CMSC 498M: Chapter 10b
Networking and MMOGs

Reading:
- Lecture notes by Mike Zyda, at USC.
- Lecture notes by Sugih Jamin at U. of Michigan.

Overview:
- Multiplayer Game Overview
- Online Game Architectures
- Distributed Virtual Worlds
Games and Networking

Industry Information:
- The top 17 game companies brought in $33B in revenue in 2005.
- Major Players: Combined share $23B.
  - Nintendo
  - EA
  - Sony (games part)
  - Microsoft (games part)

Online Games: About 50% of this is from online games:
- Massively Multiplayer Online Games (MMOGs).
- Smaller online play systems: Squad-on-squad play (under 32 players), or player-on-player games.
- Massively Multiplayer Online Role-Playing Games (MMORPGs) about $4.3B.

Chapter 10, Slide 4
MMOG Subscription Growth

Source: http://www.mmogchart.com
Chapter 10, Slide 5

MMOG Market Share

World of Warcraft
Lineage II
Lineage

Source: http://www.mmogchart.com

Chapter 10, Slide 6

MMOG Market Share by Genre

Fantasy RPG
⇒ MMOG ≈ MMORPG

Source: http://www.mmogchart.com
MMOG Rationale

Why are MMOGs Popular?

Humans are better at strategy than AI engines are:
- Less predictable for general settings.

Social Interaction:
- Peer group support.
- Natural language communication.
- Development of game "culture".

Make money:
- Professional gamers.
- Virtual economies.
- ...and the inevitable rise in "virtual crime".

Origins of MMOGs

Quake (1996):
- First widely used 3D multiplayer online game.
- Difficult to find game servers:
  - Gamers exchanged IP addresses by email or gaming websites.
- No persistent state:
  - Short-lived ad-hoc fight-to-death sessions.

Advent of MMORPGs:
- The Realm Online, Meridian 59, Ultima Online, Underlight and EverQuest in the late 1990s.

Scale:
- 2000: Thousands.
- 2007: Hundreds of thousands.
Persistent MMOGs

Persistence:
- Players can join and leave as they choose.
- Each player can affect the persistent world and be affected by it.
- Plot/events progress even while a player is offline.
- Concept started in MMOGs and later began to be used even in offline games (use the game clock).
- Requires commercial 24/7 game servers.
- Led to virtual currency, botting, ...

Challenges of MMOGs

Need to support millions of subscribers:
...a few 100K concurrently.

Back-End Networking:
- Authentication and billing.
- Ranking / black lists.
- Run-time mods/patches.
- Guard against denial of service attacks.

Front-End (in-game) Networking:
- Network topology (client-server, peer-peer).
- Persistence.
- Latency, bandwidth.
- Virtual economy (audits, gold farming, ...).
- Distributed protocols.
Overview

• Multiplayer Game Overview
• **Online Game Architectures**
• Distributed Virtual Worlds

---

Peer-to-Peer

**Peer-to-peer Architecture:**
- Each player communicates directly with all other players.

**Possible complexity:**
- $O(n^2)$.
- Limited scalability.

**Advantages:**
- Low latency, robust.

**Disadvantages:**
- Demands high bandwidth, limited scalability.
- Persistence vs. redundancy tradeoff.

Figure source: Mike Zyda
Client-Server

Client-Server Architecture:
- Each player (client) communicates with server.

Advantages:
- Scalable, usually requires less bandwidth
- Easier to provide persistence.

Disadvantages:
- Server bottleneck: Higher latency.
- Server failure: Lower reliability.

Client Types:
- Thin Client:
  - All simulation on server.
  - Good for resource-poor clients: Cell phones, PDAs.
- Simulation Client:
  - Most simulation and world distributed amongst clients.
  - Server maintains/updates state.

Multiple Servers

Multiple Server Architecture:
- Many distributed servers, each supporting a subset of the clients.

Advantages:
- Reduced latency.
- Scalability to millions of players.
- Improved robustness.

Disadvantages:
- Difficult to maintain consistency for persistence.

Possible Architectures:
- Shards
- Mirrors
- Grids
Multi-Server Shards

Multi-Server Shards:
- Each server simulates a different instantiation of the world.
- Each player plays within his own realm, and does not influence the other worlds.
- Minimal server-server traffic.

Origin:
- “…different images of the world, trapped in the shattered shards of a mystic gem…” Ultima Online story.

Examples:
- Ultima Online
- Neverwinter Nights
- Silkroad Online
- World of Warcraft (called Realms)

Multi-Server Mirrors

Multi-Server Mirrors:
- Each server mirrors/replicates the same persistent world.

Advantages:
- Common world view regardless of hosted server.

Disadvantages:
- Heavy server-server traffic required for synchronization.

Examples:
- Mankind
- PlanetSide
- A Tale in the Desert
- Entropia Universe
Multi-Server Grid

Multi-Server Grid:
- Each server (Sim) hosts a different region of the same persistent world.
- Sims communicate with each other on a grid structure.
- Users are transferred between servers as they move around.

Advantages:
- Scalability for very large worlds.

Disadvantages:
- Load balancing for high density regions.

Examples:
- Second Life

---

Overview

- Multiplayer Game Overview
- Online Game Architectures
- Distributed Virtual Worlds
Distributed Virtual Worlds

- Implementation is distributed, but appearance should be local.
- Need to hide artifacts that suggest otherwise.
- Each player sees real-time activity and reacts.
- Actions should have immediate impact on the virtual world.

**Major Challenge:**
- Latency: Delays induced by network structure.

![Diagram showing latency impact on a game scenario](image)

Distributed Virtual Worlds: Latency

**Latency:**
- All information received is out-of-date.
- Delays are variable and unpredictable.
- Maintaining the illusion of a consistent and persistent world is much harder when players interact.

**Collisions:**
- **Simple:** Bullets, missiles (linear or parabolic).
- **Complex:** Sword fighting, handshakes.
- **Nightmare:** Successive/concurrent collisions/responses.
Distributed Virtual Worlds: Latency

**Visual latency:**
- 30Hz or higher (~ 30 ms)

**Response times:** Depend on interaction speed.
- **Real-Time Strategy:** < 250 ms preferred, 250-500 ms playable, > 500 ms noticeable.
- **First-Person Shooter:** < 150 ms preferred.
- **Car race games:** < 100 ms preferred; 100 – 200 ms sluggish; > 500 ms out of control.

**Predictability:**
- Predictable (even if sluggish) response considered better than variable (even if sometimes very fast).

**Haptic (touch) latency:**
- **Preferred:** Update rates > 1 KHz (< 1 ms) in force-feedback systems.

Distributed Virtual Worlds: Challenges

**Collision detection and response:**
- Did the collision occur?
- If it did, when?
- Velocities, forces, in play at that instant.

**Environmental effects in a changing world:**
- Global illumination.
- Aural (sound) rendering.
- Both can be computationally intensive.
Distributed Virtual Worlds: Challenges

Managing Shared States: Three methods...

Shared repositories:
- All servers maintain a common consistent description of the world.
- Absolute consistency!

Blind Broadcasts:
- Owner of each state transmits its current state at regular intervals.
- Clients cache the most recent update information.
- No acknowledgements.
- Frequent state updates compensate for lost packets (hopefully).

Dead Reckoning:
- Transmit state updates occasionally.
- Extrapolate from past updates to estimate the true shared state.
- Need to correct state when an update is received.

Dead Reckoning

**Classical:**
- Navigation with just logs, compass, clock (local/dead information) without looking at the sky (global/live information).

**Distributed Worlds:**
- Each processor predicts the state of the other prior info.
- Corrects course when live information arrives.
- Predict $\rightarrow$ Correct $\rightarrow$ Converge.

---

Chapter 10, Slide 23
Copyright © David Mount and Amitabh Varshney

Chapter 10, Slide 24
Copyright © David Mount and Amitabh Varshney
Distributed Virtual Worlds: Challenges

Mutually Conflicting Goals:
- We can have either a dynamic world or a consistent world, but not both.

Design Implications:
- For a highly dynamic shared state, hosts must transmit more frequent data updates.
- To guarantee consistent views of the shared state, hosts must employ reliable data delivery.
- Available network bandwidth must be split between these two constraints.

Summary

Summary:
- Multiplayer Game Overview
- Online Game Architectures
- Distributed Virtual Worlds