Sand Simulation with Fan Influence in a Three.js Environment

Introduction

This paper explores a sand simulation algorithm developed using Three.js, a popular JavaScript library for creating 3D graphics in the web browser. The simulation dynamically updates the positions of sand particles based on gravity and the influence of a fan. The primary focus is on the technique used for updating the sand dynamics and integrating the fan's effect.

Graphics Technique

Sand Representation

The sand is represented using a 3D array, where each element indicates the presence (1) or absence (0) of a sand particle. This array serves as a discrete model of the sandbox. The array is defined as:
`sandArray}[x][z][y]`
Here, x, y, and z represent the three dimensions of the sandbox. Each coordinate in the array corresponds to a position in the 3D space of the simulation.

Gravity Simulation

Gravity is simulated by iterating through the sand array from the bottom to the top, ensuring that sand particles fall downwards if there is empty space below them. The algorithm considers both vertical and diagonal movements to simulate a more natural falling pattern. The gravity directions are defined as:

```
[1, 0, -1], & [-1, 0, -1],
[0, 1, -1], & [0, -1, -1],
[1, 1, -1], & [1, -1, -1],
[-1, 1, -1], & [-1, -1, -1]
```

For each sand particle, the algorithm checks if it can move downwards or diagonally downwards. If a valid move is found, the particle's position is updated in the array.

Fan Influence

The fan can blow sand particles in a specified direction (front, right, back, left). The fan's influence is simulated by applying a force in the horizontal direction, both directly and diagonally. The fan directions are defined as:

```
[0, 1, 0],
[-1, 0, 0],
[0, -1, 0],
[1, 0, 0]
```

The diagonal fan directions are:

```
[0, 1, -1],
[-1, 0, -1],
[0, -1, -1],
[1, 0, -1]
```

When the fan is turned on, the algorithm checks if a sand particle can move in the fan's direction. If a valid move is found, the particle's position is updated.

Mathematical Operations

The position update is calculated by adding the direction vector to the current position. This is expressed as:
new_position = current_position + direction
Bounds checking ensures that the new position is within the limits of the sandbox:
$0 \leq new\_position < dimensions\_size$
Random direction selection for gravity is performed using:
`randomDirection=gravityDirections[Math.floor(Math.random()*gravityDirections.length)]`

Sand Dynamics Algorithm

The `updateSandDynamics` function is responsible for simulating both gravity and the fan's influence on the sand particles. The algorithm iterates through the sand array, applying gravity to make the sand particles fall and the fan's influence to blow them in a specified direction.

The algorithm starts by initializing a boolean `changes` to track if any particle has moved. It defines the movement directions for gravity and fan influence.

A helper function, `applyMovements(x, y, z)`, is used to check for valid movements based on the fan's influence. If no movement due to the fan is detected, it checks for valid gravity movements. The function updates the positions of the particles in the sand array and sets `changes` to true if any movement occurs.

The algorithm then loops through the sand array based on the fan's position. Depending on the fan's index, the loop order is adjusted to apply the movements correctly. For each particle, the `applyMovements` function is called.

If any changes occurred, the `updateMesh` function is called to reflect the new positions of the particles in the mesh.

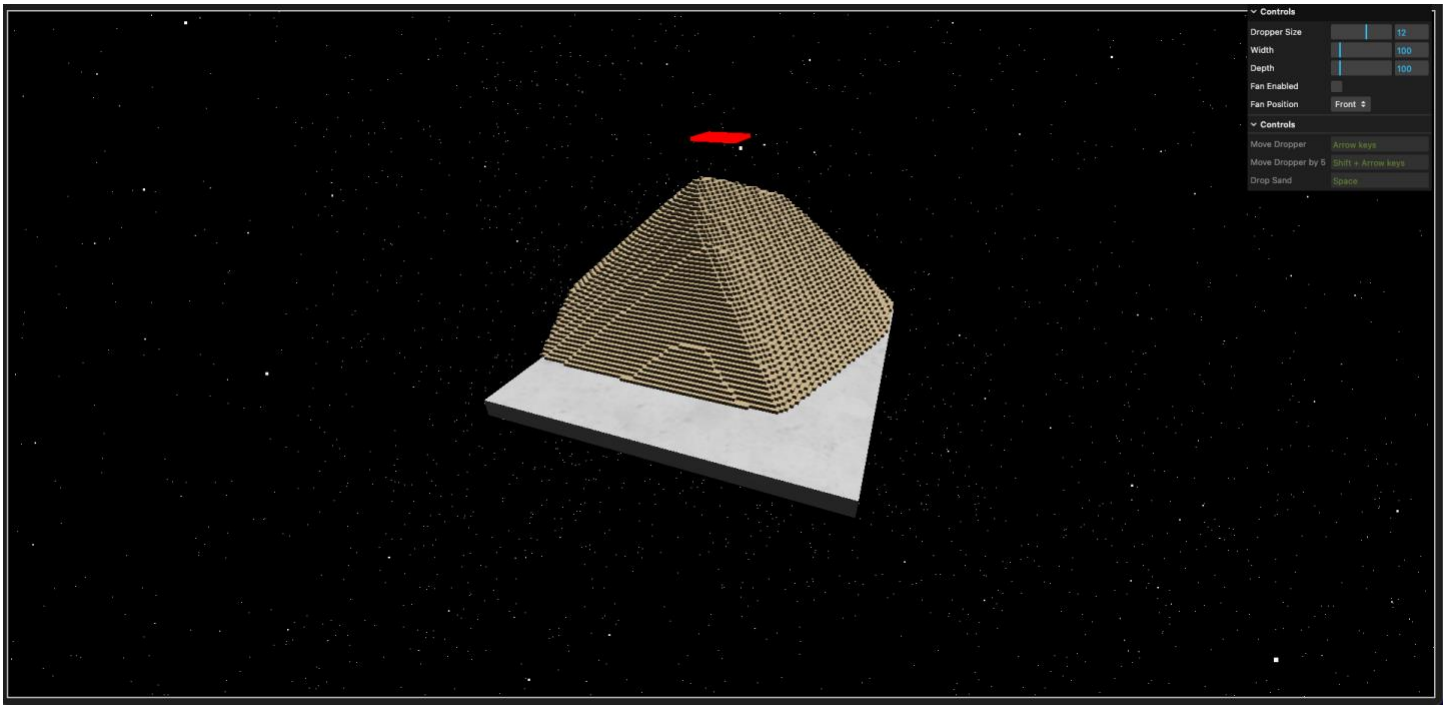
Future Improvements

To enhance the realism of the sand simulation, several improvements can be made:

1. **Improved Sand Pileup Algorithm**: This would account for the angle of slopes, sand temperature, and the speed at which the sand is falling. Implementing such factors can simulate more realistic sand behavior, such as how sand piles up at different angles based on its temperature and velocity.
2. **Improved Wind Algorithm**: This would consider wind speed and direction to determine which sand particles are affected. By simulating wind dynamics more accurately, the effect of the fan can be made more realistic, with varying degrees of influence on the sand particles based on their position and the wind's strength and direction.
3. **Larger Scale**: Expanding the scale of the simulation can enhance realism by allowing more particles to interact and form complex structures. This can create more intricate patterns and behaviors as the sand responds to gravity and fan influence over a larger area.

Conclusion

This sand simulation algorithm effectively demonstrates the use of Three.js for dynamic 3D graphics in the web browser. By integrating gravity and fan influence, it creates a simulation of sand behavior. Future enhancements can further improve the realism and complexity of the simulation, providing a richer and more immersive experience.


Controls
Dropper Size
12
Width
100
Depth
100
Fan Enabled
Fan Position
Front
Controls
Move Dropper
Arrow keys
Move Dropper by 5
Shift + Arrow keys
Drop Sand
Space