SWEN20003
Object Oriented Software Development
Workshop 4

Eleanor McMurtry

Semester 2, 2020

Workshop

This week the focus is on learning to use the Bagel library. To get started, download the starter pack on Canvas and unzip it. Open the directory as an IntelliJ project, and run the main method in `BagelTest`. If you get errors, try right-clicking the file `pom.xml` and clicking `Add as Maven project`; this will force IntelliJ to download and set up the appropriate dependencies for Bagel.

- In Bagel, the window is divided into **pixels**; by default, the window is 1024 pixels wide and 768 pixels high.
- Coordinates in Bagel are specified from the **top-left** of the window (unlike in usual mathematics[1], where the origin is the bottom-left). Therefore, by default
  - `(0, 0)` is the top-left
  - `(512, 384)` is the centre
  - (1024, 768) is the bottom-right
- The graphics in Bagel are handled by clearing the screen to blank many times per second, and re-rendering all of the graphics. One **clear-render** step is called a **frame**. The `update` method inherited from `AbstractGame` must be used to render the graphics, otherwise they will be erased at the beginning of the frame. (We will learn more about inheritance this week.)

Questions

1. Study the `BagelTest` class, using the Bagel documentation available at `https://people.eng.unimelb.edu.au/mcmurtrye/bagel-doc` to understand how it works.
   (a) Look at the documentation for the `Image` class to understand how images are loaded and rendered.
   (b) Similarly, look at the documentation for the `Input` class to understand how keyboard and mouse input is handled.
2. Create a simple game following the below instructions.
   (a) When the game starts, our player should be rendered to the screen at the **point**: `(50, 350)`, and our house should be rendered at **point**: `(850, 180)`.
      - The player image is located at `res/player.png`, and the house image is located at `res/house.png`.

   (b) The player should be able to move left, right, up, and down, using the respective arrow keys at a constant speed (in pixels per frame). Try different values for the speed (a **constant**), starting at 1.
   (c) If the player comes within 55 pixels of the center of the house door, print to the console "Welcome home!". The center of the door is located at point: `(854, 268)`. (Remember, the distance between points (x_1, y_1) and (x_2, y_2) is given by $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.)
   (d) Instead of printing the greeting to the console, draw the text to the screen using the **font** provided (`res/conformable.otf`). Draw it in size 24 font at coordinate (32, 32). (See the `Font` class in Bagel documentation.)

[1]You can blame the Microsoft programmers Craig Eisler, Alex St. John, and Eric Engstrom who created DirectX (one of the first widely-used graphics libraries) in 1995 for this. They simply preferred "left-handed" coordinates.

(e) The game should exit when the Escape key is pressed.

3. Here's a more complex game: **Catch the Ball**.

   (a) First, **clone** your workshop repository. Head to `https://gitlab.eng.unimelb.edu.au/swen20003-s2-2020/<us` and follow the steps in last week's lecture slides to clone the repository.

   (b) Copy the contents of `bagel-starter-pack` into the cloned repository. Run `git add .` to add the files, and `git commit -m "copied template"` to make your first commit. Then run `git push` to push your changes; you should commit and push for each of the following steps.

   (c) The player (using the same image as in Question 2) must move around the window catching a ball (`res/ball.png`). When the game starts, the ball should choose a random coordinate on the window to be drawn at.

   (d) When the player is within 24 pixels of the centre of the ball, the ball should move to another random coordinate.

   (e) Keep track of the player's **score**, which starts at 0 and increases by 1 every time the player catches the ball. Draw this score (using `Font`) at the top-left of the window.

   (f) Extend the code to make the ball choose a random diagonal direction, and move in that direction at a speed of 0.2 pixels per frame. When the ball reaches the left and right edges of the window, it should reverse horizontal direction. Similarly, when the ball reaches the top and bottom edges of the window, it should reverse vertical direction.

4. The below code (also on Canvas) contains bugs that stops it from working.

```java
import java.util.Arrays;

public class VideoGame {
    public final String name;
    public final String platform;

    private static final VideoGame[] games = new VideoGame[100];
    private static int numGames = 0;

    public VideoGame(String name, String platform) {
        this.name = name;
        this.platform = platform;

        games[numGames++] = this;
    }

    public int compareTo(VideoGame other) {
        int platformCompare = platform.compareTo(other.platform);
        if (platformCompare != 0) {
            return platformCompare;
        } else {
            return name.compareTo(other.name);
        }
    }

    public static VideoGame[] getSortedVideoGames() {
        // Create a copy of the array to sort
        VideoGame[] sorted = Arrays.copyOf(games, games.length);

        // Recursively sort the array
        quickSort(sorted, 0, numGames - 1);
        return sorted;
    }

    private static void quickSort(VideoGame[] array, int low, int high) {
        if (low < high) {
            // Choose a pivot
            VideoGame pivot = array[high];
            // Partition the array
            int i = low;
            for (int j = low; j < high; ++j) {
                if (array[j].compareTo(pivot) < 0) {
                    // Swap the elements
                    VideoGame temp = array[i];
                    array[j] = array[i];
                    array[j] = temp;
                    ++i;
                }
            }

            // Swap the elements
            VideoGame temp = array[i];
            array[high] = array[i];
            array[i] = temp;

            quickSort(array, low, i - 1);
            quickSort(array, i + 1, high - 1);
        }
    }

    public String toString() {
        return String.format("%s (%s)", name, platform);
    }

    public static void main(String[] args) {
        new VideoGame("Halo", "Xbox");
        new VideoGame("Ocarina of Time", "Nintendo 64");
        new VideoGame("Red Dead Redemption 2", "Xbox One");
        new VideoGame("Majora's Mask", "Nintendo 64");

        System.out.println(Arrays.toString(VideoGame.getSortedVideoGames()));
    }
}
```

Use the **IntelliJ debugger** to help you find the bugs. Click to the right of the line number to set a **breakpoint**, and click the bug icon (next to the play icon) in the top-right; the code will **break** (pause) at the line you selected. Experiment with the features of the debugger view. Particularly helpful will be the **Variables** view and the **Step Over** button.