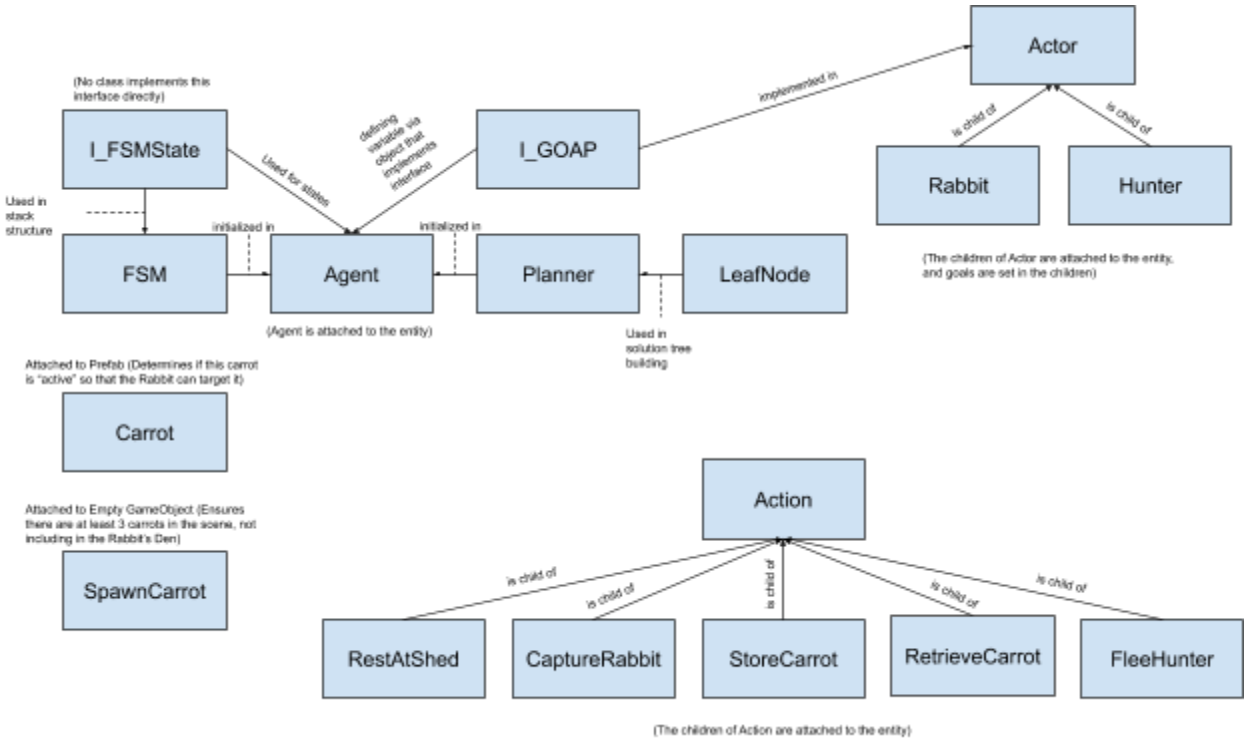
Nicholas Bazos

IGME.799.01: Advanced AI for Gameplay

Project 1 Writeup

Class Diagram



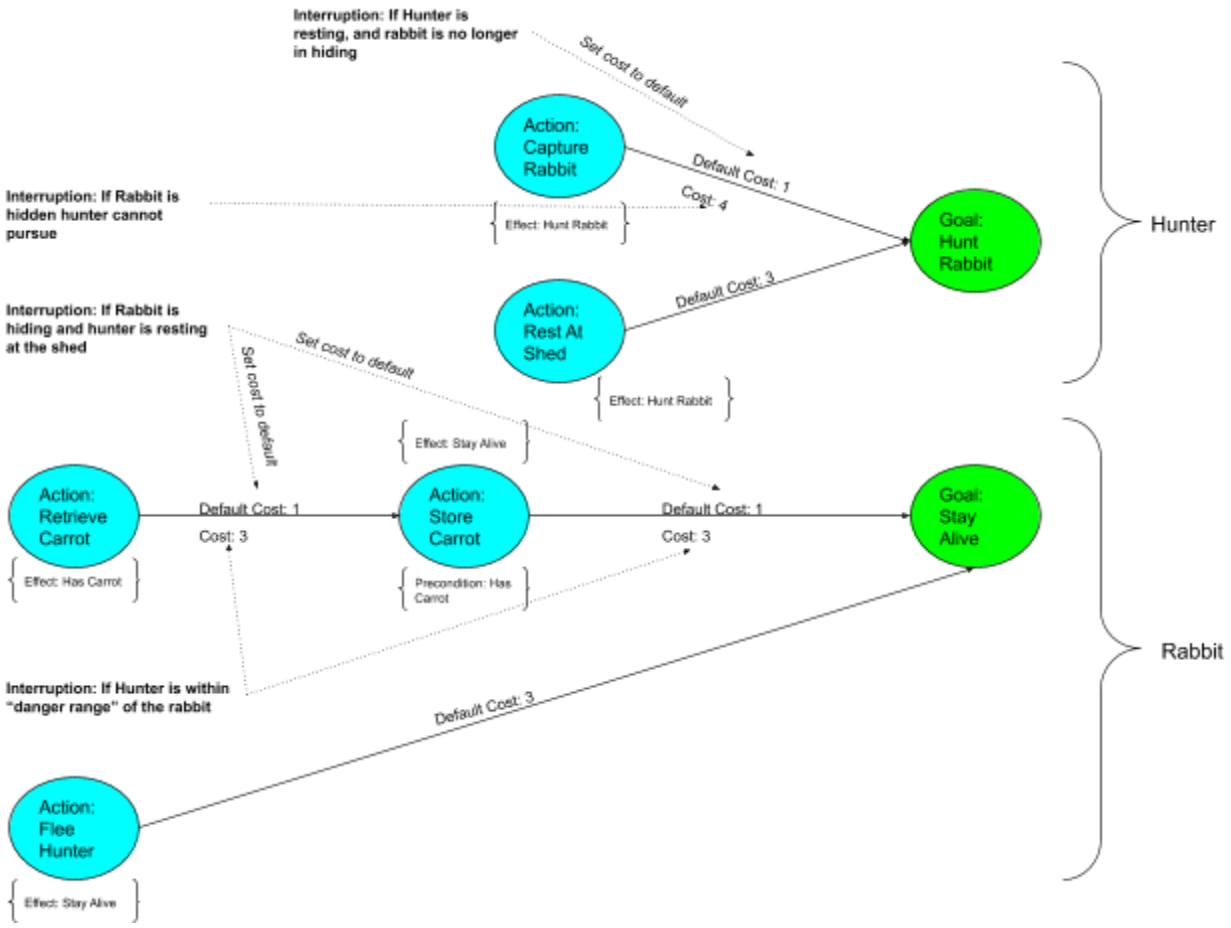
Reasoning

Based off of my research (please see "Implementation References"), I decided to go with this architecture because I felt that it had clear encapsulation of systems which would help in terms debugging and isolating technical errors or misbehavior. I also chose it because it assisted my visualization of the GOAP process that the in-game actors are going through. I think a positive aspect of this architecture is that new actions are fairly easy to make quickly and get working in the game world (as GOAP intends). I also believe that the same idea extends to making new Actors, as the only major thing that you *have* to do for the system is to set the goal (however I do anticipate things will get more complicated as Actors have more attributes and are more complex). That being said, I have definitely learned a great deal through this project and have a

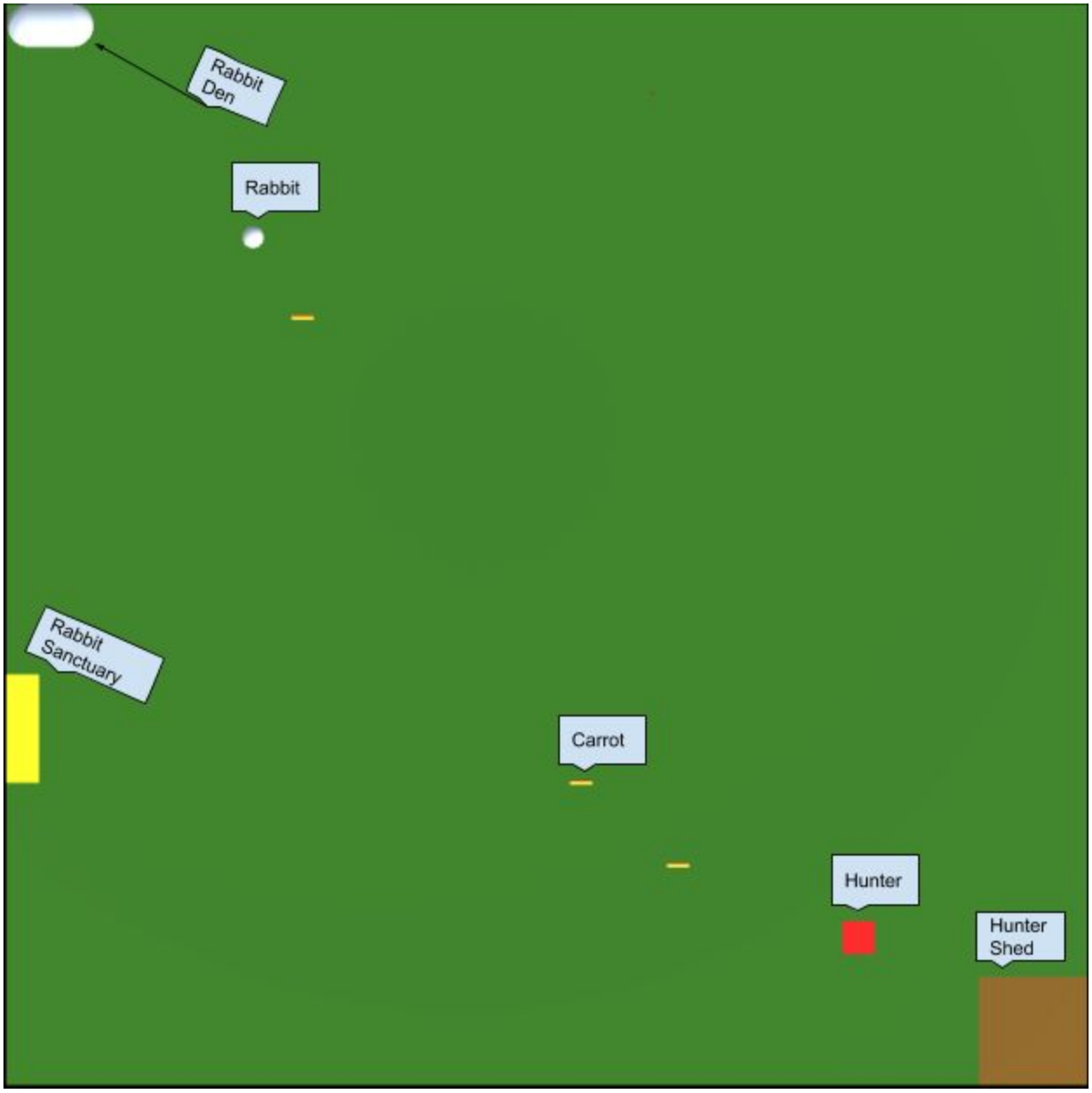
few changes I would like to make to the architecture that I feel would improve the negative elements:

- A regressive search rather than a progressive search would be more efficient/intuitive for yielding action plans based on certain goal.
- Handling interruptions in the current system is not very dynamic (as it all happens in the "Move To" state where certain conditions are checked, action costs are adjusted, and the state machine is forced into the "Idle" state where it must re-plan with the new costs). Perhaps abstracting a functionality for handling interruptions within the Agent class, or even including a method for the Actor to implement from the I_GOAP interface would be better.
- The current system can only handle one overall goal (for example, "Stay Alive") per actor with different actions serving as mini-goals towards the overall goal. This works alright for a smaller system but is really not scalable. Some sort of priority system for goals that will choose the most essential goal to pass into planning should address this issue.
- Currently the actions are not multi-usable among different actors, I have separate actions for retrieving a carrot and capturing a rabbit. However, I believe I could abstract this out to be the same action (for example, "Get Object") with different parameters passed in depending on the actor it will be attached to. I believe that following this pattern for actions would also greatly improve the scalability of the project.

AI Diagram



The rabbit will start off by the rabbit den and hunter will start off at the shed. As the hunter is not within the danger range of the rabbit, the rabbit will start collecting the closest carrots to it and will bring them to its den. However, because the rabbit is not hiding in the rabbit sanctuary the hunter will pursue it. Once the hunter is within danger range of the rabbit, carrot in mouth or not, the rabbit will flee to the rabbit sanctuary and hide from the hunter. Once the rabbit is hiding, the hunter will return to the shed to rest as it can no longer pursue the rabbit. The rabbit will wait until the hunter has returned to the shed and then either search for the next closest carrot or (if it has a carrot in its mouth) bring a carrot back to its den. The simulation stops once the hunter has caught the rabbit.



Questions

- From my research, I found that for a regressive search algorithm it may be best to use a priority queue for the state machine instead of stack structure, is this concept going in the right direction? (I need to brush up on my understanding of a priority queue as well)

- I'm curious about other architectural options that could be used instead of the I_GOAP interface to link actor goals, states, etc. to the planning.

Implementation References

- Goal Oriented Action Planning by Vedant Chaudhari
- Goal Oriented Action Planning for a Smarter AI by Brent Owens
- Finite-State Machines: Theory and Implementation by Fernando Bevilacqua
- Combat AI for Action-Adventure Games Tutorial [Unity/C#] [GOAP] by TheHappieCat
- An Introduction to Goal Orientated Action Planning by Holistic3D