Motor Schema’s Software Test Plan

# AI Robotics Final Project Phase 3 – Fall 2013

### John DiLeo

### David Mullen

### Eric Wilcox

1. **Introduction**

The purpose of this document is to provide information on the structure and execution of our software testing plan. We will show how we tested the software we developed, where to find the testing environment and exactly what tests were and were not run, including the reasons. This plan was implemented for the C# program file labeled “BlockCrowd2.cs” developed and run using MonoDevelop, the IDE which accompanies the Unity game engine. It should be noted that the software was developed for the Unity game engine to provide a simulator with which to test the behaviors for multiple Unmanned Arial Vehicles (UAV’s).

1. **Testing**
   1. **Unit Testing**

We provided various functions for individual unit testing through about the first half of what we saw was necessary. Those functions include:

1. KeepHeight
2. HoldCenter
3. Follow
4. AvoidCrowd
5. ThreatenRand2D
6. Watching\*

\*Watching is only a partial test implementation as we begin to see changes going further into these functions flowing from actual unit testing to integration testing. Accurately testing the behavioral functions requires so much framework constructed that individual unit testing isn’t really applicable. The functions for the behaviors which were not unit tested include Approaching, Threatening, Avoid, and BlockCrowd2.

**2.2 Integration Testing**

We started off with implementing what was already in place from the previous project (phase 2) and adapted from there. Implementing the new code consisted of making relevant changes then testing out the simulation for crashes. Testing occurred in phase with development during this stage. We had several interesting events occur during this process including some hard to track down bugs, but we believe we’ve eliminated them at this point.

* 1. **Testing Scripts**

We’ve put together a testing script that can be run, “Program.cs”. This creates some of the necessary framework for testing and throws values at the functions testing what are known results. This is by far not an exhaustive test but the code can easily be altered to call for and return results on any set of expected data with known results. The code for this script will be included in the same folder as this document for evaluation purposes.

1. **Additional Tests**

Additional testing was done more in style with the previous testing model we used. These relate to verification and validation testing. Multiple debugging statements are peppered throughout our code for various console logging. In the event of exceeding the bounds of critical elements console statements are logged. In successful runs these message should not be visible and would not be visible if the testing environment is changed to release from debug mode.

Long term runs of the simulation with no acceleration cap resulted in a highly effective blocking behavior. Running over 10 hours gave us an excess of 25k attempted blocks with about 28 failed objects. This has a 99% effective block rate.

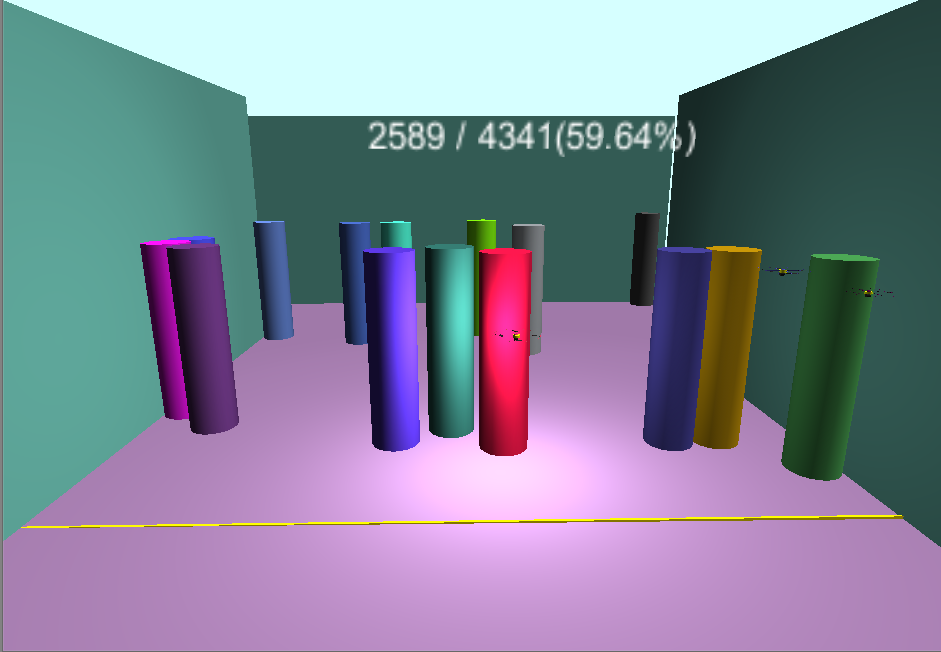
We noted some movement discrepancy in the UAV’s as they approached the crowd objects. For the sake of realism in the area of validation we capped the change in velocity for a given UAV. While this cap aids in simulating a more realistic situation it also lowers our overall success rate on blocking the crowds. Setting the velocity increase to a maximum of 4% over any iteration of the update caused the effectiveness of the overall blocking to go to approximately 60%. This figure goes up quickly as the velocity scale is changed. With 5-10% velocity change allowed the blocking goes up to 80-90% again. This aids additionally in the verification of the behavior and due to the movement speed of evaluated hardware we should be able to see a related effectiveness follow suite. Some of this is due to changes we implemented in the crowds themselves however. If porting this to real world devices this should be an element that could be removed as a UAV will have a cap to its acceleration by the nature of the hardware.

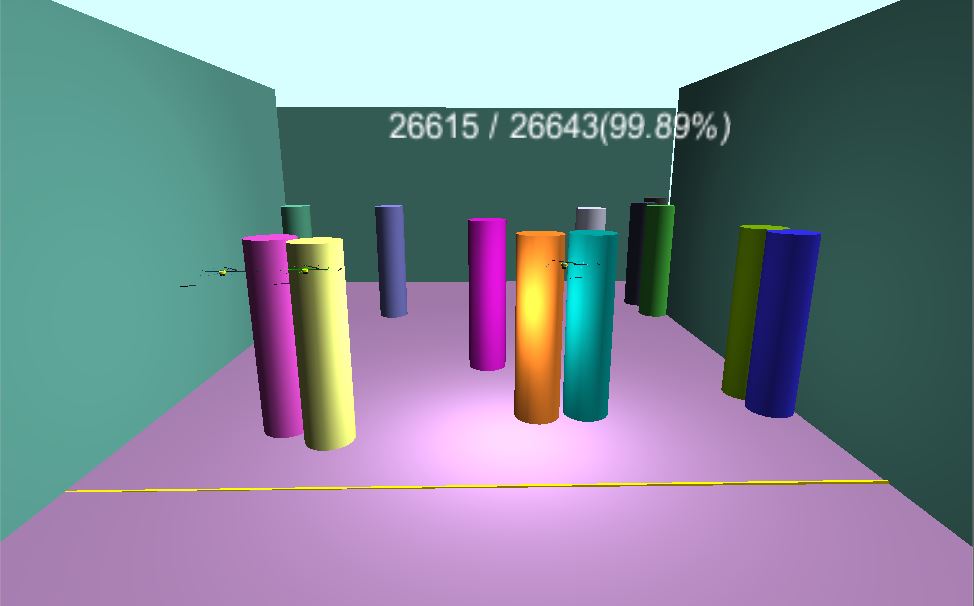
Additional long term runs could be made to evaluate the effectiveness of the system with the acceleration limits but an acceleration model should be used to provide for closer to real world results. We believe these tests have demonstrated the verification of the designed system.

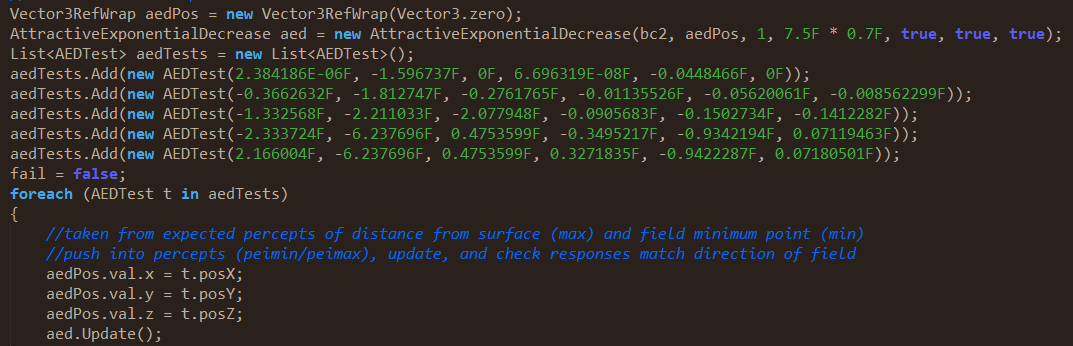
**3.1 Crowd Changes for Testing**

A significant change to the crowds has been made to aid in testing over longer simulation runs. We took an interesting approach idea from Yue Zhuo and Han Wang’s group and changed how the automatic crowds are populating. We now have 5 new crowd objects populating every 5 seconds with a randomized velocity and direction towards the block line. These crowds also now disappear much more quickly after they have been blocked or passed the line also so as to not interfere with new crowds. This effect provides for what we believe is a much better simulation which can be run over extended periods of time to test for proper behavior. Additionally the crowds have changed to respond differently to different types of behaviors from the UAV’s, again adding to the realism of the testing. When a UAV does not get in front of a person object in time the person can make it across the block line, and they do not simply turn around when close enough.

1. **Screen Captures**



****



1. **Overall results**

The testing for this phase of the project was more complete then the previous phase, but as it is well known in software engineering, you can never have enough testing to track down all the bugs in a system. Extensive runs of the simulation combined with oversight during verification and validation as well as more in depth unit and integration testing lead us to believe this project has been tested successfully. Given more time for this project we still believe more tests could be developed to look for additional drawbacks we have not yet identified.

1. **References**

All previous posted team projects contributed to some extent to this revision. The groups whose projects we evaluated are recognized (listed as the group evaluations listed them):

1. Ahamd Hawkins Probe
2. Firodiya Liu Kaul
3. Huang Anh Vedam
4. Li Pan Wang
5. Mandal Parulker
6. Nowak Wilde
7. Wang Zhou

IEEE Test Plan Outline." School of DCU Computing. Dublin City University, n.d.  Web. 7 Dec. 2013. <http://www.computing.dcu.ie/~davids/courses/CA267/ieee829mtp.pdf>.