Game Engine Materials and Lighting

Making content look amazing

Programming – Computer Graphics

Contents

- Material and Lighting Recap

- Game Engine Materials and Lighting
  - Unity3D
  - Unreal Engine 4

Materials and Lighting Recap

- Materials represent the surface properties of a model and dictate how it appears
  - Colour
  - Texture
  - Shine
  - Gloss
  - Transparency
  - Special surface effects

- Material properties are used by shader code on a GPU when rendering geometry

- Lighting properties are combined with material properties by a BRDF shader to dictate the final overall look
  - Lighting usually uses global properties for a scene
  - i.e. all models in the scene use the same lights

Game Engine Materials and Lighting

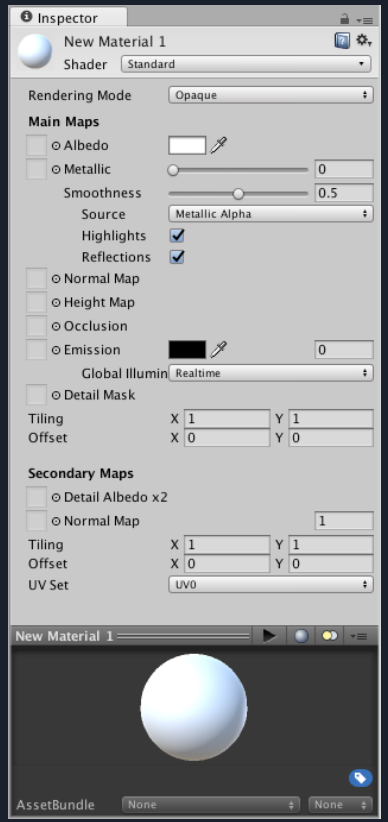
- Majority of game engines have a fixed material and lighting pipeline
  - All property transfers to the GPU are taken care of for us
  - Can often be modified with source code or plugins
- Many use a deferred lighting approach
  - Lighting is calculated globally for the scene
  - When geometry renders it can access the lighting data for each pixel fragment when calculating final output colour
- Most game engines now use physically-based material pipelines
  - Many use graph-based material editors rather than expose shader code, while some use über-shaders

Unity3D Materials

- Unity3D contains many built-in materials
  - FX
  - GUI
  - Mobile
  - Sprites / Particles
  - Toon
  - Unlit
  - Legacy

- The default material is an über-shader called the Standard Shader
  - Physically-based
  - Properties are exposed for edit via the inspector

- Custom materials can be created to use a custom shader
  - Shaders written using modified CG language in ShaderLab



Unity3D Materials

- Materials in Unity3D are a form of container that can be shared
  - Contain the Shader to use for the material
  - Contain the Textures and how they are applied
    - Tiling
    - Offsets
  - Contain the material properties
    - Albedo colour
    - Shine
    - Etc

- Shaders in Unity3D come in a few forms, written in ShaderLab
  - Surface Shader
    - Describes the final colour of a fragment, sampling from deferred pre-calculated lighting
  - Vertex and Fragment Shaders
    - Can be used if not needing to access lighting, for unlit or post-processing

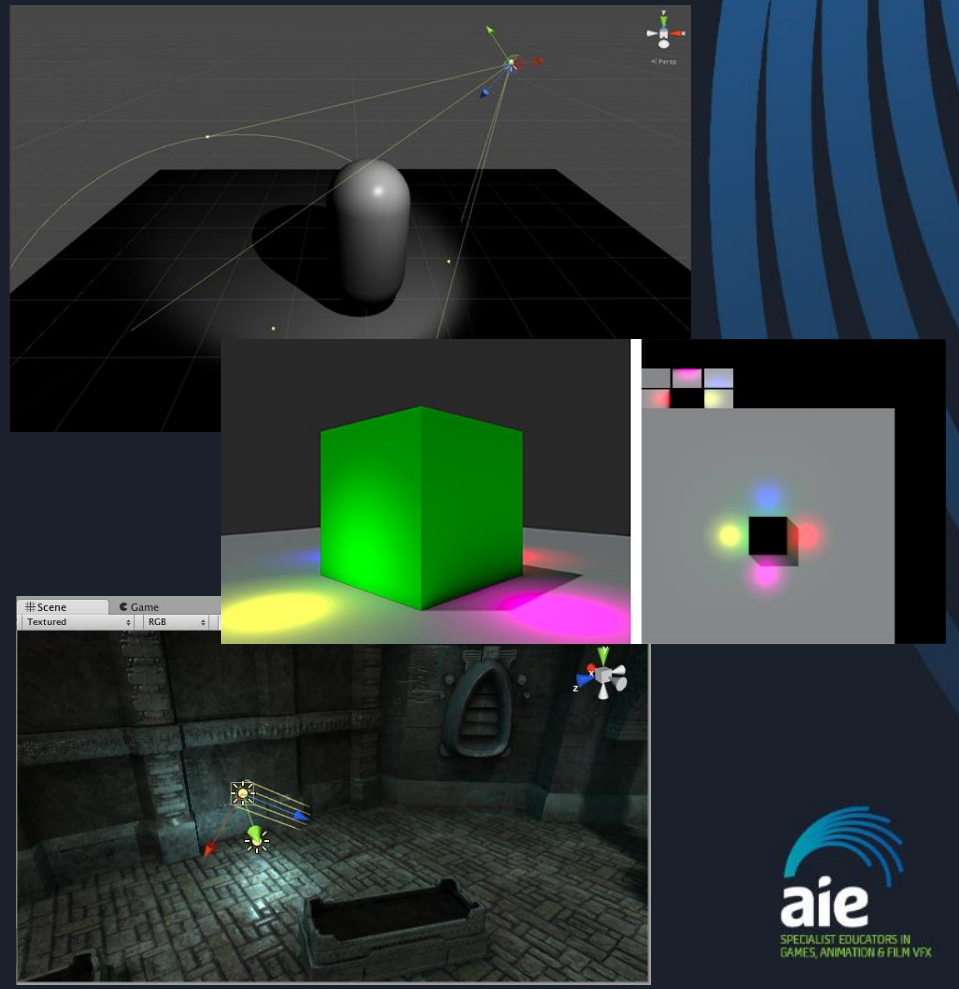
Unity3D ShaderLab

- ShaderLab is a simple language that Unity3D uses to define how a shader should work
  - Pre-defined inputs, outputs and methods
  - Actual shader code contained within a ShaderLab file is written using the CG shader language
- Lighting is pre-calculated
  - The ShaderLab simply calculates surface properties that wil be used by the lighting pipeline
  - Post-lighting modification can be made using a finalcolor modifier
- Unity3D now also includes a graph-based shader editor!

```
Shader "Example/Diffuse Texture" {
  Properties {
    _MainTex ("Texture", 2D) = "white" {}
  }
  SubShader {
    Tags { "RenderType" = "Opaque" }
    CGPROGRAM
    #pragma surface surf Lambert
    struct Input {
      float2 uv_MainTex;
    };
    sampler2D _MainTex;
    void surf (Input IN, inout SurfaceOutput o) {
      o.Albedo = tex2D (_MainTex, IN.uv_MainTex).rgb;
    }
    ENDCG
  }
  Fallback "Dif
}
```

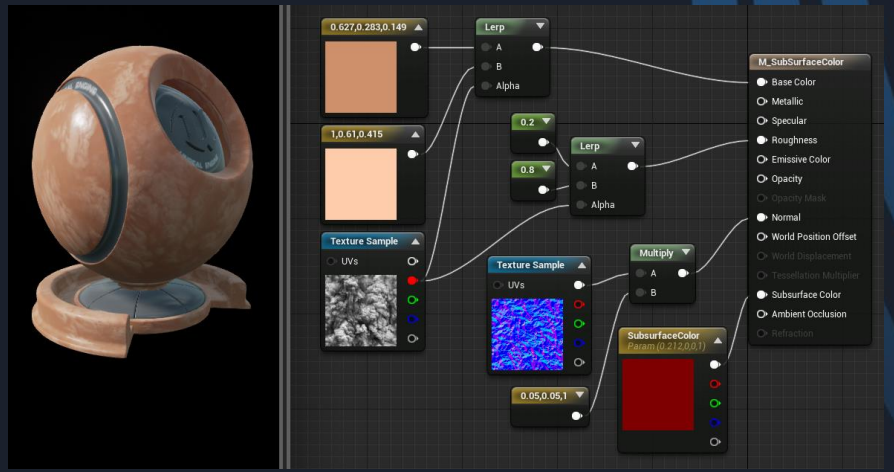
Unity3D Lighting

- Lighting is both real-time and pre-computed
  - Real-time lighting
  - Lightmaps
  - Pre-computed “real-time” Global Illumination
  - Combining techniques can have performance hits
- Basic real-time light types available
  - Point, Spot, Direction
  - All can be shadow casters



Unreal Engine 4 Materials

- Materials created using a graph-based editor
  - Internally creates non-editable HLSL shaders
- Material controls shader logic
  - Also stores textures and material properties
  - Properties can be made editable by users of the material instance
    - One material, many texture variations
- Layered materials are possible



aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Unreal Engine 4 Lighting

- Supports real-time and pre-computed lighting
- Basic real-time light types, plus a Sky Light
  - Point, Spot, Direction
  - SkyLight samples the scene to calculate fake global illumination to apply to objects as a form of ambient light

- Multiple pre-computed global illumination methods are available
- Lighting applied as a deferred pre-computed step to materials
  - Lighting applies globally to a scene, or to objects within a light's bounds

Summary

- Engines wrap up the material and lighting requirements for us
  - Rarely need to write shader code
- Most engines pre-compute lighting in a deferred step
  - Materials just dictate the surface properties, the engine combines this with the lighting

Further Reading

- Unreal Engine 4 Documentation: Materials
  - https://docs.unrealengine.com/en-us/Engine/Rendering/Materials
- Unreal Engine 4 Documentation: Lighting the Environment
  - https://docs.unrealengine.com/en-us/Engine/Rendering/LightingAndShadows
- Unity3D Manual: Creating and Using Materials
  - https://docs.unity3d.com/Manual/Materials.html
- Unity3D Manual: Lighting
  - https://docs.unity3d.com/Manual/LightingOverview.html