Due:

09/06/23

Assignment 3

Battle Mechs

thomas peters - 10810251

# Statement of completion

I Thomas Peters have completed the related assignment solely on my own. All related and modified code supplied has been written by myself or supplied by the university.

This includes:

* The related behavior tree’s (single file of 3, complete trees).
* Memory functionality.
* And any additional modifications supplied in a single file format.



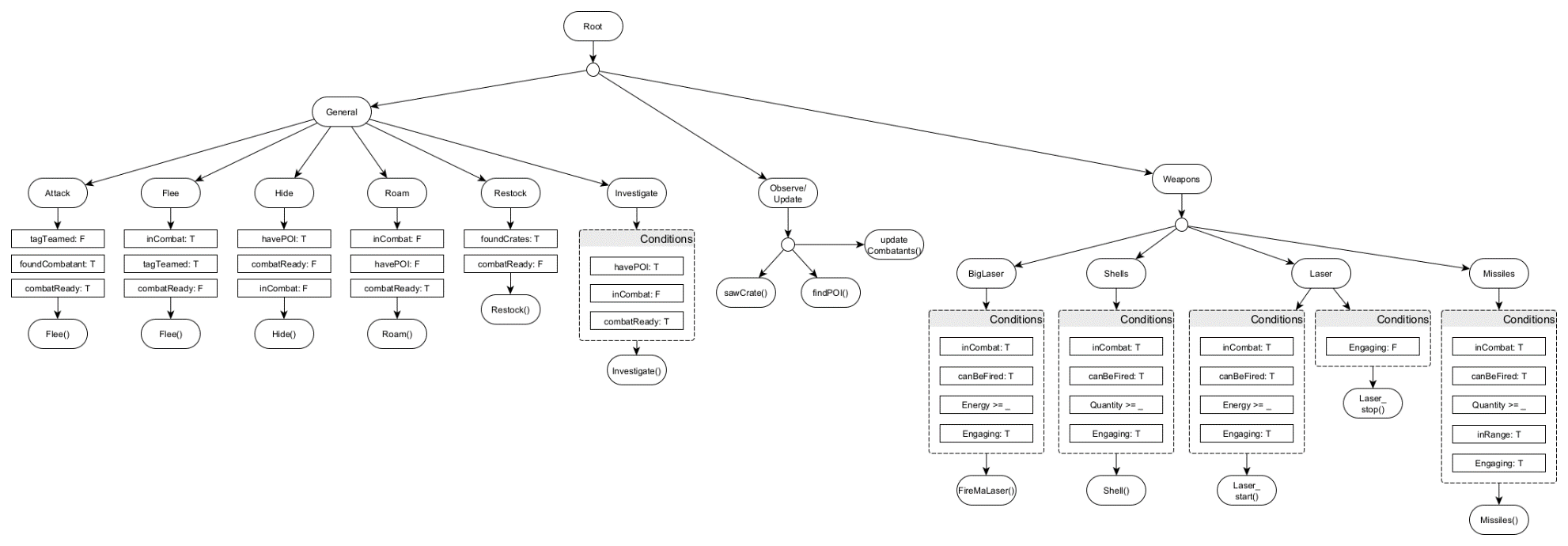
# Behavior Tree

The original concept for the bot included the capacity to run while fleeing, however, due to the restrictions implemented on the submission this feature along with the ability to hide was removed when rescoping. Any related information to the unimplemented functionality of hide remains available within the code.

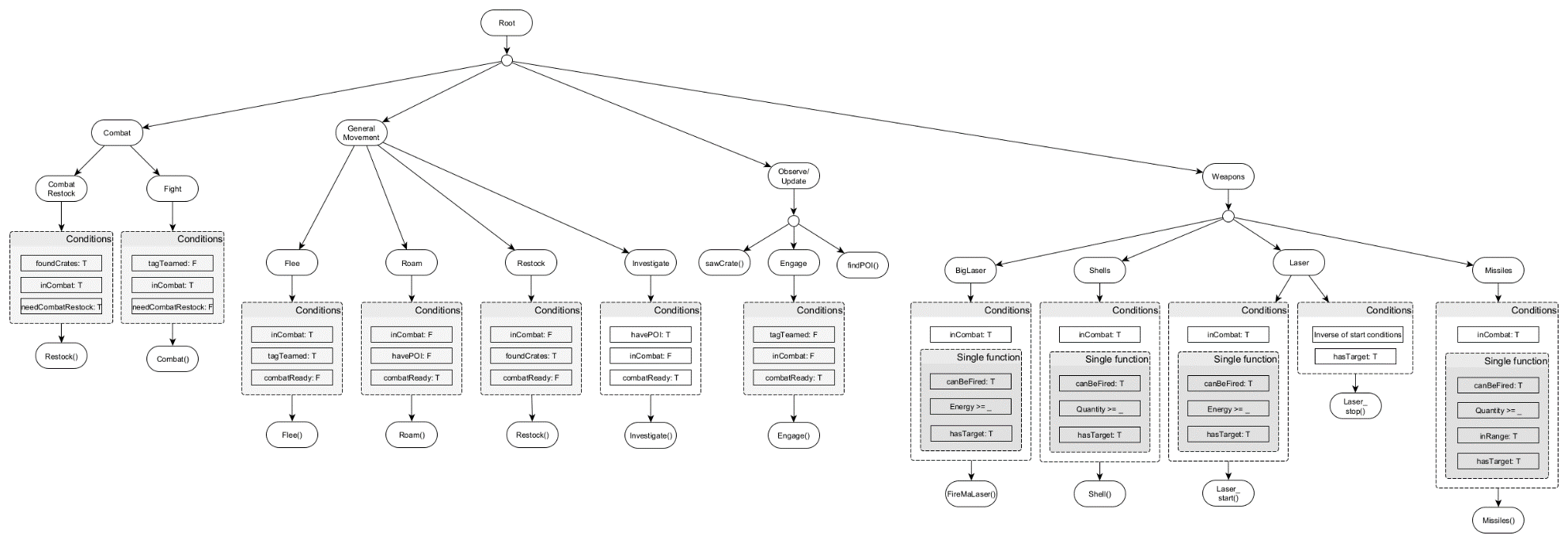
The supplied behavior tree comes in a horizontal format where the tree itself is notably shallow. The reason for this is some of the unorthodox methods of implementing two features mentioned in the following section. This is due to the mechanics behind the AI’s being more complex than that of the current behavioral model as several concepts were cut short due to accessibility such as dynamic hiding, a concept brought up in Assignment 1 using raycasts to identify an environmental object capable of blocking line of sight (LOS) with an intended target and then applying an inverse raycast to identify an accessible point on the opposing side. Which upon implementation caused numerous errors surrounding the lack of viable filtering which without, meant brute forcing the operation resulting in a O(n) time running frequently.

Instead, the AI focuses more on available knowledge than supplied knowledge.

= A Parallel operation



*Version 2 of the initial tree*

**

*Final version of the supplied tree.*

Originally I intended to implement all observational functions within Update() however, I quickly reconsidered implementing each observational function tied to the scene into the tree as it would provide a streamlined way to modify the operations behavioral conditions without recompiling or modifying the code directly. This proved to be beneficial in most cases with one exclusion related to Weapon operations. Unfortunately, as the weapons are now tied to the update of the tree compared to the update of the engine directly, when evaluating the conditions to fire a weapon it is possible for the current target to die between the actual firing call, and the initial check for a valid target. This miss timed incident results in the function attempting continuing with a null pointing object resulting in a MissingReference exception that couldn’t be fixed from within the script.

Whilst the behavioral tree works under most circumstances, due to its size and limitations, it could be easily replaced by a multithreaded FSM or even a GOAP modeled agent most likely without any notable impact on performance.

# Behaviors

Each included behavior is a minor modification from its predecessor with the exception of observational behaviors and restock.

Observational behaviors relate solely to any behavior related to vision. For this a modified and generalized version of the LOS code supplied within MechAIAiming was implemented. This version employs a predicate to allow for use in any section of code with a reliable output. Additionally, to simulate the knowledge a player would have, the bots employed the concept of persistent memory on a reduced format.

## Memories, POIs & Crates

Due to each bot being destroyed and instantiated when dying and respawning, any local systems for simulating knowledge would also be reset, thus defeating the point of simulating a player’s memory. To solve this issue a singleton was implemented to act as the persistent aspect of a player’s memory, albeit an unorthodox implementation. Points of Interest or in the case of the game, footsteps, remained local due to their short lifespan making it unrealistic to expect a player to react to following a death. Crates, however, are a different story, being the only item in the game with a predetermined spawn location and only persisting until picked up. By implementing an indexer into the singleton, accessing data unique to the mech became a simpler task.

To ensure crates were processed, LOS was used on each unknown crate within the frustum during runtime until all crates are found. Once a crate had been hit by the LOS raycast it was added to memory under KnownCrates.   
The memory of the crate included if it was present when last seen, its position, expected respawn and the last time it was observed. When attempting to restock, this allows the bot to predict if the crate has respawned.

When 1 or more crates have been found, the bot automatically begins updating its memory by applying the same visual logic to each known crate. If the crate failed the LOS check which includes a range condition to exclude an edge case where the raycast hits an opposing bot or weapon fire, the crate is marked using wasThere = false to denote its absence.

When restocking only known crates are used, and only the nearest crate with 2 seconds remaining on respawn or a crate known to exist is used.

## Restocking

Restocking functionality is used in two different nodes of the behavior tree. Specifically, the combat branch which supplies strafing and restocking during combat, and the movement tree. The reason for this is a difference in conditions when being activated where combat restocking relies on a operation that dictates if it should occur earlier than that of combatReady();



# Conclusion

In conclusion, I have learnt that I don’t like script limitations, but more importantly I have come to understand why behavior trees are preferred in development over the alternatives I am familiar with such as GOAP. Namely, it comes down to versatility and speed. Whilst GOAP is highly adaptable, the operations required are too intensive in large scale meaning GOAP has an application in smaller situations of limited size as opposed to behavior tree which are restrictive initially, but can be dynamically expanded like GOAP without the operation overhead.