Capture the flag: Multi-agent system simulation

Developed by:

- José Guerra up201706421
- Luis Ramos up201704243
- Martim Silva up201705205

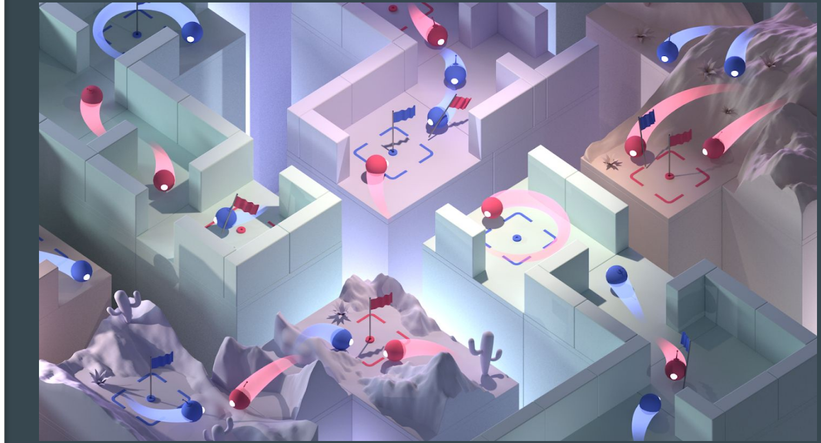
Intro

Cooperative, multiplayer action gamemode, revolving around large-scale territory battles over map control zones, inspired by Battlefield and DeepMind similar modes.

➔ At the start, both teams ...
Have the same number of tickets, which denotes the number of times a team's player can respawn.

➔ To win, one of the teams must ...
Reduce the enemy's tickets to zero or have more tickets than the enemy by the end of the round.

➔ To reduce the enemy's ticket you need to capture the zones!



Description: Image from DeepMind blog post

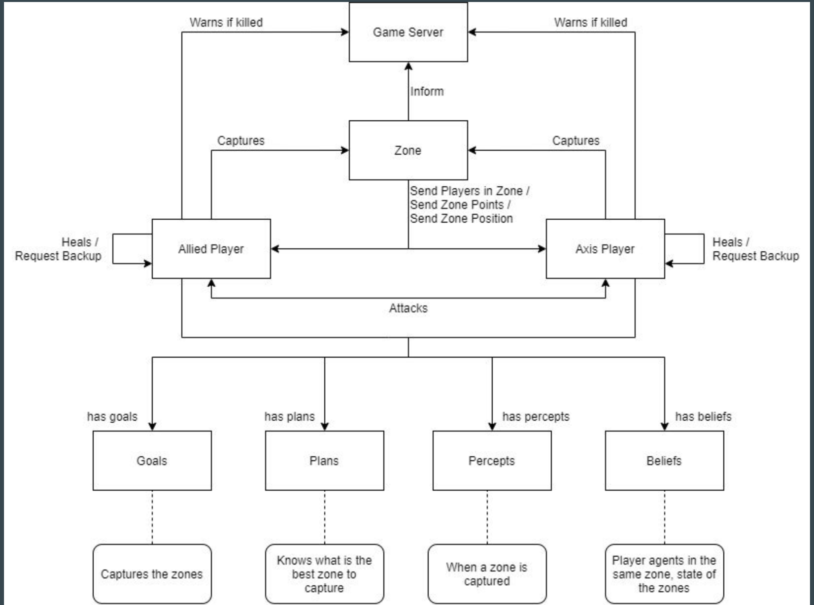
Data analysis problem description

- ➔ After developing the first project using JADE, it became clear the need to gather large portions of data and analyze it to ensure our simulation was **accurate** and **balanced**.

- ➔ On provisional analysis, we noticed that the **Medic** class was too good since it gather points for both defensive and offensive actions. We also noticed that the **Sniper** class was OP in terms of offensive output.

- ➔ It became vital to understand which **parameters to tweak** in order to get more **balance** between the available classes.

- ➔ We gathered a few key **independent variables capable of impacting the outcome** of each game match.



Description: Agents schema and interactions

Agent Strategies

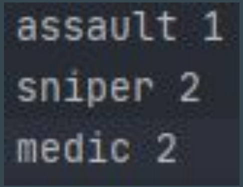
Since each class is unique in their attributes (health, attacking factor, velocity, perks), the best strategy is to get a good team configuration in order to obtain a higher chance of winning a round.

All player agents derive their strategy though the value of a **utility function, that will vary for each player** and takes into account a lot of parameters, such as (but not only):

- ➔ The distance between each zone
- ➔ The number of agents in their current zone
- ➔ The change in delta zone points for each tick
- ➔ The number of backup requests for each zone
- ➔ The team controlling each zone

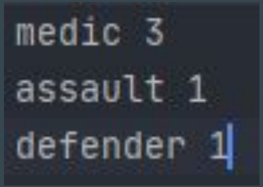
A player stays in their current zone if the current zone is the zone with the best utility value among all others. Otherwise, the player moves to the zone with the best utility value.

If the player stays in the same zone, he can perform a variety of actions such as request for backup, inform allies of current health, attack an enemy, heal the ally with least health (if it has this perk)



```
assault 1
sniper 2
medic 2
```

VS



```
medic 3
assault 1
defender 1
```

Independent Variables

Teams compositions

- There are **unlimited** team composition possibilities.
  They largely **influence** the round's **outcome** given that each class has their perks which need to be combined between different classes wisely.

- **Independent variable(s)** *ALLIED_TEAM, AXIS_TEAM*

Attack factor

- **Tweaking** each **class base attacking factor** will have a big impact for each class offensive performance.

- **Independent variable(s)** *MEDIC_ATK_FACTOR, ASSAULT_ATK_FACTOR, SNIPER_ATK_FACTOR, DEFENDER_ATK_FACTOR*

Health points

- It's possible to **update** each **class base health**.
- More health points allow players to withstand more damage.

- **Independent variable(s)** *MEDIC_HEALTH, ASSAULT_HEALTH, SNIPER_HEALTH***,** *DEFENDER_HEALTH*

Velocity

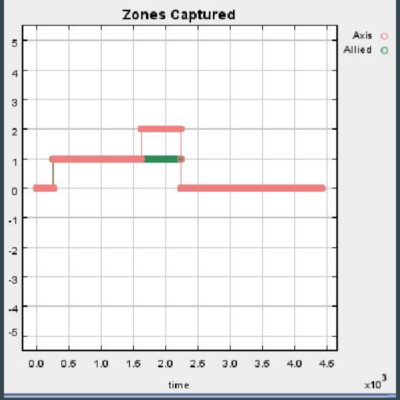
- It's possible to **change** each **class base movement speed.**
- This makes for faster movements between zones.

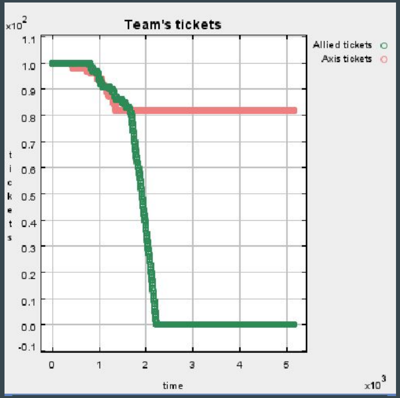
- **Independent variable(s)** *MEDIC_VELOCITY, ASSAULT_VELOCITY, SNIPER_VELOCITY***,** *DEFENDER_VELOCITY*

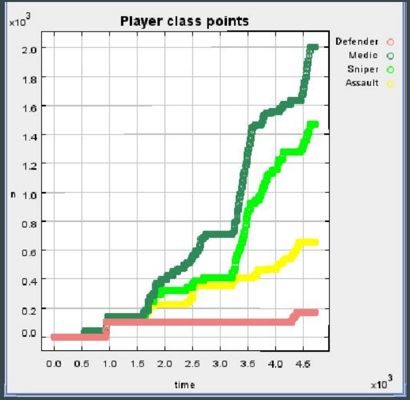
Note: We also have other independent variables, that don't affect the game nearly as much, such as the **game time**, **speed factor**, **initial team tickets** and the **map configuration**.

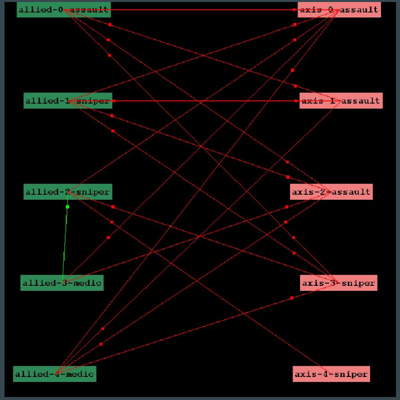
Dependent Variables

- ➔ Winning team
- ➔ Average of points per player-class
- ➔ Number of team tickets through time
- ➔ Number of controlled zones through time
- ➔ Interactions (attacking & healing) between agents





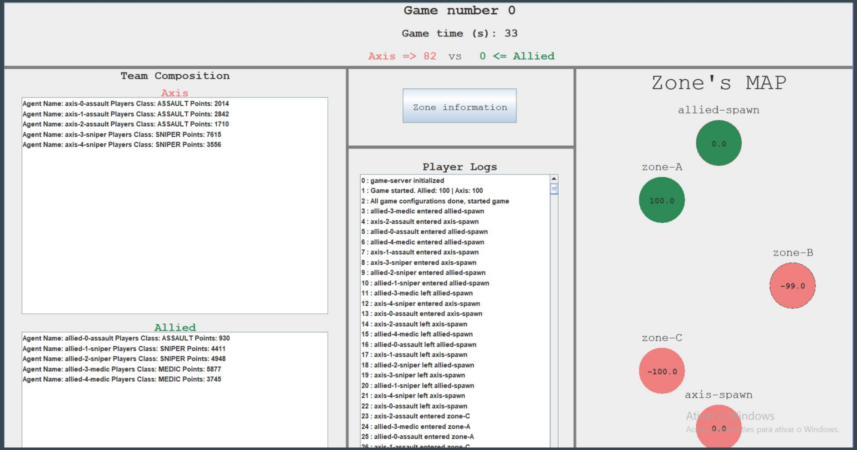




Space Interaction Specification

The space in which the agents interact are the different zones on the map. Each zone has a different position and a player can only communicate with other players on his current zone. To capture a zone, players from one of the teams need to stay there continuously and outnumber the players from the enemy team in that zone for some time.

Moving between zones take some time depending on the distance between the zones and makes the player inactive, ie, the player cannot communicate and interact.



Game number 0

Game time (s): 33

Axis => 82 vs 0 <= Allied

Team Composition

Axis

```
Agent Name: axis-0-assault Players Class: ASSAULT Points: 2014
Agent Name: axis-1-assault Players Class: ASSAULT Points: 2842
Agent Name: axis-2-assault Players Class: ASSAULT Points: 1710
Agent Name: axis-3-sniper Players Class: SNIPER Points: 7615
Agent Name: axis-4-sniper Players Class: SNIPER Points: 3556
```

Allied

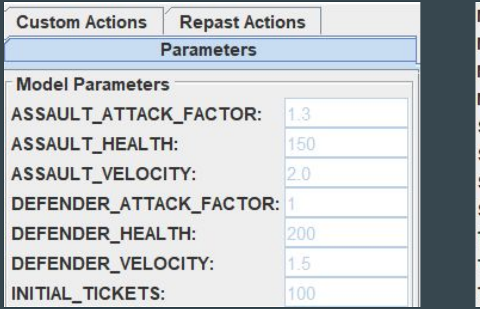
```
Agent Name: allied-0-assault Players Class: ASSAULT Points: 930
Agent Name: allied-1-sniper Players Class: SNIPER Points: 4411
Agent Name: allied-2-sniper Players Class: SNIPER Points: 4948
Agent Name: allied-3-medic Players Class: MEDIC Points: 5877
Agent Name: allied-4-medic Players Class: MEDIC Points: 3745
```

Player Logs

```
0 : game-server initialized
1 : Game started. Allied: 100 | Axis: 100
2 : All game configurations done, started game
3 : allied-3-medic entered allied-spawn
4 : axis-2-assault entered axis-spawn
5 : allied-0-assault entered allied-spawn
6 : allied-4-medic entered allied-spawn
7 : axis-1-assault entered axis-spawn
8 : axis-3-sniper entered axis-spawn
9 : allied-2-sniper entered allied-spawn
10 : allied-1-sniper entered allied-spawn
11 : allied-3-medic left allied-spawn
12 : axis-4-sniper entered axis-spawn
13 : axis-0-assault entered axis-spawn
14 : axis-2-assault left axis-spawn
15 : allied-4-medic left allied-spawn
16 : allied-0-assault left allied-spawn
17 : axis-1-assault left axis-spawn
18 : allied-2-sniper left allied-spawn
19 : axis-3-sniper left axis-spawn
20 : allied-1-sniper left allied-spawn
21 : axis-4-sniper left axis-spawn
22 : axis-0-assault left axis-spawn
23 : axis-2-assault entered zone-C
24 : allied-3-medic entered zone-A
25 : allied-0-assault entered zone-A
26 : axis-1-assault entered zone-C
```

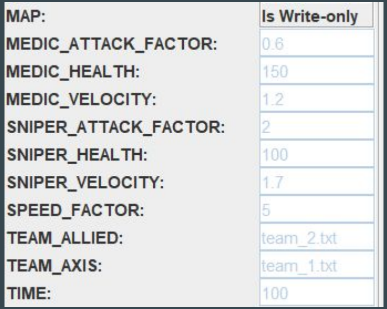
Simulation Visualization

Our application has a complex and well detailed GUI, where the user can configure the parameters of the game before starting it.
When the game starts the user is presented with multiple windows. One of them shows multiple informations about the game like the game time, the zone map, current tickets, etc...
Another window shows the past games score.
There are also multiple windows with graphs like the ones shown on the previous slide.



Model Parameters

ASSAULT_ATTACK_FACTOR:	1.3
ASSAULT_HEALTH:	150
ASSAULT_VELOCITY:	2.0
DEFENDER_ATTACK_FACTOR:	1
DEFENDER_HEALTH:	200
DEFENDER_VELOCITY:	1.5
INITIAL_TICKETS:	100



MAP:	Is Write-only
MEDIC_ATTACK_FACTOR:	0.6
MEDIC_HEALTH:	150
MEDIC_VELOCITY:	1.2
SNIPER_ATTACK_FACTOR:	2
SNIPER_HEALTH:	100
SNIPER_VELOCITY:	1.7
SPEED_FACTOR:	5
TEAM_ALLIED:	team_2.txt
TEAM_AXIS:	team_1.txt
TIME:	100

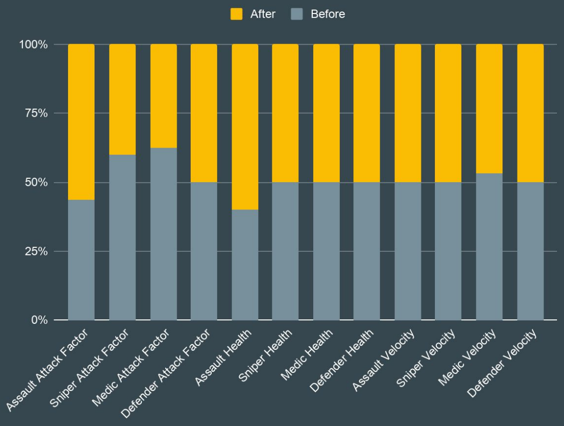
Experiments and result analysis

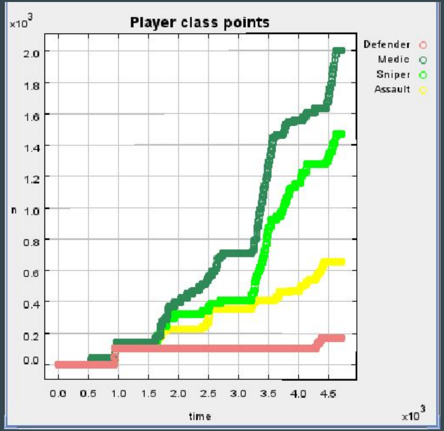
Player classes:

- The first analysis and experiments are done for each individual player class. We concluded, on the first project, that the Defender class was the weakest and that the Medic was the one with more individual points.
- We tried to fix this by tweaking the independent variables to make each class more balanced. For that, we changed each class parameters, such as velocity, attack damage, health points, actions timeouts, etc.

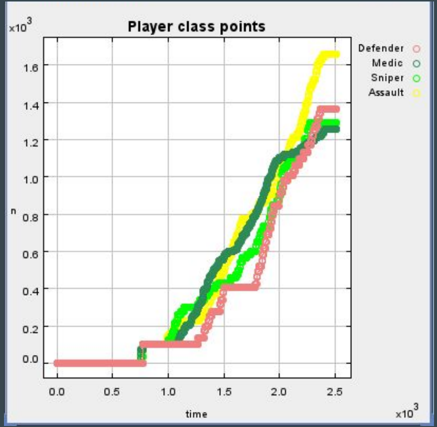
	Healing	
	Before	After
Minimum	50	25
Maximum	100	50
Timeout (ms)	5000	8000

Parameters tweak before and after (independent variables)





Description: Before tweak



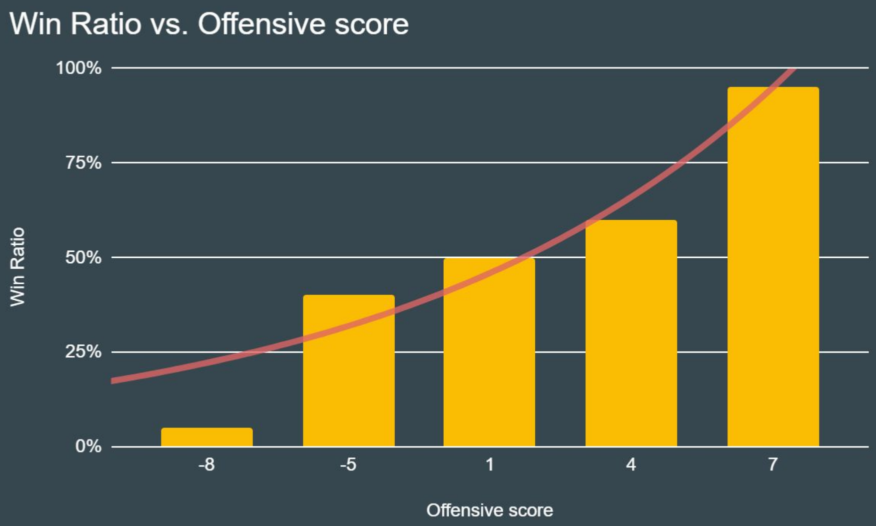
Description: After tweak

Experiments and result analysis

Team configuration:

To analyse the team configuration on our game, we decided to test different combinations of teams against each other. When doing this, we discovered that the number of offensive units on a team might have a direct correlation with the number of wins a team had, still more testing was necessary.

The testing done was simple, 5 teams, each with a different combination of player classes. Each team had a combined offensive score that came from the sum of the individual offensive rating of each unit of a certain player class (Assault: 2, Sniper: 1, Medic: -1, Defender: -2). After putting up all the teams against each other, in batch mode, the graph on the right was produced and it clearly shows that the more offensive score a team has the more likely it is to win.



		Team setup			
		Assault	Sniper	Medic	Defender
	Offensive score	2	1	-1	-2
Team1	1	2	-	3	-
Team2	-5	1	-	1	3
Team3	-8	-	-	2	3
Team4	4	1	2	2	-
Team5	7	2	3	-	-

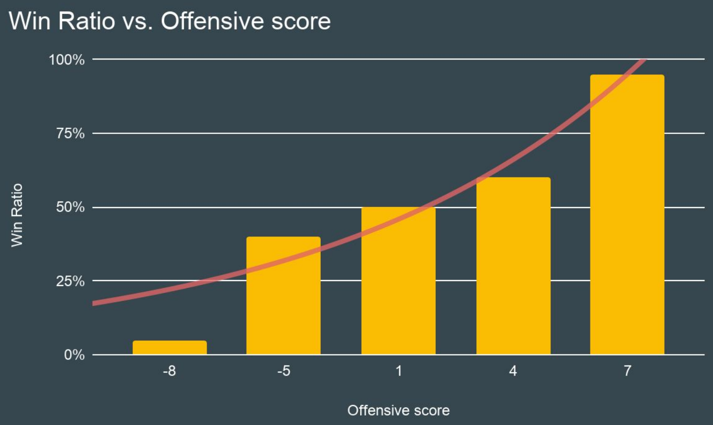
Conclusion

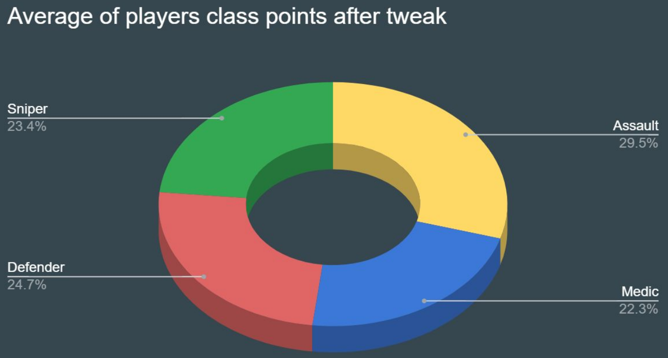
By using SaJas and Repast for the agent based simulation, we were able to extract **useful information** and **tweak many parameters.** This was done via plots, network graphs, our swing GUI and collecting and extrapolating data.

As stated before, we noticed, on the first project, that **some classes were disproportionately better than others**, that lead to **unbalanced games**. This was addressed after **tweaking the independent variables** though Repast parameters menu.

Also, we were able to further study how the **utility function** and **player strategies** behaved **for different team configuration**, to try to understand the best possible configuration using the different classes settings and perks. For that we used Repast batch mode.

In summary, using Repast and SaJas we were able to adjust our initial models and fix many of the games deficiencies by data extraction and analysis and also testing our game in a proportion much larger and more detailed than with Jade.





Observations

We have done some testing and executed, in **batch mode**, different games (can be seen on the execution examples slides) having reached some observations:

- ➔ Since all players use the same function to determine the best zone based on its utility, what can happen sometimes, is that, all players from the same team can make the **same decision** and go all together to the same zone.

- ➔ Since the MovingBehaviour is done using a WakerBehaviour this makes the agent **unaware of the changing zones situations** while its moving.

- ➔ Since we introduce some randomness, to make our game as immersive as possible, we had to **test each strategy multiple times** to make sure we extrapolated the correct data information.

- ➔ When the speed factor is very high there tends to be a **build up of unprocessed messages** from the players to other players. This can be attributed to respast being **single threaded**.

Instructions

We have created scripts for both Linux and Windows Powershell terminals. There are scripts to cleanup, compile and start the game. To make make the game as entertaining as possible you can configure the game settings to experience different outcomes after starting the app using the repast gui. NOTE: Java version need to be lower or equal to 14.0. Java 15.0 gives out errors when working with repast 3.1.

➔ Cleanup & Compile

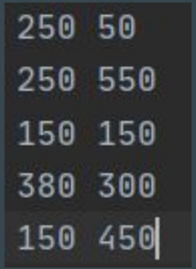
>> scripts/cleanup | >> scripts/compile

➔ Start

>> scripts/start

Text files

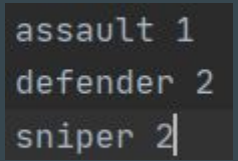
The map and team configuration can be changed by changing the name of the file in the repast gui parameters. To change the number of zones and their positions add a file in /zones/. The same can be done to each team composition located in /players/ where you can edit the number of players for each team and their respective classes (assault, medic, sniper and defender).



```
250 50
250 550
150 150
380 300
150 450
```

/zones/map1.txt

Each line corresponds to a different zone. First 2 zones are allied and axis spawn locations. The two values in each line correspond to the position of the zone (the x and y values respectively)



```
assault 1
defender 2
sniper 2
```

/players/team_1.txt

Each line corresponds to the number of players of a certain player class.
The file gives a team’s entire composition

Execution Example 1

To run this example:

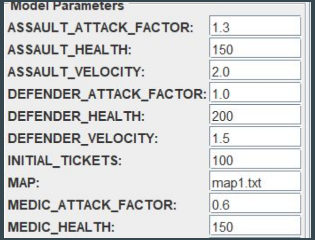
```
>>scripts/start
```

In this example, the allied team has 3 medic, 1 assault, 1 defender and the axis team had 3 sniper, 1 assault, 1 medic.

The axis team ended up winning 64% of the games against the allied. With this we can conclude some things with some degree of certainty.

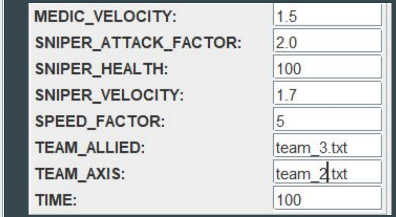
The axis team compositions is proved to be superior but just by a little when stacked against the composition of the allied team. This can be explained, by the fact that there are more offensive units in the axis team. Even though the medic is a unit which can attack and defend at the same time, there are some penalties associated with this class, such as, less velocity in travel time and less damage perform on certain attacks. The defender is proved also to be a less effective unit because its job is to protect conquered zones, which can lead to contribute less to the overall game.

With this, we can analyze that having to many defensive units is not recommended.

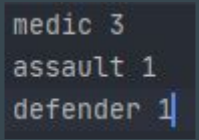


Model Parameters

ASSAULT_ATTACK_FACTOR:	1.3
ASSAULT_HEALTH:	150
ASSAULT_VELOCITY:	2.0
DEFENDER_ATTACK_FACTOR:	1.0
DEFENDER_HEALTH:	200
DEFENDER_VELOCITY:	1.5
INITIAL_TICKETS:	100
MAP:	map1.txt
MEDIC_ATTACK_FACTOR:	0.6
MEDIC_HEALTH:	150

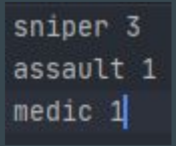


MEDIC_VELOCITY:	1.5
SNIPER_ATTACK_FACTOR:	2.0
SNIPER_HEALTH:	100
SNIPER_VELOCITY:	1.7
SPEED_FACTOR:	5
TEAM_ALLIED:	team_3.txt
TEAM_AXIS:	team_2.txt
TIME:	100



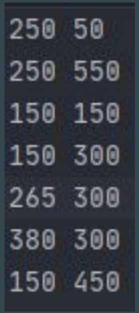
```
medic 3
assault 1
defender 1
```

Allied Team Composition



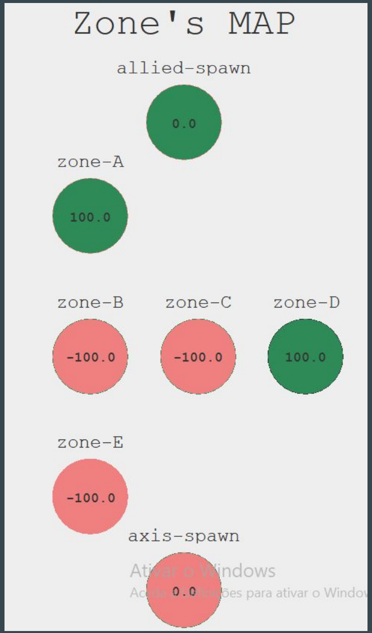
```
sniper 3
assault 1
medic 1
```

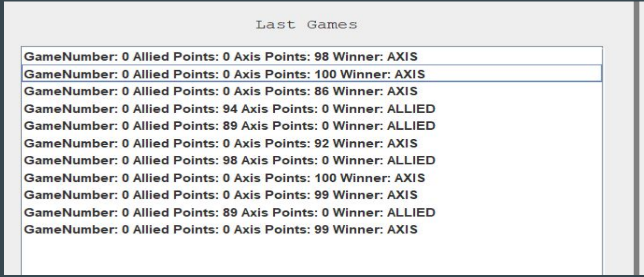
Axis Team Composition



```
250 50
250 550
150 150
150 300
265 300
380 300
150 450
```

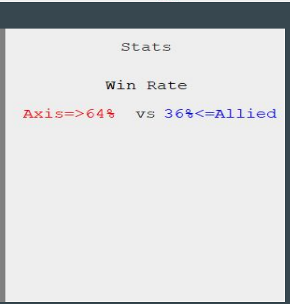
Zones position





Last Games

```
GameNumber: 0 Allied Points: 0 Axis Points: 98 Winner: AXIS
GameNumber: 0 Allied Points: 0 Axis Points: 100 Winner: AXIS
GameNumber: 0 Allied Points: 0 Axis Points: 86 Winner: AXIS
GameNumber: 0 Allied Points: 94 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 89 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 0 Axis Points: 92 Winner: AXIS
GameNumber: 0 Allied Points: 98 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 0 Axis Points: 100 Winner: AXIS
GameNumber: 0 Allied Points: 0 Axis Points: 99 Winner: AXIS
GameNumber: 0 Allied Points: 89 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 0 Axis Points: 99 Winner: AXIS
```



Stats

Win Rate

Axis=>64% vs 36%<=Allied

Win Ratio

Execution Example 2

To run this example:

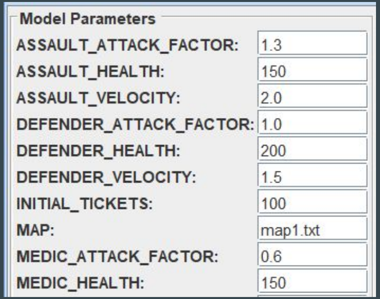
```
>>scripts/start
```

In this example, the allied team has 3 assault, 2 sniper and the axis team has 2 sniper, 1 assault, 1 medic.

The allied team ended up winning 65% of the games against the axis. With this we can conclude some things with some degree of certainty.

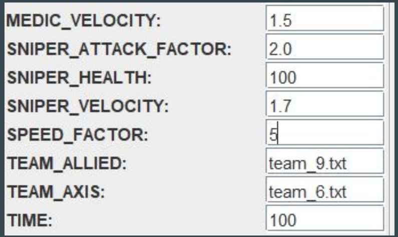
The allied team compositions is proved to be superior, when stacked against the composition of the allied team. This can be explained, by the fact that there are more attack units in the allied team.

Even though the axis team is more balanced in term of defensive and offensive units, the low number of assault units is proved to be an handicap. Since the soldier unit has an advantage in terms of velocity, it can reach any zone quicker than any other unit and this proves to be a great advantage.

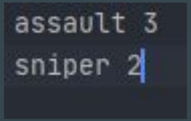


Model Parameters

ASSAULT_ATTACK_FACTOR:	1.3
ASSAULT_HEALTH:	150
ASSAULT_VELOCITY:	2.0
DEFENDER_ATTACK_FACTOR:	1.0
DEFENDER_HEALTH:	200
DEFENDER_VELOCITY:	1.5
INITIAL_TICKETS:	100
MAP:	map1.txt
MEDIC_ATTACK_FACTOR:	0.6
MEDIC_HEALTH:	150

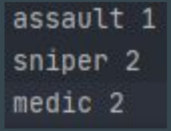


MEDIC_VELOCITY:	1.5
SNIPER_ATTACK_FACTOR:	2.0
SNIPER_HEALTH:	100
SNIPER_VELOCITY:	1.7
SPEED_FACTOR:	5
TEAM_ALLIED:	team_9.txt
TEAM_AXIS:	team_6.txt
TIME:	100



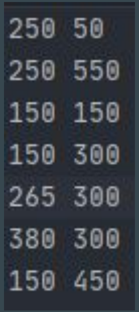
```
assault 3
sniper 2
```

Allied Team Composition



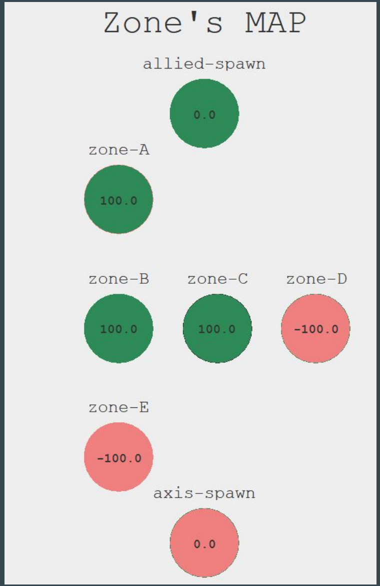
```
assault 1
sniper 2
medic 2
```

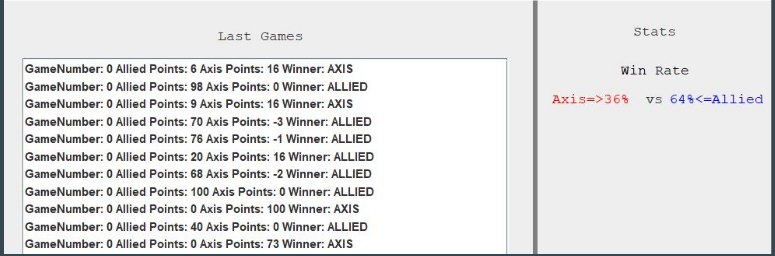
Axis Team Composition



```
250 50
250 550
150 150
150 300
265 300
380 300
150 450
```

Zones position





Last Games

```
GameNumber: 0 Allied Points: 6 Axis Points: 16 Winner: AXIS
GameNumber: 0 Allied Points: 98 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 9 Axis Points: 16 Winner: AXIS
GameNumber: 0 Allied Points: 70 Axis Points: -3 Winner: ALLIED
GameNumber: 0 Allied Points: 76 Axis Points: -1 Winner: ALLIED
GameNumber: 0 Allied Points: 20 Axis Points: 16 Winner: ALLIED
GameNumber: 0 Allied Points: 68 Axis Points: -2 Winner: ALLIED
GameNumber: 0 Allied Points: 100 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 0 Axis Points: 100 Winner: AXIS
GameNumber: 0 Allied Points: 40 Axis Points: 0 Winner: ALLIED
GameNumber: 0 Allied Points: 0 Axis Points: 73 Winner: AXIS
```

Stats

Win Rate

Axis=>36% vs 64%<=Allied

Win Ratio

GUI

After running the start script and the app is running, the following repast gui will appear. To start the game the user needs only to press to Start button.

Before starting the game it is also possible to change some of the independent variables before starting the game.

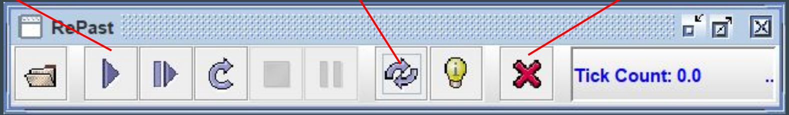
When the game has started it is possible to pause and continue the game and to restart it as well.

To close the app the user may only need to press the red cross button.

Start game Button

Restart the game

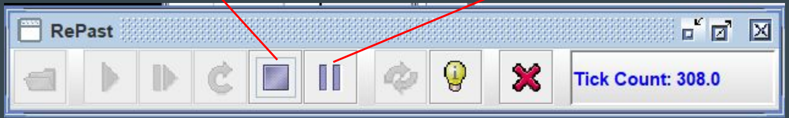
Close the app button



Start of the app

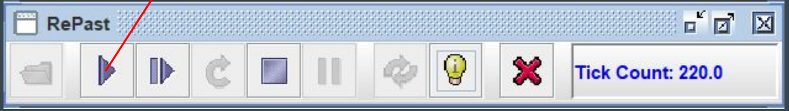
Stop the game

Pause the game



Game running

Continue the game



Game paused

GUI (cont)

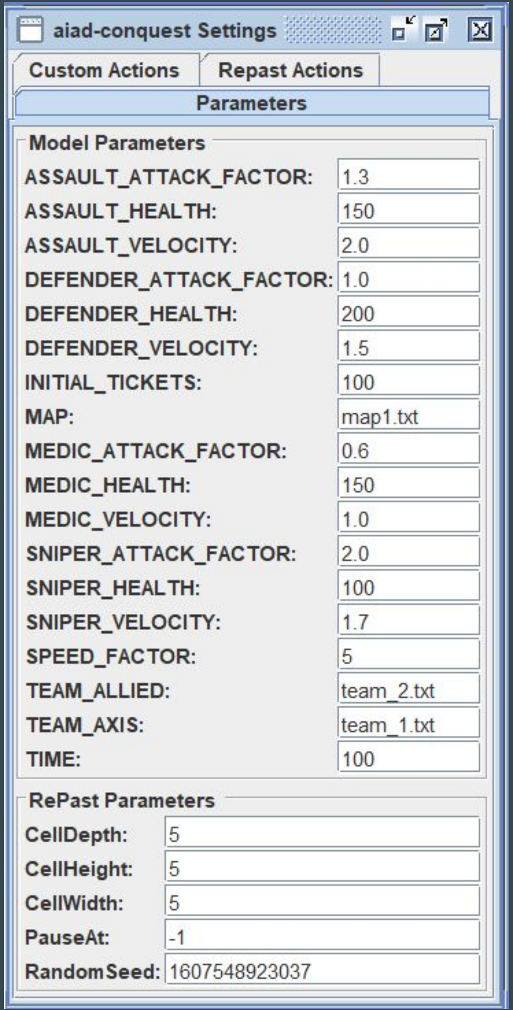
Like was said before, it is possible to change a lot of the independent variables of the game. To do this the user can use to following repast gui that appears when the app is stars running.

In this gui it is possible to change the zone map of the game as well has the configuration of each team. This is done by passing the name of the file to the parameters MAP, TEAM_ALLIED, TEAM_AXIS on the gui.

There are also parameters that affect the attack, health and velocity of each of the classes in the game.

Finally there are parameters like the time of the game, the initial tickets of each team and the speed factor that can also be configurable.

Note: All of the parameters come with default values already applied.



Model Parameters

ASSAULT_ATTACK_FACTOR:	1.3
ASSAULT_HEALTH:	150
ASSAULT_VELOCITY:	2.0
DEFENDER_ATTACK_FACTOR:	1.0
DEFENDER_HEALTH:	200
DEFENDER_VELOCITY:	1.5
INITIAL_TICKETS:	100
MAP:	map1.txt
MEDIC_ATTACK_FACTOR:	0.6
MEDIC_HEALTH:	150
MEDIC_VELOCITY:	1.0
SNIPER_ATTACK_FACTOR:	2.0
SNIPER_HEALTH:	100
SNIPER_VELOCITY:	1.7
SPEED_FACTOR:	5
TEAM_ALLIED:	team_2.txt
TEAM_AXIS:	team_1.txt
TIME:	100

RePast Parameters

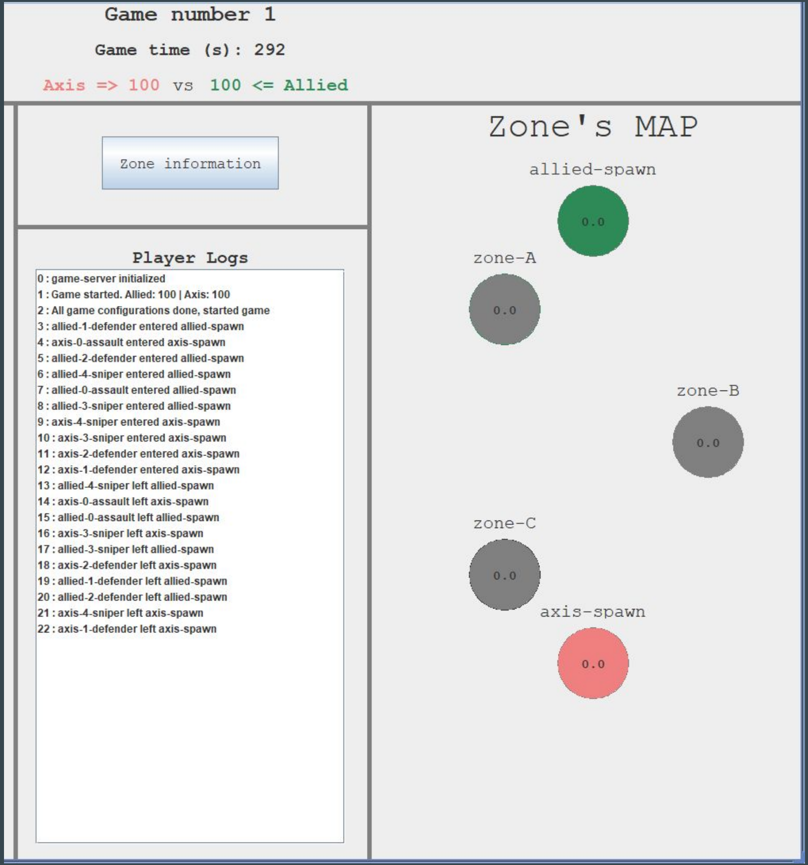
CellDepth:	5
CellHeight:	5
CellWidth:	5
PauseAt:	-1
RandomSeed:	1607548923037

GUI (cont)

The GUI helps to better understand what is happening in the current game and informs about previous games results and stats.

In one frame it is displayed the current game information with:

- Game number
- Game time
- Tickets remaining for each team
- A list of logs related to players interactions
- A zone's map which is a visual representation of the state and position of zones
- A button to switch between zone information and team composition tabs.



Game number 1

Game time (s): 292

Axis => 100 vs 100 <= Allied

Player Logs

```
0 : game-server initialized
1 : Game started. Allied: 100 | Axis: 100
2 : All game configurations done, started game
3 : allied-1-defender entered allied-spawn
4 : axis-0-assault entered axis-spawn
5 : allied-2-defender entered allied-spawn
6 : allied-4-sniper entered allied-spawn
7 : allied-0-assault entered allied-spawn
8 : allied-3-sniper entered allied-spawn
9 : axis-4-sniper entered axis-spawn
10 : axis-3-sniper entered axis-spawn
11 : axis-2-defender entered axis-spawn
12 : axis-1-defender entered axis-spawn
13 : allied-4-sniper left allied-spawn
14 : axis-0-assault left axis-spawn
15 : allied-0-assault left allied-spawn
16 : axis-3-sniper left axis-spawn
17 : allied-3-sniper left allied-spawn
18 : axis-2-defender left axis-spawn
19 : allied-1-defender left allied-spawn
20 : allied-2-defender left allied-spawn
21 : axis-4-sniper left axis-spawn
22 : axis-1-defender left axis-spawn
```

GUI (cont)

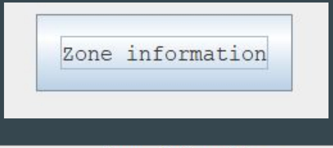
Still on the same frame, to the left there is a section dedicated either to zone information about where the player are or the teams composition. To switch between these two tabs there exists a button that is either labeled “Zone information” or “Team Comp”.

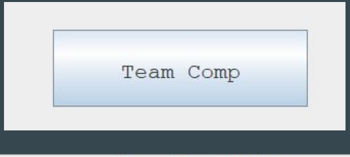
In the team composition tab, the following information is displayed for both team’s players:

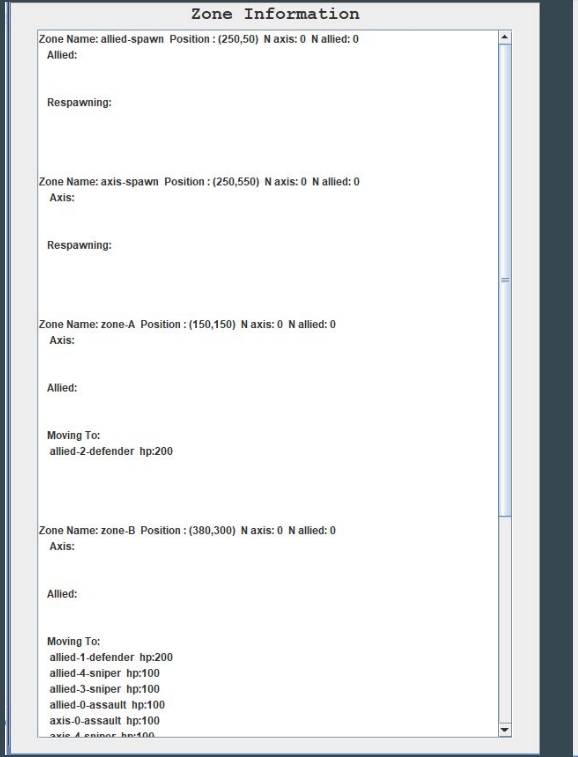
- Name
- Class
- Current Points

In the Zone information tab, there are sections for each zone with the :

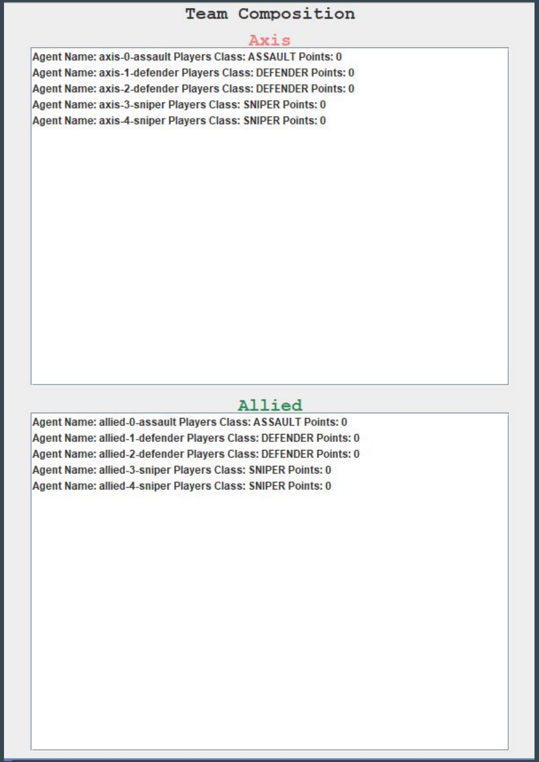
- Name of the zone
- Position of the zone
- Number of players of each team in the zone
- Specific players in that zone or coming to that zone and their health points






Zone Information

```
Zone Name: allied-spawn  Position : (250,50)  N axis: 0  N allied: 0
  Allied:


  Respawning:



Zone Name: axis-spawn  Position : (250,550)  N axis: 0  N allied: 0
   Axis:


  Respawning:



Zone Name: zone-A  Position : (150,150)  N axis: 0  N allied: 0
   Axis:


  Allied:


  Moving To:
   allied-2-defender  hp:200



Zone Name: zone-B  Position : (380,300)  N axis: 0  N allied: 0
   Axis:


  Allied:


  Moving To:
   allied-1-defender  hp:200
   allied-4-sniper  hp:100
   allied-3-sniper  hp:100
   allied-0-assault  hp:100
   axis-0-assault  hp:100
   axis-4-sniper  hp:100
```


Team Composition

Axis

```
Agent Name: axis-0-assault Players Class: ASSAULT Points: 0
Agent Name: axis-1-defender Players Class: DEFENDER Points: 0
Agent Name: axis-2-defender Players Class: DEFENDER Points: 0
Agent Name: axis-3-sniper Players Class: SNIPER Points: 0
Agent Name: axis-4-sniper Players Class: SNIPER Points: 0
```

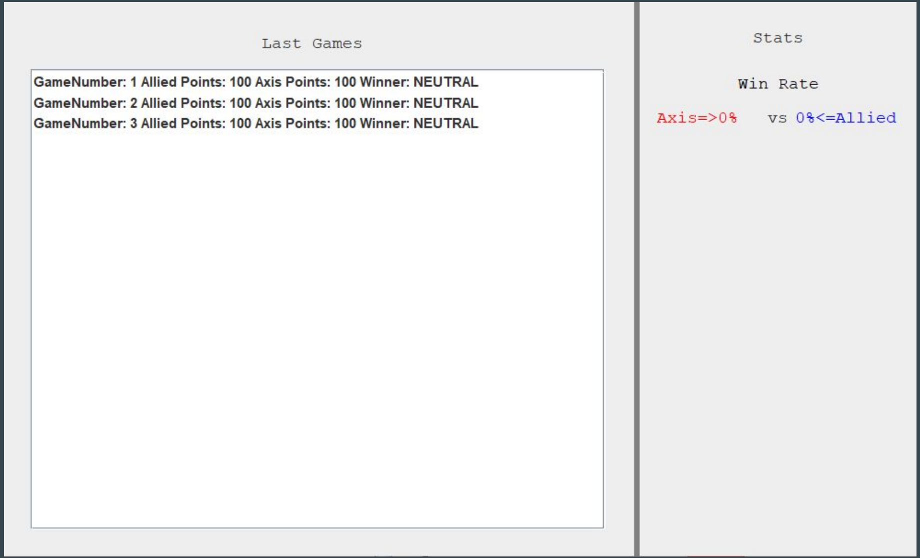
Allied

```
Agent Name: allied-0-assault Players Class: ASSAULT Points: 0
Agent Name: allied-1-defender Players Class: DEFENDER Points: 0
Agent Name: allied-2-defender Players Class: DEFENDER Points: 0
Agent Name: allied-3-sniper Players Class: SNIPER Points: 0
Agent Name: allied-4-sniper Players Class: SNIPER Points: 0
```

GUI (cont)

In another frame it is displayed information about previous games and stats about them such as win rate percentage for both teams. The information about previous games boils down to:

- Game number
- Allied Tickets remaining
- Axis Tickets remaining
- Team that won the game



```
GameNumber: 1 Allied Points: 100 Axis Points: 100 Winner: NEUTRAL
GameNumber: 2 Allied Points: 100 Axis Points: 100 Winner: NEUTRAL
GameNumber: 3 Allied Points: 100 Axis Points: 100 Winner: NEUTRAL
```

```
Axis=>0%   vs 0%<=Allied
```