CMPS 327: Introduction to Video Game Design and Development

Dr. Arun K. Kulshreshth

Fall 2020

Lecture 2: Game Architecture

Last Class

- Game
  - Play, goals and rules
- Fun factor in video games
- General design guidelines
- Today: Game Architecture

Video Game

- Video game is a computer simulation of a virtual world
- Game designers must have knowledge of the following to make people, objects, and environments behave realistically in a virtual world:
  - Computer graphics
  - Artificial intelligence
  - Human-computer interactions and simulation
  - Software engineering
  - Computer security
  - Fundamentals of mathematics
  - Laws of physics relating to gravity, sound, etc.

Game Genre

- **Gameplay**: The type of interactions and experiences a player has during the game

- Game genres, based on gameplay, include:
  - Action games
  - Shooter games
  - Racing games
  - Life-simulation games
  - Role-playing games
  - Strategy games
  - Puzzle games

Game Creation

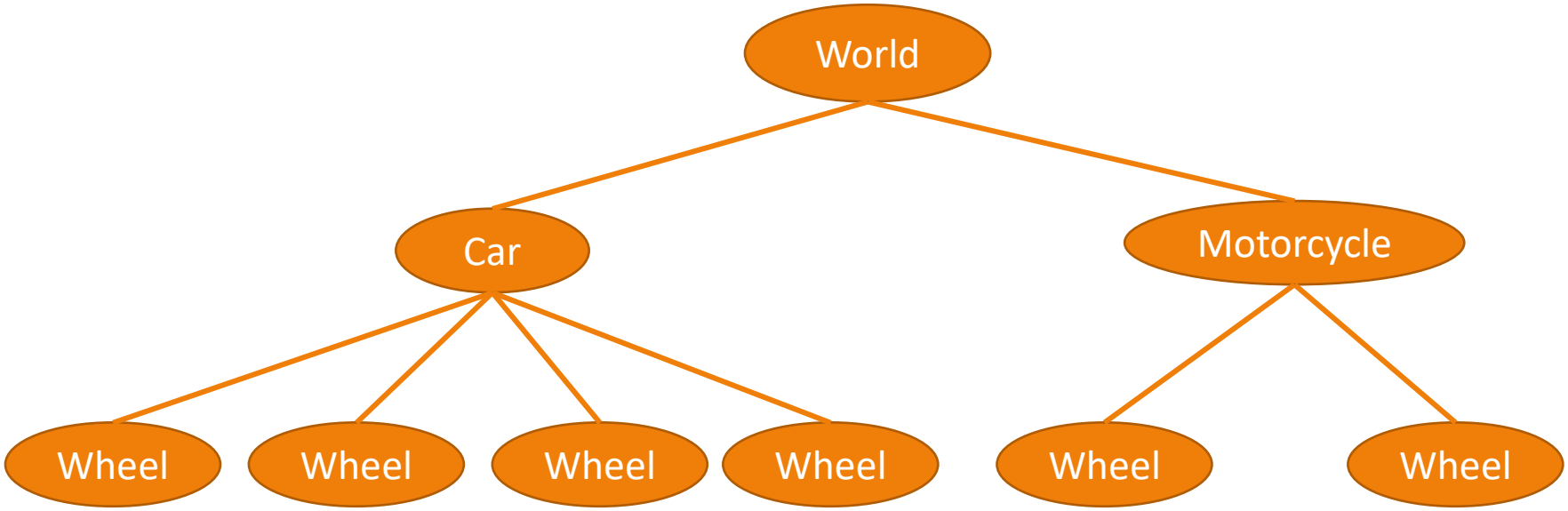
- Game engine
  - a software system within which games can be created
  - E.g. Unity 3D, Unreal engine, CryEngine, etc.

- Main components of a game engine:
  - A rendering engine for graphics
  - A physics engine to provide a collision detection system and dynamics simulations
  - A sound-generating component or sound engine

Game Engine Functionality

- A scripting language apart from the code driving the game
- Animation
- Artificial intelligence algorithms (e.g., path-finding algorithms)
- A scene graph that holds the spatial representation in a graphical sense

Scene Graph Example



Game Programming

- After all the design decisions have been finalized, programmers produce the code to create the virtual world of the game
- Popular languages include: Java, Objective C, C#, and Lua
- Some well-established game engineers have created custom languages based on their games, e.g., Epic Game's UnrealScript for the Unreal Game Engine.

Game Architecture

- The code for modern games is highly complex
- With code bases exceeding a million lines of code, a well-defined architecture is essential
- A video game has three important steps
  - Initialization/startup/loading
  - Game play
  - Shutdown/unloading

Initialization Step

- Prepares everything that is necessary to start a part of the game
- This step deals with resource and device acquisition, memory allocation, initialization of the game's GUI, and loading of art assets such as an intro video from file.

Shutdown Step

- Undoes everything the initialization step did, but in reverse order
- Unloads game data from memory, free resources

- Minimalize mismatch errors in the initialization and shutdown steps
- Creating an object acquires and initializes all the necessary resources, and destroying it destroys and shuts down all those resources

Gameplay

- This is where all the fun is!!

- Game is also a program written in a language
  - Java programs in 261
    - Take input - > process it -> generate output
  - Why the game code keeps running?
    - It is a loop with certain exit conditions
      - Player dead
      - Exit button pressed
      - …

Main Game Loop

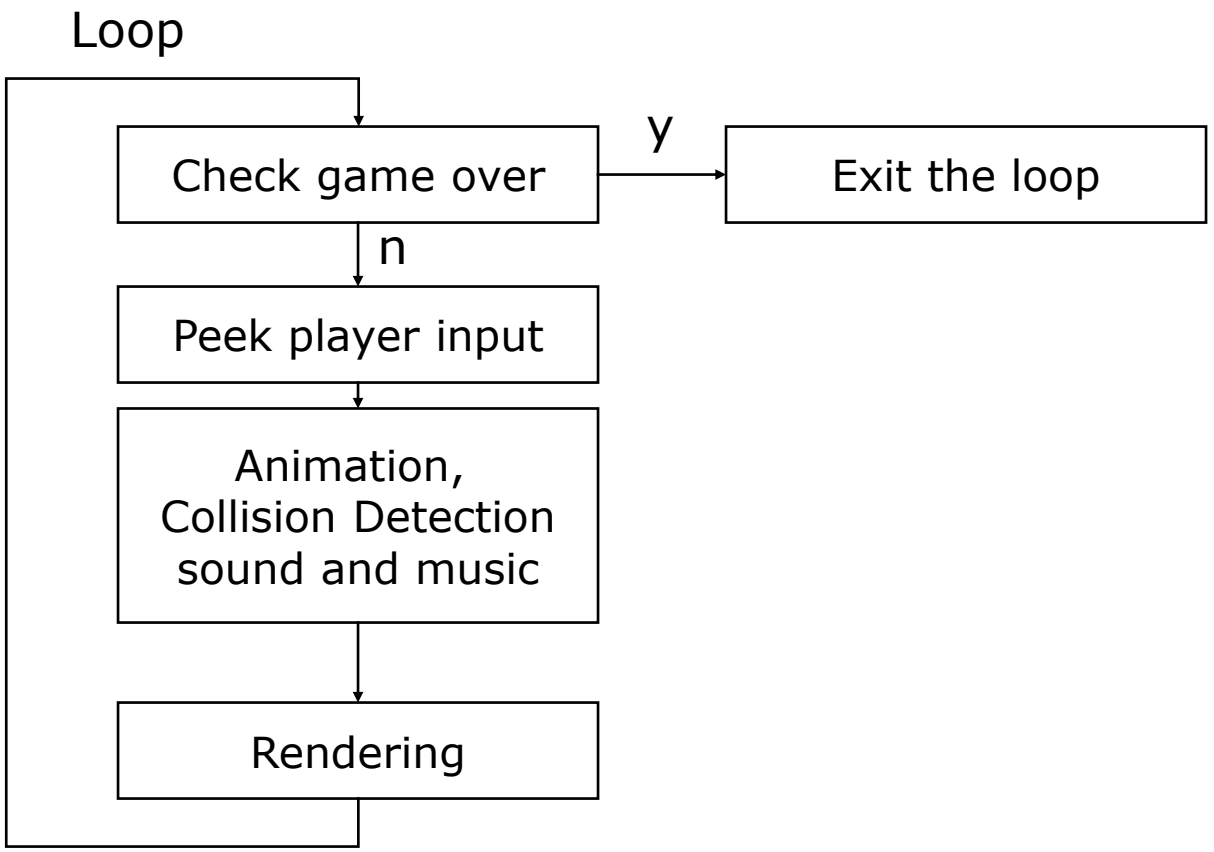
- Games are driven by a game loop that performs a series of tasks every frame
- Some games have separate loops for the frontend and the game itself
- Other games have a unified main loop

Main Game Loop

- Tasks
  - Handling time
    - Timers
    - Must be precise to at least 1 ms or less
    - 30fps = 1/30 second = 33.333… ms
  - Gathering player input
  - Networking
  - Simulation
  - Collision detection and response
  - Object updates
  - Rendering
  - Other miscellaneous tasks

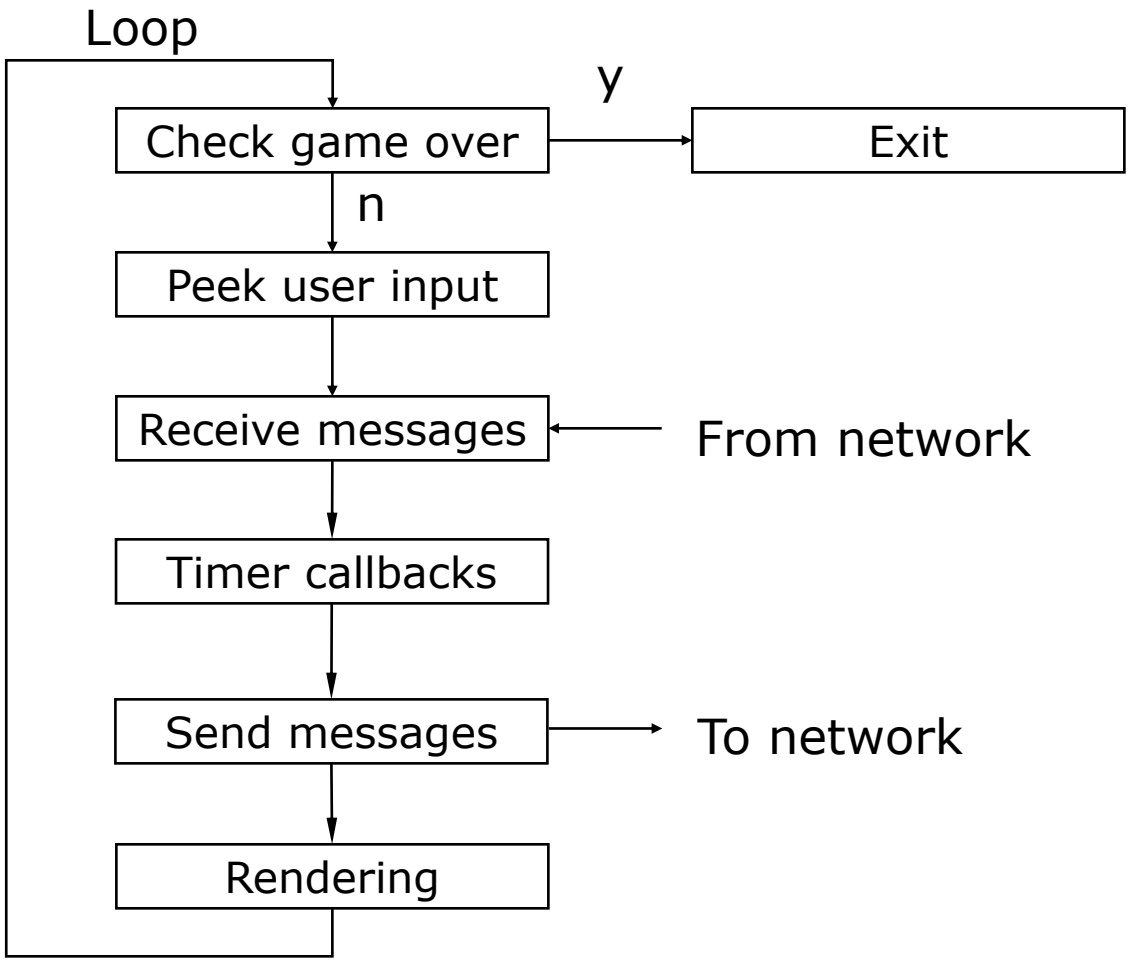
Main Game Loop

- Single Player



Main Game Loop

- Network Client



Main Game Loop

- Time Step:
  - Use a clock internal to the game to drive the game independently from the system clock.
  - Time step in game loop updates the game clock to match the hardware clock, for example at the beginning of a frame.
    - Timing allows a game to execute at a speed independent of the processor's clock speed.
  - Variable frame duration and fixed frame duration.
  - In variable: frames can last different lengths of time depending on what they have to display (from 10 ms to 50ms or more).

Main Game Loop

- Input:
  - Gather the user input and react accordingly.
  - Input devices: keyboard, mouse, gamepad, driving wheel, video camera, …
  - Input should be instantaneous.
  - Minimize the time elapsed between the moment when the input is received and the response from the game.
  - Get the input at the beginning of the game loop, right before the simulation.
  - Networking may come into play if input received from network.

Main Game Loop

- Simulation:
  - Task that brings the virtual world alive:
    - Update game state
    - Run AI behavior
    - Trigger new events
    - Run physics simulations that move objects in the game world
    - Updates particle systems in waterfalls, fires, …
    - Moves animation forward for visible characters
    - Update player's position and camera …
  - The most expensive task in the game loop.

Main Game Loop

- Collision
  - Collision detection: when objects are moved where they are supposed to be, check for collisions between entities.
  - Collision response: correctly update the entities that have collided.

Main Game Loop

- Objects update
  - Move the objects to their exact position after simulation and collision.
- Rendering
  - Display the game world on the screen.
- Miscellaneous other tasks (sound, …).

Main Game Loop

- Structure:
  - Update time
  - Get input
  - Get network messages
  - Simulate world
  - Collision step
  - Update objects
  - Render world
  - Miscellaneous tasks

The Loop in Essence

1. Game initialization
2. Main game loop
    1. Front-end initialization
    2. Front-end loop
    3. Front-end shutdown
    4. Level initialization
    5. Level game loop
    6. Level shutdown
3. Game shutdown

Front-end loop
1. Gather input
2. Render screen
3. Update front-end state
4. Trigger any state changes

Level game loop
1. Gather input
2. Run AI
3. Run physics simulation
4. Update game entities
5. Send/receive network messages
6. Update time step
7. Update game state

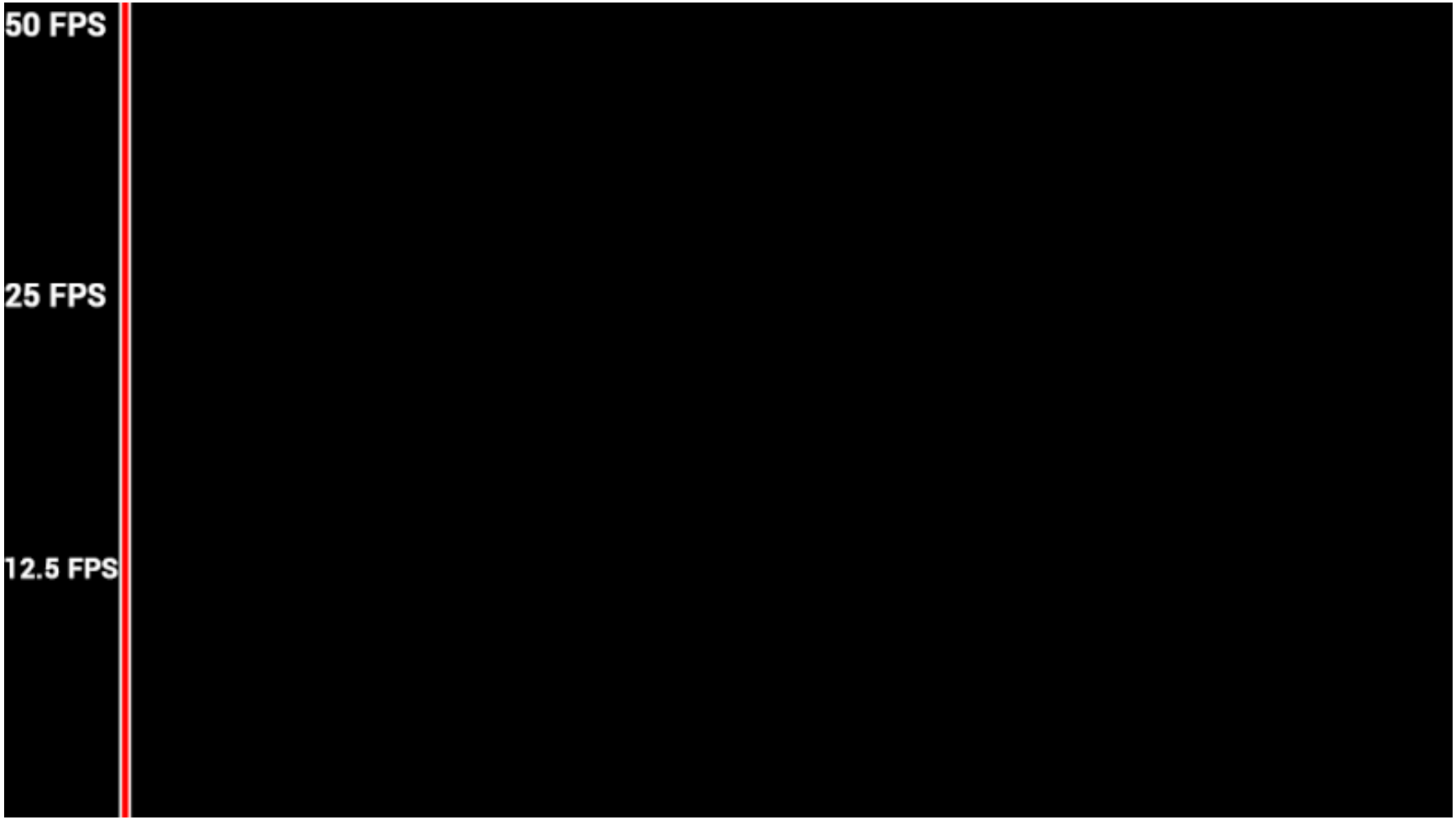
Frame Rate

- Expressed in frames per second or fps
- In the game loop each task takes time
- Frames/second
  - How many times the loop can be executed in a second?
  - E.g. 30 fps, 60 fps,
- Display also has a certain refresh rate
  - Frame rate is independent of this

Frame Rate

- The human eye is capable of differentiating between 10 and 12 still images per second before it starts just seeing it as motion.
- That is, at an FPS of 12 or less, your brain can tell that its just a bunch of still images in rapid succession, not a seamless animation.
- Once the frame rate gets up to around 18 to 26 FPS, your brain is fooled into thinking that these individual images are actually a moving scene.

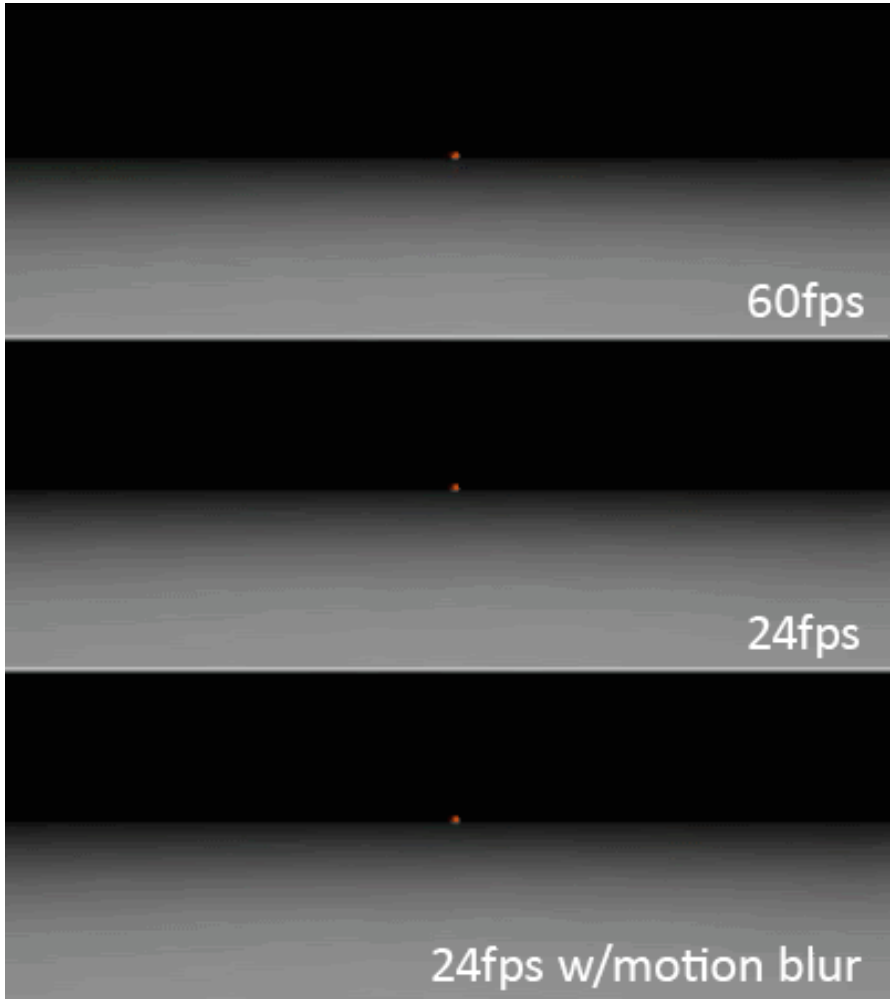
Frame Rate Comparison



Frame Rate

- Slow frame rate
  - motion looks jagged
- Too fast
  - Soap-opera effect
  - Jerky motion
- Soap-opera effect
  - motion smoothing/blurring
  - a feature of many modern televisions
  - E.g. sports content on televisions

Effect of Frame Rate and Motion Blur



Game Entity

- What are 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 Entity

- 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 Entity

- 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
- Identification
  - Strings/names
  - Unique IDs or handles

Communication Between Entities

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

Summary

- Game engine
- Game architecture
  - Initialization
  - Main Game loop
  - Shutdown
- Frame rate
- Game entity

- Next class: Unity 3D Introduction