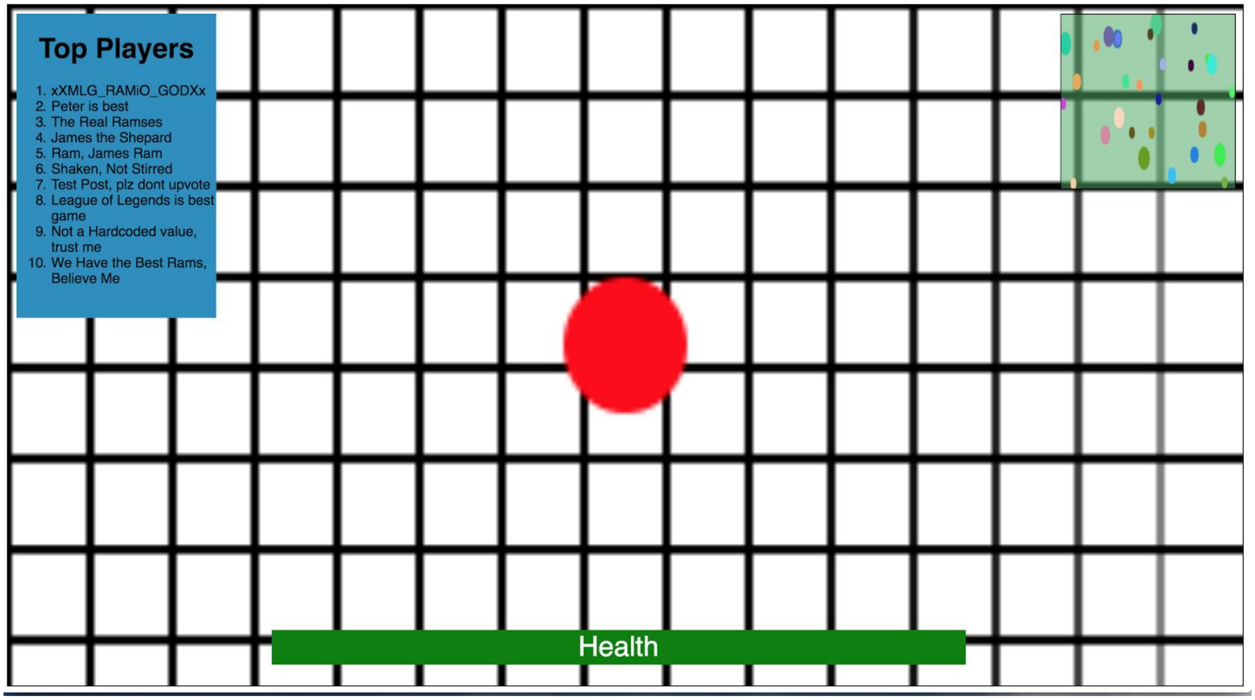
Team Name: Ram.io

Project Description:

Ram.io is a realtime multiplayer http-based, resource collection game akin to agar.io. The resources of the game are grass and fur. Players play as grass eating and fur collecting rams. Ram.io backend based off Express.js and socket.io on top of Node.js. Ram.io front end HTML5 (including Canvas as a major component to draw all of the game objects), CSS, and Vanilla JS with jQuery.

Screenshots:



Top Players

1. xXMLG_RAMiO_GODXx
2. Peter is best
3. The Real Ramses
4. James the Shepard
5. Ram, James Ram
6. Shaken, Not Stirred
7. Test Post, plz dont upvote
8. League of Legends is best game
9. Not a Hardcoded value, trust me
10. We Have the Best Rams, Believe Me

Scoreboard (Top Left):

Lists top 10 players in the current game session based on resources gathered. A SQL database will contain player profiles, their highest score achieved, last time logged in, highest placement on the scoreboard, username and password for their account, and other important information.

The current session will have a separate table in the database which will hold all the players and information pertinent to the current game. We are separating the current session data from the total set of accounts to increase database query speed which will increase responsiveness of scoreboard. After each session, relevant data is transferred to a permanent database table, and the table associated with that session is deleted.

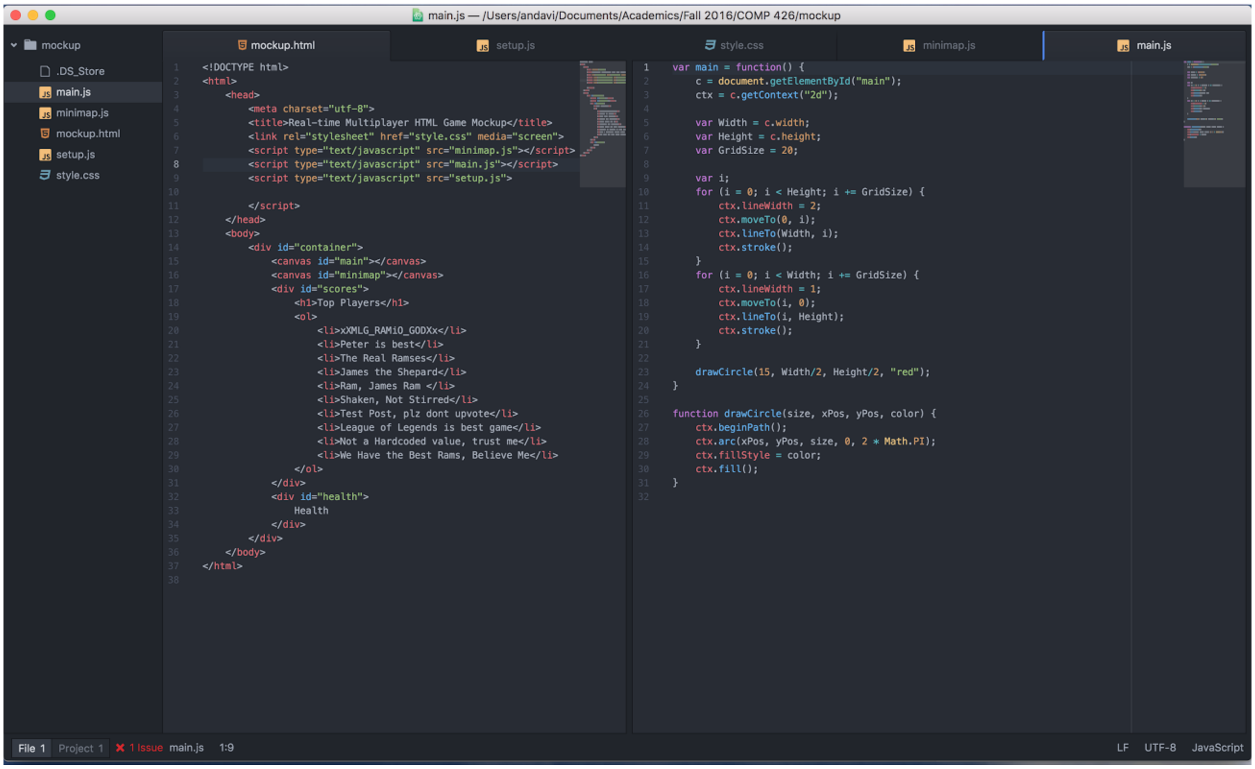
Minimap (Top Right):

Players will broadcast their current location to the server. The server will then broadcast that location to all of its listeners (the players of the game). The client side JS will then interpret this broadcast and display the location of all the other players in a minimap at the top right of the screen.

Game:

The orange dot in the center of the picture above is a placeholder for the player avatar, which we currently envision to be a stylized avatar of a ram. The green bar at the bottom is the health bar for the player. The map will be a static playing field every session, containing statically placed obstacles to give another level of strategy to the game. The static map will give a consistency to the base strategy of the game, when all other factors will change (the player's starting position, every enemy having variable sizes and starting positions, etc.).

HTML,JS,CSS Code for our mockup:



```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Real-time Multiplayer HTML Game Mockup</title>
        <link rel="stylesheet" href="style.css" media="screen">
        <script type="text/javascript" src="minimap.js"></script>
        <script type="text/javascript" src="main.js"></script>
        <script type="text/javascript" src="setup.js">

        </script>
    </head>
    <body>
        <div id="container">
            <canvas id="main"></canvas>
            <canvas id="minimap"></canvas>
            <div id="scores">
                <h1>Top Players</h1>
                <ol>
                    <li>xXMLG_RAMiO_GODXx</li>
                    <li>Peter is best</li>
                    <li>The Real Ramses</li>
                    <li>James the Shepard</li>
                    <li>Ram, James Ram </li>
                    <li>Shaken, Not Stirred</li>
                    <li>Test Post, plz dont upvote</li>
                    <li>League of Legends is best game</li>
                    <li>Not a Hardcoded value, trust me</li>
                    <li>We Have the Best Rams, Believe Me</li>
                </ol>
            </div>
            <div id="health">
                Health
            </div>
        </div>
    </body>
</html>
```

```javascript
var main = function() {
    c = document.getElementById("main");
    ctx = c.getContext("2d");

    var Width = c.width;
    var Height = c.height;
    var GridSize = 20;

    var i;
    for (i = 0; i < Height; i += GridSize) {
        ctx.lineWidth = 2;
        ctx.moveTo(0, i);
        ctx.lineTo(Width, i);
        ctx.stroke();
    }
    for (i = 0; i < Width; i += GridSize) {
        ctx.lineWidth = 1;
        ctx.moveTo(i, 0);
        ctx.lineTo(i, Height);
        ctx.stroke();
    }

    drawCircle(15, Width/2, Height/2, "red");
}

function drawCircle(size, xPos, yPos, color) {
    ctx.beginPath();
    ctx.arc(xPos, yPos, size, 0, 2 * Math.PI);
    ctx.fillStyle = color;
    ctx.fill();
}
```