ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Final Project – Behavior of Different Agents: Battlefield

Group 18:

Jhorman Arlex Pérez Buendía

Wilfredo Joshua Robinson Moore

December 24th, 2019

Table of Contents

1. Introduction

For this Final Project, we developed a simulation that used at least 50 agents with different personal traits, meetup places, and even interaction/behavior rulesets. This means this simulation got us closer than ever before to simulating a real-life scenario within GAMA. The simulation must also make agents use the FIPA protocol for long-range communication and it must be continuously running.

To achieve this feat, we had may possible scenarios from previous assignments: a festival with restaurants and bars, a music festival, an auction festival, among others. However, we decided to try out something completely new: a battlefield. The battlefield we developed has two continuously opposing armies with different soldier types, different battlefield zones where the rules change, and two commanders that do their best to win; sometimes even joining in on the fight itself if their last line of defense is breached. The battle ends when one of the two Commanders falls in battle or when an army captures the opposing home base.

To explain our approach and results in detail, we first delve into the explanation of how the battlefield works and the different species involved in it. Then, we explain how it was implemented, including a basic summary of most possible scenarios that can occur during battle. Finally, we conclude by discussing what each section has taught us, their different challenges, and what potential we see for applications similar to these in the future.

2. Approach

This chapter will discuss three main sections: the main goal, the Battlefield itself and the roles of all species interacting within it.

2.1 Main Goal

Each army has a simple goal: defeat the other army to end the battle as fast as possible. There are two possible ways to accomplish this:

1. Defeat the enemy Commander
2. Reach the end of the enemy Home Base. If this goal is reached, the battle ends without considering if the army Commander is alive or not. This goal line is represented by the edges of the map, for example: Alliance #1 must reach x = 0.0 and Alliance #2 must reach x = 100.0.

2.2 Battlefield Explanation

The Battlefield, as seen during the beginning of the simulation, can be seen in Figure 1. The Battlefield has the following sections where different agents can interact:

1. **Combat Zones**: All fighting between active soldiers is done here. Each alliance has its own combat zone, and some agent interactions between Medics, Transports, and Soldiers change depending on which combat zone they are in.
2. **Last Line of Defense**: each army has to make the utmost effort to keep the enemy army from crossing this red line. If the red line is crossed, allied soldiers will not be able to intercept the intruder as they are too far within the combat zone. The only option at this point is for the Commander to step in and fight himself if the red line is crossed. This is a great risk, as a dead Commander means the battle is lost.
3. **Home Base Area**: this is the area where each Commander will take matters into its own hands and stop enemy intruders.
4. **Goal line (edges of the map):** the goal of all soldiers is to reach this goal line and kill any enemies in their way. Once this goal is reached, they have achieved victory.
5. **Supply Area:** this area is where the Provisions agent makes his business. All agents have a limited supply of bullets, medicine, fuel, and others, and must come to their respect Provisions agent to restock while the battle is taking place.
6. **Reserve Area:** all soldiers not in active combat must stay within this area. Only an order from the Commander can make a reserve soldier act and leap into battle.

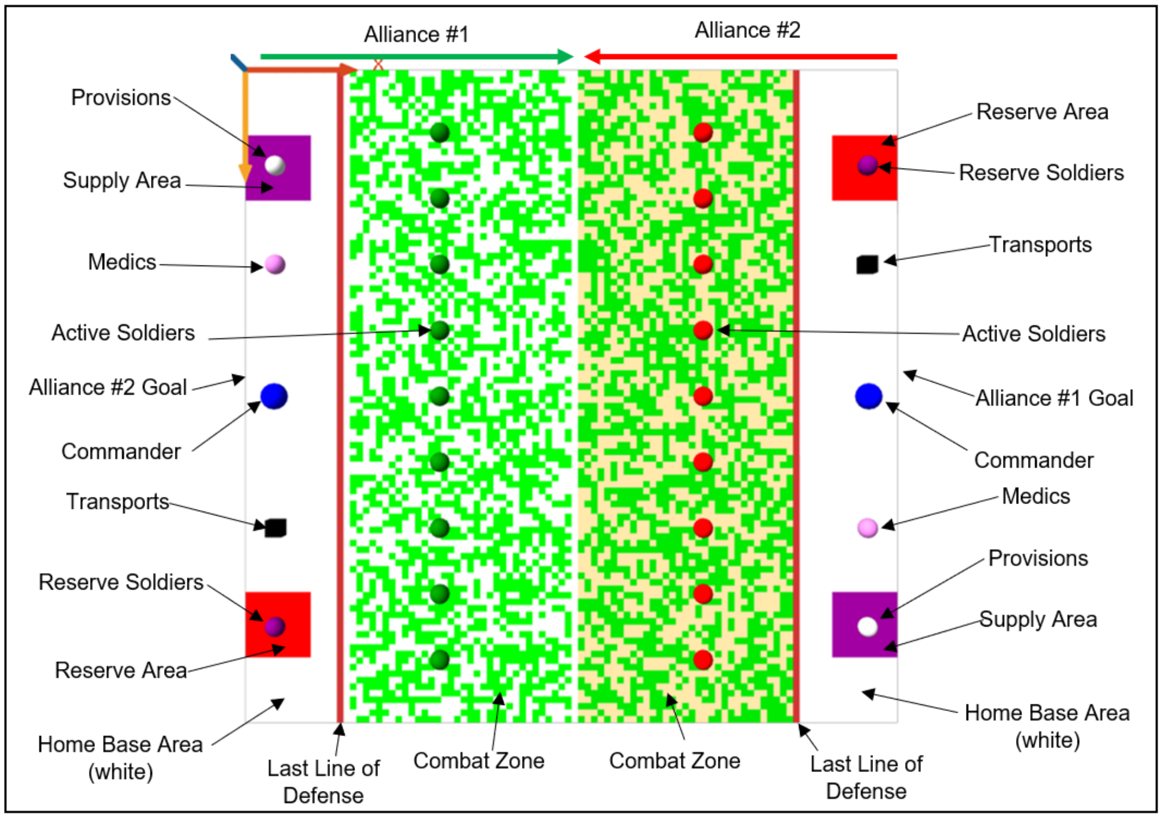


Figure 1: Battlefield Overview

2.3 Species and their Roles

Many different species are interacting at the same time within this Battlefield. Using Figure 1 as a reference, these species and their respective roles are:

1. **Soldier**:
   1.1 **Active Soldiers**: an active Soldier's color can be red or green depending on the Alliance it belongs to. An active soldier's main goal is to kill enemy soldiers and reach its respective goal line to end the battle. All active soldiers are within the different Combat Zones. There is a total of 9 active soldiers at any time per alliance, meaning the battle begins with a total of 18 active soldiers. All soldiers have a limited supply of bullets (used to fight). Once their bullets run out, they must go to their Supply Area to get more.
   1.2 **Reserve Soldiers**: a reserve Soldier's color is purple, and they can only be found within the Reserve Area. The role of reserve soldiers is to be on stand-by until they receive an order from their Commander to get their gear, replace the dead soldier in the Combat Zone, and enter the fight.
2. **Medic**: each army has a maximum of two (2) medics, colored pink. The Medic's role is to heal any injured active Soldiers as quick as possible. Since their ground speed is not the best for this, they employ the services of the Transport agent to help them accomplish their goal. However, Medics are frail and will die instantly if an enemy soldier attacks them while on the enemy Combat Zone. All Medics have a limited supply of medicine (used to heal). Once their medicine runs out, they must go to their Supply Area to get more.
3. **Transport:** each army has a maximum of two (2) transports, colored **black**. The Transport's role is to aid the Medic to get to as many active Soldiers as possible before an enemy soldier can finish off their injured comrade. A Transport's speed is much higher than a regular foot soldier's speed, so it is the perfect companion for the Medics. However, if the Medic on board is killed while traversing and enemy combat zone, the Transport will be stolen by the enemy army. All Transports have a limited supply of fuel (used to traverse around the Battlefield). Once their fuel runs out, they must go to their Supply Area to get more.
4. **Provisions**: each Army has one agent in charge of supply chain management, colored white. Soldiers, Medics and Transports all have a limited amount of a corresponding parameter that allows them to perform their actions. Provisions must make sure that it always has enough of these to give its comrades once they come to restock. To ensure this, every time Provisions is running low on stock, it will call its Commander to bring more materials and be ready for any other soldier coming in.
5. **Commander:** the Commander, colored blue, is the main strategist for each army. It is in charge of monitoring a large number of variables, such as:
   1. Active Soldiers on the Battlefield: any time a Soldier dies, the Commander must order a reserve soldier to take its place.

2. Reserve Soldiers available: the Commander must know at all times how many Reserves are available to replace his dead units.
3. Messenger: all agents communicate through the Commander using the FIPA protocol for a variety of purposes, such as:
    - Whenever a Soldier dies or is injured, or
    - Provisions runs out of stock, or
    - a Reserve Soldier needs to go into battle, or
    - a Medic needs a transport to go and heal a Soldier,
    - among others.

6. **BattleGrid:** This agent covers two main roles:
    1. Serve as a visual guide for the viewer to understand where agents are in the Battlefield and where each zone starts and ends.
    2. Allow active soldiers to have "lanes" where they can move. Active soldiers can only move through these defined lanes (only in the x axis), to make the flow of battle more understandable. The only time when Soldiers can move to other *y* coordinates is when they run out of bullets and need to go to Provisions or when they are changing from being a Reserve to an Active Soldier.
    3. Only Soldier agents are restricted by these lanes. All others move around freely within the Battlefield.

3. Implementation

Now that we know the main goal of both armies, this section will focus on detailing all possible scenarios within the simulation and how we implemented them.

3.1 General Aspects

Some general aspects to take into consideration are:

1. All long-range communication between agents is handled with the FIPA Protocol. All agents have a reflex dedicated to receiving messages. All messages have a different *message.contents[0]* in text form which identifies the purpose of the message. For example, this text may be "Injured!", "Coming to pick you up!", "I'm dead!", etc.
2. All short-range communication is handled with the "ask" command. It is simpler, faster, and ideal for agents communicating while in close range.
3. All agents have a reflex destined to receiving or sending FIPA messages. Since inform messages are deleted almost instantly after being seen by any agent, all agents save the messages from their default *informs* list to a personal, more easily manageable list. Within this list, agents can decide when to read, reply to, and delete the message at will.

4. All agents were developed as standalone agents serving their purpose within the Battlefield. Once their core behavior was established, the different interactions between them were programmed.
5. Many agents have a *resetControlVariables* action used to reset variables whenever an important event happened, such as restocking on materials or going from being injured to active again.
6. All agents except for the BattleGrid have an *alliance* variable which defines to which army they belong to. This variable also determines if another agent is an enemy or an ally.
7. All agents except for the BattleGrid have an <*agent*>MessageList, where <*agent*> depends on which agent I am receiving messages from. **Queues** are constantly implemented throughout this simulation, as agents can still receive orders from the Commander but may be *occupied* accomplishing a previous order or restocking on materials. Once the agent is not *occupied*, the first message in the **queue** is taken and used as its next order. It is **extremely important** for the Commander to have an excellent management of queues, as being the central strategist can easily make it crash if too many messages with different directives are received at once.
8. To color the Battlefield and its different sections, a *BattleGrid* agent was created. Its main roles are explained in Section 2.3.

3.2 Agents' Main Control Variables

All agents are listed together with their main control variables and how these affect their behavior and interactions with other agents in Table 1. These variables are used to execute their main purpose explained in Section 2.3.

Table 1: Agents and their Main Personal Traits

Agent	Main Control Variables or Personal Traits	Affected Behavior
Commander	*battleInitiated*	This variable becomes *true* once the Commanders sends a message to all agents in its Army. This begins the battle. All behaviors before this are basically nil.
	health	The Commander has the highest health of any soldier in the army. It is also one of the most important parameters in the whole simulation, because if the Commander's health drops to zero, the battle ends. This parameter's value drops every time the Commander fights with an enemy soldier.
	protectingZone	This completely changes the Commander's behavior from a passive strategist into the most powerful Soldier in the army. Once an enemy soldier crosses the LastLineOfDefense, the Commander leaps into action and fights with him directly. However, this is a double-edged sword, as the Commander's health decreases with every fight.

Soldier	*fighting*	This variable becomes *true* once it has encountered an enemy Soldier in its movement lane. Soldiers then start their fighting animation and any other behavior is nil while this occurs.
	inField	This defines whether a Soldier is an Active or Reserve Soldier. If a Soldier is Active, its goal will be to get to the enemy camp and kill any enemies in its way. If it is Reserve, it will wait on standby in the Reserve Area until an order from the Commander arrives.
	timeInFight	This defined the length of a battle between enemy Soldiers. Knowing this trait is important as all other reflexes have to be put on pause while the fight ensues.
	injured	After a fight has ended, if the Soldier's health is low (but still above 0), the Soldier will become injured and turn yellow. This means it is unable to move or fight until a Medic heals him. Additionally, if an enemy Soldier comes into contact with this injured Soldier, it will die.
	health	After a fight has ended, the Soldier's health decreases by an amount equal to the power of the enemy Soldier. If the fight ends and its health is still moderately high, it can continue fighting almost immediately without the need for a Medic.
	power	After a fight ends, an enemy Soldier's health decreases by the amount specified in this variable. All soldiers have different power values.
	bullets	After a fight ends, this trait gets its value reduced by 1. Soldiers can only fight as long as this trait is above 0. If it is equal to zero, it must leave its post and go to Provisions to restock on bullets.
Medic	*occupied*	If this variable is true, it means that all other order received by the Commander will be put in its queue. This variable means the Medic is solely focused on healing its assigned injured Soldier.
	injuredLocation	This variable has a value different from nil when the Medic has been assigned a Soldier to heal. It defines the Medics targetPoint, and the Medic uses it to tell the Transport where to take it at any point in the simulation.
	medicine	Extremely important, as the Medic cannot heal any Soldier if medicine = 0. If medicine = 0, all orders are put on hold and all other behaviors are overwritten. The Medic will call its Transport and tell it to quickly take it to its corresponding Provisions, to restock.
	transportAssigned	If this is true, the Medic has already been assigned a Transport and it is on its way. The Medic will stop calling other Transports once this becomes true. The Transport will essentially become the Medic's partner during its healing missions.
Transport	*occupied*	If this variable is true, it means that all other order received by Medic swill be put in its queue. This variable means the Transport is solely focused on assisting its assigned Medic.
	fuel	Extremely important, as the Transport cannot move around any Medic if fuel= 0. If fuel = 0, all orders are put on hold and all other behaviors are overwritten. The Transport will take its assigned Medic to the corresponding Provisions, to restock on fuel.

	moveControl	The Transport remains on standby until a Medic calls it and turns this variable into *true*. A Transport **will not move** unless this variable's value is true. Only Medics can do this. If the Transport is stolen by the opposing army, this value will be set to false, rendering it unusable.
Provisions	*medicineStock*	This allows Provisions to fulfill its roles to allied Medics. Provisions must make sure that it is above 0 at all times and must call the Commander if it enters dangerously low values. The Commander will then come to restock.
	fuelStock	This allows Provisions to fulfill its roles to allied Transports. Provisions must make sure that it is above 0 at all times and must call the Commander if it enters dangerously low values. The Commander will then come to restock.
	bulletStock	This allows Provisions to fulfill its roles to allied Soldiers. Provisions must make sure that it is above 0 at all times and must call the Commander if it enters dangerously low values. The Commander will then come to restock.

3.3 BattleField Zones

All Agent behaviors are also linked to which Zone in the Battlefield they are in. The effects of each Zone and how they affect Agent behaviors are summarized in Table 2.

Table 2: Summary of each Zone's effects on Agent Behavior

BattleField Zone	Effect on Agent Behavior
Combat Zone	Soldiers fight enemy Soldiers
	Medics heal allied Soldiers
	Enemy Soldiers can kill Medics if Medic is within enemy Combat Zone
	Enemy Soldiers can steal Transports if Transport is within enemy Combat Zone
	Soldiers may end up injured after a fight
	Soldiers may end up dead after a fight
	Soldiers may instantly kill other injured Soldiers in this zone
Home Base Zone	Soldiers cannot fight each other in this zone
	Commander can change from being a strategist to a brutal Soldier as a last resort
	If a Commander is killed within this zone, the battle ends
	If an enemy Soldier reaches the edge of this zone, the battle ends
	Commander only works within this Zone
Supply Area	Provisions can only interact with other agents in this zone. Wireless transmission of bullets, fuel, or medicine is not allowed in this Battlefield
Reserve Area	Only Reserve Soldiers may be in this Zone. No fights or exchanges occur in this zone

	Reserve Soldiers can receive order from Commander to replace dead soldiers while on this Zone. Once they go to the Combat Zone, they are no longer considered Reserve Soldiers.

4. Results

Considering everything mentioned in previous sections, the results can be summarized in the following section.

4.1 General Sequence of the Simulation

After many iterations of the battle, the general sequence of events is as follows:

1. Commanders issued the initiateBattle command.
2. Soldiers from both alliances start moving forward towards their objective. Soldiers clash and fight. The fight animation begins.
3. After a fight, a Soldier may end up injured:
   3.1 Is Soldier ended up injured, Commander will tell the Medic to provide support
   3.2 Medic will call for any available Transport
   3.3 Transport will pick up Medic and quickly go to the injured Soldier's location.
   3.4 Medic will heal Soldier, and its medicine will be reduced by 1. The Transport's fuel will be reduced by 1 as well.
   3.5 Medic and Transport will become available again. Soldier will resume the fight.
4. After a fight, a Soldier may end up dead:
   4.1 Soldier will notify Commander of its demise right before it dies
   4.2 Commander will order a Reserve Soldier to go and take the place of the dead Soldier
   4.3 Reserve Soldier will change from Reserve to Active and resume the previous Soldier's fight.
   4.4 This goes on until there are no more available Reserve Soldiers.
5. If an enemy Soldier touches an injured Soldier in the Combat Zone, the injured Soldier will die. Step 4 begins again here.
6. During the simulation, Soldiers, Medics, and Transports may run out of ammunition, medicine, or fuel and need to restock. They will cease their current behavior and quickly go to Provisions to restock.
7. Provisions may end up low on supplies. Provisions calls Commanders to restock.
8. Eventually, one Soldier will be able to breach the opposing Army's LastLineOfDefense. If this happens, the Commander takes an active role and proceeds to kill the opposing intruder.
9. If either the Commander dies or an opposing Soldier is able to reach the goal line, the battle ends.

10. The simulation restarts from the beginning after a certain amount of time. This results in a continuously running simulation.

4.2 Monitoring the Flow of Battle

To monitor the flow of battle, four global variables were introduced to the simulation and are plotted in real time:

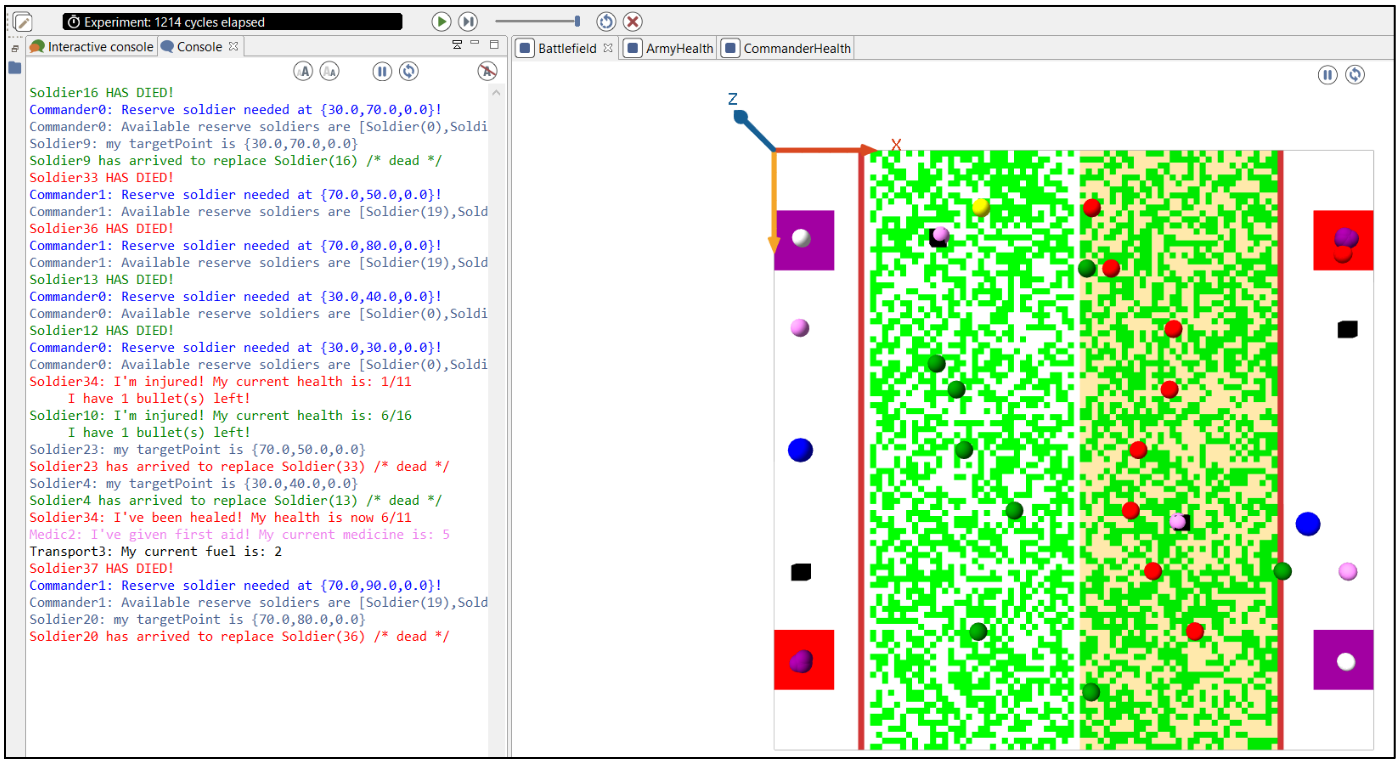
1. commanderHealthAlliance1: stores Alliance 1's Commander's health.
2. commanderHealthAlliance2: stores Alliance 2's Commander's health.
3. armyHealthAlliance1: stores the sum of all Alliance 1's Soldier's health.
4. armyHealthAlliance2: stores the sum of all Alliance 2's Soldier's health.

Pie charts are generated using the *chart* command in the *experiment* section of the program. Comparing both commanderHealthAlliance gives us an idea of how much a Commander has fought, which tells us how strong or weak the overall army has been. Usually, the alliance with the lowest commanderHealthAlliance loses.

Additionally, the same is done for both armyHealthAlliance variables. Instead of just looking at the Commander, this pie chart gives us a global overview of the whole fight. Amazingly, even if an Alliance's overall health is constantly below the other, it may still win by defeating the Commander or reaching the goal line with at least one soldier. This means **these plots give us a general idea of who has the highest chances of winning, but not of who will definitely win.**

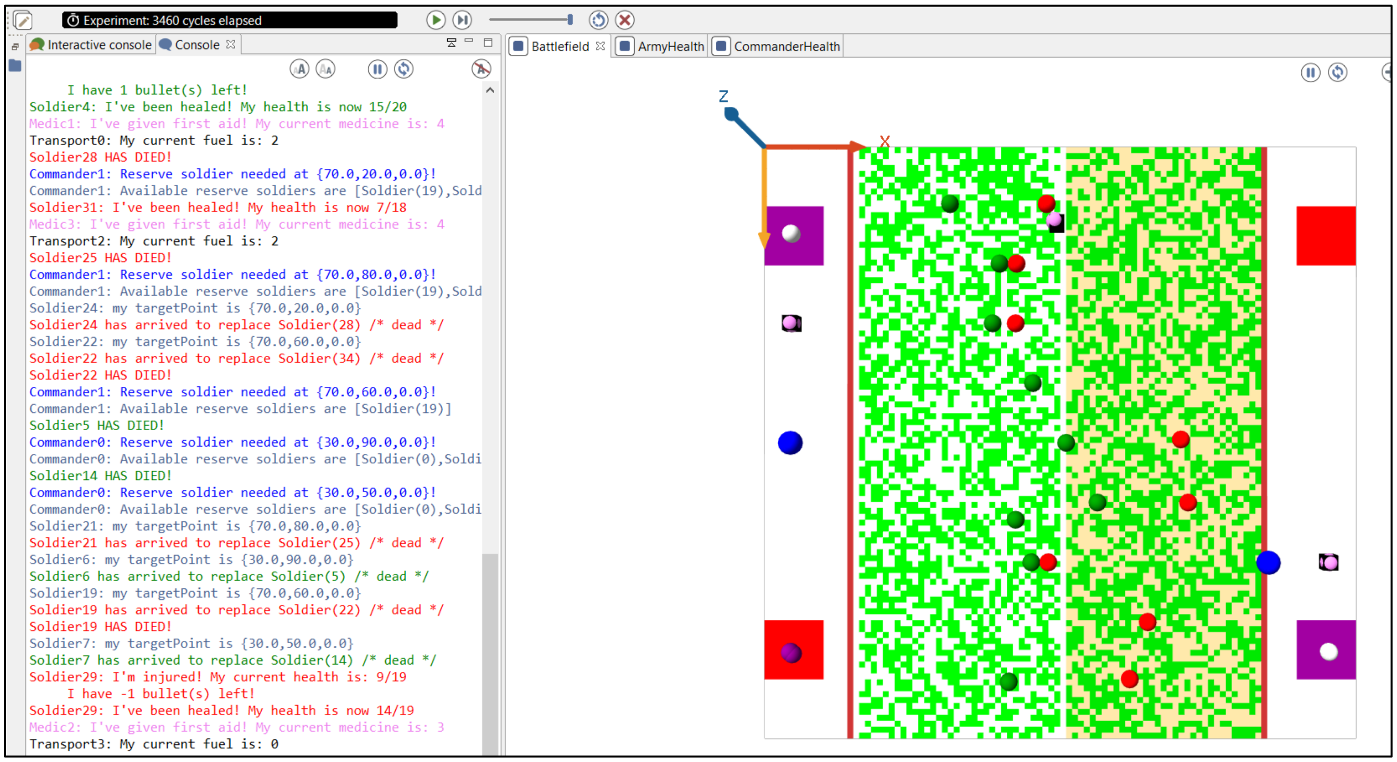
To give an example, we will use Figures 2, 3, 4, 5 and 6. First, take a look at Figure 2. Here we see a battle after just 1214 cycles, but we can already see that Commander2 (right) has already found the need to intervene. This means that most likely Alliance 1's Soldiers are more powerful than Alliance 2's. This scenario does not bode well for Alliance 2. Here we can also see some Medics with their **Transports** trying to heal injured Soldiers.

Now take a look at Figure 3. This is the same battle after 3460 cycles. We can see here that all of Alliance 2's (red) Reserve Soldiers have been decimated. Alliance 2 has been getting pushed back the whole battle, which means its overall chances of winning are very low. These low chances are also reflected by Figure 4. Alliance 1's overall health is much higher than Alliance 2's. Eventually, after 6024 cycles, Alliance #1 completely overpowers Alliance #2, even stealing some of its Transports and killing some of its Medics in the process. This can be seen in Figure 5.



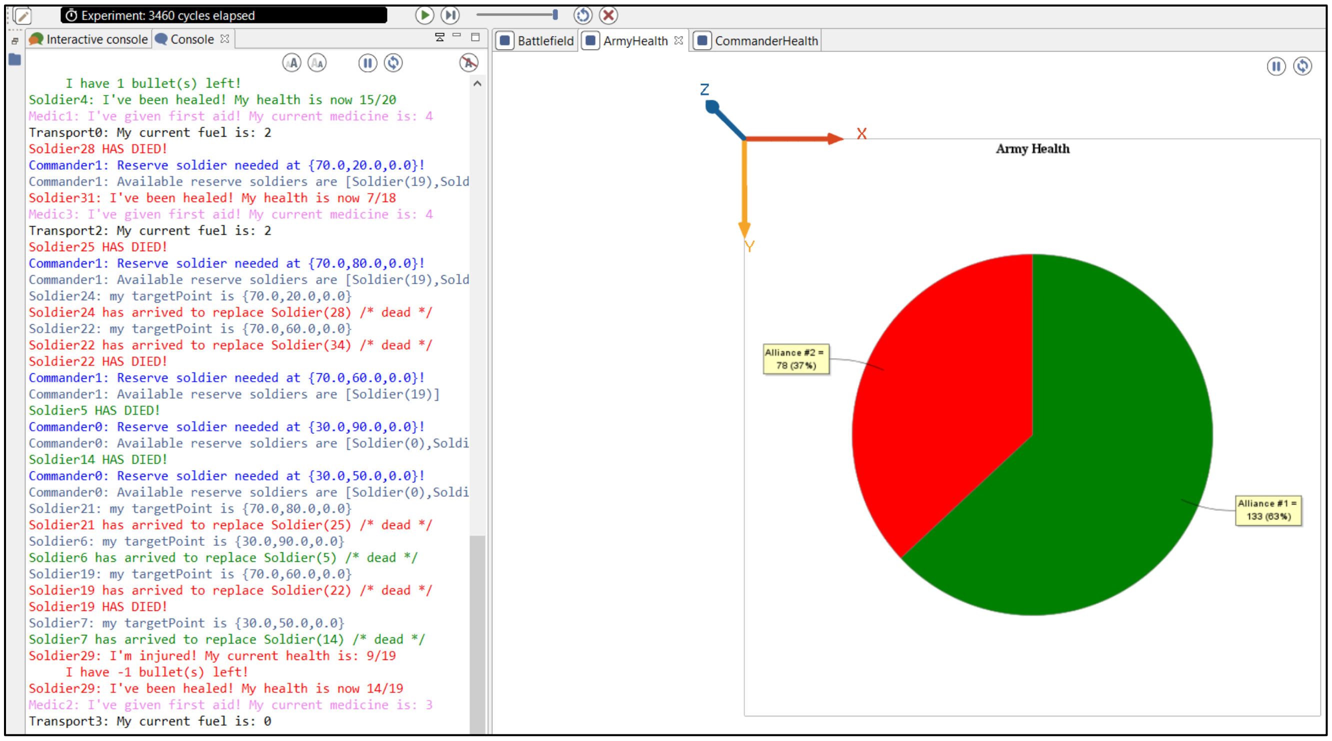
```
Soldier16 HAS DIED!
Commander0: Reserve soldier needed at {30.0,70.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier9: my targetPoint is {30.0,70.0,0.0}
Soldier9 has arrived to replace Soldier(16) /* dead */
Soldier33 HAS DIED!
Commander1: Reserve soldier needed at {70.0,50.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier36 HAS DIED!
Commander1: Reserve soldier needed at {70.0,80.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier13 HAS DIED!
Commander0: Reserve soldier needed at {30.0,40.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier12 HAS DIED!
Commander0: Reserve soldier needed at {30.0,30.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier34: I'm injured! My current health is: 1/11
     I have 1 bullet(s) left!
Soldier10: I'm injured! My current health is: 6/16
     I have 1 bullet(s) left!
Soldier23: my targetPoint is {70.0,50.0,0.0}
Soldier23 has arrived to replace Soldier(33) /* dead */
Soldier4: my targetPoint is {30.0,40.0,0.0}
Soldier4 has arrived to replace Soldier(13) /* dead */
Soldier34: I've been healed! My health is now 6/11
Medic2: I've given first aid! My current medicine is: 5
Transport3: My current fuel is: 2
Soldier37 HAS DIED!
Commander1: Reserve soldier needed at {70.0,90.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier20: my targetPoint is {70.0,80.0,0.0}
Soldier20 has arrived to replace Soldier(36) /* dead */
```

Figure 2: A battle after 1214 cycles.



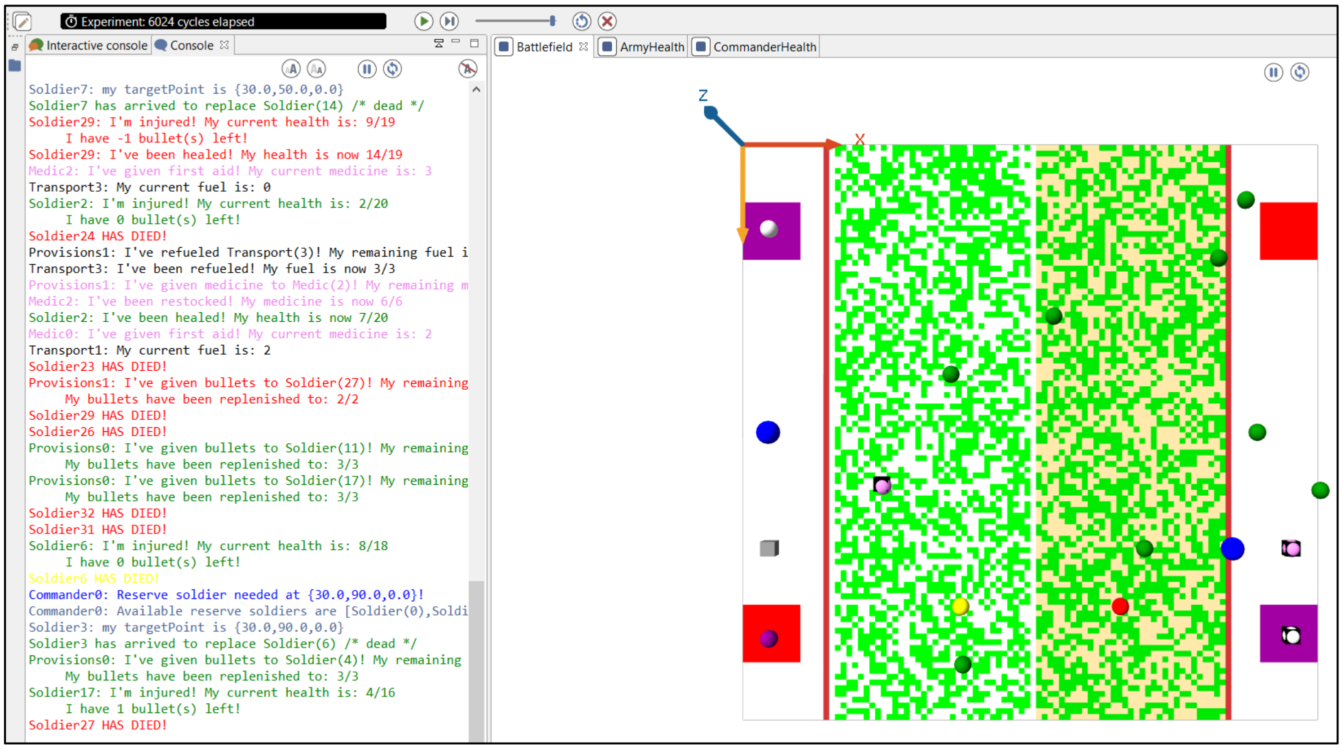
```
     I have 1 bullet(s) left!
Soldier4: I've been healed! My health is now 15/20
Medic1: I've given first aid! My current medicine is: 4
Transport0: My current fuel is: 2
Soldier28 HAS DIED!
Commander1: Reserve soldier needed at {70.0,20.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier31: I've been healed! My health is now 7/18
Medic3: I've given first aid! My current medicine is: 4
Transport2: My current fuel is: 2
Soldier25 HAS DIED!
Commander1: Reserve soldier needed at {70.0,80.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier24: my targetPoint is {70.0,20.0,0.0}
Soldier24 has arrived to replace Soldier(28) /* dead */
Soldier22: my targetPoint is {70.0,60.0,0.0}
Soldier22 has arrived to replace Soldier(34) /* dead */
Soldier22 HAS DIED!
Commander1: Reserve soldier needed at {70.0,60.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19)]
Soldier5 HAS DIED!
Commander0: Reserve soldier needed at {30.0,90.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier14 HAS DIED!
Commander0: Reserve soldier needed at {30.0,50.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier21: my targetPoint is {70.0,80.0,0.0}
Soldier21 has arrived to replace Soldier(25) /* dead */
Soldier6: my targetPoint is {30.0,90.0,0.0}
Soldier6 has arrived to replace Soldier(5) /* dead */
Soldier19: my targetPoint is {70.0,60.0,0.0}
Soldier19 has arrived to replace Soldier(22) /* dead */
Soldier19 HAS DIED!
Soldier7: my targetPoint is {30.0,50.0,0.0}
Soldier7 has arrived to replace Soldier(14) /* dead */
Soldier29: I'm injured! My current health is: 9/19
     I have -1 bullet(s) left!
Soldier29: I've been healed! My health is now 14/19
Medic2: I've given first aid! My current medicine is: 3
Transport3: My current fuel is: 0
```

Figure 3: Figure 2’s battle after 3460 cycles.



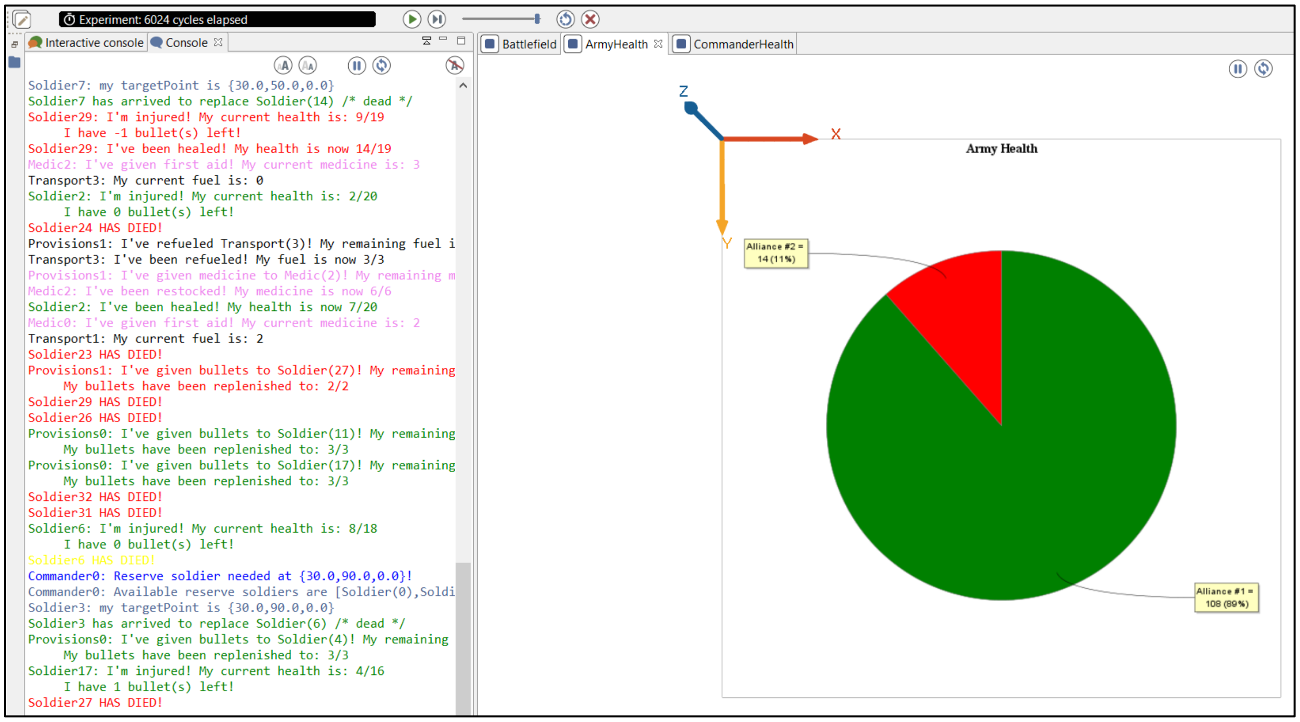
```
     I have 1 bullet(s) left!
Soldier4: I've been healed! My health is now 15/20
Medic1: I've given first aid! My current medicine is: 4
Transport0: My current fuel is: 2
Soldier28 HAS DIED!
Commander1: Reserve soldier needed at {70.0,20.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier31: I've been healed! My health is now 7/18
Medic3: I've given first aid! My current medicine is: 4
Transport2: My current fuel is: 2
Soldier25 HAS DIED!
Commander1: Reserve soldier needed at {70.0,80.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19),Sold
Soldier24: my targetPoint is {70.0,20.0,0.0}
Soldier24 has arrived to replace Soldier(28) /* dead */
Soldier22: my targetPoint is {70.0,60.0,0.0}
Soldier22 has arrived to replace Soldier(34) /* dead */
Soldier22 HAS DIED!
Commander1: Reserve soldier needed at {70.0,60.0,0.0}!
Commander1: Available reserve soldiers are [Soldier(19)]
Soldier5 HAS DIED!
Commander0: Reserve soldier needed at {30.0,90.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier14 HAS DIED!
Commander0: Reserve soldier needed at {30.0,50.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier21: my targetPoint is {70.0,80.0,0.0}
Soldier21 has arrived to replace Soldier(25) /* dead */
Soldier6: my targetPoint is {30.0,90.0,0.0}
Soldier6 has arrived to replace Soldier(5) /* dead */
Soldier19: my targetPoint is {70.0,60.0,0.0}
Soldier19 has arrived to replace Soldier(22) /* dead */
Soldier19 HAS DIED!
Soldier7: my targetPoint is {30.0,50.0,0.0}
Soldier7 has arrived to replace Soldier(14) /* dead */
Soldier29: I'm injured! My current health is: 9/19
     I have -1 bullet(s) left!
Soldier29: I've been healed! My health is now 14/19
Medic2: I've given first aid! My current medicine is: 3
Transport3: My current fuel is: 0
```

Figure 4: Pie Chart summarizing the status of Figure 3’s battle in real time



```
Soldier7: my targetPoint is {30.0,50.0,0.0}
Soldier7 has arrived to replace Soldier(14) /* dead */
Soldier29: I'm injured! My current health is: 9/19
     I have -1 bullet(s) left!
Soldier29: I've been healed! My health is now 14/19
Medic2: I've given first aid! My current medicine is: 3
Transport3: My current fuel is: 0
Soldier2: I'm injured! My current health is: 2/20
     I have 0 bullet(s) left!
Soldier24 HAS DIED!
Provisions1: I've refueled Transport(3)! My remaining fuel i
Transport3: I've been refueled! My fuel is now 3/3
Provisions1: I've given medicine to Medic(2)! My remaining m
Medic2: I've been restocked! My medicine is now 6/6
Soldier2: I've been healed! My health is now 7/20
Medic0: I've given first aid! My current medicine is: 2
Transport1: My current fuel is: 2
Soldier23 HAS DIED!
Provisions1: I've given bullets to Soldier(27)! My remaining
     My bullets have been replenished to: 2/2
Soldier29 HAS DIED!
Soldier26 HAS DIED!
Provisions0: I've given bullets to Soldier(11)! My remaining
     My bullets have been replenished to: 3/3
Provisions0: I've given bullets to Soldier(17)! My remaining
     My bullets have been replenished to: 3/3
Soldier32 HAS DIED!
Soldier31 HAS DIED!
Soldier6: I'm injured! My current health is: 8/18
     I have 0 bullet(s) left!
Soldier6 HAS DIED!
Commander0: Reserve soldier needed at {30.0,90.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier3: my targetPoint is {30.0,90.0,0.0}
Soldier3 has arrived to replace Soldier(6) /* dead */
Provisions0: I've given bullets to Soldier(4)! My remaining
     My bullets have been replenished to: 3/3
Soldier17: I'm injured! My current health is: 4/16
     I have 1 bullet(s) left!
Soldier27 HAS DIED!
```

Figure 5: Figure 2’s battle at cycle 6024. Alliance #2 has already lost. Note the stolen gray transport in Alliance #1’s side.



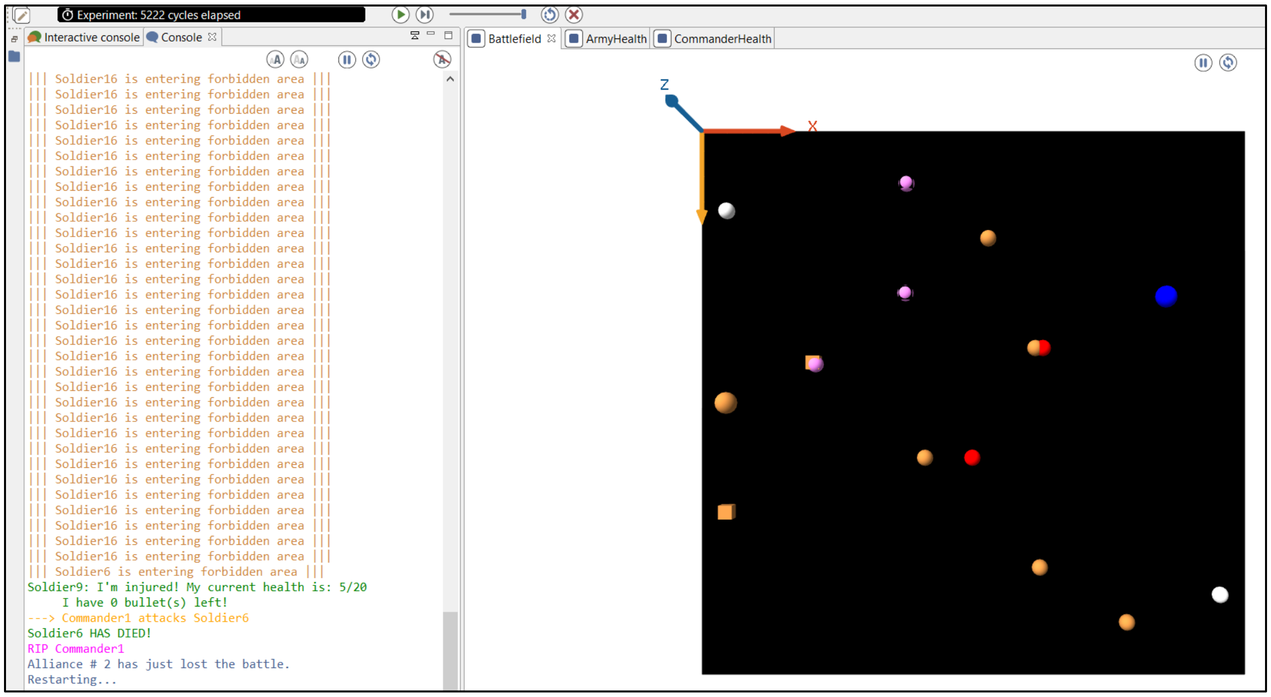
```
Soldier7: my targetPoint is {30.0,50.0,0.0}
Soldier7 has arrived to replace Soldier(14) /* dead */
Soldier29: I'm injured! My current health is: 9/19
     I have -1 bullet(s) left!
Soldier29: I've been healed! My health is now 14/19
Medic2: I've given first aid! My current medicine is: 3
Transport3: My current fuel is: 0
Soldier2: I'm injured! My current health is: 2/20
     I have 0 bullet(s) left!
Soldier24 HAS DIED!
Provisions1: I've refueled Transport(3)! My remaining fuel i
Transport3: I've been refueled! My fuel is now 3/3
Provisions1: I've given medicine to Medic(2)! My remaining m
Medic2: I've been restocked! My medicine is now 6/6
Soldier2: I've been healed! My health is now 7/20
Medic0: I've given first aid! My current medicine is: 2
Transport1: My current fuel is: 2
Soldier23 HAS DIED!
Provisions1: I've given bullets to Soldier(27)! My remaining
     My bullets have been replenished to: 2/2
Soldier29 HAS DIED!
Soldier26 HAS DIED!
Provisions0: I've given bullets to Soldier(11)! My remaining
     My bullets have been replenished to: 3/3
Provisions0: I've given bullets to Soldier(17)! My remaining
     My bullets have been replenished to: 3/3
Soldier32 HAS DIED!
Soldier31 HAS DIED!
Soldier6: I'm injured! My current health is: 8/18
     I have 0 bullet(s) left!
Soldier6 HAS DIED!
Commander0: Reserve soldier needed at {30.0,90.0,0.0}!
Commander0: Available reserve soldiers are [Soldier(0),Soldi
Soldier3: my targetPoint is {30.0,90.0,0.0}
Soldier3 has arrived to replace Soldier(6) /* dead */
Provisions0: I've given bullets to Soldier(4)! My remaining
     My bullets have been replenished to: 3/3
Soldier17: I'm injured! My current health is: 4/16
     I have 1 bullet(s) left!
Soldier27 HAS DIED!
```

Figure 6: Pie Chart summarizing the result of the Battle. Alliance #2 was completely overpowered.

4.3 Creativity Implementation

For the creative part, we implemented a "Night" criteria where once a battle has been finished, the armies wait until the next day comes to get in action again. The color of the map changes to black and all of the agents on it stay blocked for a certain time frame (2000 time units in our case). An example of the BattleField at Night can be seen on Figure 7.

For this to be possible, we had to add control variables and interact again with the grid elements in the map (BattleGrid) to create the nighttime effect. With the time frame limit reached, the map starts being colored as at the beginning (zones and boundaries) and the agents are allowed to perform their actions normally. This happens in a permanent loop during the simulation execution.



```
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier16 is entering forbidden area |||
||| Soldier6 is entering forbidden area |||
Soldier9: I'm injured! My current health is: 5/20
     I have 0 bullet(s) left!
---> Commander1 attacks Soldier6
Soldier6 HAS DIED!
RIP Commander1
Alliance # 2 has just lost the battle.
Restarting...
```

Figure 7: BattleField at night. Winning alliance is colored orange, everything is frozen

5. Discussions and Conclusions

Our discussions/reflections and conclusions can be summarized with the following points:

- The best strategy to manage FIPA messages with different contents and purposes all at once is to give each agent a reflex dedicated to receiving messages. Once a message has been detected in your preferred list (in our case, *informs*), extract it from the default program list and save it in your own list. Once extracted, the message will be automatically deleted from the default program list, cleaning the list and avoiding message repetition. Your new local list can be managed however and whenever you want.
- Using **queues** is extremely important when managing large amounts of FIPA messages, especially if you have a central strategist, like our Commanders.
- GAMA's global variable monitoring is extremely useful when dealing with simulations with many different agents. Having a simple summary of what the state of your simulation is gives you power to make better decisions (if inputs are required) and makes it much easier to troubleshoot once clashes and bugs are encountered.
- As with previous laboratories, it is extremely important to have the right amount of control variables to not have different agent behaviors clash with each other. This was the case here as well, since agents could receive different orders at the same time,

while having their behaviors modified by their own state variables, while also needing to communicate their state to others for them to adapt to the situation.

- GAMA has many limitations and quirks that make its use difficult to manage sometimes, but it is definitely a good platform for basic Artificial Intelligence programming. With GAMA's tools you can clearly see how each agent interacts with the other, how they can crash and bug out, what each of their control variables are at that point in time, how their preferences change, and much more. What we built in this project is basically the skeleton for a strategy or board game, and GAMA could handle it without issues. Having two users decide what each Commander does instead of it doing everything itself would be a great way of testing two human's fast decision-making capabilities.