

Mobile Computing

## App Development Project - Report

Student: Mike Pereira Gonçalves

ID: 0130898735

15.05.2017

# Contents

<b>1</b>	<b>Description</b>	<b>3</b>
<b>2</b>	<b>Specification Modifications</b>	<b>3</b>
<b>3</b>	<b>Implementation</b>	<b>5</b>
3.1	Architecture . . . . .	5
3.2	Object Manager . . . . .	5
3.3	Scene Layers . . . . .	5
3.4	World Generation . . . . .	6
3.5	Object Respawn . . . . .	6
3.6	Screens . . . . .	6
3.7	Camera . . . . .	8
3.8	Controller . . . . .	8
3.8.1	CPU . . . . .	8
3.9	Countdown . . . . .	8
3.10	Minimap . . . . .	9
3.11	BarIndicator . . . . .	9
3.12	Blackhole . . . . .	10
3.13	Spacestation . . . . .	10
3.14	Pause/Resume . . . . .	11
3.15	Connection loss . . . . .	11
3.16	Movement interpolation . . . . .	12
3.17	Sound effects . . . . .	13
<b>4</b>	<b>Use Case</b>	<b>13</b>
<b>5</b>	<b>Appendix</b>	<b>14</b>

# 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	Velocity	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 crystal appearing underneath. The spawn rate of this happening is denoted `spawn_rate` and is computed using this formula:

$$\text{spawn\_rate} = 0.75 * \text{mean}(m)$$

See specification section 2.4 to learn more about how to compute  $\text{mean}(m)$ .

- **Spacestation:**

Because of their round shape, the dimensions of spacestations are represented by their radius instead of their width and height. 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 sent to all the clients. See section 5 for the concrete structure of the JSON message.

- **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

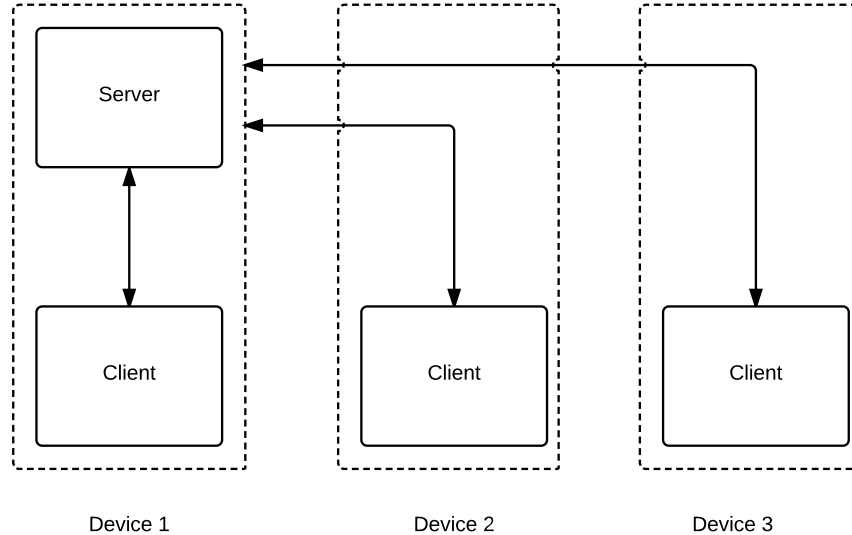


Figure 1: Server-Client architecture

Describe client server

### 3.2 Object Manager

### 3.3 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.

### 3.4 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...

setup creation, maybe as subsubsection

### 3.5 Object Respawn

### 3.6 Screens

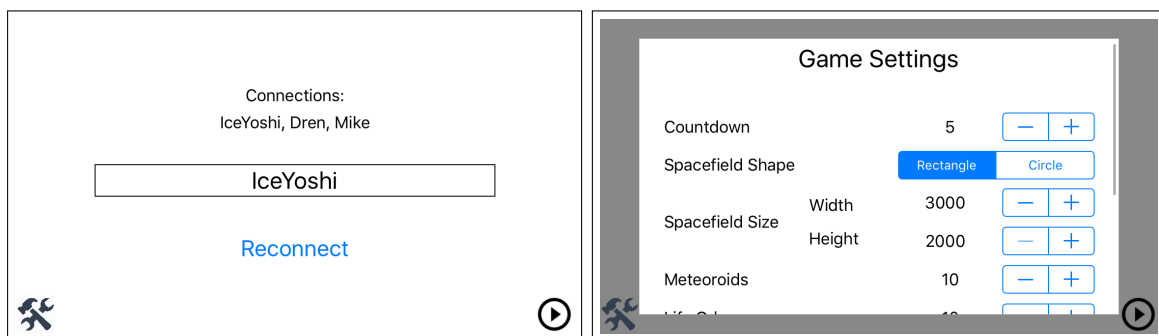


Figure 2: Lobby screen

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.

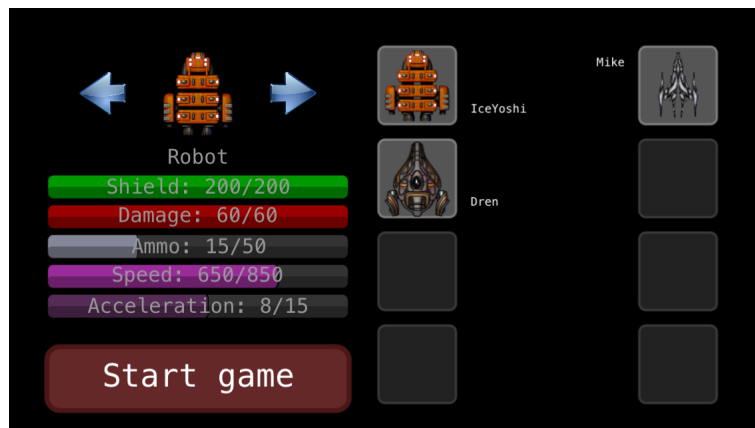


Figure 3: Ship Selection screen

In the Ship Selection screen at figure 3, the player can select between 3 different spaceship types, each with their strengths and weaknesses. On the right side of the screen, the selection choice from other players is being shown and updated in real-time. When everyone's ready, the host can generate the world and start the game by clicking on the 'Start game' button.

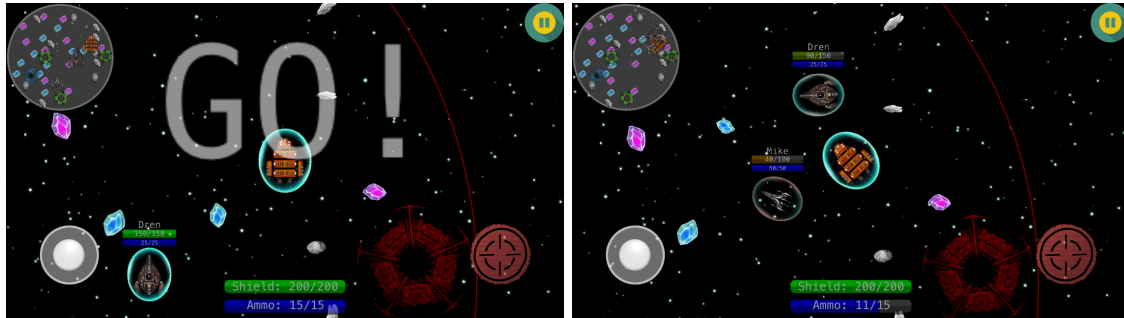


Figure 4: Gameplay screen

After the game scene has loaded, a countdown will start on every device like depicted on figure 4. The game begins when the host's counter reaches 0 (labeled "GO!"), after which a start message is sent to all the clients.

Here, the player can move around his spaceship, shoot torpedoes and pause the game. On the top-left side is a Minimap with the updated placement of all game objects. The game session ends either when no player spaceship remains or one remains with no CPU enemies left. When this condition is met, the host sends to all his clients a gameover message, resulting in a screen transition.

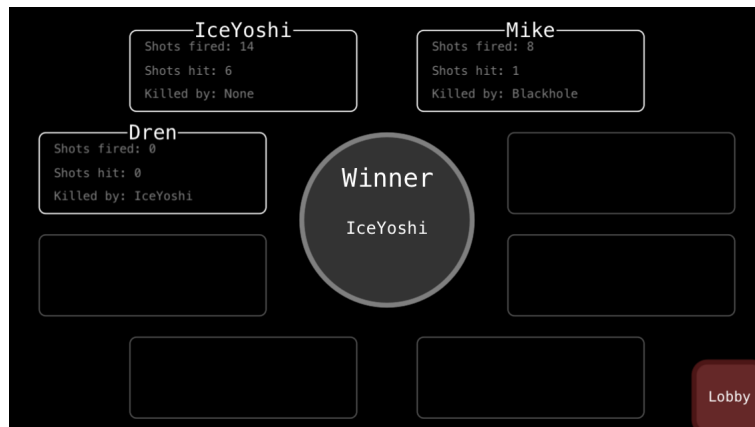


Figure 5: Gameover screen showing game statistics

Finally, in figure 5, the winner (if any) of the game session is shown, as well as game statistics per player like torpedo shot and hit count. With a click on the 'Lobby' button, the player enters the Lobby screen and the journey repeats itself.

### 3.7 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.

By performing a pinch gesture, it is possible to zoom the view point in and out. In order to avoid accidental pinch gesture recognition while simultaneously controlling the joystick and firing torpedoes, it is only possible to perform said action if at least two touch points are not on top of any overlay content.

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.

### 3.8 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.

#### 3.8.1 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 chooses 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.

### 3.9 Countdown

When transitioning from the Ship Selection to the gameplay screen, there is a countdown which primarily serves two purposes:

- It lets the player time to get familiar with the environment and to plan his first actions.



- It adds a way to synchronize the game start on all devices, regardless of the needed loading time. In fact, once the host's countdown reaches 0, he sends a start message to all his clients, whose countdown are purely visual and stop at 1, waiting for said message to arrive. In the other case, if a client's countdown lags behind and receives a start message, he simply skips the remaining count and jumps directly to 0.

The final "GO!" label can be seen in figure 4 on the left side.

### 3.10 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

### 3.11 BarIndicator

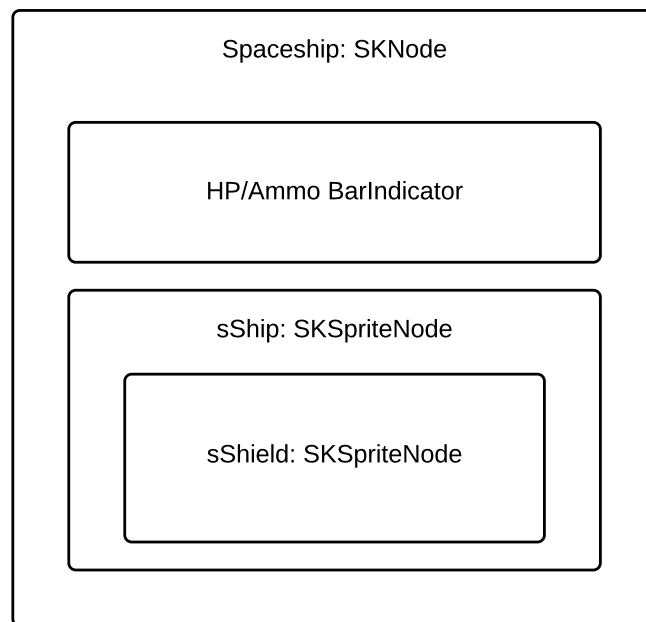


Figure 6: Spaceship Node nesting

A BarIndicator is a visual representation of a numeric value that ranges between a minimum and maximum bounds. It is used in the ship selection screen in order to show the characteristics of the individual spaceship types. During gameplay, they show the current health and ammo values of spaceships and, once damaged, the health of meteoroids.

When initializing, it is possible to give it 2 colors. As the value decreases, there will be a gradual transition between one color to the other.

The player's BarIndicators are static at the bottom center of the screen and remain there for the whole duration of the game. The adversaries' BarIndicators are drawn on top of their spaceship and thus move with the sprite node.

One technical challenge consists in making the BarIndicator move with the target node, but not let it rotate with him. In figure 6, we can see how the individual objects are nested with each other. For instance, when moving the main **Spaceship** class node, all its children will also move along with it. Same goes for the rotation.

The proposed solution is to override the **zRotation** getter and setter of the main class and to propagate those changes to the **sShip** sprite node, thus making the BarIndicator immune to the applied rotation. This solution does however not work for rotations coming from a **SKAction** or produced by physic computations as they seem to completely bypass getter and setter. This can be resolved by introducing public functions inside the **Spaceship** class who then perform the desired actions over the **sShip** sprite.

### 3.12 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 a 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.

### 3.13 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.

### 3.14 Pause/Resume



Figure 7: 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. More about connection loss in the following section 3.15.

### 3.15 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?

### 3.16 Movement interpolation

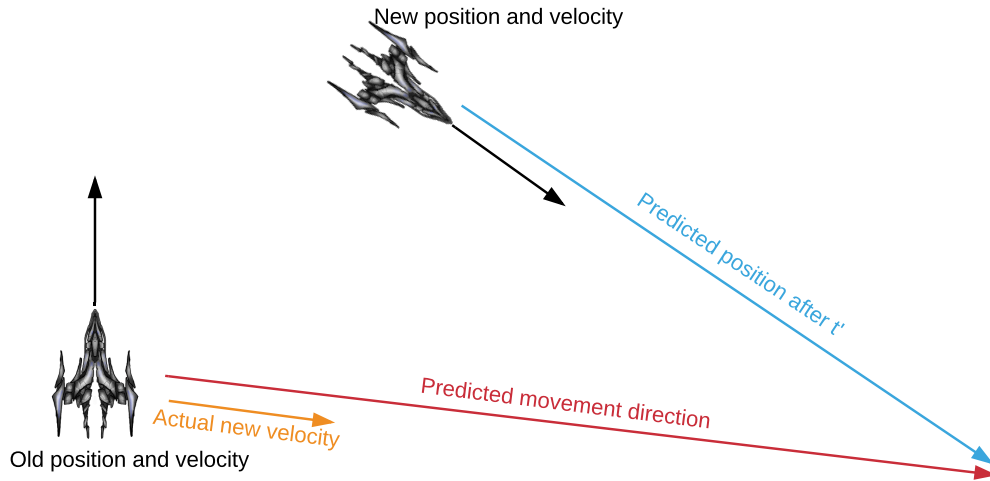


Figure 8: Movement interpolation

Every client sends his current position, together with his rotation and velocity in form of a move 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.

In order to improve the movement animation, let us consider figure 8. The client proceeds as follows:

1. First, by knowing the time delay in seconds  $\Delta t$  between two consecutive move messages and the frames per second rate  $fps$  at which the game is running locally, we can compute the mean amount of game frames that happen between two move messages like this:

$$frames = \Delta t * fps$$

For our typical case, we have:  $\Delta t = 0.1s$  and  $fps = 60$ , thus:  $frames = 6$ .

2. Now we take the newly received velocity vector and multiply its length by  $frames$ , keeping his heading direction. If we add this computed vector to the new position, we get the predicted position for the spaceship after  $\Delta t$  (shown in blue).
3. Next, we subtract the old position from the new predicted position to get the predicted movement vector (indicated in red)
4. Finally, we divide this created vector by  $frames$  and change the spaceship's current velocity to the resulting vector.

As for the rotation of the spaceship, a `SKAction` is executed for the same duration  $\Delta t$ , resulting in a smooth rotation speed.

### 3.17 Sound effects

The Application comes with a main looping background song track, as well as multiple sound effects for common actions like shooting and hitting torpedoes, clicking on buttons, countdown timer, gameover transition, getting pulled from a Blackhole, collecting Dilithium Crystals and Life orbs, recharging at a Spacestation as well as destroying meteoroids and spaceships.

## 4 Use Case

Consider the following scenario:

Dren and Mike want to play SpaceWars together and start the Application on their personal devices. Mike chooses the nickname 'IceYoshi' and hosts the game. Dren on his turn writes his own name as a nickname and joints the host, who then responds with his own name. However, Dren changes his mind and quickly alters his nickname to 'Drega'. This change is signalized via a message to the host who then updates his lobby view.

Mike adjusts the game settings and then clicks on the ship selection button, resulting in a transition to said screen on both devices. Mike chooses 'Human' as his spaceship race, and Dren decides to go with 'Skeleton'.

After hitting the start button, the game scene loads, and after waiting for the countdown to reach 0, the fun can start. Unfortunately, only a few seconds into the game, Dren receives an important phone call that he cannot dismiss. The Application goes in the background and automatically sends a pause message to the host.

After finishing the call, Dren tries to reopen the game, but closes it accidentally. Therefore he opens a fresh instance of the Application and, as expected, is once again at the Lobby screen.

In the meantime, Joe opens SpaceWars on his device. He chooses to impersonate Dren by calling himself 'Drega' and tries to join the already existing game session. However, the host notices that the device ID does not match any player in that game session and thus refuses this connection.

On the other hand, Dren enters his nickname and hits join. The host recognizes the device ID and responds with a `state_sync` message which contains the setup with all game objects and their most up-to-date position. Dren's device loads the game scene which is exactly at the same state as where he had left off. All what's left to do is for the host to click on resume and they are once again playing on the same game session.

It so happens that both players are sucked in a Blackhole at the same time, resulting in a mutual collision and thus destruction. The game ends by showing the gameover screen with the game statistics. It's a draw.

Mike suggests to start a new game session, and this time wait for Joe to also join in.

## 5 Appendix

Here is an updated sample of all possible messages that can be send to- and from the server.

```
1 { // Client -> Server
2   "type":"name",
3   "val":"Mike"
4 }
5
6 { // Server -> Client
7   "type":"name",
8   "val":[
9     {
10      "pid":1,
11      "name":"IceYoshi"
12    },
13    {
14      "pid":2,
15      "name":"Dren"
16    },
17    {
18      "pid":3,
19      "name":"Mike"
20    }
21  ]
22 }
23
24 {
25   "type":"start_ship_selection"
26 }
27
28 {
29   "pid":2, // pid only when Server -> Client
30   "type":"ship_selected",
31   "ship_type":"skeleton"
32 }
33
34 {
35   "type":"setup",
36   "pid":2, // pid of the client that receives this message
37   "countdown":5,
38   "space_field":{
39     "shape":"circle",
40     "r":1500
41   },
42   "objects":[
43     {
44       "dmg":50,
45       "rot":5.143643,
46       "size":{
47         "w":93,
48         "h":93
49       },
50       "id":32,
51       "hp":87,
52       "type":"meteoroid2",
```

```

53     "hp_max":87,
54     "pos":{
55         "x":1911,
56         "y":1255
57     }
58 },
59 {
60     "size":{
61         "w":300,
62         "h":300
63     },
64     "hp":750,
65     "acc":2,
66     "pos":{
67         "x":309,
68         "y":930
69     },
70     "speed":150,
71     "rot":0,
72     "name":"COM",
73     "damping":0.2,
74     "id":63,
75     "ammo":{
76         "max":64,
77         "available":[
78             64
79         ],
80         "min":64
81     },
82     "type":"cpu_master",
83     "hp_max":750,
84     "dmg":80
85 },
86 {
87     "dmg":83,
88     "rot":0.3719037,
89     "size":{
90         "w":76,
91         "h":38
92     },
93     "id":9,
94     "hp":142,
95     "type":"meteoroid1",
96     "hp_max":142,
97     "pos":{
98         "x":1247,
99         "y":786
100     }
101 },
102 {
103     "hp_gain":17,
104     "id":35,
105     "rot":2.557928,
106     "type":"life_orb",
107     "size":{

```

```

108         "w":43,
109         "h":59
110     },
111     "pos":{
112         "x":2287,
113         "y":2611
114     }
115 },
116 {
117     "size":{
118         "w":108,
119         "h":132
120     },
121     "hp":40,
122     "acc":15,
123     "pos":{
124         "x":2426,
125         "y":800
126     },
127     "speed":800,
128     "rot":0,
129     "name":"COM",
130     "damping":0.8,
131     "id":57,
132     "ammo":{
133         "max":58,
134         "available":[
135             58
136         ],
137         "min":58
138     },
139     "type":"cpu_slave",
140     "hp_max":40,
141     "dmg":30
142 },
143 {
144     "size":{
145         "w":49,
146         "h":68
147     },
148     "id":41,
149     "rot":4.743699,
150     "type":"dilithium",
151     "ammo_gain":13,
152     "pos":{
153         "x":1456,
154         "y":166
155     }
156 },
157 {
158     "rot":4.687587,
159     "inactive":60,
160     "size":{
161         "r":200
162     },

```



```

163     "rate":15,
164     "owner":3,
165     "id":157,
166     "active":10,
167     "type":"spacestation",
168     "pos":{
169         "x":1956,
170         "y":785
171     }
172 },
173 {
174     "size":{
175         "w":108,
176         "h":132
177     },
178     "hp":100,
179     "acc":15,
180     "pos":{
181         "x":2063,
182         "y":2498
183     },
184     "speed":850,
185     "rot":0,
186     "name":"Dren",
187     "damping":0.95,
188     "id":2,
189     "ammo":{
190         "max":140,
191         "available":[
192             103,108,116,105,107,114,97,91,132,120,127,136,111,94,121,124,118,13
193             1,98,123,106,95,104,138,133,119,92,135,137,129,102,130,125,109,
194             113,140,99,134,101,117,100,93,96,110,115,128,122,139,112,126
195         ],
196         "min":91
197     },
198     "type":"skeleton",
199     "hp_max":100,
200     "dmg":20
201 },
202 {
203     "size":{
204         "w":130,
205         "h":158
206     },
207     "hp":200,
208     "acc":8,
209     "pos":{
210         "x":2388,
211         "y":1216
212     },
213     "speed":650,
214     "rot":0,
215     "name":"Mike",
216     "damping":0.75,
217     "id":3,

```

```

216     "ammo":{
217         "max":156,
218         "available":[
219             154,144,153,146,143,145,152,155,142,150,156,151,148,147,149
220         ],
221         "min":142
222     },
223     "type":"robot",
224     "hp_max":200,
225     "dmg":60
226 },
227 {
228     "delay":5,
229     "pos":{
230         "x":716,
231         "y":1575
232     },
233     "spawn_pos":{
234         "x":1500,
235         "y":1500
236     },
237     "size":{
238         "r":129
239     },
240     "min_range":43,
241     "id":44,
242     "strength":3.44,
243     "type":"blackhole",
244     "max_range":387,
245     "dmg":34
246 },
247 {
248     "size":{
249         "w":108,
250         "h":132
251     },
252     "hp":150,
253     "acc":10,
254     "pos":{
255         "x":1984,
256         "y":617
257     },
258     "speed":700,
259     "rot":0,
260     "name":"IceYoshi",
261     "damping":0.9,
262     "id":1,
263     "ammo":{
264         "max":89,
265         "available":[
266             71,69,74,67,66,65,83,88,86,73,70,72,79,77,68,84,80,78,76,87,85,89,8
267             1,82,75
268         ],
269         "min":65
270     },

```

```

270         "type": "human",
271         "hp_max": 150,
272         "dmg": 30
273     }
274 ]
275 }
276
277 {
278     "type": "start"
279 }
280
281 { // Move Client -> Server
282     "type": "move",
283     "pos": {
284         "y": 1215.968505859375,
285         "x": 2387.930908203125
286     },
287     "pid": 3,
288     "rot": 0,
289     "vel": {
290         "dx": 0,
291         "dy": 0
292     },
293     "sqn": 54
294 }
295
296 { // Move Server -> Client
297     "type": "move",
298     "sqn": 0,
299     "objects": [
300         {
301             "pid": 1,
302             "rot": 0,
303             "vel": {
304                 "dx": 0,
305                 "dy": 0
306             },
307             "pos": {
308                 "x": 1984,
309                 "y": 617
310             }
311         },
312         {
313             "pid": 45,
314             "rot": 0,
315             "vel": {
316                 "dx": 0,
317                 "dy": 0
318             },
319             "pos": {
320                 "x": 431.0001,
321                 "y": 2244
322             }
323         },
324     ]

```

```

325     "pid":47,
326     "rot":0,
327     "vel":{
328         "dx":0,
329         "dy":0
330     },
331     "pos":{
332         "x":1257,
333         "y":1703
334     }
335 },
336 {
337     "pid":49,
338     "rot":0,
339     "vel":{
340         "dx":0,
341         "dy":0
342     },
343     "pos":{
344         "x":2180,
345         "y":1632
346     }
347 },
348 {
349     "pid":51,
350     "rot":0,
351     "vel":{
352         "dx":0,
353         "dy":0
354     },
355     "pos":{
356         "x":2528,
357         "y":1890
358     }
359 },
360 {
361     "pid":53,
362     "rot":0,
363     "vel":{
364         "dx":0,
365         "dy":0
366     },
367     "pos":{
368         "x":932.0001,
369         "y":2323
370     }
371 },
372 {
373     "pid":55,
374     "rot":0,
375     "vel":{
376         "dx":0,
377         "dy":0
378     },
379     "pos":{

```

```

380         "x":1085,
381         "y":2177
382     }
383 },
384 {
385     "pid":57,
386     "rot":0,
387     "vel":{
388         "dx":0,
389         "dy":0
390     },
391     "pos":{
392         "x":2426,
393         "y":800
394     }
395 },
396 {
397     "pid":59,
398     "rot":0,
399     "vel":{
400         "dx":0,
401         "dy":0
402     },
403     "pos":{
404         "x":1064,
405         "y":2546
406     }
407 },
408 {
409     "pid":61,
410     "rot":0,
411     "vel":{
412         "dx":0,
413         "dy":0
414     },
415     "pos":{
416         "x":666,
417         "y":2408
418     }
419 },
420 {
421     "pid":63,
422     "rot":0,
423     "vel":{
424         "dx":0,
425         "dy":0
426     },
427     "pos":{
428         "x":309,
429         "y":930.0001
430     }
431 }
432 ]
433 }
434

```

```

435 {
436   "type": "station_status",
437   "enabled": false,
438   "station_id": 157,
439   "transfer": false
440 }
441
442 {
443   "type": "pause",
444   "pid": 1 // pid only when Server -> Client
445 }
446
447 {
448   "type": "state_sync",
449   "state": "playing",
450   "setup": {
451     // Setup with up-to-date object data
452   }
453 }
454
455 {
456   "type": "object_respawn",
457   "object": {
458     "dmg": 43,
459     "rot": 4.609109,
460     "size": {
461       "w": 80,
462       "h": 80
463     },
464     "id": 159,
465     "hp": 75,
466     "type": "meteoroid2",
467     "hp_max": 75,
468     "pos": {
469       "x": 1789,
470       "y": 973
471     }
472   }
473 }
474
475 {
476   "type": "collision",
477   "id1": 5,
478   "id2": 55
479 }
480
481 {
482   "type": "gameover",
483   "players": [
484     {
485       "shots_fired": 7,
486       "shots_hit": 5,
487       "killed_by": "Meteoroid",
488       "pid": 1
489     },

```

```

490     {
491         "shots_fired":0,
492         "shots_hit":0,
493         "killed_by":"COM",
494         "pid":2
495     },
496     {
497         "shots_fired":0,
498         "shots_hit":0,
499         "killed_by":"Blackhole",
500         "pid":3
501     }
502 ]
503 }
504
505 {
506     "type":"fire",
507     "pos":{
508         "x":1331,
509         "y":2450
510     },
511     "pid":61,
512     "rot":-1.507627,
513     "fid":62
514 }

```