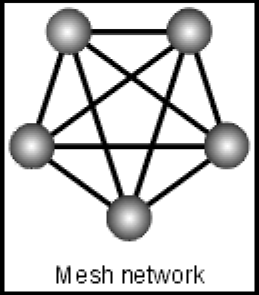
What every programmer needs to know

# GAME NETWORKING

# Peer to Peer Lockstep

- No client, no server
- Fully connected mesh topology
  - Easiest
  - First developed for RTS
- Turns and commands
- Common initial state (starting in a game lobby)
  - Beginning of each TURN
  - Send all commands to all machines
  - All machines run commands
  - End Turn



# Peer to Peer lockstep

- Simple, elegant. But…
- Non-linearity
  - Ensuring complete determinism is hard. Slight differences amplify with time
- Latency
  - All commands must be received before simulating that turn. Latency = max latency over all players!
- Command and Conquer, Age of Empire, Starcraft
  - Best over LANs

# Client/Server

- Lockstep not good for action games like DOOM over internet.
- Each player is now a client and they all communicate with a server.
- Server ran the game simulation, dumb clients interpolated between states received from the server
- All input goes from clients to server
  - Keypresses, mouse movement, presses
  - Server simulates, changes entity states
  - Client gets new entity states, interpolates between old and new states
- Players could come and go in the middle of the game. Quality of connection depends on client server connection

# Client server problems

- Latency is still the big problem

# Client-side prediction

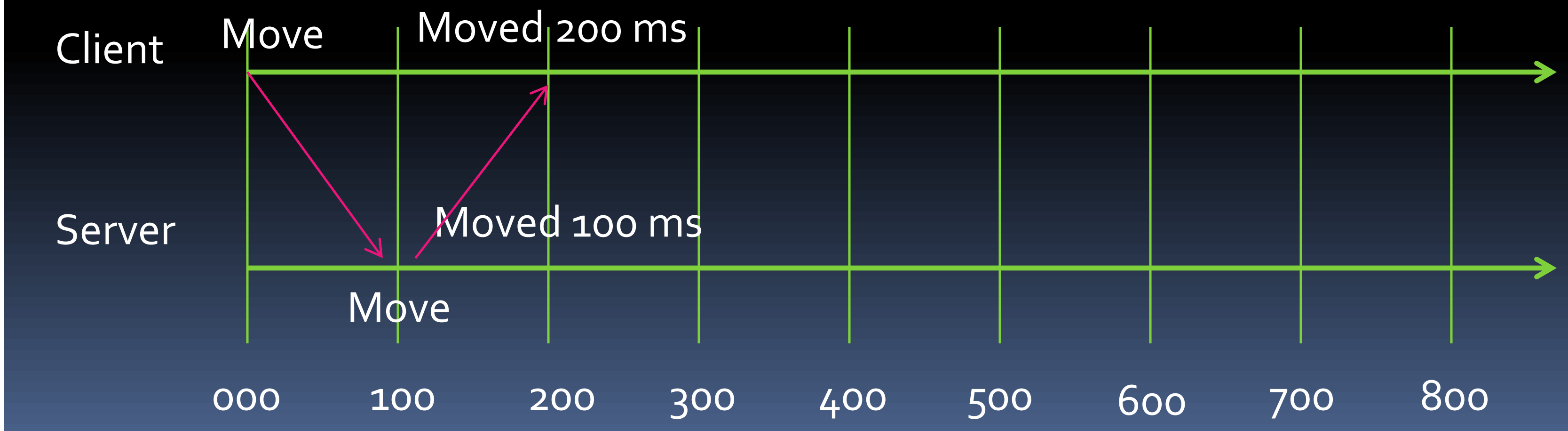
- Client-side prediction
- Latency compensation
- Interpolation
- John Carmack on QuakeWorld
  - I am now allowing the client to guess at the results of the users movement until the authoritative response from the server comes through. This is a biiiig architectural change. The client now needs to know about solidity of objects, friction, gravity, etc. I am sad to see the elegant client-as-terminal setup go away, but I am practical above idealistic.

# Client-side prediction

- All machines are the same and run the same code → no dumb clients
- One machine is still the authority → server
- So
  - Client simulates the movement of your entity locally and *immediately* in response to your input
    - No latency issue. Immediate movement
    - How do I synchronize with all the other players?
      - Communicate with server and correct your movement in response to server state messages

# Client-side prediction

- But server state is past-state
  - If it takes 200 ms for round trip message between client and server, then server correction for the player character position will be 200 ms in the past, .relative to now



# Client-side prediction soln

- Keep a buffer of past local state (and input) for each entity
- When client gets correction from server
  - Discard state older than server state
  - Simulate from server state to now
    - This is your (client entity) new predicted position using latest info from server
  - Look at the difference in position
    - Between the predicted position and your current position → pDiff
  - Note network latency
  - Keep simulating forward from current position, but interpolate to eliminate pDiff within latency amount of time
  - By the time you next get a correction from server, your predicted position should be your current position!

# Test on Monday, April 7

- Physics, AI, Networking
- Game Engine Architecture
- Start finding partners (Wednesday)
  - Prepare a short description of what you want to do
- Larry Dailey will help you brainstorm a game idea next Wednesday, April 9
- Assignment 6, AI, (pairs possible) due April 14
- April project proposal, deliverables, schedule to be posted