Principles of Computer Game Design and Implementation

Lecture 4

We already knew

- Introduction to this module
- History of video
- High-level information of a game
- Designing information for a game (Overall architecture, Game structure, scripting language)

Game Loop

Bird’s-Eye View of a Game

- 1. Game initialization
- 2. Main game loop
  - Front-end initialisation
  - Front-end loop (gather input, render screen,update front-end state, trigger any state change)
  - Front-end shutdown
  - Level initialisation
  - Level game loop (gather input, run AI, run physics simulations, update game entities, send/receive network messages, update time step, update game state)
  - Level shutdown
- 3. Game shutdown

Games and Time

- Most programs run slower than the underlying computer.
- Games run as quickly as possible.
- This is demanding on the processor and graphics capabilities.

The Importance of Frame Rate

- *Frame rate* is the speed at which the visual display updates.
- A faster frame rate leads to more fluid animation and is more computationally intensive.
- The goal is to have a fast, consistent frame rate.

Games and Space

- Games are often run in different display modes than typical programs.
- Games often use custom user interfaces.
- Games often take full control over the display and input devices

Event-driven Programming

- The program is event-driven
  - Messages = events
- We need a loop to check all incoming events
- The Loop
  - Check all incoming events (messages)
  - Handle the events
  - Check timing and do something in regular
- Incoming Events
  - Interrupts
  - System requests

Event-driven Programming

- Timers (do something in regular timing)
  - The sub-system to handle timing
  - Must be precise to at least 1 ms or less
- Events
  - Input devices
    - Mouse
    - Keyboard
  - Something coming from network
  - System requests
    - Re-draw
    - …

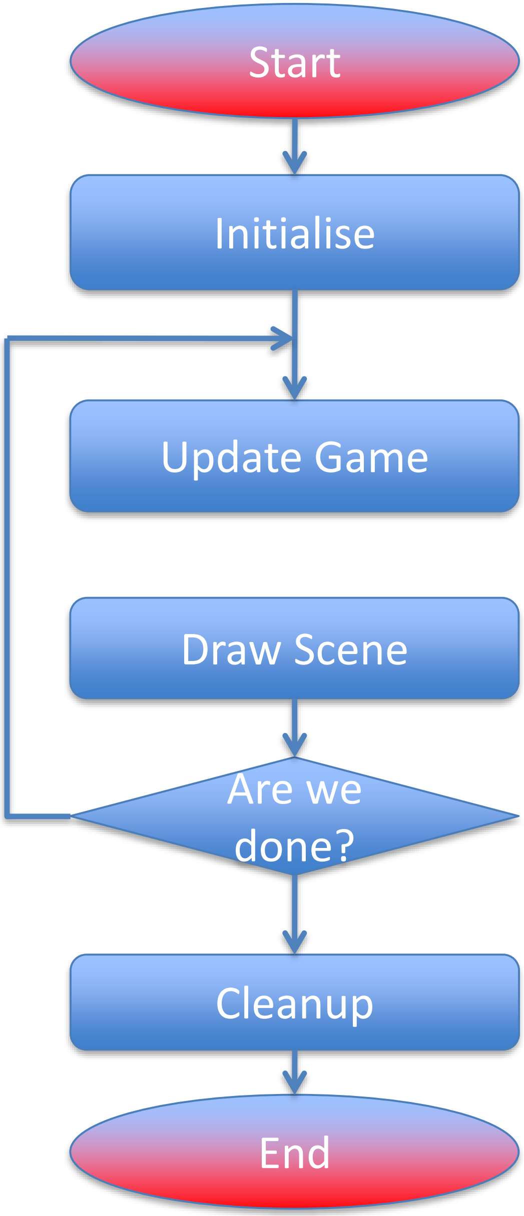
Event-driven Programming

- Therefore, we have two types of jobs:
  - In regular
    - Timers callbacks
  - By requests
    - Input device callbacks
- Same as a game main program
  - A game is an interactive application
  - A game is time-bound
    - Rendering in 30fps or 60fps
    - Motion data in 30fps
    - Game running in 30fps
    - …

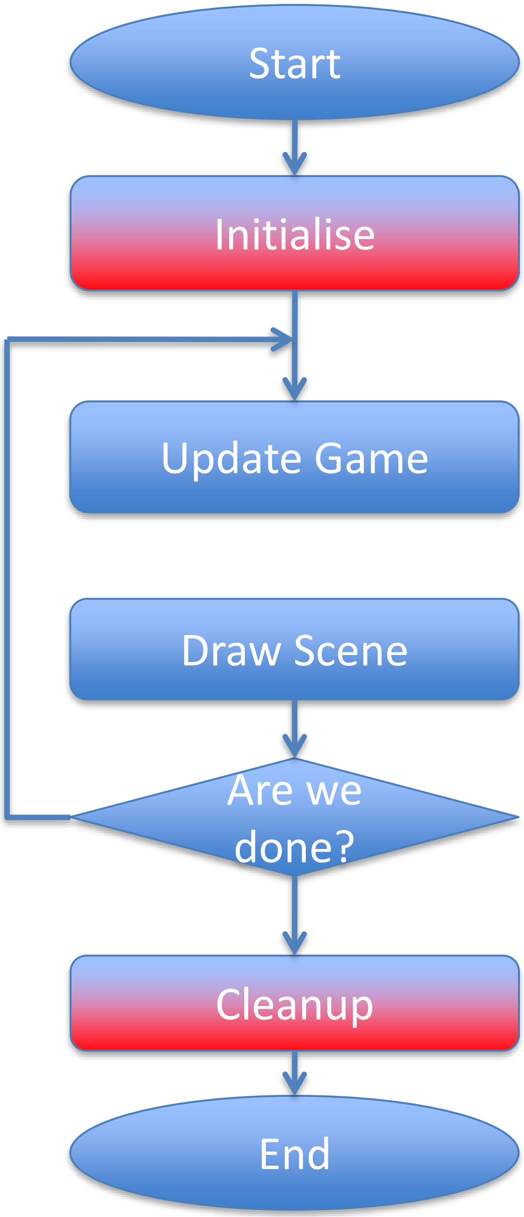
Typical Game Architecture

<u>Initialization/Cleanup</u>

- The initialization step prepares everything that is necessary to start a part of the game
- The cleanup step undoes everything the initialization step did, but in reverse order



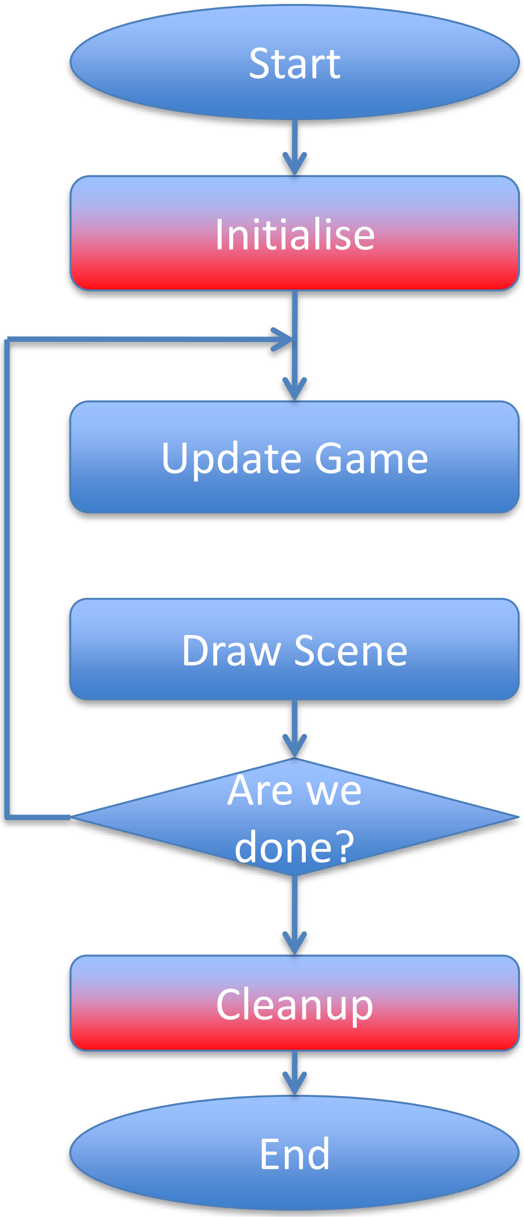
Typical Game Architecture



Initialization/Cleanup

- Resource Acquisition Is Initialization
  - A useful rule to minimize mismatch errors in the initialization and shutdown steps
  - Means that creating an object acquires and initializes all the necessary resources, and destroying it destroys and shuts down all those resources

Typical Game Architecture



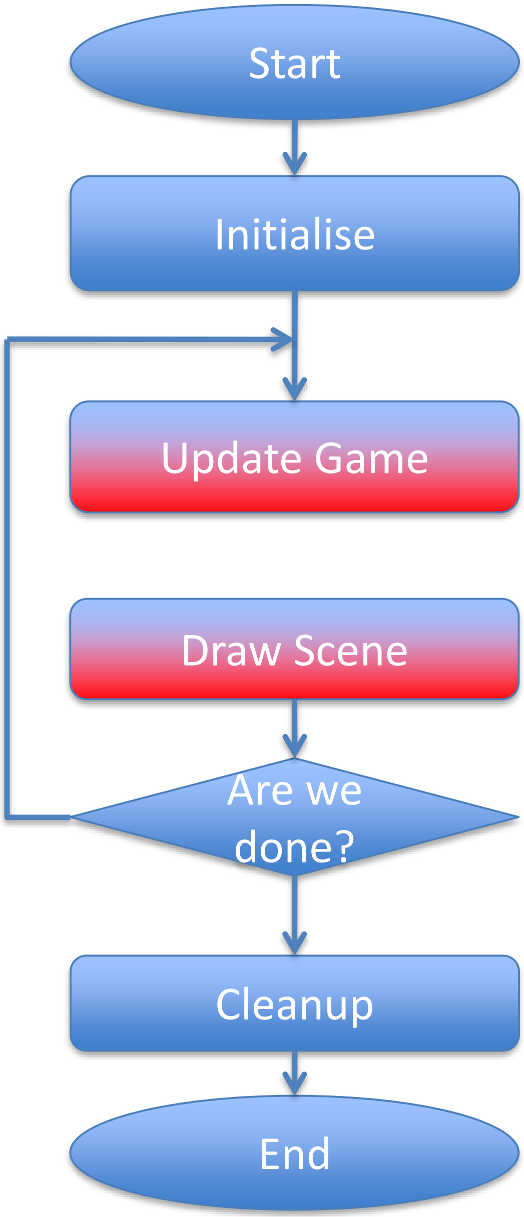
Initialization/Cleanup

- Optimizations
  - Fast shutdown
  - Warm reboot

Typical Game Architecture

<u>Main Loop</u>

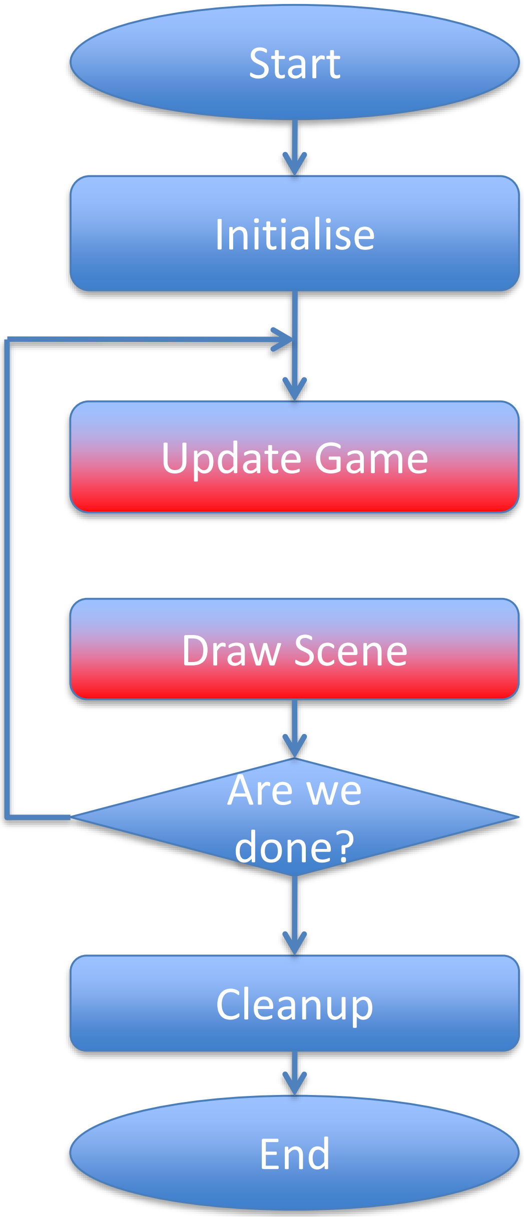
- Games are driven by a game loop that performs a series of tasks every frame
- Some games have separate loops for the front and and the game itself
- Other games have a unified main loop



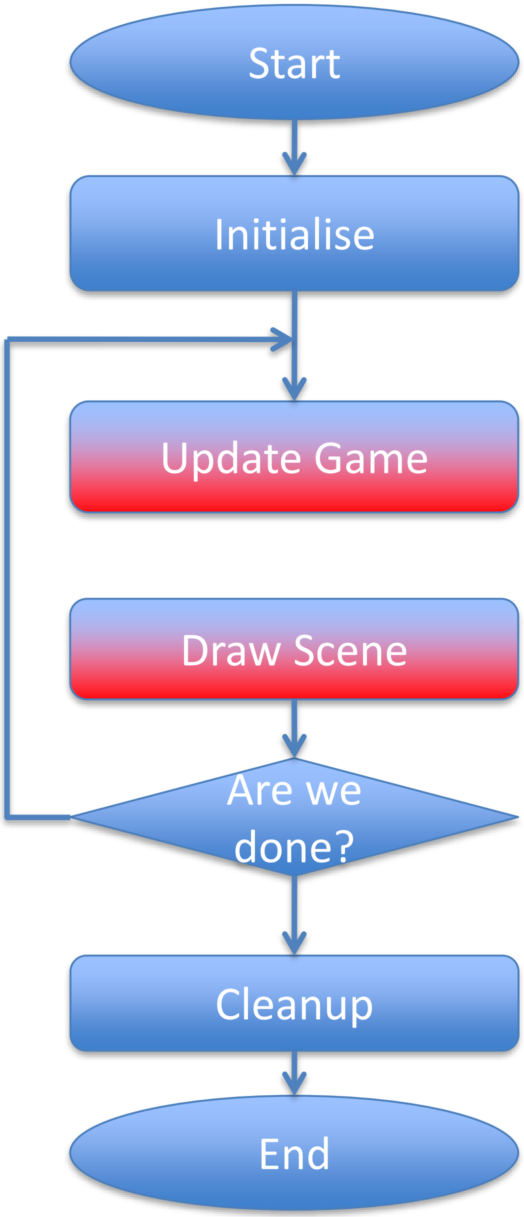
Typical Game Architecture

Main Loop

- Tasks
  - Handling time
  - Gathering player input
  - Networking
  - Simulation
  - Collision detection and response
  - Object updates
  - Rendering
  - Other miscellaneous tasks



Typical Game Architecture



Main Loop

- Structure
  - Hard-coded loops
  - Multiple game loops
    - For each major game state
  - Consider steps as tasks to be iterated through

Execution order

- Most of the time it doesn't matter
- In some situations, execution order is important
- Can help keep player interaction seamless
- Can maximize parallelism
- Exact ordering depends on hardware

Game Entities

- Game loop operates *game entities*
  - Basically anything in a game world that can be interacted with
  - More precisely, a self-contained piece of logical interactive content
  - Only things we will interact with should become game entities

Game Entities

- Organization
  - Simple list
  - Multiple databases
  - Logical tree
  - Spatial database

Game Entities

- Updating
  - Updating each entity once per frame can be too expensive
  - Can use a tree structure to impose a hierarchy for updating
  - Can use a priority queue to decide which entities to update every frame

Game Entities

- Object creation
  - Basic object factories
  - Extensible object factories
  - Using automatic registration
  - Using explicit registration

Game Entities

- Level instantiation
  - Loading a level involves loading both assets and the game state
  - It is necessary to create the game entities and set the correct state for them
  - Using instance data vs. template data

Game Entities

- Identification
  - Strings
  - Pointers
  - Unique IDs or handles

Game Entities

- Communication
  - Simplest method is function calls
  - Many games use a full messaging system
  - Need to be careful about passing and allocating messages