Game Engines

Overview

- Game engines are a significant part of the modern games industry

- Middleware
- Game engines
- Why use an engine?
- Unreal and Unity
- Why don’t we use engines in this course?

Middleware

- Some parts of a game are difficult to build, and also not very game specific
  - Rendering, physics, sound, front-end tools, etc.
- Many developers created central teams to build shared technology, allowing them to spread the development costs across several games
- Smaller developers didn’t have or couldn’t afford central development
- A market was born

A history of middleware

- Around the launch of the PS2 (2000), several companies began to license their technology:
  - Rendering
    - Renderware, Gambryo
  - Physics
    - Havok, Mathengine
  - Movie player
    - Bink
  - Sound
    - FMOD
- Mostly single purpose libraries
  - Varying effort involved in integration with existing games:
    - Bink (memory, file I/O, rendering)
    - Havok (all of the above, loading, gameplay / AI, etc)

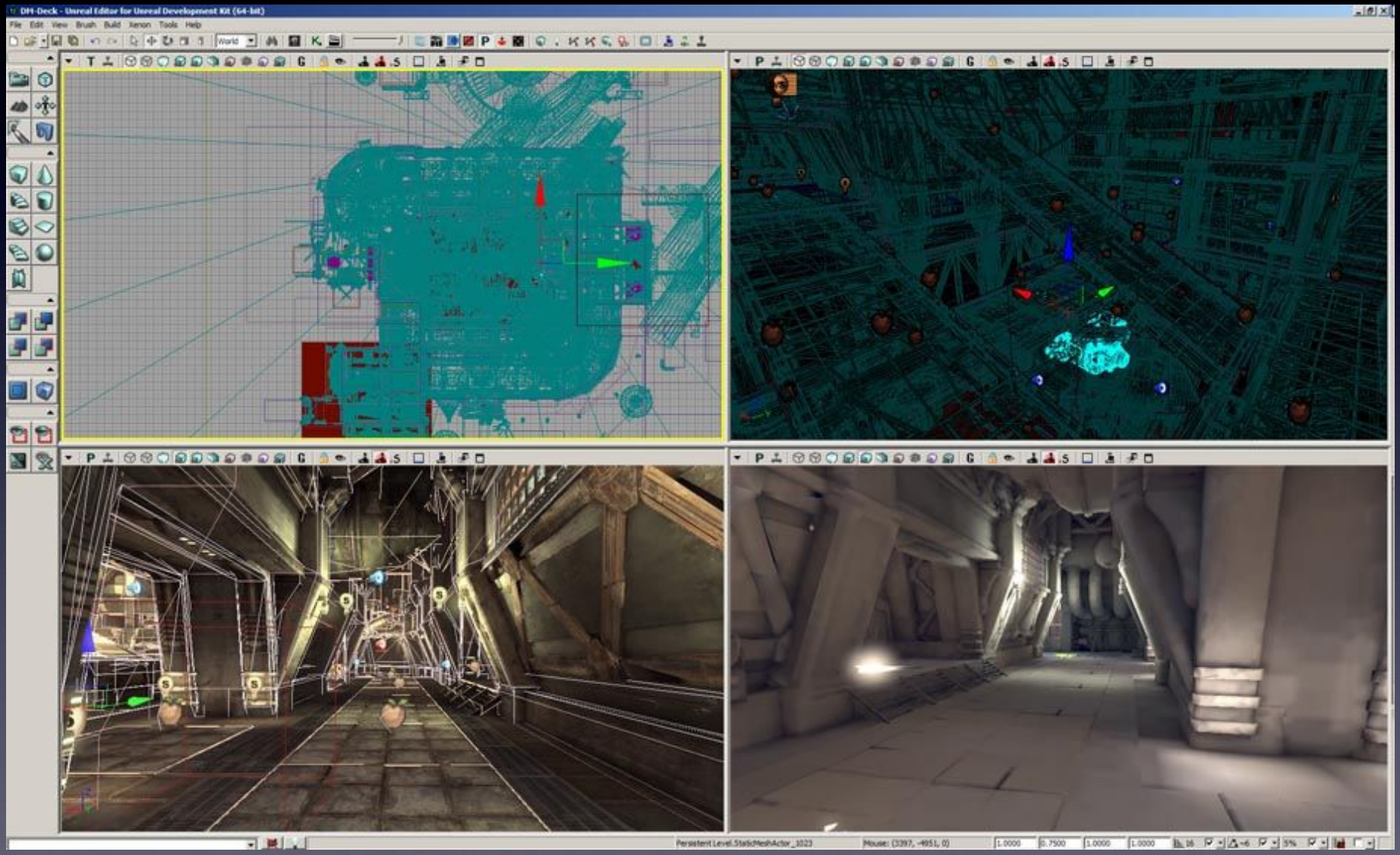
Game engines

- The Quake engine (1996)
  - PC only
  - 3D hardware acceleration added later
  - Spawned many derivative engines
    - Quake 2 engine: basis for Half-Life and Source, Call Of Duty
- Unreal Engine (1998)
  - Modular architecture
  - UnrealScript
- CryEngine (2004)
- Unity (2005)
  - Multi-platform: Web plugins, PC, mobile, consoles

Why use an engine?

- Many hard problems are solved for you
  - Content tools and pipeline
  - State-of-the-art rendering technology
  - Multi-platform support
  - Cross-domain integration
  - Easy prototyping of new game ideas
  - Trade some performance and flexibility for development time
- Buy vs build
  - Many developers license an engine
  - Others use internally-built engines (e.g. Frostbite)
- Can aid recruitment / retainment (de facto standards)
- 'Standalone' console games are now rare

Unreal Engine



History of Unreal Engine

- Dates back to 1998, used for Unreal (single-player FPS) and Unreal Tournament (fast-moving multiplayer FPS)
- Mostly the work of Tim Sweeney, who is still CTO
- Rapid prototyping tools (e.g. CSG) made it popular
- UnrealEngine 2 (2002, America's Army): cinematics, Maya/Max integration, Xbox support
  - Ported to 3DS in 2011!
- UnrealEngine 3 (2006, Gears Of War): PS3/Xbox 360, iOS, WiiU, HTML5, async rendering, shaders
- UnrealEngine 4 (2014): real-time lighting, faster iteration, script debugging

Unreal Engine highlights

- Unreal Editor, “Play In Editor” feature
- Multithreaded rendering engine
- Level construction tools
- Scaleform UI toolkit (replaced with their own UMG)
- PhysX physics integration
- Scripting
  - Kismet and UnrealScript in UE3
  - Blueprints in UE4
- UE3 was used to ship more than 300 games
- UE4 was used for the latest Gears of War
- Licensing model has opened up considerably

Unreal Engine highlights

- Built in Level/Texture streaming, asset management
- Large amount of sub(licensing) to middleware
  - LiveCode, Quixel, Maya Live Link … etc
- Blueprints compile to native C++!

Unity

Unity highlights

- Multiplatform
  - Web, Windows, Mac, Linux
  - iOS, Android, Blackberry 10, Windows Phone 8
  - Consoles (comparatively late - 2013+)
- Rapid iteration through integrated editor
- Scripting through C#, Javascript or Boo (Python)
- Broad support for different game genres
  - Pathfinding, animation, 2D components, audio, physics, terrain, visual effects
- Not an AAA competitor to UnrealEngine until recently:
  - Unstable / inadequate framerates, no console support
  - Shovelware reputation – paradoxically because of ease of use, asset store
  - But there are now games where you wouldn't guess they were in Unity without seeing the legals

Unity Game Objects

- GameObjects are containers for Components
  - Tag, Layer, Name, Static flag
  - AddComponent(TypeName)
- Components
  - Mesh imported from 3D package
  - Animation controller
  - Box collider
  - Write your own in C# very quickly

Unity Nativization

- Moving towards a ECS-style system (Untiy DOTS)
- Burst compiler transforms .NET Bytecode/IL to native assembly code

Why don’t we use an engine in this course?

- You need to know how an engine works to use it well
  - Working at a lower level evens the playing field
- Game engines are an important tool for game development, but they’re rarely the only tool you’ll need
  - Depending on the game you’re building, an engine may need significant modification
    - Everyone used to rewrite the UnrealEngine animation system the moment they licensed it
  - Understanding how the engine is built is critical
  - You'll still need to write bespoke tools, even if they plug into an engine-provided pipeline
- It will make you a better programmer!
  - In the games industry, you often need to know the low level details!

Summary

- Most games these days are built with an engine
- Unreal and Unity are the big ones, but there are other engines available
- Lots of internal engines too
- Both Unreal and Unity have free versions that are worth studying
- Plenty of high-quality free/cheap training resources online
- Consider a company's tech and engine usage when deciding whether to apply for a job