Lecture 9
Advanced Topics: Graphics and Lighting

98-127: Game Creation for People Who Want to Make Games (S19)

Written by Adrian Biagioli

Instructors:
Adrian Biagioli (abiagiol@andrew.cmu.edu)
Carter Williams (ncwillia@andrew.cmu.edu)

1 Objectives

By the end of this lesson you will be able to:

- Understand Unity's Material system, and the difference between Shaders and Materials
- Understand the concepts of Physically Based Rendering (PBR) as well as the inputs to Unity's Standard PBR shader
- Use Unity's Effects pipeline to compose complex post processing effects
- Bake lightmaps onto Unity scenes to improve performance
- Understand the basics of creating your own shaders via Unity's experimental Shader Graph system and Surface shaders

These lecture notes were written for **Unity 2018.3.8**.

2 What are Materials?

1. Creating a material in the project view
2. Material vs Shader
3. Manipulating Material Properties (textures, colors)

3 The Standard Shader

1. Discuss the standard shader from a high level – maybe talk about the history of shaders (lambert / blinn based specularity) and what motivates the Standard shader. **Define PBR (Physically Based Rendering)**
2. Big idea: The standard shader is a "meta-shader" that is able to express photoreal surfaces. Use custom shaders for non-photoreal stuff

3. Go over the material charts. Big idea: Albedo $\neq$ Color! Especially show effect of albedo on metals
4. To demonstrate PBR reflections, demonstrate changing the skybox to an HDR one

4 Finding PBR Textures

1. Find a texture from CC0Textures.com and a HDR from HDRIhaven.com, and show how to import all the maps (via GIMP)
2. Smoothness vs Roughness
3. Define HDR images (not a simple color, but an *irradiance*)
4. Demonstrate reflection probes in the room scene

5 Unity's Lighting panel and Global Illumination

1. Explain why we need to bake lighting (performance + complex prebaked lighting)
2. Bake lighting in the room scene
3. Put a big bright green cube in the middle of the scene and bake again
4. Discuss static / lightmap static
5. Point lights only support baked shadows

6 Adding Post-processing image effects

1. How to download post process stack (package manager, or asset store for v1)
2. Adding post process layers to the camera
3. Adding specific effects via a post process volume (have to create post process profile first)
4. Motion blur makes things look better for free :)
5. Remember to set the layer to the corresponding layer on the camera (show them how)

7 Writing custom Shaders: The surface shader

1. Surface Shader examples: https://docs.unity3d.com/Manual/SL-SurfaceShaderExamples.html
2. Surface Shader output struct: https://docs.unity3d.com/Manual/SL-SurfaceShaders.html

3. Write a basic rim lighting shader that writes to emission (also define emission if you havent) and then show the surface shader example page

Rim lighting example from unity docs:

```
Shader "Example/Rim" {
  Properties {
    _MainTex ("Texture", 2D) = "white" {}
    _BumpMap ("Bumpmap", 2D) = "bump" {}
    _RimColor ("Rim Color", Color) = (0.26,0.19,0.16,0.0)
    _RimPower ("Rim Power", Range(0.5,8.0)) = 3.0
  }
  SubShader {
    Tags { "RenderType" = "Opaque" }
    CGPROGRAM
    #pragma surface surf Lambert
    struct Input {
        float2 uv_MainTex;
        float2 uv_BumpMap;
        float3 viewDir;
    };
    sampler2D _MainTex;
    sampler2D _BumpMap;
    float4 _RimColor;
    float _RimPower;
    void surf (Input IN, inout SurfaceOutput o) {
        o.Albedo = tex2D (_MainTex, IN.uv_MainTex).rgb;
        o.Normal = UnpackNormal (tex2D (_BumpMap, IN.uv_BumpMap));
        half rim = 1.0 - saturate(dot (normalize(IN.viewDir), o.Normal));
        o.Emission = _RimColor.rgb * pow (rim, _RimPower);
    }
    ENDCG
  }
  Fallback "Diffuse"
}
```

8 Advanced: Using the Unity Shader Graph

1. Talk briefly about the scriptable render pipelines and how they are the future of unity rendering
2. Briefly show how to install the new pipelines (remember to enable experimental packages in the package manager)
3. Do the rim lighting example again, this time using shader graphs

9 Case Study: Legend of Zelda lighting

1. Discuss the toon shading, briefly discuss NDotL lighting and how smoothstep is important here
2. Discuss how the eyes are rendered: Start by demonstrating the shadergraph version of the pupil shader on a plane, then show the raw shader version (this will be the first time they see a vert/fragment shader). Talk about the depth buffer and how you can ignore it for artistic effects. Mention render order input in materials.
3. Show the LOZ BOTW shadergraph example: https://connect.unity.com/p/zelda-inspired-toon-shading-in-shadergraph