Network Programming

Euan Watt 1200755

Network Architecture

Hybrid client-server

Advantages:

1. Player acts as server
2. Spectators connect as clients to server
3. Allows for minimal network lag
4. Discrepancies are non-existent as server has true physics simulation

Network Architecture

Hybrid client-server

Disadvantages:

1. Player may have firewall/NAT issues when accepting connections
2. Player uses more bandwidth the more people are spectating
3. If players’ game or internet crashes, all spectators are disconnected - application problem; more on this later

Application-level Protocols

Server-side

1. TCP
2. Need a reliable connection
3. Relatively fast-paced game
    a. Out-of-order messages could affect viewing experience
    b. Major changes in lag (from UDP changing routes) are bad
4. TCP can send smaller packets, or at least smaller ‘chunks’
5. Not enough messages being sent to make UDPs efficiency worthwhile - TCPs adaptation to network conditions is more necessary for the game type

Application-level Protocols

Client-side

1. TCP
2. Could have gone for a UDP server, to accept a broadcast message
    a. Only clients in device IP range could accept message
    b. Really only suitable on LAN connection

Network API

Game Maker (v1.4.1474)

1. Uses proprietary language GML
2. Offers Box2D out of the box (used server-side)
3. Has built-in networking functions and variables - very similar to those in WinSock
4. Can use a TCP or UDP connection
5. http://gmc.yoyogames.com/index.php?showtopic=604116 - Tutorial used to set up basic connection (accessed 23/11/14) - FatalSheep?, 17 December 2013

Networking Code Structure

Server:

1. Switch case handles event types
    a. network_type_connect
    b. network_type_disconnect
    c. network_type_data
2. Multiple scripts
    a. ReceivedPackets()
        i. sends different messages depending on the type of information being sent
        ii. each packet is given an ID to dictate what type of message it is
    b. SendPegInformation() - sends initial peg positions and state

Networking Code Structure

Client:

1. Same as server - switch case handles event types
2. Single script - ReceivedPacket() reads the buffer depending on the message type that has been received

Position Prediction Techniques

1. Quadratic model
   a. Due to vertical physics in play
   b. Smoother motion client-side - important for spectator
2. Initial position, speed, direction message
3. Client predicts from there until collision occurs
4. Certain conditions dictating velocity
   a. No objects on client side will cause deceleration
   b. When moving vertically up, decelerate at constant rate
5. Once-per-second update from server (maybe less)

Interpolation Techniques

1. Linear model
   a. Despite the need for smooth movement
   b. Prediction shouldn’t be ‘that’ far off
   c. If time allows, Bézier curves would be interesting (used these for coursework last year)
2. If prediction is overly wrong, simply snap to true location and restart prediction
   a. Avoids debate on possible collisions during interpolation

Critical Evaluation

1. New clients connecting after player has started playing will only see pieces of the game
2. When the player disconnects, all play on clients will end
3. Position prediction may see client-side ball pass through objects
   1. Once-per-second updates may not be enough
   2. However, this is only to simulate latency; prediction ‘should’ keep up
4. Linear interpolation may look out of place, jumping object most definitely will; but necessary to keep state of play

Changes Since Demonstration

Demonstration

1. Working server application
2. Client connection established
3. 'Ping' between client and server

Now

1. Peg messages
2. Ball messages
3. Collision messages
4. Working client application
5. Position prediction (client)
6. Interpolation (client)

Thank You for Listening

Any questions?