START LOOP

Requirements

Design

Implementation

1. What algorithms/approach did I use
2. Programming lang/platform, why? And what effect did it have?
3. How did I plan and design
4. Stages of work?
5. Version control?
6. Code optimisations? Why or why not?
7. How did I test
8. How did I select inputs to test

Evaluation

Testing

END LOOP

NO RESULTS

State the tools used for each step (inc tutorials)

2500 words

Intro to chapter. Methodology used 1. Tools used 2. Plan and design 3. Build plan 4. Work plan 4. Version control 5. Code optimisations? Test and testing inputs? 500 words ish

Structure sub-chapters based on objective. Requirements. Design. Implementation. Evaluation. Testing. 200 WORDS MAX PER OBJECTIVE

Diagrams drawn graphically preferably

This project was developed using an agile methodology. Specifically, each major objective was run in an iterative loop of design, implementation and testing. This ensured each objective was completed to a good quality before moving on. This lines up with the build plan.

The build plan was split into 6 major sections and was structured so that the most vital parts of the project were completed first. This ensured excellent quality of the most important aspects of the projects, which was crucial so that the prototype at the end of development highlighted the best bits to potential beneficiaries. Due to the strict timeframe for this project, there was also a risk of not meeting all the objectives initially set out by the proposal. The build plan also works to help nullify this risk by placing the least important aspects of the prototype towards the end of the build.

To track the progress of the project I designed a work plan in the form of a Gantt chart and this was updated regularly to reflect the progress made on the project. Coinciding with the agile methodology, there was a reflection every weekend to ensure I was on target with the project and if I wasn`t, I could move some of the objectives around and rethink my work plan. This allowed me to be flexible and still complete the project to a high degree.

The methodology chapter is broken up into 10 different parts, one for each objective. These sub-chapters include at least the design and implementation required to complete that objective. They may also include the requirements, evaluation and testing of the objective/implementation.

Note, this project`s objectives are to address areas in which first-person stealth games are lacking. Graphics and animations are not one of these areas. For my project, I will not use any fancy models for characters, etc, as it would take attention and time away from me implementing key features that are lacking in first-person stealth games.

The version control used for this project will be GitHub. The project was backed up to GitHub at the end of every day in which any sort of work took place on the project. The reason GitHub was chosen was that I have used it a lot in the past to back up my other university work and found it to be quick and easy to use.

Enemy Detection (Objective 1C)

The first objective was to implement the enemy`s ability to slowly detect the player over time. This was done by first implementing some basic detection. This was achieved by performing a raycast from the enemy to the player and checking that the player was within a certain view angle relative to the enemy`s forward vector(ensuring the player was in front of the enemy). The next step was to check that the player was within a certain range of the enemy and that there were no obstacles in the way. To implement a more in-depth system, the enemy would have to slowly detect the player over time. To implement this, split the enemy`s view cone into different zones. This was done by first splitting the view cone into 3 different angles, wide, medium and close. The smaller the angle, the faster the player should be detected. The next step is to add 3 distinct ranges within the view cone. This allows for the player to be detected slower when they are further out from the enemy. with 15 distinct zones categorised into 5 zone types within the enemy`s view cone, all that is left to do is set a timer for each zone type to detect the player at a different speed depending on what zone the player occupies in the cone.

An example of an Enemy Vision Cone. 15 vision zones, split into 5 zone types. The higher the zone type, the slower the detection. 'a', 'b' and 'c' are the close, medium and wide angles respectively, displayed with the blue lines.

