Mobile Computing

App Development Project - Report

Student: Mike Pereira Gonçalves
ID: 0130898735

13.05.2017

Contents

1 Description

This report's focus lies on the actual implementation of the previously created specification about a Star Trek-inspired space combat multiplayer game. For the full understanding of this project, it is advised to start with the specification before reading this report as details and features explained there won't be covered here anymore.

2 Specification Modifications

During the implementation phase, some smaller details about the specification had to be adjusted. Those modifications include aspects of the game which were either not fully thought out or were missing crucial information.

- **Spaceship**:

  The linear damping property was added in the setup object configuration.

  Additionally in the setup message, the spaceship attributes were moved from the separate `ship_config` into the common `objects` section.

  Moreover, the `head` attribute of `ammo` was renamed to `available` and represents an array of torpedo IDs. This change was necessary in order to be able to send the current state of a game session to a client who for some reason got previously disconnected.

  The torpedoes shot by a spaceship are all of the same rectangular size and fly at a constant velocity. In order to reduce the fire rate, the delay between two consecutive torpedoes must be of at least 0.3 seconds.

Type	**Width**	**Height**	**Velovity**	**Shoot delay**	**linearDamping**
Torpedo	30	90	2000	$0.3s$	0

Table 1: Torpedo Properties Table

  Finally, the concept of a spaceship's shield wasn't fully covered. When the spaceship takes damage, it reduces its shield by the same amount. When the shield reaches 0 and thus becomes disabled, the ship can be destroyed by a single subsequent hit. With the exception of spaceships colliding with each other resulting in them being destroyed on the spot regardless of their shields, when the incoming damage is greater than the shield's value, the shield simply becomes disabled without killing the spaceship, for which another damage boost is necessary. This life mechanism can be compared to the one typically used on the Sonic video game franchise.

- **Meteoroid**:

  In order to be able to send the current game state to a client, the setup message does not only include the maximum but also the current health of meteoroids, respectively denoted `hp_max` and `hp`.

Additionally, when destroying big meteoroids, there is a chance of a dilithium appearing underneath. The spawn rate of this happening is denoted `spawn_rate` and is also sent via the setup message.

- **Spacestation**:

  Because of their round shape, the dimensions of spacestations are represented by their radius instead of their width and hight. Additionally, their size has been increased from 150 to 200 radius.

  Moreover, the spacestation status message now also includes whether the station is currently transferring hp to its owner.

- **Move command**:

  Periodically, every client sends a move message to the server, stating his current position, rotation and velocity. Previously, the server would simply broadcast this message to all his clients. However, the number of messages sent from the server would increase exponentially with the number of connected clients. Therefore, all the move commands have been grouped into a single message which is then send to all the clients.

- **Resume command**:

  This message type has been removed as it was sufficient to reuse the already existing `start` message.

- **Item respawn command**:

  This message type has been renamed to `object_respawn` and has the same structure as the object listing in the setup message.

- **Hp/Ammo/Kill commands**:

  These message types were all removed. It is sufficient to send a collision message to the clients who then compute the current life and ammo of the involved objects.

- **Gameover command**:

  The `winner` attribute is not necessary as it can be computed by going through the player list and checking for a `null` value in the `killed_by` property.

- **State Sync command**:

  The `paused` attribute has been removed. The default behaviour of the server is to pause the game whenever a `state_sync` is received, making this attribute redundant.

3 Implementation

3.1 Architecture

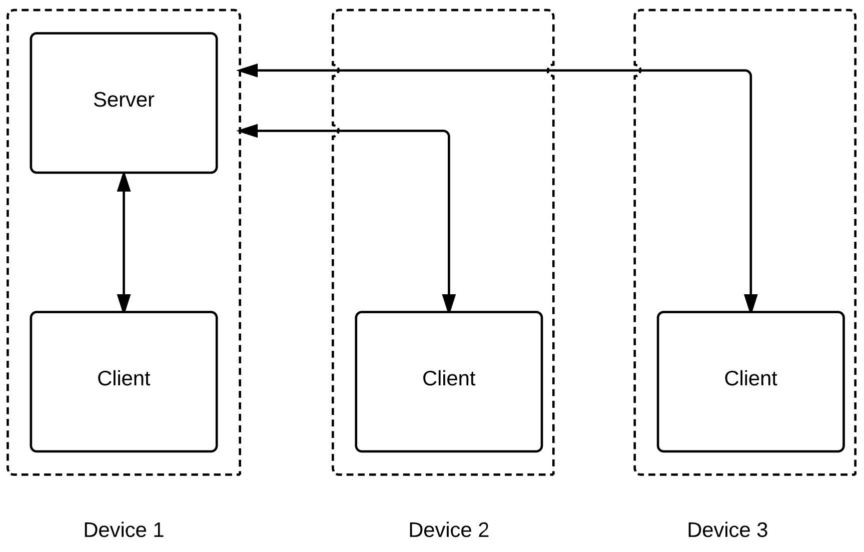
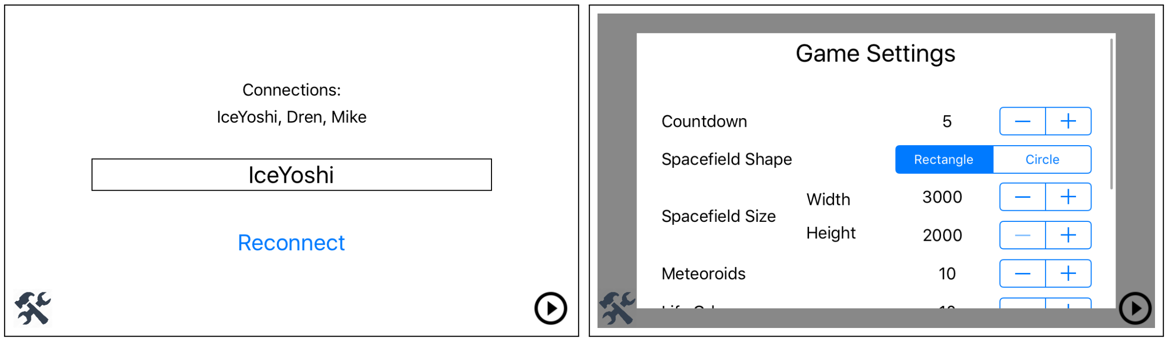


Figure 1: Server-Client architecture

Describe client server

3.2 Message processing

3.3 Screens



Connections:
IceYoshi, Dren, Mike

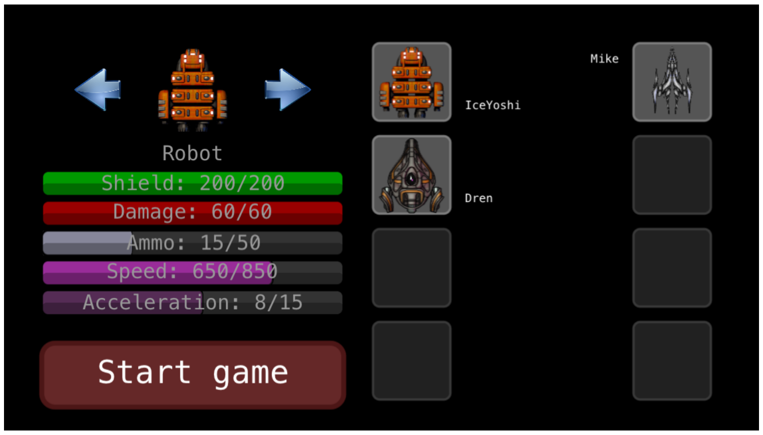
IceYoshi

Game Settings

Countdown		5	
Spacefield Shape		Rectangle	Circle
Spacefield Size	Width	3000	
	Height	2000	
Meteoroids		10	

Figure 2: Lobby screen

When first opening the Application, the user is confronted with the Lobby Screen shown on figure 2. Here, he can type in his nickname and choose whether he wants to be the server (host) or client. When connected to other devices, a list of all participants is shown. The host has additionally the ability to view and change the game settings and to start the ship selection.



Robot
Shield: 200/200
Damage: 60/60
Ammo: 15/50
Speed: 650/850
Acceleration: 8/15

Figure 3: Ship Selection screen

In the Ship Selection Screen,

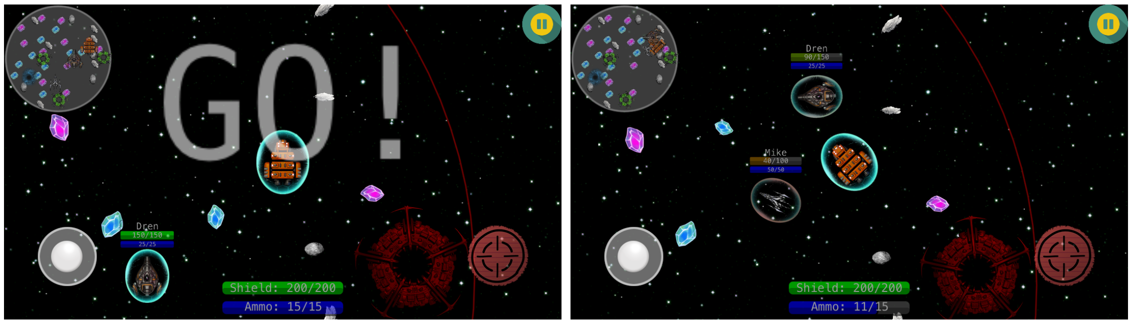
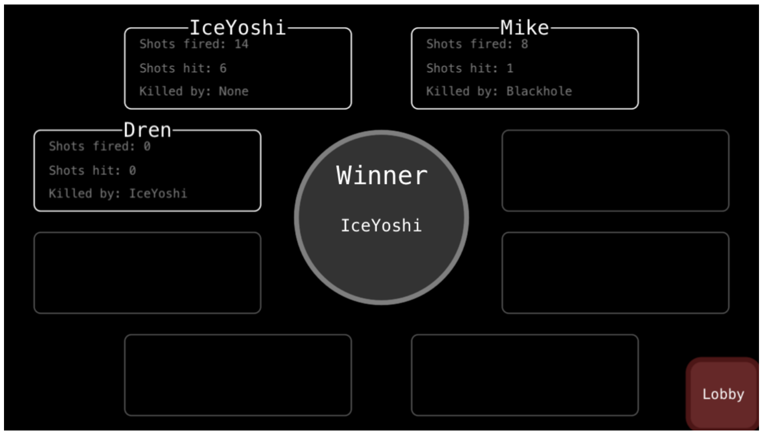


Figure 4: Gameplay screen



IceYoshi
Shots fired: 14
Shots hit: 6
Killed by: None

Mike
Shots fired: 8
Shots hit: 1
Killed by: Blackhole

Dren
Shots fired: 0
Shots hit: 0
Killed by: IceYoshi

Winner
IceYoshi

Figure 5: Gameover screen showing game statistics

Also describe screen transition condition

3.4 Scene Layers

The game scene is composed of multiple layers which all have their role. In the following is a list of all the components per layer, as well as a small description:

- **World layer**: Spaceship, Meteoroid, Life orb, Dilithium, Spacestation, Blackholes

  Includes all game objects that the player can interact with. They all can be accessed by their unique IDs and will be shown on the Minimap.

- **Background layer**: Spacefield border, Starfield

  Includes all environment objects which are purely visual. A parallax scrolling effect is added to the whole layer.

- **Overlay**: Joystick, fire and pause button, Minimap, BarIndicator

  Includes static content that should always stay in the foreground and at the same place.

3.5 Camera

The camera always follows the player's spaceship. However, instead of directly locking on the player, it changes its offset according to the spaceship's velocity in order to be slightly ahead and to show a bigger area in front of the user.

Once the player's spaceship gets destroyed, it is possible to drag the camera using the pan gesture and thus to spectate the remaining match. In this mode, it is possible to click on an adversary's spaceship to make the camera follow the selected target.

3.6 Object Manager

3.7 World generation

probability of big/small meteoroids, CPU slave and master.., how the setup message is created with idCounter

no collision between objects, freerandomposition

server decides stats...

3.8 Countdown

3.8.1 Object respawn

3.9 Minimap

shape of real spacefield, tries to aspect-fill the space given by the overlay

Composed of actual game sprites, actualizes whenever an object moves or gets removed

3.10 BarIndicators

3.11 Blackhole

A Blackhole is a round game object that pulls any nearby spaceship in. When colliding with it, the ship gets damaged and a suck-in animation is shown, before reappearing once again at a predefined spawn position, usually at the center of the universe. Finally, an invulnerability time of 2 seconds is applied on the spaceship, making it possible to avoid collision with another object at the spawned position.

When multiple spaceships enter the Blackhole simultaneously, they simply collide with each other, resulting in a mutual destruction.

The animation duration is based on a given `delay` attribute and is defined in seconds, after which the player regains control over his ship again. The Blackhole also gets temporarily disabled during that time. This makes it possible to use this object as a means of teleportation in order to escape an adversary as he won't be able to directly follow the player due to this short inactivity time.

3.12 Spacestation

Every player (except of CPU enemies) has his individual stationary space station. When the owner spaceship is on top of it, he slowly gets his shields repaired. However, after some regeneration time, the station gets temporarily disabled, thus avoiding someone camping at his station.

During this regeneration time, the player can still be shot at. Additionally, if an enemy touches a foreign station, it instantaneously gets disabled for the same amount of time.

3.13 Controller

The spaceship steering controller is inspired by a real-world joystick. Using touch gestures, it allows the player to precisely move his spaceship around. The distance of the controller head from its local origin indicates the applied thrust value, giving the player even additional movement control over his spaceship. On top of that, the controller comes with a small "dead zone" of 30% where the player can rotate but not apply thrust. This is useful when aligning the spaceship in order to shoot at a target without moving towards it.

3.14 CPU

In the game options, the host is able to choose whether CPU enemies should be included in the game session. Those enemies come with reserved spaceships that cannot be chosen by normal players. As the name indicates, they are completely autonomous and behave as if they were real players: They move towards other players and, once close enough, start to shoot at them.

The underlying implementation is quite simple. The server (host) assigns a virtual controller for each CPU enemy. Like a normal controller, the only 2 parameters that can be changed are angle and thrust. When provided with a list of spaceships with their current position, the virtual controller simply choses to move towards the target that is the closest. In order to make the CPU easier/harder, it is possible to define a shoot rate and thrust limit. For instance, in the default settings, the thrust is limited to 20% and the fire rate at only one torpedo every 3 seconds.

3.15 Pause/Resume

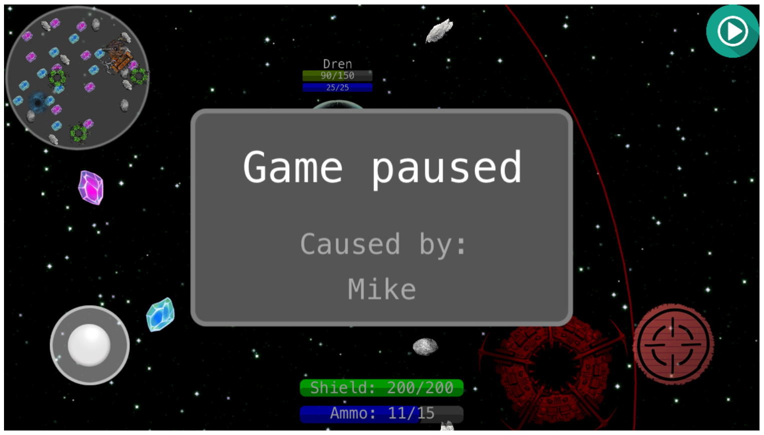


Figure 6: Paused

During gameplay, any player may hit the pause button, resulting in a series of `pause` messages in order to pause every player on that game session. By design, only the host is able to resume the game. While the game is paused, the name of the player that caused that action to happen.

However, the game may also automatically pause when the host notices a connection loss between one of his clients. Again, it is the responsibility of the host to decide when to resume the game, independently of whether the lost client rejoins.

3.16 Movement interpolation

Every client sends his current position, together with his rotation and velocity in form of a message to the server, which collects this data and broadcasts it to all his clients. Per default, those actions are done at a frequency of 10 messages per second.

While it would be sufficient for a client to just update the data of every spaceship upon receiving such a message, the resulting movement would not look smooth, as it would effectively just be teleporting the ships to the new position.

3.17 Connection loss

Describe server states.. Input diagram of allowed messages per state from spec

What exactly is send at what state

When does the server accept clients?

4 Use Case

Select name, choose server, choose game settings

Select name, choose client, change name

Server starts, everyone chooses spaceship

Game starts, countdown, someone disconnects, tries to reconnect, sends name message, name does not have to match, receives sync

Same time, another one tries to join

At the end, shows statistics

5 Appendix

```
{ // Client -> Server
  "type":"name",
  "val":"Mike"
}

{ // Server -> Client
  "type":"name",
  "val":[
    {
      "pid":1,
      "name":"IceYoshi"
    },
    {
      "pid":2,
      "name":"Dren"
    },
    {
      "pid":3,
      "name":"Mike"
    }
  ]
}

{
  "type":"start_ship_selection"
}

```

```
{
  "pid":2,  // pid only when Server -> Client
  "type":"ship_selected",
  "ship_type":"skeleton"
}

{
  "type":"setup",
  "pid":2,  // pid of the client that receives this message
  "countdown":5,
  "space_field":{
    "shape":"circle",
    "r":1500
  },
  "objects":[
    {
      "dmg":50,
      "rot":5.143643,
      "size":{
        "w":93,
        "h":93
      },
      "id":32,
      "hp":87,
      "type":"meteoroid2",
      "hp_max":87,
      "pos":{
        "x":1911,
        "y":1255
      }
    },
    {
      "size":{
        "w":300,
        "h":300
      },
      "hp":750,
      "acc":2,
      "pos":{
        "x":309,
        "y":930
      },
      "speed":150,
      "rot":0,
      "name":"COM",
      "damping":0.2,
      "id":63,
      "ammo":{
        "max":64,
        "available":[
          64
        ],
        "min":64
      },
      "type":"cpu_master",
      "hp_max":750,
      "dmg":80
    },
    {
      "dmg":83,
      "rot":0.3719037,
      "size":{
        "w":76,
        "h":38
      },
      "id":9,
      "hp":142,
      "type":"meteoroid1",
      "hp_max":142,
      "pos":{
        "x":1247,
        "y":786
      }
    },
    {
      "hp_gain":17,
      "id":35,
      "rot":2.557928,
      "type":"life_orb",
      "size":{
        "w":43,
        "h":59
      },
      "pos":{
        "x":2287,
        "y":2611
      }
    },
    {
      "size":{
        "w":108,
        "h":132
      },
      "hp":40,
      "acc":15,
      "pos":{
        "x":2426,
        "y":800
      },
      "speed":800,
      "rot":0,
      "name":"COM",
      "damping":0.8,
      "id":57,
      "ammo":{
        "max":58,
        "available":[
          58
        ],
        "min":58
    },
    "type":"cpu_slave",
    "hp_max":40,
    "dmg":30
  },
  {
    "size":{
      "w":49,
      "h":68
    },
    "id":41,
    "rot":4.743699,
    "type":"dilithium",
    "ammo_gain":13,
    "pos":{
      "x":1456,
      "y":166
    }
  },
  {
    "rot":4.687587,
    "inactive":60,
    "size":{
      "r":200
    },
    "rate":15,
    "owner":3,
    "id":157,
    "active":10,
    "type":"spacestation",
    "pos":{
      "x":1956,
      "y":785
    }
  },
  {
    "size":{
      "w":108,
      "h":132
    },
    "hp":100,
    "acc":15,
    "pos":{
      "x":2063,
      "y":2498
    },
    "speed":850,
    "rot":0,
    "name":"Dren",
    "damping":0.95,
    "id":2,
    "ammo":{
      "max":140,
      "available":[
        103,108,116,105,107,114,97,91,132,120,127,136,111,94,121,124,118,13
```

```
              1,98,123,106,95,104,138,133,119,92,135,137,129,102,130,125,109,
              113,140,99,134,101,117,100,93,96,110,115,128,122,139,112,126
        ],
        "min":91
      },
      "type":"skeleton",
      "hp_max":100,
      "dmg":20
    },
    {
      "size":{
        "w":130,
        "h":158
      },
      "hp":200,
      "acc":8,
      "pos":{
        "x":2388,
        "y":1216
      },
      "speed":650,
      "rot":0,
      "name":"Mike",
      "damping":0.75,
      "id":3,
      "ammo":{
        "max":156,
        "available":[
          154,144,153,146,143,145,152,155,142,150,156,151,148,147,149
        ],
        "min":142
      },
      "type":"robot",
      "hp_max":200,
      "dmg":60
    },
    {
      "delay":5,
      "pos":{
        "x":716,
        "y":1575
      },
      "spawn_pos":{
        "x":1500,
        "y":1500
      },
      "size":{
        "r":129
      },
      "min_range":43,
      "id":44,
      "strength":3.44,
      "type":"blackhole",
      "max_range":387,
      "dmg":34
    },
    {
      "size":{
        "w":108,
        "h":132
      },
      "hp":150,
      "acc":10,
      "pos":{
        "x":1984,
        "y":617
      },
      "speed":700,
      "rot":0,
      "name":"IceYoshi",
      "damping":0.9,
      "id":1,
      "ammo":{
        "max":89,
        "available":[
          71,69,74,67,66,65,83,88,86,73,70,72,79,77,68,84,80,78,76,87,85,89,81,82,75
        ],
        "min":65
      },
      "type":"human",
      "hp_max":150,
      "dmg":30
    }
  ]
}

{
  "type":"start"
}

{ // Move Client -> Server
  "type":"move",
  "pos":{
    "y":1215.968505859375,
    "x":2387.930908203125
  },
  "pid":3,
  "rot":0,
  "vel":{
    "dx":0,
    "dy":0
  },
  "sqn":54
}

{ // Move Server -> Client
  "type":"move",
  "sqn":0,
  "objects":[
    {
      "pid":1,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
      "pos":{
        "x":1984,
        "y":617
      }
    },
    {
      "pid":45,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
      "pos":{
        "x":431.0001,
        "y":2244
      }
    },
    {
      "pid":47,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
      "pos":{
        "x":1257,
        "y":1703
      }
    },
    {
      "pid":49,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
      "pos":{
        "x":2180,
        "y":1632
      }
    },
    {
      "pid":51,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
        "pos":{
          "x":2528,
          "y":1890
        }
      },
      {
        "pid":53,
        "rot":0,
        "vel":{
          "dx":0,
          "dy":0
        },
        "pos":{
          "x":932.0001,
          "y":2323
        }
      },
      {
        "pid":55,
        "rot":0,
        "vel":{
          "dx":0,
          "dy":0
        },
        "pos":{
          "x":1085,
          "y":2177
        }
      },
      {
        "pid":57,
        "rot":0,
        "vel":{
          "dx":0,
          "dy":0
        },
        "pos":{
          "x":2426,
          "y":800
        }
      },
      {
        "pid":59,
        "rot":0,
        "vel":{
          "dx":0,
          "dy":0
        },
        "pos":{
          "x":1064,
          "y":2546
        }
      },
      {
        "pid":61,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
      "pos":{
        "x":666,
        "y":2408
      }
    },
    {
      "pid":63,
      "rot":0,
      "vel":{
        "dx":0,
        "dy":0
      },
      "pos":{
        "x":309,
        "y":930.0001
      }
    }
  ]
}

{
  "type":"station_status",
  "enabled":false,
  "station_id":157,
  "transfer":false
}

{
  "type":"pause",
  "pid":1 // pid only when Server -> Client
}

{
  "type":"state_sync",
  "state":"playing",
  "setup":{
    // Setup with up-to-date object data
  }
}

{
  "type":"object_respawn",
  "object":{
    "dmg":43,
    "rot":4.609109,
    "size":{
      "w":80,
      "h":80
    },
    "id":159,
    "hp":75,
    "type":"meteoroid2",
    "hp_max":75,
    "pos":{
      "x":1789,
      "y":973
    }
  }
}

{
  "type":"collision",
  "id1":5,
  "id2":55
}

{
  "type":"gameover",
  "players":[
    {
      "shots_fired":7,
      "shots_hit":5,
      "killed_by":"Meteoroid",
      "pid":1
    },
    {
      "shots_fired":0,
      "shots_hit":0,
      "killed_by":"COM",
      "pid":2
    },
    {
      "shots_fired":0,
      "shots_hit":0,
      "killed_by":"Blackhole",
      "pid":3
    }
  ]
}

{
  "type":"fire",
  "pos":{
    "x":1331,
    "y":2450
  },
  "pid":61,
  "rot":-1.507627,
  "fid":62
}
```