Game Engine Render Pipelines

An introduction

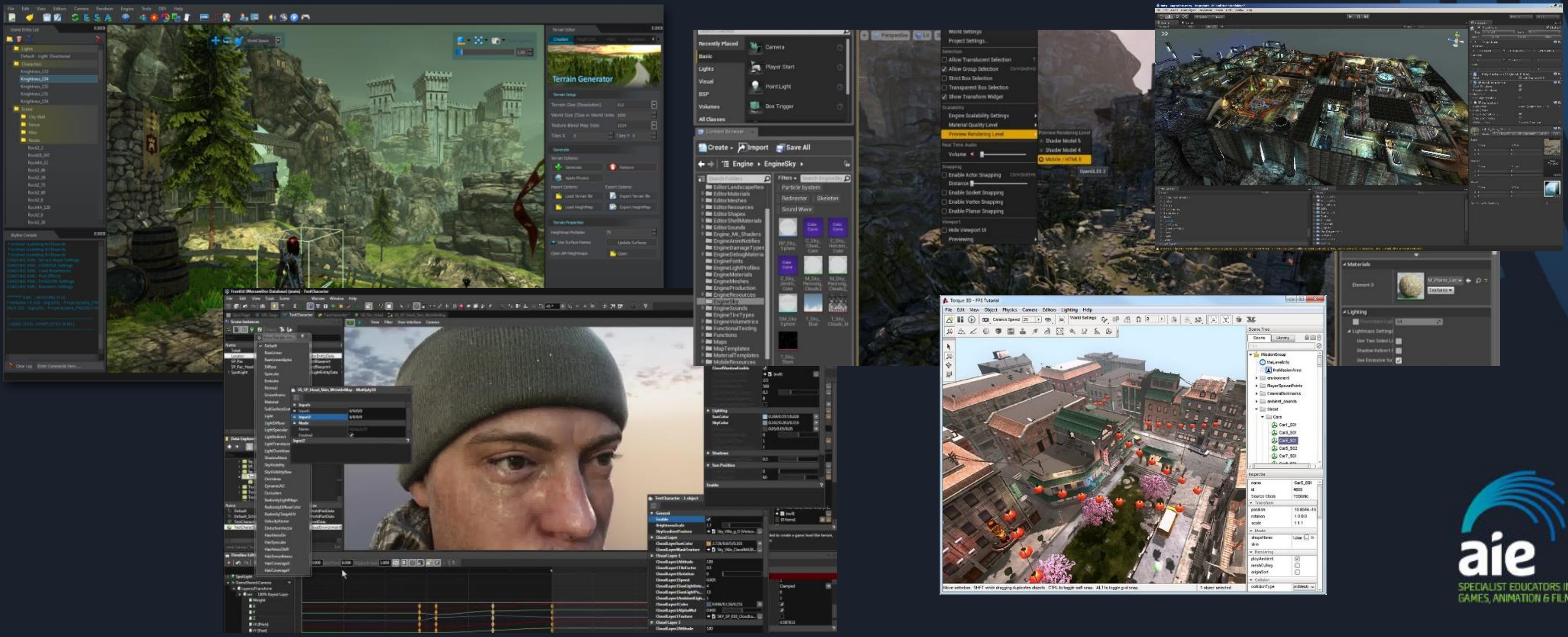
Programming – Computer Graphics

Contents

- Game Engines
- Case Study
  - Unity3D
  - Unreal Engine 4
- Considerations

Game Engines

- There are lots of game engines used in the industry today



Game Engines

- All engines wrap up lower-level APIs
  - OpenGL
  - Vulkan
  - DirectX
  - OpenGL ES
  - Metal

- Their job is to abstract away the APIs so that you can just deal with gameplay and work cross-platform
  - But we can also access many low-level features
  - Some engines restrict the way we can use the low-level APIs

- Many implement hybrid Deferred Rendering systems

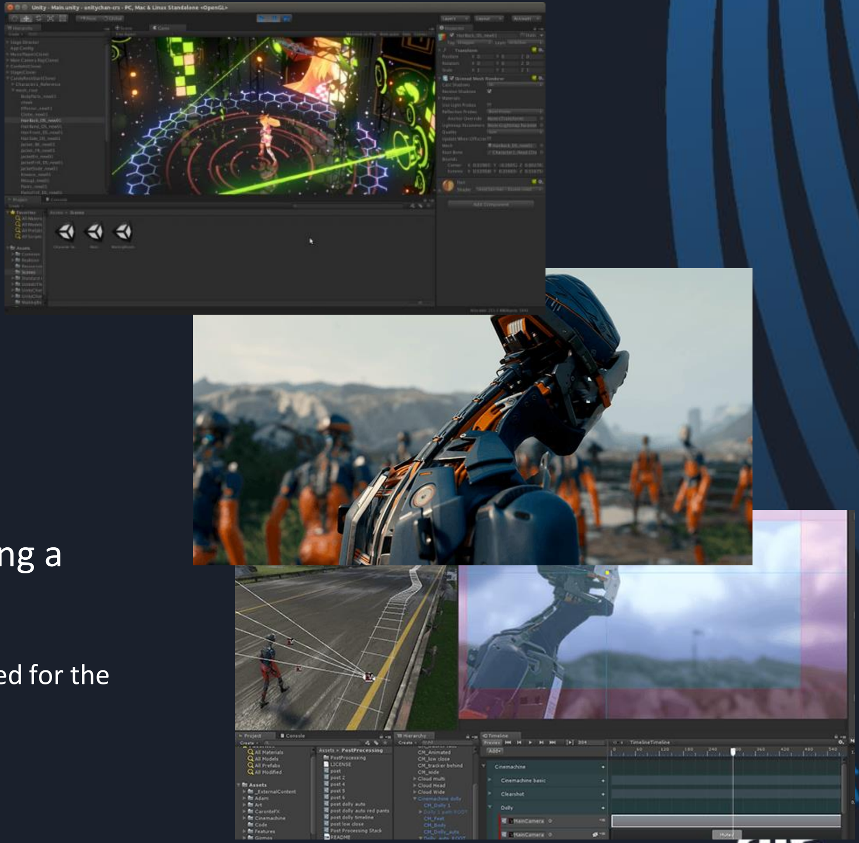
Case Study: Unity3D

- Many APIs
  - Vulkan
  - Metal
  - DirectX 12
  - nVidia VRWorks / AMD LiquidVR

- Hybrid deferred real-time Global Illumination and Physically-based Shaders
  - Extendable Post-processing pipeline

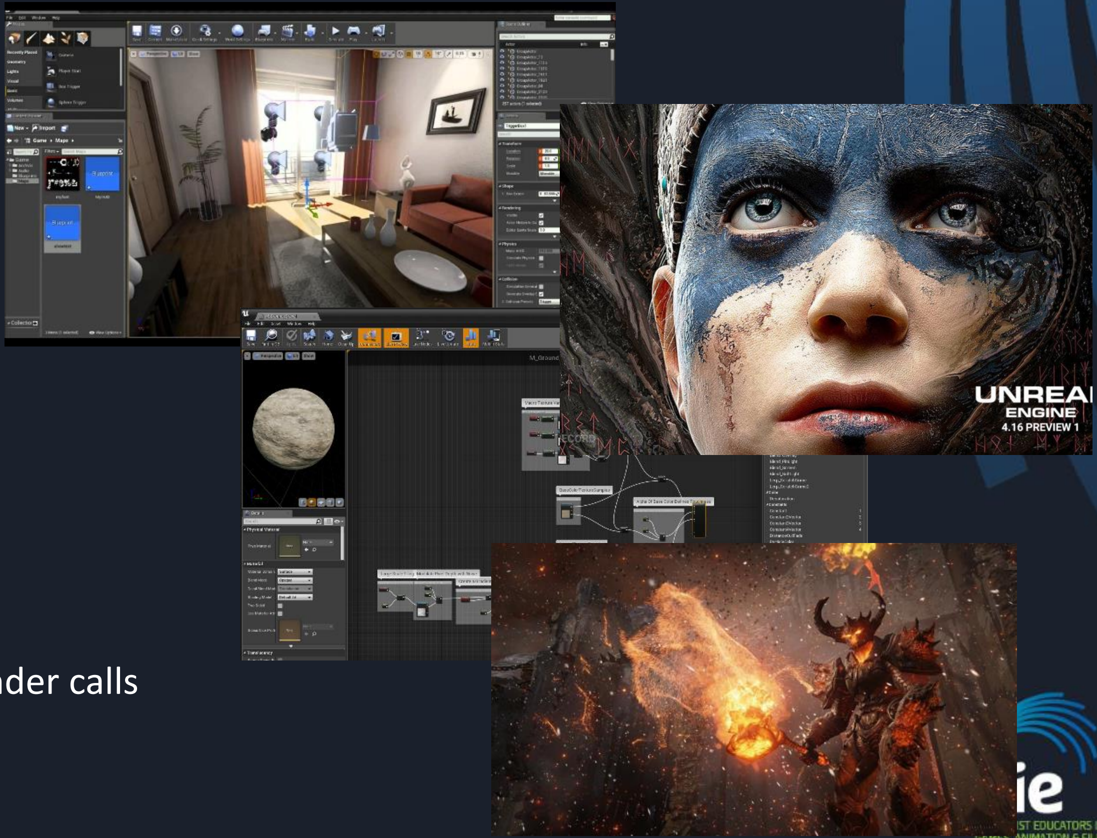
- Shaders written with a modified CG language or using a graph editor in latest versions of the engine
  - Surface Shader
    - Shaders applied after universal lighting as been calculated for the whole scene

- No hardware instancing – geometry is batched



Case Study: Unreal Engine 4

- Many APIs
  - Vulkan
  - Metal
  - DirectX 12
- Visual shader graph editor
  - Lighting calculated as a global solution using Deferred rendering
- Advanced global illumination and Post-processing effects
- GPU particle pipeline
- Supports hardware instancing for render calls



Considerations

- Even though engines wrap up the low-level APIs we still need to keep good render practices in mind
  - Lazy use of scenes and materials will cause substantial performance hits
  - Try to reuse material types using different parameters to reduce GPU cache updates
  - Complex lighting and excessive shadow casters / receivers can kill performance
- Profile render performance often
  - Can help identify bottlenecks
- Prototype materials / shaders first, then optimize
  - Not all engines allow you to write custom shader code
- In essence, all engines can do the same things graphically
  - Best not to “fanboy” over an engine’s capabilities; they change often and are outclassed often!

Summary

- Engines simply wrap up low-level API calls
  - They don’t change how the GPU works
  - All work in similar ways with the same capabilities since they use the same APIs and the same hardware

- Most engines wrap up shaders with graph-based editors
  - Some allow access to shader code for optimization

- Many perform lighting globally and apply it as a deferred step
  - Allows the engine to implement advanced global illumination techniques

Further Reading

- Unity3D Graphics Rendering
  - https://unity3d.com/unity/features/graphics-rendering

- Unreal Engine 4 Documentation: Graphics and Rendering
  - https://docs.unrealengine.com/en-us/Engine/Rendering
  - Video: “Reflections” real-time ray tracing Star Wars short film
    - https://www.youtube.com/watch?v=J3ue35ago3Y