Massively Multiplayer Online Gaming (MMOG)

- Overview
- Technical issues
  - Scalability
  - Coordination
  - Security
- Economic issues
  - What makes gaming work?
  - What are the limits (if any) to the size of a game?
- Some of the information taken from a USENIX 2005 talk by Mark Wirt (CTO, Butterfly.net)
What is a MMOG?

- Massively Multiplayer Online Game
  - These days, often Massively Multiplayer Online Role-playing Game (MMORPG)
- Thousands of players simultaneously connecting
- Sharing actions and state
  - Multiple players see the same world
- Persistent worlds
  - Persist across logins
- MMOGs are not limited to...
  - Specific game genres (not just role-playing)
  - Deep/long term commitment
  - 3D immersive environments

MMOG history

- Multi-User Dungeons (MUDs): 1978
  - Multiple users interact via text messages
- SIMNET: 1985
  - Defense Dept. project to network tank (and other vehicle) simulators
  - Users shared the same world
  - Initially had to be colocated (no longer the case)
  - I worked on this one in 1987–1988
- Meridian 59 (3DO): 1996
  - Multiple users can interact in real-time graphically
- Ultima Online (Origin Systems): 1997
  - The (commercial) MMOG is born
SIMNET

- Hundreds of “tanks” and other vehicles connected by networks
  - Each vehicle displays the outside world through “windows”
- Vehicles need to know about others in the immediate vicinity
  - No need to know about something 100 km away
- Physics has to be accurate
  - Makes it easier (!) for game to simulate reality
- Support for AI-based opponents
  - Single person can control a whole tank squadron
- Used to train military for missions in unfamiliar areas
  - Soldiers drove the Arabian desert before landing in the First Gulf War

How do MMOGs work?

- Players connect to central server
- Central server manages the game
  - More secure
  - Point of serialization (remember about replication…)
- Players’ computers display the world and allow interaction
  - State has to be updated for each player
  - State has to be consistent for all players (as far as they can tell, anyway)
Types of MMOGs

- "Text-based"
  - Users interact via text and similar messaging
  - Bulletin boards could be a type of MMOG
  - Synchronization less of a concern
- "Role-playing"
  - Players specify "higher-level" actions
  - Less demand for real-time actions
  - No millisecond-level actions (rather strategy)
- "Fast-twitch"
  - Players do many rapid actions
  - Real-time activity a must
  - Can be difficult to coordinate among multiple people

State in a MMOG

- State in an MMOG is
  - Relatively permanent environment
  - Terrain
  - Game-controlled information
    - Non-player characters
    - Dynamic environment conditions
  - User-influenced information
    - Character state
    - Objects that users interact with
- Games persist over months to years
  - State has to last that long
  - All players must see the same state!
What’s hard in an MMOG?

- Thousands of simultaneous players
- Proliferation of objects and assets
  - Need to track all of them exactly
  - Make sure they don’t get duplicated…
- “Griefers”
  - Cheating
  - Denial of Service (DoS) attacks

MMOG architecture

- “Real life”: scalable, reliable distributed systems run in a well-defined environment
- MMOG: scalable, reliable distributed systems
  - Dynamic environment
  - Non-deterministic environment
  - Insecure environment
Wire protocol

- Which protocol should be used?
  - TCP or UDP?
  - Low latency vs. reliability?
- Base it on existing protocols?
- What about security?
  - Use encryption, hashing
  - Anti-spoofing support
  - Support for verification of the client?
- Support heterogeneous platforms

Distributed transactions

- MMOG has to update state in such a way that everyone agrees
- Actions may be grouped
  - Pick up object
  - Use object
- Actions should either happen or not!
- Need to have cryptographically strong encapsulation
  - Don’t want players faking actions
  - Remote authorization is necessary
- Need to have fast updates and commit
  - Can’t keep “tentative” state for too long!
Distributed state

- State contains millions of objects distributed across thousands of clients
  - Updates must be distributed
  - Updates must be synchronized
- Updates have to be low latency!
- Limit the amount of state each player must have
  - Calculate visibility / occlusion
  - Calculate distance and possible “visibility time”: forget about objects you can’t affect soon
- Present view to each player of the current state
  - Transform into graphically-usable state

Distributed state consistency

- State on a client must be internally consistent
  - Can’t violate laws of (game) physics
  - Can’t violate game play rules
- State on a client must agree (somewhat) with that on other clients (“good enough”)
  - May differ slightly if there’s transmission delay
  - Needs to be corrected based on new information
    - Correction should be as seamless as possible
Guessing the new state

- May need to display new state before updates arrive
  - Slow network
  - Fast game
- Guess at the new state
  - Project object movement forward (using “physics”)
  - Use client to predict actions of game-controlled entities
  - Make guesses on what other players might do
- Need to correct for bad guesses
  - Immediate correction could look bad
  - Correct slowly over a second or two?
  - Other approaches?

Using physics to predict the next state

- Object is moving at a fixed velocity
  - Predict that it continues to do so
- Object is changing velocity
  - Predict that velocity will continue to change that way
  - Applies to things like direction change as well
- Collisions can also be predicted in advance (often)
  - May be difficult if projectiles are relatively slow–moving
  - Predicting collisions is very important!
- Try to distribute physics information (velocity, acceleration) to clients to let them do the calculation
- Final outcome has to be updated anyway to prevent clients from cheating or guessing wrong
Security in MMOGs

- People cheat!
  - Gain items they shouldn’t have
  - Do things they shouldn’t be able to
  - Run “bots” to play for them?
- Transport layer security
  - Prevent spoofing
  - Prevent packets from being modified
  - Perhaps authenticate the client (make sure it’s your version of the game!)
- Protocol layer
  - Use encryption to protect the data or people may get “secrets”
  - Open, documented protocols can lead to cracking
- Resource level
  - Need to secure the servers: not too hard
  - Need to secure the client against tampering: very difficult!
    - Tampered client can do things that aren’t supposed to be possible
    - May need to verify correctness at server
    - Tampered client can do things that are “difficult” (perfect aim)
- Service level
  - Guard against denial of service
  - Protect before it reaches the game servers
- Software exploits are a problem as well
Guarding against ‘bots

- Problem: users create “bots” to play the game for them
  - Faster reflexes
  - Better aim
- How can ‘bots be detected?
  - Reaction time that’s better than human?
    - Maybe it’s just a good player!
  - Rapid change in skills (bad player gets a ‘bot?)
- Design situations that ‘bots are bad at?
  - Several “targets” that humans can distinguish but ‘bots can’t?
- Remove the “fast-twitch” element from the game?

Preventing resource cheating

- Problem: players want “extra” resources: gold, items, lives…
- Solution: use authentication (public key-based signatures)
  - May be slow
  - May not prevent resource duplication
- Solution: authenticate with serial numbers
  - Flag duplicates as cheating
  - Requires “world”-wide cheat detection and DB
- Solution: keep all state on servers (including changes!)
  - Slower
  - More difficult to compromise: servers maintain consistency
When MMOG security fails

- Ultima Online had several bugs
  - Lord British (alter-ego of game designer) was murdered, which wasn’t supposed to be possible
  - Server bug in Ultima allowed cloning of gold
    - Result: hyper-inflation because of the ease of getting lots of money
- It can be difficult to undo security problems
  - Just like the real world: hard to rectify “bad things”
    - May be easier because you can raise the dead
  - Still difficult to “undo” a chain of events

Scaling MMOGs

- Static clustering probably doesn’t work
  - Load varies with time of day
  - Load may vary by which “region” is active (US vs. Japan vs. mid-Pacific)
- Solution: use ad-hoc clustering
  - May need to migrate sessions while they’re active!
  - Group players together if they’re interacting with each other
  - What happens when they travel long distances?
    - Teleportation may be real in the game!
- Interactions between physical resources can be complex
  - Updates will need to be propagated between servers as well as to clients
  - Resource management is difficult…
Game instancing

- Shared world for “large-scale” interactions
  - Outside world
  - Large-scale economy
- Individual world for more intensive interactions
  - Separate dungeon for each raiding party
- Need to make the transition from shared to individual be seamless
- Can be difficult if another party enters the same dungeon at the same time…

Upgrading the “world”

- People have different versions of the software
  - Bug fixes and upgrades
  - Perhaps even different goals
- Everyone needs to interact correctly
  - Different versions can work together
  - New features have to somewhat work in older clients
- Eventual goal: reuse the underlying platform for multiple games
  - This can be difficult because of the custom nature of games
  - Would be very useful for reducing the cost of developing a game
Testing an MMOG upgrade

- Can’t just roll out an upgrade
  - Could be (serious) bugs that cause inequities
  - Might cause the system to crash!
- Need to test with live system!
  - Can’t predict what players will (and won’t) do!
- What if an anomaly is found?
  - Expected behavior?
  - Allowed (but not expected) behavior?
  - Bug or cheating?

Monitoring an MMOG

- Need to instrument all of the pieces of an MMOG
  - Detect potential failures or problems as they happen
  - Find inefficient pieces of code
- Analyze the traces in real time!
  - Need to fix things as soon as possible (*Dream Park*, Niven & Barnes)
  - Figure out what might be **causing** the problem (may not be slow part of the system!)
- Trace game activity as well as system performance
  - Unusual game activity could be due to cheating or exploiting a bug
  - Similar to “real-world” surveillance, except that it’s (hopefully) impossible to avoid
Game economics

- Game economies are big!
  - Real Life™: economy develops naturally, goal is to predict it
  - Game: economy is simulated, goal is to simulate something that’s reasonable
- Challenges include
  - Availability of supplies and controlling the supply curve
  - Shadow economies
  - NPC's (non-player characters) and other forms of slavery
  - Hoarding
  - Taxes and fees
  - Methods of exchange and transactions
    - Transaction and contract support
    - Auctions and markets

Limits to scaling

- Too small: not very interesting
  - “Desert island” society: not enough interaction
  - Players get bored -> game dies
- Too large: may be overwhelming
  - Too much like real life!
  - Too difficult to master
  - Too difficult to program (correctly)
- Need a happy medium
  - Large enough to be interesting
  - Small enough to get familiar and be able to conquer (win) the game
MMOG summary

- Massively multiplayer online games present lots of challenges
  - Scalability
  - Security
  - Concurrency
- Still a lot of research to be done
  - Just doing it correctly is hard
  - Also need to build “framework” so the tools aren’t reinvented each time
- Can be tricky to get the game right along with the underlying technology!