Who Are We?

• We work in the engineering group at Vicarious Visions.
  – Brian led programmers on Marvel Ultimate Alliance for PSP, and implemented network play on Transformers for DS.
  – Chris goes to meetings.

• Vicarious Visions is a game development studio in the Activision corporation. Locations in Albany, NY and Mountain View, CA.
What Are We Going To Talk About?

- Architecture issues around networking and games in general.
- Practical issues with networked games on PSP and DS.
- Networking on Marvel Ultimate Alliance PSP and Transformers DS.

And we will do it all without violating more than a handful of NDAs and other confidentiality agreements in the process!
Network Architectures for Games

• Topology
  – Client/server
  – Peer-to-peer
• Memory use and performance
• Bandwidth, lag and packet loss
• Security
Client / Server

- Server receives user input from clients, runs simulation and sends game state to clients
- Clients collect user input, send to server, receive game state and display it to user
Client / Server 2

- Client / Server topology has a profound effect on the architecture of a game
- Server is single point of failure and authoritative
- N clients equals N connections
- Clients can enter and leave session freely
- Clients may extrapolate (or sometimes interpolate) game state in the presence of lag and packet loss
- Security assumes a good server and potentially malevolent clients
- Client / Server may not be practical for games with a large simulation (games that have a lot of objects driven by physics)
Peer-to-Peer

- Peers share user input with each other each tick
- Peers run simulation
- Peers are perfectly synchronized so that the simulation results match on all machines
Peer-to-Peer 2

- Peers **must** be synchronized
- $N^2$ connections
- Entering and leaving a game can be complicated
- Extrapolation to smooth over lag is complicated (generally try input buffering instead)
- Security is complicated (impossible?)
- Can be bootstrapped onto a single player game without fundamental design changes!
Practical Issues with Handheld Networking Games

• Number of players
• Usability and presentation
• Player communication
• Community
Number of Players

• System limitations
  – PSP limited to 17 connections on wireless LAN (ADHOC) play
  – DS limited to 2000 polys per frame
  – Networking libs can use a couple 100kb of precious memory

• More player disconnects expected on handhelds than fixed platforms

• Limited screen size
Usability and Presentation

• Limited screen size can make complex configuration and matchmaking screens hard to navigate

• Handheld play periods tend to be shorter so allow players to get into game quicker
Player Communication

• Text entry is rather difficult
• Voice chat a non-trivial CPU and memory cost
  – DS has built-in microphone
  – PSP has an optional headset that nobody bought
• DS prohibits communication with ‘strangers’
• Definite effects on gameplay
Community

• Extend single player experience
  – Score boards, replays, downloads

• Facilitate multiplayer experience
  – Swapping friend keys on DS
  – Custom data share

• DS has limited storage space for dynamic content. Use website as a big fileserver for custom content.
Use Cases

• Two real-world games with multiplayer
  – Transformers DS
  – Marvel Ultimate Alliance PSP

• Different situations, similar solutions
Transformers DS

- Third person action game for summer 2007 Transformers movie
- Platform doesn’t have much CPU to spare
  - So having a server is unlikely
- No need to support WiFi (Internet) play
  - So we can guarantee low latency
- Goal was to support 4-players
- Peer-to-peer (key-sharing) works great!
Key-Sharing

• Each machine reads one frame of input
• Input is sent to host…
  – … wait, you said this was peer-to-peer!
Key-Sharing

- Host collates data and broadcasts to all clients
- After receiving this data, all machines do one ‘update()’
- Data-sharing only takes 1-2 frames for local wireless
  - System is running at 60 fps, so that’s 16-33 ms. Wow!
Synchronization

• Identical simulation is tricky
• This is where most bugs occur
  – Camera-relative code
  – Randomness, ordering assumptions
  – Uninitialized variable use
  – Memory layout dependencies
  – Different builds?
• Obviously, we also need guaranteed delivery of packets.
Synchronization

• Old process for locating these bugs:
  – Tester plays game
  – Tester notices that games are wildly out of sync
  – Tester reports bug

• Imprecise, lack of useful information
• Chance to miss lots of sync bugs!
SyncChecker To The Rescue!

• Enhancement to our key-sharing system
  – Provides space for extra data
  – Transmitted along with key information
• Uses extra key-sharing buffer space
  – Max per-frame payload is about 512 bytes
  – Minus actual key-sharing data…
  – We’ve set aside 96 bytes per user
SyncChecker

• **Does**
  – Detect errors
  – Reporting of failed state synchronization

• **Doesn’t**
  – Correct errors
  – Identify bugs at the code level
Key-Sharing With SyncChecker

- Each machine reads one frame of input
- Appends checksum of important game state
- All data is sent to host
Key-Sharing With SyncChecker

- Host collates data and broadcasts to all clients
- All machines check that everyone agreed on state
  - This information is 1-2 frames old
- All machines do one `update()`
What To Check?

• Anything that’s likely to be out of sync, or summarizes lots of game information

• 16-bit “CRC” of 32-bit values, vectors…
  – Current random seed
  – Player position, state, health, …
  – Camera position, target, …
  – Number of active entities
Marvel Ultimate Alliance PSP 1

• Action RPG with an isometric view built on engine from X-Men Legends
• Networking originally introduced in the engine after X-Men Legends shipped
  – There was already a large amount of single-player code
• Needed multiplayer via the internet
• Client server would be nice, but…
Marvel Ultimate Alliance

• Retro-fitting key-sharing is much easier than proper client-server architecture
  – That just leaves latency issues

• DS had ~30 ms latency, and guaranteed delivery (for local Wireless)

• Internet has ~300 ms latency, and packet delivery is like playing the lottery
Buffered Input

- High level logic to smooth out problems with the delivery
- Input is buffered for some number of frames
- So the input being acted on is old. How is this useful?
Buffered Input 2

• All machines can detect that delivery has slowed down (or sped up)

• Subtle changes can be made to keep the buffer at a target length
  – Rate of input sampling
  – Simulation framerate

• If buffer is too small, delivery failure makes the game stop completely

• If buffer is too large, input delay is really bad
Future Work?

• Marvel Ultimate Alliance lacked join-in-progress. We need to fix that.
• More work could be done on simplifying the process of starting a game.
• More community oriented features.
The End

Thanks for coming.
We love questions!