Project plan document - (20% of the Project's final grade)

## Comprehensive, accurate, and detailed use case description that effectively communicates the project's purpose, goals, and users' needs.

“Boids” are a simulation of natural flocking behaviour exhibited by birds that was developed by Craig Reynolds in 1986[[1]](#footnote-1). It defines the motion of each boid based on the position and velocity of other nearby boids. There are 3 key concepts defining their behaviour:

1. Separation: each boid will steer away from other boids within a protected radius
2. Alignment: each boid will steer towards alignment with the common velocity of boids within a visual radius.
3. Cohesion: each boid will steer toward the centre of the group of other boids within the visual radius.

The implementation used in this project leverages a Cornell University algorithm[[2]](#footnote-2). It includes several tuneable parameters which define their behaviour.

This project will implement a basic boid program with the addition of a predator boid who will hunt the other boids based on an algorithm of our creation or alternatively be controlled by a human player. Fleeing behaviour of prey boids will also be of our creation. The goal of the player will be to hunt as many boids as possible.

This project will use SFML for the visual interface.

This simple implementation of boids will not be able to handle large flocks without significant optimisation. This optimisation is considered out-of-scope so the flock size will be limited by game performance.

## Comprehensive and well-defined list of potential classes that accurately reflect the project's requirements and demonstrate an understanding of object-oriented design principles.

The following classes will be implemented:

1. Flock. A class composed of an array of Boid classes. It will keep track of the boids and be the primary entry point for the main program to run the simulation.
2. Boid. An interface to be implemented by predator and prey classes that inherit from this as a base class. It will enforce an update function intended to calculate the position and velocity of the respective boids as well as drawing them to the window.
3. PreyBoid. A child of Boid that implements the flocking behaviour described above.
4. PredBoid. A child of Boid that implements predator behaviour over the PreyBoids.
5. PlayBoid. A child of PredBoid that accepts user input to navigate the predator boid instead of being calculated algorithmically.

Several classes from SFML will be used to provide the visual interface.

## Comprehensive and well-defined list of potential data and function members for each class that aligns with the class's responsibilities and demonstrates encapsulation.

1. Flock.
   1. Data:
      1. basic data members to hold the addresses and quantities of boids.
   2. Methods:
      1. A constructor that will create the flock based on input parameters.
      2. An update function that cycles through each boid.
      3. A destructor to ensure the clearance of memory for each boid.
2. Boid.
   1. Data:
      1. None.
   2. Methods:
      1. A pure virtual function to update the position of the boids and draw it to the screen.
3. PreyBoid.
   1. Data:
      1. sf::Vector2 attributes for position and velocity.
      2. relevant parameters for the world they occupy.
      3. Various parameters to define flocking behaviour.
   2. Methods:
      1. A constructor to initialise the boid and apply their behaviour parameters.
      2. Various functions to calculate flocking behaviour.
      3. A function to draw the boid to the window.
4. PredBoid.
   1. Data:
      1. sf::Vector2 attributes for position and velocity.
      2. relevant parameters for the world they occupy.
      3. Various parameters to define hunting behaviour.
   2. Methods:
      1. A constructor to initialise the boid and apply their behaviour parameters.
      2. Various functions to calculate hunting behaviour.
      3. A function to draw the boid to the window.
5. PlayBoid.
   1. Data:
      1. Relevant data to manage user input.
   2. Methods:
      1. A constructor to initialise the predator.
      2. Methods to convert user input to movement

## A well-defined, accurate, and comprehensive description of the relationships between the identified classes that demonstrate an understanding of the various types of relationships (e.g., inheritance, composition, association).

1. Flock will be composed of Boid interfaces
2. PreyBoid and PredBoid inherit from Boid to implement their respective behaviours.
3. PlayBoid will inherit from PredBoid.
4. Each child of Boid will contain a reference to the sf::RenderWindow in use so that it can internally draw itself onto the window (aggregation).

## A well-defined and structured list of project tasks in a logical sequence and a realistic and achievable timeline that demonstrates an understanding of the project's requirements and dependencies.

1. Develop the basic boid flocking behaviour.
2. Implement this in the structure defined above.
3. Implement the PredBoid. This will require position and velocity data to be known between PredBoid and PreyBoid.
4. Implement PlayBoid.
5. Implement UI to adjust flocking behaviour ono the fly.

## Clear, concise, and accurate description of how the user will interact with the program that demonstrates an understanding of the user's needs.

The user will either simply adjust the flocking behaviour parameters for the visual appeal, or they will play as a predator boid hunting the prey boids.

## Well-defined and comprehensive plan for unit testing and debugging.

To minimise bugs entering each class will start out only implementing necessary features. The program will be written in a way that exceptions are avoided.

1. (Reynolds, n.d.) [↑](#footnote-ref-1)
2. (Adams, n.d.) [↑](#footnote-ref-2)