CS503 Capstone Multiplayer Game – agar.io

Software Design Document

# Overview

Agar.io is a multiplayer online action game.

Player control a cell in map, the goal is to gain as much mass as possible by eating agar(randomly generated pellets) and cells smaller than the player’s cell, while avoiding larger ones which can eat player’s cells

Realize the game by using HTML5 canvas, JavaScript, Nodejs

# Major Use Case

1. User can login the game with a name
2. User can control the character to move by moving mouse
3. User can see other players
4. User can make the character to consume the food by hitting the food
5. User can make the character bigger by consume the “food”, the bigger the character is, slower it moves.
6. User can double the character’s number by shrinking the character’s size to half
7. User can eject a small ball by reducing it’s own mass
8. User can make the character to consume other smaller mass characters by hitting the other characters
9. User can check their rank on the leaderboard
10. User can use chat window to send public message

# High Level Design

|  |  |
| --- | --- |
| Stack | Technology |
| Frontend – client | HTML5 canvas, Socket.io |
| Frontend – server | Node.js, Socket.io |

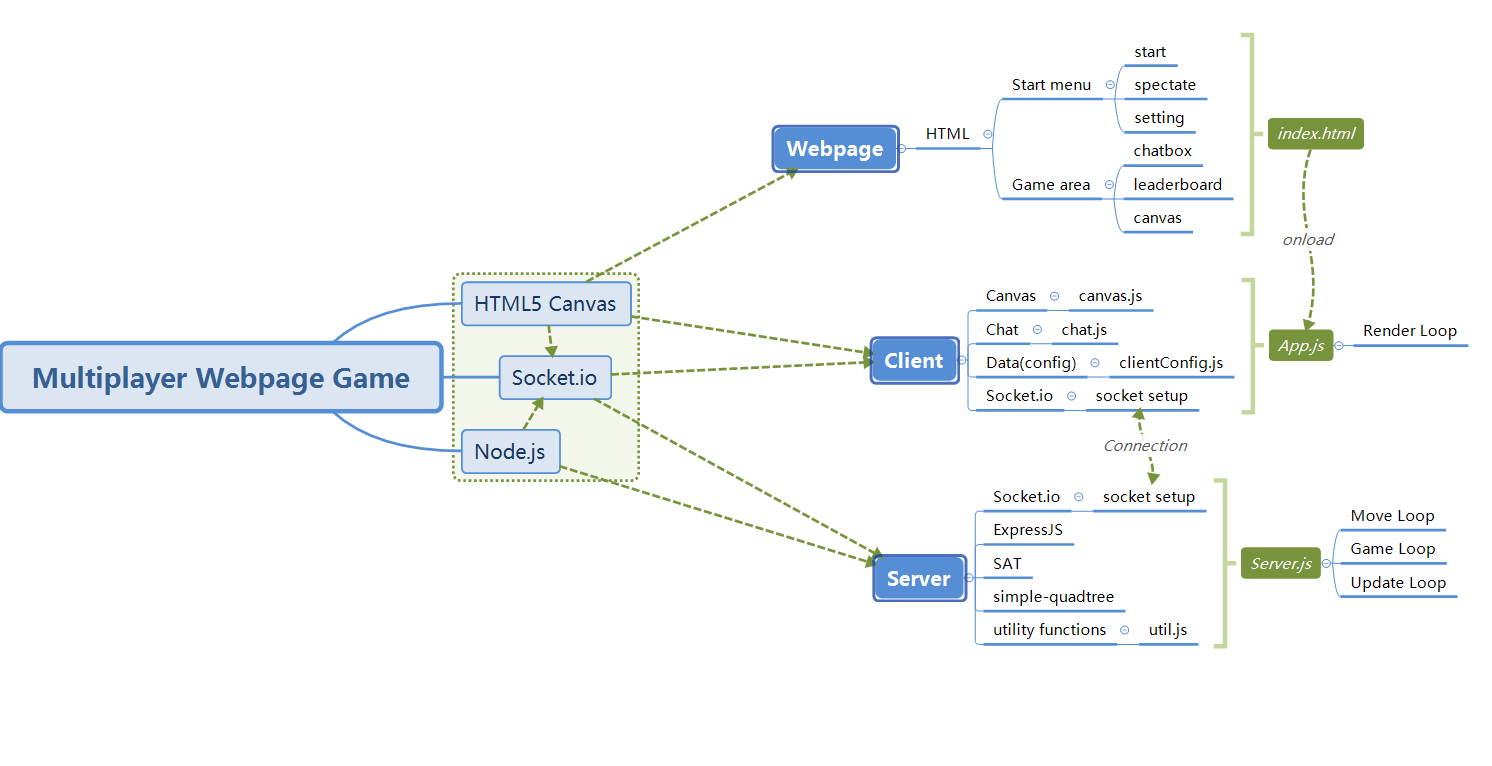
Socket.io

Socket.io

The game runs on a NodeJS and use Socket.io to create a WebSocket server to listen on certain port.

Use ExpressJS to setup a simple HTTP service that display index.html, which has Canvas element used to render the game and client side Javascript to communicate with Socket server.

# Detail design



## Webpage

Start Menu

Game Area:

Chatbox

Leaderboard

Game canvas

## Game Client

Game Canvas:

Event listener for input behaviour

Chat:

Get input from chatbox

Add line to chatbox

App:

Socket.io setup

Game render loop

## Game Server

Socket.io setup

Move loop

Game loop

Update loop

## Overall Data schema:

|  |  |  |
| --- | --- | --- |
| Name | Location | Description |
| Food[] | Client & Server | Food position |
| Virus[] | Client & Server | Virus position |
| FireFood[] | Client & Server | Eject mass position |
| Users[] { Cell[] } | Client & Server | All users id and its cell info |
| Leaderboard[] | Client & Server | Leaderboard info |

## Client side Data:

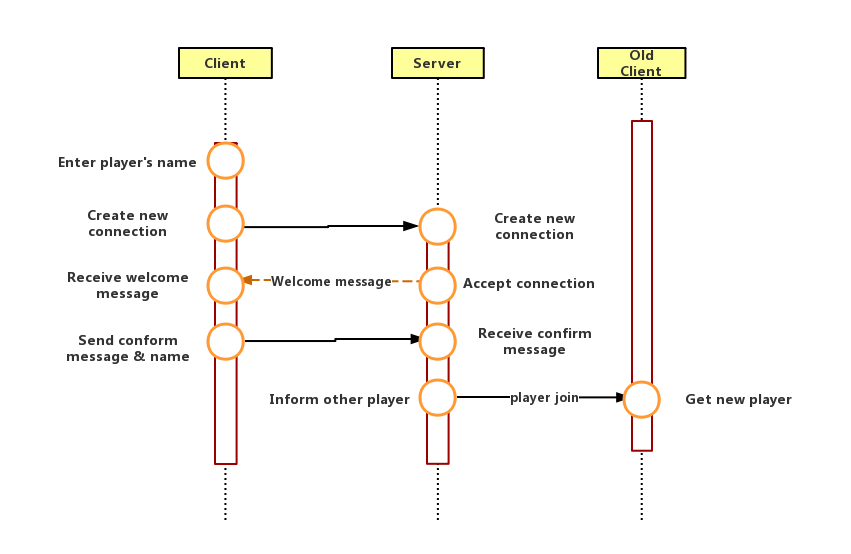
|  |  |  |
| --- | --- | --- |
| Name | Location | Description |
| Player { } | app.js | Contains  Id: socket id  x, y: position,  screenwidth, screenheight:  screen size,  target |
| global.XXX | global.js | Game status properties |

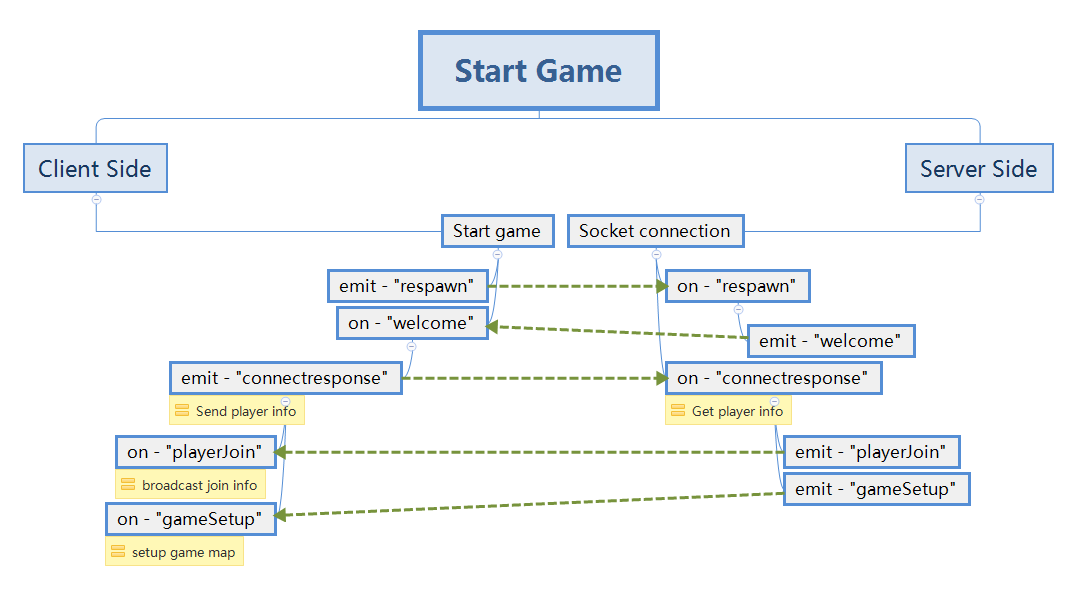
## Server side Data:

|  |  |  |
| --- | --- | --- |
| Name | Location | Description |
| CurrentPlayer {} | Server.js | id: socket.id,  x, y: position,  w, h: use for range in collision check,  cells: cells of one player,  massTotal: massTotal,  hue: color,  type: type,  lastHeartbeat: last active time,  target: { x , y } |
| Config.XXX | config.json |  |

# Basic Interaction

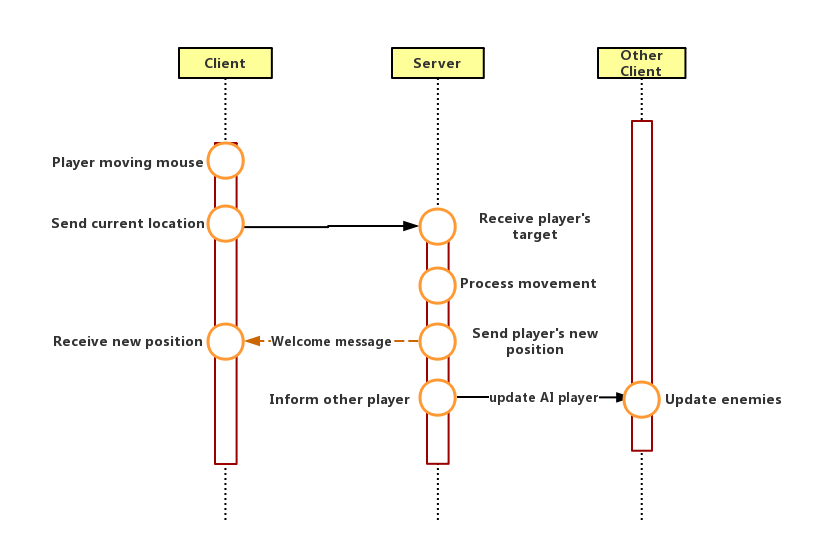
## Authentication



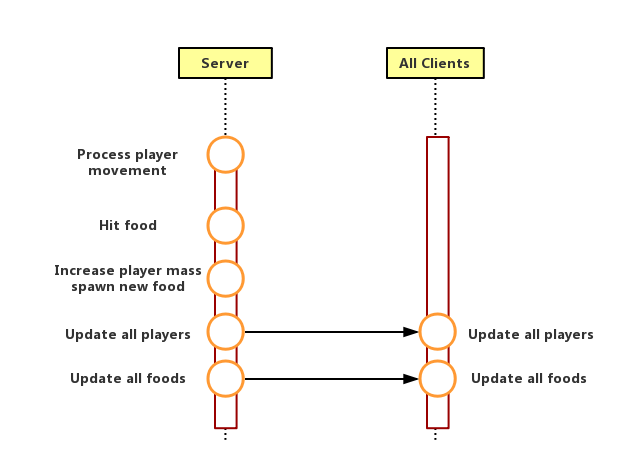


## In Game Communication

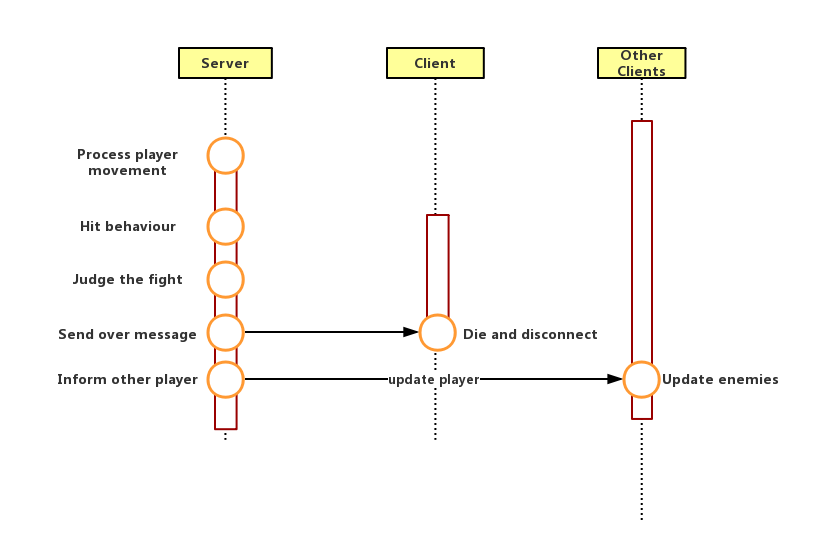
Game logic

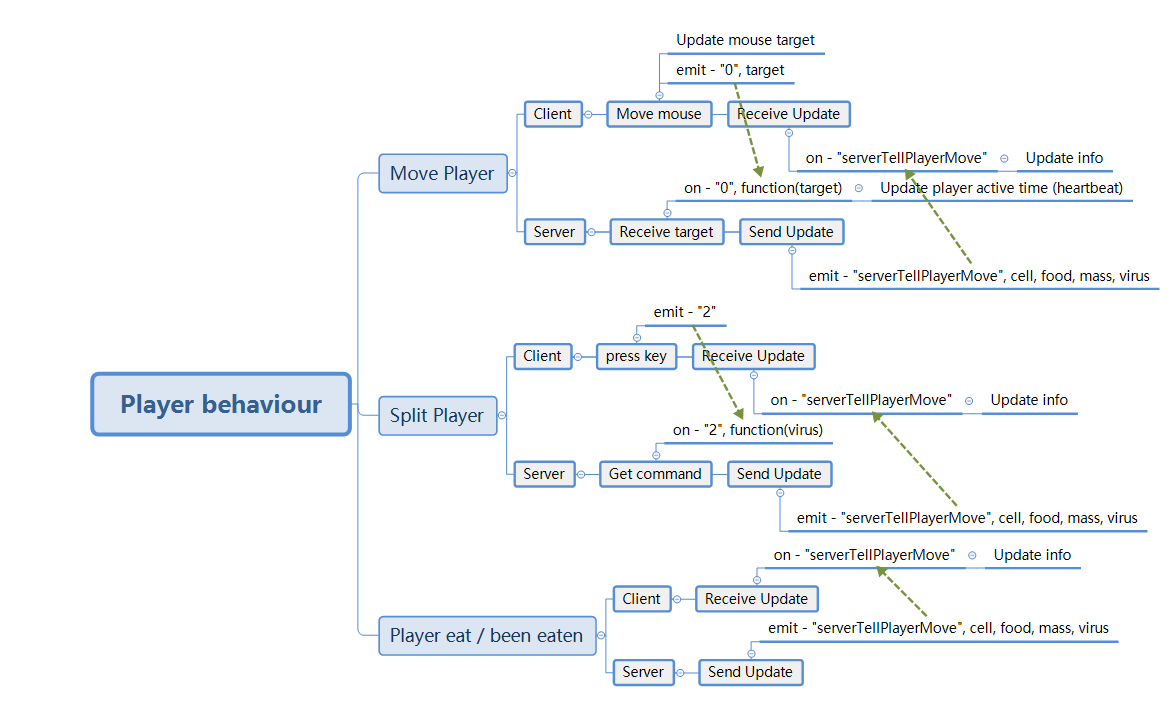


Eating food:

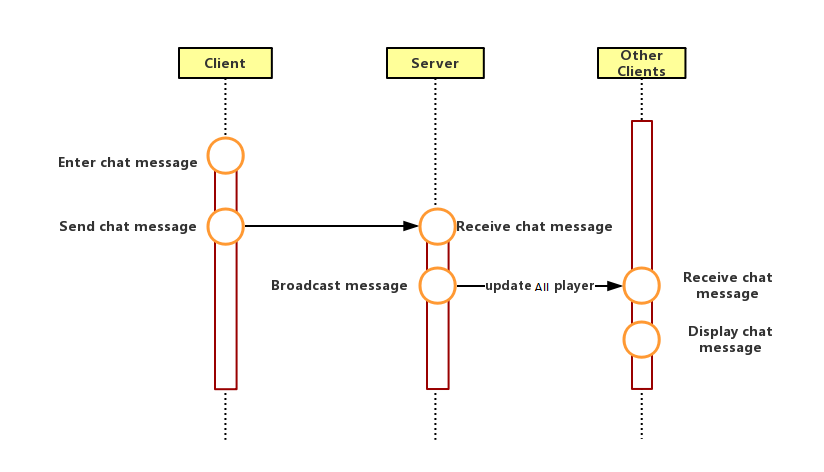


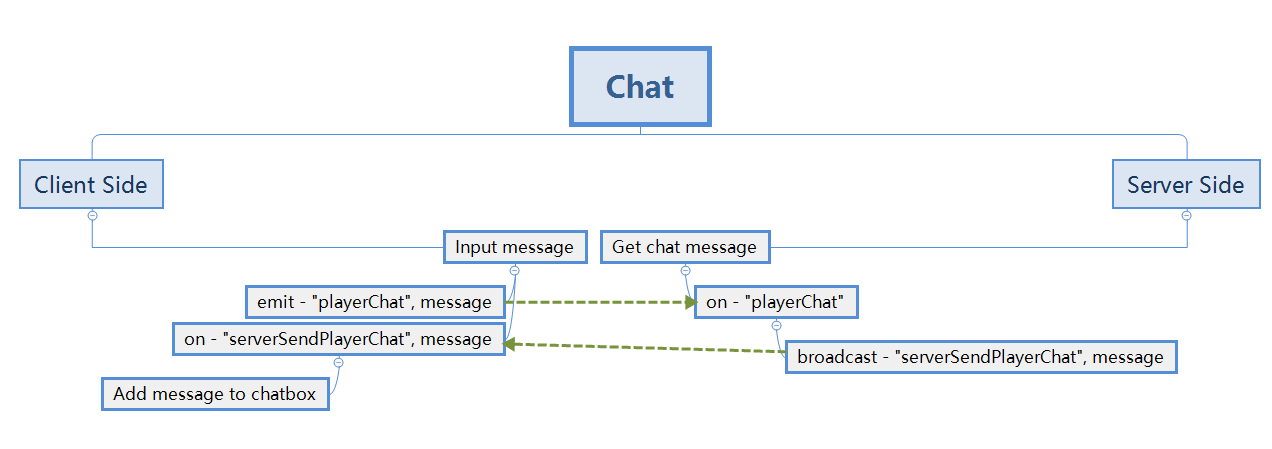
Eating player:





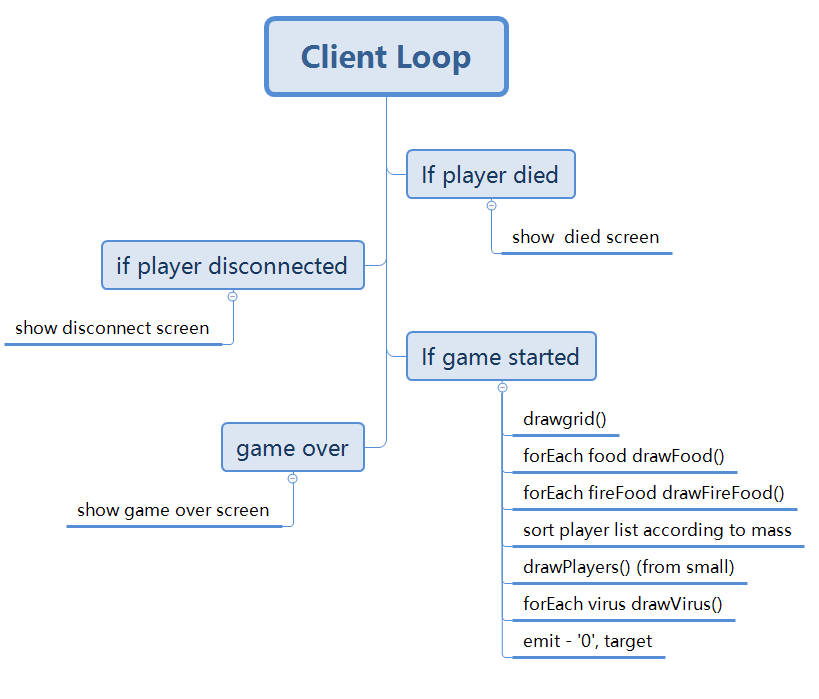
Chat:



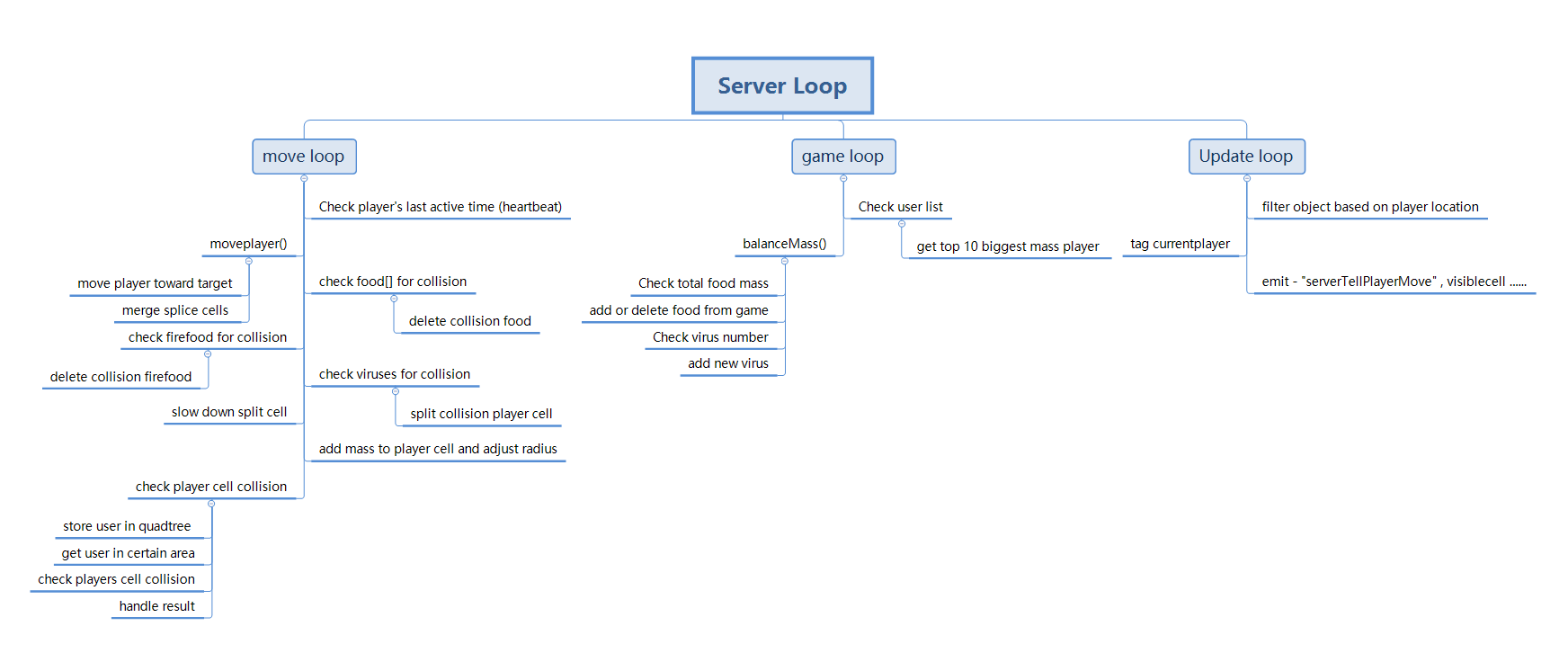


# Major Game process logic

## Client Loop Logic



## Server Loop Logic



# Future Work

Scalability:

Game feature

Game logic optimizing

Latency Compensating

Test Plan:

Add test procedure before building

Launch Plan:

Deploy project