Arctic Outbreak Developer’s Guide

Overview

Arctic outbreak is an animal interaction simulation which takes place in the arctic. It uses a GUI interface and Java backend to simulate how the animals, penguin, fish, whale, and seal would hunt and mate in the environment. The program uses many different classes including the inheritance of the animal class by the different types of animal classes. At each 50 millisecond, the game repaints the location and the number of each animal.

Classes

Main Class

This class is used to display the game screen.

`public static void main(String[] args)`

- The main method allows for the window to be visible and disables the resizability of the window.

Class MenuScreen (extends JFrame implements MouseListener)

This class is used to display the home screen of the game. It loads the background images and sets up the coordinates for the user’s mouse to select.

`public MenuScreen()`

- The method sets up the window attributes and loads the images on the home screen.

`class DrawArea extends JPanel`

- Displays the menu page and Instructions page.

`public void mouseClicked(MouseEvent e)`

- The method sets up the parameters of each button in which the user can click on to access other screens.

```
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void actionPerformed(ActionEvent e)
```

- Methods for the user’s cursor to function

`public static void main(String[] args)`

- Every main method of each class is able to set up the visibility of the screen.

class SelectScreen extends JFrame

This class is for the first screen that appears after the user selects “start”. It prompts for the user to enter the amount of each animal. It also sets up a default number for each animal in the case where there is no number entered.

`public SelectScreen(int width, int height)`

- This method prompts the user to enter numbers and provides boxes to type in the answer

`public void actionPerformed(ActionEvent ae)`

- Reads the input from user and checks for integers

`public void disposer()`

`class DrawArea extends JPanel`

- Determines the size of screen

`public static void main(String[]args)`

- Makes screen visible and non adjustable

class GameScreen extends JFrame implements ActionListener

This class sets up the actual game screen simulation after the user inputs the various amounts. It sets up objects for each component of the game the numeric variables. It also sets up an arraylist for the animals.

`public GameScreen(int width, int height, int f_amount, int p_amount, int s_amount, int w_amount, boolean debug_mode)`

- The method generates the amount of each animal according to the user input
- It sets up a timer for the game that updates every 50ms
- It also adds the options to quit the game

`public void disposer()`

- Separate exit method

`public void actionPerformed(ActionEvent e)`

- The method updates the number of animals remaining after ever update and displays the number on the screen

`class DrawArea extends JPanel`

- The class displays the images(animals, background, bars) according to their coordinate points
- It also sets up the hunger and mating bars on top of the animals if user selected to display this earlier in the simulation

class Entity

- This is the base entity class where base variables are declared that are required of general animal functionality

class Animal extends Entity

- This is the parent class of the animal where the general variables and methods are defined and can be added to with child classes

`public Animal(int max_x, int max_y)`

- This constructor is used to randomize the start coordinates of all animals

`public void update(ArrayList<Animal> animals)`

- The method updates the state of the animal, its movement, eating or mating

`private void movement()`

- The method determine the movement of the animals
- The animal starts with a randomized direction and only changes direction 5% of the time, it would move in the same direction otherwise
- It determines the coordinates moved by the move distance and velocity which differs for each animal
- It also allows the animals to move to the other side of the screen if it moves out of the screen

`private void check_action(ArrayList<Animal> animals)]`

- This method loops through each animal to check for position, hunger and mating

`void check_eat(Animal a, ArrayList<Animal> animals)`
`Animal create_child()`

- Dummy classes for the child classes to override

`private void check_mate(Animal a, ArrayList<Animal> animals)`

- The method checks if the animal interacted with its own kind
- If yes - calls the "create_child" method (adds a new animal to the screen)

`private void update_counters(ArrayList<Animal> animals)`

- This method updates the values for hunger and mating
- If the hunger is zero, the animal is removed from the game (dead)

class Fish extends Animal

- This class is a child class of Animal, it adds to the methods, making specifications for the fish animal

`public Fish(int max_x, int max_y)`

- The constructor defines the specific values for each variable according tot eh animal fish
- It defines width, height,, move distance, hunger rate, max full, max hunger, current hunger, max mate, current mate, and size
- It also loads the fish image to display

`void check_eat(Animal a, ArrayList<Animal> animals)`

- This method does not perform any function as the fish does not eat any other animal

`Animal create_child()`

- This method creates a new fish

class Penguin extends Animal

- This class is a child class of Animal, it adds to the methods, making specifications for the penguin animal

`public Penguin(int max_x, int max_y)`

- The constructor defines the specific values for each variable according to the animal penguin
- It defines width, height, move distance, hunger rate, max full, max hunger, current hunger, max mate, current mate, and size
- It also loads the penguin image to display

`void check_eat(Animal a, ArrayList<Animal> animals)`

- This method checks if a penguin interacted with a fish
- If it did, the fish is removed from the game (penguin ate the fish)

`Animal create_child()`

- This method creates a new penguin

class Seal extends Animal

- This class is a child class of Animal, it adds to the methods, making specifications for the seal animal

`public Seal(int max_x, int max_y)`

- The constructor defines the specific values for each variable according to the animal seal
- It defines width, height, move distance, hunger rate, max full, max hunger, current hunger, max mate, current mate, and size
- It also loads the seal image to display

`void check_eat(Animal a, ArrayList<Animal> animals)`

- This method checks if a seal interacted with a fish
- If it did, the fish is removed from the game (seal ate the fish)

`Animal create_child()`

- This method creates a new penguin

class Whale extends Animal

- This class is a child class of Animal, it adds to the methods, making specifications for the whale animal

`public Whale(int max_x, int max_y)`

- The constructor defines the specific values for each variable according to the animal whale
- It defines width, height, move distance, hunger rate, max full, max hunger, current hunger, max mate, current mate, and size
- It also loads the whale image to display

`void check_eat(Animal a, ArrayList<Animal> animals)`

- This method checks if a whale interacted with a seal or penguin
- If it did, the seal or penguin is removed from the game (whale ate the seal or penguin)

`Animal create_child()`

- This method creates a new whale

class Iceburg extends Entity

- This class defines the size of the iceberg and the coordinates in which the iceberg is located

```
public Iceburg(int x, int y)
```

- This constructor sets the location of the iceberg and loads the image

Future Suggestions

Some future suggestion for improvement include:

- Having the iceberg melt to make the game more realistic
- Having the whale and fish unable to go on top of the iceberg
- Adding sound
- Adding more animals