Patrolling Guard AI

# Finite State Machines:

* Patrolling
  + When the AI is simply patrolling the compound, it will loop between different points laid out as a pathway. This pathway will be given the option of either be looping or ping-ponging.
* Standing Guard
  + An AI can also stand guard. This means the AI will not move, but rather rotate in place, investigating one specific room or corridor.
* Investigating
  + If the AI notices the player briefly or hear them nearby, it will ignore its patrolling pathway and start searching for the player in the nearby corridors. It will do this until an area (determined on how sure it was it heard/saw something without directly noticing, ranging from *x* to *y*) has been searched. Once this is completed, it will return to its original state.
* Alerted
  + If the player is fully revealed to the AI, the AI will give chase. The AI will then follow the player by raycasting ahead of itself and seeing if it can see the player. If it can see the player, it will follow them. If it can’t, it will make assumptions of where the player ran for the next *z* seconds. When the player has been fully lost, the AI will be in an alternative state to “Investigating” called “Searching”.
* Call for Help
  + If the AI is in the “Alerted” state, but comes across any additional AI agents, the AI will stop and let those AI agents know of the situation. This puts the other two agents in the “Alerted” state with their target being the player. This state stops the AI agent for *å* seconds, and then returns to the “Alerted” state.
* Searching
  + Similar to “Investigating”, the AI will deviate from its patrolling pathway and search the player in the nearby corridors. It will do this in an area of *y* size. If it encounters any other AI agents, it will alert them of the intruder as well, putting them in the “Searching” state in an area of *x* size.

Any State, Generic Instructions - Any AI agent seeing another agent “Alerted” will immediately go into an investigating state. If an AI agent sees two agents in the “Alerted” state, it will enter an “Alerted” state as well, however with their target being the other agents until the player has been seen. An AI will only return to its original state – an AI standing guard will not suddenly start patrolling, or vice versa.

# Utility Theory:

The strategies will be scored between 0.0f and 1.0f

Information needed for “Patrolling” – Default State, no information needed

Information needed for “Standing Guard” – Default State, no information needed

Information needed for “Investigating”

* Nearby Audio Level (Player, Distraction, etc.)
* Line of Sight to an Unidentified Object (Player, Distraction, etc.)

Information needed for “Alerted”

* Nearby Audio Level (Player, Distraction, etc.)
* Line of Sight to the Player

Information needed for “Call for Help”

* Distance to Player
* Line of Sight to the Player
* Line of Sight to AI Agent

Information needed for “Searching”

* Nearby Audio Level (Player, Distraction, etc.)
* Line of Sight to an Unidentified Object (Player, Distraction, etc.)

Full list of information needed:

* Nearby Audio Level (Generic)
* Line of Sight to Different Objects (Generic, Player, AI Agent)
* Distance to the Player

Pseudocode - Getters:

public float GetAudioLevel(AIAgent anAgent)

{

return 1.0f – (Mathf.Clamp01(FindNearest<IAudibleObject>().transform.position - anAgent.transform.position) / agent.hearingRange);

}

public bool GetLOS(AIAgent anAgent, IObjectOfInterest anOOI)  
{  
 return Physics.Raycast(anAgent.transform.position, GetDistance(anAgent, anOOI, 0));  
}

public float GetDistance(AIAgent anAgent, IObjectOfInterest anOOI, float aRange = 10f)  
{  
 return aRange > 0 ? Mathf.Clamp01((anOOI.transform.position - anAgent.transform.position) / aRange) : anOOI.transform.position - anAgent.transform.position;  
}

Pseudocode - Desirability:

public float GetInvestigationDesirability(AIAgent anAgent) // Used for both Investigating and Searching  
{  
 return k \* ((GetLOS(anAgent, FindNearestInFront<IObjectOfInterest>()) ? 1f : 0f) + GetAudioLevel(anAgent));  
}

public float GetAlertedDesirability(AIAgent anAgent)  
{  
 return k \* ((GetLOS(anAgent, FindNearestInFront<Player>()) ? 1f : 0f) + GetAudioLevel(anAgent));  
}

public float GetCallForHelpDesirability(AIAgent anAgent)  
{  
 return k \* ((GetLOS(anAgent, FindNearestInFront<Player>()) ? -GetDistance(anAgent, FindNearest<Player>()) : 0) + (GetLOS(anAgent, FindNearestInFront<AIAgent>()) ? 3f : 0f)  
}

Level Generation

# Procedural Content Generation