

Pathfinding

06.05.2022

**─**

Ryan “Lunar” Bassil

# 

# Changelog

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0.0 | 06/05/2022 | Initial Setup, specifying document concept and author |
|  |  |  |
|  |  |  |
|  |  |  |

# 

# Contents

[Changelog 2](#_Toc100306763)

[Contents 3](#_Toc100306764)

[Introduction 4](#_Toc100306765)

[Rationale 4](#_Toc100306766)

[Background 4](#_Toc100306767)

[Terminology 4](#_Toc100306768)

[Non-Goals 4](#_Toc100306769)

[Proposed Design 4](#_Toc100306770)

[Software and Hardware Requirements 4](#_Toc100306771)

[Research 4](#_Toc100306772)

[System Architecture 5](#_Toc100306773)

[Architecture Diagram 5](#_Toc100306774)

[Interface/API Definitions 5](#_Toc100306775)

[Risks 5](#_Toc100306776)

[Alternatives 5](#_Toc100306777)

[Sign Off 6](#_Toc100306778)

[Testing 7](#_Toc100306779)

[Errors and Buggs 7](#_Toc100306780)

[Evaluation 7](#_Toc100306781)

[Reflection 7](#_Toc100306782)

# 

# Introduction

## Rationale

The purpose of this TDD is to learn and understand more about artificial intelligence and search algorithms by utilising a number of different game engine objects to present challenges to the AI agents for them to overcome while using the algorithm(s).

## Background

The object-oriented programming language, C#, has had many different versions in the past forming its basis roughly off C. Different versions slightly affects the ways in which some code concepts are written. In this project I will be using version 9.0.

Like C#, Unity has had many iterations in the past and continues to update to this day, but for this project I will be using Unity version 2020.3.27f1

According to a [Harvard University blog post](https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/), and Wikipedia, AI and machine learning have been around for a few centuries, though the foundation of AI as an actual science was coined in 1956. Alan Turing, a brilliant mathematician, was highly influential to its foundation and development. Nowadays AI is being used in almost anything with a processor – watches, phones, computers, TVs, cameras, etc. As of 2022, the world’s first robot AI was born in 2018 and still lives to this day.

In this project I will be designing and producing a working project within Unity from scratch, starting with this document. I will be using Unity’s built in AI functions and incorporating the A\* search algorithm.

## Terminology

**C#** - An object-oriented programming language

**Unity** – a game engine used in the production of games and software in the IT industry

**TDD** – a Technical Design Document (this document) is a document that aids in the critical analysis of a problem that is presented to a development team and the proposed solution, while also communicating priority and effort of tasks, and impact with various stakeholders.

**AI** – Artificial Intelligence is human intelligence demonstrated in machines, able to reason and carry out tasks based on inputs as a human would.

**OS** – Operating System of a computer or electronic device

**Processor** – essentially a computer’s brain for calculating and input/outputs of the software running on the computer

## Non-Goals

* Add sound effects and dialogue
* Add a timer
* Add particle effects

## Proposed Design

The project will have a few key elements from classwork implemented such as the NavMesh and related components, as well as some of the code structure. There will be 3 AI agents: two key searchers and one agent searcher. The project will be structured within a constructed maze that the AI must navigate through; the key searcher agents will search for key pickups, and the agent searcher will search for the other AI agents.

The objective for the key searchers is to retrieve one key each and get to the end of the maze. The objective for the agent searcher is to find the other AI agents and stop them achieving their objectives.

The agent searcher will be placed randomly on every scene play, and the key searchers start in the same place every run.

## Software and Hardware Requirements

**OS**: Windows 7 or Greater

**Processor**: Intel Core i3-12100, AMD Ryzen 3 3300X, or equivalent

**Memory**: 4 GB RAM

**Graphics**: Integrated Graphics Chip (minimum Nvidia GTX 700 series or AMD Radeon R7 200 series)

**DirectX**: Version 9.0

**Storage**: 1 GB available space

# Research

* Find 5 major AI jargon of your choice used within the games industry. List and describe the major AI terms.

**Machine Learning**

Essentially a field of Artificial Intelligence, in a broad sense, that defines and develops a machine’s ability to imitate human behaviour in complex task learning.

**Data Science**

An interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from the data in various forms, both structured(tables) and unstructured (images, videos).

**Pattern Recognition**

It is the automated recognition of patterns and regularities in the data.

**Computer Vision**

A field of AI that applies machine learning to teach computers how to interpret data from captured images and video.

**Deep Learning**

A way for an AI, or machine, to practice machine learning in a way that imitates how humans gain certain types of knowledge and behaviours.

**AI Agent**

The entity that performs, executes, and behaves with the AI code and information. Within this project, it shall be a simple game object with the attached script.

* Describe what an algorithm is, and how do you use algorithms when programming?

Algorithms are essentially step-by-step methods or procedures that are used for calculations, data processing, and automated reasoning.

To use algorithms in code you would need code that calls and utilises those calculations and instructions to benefit the program, usually by passing other variables in or using the result of the algorithm for use in the code in which it was called. Often algorithms are written as methods in C#.

* Define what is a search algorithm and a sorting algorithm.

A search algorithm is a series of steps of calculations to locate specific data within a collection of data, such as finding a specific word in a book.

A sort algorithm is a series of steps of calculations to analyse all the data in a collection and organise it based on specific requirements, for example, organising all the words in the book from shortest to longest in length.

* What is big O notation?

Big O notation as a concept can be applied to many philosophical categories of life, however when talking about AI specifically it relates to the upper boundaries or the largest scale of the code or data within the AI. In other words, it is used to describe the computation of executing time or space requirements grow based on the size of the input, and generally speaking, the “Big O Notation” is account for the worst case scenario and the largest limit of data and processing within time and space for the AI in question.

* Outline the difference between 3 different pathfinding algorithms.

**Breadth First**: An algorithm that simply searches all path possibilities equally.

**Dijkstra**: An algorithm that eliminates searching paths that it deems unnecessary to search or less likely options to finding the shortest path to the objective.

**A\***: A modification of the Dijkstra algorithm, the A\* algorithm searches for the shortest path to the objective as possible but also accounts for calculating an estimation of how far the origin to the objective distance is and uses the distance from the origin to determine the best path to use.

* Describe how object-oriented programming and inheritance could be used to create AI.

Since OOP is a language where the programmatical objects simulate or relate to those in which the behaviours and states of real-world objects relate to each other, and inheritance is used to describe how children objects (or more specific objects) possess characteristics, data, behaviours, and state of their parent objects, then using these concepts with AI can prove very beneficial. Machine learning and AI is all about how a machine learns to imitate human behaviour and how humans learn that ability, and in a similar way, humans learn by way of relating real world and theoretical objects to each other and considering the differences, then storing that information for later. In this way, an AI can be taught to identify a parent object (a general concept or over-arching item such as ‘food’) and relate it to a child (or specific) object such as ‘chicken’. Using OOP and inheritance, it would be able to recognise that chicken is a child object of food, and would thus understand its base behaviours and state, but it could then learn and store chicken’s specific characteristics separate from its parent. I.E: what makes chicken different to food as a concept, or what makes chicken different to carrots.

* Outline the development process for creating AI strategies for NPCs in a game.
* Research existing AI in games
* Think up a concept for an AI behaviour
* Draw up a flowchart of behaviour for the AI
* Draw up a UML diagram for its structure
* Write pseudocode
* Implement code
* Test

# System Architecture

## Architecture Diagram

/Diagram of the planned system – This is like a mind map or flow chart of what you are going to make or what you need to do/

## Interface/API Definitions

**Components**:

* **Image** (used to place and manipulate image files within the Unity editor and extend functionality to other components that make use of the Image component)

<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIVisualComponents.html>

* **Text Element** (used to display, define, and manipulate text within the Unity engine/editor)

<https://docs.unity3d.com/ScriptReference/UIElements.TextElement.html>

* **Button** (used to run events on a click of the game object this component is attached to)

<https://docs.unity3d.com/2018.3/Documentation/ScriptReference/UI.Button.html>

* **Mesh Filter & Mesh Renderer** (used to render the GameObject the component is attached to and display it onscreen.)

<https://docs.unity3d.com/Manual/class-MeshRenderer.html>

* **Script**

<https://docs.unity3d.com/ScriptReference/ScriptableObject.html>

* **GameObject**: base class used to describe and define all entities in a Unity Scene)

<https://docs.unity3d.com/ScriptReference/GameObject.html>

* **NavMeshAgent** (used for mobile gameobjects to navigate pathfinding elements in the game)

<https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html>

**Libraries**: (need links and descriptions of what the library allows you to do)

* **UnityEngine**: connect to Unity, attach scripts to objects, reference Unity components, access unity’s classes and methods

<https://docs.unity3d.com/ScriptReference/index.html>

<https://docs.unity3d.com/Manual/ScriptingImportantClasses.html>

<https://answers.unity.com/questions/524862/why-does-one-need-to-use-unityengine.html>

* **UnityEngine**.**UI**: to access the UI Canvas Components and manipulate gameplay user interfaces in Unity

<https://docs.unity3d.com/Manual/com.unity.ugui.html>

* **UnityEngine.AI**: to access NavMesh components and properties and manipulate the AI agents, as well as utlising pathfinding.
* <https://docs.unity3d.com/ScriptReference/UnityEngine.AIModule.html>
* **System**: to access parsing and conversion of data types (in relation to this project), but according to Microsoft: *“Contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.”*

<https://docs.microsoft.com/en-us/dotnet/api/system?view=net-6.0>

* **System.Collections**: In relation to the project, allows us access to arrays, Microsoft: *“Contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hash tables and dictionaries.”*

<https://docs.microsoft.com/en-us/dotnet/api/system.collections?view=net-6.0>

* **System.Collections.Generic**: in relation to this project it allows access to stronger typed (more generic) collections and methods for those collections (ie: List<T>, which would be much more useful for Sort()-ing through multiple specific game object types, versus an ArrayList). Microsoft: *“…allow users to create strongly typed collections that provide better type safety and performance than non-generic strongly typed collections.”*

<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-6.0>

## Risks

* The resulting graphical integrity of the game.
* Properly incorporating the SOLID principles, especially DIP.
* NavMesh and pathfinding is new

Research to be done:

* NavMesh manipulation/accessing
* A\* search algorithm and implementation
* AI agent movement based on search algorithm results
* Spawning an AI agent in a random location on areas only with baked NavMesh
* Free roam camera controls
* Playing different animations based on state

## Alternatives

* The Dijkstra search algorithm: passed up due to the improvements that the A\* algorithm has over it.

# Sign Off

Name

[Name of appropriate person approving the software]

Role

[Click/tap to select role]

Signature

Date

[Click/tap to select date]

# Testing

## Errors and Buggs

Outline the test classes used. Add rows to table as required.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | **Description of Error** | **Screenshots of testing** | **Solution** |
| [Name of Class or Object that the error is connected to] | [Description of the error/error message] | [Add and resize relevant screen shots] | [Explain solution/fix to error] |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Evaluation

## Reflection

/Provide a self-reflection on your performance. /