Methodology

**What did I do…**

I made an application

An application was produced to test the hypothesis. This application used the Games Education Framework [reference], a light framework with little overhead. This was very early on chosen over the framework chosen in the proposal as it had a much larger overhead when producing the environment space in DirectX11. This switch was confirmed as best by two lecturers.

What did the application do?

The application simulated an environment, in which two flocks interact (one of which is improved via a genetic algorithm) within a limited space of scarce resources, with the aim of finding out if a flocking algorithm improved by a genetic algorithm can outcompete one without.

How was the flocking algorithm designed? (what influenced your choices?)

The design of the flocking algorithm was produced

The boids forces were modelled primarily off of the paper [reference here]. This is a modified version of the expanded boids algorithm, optimised for the environment in which it is placed in. The forces each boid experiences are: Cohesion, Alignment, Separation, Food Attraction and Flock Avoidance.

**Cohesion**

Cohesion my bad boiii what the fuck should you even be anyway like jesus.#

Basically cohesion is used to – you guessed it- coheseeeeee. Basically its asking the question, what is the best way to represent cohesiveness in this system

Maths Equation Here…

What design process/decisions did I take to create the application/genetic algorithm? [could ref. any relevant theory here]

* No mention of results (i.e. how well the bots worked)
* But method of **how** u tested to see how well they worked
* And any **alterations** made further based on tests, i.e. X method was used; if outcome A observed then technique 3 used, but if outcome B observed then technique discontinued/technique 4 used

Sections:

1. Design of software application (maths behind the code)
   1. How I went from bad to good flocking algorithm design (This may be a result, but write it just in case anyway and then copy to results if it’s not in the right section)
   2. Iterative Design process
2. Outcome (the code/application)
3. Testing

\section{Methodology}

An application was produced to test the hypothesis. This application used the Games Education Framework (GEF) [reference] as the framework, which was chosen due to its lightweight overhead and recommendations from a couple of university lecturers.

\subsection{Application Design}

The application was designed through an iterative process.

\subsection{Test Environment Design}

The application simulated an environment, in which two flocks interact (one of which is improved via a genetic algorithm) within a limited space of scarce resources, with the aim of finding out if a flocking algorithm improved by a genetic algorithm can out-compete one without.

To answer the research question required removing as many base assumptions as possible - reducing the problem to its core necessities. To do that, three core decisions were made: The environment should be limited so as to keep the flocks in competition with each other, this means the flocks have to interact; there should be a limited amount of resources in the environment, less than that required to support two full populations of flocks, this should drive the flocks to compete over resources spawned in the environment; finally there should only be two flocks, one improved by a genetic algorithm over simulated generations and one produced through a tested expanded flocking algorithm design. This ensures a competitive environment in which the effects of decisions the genetic algorithm makes can be singled out, and the data produced analysed. Fig.\ref{fig:EnvScrnshot} displays a screenshot of this environment in action.

\subsection{Flocking Algorithm Design}

The flocking algorithm was originally designed primarily by adapting the example from \citet{flockingprocessingorg}. As this was a very clear representation of how to design a basic flocking algorithm from a code perspective. There were also clear areas for improving the efficiency of the algorithm presented too. The most major way was combining the calculations for Alignment, Separation and Cohesion. This was so each boid was not checking against all others for each force calculation, and only needed to check against all other boids once instead of three times; doing that was simple and reduced the requirement for N-Body problem solutions like the Barnes-Hut algorithm later on. The other main way was combining the calculations required for each force as they use shared variables, so calculating them once instead of multiple times for each force also increased efficiency.

This worked ok for the original three forces based on \citet{Reynolds:1987:FHS:37402.37406}. However, issues came up when expanding the application to include extra forces. Especially since in the aforementioned implementation it wasn't strictly clear how the forces would graph out due to the lack of separation between direction and weighting. This caused issues to pile up and it was clear a change in the flocking algorithm design was necessary to expand and edit the program appropriately.

The boids' forces as they are in the final program were modelled primarily off \citet{4604156}, which has a very clear mathematical representation of each force. The version of the expanded boids algorithm found in this paper was modified by optimising the algorithm for the environment in which it was placed. The forces each boid experienced were: Cohesion, Alignment, Separation, Food Attraction and Flock Avoidance. Each of these forces were the result of multiplication of a unit vector and their respective weight, mathematically represented in Eq.\ref{forcevector\_equation}

The relationship of the weight to the unit vector will become clear as the boid forces are described further on. This separation of direction vector and weight, meant it was far easier to mathematically model and graph out each force to predict their behaviour; this caused a reduction to the amount of work necessary to produce a functioning flocking algorithm. The resulting accelerative force then is the sum of these forces. That sum is then used in the Semi-Implicit Euler method to produce the updated position of each boid every frame.

This is the first of the boid forces, cohesion pulls the boid in the direction of flock members, in this case towards its local flock centre. The boids local flock centre is determined by its neighbours in communicable range and is the average position of those flock members. The further the distance away from its neighbours, the stronger the force. Below is Eq.\ref{cohesion\_equation} describing the direction vector and weighting:

Where $LFCVector$ represents the vector from the boid to the local flock centre, $LFCPos$ is the position of the local flock centre, $BoidPos$ is the position of the boid and $FlockSize$ is the number of flock members as a whole.

\subsubsection{Alignment}

Alignment is a vector that accelerates the boid in the average direction of its neighbouring boids. This produces the common direction the flock as the boids individually move about. By taking the average velocity of neighbouring boids as a direction vector and multiplying it one over the distance between the local flock centre and itself a suitable alignment vector is produced. Below is Eq.\ref{alignment\_equation} describing the two components of the alignment vector:

Where $LFVelVector$ is the average velocity of the local flock members, and $LFCPos$ and $BoidPos$ are the same as in Eq.\ref{cohesion\_equation}.

\subsubsection{Separation}

Separation is the force that stops flocks getting too tightly knit. By having enough space between flock members it grants maneuverability where there would otherwise be collisions that slow movement down; that could be from obstacles, other moving entities or other boids in the flock. The unit vector that represents the direction for the force to be applied is calculated