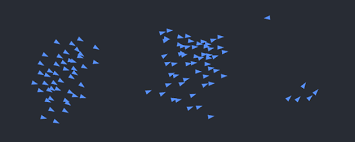
****

## **Term Project Proposal**

## **BOID algorithm - simulation of moving groups of animals**

**Project Description**

Boid algorithm was developed by [Craig Reynolds](https://en.wikipedia.org/wiki/Craig_Reynolds_(computer_graphics)) in 1986, which simulates the [flocking](https://en.wikipedia.org/wiki/Flocking_(behavior)) behaviour of birdsand related groups of animals’ motion. While it has primarily been applied in animation and visual effects for movies, often with minimal interactivity, this project takes a different approach. The main **motivation** of me undertaking this project comes from a larger Research Project that aims to train a machine learning model for accurately counting animals moving in aggregations. One of the fundamental steps of the research is to accurately imitate behaviors of moving groups of animals. By implementing this simulation by adding extra **interactivity** and **control** for the user, I aim to come up with a simulation that is both *unique in the field of boids simulation* and also a *meaningful* contribution to my larger research project.

**Competitive Analysis**

There are numerous projects that have successfully implemented the Boid algorithm to simulate the movement of animal groups. While Craig Reynolds' original paper describes the algorithm in a generic manner, the main challenge lies in **translating** these rules into a working simulation. Many implementations exist in both 2D and 3D, showcasing realistic flocking behavior. One of the best 2D examples I found is [this JavaScript-based simulation](https://eater.net/boids), which effectively demonstrates the algorithm's core principles.

My project differentiates itself by significantly increasing interactivity and complexity beyond traditional boid simulations. Unlike most existing implementations, which focus solely on the three fundamental flocking rules which are known as cohesion, separation and alignment. In addition to these, my simulation will incorporate

* predator avoidance
* user control over the number of boids
* adjustable visual range
* real-time boid creation via mouse clicks
* predator chase
* interactive obstacles that mimic real-world constraints like trees and other features as my creativity allows

Additionally, this will be the first project to integrate these advanced behaviors using cmu\_graphics, making it unique in both its scope and implementation framework.

**Structural Plan:** A structural plan for how the finalized project will be organized in different functions, files and/or objects.

1. Create a *readme* file to explain how the simulation is working, how it uses existing pseudo-code, challenges, uniqueness, major improvements etc.
2. I will have a class object for creating and controlling movement of all boids (this is a must!)
3. I will create a class object for the menu bar, and potentially child classes for functions inside the menu
4. Class object for the buttons on the screen (cohesion, alignment, separation)
5. Class object for the predators
6. Separate file for the “special game” that implements boids algorithms on top of images
7. I will use helper functions to implement edge avoidance and to apply three fundamental rules of the boids movements.
8. I also may need many other helper functions, I may create a separate file depending on the need to keep them separately

*I am aware that my current code needs better organization and better usage of OOPs, and I will be working on that.*

**Algorithmic Plan**: A detailed algorithmic plan for how you will approach the trickiest part of the project.

Probably the trickiest part of the project lies in translating the generic algorithm of Boids from the original paper into the code using cmu\_graphics. Further complications lie in adding complexities that are both interactive and simulate the natural world.

**Make the boids follow the algorithm! (done this part so far)**

(brain breaking\*)

1. Start with drawing randomly generated dots on the screen

2. Make the dots move in random directions (start using cmu-graphics)

3. Give triangular shape to the objects to simulate birds

4. Apply classes to create boids efficiently, that takes random x, y values, and x, y vectors

5. Research how to work with vectors (how they impact speed and direction angle)

6. How to give the right direction to the boids, so they move to the side of the pointer?

7. Use atan2 to get the right angle between 180 and -180 using the vector values

8. First major challenge: make boids avoid the edges

9. Manipulate x, y as the boid exceeds the margin to create a smooth edge avoidance behavior

10. Boids moving randomly avoiding the edges, apply 3 rules of boids!

11. In depth reading/watching almost all existing videos on youtube (none of them written in python)

12. All 3 rules work by slightly adjusting the vector (which impacts direction angle and speed of the boids) in each frame, based on the behavior of the neighboring boids (this means all three functions return 2-value tuples or lists to by which current boids behavior will be slightly impacted)

13. Create neighbor function that takes all boids and returns neighbors

14. In each frame, for each boid apply 3 rules of boid algorithm

15. Cohesion rule: Makes each boid steer slightly toward the average position of its nearby neighbors, promoting group togetherness.

16. Alignment rule: Adjusts a boid’s velocity to align with the average direction and speed of its neighbors, creating synchronized movement.

17. Separation rule: Steers a boid away from nearby boids if they get too close, preventing crowding or collisions.

**Upcoming challenges in general, detailed algorithm will be created as I progress:**

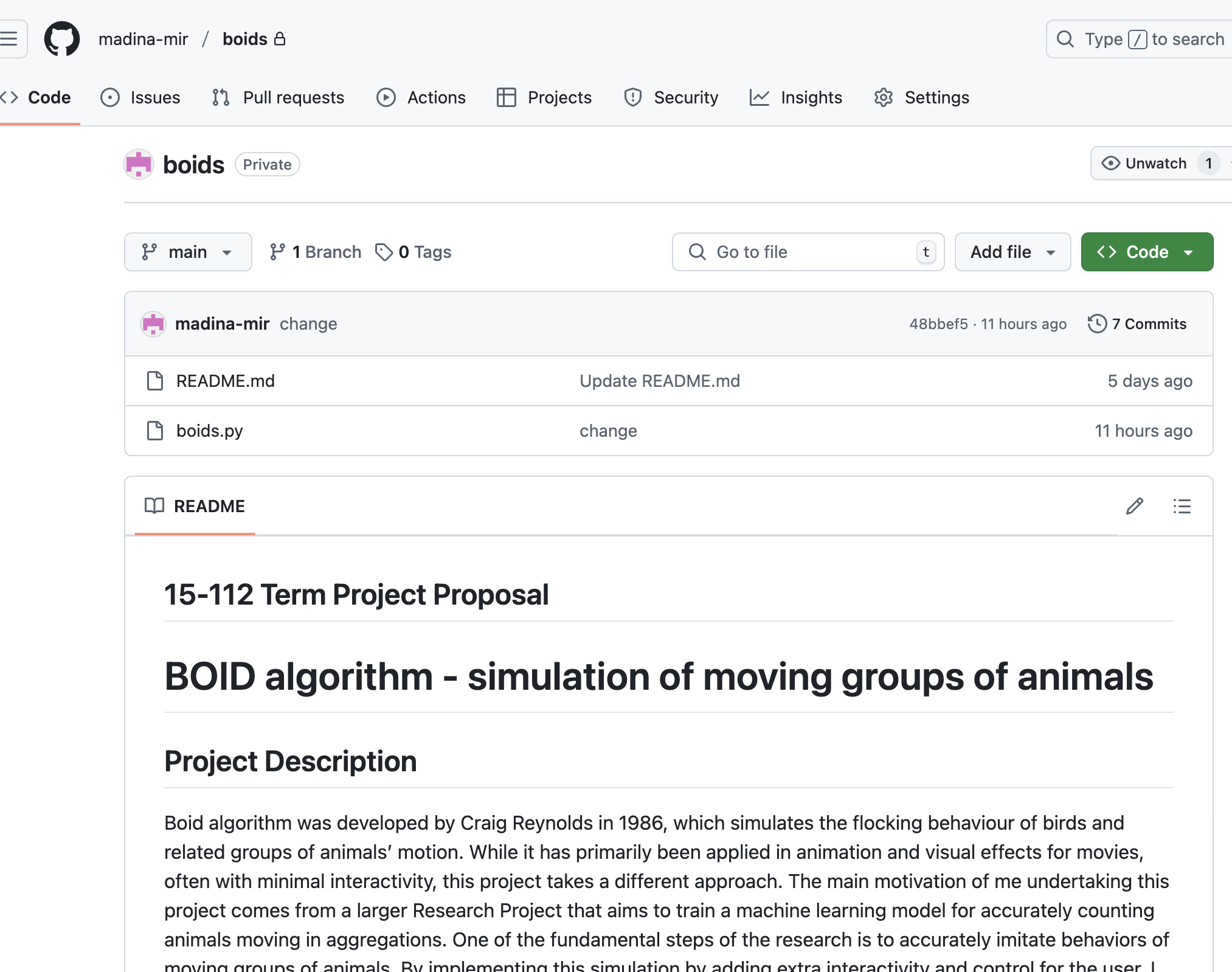
1. Create a menu bar in the top right corner with all the user-controllable features including but not limited to
   1. Predator creation
   2. Predator control using the arrows
   3. Making each boid avoid the predators
   4. Creation of obstacles
   5. Making the boids avoid the obstacles
   6. Special game button (let’s keep it for later)
2. Create interactive user-controllable Visual range changing button on the right-bottom of the screen
   1. This should include reset button
   2. +- buttons that impact the radius perspective by which each boid views its neighbors
3. Further challenge
   1. The cmu-graphics handles 150-200 moving boids well, yet it clearly suffers as the number increases
   2. **Problem: adding more features means adding more loops that loop each boid in each frame, which is extremely costly when it comes to smooth movement of my boids**
   3. Research potential **complexity reduction techniques** depending how smooth the boids will act after new features are added

**Timeline Plan: A timeline for when you intend to complete the major features of the project.**

1. Complete the *working* welcome page and menu bar appearance (though features don’t work) by 12th April
   1. During the weekend work on the design, color template, and potential music effects (optional)
2. Make most features in the menu bar work except the special game by MVP submission deadline
3. Work on the “special game”, and all the features in the menu bar by 19th April
4. Work on code optimization and organization and allow time for further creativity till 22nd of April

**Version Control Plan: A short description and image demonstrating how you are using version control to back up your code. You must back up your code somehow!!!**

I am using github!



**TP2 Update**

These are the things I worked since TP1 submission

1. Organized boid rule buttons in a class and created a new file for that (I almost re-wrote the whole thing)
2. Organized welcome page using a class and a new file
3. Organized onAppStart for better readability
4. Created the menu tab using a class object that includes extra features
5. Added add boid button and implemented its behavior
6. Added Obstacle creator button and implemented its behavior
7. Added Predator mode that gives user ability to chase the boids
8. When “R” is clicked the simulation will restart

**TP3 Update**

1. Implemented Quadtree for efficiency, now my boids freely run up-to 500 or even more boids
2. Added on/off button for drawing the grids which takes the values from the calls in Quadtree
3. Added “special game”
4. Improved user interface
5. Added Tail on and off button
6. Added Day on and off
7. Better organized the files
8. Updated the readme file
9. Added acknowledgement to the readme (tradition among boid algorithm implementers)

**Sources, notes, personal**

Original source by the author of the algorithm <https://dl.acm.org/doi/abs/10.1145/37401.37406>

Wikipedia

<https://en.wikipedia.org/wiki/Boids>

Sample : <https://eater.net/boids> **primary reference**

**Primary pseudo-code relied** <http://www.kfish.org/boids/pseudocode.html>

How it works:

* Each Boid has properties that determine how it interacts with others:
  + Separation: Boids try to avoid crowding their neighbors.
  + Alignment: Boids try to align their direction with the average heading of their neighbors.
  + Cohesion: Boids try to move towards the average position of their neighbors.
* These forces are calculated for each Boid, and a resultant force changes the Boid's direction.
* A "wandering" behavior is also implemented to make the Boid's movement more natural and less uniform.
* Boids are drawn on the screen as simple triangles, the orientation of which depends on their current velocity.

Some good sources <https://codesandbox.io/p/sandbox/github/SverreNystad/boids-in-python>

<https://github.com/meznak/boids_py>

<https://youtu.be/NTBYAs_zI4E?si=K-Kz9U5MfIUgBH6F>

<https://github.com/jackaperkins/boids>

<https://youtu.be/QbUPfMXXQIY?si=sCOAfjUBUSQao4Dy>

Math <https://youtu.be/woTqSlyJP7Q?si=IeUJX0OHkl64wBPu>

<https://youtu.be/woTqSlyJP7Q?si=mB1LRrUjf1FJUUJw>