Lecture 30

Networking: Consistency
Why Network Games?

Networking: Consistency
Architecture: Client-Server

Networking: Consistency
Architecture: Peer-to-Peer

Networking: Consistency
Basic Networking Concerns

- Networking topology
  - Client-server
  - Peer-to-peer
- Computing model
  - Distributed objects
  - Message passing
- Communication protocol
  - TCP vs. UDP
  - UDP vs. Reliable UDP

**Not Today’s Subject!**

Machine 1

Machine 2

Network: LAN/Internet

CS 5414

CS 4450
Game Networking Issues

Consistency

- Do our games agree?
  - Where do I see objects?
  - Where do you see them?
  - Who is authoritative?
- How to force agreement?
  - Do I wait for everyone?
  - Do I guess and fix errors?

Security

- What cheats are possible?
  - View hidden data
  - Enter invalid states
  - Improve player skill
- How do we cheat proof?
  - Technical solutions?
  - Community policing?


**Consistency**

- *Latency* is root of all evil
  - *Local* actions are instant
  - *Network* actions are slow

- **Example**: targeting
  - Want “geometric fidelity”
  - Fire a weapon along ray
  - Hits first object on ray
  - But movement is fast!

How to tell these cases apart?
World State vs. Local State

- **State**: all objects in game
  - **Local State**: on a machine
  - **World State**: “true” state

- **Where** is the world state?
  - Not on any one machine
  - Union of local states?

- States may be **inconsistent**
  - Local disagrees with world
  - Is this really a problem?
  - What can we do about it?
Server Side Consistency

- Server has the world state
  - Actions update world state
  - Server updates local states

- **Issue**: ordering actions
  - Order by receipt?
    - Lower latencies benefit
  - Time action is sent?
    - Need synchronized clock
    - Major open problem in CS

- When is okay to draw?
Peer-to-Peer Consistency

- Similar issues as server
  - When to render a move?
    - Do I wait for opponents?
    - How long do I wait?
  - What to render before that?
- No authoritative world state
  - Relatively easy to cheat
    - Word state = union of local
    - Alter your local state!
  - Only for small/short games
Pessimistic Consistency

- Everyone sees same world
  - Ensure local = world state
  - *Synchronization* algorithms

- *Age of Empires*/Mythology
  - Each player simulates world
  - Players send only moves

- Uses *bucket synchronization*
  - Monitors round trip time
  - Dropped packets resent
  - Adjusts future bucket sizes

Networking: Consistency
Lock-Step Synchronization

- **Algorithm**: play by “turns”
  - Players send turn actions
  - Even if no action was taken
  - Wait for response to render

- **Problems**
  - Long Internet latency
  - Variable latencies (jitter)
  - Speed set by slowest player
  - What if moves are lost?

- More common in LAN days
Bucket Synchronization

- **Algorithm**: turns w/ timeout
  - Often timeout after 200 ms
  - But can be adapted to RTT (round trip time)
  - All moves are buffered
  - Executed at end of *next* turn

- **Problems**
  - Variable latencies (> a turn)
  - Speed set by slowest player
  - What if moves are lost?

Diagram:
- Player 1
- Player 2
- Synch & Render
- Multiple Moves Possible
- Loss
Optimistic Consistency

- Best guess + roll back
  - Do not wait for others
  - Fix mistakes if needed
  - Only send major events

- **Half-Life**: client-server
  - Client acts immediately
    - Sends moves to the server
    - Guesses other’s moves
  - Server computes *true world*
    - Sends its version to clients
  - Client reconciles *differences*
Dead Reckoning

- Need some way to handle slow players
  - Want to “predict” what their actions will be
  - Use predictions if action lost or late

- Dead reckoning: movement extrapolation
  - Players send velocity, acceleration with each move
  - Assume this is unchanged if no action received

[Diagram showing network flow with nodes labeled 'Lost', 'Extrapolate']
Roll-Back

• Dead-reckoning introduces inconsistency
  • You see your intended move
  • Other players see your extrapolated moves

• Make one computer *authoritative*
  • Server in client-server architecture
  • More complicated for peer-to-peer

• **Roll-back**: Replace guess with authority
  • Players may experience “warping”
  • Can improve with smoothing algorithms
Shooting Around Corners
Consistency: Correctness

- Make multiplayer from start
  - Adding later is very difficult
- Single player is special case
- AI/network interchangeable
- Eliminate **non-determinism**
  - Random rolls are a problem
  - Generate from the server
  - Or use same random seed
- **Input** must be synchronized
  - Or else will get state “drift”

Networking: Consistency
Consistency: Smoothness

- For smoother playback
  - Decouple turn & frame rate
  - Always draw local moves
  - Design for latency and loss

- Latency masking
  - Players wait for elevator
  - Teleportation takes time
  - Many hits needed per kill
  - Bullets have flying time
  - Inertia limits movement

---

![Network Options](image_url)
Consistency: Design Solutions

- Limit possible conflicts
  - Authoritative for own state
  - Minimize amount to guess
  - Make roll-back rare/simpler

- **Game design** solutions
  - Software solutions are hard
  - So make game state *simpler*

- **Examples**
  - Area of interest management
  - Coarse state fidelity
Area of Interest Management

- Aura and Nimbus
  - **Aura**: visibility radius
  - **Nimbus**: detect radius
- Given by “technology”
  - **Aura**: cloaked ship
  - **Nimbus**: sensor
- Consistency check if
  - $B$ is within $A$'s nimbus
  - $A$ is within $B$'s aura
Course State Fidelity

- State need not be exact
  - Often just need an estimate
  - Send estimate over network
  - Handle details locally

- **Example**: tiled games
  - Just need grid location
  - Do not send movement
  - Animate motion locally

- Animation vs. gameplay
  - Many frames = one action
  - Keep interactions simple
World of Warcraft

- Coarse spatial fidelity
  - Not sure of exact position
  - Exact targeting impossible

- How to deal with this?
  - Open, airy buildings
    - Few corners to hide with
  - Attacks are automatic
    - Misses are a random roll
  - If you see it, you can hit it

- Is this a challenge?
World of Warcraft

- Make challenge strategic!
  - NPCs have well-defined AI
  - Affects order/type of attacks
  - Players must learn exploits
  - “Chess-like combat”

- Aggro Management
  - Picks who NPCs attack
  - Draw aggro to shield others
  - Replaces *spatial cover*

- Allows 1 second latencies!
Summary

- Consistency is primary issue in network games
  - **Inconsistency**: local state deviates from world state
  - Caused by latency; takes time for action to be sent
  - Requires complex solutions since must draw now!

- Two primary consistency techniques in use
  - **Pessimistic**: All clients must agree with world state
  - **Optimistic**: Can deviate from world state; roll back

- Next time: **Security**