Spike: 1

Title: Goal Oriented Behaviour with Simple Goal Insistence

Author: Steven Efthimiadis, 1627406

Goals / deliverables:

Create a simulation of a GOB using SGI.

- Code that simulates and displays the GOB using SGI
    - Demonstrate where SGI works appropriately
    - Demonstrate where SGI doesn't work appropriately

Technologies, Tools, and Resources used:

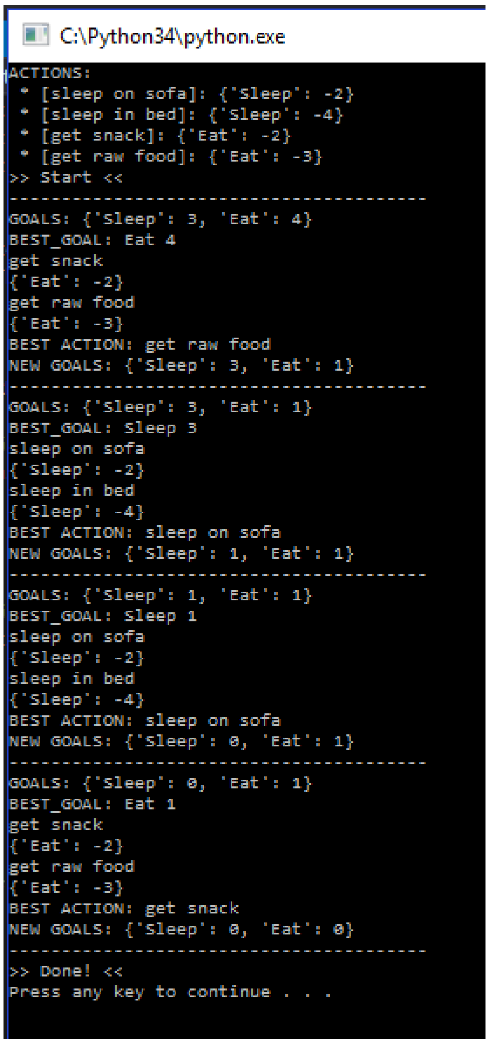
- Knowledge of python
    - https://docs.python.org/3/tutorial/
- Python Interpreter
    - Visual Studio
        - https://www.visualstudio.com/downloads/
- Knowledge of how game loops work
    - http://gameprogrammingpatterns.com/game-loop.html

Tasks undertaken:

- Added a best action before you loop though the actions for comparing each action
- Compare each action and determine which action will used
    - See which one will drop the value down by the most
    - See if the action will be equal or greater than 0
- Once an action is found complete the action

What we found out:

Good SGI



C:\Python34\python.exe

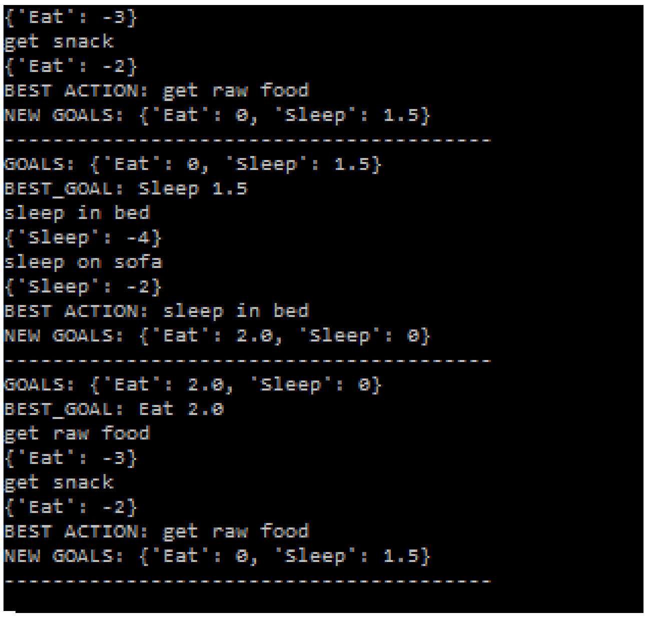
```
ACTIONS:
 * [sleep on sofa]: {'Sleep': -2}
 * [sleep in bed]: {'Sleep': -4}
 * [get snack]: {'Eat': -2}
 * [get raw food]: {'Eat': -3}
>> Start <<
----------------------------------------
GOALS: {'Sleep': 3, 'Eat': 4}
BEST_GOAL: Eat 4
get snack
{'Eat': -2}
get raw food
{'Eat': -3}
BEST ACTION: get raw food
NEW GOALS: {'Sleep': 3, 'Eat': 1}
----------------------------------------
GOALS: {'Sleep': 3, 'Eat': 1}
BEST_GOAL: Sleep 3
sleep on sofa
{'Sleep': -2}
sleep in bed
{'Sleep': -4}
BEST ACTION: sleep on sofa
NEW GOALS: {'Sleep': 1, 'Eat': 1}
----------------------------------------
GOALS: {'Sleep': 1, 'Eat': 1}
BEST_GOAL: Sleep 1
sleep on sofa
{'Sleep': -2}
sleep in bed
{'Sleep': -4}
BEST ACTION: sleep on sofa
NEW GOALS: {'Sleep': 0, 'Eat': 1}
----------------------------------------
GOALS: {'Sleep': 0, 'Eat': 1}
BEST_GOAL: Eat 1
get snack
{'Eat': -2}
get raw food
{'Eat': -3}
BEST ACTION: get snack
NEW GOALS: {'Sleep': 0, 'Eat': 0}
----------------------------------------
>> Done! <<
Press any key to continue . . .
```

If we look at figure on the left, we see that the program can determine which action is best.

Now we notice that when you sleep you should be able to sleep in bed and knock of sleep in one hit. But you can't because you need to have a value which is greater or equal to 0.

If you look at the code in Appendix (figure 1.1) we see that if the action has a value greater than the current best action's value; it will check to see if the action's value will drop your goal to under 0. If so than it's not the best action. If it's above 0 than it's the best action.

Bad SGI



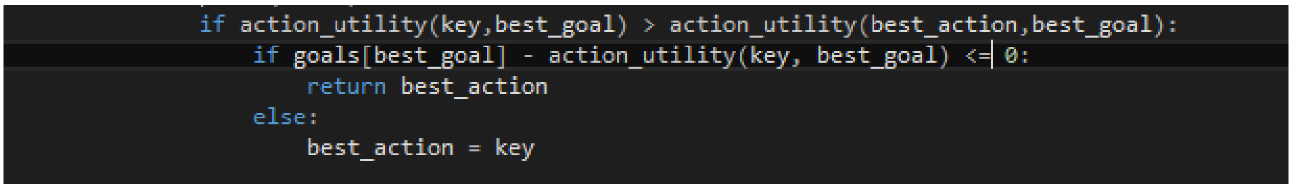
```
{'Eat': -3}
get snack
{'Eat': -2}
BEST ACTION: get raw food
NEW GOALS: {'Eat': 0, 'Sleep': 1.5}
----------------------------------------
GOALS: {'Eat': 0, 'Sleep': 1.5}
BEST_GOAL: Sleep 1.5
sleep in bed
{'Sleep': -4}
sleep on sofa
{'Sleep': -2}
BEST ACTION: sleep in bed
NEW GOALS: {'Eat': 2.0, 'Sleep': 0}
----------------------------------------
GOALS: {'Eat': 2.0, 'Sleep': 0}
BEST_GOAL: Eat 2.0
get raw food
{'Eat': -3}
get snack
{'Eat': -2}
BEST ACTION: get raw food
NEW GOALS: {'Eat': 0, 'Sleep': 1.5}
----------------------------------------
```

If you look at the picture on the left. You see that there is an endless loop which keeps repeated because as you sleep you get hungry and when you eat you get sleepy.

If you look at the code in the Appendix (figure 1.2) you see that you get half of what the change would be for the current goal added to the other goal so essentially you never completely full or tired.

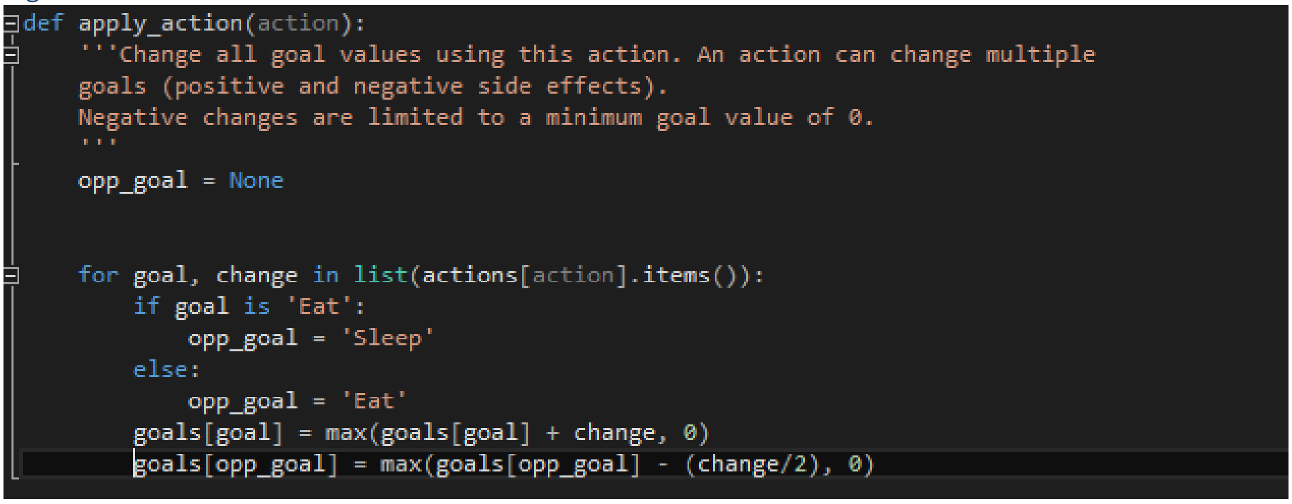
Appendix

Figure 1.1 Code for selecting best action1



```
            if action_utility(key,best_goal) > action_utility(best_action,best_goal):
                if goals[best_goal] - action_utility(key, best_goal) <= 0:
                    return best_action
                else:
                    best_action = key
```

Figure 1.2 Code for



```
def apply_action(action):
    '''Change all goal values using this action. An action can change multiple
    goals (positive and negative side effects).
    Negative changes are limited to a minimum goal value of 0.
    '''
    opp_goal = None


    for goal, change in list(actions[action].items()):
        if goal is 'Eat':
            opp_goal = 'Sleep'
        else:
            opp_goal = 'Eat'
        goals[goal] = max(goals[goal] + change, 0)
        goals[opp_goal] = max(goals[opp_goal] - (change/2), 0)
```