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.

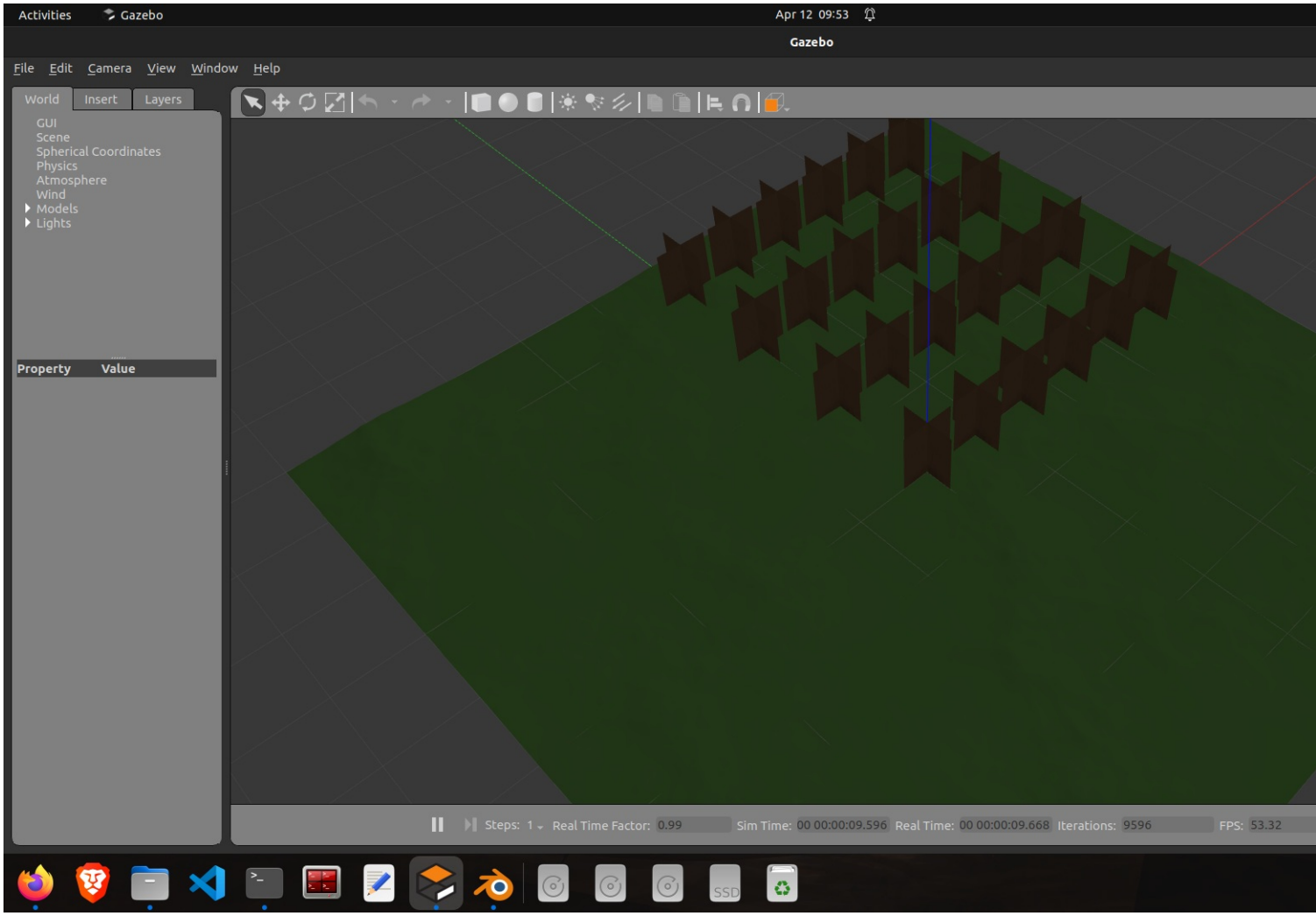
#	Action Item/ Explanation	Total Time This Week (hours)
1	Replacing the Brown ground with a Lush Green Texture and then debugging the wheat transparency	3
2	Importing and Arranging Maize Models for Dense Crop Appearance	3
3	Creating a Dense Maize Row by Manual Model Duplication in the .world File	3
4	Farmland Scene Cleanup and Maize Field Setup in Gazebo	3
5	Review of Vision-Based Navigation & Guidance for Agricultural Autonomous Vehicles & Robots	3
6	GNSS-based Localization for Autonomous Vehicles: Prospects and Challenges	3
7	Weekly Plan for Next Week	1
8	Report Writing	1
	Total hours for the week:	20

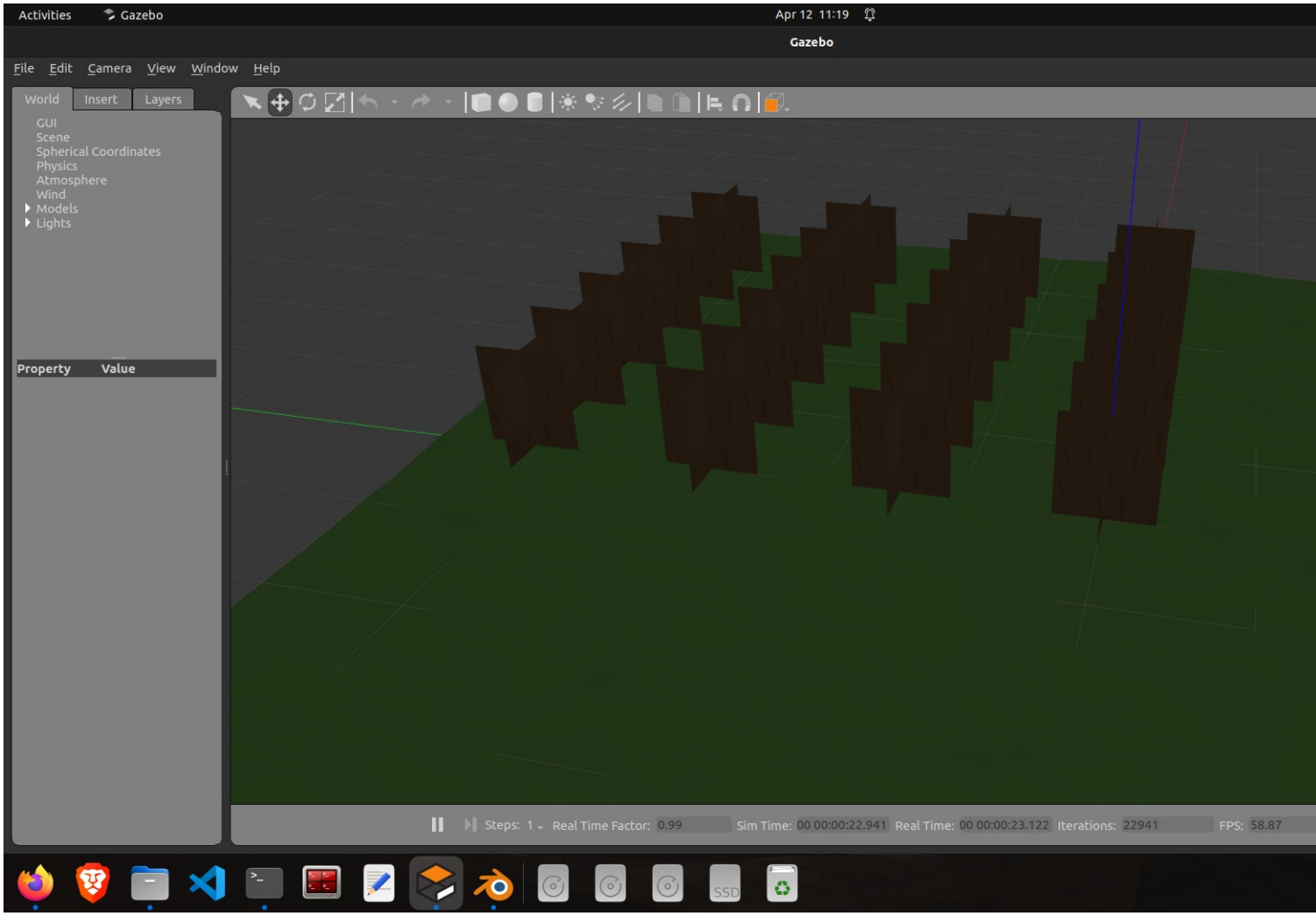
Verification Documentation:

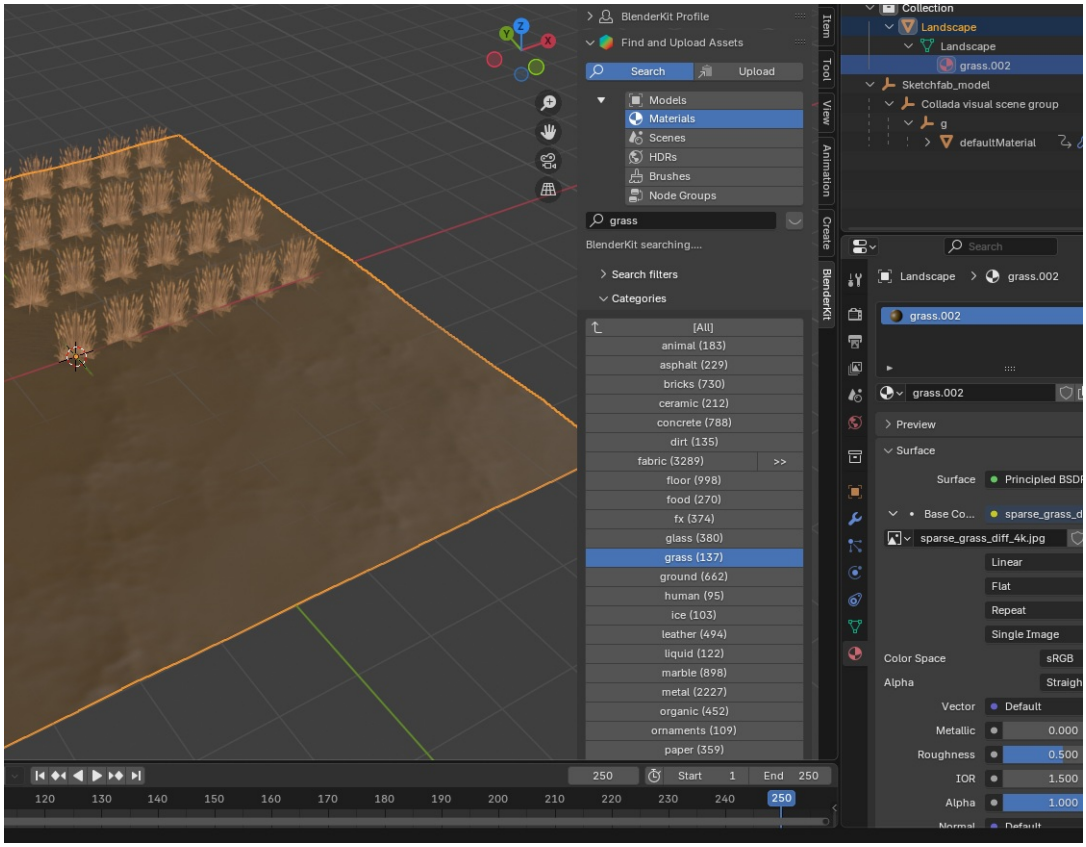
Action Item 1: Replacing the Brown ground with a Lush Green Texture and then debugging the wheat transparency – 3 hour(s).

Project Work Summary

- Began this week by searching for a rich green texture that could give the farmland ground a more natural and green appearance. After browsing through a few resources, I downloaded a realistic green ground texture from Poly Haven. It had the right tone and surface detail for what I had in mind.
- Then tested the green texture by applying it to a basic plane in Blender to see how it interacted with the default lighting. Once it looked good, I assigned the same material to the wheat model's base mesh, hoping to enhance its visual appeal and make it appear fresher and healthier.
- After assigning the texture and confirming the UV map looked correct, I exported the model as a .dae file and placed it inside the Gazebo model folder. I updated the model's .sdf to ensure it pointed to the right file and then launched Gazebo to see how it appeared.
- The wheat model loaded without issues but the wheat was still filled in with the same dull brown color as before. The transparent texture was not visible in the gazebo when imported from the blender.
- I tried several workarounds: changing the material names in Blender, checking the UV layout again, even testing a transparency shader. I applied transforms and normals, repacked textures, and verified file paths. Still, no visible change occurred when loading the model in Gazebo.
- At one point, I suspected it might be a limitation in how Gazebo reads material settings from .dae files exported by Blender. So, I reduced the model to just a simple mesh, applied the same green texture, and re-exported it—but the result was the same.
- Gazebo wasn't parsing the texture or material information as intended. So for now, I left the brown texture correctly mapped in Blender and began researching for any other plants available in the Gazebo-compatible material workflows to try again with more control over appearance in simulation.

- 
Activities
Gazebo
Apr 12 09:53
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: 0.99 Sim Time: 00 00:00:09.596 Real Time: 00 00:00:09.668 Iterations: 9596 FPS: 53.32
SSD

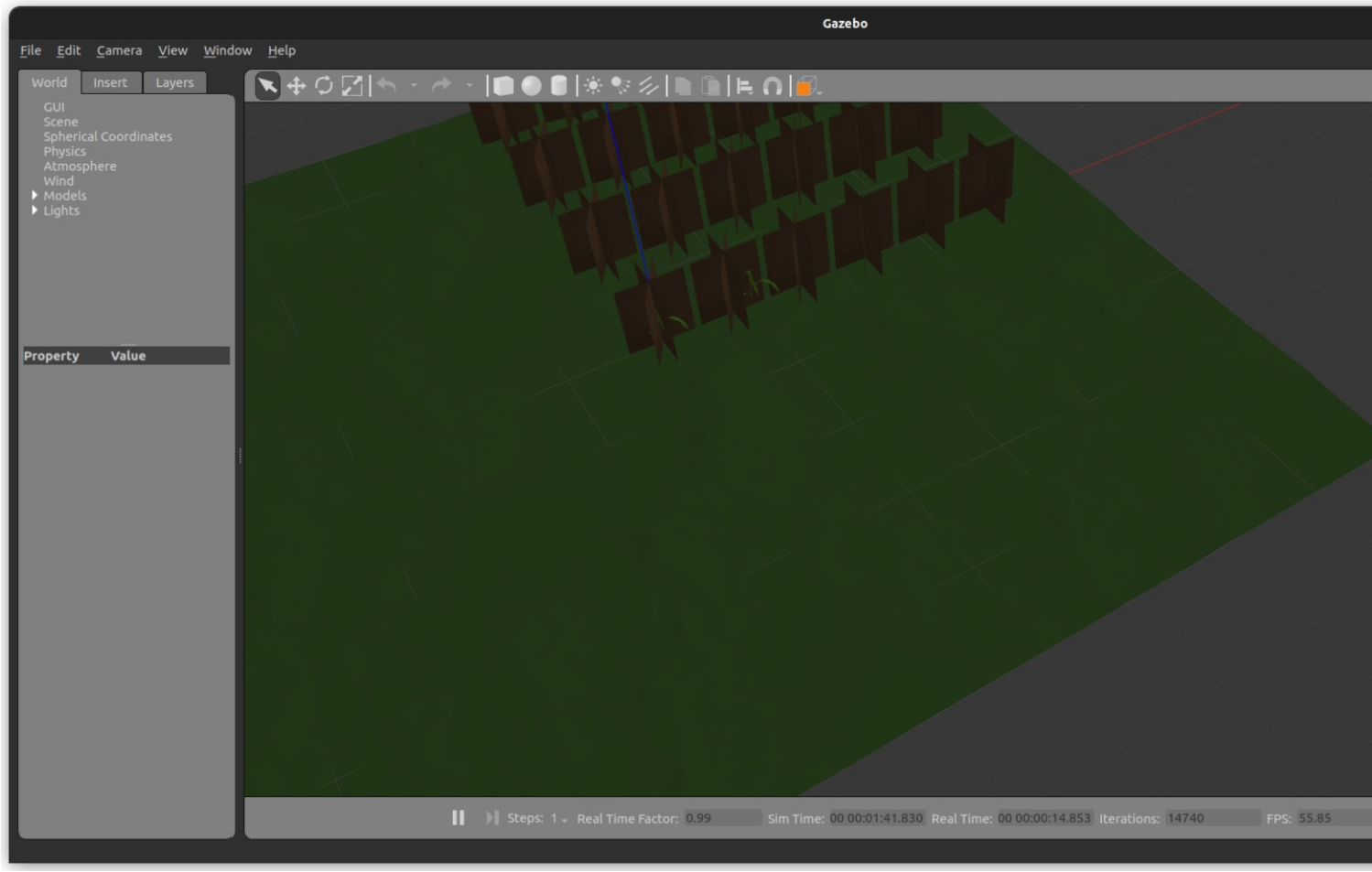
- 

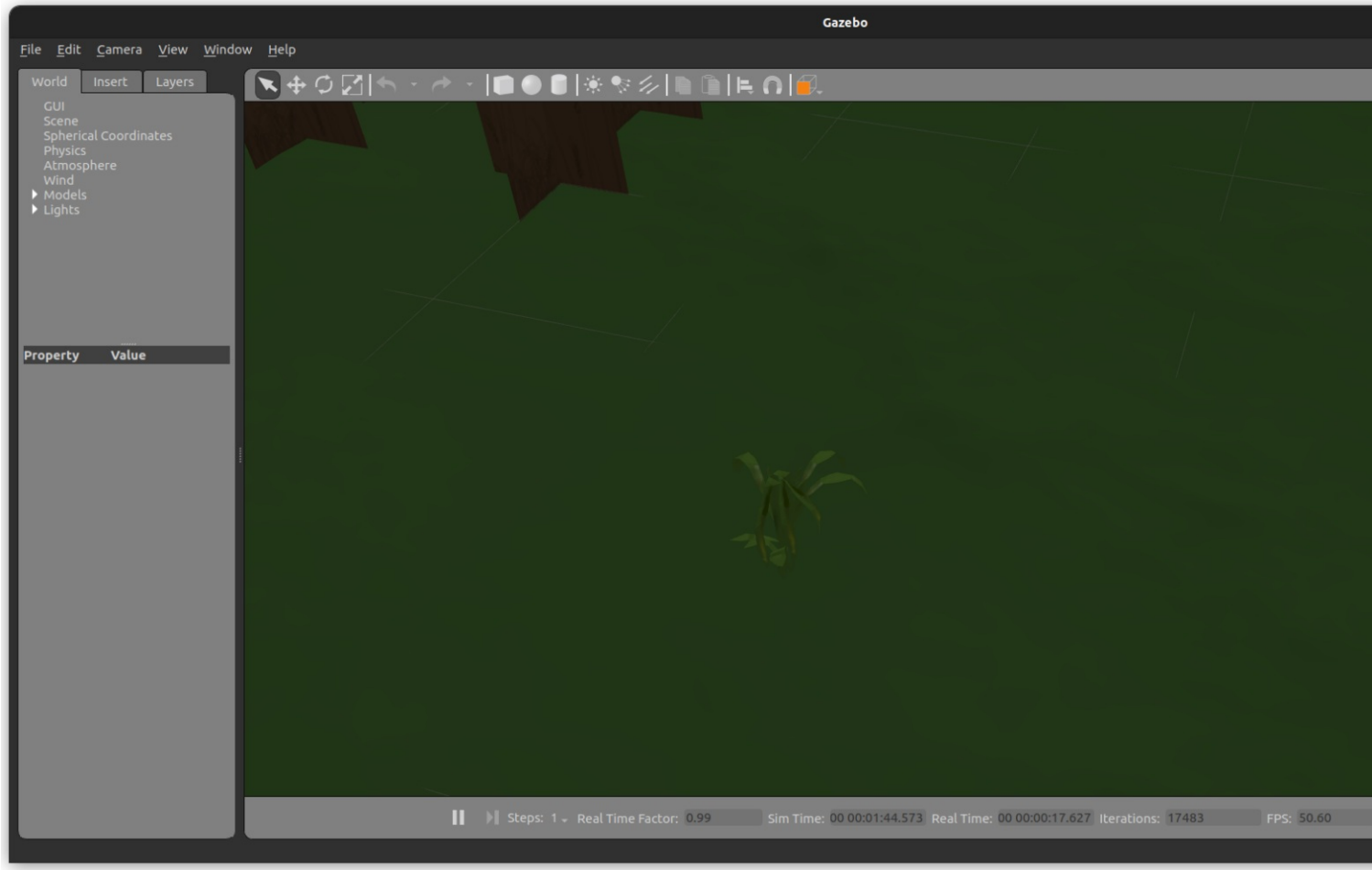
- 

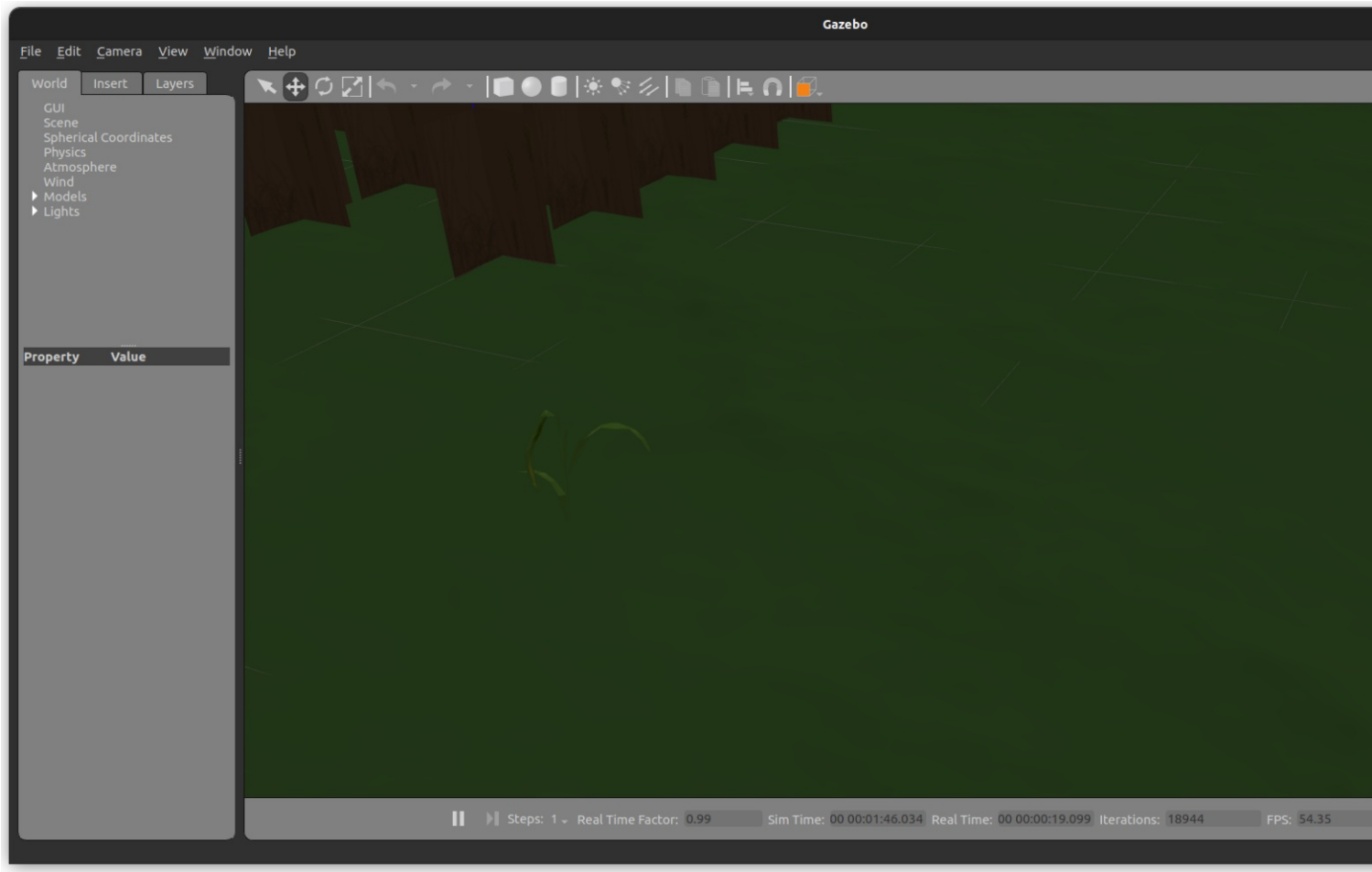
Action Item 2: Importing and Arranging Maize Models for Dense Crop Appearance – 3 hour(s).

Project Work Summary

- After setting up the base environment, I moved on to integrating maize plants into the farmland simulation. I browsed through Gazebo's built-in model library and found a maize plant model that looked detailed enough for our purposes.
- I opened the existing map using the already configured .world file to make sure all features—terrain, lighting, and previous elements—were intact. Once confirmed, I began adding the maize models directly within the Gazebo interface.
- I dragged and dropped the maize model onto the terrain. To give the crops a denser and more realistic planting pattern, I added a second maize plant at the exact same coordinates, but rotated it by π/2 radians (90 degrees). This helped offset the structure just enough to make it appear more naturally overlapping and lush.
- After placing both models, I selected them and ensured everything looked aligned and balanced from different angles in the 3D view. This double-stacking trick gave a fuller crop appearance without needing to alter the model itself.
- With the positioning done, I exported the updated environment as a new .world file. This saved all the added maize models and their orientations in place.
- I then reopened the newly exported .world file in Gazebo to verify if the maize models were properly saved and loaded. Now both instances showed up exactly as arranged, confirming that the export preserved the new additions.



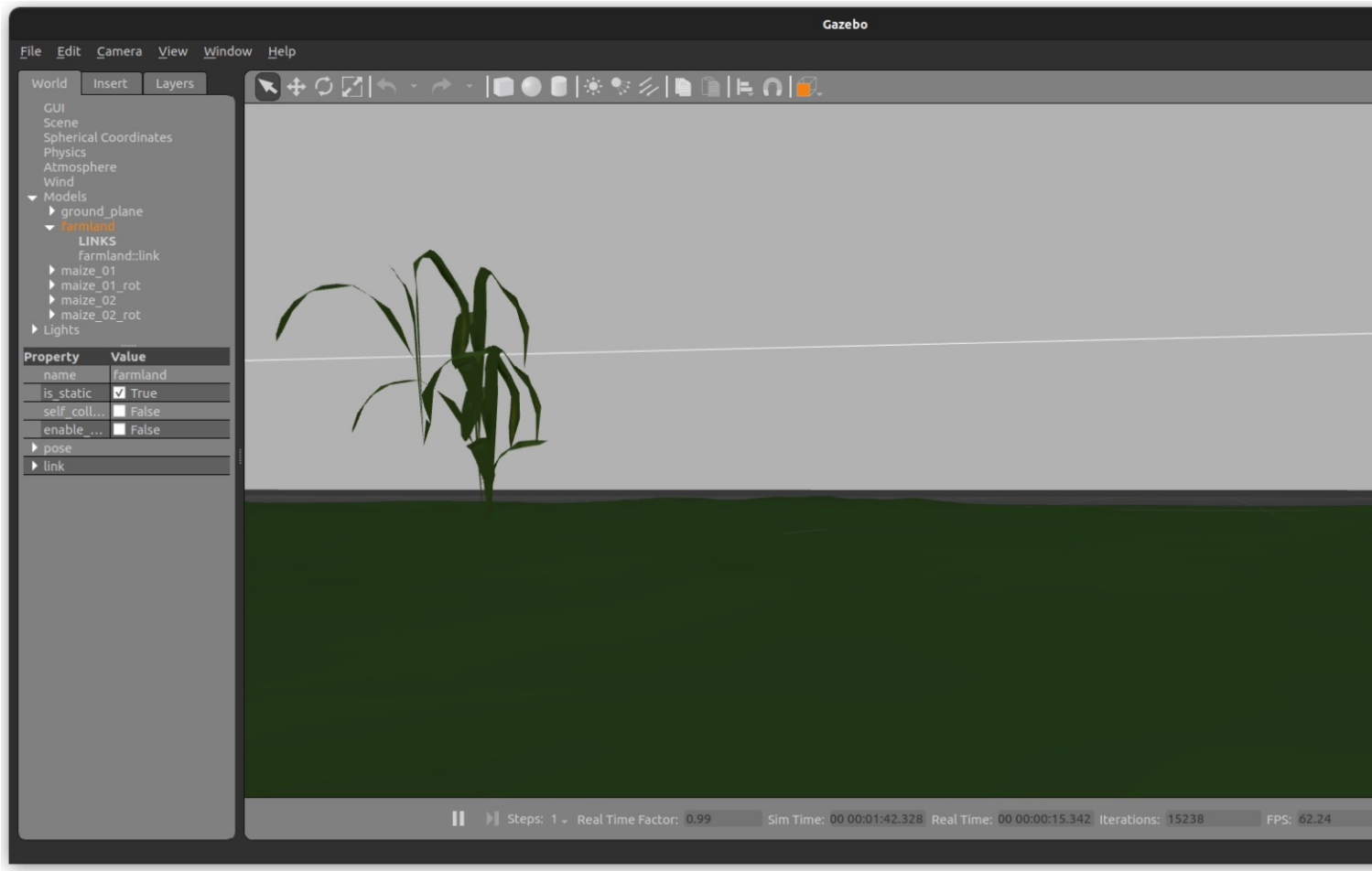


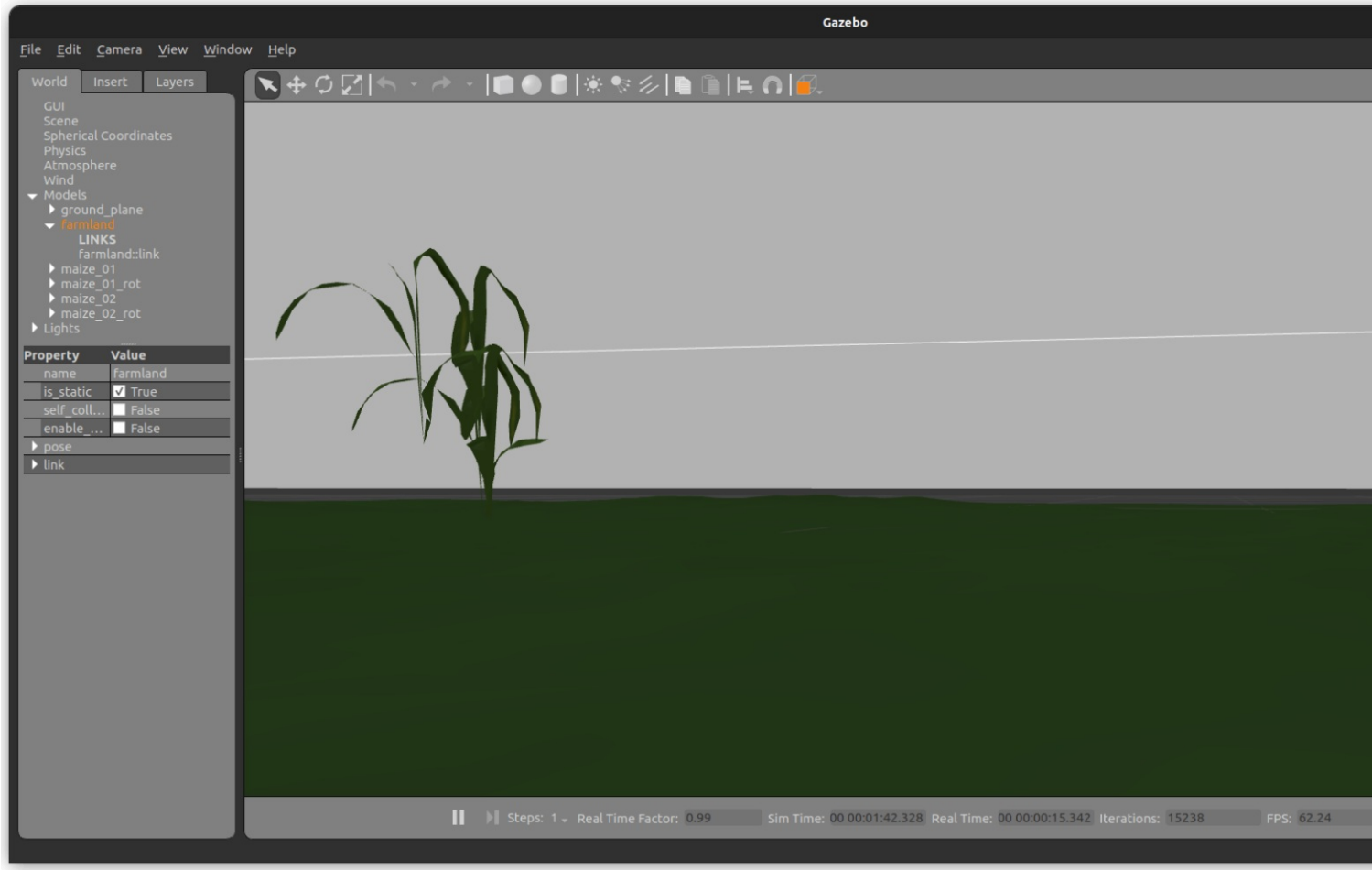


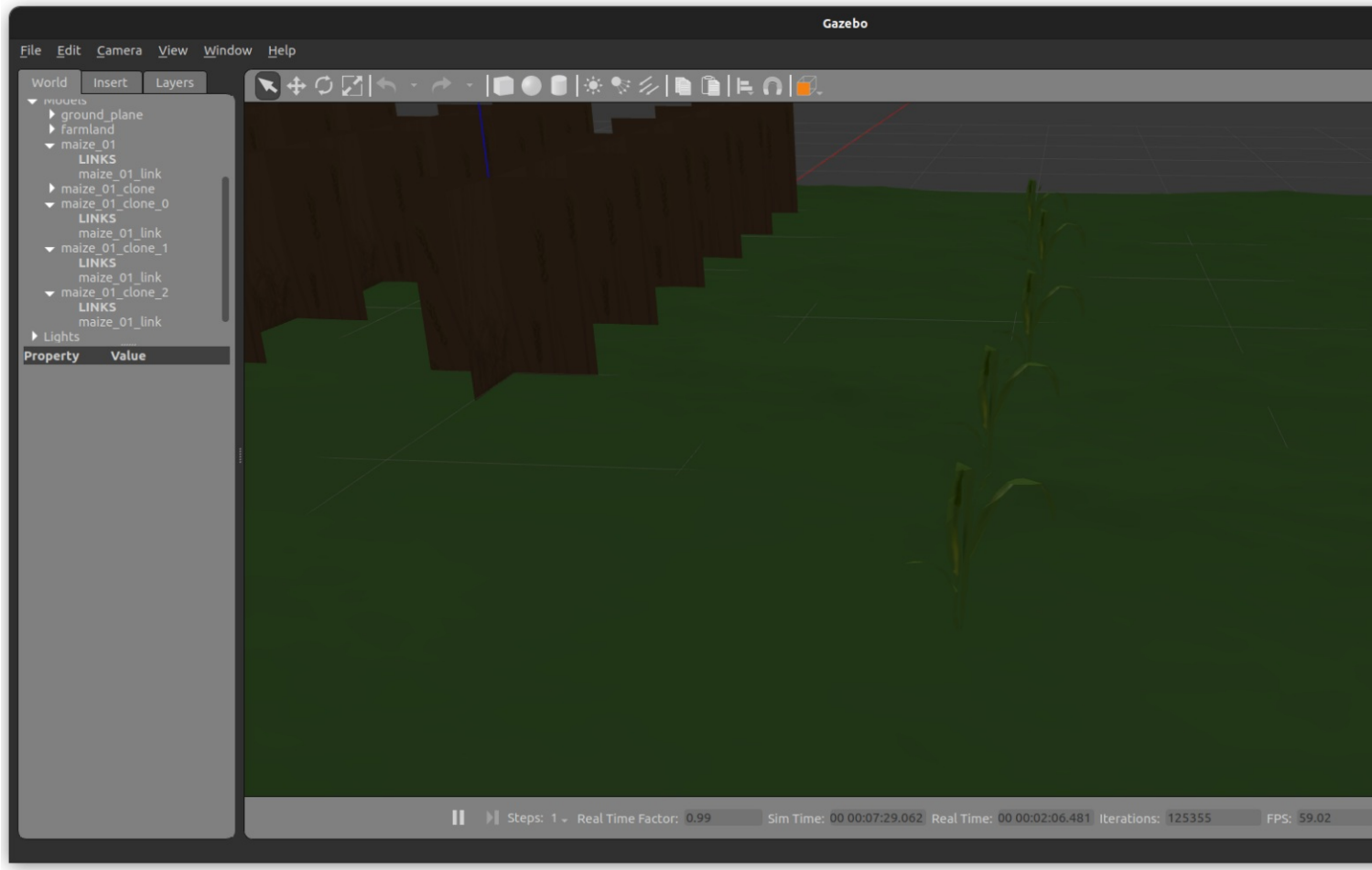
Action Item 3: Creating a Dense Maize Row by Manual Model Duplication in the .world File – 3 hour(s).

Project Work Summary

- The main motivation for this task was to bypass the issues we had encountered with transparency and mesh conversion when importing custom plant models from Blender to Gazebo. Since these problems were likely rooted in how Gazebo handled Blender's transparent textures, I figured using Gazebo's built-in maize model directly would eliminate those rendering glitches.
- After successfully adding the maize model into the simulation through the Gazebo model library, I navigated to the .world file to inspect how the maize model was represented in XML. I located the relevant <model> definitions and confirmed that the maize model and its rotated version were correctly referenced along with their meshes, poses, and collision parameters.
- To simulate a realistic row of maize plants, I planned to manually duplicate the model definitions within the .world file. I started with the existing two maize models—one normal and one rotated by 90 degrees (π/2 radians)—placed at the coordinate (-2, -2, 0). These served as the base structure for replication.
- I copied and pasted the entire <model> blocks and updated each one's name and pose. For the first row, I followed a pattern along the X-axis: placing them at (0, 0, 0), (-2, 0, 0), and (-4, 0, 0). I alternated between the normal and rotated maize models to maintain the dense crisscross look.
- I then repeated the process along the Y-axis to form a second row: placing models at (0, -2, 0), (-2, -2, 0), and (-4, -2, 0), again alternating the rotation to make the patch look more organic and overlapped, giving the illusion of a thick crop field.
- After updating and saving the .world file, I launched the world in Gazebo once again to confirm everything loaded correctly. To my relief, all the newly added maize plants appeared in the correct positions and orientations, forming a neat, dense 2x3 grid.
- This manual method gave us full control over placement, spacing, and orientation—plus, since these models were directly from Gazebo's own database, we had no transparency bugs or missing mesh issues to worry about.




Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
ground_plane
farmland
LINKS
farmland::link
maize_01
maize_01_rot
maize_02
maize_02_rot
Lights
Property Value
name farmland
is_static True
self_coll... False
enable_... False
pose
link
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:01:42.328 Real Time: 00 00:00:15.342 Iterations: 15238 FPS: 62.24

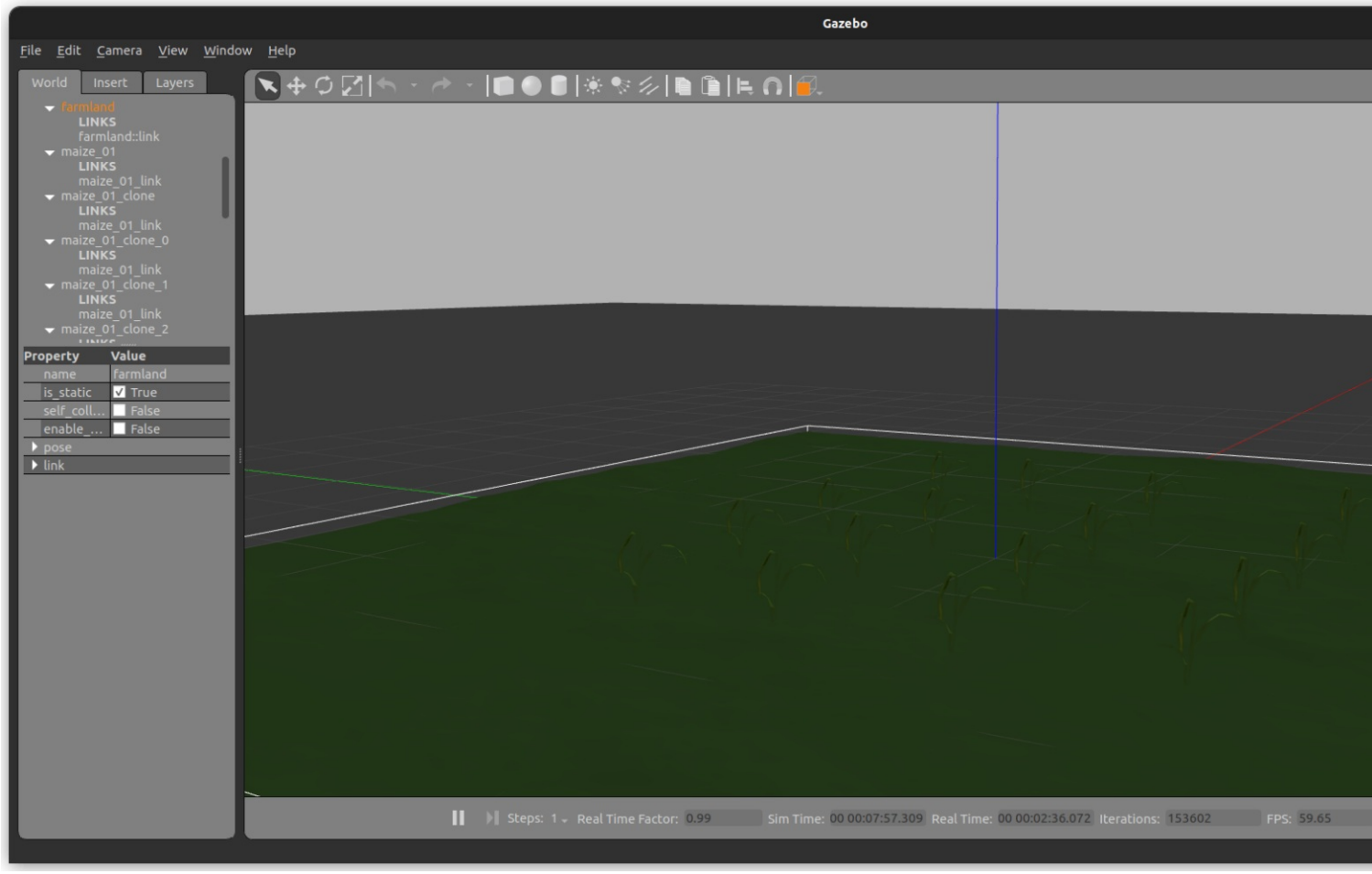

Gazebo
File Edit Camera View Window Help
World Insert Layers
ground_plane
farmland
maize_01
LINKS
maize_01_link
maize_01_clone
maize_01_clone_0
LINKS
maize_01_link
maize_01_clone_1
LINKS
maize_01_link
maize_01_clone_2
LINKS
maize_01_link
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:07:29.062 Real Time: 00 00:02:06.481 Iterations: 125355 FPS: 59.02

-
-

Action Item 4: Farmland Scene Cleanup and Maize Field Setup in Gazebo – 3 hour(s).

Project Work Summary

- With the default brown ground grass and wheat elements no longer serving the visual or functional goals of the simulation, I decided it was time for a complete refresh of the terrain. The objective was to replace the default vegetation with a cleaner layout, one that better represented a manually planted maize field and allowed for clearer robot interaction and navigation in Gazebo.
- I began by removing the existing brown ground grass that was originally part of the imported farmland. This texture had a visual clutter that didn't contribute much to the scene and often conflicted with added vegetation models, so I stripped it from the map entirely to bring visual clarity and uniformity to the simulation environment.
- Next, I focused on eliminating all instances of wheat models scattered across the scene. I ensured that every wheat mesh was deleted, leaving no residual clutter in the simulation. To validate the cleanup, I reimported the map into Gazebo and confirmed that the wheat had been successfully and completely removed.
- Once I had a clean base to work with, I introduced a new set of maize models into the environment. Instead of randomly placing them, I deliberately created a structured 4x4 grid layout to simulate an organized maize crop row — 16 plants spaced evenly to mirror real-world agricultural planting patterns.
- I paid particular attention to the positioning of each maize instance to ensure that none of them were hovering above or submerged beneath the terrain surface. This precision helped prevent physics inconsistencies and maintained the visual realism of the scene.
- After arranging the maize plants, I finalized the scene and saved the updated map configuration. Upon relaunching it in Gazebo, I was pleased to see that the new maize field integrated smoothly with the terrain, replacing the old vegetation with a neater, more simulation-friendly design.
- This setup now offers a cleaner, more purposeful testing ground for further robotics simulation and navigation tasks, with vegetation elements that are both visually realistic and technically compatible with our objectives.



Action Item 5: Review of Vision-Based Navigation & Guidance for Agricultural Autonomous Vehicles & Robots – 3 hour(s).

Research

- https://www.researchgate.net/publication/369913029_Review_of_Vision-Based_Navigation_Guidance_for_Agricultural_Autonomous_Vehicles_Robots
- Review of Vision-Based Navigation & Guidance for Agricultural Autonomous Vehicles & Robots
- Summary of Report
  - This review paper presents a detailed analysis of vision-based navigation systems in autonomous ground and aerial vehicles, emphasizing the role of camera-based perception in GPS-denied environments.
  - The authors categorize navigation approaches into monocular, stereo, RGB-D, omnidirectional, and event-based systems, outlining trade-offs in depth accuracy, computation complexity, and robustness to environmental variability.
  - The study highlights key navigation pipelines—Visual Odometry (VO), Visual Simultaneous Localization and Mapping (V-SLAM), and neural network-based scene understanding. VO and SLAM techniques are discussed with reference to feature-based and direct methods.
  - For example, ORB-SLAM and LSD-SLAM are evaluated on accuracy and resilience to scale drift. Deep learning-based models for semantic segmentation, obstacle recognition, and terrain classification are also reviewed.
  - In addition to technical architecture, the paper addresses calibration challenges, sensor fusion (e.g., with IMU), and visual localization under adverse conditions like poor lighting, motion blur, and foliage occlusion. Multiple datasets (e.g., KITTI, TUM RGB-D, EuRoC MAV) are referenced, offering benchmarking options for researchers.
  - A notable aspect is the treatment of real-time processing constraints. The authors differentiate between CPU-bound and GPU-accelerated pipelines, noting that most state-of-the-art SLAM systems run at ~10–30 FPS on embedded GPUs. Recommendations for hardware and algorithmic trade-offs in real-world deployments round out the review.
- Relation to Project
  - The surveyed methods directly inform visual localization strategies for agricultural robotics, especially in GPS-intermittent or occlusion-heavy conditions like crop canopies. The coverage of stereo and RGB-D sensors provides insights into 3D mapping approaches applicable to structured farmland, where depth estimation is crucial for row-following and ditch avoidance.
  - Techniques like ORB-SLAM2 and DSO (Direct Sparse Odometry) are viable candidates for initial SLAM modules. Additionally, the paper's evaluation of deep learning for terrain classification aligns with our objective to semantically distinguish between cultivable land, irrigation paths, and field boundaries.
  - From a system design perspective, the modularity described (e.g., integrating VO with IMU in ROS) mirrors our intended implementation. The dataset references are useful for pre-training before deployment in custom Gazebo worlds.
- Motivation for Research
  - Agricultural robots must operate reliably where GNSS signals degrade due to tree cover or signal multipath. Vision-based SLAM offers an independent and complementary localization stream.
  - This paper's exhaustive review provides a toolkit for selecting, integrating, and optimizing vision systems within a ROS2-based navigation stack.
  - Moreover, the emphasis on annotated datasets and the performance of SLAM algorithms in natural scenes (vs. urban ones) validates the need for field-specific tuning.

- The article encourages a hybrid localization strategy combining visual and inertial cues—exactly the type of fusion that could enhance our robot's performance in open farmland and orchard environments.

Action Item 6: GNSS-based Localization for Autonomous Vehicles: Prospects and Challenges – 3 hour(s).

Research

- GNSS-based Localization for Autonomous Vehicles: Prospects and Challenges
- https://www.researchgate.net/publication/337656003_GNSS-based_Localization_for_Autonomous_Vehicles_Prospects_and_Challenges
- Summary of Report
  - This survey compiles advances in GNSS-based localization for autonomous ground vehicles, with a focus on precision, redundancy, and integration with inertial and visual sensors. It classifies GNSS implementations into standard positioning services (SPS), Real-Time Kinematic (RTK), and Precise Point Positioning (PPP), detailing their accuracy ranges and latency profiles.
  - Sensor fusion strategies dominate the discussion, particularly the use of Extended Kalman Filters (EKF) and Unscented Kalman Filters (UKF) to merge GNSS data with IMU and wheel odometry.
  - Specific mention is made of open-source frameworks like Robot Operating System (ROS), where packages like robot_localization and navsat_transform_node simplify data fusion and coordinate transformations.
  - The review identifies key limitations of GNSS: multipath distortion near trees or buildings, signal dropouts, and the lack of fine-scale localization required for tasks like row navigation or docking. To counter this, multi-sensor fusion (GNSS + IMU + VO) and post-processing techniques like smoothing or map-based corrections are discussed.
  - The paper also includes case studies where GNSS localization was deployed in agricultural, urban, and off-road settings. In these, systems with fused GNSS/IMU achieved ~10 cm accuracy using RTK, with failure cases analyzed (e.g., wheel slip skewing odometry or IMU drift during signal outages).
- Relation to Project
  - The discussion on GNSS-IMU fusion directly aligns with our project, where consistent global localization is necessary for autonomous row-following and boundary detection.
  - The paper's breakdown of ROS2 integration (via EKF and transform publishers) mirrors our middleware choice, providing a reliable framework for multi-sensor data management.
  - Additionally, the reference to transforming GPS data to robot-local frames using UTM conversion or base_link alignment supports our planned implementation in farmland terrain, where real-time location must be visualized and adjusted relative to waypoints.
  - The RTK-GNSS configuration examples and challenges in agricultural fields (e.g., signal noise from vegetation) validate our approach to simulate and test GNSS modules in Gazebo prior to field trials.
- Motivation for Research
  - Reliable navigation in agriculture hinges on continuous localization in vast, repetitive, and occlusion-prone environments. This survey's insights into sensor fusion and GNSS reliability offer a roadmap for robust system design under real-world constraints.
  - By highlighting ROS-native solutions and practical fusion pipelines, the paper empowers researchers like us to accelerate development and troubleshooting.
  - It motivates the need for simulation validation (e.g., using synthetic GNSS topics in Gazebo) and the creation of fallback strategies during GNSS failures, such as VO-based dead reckoning.
  - This makes the article not only relevant but essential for structuring a resilient localization pipeline for our robot under ROS2 Humble.

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

Project Work Summary

- We will begin by installing the vision_msgs and image_pipeline ROS 2 Humble packages to process RGB-D data from a simulated Realsense D435 camera. Using the realsense2_camera ROS 2 wrapper, I'll spawn the camera on the Husky robot's front URDF mount, verifying /camera/color/image_raw and /camera/depth/image_rect_raw topics publish expected outputs when launching in a flat farmland world.
- To validate the vision pipeline,we can then run a ROS 2 node for visual odometry using viso2_ros, remapping image topics accordingly. If the estimated trajectory jitters due to lighting artifacts or noisy depth maps, I'll apply bilateral filtering and auto-exposure constraints in the camera parameters file. A fallback approach involves switching to ORB-SLAM3 for a more robust feature-based VO pipeline.
- Then fuse visual odometry with GNSS and IMU data using robot_localization's ekf_filter_node.
- Initially, we can assign low covariance to visual odometry in X-Y ([0.05, 0.05]) but reduce its influence if feature tracking degrades in uniform vegetation. The IMU (/imu/data) and GNSS (/navsat/fix) will be prioritized when the robot drives through visually ambiguous zones like crop rows.
- For field navigation, I'll create a synthetic GNSS trajectory mimicking an orchard grid layout using QGIS and export it as a CSV. Using geographic_msgs and navsat_transform_node, I'll convert this to UTM-based map coordinates.

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.