City, University of London

BSc Computer Science with Games Technology

Final Year Project Report

First-Person Stealth Game Prototype

By

Tayyab Hussain

# Table of Contents

[1. Abstract 3](#_Toc137185745)

[2. Introduction 4](#_Toc137185746)

[2.1. Description of the Problem 4](#_Toc137185747)

[2.2. Project Objectives 4](#_Toc137185748)

[2.2.1. Stealth AI 4](#_Toc137185749)

[2.2.2. Stealth UI 4](#_Toc137185750)

[2.2.3. Stealth Mechanics 5](#_Toc137185751)

[2.2.4. Movement/Traversal 5](#_Toc137185752)

[2.2.5. Balance 5](#_Toc137185753)

[2.2.6. Other Functionality 5](#_Toc137185754)

[2.3. Project Beneficiaries 5](#_Toc137185755)

[2.4. Work Performed 5](#_Toc137185756)

[2.5. Limited Scope 6](#_Toc137185757)

[3. Output Summary 7](#_Toc137185758)

[3.1. Guard Behaviour Tree 7](#_Toc137185759)

[3.2. Group Searching and Attacking Algorithm 7](#_Toc137185760)

[3.3. Detection System 7](#_Toc137185761)

[3.4. State Depiction System 8](#_Toc137185762)

[3.5. Hiding Mechanics 8](#_Toc137185763)

[3.6. Traversal 8](#_Toc137185764)

[4. Literature Review 9](#_Toc137185765)

[4.1. Introduction 9](#_Toc137185766)

[4.2. Artificial Intelligence 9](#_Toc137185767)

[4.3. User Interface 9](#_Toc137185768)

[4.4. Mechanics 10](#_Toc137185769)

[4.5. Traversal 10](#_Toc137185770)

[4.6. Balance 10](#_Toc137185771)

[4.7. Conclusion 10](#_Toc137185772)

# Abstract

Third-Person Stealth Games have been a staple in the Games industry for over a decade. Games such as Hitman 3(IO Interactive, 2021) and Splinter Cell: Blacklist(Ubisoft Toronto, 2013) are regarded by many as some of the best games ever made, not just in terms of stealth. Naturally, game studios attempted to replicate this success in the form of First-Person Stealth Games/Levels. This was largely unsuccessful. A market gap exists for an intriguing and tense First-Person Stealth Game/Level.

This project aimed to develop a First-Person Stealth Game Prototype that would implement some of the features which make blockbuster Third-Person Stealth Games great. To do this, the project was divided into five key areas to explore, Artificial Intelligence, User Interface, Game Mechanics, Movement and Traversal and Game Balance.

# Introduction

## Description of the Problem

Stealth games have been highly prevalent in the last decade. Games like Hitman 3(IO Interactive, 2021), Batman Arkham Knight(Rocksteady Studios, 2015), Metal Gear Solid V: The Phantom Pain(Kojima Productions, 2015) and Splinter Cell: Blacklist(Ubisoft Toronto, 2013) are all exceptional examples of good stealth games. However, these are all Third-Person games. There is a severe lack of First-Person Stealth games. Dishonoured 2(Arkane Studios, 2016) is the only relatively recent game that meets the First-Person Stealth criteria.

There have been many attempts by games such as Call of Duty Modern Warfare(Infinity Ward, 2019) and Battlefield 1(DICE, 2016) to include stealth levels within their FPS games. However, these often feel shoehorned in. A few areas in which these games need to improve compared to their Third-Person counterparts include a poor stealth AI, an unintuitive UI design for stealth, a lack of stealth-specific mechanics, limited movement/traversal options and a lack of balance between the player and the enemies.

This project aimed to create a prototype First-Person Stealth game that would incorporate a few of the features that make Third-Person Stealth games great. The specific areas the project looked at were the stealth AI, stealth UI, stealth mechanics, player movement/traversal and balancing the player and enemies.

## Project Objectives

The project’s main objective was to create a First-Person Stealth level using Unity. This included five main aspects, AI, UI, Mechanics, Movement and Balance. The exact objectives were defined as follows:

### Stealth AI

1. The stealth AI shall be designed with a behaviour tree with at least five states for the enemy. E.g., Cautious, Search, Attack, etc.
2. The AI shall use a pathfinding algorithm to find the player when in the searching state.
3. The AI shall not have a binary detection of the player. The AI should slowly detect the player over time and not instantly go into a state of alert when they spot the player.

### Stealth UI

1. There shall be a small icon to allow the player to determine the current state of an enemy. E.g., Red for an attack state and amber for a search state.
2. There shall be a UI element allowing the player to determine whether they are about to be spotted.

### Stealth Mechanics

1. There shall be two different stealth mechanics/gadgets that the player can use in the level. E.g., Binoculars to mark targets, Agent 47`s piano wire(IO Interactive, 2021), and Sam Fisher`s fibre optic cable(Ubisoft Toronto, 2013).
2. The player shall only be able to use the mechanic a finite number of times within the level.

### Movement/Traversal

1. The player shall have a unique and original way to traverse the level that differs from walking, running, crouching, and crawling prone. E.g., Batman`s Grapnel (Rocksteady, 2015) and Dishonoured`s Blink ability (Arkane Studios, 2016).

### Balance

1. Enemies shall be much stronger than the player in terms of health and damage they can do.
2. The weapons the player can use shall be very weak when taking on multiple enemies.
3. The player shall not regen health or have any way to replenish health in the level.

### Other Functionality

1. Main menu and pause menu.
2. Audio and Visual FX.
3. Saving and Loading.

## Project Beneficiaries

The primary beneficiary of this project was other developers specifically working on FPS games who wanted to implement a stealth level into their game. The project should give them a working prototype of a stealth level better implemented using some prominent features used successfully in Third-Person Stealth games.

Another beneficiary of the project was developers who may have wanted to build on top of the prototype to produce a full First-Person Stealth game.

## Work Performed

The project was developed using an Agile Development methodology. The development was split into stages based on the primary objectives, AI, UI, Mechanics, Traversal, Balance and Other Functionality. The stages were then sorted by their importance to the project’s success. Therefore, AI and UI were developed first, and Other Functionality was left until the end of development. This was done to ensure that if there were time overruns, the essential features of the prototype had been completed.

The features were designed, implemented and tested for each stage of the project. If the feature did not meet the objective, it was redesigned, implemented and tested again. Use case Requirements and Use Case testing were utilised to design and test the features. This worked effectively when paired with the Agile methodology as it allowed for a feature to be easily redesigned and retested if it did not meet the objectives or if the design was not up to scratch and the feature had to be improved.

## Limited Scope

The scope had to be limited in development due to time constraints. It was underestimated how much time it would take to implement parts of the prototype that were not vital to the project. Things such as a moderately comprehensive player controller and a much larger level design to incorporate some ziplines for the Traversal objective took time. Before starting the project, this time was not initially accounted for in the work plan. However, thanks to how the work plan was assembled, with the most vital stages of the prototype being completed at the beginning, the only stages removed from the development were the Balance and Other Functionality.

The scope of objective 1a regarding the Enemy AI`s Behavior Tree was also limited. The objective entailed having a minimum of five key states for the Enemy to be in. However, it soon became apparent that having four key states was comprehensive enough and having another state for the Enemy to be in would be unnecessary and likely be of a lesser quality than the previous four.

# Output Summary

## 3.1. Guard Behaviour Tree

The first output is the Behaviour Tree which controls the actions and states of each Guard in the prototype. It is software code coded in C# consisting of 15 classes and 904 lines of code(not including comments), of which I wrote 698 and 206 were re-used as detailed in the Results section. The intended recipients of the output are future developers. They can adapt the basic framework of the current Behaviour Tree for use in their games, or they can use this Guard Behaviour Tree as a base and build upon it.

## 3.2. Group Searching and Attacking Algorithm

Objective 1b entailed developing a system for the AI to use pathfinding to search for the player. For this, a searching algorithm was required to tell the AI guards where to pathfind to, to make it look like the Guards were actively searching for the player as a collective. Later this developed to also house functionality that would allow the Guards to all organise an attack together if one of the Guards spotted the player. This output was a single class I wrote in C#, consisting of 184 lines. The intended recipients of this output include future developers who can take this simple yet effective searching algorithm and make it more complex and more efficient for use in their games.

## 3.3. Detection System

The AI required functionality to spot the player slowly over time. To do this, a system was created to implement a vision cone for a Guard and then split that vision cone into five different vision zones. These vision zones would detect the player at varying speeds based on how close the player was to the Guard. The output was two classes I wrote in C# consisting of 280 lines of code.

Another aspect of the detection system was implemented in the UI stage of development. There was a UI element that would depict how close the player was to being detected based on the previously mentioned detection speed. The output consisted of two main parts. The first part was C# code consisting of 2 classes and 60 lines of code, of which I wrote 41 and 19 were reused as detailed in the Results section. The second part of the output was the sprites that would be used to display the detection amount. Two sprites were initially created as .png and then converted in Unity to sprites. I made one sprite using Piskel, and another was found online, and both were a combined size of 177KB. The sprites were detailed further in Appendix B.

The intended recipients of this output are the same as the previously mentioned future developers. With this output, they can easily modify the detection area and how quickly the player can be detected to suit their need for their specific games. The sprites can also be treated as placeholders and replaced by a more visually appealing design. The complete output is detailed in Appendix B.

## 3.4. State Depiction System

To inform the player of what state a Guard is in, a UI element was designed to depict the current state of a Guard. This output was simple and only depicted three of the four AI states. These included a UI element for the Searching state and a UI element for both the Chasing and Attacking states, called the Alerted state. The output was split into two parts. The first was C# code to enable and disable the UI elements. These were divided into two separate classes, along with a third class which handled turning the UI elements into a billboard. This last class was re-used, as mentioned in the Results section. From the three classes, 60 lines of code were written, 41 by me and 19 were re-used.

The second part of this output came in the form of sprites. These were used as the actual visual representation of a Guard’s state. As detailed in Appendix B, I designed the Searching and Alerted sprites using Piskel, and both sprites came to a total size of 201KB.

Future developers are the intended recipients of this output, and they can look to build upon this output by implementing more AI states and then complementing those states with more UI elements to depict those states. Future developers may also want to keep the current implementation of the Ui but alter the graphic design of the sprites to make them more visually appealing. The complete output is detailed in Appendix B.

## 3.5. Hiding Mechanics

A simple yet efficient and easy method of implementing hiding mechanics was implemented into the prototype. The output consists of two simple parts. The first is a layer mask called Obstacle. I added this to the layers section in the Unity Inspector Window. This allows future developers to choose an element they want the Guard not to see through, and it will enable the developers to quickly create a vast array of places the player can use to hide. The second component of this output consists of a single line of code embedded into a much larger class, coded in C#. The total amount of code in the class comprises 127 lines, and only one line is required for this output. I wrote the entire class. Future developers can also make changes here, they only have to change one line, and their AI will not be able to see through any other layer mask they choose. The complete output is detailed in the Results chapter.

## 3.6. Traversal

For objective 4a, a zipline was implemented into the prototype to allow the player to traverse the level more dynamically. This implementation consisted of 2 C# classes, both of which I wrote. The code amounted to a total of 98 lines. The intended recipients of this output are future developers who can build upon this feature and implement more complex zipline physics, animations, and a more polished output than the one in this prototype. The complete output is detailed in the Results chapter.

# 4. Literature Review

## 4.1. Introduction

Stealth games have been one of the most popular genres of video games since the turn of the millennium. Games such as Metal Gear Solid(Hideo Kojima, 1998), Tom Clancy`s Splinter Cell: Blacklist(Ubisoft Toronto, 2013) and Hitman(IO Interactive, 2016) defined the genre. An overwhelming majority of these games are in the Third-Person. The only notable recent exception is Arkane Studios` Dishonoured 2(Arcane Studios, 2016). There have been many attempts by FPS studios to include a stealth level within their action-packed games, but these often feel shoehorned in. For my project, I amde a prototype First-Person Stealth game that sets out to achieve what most FPS Stealth games/levels are missing. In this literature review, I will compare and contrast five themes: AI, UI, Mechanics, Traversal and Balance in Third-Person Stealth and First-Person Stealth games/levels. This allowed me to find gaps in current FPS stealth levels that my prototype aimed to fill.

## 4.2. Artificial Intelligence

A stealth-based AI is a critical component of a stealth game. An AI that the user perceives to be unintelligent will make the game less intense and provide less of a challenge for the player. One essential part of a comprehensive AI is how the AI detects the player. In Splinter Cell: Blacklist(Ubisoft Toronto, 2013), the team used vision cones and vision zones(Walsh, M. 2014) to allow the player to be slowly detected over time based on the player`s location within the enemy’s vision cone. This is an example of great architecture for stealth AI and provides a realistic interpretation of how someone would spot a foreign entity in real life. In contrast, Metal Gear Solid(Hideo Kojima, 1998) has a binary detection system. This means that whenever the player is within the enemy’s vision cone, they are instantly detected. This could be very frustrating to the player as even if the player`s arm was barely visible for a split second, the enemy would go into an attack state. This makes AI seem unrealistic and unfair. However, unlike some games, Metal gear solid(Hideo Kojima, 1998) does allow the player to hide after being spotted and return to the stealth aspect of the game. This is evidence of an in-depth implementation of a behaviour tree or finite-state machine(FSM) for AI(Millington, I. 2019). To sum up, the AI needs a non-binary detection system and a behaviour tree or FSM that allows the player to escape after being spotted.

## 4.3. User Interface

Stealth games must have an intuitive UI design. A good UI can help the player understand what state the enemies are in, if the player is hidden and how close the enemies are to spot the player. Dishonoured 2(Arcane Studios, 2016) has an excellent UI that uses markers above the enemy’s head to depict the enemy`s current state. If we compare this to Call of Duty: Modern Warfare(Infinity Ward, 2019), we find that during the mission ‘Going Dark’, the UI gives the player no indication of what state the enemy is currently in. This doesn’t give the player confidence and can make the player play safer and not explore the level to its fullest. Overall, it is crucial to have an intuitive UI that helps the player understand what state the enemy is in.

## 4.4. Mechanics

A game won’t be fun if its mechanics are boring and unbalanced(Adams, E. and Joris Dormans, 2012). Games are made by how good their mechanics are, and stealth games are no different. This is most clearly seen during the Battlefield 1 mission ‘Fall from Grace’(DICE, 2016). During this mission, there are only two stealth mechanics; both are overused and unbalanced. The first is throwing an item to take the guard’s attention, and the second is destroying a communication box to stop reinforcements. Not only are they overused, but they are also unbalanced. The player can find things to throw to distract the guards all over the level, and disabling the communications boxes provides no real challenge. Therefore, mechanics in stealth games must be original and balanced to prevent the player from abusing the mechanic.

## 4.5. Traversal

Traversal is a vital component of any good game. The ability to traverse the level uniquely provides more replayability for a level. Stealth games have been using unique forms of traversal for a long time. Examples include Sam Fisher`s split jump in Splinter Cell: Chaos Theory(Ubisoft, 2005) and Batman`s grapnel in Batman: Arkham Asylum(Rocksteady Studios, 2009), which the player can use to climb above the guard’s line of sight or use ventilation shafts/grates to traverse the level below the guards’ feet. This kind of traversal is largely missing in First-Person Stealth games/levels. An example is Battlefield 1`s level ‘Fog of War’(DICE, 2016). In this level, the player has no unique ways to traverse the map. There is no way to change the verticality or manoeuvre around enemies. This leaves this level feeling a little flat. Ultimately, traversal is necessary for all games, but the lack of unique traversal in stealth games can leave the player with no opportunities to tackle a level in an original way.

## 4.6. Balance

A balance between the player and enemies is even more vital in stealth games than in regular games. In stealth games, if the player feels like they can win a shootout against a large number of enemies, it defeats the point of the player trying to be stealthy. The designers made a single enemy lethal in The Last of Us(Naughty Dog, 2013). This forced the player to play stealthily. However, this alone would not be balanced when the player was fighting a large group of enemies in a non-stealthy environment. To counterbalance this, when the player was fighting a large group, only one or maybe two enemies would shoot at the player simultaneously (McIntosh, T. 2014). This meant the player was still cautious of the enemy, but at least it gave them a fighting chance. Getting the balance right in a game is a tedious and lengthy process. Getting it right in a stealth game is just as hard. But doing it correctly means the player plays the game in the way intended for them by the developer, thus resulting in a much more fun experience.

## 4.7. Conclusion

To conclude, there is a multitude of things that FPS stealth games/levels can learn from their Third-Person counterparts. Many FPS stealth levels incorporate one or maybe two good practices seen in Third-Person Stealth games, but the only First-Person game that encapsulates all these practices is Dishonoured 2(Arcane Studios, 2016). With the knowledge from this literature review, I focused my prototype on the specific areas that FPS stealth games/levels are in dire need of.