Artificial Intelligence

New Mexico High School Supercomputing Challenge
Final Report
Category A
April 4, 2001

Team 064
Alamogordo High School

Team Members:

Chris Berger

Joseph Farfel

Matt Hoppe

Vincent Hoppe

Scott Richardson

Teachers:

Albert Simon

Table of Contents

Executive Summary:

Artificial intelligence, although initially thought to be a fanciful representation of scientific concepts with little basis in fact, has been validated and its theories affirmed through the research and analysis of the logical foundations of its principles. Modern interpretations of the artificial intelligence theory have led scientific theorists to begin to accept the feasibility of such ideas

formerly thought to reside only in the realm of science fiction. In attempting this project, our group is undertaking the investigation and simulation of the concepts involving advanced computerized decision making.

The goal of this project is to create a sophisticated computerized artificial intelligence system that enables the computer to think logically and formulate the best strategy for accomplishing a task. The computer will both make "educated guesses" in-simulation based on it's likelihood for a successful outcome, and compare this to previous attempts at success, so that the computer can successfully formulate the most effective strategy possible in a real time environment.

This environment, in which this simulation was conducted, is drafted from the avionics dogfight. In this project, the computer is responsible for taking knowledge of enemy positions and, using logical paths to generalize the situation, rank its options, using possibly previous failures/successes, to make decisions (i.e. in what instance to attack or evade and to where to evade) and win the simulation. To accomplish this, the computer generalizes the location and direction of the enemy plane in relation with the player. The AI then decides the best method for flight, whereby subsequently positioning the plane behind the enemy so as to effect a kill. Using a series of "evaluators", the AI then determines the effectiveness of possible actions and concludes the most strategic action. This will empower the computer to think in the same general manner a human does in order to, in the most effective way possible, determine the best method for completing its function.

Two "intelligent" AI's were pitted against each other in the simulation to test their interactions with each other. As an artificial intelligence was implemented, it was found that the planes did become more effective at attacking and evading when compared both to a constant, controlled AI and against each other. Thus, in turn, it was shown that using a series of generalities and evaluators of favorableness, and logical paths of deduction, a computer could be made more efficient.

Introduction:

Artificial intelligence, although initially thought to be a fanciful representation of scientific concepts with little basis in fact, has been validated and its theories affirmed through the research and analysis of the logical foundations of its principles. Modern interpretations of the artificial intelligence theory have led scientific theorists to begin to accept the feasibility of such ideas formerly thought to reside only in the realm of science fiction. In attempting this project, our

group is undertaking the investigation and simulation of the concepts involving advanced computerized decision making.

'AI' is defined as a method whereby an apparent intelligence is created; the ability of a machine to perform those activities that are normally thought to require intelligence; in its most advanced form would pass the Turing test, a test to perform such intelligent tasks. If a computer can mimic human responses to such an extent that it cannot be said by an outside party, with any great certainty, that it is not human, then it is said to be artificially intelligent. Our AI simply is able to use a series of evaluators to conclude the effectiveness of certain actions, and carry out the most effective sequence of these moves.

The goal of this project is to create a sophisticated computerized artificial intelligence system that enables the computer to think logically and formulate the best strategy for accomplishing a task. The computer will both make "educated guesses" in-simulation based on its likelihood for a successful outcome, and possibly compare this to previous attempts at success, so that the computer can successfully formulate the most effective strategy possible in a real time environment. This will empower the computer to think in the same general manner a human does in order to, in the most effective way possible, determine the best method for completing its function.

Description:

The environment in which this simulation was conducted is drafted from the avionics dogfight. This simulation was constructed as a clear (no obstructions) 3D square map in which the planes "fly." The players start at opposite corners (in the x-y coordinate plane) of the map, and in the simulation, their goal is to destroy each other. The planes controlled by the AI's can move in all

three dimensions, in order to out-maneuver each other and position themselves in good locations from which to effect a kill. The environment in which the planes move, the plane movement and attack algorithms, rudimentary physics, constructing/destroying planes and players, and other parts of the simulation have been coded.

The simulation allows for a few accepted truths or constants, so as to not overwhelm the code, computer, or the programmers. Firstly, an ideal environment was created; one in which the two planes engage at an altitude above all interactions with other environmental features (i.e. the ground, weather conditions). There are, however, equations to take into account the principals of physics, such as air resistance on the plane during maneuvers. The planes have also been given constants of maneuverability, speed, and drag. There are no other variables accounted for concerning the environment or plane.

The AI is organized simply to maximize the effectiveness of an offensive - attacking - or a defensive strategy through the best use of the particular variables and its knowledge of the current situation. The computer is responsible for taking knowledge of enemy positions and, using logical paths to generalize the situation, rank its options, using possibly previous failures/successes to make decisions (i.e. in what instance to attack or evade and to where to evade), and win the simulation. To accomplish this, the computer generalizes the location and direction of the enemy plane in relation with the player. The AI then decides the best method for flight, whereby subsequently positioning the plane behind the enemy so as to effect a kill. Using a series of “evaluators”, the AI then determines the effectiveness of possible actions and concludes the most strategic action. An effective maneuver is defined as one in which the plane is put in a better position to fire at the enemy, or increases the distance between the planes’ azimuths to evade fire. Eventually, two "intelligent" AI's will be pitted against each other in the

simulation to test their interactions with each other. To view the interactions between the planes, a layout for displaying the simulation graphically through the OpenGL 3D programming API was designed.

Initially we implemented a very simple AI: each plane simply would fly towards the other. Further development of the AI continued firstly in two areas, attack and evasion. Attack AI is used to direct a plane in the most efficient way towards its target, and Evasion AI is used to make a plane attempt to maneuver out of its attacker's line of fire as quickly as possible. The Attack AI drives one player's plane, and the Evade AI drives the others for testing purposes. Eventually, both AI's were combined into a final true AI that was capable of both attacking and evading the other when necessary.

The Program

Our interest in artificial intelligence spawned the thought that a dogfight would be a very practical and possible shell from which to attempt an endeavor such as this. The program was written in C++, using OpenGL as a graphic output Application Programming Interface (API) (this would allow us to visualize how our planes were moving, instead of looking at a list of X, Y, and Z coordinates). (Explanation on OpenGL see Appendix I).

Early on in the coding process we had to decide how to organize the project. Initially, structs seemed sufficient to keep track of the plane's attributes, but this proved to lack some of the versatility needed. Instead, we converted to classes, which we found to have a few extra benefits. Firstly, classes are multi-file compatible, so a function for keeping track of all unit variable changes was not needed. We also found classes to provide a few time saving techniques concerning class objects and class member functions. This allowed the code to be organized very logically: a struct contained all the units' attributes and variables, such as speed, X, Y, Z

position, damage etc., while the players that used these attributes were organized into a class. The functions that the players used to interact with the environment, such as movement, seeing, attacking functions, were declared class member functions. Thus, players used a defined plane, to interact, through privileged functions, with the environment, similar to the hierarchy of a plane in the real world.

Once we created this structure, we set about creating the environment in which the planes would fly. Functions that would initialize the players with the desired attributes depending upon plane type, place them in the environment, determine where the plane was and how to move it to where it wanted to go (how much to rotate etc.), under what circumstances a plane could attack and when a plane was hit had to be defined, created, and tested. The specifics of the simulation went through many revisions, especially how the planes move in the environment, that is to say, through what calculations will the plane reach its destination (this will be discussed further in proceeding paragraphs). Furthermore, problems arose as we tried to define a 3-Dimensional scenario. One by one, these functions were crafted and a simulation in which planes could interact was created. (The individual functions of the simulation will be discussed more thoroughly momentarily.)

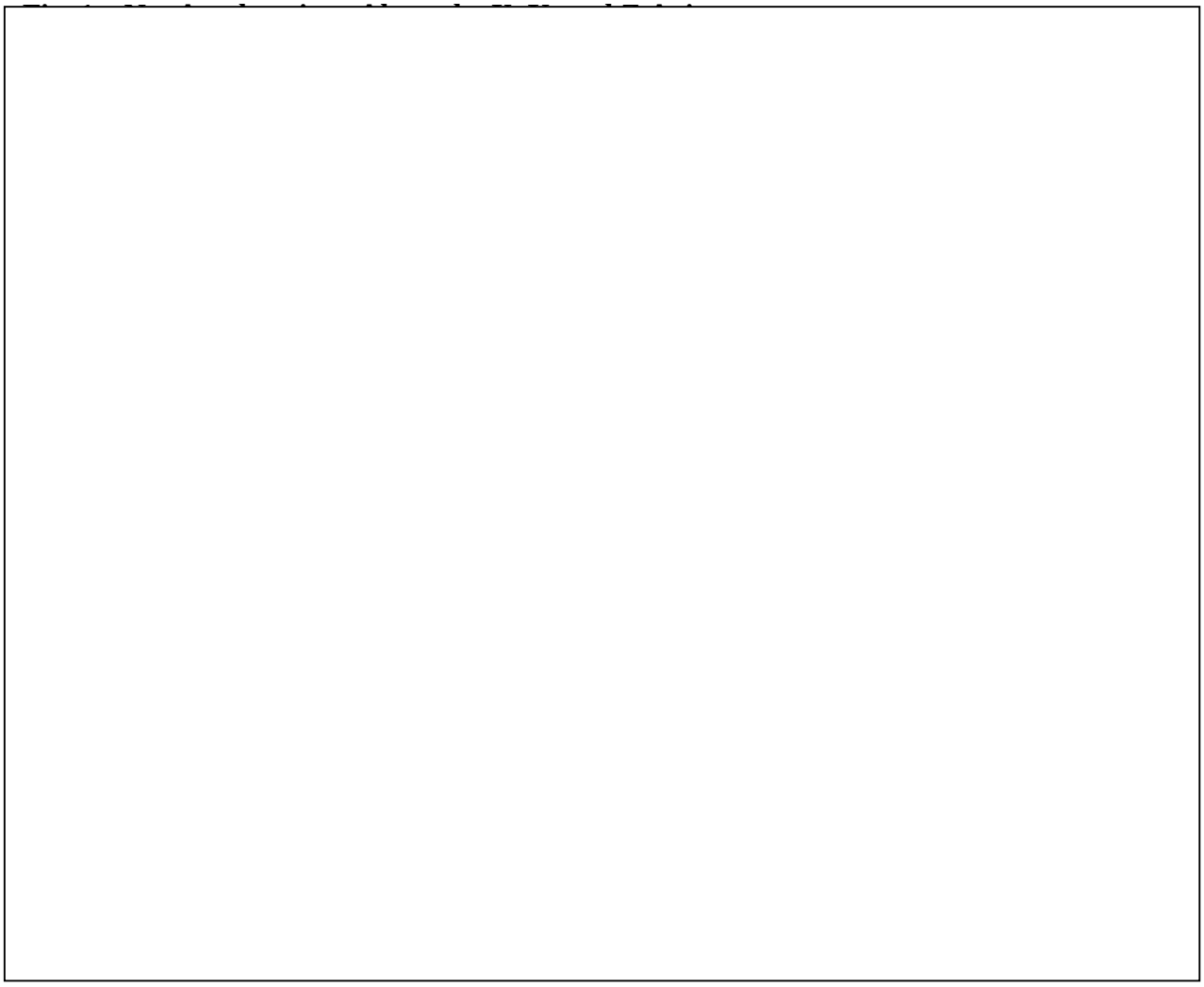
With the simulation completed, the work on the computer AI began; this code contains all the functions which are involved with the decision making process of the computer. This section of the code was far more intensive in terms of analytic creativity, whereas the environment was mostly the labor-intensive task of hacking away at functions until they worked. The AI was coded on the principle of allowing the computer to make educated guesses as to the best next move. To accomplish this, a series of evaluators were used to determine the effectiveness of a specific maneuver based on its favorable characteristics. Eventually, these

evaluators will log this effectiveness and the program will refer to this data as supplemental information for the decision. This should allow the computer to combine the most effective maneuvers and therefore be the most successful.

- Equations

As was mentioned, equations were used to define the physics of the environment to simulate the real flight of the plane, such as the effect of drag as turns are made. In this way, the planes will interact in a realistic environment, one in which outrageous maneuvers cannot be preformed, and one where the planes will accelerate, decelerate, turn, and loose altitude as naturally as possible. In this way, the players will be forced to take into account that depending upon how they turn, the plane will loose or gain altitude and speed and therefore possibly allow the enemy an opening to advance, or force it to over maneuver and loose the edge.

The function most intensely related to these forces was the movement function, which controlled how the plane would translate from one point to the next. The forces controlling movement are those which would act on a real plane, and they include thrust, drag, lift, and weight; the first three are vectors, and weight is always straight down. To calculate the net acceleration in the X, Y, and Z directions, these force vectors are resolved. That is to say, the cosines of the plane's horizontal and vertical azimuths are multiplied by the vectors to find the magnitude of the vector in the X, Y, or Z direction (see Fig. 1). These accelerations in the specific direction are then totaled and added to the current speed in that direction. In this way, the movement for the plane is calculated. The forces acting on the plane have essential the same effect as they would a real plane. (i.e. The planes reach a max speed as a result of the drag vector canceling out the thrust vector).



The functions of the simulation are as follows:

• Initialization – The players are initialized and assigned attributes based on which type of planes the user selects. The players' initial positions are also defined. Such variables as sight, range, hit points, speed in the X, Y, and Z directions, lift and drag coefficients, mass, thrust, etc. are initialized to values that are appropriate and relative. For example, the mass and thrust of the plane are those of the F-16 Fighting Falcon.

• Movement – This function takes as input an azimuth that the plane will match, and a speed. The degree rotation for each turning axis is then calculated. Initially, the program

utilized a geometric deduction to calculate how the plane should turn. The arcTan of the slope between the plane's position and the destination left the difference by which the plane's azimuth would have to translate to line up with the destination. A turn radius to determine how much the azimuth should change each turn then divided this difference. The movement functions, however now, takes advantage of resolving vectors as its method of determining how the plane should be translated. This will make it easier to implement such variables as gravity and drag. To handle these calculations, there is a 'Move' and 'ContinueMoving' function.

• Seeing – If a plane can see the enemy then it gathers that information then used by the AI it to determine how to move. It also controls which plane it should try to attack.

• Attacking – This function determines if a player has been hit. It uses the strait line trajectory of the attacking plane and calculates to see if it intersects the opponent's position. As a hit is encountered the opponents hit points are accordingly docked.

• Player Elimination – This function eliminates a killed player from the Player Array, effectively moving all players of a greater 'i' value (i value being the players value in the array i.e. Player[2]) down in the list. When one 'i' value is left, that player, who is identified by a constant value, is the victor.

- AI Functions

The AI was used to decide where, in terms of pitch, yaw, and roll, to move; however, this is not the single decision, merely the output. The AI must determine, based on the current situation of position, direction, and speed, to either engage the opponent or evade with the prospective intent being to engage the opponent. From this conclusion, the plane will then be directed as to how to

move. The AI initially was divided into two procedures, attacking and evading. This was to ensure a successful implementation of the environment and plane to plane interactions. After this phase of testing was completed, and we found that in fact the plane did turn in opposition to the opponent or into the opponent, dependent upon the AI's goal, we felt free to move along with the advancement of the AI.

The problem with a separate AI for attacking and evading is that it is actually not that difficult to do one or the other. If the player is in the attacking sequence, it maneuvers to get behind the enemy, as being the position most likely produce a successful and swift kill; while the evade sequence contains processes by which to avoid fire and increase the distance between the two plane's azimuths, therefore ensuring the player will not come under fire soon after. The evading plane thus flew the opposite direction as the opponent thereby creating the greatest probability that it would not come under attack. This results in the evading plane simply circling so as to maximize the distance between its azimuth and the enemy's, while the enemy attempted to follow this procedure. As this was actually the most effective thing for each plane to do, it was not the intended result. Although this is technically the most efficient strategy, it is not the best because the plane simply is running from the opponent; it does not take into account the fact that in reality, a plane would not merely want to evade, but attack and kill the enemy.

A true AI must ensure its own survival, but also, the lack thereof for the enemy. Therefore, the plane must be put in a position to intercept the opponent, while not exposing itself to fire, and thus decide when to invoke the sequence for attacking or evading. To accomplish this, the AI considers the situation generally, and has simple routines for making decisions based on a situation. The AI will determine where and how to move based on a quadrant system (as the program becomes more sophisticated the quadrants will be subdivided). The quadrants allow

the computer to generalize locations and then provide a general response, as opposed to an exact one. This works by allowing the computer to recognize a situation and provide a response that would be similar regardless as to where the plane was actually and exactly positioned. For example, if the opponent was in a quadrant above and to the right of the player in the mode to evading, then that plane should switch to the attacking mode as it is in no position to be attacked, and turn to intercept it. This is the same response regardless of where the plane was actually located in this quadrant. This same opponent would, in turn, recognize the positioning of the enemy as being in a quadrant conducive to attacking, and thus alters its position to another quadrant which moves it out from in front of the enemy and thus sets it up for a move to attack. In the end, the flight pattern most likely to avoid attack and effect a kill would be utilized.

This is essentially the same way a human would react to such a situation (i.e. should a pilot see a bogie, he recognizes it as being in a general location, not a specific point, and thus moves in a manner that will position his plane in a better position either to avoid fire, or fire at the enemy).

Future Plans

This program we would like to see be expanded to account for multiple players and even teams, and possibly learn from previous runs of the simulation the most effective maneuvers. The program should also look at each opponents attributes, situation and the possible outlook for moves, while also considering the future plans for its own strategy, and then compile an overall outlook as to the effectiveness of a specific maneuver in a situation, and then, based on this determination, progress accordingly.

Eventually, a function should be created that looks at previous games, through text files which save a record of previous attempts, and from this data, conclude the effectiveness of certain strategies/attempts, categorizing them as a success or failure, and from this information make a decision upon which previous attempts were successful and would possibly be successful if attempted again.

Results

• Tell how we tested the program.

As we reached a stopping point in the program, we found that there were more problems then ever anticipated. Essentially, we worked extensively on the environment and unfortunately had less then foreseen time to conduct the same effort to the routines of the artificial intelligence. Nonetheless, the program is still notable for its many functions.

- The program effectively simulates the aircraft in flight through its various maneuvers with real physics affecting the situation.
- The AI's both looked upon their situation in much the same way a human would, generalizing situations and positions so as to produce an educated proceeding action.
- The AI's both improved their performance, as they, like a human, "learned" from experience.
- Using the AI's generalization of its situation, and then being able to quantify results based on effectiveness, the AI can successfully provide a valid response to may general scenarios.

Conclusion

As an artificial intelligence was implemented it was found that the planes did become more effective at attacking and evading when compared both in contrast to a constant, controlled AI and against each other. Thus, in turn, it was shown that using a series of generalities and evaluators, and logical paths of deduction, a computer could be made more efficient. After programming, testing, and refining of the artificial intelligence system being created in this project, this final system could blaze the trail for future, more advanced systems that could develop the "best" solutions for any given problem much more quickly and efficiently than could a human (these decisions would often be better than those a human would make, as well, because the computer can base its choices upon many more variables than a human could comprehend). Using such a technology, many tasks could be accomplished by computers alone. This technology could be implemented in unmanned spacecraft, mechanical probes entering areas too dangerous for humans to traverse, and even in machines that would perform household tasks. Such implementations would take input from the real world and logically decide what method of action would be the best to take, just as the players in the simulation described above take input from its simulated surroundings and determine the best route to success, and just as humans take input from their surroundings to formulate the best route to accomplish their goals. This program showed us the possibilities to which a computer could be applied, and has introduced us to the type of algorithms that might be necessary, involving generalization and evaluation, should a more complex AI be developed.

Acknowledgments

We would like to take the time to thank all those who were involved in with our project, specifically Mr. Simon for sponsoring us and working with us.

We would also like to express our gratitude to all the staff at Los Alamos National Laboratories and New Mexico Technet.

* * *

Bibliography

The OpenGL Handbook. www.openGL.org/Handbook. Jan 2001.

Bronson, Gary J. A First Book of C++. 2000 Cole Publishing Company.

***

Appendix I

OpenGL is a three dimensional graphics rendering Application Programming Interface (API), which is simply a header file and functions that can be called in C++. These functions coordinated and conduct any output rendering. This implementation allowed us to initialize a window in which two planes were drawn, and were visual representations of the simulation as the AI moved the planes in the environment. Obviously, this 3-Dimensional rendering of the planes as they flew along on their flight patterns provided a huge advantage over an alternative

text based system. Essentially, OpenGL provided us with the procedures by which to see what the computer was accomplishing, rather then simply trying to decipher X, Y, Z outputted coordinates, as undoubtedly, this would have been extremely inefficient to decide how effective each run of the simulation was. To create a ship, we utilized the exported vertices and indices, which code for the order in which the vertices should be connected, of a plane we designed in a 3D-rendering program, and imported these from a file to be drawn (see Fig 2). We were also able to apply a color to each plane so as to be able to distinguish them in flight and make it apparent when one plane was under fire and being hit by enemy bullets. To output this, however, is actually quite complicated. We had to learn and understand the many nuances of OpenGL's programming technique, that is to say, using what lines of code could a scene be outputted. In its simplest form, this is actually quite simple and is outlined in the OpenGL handbook, but to provide a more efficient and functional display, we utilized Macintosh Input Sprockets along with OpenGL. In the end, we were able, through rather extensive coding, to visualize our creation.

Fig. 2b. - Vertices

```
Vertices: 822
268.24 6.668 -52.415
-231.378 11.418 -70.806
268.464 7.761 -52.449
20.003 45.975 -77.25
267.386 5.488 -52.283
```

An example of the Vertices of the plane. Specified by and X, Y, and Z coordinate.