Delay Compensation in Physically Realistic Networked Video Games

Masters Project: Robert F. Buchheit
Adviser: Vincenzo Liberatore

Motivation

- The computer gaming industry is experiencing rapid growth, particularly in the area of networked games
- Many of these games are physically realistic or evolve according to similar sets of equations
- Such games usually require an excellent level of network service - low latency and moderate bandwidth in order to be playable

Motivation

- As a result, many players choose to play only over LANs to avoid problems associated with network latency
- To improve quality of gaming over WANs and allow games to reach a wider market it is necessary to compensate for less than optimal network service

Project Goals

- The goal of this project is to develop a demonstrative physically realistic network game
- Develop and studying delay compensation techniques
- Provide a framework for further study

Delay Compensation - Background

- Human interaction with the physical world is based on a sense and response cycle
- People behave very similarly when interacting with virtual environments
- The sense and response cycle is very dependent on accurate perception and immediate feedback in guiding decision making

Delay Compensation - Background

- When a user is separated from the virtual game environment over an IP network, delay is introduced to the interaction
- Small delays are generally not a problem as users will rarely notice them.
- As delay climbs interaction becomes more difficult and introduces several problems

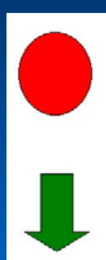
Delay Compensation - Background

- 1. User’s view of the environment is dated by 1/2 RTT
- 2. User’s interactive inputs require additional 1/2 RTT to reach remote server and be applied
- 3. User sees no control feedback for 1 RTT

Delay Compensation - Background

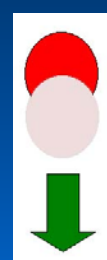
- Goal of delay compensation is to minimize these problems and make higher delay interaction as similar as possible to low-delay interaction
- The basic strategy to overcome these problems involves extrapolation of the game state

Delay Compensation - Background



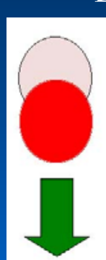
- By extrapolating the game state forward, the game client can help limit the effects of delay and make the user’s interaction more natural.

Delay Compensation - Background



- Using the information provided by the server, the game client projects the state of the environment at a point 1 RTT ahead of the current state
- This causes the state displayed to the user to be 1/2 RTT ahead of the remote state

Delay Compensation - Background



- The extrapolated state is then shown to the user who interacts normally
- When the user’s inputs reach the server 1/2 RTT later the actual state should closely match the extrapolated state

Delay Compensation - Background

- This solves the first 2 problems
- To solve the 3rd, the user’s input is buffered locally and applied to incoming data.
- This allows the user to see the results of their control input and modify their subsequent inputs appropriately

Delay Compensation - Limitations

- The strategy outlined above works well in many situations but is limited in its effectiveness by the nature of physically realistic games
- Similar to the real world, there are many events which can't be predicted
- Thus, attempts to extrapolate game state ahead will always be somewhat flawed

Delay Compensation - Limitations

- When an unpredicted event occurs the user will not receive notification of it until 1/2 RTT later
- Furthermore, they cannot react to the event until 1 RTT has passed (at the earliest)
- If the event has critical timing (such as a collision) this may be too long

Delay Compensation - Limitations

- Only way to combat this would be a form of server-applied "contingency control"
- Events with less critical timing as well as smaller extrapolation errors must also be dealt with
- Despite limitations, delay compensation is still a useful tool

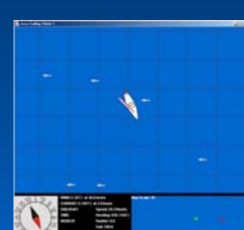
Experience

- For the purpose of this project two games were developed
- The first game was a prototype for the second and provided experience as well as a chance to test DC with a fast-paced game
- The second game is a realistic sailing game with somewhat slower dynamics.

Experience - Prototype

- Control a "JetCar"
- Avoid obstructions to gain points
- Loose points for collisions
- Critical timing of collisions poses difficulties for DC

Experience - Final



- Control a Sailboat
- Sail and rudder controls
- Race to a buoy given varying wind and current conditions
- Also, can compete against AI

Experience - Final

- Delay compensation is very effective for this game
- This is aided by the GUI design of the game as well as the nature of the physically realistic system (fewer critically-timed events)
- More critically timed events can be added to the game with the addition of islands to the open ocean

Final Game Systems

- The final game incorporates a number of subsystems in addition to the DC system
- Physics and Game play
- Animation/Rendering
- Network Connectivity and Protocol
- Clocking/Timing

Delay Compensation System

- The DC system for the sailing game has three main features
- First, it attaches timestamps to each clocking/input packet sent to the server
- Second, it uses a ring buffer to store client inputs using the same timestamps
- Finally, a local copy of the server-side system is used to evolve the system forward using the buffered client controls

Physics System

- The physics of the system are based on several simple equations
- The force generated by the sail is given by the equation: $F = C (\sin \alpha) 0.5 \rho V^2 A$ where V is the velocity of the apparent wind and α is the sail's angle to it.
- The forward component of the force is then calculated as: $Force = F \sin(\theta)$ where θ is the angle between the sail and the boat

Physics System

- Leeway forces are ignored at the present time
- The drag experienced by the boat is given by: $Drag = 0.5V^2$ where V is the boat's velocity
- The turn caused by the rudder is given by: $Turn = C V \sin(\theta)$ where θ is the angle of the rudder
- The turn directly redirects the boat's Vel.

Physics System

- For purposes of game play the system is tuned to evolve relatively quickly (low boat mass for instance)
- Additionally, the boat's speed is not limited by more complex physical factors such as hull length
- Overall, the boat's behavior is more similar to a light sail board than a heavier yacht

Game Play System

- The player uses they keyboard to control the boat
- ‘A’ pulls the sail in, ‘D’ lets it out
- Left and Right arrow keys move the rudder accordingly while Up or Down center the rudder
- If the player chooses ‘God-mode’ they can also control the wind direction and speed

Animation/Rendering System

- This system is responsible for displaying the game state to the user
- Each component of the display is rendered by its own function to enhance code clarity and make changes convenient
- Double buffering is used to eliminate animation flicker
- To help accommodate all systems the user can run the game in two display modes

Networking System

- The network “layer” of the game system is designed primarily to minimize end-to-end latency between the client and server
- Packets are sent using UDP
- All data is transmitted in string format for simplicity and ease of transport
- Both client and server maintain network listener threads which place incoming packet data into a buffer to be read

Networking System

- The buffer is given only one slot as, at the time of update, only the newest data is relevant
- Additionally, packets delivered out of order are discarded
- The network class also implements a priority tag system to make sure game control messages (Start/End) are not over written accidentally

Clocking System

- The update clocking of the game system is designed around the fact that the JVM runs different on different platforms
- To help minimize issues the client’s inputs are used to clock the server
- The client is designed to clock in a reliable manner while the server is designed to update the game as continuously as possible and send data on client inputs.

Conclusions

- Despite certain limitations delay compensation is a useful tool for improving network game playability
- Contingency controls mechanisms coupled with delay compensation stand to make remote operation of real systems over IP networks feasible
- This sailing game will provide a solid framework for further study of the topic