Decision making AI in Games

Callum Burley

BSc (Hons) Computer Games Development

Project: A turn-based game with two different AI's fighting each other

Supervisor: Nick Mitchell

21 November 13

*Abstract*

*This paper will start by discussing the hierarchical task network, goal oriented action planning and finite state machine methods for making believable AI in video games. To do this there will be a description of each and their advantages and disadvantages. There will also be a brief overview of the first planning algorithm STRIPS. From this a conclusion will be drawn about why* hierarchical task network *is better than goal oriented action planning and finite state machines for developing believable AI.*

## . Introduction

### Context

Decision making is important when creating an artificial intelligent for a game agent or NPC so that the player considers them believable and perceives them to be acting in a human-like manner as predictably is boring. As Johann Alvarez says “We as humans have a natural instinct to compete, so when something becomes too easy to predict for someone, it becomes too easy to defeat and we lose the drive to fight against it” (Alvarez, 2013).

There are many ways that developers use in order to create a game agent that seems to have human-like decision making skills. The most commonly used is finite state machines but it is not the best option to do so, “There is no doubt that FSM can easily produce challenging game opponent with good intelligence, however, it does not always provide the best solution”( Pathak, Zamani, Shah, 2013). GOAP and HTN as planning systems allow the agents to control the decision making and unlike GOAP, HTN is not as computationally expensive.

### Overview

When creating a video game the game agents usually need to do some sort of decision making to allow them to provide a challenge to the player. There are three methods that will be analysed in this paper each with their own section, in order they are finite state machines, goal oriented action planning and hierarchical task networks. Each of these will be discussed in sections 2 through 4 along with their advantages and disadvantages. In section 5 there will be an overview of what STRIPS is so that we can see what both HTN and GOAP were based on and to make it easier to see why they are better. After this there will be a conclusion that will be made on which I believe is better suited to create a believable AI.

## . Finite State Machines

### What is a finite state machine (FSM)?

A finite state machine consists of a finite number of states, when certain conditions are met, depending on what the current state is it will transition to another state. The only state that the system needs to know about is the current state. Because you can have any number of states you can create it can track a lot of transitions. You only need a state for each of the possible transitions because of this “it has been argued that it is a sufficient model of human behaviour i.e. humans are finite state machines” (James, N.A). a finite state machine has four parts the states that are used to describe the agents behaviour, the transition that allow for the agent to change states, the rules that say when a state may change to another and the input from the world or user that can trigger state transitions or have the agent check the rules within the current state. Since the method of creating a finite state machine is procedural you have to tell the system exactly what to do in a given circumstances. Finite state machines originated in computational theory and mathematical models to support the fields in bioscience. It was George H.Mealy and Edward F.Moore that contributed to the popularity in computer science and engineering. They did this by proving that finite state machines concepts are valuable in two important engineering disciplines. The first was language parsing or compliers and sequential circuit design. A finite state machine is usually shown as a state diagram with a state circle/bubble and the transition arrows connecting the states together.

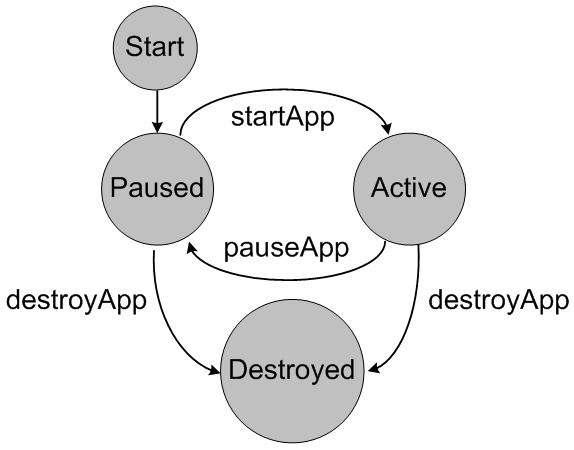


Figure 2.1.1 shows an example of FSM.

(Ortiz, 2004).

### The advantages and disadvantages of finite state machines.

## A downside to these is that the transitions from one state to another are rigid. Finite state machines are helpful as they can be easily tested since the behaviour of them is predictable and the states, transitions and rules have to be known before it can be created. At the same time though this can be a disadvantage as predictability is not fun or engaging and can take away for the believability of them. On the other hand since they are flexible and there are many ways to implement them, they are also quick to design, implement and in their execution which is beneficial as the time a game has to execute the AI code is very small so the more time that can be saved the better. The knowledge that most programmers have does not apply to finite state machine “a surprisingly small amount of mainstream programming knowledge applies to finite state machines.”( Champandard, 2007). This is because each state is required to be wired to the others with an explicit transition that no programming language requires because everything is done implicitly since the complier builds the sequences from statements. Game developers also have to often use extensions to make finite state machines useful in practise and these are not as easy to understand or well documented unlike the academic foundations of finite state machines. It is also hard to re-use finite state machines in other games or in different parts of an engine as they need to be change to deal with every case of a problem.

## . Goal Oriented Action Planning

### What is goal oriented action planning (GOAP)?

At the simplest level goal oriented action planning is made of two parts the goal and the actions to achieve that goal. The goal is the desired state of the world to the agent, actions are the methods needed to accomplish that goal. Each action will have preconditions in order for that task to be executed these will need to have been successful. A way of creating goal oriented action planning is to use the fundamentals of the A\* path finding algorithm, which uses a heuristic and a cost to create the path to a goal and orders them be ascending cost resulting (hopefully) in the lowest cost route being taken. The algorithm also allows for back tracking if it finds itself at a dead-end or going down a path that does not result in the desired goal. Monolith’s game F.E.A.R. was one of the first games to use goal oriented action planning in 2005.

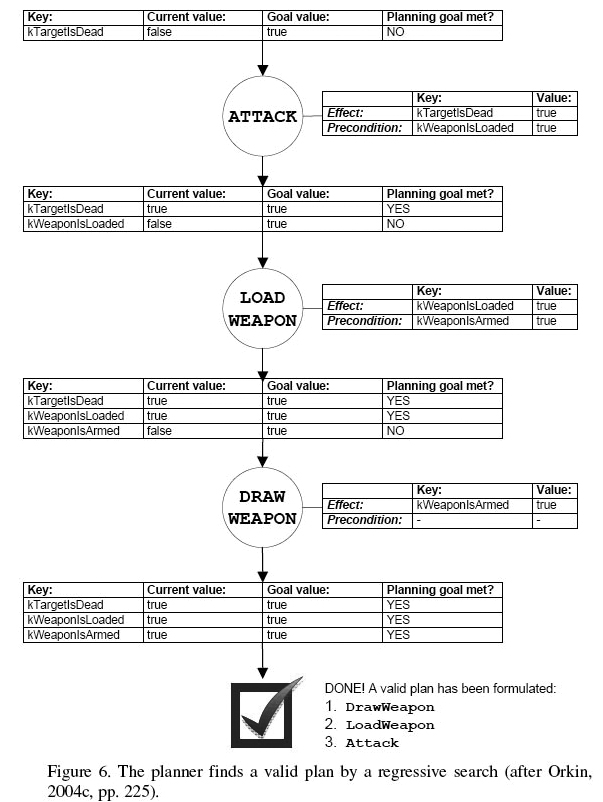


Figure 3.1.1 shows an example of a GOAP.

(Hardwidge, 2009).

### The advantages and disadvantages of goal oriented action planning.

Goal oriented action planning allows game agents to dynamically plan and re-plan their action depending on how the game world changes. “A character that determines his own plan at runtime can custom fit his problems to his current surroundings, and dynamically finds alternate solutions to problems” (Orkin, 2003). The problem with using a planning system like this is that we need a fully determined world to make planning feasible. This is not always the case as even in game systems there can be unpredictable behaviour and this can cause the planning system to make mistakes, though, since humans make are known for making mistakes, can add to the believability of an AI. Another disadvantage is that the A\* algorithm although it is likely to find the best solution to a problem. It also requires a lot of memory due to needing to keep all the possible routes open until it reaches the end of that route while extending that and others. goal oriented action planning can also help with the development process as the goal and actions defined in it can be used multiple times with one definition making it more efficient during development.

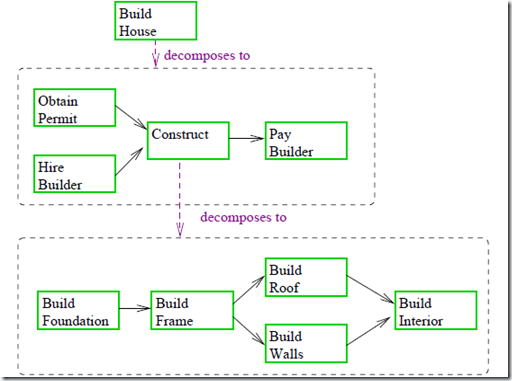
## . Hierarchical task network

### What is a hierarchical task network (HTN)?

A hierarchical task network, unlike goal oriented action planning, uses a set of tasks, with the constraints on the tasks, to create a plan. It does this be repeatedly decomposing the tasks until they are primitive. The way to do this is to recursively call a method that receives a task plan that then executes it if it is primitive or calls the method again with the sub task/s replacing the plan. The task decomposition techniques that hierarchical task network uses were introduced by Sacerdoti in the mid-70s in a system called NOAH and were later enhanced by Tate in 1977 in a system called NONLIN. NOAH represented planning problems as partially ordered task lists. It had procedurally encoded soup code on how to decompose tasks. Soup code was a collection of procedures that did the decomposition. A task network is a set of tasks that need to be executed with constraints that decide when each task will be executed. Primitive tasks are tasks that require one action to accomplish, while a non-primitive task also called an activity are tasks that cannot be executed with one action.

Figure 4.1.1 shows an example of a HTN for building a house

(Elogeel, 2010).



### The advantages and disadvantages of hierarchical task network.

The main advantage of hierarchical task network planning is the ability to reason on a high level of abstraction compared to most methods of planning. “Since hierarchical planning works in a similar way to how it is believed that human planning works” (Marthi, Russel, Wolfe, 2007) it makes it easier to implement. It does is like this because like humans it breaks down complex problems into easier to manage tasks allowing the designers/developers to design/implement it in a way that they themselves think. A disadvantage of this is that it requires, just like other planners, a deterministic environment and it cannot handle uncertain outcomes. Hierarchical task network is also more efficient that FSM as compound tasks can do something that going through a state many times would be able to do ( **Hagström** 1997).

## . What is STRIPS?

STRIPS stands for Stanford Research Institute Problem Solver and was created in the early 1970s. Though it is an old technique most modern planning algorithms have been inspired by it. It was developed by Richard Fikes and Nils Nilsson from the AI centre at SRI international. The STRIPS methods roots are from robotics. They were trying to make a system to control a robot that could solves problems in real-time with the limited hardware of their time and capable of performing task like navigating an environment and moving objects. The method consists of two major parts the first is a representation language for planning the problems which changes the world with functions when certain preconditions are met by the side effects of those functions. The second is that is uses two search algorithms. One to do the planning of when the actions should be run and the second to check if the preconditions have been met for the task in accordance to what is currently true in the world. Like goal oriented action planning, STRIPS also makes use of the A\* algorithm as it applies a heuristic to help decide what should be executed next. A full STRIPS solution is not the best for games as not many games need the complex power of the algorithm and any that does will have performance issues and because of this the less complex but more efficient ones like the hierarchical task network and goal oriented action planning algorithm is used.

## . Conclusion

This paper started by analysing finite state machines and from this is it easy to see that they have their uses but are not as helpful as planning algorithms as they require everything to be hard coded. After that goal oriented action planning was discussed and that it creates a dynamic plan of actions that can be changed, although it is computationally expensive and though likely, it is not assured to find the best solution. Then it went on the hierarchical task network analysis and from this we can see that it works in a similar way that human decision making works which adds to its believability. It also has the ability to reason on a high level of abstraction. In conclusion though both finite state machines and goal oriented action planning are useful algorithms, hierarchical task network is not as expensive as goal oriented action planning and still allows for dynamic plans to be created.

## . Relevance to project

The review is relevant to my project as I am implementing different AI’s and needed to research different types of AI algorithms to see how they are done and which are better than the other.

## . References

Machine Learning in Video Games:

The Importance of AI Logic in Gaming

Johann Alvarez

http://www.cs.fsu.edu/~cop4601p/project/students/johann-alvarez/researchpaper.pdf Accessed [13/11/2013]

International Journal of Science and Research (IJSR)

ISSN (Online): 2319-7064

Volume 2 Issue 10, October 2013

Automatic Game AI Design

Pranav kumar Pathak, Abu Sarwar Zamani and Dipthi Shah

Mark James

http://www.i-programmer.info/babbages-bag/223-finite-state-machines.html

Accessed [13/11/2013]

Gideon Maillette de Buy Wenniger

Attila Houtkooper 5 January 2008

http://staff.science.uva.nl/~gemaille/Finished\_Projects/GOAP-Report.pdf

Accessed [13/11/2013]

Shirin Sohrabi Jorge Baier Sheila A. McIlraith

http://www.cs.toronto.edu/~sheila/publications/soh-bai-mci-icaps08ws.pdf

Accessed [13/11/2013]

Abdelrahman Elogeel February 5, 2010

http://abdelrahmanogail.wordpress.com/2010/02/05/hierarchical-task-network-htn Accessed [13/11/2013]

C. Enrique Ortiz

August 2004

http://www.oracle.com/technetwork/systems/fsm-156381.html Accessed [21/11/2013]

5th March 2009

Ben Hardwidge

http://www.bit-tech.net/gaming/2009/03/05/how-ai-in-games-works/3 Accessed [21/11/2013]

TeroHagström11.12.1997

Tik-76.275 Seminar on Knowledge Engineering

http://www.cs.hut.fi/~sto/planning-seminaari/hagstrom/hierarchical-planning.html Accessed [21/11/2013]

Orkin, J. 2003

*Applying Goal-Oriented Action Planning to Games* [pdf]

http://web.media.mit.edu/~jorkin/GOAP\_draft\_AIWisdom2\_2003.pdf Accessed [21/11/2013]

## Alex J. Champandard December 28, 2007

http://aigamedev.com/open/article/fsm-age-is-over/ Accessed [21/11/2013]