Shots Fired

Group 17

scmaier, d2lal, samiskin, wcakeize

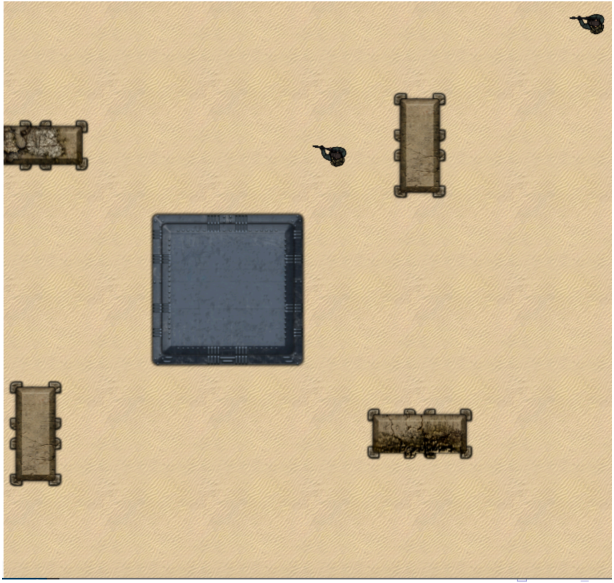
University of Waterloo

September 29, 2016

Brief Project Description

1. Top-down arena style 2D online multiplayer shooting game
2. Can easily join a game with friends (low barrier to entry)
3. 2-4 player game played on separate machines connected online
4. When a player loses all their health they are out
5. Last person standing wins!

Game Screen Capture



Gameplay

1. Each player has x amount of health
2. Move with WASD
3. Move mouse to aim.
4. Click to fire.

Design Challenges

1. Creating a game with character interactions, animations, controls, physics, online connections
2. Synchronizing differences between client and server game states
3. Running and managing multiple game servers simultaneously
4. Ensuring easily extendable game object interaction.

Handling Game Operations

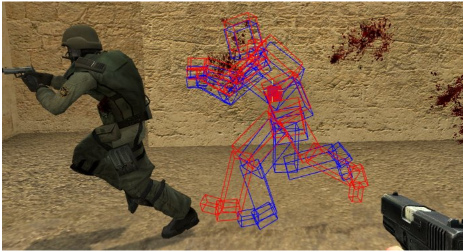
1. Various operations were split into stages, each modifying the game state and passing it to the next stage, resembles pipe and filter
    1. Apply game inputs
    2. Handle any game object deletions
    3. Update individual entities
    4. Resolve interactions between entities
2. Interactions between entities handled using events
3. Attempted a more functional approach, however mutability was still allowed for performance reasons, as updates are made at 60 frames per second.

Client Synchronization

1. Clients forward **only their inputs** to the server
2. Game continues to be simulated on the clients
3. Server collects inputs from all clients
4. Server applies inputs and updates
5. Server forwards the entire state of the game to clients (the game state acts as a blackboard)
6. Clients re-sync on receiving the server copy

Mitigating Latency Effects

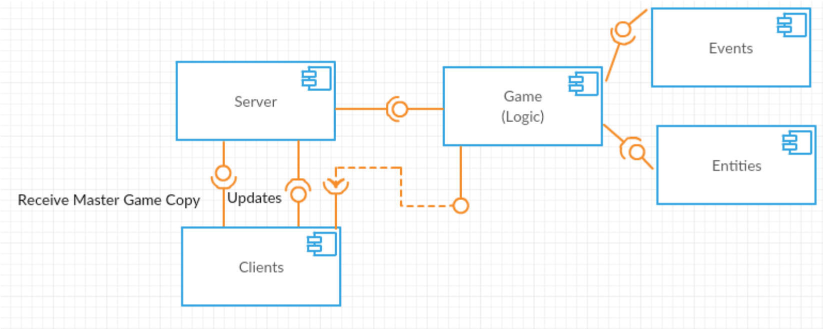
1. The game state received by the server may be significantly delayed
2. The server copy is simulated up to the current time on the client
3. Inputs from the client which occured after the time of the server update are reapplied



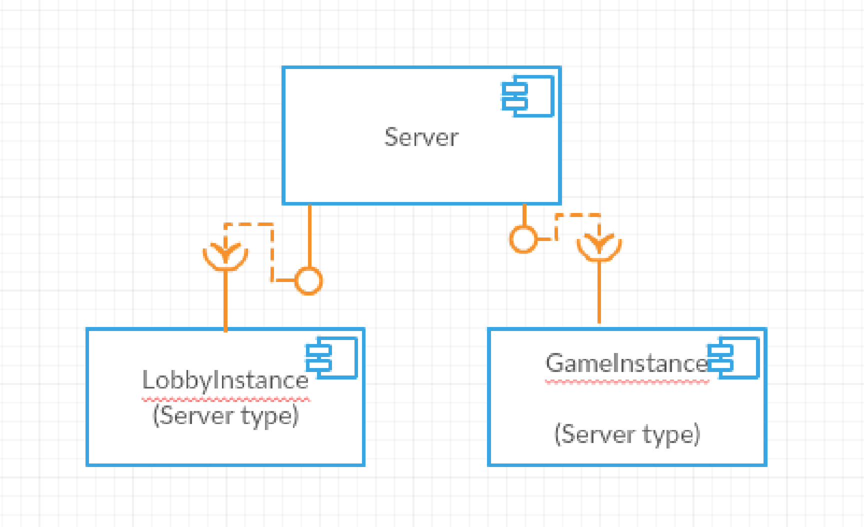
Node.js Concurrency

1. Efforts were made to improve performance through concurrency, using multiple cores
2. Problems due to JavaScript being single-threaded by design and Node.js having poor multi-process support
3. Issues also arise with deployment due to only one port being allowed

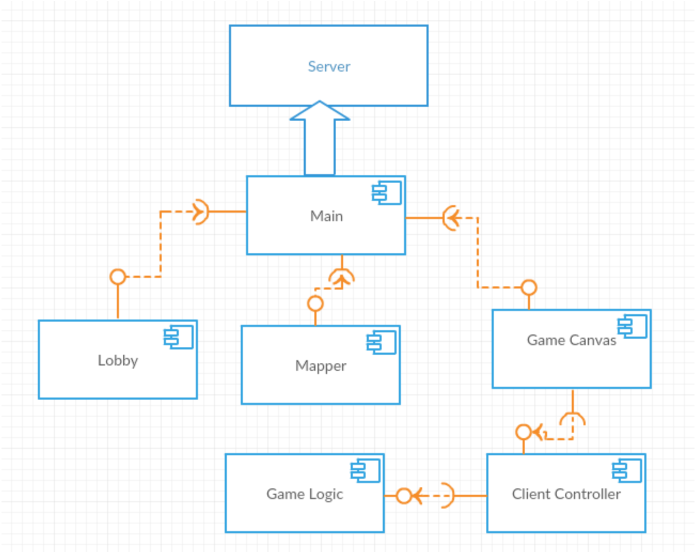
Overall Architecture



Server Architecture



Client Architecture


Server
Main
Lobby
Mapper
Game Canvas
Game Logic
Client Controller