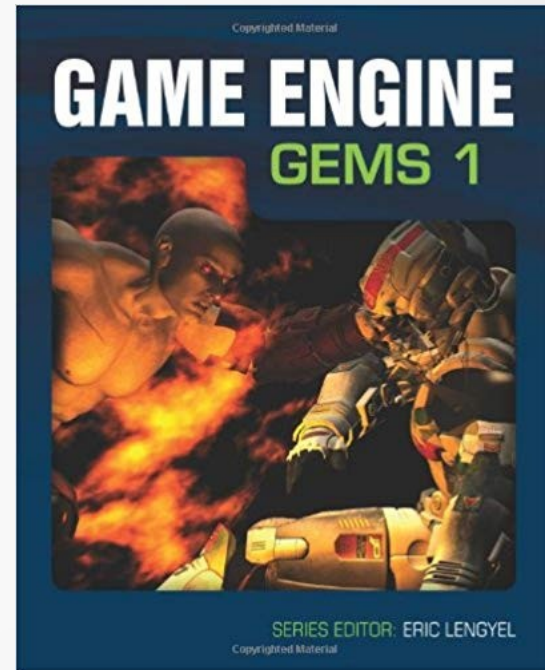
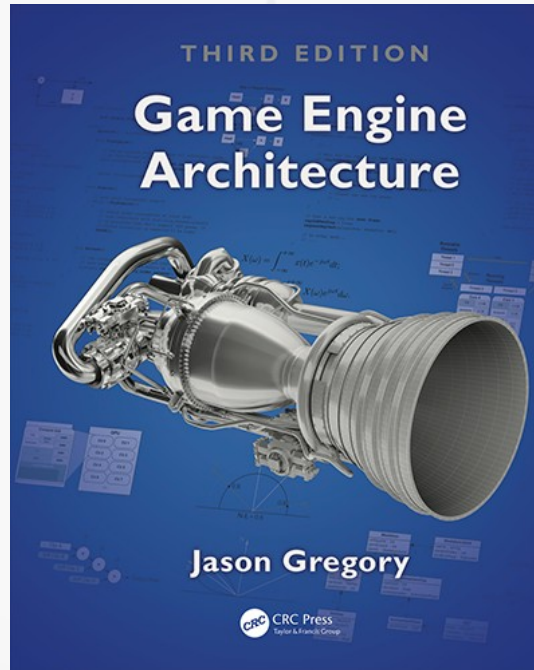
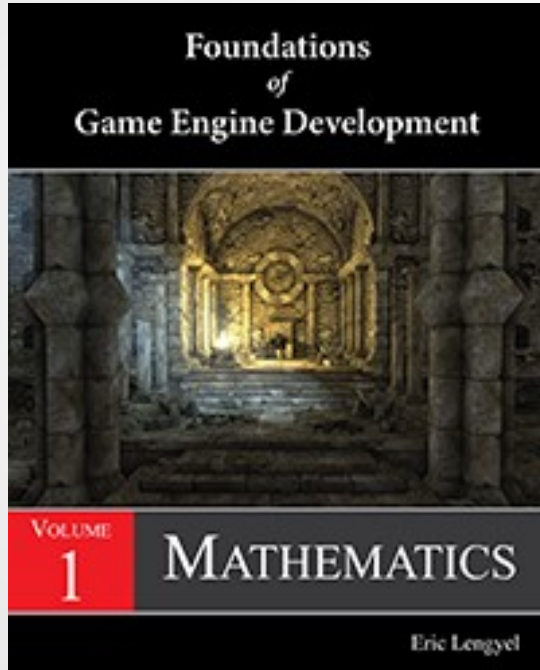


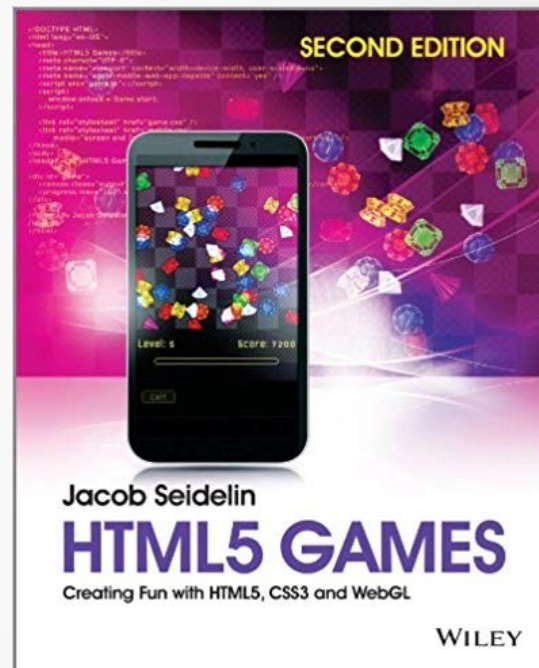
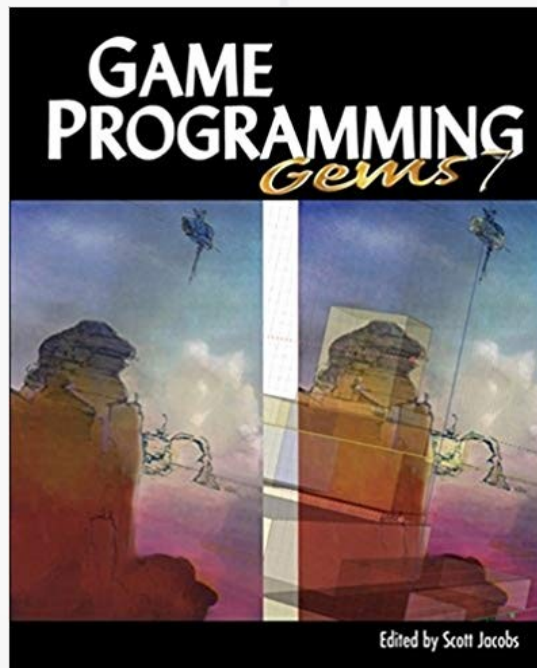
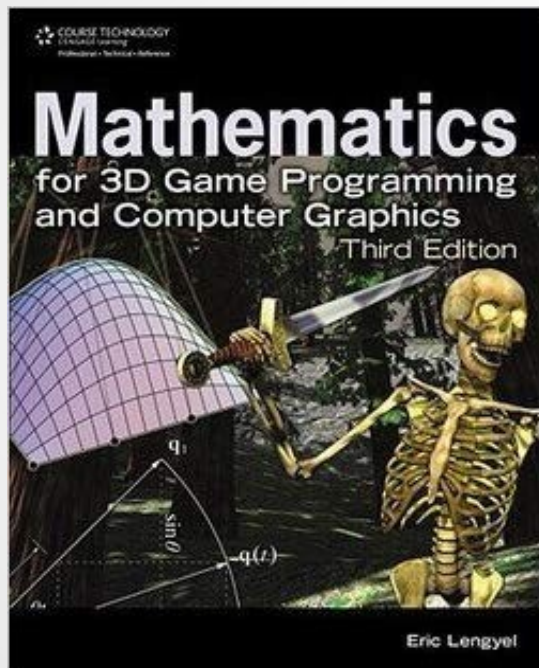
Game Components & Architecture



Book Resources

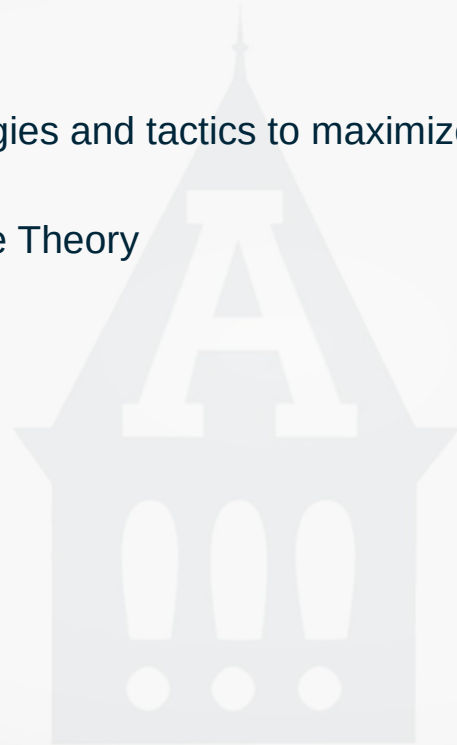


Book Resources



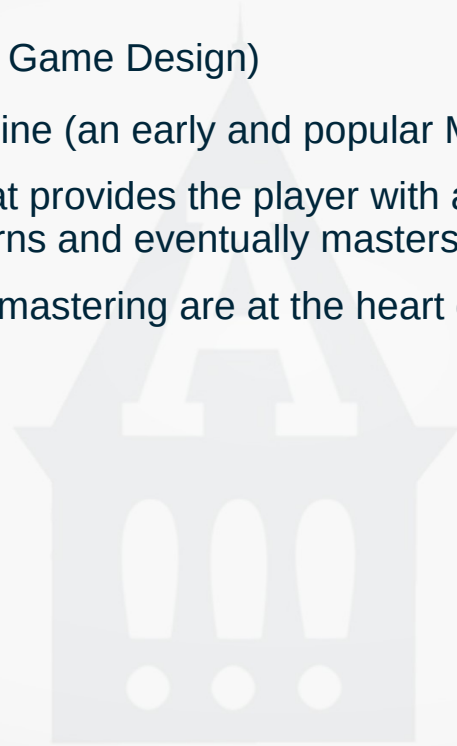
What is a Game?

- Game Theory
 - Multiple agents select strategies and tactics to maximize their gains within a framework of a well-defined set of game rules
 - This class is not about Game Theory



What is a Game?

- Ralph Koster (A Theory of Fun for Game Design)
 - Lead designer for Ultima Online (an early and popular MMO)
 - An interactive experience that provides the player with an increasingly challenging sequence of patterns which he or she learns and eventually masters
 - He argues that learning and mastering are at the heart of what we call “fun”



Game Development

- Games are software, therefore game development is software development
 - Software engineering (design & development methodology)
 - Programming
 - Content generation (art, models, levels, etc)
 - Source code control
 - Team collaboration & tools
 - Project management & tools
 - Budgeting, sales...
- A key difference in game development vs general software development might be the wide scope of topics involved, not necessarily the complexity or performance considerations

Who is Involved?

- Engineers (obviously the most important!)
 - Technical Directors
 - Engine Developers
 - Gameplay/Scripting
 - Tools
- Game Designers
- Producers
- Publishers & Studios
- Content Providers
 - Concept Artists
 - Texture Artists
 - Lighting Artists
 - 3D Modelers
 - Animators
 - Motion Capture Actors
 - Sound Designers/Composers



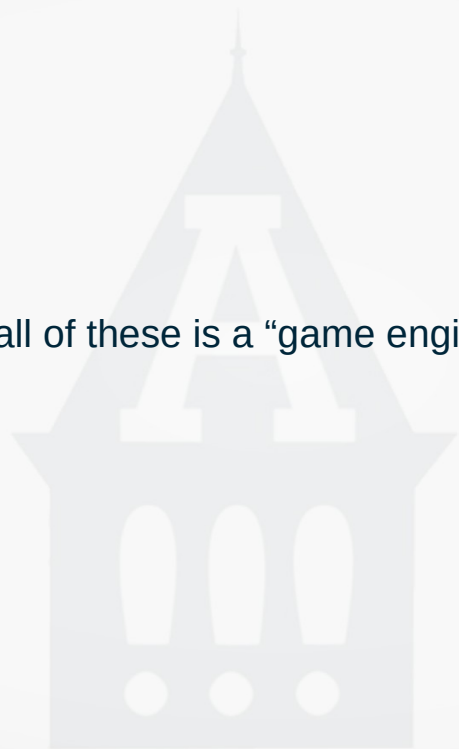
What is a Game Engine?

- Write a game...
 - Write another game...
 - Write another game...
 - Write another game...



What is a Game Engine?

- Write a game...
 - Write another game...
 - Write another game...
 - Write another game...
- What you use in common among all of these is a “game engine”

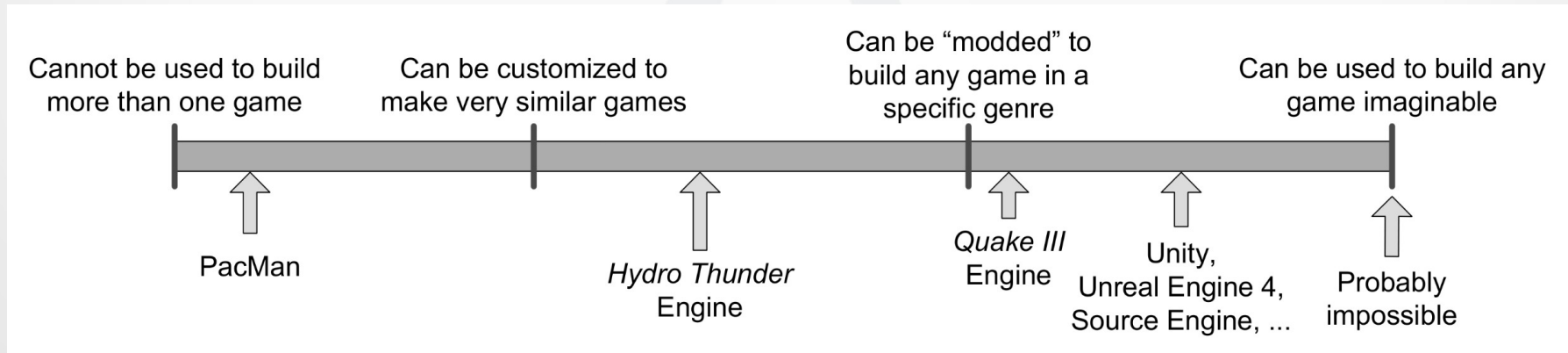


Game Engine

- Concept probably started in mid-1990's with Doom
 - Designed to separate core software components and assets
 - Started with community developing “mods” for the game (also DeHackEd)
 - Evolved into the full concept of a “game engine” as it was licensed
 - Further evolution with Quake. Designed from the ground up to be extendable, using a scripting language called, “QuakeC”
- Now we have fully supported and commercialized Game Engines
 - Unreal
 - Unity
 - Many others...

Game Engine

- A data-driven architecture is key to developing a true game engine
 - Want to eliminate hard-coded: logic, rules of play, characters, weapons, and other assets




Game/Game Engine Categories

- First/Third Person Shooters
- Adventure, RPG
- Action RPG
- Platformers
- Fighting Games
- Simulation, Racing Games
- Real-Time Strategy
- Massively Multiplayer Online Games



Major Game Components

- 
- Rendering
 - Input
 - Audio
 - AI Framework
 - Networking
 - Physics
 - Scripting
 - Resource (asset) management
 - Platform abstraction
 - Memory Management
 - Event Model
 - Menuing
 - Configuration
 - Database
 - Localization
 - Testing, Profiling, & Debugging & Tools
 - Content Tools
 - Level editors
 - Modeling, Texture
 - Pre-Computed Simulations

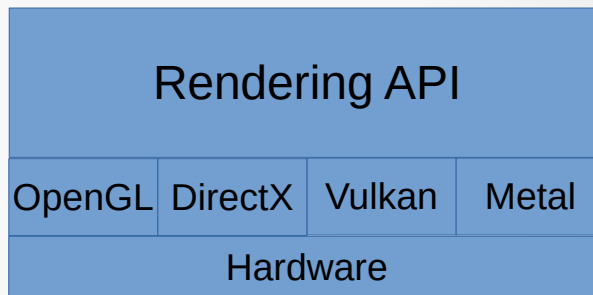
Resource Management

- System responsible for manipulation of content/assets
 - Textures
 - Models
 - Animations
 - Geometry
 - Sounds
 - Strings (e.g., language and localization)
- Usage optimization
- Asset streaming (prediction)



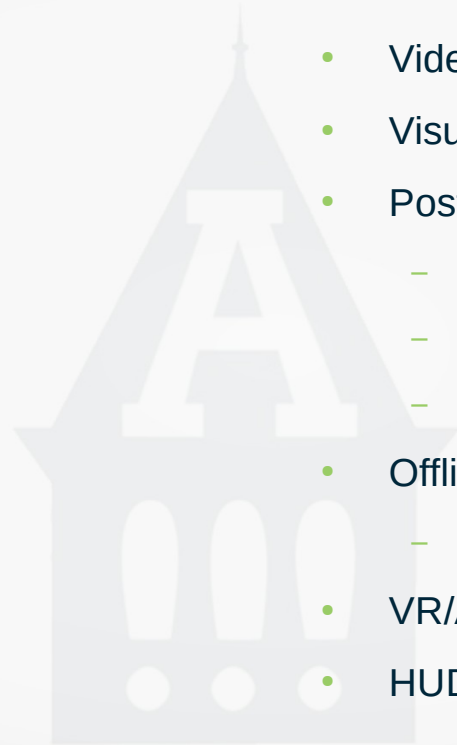
Rendering - Overview

- This is what draws things to the screen
- Interface/abstraction to low-level API
 - Game systems use this interface, do not directly use low-level API
 - What is a low-level API: OpenGL, DirectX, Vulkan, Metal
- Configuration
 - Screen resolutions
 - Aspect Ratio
 - Color depth
- Font Rendering



Rendering - Features

- Scene Graph
- 2D Rendering
- 3D Rendering
- Camera model
 - Positioning
 - Depth of field
- Animation
- Particles
- Video
- Visual Effects
- Post Processing
 - Film grain
 - Motion blur
 - 4K checkerboard reconstruction
- Offline rendering
 - Shadows, lightmaps
- VR/AR
- HUD



Audio

- This is what makes your speaker(s) do something
- Interface/abstraction to low-level API
- Game systems use this interface, do not directly use low-level API

- Features

- Output to standards (e.g., all things Dolby)
- Point source spatialization
- Propagation
- Mixing
- Multi-channel (4.0, 5.1, 6.1, 7.1)

- Uses

- In-Game effects
- Music, dynamic
- Menu clicks
- Screen transitions

Input

- Input Sources

- Keyboard
- Mouse (+motion smoothing)
- Game Controllers
- Sound (voice recognition)
- Video (Eye Toy, Kinect - RIP)

- Technical Approaches

- Publisher – Subscriber model
- Input queue and pull from queue
- Multiple queues, one for each input type



Networking

- Uses

- Sharing of game data
 - characters, stats, saved games, etc.
- Social communities
 - chat, mail, clans, guilds
- Multiplayer
 - Client/Server
 - P2P
 - MMO (still client/server)

- Issues/Technical

- TCP/IP using TCP or UDP
- Bandwidth: how much data
 - limits # of players
- Latency: round trip time
 - limits speed of interactions
- Packet Loss
 - Affects game types and experience of players
- Synchronization
 - Maintaining a consistent world state across all players

Physics

- Used to describe the movement of objects in the game environment
- Terms/Uses
 - Rigid Body Physics : Think of the ubiquitous crate
 - Rag Doll Physics : Describe non-animated movement of models
 - Vehicle Simulations
 - Cloth Simulation
 - Explosion Simulation : Instead of artists manually animating
 - Collision Detection : Detect when objects interact. Used to be part of the graphics part of the engine, but not really anymore.
- Doesn't have to be realistic, instead it needs to be **fun**
- JavaScript Physics Library : <http://brm.io/matter-js/>

Scripting

- Loosen coupling between the gameplay and the core game services
 - Game services (“engine”) usually written in C/C++
 - Gameplay likely written in something else; but not necessarily
- Think of it as the “glue” between the game and the game engine
- Examples of game scripting languages
 - JavaScript
 - C#
 - Python
 - Lua
 - Custom (e.g., GOAL, UnrealScript)
 - GOAL: https://en.wikipedia.org/wiki/Game_Oriented_Assembly_Lisp
 - UnrealScript : https://en.wikipedia.org/wiki/Unreal_Engine#UnrealScript
 - C++

Artificial Intelligence

- General: Collection of techniques used to produce the illusion of intelligent behavior in non-player controlled game characters
- Possible Inclusions (in an engine)
 - Actions – The gameplay/engine coding interface
 - Information gathering – what can be sensed by the components
 - Goals, planning, decision making
 - Navigation, path finding
 - Finite State Machines
- Lots of approaches to AI, whatever works

Other Components/Systems

- Event System (spatial game triggers)
- Menuing System
- Configuration (e.g., keyboard mapping)
- Database System (persistent data)
- Localization
- Testing, Debugging, and Performance Tools
- Memory Management
- Platform Abstraction + Platform Specific (think PC, console, mobile)
- Analytics



Game Development Frameworks & Engines



Frameworks

- MonoGame (OpenSource XNA Framework)
 - <https://www.monogame.net>
 - PC, Mobile, Console
 - Bindings: C#
- Simple Direct Media Layer (libSDL)
 - <https://libsdl.org>
 - PC, Mobile
 - Bindings: C, C#, Lua, Python, Rust, others..
- Simple Fast Multimedia Library (SFML)
 - <https://www.sfml-dev.org>
 - PC, Mobile (experimental, but in good shape; so they say)
 - Bindings: C, C#, Java, Python, Rust, others...

Engines

- Unreal Engine
 - <https://www.unrealengine.com>
 - PC, Web, Mobile, Console, VR
 - Bindings: C++
 - Other: Blueprints – Visual Scripting
- Unity
 - <https://unity3d.com>
 - PC, Web, Mobile, Console, VR
 - Bindings: C#
 - Other: playMaker (third party) – Visual Scripting

Engines

- CryEngine
 - <https://www.cryengine.com>
 - PC, Console, VR
 - Bindings: C++
 - Other: Flowgraph – Visual Scripting
- Open 3D Engine
 - <https://www.o3de.org/>
 - Successor to Amazon Lumberyard, itself based on CryEngine, but I believe an “all new” engine
 - Windows, Linux, macOS (coming), Android (coming), iOS (coming)
 - Bindings: Script Canvas, Lua