Developer's Manual

Overview

Boids is a program that simulates the movement behaviour of flocking animals. The movement of the entities in the Boids simulation is calculated by vector arithmetic. Each entity has a position and direction vector, and the movement of the entities is controlled by adding or subtracting direction vectors from the entity's position vector. These direction vectors vary depending on the type of entity, the other entities and obstacles surrounding it, and the user-defined custom settings.

Classes

Vector2D

A small class with an x and y field, used to represent the position and direction of each entity. Supports basic vector operations.

Fields

double x, y
Horizontal and vertical components (lengths), respectively.

Methods

public void add(Vector2D v)
Adds vector ***v*** *to the current vector (head-to-tail vector addition).*

public void sub(Vector2D v)
Subtracts vector ***v*** *from current vector (subtract v.x and v.y from x, and y respectively).*

public void mult(double n)
Multiplies the current vector by a constant (only changes magnitude).

public void div(double n)
Divides the current vector by a constant (only changes magnitude).

public void dist(Vector2D v)
Finds the distance between two points (represented here as position vectors) using Pythagorean Theorem.

public double magnitude()
Returns the magnitude of the vector (found using Pythagorean Theorem).

public void setMag(double n)
Sets the value of x and y fields given a magnitude, n (by dividing the current vector by a constant m = magitude() ÷ n [unless n = 0, in which case it sets x and y to 0]).

public void limit(double n)
*Changes the magnitude to the lesser of the two values, **n** and **magnitude()**.*

public void normalize()
*Changes the magnitude to 1 (using **setMag()**).*

public Vector2D copy()
*Returns a copy of the current vector (same **x** and **y** values).*

Flock

An object containing an "environment" (boids, predators, obstacles, walls).

Fields

LinkedList<Boid> boids

LinkedList<Predator> predators

LinkedList<Obstacle> obstacles

LinkedList<Obstacle> walls

Lists containing the respective objects.

double nearbyRadius, avoidRadius, coheseRadius, obstacleRadius, wallRadius
Radius/distance in which each vector (affecting the move vector) is applicable.

double predatorSpeed

double boidSpeed

The maximum speed that the predator/boid can move at.

double eatRadius
Distance around a predator in which boids are eaten (when a boid is within the distance, it "dies").

double avoidStr, coheseStr, obstacleStr, randomStr, wallStr
The strengths (multiplying factor for the magnitudes) of the avoid, cohese, obstacle, random, and wall vectors.

Constructor

public Flock()
Initializes LinkedLists: **boids, predators, obstacles,** and **walls**.

Methods

public static void defineWalls()

*Removes the previous obstacles in **walls**, then creates the new walls based off of **GUI**'s **AREA_WIDTH** and **AREA_HEIGHT**.*

public void addBoid(Boid b), removeBoid(int pos), addPredator(Predator p), addObstacle(Obstacle o)

Adds or removes the implied object from its respective LinkedList in this class.

public void updateAllPos()

Updates the position of all boids and all predators.

public void clrBoids(), clrPreds(), clrObs()

Clears the respective objects.

public boolean toggleWalls()

*Turns walls on (sets **wallStr** to 10) and returns true if **wallStr** is 0 (aka the walls are off). Turns walls off (sets **wallStr** to 0) and returns false if **wallStr** is 10 (aka if the walls are on).*

public void show(Graphics g)

*Draws **boids**, **predators**, and **obstacles** based on their set color, and positions. Also draws trails of the boids and predators if GUI's **trail** flag is true.*

Entity

Parent class of Boid and Predator.

Fields

Vector2D pos

Represents the position (x, y) of the entity.

Vector2D move

Represents the direction and speed (magnitude) in which the entity is heading.

Flock flock

The flock to which the entity belongs.

Queue<Vector2D> trail

A queue representing the positions of the particles left behind in the trail of the entity.

Methods

Vector2D getObstacleAvoid()

Returns a direction vector pointing away from nearby obstacles.

Vector2D getWallAvoid()

Returns a direction vector pointing away from nearby walls.

void addVectors(Vector2D... vectors)

*Adds each vector v in **vectors** to **move**.*

Vector2D getAvoid()

Returns a direction vector pointing away from nearby entities of the same type.

abstract void updatePos()
*Updates the **pos** vector of the entity by adding the **move** vector to it.*

abstract void updateMove()
*Updates the **move** vector of the entity, considering nearby entities and obstacles.*

void generateTrail()
*Gives the entity a trail by adding the current position to the queue **trail** and removing boids as necessary to keep the queue's size 50.*

void changeColour()
Changes color to a random colour.

void adjust()
*Updates the **pos** vector of the entity to wrap around if it is offscreen.*

Boid extends Entity

The main 'Entity' object (extends Entity class).

Fields

LinkedList<Boid> nearby
*Contains all boids within a distance of flock.**getNearbyRadius()**. When updating its position, the boid will only consider other boids that are in **nearby**.*

Constructor

public Boid (double x, double y)
*Creates an **Entity** object at position (x, y). Initializes **nearby** (linked list), **color**, and **trail** (queue).*

Methods

public void updateMove()
*Adds **align**, **cohese**, **avoid**, **predatorAvoid**, **obstacleAvoid**, **wallAvoid**, and **rand** vectors to obtain the latest heading of the boid (direction in which the boid is heading). Adds the vector, **pos**, of current boid to **tail** (queue), and removes the head of the queue **tail** if its size reaches 50 (not counting the vector **pos** that was just added).*

public void getNearby()
*Updates the ArrayList, **nearby**, to include all the boids within a distance of flock.**getNearbyRadius()** of the current boid, excluding the boid itself.*

Vector2D "getters"

- getAlign()
    - move towards the average heading of the nearby boids

- getCohesion()
    - move towards the average position of the nearby boids (cohese)
- getAvoid()
    - avoid other boids within a distance of flock.**getAvoidRadius()**
- getPredatorAvoid()
    - avoid predators within a distance of flock.**getObstacleRadius()**

public void updatePos()
*Updates the ArrayList of nearby boids with **getNearby()**, updates the heading (direction in which the boid moves) using **updateMove()**, and gets the new position by adding the **move** vector to the current position.*

Predator extends **Entity**

Represents a "predator," which attempts to chase and eat the nearest boid to it.

Constructor
public Predator(double x, double y)
*Creates an **Entity** object at position (x, y) and sets colour to a semi-random color.*

Methods
public void updateMove()
*Updates the **move** vector by adding vectors, **avoid**, **obstacleAvoid**, **wallAvoid**, **rand**, and **closest**. The vector **closest** is obtained by finding the vector between the predator and the boid nearest to it, and changing the magnitude of that vector to 0.1.*
public Vector2D getAvoid()
Returns a vector with its tail at the position of the predator and pointing away from the nearby predators.
public void updatePos()
*Updates the position of the predator by adding the latest **move** vector to the **pos** vector. Removes the boids within a distance of flock.**getEatDistance()** of the predator.*

Obstacle

Represents an obstacle which entities (boids and predators) avoid.

Fields
private Vector2D pos

The position of the obstacle.

Methods
All are getters and setters, as obstacles do not move.

GUI

All the visual aspects of the program are created in this class.

Fields
All the sliders, some of the buttons, a timer are declared within the scope of the class for use in multiple methods.

boolean trail
The state of whether trails are turned on or off. Also used in Flock class.

boolean start
The state of whether the simulation has been started or stopped.

Font textFont
The font used for all text displayed.

Color textColor
The background colour for text boxes.

Color bgColor
The background colour for panels. This is the default beige colour.

int AREA_HEIGHT, AREA_WIDTH
The dimensions of the simulation window. Also used in Flock class.

Timer t
Used to time the drawing updates so they happen consistently.

Flock f
The flock that all entities belong to, required for the Movement class.

Constructor (Main Window)
The timer is initialized to a speed of 15 ms.

JPanel content
*Contains the three main areas: **north**, **centre** and **south**.*

JPanel north
Contains the buttons in the upper section of the simulation window.

DrawArea board
Contains the simulation area where entities and obstacles are placed.

JPanel south

Contains the buttons in the upper section of the simulation window.

Methods

public void stateChanged(ChangeEvent e)

Handles any changes made to a slider and updates the corresponding values through Flock methods.

public void mouseClicked(MouseEvent e)

Detects where the mouse was clicked, determines whether there were any modifiers and creates the appropriate object:

- *Left-clicking creates a Boid. This is the default click.*
- *Right-clicking creates an Obstacle.*
- *Clicking with the Shift button held down creates a Predator.*

public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)

Required methods from MouseEvent. These are unused.

public void actionPerformed(ActionEvent e)

Detects when a button is clicked and executes the appropriate response.

- *Simulate: Determines the state of **start**, then toggles it and starts/stops the simulation.*
- *Settings: Declares and initializes the sliders. Each slider has a corresponding JTextArea used as a title, which is formatted using the **textFormat()** method. A new JFrame is then created and the components are added to it.*
- *Info: A simple JFrame that contains information about this simulation.*
- *Walls (seen as Walls On/Off): Determines the state of **walls**, then toggles it and turns on/off the walls. The text of the button is also changed after clicking to correspond with the current state.*
- *Boids, Predators, Obstacles: Clears the appropriate group using Flock methods.*
- *Theme (seen as Dark/Light): Determines the current colour of the background, then changes it to the other colour. The colour of the entities and obstacles are also changed using Flock methods. The text of the button is changed after clicking to correspond with the current state.*
- *Trail (seen as Trails: On/Off): Determines the state of **trail**, then toggles it and turns on/off the trails. The text of the button is also changed after clicking to correspond with the current state.*

private void textFormat(JTextArea... textAreas)

Configures settings for text areas, making them uneditable and changing the background colour and font.

private void buttonFormat(JButton... buttons)

Adds an ActionListener to buttons and turns off focus painting.

private void addCListeners(JSlider... sliders)

Adds a ChangeListener to sliders.

public static void setDimensions(int width, **int** height**)**

*Sets **AREA_WIDTH** and **AREA_HEIGHT** to the width and height specified in the parameters.*

private void init_font()

Imports the font used in the simulation- prints out an error if an exception occurs.

Movement (nested in GUI)

A helper class for drawing the simulation in time with the Timer.

Constructor

public Movement(Flock f**)**

Sets the flock to the current one provided by the GUI class.

Methods

public void actionPerformed(ActionEvent e**)**

Updates all positions of entities using Flock's **updateAllPos()** method, then repaints the JPanel.

DrawArea (nested in GUI)

Stores the simulation.

Constructor

public DrawArea(int width, **int** height**)**

Creates a default JFrame. Sets the preferred size of the window to the dimensions specified.

Methods

protected void paintComponent(Graphics g**)**

Draws the new image returned from **createImage()** onto the window.

private BufferedImage createImage()

Creates a new BufferedImage the size of the window and draws the current state of the simulation, using Flock's **show()** method.

Boids

The main class.

Fields

GUI window

The main window.

Methods

public static void main(String[] args)

Initializes the main window and flock. Adds a ComponentListener to the window for registering resizing updates.

FrameListener (nested in Boids)

A helper class to detect when the window is resized.

Methods

public void componentResized(ComponentEvent e)

Called when the main window is resized. Updates **AREA_WIDTH**, **AREA_HEIGHT** and the walls using **GUI** and **Flock** methods.