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.

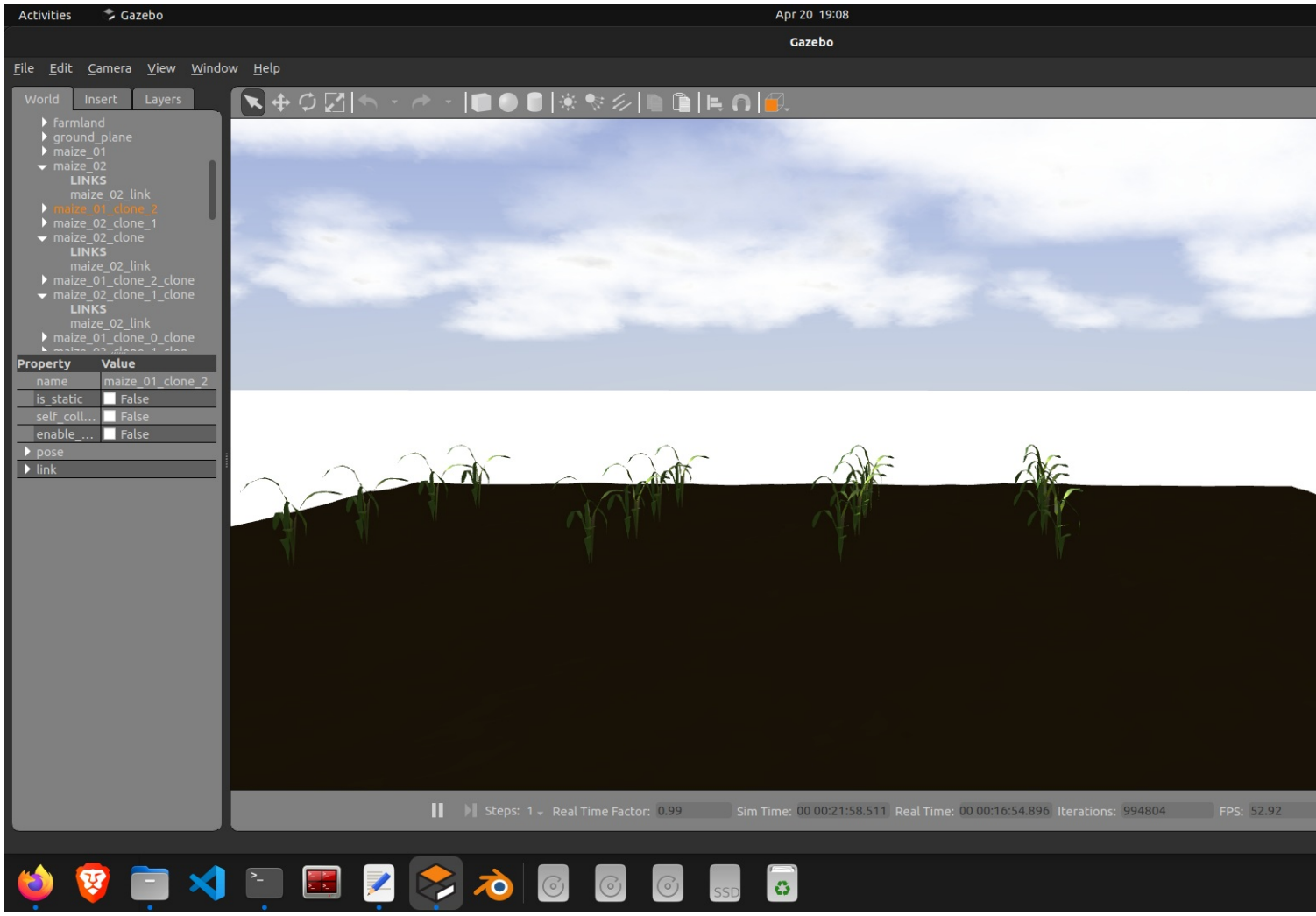
#	Action Item/ Explanation	Total Time This Week (hours)
1	Preparing the Base Farmland Map Using Blender and Gazebo	3
2	Enhancing Farmland Visual Odometry Compatibility with Structural	3
3	GNSS-Stereo-Inertial SLAM for Arable Farming	3
4	Integration of the Husky Model and final enhancements in the gazebo map	3
5	Camera Integration into Husky and Debugging the Camera Feed in ROS2-Gazebo	3
6	Aligning the Husky with the Crop Row and Capturing Visual Data for VO	3
7	Preparing for Sensor Fusion and Navigation Integration with GNSS and Visual Odometry	1
8	Report Writing	1
	Total hours for the week:	20

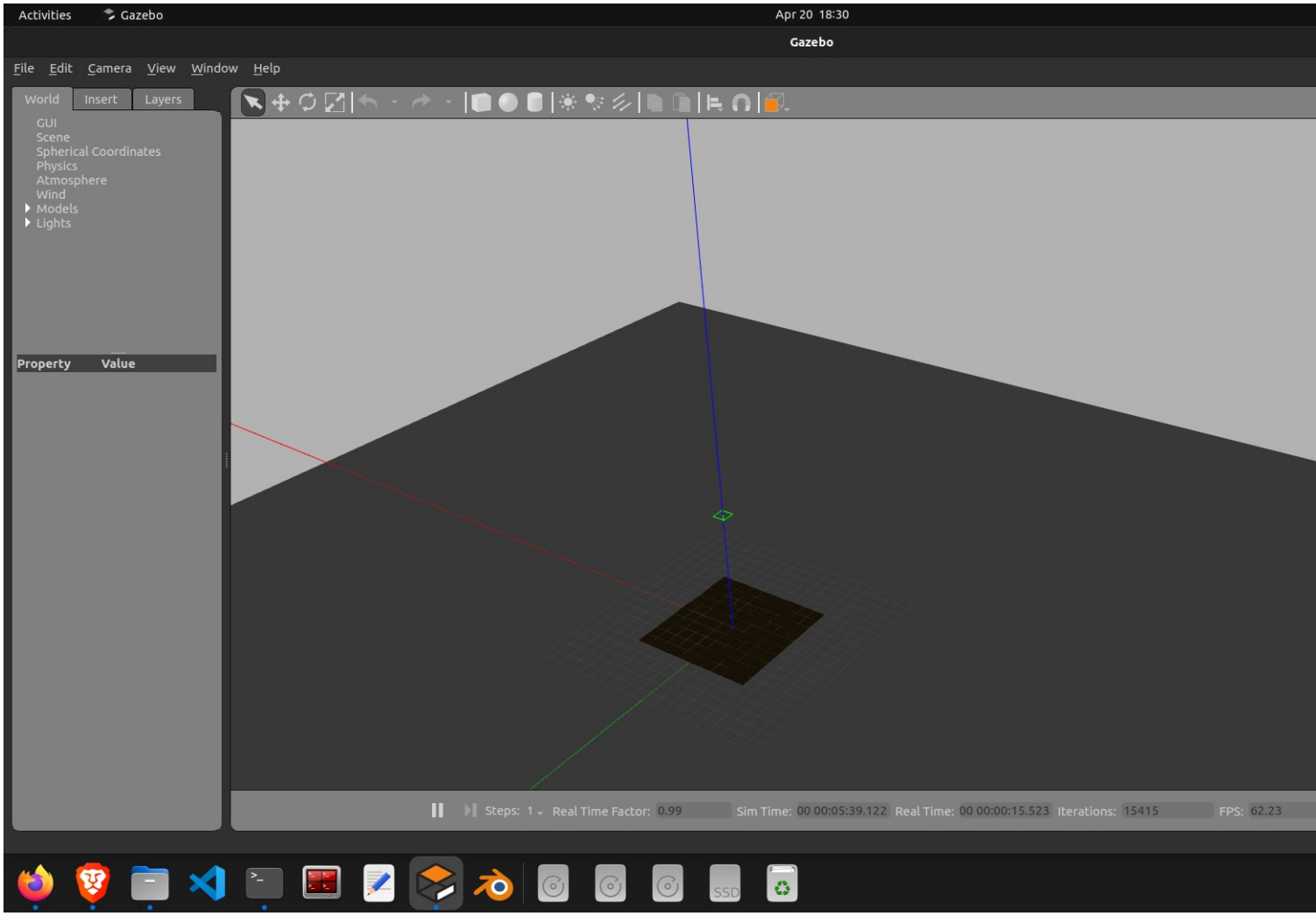
Verification Documentation:

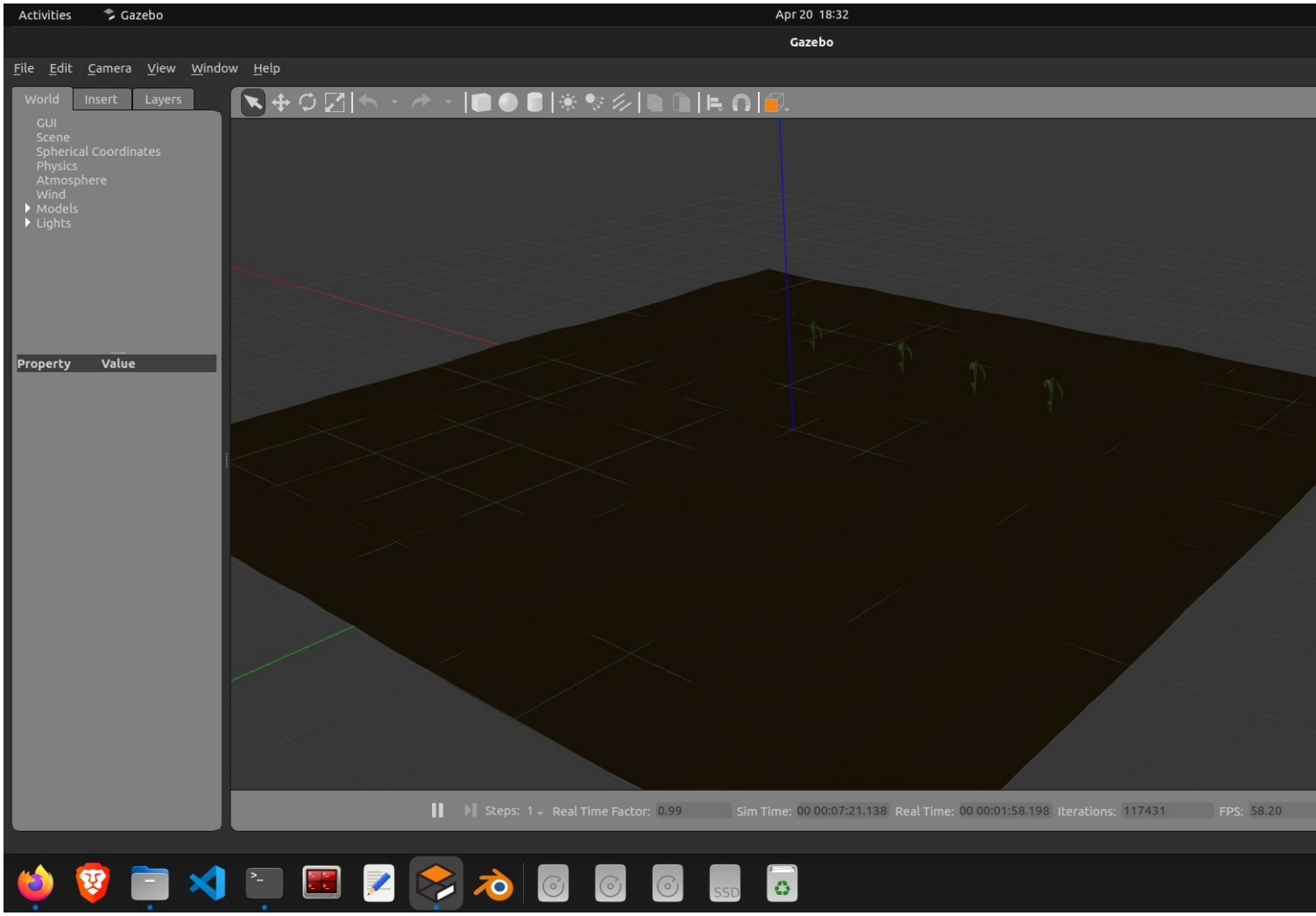
Action Item 1: Preparing the Base Farmland Map Using Blender and Gazebo – 3 hour(s).

Project Work Summary

- I started by focusing on the visual and structural foundation of our simulated farmland. The goal was to create a clean, realistic environment for future use in robotics simulation and visual odometry testing. The first step involved opening Blender and importing our base terrain.
- I had a grass-textured plane that needed more control and realism, so I decided to replace the default grey floor of Gazebo with a custom terrain exported directly from Blender.
- In Blender, I applied a brown soil-like material to the plane, favoring high contrast over visual complexity, which is important for VO systems. I took care to adjust UV mapping for a clean texture layout, using the Smart UV Project to optimize how the texture wrapped across the geometry.
- After confirming the visual quality in Blender's viewport, I exported the object as a .dae file, making it compatible with Gazebo's mesh import system.
- With the model ready, I moved over to the .world file in Gazebo and removed the default ground plane model entirely. Since this could break physics by causing objects to fall indefinitely, I replaced it with a custom invisible ground: a static collision plane with fully transparent visual material. This preserved physics without interfering with the visual layout of the scene.
- Next, I wanted to enhance the sky environment to better simulate an open-air farm. I edited the <scene> tag to include a light blue sky background and enabled cloud rendering using Gazebo's built-in <sky> and <clouds> features. I then added the standard sun model using <include uri="model://sun"/>, which cast realistic shadows and improved depth cues for monocular VO.
- I noticed that the default GUI overlays like the axis and grid lines were distracting, especially for visual testing. Since these couldn't be removed directly from the .world file, I toggled them off manually through the Gazebo GUI under the "View" menu.
- To complete the scene, I arranged equidistant rows of maize crops across the soil. I ensured spacing consistency by using Gazebo's clone and snap-to-grid tools. This not only created a uniform crop layout but also provided a repeatable structure that would later support path planning and row-following tasks. Once finalized, I saved the entire setup as a clean .world file, creating a solid base to build the rest of the simulation stack on.


Activities
Gazebo
Apr 20 19:08
Gazebo
File Edit Camera View Window Help
World Insert Layers
farmland
ground_plane
maize_01
maize_02
LINKS
maize_02_link
maize_01_clone_2
maize_02_clone_1
maize_02_clone
LINKS
maize_02_link
maize_01_clone_2_clone
maize_02_clone_1_clone
LINKS
maize_02_link
maize_01_clone_0_clone
Property Value
name maize_01_clone_2
is_static False
self_coll... False
enable_... False
pose
link
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:21:58.511 Real Time: 00 00:16:54.896 Iterations: 994804 FPS: 52.92
SSD

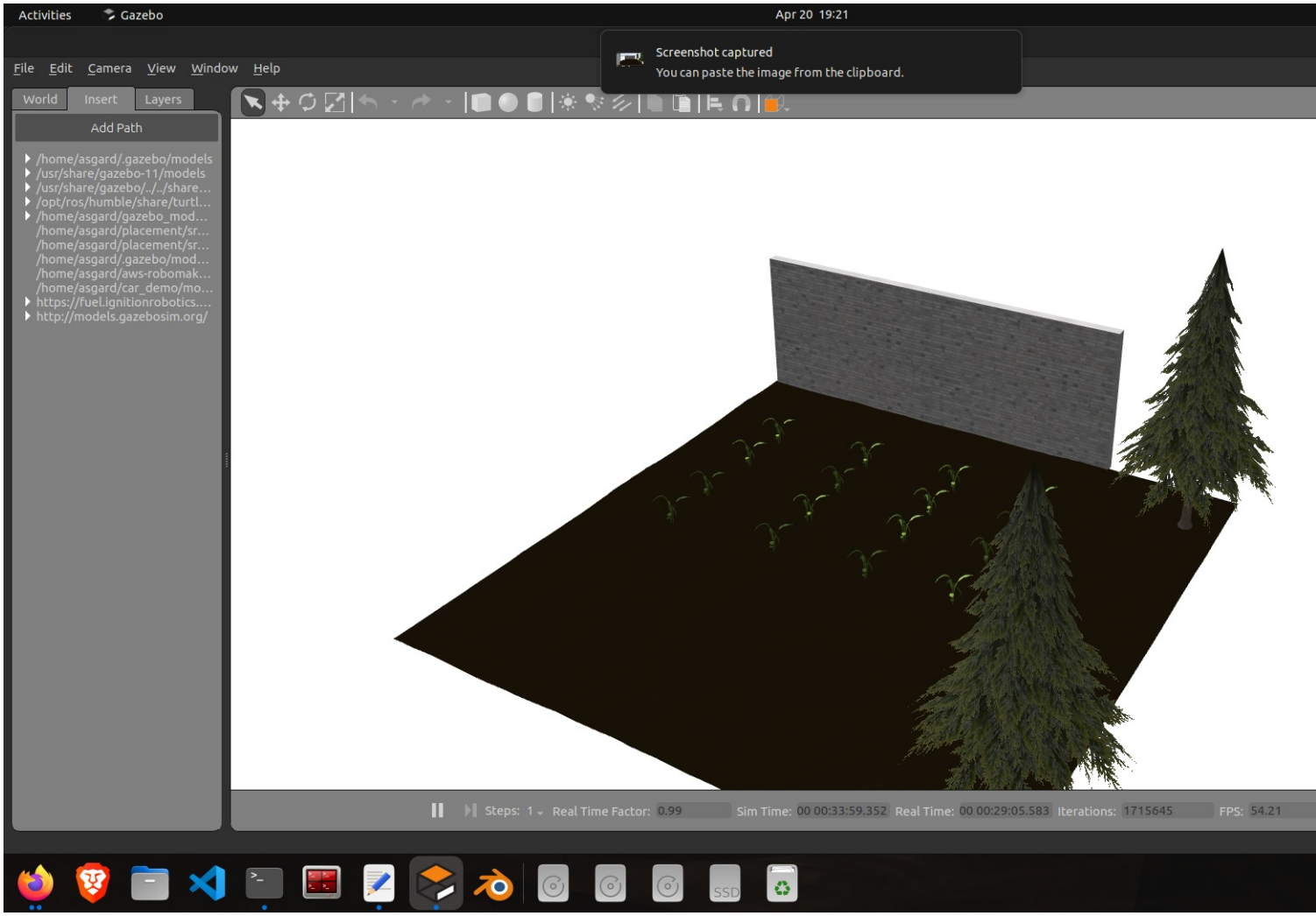

Activities
Gazebo
Apr 20 18:30
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:05:39.122 Real Time: 00 00:00:15.523 Iterations: 15415 FPS: 62.23

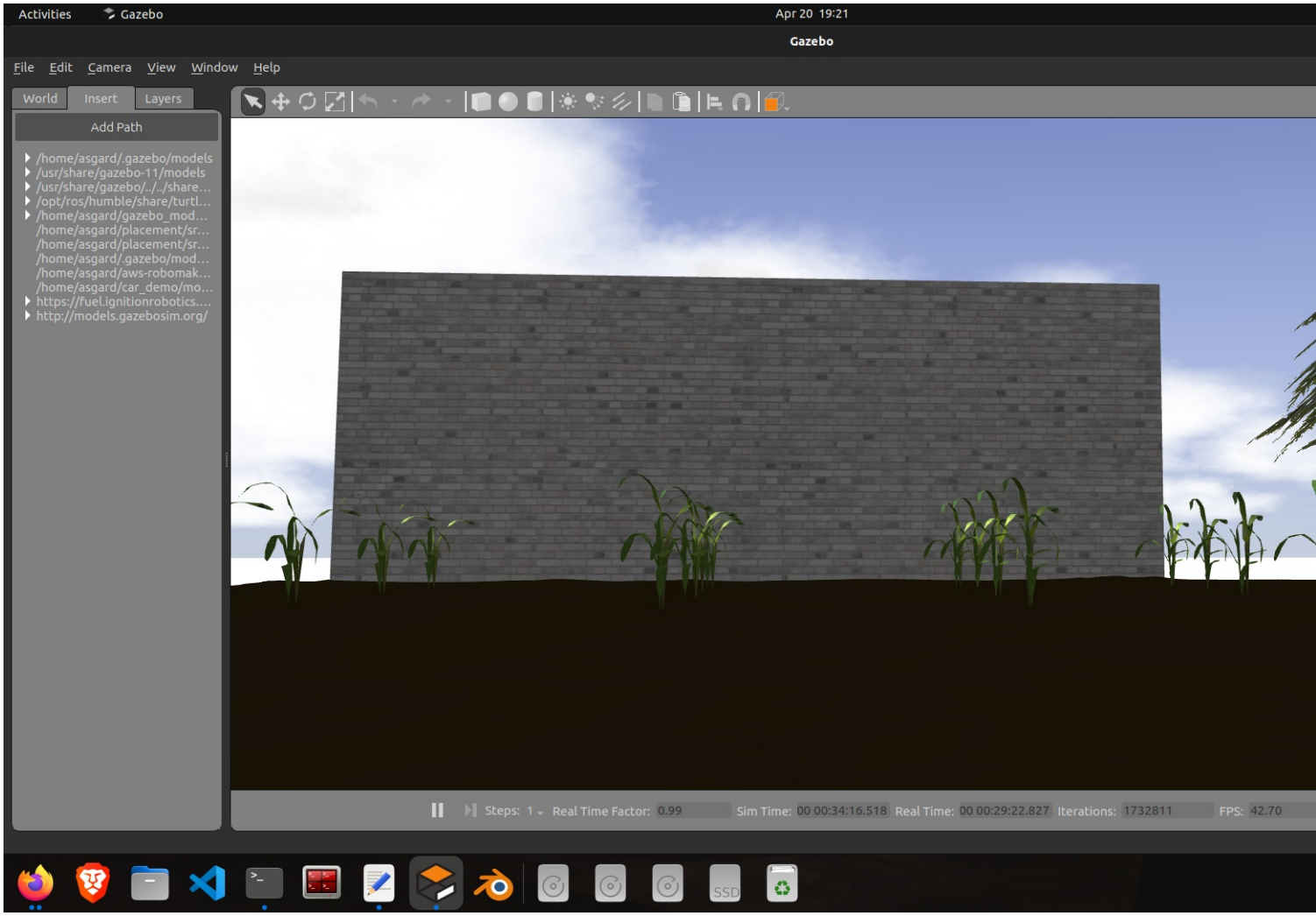


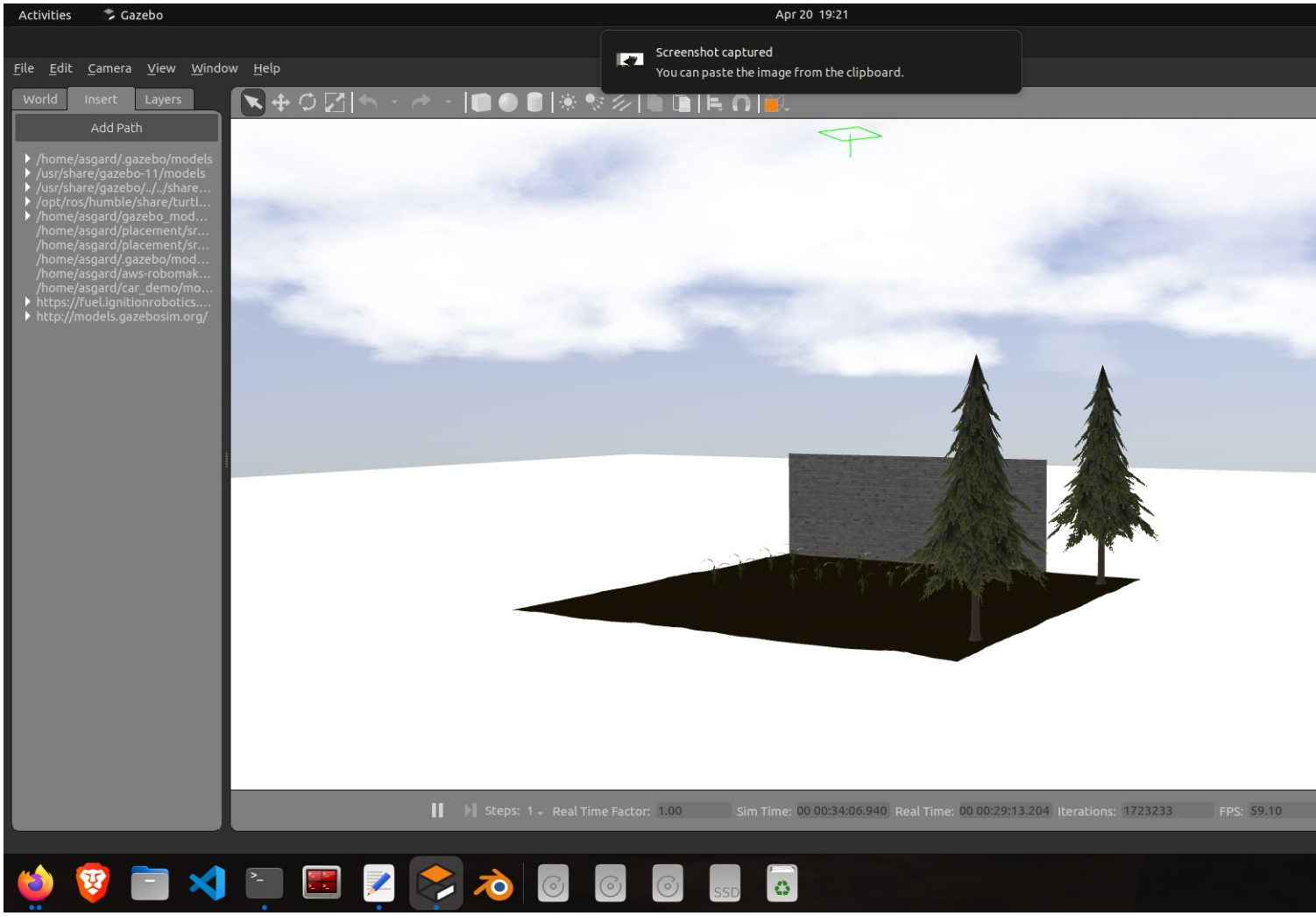
Action Item 2: Enhancing Farmland Visual Odometry Compatibility with Structural – 3 hour(s).

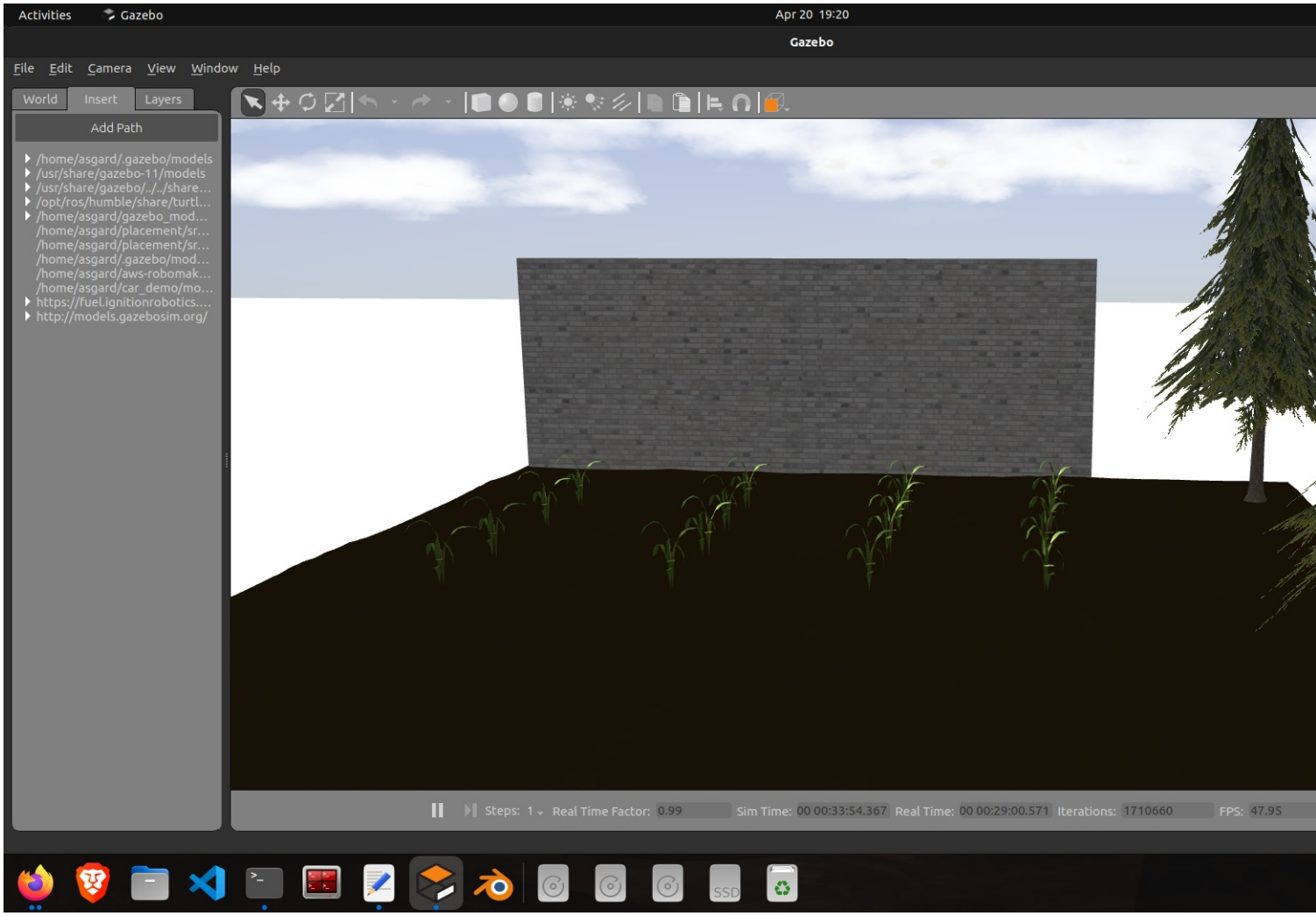
Project Work Summary

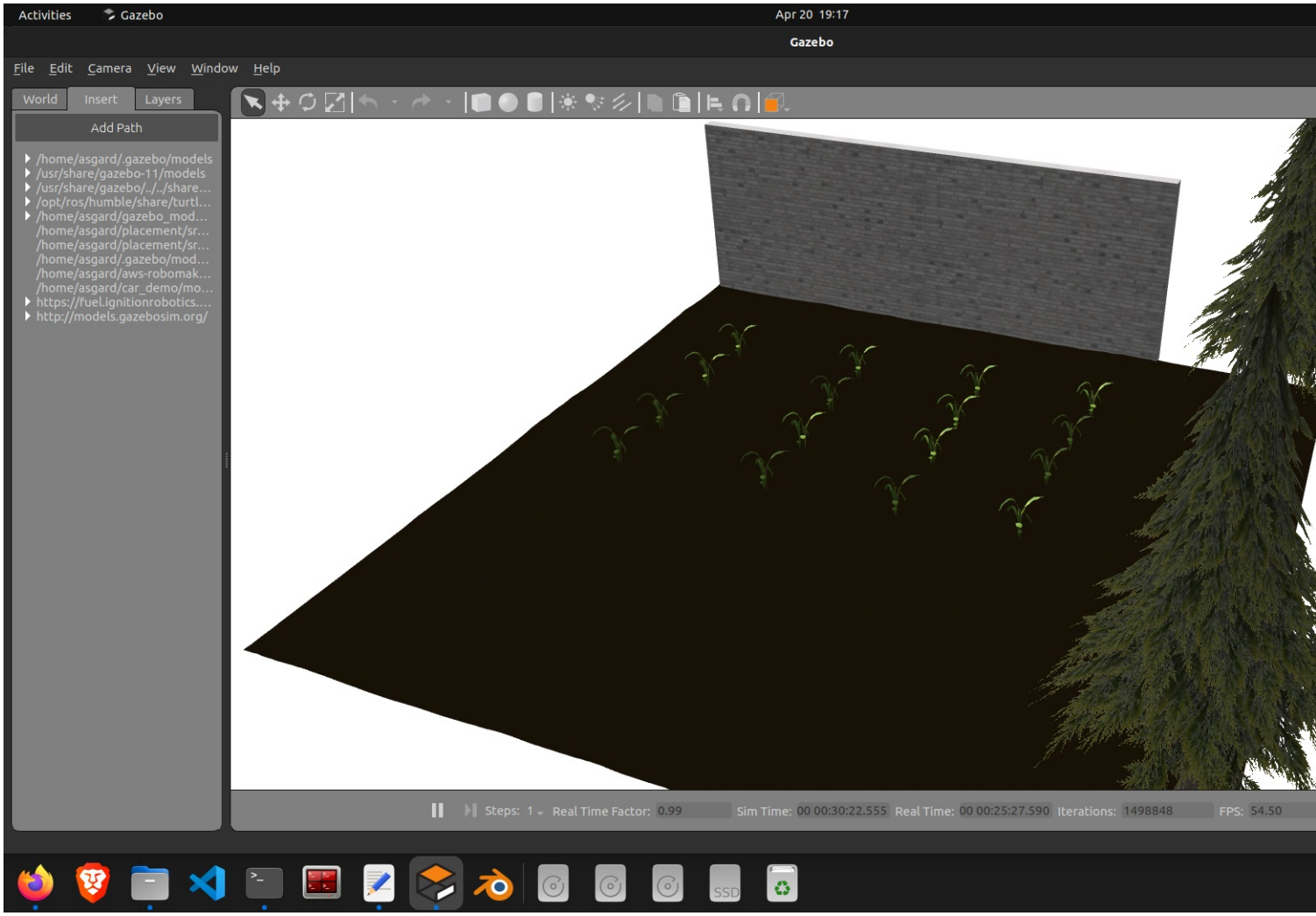
- After finalizing the soil, sky, and crop layout, I shifted my focus to optimizing the map specifically for monocular visual odometry (VO). While the base terrain worked well, monocular VO algorithms thrive on visual richness and structural cues, especially vertical features and high-contrast boundaries. I then began brainstorming which elements in a real-world field could offer this — and trees, walls, and landmarks immediately came to mind.
- I started by testing an oak tree model. Its wide, uneven branches seemed perfect at first, but when loaded into Gazebo, the sheer scale and volume of its mesh overwhelmed the map. The oak's shadowing and leaf density introduced more clutter than helpful feature points. After evaluating the visual results and checking the simulation frame rate, I decided to retire the oak and look for a more efficient alternative.
- That's when the pine tree model caught my attention. It had a tall, slim profile with layered branches and consistent geometry — ideal for providing vertical structure without overcomplicating the visual scene. I positioned it strategically along one corner of the farm where it wouldn't interfere with navigation paths but would be highly visible to the camera.
- Next, I introduced a stone wall to act as a field boundary. Its brick texture created sharp horizontal and vertical edges that offered ideal contrast and geometry for feature tracking. I placed the wall along one edge of the map, anchoring the crop rows visually and giving the VO system a stable reference in one direction of travel.
- With these additions, the farmland gained a meaningful 3D structure — one that VO algorithms could lock onto over time. I carefully checked their alignment and scaling, ensuring the tree and wall didn't intersect with the maize crops or dominate the field of view. The pine tree, in particular, balanced environmental realism and simulation performance more effectively than the oak ever could.
- Finally, I saved the new layout as an updated `.world` file, preserving the placement of all visual structures. The map was now not only visually consistent but also enriched with features that would significantly improve the performance of localization algorithms relying on monocular input.


Activities
Gazebo
Apr 20 19:21
Screenshot captured
You can paste the image from the clipboard.
File Edit Camera View Window Help
World Insert Layers
Add Path
/home/asgard/.gazebo/models
/usr/share/gazebo-11/models
/usr/share/gazebo/../../share...
/opt/ros/humble/share/turtl...
/home/asgard/gazebo_mod...
/home/asgard/placement/sr...
/home/asgard/placement/sr...
/home/asgard/.gazebo/mod...
/home/asgard/aws-robomak...
/home/asgard/car_demo/mo...
https://fuel.ignitionrobotics....
http://models.gazebosim.org/
Steps: 1
Real Time Factor: 0.99
Sim Time: 00 00:33:59.352
Real Time: 00 00:29:05.583
Iterations: 1715645
FPS: 54.21
SSD


Activities
Gazebo
Apr 20 19:21
Gazebo
File Edit Camera View Window Help
World Insert Layers
Add Path
/home/asgard/.gazebo/models
/usr/share/gazebo-11/models
/usr/share/gazebo/../../share...
/opt/ros/humble/share/turtl...
/home/asgard/gazebo_mod...
/home/asgard/placement/sr...
/home/asgard/placement/sr...
/home/asgard/.gazebo/mod...
/home/asgard/aws-robomak...
/home/asgard/car_demo/mo...
https://fuel.ignitionrobotics....
http://models.gazebosim.org/
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:34:16.518 Real Time: 00 00:29:22.827 Iterations: 1732811 FPS: 42.70
SSD


Activities
Gazebo
Apr 20 19:21
Screenshot captured
You can paste the image from the clipboard.
File Edit Camera View Window Help
World Insert Layers
Add Path
/home/asgard/.gazebo/models
/usr/share/gazebo-11/models
/usr/share/gazebo/../../share...
/opt/ros/humble/share/turtl...
/home/asgard/gazebo_mod...
/home/asgard/placement/sr...
/home/asgard/placement/sr...
/home/asgard/.gazebo/mod...
/home/asgard/aws-robomak...
/home/asgard/car_demo/mo...
https://fuel.ignitionrobotics....
http://models.gazebosim.org/
Steps: 1
Real Time Factor: 1.00
Sim Time: 00 00:34:06.940
Real Time: 00 00:29:13.204
Iterations: 1723233
FPS: 59.10
SSD


Activities
Gazebo
Apr 20 19:20
Gazebo
File Edit Camera View Window Help
World Insert Layers
Add Path
/home/asgard/.gazebo/models
/usr/share/gazebo-11/models
/usr/share/gazebo/../../share...
/opt/ros/humble/share/turtl...
/home/asgard/gazebo_mod...
/home/asgard/placement/sr...
/home/asgard/placement/sr...
/home/asgard/.gazebo/mod...
/home/asgard/aws-robomak...
/home/asgard/car_demo/mo...
https://fuel.ignitionrobotics....
http://models.gazebosim.org/
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:33:54.367 Real Time: 00 00:29:00.571 Iterations: 1710660 FPS: 47.95
SSD



Action Item 3: GNSS-Stereo-Inertial SLAM for Arable Farming – 3 hour(s).

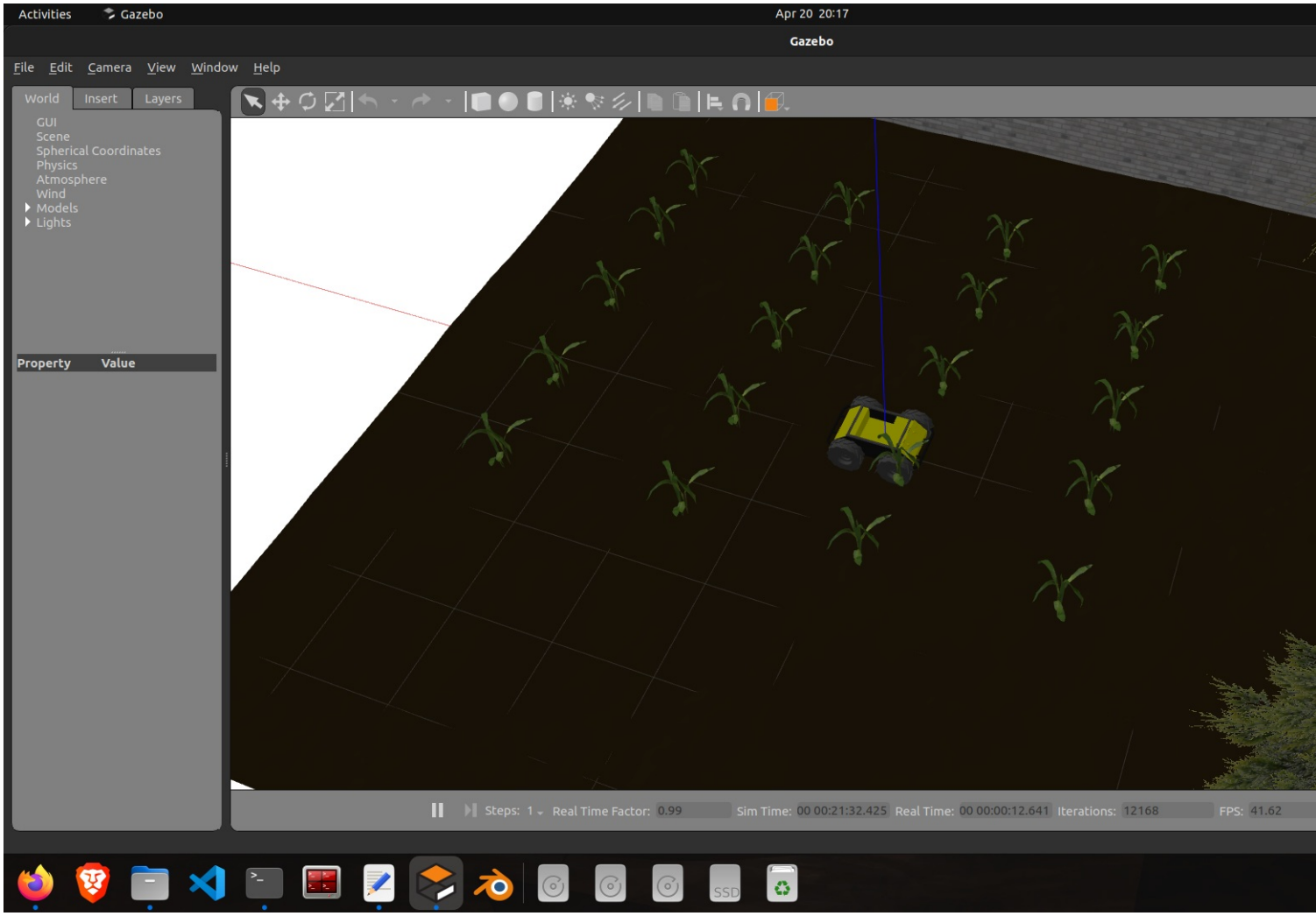
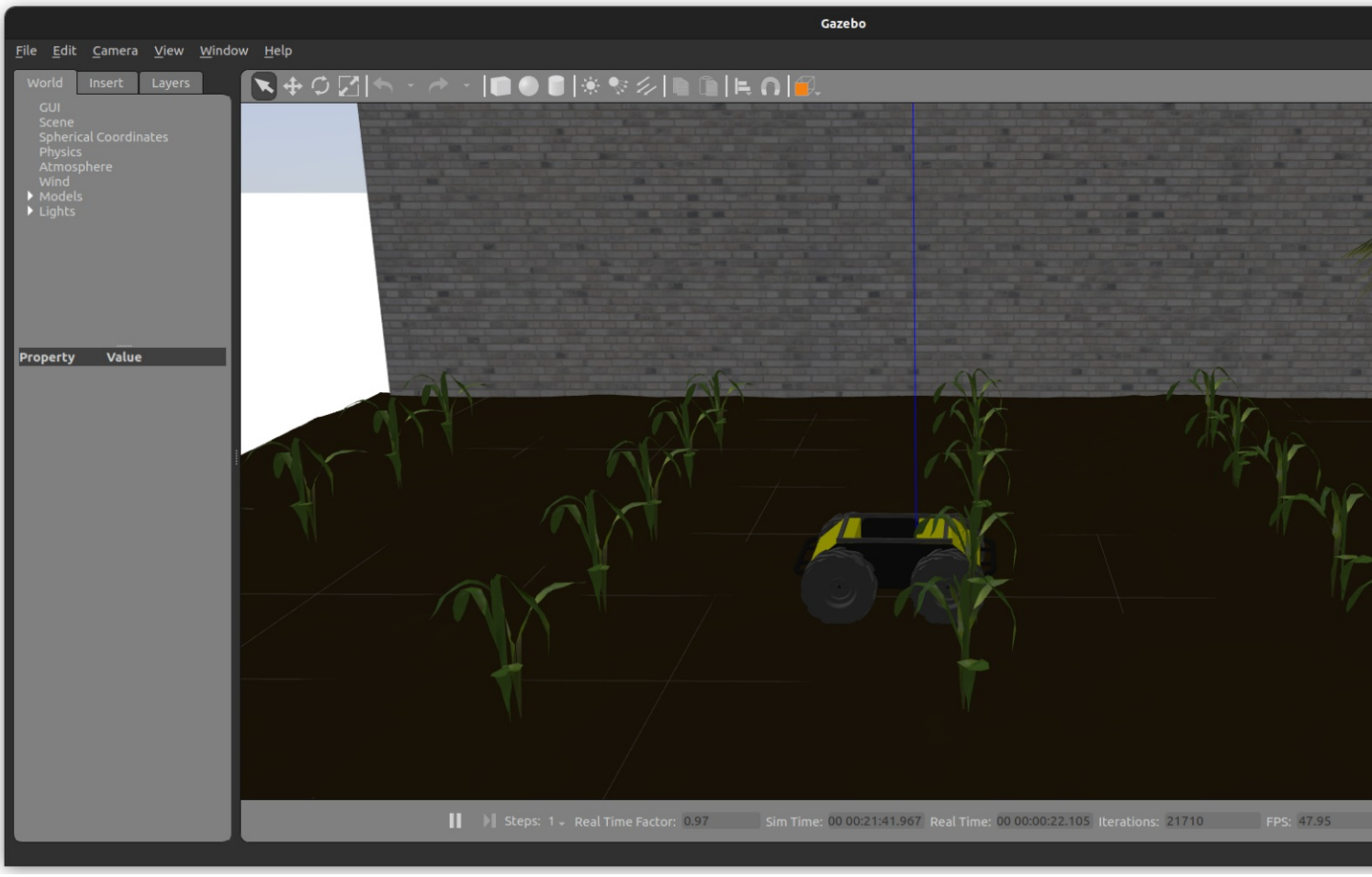
Research

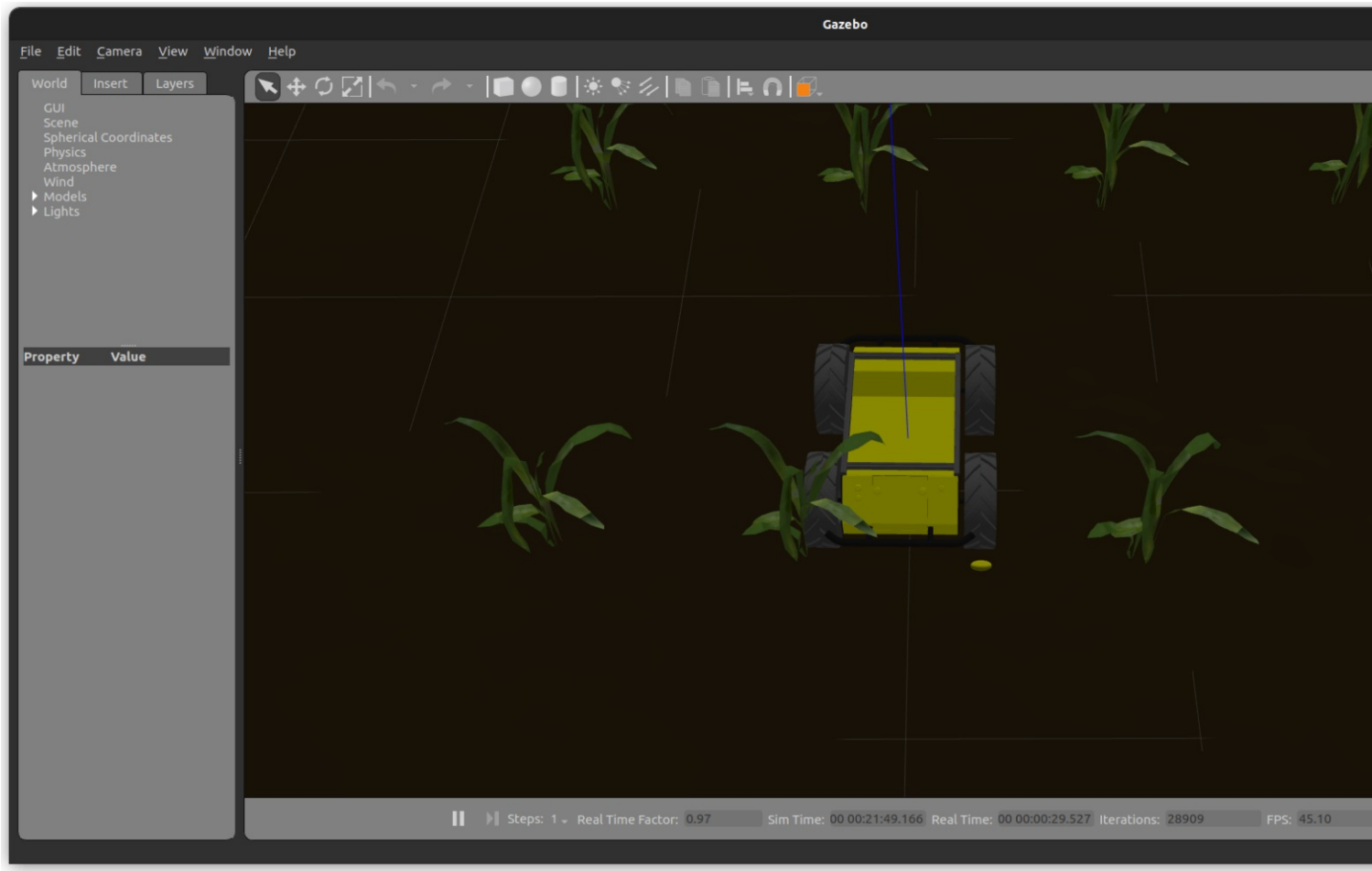
- https://arxiv.org/abs/2307.12836
- GNSS-Stereo-Inertial SLAM for Arable Farming
- Summary of Report
  - The paper proposes a tightly-coupled GNSS-stereo-inertial SLAM system built on top of ORB-SLAM3, integrating stereo vision, IMU, and GNSS readings for drift-less and robust localization, particularly in repetitive and texture-sparse agricultural environments.
  - Unlike loosely-coupled approaches that process each sensor stream independently, this method fuses all sensor data within a single optimization pipeline, ensuring cross-sensor correlations are respected and reducing cumulative pose error over time.
  - The authors evaluate the system on the Rosario dataset and custom soybean field trials, demonstrating up to 30% improvement in Absolute Trajectory Error (ATE) over stereo-inertial SLAM and VINS-Fusion. Notably, their results were achieved using a conventional GNSS, not a high-precision RTK module, which underscores the practicality of the method.
- Relation to Project
  - This work is a near-direct inspiration for our Agribot localization pipeline. Our goal is to combine GNSS and monocular VO, and this paper reinforces the idea that stereo or monocular VO fused with inertial and GNSS data is the most resilient architecture for open-field navigation.
  - The integration of GNSS as a mapping constraint within the SLAM loop (instead of a post-fusion or external localization aid) is something I aim to replicate using ROS2. It matches our planned use of GNSS correction to periodically reset local SLAM drift without waiting for loop closures.
  - Their field experiments in soybean farms (with repetitive row patterns and minimal visual features) mirror the simulated maize plots we've been building. The challenges they describe — such as perceptual aliasing and failed loop closures — directly validate our strategy of adding artificial landmarks (trees, walls) into the Gazebo map for monocular VO to lock onto.
- Motivation for Research
  - One of my biggest concerns was how to structure a localization pipeline that works despite repetitive textures and lack of strong loop closure — a recurring pain point in agriculture. This paper shows that tightly-coupled fusion can address both and achieve state-of-the-art performance.
  - It’s particularly motivating that the authors used real, noisy GNSS (not RTK or idealized data), yet still achieved robust and smooth trajectories. This proves that simulation using Gazebo + synthetic GNSS is a valid pre-deployment strategy — something we’re already doing in our farmland setup.
  - Finally, the public release of their implementation opens a powerful door for me. I plan to study their open-source GNSS factor model and eventually build a ROS2-compatible version that supports monocular visual odometry for the Agribot, possibly replacing stereo where needed due to hardware constraints.

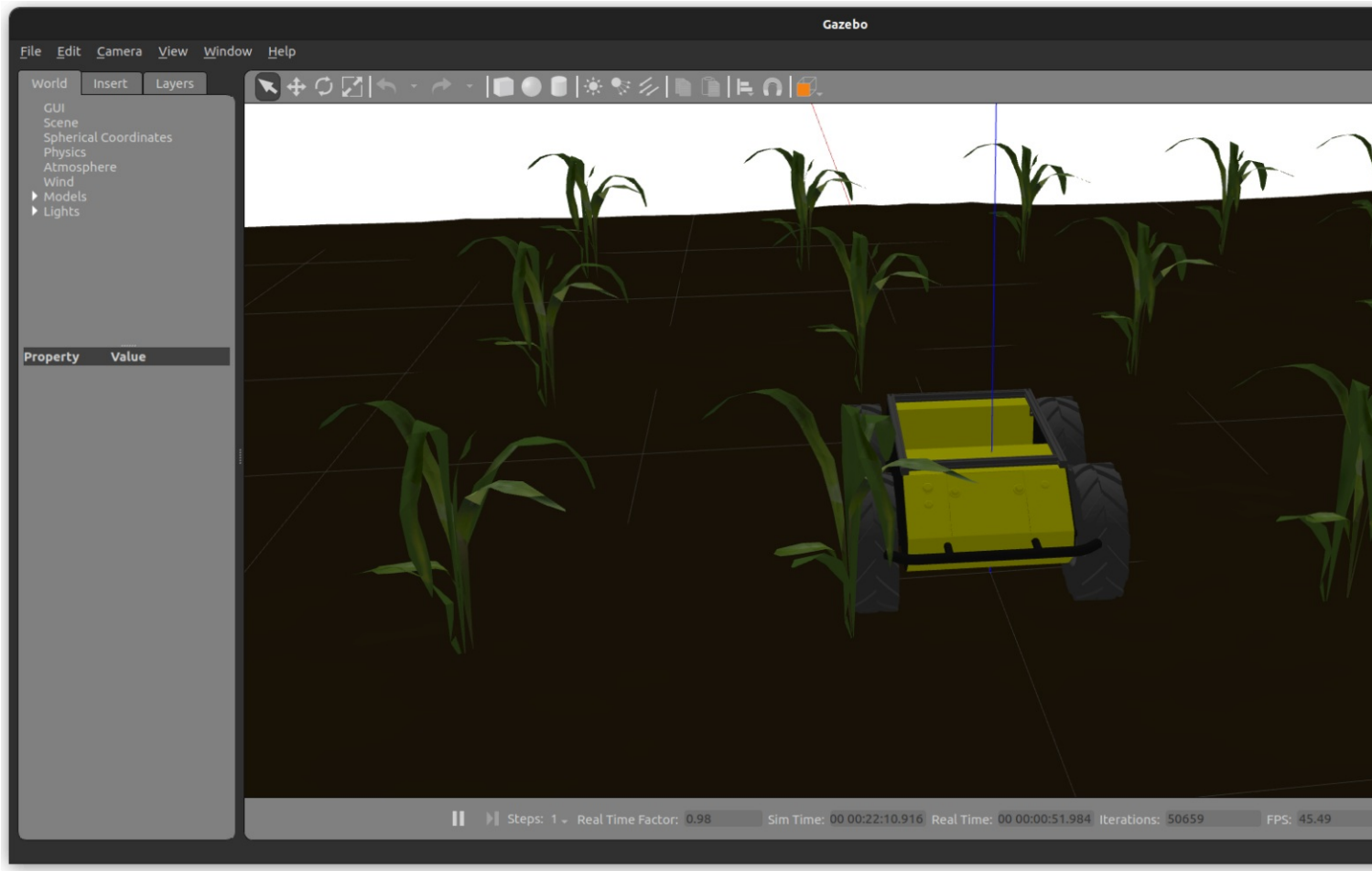
Action Item 4: Integration of the Husky Model and final enhancements in the gazebo map – 3 hour(s).

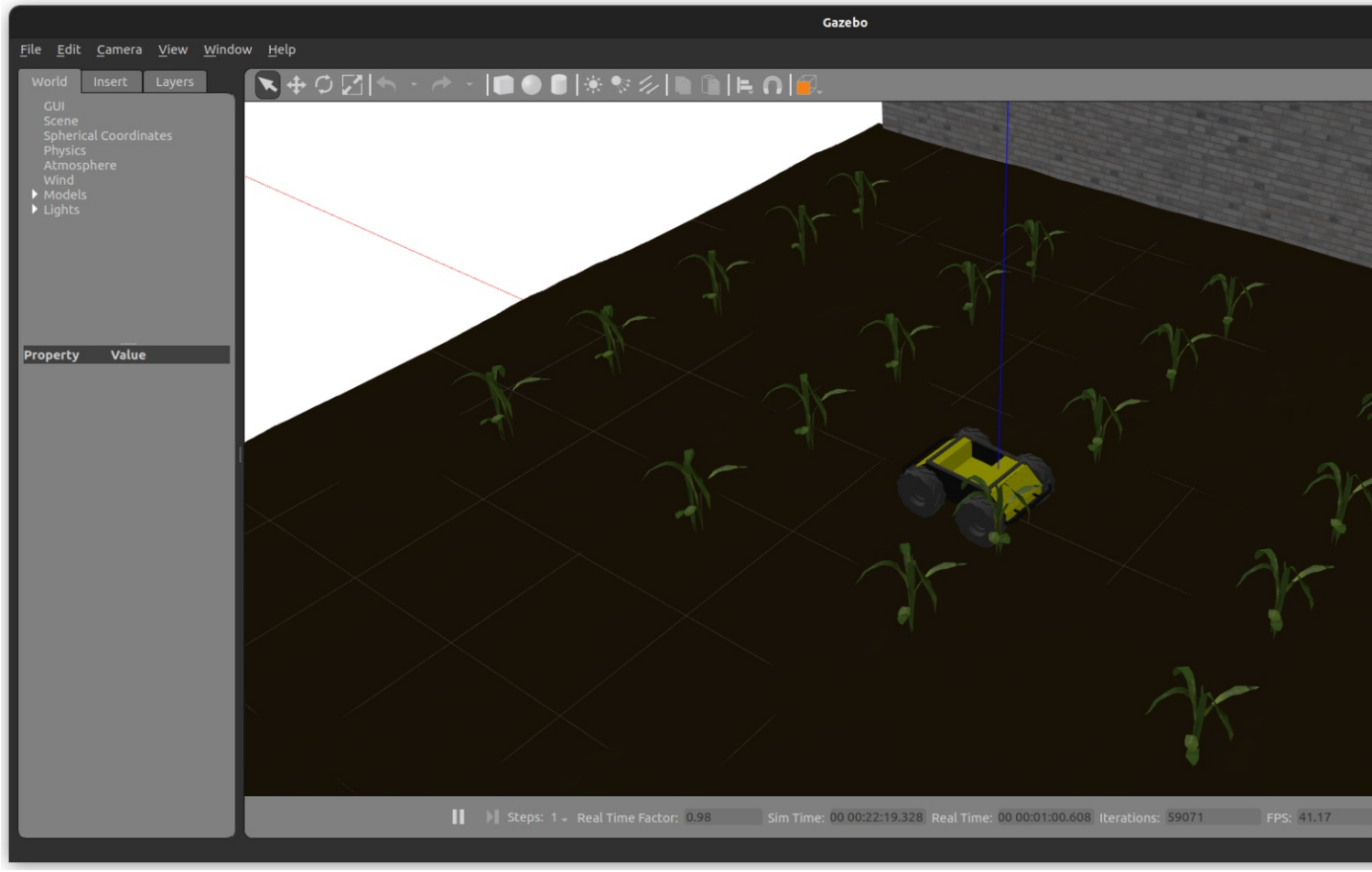
Project Work Summary

- Once the base farmland map was complete with maize rows and soil textures, I began refining it visually and structurally to support monocular visual odometry. I decided to introduce a wall as a vertical landmark to create strong visual features that VO systems can latch onto. This was especially necessary because the crop rows themselves were visually repetitive, leading to potential perceptual aliasing.
- Using Gazebo’s model insertion interface, I placed a wall along the boundary of the field. I fine-tuned its position and orientation by adjusting the pose parameters directly within the .world file. I ensured it was aligned with the edge of the farmland and didn't interfere with the crops or rover path.
- After placing the wall, I manually added a directional light source and sky settings to simulate realistic outdoor lighting and saved the entire environment as a new .world file.
- Next, I attempted to integrate the Husky robot into the scene. I added an <include> block for the model://husky URI to the top of the .world file and specified its pose. However, upon launching the simulation, I noticed the robot didn’t appear.
- This led me to re-inspect the world file, and I realized the <include> was accidentally placed outside the <world> tag. Since Gazebo silently ignores anything outside that scope, the model wasn't loaded.
- I moved the Husky <include> inside the <world> block and relaunched the simulation. This time the robot spawned correctly in the environment. After verifying its position and ensuring it sat just above ground level, I saved the corrected .world file as the working simulation environment. This process reinforced the importance of scope-aware editing in Gazebo and the subtle but critical syntactic rules in .world files.

- 
Activities
Gazebo
Apr 20 20:17
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 00:21:32.425 Real Time: 00 00:00:12.641 Iterations: 12168 FPS: 41.62
- 
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: 0.97 Sim Time: 00 00:21:41.967 Real Time: 00 00:00:22.105 Iterations: 21710 FPS: 47.95


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: 0.97 Sim Time: 00 00:21:49.166 Real Time: 00 00:00:29.527 Iterations: 28909 FPS: 45.10


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: 0.98 Sim Time: 00 00:22:10.916 Real Time: 00 00:00:51.984 Iterations: 50659 FPS: 45.49

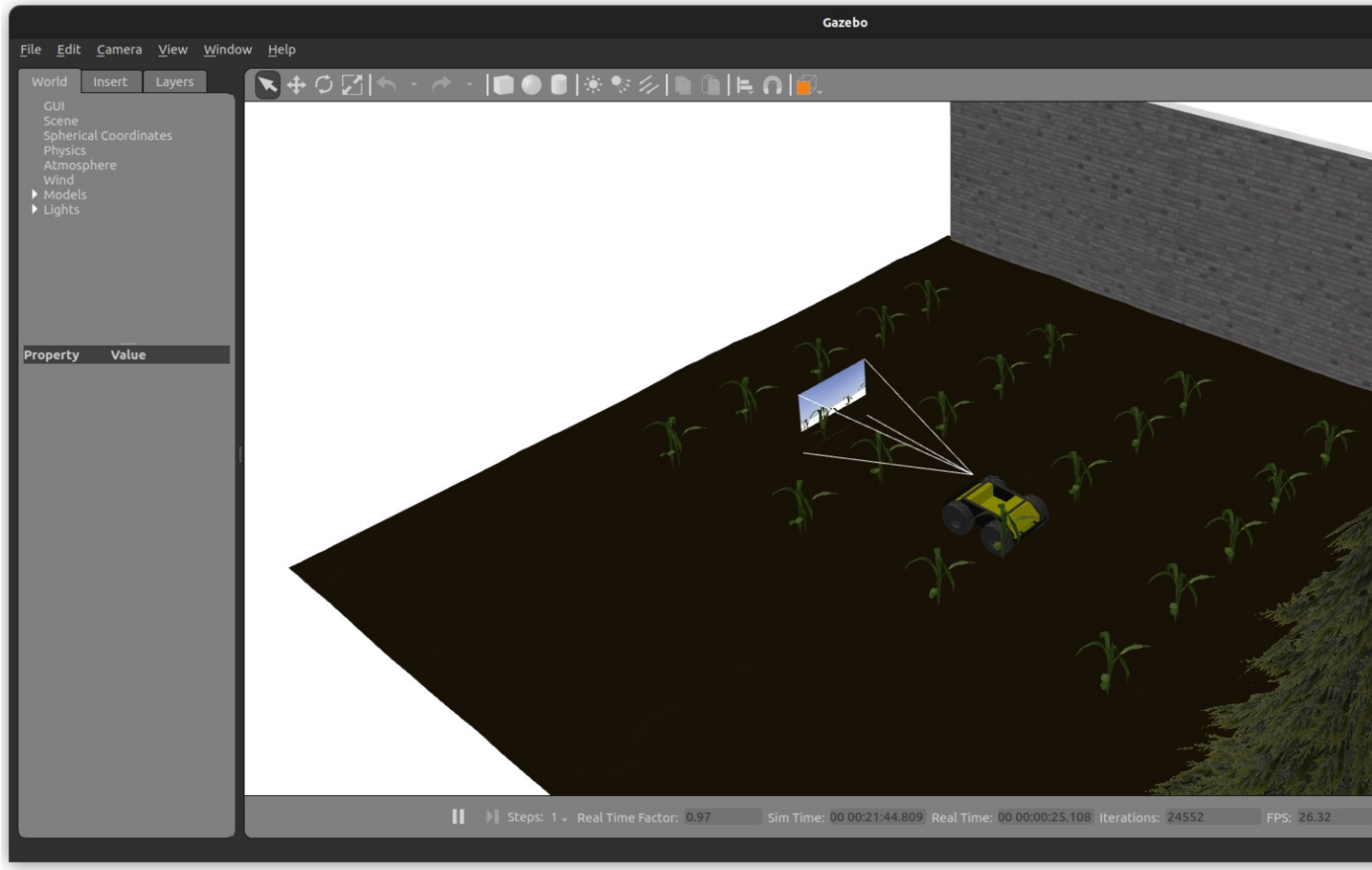


-
-

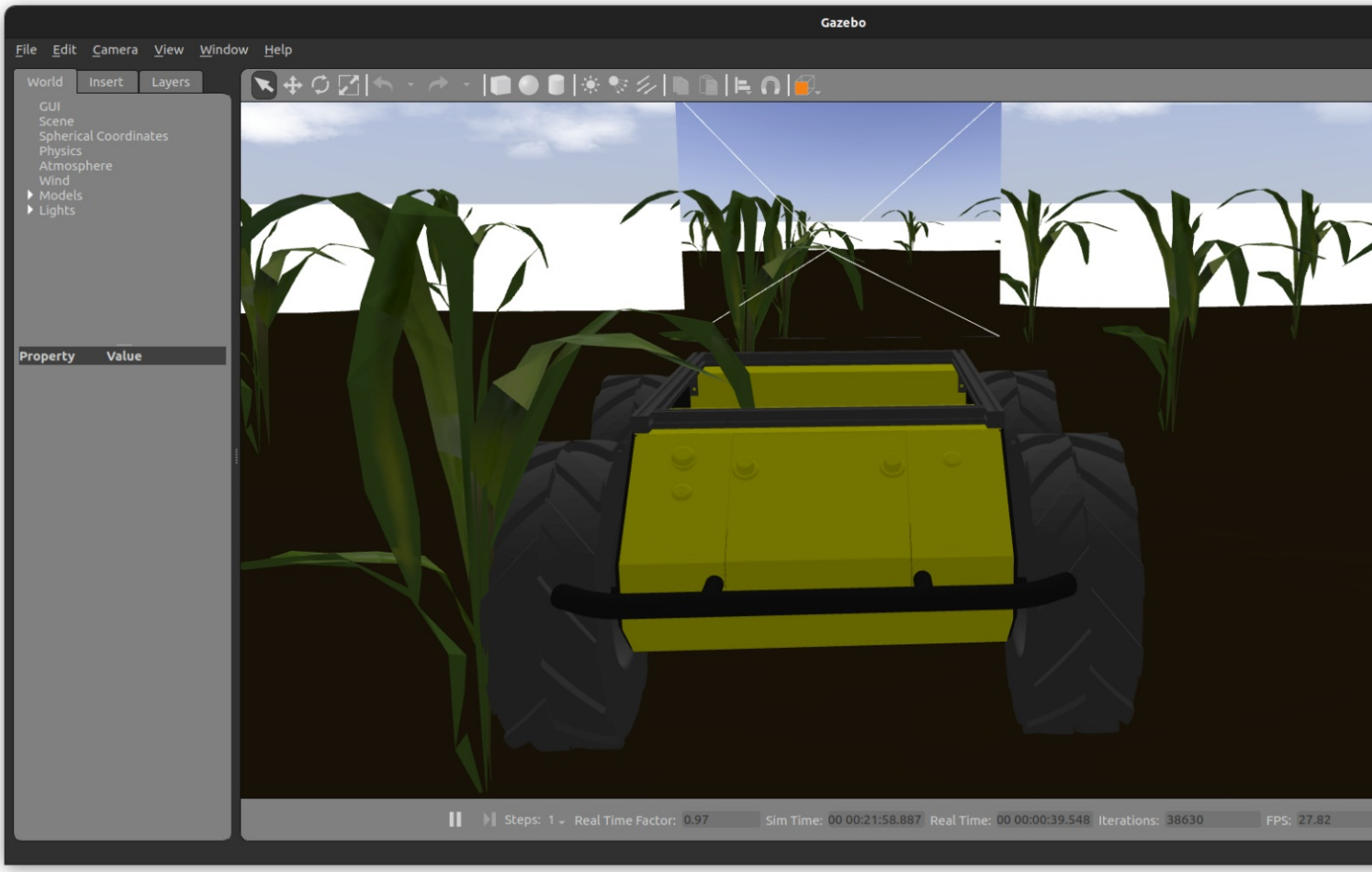
Action Item 5: Camera Integration into Husky and Debugging the Camera Feed in ROS2-Gazebo – 3 hour(s).

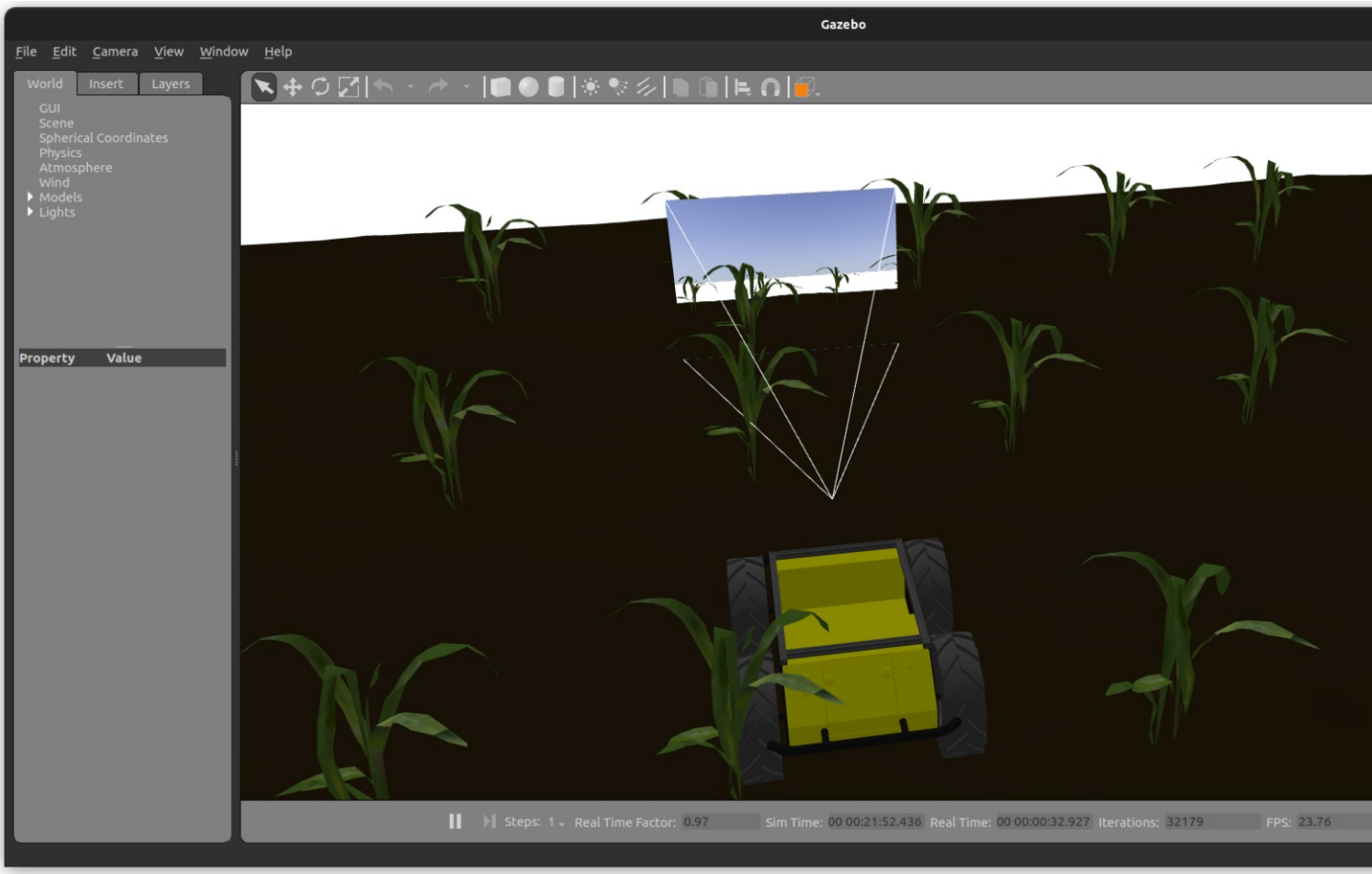
Project Work Summary

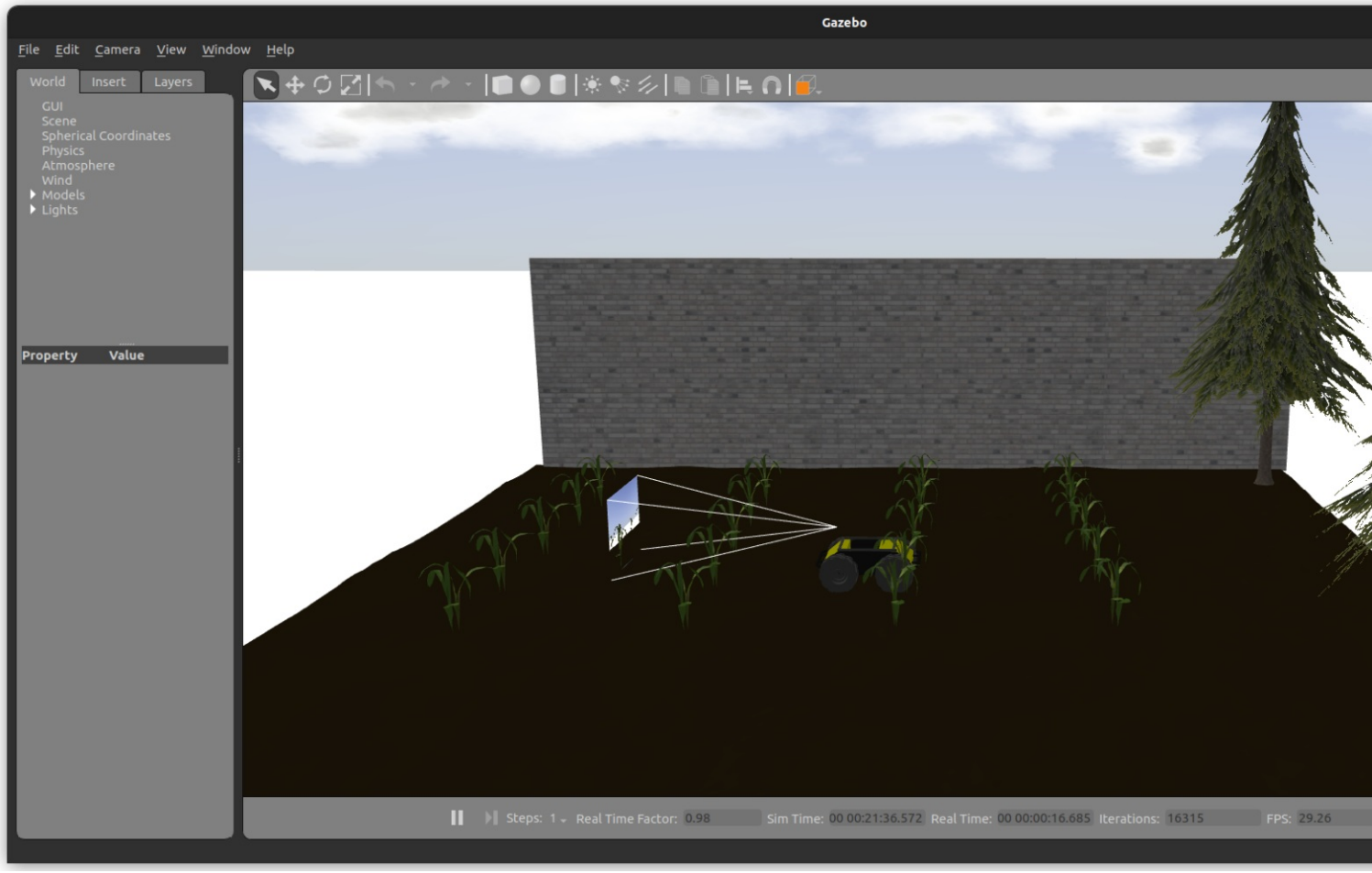
- After successfully loading the Husky robot into the simulation, I moved on to integrating a front-facing camera directly onto the robot. The goal was to simulate a monocular camera feed that could later be used for VO or dataset collection.
- Initially, I tried to add the camera as a new <model> block directly into the .world file, placing it near Husky and linking it via a fixed joint. However, this did not work as expected. The camera spawned in the world but remained static, disconnected from the Husky robot. It didn't follow the robot's movement and was effectively just a hovering GoPro in the middle of the field.
- Realizing this wasn't ideal, I opted for a more permanent and structurally sound solution — embedding the camera directly into Husky's own model. I navigated to the Husky's model.sdf file inside the ~/.gazebo/models/husky/ directory. Within the <link name="base_link">, I inserted a <sensor> block with a <camera> element and the necessary gazebo_ros_camera plugin. I ensured the camera had a slight forward and upward offset using the <pose> tag to mimic realistic mounting.
- After saving the file, I relaunched the simulation using ros2 launch gazebo_ros gazebo.launch.py, but I wasn't seeing the expected image topics. Suspecting a plugin issue, I checked the console and verified that there were no errors about libgazebo_ros_camera.so. Then I realized I had launched Gazebo using the gazebo command rather than the ROS2 interface — and thus the ROS2 topics were not being bridged.
- I relaunched the simulation using the ROS2-native launch command and confirmed that /husky_cam/image_raw and /camera_info were now visible. I further validated the camera feed using rqt_image_view, confirming that the embedded camera was correctly publishing to ROS2. This process helped solidify my understanding of ROS-Gazebo plugin lifecycles, and the importance of ROS2-aware simulation launch methods.




```
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~/farmbot/src 104x25
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot/src$ ros2 topic list
/clock
/front_camera/camera_info
/front_camera/image_raw
/parameter_events
/rosout
asgard@asgard-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/farmbot/src$
```


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: 0.97 Sim Time: 00 00:21:58.887 Real Time: 00 00:00:39.548 Iterations: 38630 FPS: 27.82

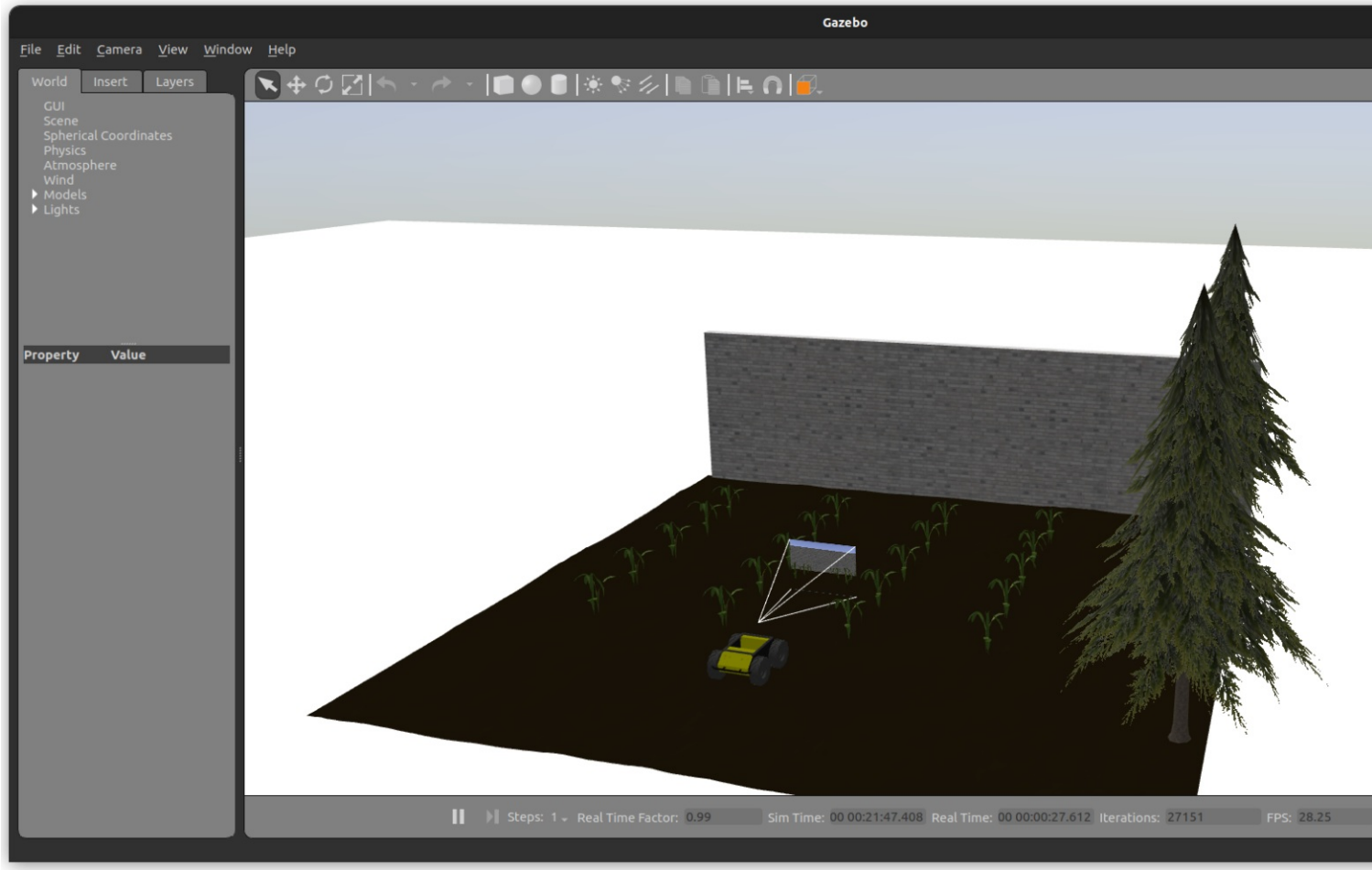

Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.97 Sim Time: 00 00:21:52.436 Real Time: 00 00:00:32.927 Iterations: 32179 FPS: 23.76

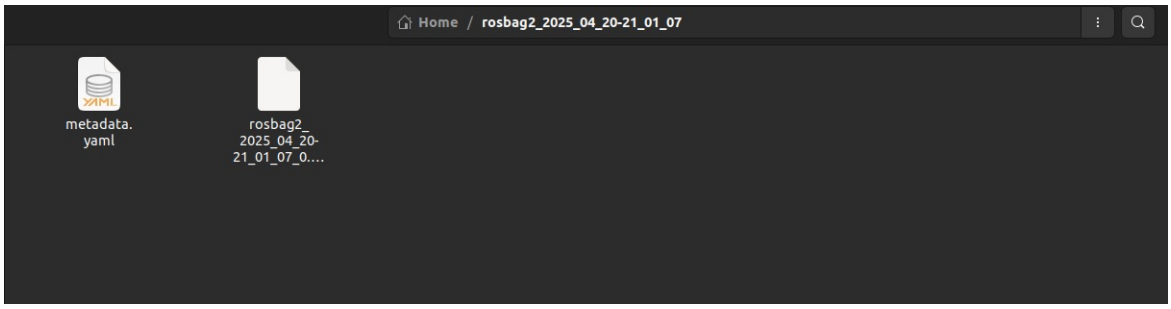


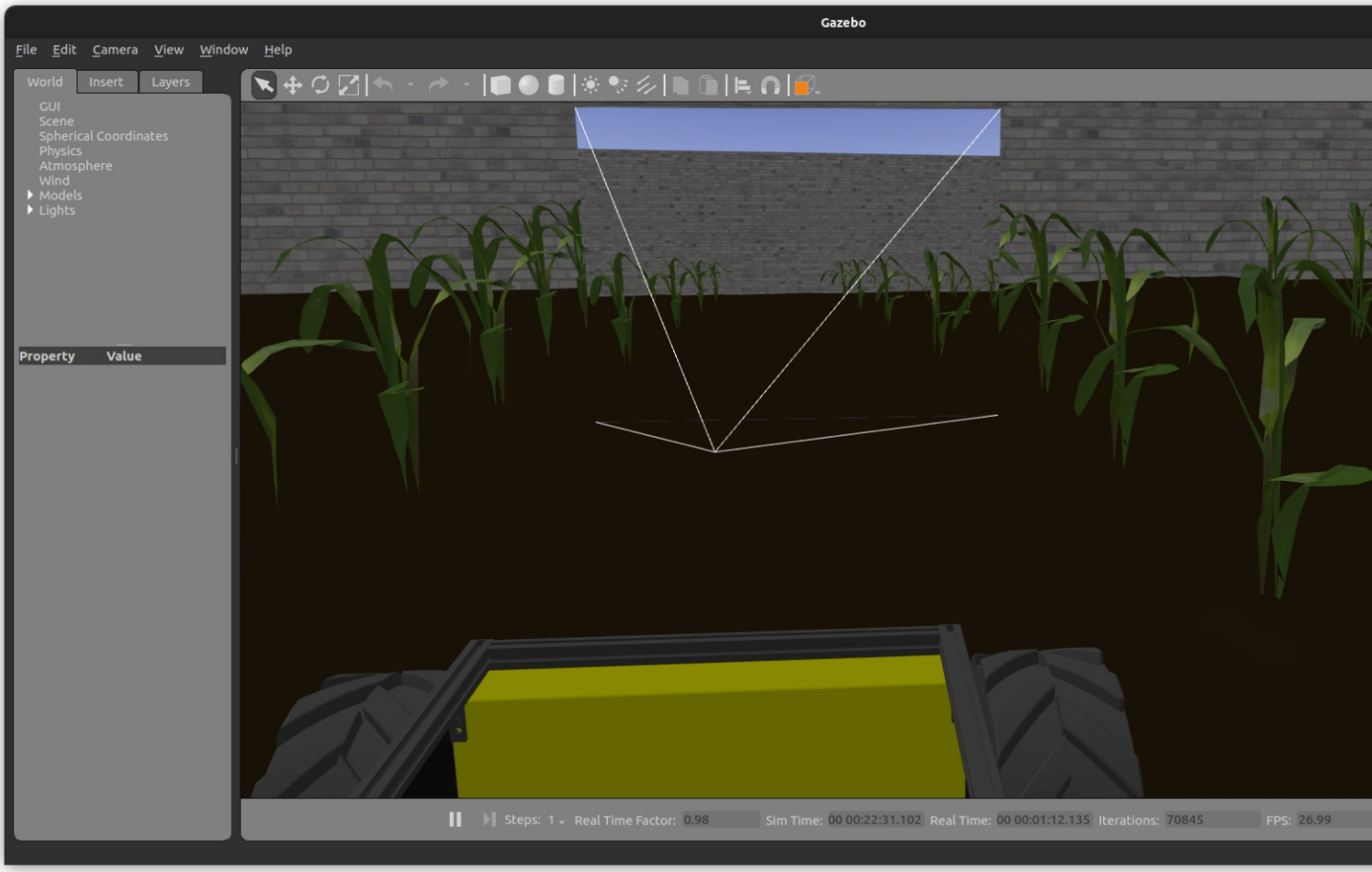
Action Item 6: Aligning the Husky with the Crop Row and Capturing Visual Data for VO – 3 hour(s).

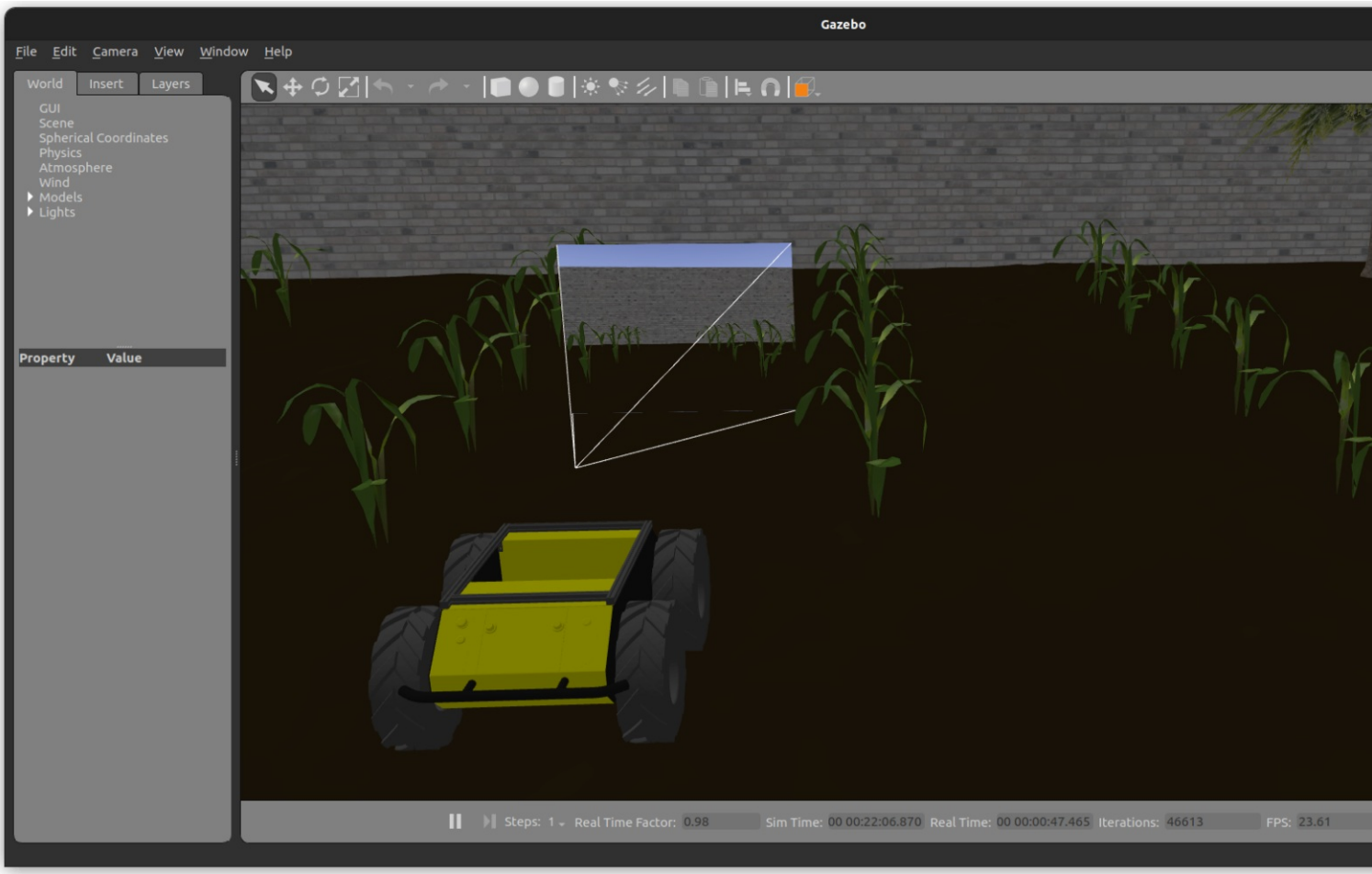
Project Work Summary

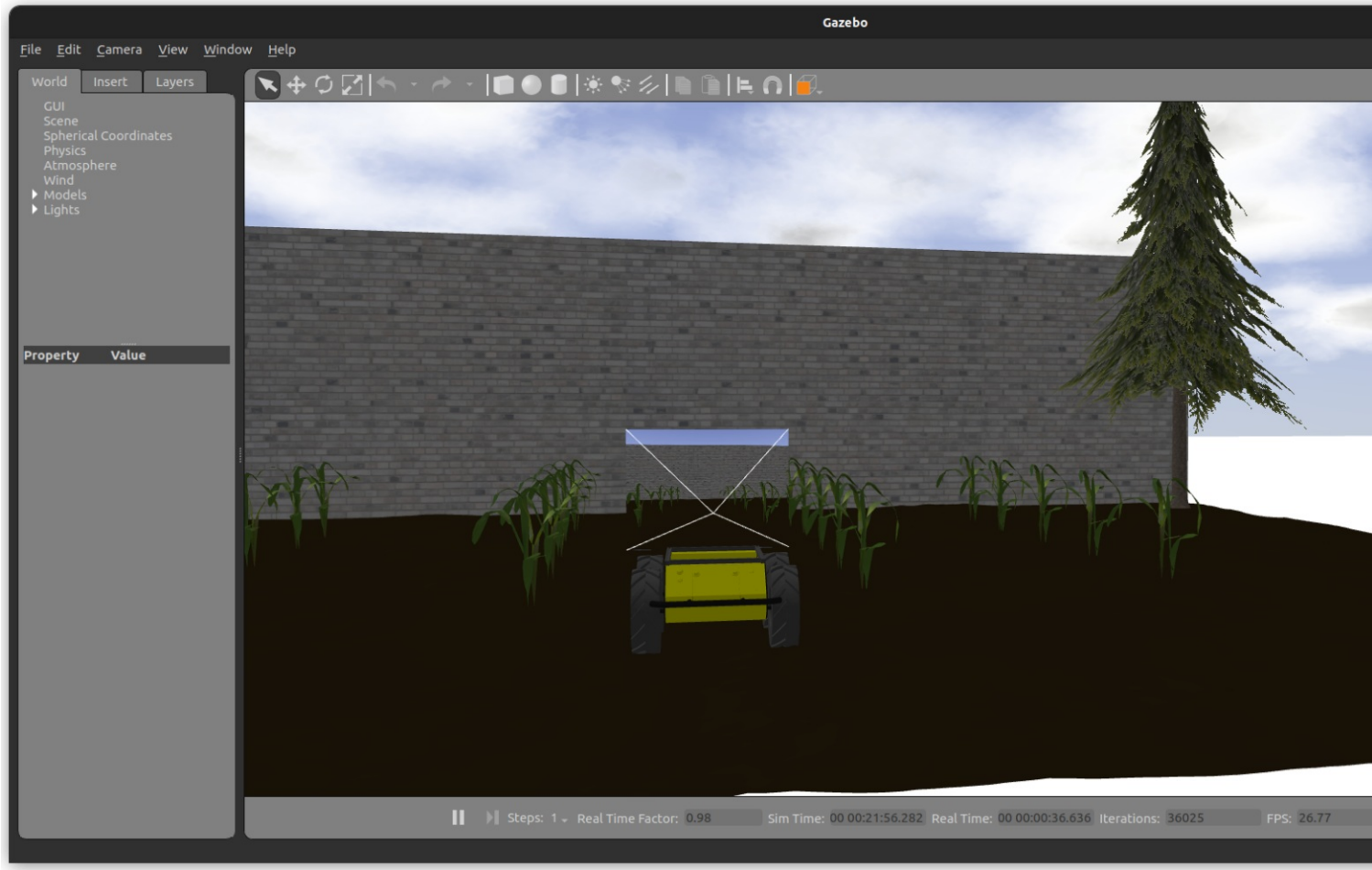
- After embedding the Husky robot into our custom farmland map, the next step was to ensure it was precisely aligned with the crop row to simulate forward-facing monocular camera movement for visual odometry tasks. The idea was to position the robot such that its camera looked directly down a row of maize crops with minimal angular offset — just as it would in a real field traversal scenario.
- To begin, I experimented with the <pose> values in the Husky <include> block within the .world file. My first attempt was <pose>-0.5 -2.5 0 0 0 1.5708</pose>, which I expected would place the robot near the start of a row and point it toward the crops. However, this positioning was too far out and possibly below the visible soil plane, so Husky didn't appear correctly in the simulation.
- I continued adjusting the x, y, and yaw values iteratively. Each time, I reloaded the world in Gazebo and inspected the alignment visually. After a few tweaks, I found an optimal pose: <pose>0.6 2.5 0 0 0 -1.5708</pose>
- This placed the Husky on the soil patch, directly in front of a crop row, and oriented it to face downward along the y-axis. Once the position looked visually centered and functional, I saved the .world file to preserve the alignment for future sessions.
- With the robot aligned, I proceeded to launch the camera feed embedded in the Husky model. I verified that /husky_cam/image_raw was publishing correctly and used rqt_image_view to confirm live camera output from the simulated field. The view provided a consistent sequence of images with maize rows ahead — suitable for training or evaluating visual odometry algorithms.
- This saved the camera output as a .db3 file in a timestamped rosbag2_ directory within the current working directory. I also ensured I had access to camera_info for lens calibration parameters. This dataset would serve as a foundation for downstream VO testing, dataset annotation, or even feature extraction pipelines.
- This task wrapped up the physical positioning and initial data acquisition phase, setting the stage for integrating pose estimation and fusing visual data with GNSS in future stages.

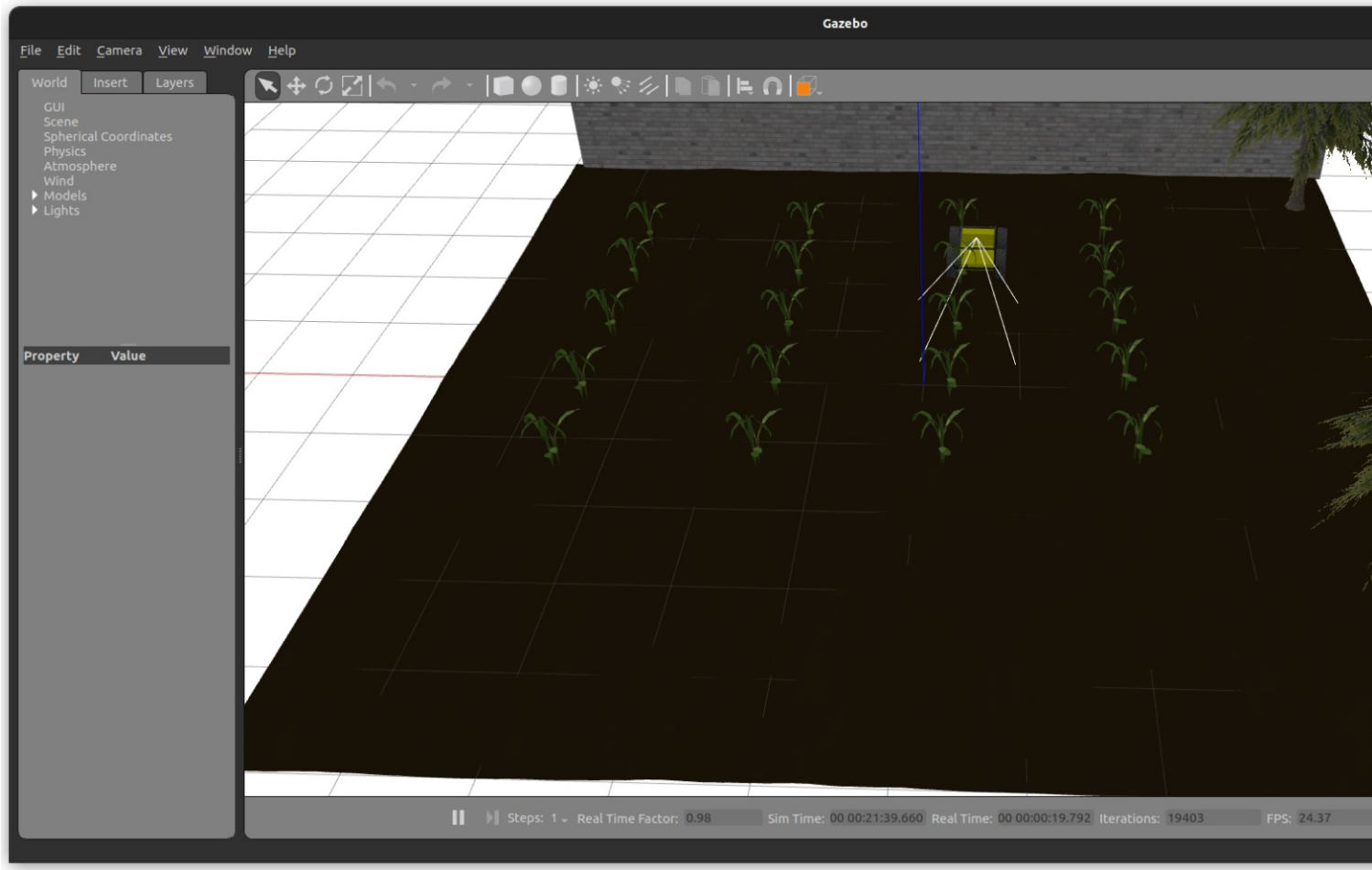

Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:21:47.408 Real Time: 00 00:00:27.612 Iterations: 27151 FPS: 28.25


Home / rosbag2_2025_04_20-21_01_07
metadata.
yaml
rosbag2_
2025_04_20-
21_01_07_0....


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: 0.98 Sim Time: 00 00:22:31.102 Real Time: 00 00:01:12.135 Iterations: 70845 FPS: 26.99


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: 0.98 Sim Time: 00 00:22:06.870 Real Time: 00 00:00:47.465 Iterations: 46613 FPS: 23.61


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: 0.98 Sim Time: 00 00:21:56.282 Real Time: 00 00:00:36.636 Iterations: 36025 FPS: 26.77


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: 0.98 Sim Time: 00 00:21:39.660 Real Time: 00 00:00:19.792 Iterations: 19403 FPS: 24.37

-
-

Action Item 7: Preparing for Sensor Fusion and Navigation Integration with GNSS and Visual Odometry – 1 hour(s).

Project Work Summary

- With the camera feed successfully streaming from the Husky robot and the vehicle aligned with the maize crop row, I'm now ready to transition toward autonomous navigation using a fused localization system. The goal for the upcoming phase is to lay down the groundwork that will eventually allow the robot to navigate autonomously using GNSS and visual input, both in simulation and (eventually) in real-world field deployments.
- The first task will be to configure the GNSS module inside Gazebo. This involves adding a GPS sensor plugin to the Husky robot, similar to how the camera was embedded. I plan to use the gazebo_ros_gps plugin to simulate satellite signals and ensure the /fix topic publishes data in the standard sensor_msgs/NavSatFix format. Verifying this stream will be essential before introducing fusion logic.
- Once GNSS is active, I'll review the available sensor fusion frameworks — most likely robot_localization with an Extended Kalman Filter (EKF). I'll begin with a standalone test of EKF on simulated GNSS and odometry, to ensure the configuration is robust and outputs a usable /odometry/filtered topic. This step will help me understand the tuning parameters before adding visual odometry to the mix.
- Simultaneously, I'll revisit the visual odometry pipeline and select a suitable lightweight ROS2-compatible VO package. Monocular VO options like VSLAM or OpenVINS will be shortlisted. I will either launch the VO node on saved bag data from Husky or stream real-time camera images into the pipeline to test how well it tracks the rover's motion along the crop row.
- After verifying that both the GNSS and VO sources are working independently, I'll move toward synchronizing them via fusion. This will include configuring transforms, broadcasting tf frames (e.g., base_link, map, odom), and ensuring time synchronization between topics. This step may also involve building a basic launch file that brings up the entire localization stack in ROS2 Humble.
- Lastly, I aim to validate the fused navigation by visualizing the robot path in RViz using markers or trajectory traces from the filtered output. Even if motion control isn't yet active, I want to confirm that the estimated trajectory matches Husky's movement through the simulated field.

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.