29.04.2018

Fatih Mustafa Kurt 2016400294

CMPE 160 Project 2

1. Introduction

In this project, we will implement a game called Snakes! It is a game like the classic game Snake with some important differences and enhancements. Instead of being played by a person, it will be played by computer. Game develops itself until lock itself by crawling snakes on themselves.

2. Hierarchy

Project contains five packages which control the game's different parts. These packages are bodies, game, huntersnake, main and ui.

1.1 Bodies

Bodies package contains the the drawable game objects. Game objects are Snake and Food.

1.1.1 Body

Body is the main class to draw a square inside the grid. It implements the Drawable. Thus, any class which will be drown as square should extend the Body. It has point and color to draw inside the grid.

Body gets x and y for its coordinate and color.

1.1.2 Food

Food is basically extending Body, to be drown as square, without overriding and having no methods.

1.1.3 Snake

Though Snake should be drown into the grid, it does not extend Body directly. Since Snake consists of bodies, it should not extend It. For that purpose, it has a LinkedList property called snakeBody. This property contains bodies the Snake has. Thus it has a sequence of bodies.

It has four ability: reproduce, attack, move and stay. These are the fundamental action for snake. The conditions for these actions in a sequence are:

- It should reproduce if its size is equal to its max size, which is 8.
- it should eat the food if the food is away one square to it.
- it should move to the food through the shortest way if it has free direction and it will not stuck by the movement.
- Otherwise it should stay.

Snake gets array of points to create its body.

1.2 Game

Game package contains game logic components

1.2.1 Action

Action defines four different type for Snake. That Type enum contains:

- Reproduce
- Attack
- Move
- Stay

Action gets Type enum if Type is stay, otherwise it will get Type and Direction. Type defines which type of action it is.

1.2.2 Direction

Direction is an enum. It defines the four different direction for Snake movement.

1.2.3 Drawable

Drawable is an interface. The classes which are drown must be implement this interface.

1.2.4 GridGame

Grid game is an abstract class. It gives us a basic game board without any game logic. Its job is to create the game world, start the timer and redraw the game board. It will draw the drawables after each timer clock. Timer task is to delay processes for a given time.

GridGame gets gridWidth and gridHeight for the world size, gridSquareSize for one square size and frameRate for timer clock.

1.2.5 Point

Point is a basic component of the game package. It will give you a point with x and y. That will be useful for drawing items on the board.

Point gets x and y for its coordinates.

1.3 HunterSnake

HunterSnake contains game itself. It is the game's main part. It will initialize and play game.

1.3.1 HunterSnake

HunterSnake runs the game itself. Since this is a grid game, it extends GridGame. For one timer clock, It loops over every snakes. It will create local information foreach snake and wait for them to decide an action, then it will check if the action is valid. Finally it will run the action. Basically it defines the game logics, checks the rules and runs the actions.

HunterSnake gets the same parameters like GridGame.

1.3.2 LocalInformation

LocalInformation contains information for one square inside the grid. It gives free direction for the point and neighbor bodies. It also defines optimal way going to the food. Another functionality is selecting randomly free directions If exists.

LocalInformation gets food on the map, free directions for the point and neighbor bodies.

1.4 Main

Main will run the program, initialize the application window and start the game with specified config.

1.4.1 Main

Firstly, main detach the game from current thread and run it another thread. It calls create the HunterSnake game and add one snake to it. Then initialize the application Window with the HunterSnake game. Afterwards, game starts until some exception is raised or game stop method is called.

1.5 UI

UI will create the application window and draw the canvas inside it.

1.5.1 ApplicationWindow

ApplicationWindow will create a GUI frame. It sets the frame's boundaries and some options like resizable and exit on close.

ApplicationWindow gets GridPanel to place it inside the frame.

1.5.2 Colors

Colors defines general colors, like snake head, sneak body and food, to set consistency and avoid redundancy.

1.5.3 GridPanel

GridPanel is the GUI part of the GridGame. It will create a BufferedImage and paint it. After each timer clock, this panel will be cleared and repainted. It has a basic drawSquare method to draw with given coordinate and color.

GridPanel gets the quantity of the squares for x and y axis, and gets one square size.

---

3. Game

When game is started, a food will be placed randomly into the game board, then a snake will placed to the given coordinations. Snake will try to find food and eat it. After each eaten food, snake will grow one square size. If it reaches up to eight body, it will divide into two different snake. Game will go like this until all snakes are stuck or there is left no point to place food. We will explore the game logic and its under-hood.

3.1 Colors

Colors Class shows us the colors used in the game. Let's touch on them.

Snake Head: rgb(0, 0, 200) blue
Snake Body: rgb(120, 255, 0) greenish
Food: rgb(255, 0, 133) purple
Canvas: rgb(255, 255, 255) white
Grid: rgb(0, 0, 0) black

3.2 Snake Implementation and AI

3.2.1 Snake Implementation

Snake is a group of bodies. It consists of multiple bodies inside it to be drawn like line. These bodies are stored in a LinkedList object. The last item of list is the head of snake. The movement of snake is a sequential movement. The first item of the list will take the place of its ancestor. And the head will move to the desired direction. If it attacks to food, it will not move but create a new head and place it over food.

Snake will choose one action for one timer clock. This actions are defined inside the Action class. For movement of the snake, it will choose one of the direction inside of the Direction enum.

3.2.2 AI

3.3 UI

UI implementation is based on JFrame and BufferedImage library. It will create an application frame and place BufferedImage canvas into this frame. After each timer clock, the canvas will be redrawn. To draw the bodies, we have a drawables object which consist of bodies. The bodies inside the drawables will be drawn according to their Point.

3.4 Game Config

Game has four basic configurations, grid width, grid height, grid size and frame rate. Grid width and grid height will determine the frame width and height. Grid size is the size of one square inside the frame. Frame rate is the timer frequency which determine the action per second for one snake.

---

4. Result

The most critical point of this project is implementing the Snake AI. Since any possibility can occur in the game, we cannot be sure about AI if it is working as expected. Sometimes it fails and sometimes works perfectly. Also it depends on the food distribution, and it is randomly.

This project showed me GUI capability of the Java. Maybe it is not so efficient but enough to create some basic GUI application.