InternPro Weekly Progress Update

Name	Email	Project Name	NDA/ Non-NDA	InternPro Start Date	OPT
Adharsh Prasad Natesan	anatesan@asu.edu	IT-Core Foundation Suriname	Non-NDA	2024-08-05	Yes

Progress

Include an itemized list of the tasks you completed this week.

#	Action Item/ Explanation	Total Time This Week (hours)
1	Double layered particle collision simulation	3
2	Optimizing Particle Simulation Computation	3
3	Enhancing Particle Simulation with Increased Density and Layers	3
4	Soil Simulation with Particles in Python	3
5	Simulating Friction Between Soil and Tools	3
6	Efficient Multi-Particle Handling with GPU Acceleration	3
7	Next week plan	1
8	Report writing	1
	Total hours for the week:	20

Verification Documentation:

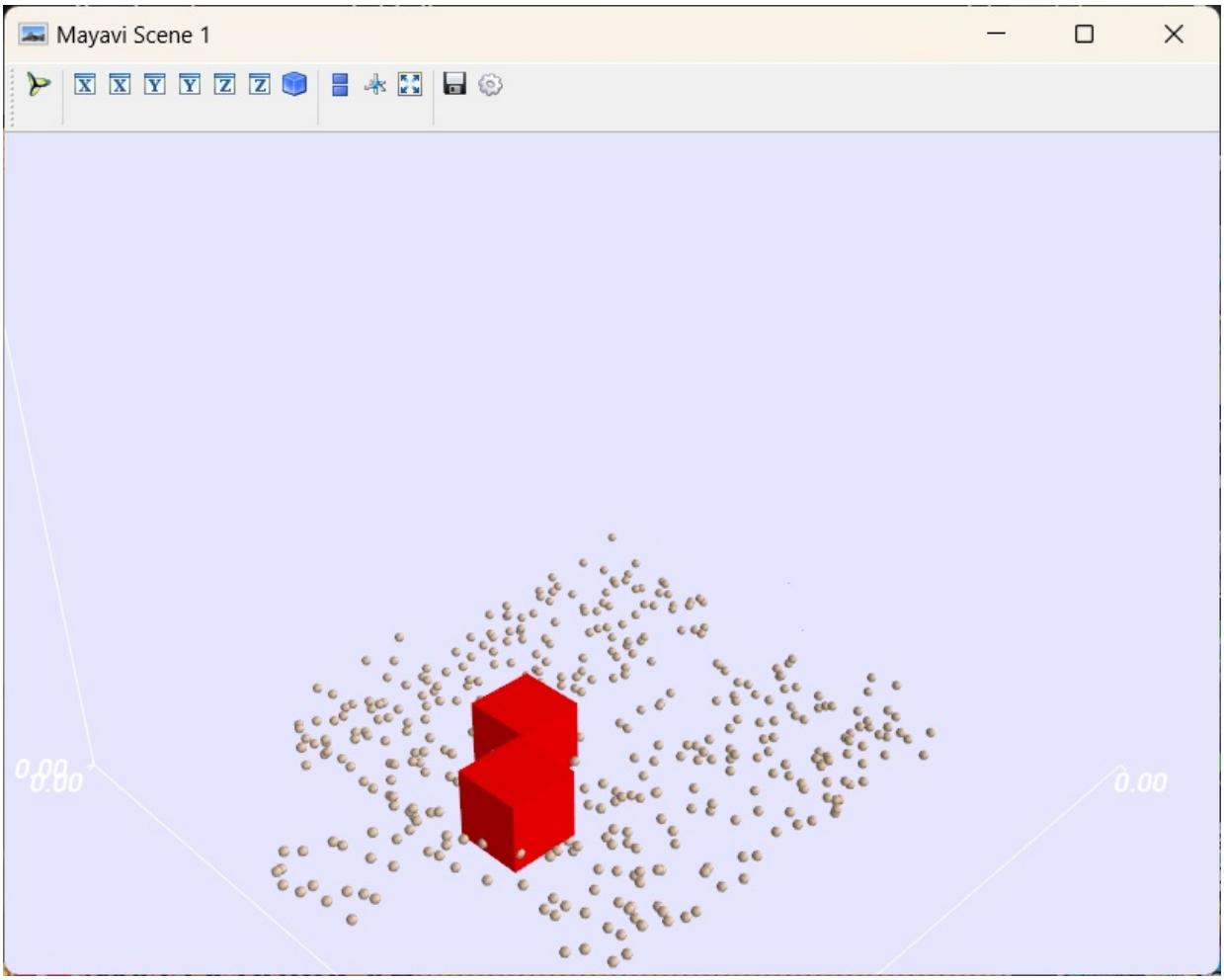
Action Item 1: Double layered particle collision simulation – 3 hour(s).

Project Work Summary

- Environment Setup
  - Define a 3D space (e.g., 3×3×2 meters) to accommodate the rover and two layers of particles.
  - Initialize two distinct layers of particles, each at rest with zero initial velocity.
- Layered Particle Structure
  - Generate two separate layers of particles:
  - Lower layer: particles positioned between z = 0.1 and z = 0.2 meters.
  - Upper layer: particles positioned between z = 0.2 and z = 0.3 meters.
  - Ensure uniform distribution of particles within each layer's x-y plane.

- Simulation Dynamics
  - Keep particles static until acted upon by the rover or other moving particles.
  - Propagate motion through both layers based on initial rover impact and subsequent particle collisions.
- Performance Considerations
  - Optimize collision detection algorithms to handle the increased number of particles efficiently.
  - Consider using spatial partitioning techniques to reduce the number of collision checks required.
- Extensions
  - Introduce settling behavior for particles after being displaced, simulating sand-like properties.
  - Implement variable particle densities or sizes between layers to represent different soil compositions.
  - Add an option to visualize force propagation through the layers using color gradients or particle size variations.

  This approach builds upon the single-layer model, introducing vertical stratification of particles to create a more complex and realistic sand-like environment for rover interaction.



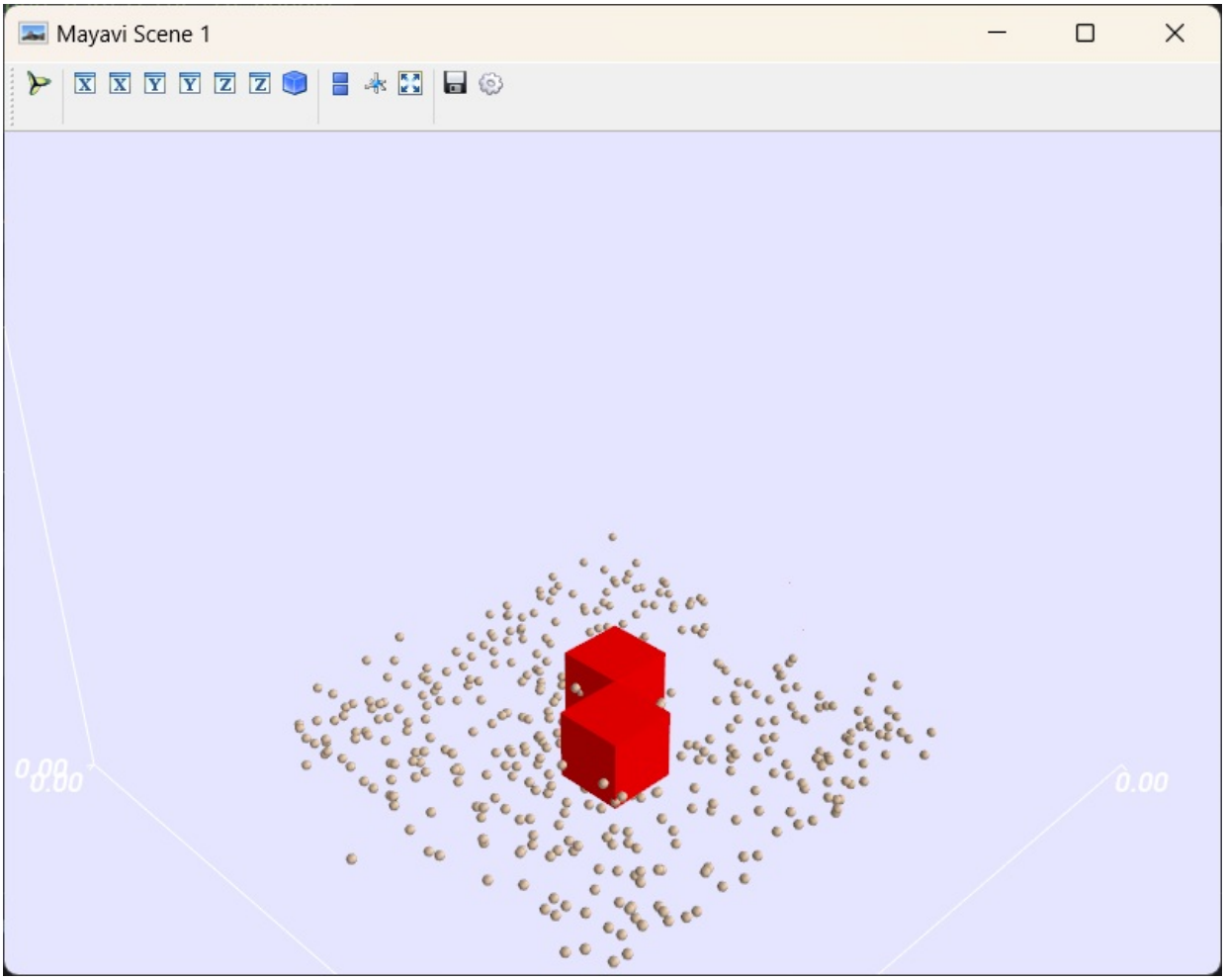
Action Item 2: Optimizing Particle Simulation Computation – 3 hour(s).

Project Work Summary

- Spatial Partitioning
  - Implement a grid-based or tree-based spatial partitioning system (e.g., uniform grid, octree, or k-d tree).
  - Use this structure to efficiently query nearby particles, reducing the number of collision checks.
- Parallel Processing
  - Utilize multi-threading or GPU acceleration (e.g., CUDA with PyCUDA) for particle updates and collision detection.
  - Implement parallel algorithms for particle-particle and rover-particle interactions.
- Vectorization
  - Leverage NumPy's vectorized operations to replace loops where possible.
  - Use SIMD (Single Instruction, Multiple Data) operations for particle updates.
- Collision Detection Optimization
  - Implement broad-phase collision detection using bounding volume hierarchies or sweep and prune algorithms.

- Optimize narrow-phase collision detection with more efficient algorithms (e.g., GJK algorithm for complex shapes).
- Numba Optimization
  - Refine Numba JIT compilation by specifying exact types and using parallel options where applicable.
  - Experiment with different Numba decorators (@njit, @vectorize) for various functions.
- Time Step Optimization
  - Implement adaptive time-stepping to allow larger time steps in less dynamic regions of the simulation.

This task aims to transform the particle simulation into a high-performance computational model, enabling more complex and larger-scale simulations while maintaining real-time interactivity.



Action Item 3: Enhancing Particle Simulation with Increased Density and Layers – 3 hour(s).

Project Work Summary

- Environment Setup
  - Define a 3D space (e.g., 3×3×2 meters) to accommodate the rover and multiple layers of particles.
  - Initialize four distinct layers of particles, each at rest with zero initial velocity.
- Particle Generation
  - Increase the number of particles per layer to 500 (up from 200).
  - Create four layers of particles (up from two), resulting in a total of 2000 particles.
  - Distribute particles uniformly within each layer, with slight vertical offsets between layers.
- Layered Particle Structure
  - Generate four separate layers of particles:
  - Layer 1: particles positioned between z = 0.1 and z = 0.2 meters
  - Layer 2: particles positioned between z = 0.2 and z = 0.3 meters
  - Layer 3: particles positioned between z = 0.3 and z = 0.4 meters
  - Layer 4: particles positioned between z = 0.4 and z = 0.5 meters
  - Ensure uniform distribution of particles within each layer's x-y plane.
- Collision Handling
  - Adapt collision detection algorithms to efficiently handle the increased number of particles.

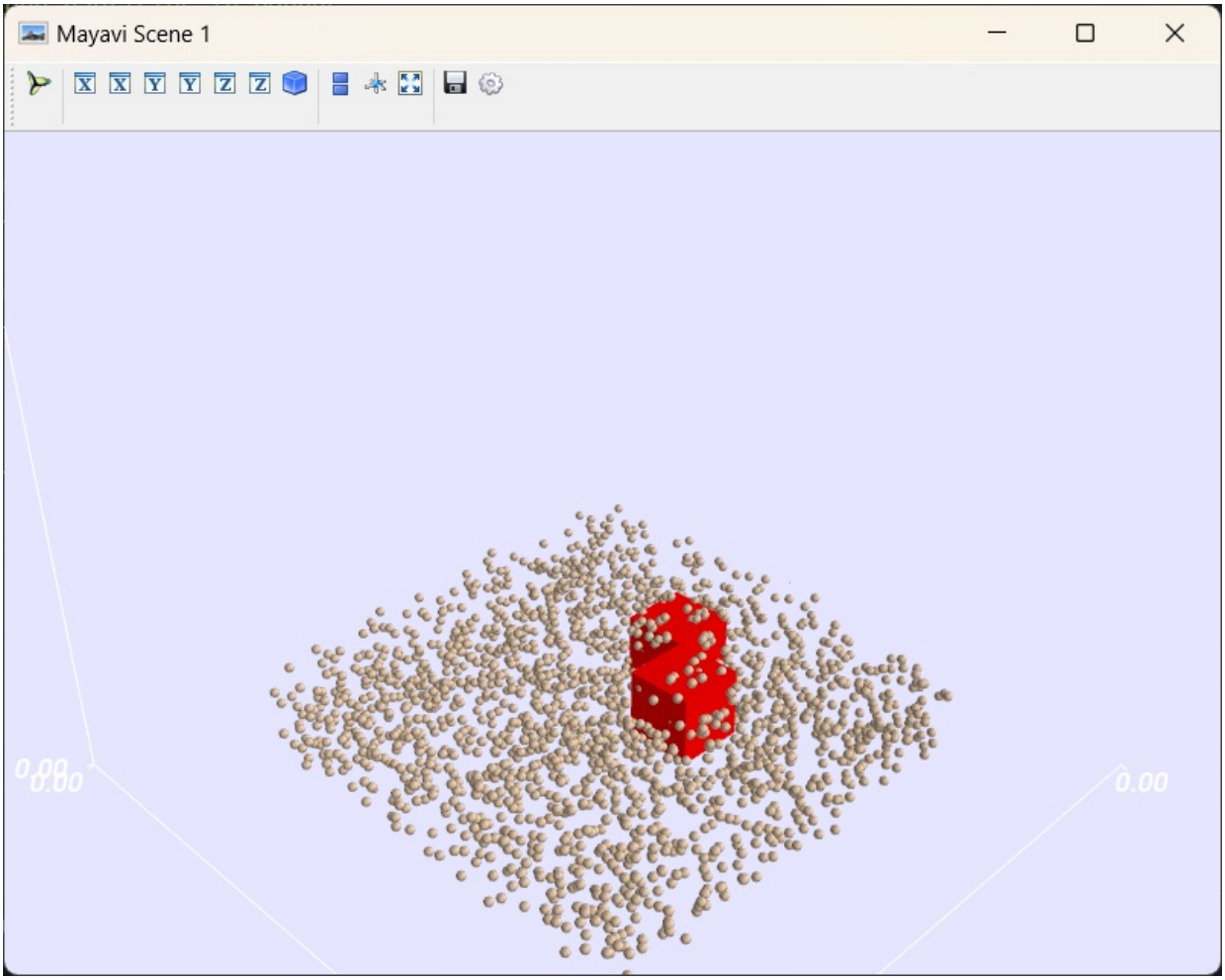
- Implement rover-particle interactions across all four layers simultaneously.
- Enable particle-particle collisions within and between all layers when set in motion.

- 3D Visualization
- Render the increased number of particles as small spheres.
- Optionally, use slightly different shades for each layer for visual distinction.
- Ensure the visualization can handle and smoothly display the increased particle count.

- Simulation Dynamics
- Maintain the existing physics model for particle interactions and movement.
- Verify that the increased particle count doesn't significantly impact the simulation's physical accuracy.

- Rover Interaction
- Ensure the rover can effectively interact with the denser particle environment.
- Adjust rover movement parameters if necessary to account for the increased resistance from more particles.

- This task aims to significantly increase the complexity and realism of the particle simulation by dramatically increasing the number of particles and layers, while maintaining the core functionality and physical accuracy of the original model.



Action Item 4: Soil Simulation with Particles in Python – 3 hour(s).

Project Work Summary

- https://dl.acm.org/doi/10.1145/3460773
- Article: "A Python-based Framework for Smoothed Particle Hydrodynamics" (ResearchGate, 2021)
- Summary of Report
  - Introduces PySPH, an open-source Python framework for particle-based simulations, including Smoothed Particle Hydrodynamics (SPH).
  - Supports multi-physics simulations (e.g., fluid-structure interaction) and integrates with visualization tools like Mayavi.
  - Provides modular components for custom solvers, boundary conditions, and particle interactions.
- Relation to Project
  - Python Compatibility: PySPH’s Python-native implementation aligns with our codebase, enabling seamless

integration of soil particle dynamics.
    - Particle Interaction Models: Offers SPH-based methods for simulating soil as a granular material with cohesion and friction.
    - Visualization Tools: Mayavi integration matches our existing 3D visualization pipeline for rover-soil interactions.
- Motivation for Research
    - Need for accessible tools in Python to prototype agricultural soil simulations.
    - Modular architecture allows customization for stratified soil layers and rover interactions.
    - Validated SPH methods ensure physical accuracy for soil deformation and particle motion.

Action Item 5: Simulating Friction Between Soil and Tools – 3 hour(s).

Project Work Summary

- https://www.mdpi.com/2076-3417/12/5/2524
- Article: "Simulation of Track-Soft Soil Interactions Using a Discrete Element Method" (MDPI, 2022)
- Summary of Report
    - Calibrates Discrete Element Method (DEM) parameters (cohesion, static/rolling friction) to replicate real soil behavior.
    - Validates friction models through shear stress-displacement tests and soil accumulation experiments.
    - Uses scaled-up particles (1 mm) to balance computational efficiency and mechanical accuracy.
- Relation to Project
    - Friction Calibration: Directly applicable to modeling rover-soil friction during ploughing.
    - Parameter Optimization: DEM calibration methods ensure realistic traction/resistance forces.
    - Validation Metrics: Soil accumulation tests mirror our terrain generation requirements.
- Motivation for Research
    - Critical to model traction dynamics for rover movement on multi-layered soils.
    - Scaled-up particle methods reduce computational load while preserving accuracy.
    - Experimental validation bridges simulation and real-world soil behavior.

Action Item 6: Efficient Multi-Particle Handling with GPU Acceleration – 3 hour(s).

Project Work Summary

- https://arxiv.org/html/2406.16091v1
- Article: "Efficient GPU Implementation of Particle Interactions with Cutoff Radii" (arXiv, 2024)
- Summary of Report
    - Proposes GPU-optimized algorithms for particle systems with cutoff radii (short-range interactions).
    - Achieves 14x speedup over CPU methods by minimizing thread divergence and maximizing memory coalescing.
    - Validates performance with large-scale simulations (millions of particles) on NVIDIA A100 GPUs.
- Relation to Project
    - GPU Acceleration: Enables real-time simulation of dense soil layers (>100k particles).
    - Cutoff Optimization: Reduces collision checks from $O(n^2)$ to $O(n)$ via spatial partitioning.
    - Scalability: Supports future expansion to larger terrains and complex rover mechanics.
- Motivation for Research
    - Performance Demands: GPU methods are essential for interactive agricultural simulations.
    - Resource Efficiency: Minimizes VRAM usage while maintaining high particle counts.
    - Industry Alignment: NVIDIA’s Omniverse/Isaac Sim compatibility ensures scalability for robotic applications.

Action Item 7: Next week plan – 1 hour(s).

Project Work Summary

- **Next Week's Development Tasks**

  ### 1. **Validate and Stabilize Core Simulation**
  - **Dependency Verification**
  - Ensure all required imports (e.g., `numpy`, `numba`, `mayavi`) are consistently defined across modules.
  - Create a unified initialization script to prevent `NameError` issues (e.g., missing `Rover` class or `jit` decorators).

- **Error Handling**
- Add try-except blocks to critical functions (collision detection, visualization) with descriptive error logging.
- Standardize variable names and scopes to avoid undefined function errors (e.g., `rover_particle_interact`, `particle_collision`).

---

2. **Optimize Particle-Particle Collisions**
- **Spatial Partitioning**
- Implement a grid-based collision detection system to replace brute-force pairwise checks.
- Use `numba`-optimized neighbor search within grid cells.
- **Performance Benchmarking**
- Measure simulation speed (steps/second) for varying particle counts (1k to 10k).
- Profile CPU/GPU usage to identify bottlenecks.

---

3. **Simulate Realistic Farmland Structure**
- **Soil Layer Properties**
- Expand soil types (`sand`, `clay`, `silt`, `loam`) with density, friction, and moisture parameters.
- Assign unique particle properties per layer (e.g., `clay` particles stick together with cohesion forces).
- **Terrain Visualization**
- Render soil layers with distinct colors (e.g., brown for loam, gray for clay).
- Add elevation contours using the `generate_complex_terrain` function.

---

4. **Implement Complex Land Structures**
- **Procedural Terrain Generation**
- Integrate hills/valleys using the `generate_complex_terrain` function into the 3D simulation.
- Adjust particle placement to follow terrain elevation (e.g., particles settle in valleys).
- **Dynamic Soil Distribution**
- Simulate erosion by allowing particles to slide downhill based on slope angle and soil type.
- Add "tilled soil" regions with randomized particle positions for farmland realism.

---

5. **Enhanced Validation & Testing**
- **Automated Test Suite**
- Create unit tests for critical functions (e.g., `Rover.move()`, `particle_collision`).
- Validate boundary conditions with edge-case scenarios (e.g., particles at domain edges).
- **Performance Metrics**
- Track memory usage and computation time for large particle counts.
- Compare simulation accuracy against real-world soil displacement data.

---

6. **Advanced Visualization Tools**
- **3D Cross-Section Views**
- Add toggleable layer visibility to inspect subsurface particle interactions.
- Implement a cutting-plane tool to slice through terrain layers.
- **Real-Time Metrics Dashboard**
- Display kinetic energy, particle displacement, and rover velocity in a side panel.
- Export simulation data (CSV/JSON) for post-analysis.

---

7. **Prepare for Future Expansions**
- **Modular Architecture**
- Decouple soil properties, terrain generation, and collision logic into separate modules.
- Create configuration files for easy parameter tuning (e.g., `farmland_config.yaml`).
- **Documentation**
- Write a user guide for adding new soil types or terrain features.
- Publish performance benchmarks for reference.

Stretch Goals
- Integrate vegetation roots as fixed particles that resist displacement.
- Add farming machinery models (plows, seeders) with specialized particle interaction logic.
- Simulate weather effects (rain softening soil, wind erosion).

This plan builds on existing work while addressing stability, performance, and realism gaps. Each task aligns with the goal of scaling particle counts and simulating complex farmland environments.

Action Item 8: Report writing – 1 hour(s).

Project Work Summary

- Created word document layout to write contents of the weekly progress. Created relevant subsections in the epicspro website and documented 20 hours of weekly progress. Collected relevant documents research papers, relevant links and company's objective from their portal.