CS 451: Computational Intelligence

Spring 2024

Dhanani School of Science and Engineering

Habib University

Assignment 2 - Swarm Intelligence

Muhammad Yousuf Uyghur

1 Particle Systems

Particle systems are dynamic computational models that simulate the behavior of individual particles in virtual environments. These systems, inspired by natural phenomena like smoke, fire, and flocking behavior, are widely used in computer graphics and simulations.

2 Problem Formulation

In this report, we explore a particle system where each particle represents a ball navigating through a maze. The maze is depicted as a 2D grid with empty and blocked cells. Particles move through the grid, aiming to reach the maze's endpoint while avoiding obstacles. Through visualization, we gain insights into how particles interact and adapt to changes in the environment, offering opportunities for experimentation and exploration.

2.1 Algorithm Description

The Particle Systems algorithm can be described as follows:

For Maze Generation

1. The Depth-First Search algorithm is implemented for maze generation using backtracking.
2. Given a current cell as a parameter.
3. Mark the current cell as visited.
4. While the current cell has any unvisited neighbour cells.
5. Choose one of the unvisited neighbours.
6. Remove the wall between the current cell and the chosen cell.
7. Invoke the routine recursively for the chosen cell.
8. Which is invoked once for any initial cell in the area.

For Particle Systems

1. Initialize the environment, including the canvas or grid where the particles will be simulated.
2. Create the particles with initial positions, velocities, and other properties.
3. Update the position of each particle based on its velocity and any external forces acting on it.
4. Handle collisions with obstacles or boundaries.
5. Continuously update the simulation in a loop, allowing particles to move, interact, and be visualized.
6. Handle any termination conditions or stopping criteria.
7. Termination conditions in this case is if a particle reaches the last cell of the maze.
8. Observing the impact of parameter changes on particle behavior is essential for understanding the system's dynamics.

2.2 Implementation

To implement the Particle Systems and Maze Generation algorithm, I used JavaScript and its library P5js, which provides a simple and intuitive environment for visualizing the particles behavior. The code snippet below shows the main components of the implementation:

```javascript
// CONSTANTS
const CANVAS_WIDTH = 1000;
const CANVAS_HEIGHT = 800;
const PARTICLE_MIN = 300;
const PARTICLE_MAX = 10000;
const PARTICLE_STEP = 50;

let noOfParticles;
let speed;

class Particle {
    constructor() {
        // Initialize particle position and velocity
    }

    show() {
        // Display the particles on the canvas
    }

    move() {
        // Adds velocity to position
    }
    handleCollision() {
        // Collision detection with the walls
    }
}

function setup() {
    // Initialize the canvas and create particles
}

function draw() {
    // Update and display particles
}

function generateMaze() {
    current = grid[0];
    stack = [];
    while (true) {
        current.visited = true;
        var next = current.checkNeighbours();
        if (next) {
            next.visited = true;
            stack.push(current);
            removeWall(current, next);
            current = next;
        }
        else if (stack.length > 0) {
            current = stack.pop();
        }
        else {
            break;
        }
    }
}
```

Listing 1: Particle Systems and Maze Generation Implementation

3 Simulation

The visualization of the Particle Systems is crucial for understanding and analyzing particle behavior. In our simulation, each particle is represented as a small entity on the canvas, typically depicted as a shape like a circle or a point. The position of each particle corresponds to its location within the simulated environment.

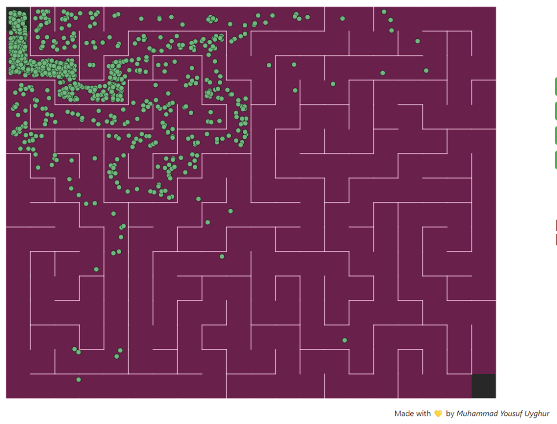
3.1 Simulation Control

Implementing controls for the simulation allows users to interact with and manipulate the environment. These controls typically include options for starting, pausing, resetting, or stopping the simulation. Providing feedback to the user regarding the simulation's status, elapsed time, and frame rate enhances and facilitates understanding.

3.2 Parameter Adjustment

One of the advantages of the simulation is the ability to adjust different parameters and observe their impact on the algorithm's execution and behavior. In our simulation, we allow the user to adjust the number of particles in the swarm (`noOfParticles`) using a slider. This parameter affects the exploration and exploitation capabilities of the swarm. Furthermore, we provide a speed control slider (`speed`) that allows the user to adjust the particle's speed, enabling fast or slow the swarm's movement and interactions.

3.3 Demonstration



Particles: 1000
Speed: 6

FPS: 45
ELAPSED TIME: 00:00:10

Made with 💛 by *Muhammad Yousuf Uyghur*

Figure 1: Visualization of Particle Systems

3.4 Visualization Link

To visualize the particle system in action, you can visit the following link:
`https://alfen-yu.github.io/ParticleSwarmMazeSolver/`

3.5 GitHub Repository

The source code for the particle system simulation is available on GitHub at the following repository:
`https://github.com/alfen-yu/ParticleSwarmMazeSolver`

4 Future Plans

In the current implementation, the collision detection mechanism exhibits imperfections, resulting in particles leaking from corners or edges of walls. Despite efforts to address this issue, it persists as a challenge. Future plans include dedicating more resources to refining the collision detection algorithm to achieve more accurate and robust results. Additionally, I plan to integrate Particle Swarm Optimization (PSO) with the existing particle systems framework. By leveraging PSO algorithms within the context of the maze environment, I aim to evaluate its efficacy in optimizing particle movements and navigating through the maze. This comparative study will shed light on the performance differences between the current particle systems approach and PSO-enhanced strategies, providing valuable insights.

5 Conclusion

In summary, exploring particle systems through this project has been insightful. By simulating particles navigating a maze, I've grasped the basics of their behavior in dynamic environments. While my focus was on basic particle movement and interaction, there's room for improvement. I aim to enhance collision detection and potentially integrate advanced optimization algorithms like PSO in future iterations. In conclusion, this project showcases the versatility of particle systems.

6 References

- p5.js. (n.d.). p5.js — home. Retrieved from `https://p5js.org/`
- Wikipedia contributors. (2022, January 7). Maze generation algorithm. In Wikipedia, The Free Encyclopedia from `https://en.wikipedia.org/wiki/Maze_generation_algorithm#Randomized_depth-first_search`