Real-Time Massively Multiplayer Games

Dr Richard A. Bartle

Visiting Fellow

11th December, 2002

Introduction

- Games
  - Things you play, supposedly for fun
- Multiplayer Games
  - Capable of having at least 2 players at once
  - But actual number may vary (could even be 0)
- Massively Multi-player Games
  - Having > *n* players at once
  - *n* is (currently) 128
    - Whatever it takes to rule out LAN games…
  - These games are computer-moderated
- Real-Time Massively Multiplayer Games
  - Games with a response time of < 4 seconds
    - Whatever it takes to rule out WWW and email games

In practice…

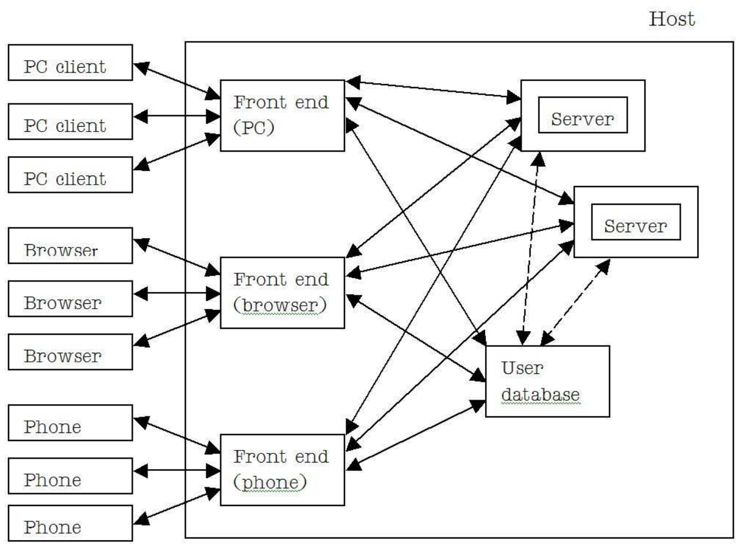
- Role-playing games
  - E.g. *EverQuest*, *Ultima Online*, *Dark Age of Camelot*
  - Hugely popular
    - *EverQuest* has nearly 440,000 subscribers
    - *Lineage* (Korea) has over a million
  - Lots of clones in development
- First-person shooter wannabes
  - E.g. *World War II Online*, *Neocron*
  - Pedestrian by *Counterstrike* standards
- God games
  - E.g. *The Sims Online*. Still in beta test…
  - Not entirely clear what people will do in them

Gratuitous Screenshot

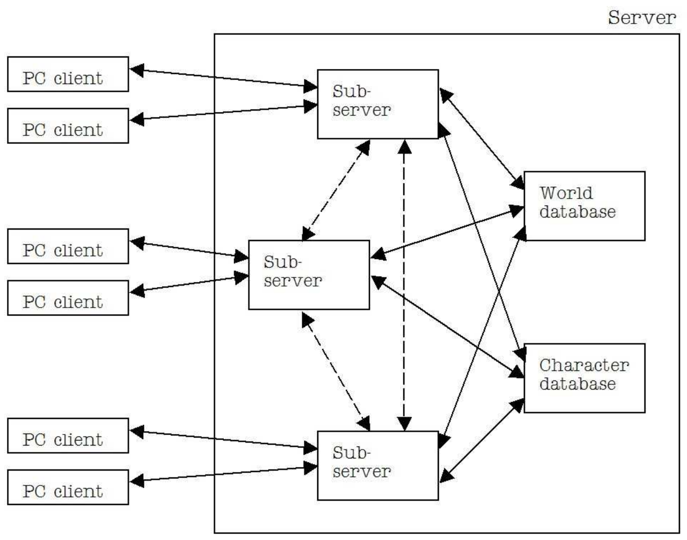
Architecture

- Uses a client/server approach
- All decisions are made by the server
  - But newbie developers still don't fully get it
  - That's *all* decisions, guys
- Clients are only given data the player needs to know
  - In theory…
- Sadly, the client needs to know more
  - E.g. what's in the immediate locale
  - So with a hack, the player finds out anyway
  - Therefore provide it officially for them?
- Still opportunities to cheat
  - Gamma correction for night sight

Overall Configuration



Shard Configuration



Notes

- Some connections vary in ways not shown
  - Which I’m not going to show, either…
- Other (boring) parts of the system omitted
  - Billing
  - Customer service
    - Deals with people, not characters
    - “Log everything”?
  - Statistics gathering
  - Patch management
- Serious AI may require its own box
  - Does it get access to the world database?
  - *Mobiles* rather than *bots*

Why are Shards Clusters?

- Ideally, one computer runs the whole game
  - This works for textual worlds already
- But for 250,000 players?
  - Or for realistic physics?
  - Or for passable AI?
  - Plenty of ways remain to soak up CPU!
- *Could* buy a supercomputer
  - But 8 machines of power *p* cost less than 1 of power 8**p*
  - And 64 machines of power *p*/8 degrade in efficiency
    - So we're told. No-one has actually tried it…
- So shards will be clusters for some time

Load Balancing

- How to ensure work is done equitably?
- Greatest source of load is player activity
  - So assign incoming players to least loaded box?
- For each command, a server needs to:
  - Lock all database records it may need
  - Perform precondition tests
  - Update data for effects
  - Unlock the records
- This means heavy overheads
- Idea! Most commands are movement
  - So partition load by virtual geography
  - Don't need to lock records if you own them

Two Approaches

- Fixed load balancing
  - Each sub-server handles a set area
  - Easy to implement
  - Can partition world database local to sub-servers
  - Allows client to pre-load texture maps
  - Popularised by *EverQuest* and its *zones*
- Dynamic load balancing
  - Each sub-server handles several smaller areas
  - Over-used sub-servers pass control of an area to under-used sub-servers
  - Allows for seamless terrain
  - Has no “physical” borders
  - Popularised by *Asheron’s Call*

Designer Solutions

- Fixed load balancing is best 90% of the time
  - Because of single-machine efficiencies
- However, it has major flash crowd problems
  - 1,000 people in one zone? Oh-oh…
- Design of the game can address this though
  - *Dark Age of Camelot* has 3 realms + borderlands
  - Characters can’t enter other realms
  - Therefore at most 1/3 of players per zone
- Also, use smaller zones, spread by geography
  - No sub-server gets geographically adjacent zones
- *DAoC* players don’t notice zone transitions
- *EQ* players do (for 40 seconds or more)

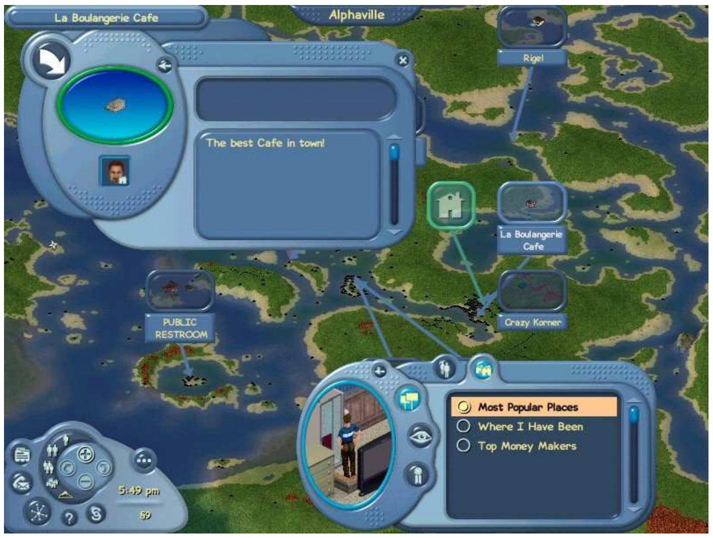
Synchronisation

- Internet is laggy
- Predictive algorithms to handle this
  - Smooth position changes to avoid warping
  - So what you see on your screen may be wrong!
- Implies use of absolute co-ordinate frame
  - Travel to a point, not travel “forward”
- But some actions are relative
  - Shoot an arrow *at* someone
  - You might not be where you think you are
  - They might not be where you think they are
- Most designs target objects not places
  - Arrow flights are not timed
  - So arrows sometimes go through obstacles…

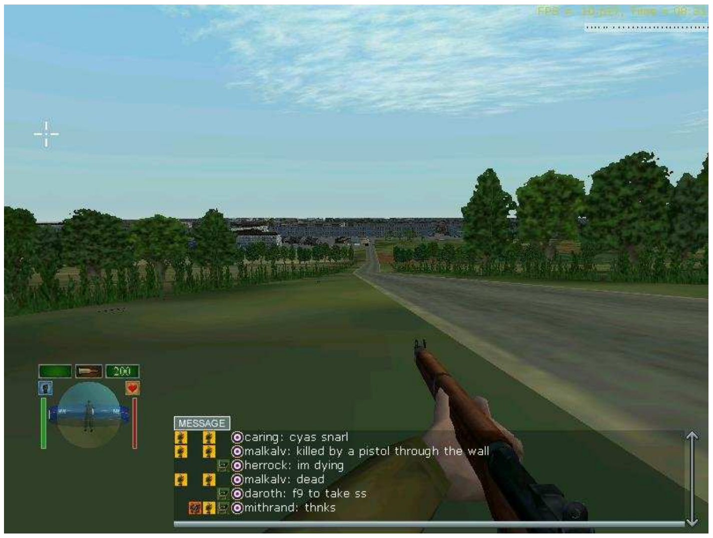
Conclusion

- Many of the problems with games occur in real world situations too
- Tried-and-trusted solutions exist
- But game developers love re-inventing the wheel…
- Thoughtful designers can help alleviate the situation
  - But most designers aren’t thoughtful…
- Great opportunity for non-game experts
- But it’ll take a while for developers to figure this out…
  - Sorry, folks!

Added Extra 1



Added Extra 2



```
caring: cyas snarl
malkalv: killed by a pistol through the wall
herrock: im dying
malkalv: dead
daroth: f9 to take ss
mithrand: thnks
```