SPIKE REPORT SPIKE 13

Jake Lucic

2103435 AI for Games

Number: Spike 13
Spike Title: The Planning in GOAP
Personal: Jake Lucic (2103435)

Context:

Goal Oriented Action Planning (GOAP) is an extension to simple goal insistence action planning that considered the outcome of actions and creates plans instead of simple reactions to goal insistence levels.

Knowledge/Skill Gap:

Developers need to be able to create and use a GOAP system for agent control.

Goals:
Create a GOAP simulation that demonstrates the effectiveness of the technique in considering long-term outcomes of actions (related to side-effects and/or time delays) and can plan and act intelligently.

Technologies, Tools, and Resources used:

In this task, the technologies used are listed below:

- Simple Code base, the not-yet-functional code that required fixing.
- Python IDLE v3.6.4 / Python language / PyCharm
- GOB and GOAP Lectures
- *Artificial Intelligence for Games*" Book by Ian Millington and John Fung. *Chapter 5 p412-p417*

Tasks undertaken:

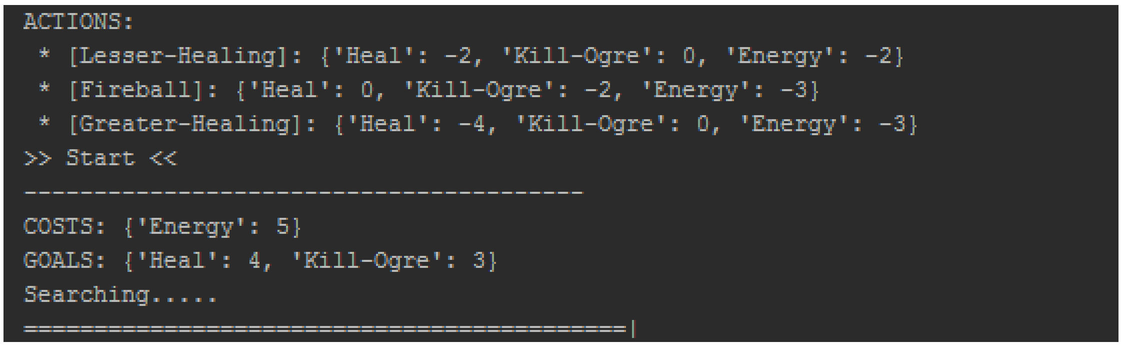
- Reassess the code base and understand what each function does
- Study the different key words and variable names
- Follow the lectures on GOAP and Read the pages in *Artificial Intelligence for Games*" Book by Ian Millington and John Fung. *Chapter 5*
- Approach Task using similar process below

Process:

Much like lab03, I found this task quite difficult to complete. However, in the end I was able to achieve the result I was looking for.

In lab03, we were instructed to make an Agent that uses simple goal insistence with the use of discontentment values to achieve its immediate need first, only looking to its next step.

It was my goal to create an Agent that could plan the best set of actions to lower its discontentment value whilst keeping an eye on its costs used. The perfect example is much like the examples shown in the lecture and in *Artificial Intelligence for Games*" Book by Ian Millington and John Fung. *Chapter 5.*



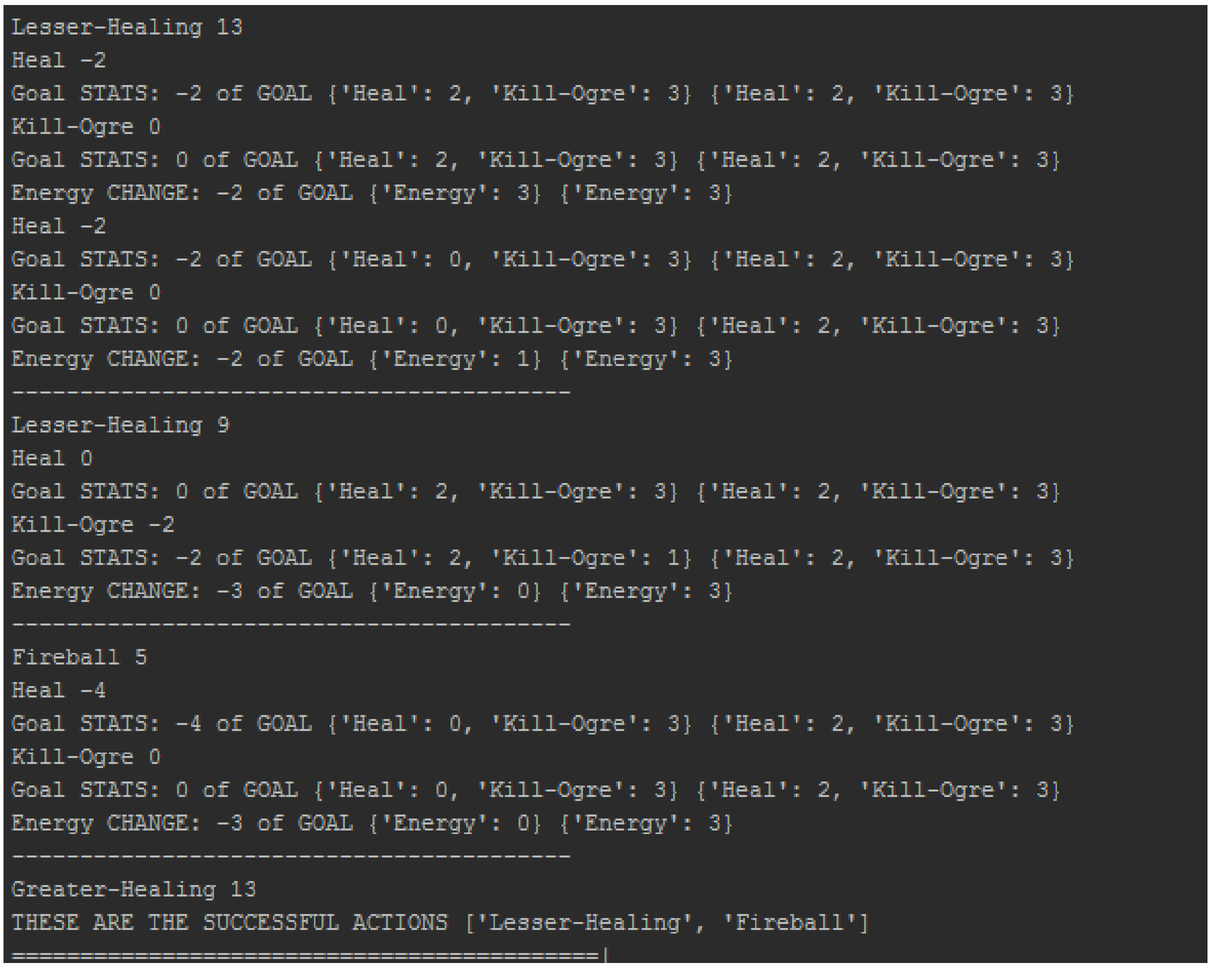
```
ACTIONS:
 * [Lesser-Healing]: {'Heal': -2, 'Kill-Ogre': 0, 'Energy': -2}
 * [Fireball]: {'Heal': 0, 'Kill-Ogre': -2, 'Energy': -3}
 * [Greater-Healing]: {'Heal': -4, 'Kill-Ogre': 0, 'Energy': -3}
>> Start <<
----------------------------------------
COSTS: {'Energy': 5}
GOALS: {'Heal': 4, 'Kill-Ogre': 3}
Searching.....
============================================|
```

As you can see here, this agent has a set of action it can do that will influence its goals with a cost. It can lesser heal to satisfy its healing goal by 2, with a cost of 2, it can also cast a fireball which will satisfy its kill-ogre goal by 2 with a cost of 3. Finally it can also cast a greater-healing with double the amount that lesser-healing does, but with a shorter cost.

If I had used this example in the lab03 code, the Agent would cast Greater-Healing, because it wanted to satisfy its *next* discontentment value. This would leave the Agent at having a discontentment of a value of 9. Kill-Ogre 3*3. Because then it could not cast a fireball, because its energy cost of casting fireball after using greater healing is 6.

Behaviour Tree

However, with GOAP, an agent is able to assess sets of combinations of actions to determine how to reduce its discontentment overall using its cost as a resource. It does this by making use of a decision tree, it evaluates an action, then steps into the hypothetical set of actions to assess its discontentment and energy costs if it were to choose another action.



```
Lesser-Healing 13
Heal -2
Goal STATS: -2 of GOAL {'Heal': 2, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Kill-Ogre 0
Goal STATS: 0 of GOAL {'Heal': 2, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Energy CHANGE: -2 of GOAL {'Energy': 3} {'Energy': 3}
Heal -2
Goal STATS: -2 of GOAL {'Heal': 0, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Kill-Ogre 0
Goal STATS: 0 of GOAL {'Heal': 0, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Energy CHANGE: -2 of GOAL {'Energy': 1} {'Energy': 3}
----------------------------------------
Lesser-Healing 9
Heal 0
Goal STATS: 0 of GOAL {'Heal': 2, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Kill-Ogre -2
Goal STATS: -2 of GOAL {'Heal': 2, 'Kill-Ogre': 1} {'Heal': 2, 'Kill-Ogre': 3}
Energy CHANGE: -3 of GOAL {'Energy': 0} {'Energy': 3}
----------------------------------------
Fireball 5
Heal -4
Goal STATS: -4 of GOAL {'Heal': 0, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Kill-Ogre 0
Goal STATS: 0 of GOAL {'Heal': 0, 'Kill-Ogre': 3} {'Heal': 2, 'Kill-Ogre': 3}
Energy CHANGE: -3 of GOAL {'Energy': 0} {'Energy': 3}
----------------------------------------
Greater-Healing 13
THESE ARE THE SUCCESSFUL ACTIONS ['Lesser-Healing', 'Fireball']
===========================================
```

In this image, you can see the first attempted action was lesser-healing. It assesses its hypothetical discontentment and energy situation stores it in a copy of its goals and cost, and then moves inside the tree (to the next leaf node) to see how its changes to the discontentment and energy would fare with making a next choice.

At the end, you are able to see the actual best path, which would be to cast Lesser-Healing and Fireball. This makes use of its energy in the most efficient way, whilst making its discontentment the lowest it can be with using 5 energy. This leaves it at 5 discontentment.