Ch 1: What is Game Programming Really Like?
Ch 2: What’s in a Game?

Quiz #1
Discussion
Developing a Game

Game Architecture

Resources:
Chapter 2 (Game Coding Complete)
What was your last game architecture like?

- Probably didn’t worry about
  - efficiency
  - Graphics optimization
  - Memory optimization

- Why are these important?

- How about modularity of the code?

Lesson: many of the engines you work with will optimize for certain scenarios but if you are working on a AAA title that doesn’t fit the norm, you will need to override the standard way engines do things.
Example Game Architecture

Based on Figure 2.1 from Mike McShaffry Game Coding Complete
Example Game Architecture

Based on Figure 2.1 from Mike McShaffry Game Coding Complete
Game Application Layer (usually handled by an Engine)

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Application Layer (usually handled by an Engine)

Based on Figure 2.2 from Mike McShaffry Game Coding Complete

DEVICES
• Input
• Files
• RAM
• Time

• input devices
• Did you ever look at a device driver code?
• Abstraction of input to actions:
  • device driver input -> game actions
  • send game actions to game logic subsystem
Game Application Layer
(usually handled by an Engine)

Based on Figure 2.2 from Mike McShaffry
Game Coding Complete

- loading/saving game state
- assets loading
- resource caching
- memory management
- Virtual memory use

Is memory still an issue?
Game Application Layer (usually handled by an Engine)

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Application Layer
(usually handled by an Engine)

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Application Layer
(usually handled by an Engine)

- Devices:
  - Input
  - Files
  - RAM
  - Time

- OS:
  - Language
  - DLL
  - Threads
  - Network

- Game Lifetime:
  - Core Lib
  - Main Loop
  - Init & Shutdown

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Application Layer
(usually handled by an Engine)

Game Application Layer

Based on Figure 2.2 from Mike McShaffry's Game Coding Complete

- Queuing input
- Tick game logic
- Keep simulation time
- Events/Triggers

GAME LIFETIME
- Core Lib
- Main Loop
- Init & Shutdown
Game Application Layer
(usually handled by an Engine)

- **Init:**
  - Initialize game logic
  - Initialize game views
  - Reset simulation time
  - Handshake network, input devices

- **Shutdown:**
  - Memory de-allocation
  - Close files, network, etc.

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Example Game Architecture

Based on Figure 2.1 from Mike McShaffry Game Coding Complete
Game Logic

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Logic

- What are Data Structures?
- Common to Game Design
  - Arrays
  - Linked Lists
  - Hash Tables
  - Trees, B-Trees
  - Graphs
  - Stacks and Queues
Game Logic

Based on Figure 2.2 from Mike McShaffry Game Coding Complete

- Basic physics
  - What does that include?
- Collision
  - Why is that important?
- Mostly handled by the engine
Game Logic

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Logic

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Game Logic

Based on Figure 2.2 from Mike McShaffry Game Coding Complete
Example Game Architecture

Based on Figure 2.1 from Mike McShaffry Game Coding Complete
Based on Figure 2.3 from Mike McShaffry Game Coding Complete
Game View: Human

Display
• 3D
• UI
• Video

Audio
• SFX
• Music
• Speech

Options

Process Manager

Input Interpreter

Based on Figure 2.3 from Mike McShaffry Game Coding Complete
Based on Figure 2.3 from Mike McShaffry Game Coding Complete
Game View: Human

Based on Figure 2.3 from Mike McShaffry Game Coding Complete
Game View: AI

Based on Figure 2.6 from Mike McShaffry Game Coding Complete
Flexibility

Based on Figure 2.1 from Mike McShaffry's Game Coding Complete

---

Diagram:

- Game Logic
- Game View
- Game Application Layer
- Operating System/Hardware
- Game State
- Actions
- Input/Output
- Player AI
- Player over network
Game Engines

What engines have you worked with?
Components (coding and scripting)

- Libraries, e.g. XNA
  - API that you can use
  - Need to create the art assets
- Engines, e.g. Unreal Tournament
  - Scripting language
  - Can create modifications by extending their classes & telling the engine to use your modification instead
  - Attach scripts to objects or entities
Assets (Art and Design)

❖ Models and Animation

❖ Level Design: done in a tool they give you
❖ Models: typically done in 3Dstudio max or Maya
❖ Animation: typically done in 3Dstudio max or Maya
❖ Frames mapped to an animation cycle
❖ Exported to a game engine format

❖ Textures

❖ Music & Audio

❖ Composed in MIDI or WAV files
❖ 3D audio
❖ Interactive audio patterns
Architecture of Game Engines

Database/Resources:
- Game State
- Inventory
- Scene Graph

Systems:
- Character Controller
- Physics Engine
- Camera Controller
- 3D Audio Engine
- Game Rules
- Action Handlers
- Rendering Engine
- Weapons/Objects
- Game Rules
Architecture of Game Engines

Database/Resources:
- Game State
- Inventory
- Scene Graph
- Weapons/Objects

Processes and Methods:
- Character Controller
- Physics Engine
- Camera Controller
- Action Handlers
- 3D Audio Engine
- Game Rules
- Rendering Engine

- joints and inverse Kinematics
- Lighting Calculation
- Project to 2D
Architecture of Game Engines

Database/Resources:
- Weapons/Objects
- Game State
- Inventory
  - For Controlling bots (AI)
  - For specifying keyboard directions for user control of player
  - For Controlling Camera
  - In UT3K:
    - Controller
    - PlayerController
    - Camera
    - xPlayer
    - AIController
    - Scripted Controller
    - Bot
    - xBot
- Scene Graph
  - Navigation
- Character Controller
- 3D Audio Engine
- Action Handlers
- Camera Controller
- Rendering Engine
Architecture of Game Engines

Database/Resources:
- Game State
- Weapons/Objects
- Inventory
- Scene Graph

Processes and Methods:
- Character Controller
- Physics Engine
- Game Rules
- Camera Controller
- Action Handlers

For in game physics
- In UT3K:
  - Karma Physics
  - Bone rotation
  - Collisions

For controlling effects
- 3D Audio Engine
- Rendering Engine
- Game Rules
- Action Handlers
Architecture of Game Engines

Database/Resources:
- Game State
- Inventory

Processes and Methods:
- Weapons/Objects
- Physics Engine
- Game Rules
- Scene Graph
- Camera Controller
- Action Handlers
- Rendering Engine

Game Rules
- In UT3K:
  - Triggers
  - WeaponFire
  - xEmitter
  - Pickup

For controlling triggered actions
For handling actions, e.g.
Pickup, firing, etc.
Assignment #1

Building a simple 2D Game