2D Sand Physics Engine

Team Members

Xander Ede

Objective / Problem Statement

The goal of this project is to create a visually interactive 2D sand simulation where particles behave according to simple physics rules. The simulation will allow users to drop, draw, and interact with materials and watch them settle and interact under gravity.

The objective is to explore how physics-based objects can be implemented efficiently in C++, while also providing a real-time graphical visualization.

Motivation

I enjoy creating simulations of the real world that can lead to chaotic behavior. I think that a similar like this has potential to expand my knowledge of particle simulations and has the option to expand into a bigger project.

Key Features

- **Particle Simulation:** Simulates thousands of particles with simple physical rules.
- **Gravity and Collision:** Sand particles fall due to gravity and settle naturally on other materials.
- **User Interaction:** Click and drag to place or erase particles.
- **Real-Time Visualization:** Render the simulation in a 2D window.
- **Pause/Resume Controls:** Option to pause and step through frames for debugging and observation.
- *(Optional extension 1)*: Saving/loading simulation states.
- *(Optional extension 2)*: Add multiple materials to the simulation

Tools / Libraries

- C++17 or later
- OpenGl or Vulcan for rendering
- VSCode
- Github

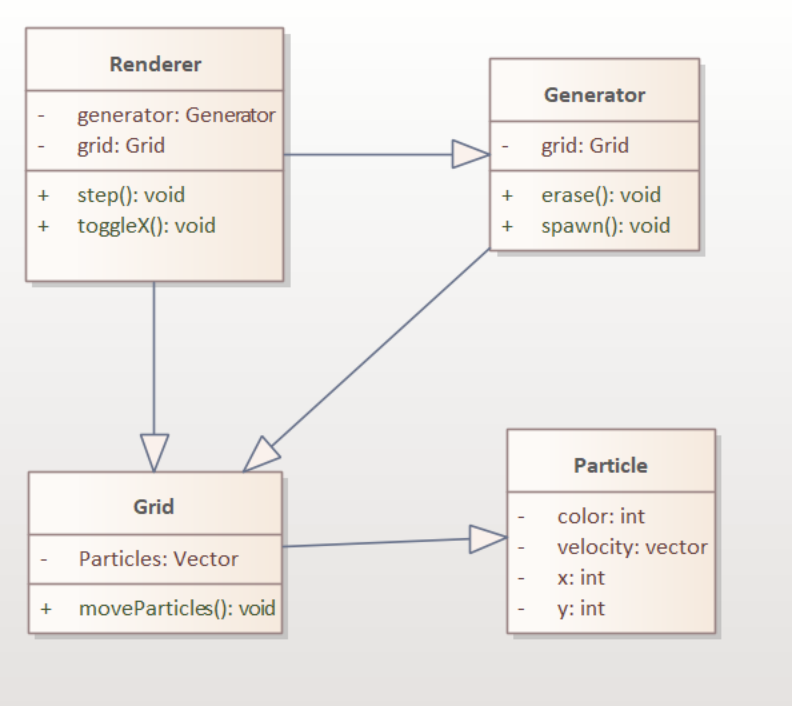
Timeline – Major Milestones

Week	Milestone
Week 6–7	Set up project, create window, basic rendering loop
Week 8	Implement grid and sand particle representation
Week 9	Add basic gravity and movement rules for sand
Week 10-11	Add user interaction (placing and erasing particles)
Week 12	Optimize performance, add polish and optional features
Week 13	Testing and documentation

Potential Challenges or Questions

- **Performance:** Ensuring the simulation runs smoothly when handling thousands of particles.
- **Boundary Conditions:** Preventing particles from clipping through edges or each other.
- **Scalability:** Deciding how large the grid can be without sacrificing performance.
- **Visual Clarity:** Balancing realism with clear visual representation of particles.

UML


Renderer
- generator: Generator
- grid: Grid
+ step(): void
+ toggleX(): void
Generator
- grid: Grid
+ erase(): void
+ spawn(): void
Grid
- Particles: Vector
+ moveParticles(): void
Particle
- color: int
- velocity: vector
- x: int
- y: int