Utility Based AI for a First-Person Shooter

By Max Oates

The Problem

Most AI behavior can be predictable after playing the game a few times. This reduces the chances of replayability and possibly the amount of entertainment the player is experiencing over time. Research has been done using other AI techniques like reenforcement learning to solve this problem (McPartland 2013).

The Solution – Dual Utility

My solution is to create AIs that use Dual Utility Reasoning which is a technique in Utility Based AI that uses both weights and scores to affect its decisions instead of just weights (Dill 2015). This is to make them behave differently when experiencing the same changes to the environment .

The actions are all separate classes which are then stored in another class called a bucket, these buckets contain actions specific to their specialism such as health (Graham 2020).

Each bucket and action is accessed using a factory pattern which allows the enemy to check its utility. It may pick the action with the lowest score or pick one at random (Graham 2020). Figure 2 & 3 shows every child class and the base class it inherits from. Figure 1 shows how the decision-making process works when finding the smallest utilities.

Practice Based Research

I experimented with four different ways for the AI to make decisions:

- picking both the smallest score and weight.
- picking both the weight and score randomly.
- picking the weight at random and the smallest score.
- picking the smallest weight and a random score.

I also experimented with different equations to calculate the weights:

- Linear equation.
- Quadratic equation.

Picking the smallest weight and a random score shows the best results, in terms of making it more unpredictable, but its slow at making decisions and would make stupid ones at times, such as trying to heal when at maximum health. Picking the smallest weight and score has shown to be very good and faster at making decisions but is more predictable.

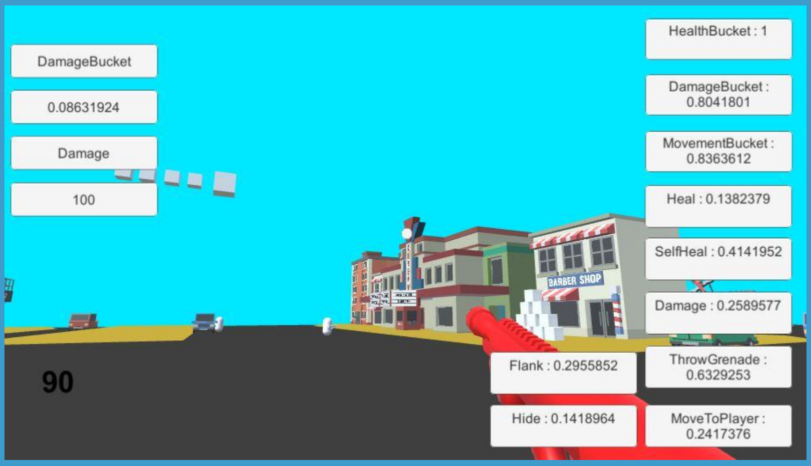

Fig4: Oates, Max. 2020. Example of gameplay

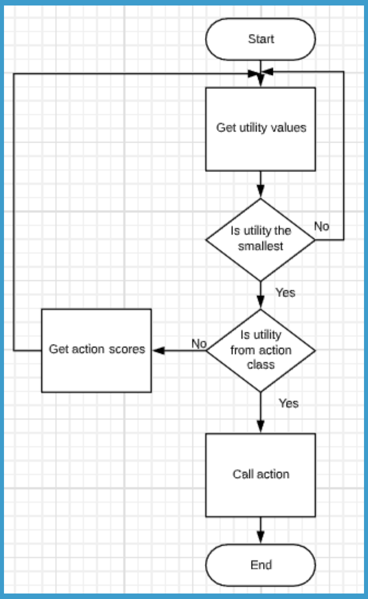

Fig1: Oates, Max. 2020. Decision making Flow chart

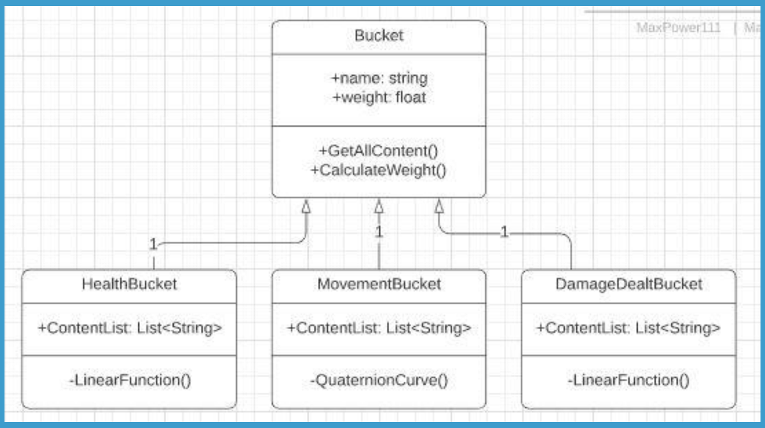

Fig2: Oates, Max. 2020. UML Diagram of bucket classes

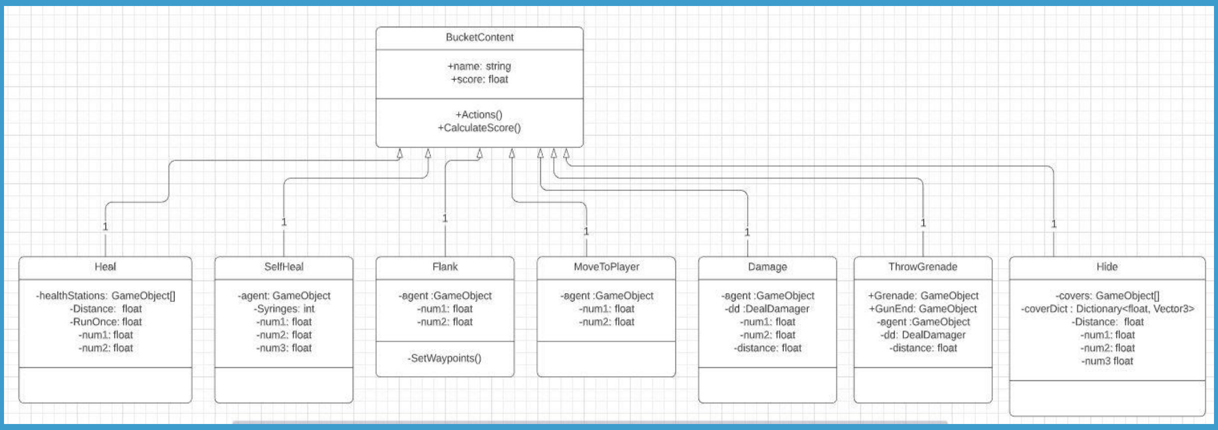

Fig3: Oates, Max. 2020. UML Diagram of action classes

Conclusion & Improvements

The AIs do show promise due to their ability to make decisions and perform actions. I feel like it was able to solve the problem but not as well as expected due to it still feeling robotic at time. However this may be due to the limitations of Utility Based AI. A picture of the game is shown in figure 4.

If I were to improve, I would:

- Use Dictionaries instead of factory patterns to permanently store buckets and actions when organizing them.
- Spend more time experimenting with different ways it makes decisions.
- Use the standard Utility Based AI technique instead of Dual Utility Reasoning.

Bibliography

DILL, Kevin. 2015. *Game AI Pro 2: Collected Wisdom of Game AI Professionals.* CRC Press.

GRAHAM, David “. 2020. 'An Introduction to Utility Theory'. In Anonymous *Game AI Pro 360.* (1st edn). CRC Press, 67-80.MCPARTLAND,

Michelle. 2013. 'Beaten by Bots – Training AI for First-Person Shooter Games'. Available at: http://theconversation.com/beaten-by-bots-training-ai-for-first-person-shooter-games-11176. [Accessed Mar 7,].

Fig. 1: Oates 2020. Decision making Flow chart [Flow Chart]
Fig. 2: Oates 2020. UML Diagram of bucket classes [UML Diagram]
Fig. 3: Oates 2020. UML Diagram of action classes [UML Diagram]
Fig. 4: Oates 2020. Example of gameplay [Screenshot]