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	Physically-Based Interactive Sand Simulation	3
2	Pygame Installation and Basic Setup	2
3	Setting up Sand Particle Simulation	3
4	Terrain Generation and Visualization	3
5	Interactive Terrain Manipulation	3
6	Rover to the 3D Farmland Simulation	3
7	Migrate Soil-Rover Interaction Simulation from MATLAB to Python	1
8	Next week plan	1
9	Report Writing	1
	Total hours for the week:	20

Verification Documentation:

Action Item 1: Physically-Based Interactive Sand Simulation – 3 hour(s).

Research

- https://www.researchgate.net/publication/235223044
- Physically-Based Interactive Sand Simulation
- Summary of Report:
  - The paper proposes a physically-based model for real-time, interactive simulation of sand dynamics in 3D environments.
  - It combines a discrete model for sandpile evolution with a soil-tool interaction model based on Perumpral's approach.
  - The authors implement and test their model, demonstrating its efficiency and suitability for real-time applications like driving simulators.
- Relation to Project:
  - Provides a framework for simulating sand physics in real-time, which could be adapted for use with Pygame.
  - Offers insights into handling sand-object interactions, crucial for creating realistic sand simulations.
  - Demonstrates the feasibility of integrating complex physical models into interactive graphical applications.
- Motivation for Research:
  - To develop a more physically accurate model for sand simulation that can still run in real-time.
  - To improve upon existing models by incorporating both vertical and horizontal forces in sand-object interactions.
  - To create a sand simulation model suitable for integration into demanding applications like driving simulators.

Action Item 2: Pygame Installation and Basic Setup – 2 hour(s).

Project Work Summary

- Install Pygame and create a single instance to serve as a foundation for a modular, scalable agricultural simulation environment.
- Pygame Installation:
  - Ensure Python is installed on your system.
  - Use pip to install Pygame: `pip install pygame`
  - Verify the installation by importing Pygame in a Python interpreter.
- Create a Basic Pygame Instance:
  - Import the Pygame module.
  - Initialize Pygame.
  - Set up a display window with appropriate dimensions.
  - Implement a basic game loop that handles events and updates the display.
- This task will establish the foundational elements needed to transition from a MATLAB-based simulation to a Pygame-powered multi-rover agricultural simulation. The modular approach will facilitate future expansions and the implementation of more complex behaviors and interactions.

Action Item 3: Setting up Sand Particle Simulation – 3 hour(s).

Project Work Summary

- Create an interactive sand simulation where users can add sand particles to a 2D grid. The particles should fall and pile up realistically, simulating basic granular behavior.
- Key features:

    - Use a 1920x1080 pixel window for the simulation.
    - Allow users to add sand particles by clicking the mouse.
    - Implement gravity and simple pile-up mechanics for the particles.
    - Render the particles as small circles on a black background.
    - Update the simulation at 60 frames per second.
- Implementation details:
    - Use a 2D grid to track particle positions.
    - Create a SandParticle class to represent individual sand grains.
    - Update particle positions based on simple rules:
        - Fall straight down if the space below is empty.
        - Fall diagonally left or right if blocked directly below.
        - Draw particles as 1-pixel radius circles with a sand-like color.
        - Handle user input to add new particles at the mouse click position.
        - Continuously update and redraw the simulation.
- This simulation provides a basic framework for more complex granular material interactions, which could be extended to include features like different particle types, obstacle interactions, or more advanced physics calculations.

Action Item 4: Terrain Generation and Visualization – 3 hour(s).

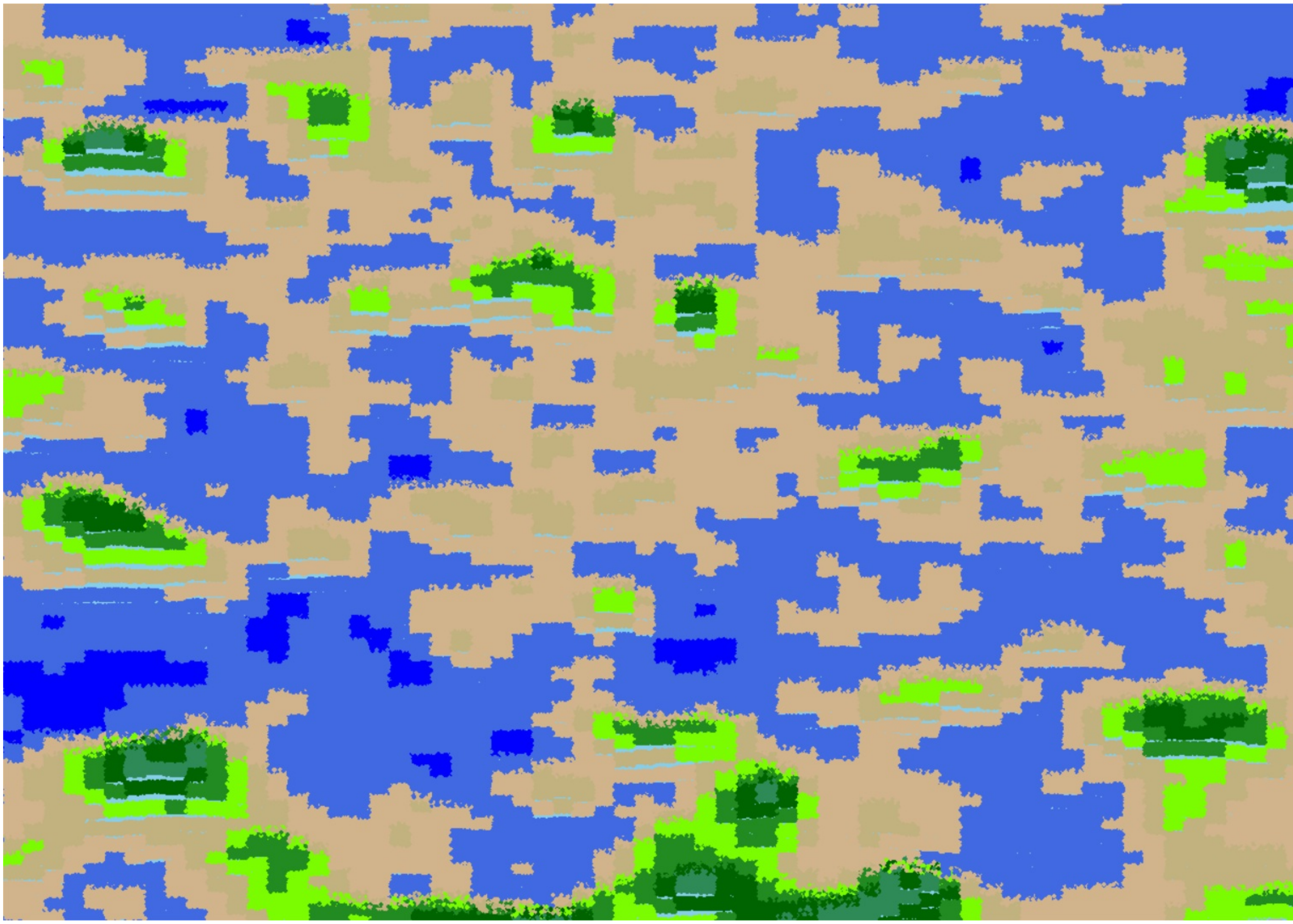
Project Work Summary

- Create a realistic 3D terrain using a height map and visualize it using colored particles.
- Key features:
  - Use a 1920x1080 pixel window for the simulation.
  - Generate a 10m x 10m terrain using a combination of sine waves and Gaussian bumps.
  - Represent the terrain using millions of colored particles.
  - Implement a color gradient based on terrain height.
  - Render the terrain in a pseudo-3D view.
- Implementation details:
  - Use NumPy to generate a height map with random variations.
  - Apply Gaussian filtering to smooth the terrain.
  - Create a SoilParticle class to represent individual terrain points.
  - Implement a get_color method that assigns colors based on particle height.
  - Use Pygame to render particles as small circles with appropriate colors.
  - Project 3D coordinates onto a 2D screen for visualization.

Action Item 5: Interactive Terrain Manipulation – 3 hour(s).

Project Work Summary

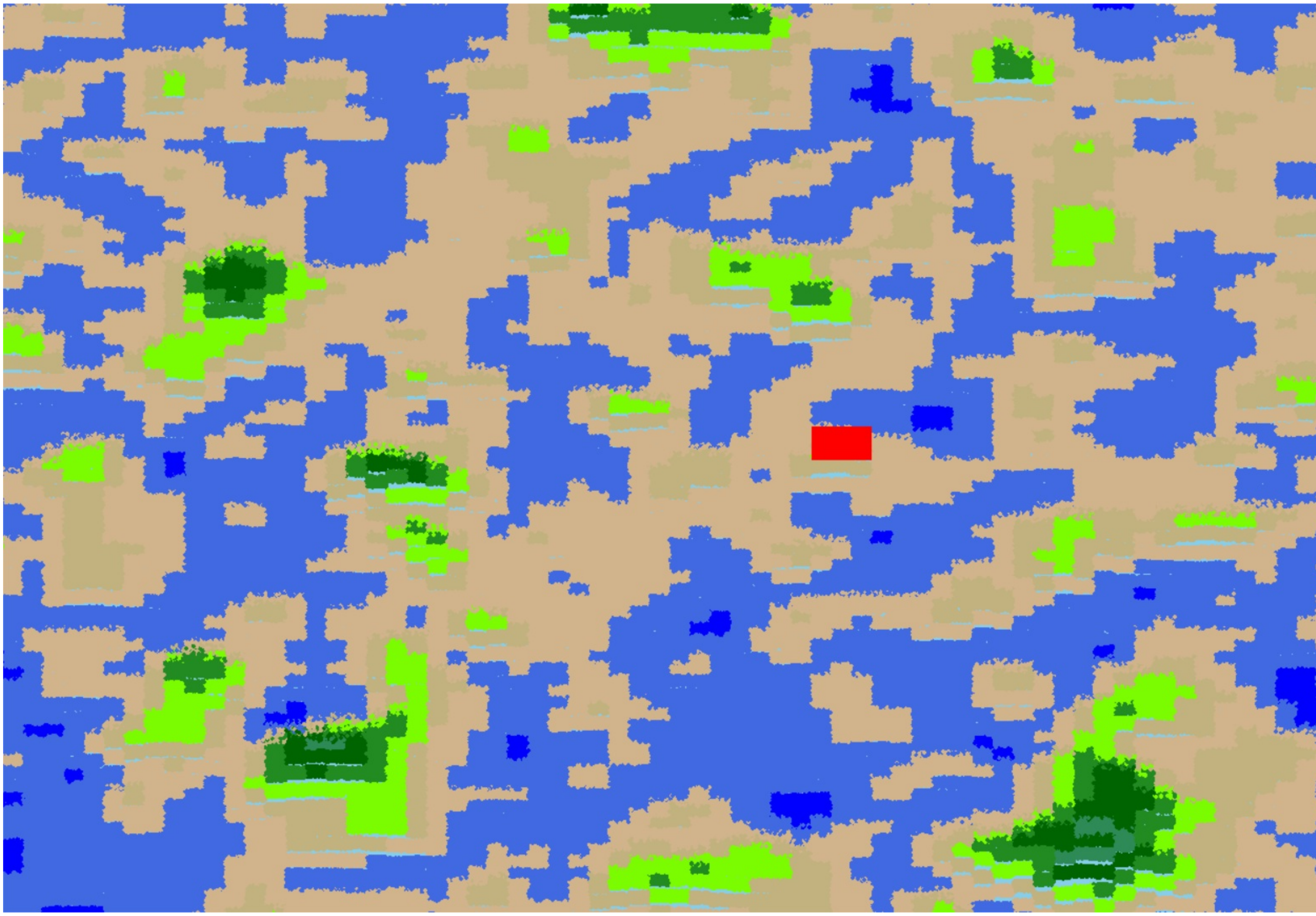
- Implement user interactions to manipulate the terrain in real-time, simulating basic soil mechanics.
- Key features:
  - Allow users to add or remove soil using mouse input.
  - Implement simple soil physics, including gravity and pile-up mechanics.
  - Update the terrain in real-time at 60 frames per second.
  - Visualize changes in the terrain as particles move and settle.
- Implementation details:
  - Handle mouse input to add or remove particles at the clicked position.
  - Update particle positions based on simple physics rules:
    - Apply gravity to make particles fall.
    - Implement pile-up mechanics when particles collide.
  - Use a grid system to track particle positions and optimize collision detection.
  - Continuously update particle positions and redraw the scene.
  - Implement a simple erosion model to simulate natural terrain changes over time.



Action Item 6: Rover to the 3D Farmland Simulation – 3 hour(s).

Project Work Summary

- Implement a rover object in the existing 3D farmland simulation using Pygame.
- Key features:
    - Create a Rover class with dimensions 0.3m x 0.3m x 0.15m (length x width x height).
    - Position the rover on the terrain surface.
    - Render the rover as a 3D cuboid in the simulation window.
    - Enable basic movement controls for the rover.
- Implementation details:
    - Define a Rover class with attributes for position, dimensions, and color.
    - Implement a draw method for the Rover class that projects the 3D cuboid onto the 2D screen.
    - Convert rover dimensions and position from world coordinates to screen coordinates.
    - Use Pygame's drawing functions to render the rover as a rectangle from the top-down view.
    - Add the rover to the main simulation loop for continuous rendering.
    - Implement basic keyboard controls to move the rover across the terrain.
    - Ensure the rover's z-position adapts to the terrain height at its current x-y position.
- This task will integrate a movable rover into the existing farmland simulation, allowing for future interactions between the rover and the soil particles.



Action Item 7: Migrate Soil-Rover Interaction Simulation from MATLAB to Python – 1 hour(s).

Project Work Summary

- Transition the existing MATLAB-based terrain generation and rover simulation to Python, leveraging Pygame for visualization and physics simulation capabilities.
- Key features:
  - Implement a particle-based soil representation for more realistic ploughing mechanics.
  - Utilize Pygame's 2D rendering capabilities to visualize the 3D terrain from a top-down perspective.
  - Integrate basic physics simulation for soil-rover interactions, including ploughing effects.
- Implementation details:
  - Use NumPy to recreate the terrain generation process, including sine wave patterns and Gaussian bumps.
  - Implement a SoilParticle class to represent individual soil elements, allowing for more granular interactions.
  - Leverage Pygame's drawing functions to render soil particles and the rover on a 2D screen.
  - Develop simple physics rules for particle movement and interactions, simulating soil displacement during ploughing.
  - Implement collision detection between the rover and soil particles using Pygame's built-in functions.
  - Create a main simulation loop that updates particle positions, handles rover movement, and redraws the scene at 60 FPS.
- **This migration allows for more accurate soil physics simulation, particularly for ploughing mechanics, which was not feasible in the original MATLAB implementation. The use of Python and Pygame provides a more flexible and interactive environment for real-time soil-rover interactions.**

Action Item 8: Next week plan – 1 hour(s).

Project Work Summary

- Develop an integrated path planning algorithm that considers:
  - Terrain characteristics and soil properties
  - Multiple robot coordination
  - Dynamic obstacle avoidance
  - Energy efficiency
- Design a soil interaction model for rovers that:
  - Simulates realistic soil deformation during plowing
  - Predicts soil compaction and its effects on crop growth
  - Adapts to different soil types and moisture levels
- Implement a multi-robot coordination system for agricultural tasks that:
  - Optimizes task allocation based on individual robot capabilities
  - Enables real-time communication and data sharing between robots
  - Adapts to changing environmental conditions and task priorities

Action Item 9: 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. Total hours : 20