# ***Using A Behaviour Tree to Control the Behaviours of Artificial Intelligence in a Stealth Environment***

**Development Blog:** [**https://henryfinalproject.blogspot.com/**](https://henryfinalproject.blogspot.com/)

**Outline of Work:**

The aim of my *Final Project Artefact* is to test and improve my knowledge of *Artificial Intelligence* (AI) specifically how *Behaviour Trees* (BT) can be used to control an agent’s decision choices in a stealth environment. I will be using the *Unity3D Game Engine* to build the project, this will allow me to quickly build and iterate the test environment, I will also be using *Unity* to gain access to the built-in *NavMesh* tool which “can be used to do spatial queries, like pathfinding and walkability tests(“Unity - Scripting API: NavMesh,” 2018).

To begin the project, I will create a small test scene within *Unity* to learn how to use the built-in *NavMesh* system. To test my knowledge of the *NavMesh* I will start with a simple scene and begin increasing the complexity to test Line of Sight (LoS) to objects around the scene, and watch the agent move to set locations. If there is time as a stretch goal, I would like to try and implement my own pathfinding algorithms.

Once the *NavMesh* has been set up, I will start building a simple BT to allow the agent to enact certain behaviours, such as a *seek* state causing the agent to move to a certain location, and a *patrol* state causing the agent to wander around the scene. I have chosen to implement a BT since they are easy to expand upon if “the behaviours are written generically enough, many different agents could share not only the behaviours, but even whole trees.” (Dawe, 2014, p95).

After implementing a basic BT, I will begin implementing a *Blackboard* *System,* this will allow each agent to have “a shared memory space which various AI components can use to store knowledge that may be of use to more than one of them” (Dill, 2014, p66). Using the *blackboard*, I will be able to simulate agents talking to each other, such as sharing the player’s last known position. The blackboard can also be used to store expensive checks such as pathfinding or LoS so the agents “can run the check once and then cache it on the blackboard” (Dill, 2014, p67).

Once I have a working BT and *blackboard* I will begin expanding the agent’s available behaviours, thereby improving the illusion of intelligence. I will do this by introducing decorator, sequence and selector nodes as described by Champandard and Dunstan (2014, pp78-81). This will allow the agent to make choices based on what is currently happening around them, executing behaviours when certain conditions are met.

Sequence and selector nodes are derived from a base class called a composite node, these can be used to make “more interesting, intelligent behaviours by combining simpler behaviours together.” (Champandard and Dunstan, 2014, p78). This will allow the agent to make decisions by traversing the tree using predefined conditions, this can be used to break the *seek* state down further by using a selector node, with the children behaviours being *hunting player*, *searching last know position,* and *investigate*. While they all perform the basic *seek* state, they will behave differently depending on the conditioning, for example *searching last known position* will cause the agent to move towards where the player was last sighted acting cautiously, ready to run to the nearest cover, whereas the *investigate* state will run if a trigger has been caused such as a noise near the agent, but the agent did not see the player, then the agent will move towards the location in a confused state.

**Project Rationale**

As part of the *Final Project Module* for Computer Games Programming we are asked to consider what area of games programming we are interested in on a personal, academic and work-related level. The area that I have been interested in doing since the start of semester two during my second year of University is Artificial Intelligence (AI).

A well implemented AI system can impact heavily on the player’s overall experience of a game, for example if the system is too “dumb” and keeps running into walls, this will “breaks [*sic*] the illusion of intelligence” (Hilburn, 2014, 102). However, the game can also be too “smart”, leaving the player feeling as if the game has cheated, both outcomes will result in the game seeming unfinished or broken.

I have chosen to Implement an AI artefact for my Final Project at a personal level to improve my own knowledge of how an AI system works and implement that within a game setting of my choice. I would like to implement a *behaviour tree, blackboard* and *steering behaviours* to make the agent autonomously move around the scene using the *NavMesh*.

Academically AI interests me as I would like to see the different AI systems across multiple games, such as the Halo series, Alien Isolation and the Sims work, how the AI affects player enjoyment and how I can implement and test these examples within my project.

The reason I would like to expand upon my knowledge of AI in a work-related situation is because although there are quite a few jobs searching for AI programmers, the games industry itself is very competitive and requires potential candidates to be up-to-date with multiple AI systems. With the AI sector always growing AI programmers are starting to create smarter AI-bots for eSports, “however, unlike a human playing against a computer, AI Gaming promotes humans creating bots to challenge each other in games to see which is the most technologically advanced” (Heitner, 2018).

**Common Requirements of *Artificial Intelligence* Job Postings (Appendix B)*;***

* Proficiency with C# or C++.
* Broad familiarity with AI in game development.
* Experience with pathfinding and steering behaviours
* Knowledge of state machines and behaviour trees.

I will be tailoring my final project to show my understanding and skill within these areas by building a behaviour tree in C# using Unity3D.

**Marking Criteria**

I would like to be marked on;

* My use of Object Orientated Programming
* Structure of Behaviour Tree
* Robustness of AI agents
* Intelligence of AI at runtime – Believability.

I would not like to be marked on;

* Models
* Design of the test scene
* Player Controller

**MVP**

A test scene built in Unity with AI using a basic tree structure.

**Realistic Goal**

A test scene built in Unity with AI using a more advanced tree with states to make the agent seem intelligent.

**Stretch Goal**

Create a basic game level within Unity where the player must avoid the AI to reach the end of the level.

Appendix A:

|  |  |  |
| --- | --- | --- |
| **Use this grid to plan your project milestones** | | |
| **2018–19** | **Week #** | **Milestone Deliverables and Tasks** |
| 24 – 28 Sept | Week 1 | Research into AI Techniques |
| 1 – 5 Oct | Week 2 | Research into AI Techniques |
| 8 – 12 Oct | Week 3 | Start Proposal |
| 15 – 19 Oct | Week 4 | Research AI in Other Games |
| 22 – 26 Oct | Week 5 | Develop Proposal |
| 29 Oct – 2 Nov | Week 6 | Receive feedback on Proposal and Update Accordingly |
| 5 – 9 Nov | Week 7 | Submission of Final Project Proposal: by Noon, Friday 9th Nov 2018 |
| 12 – 16 Nov | Week 8 | Set up Unity Project with a Basic Scene to Test NavMesh |
| 19 – 23 Nov | Week 9 | Implement Line of Sight Functionality |
| 26 – 30 Nov | Week 10 | Build Simple Behaviour Tree with Seek and Patrol States |
| 3 – 7 Dec | Week 11 | Contingency Week |
| 10 – 14 Dec | Week 12 | Start Implementing a Basic Blackboard System |
| 17 – 21 Dec | **Mid-Winter Festival** | |
| 24 – 28 Dec |
| 31 Dec – 4 Jan |
| 7 – 11 Jan | Null |  |
| 14 – 18 Jan | Null |  |
| 21 – 25 Jan | Week 13 | Test Blackboard Across Multiple Agents |
| 28 Jan – 1 Feb | Week 14 | Update Unity Project Scene to Place Triggers Around the Scene |
| 4 – 8 Feb | Week 15 | **Seminar Presentations TBC** |
| 11 – 15 Feb | Week 16 | **Seminar Presentations TBC** |
| 18 – 22 Feb | Week 17 | Breakdown Seek Behaviour to Hunt, Alerted, Investigate |
| 25 Feb – 1 Mar | Week 18 | Continuation from Previous Week |
| 4 – 8 Mar | Week 19 | Breakdown Patrol Behaviour to Have an Alerted and Normal State |
| 11 – 15 Mar | Week 20 | Continuation from Previous Week |
| 18 – 22 Mar | Week 21 | Implement Behaviours to Allow Agents to Interact with Each Other |
| 25 – 29 Mar | Week 22 | Implement Security Cameras with Triggers Agents Will React to in the Scene |
| 1 – 5 Apr | Week 23 | Polish Unity Project |
| 8 – 12 Apr | Week 24 | Submission of Final Product and Blog: by Noon, Friday 12 Apr 2019 |
| 15 – 19 Apr | **Spring Fertility Festival** | |
| 22 – 26 Apr |

Appendix B:

|  |  |
| --- | --- |
| Job Title | Job Posting Link |
| A.I. Programmer | <https://www.indeed.co.uk/jobs?q=artificial%20intelligence%20games&vjk=b6b3211389efd150> |
| Principle AI Programmer | <https://www.indeed.co.uk/jobs?q=artificial%20intelligence%20games&l&vjk=39b41fe51625325a> |
| AI Programmer | <https://www.indeed.co.uk/jobs?q=artificial%20intelligence%20games&l&vjk=666be6966f1dd712> |
| Graduate Games Developer | <https://www.indeed.co.uk/viewjob?jk=73b5b06c4aa764e7&from=tp-serp&tk=1crprd0q69npb802> |
| Game Developer (London, UK) | <https://www.indeed.co.uk/jobs?q=artificial%20intelligence%20games&start=10&vjk=ca2025123ea0b8dd> |
| Junior Simulation Developer Unreal Unity | <https://www.reed.co.uk/jobs/junior-simulation-developer-unreal-unity/36044361?source=searchResults#/jobs/unity-ai-jobs> |
| AI Software Engineer (Frostbite) found on Electronic Arts Guildford | <https://www.monster.co.uk/jobs/search/?q=games-ai-programmer&cy=uk&client=power&jobid=76144984-bb61-448d-aa0c-3d3156709cc8> |
| AI Software Engineer (Frostbite) | <https://ea.gr8people.com/index.gp?method=cappportal.showJob&layoutid=2092&inp1541=&opportunityid=152737&sid=https://www.monster.co.uk/jobs/search/> |
| AI Programmer – Games – Nottingham UK | <https://www.google.co.uk/search?q=ai+programmer+jobs+in+games&rlz=1C1CHBF_en-GBGB796GB796&oq=ai+programmer+jobs+in+games&aqs=chrome..69i57j69i60l2.4700j0j4&sourceid=chrome&ie=UTF-8&ibp=htl;jobs#htidocid=of8KtktB4q89ZalNAAAAAA%3D%3D> |
| Junior AI Programmer | <https://www.google.co.uk/search?q=junior+ai+programmer+jobs+in+games&ibp=htl;jobs#fpstate=tldetail&htidocid=KafJHi8VDCGg8xEQAAAAAA%3D%3D&htivrt=jobs> |
| AI Programmer – Console and PC – Oxford | <https://www.google.co.uk/search?q=junior+ai+programmer+jobs+in+games&ibp=htl;jobs#fpstate=tldetail&htidocid=BDVMlSlTW26vqC72AAAAAA%3D%3D&htivrt=jobs> |

**Bibliography**

Buckland, M. (2005) *Programming Game AI by Example*. Wordware Pub.

***Summary of Book***

Programming Game AI by Example was a good starting point for my research, having used the book before in my previous years, I knew I could find relevant information here. For my final project I have used Buckland for research into finite-state machines, hierarchical finite-state machines and steering behaviours to create autonomous agents. Buckland goes into a good amount of detail in both subjects referencing other notable pieces of work such as Craig Reynolds.

Champandard, A. J and Dunstan, P. (2014) “The Behaviour Tree Starter Kit” in *Game AI Pro,* edited by Steve Rabin. Boca Raton, FL: CRC Press 2014, pp. 73-91.

***Summary of Article:***

In chapter 6 of Game AI Pro, Champandard and Dunstan provide information to start creating a behaviour tree, giving examples of how to create a first and a second-generation tree, keeping the memory optimised, such as controlling when the tree updates. They then explain the concepts of composite and selector nodes and how to use them, finally they discuss ways to make the behaviour tree event-driven.

Champandard, A. J (2007, July, a). The Flexibility of Selectors for Hierarchical Logic. *AiGameDev.* [online] Available at http://aigamedev.com/open/article/selector/ [Accessed 2 Nov. 2018]

***Summary of Article:***

In this online article Champandard explains how important selector nodes are in a Behaviour Tree. Champandard then goes on to talk about how selectors can be configured using probability and priority to select the appropriate child, giving designers some control over how the tree executes.

Champandard, A. J (2007, July, b). The Power of Sequences for Hierarchical Behaviors. *AiGameDev.* [online] Available at http://aigamedev.com/open/article/sequence/ [Accessed 2 Nov. 2018]

***Summary of Article:***

Champandard explains how vital sequence nodes are, and how they follow the execution order of their children making them easy for designers to use. Champandard then goes on to talk about the robustness of a sequence node, and how the engine should only use one, that is able to handle all the codes returned by its children.

Champandard, A. J. (2007, September, b). Understanding Behavior Trees. *AiGameDev.* [online] Available at http://aigamedev.com/open/article/bt-overview/ [Accessed 2nd Nov. 2018]

***Summary of Article:***

This article discusses the differences between finite-state machines (FSM), hierarchical FSM and behaviour trees (BT). It explains how a BT is modular and able to add extra behaviours without causing too much trouble while still being very powerful and accessible.

Dawe, M. (2014) “Real-World Behaviour Trees in Script” in *Game AI Pro,* edited by Steve Rabin. Boca Raton, FL: CRC Press 2014,pp. 93-98

***Summary of Article:***

Dawe explains how powerful behaviour trees can be to control an agent’s decision-making process using example from *Kingdoms of Amalur: Reckoning* and explaining how using an on\_enter and on\_exit function within the behaviour can help perform functions on first time set up or a clean-up function as (Buckland, 2005) also talks about.

Dill, K. (2014) “Structural Architecture – Common Tricks of the Trade” in *Game AI Pro,* edited by Steve Rabin. Boca Raton, FL: CRC Press 2014, pp 61-71.

***Summary of Article:***

A paper that discusses many different common techniques that are used within the AI industry. Goes into details about common architectures such as; *goal-oriented action planners (GOAP), finite-state machines(FSM)* and *behaviour trees (BT).*

Game Developers Conference*. (2015). Less is More: Designing Awesome AI for Games*. [online] Available at: https://www.youtube.com/watch?v=1xWg54mdQos&t=51s [Accessed 18th Oct. 2018].

***Summary of Article:***

In this short video Kimberly Voll explains the problems she faced while trying to overcomplicate the AI on a game she is working on called ROCKETSROCKETSROCKETS and how taking a step back and not overthinking the problem can produce you with the solution, always make sure the AI is performing believable functions.

Game Developers Conference*. (2015). Tales from the Trenches: AI Disaster Stories.* [online] Available at: https://www.youtube.com/watch?v=\_\_5whYgSTV0&t=959s [Accessed 3rd Oct. 2018].

***Summary of Article:***

During this section of the GDC known AI programmers took to the stage to share their stories and failures that they have experienced or witnessed during their career and hope that having the knowledge can help prevent others from doing the same, such as commenting a second of code out and forgetting about it.

Game Developers Conference*. (2015). The Simplest AI Trick in the Book. [online]* Available at: https://www.youtube.com/watch?v=iVBCBcEANBc&t=23s [Accessed 15th Oct. 2018].

***Summary of Article:***

During this conference in 2015 5 different AI programmers stood up and explained some of their tips and tricks that they use to help them develop the AI in their game.

Heitner, D. (2018, May). How Artificial Intelligence May Further Develop the Fast-Growing eSports Industry. *Forbes*. [online] Available at https://www.forbes.com/sites/darrenheitner/2018/05/13/how-artificial-intelligence-may-further-develop-the-fast-growing-esports-industry/#a8db3d538581 [Accessed 15th Oct. 2018]

***Summary of Article:***

This article talks about how AI might be developed in the future due to recent developments in areas such as *Machine Learning* and *The Internet of Things (IoT)*.

Hilburn, D. (2014) “Simulating Behaviour Trees” in *Game AI Pro,* edited by Steve Rabin. Boca Raton, FL: CRC Press 2014,pp. 99-111

***Summary of Article:***

Hilburn discusses the strengths and weaknesses between Behaviour Trees and Planners. As per his examples, Behaviour Trees are rigid giving designers and programmers control over what the AI can do, while planners instruct the AI in what it should be doing using heuristics to formulate a plan of behaviours stung together to make an action.

Isla, D. (2005, March). GDC 2005 Proceeding: Handling Complexity in the *Halo 2* AI. *GAMASUTRA* [online] Available at http://gamasutra.com/view/feature/130663/gdc\_2005\_proceeding\_handling\_.php?page=2 [Accessed 16th Oct. 2018]

***Summary of Article:***

In this article on Gamasutra, Isla talks about how Bungie handle the AI in their game, using

A hierarchal finite-state machine / behaviour tree. Using this they can compare behaviours against one another using different methods, such as a prioritized list, sequential and others. Because of this they can implement behaviour impulses and other strategies to make the AI react to the players actions and current state of the world.

Unity – Scripting API: NavMesh. (2018) Available at https://docs.unity3d.com/ScriptReference/AI.NavMesh.html [Accessed 24th Sep. 2018]

***Summary of Article:***

This webpage gives directions on how to use the NavMesh and related documentation pages. It also gives directions to some of the methods that can be used in code to control the NavMesh agent.