222 Project Write-Up

Explanation: My project is about making a concurrent Snake Game.

Rules:

1. My Snake Game is made up of several rules:
2. The goal of the game is for every snake to eat the most amount of food.
3. Snakes independently move around the screen.
4. The snakes are lazy, so they do not like being given too many actions.
5. If a snake moves off the screen it resurfaces in the other side.
6. Snake food is randomly generated on the screen.
7. A snake can either be Human or CPU-controlled.
8. Only one human can play at a time.
9. The game has no time limit.

Implementation:

The game implements the slither-able interface which is used both to track the information of the snakes in the back end while changing the state of the GUI presentation in the GUI classes. For example, there is a snake class and a GUI Snake class which both implement specific actions that snakes would do.

Component Description:

Snake: The snake Component of LES is a Gui snake. It is made of GuiCircles which is a subclass of the Gui Ellipse class. It has a diameter of 10 both in length and height which makes it a circle. The circles are connected together to form a snake. The head and tail are saved and the next and previous pointers are set to establish the connection of the snake. The snake moves around with the travel method which will be explained more in the slides.

As for the simple snake component, it is a class that saves the general information of each snake without interfering with the GUI snake and complicating the code. The

snake saves its length. I wanted to increment the length of the snake as it eats more food but I decided not to, as it might negatively affect the movement of the snakes. It also saves the amount of food that has been eaten to know which snake will win the game without placing too much information in the GUI class. This class also keeps track if this specific snake object is used by a human player or CPU interface. This information will distinguish how the action of the corresponding GUI snake will act.

Score Component: The Score is not implemented as a GUI Component. It is only a class that saves the score of the corresponding snakes and changes the values as the game continues. The announcement of who wins the game is also done in this class.

Food Componenet: This Componenet is in charge of creating food pieces to be displayed for the GUI and soon eaten by a snake. Once the food has been eaten by a snake the food component remakes a new food object and places it on a random place in the game board. Originally the food class was not supposed to do anything, but it later became the sole thing controlling everything concerning snake food.

Board Component:

The Board is just the blueprint for what the Gui game board will be in the GUI. It takes the value of the width and length for the game board initialization.

Player Component:

The player class makes objects of players, each player has a snake/GUI snake and a thread to run the snake on. It helps to have a player class to make all the attributes of a player connected to one class for easy access/reference.

Game Component: This class holds all the above components together. Similar to how an animal cell has a cell wall around it for protection and cohesion, this class will keep all of the other classes together to simplify the implementation of the game without dealing with all the technical operations happening behind the scenes.

Component connection:

Most of the game connects itself through the preparation class.

This class collects all of the major components of the game and places them in one place. It is similar to a Swiss army knife for the game implementation. All of the necessary pieces that could be needed at any time in the game are passed to all components that are vital to the implementation. For example, a preparation object has a method that makes all the necessary components of the game and it is then made part of every part of the game such as the GUi JFrame, Jcomponenet, Gui snake etc.

Concurrency implementation:

The movement of the GUI snakes is done concurrently. Each GUI snake has a thread that does the movements of each of the snakes. Each snake moves in a specific direction until told to do otherwise through keyboard board input or CPU suggestion. Once a change in direction occurs each snake accesses the travel method. At any given time in the game, several snakes need to use the travel method. If they are allowed access without any organization they will make errors in their movement which normally leads to the break up of the game presentation. Hence a lock needs to be used on the method and its helper methods that one snake moves at a time by having a specific amount of access time to the method. Even though this lock organizes the snakes to share their usage of the method one by one, it seems that they are all accessing it at the same time due to the speed of the computer.

The Hardest part:

Disclaimer: This explanation might not be completely accurate as there were several problems. The problems may have merged in my memory due to the large amount of obstacles faced in the project. Please take this explanation as a rough description.

The travel method was my biggest problem. The travel method controls the movement of the snakes and the mechanics of changing their direction. I started by simply trying to get the snakes to move in a horizontal line. It took maybe a week to get this running due to the large number of prerequisites needed to do this, such as a viewable

GUI snake, GUI circles, GUI component etc. This part left me stumped with confusion as the snake simply would not move. Perhaps there was a problem with how I was repainting my snakes. Moreover, the snakes were not functioning with multiple threads yet so it was harder to see a change with the GUI. Thankfully i got them to move, just not the way I wished. Instead of moving in an organized fashion, they spasmed across the screen until all the circles faded. Fortunately, that meant I was progressing. I later realized that my travel method did not have a lock yet so the snakes did their instructions out of order. Thankfully the locks made the game more organized. After getting this part working I tried to see if I could make them turn properly.

Unfortunately, they would not turn like normal snakes. Instead, they moved like flying sticks that simply raised up or descended down without slithering or changing orientation. Moreover, after the snakes left the border of the game only the head returned into the frame. I kept trying to make little changes in the method but it would not work, only leaving me more confused with my code. I started systematically testing the intricacies of the method with print statements to see how the game logic presents itself with the GUI. I also organized the method with a switch statement to handle the four directions the snakes move in. Slowly the method became more simplified. I also created different methods to test the correlation for the snakes. “Correlated” means that all the portions of a snake are lined up either on the x or y plane and are going in the same direction. After getting secondary advice on the method. I later found out my testing or correlation and instructions to change correlated status were not accurate. After fixing these problems over several weeks, the snakes started moving well.

Future Improvements:

Background sound: Adding background sound to the game will make it more entertaining and interesting. The sound of games is a huge part of how players will view the game. It will make it a more presentable and fun game.

Make a Non-Lazy Version: Probably it would be good to make a game of snakes that are not lazy which would be version 2 of the game.

Place winner announcement on Gui: Announcing the winner at the end of the game on the GUI will be a better of ending the game instead of the threads stoping and freezing the screen.

Increase restrictions: Making more rules on the game such as increasing the different types of food and different types of points. Perhaps implementing an acceleration feature for the snakes might be an interesting thing to add.

New Features:

1. Accelerations option.
2. Energy Level bar.
3. Different types of food that give different benefits to their eaters.
4. Penalize snakes that move outside the game frame.
5. Increase the size of snakes that eat food.
6. Snakes change their colour depending on the type of food they eat.
7. Make the game human-multiplayer