Strategies for Strategy Game AI

Ian Lane Davis, Technical Director

Activision, Inc.
3100 Ocean Park Blvd, Santa Monica, CA 90405
idavis@activision.com, akiam@alum.dartmouth.org

Abstract

The biggest challenge in computer strategy games is the creation of a fun computer opponent. The hardest element of "fun" is "good". For the Artificial Intelligence to be good, it must do much of what a human player would do: situational & map analysis, resource allocation, and strategy execution. We present here the framework and some of the strategies we have developed at Activision in the development of games such as **Dark Reign: The Future Of War**, **Battlezone**, **Civilization: Call To Power**, and others.

Introduction

Recently at Activision we've had the opportunity to develop a relatively large number of high profile military strategy games including *Dark Reign: The Future Of War*, *Battlezone*, and *Civilization: Call To Power***.** Perhaps the hardest part of developing strategy games is the design and implementation of the Artificial Intelligence. Thus, over the last few years we've developed the "Dark Reign Model" for strategy game AI. We have applied this model to all of our strategy games with great success and significant code re-use. This model first breaks down the AI tasks into Tactical AI and Strategic AI, and then further breaks down the Strategic AI into three parts: Analysis, Resource Allocation, and High Level AI (personality).

Problem Definition

Before describing the details of the Dark Reign Model, we would like to define the context of the problem and our solutions. To generate a successful computer opponent in a strategy game (or any computer game, for that matter), it is important to focus on what the developer's true goal is:

Davis' First Law of Computer Game AI: The goal of any AI is to lose the game.

This may seem counter-intuitive! Most programmers and researchers want to develop algorithms and approaches that are optimal and perfect. There are two problems, however. The first problem is that optimal is always hard and often impossible. In classic games, such as chess and checkers, all players can see the entire board at once, whereas in most modern computer strategy games all players (including the computer) have incomplete knowledge of the location and composition of the opponents' pieces. And while it's true that the AI players can cheat knowledge, it's very hard to hide that cheating from players, and if the cheating is obvious, the game is not fun. Incomplete knowledge precludes optimal decision-making. Furthermore, in comparison to tradition games, the number of pieces, the number of types of pieces, and the number of actions available to each piece are colossal. Optimal solutions such as game trees [Tanimoto87] that try to predict several moves ahead (useful for tic-tac-toe, and some other small games) are inappropriate and infeasible due to the relatively enormous branching factor.

The second problem with trying to make an optimal and perfect AI player is that, even if it were possible, it would be undesirable. Although most players want a challenging AI opponent, *nobody likes to lose all the time*. The goal is to make the player feel threatened, but then to make the player feel as if she has heroically fought off a superior enemy [Millar96]. Frequently this involves pushing the player to the edge and then intentionally backing off. Such a tactic makes the player feel like they just barely held off the enemy horde, but sent them fleeing with a to-the-last-man defense.

Fortunately or not, it's easy enough to create an AI player that will lose a game. The trick is to make an opponent that can create an exciting ebb and flow of power. Towards this end, the computer player needs to be able to play a devastatingly good game, and be able to back off when needed. The AI must dynamically evaluate its situation, formulate a plan, and execute it. As a final wrinkle, the AI system must be straightforward to use, as the game's level designers are often non-technical.

Turn-Based versus RTS

We call our approach the Dark Reign Model because it was first used on *Dark Reign: the Future of War.* In the realm of computer games, *Dark Reign* is known as an RTS, or Real-Time Strategy game. In an RTS, you can instruct a unit, or a number of units, to move across the board, and as they move you have time to move other pieces, and the opponent may also move his/her/its pieces at the same time. In contrast, *Civilization: Call To Power* is a Turn-Based game, which means that only one player can move

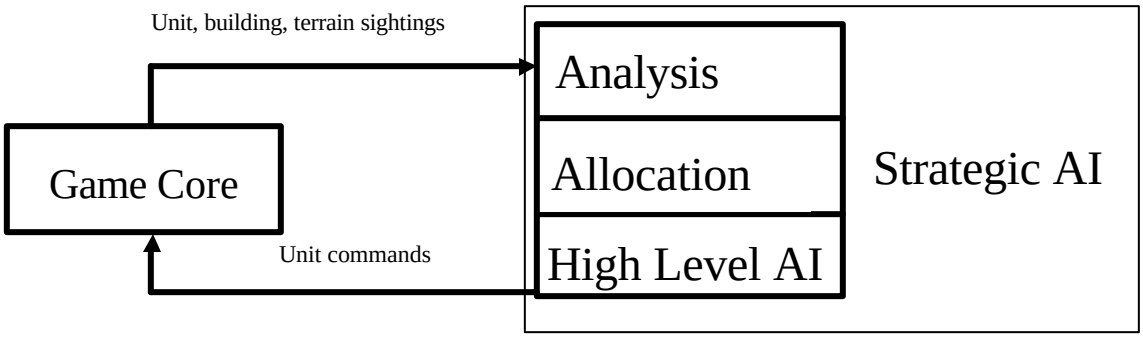

Figure 1. Strategic AI Diagram

units at a time, and within a team, only one unit moves at once. The fundamental difference between an RTS game and a Turn-Based game for an AI developer is that in a Turn-Based game the feedback for any decision is immediate: if you decide to move a unit, that unit can be moved before anything else happens. However, in an RTS, you can (and often have to) separate strategic troop and resource commitment decisions from the tactical/low-level actions of units, whereas in a Turn-Based game there is no such distinction.

Tactical AI

Although it is not our focus here, it is worth briefly describing the Tactical AI system for an RTS such as *Dark Reign* or *Battlezone*. The Tactical AI in an RTS game is tightly coupled with the core game engine. For each game, there are several parameters (controllable by potentially non-technical level designers) that define how aggressively units pursue targets. In *Dark Reign*, the units controlled by the AI constantly look to see if any enemy units were nearby. If so, each unit chooses its best target and attack it. "Best" target was defined by maximizing how much enemy firepower could be removed from the playing field most quickly. Thus a heavily armed, but lightly armored unit would be an early target for the tactical AI, but a heavily armored non-combat unit would be the last thing shot at. As a side note, in most strategy games, the offensive effectiveness of a unit does not diminish at all with damage until the unit is totally removed from the game. This means it virtually always makes sense to concentrate firepower at one target until it's destroyed and then to move on to the next.

Strategic AI

The goal of the Dark Reign Model for Strategic AI is to allow the creation of computer opponents that respond to changes in fortune and circumstance. If a computer opponent can retreat and regroup when things look bleak or recognize a weakness in the enemy's position and attack it, then the gameplay can be vastly improved. To create this dynamic gameplay, our strategic AI consists of three modules, an analysis module, a resource allocation module, and a high level AI module. The first two modules' performance is defined by a set of parameters called an AI Personality (AIP, pronounced "ape"), and the third module is a logical system which is responsible for changing AIPs when new situations arise, and for triggering special actions based on certain events occurring. Each module requires a different kind of AI techniques, and each is described below.

Analysis Module

The ultimate goal of the analysis module is to define the current strategic goals and rate them by priority. The strategic goals can include exploration, reconnaissance, base construction, defensive goals, and offensive goals. In *Dark Reign*, the goals were geographically defined: the map was divided into a number of regions and each region has some offensive, defensive, information, and/or resource values. These values are combined to make some overall priority for each region. The analysis system would simply tell the resource allocation system which areas troops should be sent to (and the tactical AI handles attacks). In *Civilization: Call To Power*, the goals are target-oriented, which means that each enemy city or unit can become its own goal. Each goal is still rated by several values. The contribution of each value to the importance of a goal is set by the level designer through an AIP configuration file.

The question is "How do we arrive at these values, and what techniques are useful?" We need to know things such as how far each spot on the map is from our empire, how far each spot is from the enemy bases, which areas of the map are dangerous, and which areas can we reach from which other areas. These are map analysis problems, which are quite similar to computer vision problems, and our solutions come from that realm (see [Davis96] for applications of the following techniques in computer vision).

For example, if we want to concentrate our troops near areas we control, we need to know how far any potential

target is from those areas. The naïve approach is to look at a target and search for the nearest piece of our territory. This is a major computational burden, so we can use a technique known to the computer vision community as the grassfire algorithm [Duda73] to compute the distance to our territory from every other spot on the map in a simple two-pass raster scan algorithm.

Next, if we want to know how dangerous a given section of map is, we start by finding the locations of all known enemy troops. We can divide the map into regions and sum up the threat value in each region. But the danger is that a region with no enemy units in it may be next to an area filled with enemies. Thus, we can use relaxation techniques to spread the danger to nearby regions [Press90].

As a final example, one pitfall is that you may be able to see a target that is strategically important, but unreachable with your current units (such as an island when you have no boats). Running a path finding algorithm from each unit of yours to each potential target is too expensive, so you can use a simple connected region extraction technique to label each movement modality's (air, land, water, etc.) contiguously reachable areas. If the label for the land under a unit is different from the label under a target, it can't reach the target. These regions can often be computed once at the beginning of the game, and just stored for constant time look up.

These are just a few examples of analysis techniques. Some games may require more complicated statistical analyses of terrain and troop distributions, predictive extrapolations on the positions of troops that cannot be seen any longer, clustering techniques for proper target definitions, etc. Various computational geometry techniques such as Voronoi diagrams and convex hull techniques are also applicable for computing distance related metrics and cluster analysis [Preparata85]. The goals of the analysis are simple, however: find the enemy & rank targets by importance.

Resource Allocation

The resource allocation module takes the strategic goals from the analysis module, and allocates available troops to those goals. This module outputs commands to the Tactical AI (or directly to the units). This system handles matching forces against enemy forces, taking over terrain resources, attacking and defending bases, etc. The actual decision on what troops to use for what happens here.

The simple Bipartite Matching problem

Matching our available forces to our strategic goals can be thought of as a variant on the bipartite matching problem [see Sedgewick92]. In that problem, you have resources that you must maximally match to some set of tasks (see Figure 2). For us, the resources are the groups of units (and the individual units) and the tasks are the strategic goals, such as "attack/counter an enemy group" or "take over a mineral mine". In the simplest version of the bipartite matching problem, you construct a graph (a collection of nodes and direction connections between them) in which each resource that is capable of addressing a task is connected to that task, as in Figure 2.

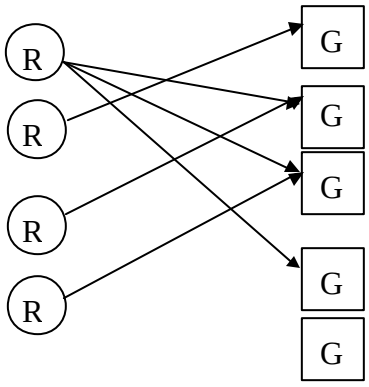

Figure 2. Resource Allocation: Here we have four units (resources) and five targets (goals).

Our matching problem

Our matching problem is a bit more involved for a few main reasons. First, each task (strategic goal) can require a different amount of resources (troop strength & composition) to accomplish. Second, each resource (group of our units) can have a different ability to satisfy each goal (troop strength and composition). Third, some tasks are more important than others. Also, if we cannot commit enough resources to a task to fully achieve the goal of that task, we do not want to commit any units (an insufficient force) to that task. Thus, we are not concerned with *just* committing the maximum amount of resources to tasks, but we also prefer to achieve the most important strategic goals before the less important ones. Two more wrinkles are that a single group's forces could be split between two tasks, and/or a single task could be tackled by forces from a number of different groups.

There are many possible solutions to this problem. Our solutions tend to be variants on network flow solutions [see Sedgewick92]. Not only are all of the goals ranked in priority, but all of the matches between troops and goals are ranked, too. The largest term in the match's scores is the raw goal priority, but the appropriateness of a particular unit type and the distance from each unit to the goal (for example) also contribute. This helps a unit right next to a medium priority target choose that goal instead of a high priority one very far away. We run a greedy network flow bipartite matching algorithm to commit troops to particular

goals. In an RTS such as Dark Reign, after the full graph is matched as best as we can, we send all the troops on their way. For a Turn-Based game, in which one decision can immediately affect our next decision, as soon as one goal is satisfied (has enough troops) we send them immediately, and then update some of the other matches based on the results of our action.

High Level AI

The final system of the Strategic AI for strategy games is what we call the High Level AI. The High Level is responsible for switching AI modes and personalities. It can look at large scale factors (such as the ratio of our strength to the enemies' strengths) to set major modes through AIPs (an AIP that favors defense or one that wants to explore). It can also be used to set specific triggers in the game; this allows the designer to specify commands such as, "when an enemy enters this region, attack with these forces".

The High Level system can be anything from a simple Finite State Machine [Sedgewick92] to a Rules-Based System [Charniak86] to a Fuzzy Logic System [Kosko97], or something even more complex. In building a system, two things should be kept in mind. First, level designers who can be largely non-technical will need to tweak the parameters of the High Level AI frequently, if not write the rules/FSMs themselves. Thus, debugging ease and readability should be principal concerns. Second, the system is only as good as its inputs and outputs.

For outputs, you may be able to use just switching AIPs, but we found it necessary to supplement that action with some to control particular mission/map specific troops for *Dark Reign*. One of the advantages that a human player has over the computer is that the human can remember what happened in the last game played on a particular map. Thus, the level designer may want to program into the High Level AI some map knowledge (for example, "If the enemy troops pass over the southernmost bridge, we should send troops to attack the northwest quadrant"). For inputs, you will want some simple things such as Overall Threat, Power Ratio (our strength to the enemy's strength), Game Time, etc. But more specific inputs and triggers may be required: the enemy is in a pre-defined region, when some particular units of ours are destroyed, or when the enemy has created a super weapon.

Conclusions

The Dark Reign Model breaks down strategy game AI into Tactical and Strategic AI, and further decomposes Strategic AI into Analysis, Allocation, and High Level (personality). Even in a broad overview such as this, it should be obvious that there are a large number of systems which must be implemented, and a large number of parameters which need to be adjusted in order to provide decent AI behavior. The level designers must have all the tools that they need to analyze the playing field and commit resources, but given the complexity of the various systems and their interactions it is also important to keep the number of parameters to adjust manageable.

Since "decent AI behavior" is not simply winning the game (which could be accomplished by simply giving the AI player combat or resource advantages), the AI players must be designed with some subtlety. The goal is to provide an entertaining, challenging, and defeatable foe. Following the Dark Reign Model we've been able to create a number of successful games played by hundreds of thousands of gamers (and even with significant code re-use).

Acknowledgements

The author would like to thank Gordon Moyes, Karl Meissner, and Richard Myers for their work and contributions to the Dark Reign Model and their own innovative strategic AI work, and Scott Lahman, VP of Production for supporting advanced AI work at Activision and encouraging the publication of results.

References

[Charniak86] E. Charniak and D. McDermott, *An Introduction to Artificial Intelligence*, Addison-Wesley, 1986.

[Davis96] I. Davis, *A Modular Neural Network Approach to Autonomous Navigation*, Ph.D. diss., Robotics, School of Computer Science, Carnegie Mellon University, 1996.

[Duda73] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley and Sons, 1973.

[Kosko97] B. Kosko, *Fuzzy Engineering*, Prentice Hall Press, 1997.

[Millar96] R. Millar and J. Watkins III, Personal discussions, 1996.

[Preparata85] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[Press90] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vatterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1990.

[Sedgewick92] R. Sedgewick, *Algorithms in C++*, Addison-Wesley, 1992.

[Tanimoto87] S. Tanimoto, *The Elements of Artificial Intelligence*, Computer Science Press, Rockville, Maryland, 1987.