**A Real Time Approach to War Games**

*Walter B. Gress V*

*Drexel University*

*Friday, June 7 2013*

This concept of operations details a “Divide and Defend” Firewall. Recently there has been a surge of malware attacks, those attacks typically being orchestrated through botnets. Modern enterprise and personal firewalls are not designed by default to handle massive number of attacks at once like botnets can incur. D&D is designed with that very thought in mind.

# introduction

This document details an innovative way to handle artificial intelligence in a tactical military strategy game. Our approach uses an innovative approach to generate a multitude behaviors from an intial set of four attributes per soldier.

# DECISION TREE

Instead of a classic approach: the use of state machines to decide the behavior best suited for the soldier, we found this approach primitive and poor at best. Certain published games such as the Close Combat series use this approach. While functional, they require the hard wired coding in which every instance for every behavior is hard coded, leaving us with a monolithic application code base. Difficult to keep track and eventually bloated we are attempting a different approach. To begin with, instead of state machines, the behavior has been encapsulated as a pseudo-decision tree. In this case, leaves are chosen behaviors, the root is the beginning of the decision process. The root has two sets of four variables representing the state of the soldier as well as “qualities”. Each time we progress from node to node in the tree, a decision is made based off of the qualities and the sets of variables. They are HEALTH, …., anda then (QUALITIES). At each branch in the tree a set of comparisons between variables, deciding which way to traverse the tree further. This continues until a behavior is chosen.

# A\* Search

So far we have established an alternate way of making decisions from state machines to a decision tree. However, have we gained that much? We still have to write all the possible tests at each branch of the tree. A better way may be, instead of just hard coding the tree, to use A\* search to determine the best way to a leaf (behavior/solution). Unfortunately still we have the inbetweens of the root and the leaves. We still have to figure out a way to avoid hard coding all the intermediate decisions.

Should we choose to go the route of A\* search, we could use the variables at each junction as a method to generate costs g(x). The path taken is the best score.