~~You are~~ surviving ~~members of the crew of~~

~~The Survivin~~

You are the surviving crew members of the Star Ship $RAND_NAME. THE Captain is dead! You have ~~decided~~ Agreed to vote on decisions, in the hopes of getting back Home. The ship's solar sails $ARE|Arent functional

```
Backstory & Start Event
Vote → event
While ( ~~event~~ state != WiNconditon)
    event.describe
    event.vote → event
Save ~~Stats~~    }
Compare ~~Stats~~ } Do Highscore
display Score     }

Game = { Backstory, {events}, state, ~~Win condition~~, High Scores }

event = { description, votes, state modifier }
      = Start, ~~WIN~~, ...

Votes = { options : events }, count? }

State = { F → Score }
```

Game/Backstory/*.TXT, *.PNG — "tweeted" in order by Name

/HighScores, state — data

/Events / ...

/Config — define state structure & win condition, etc...

Event	description	state	options : events
Start	Backstory	init	

⋮

eg:

Config: win_condition = state.distance -le 0
State = { distance = 0, time = 0, cargo = 0, fuel ... }

Events: start; BKstory; enter system
enter system; RND

Server of Game:

- universe state - tweeted cocassionally (timed or turns)
- Action Response - update state, Report changes
- player state -

Player Actions:

- Join

planet / Player state:

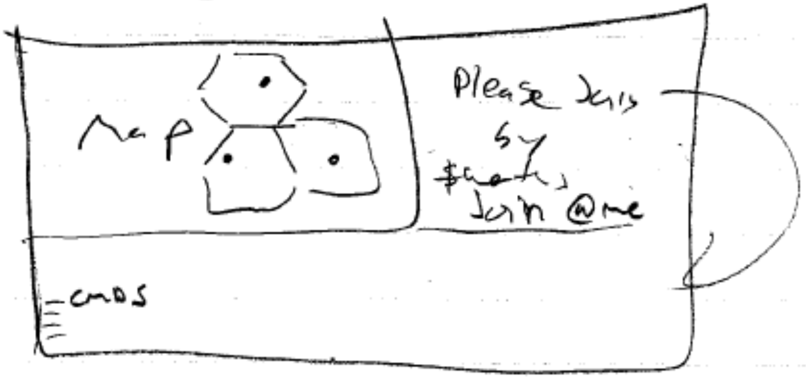
Universe state:

- data & img representation
  - planet & player Locations

MAP
plays
CADS

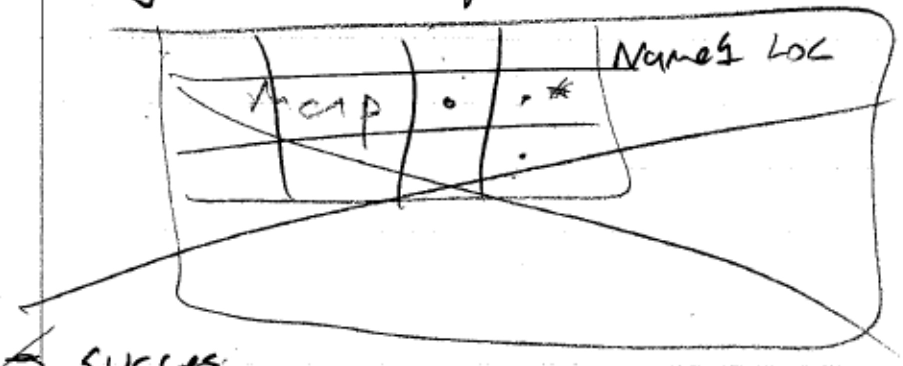
~~@Gamebot Join~~

Server: - Start the game Running
- generate initial universe state
- Advertise avaliability
    tweet game state



Client
Server

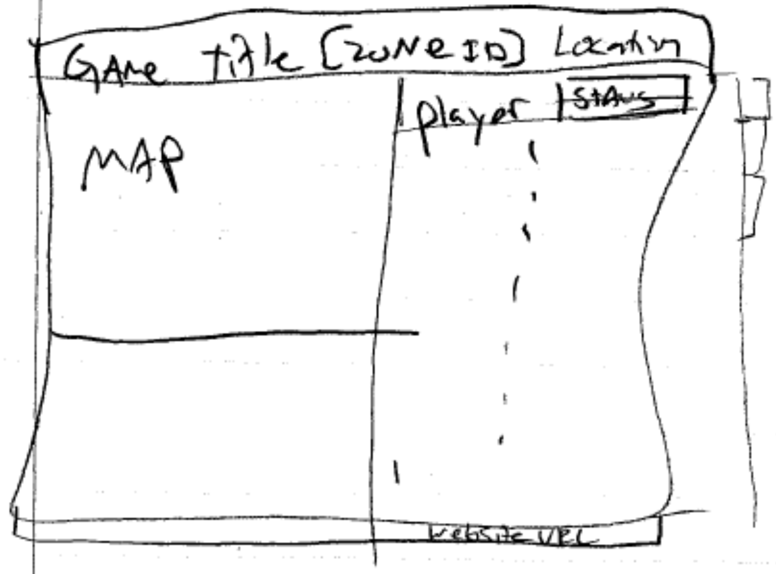
@game Join-Request



- success
    notify All players of Addition player
- Fail
    notify client of failure

welcome to the Game
    Stats

Location



time zones - divide map - increase player
~~[illegible]~~ Limit players per zone
Limit zones per universe

map: planet data
player: Location by color

website displays entire state real time
~~[illegible]~~ tweets only used for input and
ocasional IO

balance ocassions s.t. playable w/o site
if Advanced yet minimal

server
Respond within time limit
aggregate inputs and respond to "all" at once

```
switch(Server.event.queue.pop):

    Timer:
     if((Time.current - Time.stamp)> Conf.Time.wait.max):
          Advertize game state
     else
```