# Inspiration

Unexpected results in games are sometimes the most entertaining moments. Unexpectedness can emerge from complex interactions between simple elements such as physics simulations. Directly designing unexpectedness has the drawback of non-replayability, when the interactions are not free enough to have emerging unforeseen results.

Artificial intelligence (AI) in games typically contains the case of unexpected behavior in arbitrary situations. Ranging from role-playing games (ex. The Elder Scrolls series, GTA, etc.) to simulations (The Sims, etc.) to strategy (Warcraft, Command & Conquer, Total War series, Civilization, etc.), artificially intelligent agents (even if simple) can be a source of fun because of the uncanniness of their actions. Sometimes, they can surprise with clever solutions or skillful displays, other times they can do goofy things that a real player might never achieve (such as combining unexpected animations or sounds, or the like).

But AI in games is rarely truly intelligent, because the designs often specify behaviors that the agents need to follow in order to fit the other design decisions, such as story or game mechanics. Academic AI, as opposed to game AI, is focused towards creating more “truly” intelligent behavior, maybe in some cases one could call it more human-like. Examples of this could be search engines, advertisement systems that learn from the preferences of users, mechanical robots that can navigate, etc. However, this is rarely used for entertainment purposes, even though the potential is there (ex. Physical movement simulations, chat-bots, etc.). One of the sources of inspiration for this project is the unexpected fun of “serious” systems.

# Self-improving artificial intelligence

A self-improving ultra-intelligent agent has been said to be “the last invention that man need ever make” (Good, 1965). Self-improvement can be a very broad notion, but in essence, it means that the agent should be able to perform better on arbitrary tasks based on self-evaluation and adaptation. By self-improving, in addition to choosing the best course of action for different situations, the agent can modify its internal structure to better suit the needs of its functioning and to better fulfill its goals.

An analysis of self-improving systems by Stephen Omohundro, though abstract and somewhat futuristic, describes how a self-improving system will converge towards predictable behaviors founded on theories of microeconomics (Omohundro, 2008, p. 4). It also describes 5 stages of technology as *inert, reactive, adaptive, deliberative and self-improving*, relating to the way systems interact with their surrounding medium (Omohundro, 2008, p. 5). The article describes many generalities about self-improving systems, which can be used to form a list of requirements for a concrete implementation.

## Utility function

Omohundro’s article (Omohundro, 2008) demonstrates the need that an intelligent system must have a goal, and it formalizes this goal as a *utility function U*, which returns a score of how efficient an action (or series of actions) is towards achieving that goal (Omohundro, 2008, p. 8). The utility function, in the most generalized case, is defined on the domain of all possible histories of the universe, *S*, and uses a vector *P* to hold the probability that each history will occur. Even though this sounds very over-the-top, S is referring to the universe that the agent is located in (and interacts with). For example, a chess-playing machine would choose U to be the total number of chess games that it wins, S is the total amount of possible games that it can play. The system would therefore choose to pursue the scenario from S which is most likely, according to P, to achieve the goal of winning. P represents the system’s beliefs about the likelihood of each universe history, and at any moment in time, the system would have a subset of S which is compatible with the system’s current knowledge and actions that can lead to those situations (Omohundro, 2008).

## Set of actions

A fundamental requirement for an intelligent system is to have a set of actions that it can perform freely; it needs to use them most efficiently to fulfill its goals. For a self-improving system, those actions could even be modified and improved. The level of the actions can be very simple, such as activating a motor, to more elaborate, such as traveling the shortest path towards a destination. The way the actions are defined will affect how they interact with the environment and their effect as perceived by the agent. If an action does not have a perceivable effect in the environment, the agent will have no reason to perform it and thus it will end up unused/discarded.

## Perception

Another required component of such a system is perception. The information the agent has about the environment is what will determine the accuracy of its estimations about the utility of its actions and about the environment. The perception system must also contain information about the agent’s own state, such as health and other status parameters. This data will be used by multiple actions and goals, so it must be centralized in a different system than actions or goals.

For every unknown in the environment, the system will have to compute a probability, thus rendering the estimation a little more unreliable with each unknown factor. For simple scenarios, it could be possible to describe all parameters of the environment and have the agent make purely rational choices, but in most cases it is unfeasible from either computational or memory reasons to do so, therefore approximations must be used instead. It was shown in (Omohundro, 2008) that probabilities can be used for unknown factors in such deliberations.

# Game-oriented system

A simple way to test if such a system ­­­works is in a controlled environment such as a game, with few parameters, and having agents with few basic actions they can perform. The goal that the agents need to achieve should be something well defined according to the game universe. The proof that the system is working is shown if agents can achieve some degree of success based on the utility function, and can learn to use the correct actions, incorporate new actions in their behavior, or discard inefficient actions.

After the system is shown to work in a simple environment, a more elaborate design can be made to test it in more broad situations.

# Overview of game AI architectures

Based on this article (Mark, 2012), there are several examples of AI systems implemented in games. The purpose of a game-oriented system is entertainment, which is different from academic AI; the agent is not supposed to be better than the player, surpassing their skills and defeating them easily. They are supposed to provide a fun challenge, seem “intelligent” but not overly so.

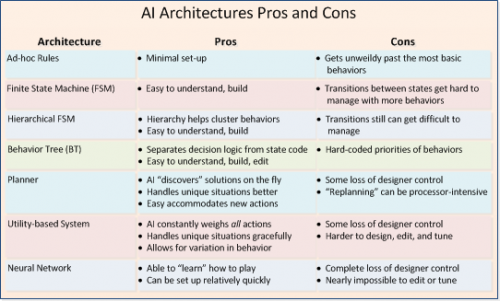


Figure Table showing pros and cons of different architectures as described in (Mark, 2012)

The most common systems are linear, and as described by (Omohundro, 2008) could be categorized as *reactive* rather than intelligent. Basic rule systems and finite state machines would fall under this category. More complex systems such as behavior trees would split decision logic from actions, which means they would deliberate about situations and act accordingly, which makes them *deliberative* though still predictable. Finally, planner or utility based systems can have very intricate mechanisms of adapting the weights of actions usefulness, so they would be able to incorporate new actions as well as change their behavior after trial and error, which would make them more genuinely intelligent, but harder to stick to the game design. Neural networks are not described thoroughly in the article, but most likely would be very adaptive and unpredictable, and if used correctly might provide the most rational behavior.

## Goal Oriented Action Planning

A system very similar in concept to the requirements taken from (Omohundro, 2008) is Goal Oriented Action Planning (GOAP) (Orkin, Goal Oriented Action Planning (GOAP) , 2011), a variation on the STRIPS planning system (Nilsson & Fikes, 1971) developed as a standard academic AI at Stanford University in 1971, first used in the game F.E.A.R. (2005), but also used in other major titles such as Fallout 3, Empire: Total War, Deus Ex: Human Revolution, etc.

The GOAP system is based on the idea that the agent must use its actions in order to fulfill its goals, and the designer need only provide those actions and specify their effects (Orkin, 2006).

In F.E.A.R., the system is set up in terms of goals and actions, therefore agents can have common goals and actions but not behave the same way.

# Implementation

This chapter will guide the reader through the process of implementing the system, roughly, based on redesigns and reiterations.

## Required systems

Based on (Omohundro, 2008) and (Orkin, Goal Oriented Action Planning (GOAP) , 2011), the requirements for this system are as following:

* Basic actions
* Goals
* Perception

Making those systems separate has the benefit of being able to alter them independently for purposes of optimization, not depending on each other means they can be used in other contexts or generalized, and subsystems or techniques can be implemented inside each system that would make them more dynamic and/or adaptive. A self-improving mechanism can then be implemented by creating a separate virtual model of the entire system or each subsystem which would allow the agent to modify the structure of the systems while being able to know/learn the effects of such modifications.

## The connection between goals and actions

The agent’s actions will need to have specific effects described in some data structure, while goals have requirements that need to be fulfilled, which should be described in the same structure. This way, the agent will know what actions to pursue in order to fulfill its goals, without trial and error for every single action. Of course, trial and error can work by automatically updating this information in the structure after every performance of the action.

In (F.E.A.R., 2005, p. 12), the requirement/effect structure is four bytes, representing basic data such as a node in the navigational path, a vehicle type, a weapon type and a Boolean specifying if the target is dead. Depending on the complexity of the awareness system, this structure can get very complex; if intended for general use, it would most likely require all parameters of perception regardless if the action is manipulating them in any way. For example, an eating action would not require to know about the state of enemies around the agent, an attack state would not require info about food, but a defensive action would require both sets of data, so the simplest way to implement is to always use the full set of data, if its size is reasonable.

# Introduction (WIP)

This project is a learning system which uses pre-defined actions in the best way it can find in order to fulfill a specific predetermined goal, implemented in a game. An arbitrary but main requirement of the system is that each module/subsystem must be kept separate from the others, for ease of optimization, generalization, improvement, adaptation and reusability. Proof that the system is working is shown if agents can achieve some degree of success based on its goal, and can learn to use the correct actions, incorporate new actions in their behavior, or discard inefficient actions.

# Bibliography

F.E.A.R. (2005). Monolith Productions/VU Games.

Good, I. (1965). Speculations concerning the first ultraintelligent machine. *Advances in Computers*, 31-88.

Mark, D. (2012, 11 1). *AI Architectures: A Culinary Guide (GDMag Article)*. Retrieved from intrinsicalgorithm.com: http://intrinsicalgorithm.com/IAonAI/2012/11/ai-architectures-a-culinary-guide-gdmag-article/

Nilsson, N., & Fikes, R. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 189-208.

Omohundro, S. (2008). *The Nature of Self-Improving Artiﬁcial Intelligence.* Palo Alto, California: Self Aware Systems.

Orkin, J. (2006). Three States and a Plan: The A.I. of F.E.A.R. *GDC 2006.* Monolith Productions / M.I.T. Media Lab. Retrieved from http://web.media.mit.edu/~jorkin/gdc2006\_orkin\_jeff\_fear.pdf

Orkin, J. (2011). *Goal Oriented Action Planning (GOAP)* . Retrieved from Jeff Orkin: http://web.media.mit.edu/~jorkin/goap.html