

# **AI NPC - Directed project**

**By:** Joaquin De Losada

# Table of Content

## [Table of Content](#)

### [Introduction](#)

- [Purpose of Project](#)
- [External Important Links](#)
  - [GitHub Repo](#)
  - [Schedule](#)
  - [Resources for making AI](#)
- [Note from the author:](#)

### [Main Research on AI](#)

- [Navigation Mesh](#)
- [Finite State Machine \(FSM\)](#)
- [Behavior tree](#)
- [Barking](#)
- [Other general AI research](#)
- [Extra Links](#)

### [AI Design](#)

### [Game Objects and Scripts](#)

- [Player](#)
  - [Movement script](#)
  - [Basic inventory](#)
- [Enemy AI](#)
  - [Variables to transition State Machine](#)
  - [Methods](#)
  - [AI Tasks \(States\)](#)
- [Explosive Barrel](#)
- [Guns](#)
- [Vehicles \(If I have time\)](#)

### [Credit](#)

# Introduction

## Purpose of Project

The main purpose of this game is in the next 26 weeks, I will be implementing some of the more commonplace systems for creating AI in the gaming industry (State Machines and Behaviour trees). I plan to demonstrate this by creating a level in Unity where the player must infiltrate a base and kill enemies. While they are attempting to infiltrate the base, there will be various enemies that are both patrolling the area and working on some everyday tasks that can possibly spot the player and engage. There will be two different versions of the level, one that will have the AI run on state machines and the other one will have the behavior tree instead that decides the actions of the AI. With these two levels, I will demonstrate that I have experience building both types of AI while demonstrating how they can work differently.

## External Important Links


GitHub Repo

<https://github.com/Montainproductions/AI-NPC-Implementation>

Schedule

 Schedule for AI Game

Resources for making AI

 Resource Suggestions - From AI and Game

Voice acting lines

 AI Lines

## Note from the author:

When initially working on this project, I thought that I could make this project in Unreal so that I could also learn how to program in C++, but because of the way, I was learning Unreal and other challenges that were occurring throughout the first part of the project when starting I have decided to work in Unity since I already have more experience using Unity which will cut down on time to work on the project.

# Main Research on AI

The following part of the document contains a large chunk of the research I have done over the last while on how different systems are being used to create and maintain AI in games that allow for a challenging experience to the player while still allowing them to feel in control when necessary.

## Navigation Mesh

### ▶ How A Navigation Mesh Works in 3D Games | AI 101

An invisible surface on top of the game's map tells the AI what areas they can walk through and how they can move from one area to another. Usually, it is encoded in polygons. Usually, it is pre-baked to make it less task-intensive when running the game on pcs and is saved as data.

Must be able to:

What area is accessible,

How do different things connect,

Ability to tell what cost to move through certain areas.

Nav mesh was initially made for robotics in 1980 as a way to read map renders that are given to it. It breaks the map into areas to more easily determine what areas it can walk from and to using a search algorithm.

Quake 3 uses **Area Awareness System (AAS)** to load the map and determine what areas are accessible and what would need to be done to transition from one area to a different area.

Recast and Detour do similar things and are more modern

One problem of pre-baking is that if another item enters the navmesh and it doesn't update it might brake an AI that is walking nearby

Combat: In doom, the demons move around the map in a way that allows them to be in the line of sight of the main character enticing them to follow them and get closer.

Alien Isolation uses vents to "Enter the map" in the front stage view so it can walk around the map and make it seem like it's trying to find you. But in what's called backstage view, only a box collider and audio are playing and will walk in straight lines even through walls so that it seems like it's crawling through the vents. (Sus)

It uses Recast and Detour for the navigation but also adds the amount of noise the player makes so that it seems the alien can hear you. I will sample the nav mesh to see what types of attacks it can do.

In Tom Clancy's The Division, the AI might move depending on their current faction and class, affecting which AI to target. There is also a limit on the distance that AIs are allowed to walk away from their spawn points to have certain enemies only appear in certain areas and have no confusion.

When in open-world games that have companions and you're far away, then the companions might teleport close to you outside of your field of view, which means they will always be able to help you and reach you.

## Finite State Machine (FSM)

▶ **The AI of Half-Life: Finite State Machines | AI 101**

[Basic Explanation and implementation](#)

[More explanation of building FSM](#)

[The codebase for the Half-Life 1](#)

[Explains the AI section of the Half-Life code base](#)

Different predefined states have some activity or action, and some transition from one state to another. Usually, a state might be walking, picking something up, etc.

Only valid transitions can move from the current state to a new and different one.

An example would be an enemy patrolling an area (Current state) and then seeing the player so it transitions to the attacking state and will go toward the player and attack.

AI can react to events in the world and internal data stored before, making them relatively fast and easy ways of making basic AI that roams around the world and reacts to players.

More manageable to scale since there might be a lot more transitions and states, which creates a cluster of different states interacting with each other. Hierarchical Finite state machines (HFSM) are just FSM that also groups certain states into their states, allowing for a slightly easier time to organize the states meaningfully.

A more code-intensive system which, as stated, can get confusing to visualize

Half-Life State Machines

All AI is derived from a base class called a monster.

The base monster as a "State" is what the AI was doing then. (I.E Attacking, Idling, Dead, etc.)

Some Conditions contain all info the AI currently has on the world around it and a few ways of updating said info

The tasks in Half-Life would be the state in an FSM where the AI is completing at that moment. Some examples are Crouching, going to a specific position, and Reloading. These tasks in the base monster class are very common and generic states that every AI in the game can use, and any more specific things can be added or changed for each AI type.

In the game, specific ways allow the AI to have multiple tasks one after the other, allowing for a more realistic acting AI. This is usually done with schedules and goals.

With schedules, multiple tasks are put together, allowing them to do general tasks that might take multiple tasks. IE

## Behavior tree

▶ Behaviour Trees: The Cornerstone of Modern Game AI | AI 101

[Making Behavior Tree](#)

[More example Behavior trees](#)

[Visual Behavior Tree](#)

Based on the tree data structure where there is a root node with no parents, only children will be linked. The leaf node only has parents and will execute the code and tell what the object needs to do. The following nodes are composite nodes that might run some code to decide what actions need to occur and what to do.

Selector nodes will run the code that decides (And selects) what behavior to do. For example, if the AI is in range to attack the enemy or if it should get closer.

Sequence nodes might do multiple actions one after another when it's activated. For example, cooking where they will go to the kitchen, get the food, cut stuff, and cook it.

Decorative nodes might get already existing nodes and slightly change them in some way for a certain scene or action.

In Comp Sci, it is also called: Directed Acyclic Graph because it will go in one direction (Usually downwards) and doesn't have a way to go back in the graph.

Behavior Trees allow for having multiple and more complex behaviors, while other ways of making AI tend to be more confusing and harder to keep in one's head. Note: I was thinking through how to do some tasks for a game, and it does seem to be a bit easier if I wish to do multiple things one after the other but still have them separate (For example: Walking from points A - B, idling and being a guard are things that I might have bundled together but it is more convenient to have them modularly made, so I have other AI also walk and idle).

At the same time, it might be easier for designers to implement different behaviors if everything is packaged in a way that can be easily conveyed.


Initially in behavior trees, each frame would have the game run through the entire tree, checking if it has changed or not, which can become intensive on the computer resources.

But now there will be a reference node of what is currently being done and will check if it has already finished that action, and if it hasn't, then continue or else start again from the top of the tree.

In halo, they try to have the behavior tree also get affected by events that might happen throughout the game and will force the AI to reevaluate the behavior tree with the new information.

Use a blackboard that contains relevant information for different parts of the tree, for example, where areas it can go, where the player is, or other events that might have happened recently. It can be used for multiple behavior trees and AI.

## Barking

 How Barks Make Videogame NPCs Look Smarter | AI 101

## Other general AI research

Applying Behavior Characteristics to Decision-Making Process to Create Believable Game AI

[shorturl.at/IPX68](http://shorturl.at/IPX68)

AI in games can react more realistically as humans when they are allowed to take into account their surroundings and the different traits they might have (Whether they are aggressive or cautious, they have more experience using weapons or other characteristics). Games that are part of the open world genre are games that require a vast amount of AI and traits for said AI to allow them to make them as diverse as possible so that each area seems unique when compared to the other ones. But a constraint of more modern open-world titles is that the number of traits and uniqueness required to make as many of the AIs feel unique can be taxing to both create and process in the game.

The article talks about what makes "believable behavior" accurate in the context of AI in video games. The ability for the behavior of an AI to be believable depends significantly on the player's ability to observe the behavior of said AI and how it correlates to what the player would generally expect that another player would reach in the same situations. This ability for the

player to determine whether the AI system is acting similarly to a human is a similar test to that of the Turing test, which has been used over the years to test AI in general and their ability to appear human.

The paper cited another paper under the title of "Metrics for Character Believability in Interactive Narrative In proceedings," stating that some of the characteristics that an AI needs to have to make it "more realistic" needed the following:

1. The AI needs to follow the game rules.
2. Adjust the current behavior dynamically.
3. A way to perceive the environment similar to the player
4. Traits that affect its decision making
5. A way for AI to interact with each other
6. To a certain extent, the ability can predict what the AI might do next.
  - a. This means that the AI has a set of actions and changes between each in a realistic manner.
  - b. Because people normally will do an action based on a limited amount they can, but the one they choose depends on specific, relatively realistic reasons.

But this is still highly dependent on how the observer perceives said AI system.

Als require a decision-making system that allows them to take all the required inputs and then go through a method to determine which of the available actions they should take. Some examples of decision-making systems are below.

#### Planned Behavior:

The most rudimentary form of decision-making for AI, as they will only perform specific actions based on the current schedule. Depending on the in-game time, this is used if you need the AI to do certain things. Though this makes it highly undynamic behavior, they will tend not to be reacting to the surrounding environment as it changes by both player and other AI. For example, if an AI is on a planned behavior to always go from points A-B-C-D with some idle action occurring between the points, but then the player comes by and starts to attack the AI's allies, this doesn't affect the wandering AI. It makes for unrealistic behavior for an AI system and frustrates the player.

#### Finite State Machines (FSM)

This decision-making system is one of the most widespread forms of making AI. It is relatively easy to implement and valuable when only a few actions are implemented. However, as one begins implementing a more complex system, ensuring that all the conditions interact correctly becomes more burdensome. It might require some diagram to remember how everything interacts if it gets incredibly complicated or if new people start working on the AI. They need to learn how everything interacts with each other.

#### Behavior Trees

Unlike FSMs, Behavior trees split tasks into smaller nodes containing smaller code snippets that would run before going to the subsequently required node. This system allows for more



modularity as the nodes create even more complex behavior when compared to FSMs. At the same time, it enables the designers to have more control over how the AI reacts to different situations while also allowing the tree to be visualized similarly to a flow chart which provides an easier time following the decision-making of the AI systems (Unless the tree becomes enormous which can make it harder to debug as many options can occur). However, it also requires a Depth-first search which can be costly to run each game frame.

#### Machine Learning methods

Due to the increased ability of Artificial Neural Networks (ANN), more games have started to implement them to challenge the player even more than other decision systems allow. 2 good examples are the AI created for DOTA 2 using OpenAI and StarCraft 2 using DeepMind. However, these AI systems require significant computational resources to develop, requiring much time during production.

#### Utility AI

Utility AI tends to reach its decision based on estimating the profit values of different actions. So it works by going through the various choices/actions it can do and calculating the benefit it might incur when executed, and the AI will perform the action with the highest value. The calculations for each action are based on the AI's internal factors and surrounding environment, which means that the decisions can vary greatly depending on each AI circumstance and alleviate the need to design for many unique events that might not happen in the game often. The usefulness of Utility AI is the ability to affect its score based on the behavior rules using the superposition graph of multiple rules or tables that dictate how two or more rules interact. Allows for reusable and quickly expanding the current set of behavior rules by adding new options.

#### Method

The AI designed in the paper is based on the following concepts: Observe-Orient-Decide-Act (OODA) decision cycle, Utility AI, multi-agent modeling, and how the game industry typically attempts to create distinct behaviors.

#### Two Factor decision cycle

Orient makes a decision based on the AI's characteristics and "feelings" and might change depending on

Simonov, Andrey, et al. "Applying Behavior Characteristics to Decision-Making Process to Create Believable Game AI." *Procedia Computer Science*, vol. 156, 2019, pp. 404–413., <https://doi.org/10.1016/j.procs.2019.08.222>.

# AI Design

For the AI, I am aiming to have the enemy NPCs that, while not alert, will be patrolling around an area or idling with other enemy NPCs as if they were talking with each other. This will change when the player enters the line of sight with one of the enemies and/or starts combat, at which point the NPCs will start combat with the player and attempt to kill the player. Each enemy will then determine whether they should attack or go behind cover and hide (Or any other actions) and this will be sent to the AI Director, who will decide who will be allowed to attack or if any of the actions should be overridden. This will cause both aggressive enemies and enemies that will stay behind cover in better positions to attack the player.

## Finite State Machine (FSM)

### Enemy AI

#### Traits

The following are a set of traits I have started implementing to attempt to give even more variety to each AI so that they seem to have more personality and make them seem more unique.

#### Aggressive

- More likely to attack and rush the player.
- +3 points for aggression state decision value
- +2 distance to get closer to the player (Read attack state).

#### Bold

- Likely to attack but would rather stay in place or not get as close to the player in case it shoots back.
- +1.5 points for aggression state decision value
- +0.5 distance to get closer to player (Read attack state)

#### Cautious

- Will try to go to cover to be safe and is more likely to attack from cover.
- -1.5 points for aggression state decision value
- -0.5 distance to get closer to player (Read attack state)

#### Scared

- More likely to try and go to cover and try to not attack the player
- -3 points to go to cover
- -2 distance to get closer to the player (Read attack state)

#### Accuracy

#### Reload consistency

## AI Director

- Controls all the AI in the map as groups.
- Helps control the amount of enemies that can attack the player at once.
- Sets if more enemies need to be spawned if necessary.

## AI Manager

- Knows the information about the AI and world which are required no matter the state
- Knows current health, weapon, player position and other similar information.
- Knows what its current state is and what other states it can transition to.
- Can determine if it can see the player or not.

## State Machines

### Aggression

- When an enemy detects the player the enemy will enter this state where the enemy will calculate a value to determine whether they should go to cover or attack the player.
- This state actually operates as a utility AI so it can decide what to do next

### Attacking

- Is in charge of attacking the player with its current weapon.
- Will get closer to the enemy if it's too far to attack.
  - This works by getting the distance between the player and the farthest distance the AI can attack the player and still hit.
  - Then sending it to the common methods script (See description below) where it will choose a new random distance between the difference of attack + 1 and the difference of attack + 6 + trait bonus.
  - These additions on top of the minimum distance to attack the player are meant to represent the AI's ability to estimate how far they need to be from the player to attack and damage. These values can be changed depending on traits each AI has.
- Will reload if there is no more ammo in the gun.

### Cover

- Will choose and go to the closest free cover point.
- Determine which cover point is behind cover compared to the enemy
- Will randomly stand up to shoot the player.
- Will randomly decided it's better to choose a new position

### Idle

- Will randomly stop in place for some time
- Look around to seem like its just waiting and looking around

### Patrol

- Choose a set of points from array to patrol through
- Will go through the chosen points

## AI Tasks

### Attacking (Shooting) Enemy (Basic) Complete

- Will see enemy
- Set target
- Shoot
- If no ammo then reload

### Patrolling (Basic) Complete

- Will have an array of locations to choose from.
- Walk to that position.
- Once arrived go to the next location

### Idle (Basic) Complete

- Stand in a spot and play the idle animation

### PickUpItem (Medium) Complete

- Go to Location
- Play pick-up animation
- Add to simple inv

### ThrowItem (Medium)

### Find Cover (Medium) Complete

### Attack from cover (Advance [Basic if I can get find cover state working]) Complete

- If the time since the last player check is more than x amount then check
- Else stand up and shoot
- If no ammo then reload

### LostEnemy (Advance) In progress

- Will choose the last known position as where to walk
- Go to that location
- Stand around looking
- Choose new location

## Guns

- Firing rate
- Current ammo left
- Max ammo amount
- Ammo in clip left
- Max ammo in clip

## Credit

Professor Aaron

- Professor in charge of the directed project.

Tany Dourv

- Voice acting for aggressive enemies.