3D Particle Simulation Project

This project is a 3D particle simulation that lets us see how different types of particles interact with each other in a 3D space. It's built using Rust with a graphics library for rendering.

The simulation creates a 3D world where particles can attract/repel each other based on their types. It manages thousands of particles with different properties like position, velocity, and type. The Border file uses GPU acceleration through WebGPU (wgpu for sharder) to render particles as spheres.

There is an interface with sliders and controls to adjust:

- Number of particles
- Simulation boundary size
- Interaction strength and radius
- Friction
- Attraction/repulsion values between different particle types
- Particle changeable colors
- Gravity effects

The simulation creates many particles in random positions where each particle has a color and type. The program calculates forces between nearby particles and they move based on these forces. Each particle interacts with nearby particles according to type-specific attraction values. The positive values cause attraction, negative values cause repulsion and particles speeds up toward/away from each other based on distance and interaction strength based on the particles. The friction slows particles over time.

Instead of iterating through all particles in a nested loop, it regulates grid that will store particles efficiently. The particle data is transferred to the GPU for parallel processing.

There is user interface side panel with sliders and controls for adjusting simulation parameters which changes the movement of particles. The performance metrics shows the frame rate and update times and window is for adjusting attraction values between particle types. The frames per second reflects the use of spatial partitioning and parallel execution. The parallel execution should improved the time taken per frame, force calculations, and reduced the processing time. Spatial hash table is used to divide the 3D space into a grid of cells and keeping track of which particles are in each cell. For each particle, it looks at all nearby particles, calculates the force from each neighbor based on how far/near it is and their particle type. Then it adds up all these forces, and updates the particle's velocity based on the total force and adds gravity effects.