Project Evaluation Report

Artificial Intelligence for Games

Academy of Interactive Entertainment

Lewis Comstive, July 2021

# Project Overview

Use this section to give a high-level overview of your project and its development.

Briefly (in one or two sentences) describe what your project is.

Then address the following questions in report form (i.e., write well-formed paragraphs that have a logical flow, taking note to avoid spelling or grammatical errors).

* Did development adhere to your pre-planned timeline?
* What A.I. algorithms did you implement, or attempt to implement?
* What difficulties did you have in implementing these algorithms?  
  Possible difficulties worth mentioning might include:
  + Difficulty in understanding all details of the algorithm
  + Difficulty with programming / debugging
  + Performance issues, including memory management
  + Unexpected or incorrect agent behaviour
* If you did not experience difficulties implementing the A.I algorithms, then explain why you feel these algorithms were easy to implement

*Slime Survival™* is a simulation involving innocent slime creatures that are just trying to live in harmony, until *skeletons* start attacking and destroying their population!

Initial development of the application went well, which was then stretched out over many tedious hours with minor bugs and issues. With the plan to implement behaviour trees as the “brain” of the NPCs, accompanied with the A\* (AStar) algorithm for pathfinding around a grid-based map, I went full steam ahead and developed the pathfinding first – to stress test was a 1000x1000 unit grid that a single point could use A\* to navigate to any other point within a few frames.

Next on the list was implementing behaviour trees, which had very little information on implementation but rather *a lot* of higher-level concepts (predominantly Unreal Engine’s system…).  
Gamasutra came in clutch as per usual, with [Chris Simpson’s article](https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php) it started making more sense and I was able to develop (*what I believe is*) a solid, flexible framework that used a hashmap per tree as a contextual pool of information for each node (*akin to a blackboard, but per AI instead)*.

While trying to test the behaviour tree I realised that a physics library would be beneficial, so [*Box2D*](https://box2d.org/) was added as a submodule to the GitHub repository; now the basic game framework had collisions and the ability to raycast, which was then implemented in a behaviour tree node.

With the blackboard implemented and some chicken sprites running after each other using viewcones with raycasting, everything was coming to life. Next was to combine the tree with pathfinding, with absolutely no idea on how to do so...

After many, many tedious hours of trial and error the behaviour nodes now had options for navigation via pathfinding, as well as finding a path to certain other entities within the scene.

With enough time an animation system was planned to be implemented by changing a value in the contextual map, akin to states, that the sprite would read and change the spritesheet offset it was reading from; but alas, spending around 60 hours on this project over two weeks I had to let personal life commitments take priority closer to the submission date.

There is a bug that has lingered throughout the codebase for the past week or so that has caused the application to crash randomly, but *only* in release mode on the Windows platform! Tests have been done in Debug mode and Release mode w/ Debug info, but the crash only happens when there are *no debug symbols loaded* – the issue was not present while testing on an M1 Mac OS system.

Another bug that occurred was a very high memory usage, for instance after spawning about a hundred units the memory usage rose to sometimes a gigabyte or more – once Visual Studio crashed and reported 164GB used by the application (on a 32GB memory machine)!  
The issue was resolved thanks to the magic of pointers, and now the simulation has a consistent sub-100MB memory footprint.

# Performance Analysis

Use this section to analyse the performance of your algorithm(s) or techniques.

Provide a brief description of the memory footprint of your agent class(es). Explain if this is efficient or could be improved upon.

Analyse you A.I algorithm and identify any performance bottlenecks or places for improvement. If possible, list the efficiency of your algorithms using Big O notation.

Possible topics for inclusion in this section are:

* Is it efficient for a lot of agents to use the same pathfinding algorithm?
* Should pathfinding be done every frame?
* How can you improve the performance of your pathfinding algorithm in the context of your game?
* Are all your algorithms efficient? Why/why not.
* How many agents could you have in your game before you start seeing performance issues, and have you tested this?

# Future Improvements

Did you get enough time to completely implement your A.I. as planned? What work did you not complete (and why)?

Can you see ways to improve your program/algorithms?

Were your algorithms good choices? Do you plan to use them in future projects (why/why not)?

There is no word limit for this report, but it is expected that you provide enough detail for 1 to 2 full pages.