Development of an Hierarchical Task Network AI in a Fighting Game

Research and Development Project
Bradley Ramsay
Supervisor: Paul Parry

Opening Statements

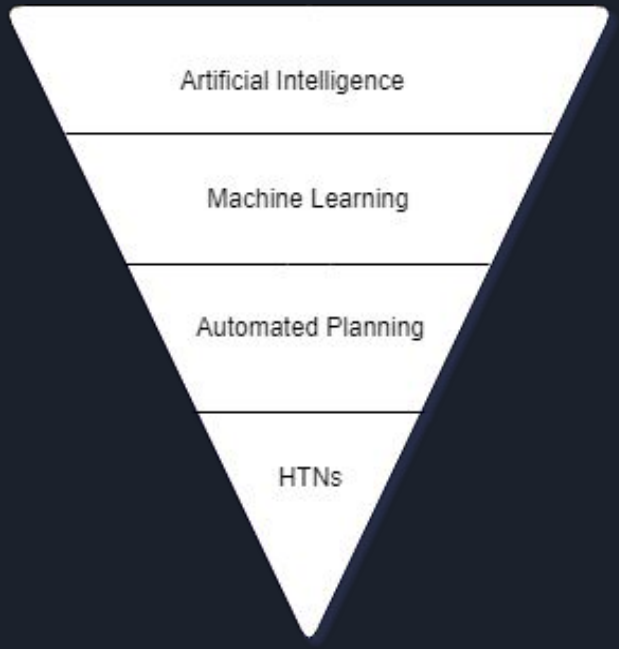
Introduction

The Problem: AI in fighting games is too predictable, making it ill-suited to be a source of entertainment or training for skilled players.

Solution: Create AIs that are well suited to providing a challenge to any player.

This is done by having the AI adapt to the opponent's strategies whilst employing strategies of their own.

Introduction



Background Context

Why This Topic?

- AI has been a staple in video games since the 70s (Pac-Man, Galaxian)
- Some implementations of it have been considered unfair, because of “cheating”
- AI in fighting games has been guilty of this problem, and has also faced criticism for being too predictable. This is because AI in fighting games tend to be created with Finite State Machines (FSMs)

Why provide Smarter AI?

Providing smarter AI can lead to increasing the longevity of a fighting game:

- because players will have an “offline” option for facing challenging opponents.
- It can entice players who prefer single player games
- If smart enough, the AI could come up with strategies that were previously unthought of by master players
- Smarter AI could open the avenue for other ways to market fighting games, through having AI competitions in much the same way that there are human fighting game competitions
- The research produced on this problem can be applicable to AI in fields outside of video games, as the core theme is improving tactical ability

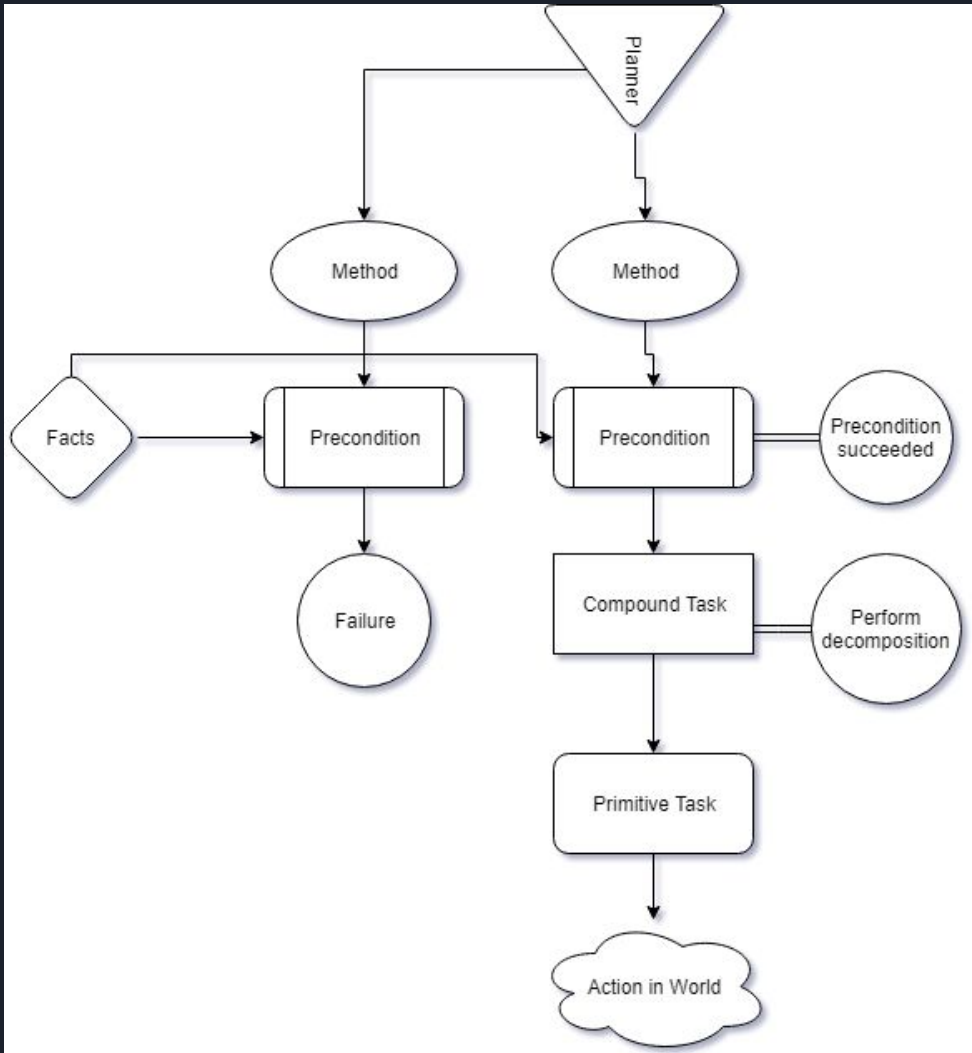
What are Hierarchical Task Networks (HTNs)?

Automated planning is a branch of AI focusing on the creation and execution of strategies or action sequences by intelligent agents, autonomous robots and unmanned vehicles.

HTNs are an approach to automated planning, where each individual action of a strategy and the dependency among actions can be represented as hierarchically structured networks

The Structure of The HTN

- Primitive Tasks,
- Compound Tasks,
- Methods,
- Preconditions,
- Simulations,
- Facts (World Space)
- Planner



Why use HTNs?

- Allows AIs to be more proactive instead of being reactive, which is ideal in creating a more human-like AI
- This is in contrast to most successful AIs developed on FightingICE being reactive in strategy
- Prior research of it used in fighting games is scarce.
- It has also been successfully implemented outside of video games, in the field of robotics

Why Fighting Games?

- Fighting Games are a very popular genre in the video game industry, as they are both fun to play and fun to watch. The numbers speak for themselves, as is the case with Street Fighter, Capcom's third-best-selling software franchise.
- Their popularity led to the creation of tournaments like the Evolution Championship Series (EVO, http://evo.shoryuken.com/) which holds tournaments for several fighting games. EVO tends to bring in an audience of millions each year (https://capcomprotour.com/street-fighter-v-evo-2017-finals-viewership-ever-fgc-event/)

Street Fighter

$10.6 bn(US)
40 m units sold

Street Fighter 2's sales numbers:
http://uk.businessinsider.com/the-11-top-grossing-video-games-of-all-time-2015-8/?r=US&IR=T/#4-world-of-warcraft-pc-2004--85-billion-8

Total units sold for the whole franchise:
http://capcom.co.jp/ir/english/finance/salesdata.html

What is FightingICE?

A Java based fighting game developed and maintained by the Intelligent Computer Entertainment Lab of Ritsumeikan University in Japan. (http://ice.ci.ritsumei.ac.jp/~ftgaic/index-1.html)

It was created with the goal to answer the following research questions:

“whether or not it is possible to realize general fighting game AIs”

“and if so how to realize them.”

General fighting game AIs would be AIs that are strong against any opponents - AIs or human players, at any play mode using any character data.

Why use FightingICE?

- It is one of the only engines developed with programmers in mind. Other similar fighting game engines, like M.U.G.E.N and zero2d, were designed for persons with little to no programming experience.
- There is a wealth of documentation available for debugging AIs, moves, health and the individual frames of each round, making it easier to collect useful data
- FightingICE relies on the same input system for both AIs and humans
- AIs developed on FightingICE can be submitted as nominations for the Computational Intelligence and Games (CIG) Competition held annually by IEEE

Past Research done with FightingICE

Several AIs have been developed on FightingICE, using methods such as:

- the Monte Carlo Tree Search algorithm,
- K-nearest neighbour algorithm,
- Q-learning techniques,
- and even visual networks.

CIG Competition

IEEE hosts a number of competitions related to AI and computational intelligence techniques. One such competition is the Fighting Game AI competition, which aims to find high skilled AI controllers by allowing users to submit AIs that would work on FightingICE

- Held annually since 2013
- Provides a set of rules that standardize the results from submissions

Objectives

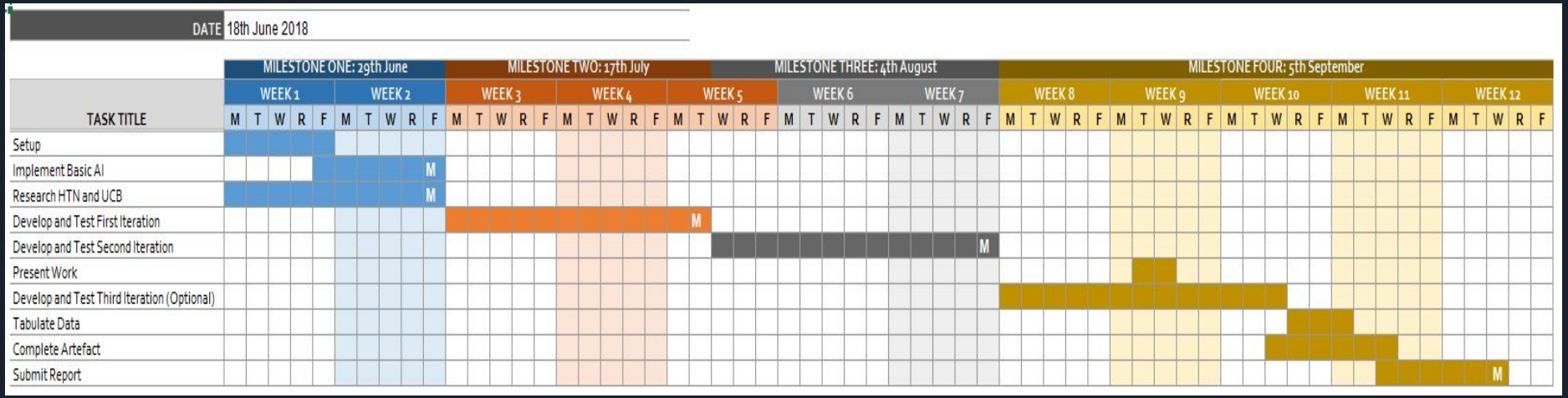
Goals

- Create an AI for the FightingICE engine, implemented with the use of Hierarchical Task Networks, and Upper Confidence Bounds as the decision policy
- The strategy the AI will follow is “Keep Away”
- Test the AI against sample AIs provided by the university. This test will follow competition rules set out by FightingICE
- Test the AI against winners of past competitions
- Analyse the results from the matches, determining (1) If the AI was more successful than its opponents (2) If it attempted to follow the strategy prescribed to it, (3) and if it did, how successful was the strategy. (4) If in situations when the strategy failed, it was able to adjust accordingly

Competition Rules

- AI Training restricted to 10 matches of 3 rounds
- The two AIs compete against each other for 100 matches of three 60 second rounds: 50 starting from the left side, 50 starting from the right
- Both AIs start a round with 400 health, and the winner is determined by who can reduce the other's health to 0 first before time is up
- The AI with the highest number of rounds won becomes the winner, with HP amounts becoming the tie-breaker in the case that there are an equal number of rounds won
- The game is played asynchronously in real-time, with a 15 frame delay on any information sent to the AIs. There are 60 frames per second, and 16.67ms per frame, meaning AIs have that much time to receive an action per frame.

Schedule



DATE	18th June 2018

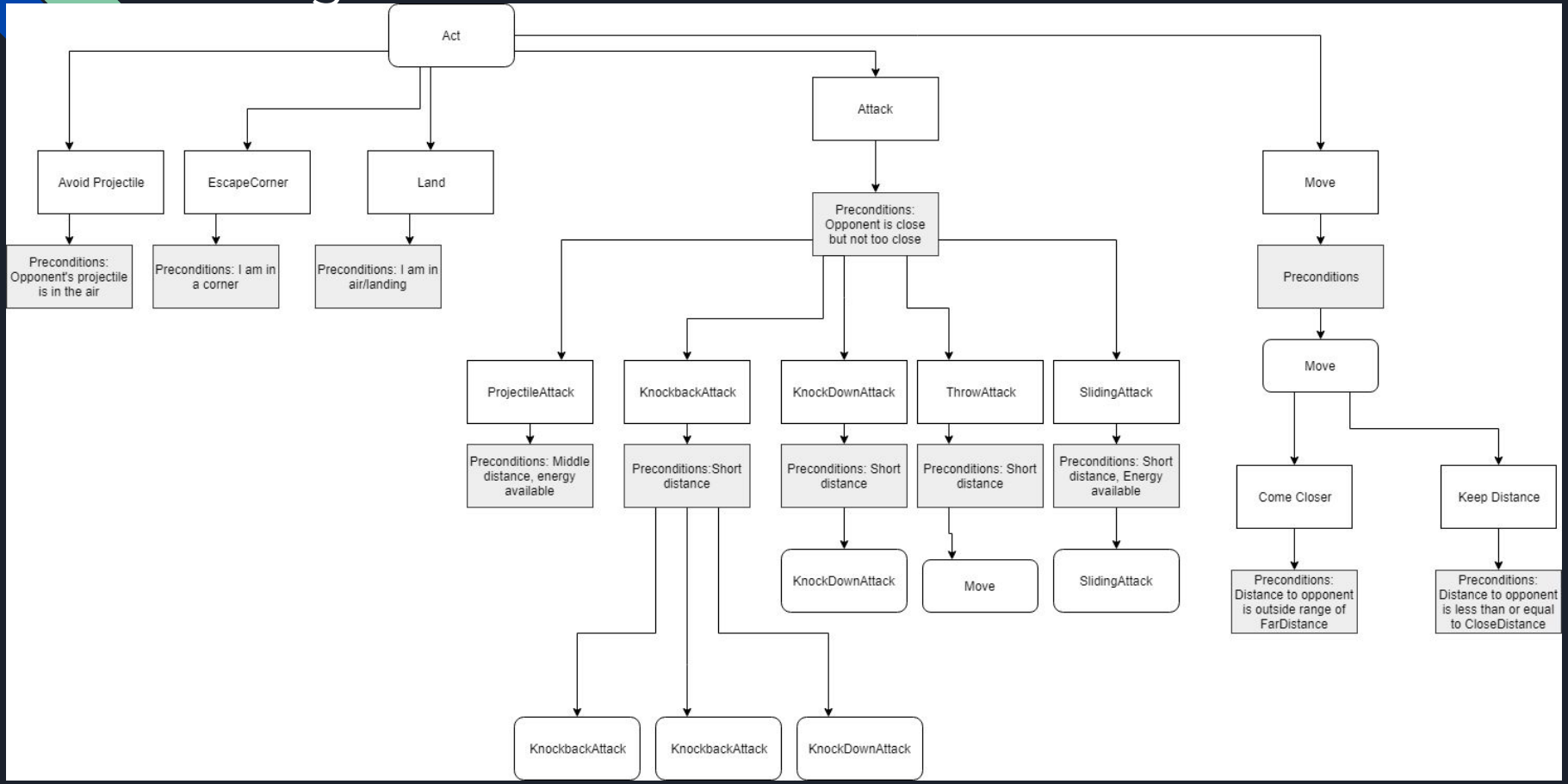
	MILESTONE ONE: 29th June										MILESTONE TWO: 17th July										MILESTONE THREE: 4th August										MILESTONE FOUR: 5th September																			
	WEEK 1					WEEK 2					WEEK 3					WEEK 4					WEEK 5					WEEK 6					WEEK 7					WEEK 8					WEEK 9					WEEK 10					WEEK 11					WEEK 12				
TASK TITLE	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
Setup																																																						
Implement Basic AI										M																																												
Research HTN and UCB										M																																												
Develop and Test First Iteration																						M																																
Develop and Test Second Iteration																																M																							
Present Work																																																						
Develop and Test Third Iteration (Optional)																																																					
Tabulate Data																																																					
Complete Artefact																																																					
Submit Report																																																					

M1: 29th June M2: 17th July M3: 4th August M4: 5th September

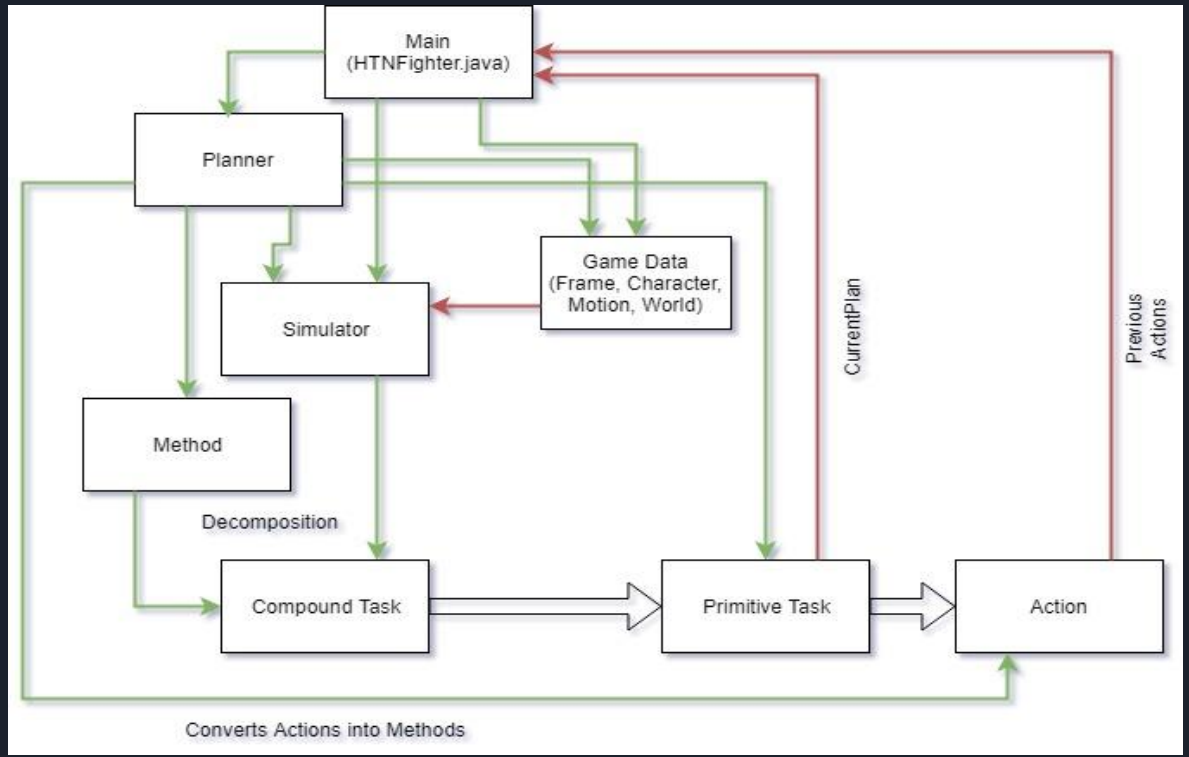
Product Design and Demonstration

Video

Diagram of the HTN



Structure of the AI



The Multi-armed bandit problem

A popular problem in probability theory that has applications in AI.

Imagine a row of slot machines that each give rewards. There is an equal cost to pulling each arm, however the reward given by each machine is random.

Given a set amount of money (say enough to pull an arm 50 times), and a choice of 5 machines, how would you distribute your money to gain the highest total reward possible, without knowing what reward each machine would give at any specific time?

What's to Come: the Upper Confidence Bound (UCB) Algorithm

A proposed solution to the multi-armed bandit problem.
This algorithm balances the concepts of exploration and exploitation when it comes to machine learning.

When a learner is faced with several actions in an unknown environment, approach the actions with as optimistic an expectation as is plausible.

Because when acting optimistically one of two things will happen: either the optimism is justified (meaning an optimal reward for the learner), or the optimism was not justified.
http://banditalgs.com/2016/09/18/the-upper-confidence-bound-algorithm/

Optimism in the face of uncertainty

Step 1: At each round ***n***, we consider two values for each action ***a***:

- $N_a(n)$ - the number of times action ***a*** was selected up until round ***n***,
- $R_a(n)$ - the sum of rewards of the action a up to round ***n***,

Step 2: From these two values calculate:

- The average reward of action ***a*** up to round ***n***

$$\overline{r_a}(n) = \frac{R_a(n)}{N_a(n)}$$

UCBs

- The confidence interval at round ***n***:

$$[\overline{r_a}(n) - \triangle_a(n), \overline{r_a}(n) + \triangle_a(n)]$$

Where delta ***a*** at round ***n*** is the degree of change from the average reward that is expected. These two confidence intervals are the confidence bounds. Delta ***a*** at round ***n*** is represented as:

$$\triangle_a(n) = \sqrt{\frac{3}{2}\frac{\log(n)}{N_a(n)}}$$

Step 3: Select the action ***a*** that has the maximum UCB: $\overline{r_a}(n) + \triangle_a(n)$

UCBs

Confidence bounds show with a high degree of certainty, the range of values that the true average reward of an action can be found in. The more times an action ***a*** is chosen, the more data we get on the expected value of the reward. It also makes the confidence bound for that action smaller. If the chosen action returns a reward, then its value increases while the interval decreases, and vice versa if it does not return a reward.

Thus with enough observations on each possible action, we can get a good idea of which action has the highest upper confidence bound, and the greatest likelihood of giving the highest reward.

By finding the action with the highest UCB, we can exploit that action to get the best reward.

UCB as applied to HTNs

While the strategy should largely remain the same (win via keeping the opponent away from the player while still dealing damage) the individual tasks used to do this can change depending on the UCBs for each task

AI Training will be used to assist in this process, by using competition rules for training: 10 matches of 3 60 second rounds against the sample AI provided by FightingICE.

Closing Statement

Conclusion

After conducting this research:

- More insight should be gained on HTNs as an application of AI in video games
- We can see how different strategies and decision policies affect an HTN
- We will be closer to having a general, more proactive AI (if it works), or closer to understanding what limits us from creating such (if it doesn't work)
- The techniques employed on this AI may also be applicable to fields other than gaming, such as robotics

Thank you for listening!