Hunger Games: May the Odds Be Ever in Your Favor

Christine Samson (casamson)

Nolan Cretney (nokynokes)

Evan Su (hexacyanide)

Michael Xiao (MDXiao)

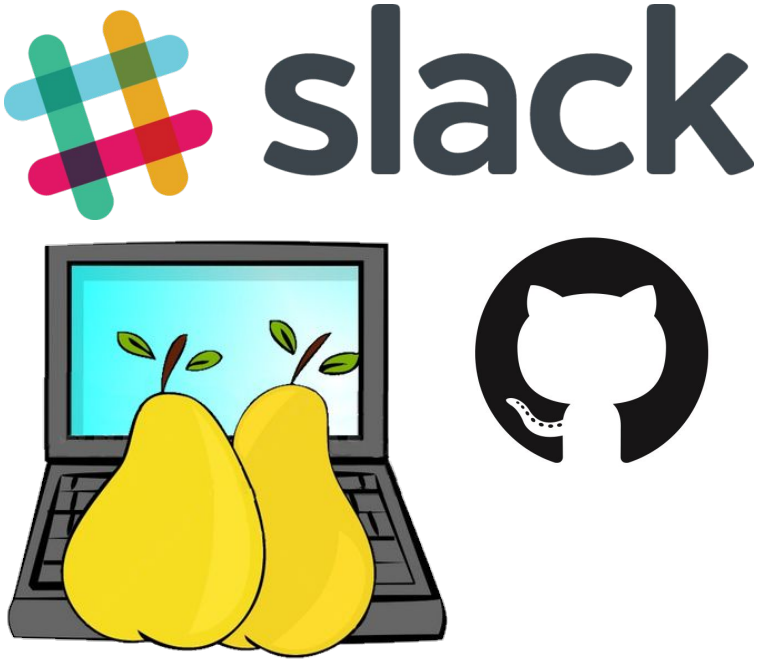
David Kleckner (D-Kleck)

Game Vision

- To provide a source of entertainment
- Make available a platform to individuals that desire to participate in larger collaborative gaming environments
- Hunger Games style elimination game
- Play to survive in the Hunger Games

Methodologies

- Slack
  - For better team communication
  - Schedule team meetings
  - 4/5, We mainly saw each other day every day.
- GitHub Issues/Milestones
  - To keep track of project priorities
  - 3/5, needed more contribution.
- Pair Programming
  - Debug code
  - Fix issues
  - Team Collaboration
  - 5/5, very helpful, as we debugged issues faster.



Two Districts

Instance Server

- Written in server-side JavaScript (Node.js)
- Responsible for recording client events
- Maintains a persistent record of client interactions

Game Client

- Created with Unity 3D engine
  - 4/5
- The visuals of the game - what the user sees while playing
- Written with C#
- Made weapons and characters with Blender: 4/5

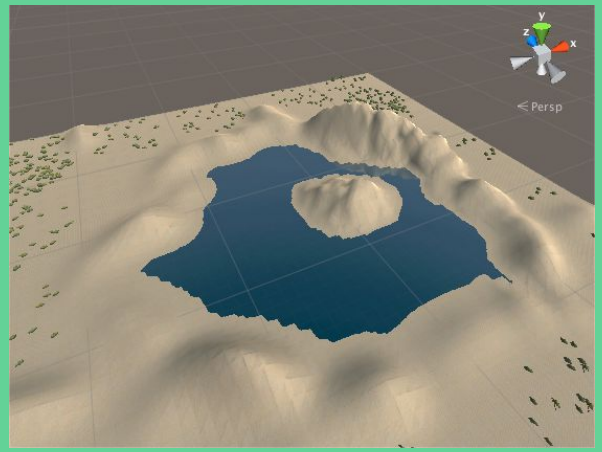
Instance Server - Why Node.js?

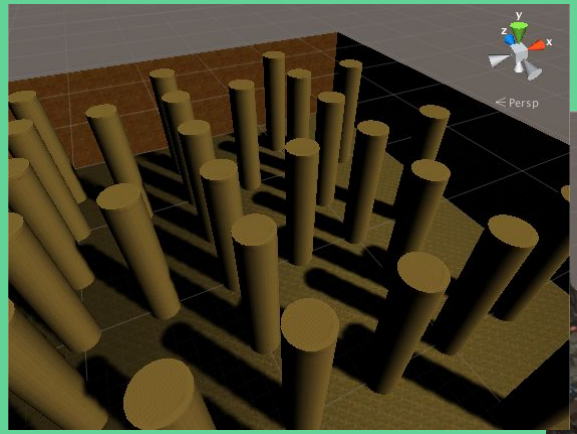
- All JavaScript
  - Uses the V8 engine developed by Google. V8 compiles and executes JavaScript at lightning speeds mainly due to the fact that V8 compiles JavaScript into native machine code
  - Use the same language on server and client, and share some code between them
- Event Loop
  - Single thread that performs all I/O operations asynchronously. It sends an asynchronous task to the event loop, along with a callback function, and then continues to execute the rest of its program. This allows for non-blocking code
- Tools and Libraries
  - Npm stands for Node.js package manager. It is fast and very efficient when it comes to searching for libraries and saving dependencies.
  - Includes an ever-growing pool of packages to pull from

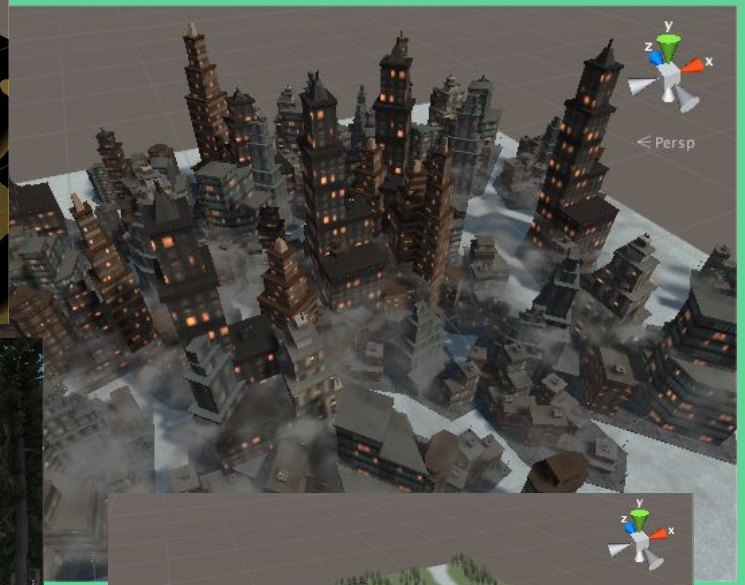
Instance Server - Specification

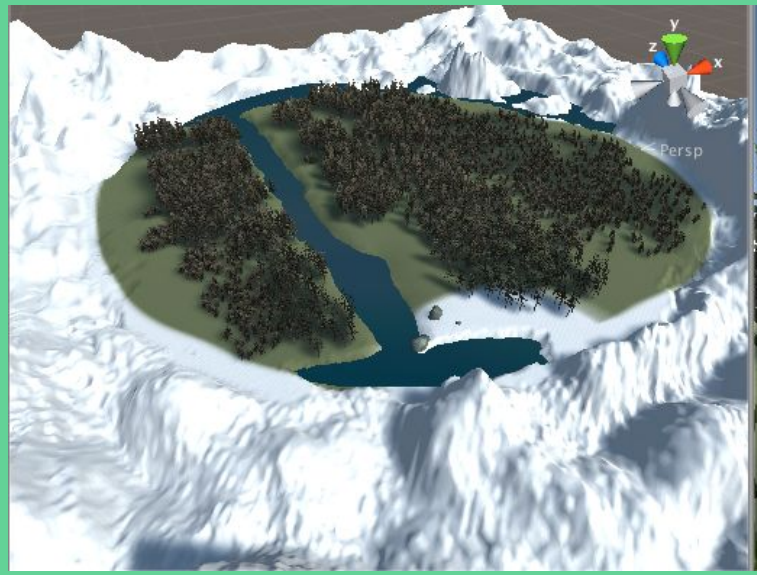
- Data Tracking
  - Usernames, sessions, characters
  - Character metadata (e.g. health, position)
  - Level metadata (coordinates, time)
- Session Handling
  - User authorization, username validation
- Distributed Computation (multiplayer)
  - Broadcasting data (one-to-many)
  - Server-side non-player entity AI
  - Client state synchronization

Game Client









Game Client Specs

- Five Fighting Arenas
  - cityArena
  - concreteArena
  - desertArena
  - forestArena
  - snowArena
- One Training Arena
  - trainingArena
- 3D Playable Characters with Weaponry
  - Spawnable enemies
  - Health indicator

Challenges Encountered

- Network Issues
  - Server and client had to be developed independently.
  - Time constraints for project completion.
- Learning Unity/Blender/Node.js
  - Starting from scratch on C#, learning to navigate the environment.
  - Learning that everything how everything is connected.
  - Playing around on the hundreds of features that Unity had to offer.
  - Trying to understand web frameworks, how client and server communicate
- Familiarization with GitHub.
  - Learning to use the Git shell to commit and pull changes.
- Time Constraints
  - We need a little more time to think of a game that we would find fun to build.