Bounded Particle Simulation JavaFX

Project Overview:
The Bounded Particle Simulation JavaFX project aims to create an interactive particle simulation using JavaFX. The simulation includes colored particles bounded within a specified window. Users can interact with the simulation by adding particles, changing particle colors, and influencing particle movement through mouse clicks.

Project Components:

1. **Main Application Class:** `BoundedParticleSimulationJavaFX`
   - Responsible for initializing the JavaFX application and managing the simulation.
   - Attributes:
     - `WIDTH`: Width of the simulation window.
     - `HEIGHT`: Height of the simulation window.
     - `NUM_PARTICLES`: Number of particles initially in the simulation.
     - `particles`: List of `ColoredParticle` objects representing particles in the simulation.
   - Methods:
     - `start(primaryStage: Stage): void`: Initializes the JavaFX application, sets up the scene, and starts the animation timer.
     - `initializeParticles(): void`: Initializes the particles with random positions, colors, and velocities.
     - `handleMouseClicked(event: MouseEvent): void`: Handles mouse clicks to apply forces to particles.
     - `addParticles(): void`: Adds a specified number of particles to the simulation.
     - `changeColors(): void`: Changes the colors of all particles.
2. **Particle Class:** `ColoredParticle`
   - Represents an individual colored particle in the simulation.
   - Attributes:
     - `DIAMETER`: Diameter of the particle.
     - `x`, `y`: Current position of the particle.
     - `vx`, `vy`: Current velocity of the particle.
     - `color`: Color of the particle.
     - `minX`, `minY`, `maxX`, `maxY`: Bounding box for the particle.
   - Methods:

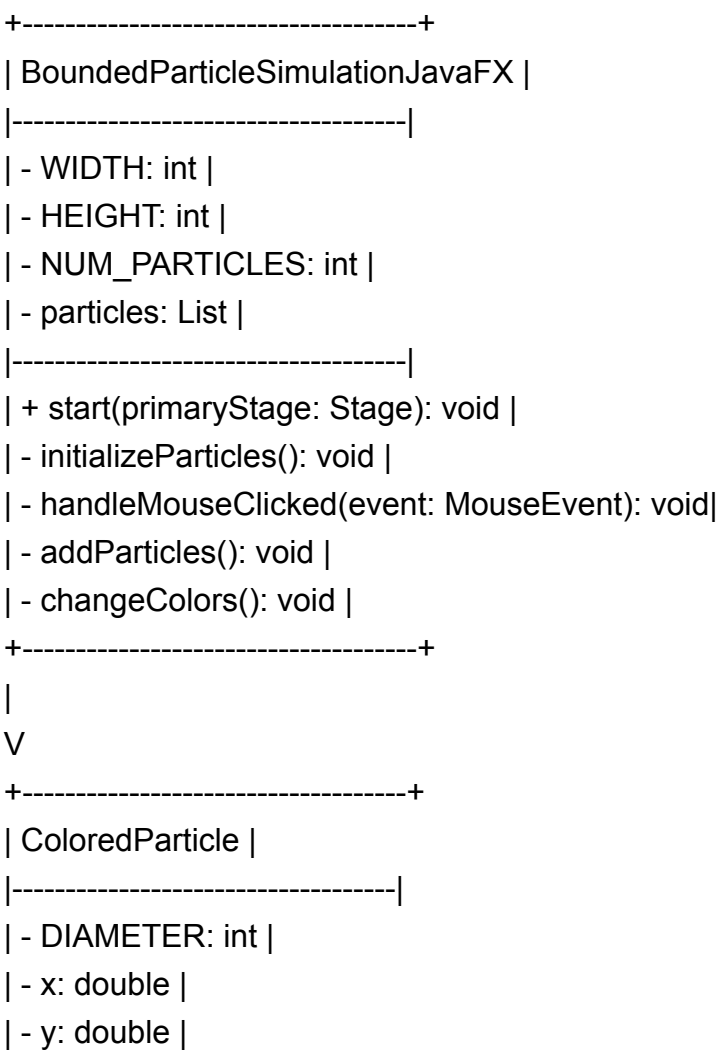
- `ColoredParticle(x, y, color, minX, minY, maxX, maxY): void`: Constructor to initialize particle attributes.
- `update(): void`: Updates the position of the particle based on its velocity.
- `applyForce(forceX, forceY): void`: Applies external forces to the particle.
- `draw(gc: GraphicsContext): void`: Draws the particle on the canvas.

User Interaction:

- Users can interact with the simulation using the following buttons:
  - "Add Particles": Adds a specified number of particles to the simulation.
  - "Change Colors": Changes the colors of all particles.
- Clicking on the simulation canvas applies forces to particles, influencing their movement.

Conclusion:
The Bounded Particle Simulation JavaFX project provides a foundation for an interactive and visually engaging particle simulation. Future enhancements and improvements can further enhance the project's features and user experience.

UML DIagram



| - vx: double |
| - vy: double |
| - color: Color |
| - minX: double |
| - minY: double |
| - maxX: double |
- maxY: double
+ ColoredParticle(x, y, color, minX, minY, maxX, maxY): void
+ update(): void
+ applyForce(forceX, forceY): void
+ draw(gc: GraphicsContext): void
+----------------------------------+

Code

```
package com.example.multicoloredparticlesimulation;

import javafx.animation.AnimationTimer;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

class ColoredParticle {
private static final int DIAMETER = 20;
double x;
double y;
private double vx, vy;
Color color;
private double minX, minY, maxX, maxY;
```

```
    public ColoredParticle(double x, double y, Color color, double minX, double
    minY, double maxX, double maxY) {
        this.x = x;
        this.y = y;
        this.vx = 0;
        this.vy = 0;
        this.color = color;
        this.minX = minX;
        this.minY = minY;
        this.maxX = maxX;
        this.maxY = maxY;
    }

    public void update() {
        x += vx;
        y += vy;

        x = Math.max(minX + DIAMETER / 2, Math.min(x, maxX - DIAMETER / 2));
        y = Math.max(minY + DIAMETER / 2, Math.min(y, maxY - DIAMETER / 2));
    }

    public void applyForce(double forceX, double forceY) {
        vx += forceX;
        vy += forceY;
    }

    public void draw(GraphicsContext gc) {
        gc.setFill(color);
        gc.fillOval(x - DIAMETER / 2, y - DIAMETER / 2, DIAMETER, DIAMETER);
    }
```

}

public class BoundedParticleSimulationJavaFX extends Application {
private static final int WIDTH = 800;
private static final int HEIGHT = 600;
private static final int NUM_PARTICLES = 5;

```
    private List<ColoredParticle> particles = new ArrayList<>();
```

```
@Override
public void start(Stage primaryStage) {
    StackPane root = new StackPane();
    Canvas canvas = new Canvas(WIDTH, HEIGHT);
    root.getChildren().add(canvas);

    Button addParticlesButton = new Button("Add Particles");
    addParticlesButton.setOnAction(e -> addParticles());

    Button changeColorsButton = new Button("Change Colors");
    changeColorsButton.setOnAction(e -> changeColors());

    HBox buttonBox = new HBox(10, addParticlesButton, changeColorsButton);
    buttonBox.setAlignment(Pos.CENTER);

    VBox vbox = new VBox(canvas, buttonBox);
    vbox.setAlignment(Pos.CENTER);

    // Adjust the alignment to bring the buttons slightly up
    vbox.setTranslateY(-10);

    root.getChildren().add(vbox);

    Scene scene = new Scene(root, WIDTH, HEIGHT);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Bounded Particle Simulation");
    primaryStage.show();
    primaryStage.setResizable(false);

    initializeParticles();

    scene.setOnMouseClicked(this::handleMouseClicked);

    GraphicsContext gc = canvas.getGraphicsContext2D();

    new AnimationTimer() {
        @Override
        public void handle(long now) {
            for (ColoredParticle particle : particles) {
                particle.update();
```

```
            }

            gc.clearRect(0, 0, WIDTH, HEIGHT);

            for (ColoredParticle particle : particles) {
                particle.draw(gc);
            }
        }
    }.start();
}

private void initializeParticles() {
    Random random = new Random();

    for (int i = 0; i < NUM_PARTICLES; i++) {
        double x = random.nextDouble() * WIDTH;
        double y = random.nextDouble() * HEIGHT;
        Color color = Color.rgb(random.nextInt(256), random.nextInt(256),
random.nextInt(256));
        particles.add(new ColoredParticle(x, y, color, 0, 0, WIDTH,
HEIGHT));
    }
}

private void handleMouseClicked(MouseEvent event) {
    for (ColoredParticle particle : particles) {
        double forceX = (event.getX() - particle.x) * 0.01;
        double forceY = (event.getY() - particle.y) * 0.01;
        particle.applyForce(forceX, forceY);
    }
}

private void addParticles() {
    Random random = new Random();
    for (int i = 0; i < 5; i++) {
        double x = random.nextDouble() * WIDTH;
        double y = random.nextDouble() * HEIGHT;
        Color color = Color.rgb(random.nextInt(256), random.nextInt(256),
random.nextInt(256));
        particles.add(new ColoredParticle(x, y, color, 0, 0, WIDTH,
```

```
HEIGHT));
    }
}

private void changeColors() {
    Random random = new Random();
    for (ColoredParticle particle : particles) {
        particle.color = Color.rgb(random.nextInt(256), random.nextInt(256),
random.nextInt(256));
    }
}

public static void main(String[] args) {
    launch(args);
}
```

}