

Tech Lab Project

Ravi Kappiyoor, Charles Albert, Robert Leith, Alberto Pareja-Lecaros, Akshay Joshi

June 8, 2006

Abstract

The purpose of this project is to make a game that simulates a war fought in outer space. There will be several solar systems through which the ships can fly through, and can be controlled by one army or another, depending on who has more ships in the area. If a particular army has control of the area, then it can make ships, build infrastructure, etc.

This project has two main goals, for the five of us to figure out how to work together as a group, and for each of us to figure out how the respective parts of our game work.

1 Introduction

1.1 Purpose

As stated above, this project has 2 main goals. They are for the five of us to figure out how to work together as a group, and for each of us to figure out how the respective parts of our game work.

In the work force, people don't work in their own isolated little part of the world, but in the real world, where they need to work as a group to make one working product. There are several products in the market today that, while they have features that work great by themselves, when put together make a mediocre final product. Our goal is to make sure that that doesn't happen with our program, that it has great features that work by themselves, and for our final product to be great as well.

The other goal of our group was for each of us to figure out how the respective parts of our game work. For example, the group member assigned the physics engine of our game plans on majoring in physics when he goes to college. All of us are working on parts that we think will be useful to us later on in life.

1.2 Scope of Study

1.2.1 Physics Engine

1.2.2 Artificial Intelligence

The project will require the programming of numerous different strategies and tactics, such as being in a certain sector with respect to the enemy ship, or a certain shell distance, etc; it is not so much of a strategy, but more of a goal state for the ship to reach, and each move the ship makes is towards achieving said state. Then in each instance, the remote game has a random array of goal states from which the NPCs can draw upon, every couple minutes the states with the stronger record will reproduce while the states with the weaker record are jettisoned. Difficulty levels can be added in by changing the accuracy, aggressiveness, and decision time of the NPCs.

1.2.3 Graphics Engine

1.2.4 Server

1.2.5 Game Engine

This part of the project contains the information that all of the other packages use. For example, the class Universe is where all of the objects are held such as rocks, ships, and their movements. In order to do all of this, a basic object-oriented design structure is needed. In order to make this design, it is helpful to use something such as the Unified Modeling Language (UML). Learning UML, along with figuring out when and where to call the other packages to run is the scope of study for this part of the project.

2 Procedure and Methodology

2.1 Physics Engine

2.2 Artificial Intelligence

The first task will be to identify as many goal states as possible, as in should one attack from a sector, shell, parallel, perpendicular, kamikaze, etc. Then to create functions to analyze the environment that are of direct use to assessing the current state and achieving the goal state, such as if there is an enemy ship is alone and in the distance, should the NPC run away, wait for help, attack from a certain sector, from a certain shell, parallel, perpendicular, etc. Then to code what responses the NPC should have towards achieving each goal state, intercept code, aiming code, etc. After this, testing must be done to make sure that each goal state function is working properly, and to make the necessary changes. The first testing cycle will be with controlled variables, i.e. a ship that is stationary; the object will be to see if the ship reacts appropriately. Next will be with a ship moving in a straight line. Then

with one enemy ship or player at a time. Followed by multiple enemy ships and players, and, finally, with multiple factions and players.

2.3 Graphics Engine

2.4 Server

2.5 Game Engine

The first task was to design the different parts that were needed to make a game engine. This was done using UML. In order to do this, we first needed to figure out which classes should do what. For example, should the Ship class contain things such as turn jets and a control panel, or should those be completely separate classes?

The next task was to actually code everything that had been designed, along with documenting the code as we went along. Along the way, however, some flaws were found with the design. As such, it was necessary for us to go back to the original design and rework some of it, and make changes accordingly.

3 Results

The results for this project are preliminary since we are not yet finished and plan to work on this project through college. However, currently, we seem to be working relatively well together, and each of us seem to have an adequate grasp on our respective parts of the game.

4 Conclusion

The conclusion for this project is also preliminary since we are not yet finished, however, we believe that we have learned to work together as a group somewhat effectively, and each of us have learned something valuable in our own respective parts of the game.

References

- [1] Introduction to Software Testing by Paul Amman and Jeff Offut. In <http://ise.gmu.edu/~ofut/forTJ/>
- [2] About.com by John Kappman. In <http://www.about.com/>
- [3] Brad Appleton's Software Process Links, author: Brad Appleton. In http://www.cmcrossroads.com/bradapp/links/sw-proc-links.html#Proc_Tut_Train_Pub
- [4] UML Toolkit by Hans-Erik Eriksson and Magnus Penker.

- [5] The Unified Modeling Language Reference Manual by James Rumbaugh and co.
- [6] A Simple Way to Read XML in Java by Kiran Pai. In <http://www.developerfusion.co.uk/show/2064/>