STANDBY

AI PLAYER DETECTION - Lucy Stent (1506919)

Introduction

Standby is a game in which the Robot butler has malfunctioned and now wants to kill its master

It was very important for the NPC AI to behave realistically by showing a searching and detection behaviour

Method

The Robot butler is controlled by AI which was given a PawnSensing component to detect the player through a field of vision and hearing.

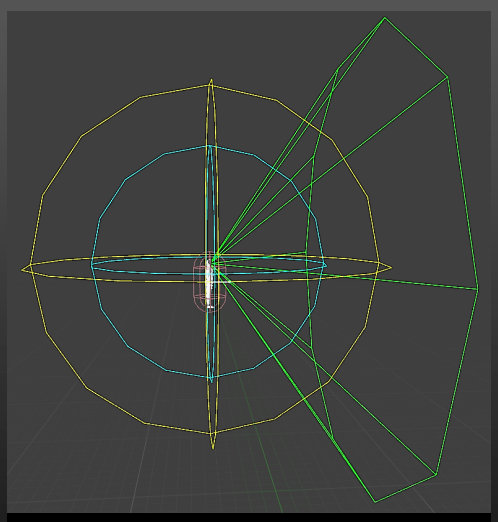

Fig 1: PawnSensing

Fig 1 is the PawnSensing component attached to the enemy NPC shows the vision field in green.

The yellow sphere shows the how far the NPC can hear unobstructed.

The blue sphere is how far the NPC can hear In an obstructed environment.

Blueprints were then constructed so the AI controller new what to do with the information detected by the component and also what to do when the component detects nothing.

Fig 2 is responsible for what happens when a character enters the vision field, Fig 3 is the reaction to hearing a noise the result of both was to chase the player.

When nothing was detected the NPC displayed a searching behaviour by moving to random locations within its environment.

Fig 4 is a simple state machine diagram of the behaviour displayed by the AI, of course if the player is heard or seen at anytime including when the AI is walking to a new location this would over ride that behaviour and lead to a chase.

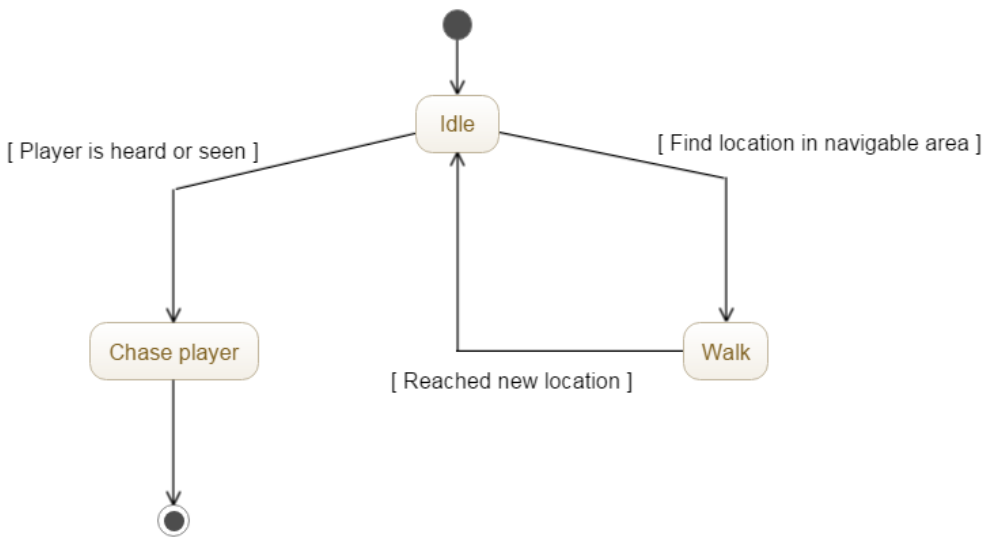

Fig 4: State machine diagram of AI behaviour

Chain of responsibility pattern

The hearing and vision work together to find the player, e.g. if the player is behind the AI the player will be heard which will turn the AI to face to player then the vision will take over the detection.

This is also true for the roaming behaviour, it first checks if the chase is being performed, if not it will then check if the AI controlled NPC is already moving, if that is also false the roaming behaviour can then start.

Command Pattern

Both the OnSeePawn (Fig 2) and OnHearNoise (Fig 3) are command patterns as they do not control when or how they all called but only what happens when they are called.

Observer Pattern

The Vision blueprint is an example of the observer pattern as it sends a message via a boolean that the vision is currently playing this is picked up by the Roaming blueprint so it doesn't play at the same time.

Memento Pattern

This pattern is used by the Roaming event which stores its own run state called moving, when the roaming begin the play it sets moving as true and when it has finished it sets moving back to false so it knows it can make another move.

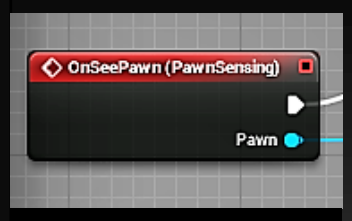

Fig 2: Vision

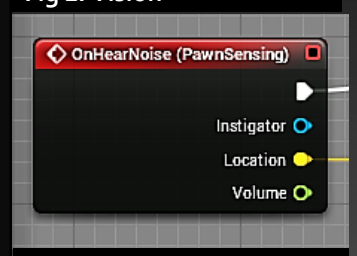

Fig 3: Hearing

Strategy Pattern

All three of the algorithms are a strategy pattern as they work together and depending on the in game situation depicts which algorithm should play.

Test Driven Development (TDD)

The NPC AI player detection was continually tested and changed during the development, the Random Roaming speed had to be changed to make it slower and the navigable area radius had to be tested to find if it was too small, the time in which the NPC waited before moving to a new location first had to be scaled up as it didn't wait at all before moving to a new location and then had to be scaled down a little as the delay was too long.

The vision field was much to large in the beginning to the point where the NPC could see the player over the other side of a large area, so it was scaled down. The chase speed was also changed twice, once up and then down to get the perfect speed.