Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

A Project Proposal

on

"Water Fountain Simulation"

[Code No: COMP 342]

Submitted by

Arun Bhandari (07)

Samir Wagle (60)

Submitted to:

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date: 15/06/2024

Table of Contents

List Of Figures

Chapter 1 Introduction

1.1 Project Description

Computer graphics is a fascinating field that involves the creation, manipulation, and representation of visual content using computational techniques. This mini-project is designed to delve into the core principles of computer graphics by simulating a water fountain. The simulation is implemented using Python, leveraging the Pygame library, which is a cross-platform set of Python modules designed for writing video games.

Pygame is built on top of the Simple DirectMedia Layer (SDL) library, providing functionalities for rendering graphics, handling input, and managing game events. In this project, Pygame is used to create and animate particles that simulate water droplets in a fountain. The simulation involves generating particles, applying physics to mimic gravity, and rendering the visual elements to create a dynamic and realistic water fountain effect.

The primary components of the simulation include:

1. **Particle System**: Particles are generated at a specific rate, each with random initial velocities. These particles represent individual water droplets.

2. **Physics Simulation**: Each particle's motion is influenced by gravity, causing them to follow a parabolic trajectory. The particles are also re-initialized when they move out of the screen bounds to create a continuous flow.

3. **Graphical Rendering**: Pygame's drawing functions are utilized to render the particles and the fountain structure, including the cylinder and triangular top from which the particles emanate.

This project provides an interactive simulation where users can adjust parameters such as particle density, spread, and gravity to observe different behaviors of the water fountain. By exploring these concepts, the project highlights the intersection of physics and graphics programming, demonstrating how computational techniques can be used to create engaging and realistic visual simulations. This report will cover the design, implementation, and results of the simulation, discussing the challenges encountered and the learning outcomes derived from this hands-on experience in computer graphics

1.2 Objectives

- To Simulate a Dynamic Water Fountain Using Pygame
- To Explore and Apply Graphics Programming Concepts
- To Demonstrate the Integration of Physics and Graphics

1.3 Expected Outcomes

- Accurate portrayal of particle movement with gravity and wind effects.
- Adjustable settings for particle density, spread, gravity, and wind.
- Real-time display of fountain structure and particle movement.
- Intuitive command-line interface for easy parameter adjustment.
- Demonstrates physics principles such as gravity and projectile motion.

Chapter 2 Library and Language used

Here are the list of libraries and programming languages used for particle physics simulation.

2.1 Python

Python served as the backbone of the project providing a versatile and powerful programming environment. It's simplicity and extensive library support facilitated the development of entire simulation framework, from particle generation and physics calculations to graphical rendering and user interaction.

2.2 OpenGL

OpenGL stands as a robust graphics library renowned for its prowess in rendering both 2D and 3D graphics, making it ideal for crafting interactive and visually captivating animations such as simulating particle physics in a water fountain. Leveraging OpenGL's capabilities empowers developers to achieve smooth animations, realistic rendering of water droplets and splashes, and interactive controls, thereby enhancing the overall user experience of the water fountain simulation. Integrating OpenGL into the project entails a solid grasp of computer graphics principles and OpenGL programming, making it a suitable choice for creating dynamic and immersive visualizations of the water fountain's behavior.

2.3 PyGame

Pygame managed the user interface and interaction, offering a seamless and interactive experience. It provided functions for window management, event handling, and basic drawing operations. Pygame's ease of use allowed for the smooth integration of user inputs, such as adjusting particle density, spread, and gravity, enhancing the overall interactivity of the simulation.

2.4 Math

The Math library was crucial for performing the necessary calculations to simulate realistic particle motion. It provided functions for vector arithmetic, trigonometry, and kinematic equations, which were essential for updating particle positions and velocities based on initial conditions and gravity. This mathematical foundation ensured that the particles followed accurate trajectories, contributing to the realism of the water fountain effect.

Chapter 3 Methodology

The Fourier Series Waveforms Animation project aims to visually represent the complex concept of the Fourier series through interactive animation. The methodology involves the following steps:

3.1 Design and Planning Phase

The design and planning phase began with defining the primary objectives of the project, which included simulating realistic water fountain dynamics and enabling user customization. Key features such as particle physics, rendering, and user input handling were identified. thorough requirement analysis was conducted to determine the necessary software and hardware, leading to the selection of Pygame for its capabilities in graphical rendering and user interaction management. The system architecture was designed to include the particle system, rendering engine, and user interface components.

3.2 UI Design Phase

The user interaction design phase focused on developing a command-line interface that allowed users to configure various simulation parameters such as particle

density, spread, gravity, and wind speed. Additional customization options were provided, including background color changes and modifications to the fountain structure. User inputs were validated to ensure they were within acceptable ranges, and any invalid inputs were handled gracefully. This phase ensured that the simulation was not only functional but also user-friendly and customizable.

3.3 Testing and Debugging Phase

Driving the simulation's overarching structure is the Fountain class, designed to manage the fountain's behavior and interactions. Upon instantiation, the Fountain class is initialized with critical parameters such as max_particles, particles_per_frame, spread, gravity, and wind. These parameters govern fundamental aspects of particle generation, physics simulation, and visual presentation within the fountain.

An integral method within the Fountain class is update, responsible for orchestrating particle generation and updating each particle's state over time (dtime). This method dynamically generates new particles based on the particles_per_frame parameter and updates existing particles by invoking their individual update methods. Additionally, the render method within the Fountain class oversees the visualization of the fountain's structural components—such as the base, cylinder, and triangular top—alongside all active particles. By leveraging Pygame's drawing functions, the render method ensures the comprehensive depiction of the fountain's dynamic elements on the simulation screen.

3.4 Documentation Phase

Documentation was created to ensure clarity and maintainability of the code. Code documentation included detailed comments and explanations for key functions and classes. A comprehensive user guide was developed, providing instructions on configuring and running the simulation, along with examples of different configurations and their effects. This phase ensured that users could easily

understand and interact with the simulation, contributing to a positive user experience.

Chapter 4 Implementation Details

The implementation details for the program are given below:

4.1 Particle Class

Central to simulating the water fountain's dynamic behavior is the Particle class. This class serves as the blueprint for individual particles within the fountain, encapsulating essential attributes and behaviors. Each particle is initialized with specific properties including its position (x and y coordinates), velocity (v_x and v_y components), and color (color). These attributes define the particle's initial state within the simulation

The Particle class features an update method responsible for computing the particle's movement over time. This method integrates physics principles such as gravity (gravity) and wind effects (wind), adjusting the particle's position based on its velocity and environmental factors. Additionally, the draw method utilizes Pygame's graphical capabilities to visually represent each particle on the simulation screen. By rendering small circles at the particle's coordinates using Pygame's circle drawing function, the draw method brings the particles to life within the fountain environment.

4.2 Fountain Class

Driving the simulation's overarching structure is the Fountain class, designed to manage the fountain's behavior and interactions. Upon instantiation, the Fountain class is initialized with critical parameters such as max_particles, particles_per_frame, spread, gravity, and wind. These parameters govern fundamental aspects of particle generation, physics simulation, and visual presentation within the fountain.

An integral method within the Fountain class is update, responsible for orchestrating particle generation and updating each particle's state over time (dtime). This method dynamically generates new particles based on the particles_per_frame parameter and updates existing particles by invoking their individual update methods. Additionally, the render method within the Fountain class oversees the visualization of the fountain's structural components—such as the base, cylinder, and triangular top—alongside all active particles. By leveraging Pygame's drawing functions, the render method ensures the comprehensive depiction of the fountain's dynamic elements on the simulation screen.

4.3 Simulation Control

Pygame serves as the foundation for initializing and managing the simulation's graphical interface. The make_renderer function is instrumental in initializing Pygame, configuring the display window (screen), and assigning a descriptive title to the simulation window. This function sets the stage for rendering the fountain simulation within a user-friendly graphical environment.

Maintaining a consistent frame rate (FPS) is crucial for preserving the simulation's smooth animation. The limit_frame_rate function is implemented to regulate the rendering speed, ensuring that each frame is displayed at the specified FPS. By calculating elapsed time between frames and incorporating appropriate delays, this function optimizes the simulation's performance and visual coherence.

User interaction within the simulation is managed by the handle_events function. This function monitors user inputs and responds to events triggered by interactions with the simulation window. Notably, it captures events such as closing the simulation window, enabling smooth user interaction and intuitive control over the simulation's execution.

4.4 User Interaction:

To enhance user engagement and customization, the simulation provides interactive functionalities accessible through dedicated functions. The prompt_user function prompts users to input specific parameters—such as particle density, spread, gravity, and wind speed—via a command-line interface. This interactive approach empowers users to tailor the simulation's characteristics according to their preferences, fostering a personalized and immersive experience.

The main_loop function orchestrates the simulation's core animation loop, orchestrating the sequential execution of particle updates, rendering fountain components, and handling user events. This function encapsulates the simulation's operational logic, ensuring continuous execution until the user opts to exit the simulation environment.

4.5 Execution and Main Functionality:

The seamless execution of the water fountain simulation is orchestrated by the main function, serving as the entry point for initiating and managing the simulation's lifecycle. Within this function, comprehensive instructions are displayed to guide users through configuring simulation parameters and exploring additional features. By integrating user prompts, interactive functionalities, and core simulation mechanics, the main function ensures a cohesive and engaging experience for users interacting with the water fountain simulation.

Chapter 5 Usage Instructions

The following usage instructions provide a comprehensive guide to set up, configure, and run the Water Fountain Simulation project. This section ensures that users can effectively interact with the simulation and customize it according to their preferences.

5.1 Requirements

- Python 3.x
- Pygame library

5.2 Setup

- Install Python:
- Install the Pygame library using the following command: pip install pygame

5.3 Running the Simulation

Download the Project Files. Ensure you have all the project files in a single directory.
Start the Simulation
After running the script, you will be presented with a command-line interface to configure the simulation. Follow the prompts to enter the desired parameters:

Particle Density: Enter a value between 1 and 10000 to set the particle density.
Particle Spread: Enter a value between 1 and 20 to set the spread of particles.
Gravity: Enter a value between 0 and 20 (the value will be scaled by 10 for a realistic effect).
Wind Speed: Enter a value between 0 and 10 to set the wind effect.

5.4 Additional Features

After setting the main parameters, you can choose to change the background color or modify the fountain's structure:

Background Color: Sky Blue, Light Green, Light Yellow, Light Pink

The Pygame window will open, displaying the water fountain simulation with the configured parameters.

Chapter 6 Conclusion and Recommendation

The simulation of a water fountain developed in this mini-project successfully integrates fundamental principles of physics, mathematics, and computer graphics to create a dynamic and visually engaging experience. By leveraging the Pygame library, we were able to implement a realistic particle system that mimics the behavior of water droplets under the influence of gravity and projectile motion. Key functionalities, such as particle initialization, motion updates, and real-time rendering, were effectively utilized to achieve smooth and continuous animation. This project not only demonstrates the practical application of theoretical concepts but also highlights the potential of combining physics and graphics programming to create interactive simulations. Through this hands-on experience, valuable insights were gained into the complexities and nuances of simulating natural phenomena, providing a solid foundation for further exploration and development in the field of computer graphics.

6.1 Limitations

While the water fountain simulation effectively demonstrates key principles of physics and computer graphics, it also has several limitations. One significant limitation is the lack of air resistance, which in reality would affect the particles' trajectories, potentially leading to a less realistic simulation. Additionally, the simulation does not account for particle collisions or interactions, which would occur in an actual water fountain as droplets collide and merge. The simplistic representation of particles as circles and the basic rendering techniques used may not capture the full aesthetic appeal of a real fountain. Furthermore, the computational performance may degrade with higher particle densities or more complex simulations, limiting the scalability of the project. Lastly, user interactions are limited to adjusting initial parameters without the ability to dynamically alter conditions during the simulation, restricting the overall interactivity and user experience.

Addressing these limitations could lead to more accurate and engaging simulations in future iterations of the project.

6.2 Future Enhancement

The future enhancements of the project are:

- Integrate fluid dynamics algorithms for realistic water behavior.
- Upgrade to 3D rendering for an immersive experience.
- Allow real-time adjustment of simulation parameters.
- Allow customization of particle colors and shapes.
- Provide multiple fountain designs to choose from or customize.
- Add sound effects for an immersive experience.

References

About Pygame. (n.d.). Retrieved 6 15, 2024, from https://www.pygame.org/wiki/about

Pygame. (n.d.). Retrieved from Pygame Front Page: https://www.pygame.org/docs/

pygame documentation: Tutorials. (n.d.). Retrieved 6 15, 2024, from http://www.pygame.org/docs/

Shinners, P. (n.d.). *Python Pygame Introduction - History*. Retrieved 6 15, 2024, from http://www.pygame.org/docs/tut/PygameIntro.html

APPENDIX

```
PS C:\Users\L E G I O N\Desktop\New folder> & "C:/Program Files/Python312/python.exe" "c:/Users/L E G I O N/Desktop/New folder/completed2.py"
pygame 2.5.2 (SDL 2.28.3, Python 3.12.3)
Hello from the pygame community. https://www.pygame.org/contribute.html

Instructions:
1. Use the prompt to configure the simulation.
2. Enter the desired particle density, spread, gravity, and wind speed.
3. Watch the fountain simulation in the Pygame window.
4. Close the window to exit the simulation.

Additional Features:
1. Change the background color
2. Modify the fountain's structure

Change Background Color:
1. Sky Blue
2. Light Green
3. Light Yellow
4. Light Pink
Enter your choice: 1
Welcome to the Water Fountain Simulation!

1. Start simulation
2. Exit
Enter your choice: 1
Enter particle density (1-10000): 2000
Enter particle spread (1-20): 15
Enter gravity (0-20): 10
Enter wind speed (0-10): 7
```

Figure 1 User Interface Page

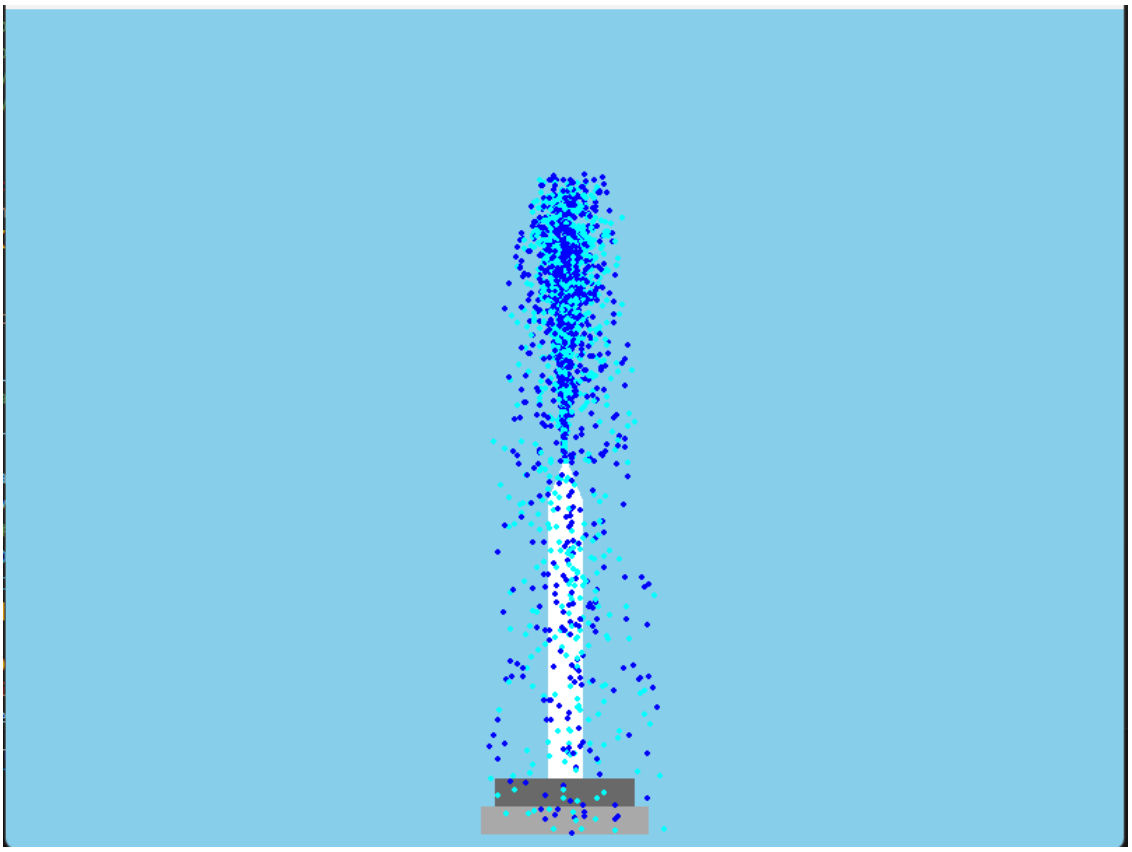

Figure 2 Basic Water Fountain Simulation

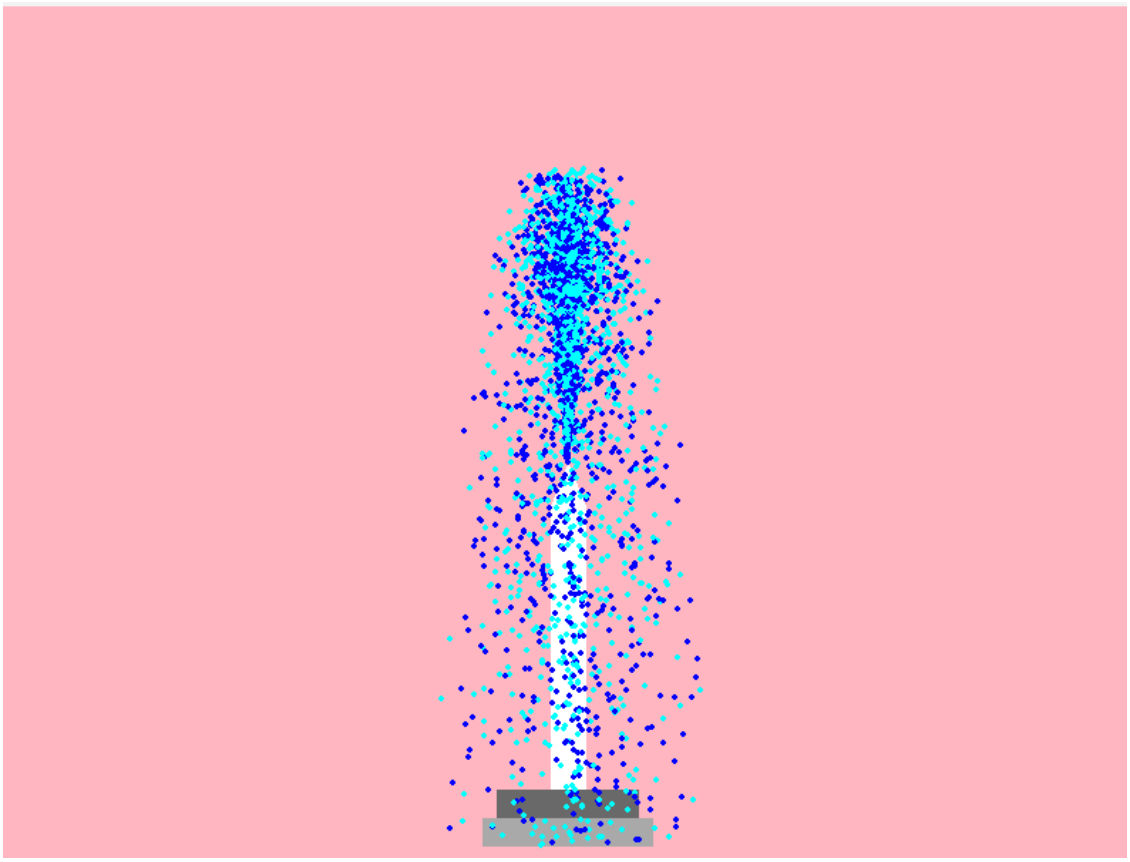

Figure 3Water Fountain Simulation with Background Color chosen Light Pink with particle density 500, Particle Spread 20, Gravity value 9 (Reference:9.8=Earth's gravity) and wind speed 1.

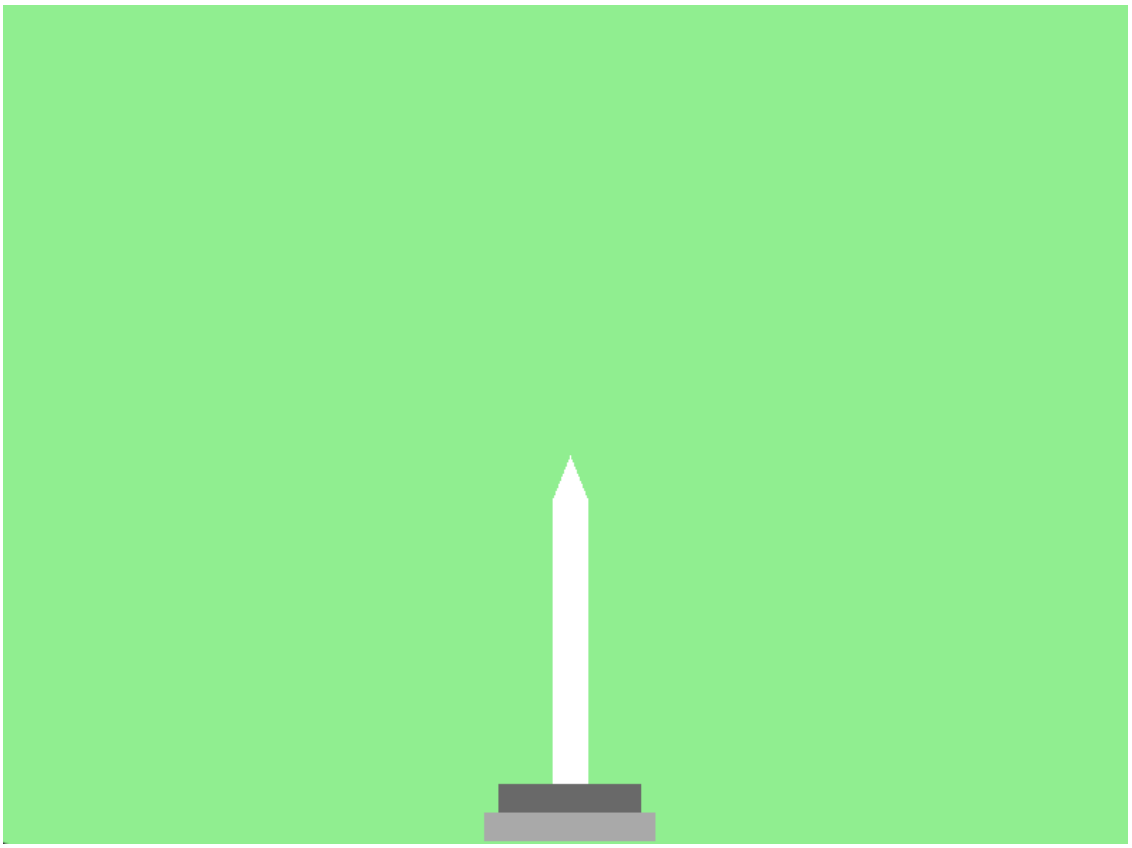

Figure 4 Water fountain structure with no water flow.

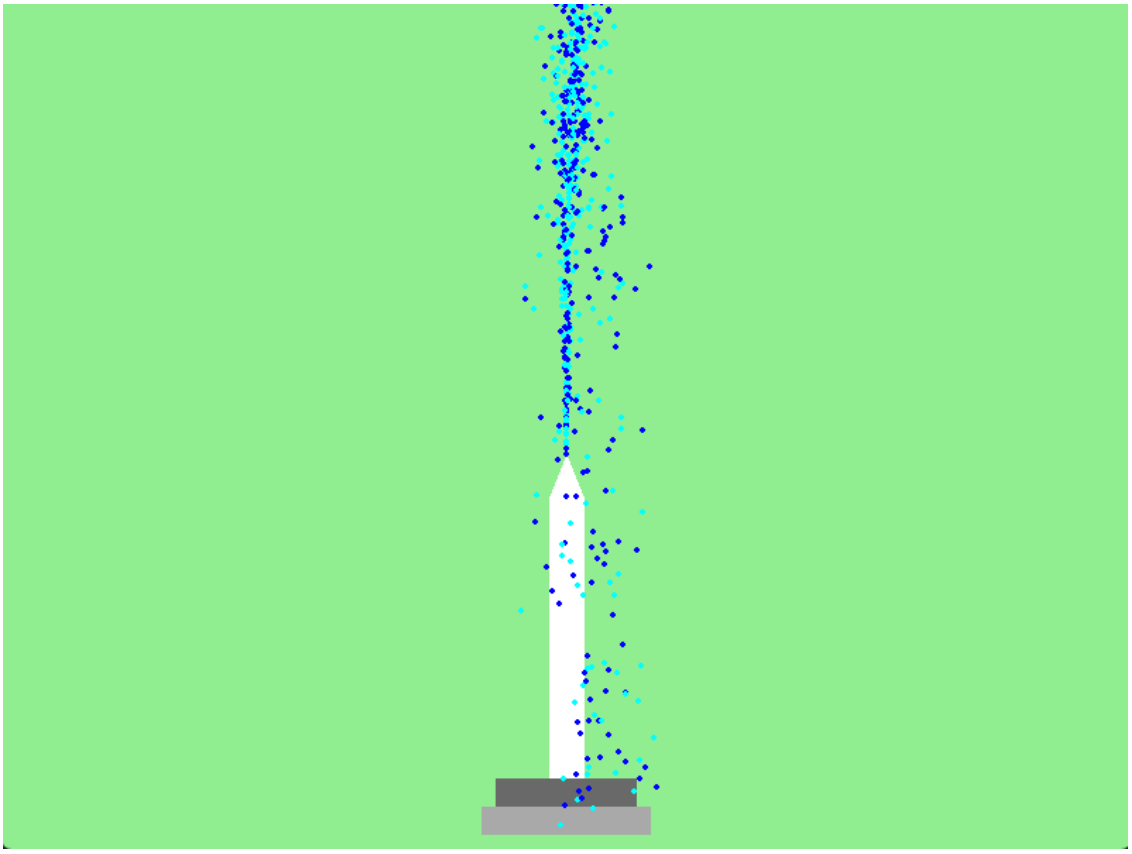

Figure 5:Water Fountain Simulation with Background Color chosen Green with particle density 100, Particle Spread 7, Gravity value 5 (Reference:9.8=Earth's gravity) and wind speed 2.

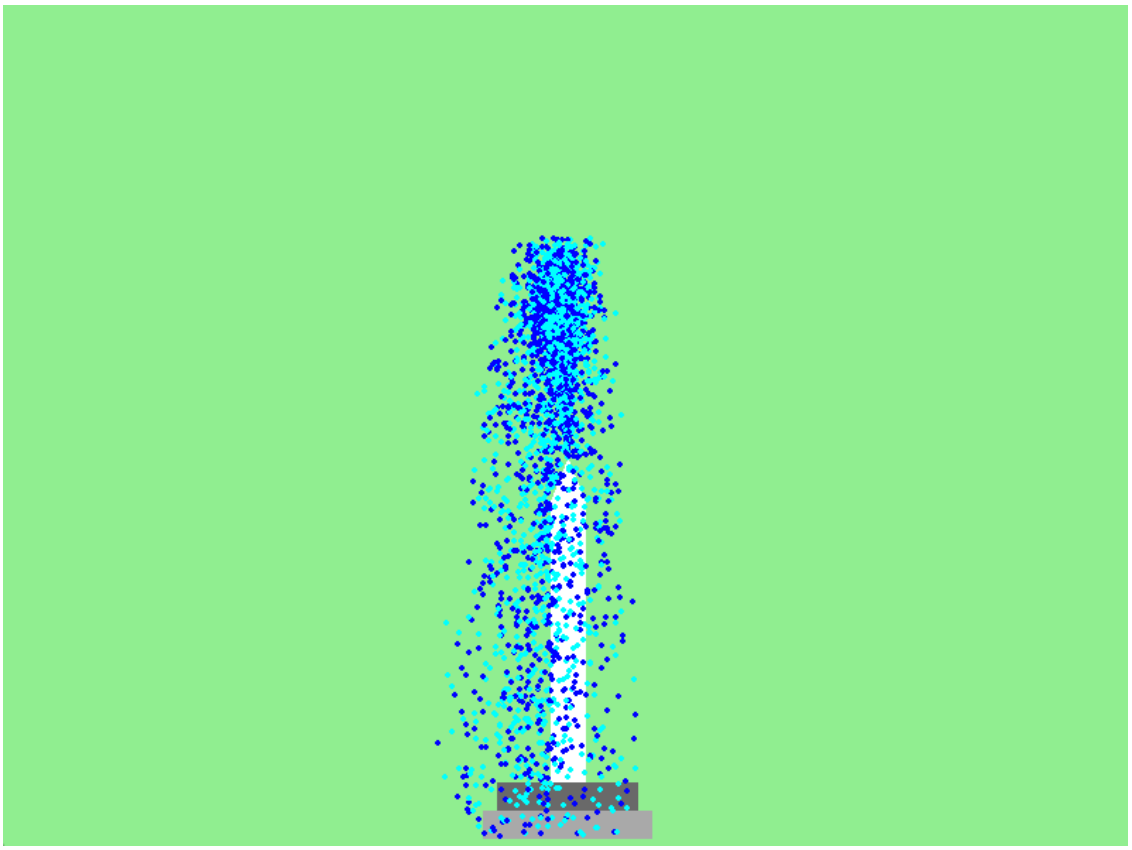

Figure 6:Water Fountain Simulation with Background Color chosen Green with particle density 700, Particle Spread 15, Gravity value 13 (Reference:9.8=Earth's gravity) and wind speed -5.

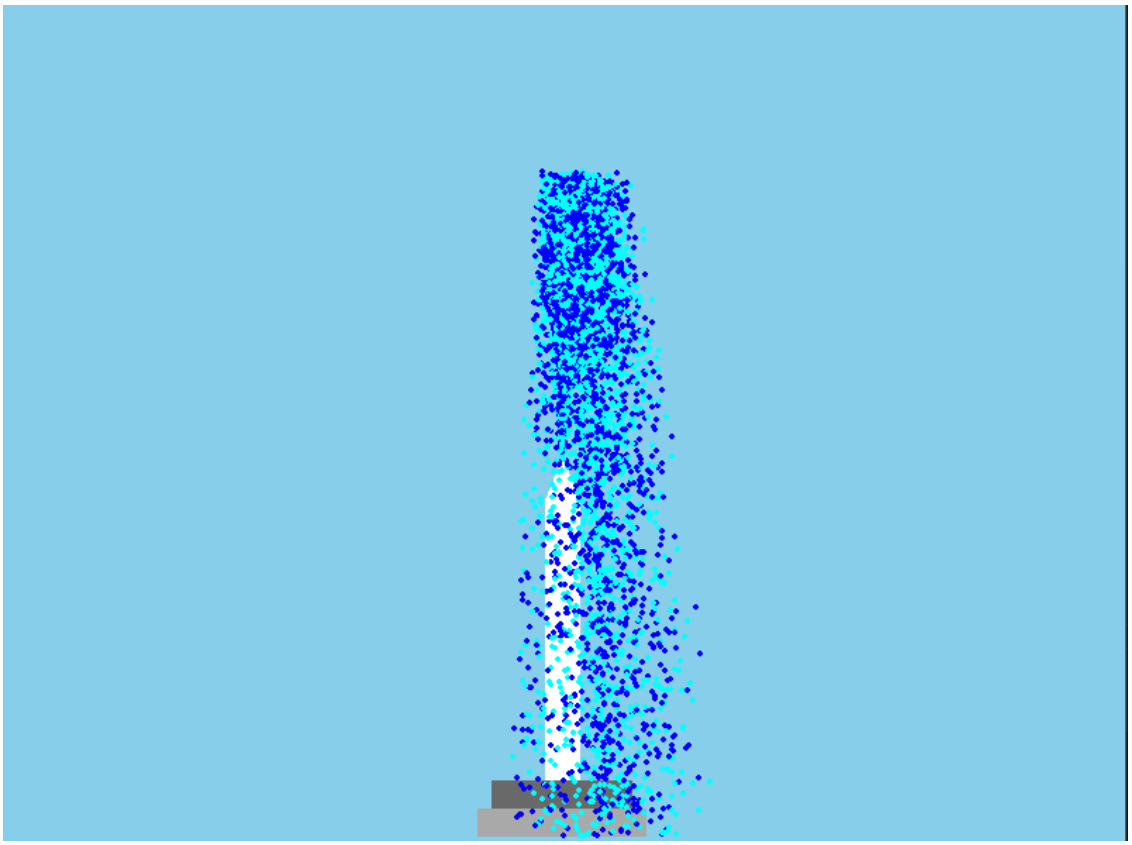

Figure 7: Water Fountain Simulation with Background Color chosen Light Blue with particle density 400, Particle Spread 20, Gravity value 10 (Reference:9.8=Earth's gravity) and wind speed 7.

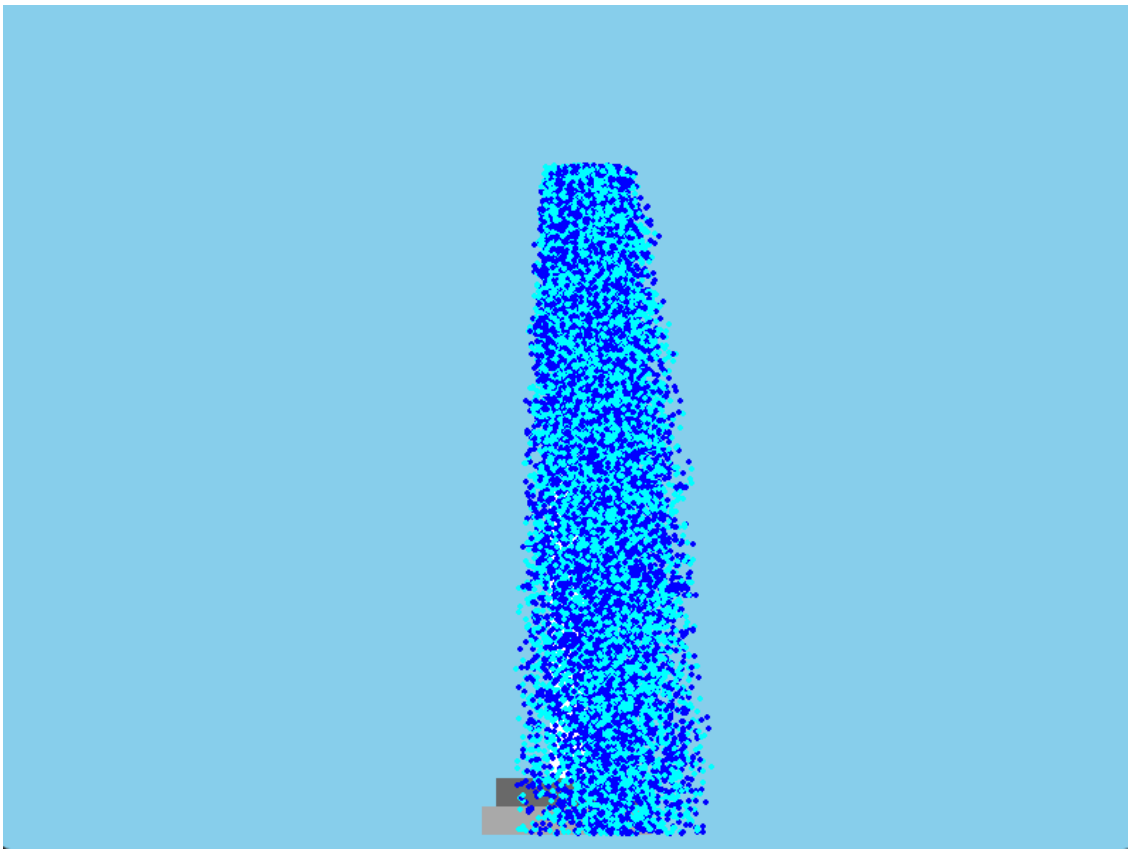

Figure 8:Water Fountain Simulation with Background Color chosen Light Blue with particle density 2000, Particle Spread 20, Gravity value 10 (Reference:9.8=Earth's gravity) and wind speed 7.

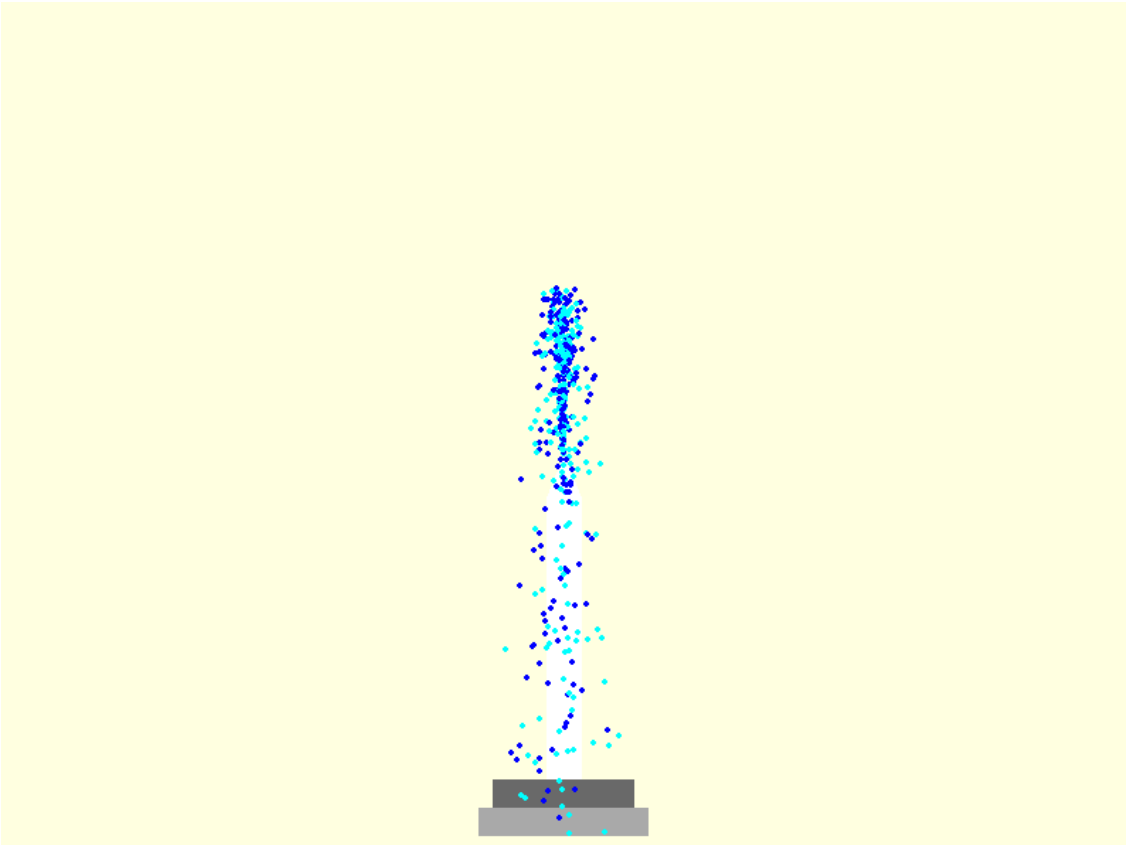

Figure 9:Water Fountain Simulation with Background Color chosen Light Yellow with particle density 100, Particle Spread 15, Gravity value 17 (Reference:9.8=Earth's gravity) and wind speed 0.

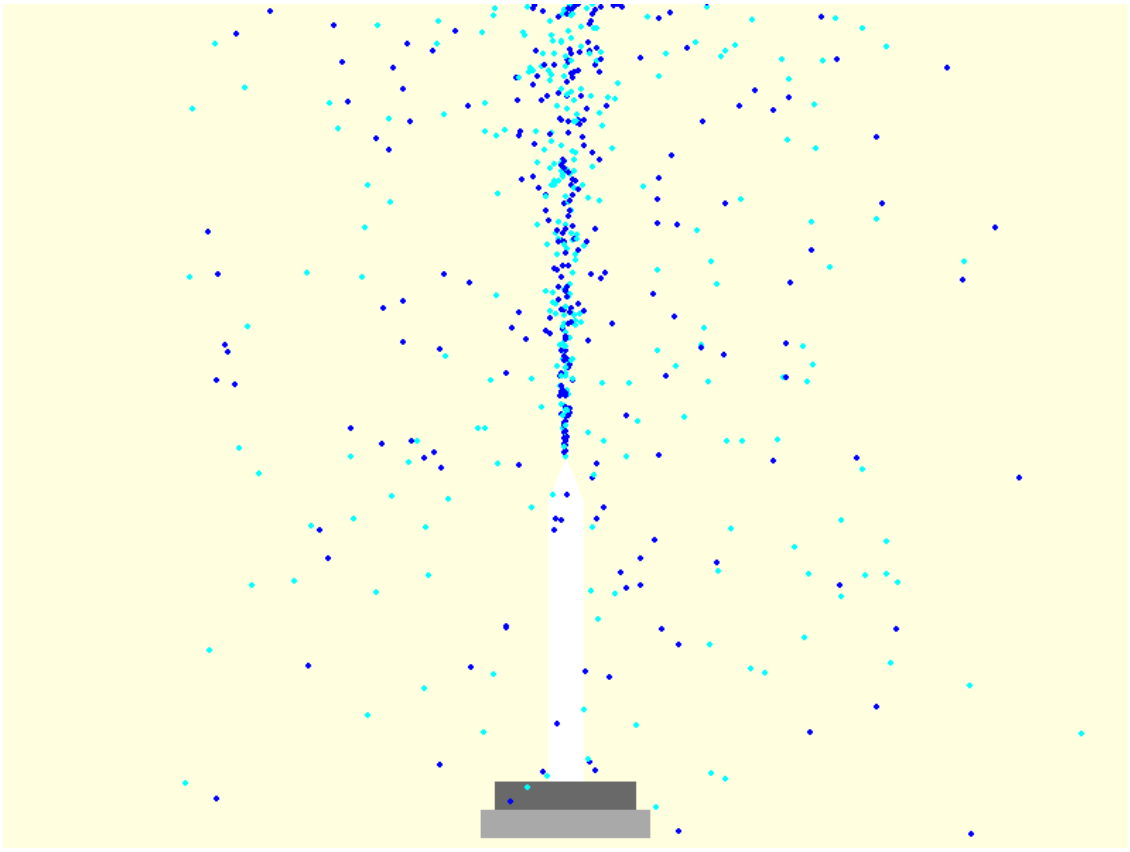

Figure 10: Water Fountain Simulation with Background Color chosen Light Yellow with particle density 200, Particle Spread 15, Gravity value 2 (Reference:9.8=Earth's gravity) and wind speed 0.

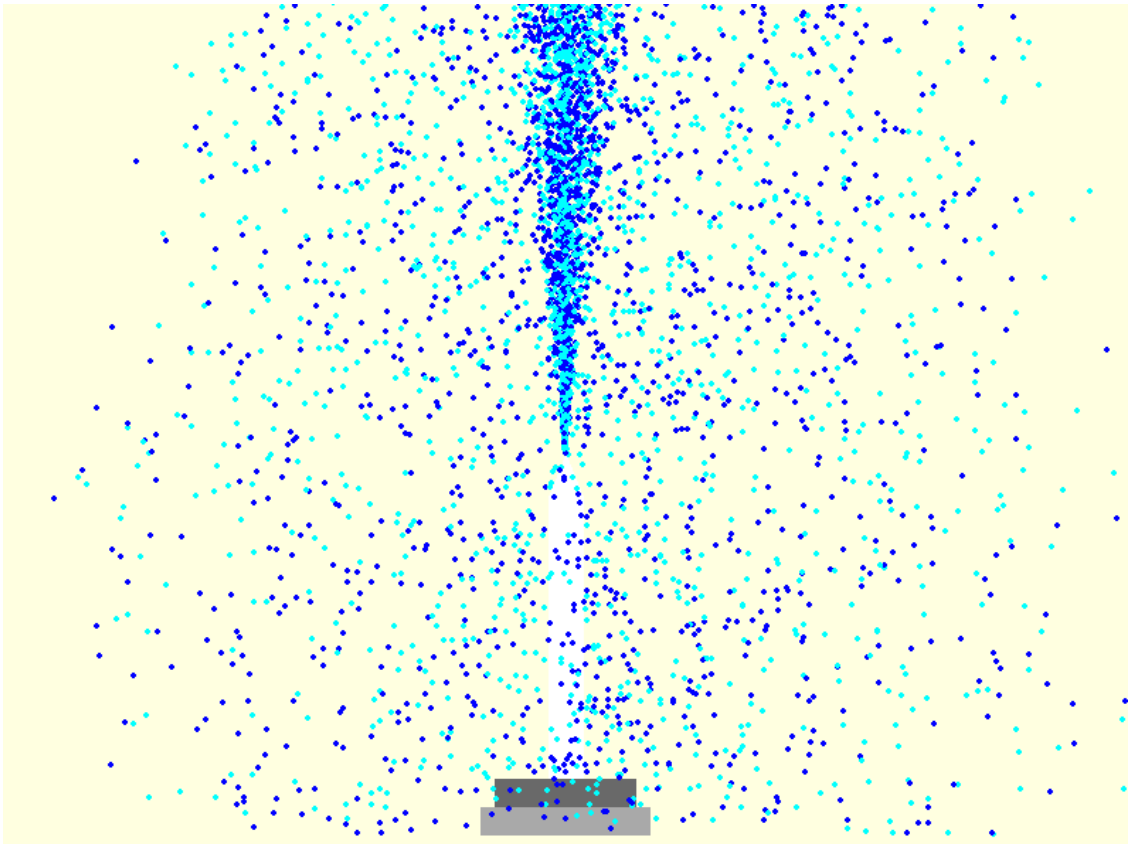

Figure 11:Outcome of above simulation after 2 mi

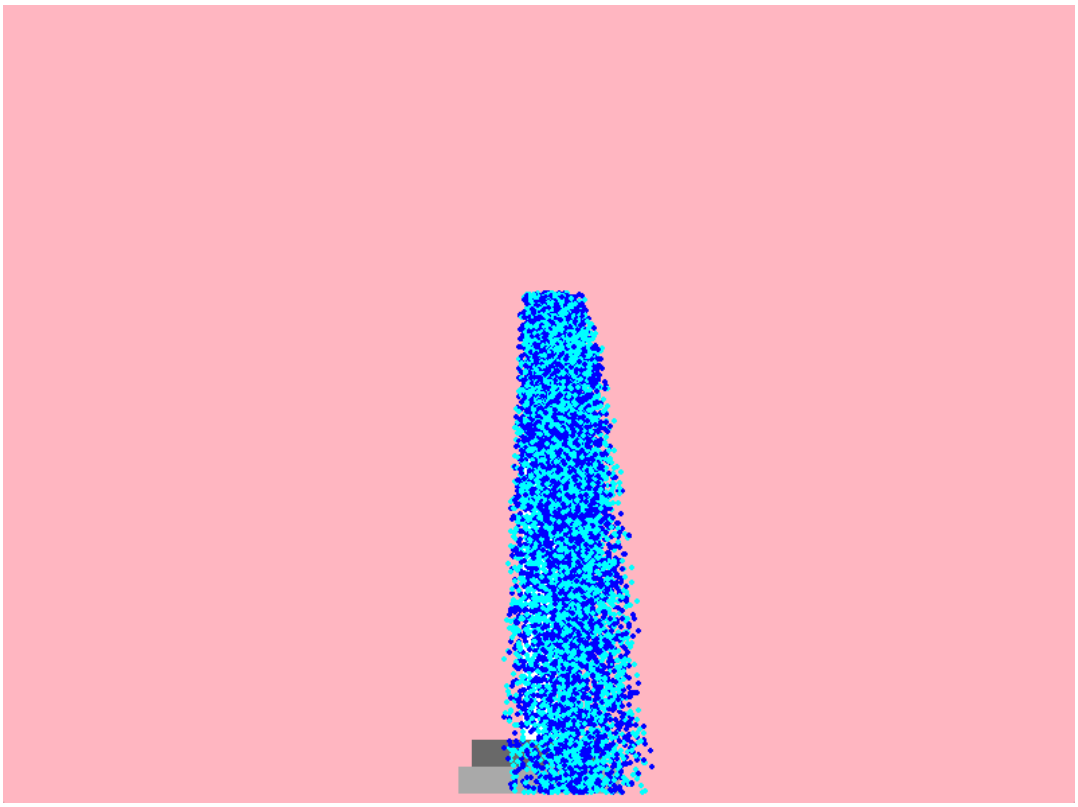

Figure 12: Simulation with 8000 particles, 20 gravity, 20 spread and wind speed 10