Corey Hall, Devin Tuchsen, Sergio Avedillo, Miguel Clement

Design Document

Tools and Technologies

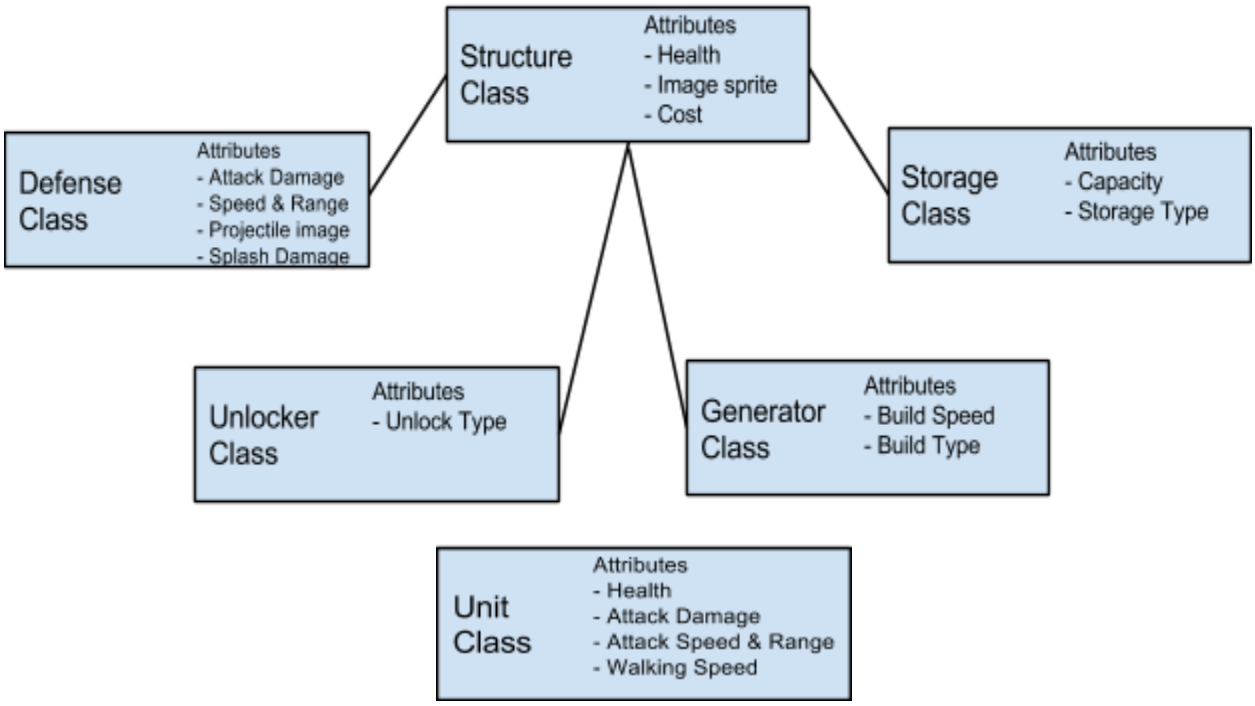
- CreateJS
  - Client side framework for creating all graphics and animations
- Socket.IO
  - Client socket connections
- NodeJS
  - All server code
- HTML5
  - Views
- CSS3
  - Styling

Data Structures

Grid square - a class containing information on whether or not the square is occupied and if so what type of object is on it (defensive or offensive).

Game grid - two dimensional array or grid squares.

Structure class hierarchy:



WebSockets Implementation

All communication will take place using Socket.IO and NodeJS. A number of listeners will be set up on the clients and server, which will handle the following:

Nodejs/Socket.io listeners

- Game lobby listener
  - Player joins game
  - Player readies up
- Game state listener
  - Player status
    - In lobby
    - In game
    - In victory/defeat screen
- Unit performed action
  - Unit location changed
    - x, y pos
  - Unit attribute changed
    - Health
    - Alive/Dead

Algorithms

- **A***

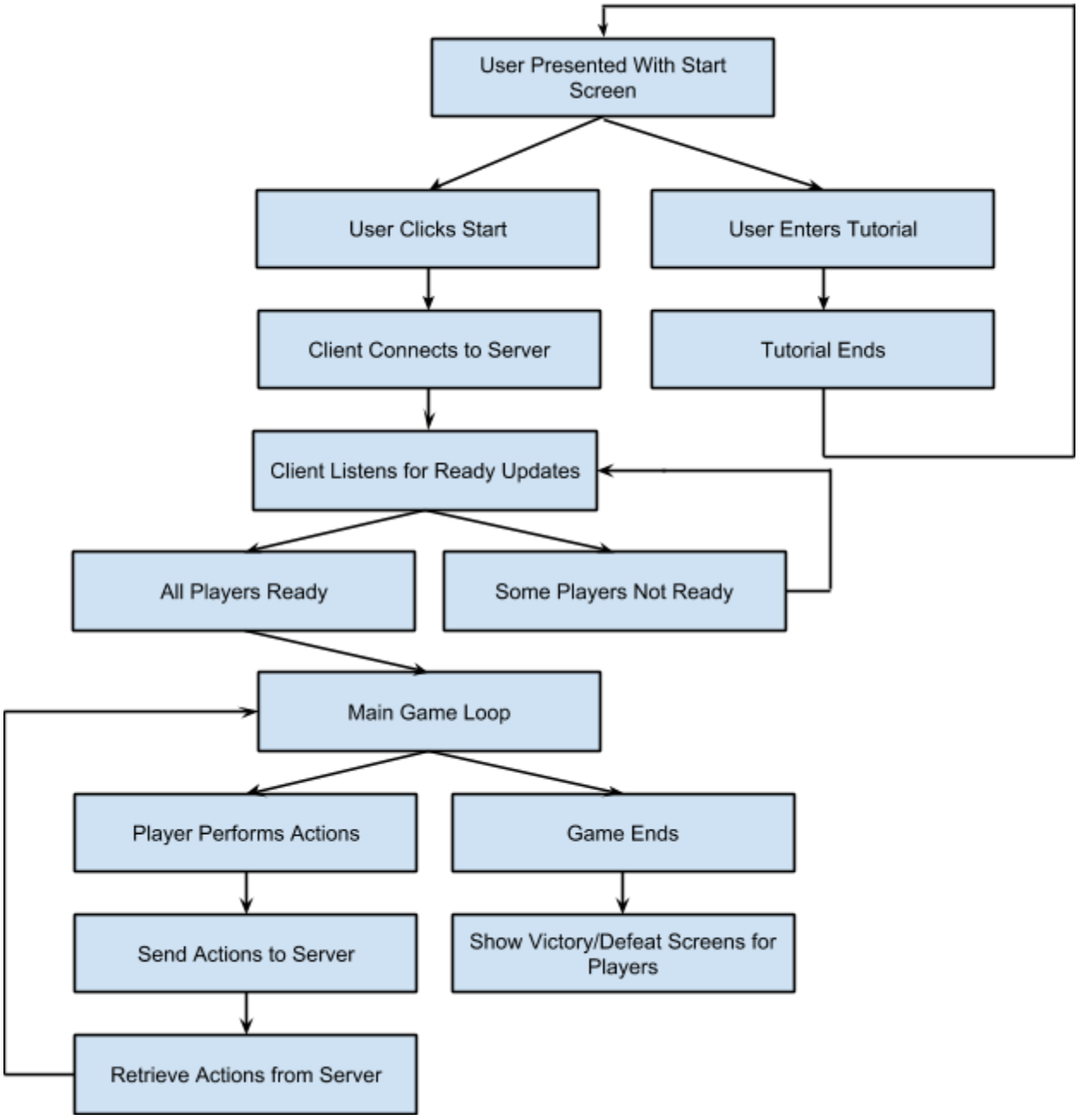
All zombie movement will be controlled by the A* pathfinding algorithm. The game grid will be broken up into a graph structure where squares that are occupied by structures will be blocked and all other spots will be free. This algorithm will allow the zombies to find the shortest path to their target and how to get there without moving through walls and buildings.

- **Opponent Targeting**

Zombies and defense structures will choose their targets based on euclidean distance. They will swarm to their nearest opposing target and strike. This allows zombies and defense turrets to attack different targets and not just focus on a single goal.

Hosting: http://students.cse.tamu.edu/devint1/outbreak/
Repo: https://github.com/clement360/Outbreak
Development Log: https://github.com/clement360/Outbreak/commits/master

High-Level Process Flow



When the player first starts the game, he or she will be presented with the splash screen. From there, the player will be able to either start the game or watch a tutorial. If the user starts the game, the client will connect to the server and continually refresh ready statuses until all players are ready. Once all players are ready, the client will enter the main game loop. Each player will perform his or her actions, which will get sent to the server. At the same time, the client will also be continually refreshing the game state from the server to update other player's actions. Once a victory condition is met, the game will end and victory/defeat screens will be shown to the players.

Corey Hall, Devin Tuchsen, Sergio Avedillo, Miguel Clement

Shortcut:

\\filer.cs.tamu.edu\ugrads\d\devint1\web_home\outbreak

Tutorials:

Node for beginners:

https://code.tutsplus.com/tutorials/nodejs-for-beginners--net-26314

Using Create Js

https://code.tutsplus.com/tutorials/using-createjs-easeljs--net-34840

Sockets and node

https://code.tutsplus.com/tutorials/real-time-chat-with-nodejs-socketio-and-expressjs--net-31708

Using Node Js and Socket.oi for a chat service

https://code.tutsplus.com/tutorials/using-nodejs-and-websockets-to-build-a-chat-service--net-34482

Sockets and Chat with Node js 2011 Date

http://martinsikora.com/nodejs-and-websocket-simple-chat-tutorial

JavaScript Tutorials

http://stackoverflow.com/questions/2353818/how-do-i-get-started-with-node-js

Multi-player
http://code.tutsplus.com/tutorials/connect-4-with-socketio--cms-19869

TO-DO:

1. pathfinding
2. Drag drop of buildings and defenses **--- I think I can work on this** *Sergio*
3. backend related to bulding (refresh everyone elses view). ***And I guess this is tied to it :)***
4. Menu Stats (health, money..etc)
5. Building images, zombie images (In the meantime use squares and circles or something, we'll add the images after)
6. Add more to the to-do list..