CS 152 Computer Programming Fundamentals Project 8: Particle Simulation[*]

Brooke Chenoweth

Spring 2024

In this project, you'll create a particle simulation program. The software resembles a paint program, except that the user is painting particles into the world. The software simulates the physical behavior of those particles, which may move (perhaps falling like grains of sand), change, clone, disappear, interact, etc.

1 The `ParticleDisplay` class

The `ParticleDisplay` class will display your simulation and capture mouse input from the user. You will not change this class.

- `public ParticleDisplay(String title, String[] toolNames, int numRows, int numCols)`

  The constructor creates a display object from a given window title, an array of strings to use as names for the tool selection buttons, and the number of rows and columns for the display.

- `public void setColor(Color color, int row, int col)`

  Sets the cell at given (row, col) location to the specified color. Row 0 is at the top, column 0 is on the left side, and coordinates increase going right and down. Does not actually repaint the screen. (see repaintAndPause, below)

  The `java.awt.Color` class has many predefined color constants for you to to choose from. If you aren't satisfied with the existing colors, you can call the constructor, specifying red, green, and blue values ranging from 0 to 255.

  `Color(int red, int green, int blue)`

- `public void repaintAndPause(int milliseconds)`

  Repaints the screen with the current colors.

- `public ParticleDisplay.Location getMouseLocation()`

  Returns an object holding the row and column of the last mouse click location.

[*]This project is based on the project at `http://nifty.stanford.edu/2017/feinberg-falling-sand/`

- `public String getToolString()`

  Gets the currently selected tool string. Tool names were specified in the constructor.

- `public int getSpeed()`

  Get the current speed selected by the slider at the bottom of the window.

2 What you have to do: Edit the `ParticleSimulator` class

To start off, compile and run ParticleSimulator. This will run ParticleSimulator's main method, which constructs a new ParticleSimulator and calls its run method. You should see a window pop up. On the left side is a black rectangular canvas which will soon be inhabited by particles. On the right side there is one button for each tool you will be able to paint with: Empty (for erasing) and Metal (for creating metal particles). You can't actually paint now, because you haven't written the code yet.

Inside the ParticleSimulator.java class, you'll see I have defined an enum type `Material` to represent the different particle types. Initially, it only contains two values, but you will add more as you proceed with the project.

You'll see that a ParticleSimulator object has two member variables:

- grid – a 2-dimensional array of Material values that represent the type of particle found at each location

- display – the ParticleDisplay used to show the particles on the screen

Work through the following exercises to implement the particle simulation.

2.1 Constructor

The ParticleSimulator constructor already initializes the display field to refer to a new ParticleDisplay with appropriate dimensions and tool names. Insert code to initialize the grid field to refer to a 2-dimensional array of the same dimensions filled with empty material. (You won't be able to test this code yet.)

2.2 `updateFromUser`

The updateFromUser method is called (by the run method) after the user clicks on some part of the canvas. The selected tool string (Empty, Metal, etc.) is passed to the method. Store the corresponding Material value in the corresponding position of the grid array. (You won't be able to test this code yet.)

2.3 `refreshDisplay`

The refreshDisplay method is called (by the run method) at regular intervals. Its job is to draw each particle (and empty space) found in grid onto the display, using ParticleDisplay's setColor method. Complete this method so that empty locations are shown in one color (probably black) and metal locations are shown in another color (probably gray).

Test that you can now paint metal particles and erase them.

2.4 Sand

Modify your program so that you can also paint with sand particles (probably in yellow). For now, these particles won't actually move.

2.5 `updateRandomLocation`

The updateRandomLocation method is called (by the run method) at regular intervals. This method should choose a *single* random valid location. (*Do not use a loop.*) If that location contains a sand particle and the location below it is empty, the particle should move down one row. (Metal particles will never move.) This code should only modify the array. Do not set any colors in the display. Test that your sand particles fall now.

Tip: If particles fall too quickly or too slowly, the speed can be adjusted by adjusting the slider in the display or by changing the dimensions passed to the ParticleSimulator constructor (from main).

Note: Because the updateRandomLocation method picks a single random particle to move (or act in some way) each time it is called, it is possible that some sand particles will move several times before others have the chance to move at all. In practice, the updateRandomLocation method is called so rapidly that you are unlikely to notice this effect when you run the code.

2.6 Water

Modify your program so that you can also paint with water particles, which move in one of three randomly chosen directions: down, left, or right.

In the updateRandomLocation method, when the randomly chosen location contains a water particle, pick one of three random directions. If the location in that randomly chosen direction is empty, the water particle moves there. (Look for ways to minimize duplicate code in your updateRandomLocation method. *You will not receive full credit if you copy large amounts of code instead of using helper methods to organize your implementation.*)

Test that the water behaves roughly like a liquid, taking the shape of a container.

2.7 Dropping Sand Into Water

What happens now when you drop sand particles into water? Right now, sand is only allowed to move into empty spaces. Modify your code so that a sand particle can also move into a space containing a water particle (by trading places with the water particle). (Look for ways to minimize duplicate code in your updateRandomLocation method.) Test that you can drop sand into water now (without destroying the water).

2.8 Sliding Sand

In real life, sand doesn't form vertical stacks, but falls into piles. Modify your code to have the sand sliding sideways a little so that it forms a more natural looking pile. It should still fall down into empty space (and water!), but if the particle is sitting on top of something solid, pick randomly between left and right, and if that location and the location beneath it is empty (or water, etc.), swap the sand into it.

So, if the sand is right on the edge of a solid surface, it'll sometimes randomly slip off (and later fall down), but in the middle of something solid, it'll stay put.

2.9 Additional behaviors

Now implement other behaviors. Get creative!

I want to see at least 3 new behaviors beyond the original metal, sand, and water. Document the expected behavior of each of the material types you include in your program so we know what to look for when testing it. (Just add more description to the class comment at the top of the file.) *You will not receive full credit without proper documentation!*

Here are some ideas to get you started.

- Other liquids besides water. Oil that floats on water, acid that destroys metal, etc.
- Chemical reactions. Acid/base interactions, freeze/boil (adding heat/cold?)
- Gasses. Steam, clouds?
- Life. Plants that grow, animals that move and eat, death and decay?

3 Turning in your assignment

Submit your **ParticleSimulator.java** file on Canvas. Do not attach `.class` files or any other files.