Purpose

The purpose of this design document is to note all details of this implementation, patches, frameworks, and design patterns in order to effectively manage code and maintain code health. This document is used so our team is effective when creating the project.

People
Logan Scott
Paul Morgan
Ibrahim Ayoob
Vivek Ganesan

Project Name
4 Player Pac Man

Technologies Used
node.js
packages.json
PNG

Files Used
Game.js - contains actual implementation of the game
Server.js - contains server view for the 4 players playing the game
Client.js - contains the code for each client play
Lobby.css - contains the layout for the board
Account.js - contains the accounts for each of the players
Index.js - contains the indexing for the board
Lobby_client.js - allows client to choose physical appearance
Lobby_server.js - waits for at minimum 4 clients to connect

Folders Used
->node_modules - contains the folders, modules, and packages needed to execute servers, clients, connections, GUI, and UI interface
->assets - contains all the external images needed to upload into js files
->js - contains all of the node.js files used in this software design project
->test - contains miniature test suites used for testing the Account.js files

Game.js

1. preload() - preloads all the necessary setup materials for the game

2. create() - creates the board, the game, the animations, the sockets, clients, servers, utilizes server.js in order to create a working server
3. update() - updates the game board each time a button is pressed that will cause an event driven action to occur
4. addPlayer() - main purpose of this function is to add a player into the game screen
5. addOtherPlayers() -focuses on the multiplayer aspect of the game. This function incorporates more players into the game.

Server.js

Server.js builds a local connection for all players to connect to 1 server which runs the game. Through this connection, all players have access to a single unit of the game.

Variables:

1. Express = require('express) create user interface
2. App = express(); create user interface
3. Players = {}; empty because no players at the start
4. IO = require('socket.io').listen(server)
5. Server = require('http').Server(app);

Client.js

Client.js represents the clients that are connected to the server. These individual clients represent the individual players where the data regarding each client is maintained and directed by the server.

This big file contains several hundred lines of code that determines how the clients should act. Information is passed in the form of a buffer. When information goes to the server, a buffer stops it and passes that onto the server. The server will then return the information from the client in the form of a move.

Lobby.css

Lobby.css contains the variables, templates, and tools needed to create the board. The board is the same for every play. Lobby.css contains boxes, mazes, and barriers of different colors. These colors all have hard-coded values indicating the the board layout is the same for every player. Each box is indicated in the comments of how the layout will appear once this ccs stylesheet is accepted by the node.js platform.

Account.js

Account.js contains a single class called Account. The name of this class is self-explanatory. In this class, an account class is used to keep track of each of the players. Each of the players is given an ID, a name (chosen by the client), a position, a color, and a score. Each time the game updates, the account for each client will be updated.

Index.js

Index.js is what keeps track of all the positions on the board. The board is guided by a series of barriers. These barriers serve as regulations in order to prevent clients from going past barriers or drifting off the board. This file also loads a series of pictures from the UIC CS Department Faculty Website (for the Hallenbeck photo) as well as creative pictures saved as png files made by Logan Scott.

Lobby_client.js

Lobby_client is a js file that will allow the clients to pick and choose their features. One feature in this game that has been allowed is the ability for clients to choose how they look on the board. Lobby_client.js has a series of functions where each function will return a separate color and that color will allow the client to appear a different way on the game board.

Lobby_server.js

Lobby_server.js is a listening file that will initiate a game only when at minimum 4 clients are connected to the game screen. Using connection logic, the server will update the number of clients in real time and will initiate game play when at least 4 are connected. The same logic used in the server.js to create a server client connection is used in Lobby_server.js.