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.

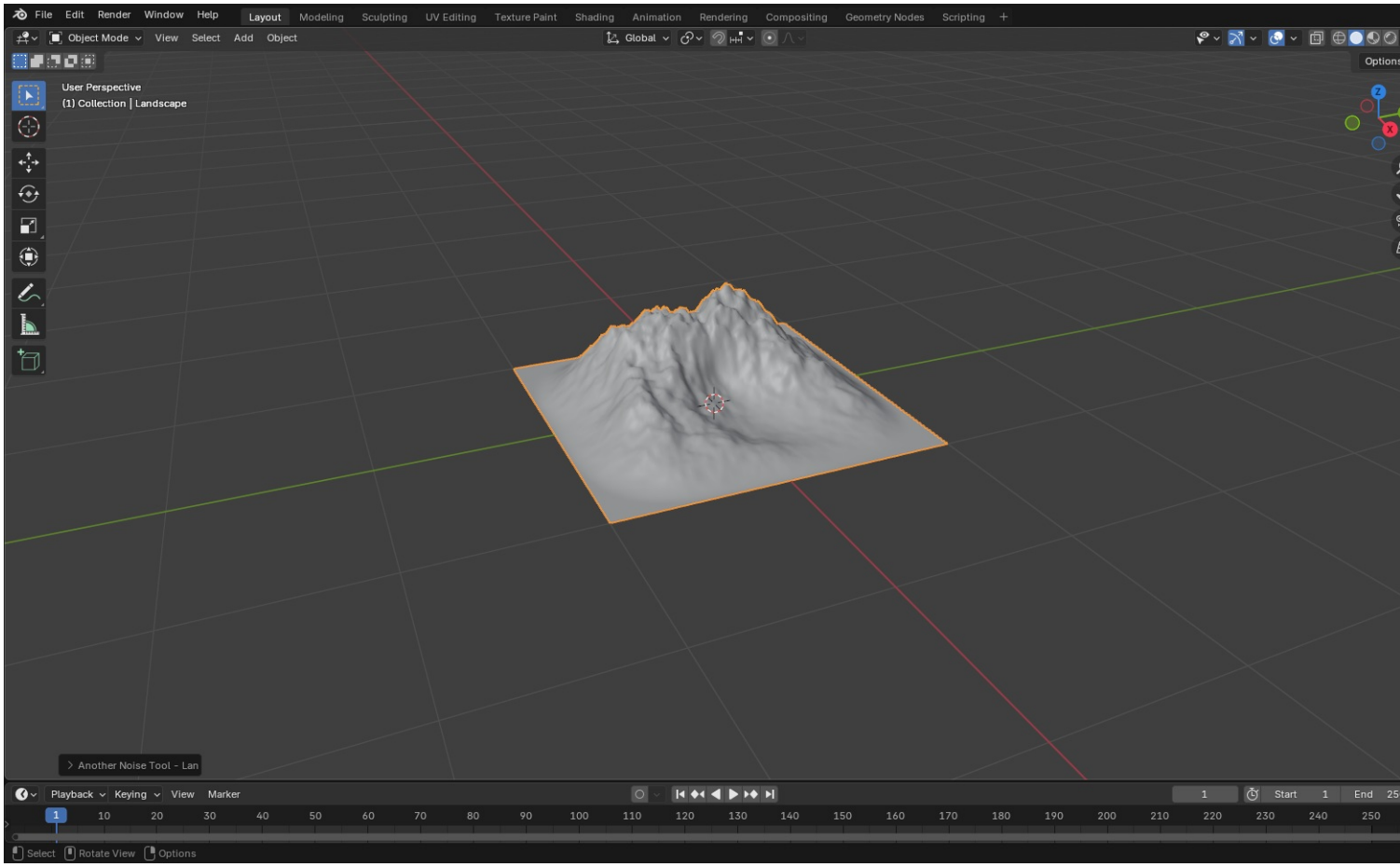
#	Action Item/ Explanation	Total Time This Week (hours)
1	Installing and Setting Up the A.N.T. Landscape Add-on	3
2	Creating Terrain with Hills and Dunes Presets	3
3	Automatically Annotated Dataset of a Ground Mobile Robot in Natural Environments	3
4	Creating Realistic Wheat Fields with Particle Systems	3
5	Exporting the Farmland Scene from Blender to Gazebo	3
6	Setting Up Gazebo for Farmland Simulation	3
7	Weekly Plan for Next Week	1
8	Report Writing	1
	Total hours for the week:	20

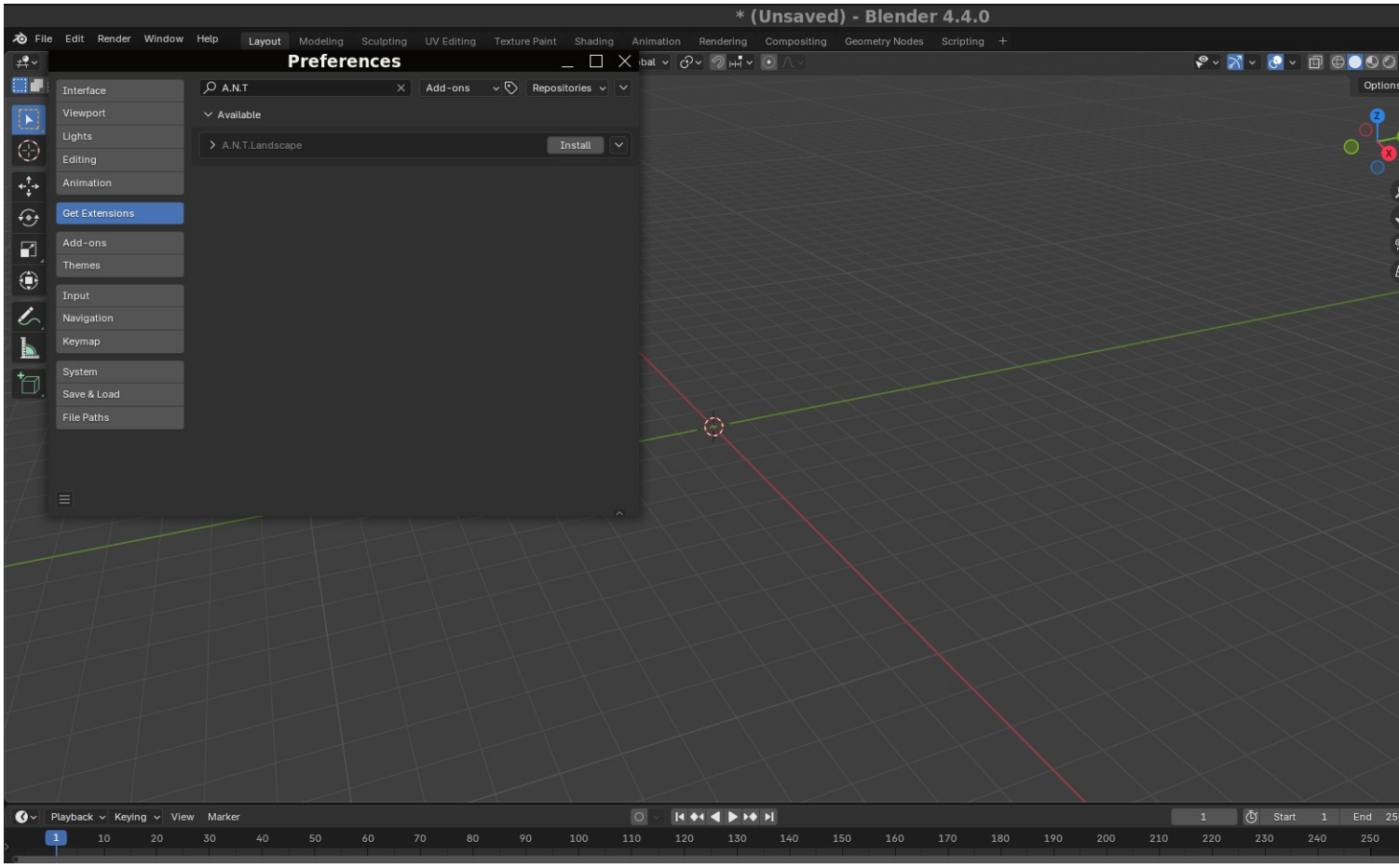
Verification Documentation:

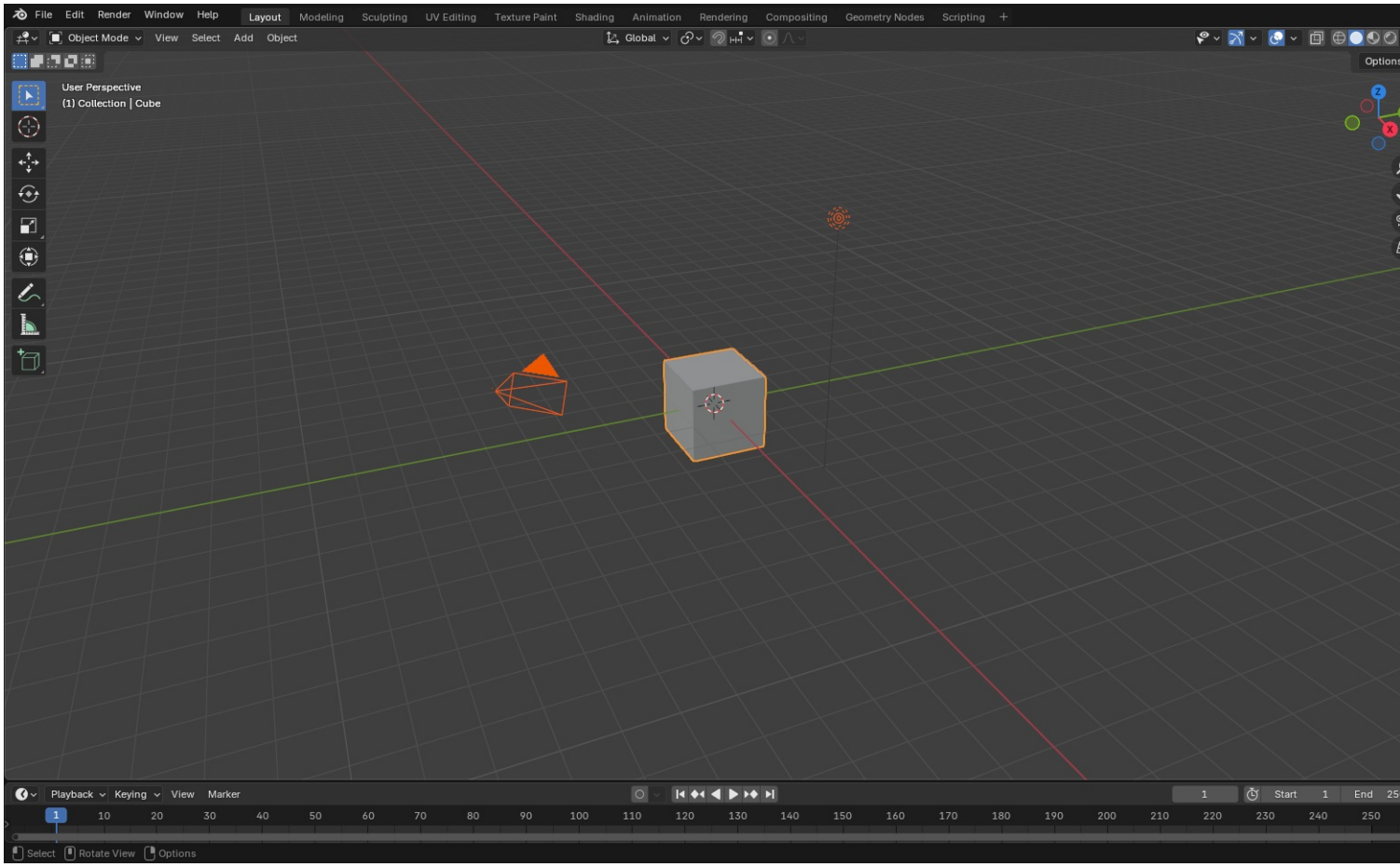
Action Item 1: Installing and Setting Up the A.N.T. Landscape Add-on – 3 hour(s).

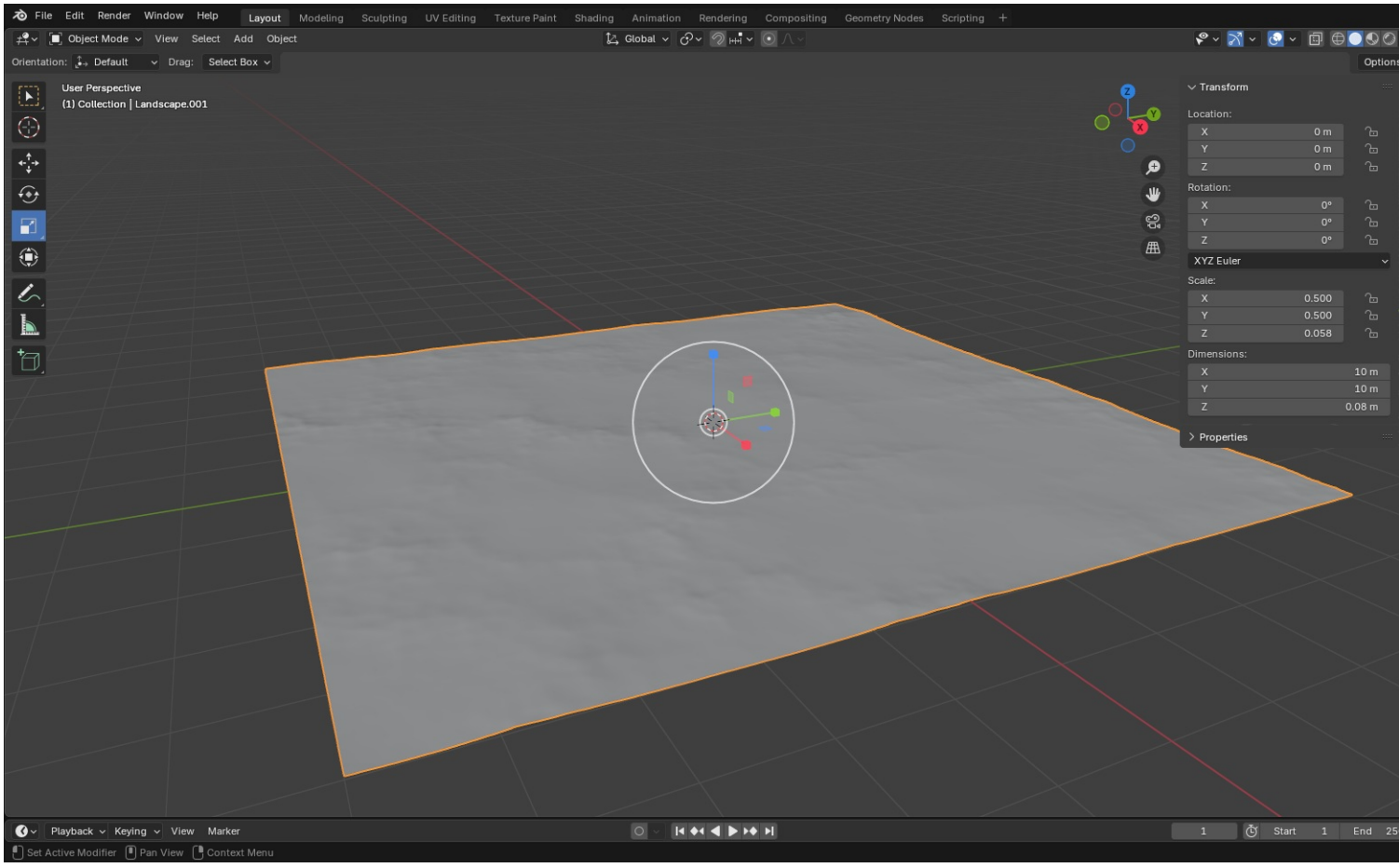
Project Work Summary

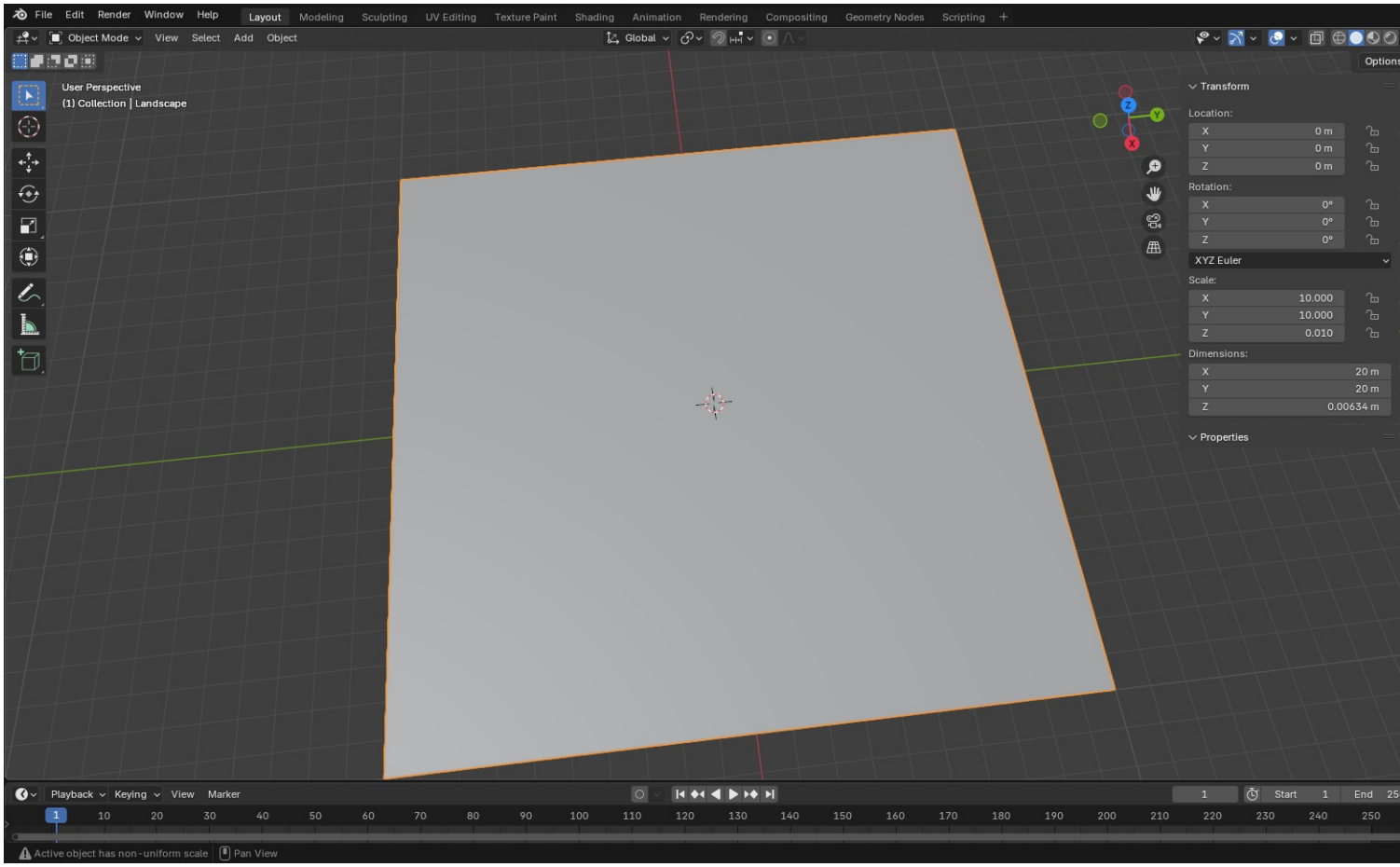
- I decided to leverage Blender's powerful add-on system to create our farmland terrain more efficiently. The A.N.T. Landscape generator is perfect for this project since it allows us to quickly generate natural-looking terrains without the painstaking process of manual modeling from scratch.
- First, I opened Blender and familiarized myself with the interface. Before diving into any modeling, I needed to make sure the A.N.T. Landscape add-on was available. I navigated to Edit → Preferences to access Blender's settings, where all the powerful add-ons are managed.
- In the Preferences window, I clicked on the Add-ons tab where Blender keeps its extensive library of both official and community-created tools. The search bar at the top came in handy - I typed "ANT" and watched as the results filtered down to show the add-on I needed.
- Looking through the list, I found "Add Mesh: A.N.T. Landscape" - this powerful terrain generator comes pre-packaged with Blender but isn't enabled by default. I simply clicked the checkbox next to it to activate the add-on, watching as it illuminated to indicate the add-on was now ready for use.
- After enabling the add-on, I took a moment to look at its settings. The preferences showed some configuration options, but I left them at their defaults since they'd work fine for our initial terrain generation needs.
- With the add-on now activated, I returned to the main Blender interface to start using it. I cleared my workspace by selecting the default cube and deleting it (pressing X and confirming), giving myself a clean slate to work with for our farmland project.
- To verify the add-on was properly installed, I pressed Shift+A to open the Add menu, then navigated to Mesh → Landscape. Seeing this option appear confirmed that A.N.T. Landscape was successfully installed and ready to use for creating our terrain for the Husky robot simulation.

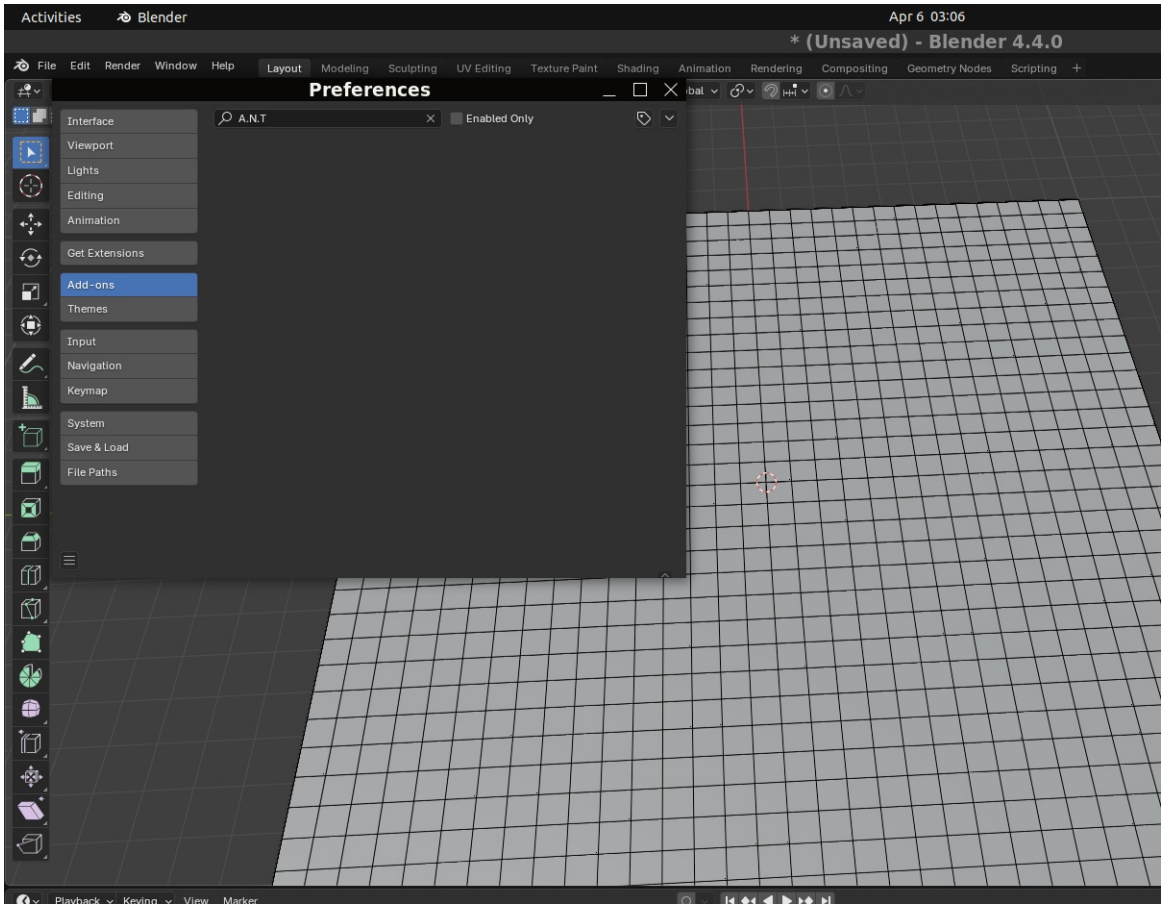

File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection | Landscape
Another Noise Tool - Lan
Playback Keying View Marker
Start 1 End 25
Select Rotate View Options


* (Unsaved) - Blender 4.4.0
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Preferences
Interface
Viewport
Lights
Editing
Animation
Get Extensions
Add-ons
Themes
Input
Navigation
Keymap
System
Save & Load
File Paths
A.N.T
Add-ons
Repositories
Available
A.N.T.Landscape
Install
Options
Playback Keying View Marker
Start 1 End 25


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection | Cube
Playback Keying View Marker
1 Start 1 End 25
1 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250
Select Rotate View Options


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
Orientation: Default Drag: Select Box
Options
User Perspective
(1) Collection | Landscape.001
Transform
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler
Scale:
X 0.500
Y 0.500
Z 0.058
Dimensions:
X 10 m
Y 10 m
Z 0.08 m
Properties
Playback Keying View Marker
1 Start 1 End 25
Set Active Modifier Pan View Context Menu

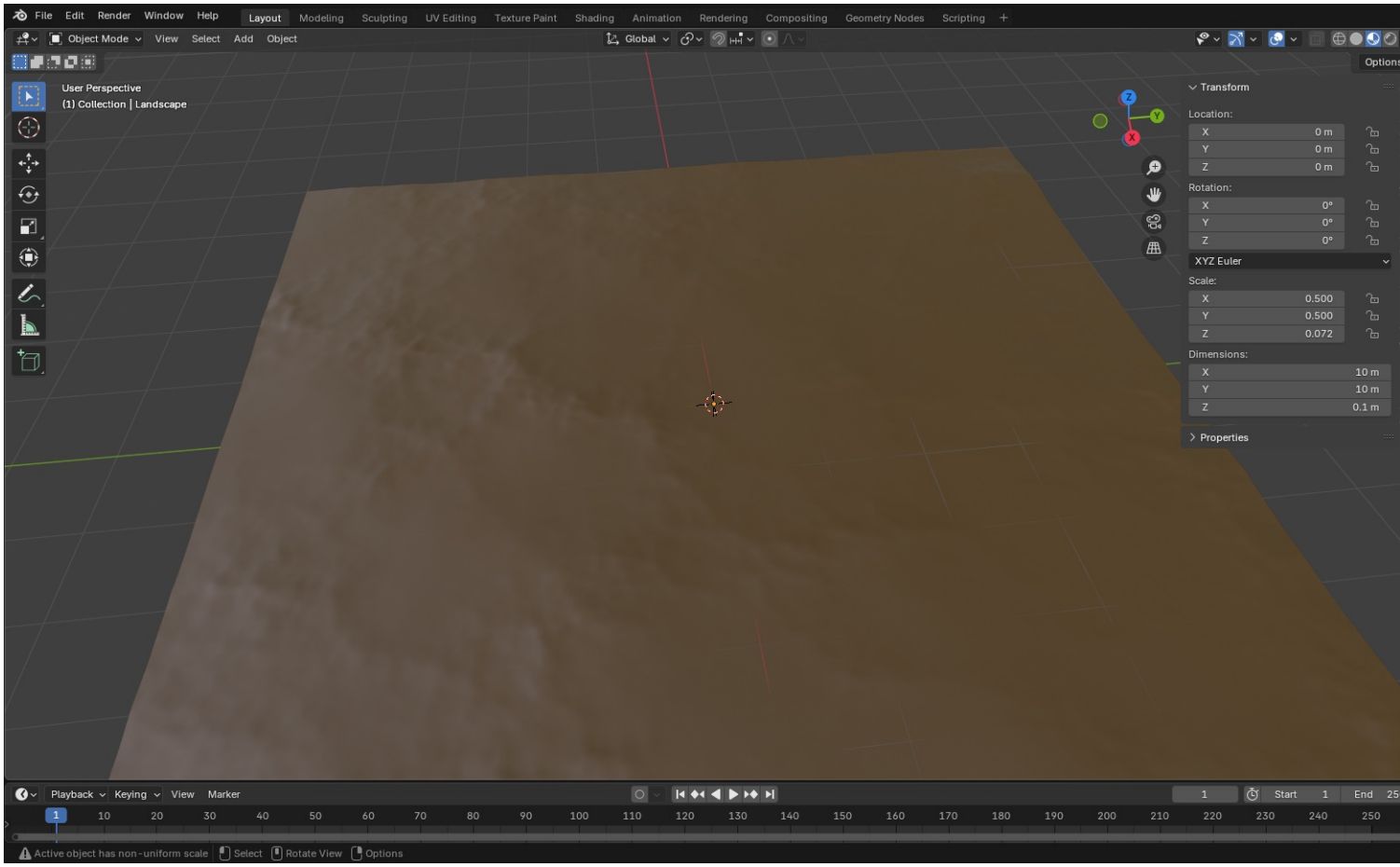


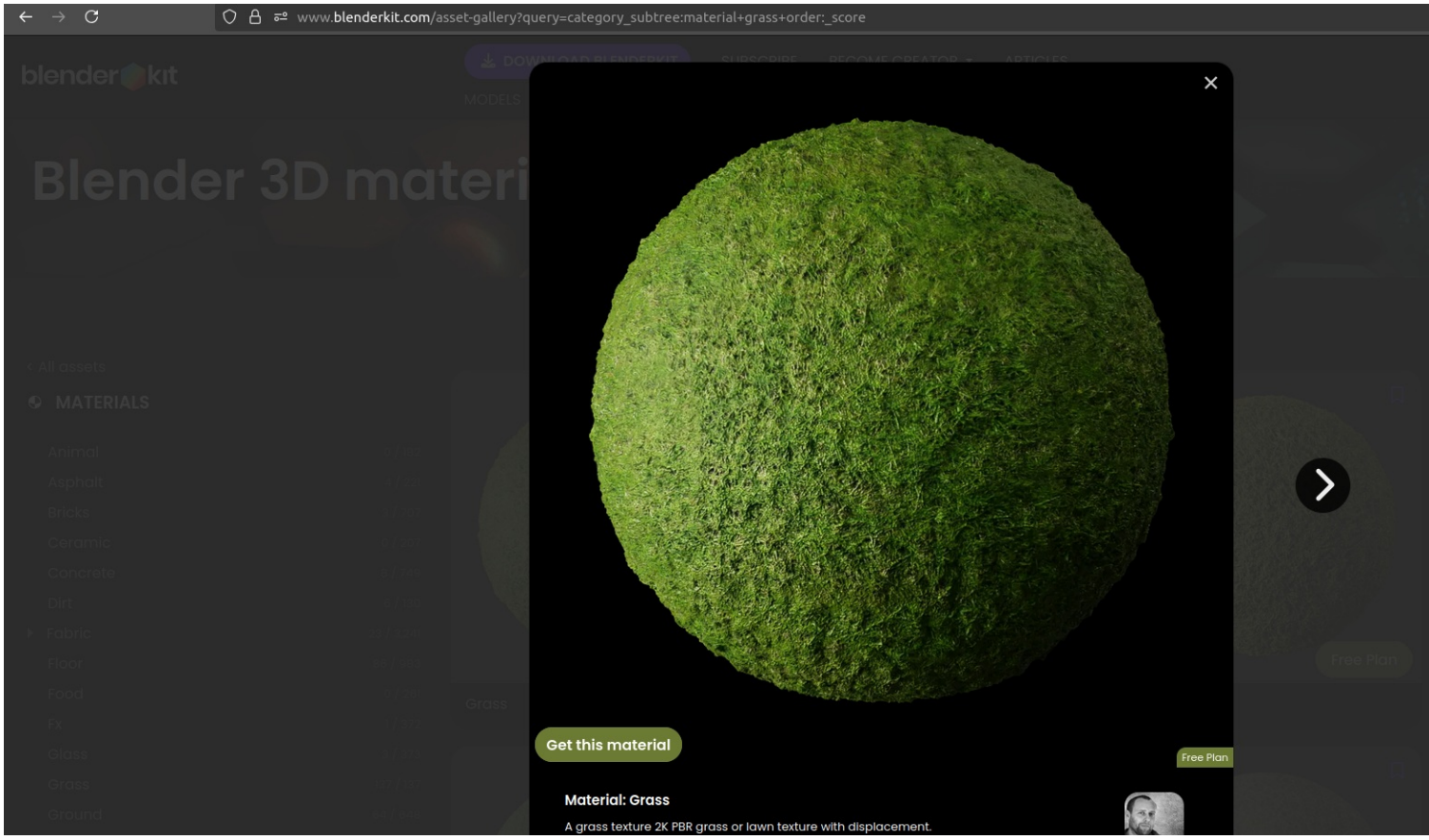


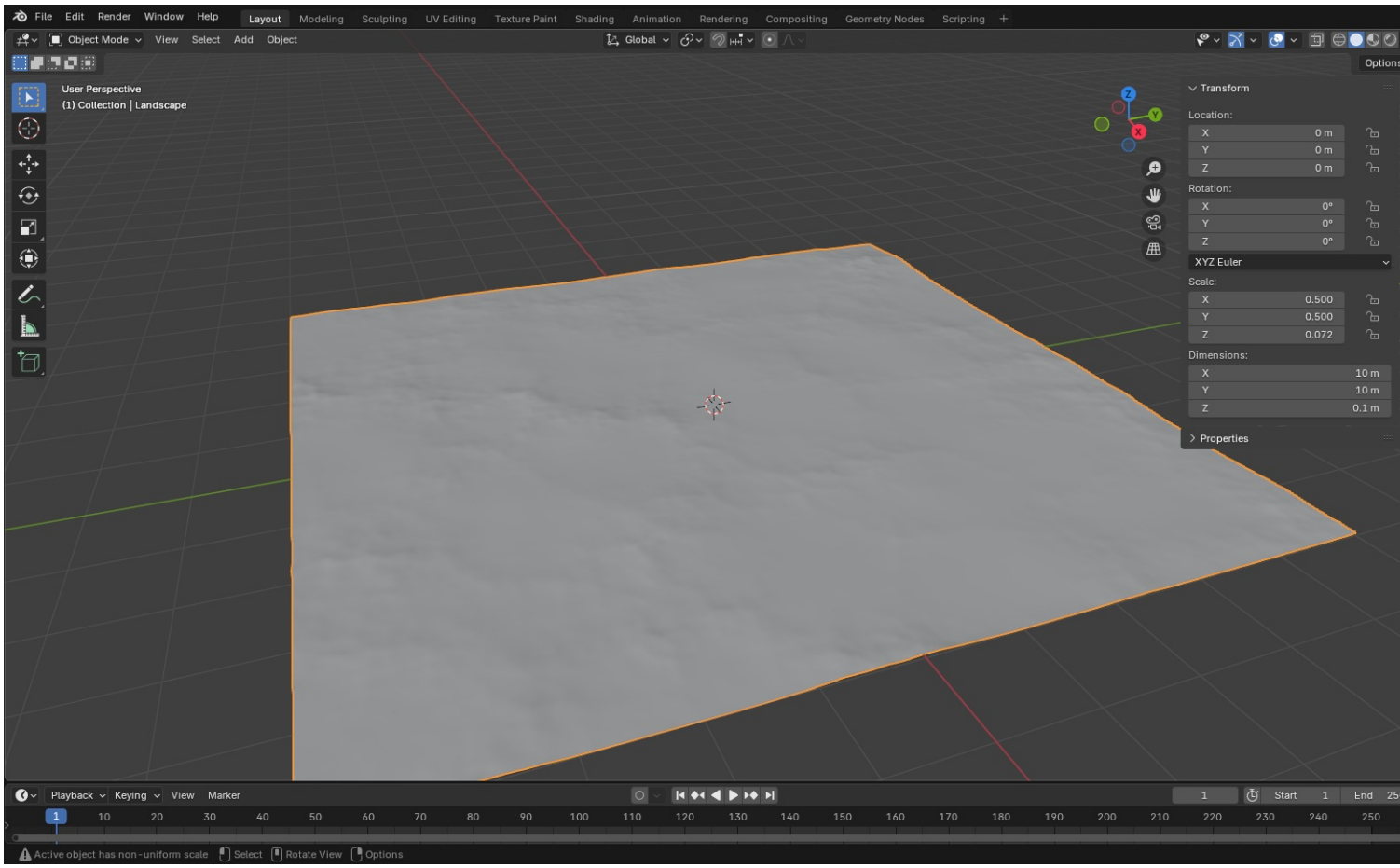
Action Item 2: Creating Terrain with Hills and Dunes Presets – 3 hour(s).

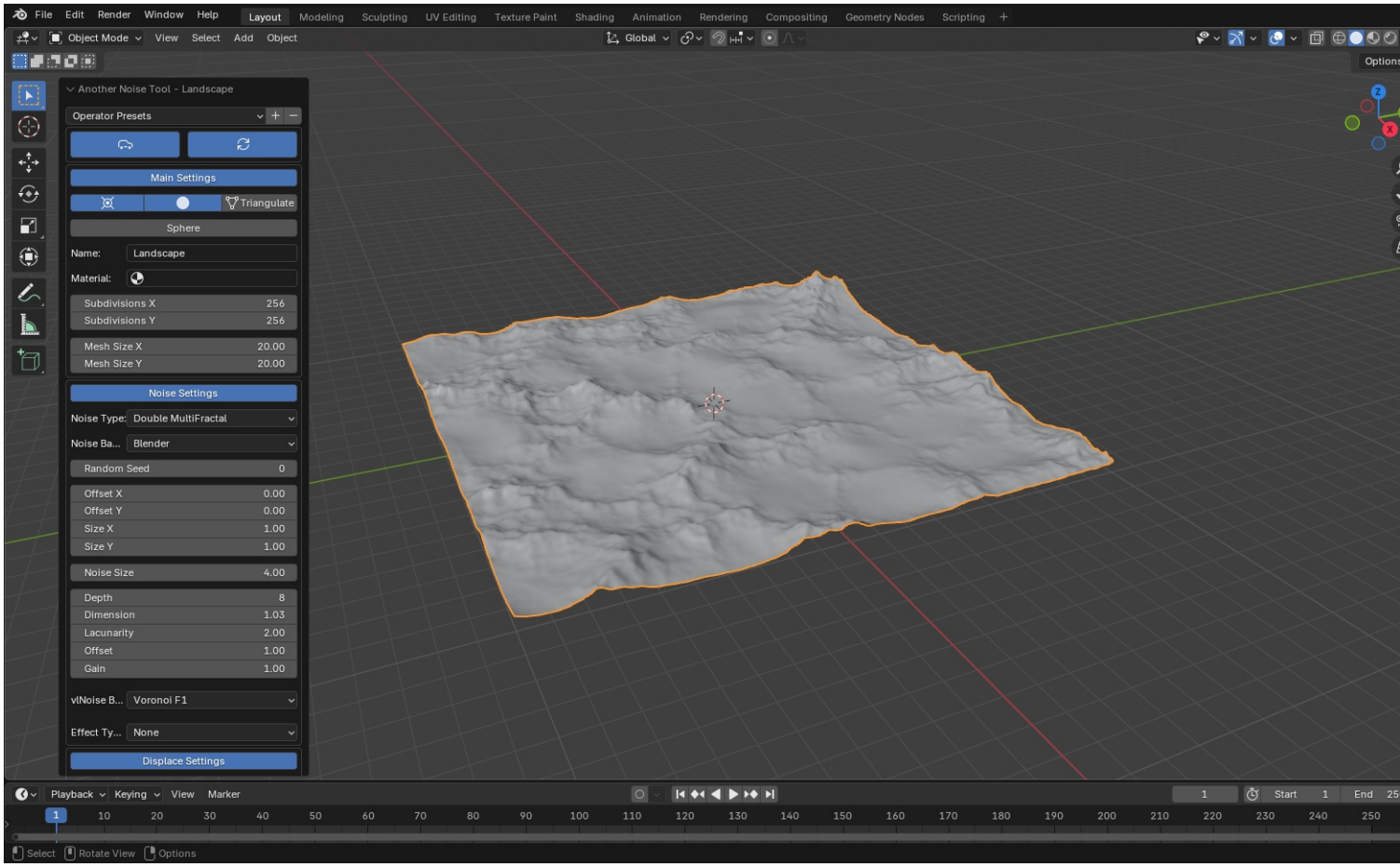
Project Work Summary

- With the A.N.T. Landscape add-on successfully installed, I was ready to create the ideal terrain for our Husky robot simulation. The goal was to craft a realistic landscape with the right dimensions and subtle elevation changes that would challenge the robot without making navigation impossible.
- I started by adding a new landscape to my scene using Shift+A → Mesh → Landscape. Immediately, the add-on generated a default terrain and displayed an operator panel in the bottom-left corner with all the customization options - perfect for tailoring the terrain to our specific needs.
- In the operator panel, I first looked for the Presets dropdown menu, which offers several terrain types. I experimented with both the "Hills" and "Dunes" presets, finally settling on "Dunes" as it provided gentler slopes that would be ideal for our Husky robot to navigate without excessive strain on its motors.
- For dimensions, I needed to ensure our terrain matched our planned simulation size. I set both the X and Y dimensions to 10 meters in the Mesh Size section, giving us a perfect 10x10m square terrain that would provide adequate testing space while keeping the simulation computationally efficient.
- The height scale was crucial for our Husky robot - too high and it would struggle to climb, too flat and it wouldn't challenge the robot's capabilities. I adjusted the Height Scale parameter down to 0.1m, ensuring the bumps and ridges were subtle enough for the Husky to traverse while still providing realistic terrain challenges.
- To add variation to the landscape, I played with the Random Seed value, clicking the randomize button several times until I found a pattern of dunes and small hills that looked natural and provided interesting navigation challenges. Each click generated a completely different terrain configuration, allowing me to "shop" for the perfect layout.
- With the basic terrain configured, I turned my attention to texturing. I opened the BlenderKit add-on (which I had previously installed) by clicking on the BlenderKit icon in the sidebar. In the search field, I typed "PBR grass texture" and browsed through the high-quality material options that appeared, looking for something that would give our farmland a realistic appearance.
- After downloading a suitable grass texture from BlenderKit, I applied it to my terrain by selecting the landscape, switching to the Material Properties tab, and assigning the new PBR grass material. The texture immediately transformed our simple mesh into a more convincing representation of farmland, complete with the detailed blade structures and color variations found in natural grass.




www.blenderkit.com/asset-gallery?query=category_subtree:material+grass+order:_score
Get this material
Free Plan
Material: Grass
A grass texture 2K PBR grass or lawn texture with displacement.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
User Perspective
(1) Collection | Landscape
Transform
Location:
Rotation:
XYZ Euler
Scale:
Dimensions:
Properties
Playback Keying View Marker
Start 1 End 25
Active object has non-uniform scale
Select
Rotate View
Options



Action Item 3: Automatically Annotated Dataset of a Ground Mobile Robot in Natural Environments – 3 hour(s).

Research

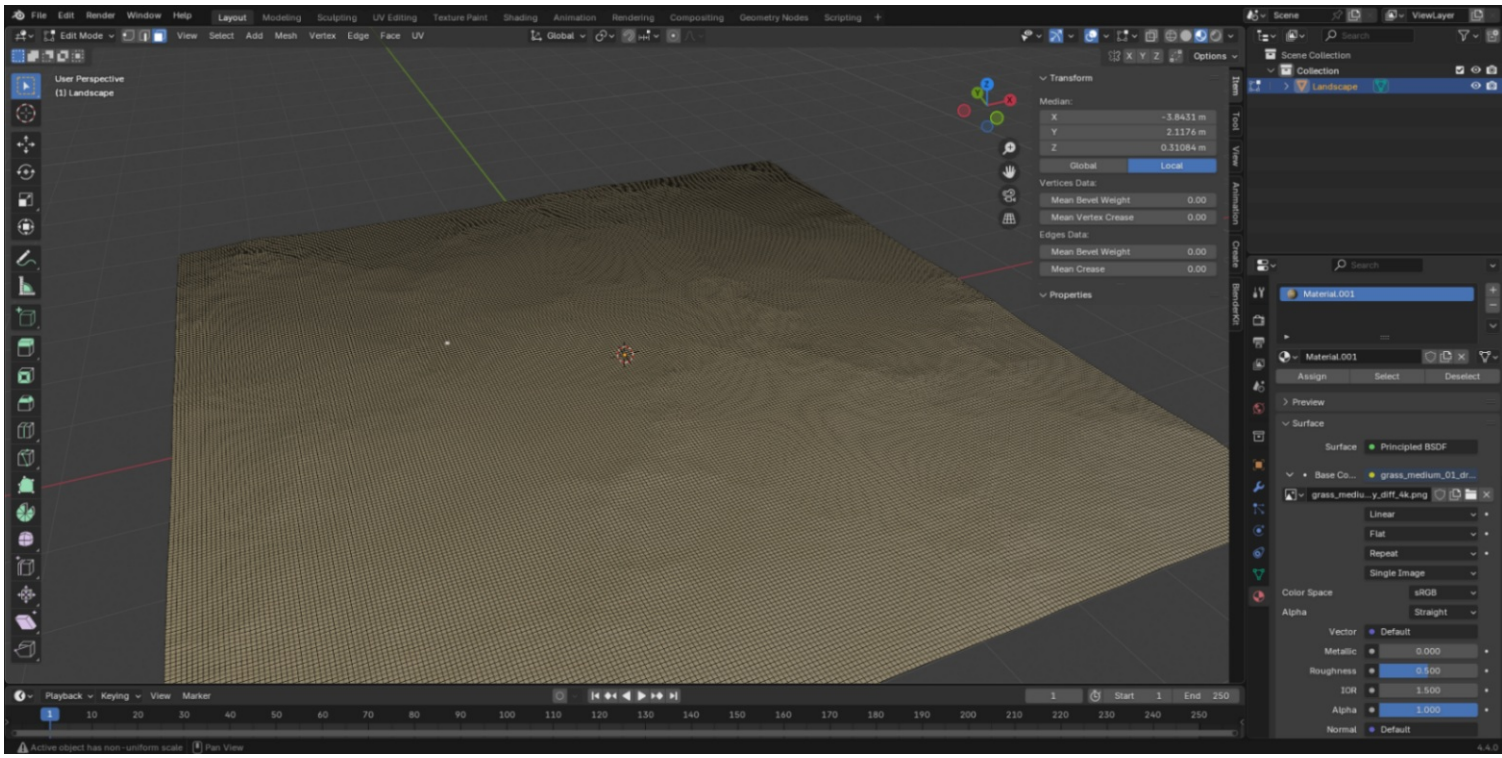
- https://pmc.ncbi.nlm.nih.gov/articles/PMC9331783/
- *Automatically Annotated Dataset of a Ground Mobile Robot in Natural Environments*
- Summary of Report
  - The paper introduces a synthetic dataset created using Gazebo simulations of a Husky UGV navigating natural terrains. By leveraging ROS, the authors synchronized sensor data (LiDAR, stereo camera, GNSS, IMU, wheel encoders) and automated labeling of terrain features.
  - This approach ensures high-fidelity ground-truth data for training and validating navigation algorithms. The integration of Gazebo's ODE physics engine with ROS topics like `/imu/data` and `/navsat/fix` enables realistic emulation of outdoor navigation challenges, such as uneven terrain and vegetation interference.
  - The Husky robot's GNSS receiver was simulated using Gazebo's `NavSat` plugin, which publishes geodetic coordinates via ROS topics. The authors configured the sensor to output WGS84 coordinates at 2 Hz, aligning with real-world GPS update rates.
  - To ensure accuracy, they defined spherical coordinates in the Gazebo world file, setting reference latitude/longitude values to match the simulation's origin. This setup mimics how GNSS data would be used in real farmland navigation, where global positioning is critical for path planning.
  - A standout feature of this work is the automatic annotation of LiDAR points and camera pixels. By assigning unique reflectivity values and flat colors to objects (e.g., grass, soil, crops), the team generated labeled datasets without manual effort. For example, crop rows were tagged with specific intensity values in LiDAR scans, enabling direct use in supervised learning pipelines for semantic segmentation or obstacle detection.
- Relation to Project
  - The paper's methodology directly applies to testing GNSS modules in agricultural environments. By replicating their Gazebo world setup—defining reference coordinates and integrating noise-free GNSS data—we can simulate how a robot would navigate crop rows while maintaining global positioning accuracy.
  - The automatic labeling technique solves a key challenge in agricultural robotics: obtaining annotated training data for crops and terrain. By adopting their approach, we could generate custom datasets for farmland-specific navigation tasks, such as distinguishing between crop rows and irrigation ditches.
  - The authors' use of ROS topics (e.g., `/os1_cloud_node/points` for LiDAR) demonstrates how to decouple sensor data processing from simulation. This modularity allows us to swap GNSS plugins or test alternative localization algorithms without restructuring the entire system.
- Motivation for Research
  - Farmland navigation requires robust systems capable of handling unpredictable terrain and GPS signal drift. This paper provides a blueprint for stress-testing GNSS-dependent algorithms in Gazebo before real-world deployment, reducing risks of field failures.
  - The authors highlight the scarcity of labeled datasets for natural environments. Their work motivates the creation of farm-specific synthetic data, which is critical for training ML models to recognize crops, soil types, and obstacles.
  - By open-sourcing their Gazebo worlds and ROS configurations, the paper encourages collaboration. Researchers can build upon their work to refine GNSS integration or test multi-robot coordination in shared farmland simulations.
  -

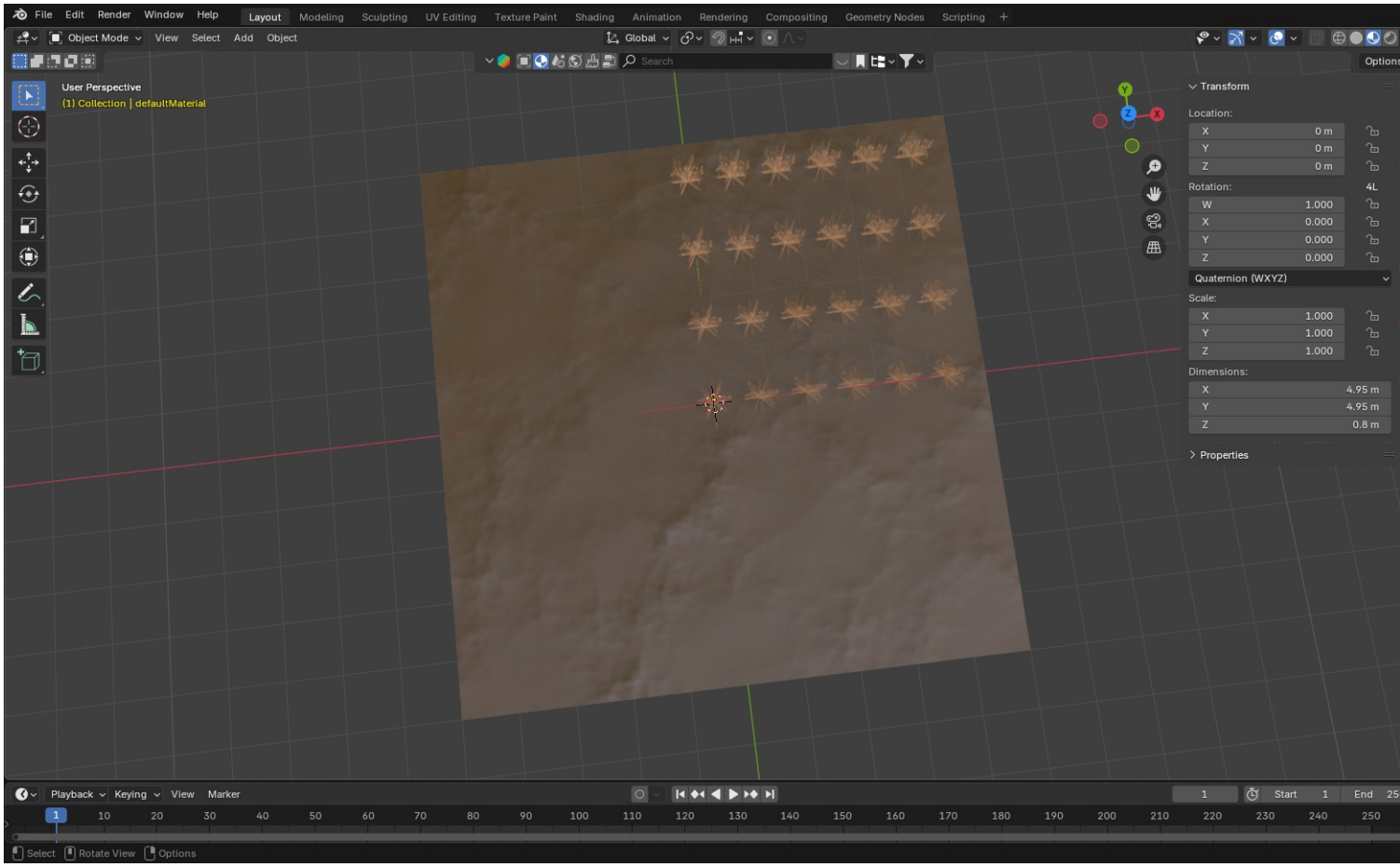
Action Item 4: Creating Realistic Wheat Fields with Particle Systems – 3 hour(s).

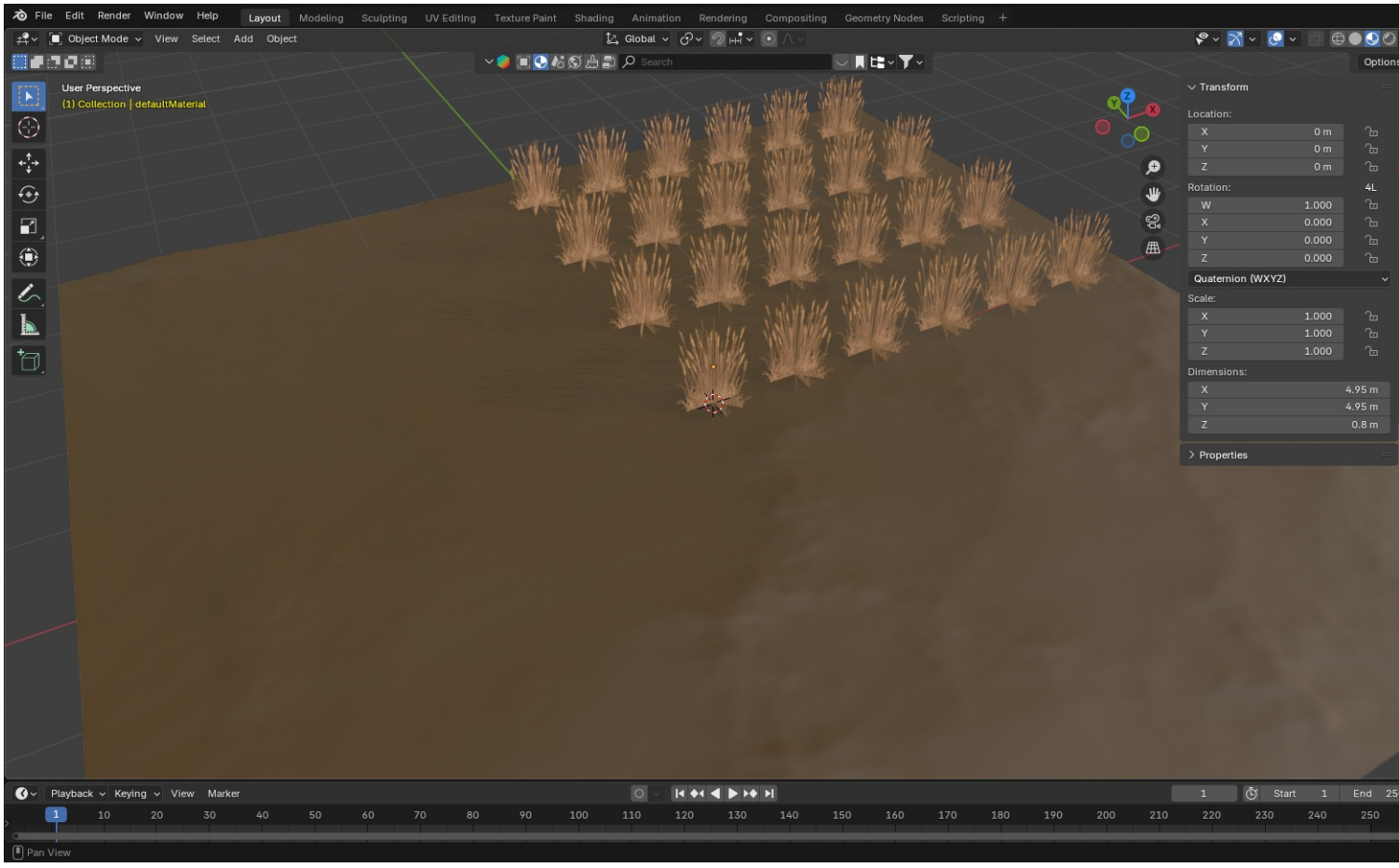
Project Work Summary

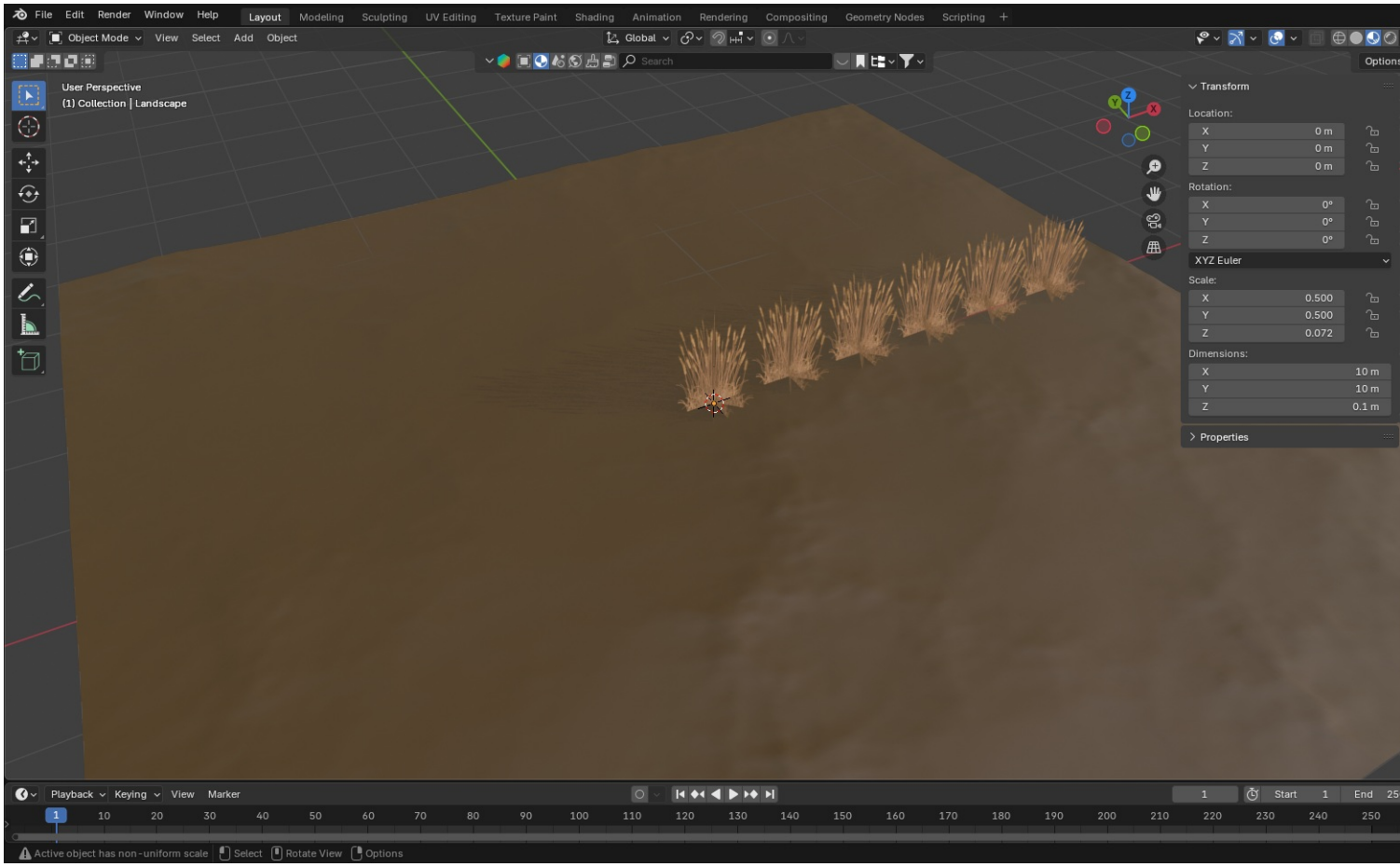
- After establishing my base terrain, I needed to add realistic wheat crops in organized rows. I began by importing a pre-made wheat model into Blender using File > Import, navigating to my downloaded asset folder, and selecting the wheat OBJ file. I immediately noticed its strong yellow-golden color would give our farmland that distinctive harvest-ready appearance.
- Before distributing wheat across the terrain, I needed to optimize the model. I selected the wheat and pressed Tab to enter Edit Mode, where I reduced some of the polygon count using Edge Dissolve on non-essential geometry. This optimization would be crucial later when thousands of wheat instances would be rendered simultaneously.
- I positioned a single wheat model exactly where I wanted it—just slightly above the ground plane to avoid intersection issues. Using the Move tool (G key), I carefully adjusted its Z-position so the roots appeared to emerge naturally from the soil rather than floating above or sinking below the surface.
- Now for the exciting part—creating the particle system! I selected my terrain, navigated to the Particle Properties panel, and clicked the "+" button to add a new particle system. I renamed it "WheatField" for better organization and changed the Render Type from "Halo" to "Object" since I wanted to use my wheat model.

- Under the Render section, I set the Instance Object to my wheat model. I adjusted the Scale value to 0.8 to create some size variation in my wheat stalks. To further enhance realism, I enabled Random Size and set it to 0.2, giving natural variation to my wheat field.
- Creating organized rows was my next challenge. I added a black and white striped texture as a density map in the Texture panel and linked it to my particle system. By adjusting the texture's scale and rotation, I was able to control exactly where wheat would and wouldn't grow, naturally forming those distinct agricultural rows that define farmland.
- Fine-tuning the particle count was crucial—too few stalks would look sparse and unrealistic, while too many would kill performance. I settled on 10,000 particles after experimenting, which provided dense wheat rows without overloading my system when I hit Alt+A to preview the animation.
- As a final touch, I adjusted the particle rotation settings, enabling "Rotation" and "Random" options to create slight variations in how each wheat stalk was oriented. This small detail made the difference between a natural-looking field and an artificially uniform one.




File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
Search
Options
User Perspective
(1) Collection | defaultMaterial
Transform
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
4L
W 1.000
X 0.000
Y 0.000
Z 0.000
Quaternion (WXYZ)
Scale:
X 1.000
Y 1.000
Z 1.000
Dimensions:
X 4.95 m
Y 4.95 m
Z 0.8 m
Properties
Playback Keying View Marker
1
Start 1
End 25
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250
Select Rotate View Options

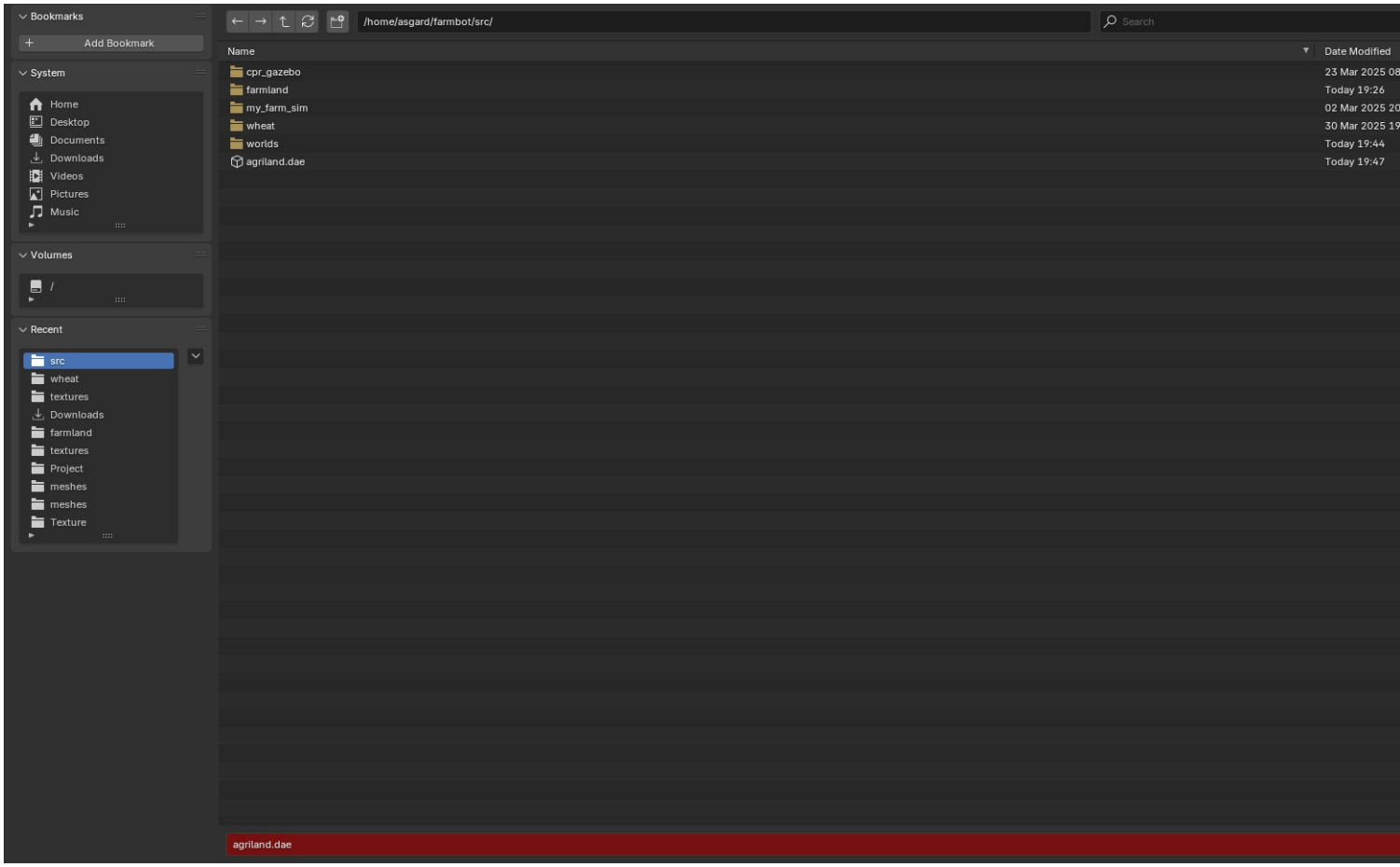

File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
Global
Search
Options
User Perspective
(1) Collection | defaultMaterial
Transform
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
4L
W 1.000
X 0.000
Y 0.000
Z 0.000
Quaternion (WXYZ)
Scale:
X 1.000
Y 1.000
Z 1.000
Dimensions:
X 4.95 m
Y 4.95 m
Z 0.8 m
Properties
Playback Keying View Marker
1
Start 1
End 25
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250
Pan View

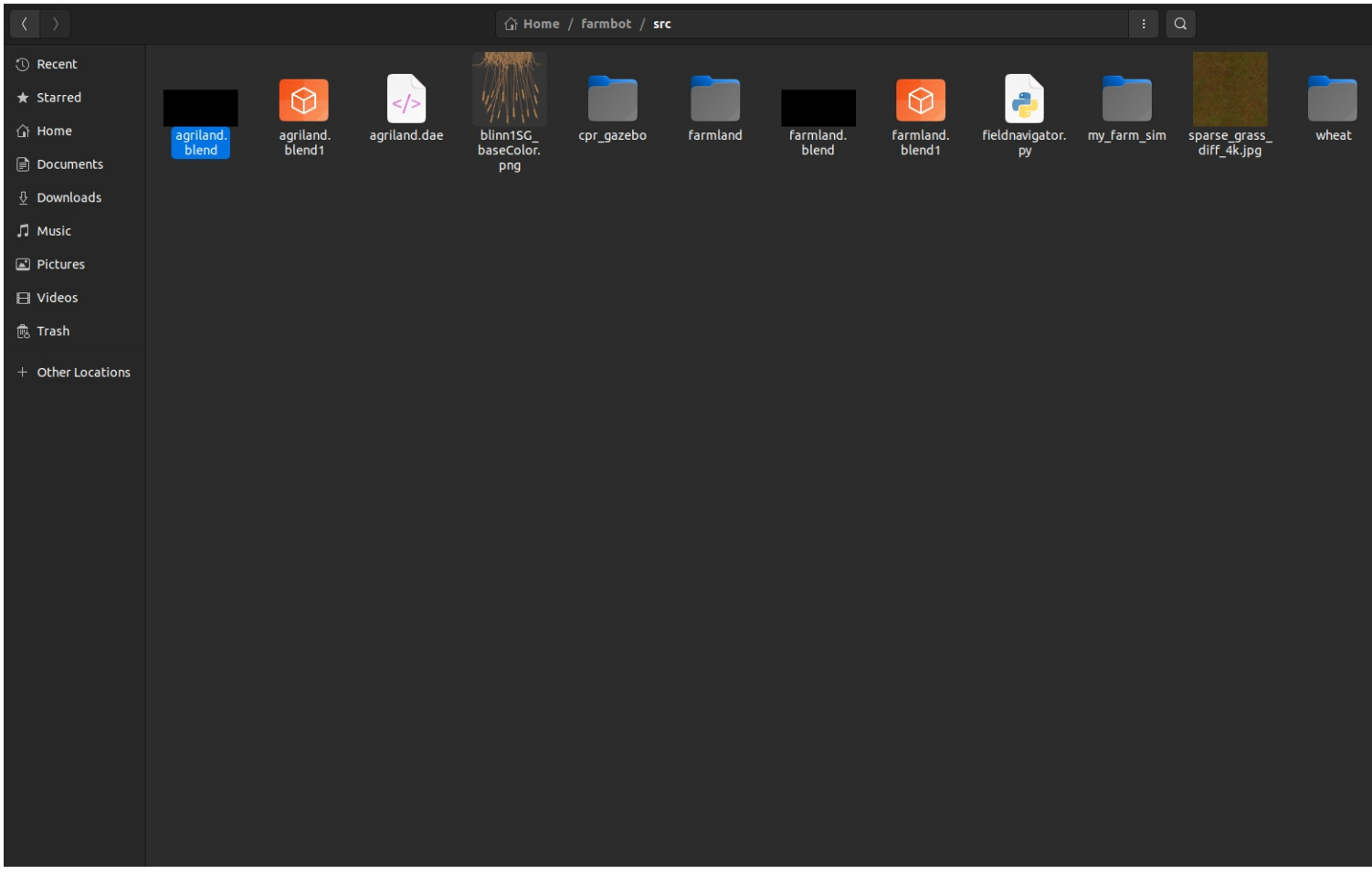


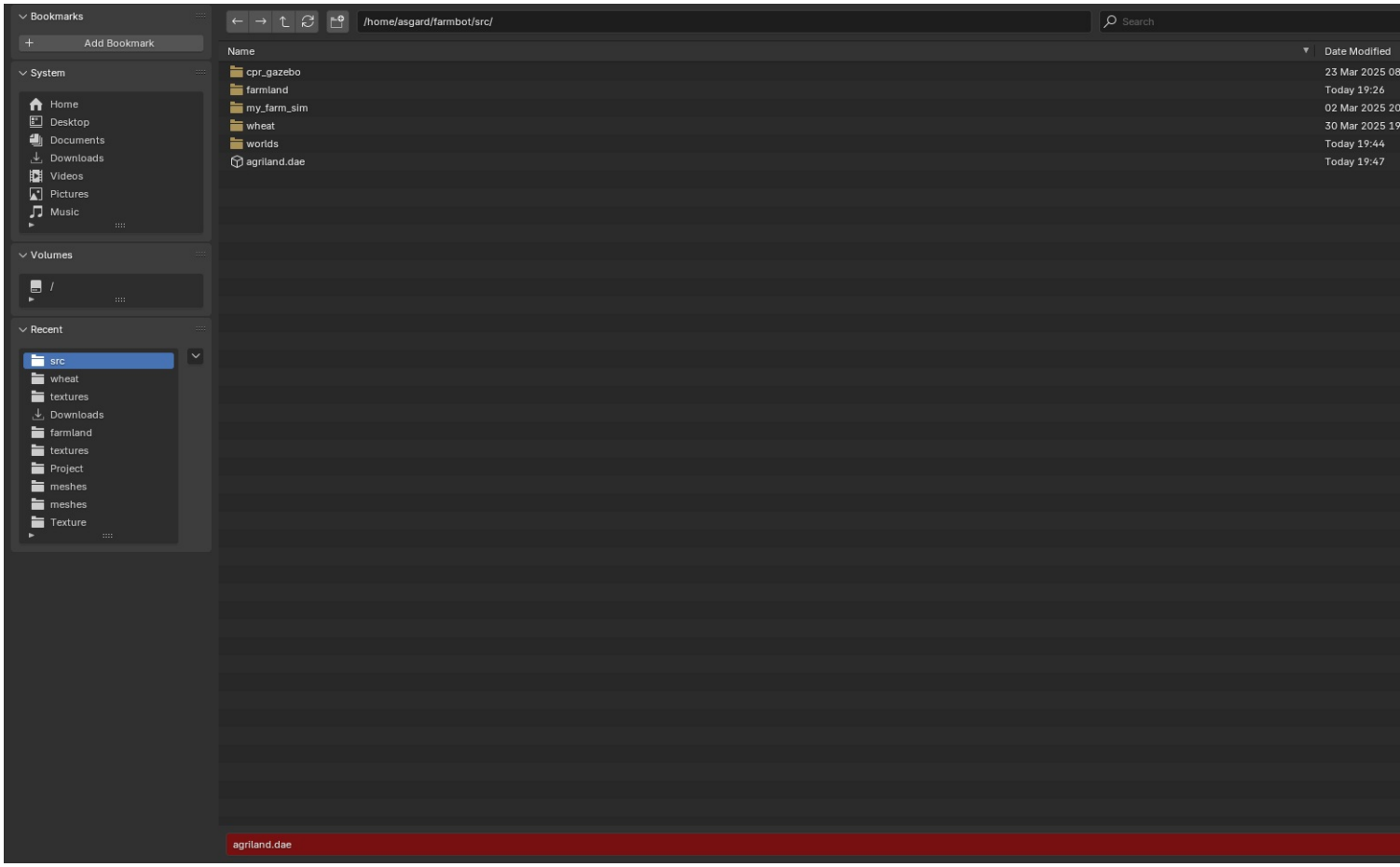
Action Item 5: Exporting the Farmland Scene from Blender to Gazebo – 3 hour(s).

Project Work Summary

- With my farmland scene complete, I needed to prepare everything for export to Gazebo. First, I pressed A to select all objects in my scene, including the terrain, wheat particles, and any other elements I'd added. I then verified in the Outliner panel that everything I needed was indeed selected.
- Before exporting, I needed to apply all modifiers and particle systems. I right-clicked on my terrain object and selected "Convert to Mesh" for the particle system. This transformed the wheat from a particle simulation to actual geometry that could be exported properly—a crucial step that I've learned the hard way!
- Since Gazebo uses a different coordinate system than Blender, I needed to correct my object orientations. I selected all objects and pressed Ctrl+A, then chose "All Transforms" from the menu. This baked all rotations, scales, and positions into the mesh data, preventing any unexpected transformations during export.
- For materials and textures, I needed to ensure Gazebo would recognize them. I went to File > External Data > Pack All Into .blend to embed all my textures in the file. Then I checked each material to make sure it used simple diffuse shaders that would translate well to Gazebo's rendering system.
- Now for the actual export! I navigated to File > Export > Collada (.dae), as this format preserves materials and object hierarchies needed for Gazebo. In the export dialog, I enabled options for "Selection Only" (to export only what I'd selected) and "Include Material Textures" to preserve my carefully crafted farmland appearance.
- I carefully named my export file "farmland.dae" and saved it directly to my designated Gazebo models directory (~/gazebo/models/farmland/) to maintain a clear organizational structure. This would make importing into Gazebo much more straightforward.
- After exporting, I opened the DAE file in a text editor to verify that all texture paths were relative rather than absolute—a common issue that can break texture references in Gazebo. I adjusted any problematic paths to ensure they'd work correctly in the target environment.


Bookmarks
Add Bookmark
System
Home
Desktop
Documents
Downloads
Videos
Pictures
Music
Volumes
/
Recent
src
wheat
textures
Downloads
farmland
textures
Project
meshes
meshes
Texture
/home/asgard/farmbot/src/
Search
Name
Date Modified
cpr_gazebo
23 Mar 2025 08
farmland
Today 19:26
my_farm_sim
02 Mar 2025 20
wheat
30 Mar 2025 19
worlds
Today 19:44
agriland.dae
Today 19:47
agriland.dae

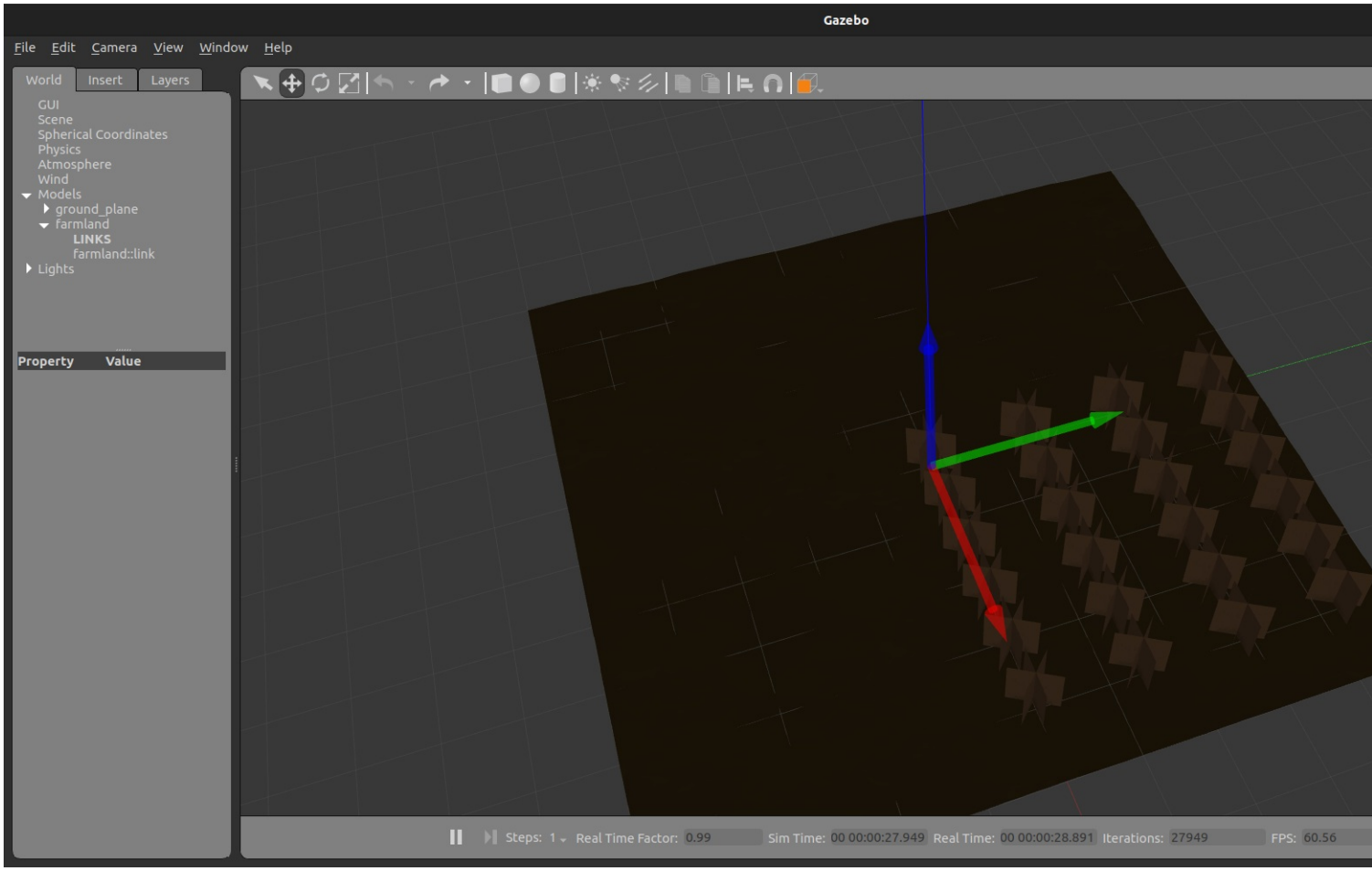

Home / farmbot / src
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
agriland.blend
agriland.blend1
agriland.dae
blinn1SG_baseColor.png
cpr_gazebo
farmland
farmland.blend
farmland.blend1
fieldnavigator.py
my_farm_sim
sparse_grass_diff_4k.jpg
wheat

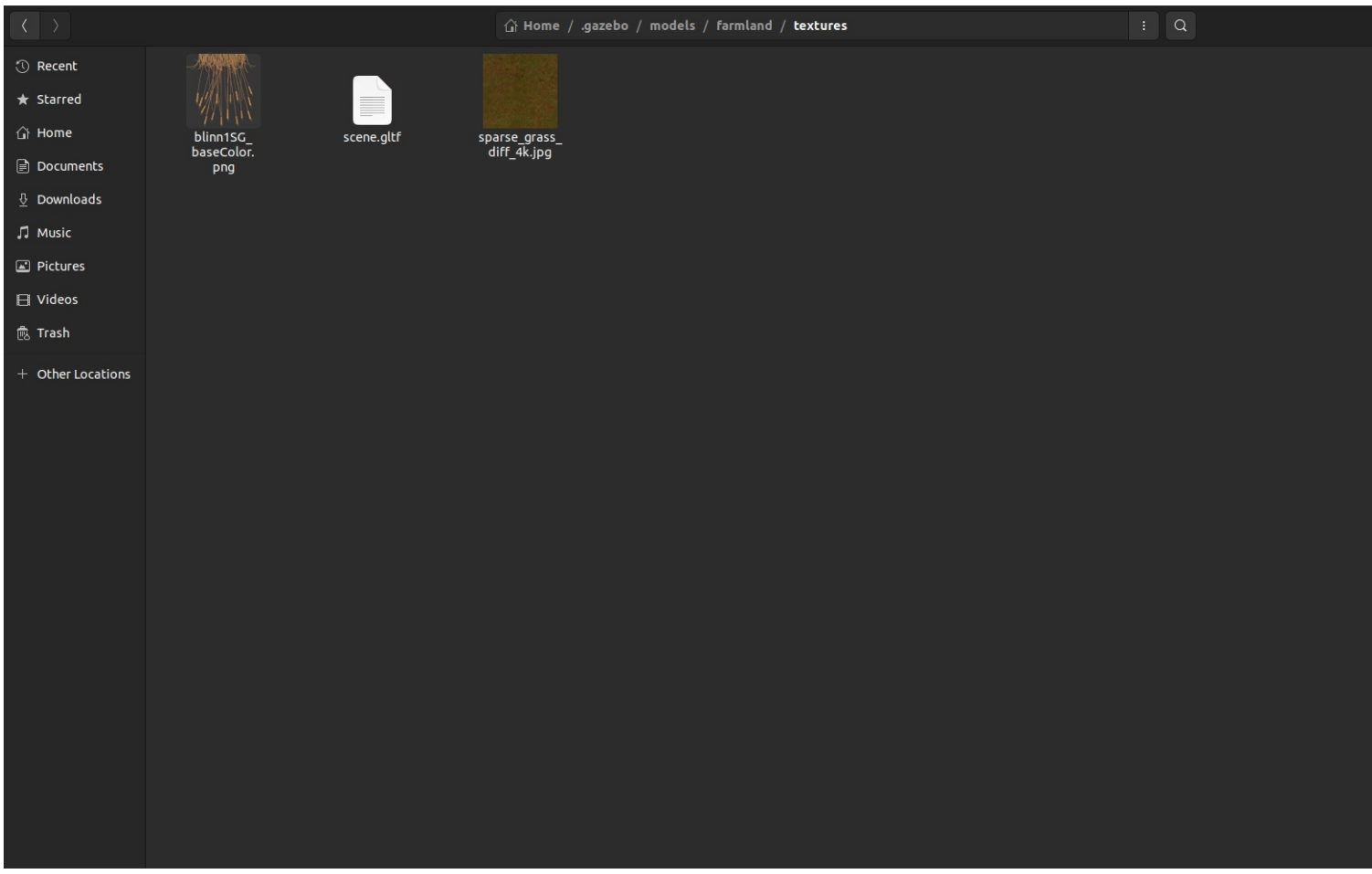


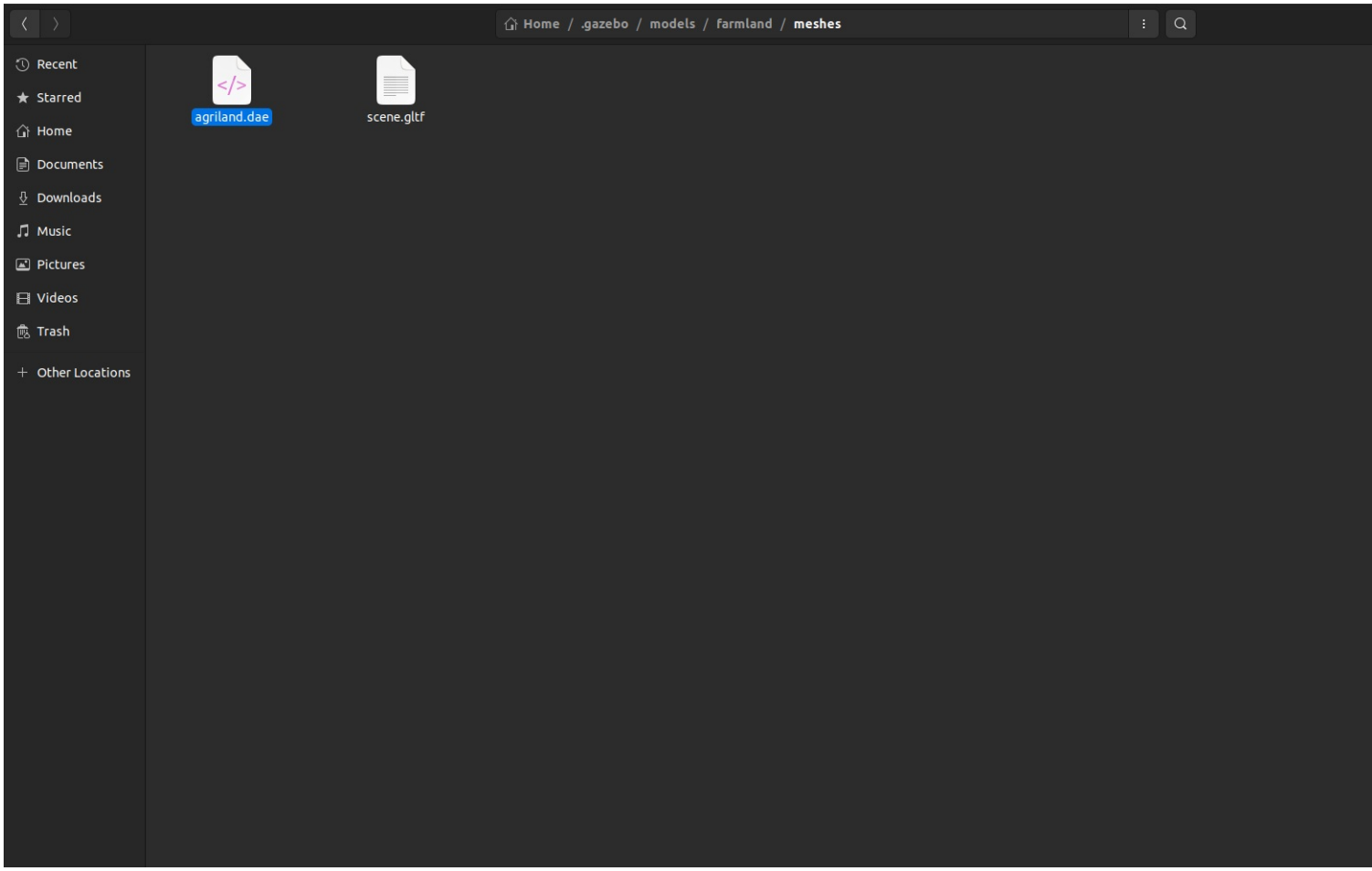
Action Item 6: Setting Up Gazebo for Farmland Simulation – 3 hour(s).

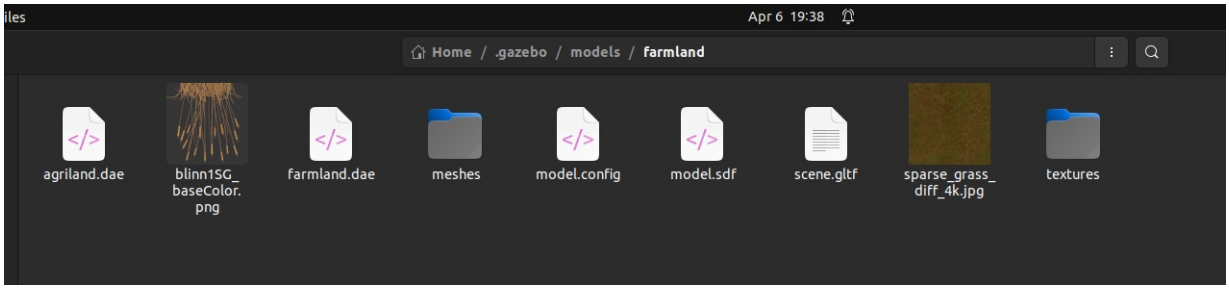
Project Work Summary

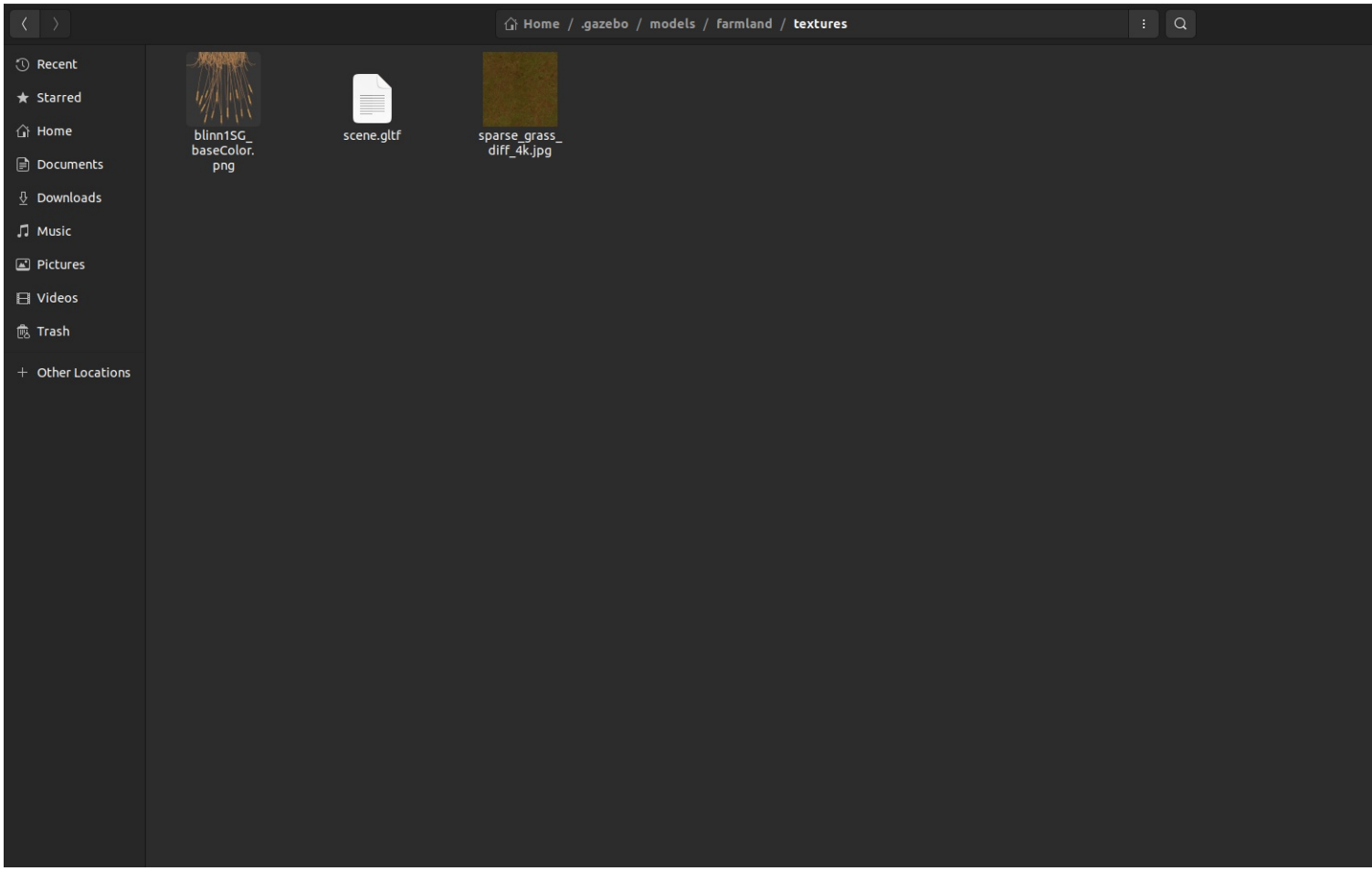
- With my farmland model exported, I needed to set up the proper directory structure in Gazebo. I opened a terminal and created a new directory in the Gazebo models folder using mkdir -p ~/.gazebo/models/farmland. This would house all my farmland-related files in a location Gazebo recognizes automatically.
- Next, I copied my exported DAE files to this location with cp ~/Downloads/farmland.dae ~/.gazebo/models/farmland/. I also needed to ensure any additional meshes were properly organized, so I created a meshes subdirectory with mkdir ~/.gazebo/models/farmland/meshes and moved appropriate files there.
- Textures are crucial for realistic simulation, so I created a textures directory with mkdir ~/.gazebo/models/farmland/textures. I copied my wheat texture and grass texture files into this folder, maintaining the same relative paths referenced in my DAE file to prevent broken texture links.
- Every Gazebo model needs a model.config file, so I created one using nano: nano ~/.gazebo/models/farmland/model.config. Inside, I added the XML metadata including the model name, version, description, and author information. This metadata helps Gazebo properly identify and categorize my model.
- The SDF file is what actually defines how the model appears in Gazebo. I created model.sdf with nano ~/.gazebo/models/farmland/model.sdf and carefully crafted the XML structure to include my mesh, collision properties, and visual elements. I made sure to reference the correct paths to my DAE file and textures.
- For convenient loading, I created a custom world file: mkdir -p ~/.gazebo/worlds followed by nano ~/.gazebo/worlds/farmland.world. In this file, I included references to standard elements like the sun and ground plane, along with my custom farmland model, positioned at the origin for easy access.
- To test everything, I launched Gazebo with my custom world using gazebo ~/.gazebo/worlds/farmland.world. The moment of truth came when Gazebo loaded, and I saw my carefully crafted farmland appear exactly as I had designed it, with realistic wheat rows stretching across the terrain.
- As a final step, I created a simple launch file for ROS integration, allowing me to easily spawn a Husky robot onto my farmland for navigation tests. I created a new package in my catkin workspace and added a launch file that combined my world with the Husky robot's launch configuration, ready for our simulation experiments.


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
ground_plane
farmland
LINKS
farmland::link
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:00:27.949 Real Time: 00 00:00:28.891 Iterations: 27949 FPS: 60.56


Home / .gazebo / models / farmland / textures
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
blinn1SG_ baseColor. png
scene.gltf
sparse_grass_ diff_4k.jpg


Home / .gazebo / models / farmland / meshes
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
agriland.dae
scene.gltf


Apr 6 19:38
Home / .gazebo / models / farmland
agriland.dae
blinn1SG_ baseColor. png
farmland.dae
meshes
model.config
model.sdf
scene.gltf
sparse_grass_ diff_4k.jpg
textures



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

Project Work Summary

- I'll start by installing the husky_gazebo ROS package and modifying its URDF to include our farmland world. Using roslaunch husky_gazebo husky_empty_world.launch, I'll verify the robot spawns correctly. If the wheels sink into the terrain, I'll adjust the ground plane friction parameters in the .world file to match our farmland's soil properties.
- For initial testing, I'll set up teleoperation using the joy package with a USB controller. If the robot drifts on uneven terrain, I'll tweak the husky_control PID gains in /etc/ros/urdf/husky_control.yaml to improve stability. A backup plan involves using teleop_twist_keyboard for basic WASD control if joystick calibration becomes time-consuming.
- Using Gazebo's libgazebo_ros_gps.so plugin, I'll add a GNSS module to the Husky's roof rack in its URDF. To simulate real-world inaccuracies, I'll inject Gaussian noise (σ=2m) into the /navsat/fix topic using a ROS node. Testing will involve driving the robot along a 10m straight path while logging GNSS data to assess drift patterns.
- I'll configure the robot_localization package's ekf_localization_node to fuse IMU, wheel odometry, and GNSS data. The challenge will be tuning the covariance matrices—initially trusting GNSS more outdoors ([0.5, 0.5, 1.0] for XYZ variances) but switching to odometry if signal dropout occurs under simulated tree canopies.
- Using the move_base stack, I'll create a .csv file of GNSS coordinates corresponding to crop rows. The geographic_msgs package will help convert these to map-frame coordinates. If the robot veers off course, I'll adjust the base_local_planner parameters to prioritize smoother turns on uneven terrain.
- Since crops are traversable but should be avoided, I'll define a custom voxel_grid layer in the costmap that treats vegetation as low-cost obstacles. The obstacle_layer will use Husky's simulated 3D LiDAR to detect non-crop obstacles like irrigation equipment, setting their cost higher.

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.