Developer’s Guide

Contents

1. Overview

This program uses a square-based ray casting 3D engine built by us to display graphics. Please keep in mind that this is not a full 3D engine, nor is it a 3D ray tracing engine, due to the high level of complication involved with those two.

2. Sections for each feature of the program or class

a. Structure

The program is structured so that making dinosaurs and merging logic and graphics is as easy as possible, each dinosaur has its own class for ease of recognition, enums were implemented to make it easy to figure out what is happening without needing to memorize numbers and their representations. The GUI and logic were developed separately and are part of different classes to make editing easier and adhering to the single responsibility principle of object oriented programming.

b. Dinosaurs

Each dinosaur is premade with specific statistics and there are some random elements to it to make the game more dynamic, such as age, below is a table with the dinosaurs and their values:

Dino	Size	Speed	Range	Reproduction Chance	Carnivore
Ankylosaurus	5	1	10	20%	No
Apatosaurus	16	1	10	10%	No
Gallimimus	1	2	15	40%	No
Stegosaurus	5	1	10	20%	No
Triceratops	5	1	10	20%	No
Velociraptor	1	2	15	20%	Yes
TRex	11	1	10	10%	Yes
IRex	50	1	15	0.01%	Yes

These values are the original values assigned to the dinosaurs, some may be modified later in the program to balance the game. Age also plays a role as a modifier to these stats, with babies receiving a 0.5x multiplier, juveniles receiving 1.5x, adults receiving 2x, and elders receiving 1x, these multipliers do not affect speed which determines how far a dino can move per turn.

c. Movement and Interactions

Movement in the game is governed by the needs of each dino, if their food value is under 80%, they will actively seek out food, otherwise they will move around randomly. If 2 dinos are close to each other and are the same species, reproduction may occur. If 2 dinos are near each other and one of them is carnivorous, an interaction will occur which will determine whether one or other dino survives the encounter, if the non-carnivorous dino dies and the carnivorous survives, another interaction will trigger which will fill up the food meters of all surrounding dinos to full.

Something else that is also possible in the game is to call down a meteor strike and to infect a random healthy dino. Casting a meteor uses a centre and effect radius to determine which dinos and life forms survive the blast while infect method picks a random index from the lif arraylist until it finds a healthy dino or all dinos are infected. If one of the parents of a baby dino is infected, the baby dino may also become infected after being born.

d. Divine Intervention

Sometimes, when the number of dinos drop too low or rise too high, divine intervention will be called upon to keep the population in check and at a level that is actually good. If the numbers are too high, meteors will be called down to random places until the dino

population is acceptable. This is done to keep the time it takes to process the array short and sweet. If the population dips too low, or one link in the food chain has become extinct, some dinos will migrate into the region to prevent the other from dying out or populating without checks.

e. GUI Class

The GUI class is the primary class, and is in charge of controlling the Thread on which the game runs, and controlling the regulation of advancements. It is also in charge of initializing the game and screen at the right size in the beginning of the game.

It has the start(), stop(), and run() methods from the Runnable interface, as this application can be used java applet. There is also a method available to change it's size, and render elements onto the screen (used by run).

f. Screen Class

This class is in charge of converting the Render data and putting it on the screen, and resetting the pixels when necessary.

g. Render Class

Render is an integral part of generating graphics, as stores all the display data. Images are converted into Renders so that the 3D content and 2D content can be seamlessly integrated onto a 2D panel. It stores the pixel information in an array (not a 2D to save memory).

h. Render3D Class

This class is arguably the most complex class in this program, as it deals with all the 3D graphics. We decided to build a 3D ray casting engine from scratch without the use of openGL, which compounds to the complexity. The floor() method is what renders all the necessary artifacts, including the sky, floor, sprites, and trees. There are methods available to render the sprites and trees based on their coordinates. Further, there is a distance limiter implemented, which makes objects that are further away appear foggier, which is prevents graphic aliasing and reduces memory impact. There is also a method used to darken colours to display sick dinosaurs.

i. Controller Class and InputHandler class

The Controller class uses the input from the InputHandler class to set the player's coordinates in 3D space. This is accessed by the Render3D class to render the objects appropriately around the player.

j. Level Class

This class is uses the Game information to place sprites and blocks in the appropriate locations. This class is used at the very beginning to create the level, and is used by the Render3D class throughout the game to get the locations of the trees.

k. Launching the Game

There are a few primary classes involved with launching: Launcher, StartGame, RunGame, Configuration, and Options. Launcher displays a friendly greeting screen to the player, with the option to start, change options, view instructions, and exit. If they choose Options, the Options class is initialized. If they choose start, the StartGame class is initialized and they must choose the game specifications before beginning. After this, the RunGame class is initialized and the game begins.

l. Options

The Options class allows the user to change the screen resolution and/ or toggle the automatic population control. This class implements ActionListener, so the primary functionality can be found in the actionPerformed class.

m. Configuration Class

The configuration class uses the config.xml file to load presets from previous game launches. This ensures that the user's settings are saved so they remain consistent between runs. This class can either save or load configurations given a key.

3. Suggestions for improvement

Some suggestions for improvement are to create 3D models of the dinosaurs, make more dinos such as air and water dinosaurs, and make a more dynamic world. Another improvement could be to also model pack behaviour in particular dinos and communicate in more advanced ways, as currently their only forms of communication are reproduction and predator-prey. Further, advanced options could be implemented such as multiple textures. More keyboard shortcuts could be implemented for actions. Resizability can also be improved, as sprite movement becomes odd at various sizes.