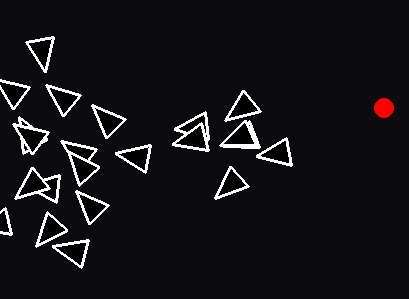
BOGICHIB  
A SIMULATION BY THE *GEOMATICS*



By: [Nehan Mohammed](mailto:nehan.mohammed@student.tdsb.on.ca), Simon Akhter, & Tahseen Rayhan

**Program Info**

BOGICHIB is a Python program that simulates the collective behavior of a flock of birds. The simulation is set in a 2D environment where birds interact with each other and with a predator. The objective of the simulation is to showcase behavior observed in real-world bird flocks, such as alignment, cohesion, separation, and predator avoidance.

To create the simulation, we primarily used the Python programming language along with the Pygame library for graphics rendering. The simulation's assets, including bird sprites and environment backgrounds, were created using pixel art tools. The Replit IDE was utilized for writing and testing the code, providing an efficient development environment.

**Libraries**

**pygame**: Utilized for creating the graphical user interface and rendering sprites within the simulation environment.

**random**: Employed for generating random values used in initializing bird positions, velocities, and other stochastic processes within the simulation.

**math**: Used for mathematical calculations required for bird movement and interaction logic.

**copy**: Utilized for creating deep copies of objects to prevent unintended mutation during simulation updates.

**Analysis**

**Technical Analysis**:

* Evaluate different algorithms and techniques for simulating flocking behavior, considering factors such as performance and accuracy.
* Assess the feasibility of implementing future extensions, such as predator/prey simulation and obstacle avoidance, within the existing framework.
* Identify potential challenges and risks, such as computational complexity and optimization requirements.

**Design**

**System Architecture**:

* Define the overall structure of the simulation system, including modules for rendering, simulation logic, user interaction, and parameter control.
* Determine the interaction flow between different components of the system, such as how user input affects object behavior and parameter adjustments Consider the use of a modular design, involving object classes and functions, to facilitate future extensions and modifications.

**PSEUDOCODE**

class Bird:

def\_\_init\_\_

# Initialize position, velocity, maxSpeed, acceleration,

def compute\_acceleration

# Reset acceleration

# Calculate and add alignment acceleration

# Calculate and add cohesion acceleration

# Calculate and add separation acceleration

# If predator exists, calculate and add avoidance acceleration

def update

# Compute new acceleration

# Update velocity and position

# Ensure the bird stays within boundaries

# Update the angle based on velocity direction

def draw

# Calculate the rotated vertices for the bird shape

# Draw the bird on the screen

class Predator:

# Initialize position, max\_speed, history

def recursive\_update

# If depth exceeds limit or no targets, return current position

# Calculate future position based on target position

# Update position and recursively call with the next target

def update

# If target positions exist, update position using recursive update

# Add the current position to history

def draw

# Draw a circle representing the predator on the screen

class Boundary:

def \_\_init\_\_

# Set minimum and maximum x and y values

def size\_x

# Calculate the size in x direction

def size\_y

# Calculate the size in y direction

def periodic\_project

# Ensure a point stays within boundaries using periodic wrapping

def periodic\_displacement

# Calculate the shortest displacement between two points considering periodic boundaries

class UIManager:

def \_\_init\_\_

# Set up sliders for number of birds and bird speed

# Set up button for adding a predator

def process\_events

# Handle UI events like slider movements and button presses

def update

# Update UI elements

def draw

# Render UI elements on the screen

class Simulation:

def \_\_init\_\_

# Set up the window, clock, and initial parameters

# Initialize UI manager

# Create an initial group of birds

def run:

while True:

# Loop until exit condition:

# Process events

# Update UI manager

# Clear the screen

# Update and draw predator if it exists

# Update and draw each bird

# Handle bird-predator interactions

# Draw UI elements

# Update display

# Main script

if \_\_name\_\_ == "\_\_main\_\_":

# Initialize Pygame

# Create a simulation instance

# Run the simulation

**UML DETAILED DIAGRAMS**

| Class Name: Bird | | | |
| --- | --- | --- | --- |
| Method Name | Parameters | Returns | Description |
| \_\_init\_\_ | (x: float, y: float) | n/a | * Description: Initializes a Bird object with the given position. |
| setRuleWeights | n/a | n/a | Sets the weights and radii for the bird's flocking rules. |
| computeAcceleration | group: list - List of Bird objects representing the flock.  - predator: Predator - An optional instance of the Predator class representing a predator. | n/a | Computes the acceleration based on the flocking rules and predator avoidance. |
| ruleAlignment | group: list - List of Bird objects representing the flock. | Acceleration limit(acceleration, self.maxAcceleration) | Implements the alignment rule to steer towards the average heading of neighbors. |
| ruleCohesion | group | Acceleration  limit(acceleration, self.maxAcceleration) | Implements the cohesion rule to steer towards the average position of neighbors. |
| ruleSeparation | group | Acceleration  limit(acceleration, self.maxAcceleration) | Implements the separation rule to steer away from nearby neighbors. |
| ruleAvoidPredator | predator | avoidance | Implements the avoidance rule to steer away from predators. |
| Draw | screen  - distance:  - scale: | n/a | Draws the bird on the screen with appropriate transformations. |

\*ARROW TO SIMULATION

| Class Name: Predator | | | |
| --- | --- | --- | --- |
| Method Name | Parameters | Returns | Description |
| \_\_init\_\_ | x, y, bird\_max\_speed | n/a | - Initializes a Predator object with the given position and maximum speed. |
| recursive\_update | target\_positions, depth=0, max\_depth=3 | Self.position- The updated position of the predator. self.recursive\_update(target\_positions[1:], depth + 1, max\_depth) | Recursively updates the predator's position based on target positions. |
| update | target\_positions | n/a | Updates the predator's position based on target positions. |
| Draw | screen | n/a | Draws the predator on the screen as a circle. |

\*ARROW TO SIMULATION

| Class Name: Boundary | | | |
| --- | --- | --- | --- |
| Method Name | Parameters | Returns | Description |
| \_\_init\_\_ | min\_x, max\_x, min\_y, max\_y | n/a | Initializes a Boundary object with the given bounds. |
| size\_x | n/a | self.max\_x - self.min\_x | Computes the size of the boundary along the x-axis. |
| size\_y | n/a | self.max\_y - self.min\_y | Computes the size of the boundary along the y-axis. |
| periodicProject | p - pygame.Vector2 - The vector to project periodically within the boundary. | n/a | Projects the given vector periodically within the boundary. |
| periodicDisplacement | p: pygame.Vector2 - The first vector.  - q: pygame.Vector2 - The second vector. | displacement | Computes the periodic displacement between two vectors. |

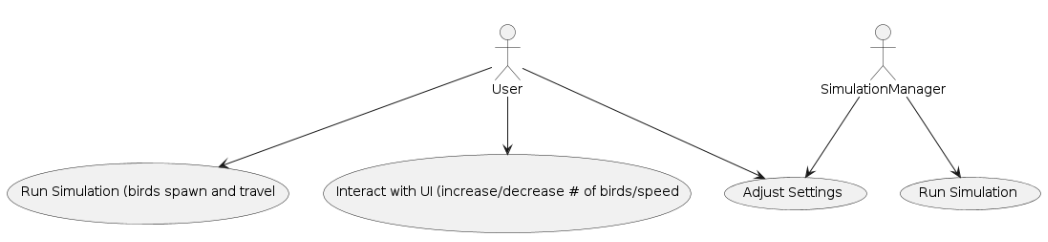
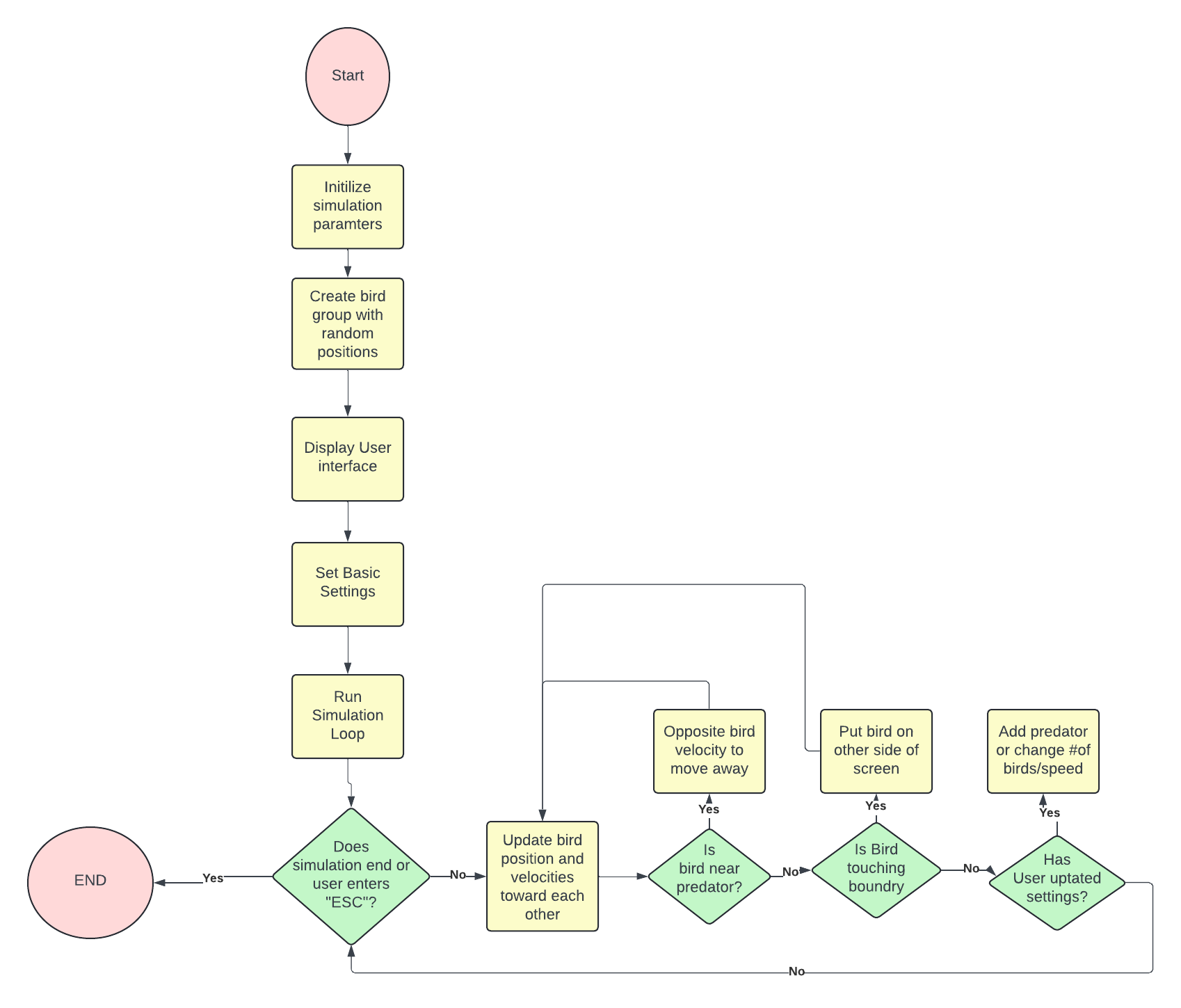
\*ARROW TO SIMULATION

| Class Name: UIManager | | | |
| --- | --- | --- | --- |
| Method Name | Parameters | Returns | Description |
| \_\_init\_\_ | width, height | n/a | Initializes a UIManager object with the given width and height. |
| process\_events | event | n/a | Processes events for the UI elements. |
| update | time\_delta | n/a | Updates the UI elements. |
| draw\_ui | window | n/a | Draws the UI elements on the window. |

\*ARROW TO SIMULATION

| Class Name: Simulation | | | |
| --- | --- | --- | --- |
| Method Name | Parameters | Returns | Description |
| \_\_init\_\_ | width, height | n/a | Initializes a Simulation object with the given width and height. |
| run | n/a | n/a | Runs the simulation loop, handling events, updating the flock and predator, and drawing the scene. |

**FLOW CHART FINAL COPY**

**USE CASE DIAGRAM** 

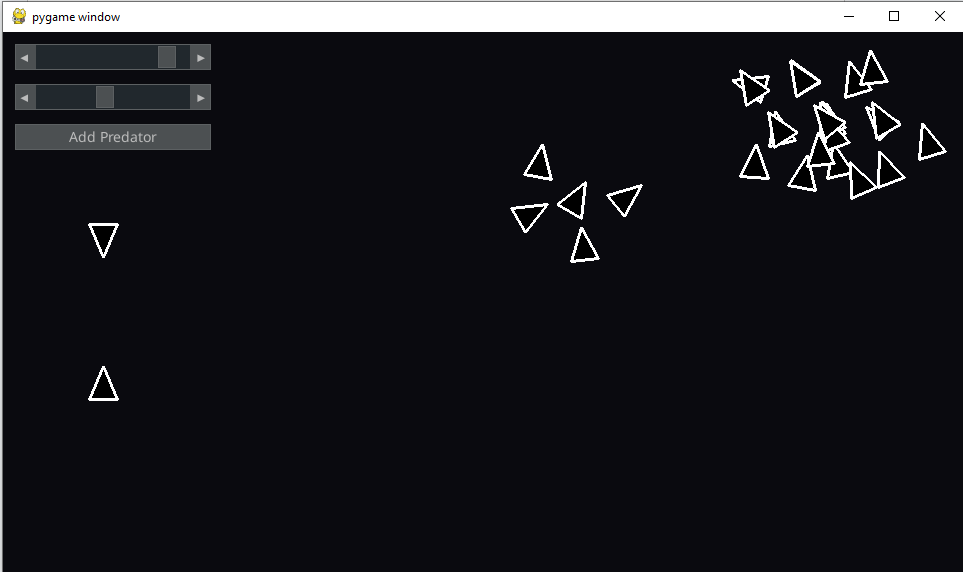
# Test Cases

## **One**

1. Start the simulation.
2. Keep default number of birds (30) and do not change the slider position
3. Keep default speed of birds (5) and do not change the slider position
4. Observe the flocking behavior of the birds for 60 seconds.

***Expected Behavior:***

* Birds should exhibit alignment, cohesion, and separation behaviors.
* Birds should move smoothly within the boundary without getting stuck.
* The flock should generally stay together, adjusting their directions as a group.



## Two

1. Start the simulation.
2. Set the number of birds to 20 using the slider.
3. Set the bird speed to 4 using the slider.
4. Click the "Add Predator" button to introduce a predator into the simulation.
5. Observe the interaction between the predator and the birds for 60 seconds.

#### Expected Behavior:

* Birds should exhibit avoidance behavior when the predator is nearby.
* The predator should move towards the birds and attempt to follow them.
* Birds close to the predator should change direction to avoid it.
* If a bird comes in contact with the predator, it should be removed from the simulation.



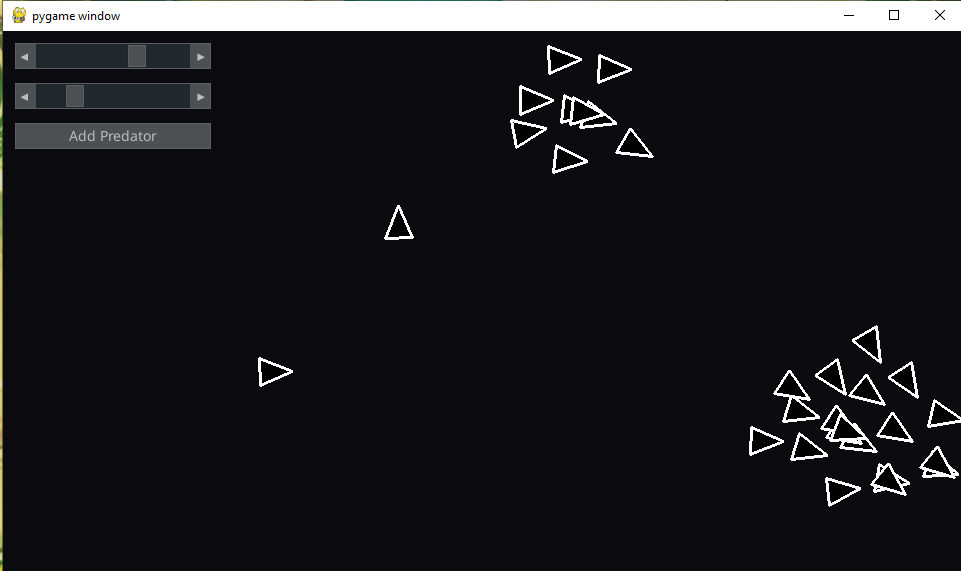
## 

## Three

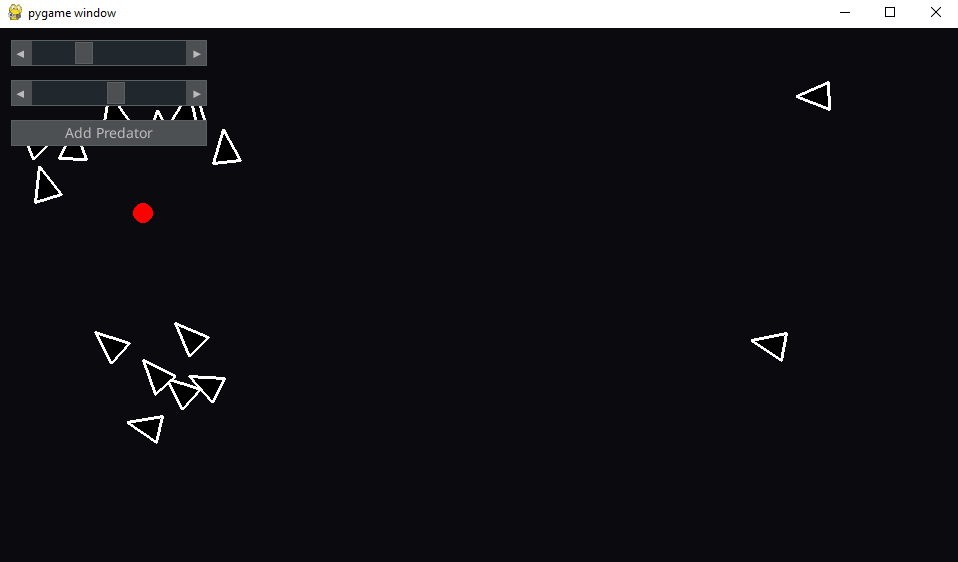
1. Start the simulation.
2. Set the number of birds to 25 using the slider.
3. Set the bird speed to 3 using the slider.
4. Observe the flocking behavior for 20 seconds.
5. During the simulation, change the number of birds to 15 using the slider.
6. Change the bird speed to 7 using the slider.
7. Click the "Add Predator" button.
8. Observe the interactions for another 30 seconds.

***Expected Behavior:***

* Initially, the simulation should have 25 birds moving at speed 3.
* After changing the number of birds, the simulation should adjust to 15 birds.
* Birds' speed should change dynamically to 7 after adjusting the slider.
* Adding the predator should introduce the avoidance behavior immediately.
* The simulation should continue to run smoothly with the updated parameters.



After Changes:



**SCRUM MEETING ONE ANSWERS (Wed. May 22)**

**Nehan:**

1. What they did since last meeting

* Implemented the ruleSeparation function in the Object class to calculate separation acceleration between objects to avoid crowding.
* Updated the setRuleWeights function to set default values for object parameters such as maxSpeed, maxAcceleration, separationWeight, and separationBoundary.
* Assisted in debugging the update function in the Object class to ensure smooth object movement and velocity calculations.

1. What they are currently working on for the project

* Attempting to change the shape of the object, from a square to a more “living” type object that moves smoother and turns

1. What issues are blocking them

* Having to apply vector concepts to for the object to turn and move in groups
* Struggling to use sine and cosine functions for object to rotate
* Numpy library is hard to use

**SCRUM MEETING TWO ANSWERS (Mon May 27)**

**Nehan**

1. What they did since last meeting

* Assisted on changing the boundaries for birds to go on the other side
* Assisted in the group following algorithm for flocks
* Changed the shape of object

1. What they are currently working on for the project

* Attempting to add a predator for birds to run from

1. What issues are blocking them

* Determining the correct size and speed for the predator
* Algorithm for running chasing specific birds (use recursion or queue)??

**SCRUM MEETING THREE ANSWERS (Fri. May 31)**

**Nehan**

1. What they did since last meeting

* Created the predator but doesn't function properly

1. What they are currently working on for the project

* Attempting to fix the predator to chase the flock of birds
* Attempt to make work for object oriented

1. What issues are blocking them

* Algorithm to chase birds isn't working properly
* Many bugs

**SCRUM MEETING FOUR ANSWERS (Thurs. Jun 4)**

**Nehan**

1. What they did since last meeting

* Fixed predator issues and will now chase the flock of birds targeting one bird and delete if caught
* Made code object oriented, split it into different files

1. What they are currently working on for the project

* Attempting to add sliders to change speed and number of birds on screen

1. What issues are blocking them

* Struggling to use pygame GUI library and its contents
* Many bugs

**Scrum Meeting One Answers (Wed. May 22)**

**Simon**

**what they did since last meeting**

* Assisted with the implementation of ruleSeparation function.
* Assisted in debugging the update function in the Object class to ensure smooth object movement and velocity calculations.

**what they are currently working on for the project**

* Adjusting number of birds and speed of birds through num\_birds\_slider and speed\_slider

**what issue s are blocking them**

* Finding speed ranges and number of birds appropriate for the window size

Scrum Meeting Two Answers **(Mon. May 27)**

**Simon**

**What I did since last meeting:**

Assisted with the ruleAlignment and ruleCohesion functions to ensure proper flocking behavior.

Integrated pygame\_gui components for the num\_birds\_slider and speed\_slider to allow dynamic adjustment of bird parameters.

Conducted initial tests to verify the behavior of birds with the new GUI controls.

**What I am currently working on for the project:**

Developing the Predator class to add a predator entity that interacts with the flock.

Implementing the ruleAvoidPredator function to enable birds to avoid the predator.

Fine-tuning the visual representation of birds and the predator for better clarity.

**What issues are blocking them:**

Balancing the predator's speed and avoidance radius to ensure realistic interactions without making the simulation too easy or too difficult.

Ensuring the periodic boundary conditions work seamlessly when the predator is introduced.

**Third Scrum Meeting Notes (Fri, May 31)**

**Simon**

**What I did since last meeting:**

Assisted the Predator class and integrated it into the simulation.

Implemented the ruleAvoidPredator function to handle birds' avoidance behavior towards the predator.

Enhanced the visual representation of birds and the predator, making them more distinguishable.

**What I am currently working on for the project:**

Debugging the collision detection mechanism to accurately remove birds that come in contact with the predator.

Conducting extensive testing to ensure all flocking behaviors (alignment, cohesion, separation, avoidance) are functioning as intended under different conditions.

Adding final touches to the user interface, ensuring sliders and buttons are responsive and intuitive.

**What issues are blocking them:**

Occasional crashes or unexpected behavior when multiple birds collide with the predator simultaneously.

Performance optimization to maintain smooth frame rates as the number of birds increases, especially with the predator actively chasing the flock.

Scrum Meeting Notes Four (**Thurs. Jun 4**)

**What I did since last meeting:**

Successfully debugged the collision detection mechanism to accurately remove birds that come in contact with the predator.

Optimized performance to ensure smooth frame rates, even with higher numbers of birds and an active predator.

**What I am currently working on for the project:**

Finalizing the project documentation, including detailed comments in the code and a user guide for the simulation.

Preparing a demonstration of the simulation to showcase all functionalities and features.

Gathering feedback from initial testers and making any necessary adjustments based on their input.

**What issues are blocking them:**

No major issues are currently blocking progress; focus is on final refinements and ensuring a polished end product.

Minor adjustments may be needed based on feedback from the final round of testing and demonstration.

**Scrum Meeting One Notes (Wed. May 22)**

*Tahseen Rayhan*

**What I did since last meeting:**

* Developed the rules, like Cohesion, in the Bird class to ensure birds move towards the average position of the neighboring ones.
* Assisted with testing the updated functions in the Bird class, to make sure that the bird velocity and position updates correctly

**What they are currently working on for the project:**

* Experimenting with different bird shapes, like squares and triangles, and how to make them rotate
* Started to investigate how we could implement an obstacle avoidance rule for the birds.

**What issues are blocking them:**

* Have to use trigonometry to rotate the birds and posed as a really big challenge

**Scrum Meeting Two Notes (Mon. May 27)**

*Tahseen Rayhan*

**What they did since last meeting:**

* Helped with the team to “map” screen boundaries, so a bird that traveled off the screen reappears on the other side

**What they are currently working on for the project:**

* A predator feature, which the birds react to, and run away

**What issues are blocking them:**

* Finding out what the appropriate parameters are for predator speed
* Finding an efficient algorithm for the predator-prey interactions, especially for the list of bird objects, seems time consuming

**Scrum Meeting Three Notes (Fri. May 31)**

*Tahseen Rayhan*

**What they did since last meeting:**

* Incorporated the predator in the main simulation loop to chase birds

**What they are currently working on for the project:**

* Debugging and fixing the predator's behavior so it pursues the birds.

**What issues are blocking them:**

* The predator's chasing algorithm was malfunctioning and became really unpredictable
* Got many bugs when the bird and predator interacted.

**Scrum Meeting Four Notes (Thurs. Jun 4)**

*Tahseen Rayhan*

**What they did since last meeting:**

* Helped with making the project object-oriented

**What they are currently working on for the project:**

* Added sliders to the UI for adjusting the speed and number of birds in real-time during the simulation

**What issues are blocking them:**

* Bugs when integrating the new UI elements with the existing code.

Summary

The primary goal of this program is to simulate the flocking behavior of birds, incorporating natural behaviors such as alignment, cohesion, and separation. Additionally, the program includes the ability to introduce a predator, adding a dynamic element where birds exhibit avoidance behavior. Simulation is run through a Pygame window with Pygame GUI components for user interaction. Birds steer towards the average heading of the nearby flock. Birds steer to move towards the average position of nearby flock. Birds steer to avoid crowding nearby flock using a separation radius variable that manages the distance between. A predator can be added which chases birds. Its speed is set to 80% of the maximum bird speed. This simulation was handled using Pygame’s vector class for position and velocity calculations.