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.

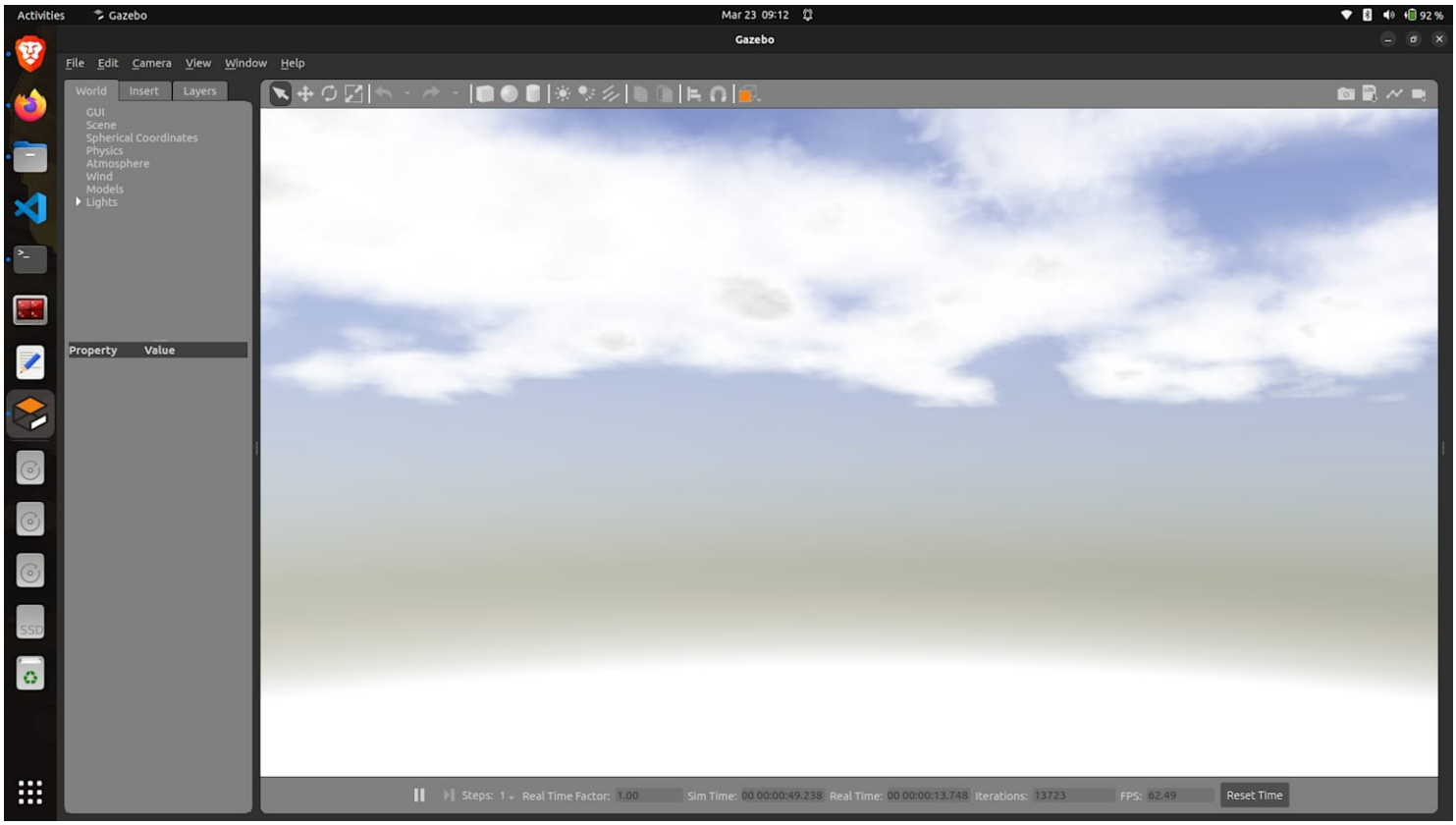
#	Action Item/ Explanation	Total Time This Week (hours)
1	cpr gazebo Farm land as a replacement for our virtual maize farmland	3
2	Exploring other Farmland Maps Compatible with ROS 2 Humble	3
3	Setting Up and Enhancing the Agricultural Simulation with Husky Robot	3
4	Adding Collision, Gravity, and Creating the Node for Husky Movement	3
5	Mapping, Navigation and Creation of Farmland Maps	3
6	Creating Realistic Farmland Maps in Gazebo with ROS 2	3
7	Weekly Plan for Next Week	1
8	Report Writing	1
	Total hours for the week:	20

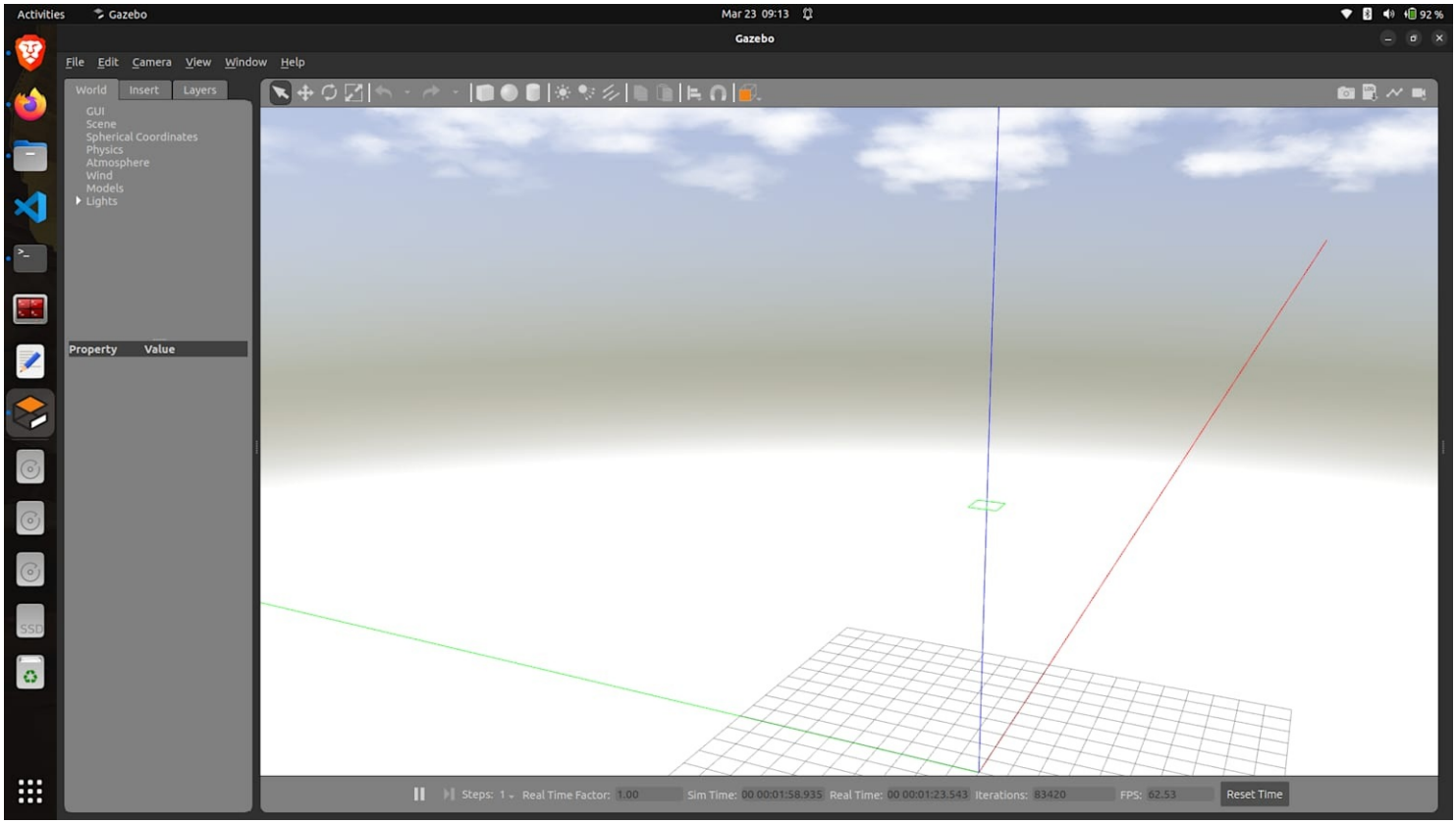
Verification Documentation:

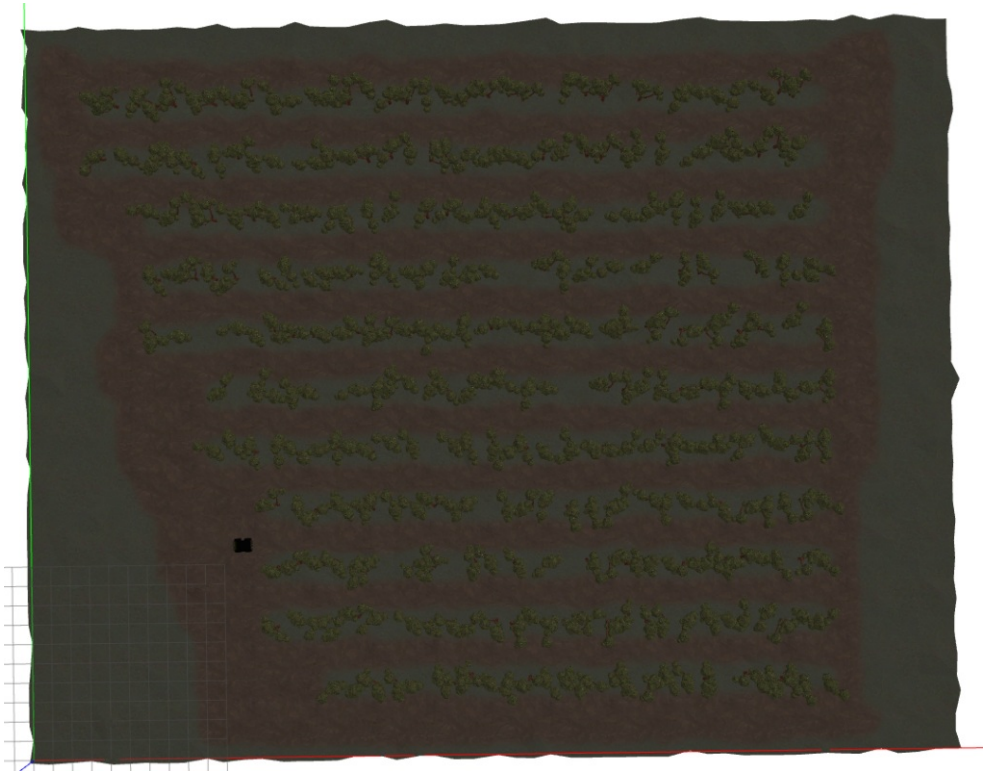
Action Item 1: cpr gazebo Farm land as a replacement for our virtual maize farmland – 3 hour(s).

Project Work Summary

- We found last week how husky found difficult to climb the heightmap of the terrain of the virtual maize map. Hence I found another alternative map from the clear path robotics gazebo repository for its prebuilt agricultural simulation worlds designed specifically for robots like Husky, Jackal, and Warthog.
- Initially, the Clearpath Robotics Gazebo repository seemed like an ideal choice because it has prebuilt agricultural simulation environments and tailor made for Husky. However, several roadblocks and technical challenges made it difficult to implement successfully.
- Compatibility Issues
  - This repository was built for ROS 1 (Kinetic/Noetic), while I was using ROS 2 (Humble) for the project purposes. This mismatch in the ROS is a significant problem for both building process and runtime.
  - Then instead of updating the ROS2 to ROS1, I manually update the repository to ROS 2 by modifying CMakeLists.txt and replacing catkin dependencies with ament_cmake. This approach has worked for me in the path for a different project, but it did not work for this repository, making the process cumbersome and unreliable.
  - I even was able to open the map with the gazebo and was able to visualize the cloud and the overall map but was not able to see the heightmaps.
- Missing Terrain in Simulation
  - The Gazebo simulation did launch successfully, but the expected farmland environment failed to render properly. Instead of the agricultural scene, only the sky and a few other elements such as the sun were visible.
  - To troubleshoot, all model paths were correctly set in the GAZEBO_MODEL_PATH environment variable. Additionally, heightmap .tif files from the cloned file were preprocessed using GDAL to meet Gazebo's format requirements.
- Despite making a lot of efforts to adapt clearpathrobotics/cpr_gazebo for ROS 2 Humble, but none of the method worked due to compatibility challenges and missing terrain made it not a viable choice.


Activities
Gazebo
Mar 23 09:12
92 %
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:00:49.238 Real Time: 00 00:00:13.748 Iterations: 13723 FPS: 62.49
Reset Time

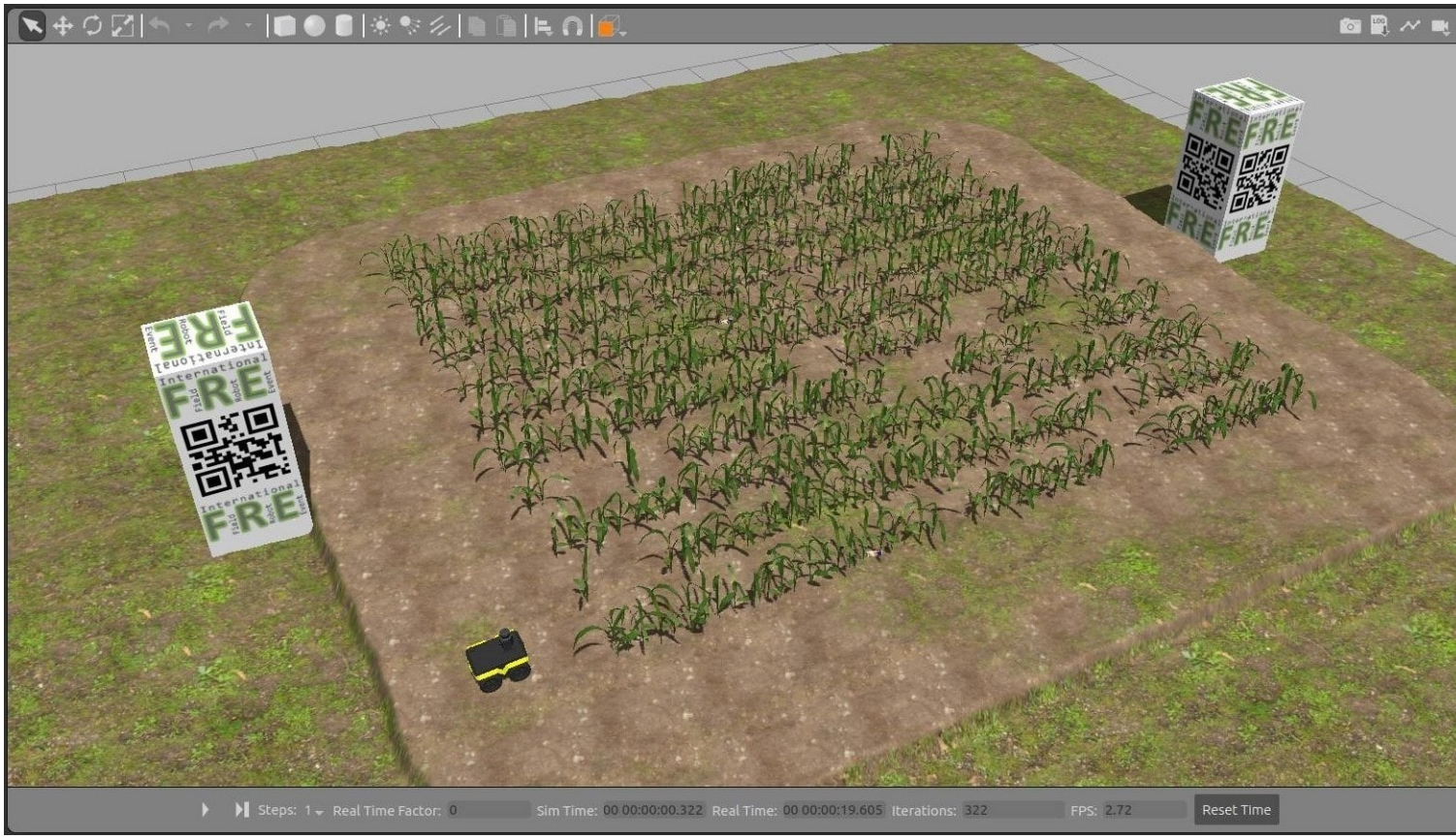

Activities
Gazebo
Mar 23 09:13
92 %
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:01:58.935 Real Time: 00 00:01:23.543 Iterations: 83420 FPS: 62.53
Reset Time



Action Item 2: Exploring other Farmland Maps Compatible with ROS 2 Humble – 3 hour(s).

Project Work Summary

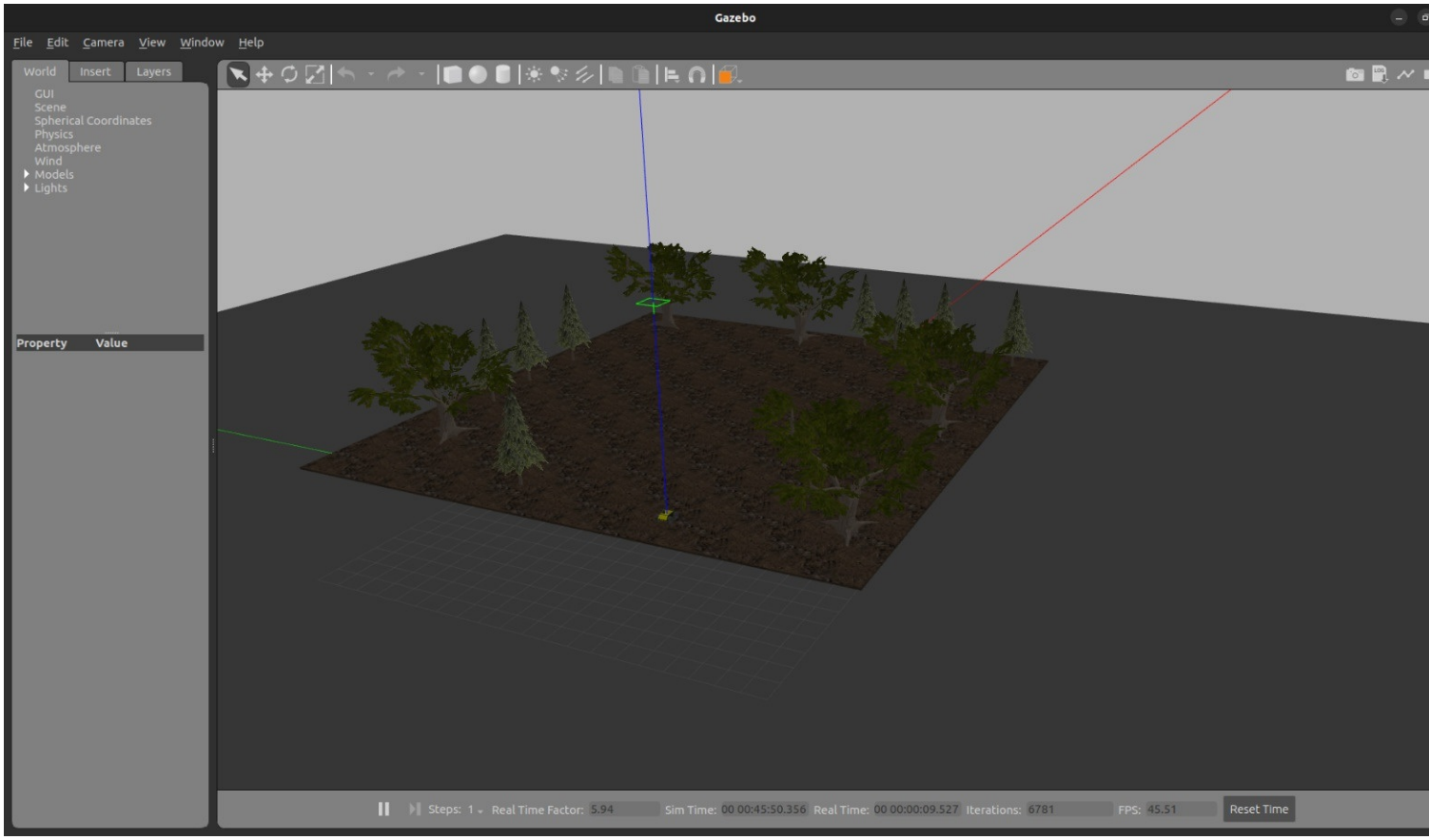
- Hence now the new primary objective is to find a well-suited agricultural simulation map for ROS 2 Humble that would allow the Husky robot to move smoothly across the terrain, avoiding unnecessary obstacles or rough patches that could interfere with its movement.
- Virtual Maize Field
  - The first repository explored was FieldRobotEvent/virtual_maize_field, which consisted of procedurally generated maize fields designed that was also compatible with ROS 2 Humble. While this map allowed for customization of row spacing and terrain properties, it presented several challenges:
  - The terrain had large heightmaps, with significant climbing heights for the tires, making it difficult for Husky to traverse smoothly.
  - Attempts were made to increase the speed and at different locations through the generate_world.py scripts did not help. Husky still struggled, especially when navigating sharp curves or areas with the maize plants.
- AOC Tomato Farm
  - Next, the LCAS/aoc_tomato_farm repository was considered, which is basically a simulated tomato farm and greenhouse generator designed for ROS 2 in Gazebo.
  - The generated environments were optimized for indoor farming, not open our medium scale outdoor agricultural fields.
  - The layout featured densely packed plants and confined spaces, which will be an obstacle for Husky to maneuver.
- FarmWithCropRow.world
  - The final map tested was FarmWithCropRow.world, sourced from the agribot_croprow_generator github repository. This map provided a structured farmland environment with well-defined crop rows and a more suitable terrain for Husky’s movement.
  - A flat terrain is the most important consideration for choosing this map with minimal elevation variations, ensuring smoother navigation.
  - Evenly spaced crop rows, which aligned well with Husky’s modular design and payload requirements.
- The Virtual Maize Field provided realistic farmland details, its excessive terrain complexity made it impractical. On the other hand, AOC Tomato Farm was too specialized for indoor scenarios, limiting its versatility. Ultimately, FarmWithCropRow.world was chosen out as the best option, offering a structured, flat, and well-designed environment that fully supports Husky’s navigation and motion planning tasks.

- 

Action Item 3: Setting Up and Enhancing the Agricultural Simulation with Husky Robot – 3 hour(s).

Project Work Summary

- Now we have finally fixed FarmWithCropRow.world, sourced from the agribot_croprow_generator repository, as the most suitable agricultural simulation environment.
- To access the necessary files, the PRBonn/agribot github repository was cloned locally, specifically from the agribot_croprow_generator directory. The repository consist Python-based crop row generator script (GazeboCropRowGenerator.py), which could be used for the procedural generation of farmland environments.
- There was also another world filed named farm.world, as it appeared to be a useful setup for the simulation. However, launching the world led to lot of parsing errors due to invalid XML constructs and missing file references. Despite multiple attempts to debug and manually integrate the crop row definitions, rendering issues persisted, making the world difficult to work with.
- Then I found the FarmWithCropRow.world, which was able to successfully load the Gazebo world map without errors. This world provided a stable agricultural environment, featuring structured crop rows and a flat terrain, making it far more practical for further development.
- To integrate robot navigation within the simulation, the Husky robot model was added directly to the FarmWithCropRow.world file. The following steps were taken:
  - Installed the necessary Husky simulation packages to ensure the model files were available.
  - Placed the Husky robot within the world file, fine tuning its initial position and orientation to avoid collisions with crop rows.
  - Verified the placement visually in Gazebo, ensuring that the robot aligned correctly with the farmland layout.
- With the help of FarmWithCropRow.world, I was finally able open an environment that now supports both functional crop rows and the Husky robot model. Now we can motion planning along predefined crop rows, navigation testing across uneven farmland terrain and sensor-based data collection.

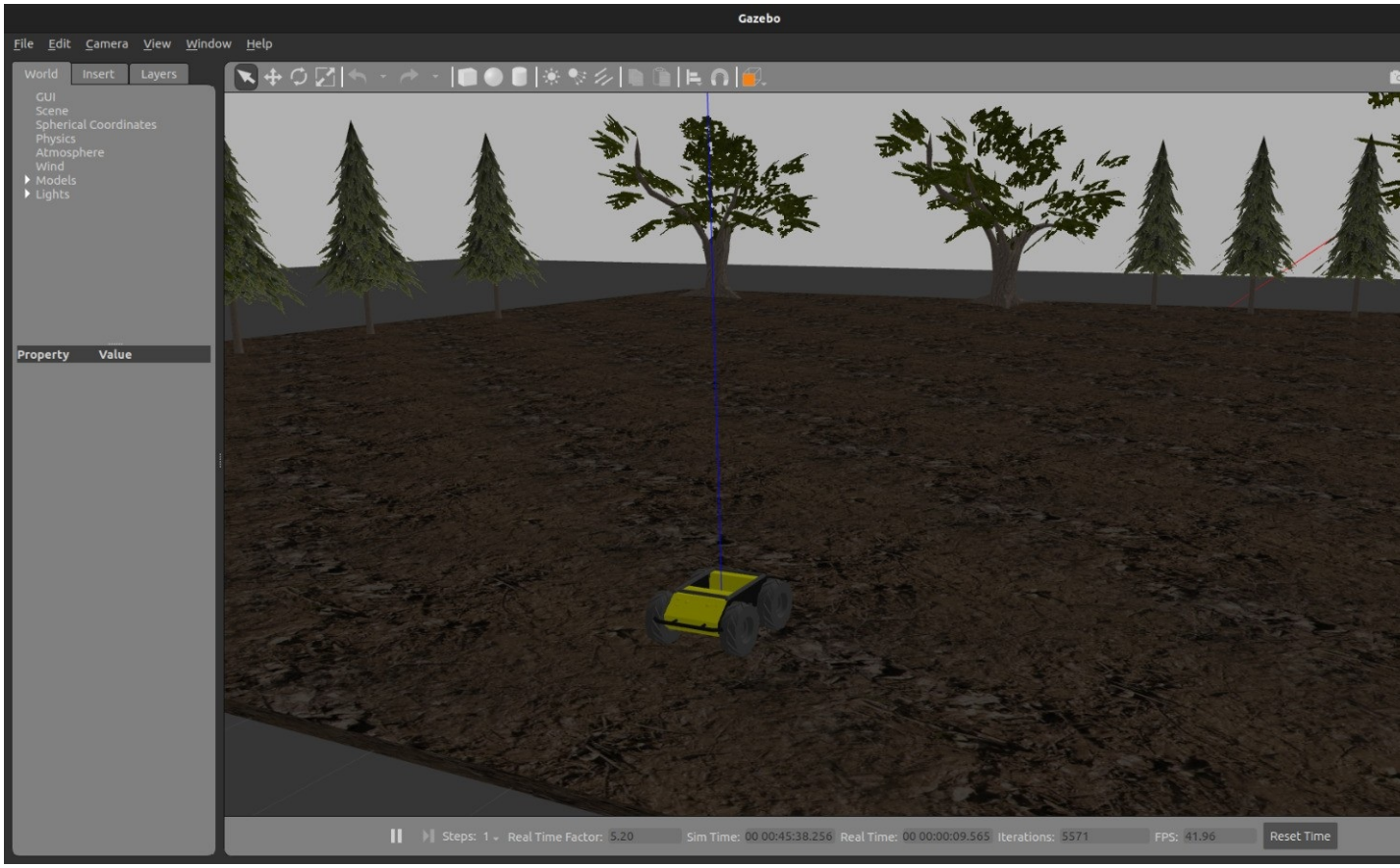
- 

-

Action Item 4: Adding Collision, Gravity, and Creating the Node for Husky Movement – 3 hour(s).

Project Work Summary

- Though the map and the husky was created easily, the husky started to wobble and move automatically due to absence of phyisics based changes needed in the final world file.
- Collision and Gravity Implementation
  - Adding a <collision> block in the SDF file to match Husky's real-world dimensions to make a proper rigid contact with the road.
  - Setting appropriate friction coefficients (mu and mu2) to make the rover stay in one place and not move.
  - Enabling gravity effects to ensure Husky's wheels stayed rigidly in contact with the ground during navigation, particularly on uneven terrain.
- Developing the Movement Node
  - To test if the husky movement, a ROS 2 node was developed to command Husky to perform a simple movement from point A to point B across the farmland. This node published velocity commands (geometry_msgs/Twist) to the /husky_velocity_controller/cmd_vel topic, allowing precise motion control.
  - Node Implementation (husky_mover.py) was made as a Python-based ROS 2 node was created within the husky_control package to published linear velocity commands to drive Husky forward.
- Execution and Testing
  - Once the node was integrated and executed alongside Gazebo, Husky successfully navigated from one edge of the crop rows to the other.
  - After implementing collision physics, gravity, and movement control, the following results were observed:
    - Visual verification in Gazebo confirmed Husky's realistic interaction with the terrain.
    - Movement across the crop rows was smooth and controlled, with no unintended collisions or erratic behavior.
    - Friction coefficients and velocity commands were fine-tuned for optimal navigation on dirt paths.



Action Item 5: Mapping, Navigation and Creation of Farmland Maps – 3 hour(s).

Project Work Summary

- https://webthesis.biblio.polito.it/33156/
- ROS2-Based AMR System for Mapping and Navigation in Unknown Environments
- Summary of Report
  - Comparing SLAM Algorithms
    - This research paper takes a deep dive into Simultaneous Localization and Mapping (SLAM) algorithms within ROS 2's Nav2 library, evaluating grid-based and topological mapping techniques, as well as AMCL (Adaptive Monte Carlo Localization) and EKF (Extended Kalman Filter). The study highlights each algorithm's strengths in mapping accuracy, computational efficiency, and adaptability to dynamic environments.
  - Simulating SLAM in Gazebo
    - The study showcases that Gazebo can be used a powerful simulation tool for testing SLAM algorithms in a controlled setting. By leveraging the integration with ROS 2, the research explores how robots can actively map their surroundings, avoid obstacles, and navigate efficiently.
  - From Simulation to Reality
    - To validate the system's performance, the research also provides parameters to find the effectiveness with simulation and tests SLAM in a controlled lab environment. These real-world trials assess how well the robot can navigate unknown spaces and adapt to dynamic obstacles, proving the system's robustness and reliability.
- Relation to Project
  - Mapping Accuracy for Farmland
    - The study's insights are critical for developing precise maps of agricultural environments, which could be used by out robot to navigate through crop rows, ditches, and uneven terrain.
  - Gazebo as a Simulation Tool
    - The research proposed Gazebo as a reliable mode for simulating farm-like conditions, ensuring robots are thoroughly tested before being deployed in the field.
  - Harnessing ROS 2 Capabilities
    - The study highlights how ROS 2's distributed computing and modular design can improve path planning and mapping efficiency—key elements for robotic navigation in farmland.
- Motivation for Research
  - Navigating Dynamic Terrain
    - Farmland created for the paper isn't just open space it has obstacles like crops, irrigation systems, and uneven ground. The SLAM techniques proves to be helping robots move accurately in such challenging environments.
  - Ensuring Reliability Before Deployment
    - By testing mapping algorithms in Gazebo first, we can refine the system before implementing it to the real-world farm land, reducing unexpected failures.
  - Selecting the Right SLAM Method for Agriculture
    - The comparison of different SLAM approaches provides a solid foundation for choosing the best algorithm to enhance robotic navigation in farm settings.

Action Item 6: Creating Realistic Farmland Maps in Gazebo with ROS 2 – 3 hour(s).

Project Work Summary

- https://github.com/FieldRobotEvent/virtual_maize_field
- Virtual Maize Field: Procedural Generation of Agricultural Worlds in Gazebo
- Summary of Report
  - Procedural Generation of Farmland Worlds
    - This github repository offers an in-depth Python script (generate_world.py) that can automatically create randomized maize fields with adjustable settings. We can fine-tune key parameters like row length, plant spacing, ground elevation, and ditch depth, making it easy to design custom farm environments for simulation. The script also introduces randomized placement errors to mimic real-world agricultural conditions, ensuring a more realistic testing environment for robotic applications.
  - Customizable Terrain Features
    - The package gives us complete control over terrain details, allowing adjustments to ground resolution and headland size and ditch depth and plant height variations
    - These features create highly adaptable and realistic test environments, making the package an excellent tool for evaluating robot navigation, motion planning, and obstacle avoidance in farming scenarios.
  - Seamless ROS 2 Integration
    - The generated farm worlds are fully compatible with ROS 2 Humble and work smoothly with Gazebo Classic and Ignition Gazebo. Users can either use predefined robot configurations or customize their setups for seamless integration with ROS 2-based agricultural robots.
- Relation to the Project
  - Automated Farmland Map Creation

        - Instead of spending time manually designing virtual farmland, this package automates the process, generating realistic and customizable maps tailored to specific research needs.
    - Simulating Real-World Terrain
        - The tool enables the creation of detailed and diverse farm landscapes, including curved crop rows, uneven ground, and obstacles like weeds and ditches—perfect for testing robot navigation strategies.
    - ROS 2 Compatibility
        - The seamless integration with ROS 2 workflows ensures that robots can interact naturally with the environment, making simulation-based testing more efficient and reliable.
- Motivation for Research
    - Dynamic Terrain Generation
        - Researchers can quickly create multiple farm layouts without manually designing each one, allowing for extensive experimentation and testing.
    - Customizable Farming Scenarios
        - The ability to tweak parameters like row spacing, plant placement, and elevation ensures that maps can be tailored to specific robot navigation and farming applications.
    - User-Friendly for Agricultural Robotics
        - The package simplifies map creation and robot testing, allowing researchers to focus on improving robotic performance instead of spending time manually setting up test environments.

Action Item 7: Weekly Plan for Next Week – 1 hour(s).

Project Work Summary

- Edit the already created husky_mover.py within the husky_control package to bring out better results for the husky movement. This allows the robot to traverse the maize field in FarmWithCropRow.world.
- Updated Husky’s SDF file to define accurate collision boundaries for crops and for the husky, ensuring the robot interacts properly with crop rows and obstacles in Gazebo.
- Launched FarmWithCropRow.world in Gazebo and testHusky’s movement from one corner to other corner to other of the farm to the other, confirming smooth and realistic navigation.
- Installed and configured and start exploring Nav2 for autonomous movement and researched SLAM tools like SLAM Toolbox and Cartographer for future mapping and navigation.
- Utilize GazeboCropRowGenerator.py from agribot_croprow_generator to create more complex custom made farm layouts, to better validating compatibility with Husky’s control node.
- Recorded all simulation updates, including collision handling, friction adjustments, and movement testing, while also resolving any errors encountered.

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.