LightSpot A.I. Analysis

LIGHTSPOT

High Concept

I want to create a game that fits in a common modern genre, and also requires the utilization of AI, so I decided to make a top-down stealth game.

The AIs are the guards patroling the level and trying to stop the player from collecting all the "Intels". So it is obvious that they need proper path following. I also used state machines for the AI to start chasing the player once they see him. Also all the AI has shared information, and they are all basing their actions on the shared information.

Instructions

- Avoid the guards and the cameras.
- Go to the desks and press 'E' to switch off a camera
- Go to the laptops and press 'E' to collect the intel
- Go to the stair with all the intel and pree 'E' to escape the level

Features

- **Path Finding / Following**

  The AIs are using standard A* algorithm to navigate through the level.

  The level has invisible grids on the floor. Each grid has two states: **Normal**, **Blocked**. Only grids on normal state are used in the algorithm. In the game, walls and some objects is making some of the grids blocked, so the AI cannot move on them.

  Even though it is not used in the current level, each grid has a cost value that could be used in A* algorithm. Right now every grid node has the same cost value of 1. I saved this feature in case it is needed for future design.

- **State Machine**

  Each guard AI has a state machine. The state machine right now is pretty simple, it cosists of two state: **Patrol State** and **Pursue State**.

  - **Patrol State**: Each guard has a list of grid nodes that he is supposed to patrol in between. Upon reaching one grid node, he wait for a small while, switch his target position to the next grid node, find the path and following that path. Once the guard sees the player, he switch to **Pursue State**.
  - **Pursue State**: In this state the guard just goes after the player. He updates his path to the player's current location every time he reaches to a new grid node. If the guard is close enough to the player, the player loses.

- **Group Logic**

  - Each AI shares a set of information. There is a class called '**Situation**' that every AI has reference to. The guards base their actions on the information within. The cameras can spot the player and notify all the guards through the 'Situation'. For now the available information in 'Situation' are '**Is the player spotted?**' and '**If so, where is the player?**'. It could be easily expanded to adapt to different complicated AI behavior designs.

    - There is also a higher-level AI called '**Security Level**'. Base on how high the security level is, the action of the AI actors could change. For now the security level increases with time, also when the player is spotted, it jumps to the maximum. For now it only does one thing: spawn a new guard when it reaches to maximum, but this is a feature that could be easily expanded base on various game design.

- **Field of View**

  This is probably the biggest problem I solved. It is absolutely crucial in a stealth game. At the beginning I used a grid based field of view system. It works in a way, but it has the worst visual presentation and just doesn't feel right. After some research I decided to use Raycast to detect objects, and a third-party asset to create the dynamic light.

Extra

I know this is totally unrelated, but I also took this chance to learn how to write shaders, since I happened to want to create a certain visual style for this project. So the simple camera shader I wrote does two things: **Color Aberration** and **Light Dimming** for the entire screen. The further it is to the player, the stronger the effect.