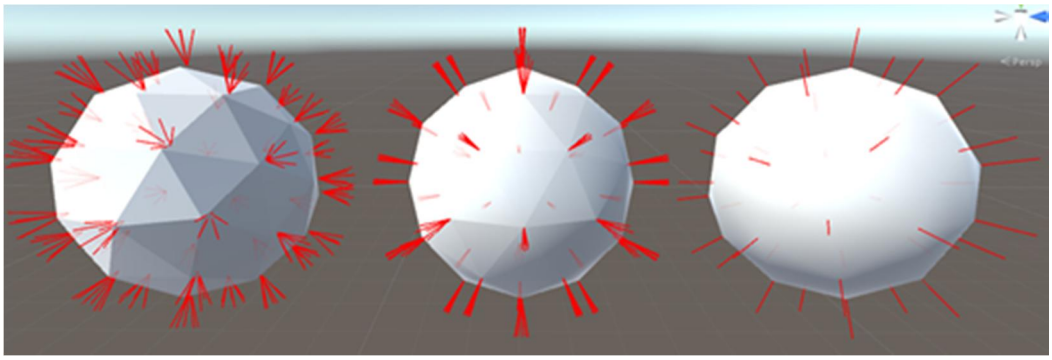


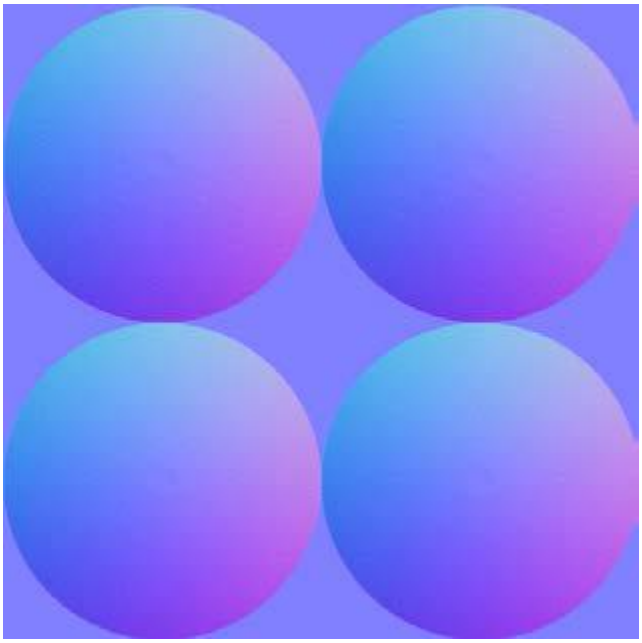

U
V
Triangle
2D Texture

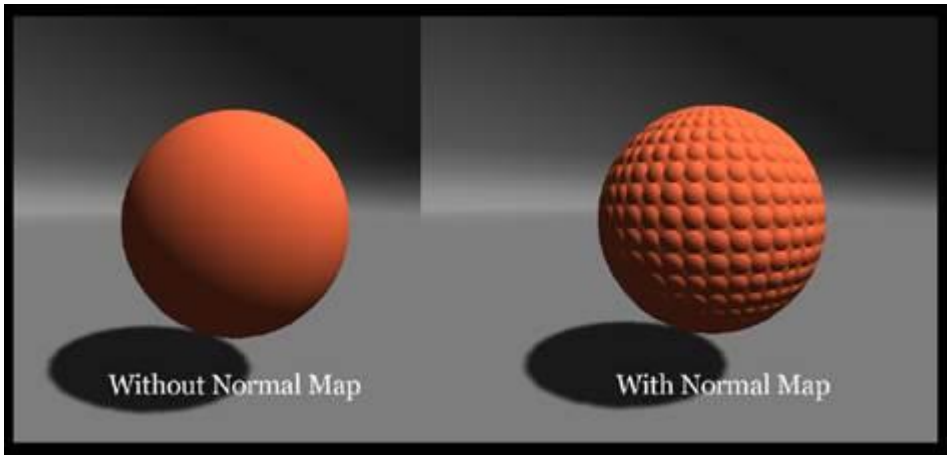

Inspector
Lighting
TexturedMaterial
Shader
CookbookShaders/TexturedShader
Color
Albedo (RGB)
Select
Tiling
X 1
Y 1
Offset
X 0
Y 0
Smoothness
0.181
Metallic
0

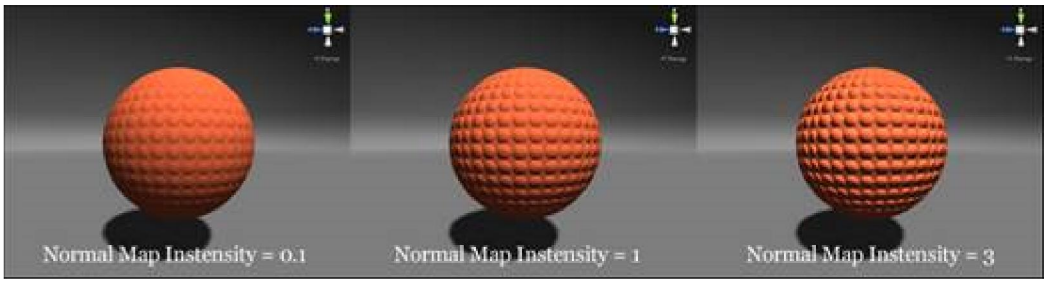

Inspector
Lighting
Unity_soldier_Head_DIF_01 Import Settings
Open
Texture Type
Advanced
Non Power of 2
None
Mapping
None
Convolution Type
None
Fixup Edge Seams
Read/Write Enabled
Import Type
Default
Alpha from Grayscale
Alpha Is Transparency
Bypass sRGB Sampling
Encode as RGBM
Auto
Sprite Mode
None
Generate Mip Maps
In Linear Space
Border Mip Maps
Mip Map Filtering
Box
Fadeout Mip Maps
Wrap Mode
Repeat
Filter Mode
Bilinear
Aniso Level
1
Point
Bilinear
Trilinear
Default
Max Size
Format
Automatic Compressed
Revert
Apply

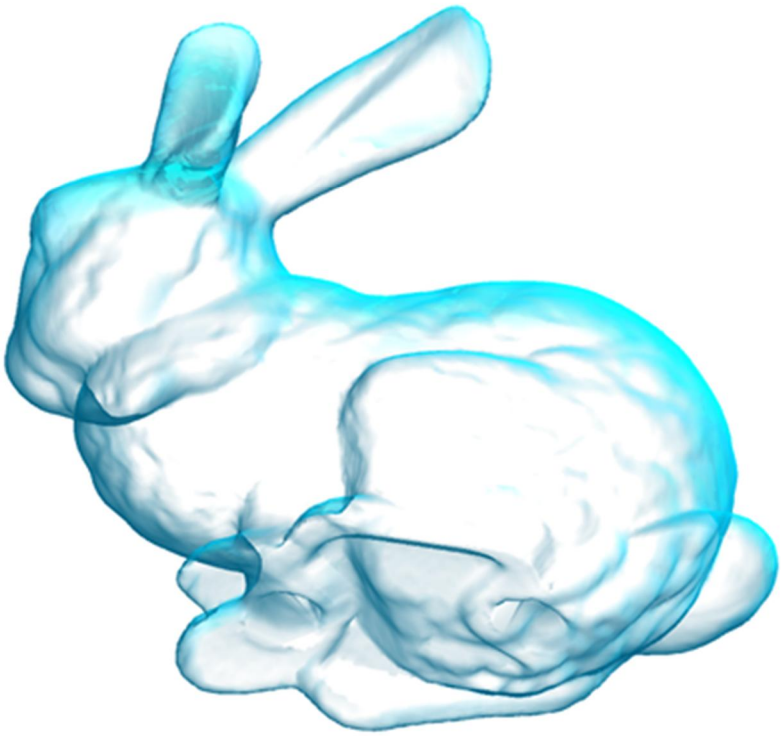


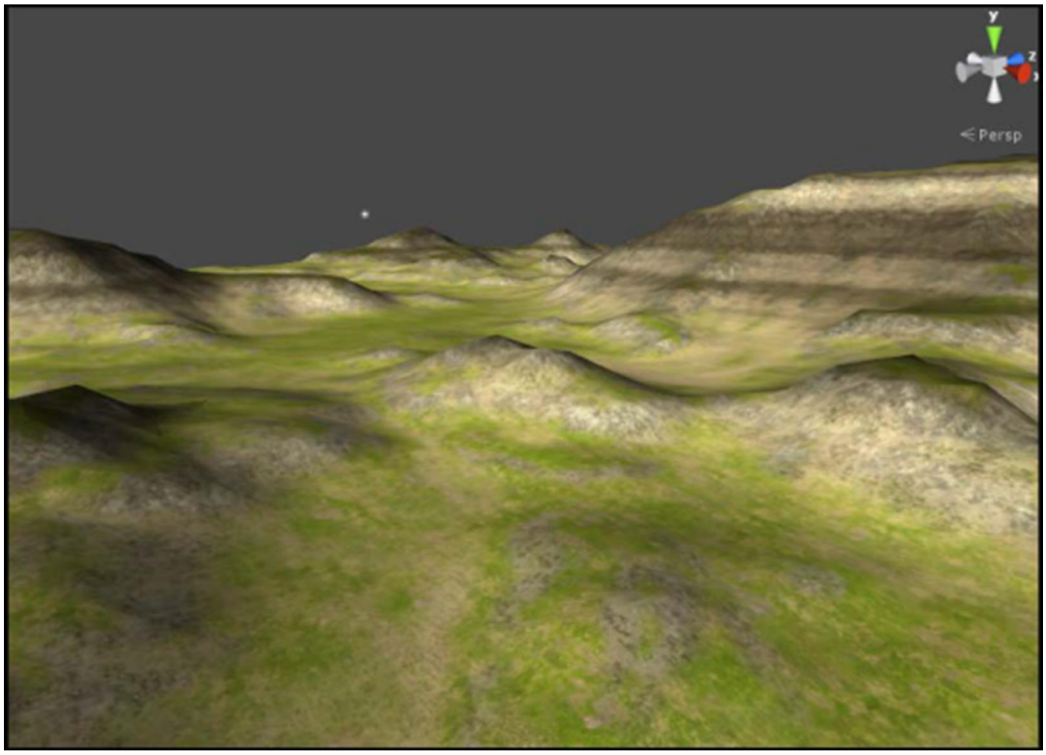


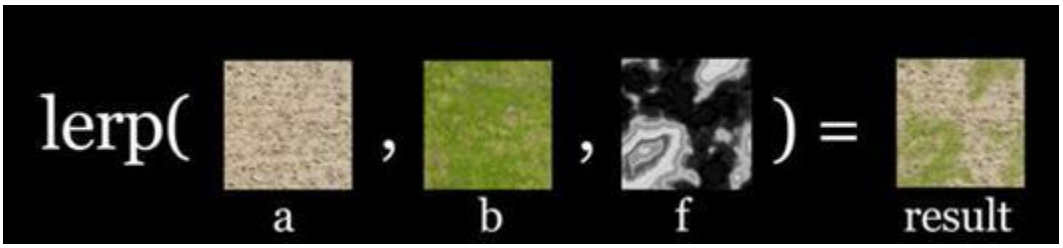


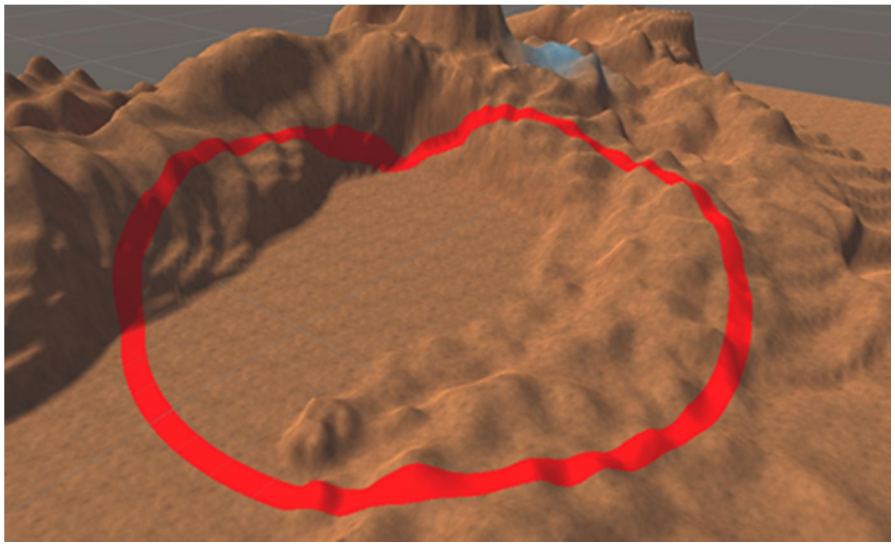

Without Normal Map
With Normal Map

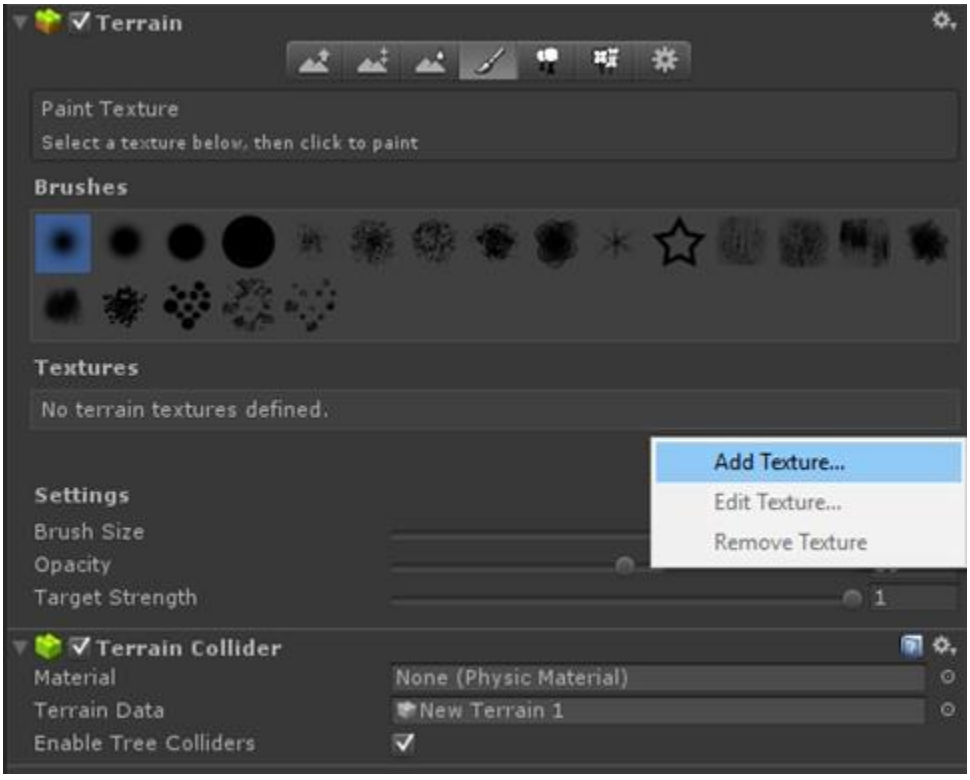

Normal Map Instensity = 0.1
Normal Map Instensity = 1
Normal Map Instensity = 3



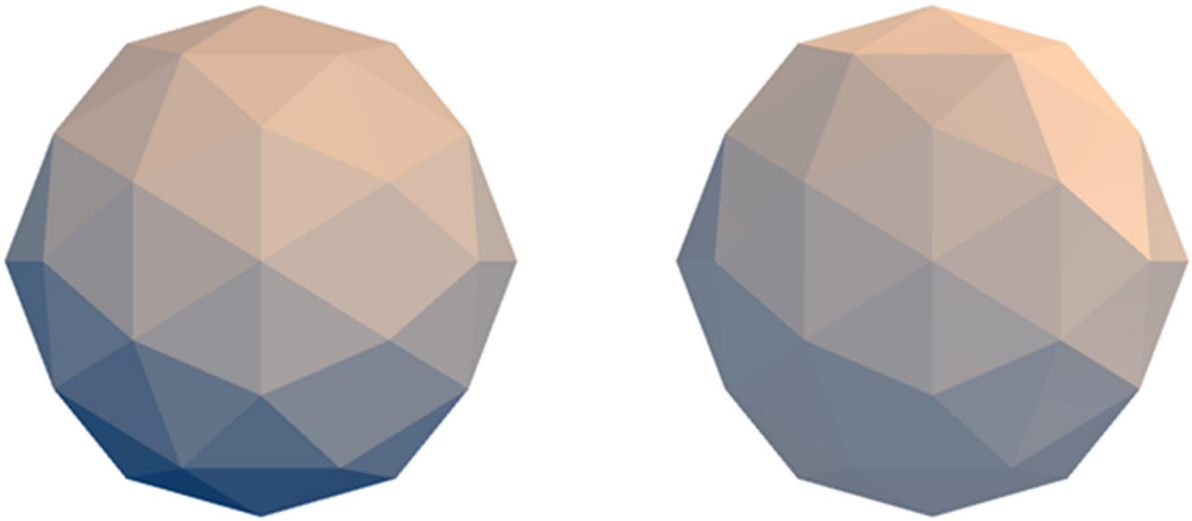

y
z
Persp

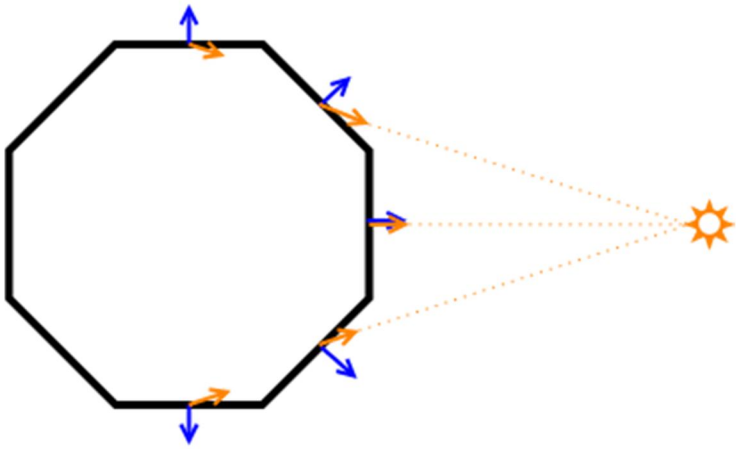

lerp(, ,) =
a
b
f
result

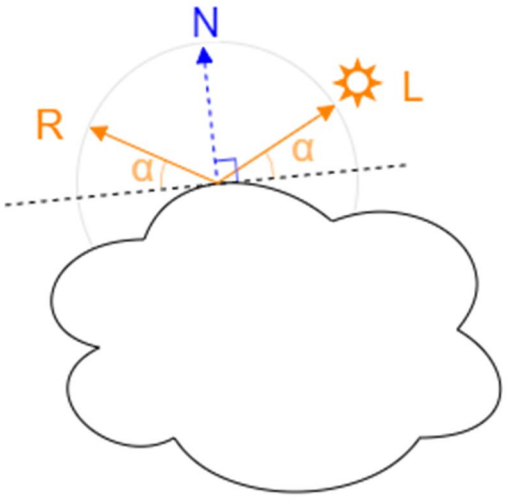



Terrain
Paint Texture
Select a texture below, then click to paint
Brushes
Textures
No terrain textures defined.
Add Texture...
Edit Texture...
Remove Texture
Settings
Brush Size
Opacity
Target Strength
1
Terrain Collider
Material
None (Physic Material)
Terrain Data
New Terrain 1
Enable Tree Colliders

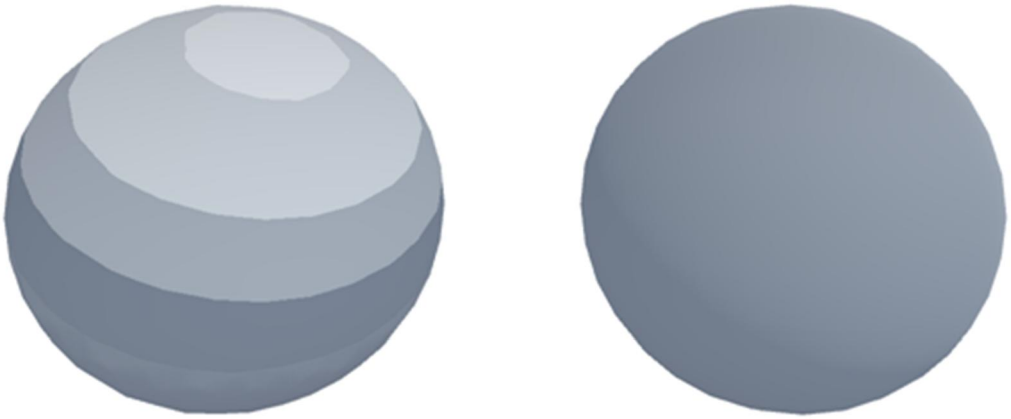
Chapter 3: Understanding lighting models

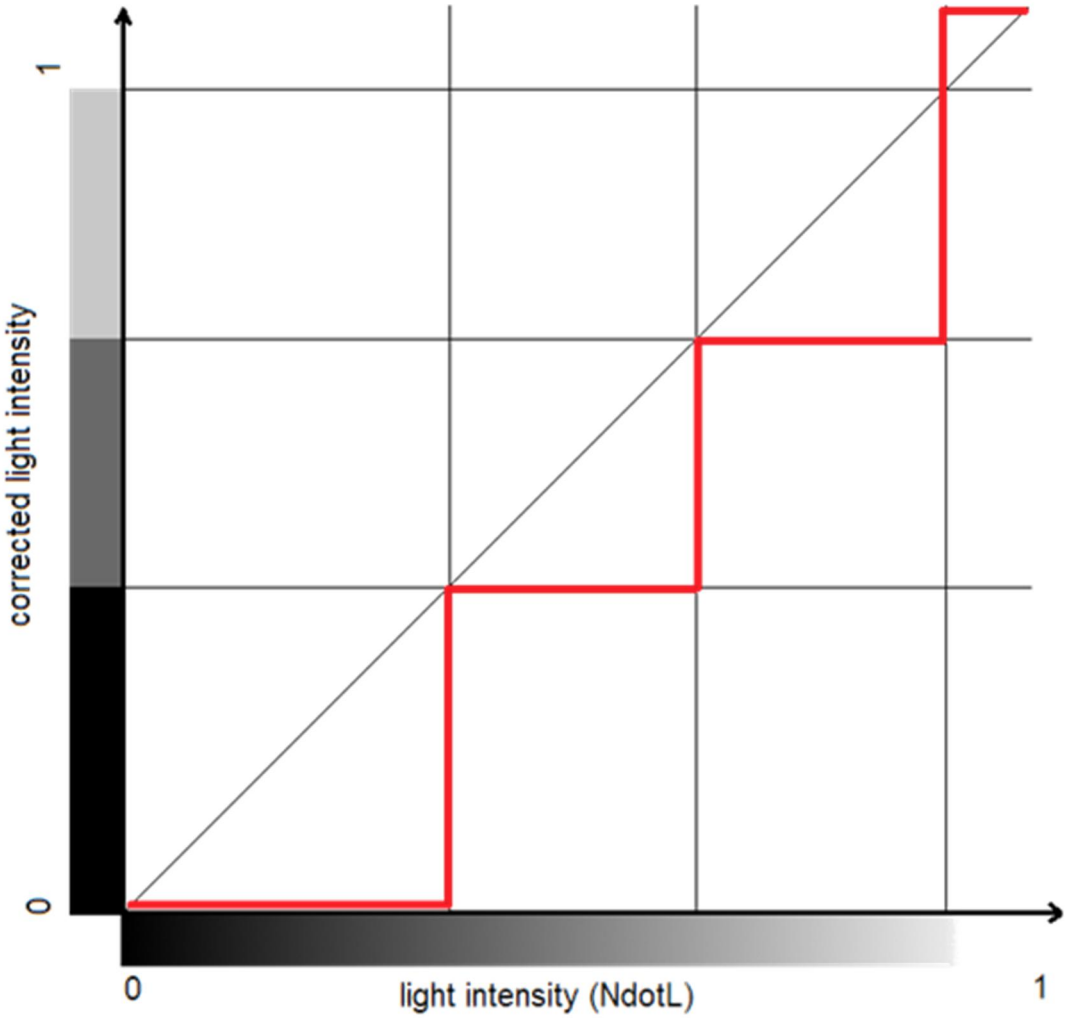


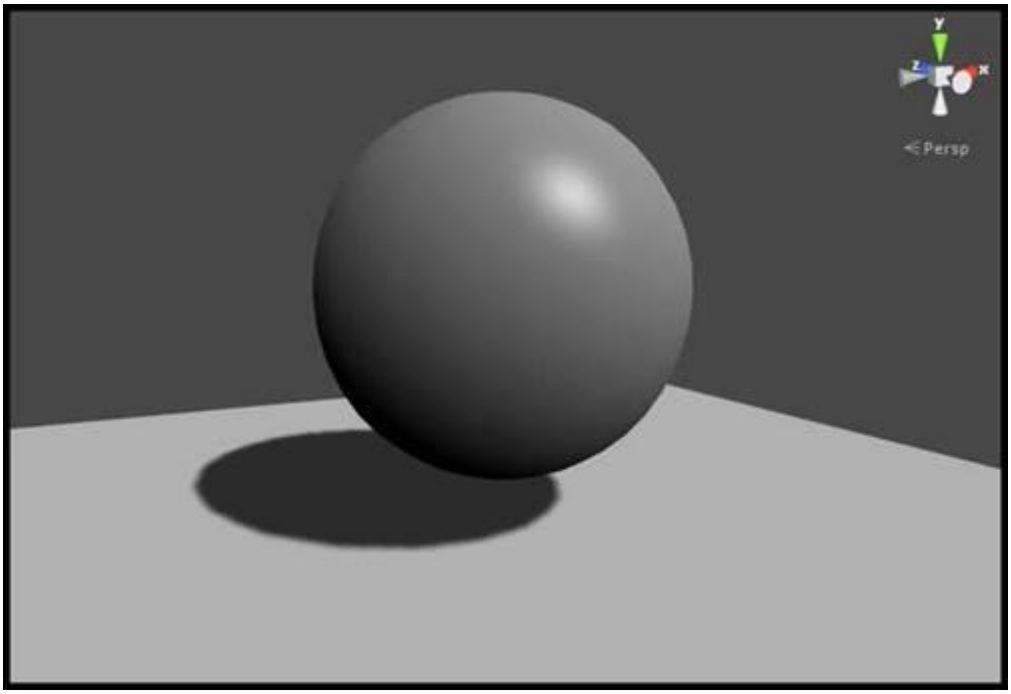


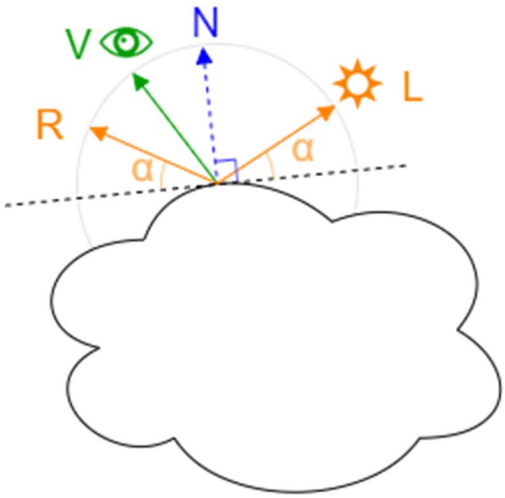


$$I = N \cdot L$$




1
corrected light intensity
0
0
light intensity (NdotL)
1

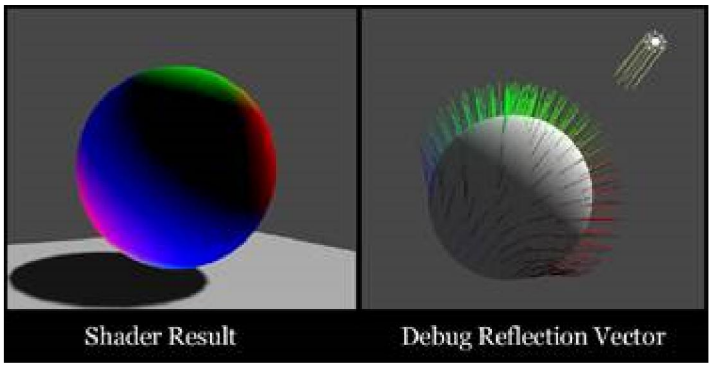


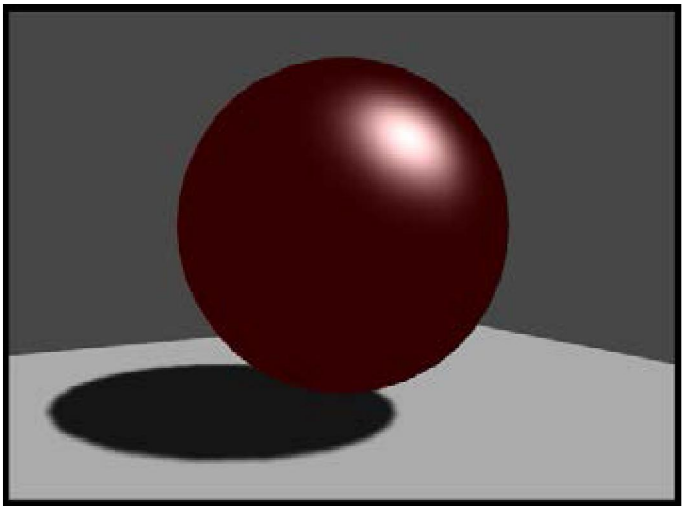


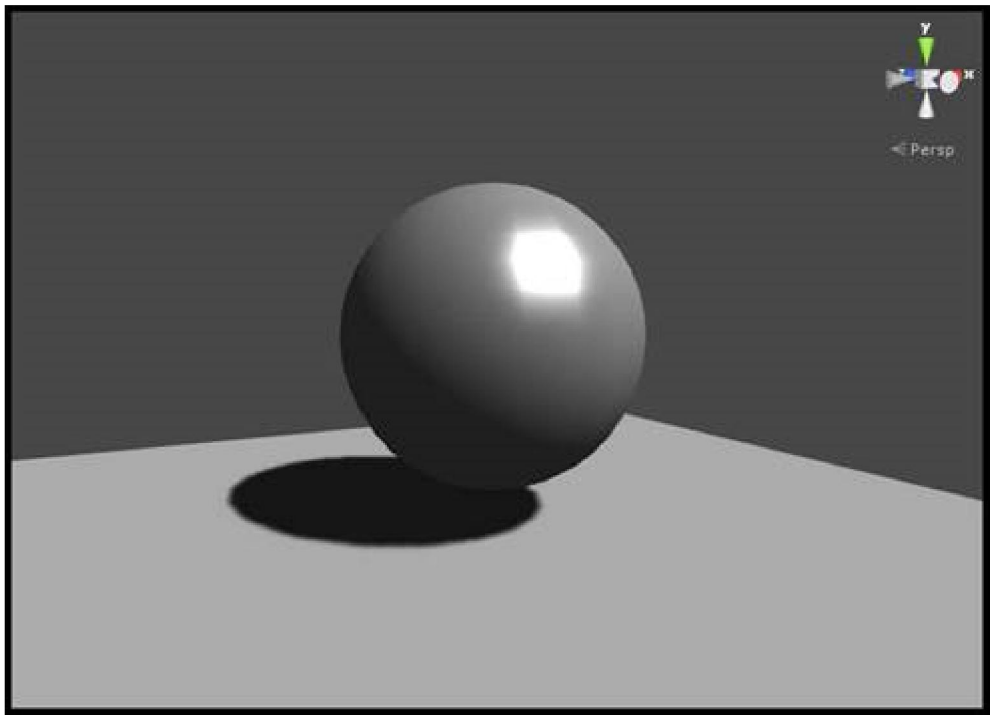
$$I = D + SD = N \cdot L$$

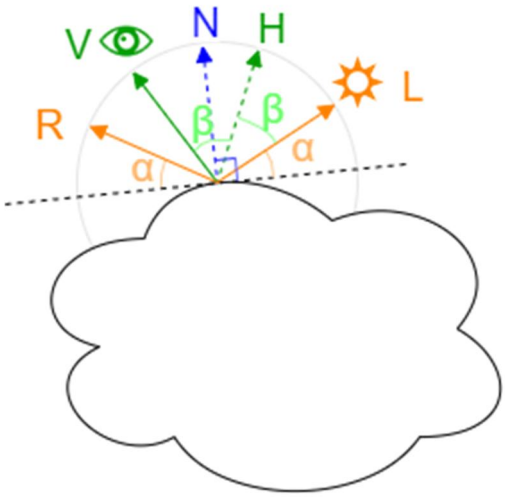
$$S = (R \cdot V)^p$$

$$R = 2N \cdot (N \cdot L) - L$$


Shader Result
Debug Reflection Vector




Y
Persp



$$S_{Phong} = (R \cdot V)^p, \qquad S_{BlinnPhong} = (N \cdot H)^p$$

$$H = \frac{V + L}{|V + L|}$$

$$|V + L|$$

$$V + L$$

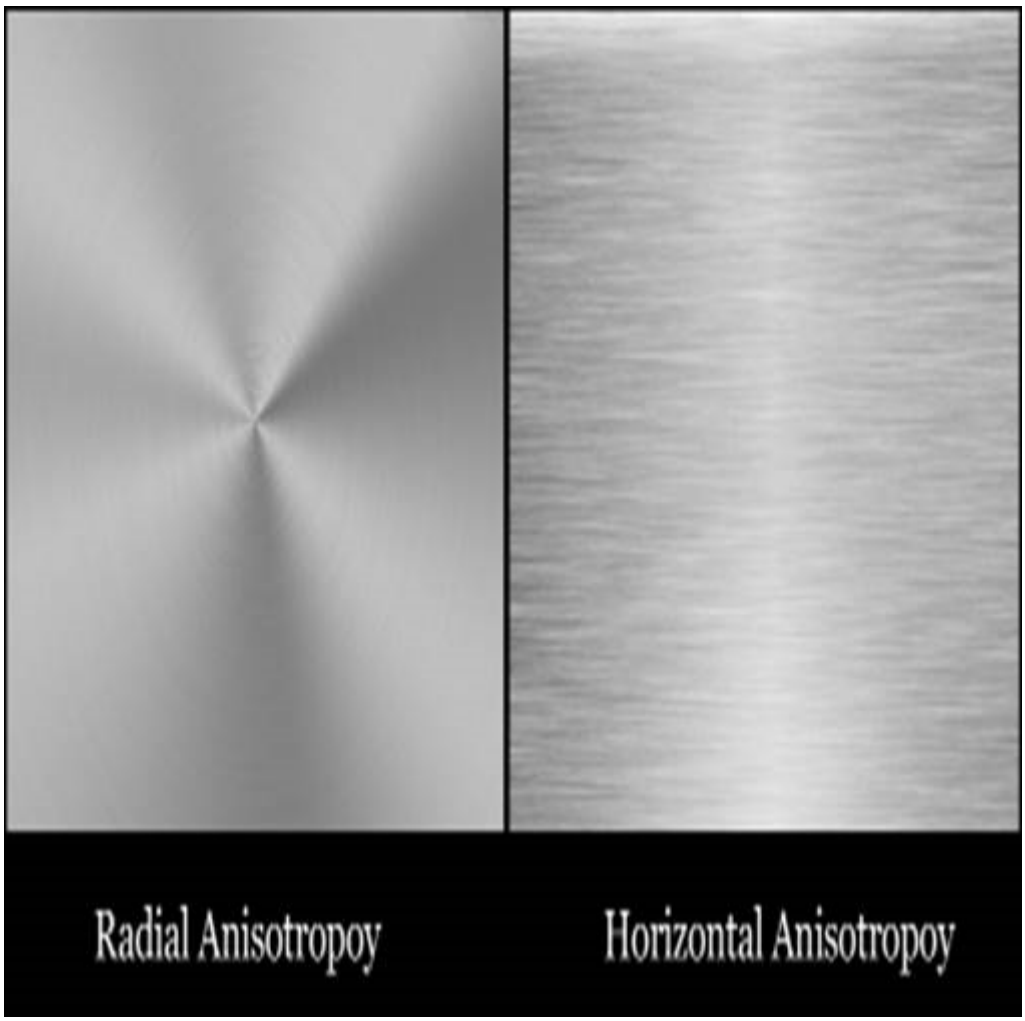
$$I = N \cdot L$$

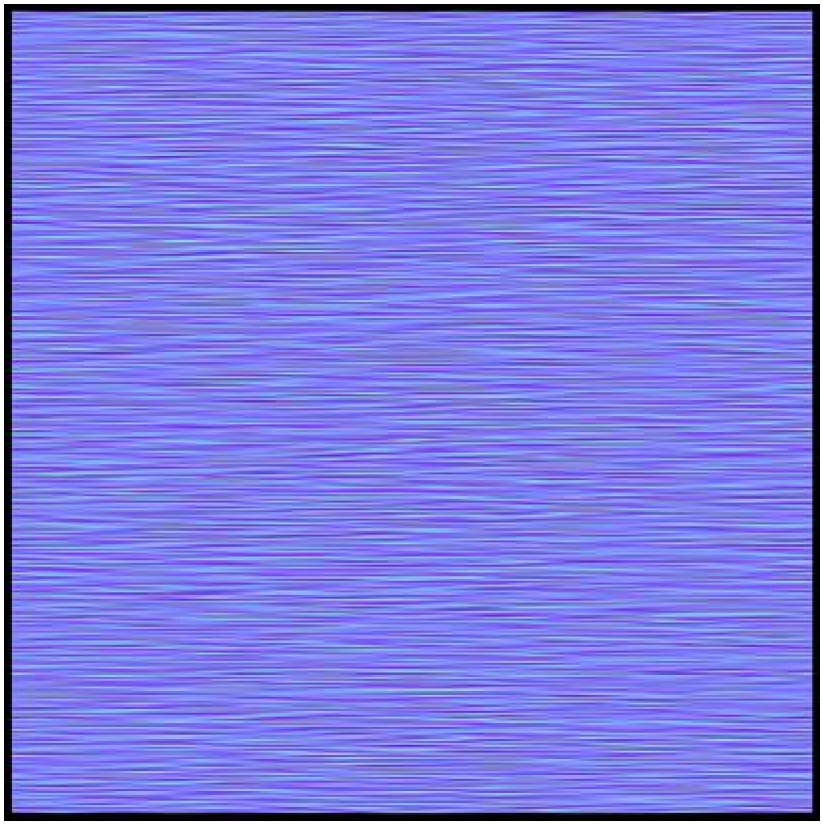
$$I = N \cdot L + (R \cdot V)^p$$

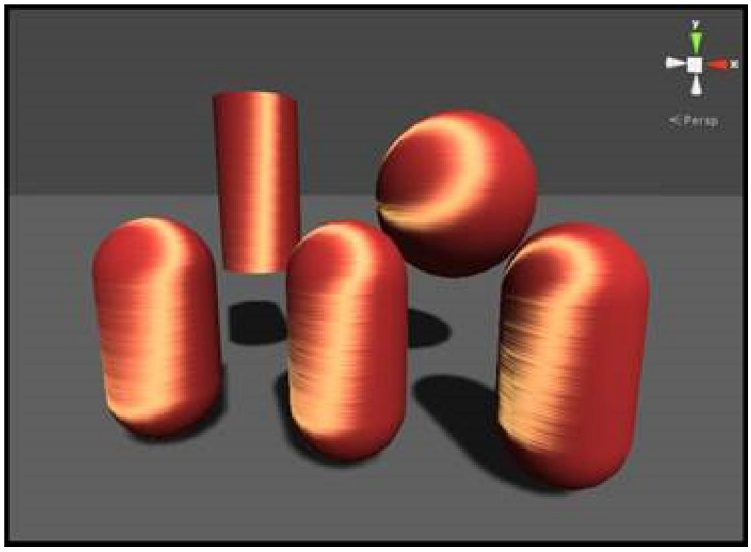
$$R = 2N \cdot (N \cdot L) - L$$

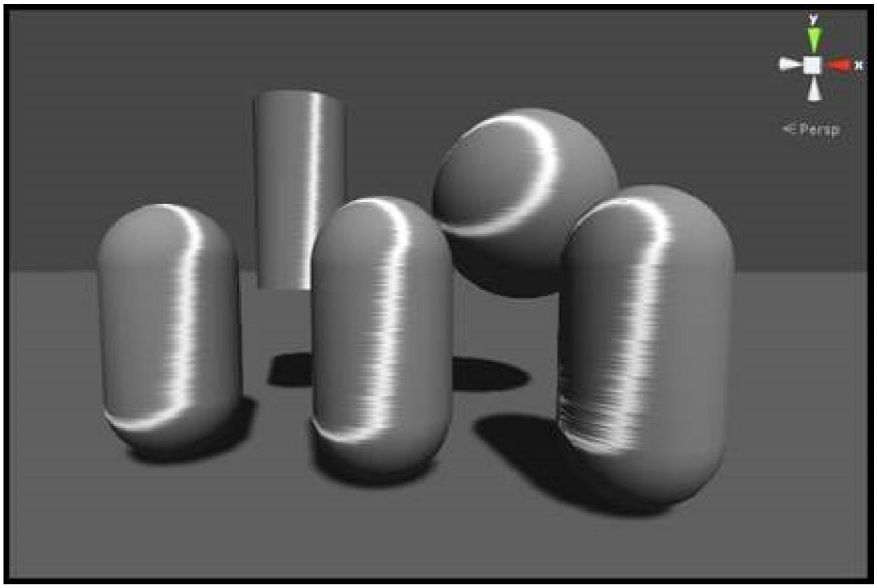
$$I = N \cdot L + (N \cdot H)^p$$

$$H = \frac{V + L}{|V + L|}$$

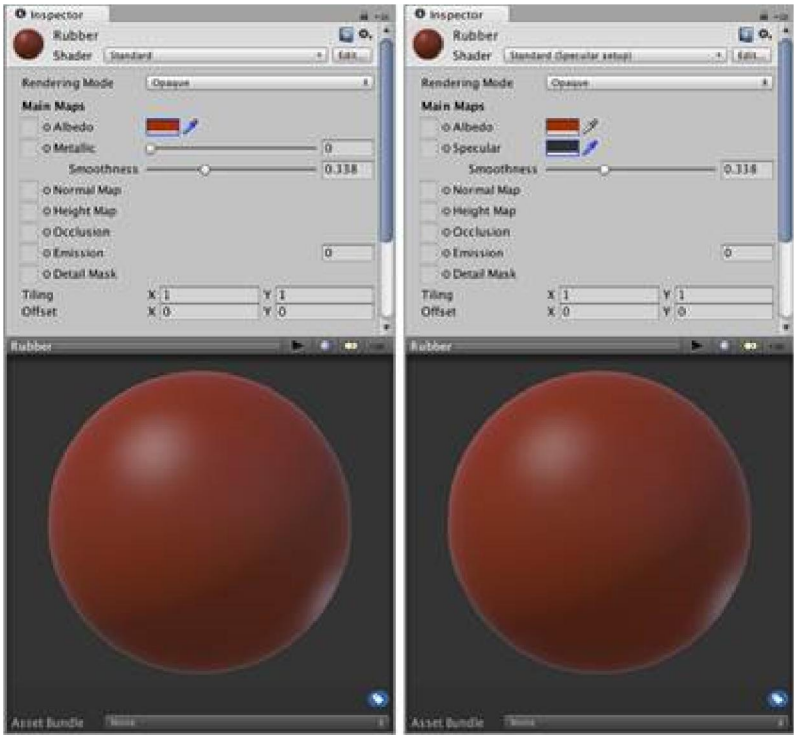

Radial Anisotropoy
Horizontal Anisotropoy

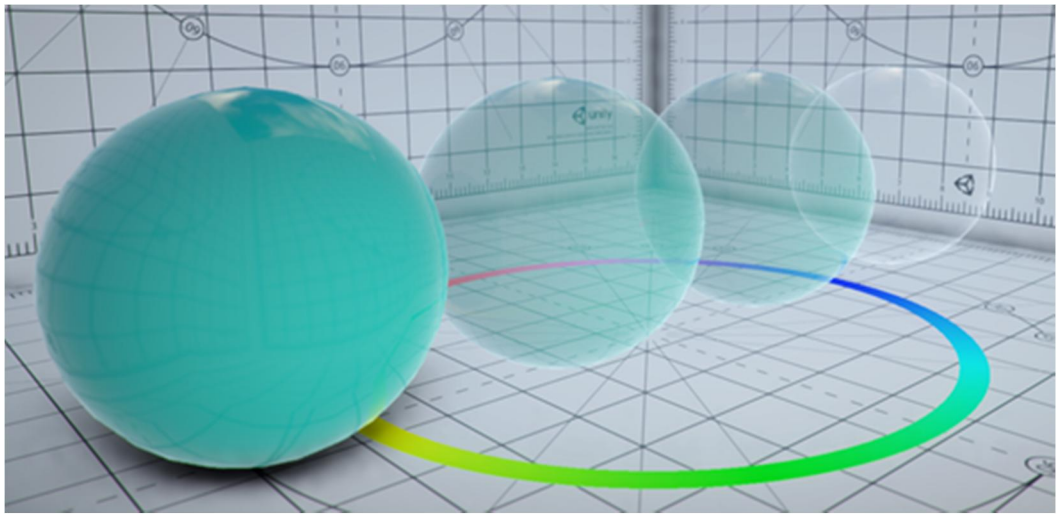


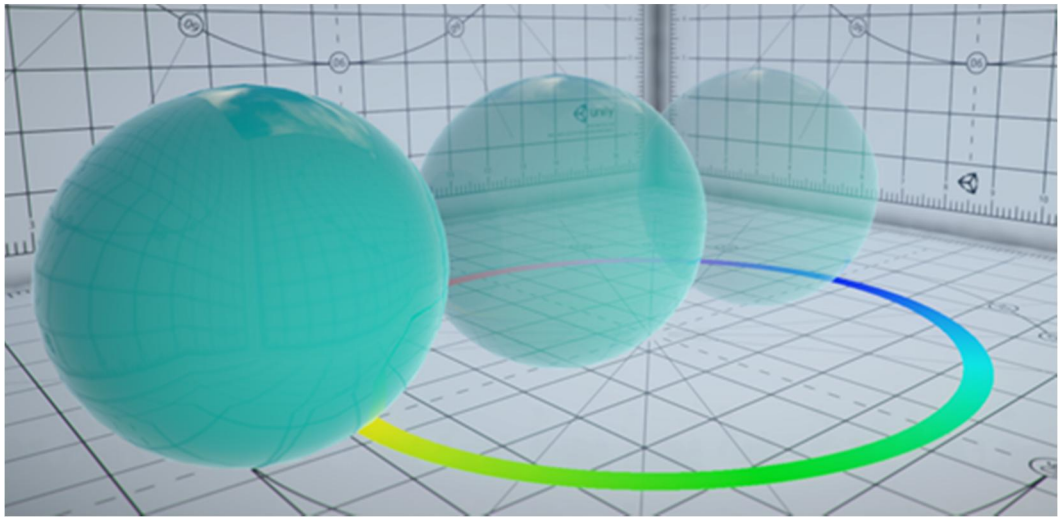

Persp

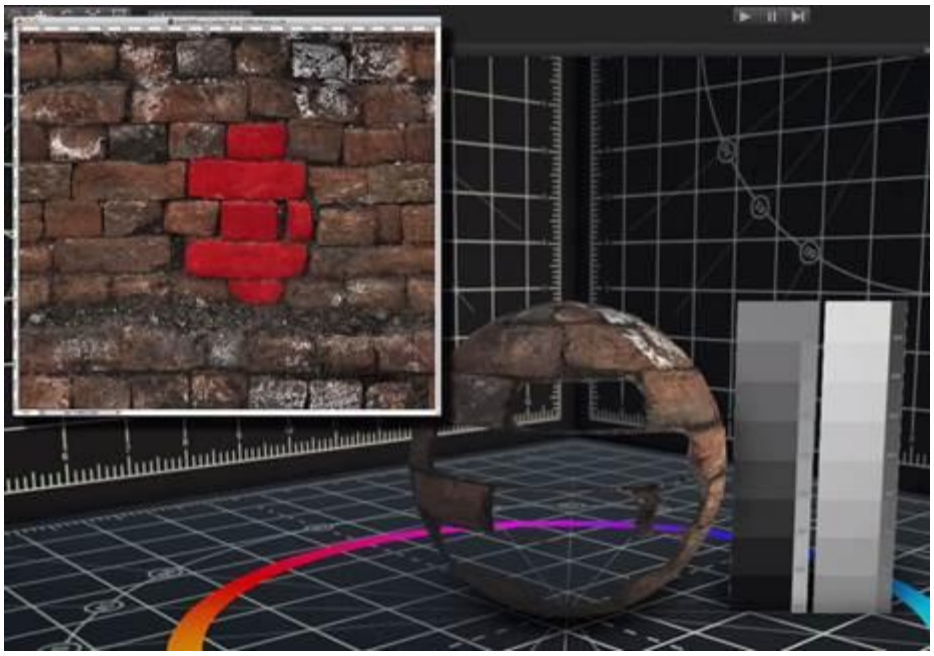


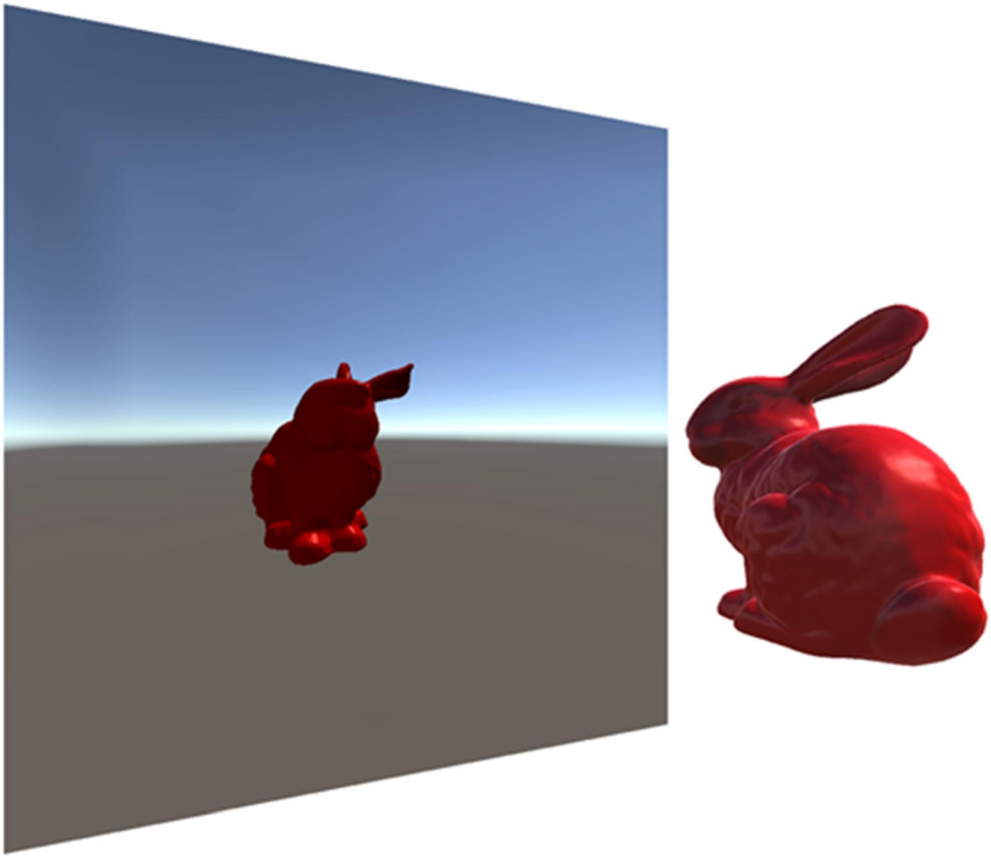
Chapter 4: Physically Based Rendering in Unity5

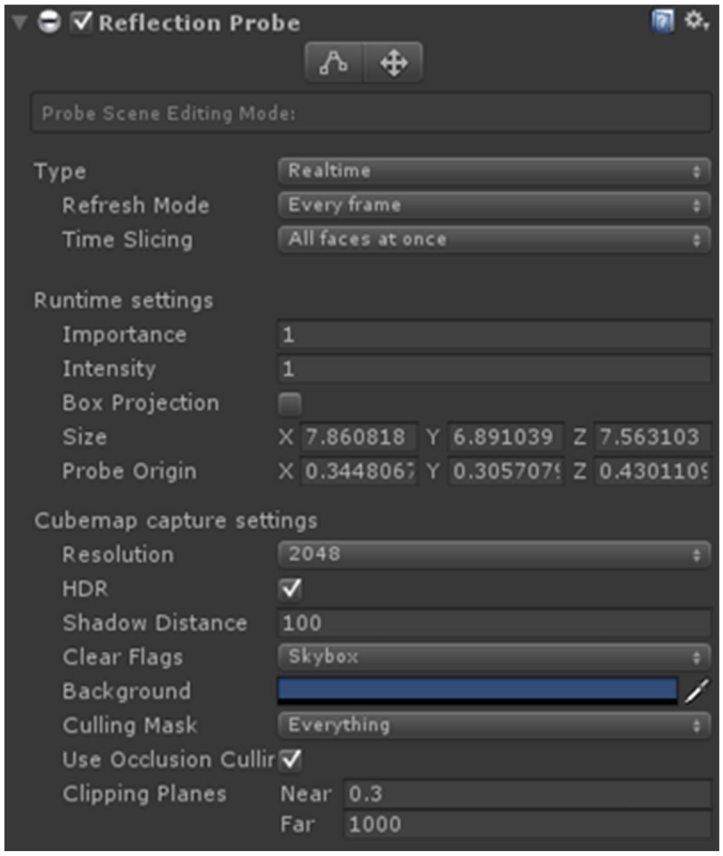


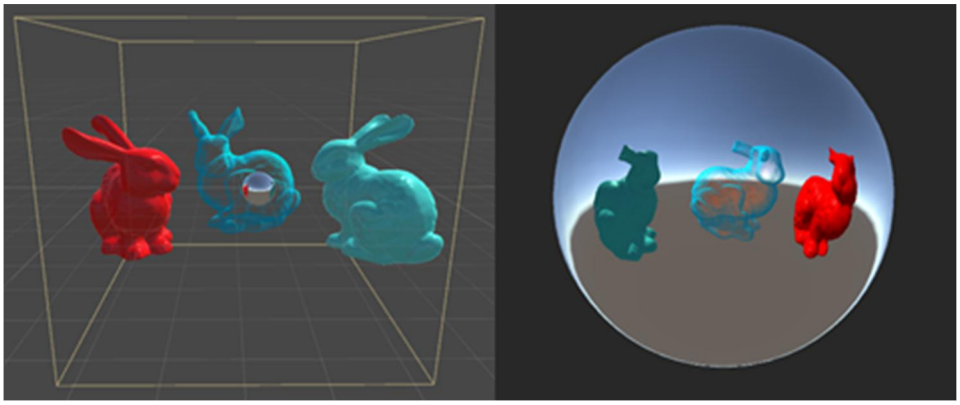


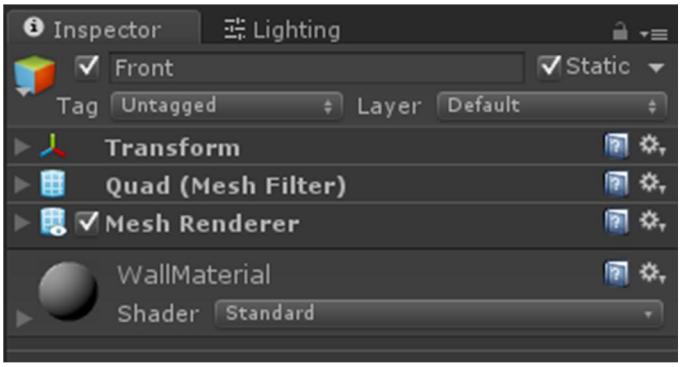


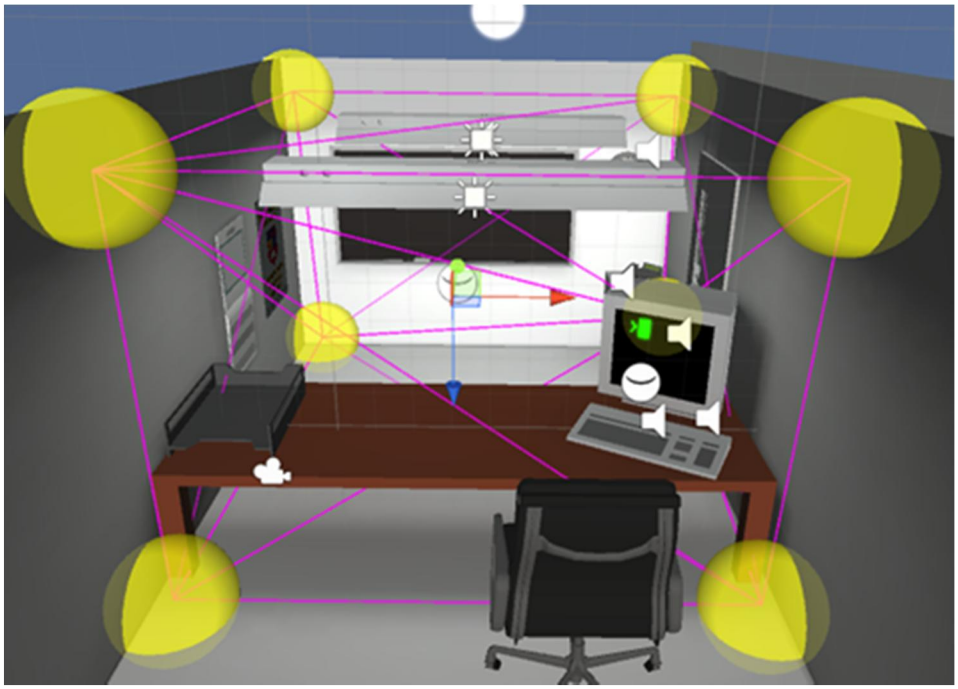


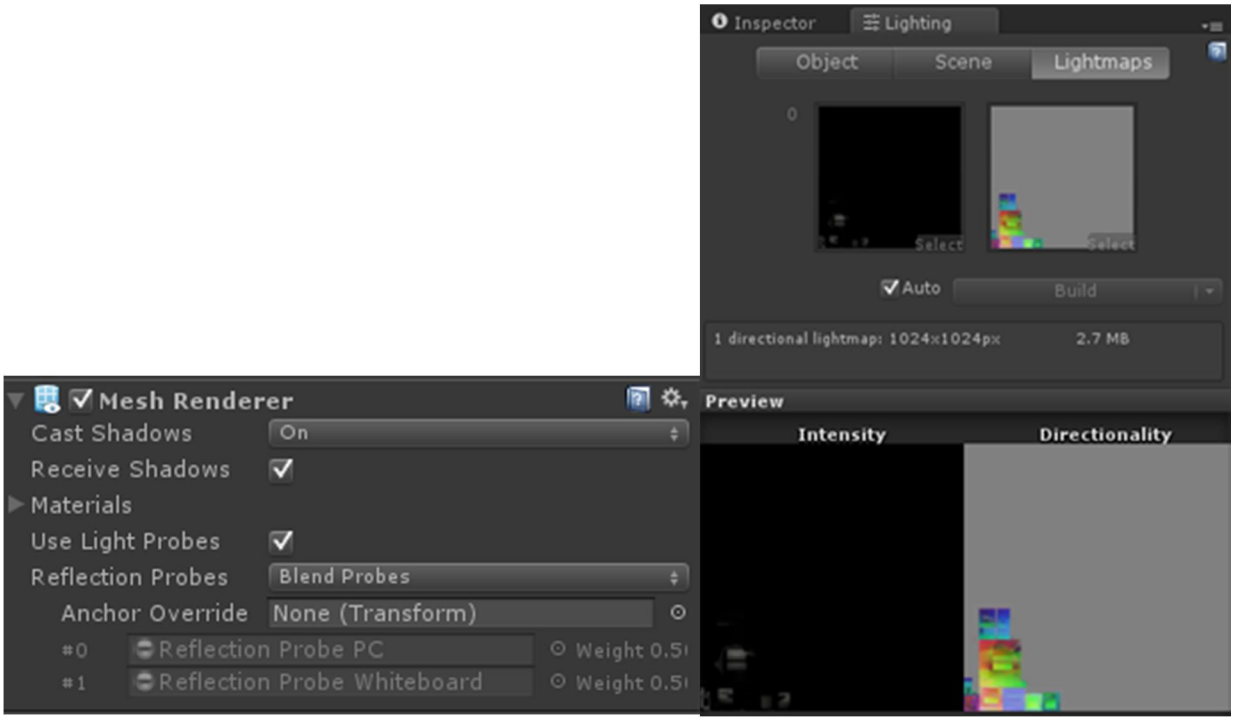



Reflection Probe
Probe Scene Editing Mode:
Type
Realtime
Refresh Mode
Every frame
Time Slicing
All faces at once
Runtime settings
Importance
1
Intensity
1
Box Projection
Size
X 7.860818 Y 6.891039 Z 7.563103
Probe Origin
Cubemap capture settings
Resolution
2048
HDR
Shadow Distance
100
Clear Flags
Skybox
Background
Culling Mask
Everything
Use Occlusion Cullir
Clipping Planes
Near 0.3
Far 1000

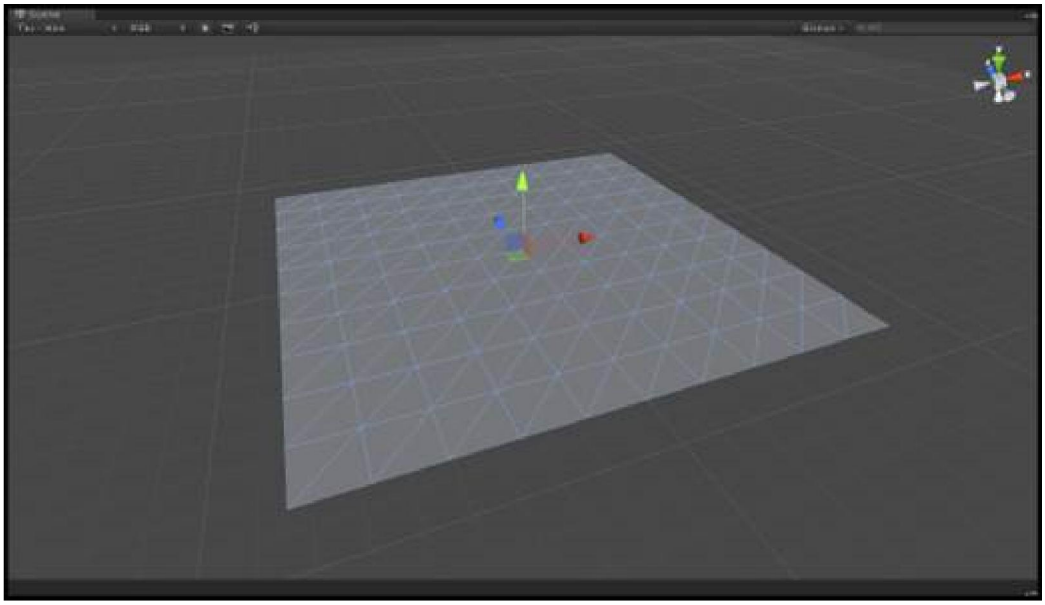


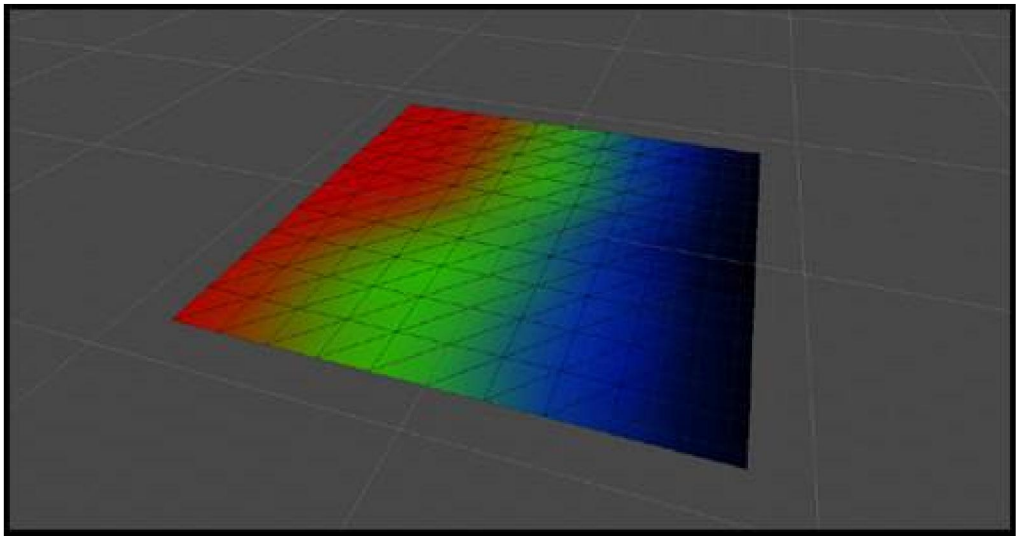

Inspector
Lighting
Front
Static
Tag
Untagged
Layer
Default
Transform
Quad (Mesh Filter)
Mesh Renderer
WallMaterial
Shader
Standard

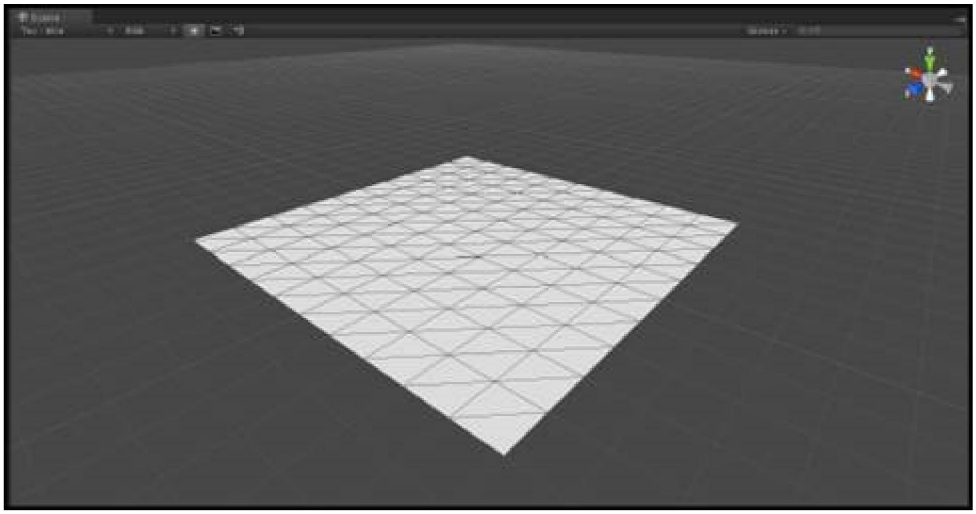


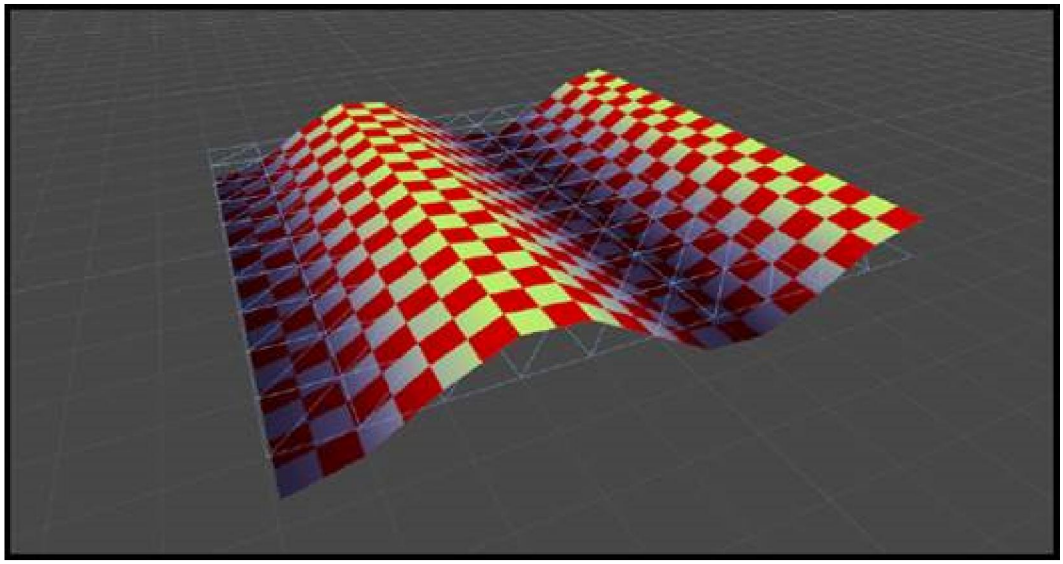

Inspector
Lighting
Object
Scene
Lightmaps
0
Select
Select
Auto
Build
1 directional lightmap: 1024x1024px
2.7 MB
Mesh Renderer
Cast Shadows
On
Receive Shadows
Materials
Use Light Probes
Reflection Probes
Blend Probes
Anchor Override
None (Transform)
#0
Reflection Probe PC
Weight 0.5
#1
Reflection Probe Whiteboard
Weight 0.5
Preview
Intensity
Directionality

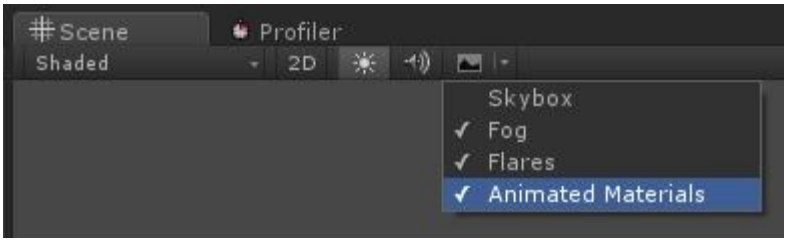
Chapter 5: Vertex modifiers

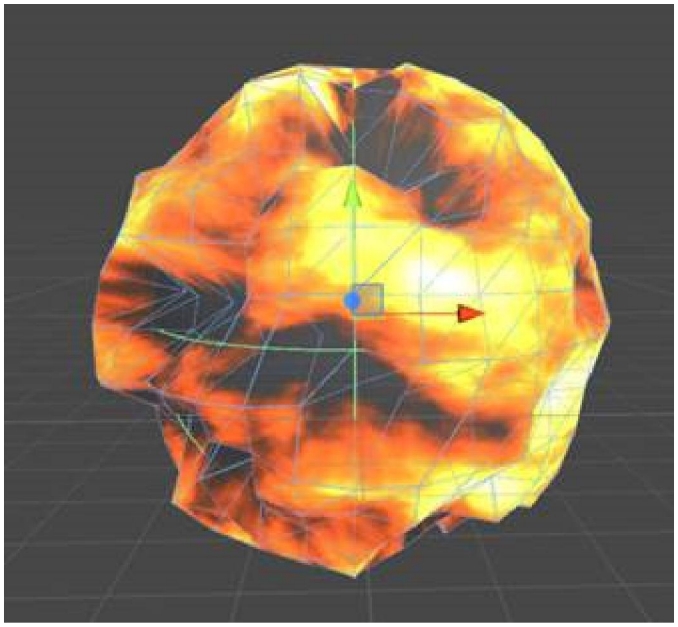




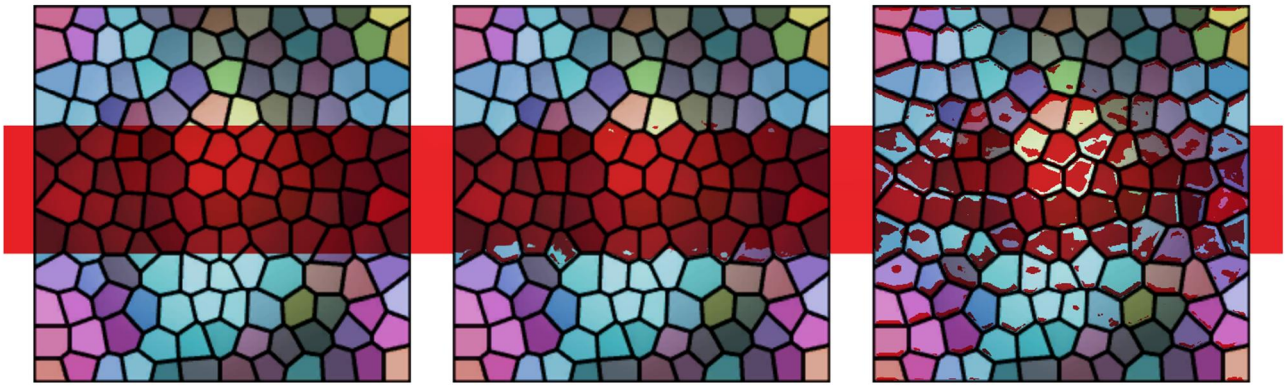


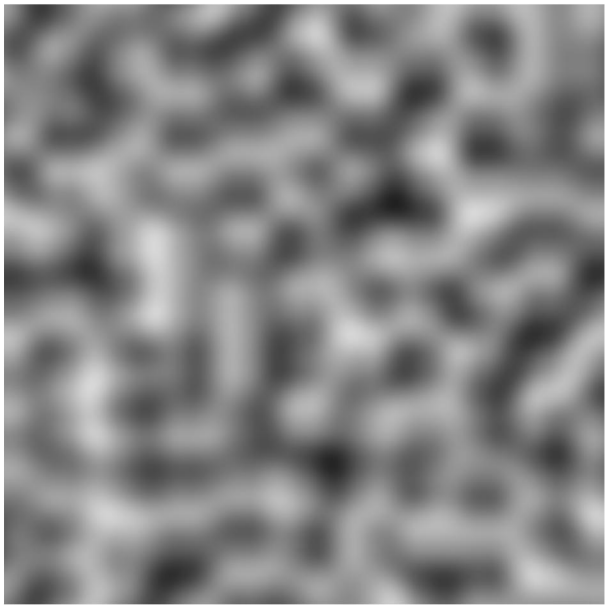



Scene
Profiler
Shaded
2D
Skybox
Fog
Flares
Animated Materials



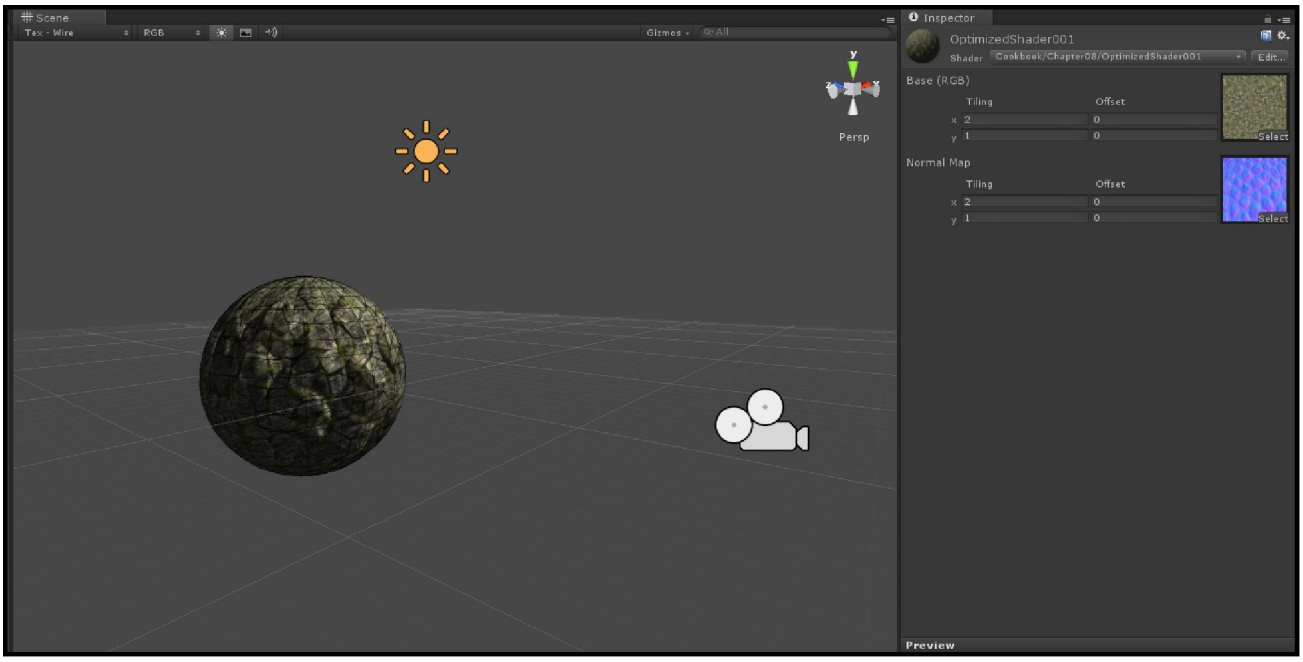
Chapter 6: Fragment shaders: water and glass





Chapter 7: Mobile Shader Adjustment

```
Shader "Cookbook/Chapter08/OptimizedShader001"
{
    Properties
    {
        _MainTex ("Base (RGB)", 2D) = "white" {}
        _NormalMap ("Normal Map", 2D) = "bump" {}
    }

    SubShader
    {
        Tags { "RenderType"="Opaque" }
        LOD 200

        CGPROGRAM
        #pragma surface surf SimpleLambert

        sampler2D _MainTex;
        sampler2D _NormalMap;

        struct Input
        {
            float2 uv_MainTex;
            float2 uv_NormalMap;
        };

        inline float4 LightingSimpleLambert (SurfaceOutput s, float3 lightDir, float atten)
        {
            float diff = max (0, dot (s.Normal, lightDir));

            float4 c;
            c.rgb = s.Albedo * _LightColor0.rgb * (diff * atten * 2);
            c.a = s.Alpha;
            return c;
        }

        void surf (Input IN, inout SurfaceOutput o)
        {
            float4 c = tex2D (_MainTex, IN.uv_MainTex);

            o.Albedo = c.rgb;
            o.Alpha = c.a;
            o.Normal = UnpackNormal(tex2D(_NormalMap, IN.uv_NormalMap));
        }
        ENDCG
    }
    FallBack "Diffuse"
}
```




```
struct Input
{
    half2 uv_MainTex;
    half2 uv_NormalMap;
};
```

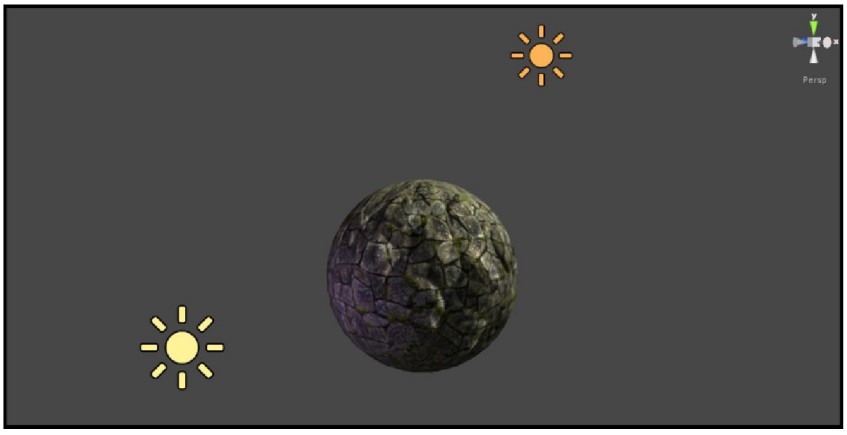
```
inline fixed4 LightingSimpleLambert (SurfaceOutput s, fixed3 lightDir, fixed atten)
{
    fixed diff = max (0, dot (s.Normal, lightDir));

    fixed4 c;
    c.rgb = s.Albedo * _LightColor0.rgb * (diff * atten * 2);
    c.a = s.Alpha;
    return c;
}
```

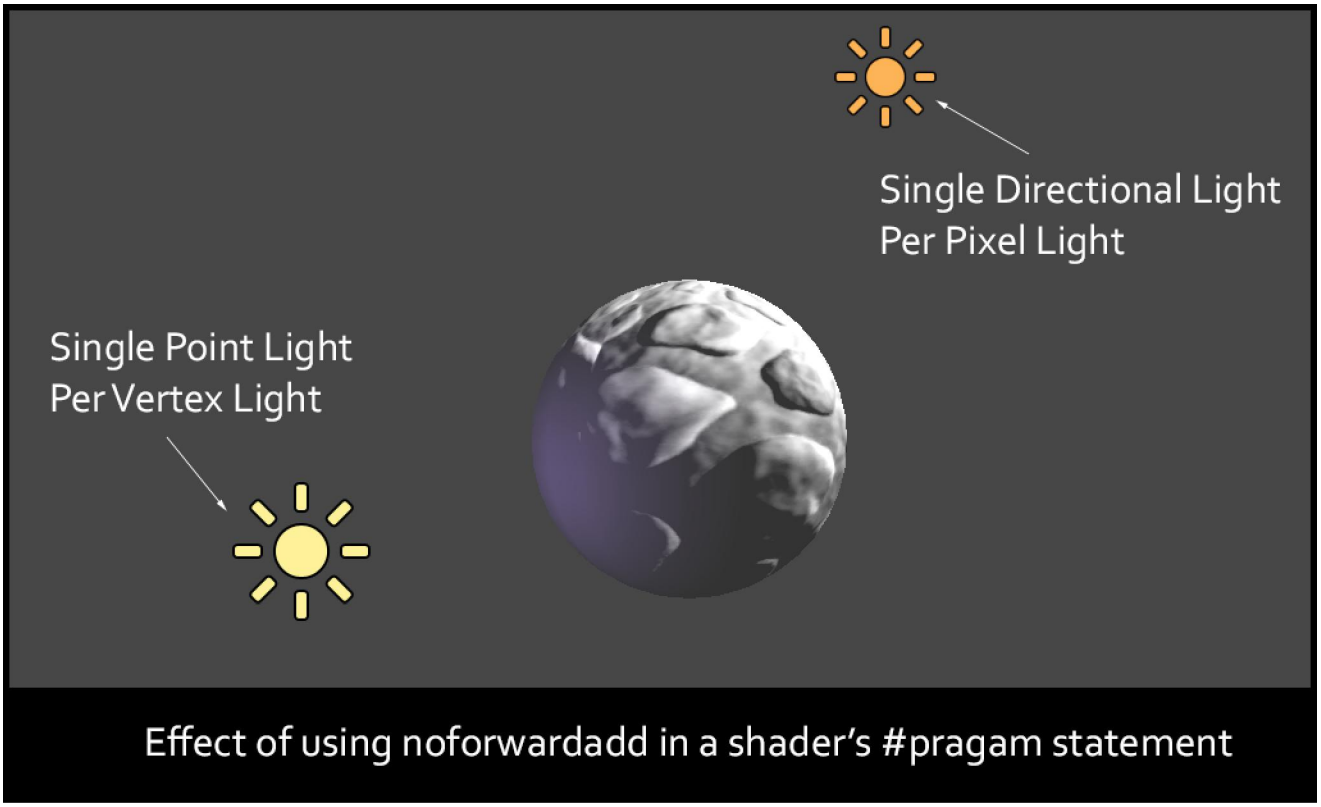
```
void surf (Input IN, inout SurfaceOutput o)
{
    fixed4 c = tex2D (_MainTex, IN.uv_MainTex);

    o.Albedo = c.rgb;
    o.Alpha = c.a;
    o.Normal = UnpackNormal(tex2D(_NormalMap, IN.uv_NormalMap));
}
```

```
CGPROGRAM
#pragma surface surf SimpleLambert noforwardadd


void surf (Input IN, inout SurfaceOutput o)
{
    fixed4 c = tex2D (_MainTex, IN.uv_MainTex);

    o.Albedo = c.rgb;
    o.Alpha = c.a;
    o.Normal = UnpackNormal(tex2D(_NormalMap, IN.uv_MainTex));
}
```

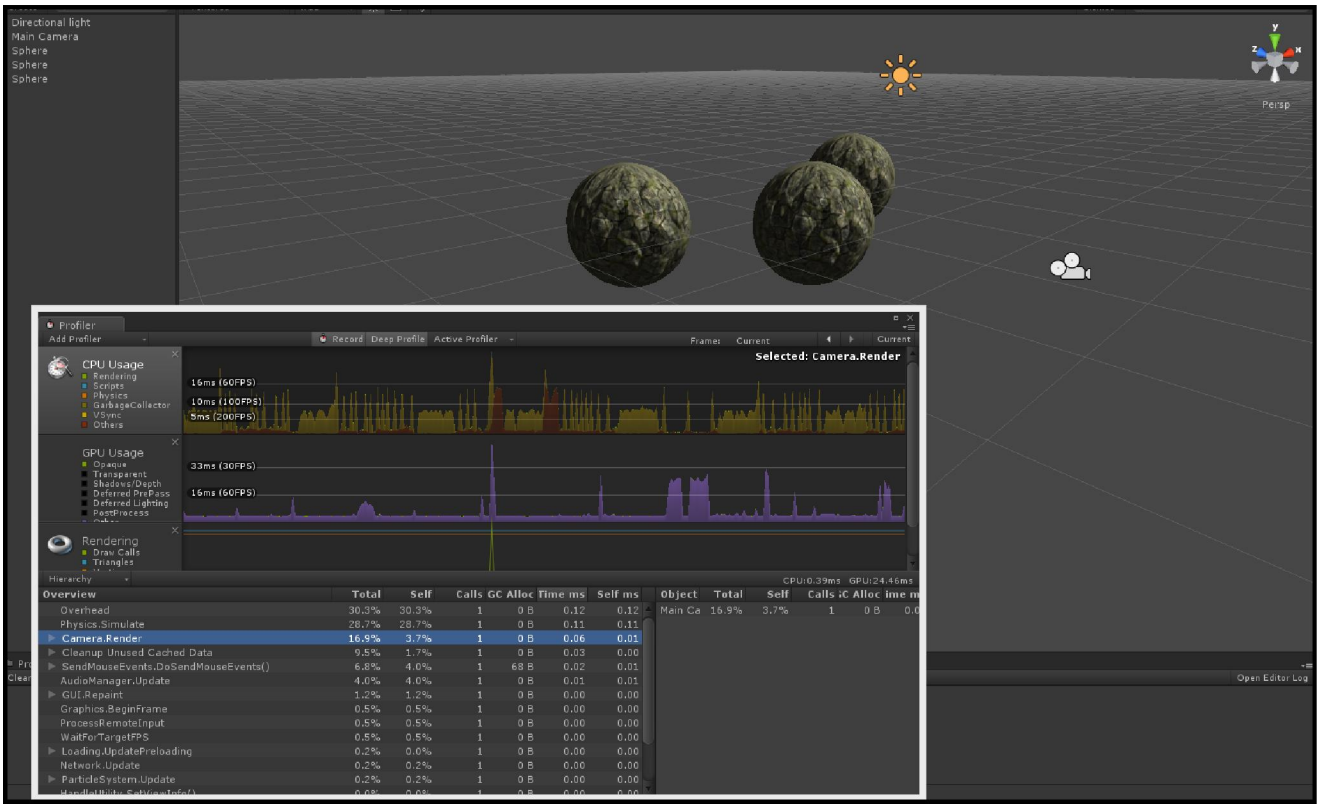
```
struct Input
{
    half2 uv_MainTex;
};
```

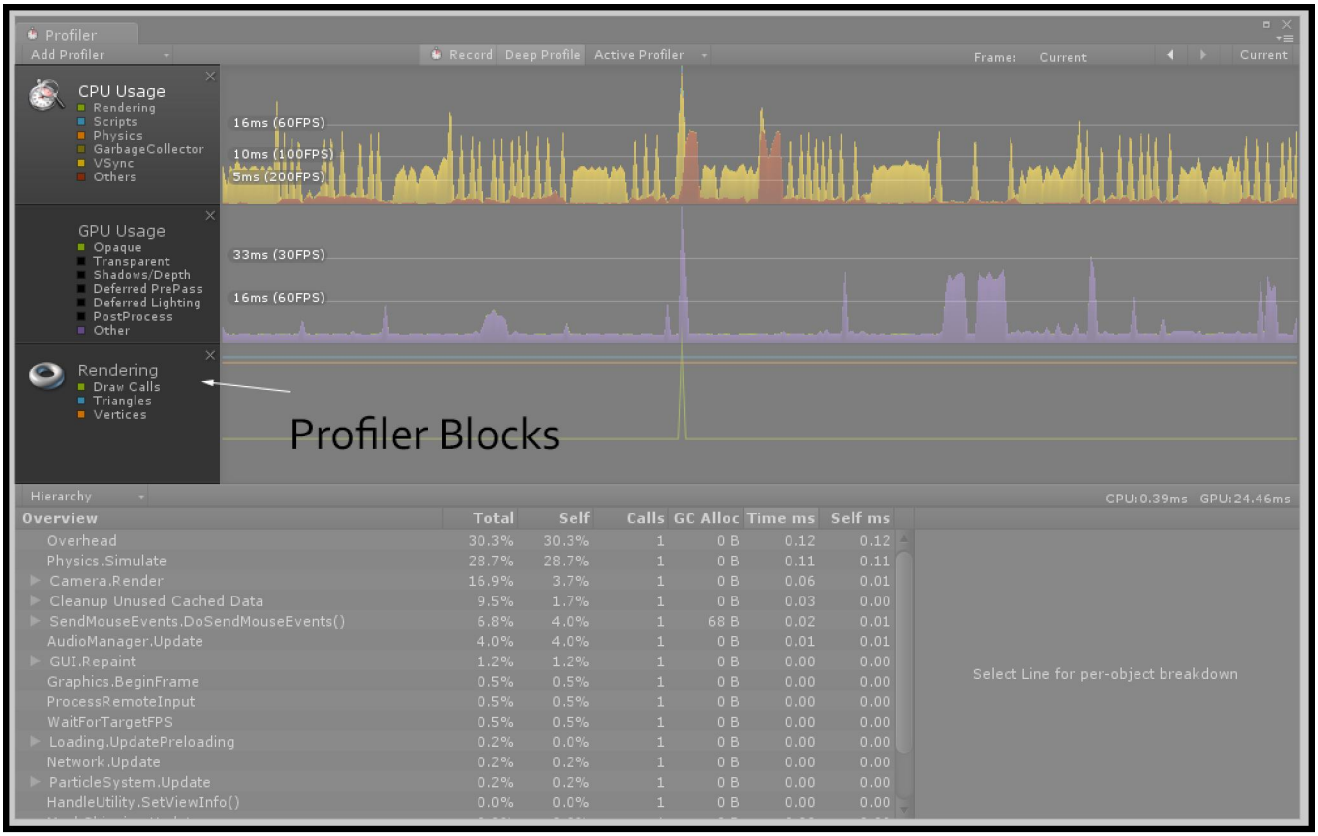
```
CGPROGRAM
#pragma surface surf SimpleLambert exclude_path:prepass noforwardadd
```

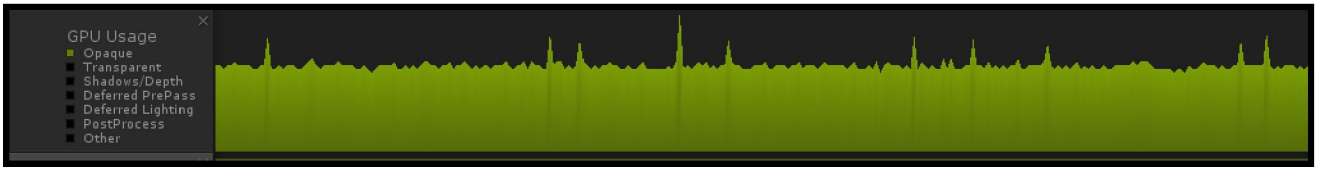


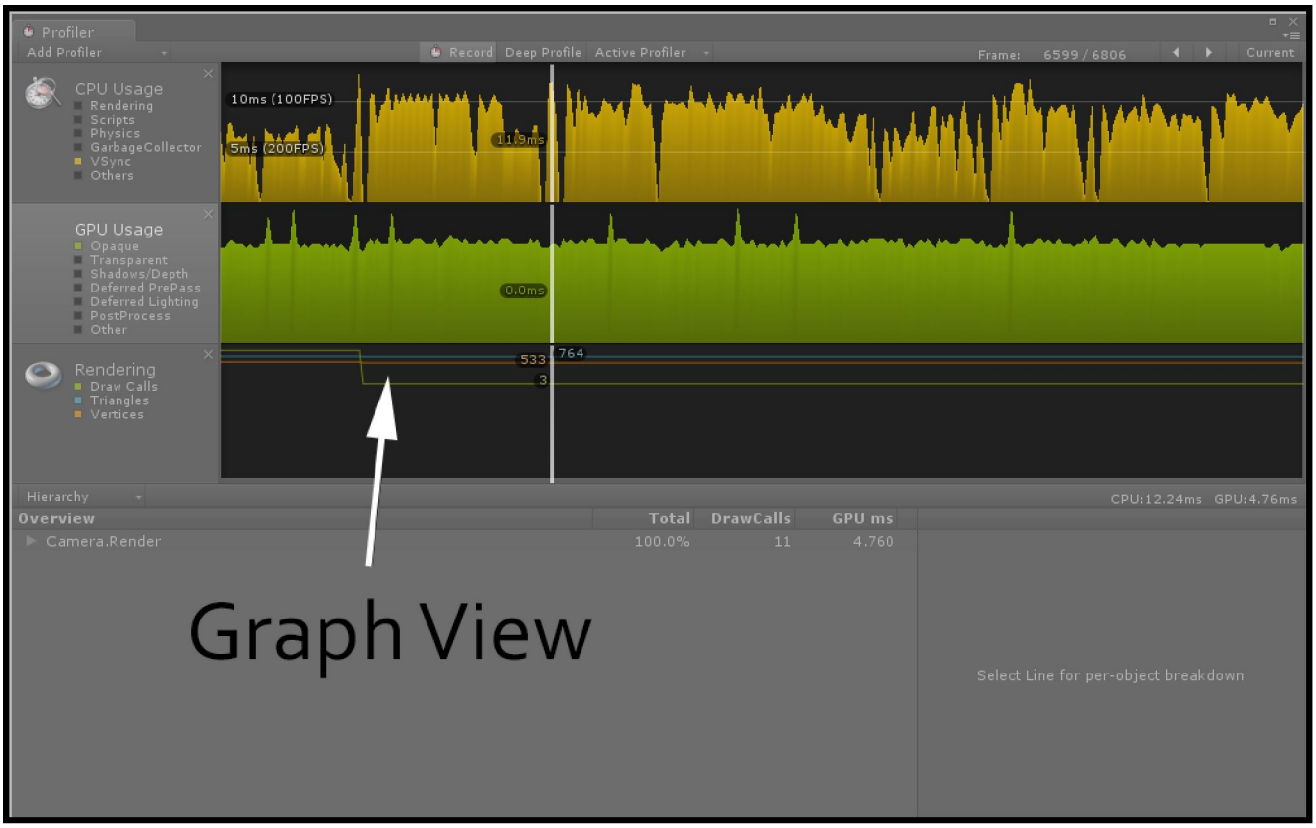


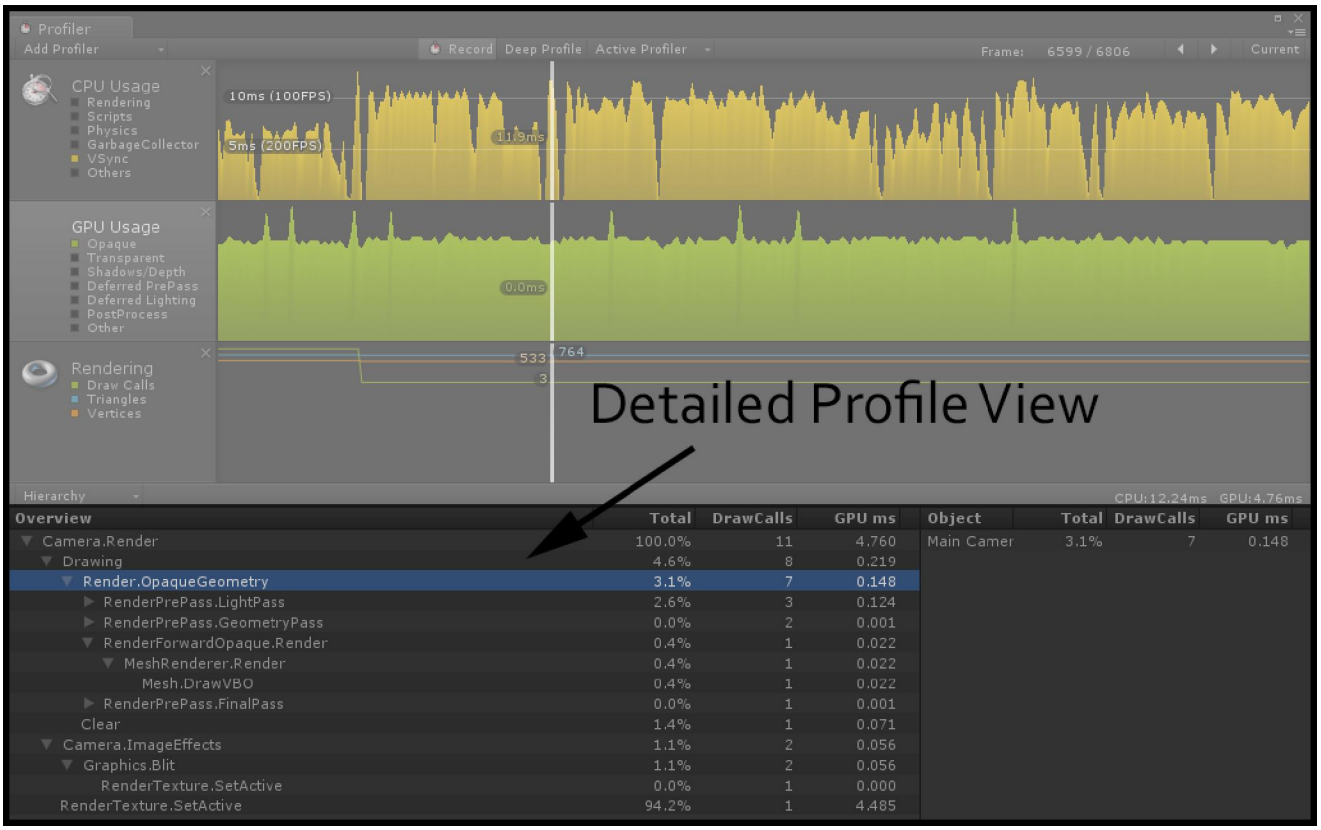
Effect of using noforwardadd in a shader's #pragam statement




Profiler
Add Profiler
Record
Deep Profile
Active Profiler
Frame:
Current
Current
CPU Usage
Rendering
Scripts
Physics
GarbageCollector
VSync
Others
16ms (60FPS)
10ms (100FPS)
5ms (200FPS)
GPU Usage
Opaque
Transparent
Shadows/Depth
Deferred PrePass
Deferred Lighting
PostProcess
Other
33ms (30FPS)
16ms (60FPS)
Rendering
Draw Calls
Triangles
Vertices
Profiler Blocks
Hierarchy
CPU:0.39ms GPU:24.46ms
Overview
Total
Self
Calls
GC Alloc
Time ms
Self ms
Overhead 30.3% 30.3% 1 0 B 0.12 0.12
Physics.Simulate 28.7% 28.7% 1 0 B 0.11 0.11
Camera.Render 16.9% 3.7% 1 0 B 0.06 0.01
Cleanup Unused Cached Data 9.5% 1.7% 1 0 B 0.03 0.00
SendMouseEvents.DoSendMouseEvents() 6.8% 4.0% 1 68 B 0.02 0.01
AudioManager.Update 4.0% 4.0% 1 0 B 0.01 0.01
GUI.Repaint 1.2% 1.2% 1 0 B 0.00 0.00
Graphics.BeginFrame 0.5% 0.5% 1 0 B 0.00 0.00
ProcessRemoteInput 0.5% 0.5% 1 0 B 0.00 0.00
WaitForTargetFPS 0.5% 0.5% 1 0 B 0.00 0.00
Loading.UpdatePreloading 0.2% 0.0% 1 0 B 0.00 0.00
Network.Update 0.2% 0.2% 1 0 B 0.00 0.00
ParticleSystem.Update 0.2% 0.2% 1 0 B 0.00 0.00
HandleUtility.SetViewInfo() 0.0% 0.0% 1 0 B 0.00 0.00
Select Line for per-object breakdown

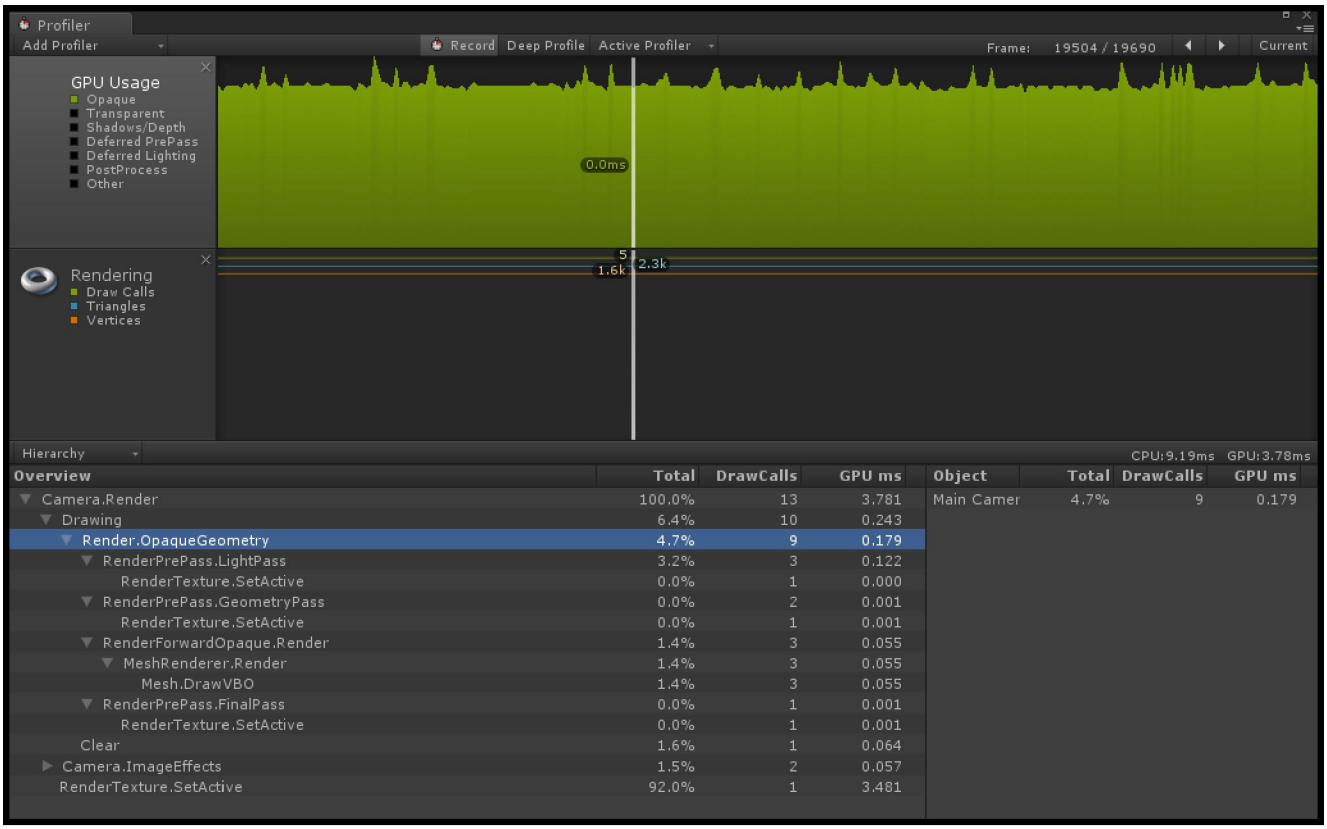

GPU Usage
Opaque
Transparent
Shadows/Depth
Deferred PrePass
Deferred Lighting
PostProcess
Other

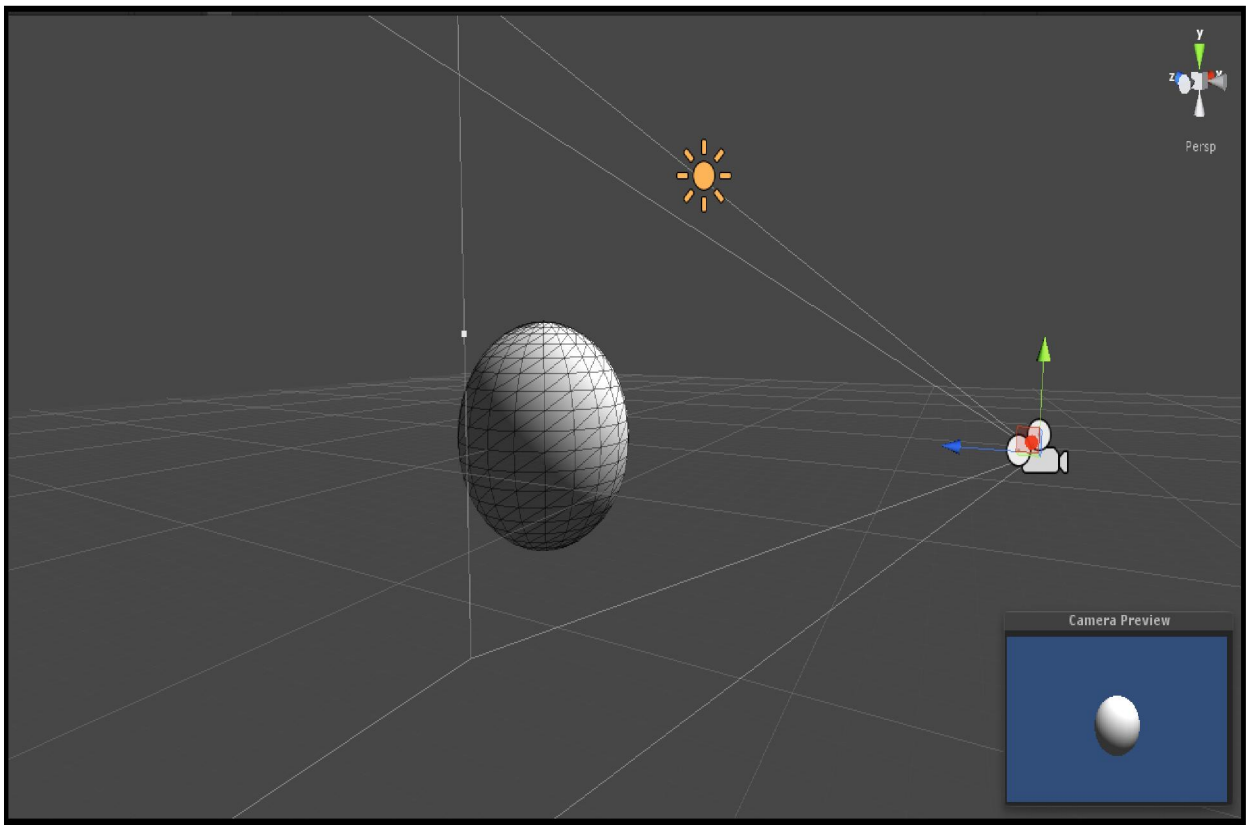

Profiler
Add Profiler
Record
Deep Profile
Active Profiler
Frame: 6599 / 6806
Current
CPU Usage
Rendering
Scripts
Physics
GarbageCollector
VSync
Others
10ms (100FPS)
5ms (200FPS)
11.9ms
GPU Usage
Opaque
Transparent
Shadows/Depth
Deferred PrePass
Deferred Lighting
PostProcess
Other
0.0ms
Rendering
Draw Calls
Triangles
Vertices
533
764
3
Hierarchy
CPU:12.24ms GPU:4.76ms
Overview
Total
DrawCalls
GPU ms
Camera.Render
100.0%
11
4.760
Graph View
Select Line for per-object breakdown


Profiler
Add Profiler
Record
Deep Profile
Active Profiler
Frame: 6599 / 6806
Current
CPU Usage
Rendering
Scripts
Physics
GarbageCollector
VSync
Others
10ms (100FPS)
5ms (200FPS)
11.9ms
GPU Usage
Opaque
Transparent
Shadows/Depth
Deferred PrePass
Deferred Lighting
PostProcess
Other
0.0ms
Rendering
Draw Calls
Triangles
Vertices
533
764
3
Detailed Profile View
Hierarchy
CPU:12.24ms GPU:4.76ms
Overview
Total
DrawCalls
GPU ms
Camera.Render 100.0% 11 4.760
Drawing 4.6% 8 0.219
Render.OpaqueGeometry 3.1% 7 0.148
RenderPrePass.LightPass 2.6% 3 0.124
RenderPrePass.GeometryPass 0.0% 2 0.001
RenderForwardOpaque.Render 0.4% 1 0.022
MeshRenderer.Render 0.4% 1 0.022
Mesh.DrawVBO 0.4% 1 0.022
RenderPrePass.FinalPass 0.0% 1 0.001
Clear 1.4% 1 0.071
Camera.ImageEffects 1.1% 2 0.056
Graphics.Blit 1.1% 2 0.056
RenderTexture.SetActive 0.0% 1 0.000
RenderTexture.SetActive 94.2% 1 4.485
Object
Total
DrawCalls
GPU ms
Main Camer 3.1% 7 0.148

```
Properties
{
    _MainTex ("Base (RGB)", 2D) = "white" {}
    _BlendTex ("Blend Texture", 2D) = "white" {}
    _NormalMap ("Normal Map", 2D) = "bump" {}
}
```

```
sampler2D _MainTex;
sampler2D _BlendTex;
sampler2D _NormalMap;

void surf (Input IN, inout SurfaceOutput o)
{
    fixed4 c = tex2D (_MainTex, IN.uv_MainTex);
    fixed4 blendTex = tex2D (_BlendTex, IN.uv_MainTex);

    c = lerp(c, blendTex, blendTex.r);

    o.Albedo = c.rgb;
    o.Alpha = c.a;
    o.Normal = UnpackNormal(tex2D(_NormalMap, IN.uv_MainTex));
}
```


Profiler
Add Profiler
Record
Deep Profile
Active Profiler
Frame: 19504 / 19690
Current
GPU Usage
Opaque
Transparent
Shadows/Depth
Deferred PrePass
Deferred Lighting
PostProcess
Other
0.0ms
Rendering
Draw Calls
Triangles
Vertices
5
1.6k
2.3k
Hierarchy
CPU:9.19ms GPU:3.78ms
Overview
Total
DrawCalls
GPU ms
Camera.Render 100.0% 13 3.781
Drawing 6.4% 10 0.243
Render.OpaqueGeometry 4.7% 9 0.179
RenderPrePass.LightPass 3.2% 3 0.122
RenderTexture.SetActive 0.0% 1 0.000
RenderPrePass.GeometryPass 0.0% 2 0.001
RenderTexture.SetActive 0.0% 1 0.001
RenderForwardOpaque.Render 1.4% 3 0.055
MeshRenderer.Render 1.4% 3 0.055
Mesh.DrawVBO 1.4% 3 0.055
RenderPrePass.FinalPass 0.0% 1 0.001
RenderTexture.SetActive 0.0% 1 0.001
Clear 1.6% 1 0.064
Camera.ImageEffects 1.5% 2 0.057
RenderTexture.SetActive 92.0% 1 3.481
Object
Total
DrawCalls
GPU ms
Main Camer 4.7% 9 0.179


y
z
x
Persp
Camera Preview

```
Properties
{
    _Diffuse ("Base (RGB) Specular Amount (A)", 2D) = "white" {}
    _SpecIntensity ("Specular Width", Range(0.01, 1)) = 0.5
    _NormalMap ("Normal Map", 2D) = "bump"{}
}
```

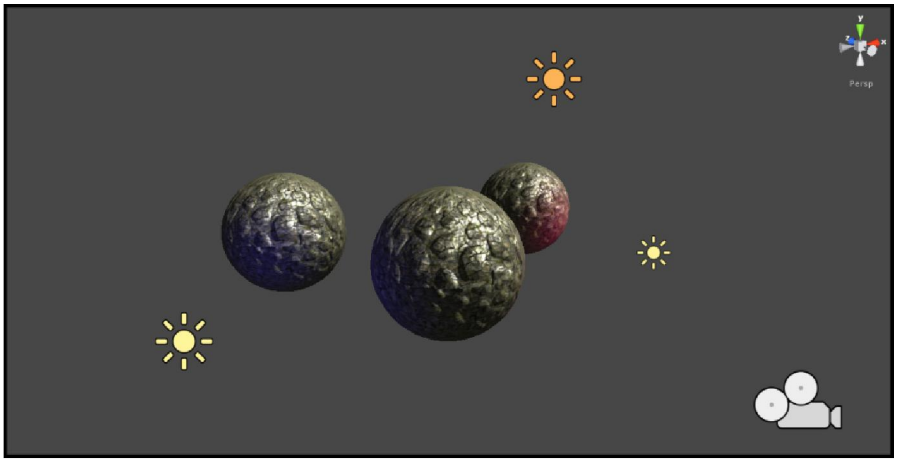
```
CGPROGRAM
#pragma surface surf MobileBlinnPhong exclude_path:prepass nolightmap noforwardadd halfasview
```

```
sampler2D _Diffuse;
sampler2D _NormalMap;
fixed _SpecIntensity;
```

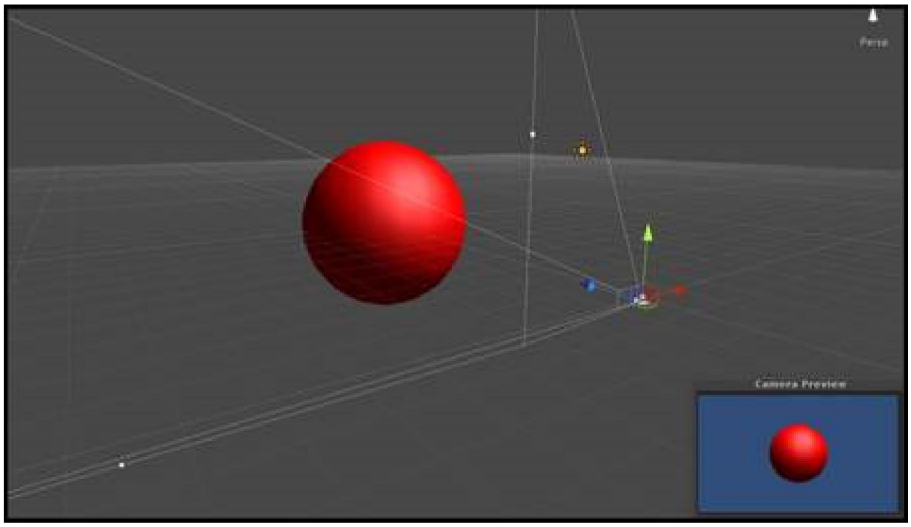
```
struct Input
{
    half2 uv_Diffuse;
};
```

```
inline fixed4 LightingMobileBlinnPhong (SurfaceOutput s, fixed3 lightDir, fixed3 halfDir, fixed atten)
{
    fixed diff = max (0, dot (s.Normal, lightDir));
    fixed nh = max (0, dot (s.Normal, halfDir));
    fixed spec = pow (nh, s.Specular*128) * s.Gloss;

    fixed4 c;
    c.rgb = (s.Albedo * _LightColor0.rgb * diff + _LightColor0.rgb * spec) * (atten*2);
    c.a = 0.0;
    return c;
}
```

```
void surf (Input IN, inout SurfaceOutput o)
{
    fixed4 diffuseTex = tex2D (_Diffuse, IN.uv_Diffuse);
    o.Albedo = diffuseTex.rgb;
    o.Gloss = diffuseTex.a;
    o.Alpha = 0.0;
    o.Specular = _SpecIntensity;
    o.Normal = UnpackNormal(tex2D(_NormalMap, IN.uv_Diffuse));
}
```



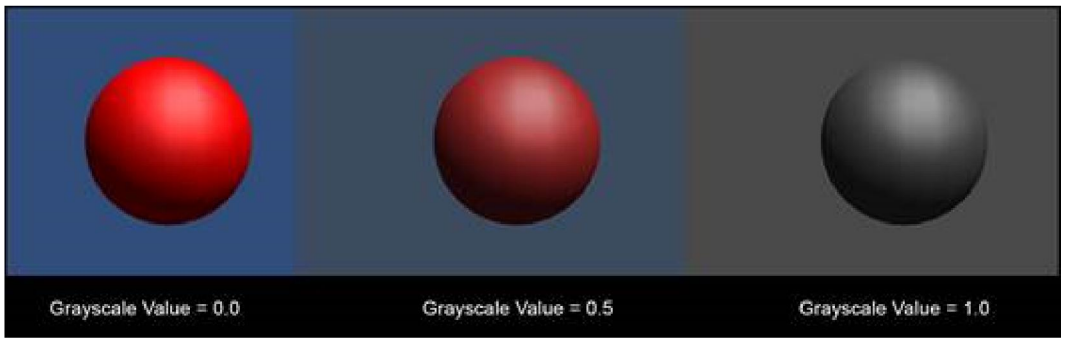
Chapter 8: Screen Effects with Unity Render Texture

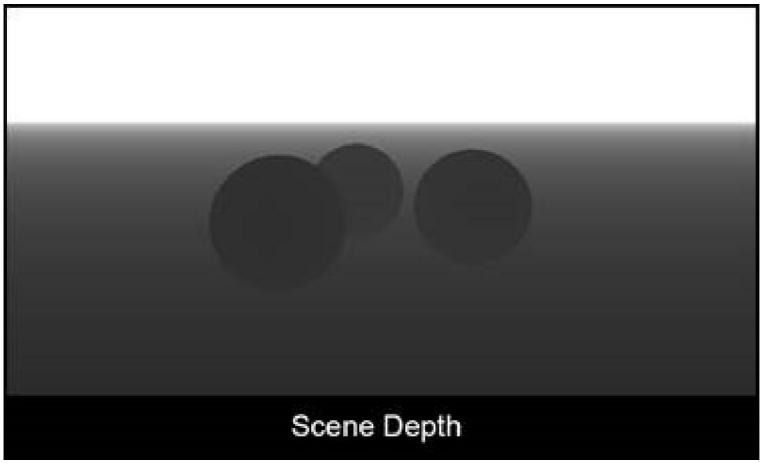


```
public class TestRenderImage : MonoBehaviour
{
    #region Variables
    public Shader curShader;
    public float grayScaleAmount = 1.0f;
    private Material curMaterial;
    #endregion
```

```
[ExecuteInEditMode]
public class TestRenderImage : MonoBehaviour
{
#region Properties
Material material
{
    get
    {
        if(curMaterial == null)
        {
            curMaterial = new Material(curShader);
            curMaterial.hideFlags = HideFlags.HideAndDontSave;
        }
        return curMaterial;
    }
}
#endregion

 void Start()
 {
     if(!SystemInfo.supportsImageEffects)
     {
         enabled = false;
         return;
     }

     if(!curShader && !curShader.isSupported)
     {
         enabled = false;
     }
 }
void OnRenderImage(RenderTexture sourceTexture, RenderTexture destTexture)
{
    if(curShader != null)
    {
        material.SetFloat("_LuminosityAmount", grayScaleAmount);
        Graphics.Blit(sourceTexture, destTexture, material);
    }
    else
    {
        Graphics.Blit(sourceTexture, destTexture);
    }
}

void Update()
{
    grayScaleAmount = Mathf.Clamp(grayScaleAmount, 0.0f, 1.0f);
}

void OnDisable()
{
    if(curMaterial)
    {
        DestroyImmediate(curMaterial);
    }
}
```

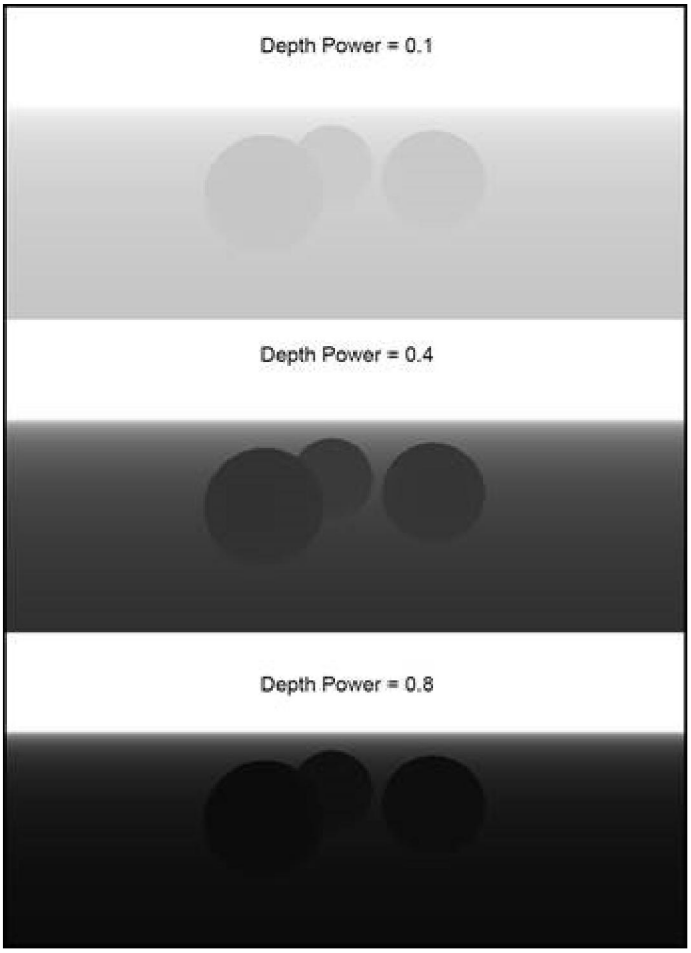
```
Properties
{
    _MainTex ("Base (RGB)", 2D) = "white" {}
    _LuminosityAmount ("GrayScale Amount", Range(0.0, 1)) = 1.0
}

SubShader
{
    Pass
    {
        CGPROGRAM
        #pragma vertex vert_img
        #pragma fragment frag
        #pragma fragmentoption ARB_precision_hint_fastest
        #include "UnityCG.cginc"


            uniform sampler2D _MainTex;
            fixed _LuminosityAmount;

    fixed4 frag(v2f_img i) : COLOR
    {
        //Get the colors from the RenderTexture and the uv's
        //from the v2f_img struct
        fixed4 renderTex = tex2D(_MainTex, i.uv);

        //Apply the Luminosity values to our render texture
        float luminosity = 0.299 * renderTex.r + 0.587 * renderTex.g + 0.114 * renderTex.b;
        fixed4 finalColor = lerp(renderTex, luminosity, _LuminosityAmount);


        return finalColor;
    }
```

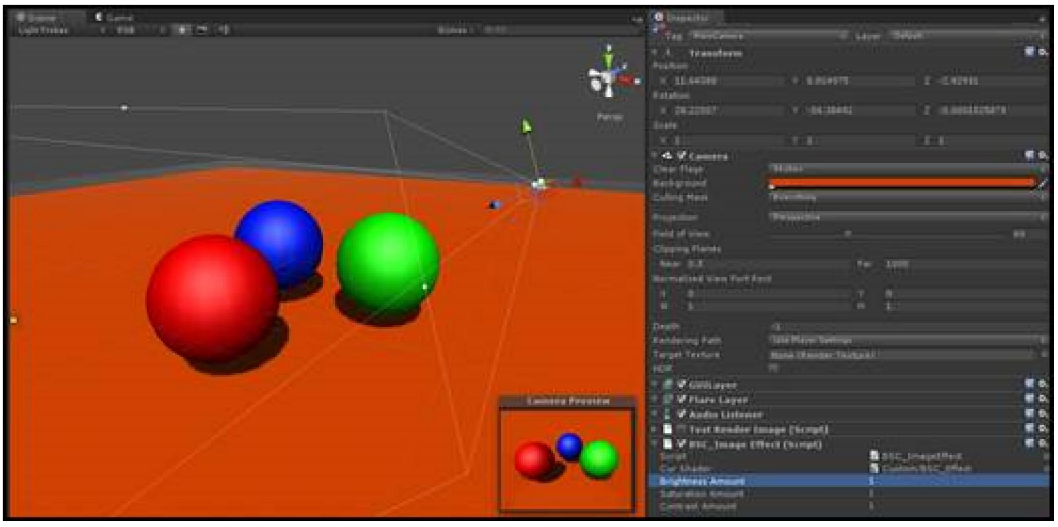




```
        Properties
        {
            _MainTex ("Base (RGB)", 2D) = "white" {}
            _DepthPower ("Depth Power", Range(1, 5)) = 1
        }

        Pass
        {
            CGPROGRAM
            #pragma vertex vert_img
            #pragma fragment frag
            #pragma fragmentoption ARB_precision_hint_fastest
            #include "UnityCG.cginc"

            uniform sampler2D _MainTex;
            fixed _DepthPower;
            sampler2D _CameraDepthTexture;

fixed4 frag(v2f_img i) : COLOR
{
    //Get the colors from the RenderTexture and the uv's
    //from the v2f_img struct
    float d = UNITY_SAMPLE_DEPTH( tex2D(_CameraDepthTexture, i.uv.xy) );
    d = pow(Linear01Depth(d), _DepthPower);

    return d;
}
```

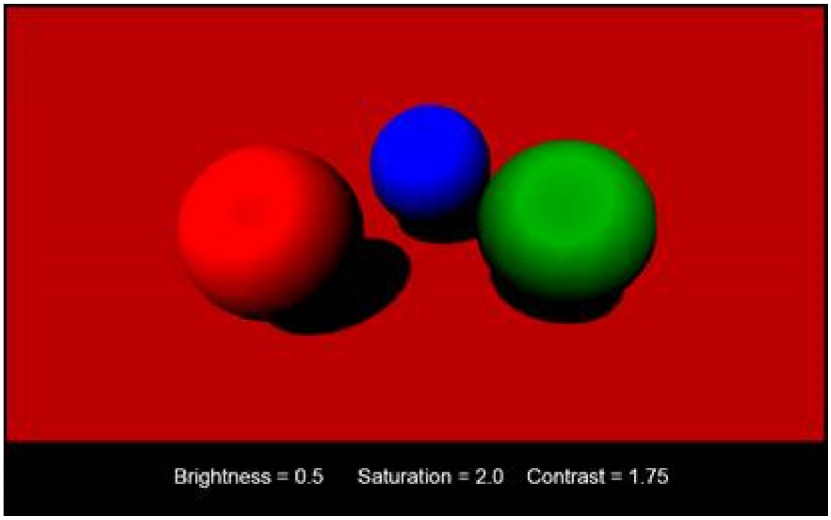
```
            #region Variables
            public Shader curShader;
            private Material curMaterial;

            public float depthPower = 1.0f;
            #endregion

  void OnRenderImage(RenderTexture sourceTexture, RenderTexture destTexture)
  {
      if(curShader != null)
      {
          material.SetFloat("_DepthPower", depthPower);
          Graphics.Blit(sourceTexture, destTexture, material);
      }
      else
      {
          Graphics.Blit(sourceTexture, destTexture);
      }
  }


   void Update()
   {
       Camera.main.depthTextureMode = DepthTextureMode.Depth;
       depthPower = Mathf.Clamp(depthPower, 0, 5);
   }
```

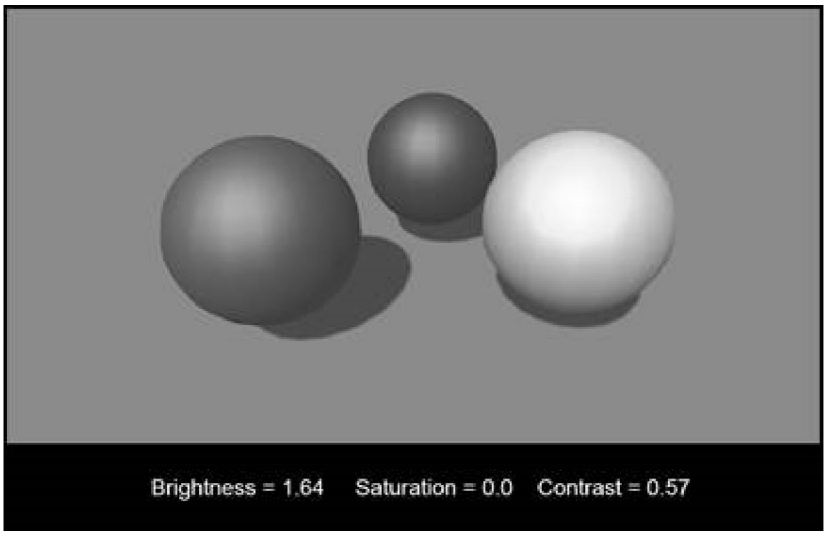

Depth Power = 0.1
Depth Power = 0.4
Depth Power = 0.8

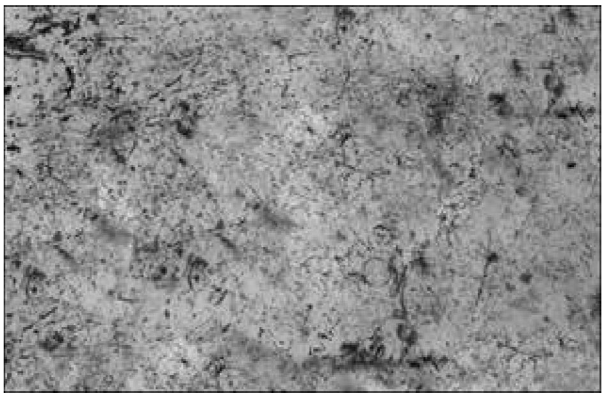


```
Properties
{
    _MainTex ("Base (RGB)", 2D) = "white" {}
    _BrightnessAmount ("Brightness Amount", Range(0.0, 1)) = 1.0
    _satAmount ("Saturation Amount", Range(0.0, 1)) = 1.0
    _conAmount ("Contrast Amount", Range(0.0, 1)) = 1.0
}
Pass
{
    CGPROGRAM
    #pragma vertex vert_img
    #pragma fragment frag
    #pragma fragmentoption ARB_precision_hint_fastest
    #include "UnityCG.cginc"

    uniform sampler2D _MainTex;
    fixed _BrightnessAmount;
    fixed _satAmount;
    fixed _conAmount;



float3 ContrastSaturationBrightness(float3 color, float brt, float sat, float con)
{
    // Increase or decrease theese values to
    //adjust r, g and b color channels seperately
    float AvgLumR = 0.5;
    float AvgLumG = 0.5;
    float AvgLumB = 0.5;

    //Luminance coefficients for getting lumoinance from the image
    float3 LuminanceCoeff = float3(0.2125, 0.7154, 0.0721);

    //Operation for brightness
    float3 AvgLumin = float3(AvgLumR, AvgLumG, AvgLumB);
    float3 brtColor = color * brt;
    float intensityf = dot(brtColor, LuminanceCoeff);
    float3 intensity = float3(intensityf, intensityf, intensityf);

    //Operation for Saturation
    float3 satColor = lerp(intensity, brtColor, sat);

    //Operation for Contrast
    float3 conColor = lerp(AvgLumin, satColor, con);
    return conColor;
}


  fixed4 frag(v2f_img i) : COLOR
  {
      //Get the colors from the RenderTexture and the uv's
      //from the v2f_img struct
      fixed4 renderTex = tex2D(_MainTex, i.uv);

      //Apply the Brughtness, saturation, contrast operations
      renderTex.rgb = ContrastSaturationBrightness(renderTex.rgb,
                                                   _BrightnessAmount,
                                                   _satAmount,
                                                   _conAmount);


      return renderTex;
  }
```

```
        #region Variables
        public Shader curShader;
        public float brightnessAmount = 1.0f;
        public float saturationAmount = 1.0f;
        public float contrastAmount = 1.0f;
        private Material curMaterial;
        #endregion

void OnRenderImage(RenderTexture sourceTexture, RenderTexture destTexture)
{
    if(curShader != null)
    {
        material.SetFloat("_BrightnessAmount", brightnessAmount);
        material.SetFloat("_satAmount", saturationAmount);
        material.SetFloat("_conAmount", contrastAmount);

        Graphics.Blit(sourceTexture, destTexture, material);
    }
    else
    {
        Graphics.Blit(sourceTexture, destTexture);
    }
}


void Update()
{
    brightnessAmount = Mathf.Clamp(brightnessAmount, 0.0f, 2.0f);
    saturationAmount = Mathf.Clamp(saturationAmount, 0.0f, 2.0f);
    contrastAmount = Mathf.Clamp(contrastAmount, 0.0f, 3.0f);
}
```







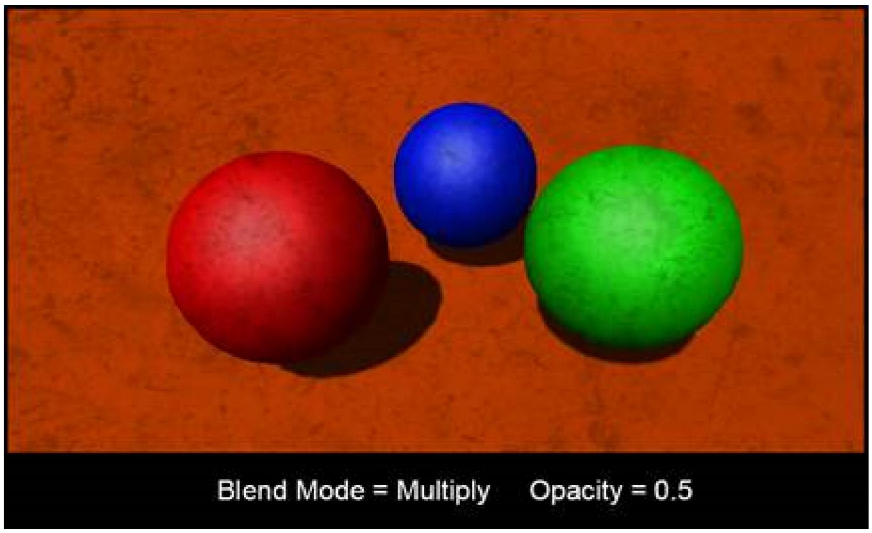
```
Properties
{
    _MainTex ("Base (RGB)", 2D) = "white" {}
    _BlendTex ("Blend Texture", 2D) = "white"{}
    _Opacity ("Blend Opacity", Range(0,1)) = 1
}


    Pass
    {
        CGPROGRAM
        #pragma vertex vert_img
        #pragma fragment frag
        #pragma fragmentoption ARB_precision_hint_fastest
        #include "UnityCG.cginc"

        uniform sampler2D _MainTex;
        uniform sampler2D _BlendTex;
        fixed _Opacity;
```

```
    fixed4 frag(v2f_img i) : COLOR
    {
        //Get the colors from the RenderTexture and the uv's
        //from the v2f_img struct
        fixed4 renderTex = tex2D(_MainTex, i.uv);
        fixed4 blendTex = tex2D(_BlendTex, i.uv);

        //Perform a multiply Blend mode
        fixed4 blendedMultiply = renderTex * blendTex;

        //Adjust amount of Blend Mode with a lerp
        renderTex = lerp(renderTex, blendedMultiply, _Opacity);

        return renderTex;
    }
```

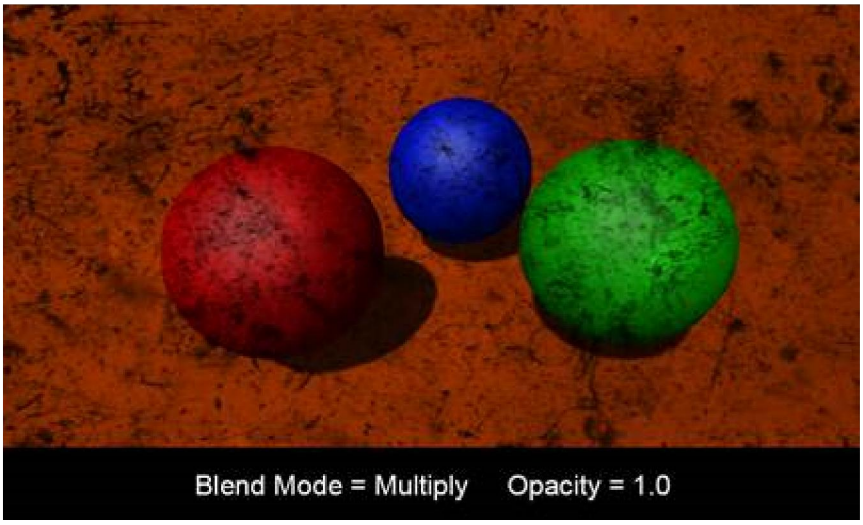
```
    #region Variables
    public Shader curShader;
    public Texture2D blendTexture;
    public float blendOpacity = 1.0f;
    private Material curMaterial;
    #endregion
```

```
void OnRenderImage(RenderTexture sourceTexture, RenderTexture destTexture)
{
    if(curShader != null)
    {
        material.SetTexture("_BlendTex", blendTexture);
        material.SetFloat("_Opacity", blendOpacity);

        Graphics.Blit(sourceTexture, destTexture, material);
    }
    else
    {
        Graphics.Blit(sourceTexture, destTexture);
    }
}
```

```
void Update()
{
    blendOpacity = Mathf.Clamp(blendOpacity, 0.0f, 1.0f);
}
```

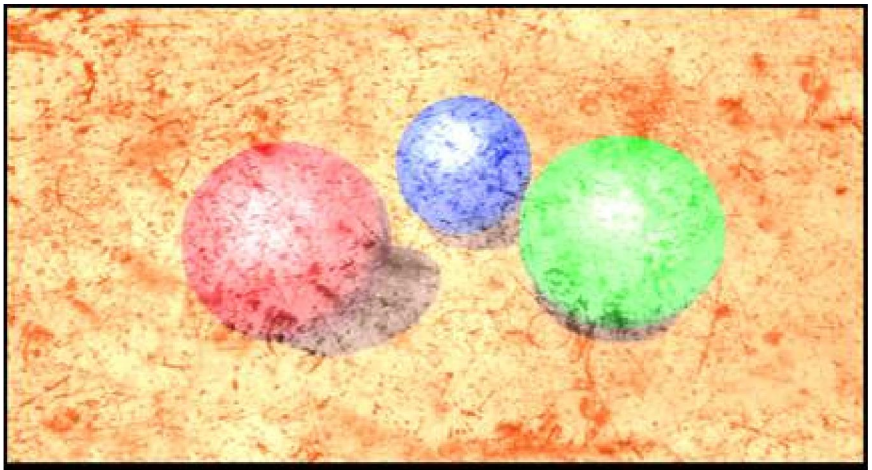


Blend Mode = Multiply Opacity = 0.5



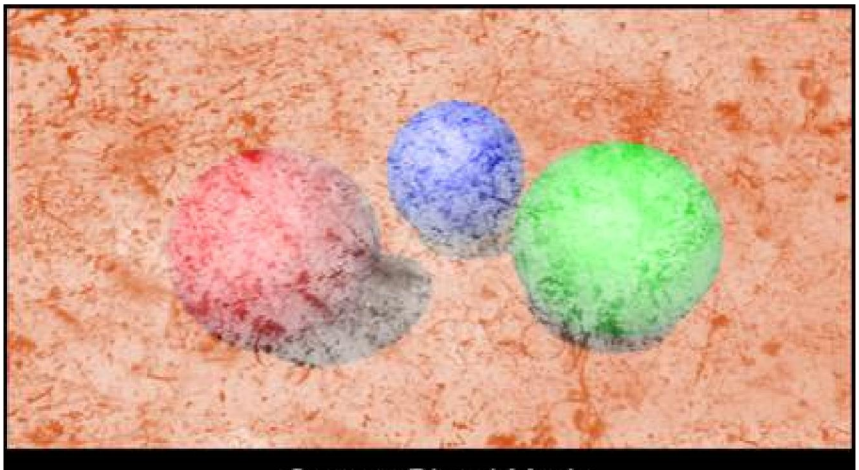
Blend Mode = Multiply Opacity = 1.0

```
fixed4 frag(v2f_img i) : COLOR
{
    //Get the colors from the RenderTexture and the uv's
    //from the v2f_img struct
    fixed4 renderTex = tex2D(_MainTex, i.uv);
    fixed4 blendTex = tex2D(_BlendTex, i.uv);

    //Perform a multiply Blend mode
    //fixed4 blendedMultiply = renderTex * blendTex;
    fixed4 blendedMultiply = renderTex + blendTex;

    //Adjust amount of Blend Mode with a lerp
    renderTex = lerp(renderTex, blendedMultiply, _Opacity);

    return renderTex;
}
```



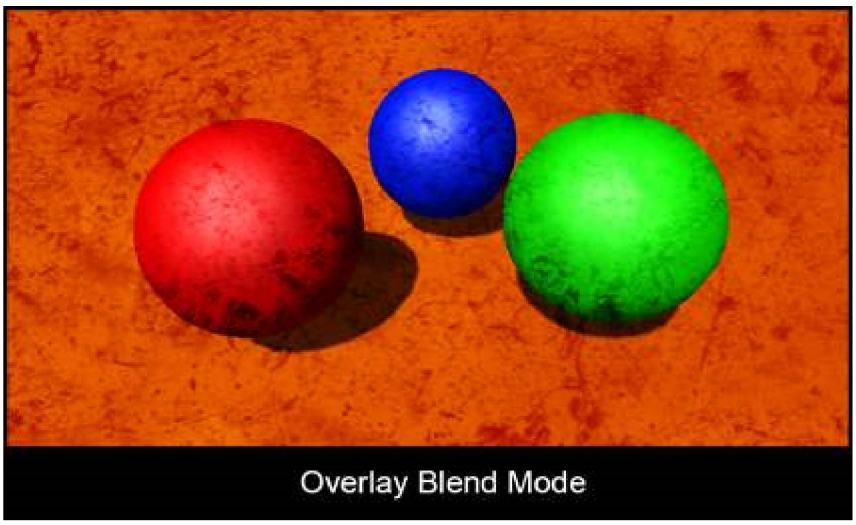
Add Blending Mode

```
fixed4 frag(v2f_img i) : COLOR
{
    //Get the colors from the RenderTexture and the uv's
    //from the v2f_img struct
    fixed4 renderTex = tex2D(_MainTex, i.uv);
    fixed4 blendTex = tex2D(_BlendTex, i.uv);

    //Perform a multiply Blend mode
    //fixed4 blendedMultiply = renderTex * blendTex;
    //fixed4 blendedAdd = renderTex + blendTex;
    fixed4 blendedScreen = (1.0 - ((1.0 - renderTex) * (1.0 - blendTex)));

    //Adjust amount of Blend Mode with a lerp
    renderTex = lerp(renderTex, blendedScreen, _Opacity);

    return renderTex;
}
```

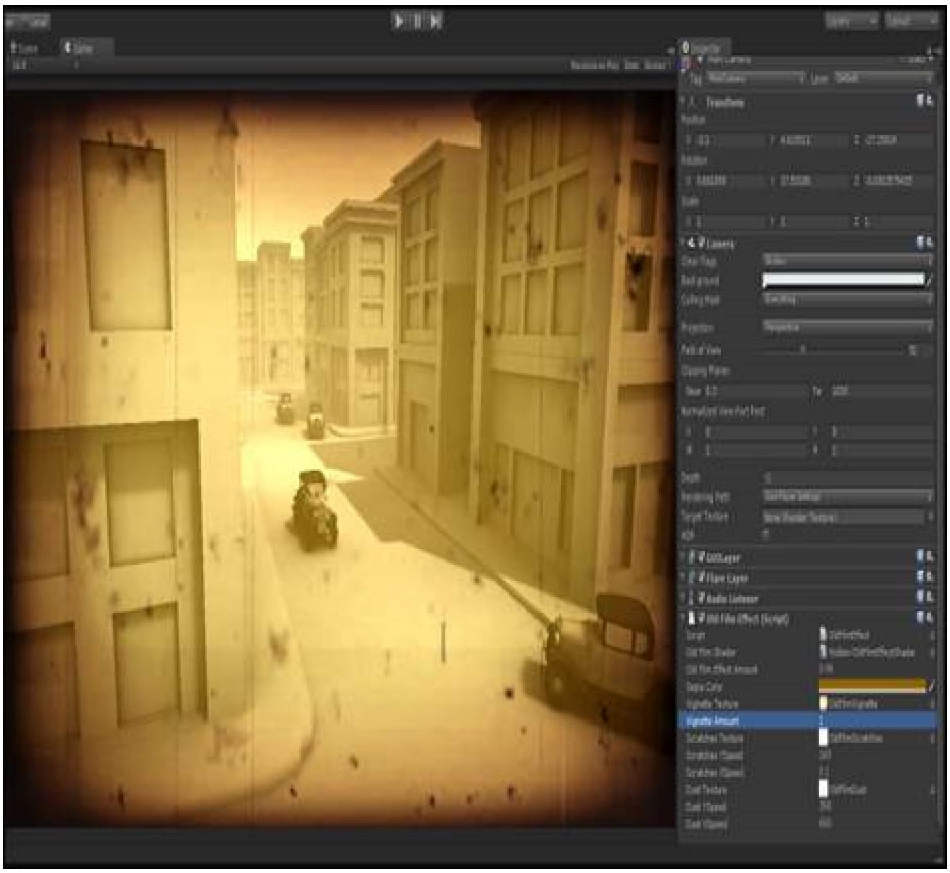


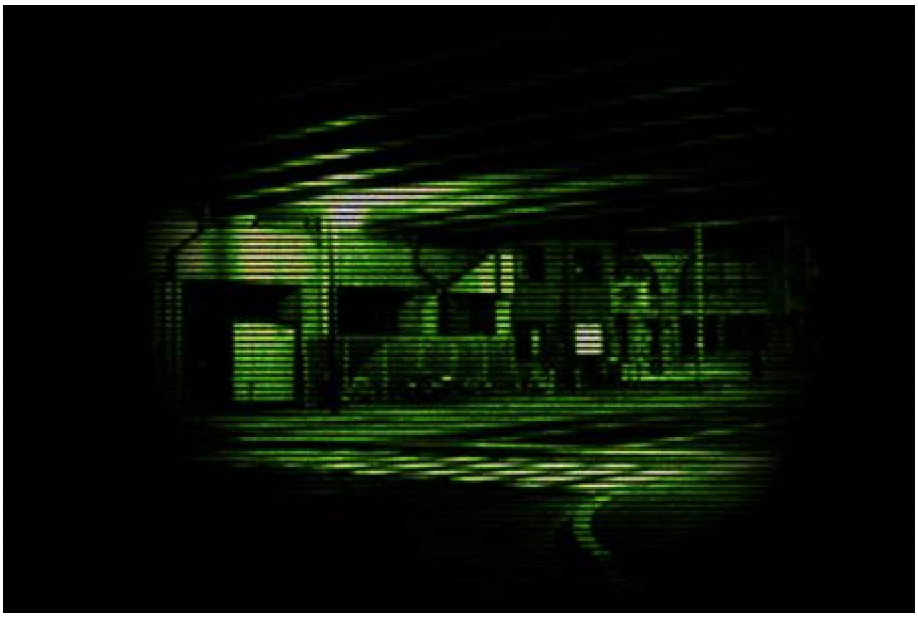
Screen Blend Mode

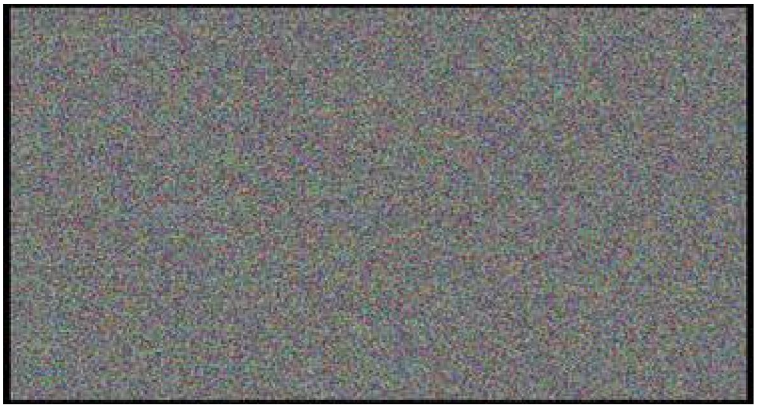
```
        Properties
        {
            _MainTex ("Base (RGB)", 2D) = "white" {}
            _BlendTex ("Blend Texture", 2D) = "white"{}
            _Opacity ("Blend Opacity", Range(0,1)) = 1
        }

    Pass
    {
        CGPROGRAM
        #pragma vertex vert_img
        #pragma fragment frag
        #pragma fragmentoption ARB_precision_hint_fastest
        #include "UnityCG.cginc"

        uniform sampler2D _MainTex;
        uniform sampler2D _BlendTex;
        fixed _Opacity;


fixed OverlayBlendMode(fixed basePixel, fixed blendPixel)
{
    if(basePixel < 0.5)
    {
        return (2.0 * basePixel * blendPixel);
    }
    else
    {
        return (1.0 - 2.0 * (1.0 - basePixel) * (1.0 - blendPixel));
    }
}


     fixed4 frag(v2f_img i) : COLOR
     {
         //Get the colors from the RenderTexture and the uv's
         //from the v2f_img struct
         fixed4 renderTex = tex2D(_MainTex, i.uv);
         fixed4 blendTex = tex2D(_BlendTex, i.uv);

         fixed4 blendedImage = renderTex;

         blendedImage.r = OverlayBlendMode(renderTex.r, blendTex.r);
         blendedImage.g = OverlayBlendMode(renderTex.g, blendTex.g);
         blendedImage.b = OverlayBlendMode(renderTex.b, blendTex.b);

         //Adjust amount of Blend Mode with a lerp
         renderTex = lerp(renderTex, blendedImage, _Opacity);

         return renderTex;
     }
```

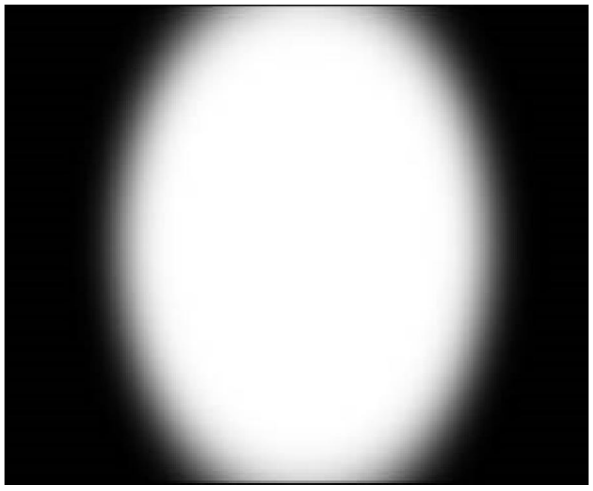


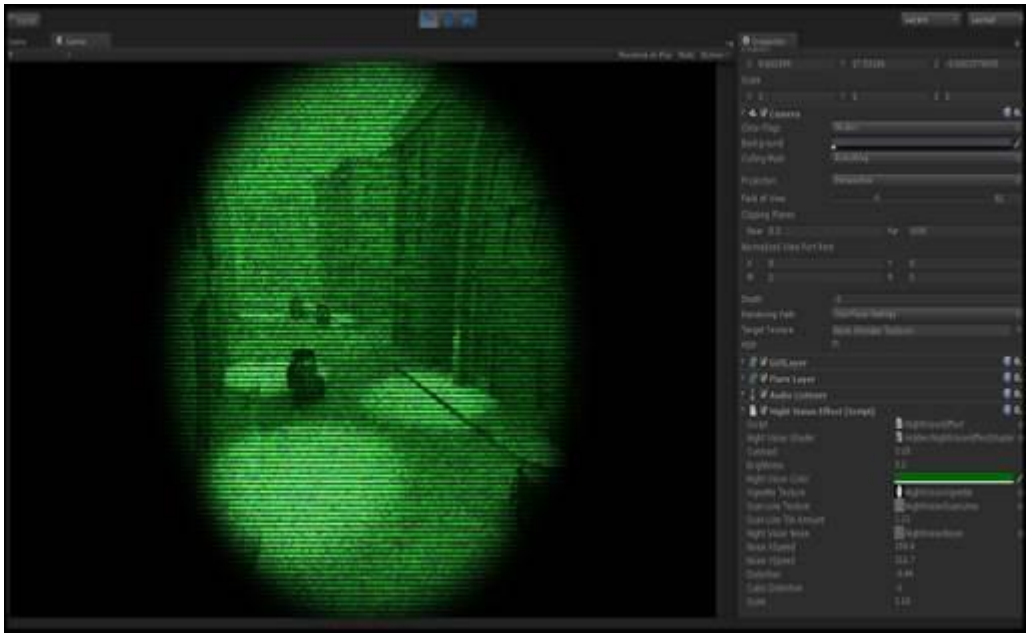
Chapter 9: Gameplay and Screen Effects



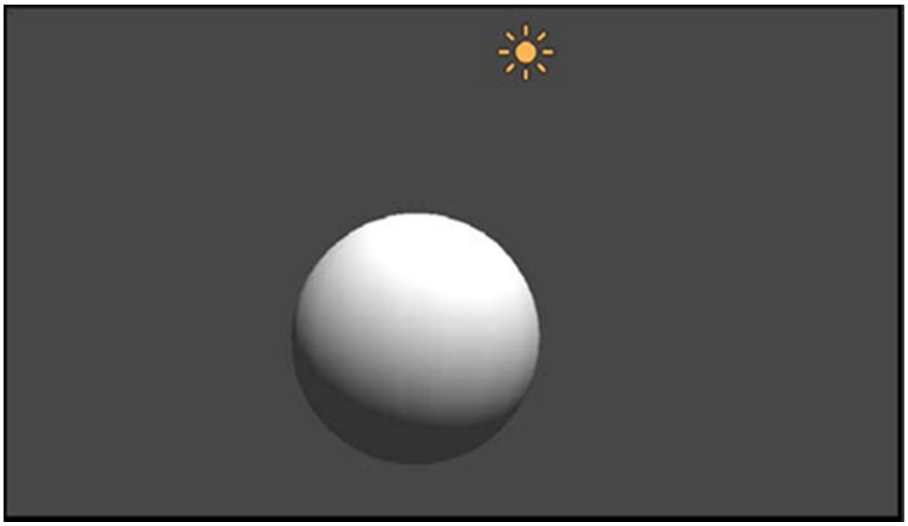




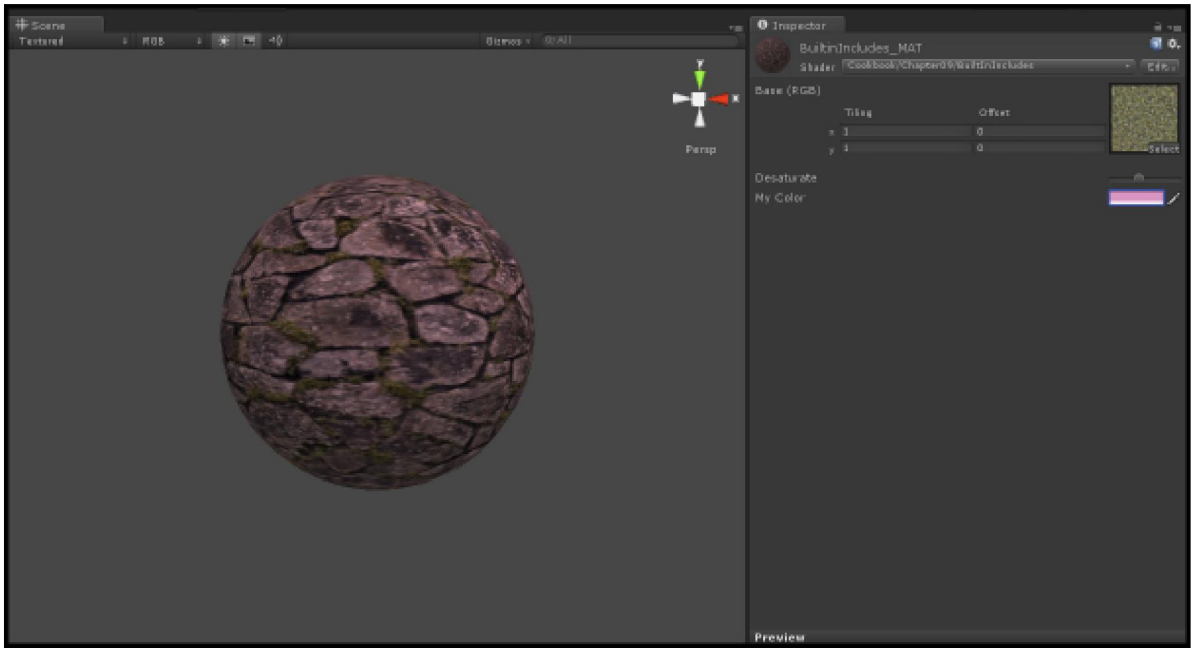


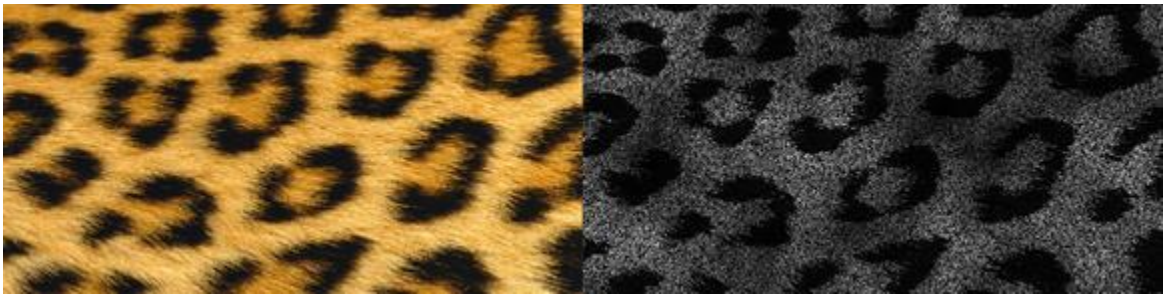


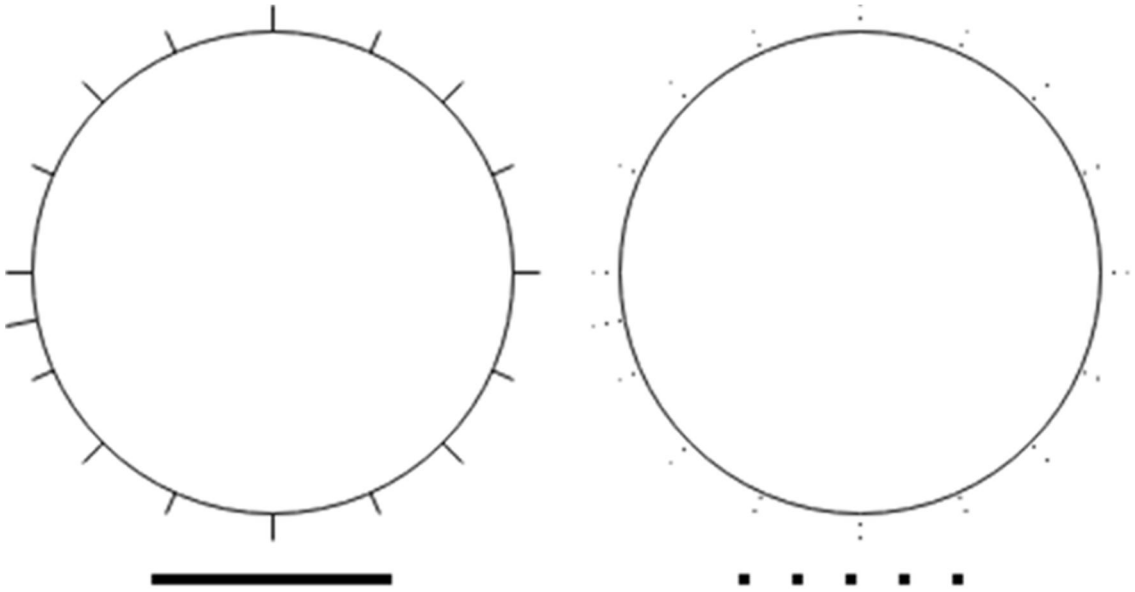
Chapter 10: Advanced shading techniques

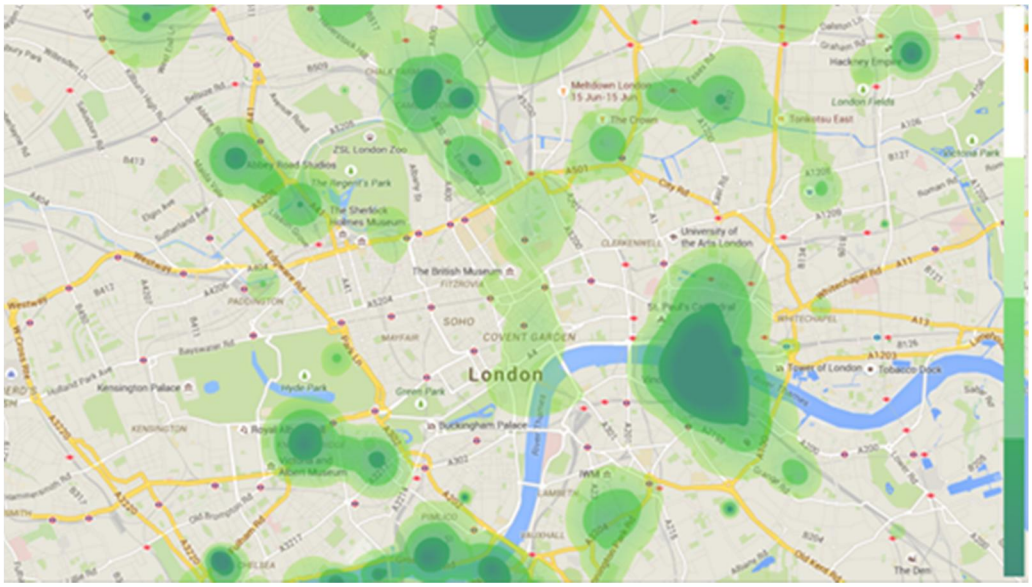


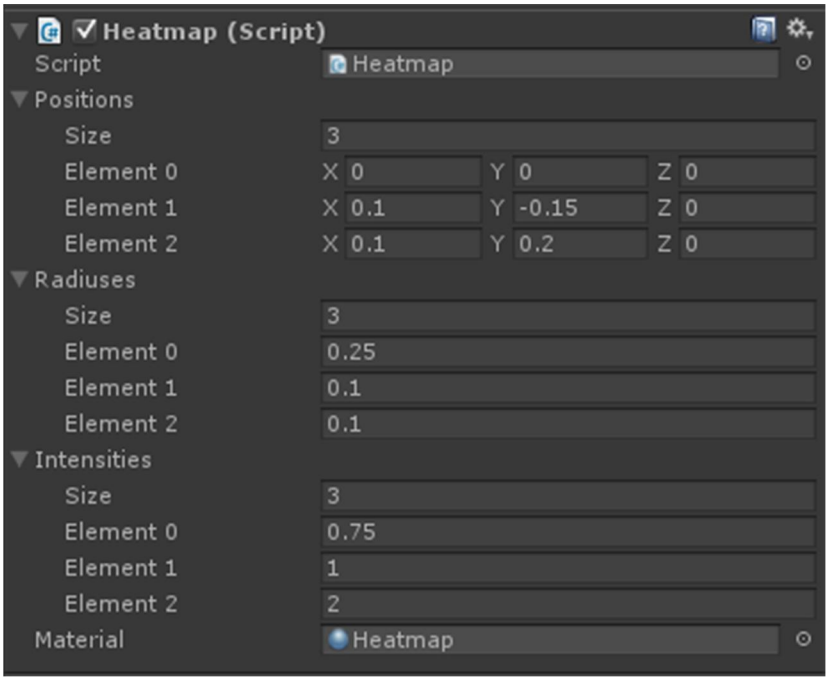

Luminace Value = 0.0
Luminace Value = 0.5
Luminace Value = 1.0


Scene
Textured
RGB
Persp
Inspector
BuiltinIncludes_MAT
Shader
Base (RGB)
Tiling
Offset
Select
Desaturate
My Color
Preview




Real fur: solid geometry
Shell fur: several spheres


London


Heatmap (Script)
Script
Heatmap
Positions
Size
3
Element 0
X 0
Y 0
Z 0
Element 1
X 0.1
Y -0.15
Z 0
Element 2
X 0.1
Y 0.2
Z 0
Radiuses
Size
3
Element 0
0.25
Element 1
0.1
Element 2
0.1
Intensities
Size
3
Element 0
0.75
Element 1
1
Element 2
2
Material
Heatmap