CAP4053 Spring 2024 Robocode
Team Post Mortem
Group 6: Doombas

Bots:
BFG9000 (Eddy Rosales),
Harvester (Kyle Lund)
GuardianOfTom (Elli Ludwin),
LordOfHeresy (German Dominguez)

---

1. Team Strategy & Design

We capitalized on the “black box” nature of this team development cycle, which prevented us from directly sharing code of our individual bots, in order to ensure our different data structures and algorithms would result in bots with diverse behaviors. Our primary goal was for none of our individual bots to share the same tactics or styles of play, and that our black box would lead to advantageous teamwork, less predictable behaviors, a reduced chance of overlapping weaknesses, and make our team more difficult for opponents to counter.

We all selected the different data structures and behaviors that we wanted to implement in order to inform our overall individual and team design process. Although many of our final implementations did not incorporate these exact data structures, for example Kyle reverted from Q-learning to an easier to implement combination of FSM and behavior trees, all our resulting structures were vastly different.

Our individual bots implement a variety of differing designs: a Behavior Tree implementation (Eddy Rosales), a single Finite-State-Machine***** (German Dominguez), and a combined double Finite-State-Machine with a simplified Behavior Tree (Kyle Lund).

***** Some hand-holding was needed during testing to optimize the FSM and its initial Search state.

In order to mesh these differing, independent robots together, we decided early on to implement some sort of message broadcasting and coordination to gain teammate information and coordinate simple attacks such as **double teaming** on a target.

In short, our main team strategies were:

1) **Be different and diverse in behavior.**
2) **Communicate basic friendly and enemy information for coordination.**

2. Architecture

Our teamwork architecture focused primarily on information sharing. We settled on this approach for a wide variety of reasons. The primary one is that each bot was initially developed independently, in order to ensure that it could succeed in its duel against failBot. This naturally resulted in each of us developing and using differing data structures and data storage formats according to what fit best into the bot's design. Additionally, the behaviors programmed into each of the bots required different information and used the information it did have in different ways. Because of this divergence, we focused on passing standardized informational messages between us and allowed each bot to use this data to best achieve its and the team's goals.

Because of the differences in data structures and required information between each of the bots, we were required to create message structures which pushed the maximum amount of useful information. For messages whose purpose was to inform the rest of the team about a bot's current status; information such as energy level, current position, heading, and velocity were passed. We also passed targeting information whenever a hostile bot is detected which includes most of the information present in the scanned robot event. Providing this amount of information allows each bot to respond to the messages as needed. In the case of the friendly bot status message, that includes behaviors such as not taking a shot that is likely to hit a friendly bot, and moving towards and assisting friendly bots with low energy. For the targeting information message, it allows bots to improve their situational awareness of the battlespace, and also reprioritize targets, even if the bot was not previously aware of it. The combination of these two messages allows additional emergent behavior, such as prioritizing targets which are close to low health friendlies.

3. Emergent Behavior

Our first test battles together were to face our bots against each other. There was not one clear winner but it was obvious from watching the game play that our bots were vastly different in movement, attack strategy, and defensive maneuvers. For instance, we noticed BFG9000 was more of a sneaky sniper, while LordOfHeresy was eager to ram and moved constantly, and Harvester was somewhere in between calculating carefully when to create distance and when to hone in.

Our next battles had us working together in a team against four failbots, and it was obvious that somewhere in our code there had to be a check for teammates when scanning any robot event. Luckily, this was fixed with a simple if statement that could easily be placed somewhere in our scanned robot logic:

```
// Check if the scanned bot is a teammate
if (!IsTeammate(scannedBot.Name))
      // Do some sort of attacks.
```

During team battles against a team of four Failbots, our individual bots, which performed well against Failbot in solo scenarios, exhibited several emergent behaviors when grouped. While our bots were initially fine-tuned for individual confrontations, the battlefield complexity increased when the bots were grouped into teams. This new dynamic introduced a variety of scenarios that our initial programming did not anticipate.

3a. Testing

Observing interactions with the SharpSocket team was crucial as it helped us identify distinct strategies and patterns. Initial testing uncovered issues such as bots colliding with each other, causing damage, and several movement patterns that resulted in loops or other undesirable behavior. There were also instances where team bots accidentally fired at each other when caught in the line of sight. Debugging and console logging proved essential, ensuring that our messaging system was implemented correctly and that battle data could be effectively and accurately exchanged between team bots.

3b. Adjustments and Rework

To foster a cohesive team dynamic, we implemented several key methods: '*IsTeammate*', *'OnMessageReceived'*, '*SendMessage*' and *'BroadcastMessage'*. These methods enhanced our bots' ability to communicate effectively with each other and acknowledge each other as a team. Establishing proper and matching communication protocols and message structures was a significant step, allowing for the accurate sharing of battlefield intelligence amongst the team. We also made corrections to the pathing and interaction scenarios in the individual bot code to eliminate undesirable behaviors. With improved battlefield intelligence, we could adjust the individual bot code to transition from acting as solitary attackers to engaging in coordinated efforts. These adjustments included tacts like shifting lines of sight to avoid friendly fire and strategically supporting teammates.

4. Successes & Failures

One major impediment to building an effective team of battle bots was how long it took for each of us to build our individual bots, though this was a natural and expected outcome due to our grades being dependent on our bot's individual performance in the failbot duel. Our initial meetings involved discussions of possible roles that our bots could fill, and how they would work together as a team. But until they were built we had no way of verifying which roles our bots would be good at, limiting how effective any sort of plan for building a team strategy could be. Additionally, most individual bots did not reach a state where their performance was good enough against the failBot to focus on team performance until a few days before the assignment was due.

Another issue that we encountered was that while we were able to have initial discussions about what sort of messages should be passed back and forth between our bots, we weren't knowledgeable enough about roboCode to accurately evaluate what information would be beneficial to pass and how our bots could best utilize that information. Without previous experience in roboCode, we were very reliant on two primary sources of information. The roboCode wiki, which mostly lists what sort of behaviors have been used previously, and was extremely helpful in that regard, but it generally does go into detail about which methods are more effective than other methods, particularly in the context of team battle. The other source, which we probably relied on more than we should have, was guess and check. Implementing something and then determining how effective it was. However, this is an exceptionally slow method considering how many possible ways of accomplishing even basic actions there are. To make matters worse, coding momentum is a real danger. When

enough time and code has gone into a specific way of doing things, it is usually easier to simply iterate on it in the hope that it improves, rather than scrap it entirely and try an entirely different approach.

To overcome these obstacles, we focused our teamwork on information sharing, rather than hard coded interactions. This allowed us both the opportunity to maximize the effectiveness of our individual bots, as well as granting our bots the maximum amount of flexibility to operate independently at a time and place of their choosing, with the goal of maximizing their effectiveness in battle. Another important activity of ours was to build an extensive knowledge repository. While we were conducting our initial research and coding our solutions, we shared links to useful roboCode resources that we came across, as well as a well-made roboCode tutorial video series. We also had a section for information needs, where we could let the rest of the group know that we had a question or lack of thorough understanding about a topic, and the rest of us would attempt to provide an answer to this question. This collection was invaluable as we slowly learned how to build better and better bots.

5. Individual Reflections

a. **Eddy Rosales:** Working on this project proved to be a challenging yet enriching experience, made smoother through effective team dynamics. Collaborating within the team not only clarified complex aspects of the project but also sparked innovative ideas that would have been difficult to conceive conceptually. Coordinating regular team meetings and maintaining open discussions were instrumental in advancing the project collectively and achieving our goals. This assignment provided a valuable learning opportunity with developing complex data structures and discussing them within the team context greatly enhanced my understanding and solidified the concepts.

b. **Kyle Lund:** It's commonly said that computer science is just a variant of applied mathematics, and I must admit that I have been fairly dismissive of this definition in the past. But this project has been far and away the most unexpectedly math-heavy project that I've been involved with. My trigonometry was certainly rusty, and this project has been good practice for knocking off that rust. This project also provided a surprisingly good playground for implementing the various behaviors and data structures that were discussed in the course. More importantly, it was extremely

helpful to be able to see the advantages and disadvantages of the various techniques we have learned in real time, and experience how they fit together in practice.

c. **Elli Ludwin:** Working on this project proved to be more challenging than anticipated due to my underlying personal circumstances as I have welcomed a newborn in the beginning of April. I believe there was plenty of time and resources provided to start and work on the project, however I was unable to complete it by the deadline due to the above mentioned reasons. Other than that, the project provided an interesting way of implementing freshly learned AI concepts such as FSM in an engaging way.
d. **German Dominguez:** Having spent two plus days simply trying to get wall avoidance working proved to me that getting a decent robot to work would be no simple task, and I think recalibrating our goals early on and simplifying the work was much needed. My teammates were very helpful, creative, and eager to share what information they could about their own implementations and that motivated and informed my own ideas. If I were to do this challenge again I think I would take a note from Kyle's implementation of a "BlackBoard" to keep track of world state information, friendly bot data, and most importantly overall enemy bot data. His bot's knowledge of the world state and its ability to pattern match was truly impressive and I think made the later endeavor of incorporating team coordination much easier and more practical.

6. Team Theme Song

Oompa loompa doompety dee
Here come the Doombas, listen and see
Oompa loompa doompety doo
With metallic might, they're coming for you

Doombas, Doombas, their name rings loud
In the arena, they're ever so proud
Harvester, LordOfHeresy, and BFG9000
Watch out, opponents, for the Doombas' might

Harvester roams, seeking to mow

Gathering energy, ready to go
LordOfHeresy, a strategic mind
With tactics so cunning, victory they'll find

BFG9000, a force to be reckoned
In the heat of battle, they're truly second to none
Together they stand, in the arena they thrive
Their opponents tremble, knowing they'll survive

Oompa loompa doompety dah
If you're not careful, you won't get far
The Doombas are coming, you better take heed
For in the battle of robots, it's them who succeed!