THE PHYSICS OF WARP ANCESTOR – UNITY TECHNICAL DOCUMENTATION

Preface on the Simulation of Warp Ancestor

Classical Limits, Spatial Constraints, and the Architecture of Simulation

1. The Classical Limitation

Every classical computer – no matter how advanced – is still a finite machine built on electrical signals moving through interconnects. Those signals propagate far below the speed of light because they're bound by:

- Earth's gravity, which dictates structural and thermal design
- Atmospheric oxygen, which enables oxidation and heat transfer
- Hardware temperature ceilings, which cap clock speeds and density

A motherboard is a controlled environment engineered to survive *Earth*, not the universe.

2. The Space Advantage

Move that same architecture into orbit and the constraints shift:

- Gravity becomes negligible (≤ 0.09 g)
- Atmospheric oxygen disappears
- Ambient temperatures drop below -100^{o}C, even in sunlight

Suddenly, the three major bottlenecks of classical computing – gravity, oxygen, and heat – are removed by the universe itself.

Space is not just a frontier for spacecraft. It's a frontier for computation.

3. The Fundamental Constraint

The universe is effectively infinite: an unbounded set of points, each moving along its own trajectory.

A computer is finite: a bounded lattice of transistors and memory cells.

You cannot fit an infinite system inside a finite substrate without abstraction. To attempt otherwise would violate physical reality.

4. The Revelation: Scaling as a Necessity

Every simulation – from orbital mechanics to galactic traversal – must introduce scaling factorization:

- compressing distances
- compressing velocities
- compressing time
- compressing mass and energy domains

This is not a flaw. It is the only way to represent the unrepresentable using the tools we have today.

Warp Ancestor embraces this honestly: the simulation is built on real physics, real ephemeris, and real NASA topography – but scaled through deterministic factorization so the universe can fit inside a finite machine.

5. The Future: Quantum Computing Belongs in Space

Quantum computing will not reach its full potential on Earth. It is fundamentally incompatible with:

- atmospheric noise
- thermal instability
- gravitational interference
- magnetic and vibrational contamination

But in orbit – or deep space – the universe provides:

- natural cryogenic temperatures
- vacuum isolation
- microgravity
- minimal decoherence sources

The next era of computation will be off-planet, built for the environment where quantum systems can finally operate at scale.

THE TECHNICALS OF THE 3 BASE SYSTEMS THAT POWER THE GAME

I

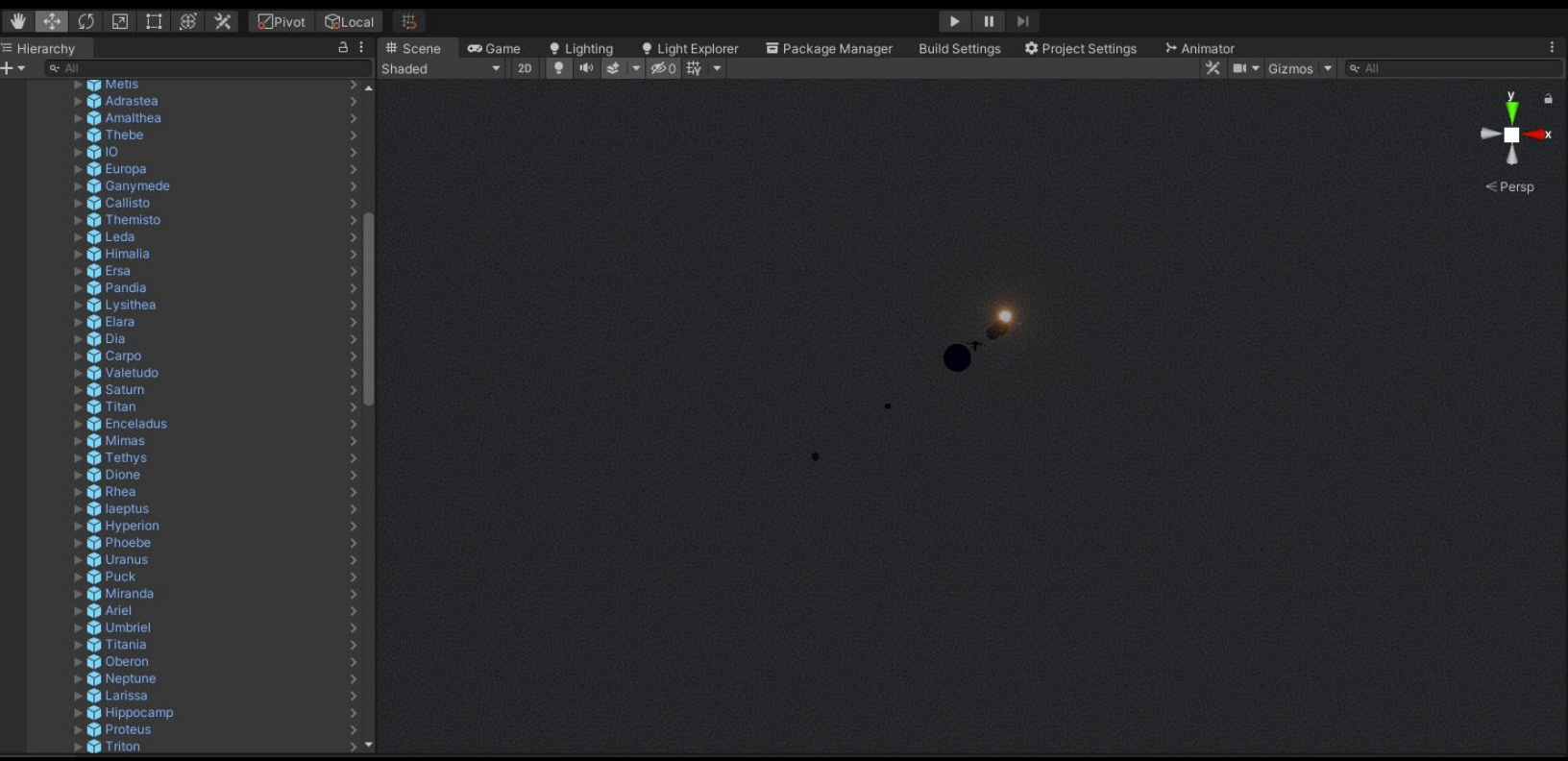
REAL EPHEMERIS…

Ephemeris (Ephemerids):

An ephemeris is a precise table or dataset that provides the calculated positions of astronomical bodies at specific times.

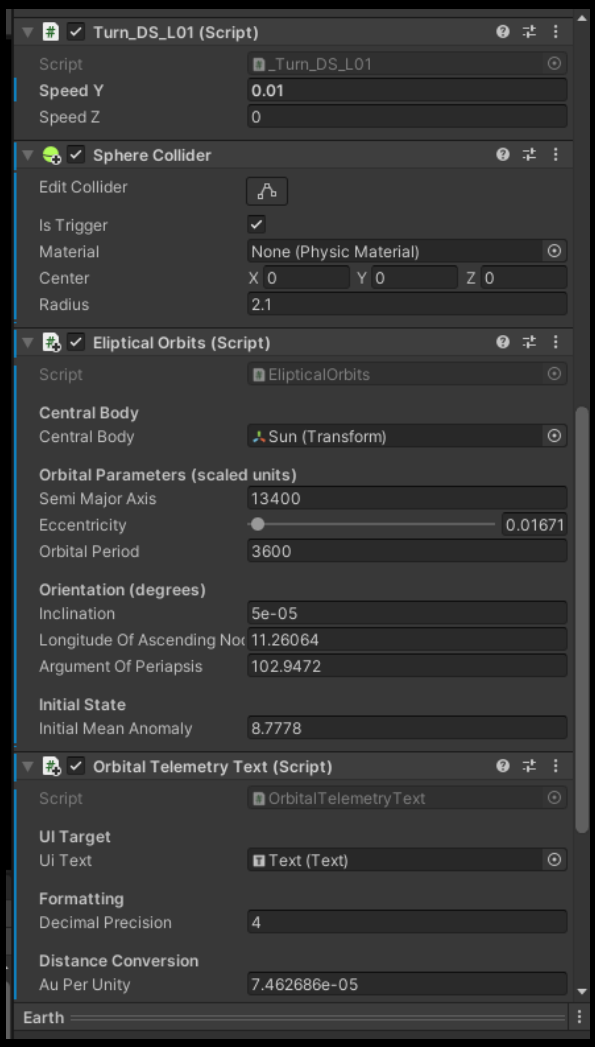
It includes coordinates, velocities, and orbital parameters for objects such as planets, moons, asteroids, and spacecraft, allowing their locations to be predicted or reconstructed for any given moment.

Runs on all spheroids (planets, moons, dwarf planets):



REAL EPHEMERIS

The Orbital Scripts



Turn_*.cs

The self orbiter. A script which allows rotation on the 2 main axes in Unity 3D space, relative to the native orientation of the 3D object.

Essentially simulates centrifugally enhanced gravity.

Sphere Collider

Standard Unity Collider which prevents going through the 3D objects. And allows for the simulation of magnetic field collision effects.

EllipticalOrbits.cs

...

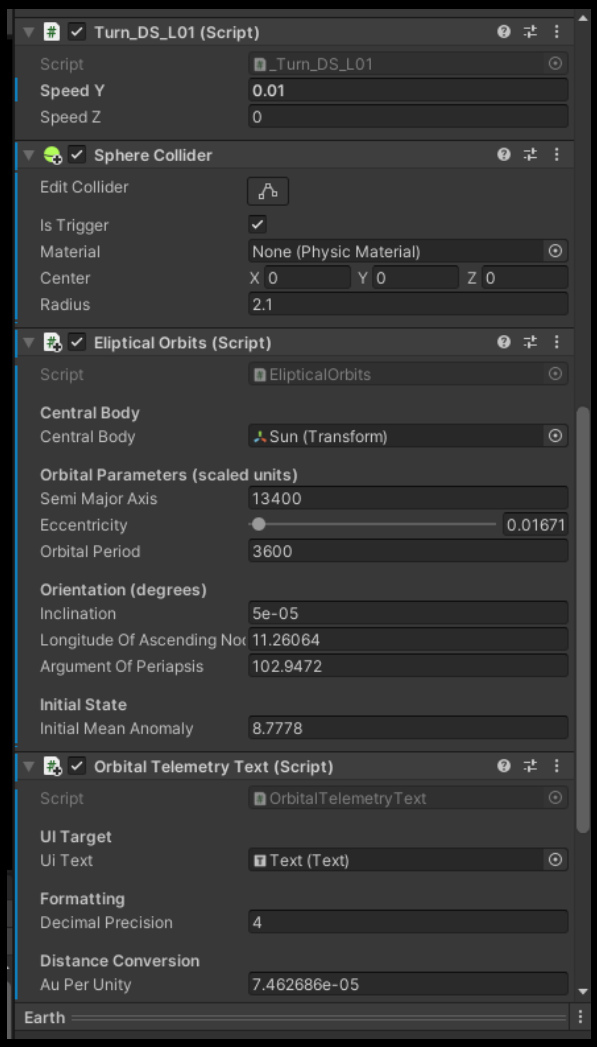
…EllipticalOrbits.cs

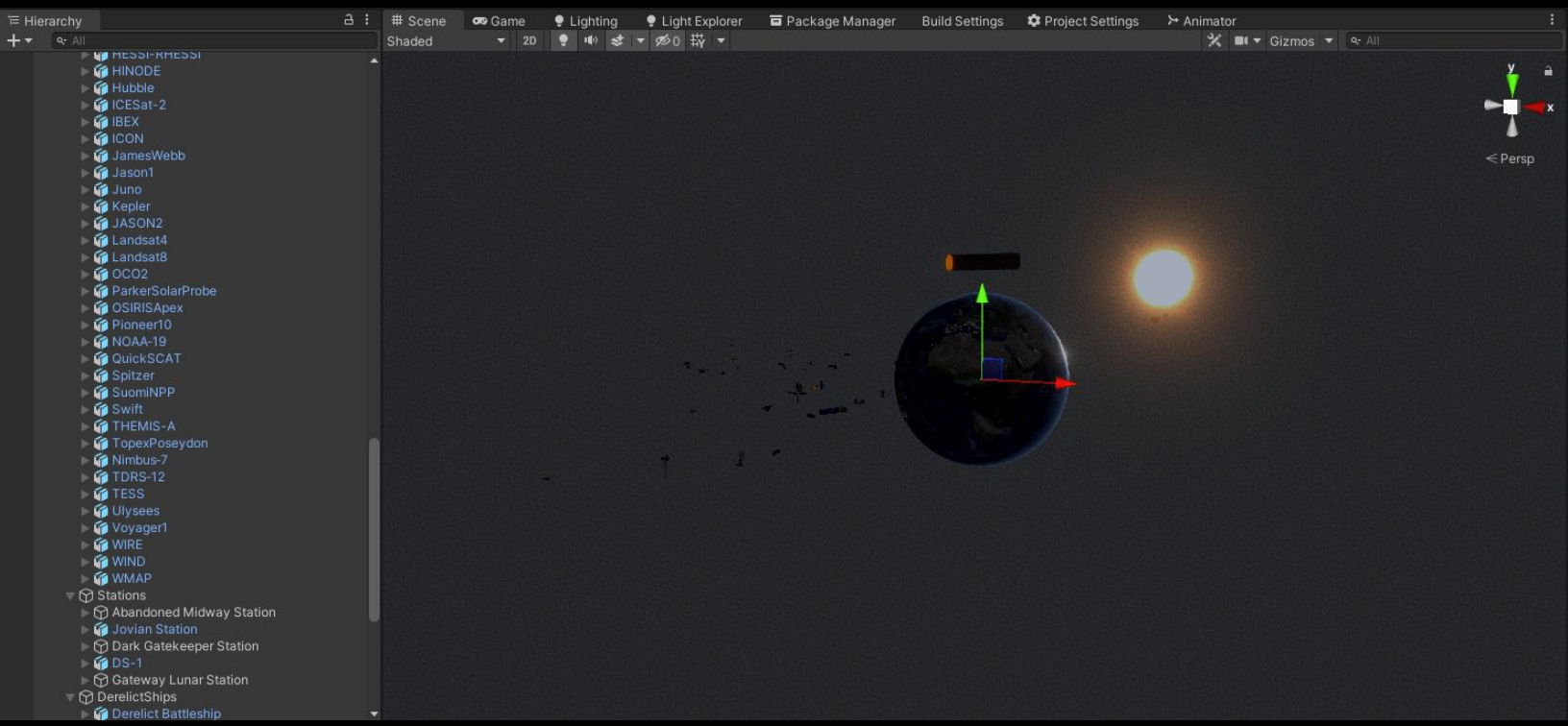
A central-body orbital mechanics component that computes the real-time position of an orbiting object using classical orbital elements. It requires the following parameters:

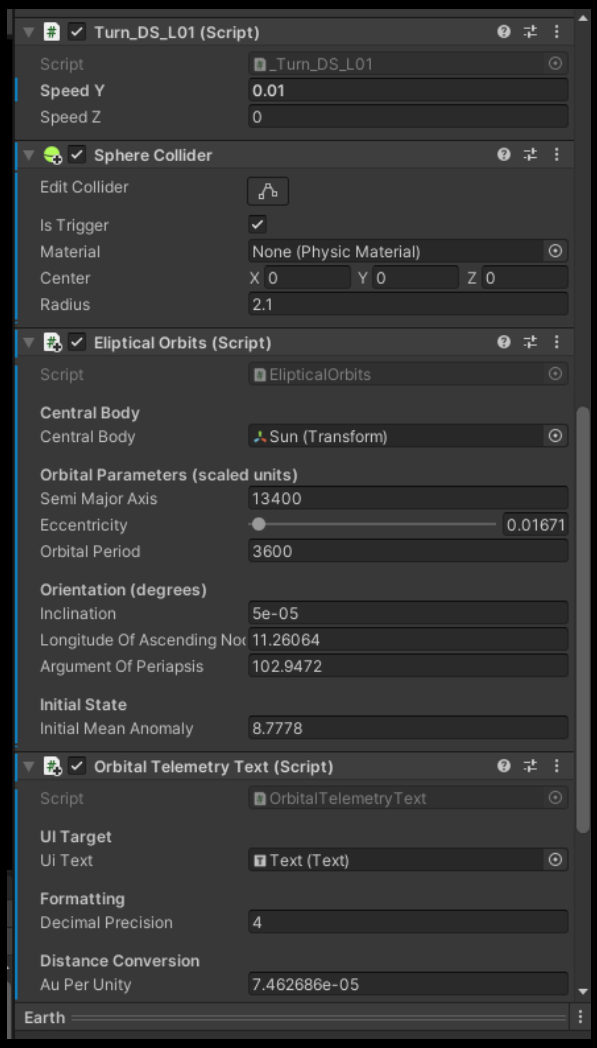
1. Orbital Parameters (scaled units)

These values are fed through the OrbitalTelemetryText.cs system, which applies the distance-conversion factor to translate astronomical units into Unity-scaled space.

- a) Semi-Major Axis The average orbital radius. Defines the size of the ellipse around the central body.
- b) Eccentricity The shape of the orbit.
  - 0 = perfect circle
  - 0 < e < 1 = ellipse Higher values produce more elongated orbital paths.
- c) Orbital Period The time (in scaled seconds) required to complete one full revolution around the central body. This drives the angular velocity of the orbit.







2. Orientation (degrees)

These parameters rotate the orbital plane into its correct 3D orientation relative to the simulation's ecliptic.

- Inclination Tilt of the orbital plane relative to the reference plane.
- Longitude of Ascending Node The angle from the reference direction to the point where the orbit crosses upward through the reference plane.
- Argument of Periapsis The angle from the ascending node to the orbit's closest-approach point.

3. Initial State

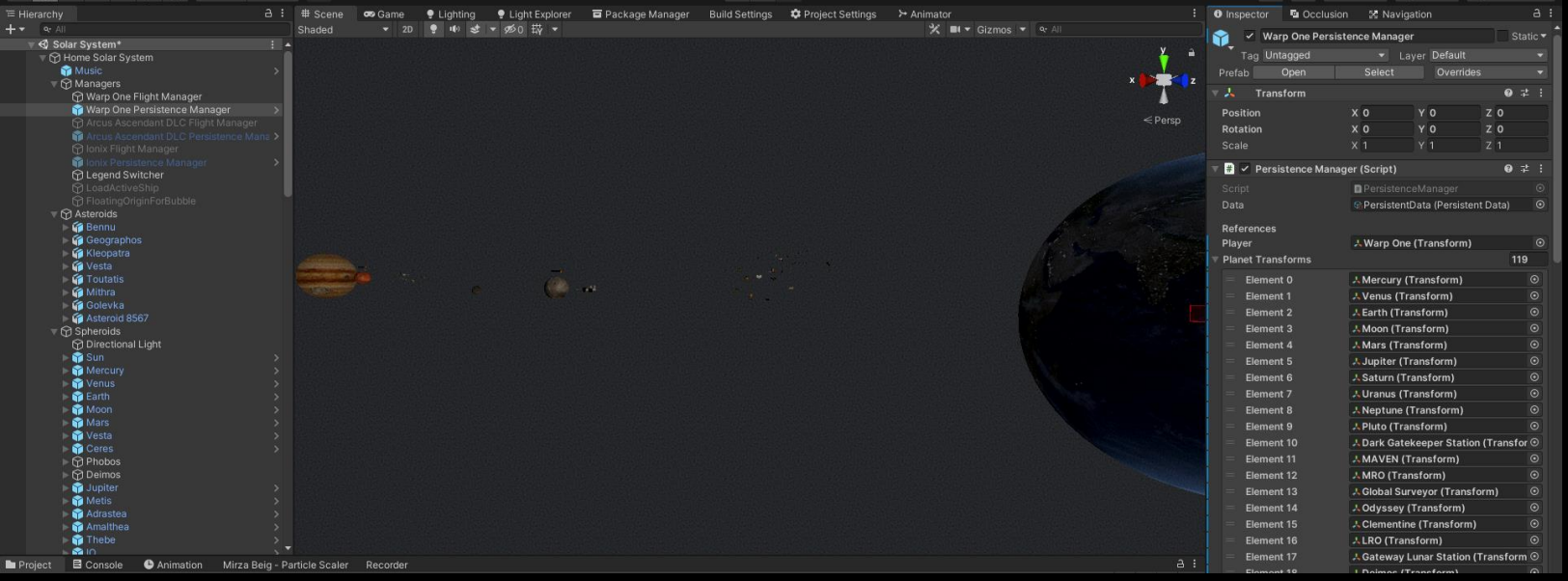
- Initial Mean Anomaly Determines where along the orbital path the object begins at simulation start. This ensures planets, moons, and satellites begin at their correct ephemeris-derived positions.

4. OrbitalTelemetryText.cs

A UI-driven telemetry system that converts astronomical distances into readable values using a defined AU-per-Unity ratio. It ensures that:

- orbital distances
- periapsis/apoapsis
- velocity readouts are displayed with scientific precision while still matching the scaled simulation space.

The Decimal Precision field controls the formatting of all telemetry outputs.

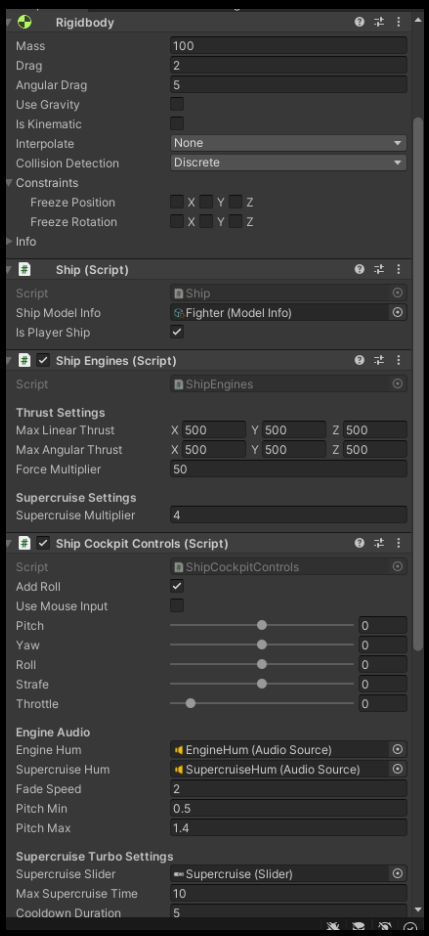


*Proprietary Persistence Managers to remember last orbiting position across sessions and load those positions for continuance across sessions.

II

Proprietary Superstring Engines

SWn (Superstring Warp Notations):



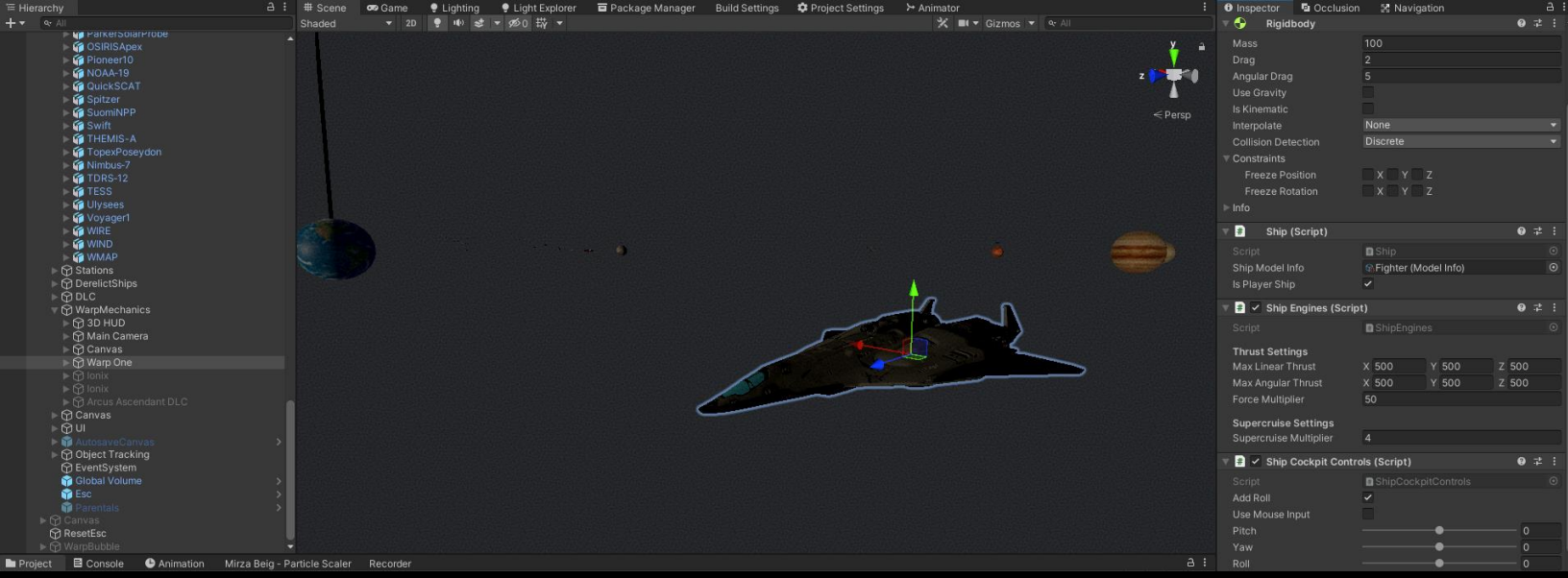
Rigid body-driven scripts that compute velocity by the application of thrust.

Basic x,y,z exposure of Linear and Angular thrust through 3D space.

Super cruise (Superstring Burns) which apply multipliers to the exponentiated readouts in the HUD.

Ship Cockpit Controls that allow the fine tunning of ship navigational feel based on Engine and proprietary and persisted Ship.cs configurations.

Other synchronous scripts that include, but are not limited to: Flight Manager per ship, with each ship sustaining an escalating configuration.

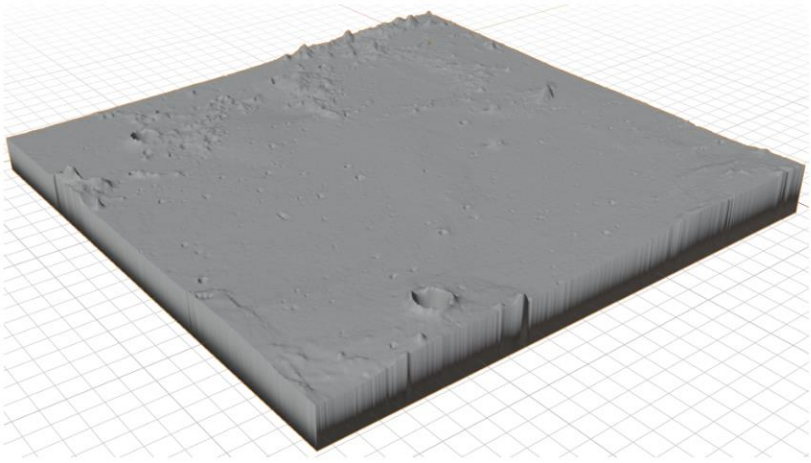

Hierarchy
Scene
Game
Lighting
Light Explorer
Package Manager
Build Settings
Project Settings
Animator
Inspector
Occlusion
Navigation
Shaded
2D
Gizmos
Persp
ParkerSolarProbe
OSIRISApex
Pioneer10
NOAA-19
QuickSCAT
Spitzer
SuomiNPP
Swift
THEMIS-A
TopexPoseydon
Nimbus-7
TDRS-12
TESS
Ulysees
Voyager1
WIRE
WIND
WMAP
Stations
DerelictShips
DLC
WarpMechanics
3D HUD
Main Camera
Canvas
Warp One
Ionix
Ionix
Arcus Ascendant DLC
Canvas
UI
AutosaveCanvas
Object Tracking
EventSystem
Global Volume
Esc
Parentals
Canvas
ResetEsc
WarpBubble
Rigidbody
Mass 100
Drag 2
Angular Drag 5
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Freeze Position X Y Z
Freeze Rotation X Y Z
Info
Ship (Script)
Script Ship
Ship Model Info Fighter (Model Info)
Is Player Ship
Ship Engines (Script)
Script ShipEngines
Thrust Settings
Max Linear Thrust X 500 Y 500 Z 500
Max Angular Thrust X 500 Y 500 Z 500
Force Multiplier 50
Supercruise Settings
Supercruise Multiplier 4
Ship Cockpit Controls (Script)
Script ShipCockpitControls
Add Roll
Use Mouse Input
Pitch 0
Yaw 0
Roll 0
Project
Console
Animation
Mirza Beig - Particle Scaler
Recorder

NASA STL CLEANUP & UNIFICATION USING BLENDER

Example Process:

1. Download of STLs from the NASA public repository e.g. the Apollo 12 Landing Site Topology

Apollo 12 – Landing Site

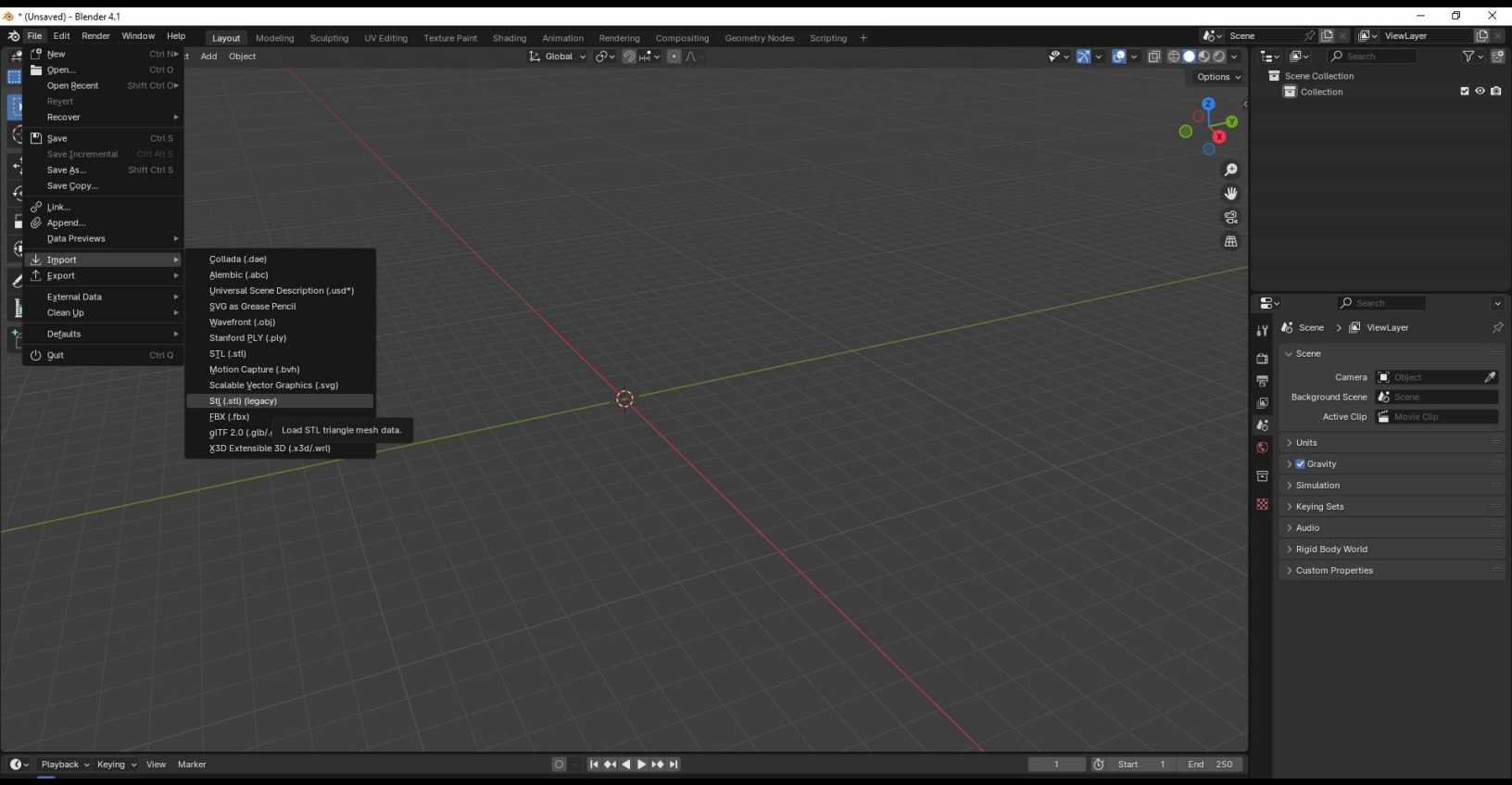


Ocean of Storms: Height map of the area around the Apollo 12 landing site (60x60 km). Exaggerated 50 times in the Z axis for better visualization.

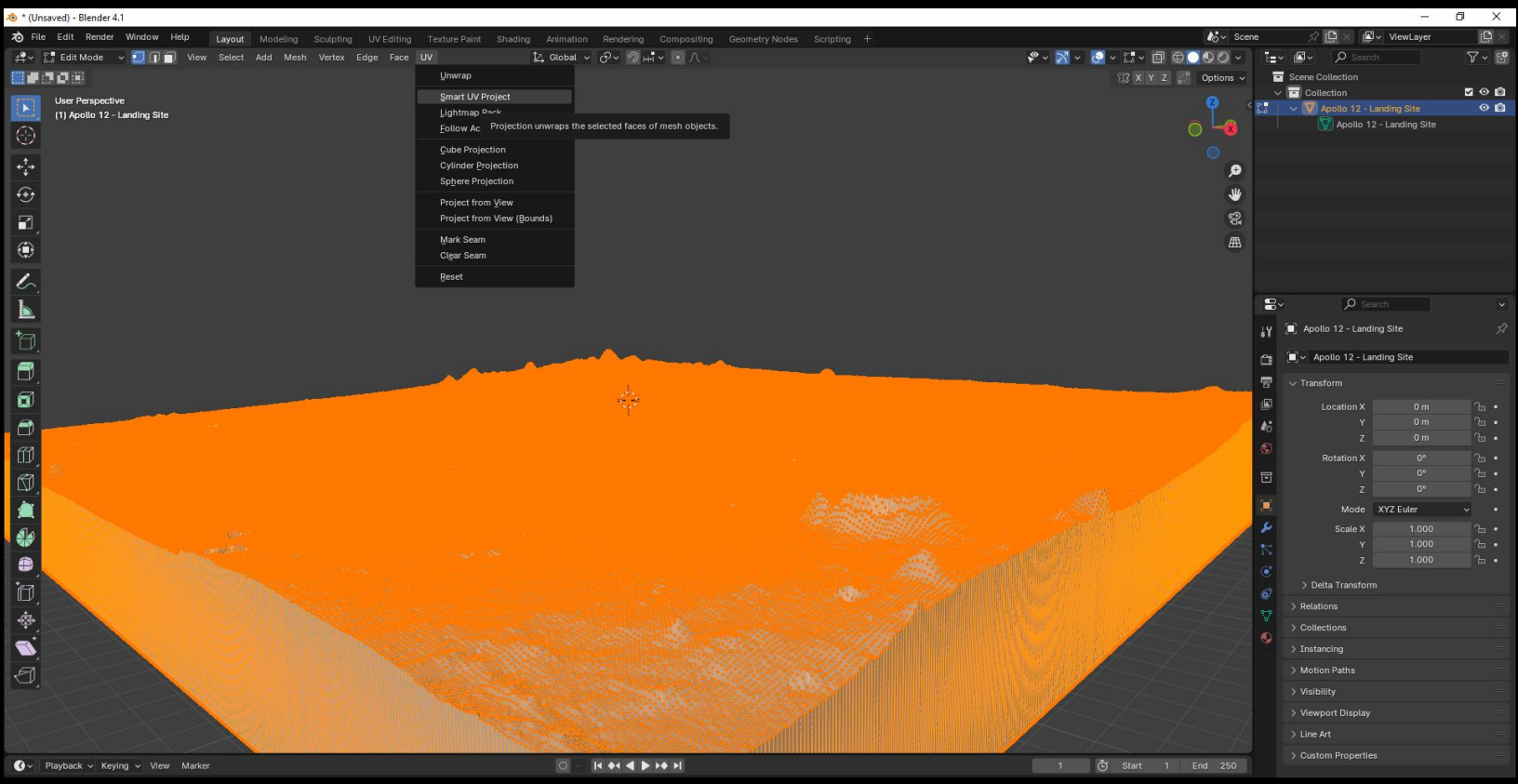
Read more about the mission

Apollo 12 - Landing Site.stl	Apr 21, 2025	(25.19 MB)	

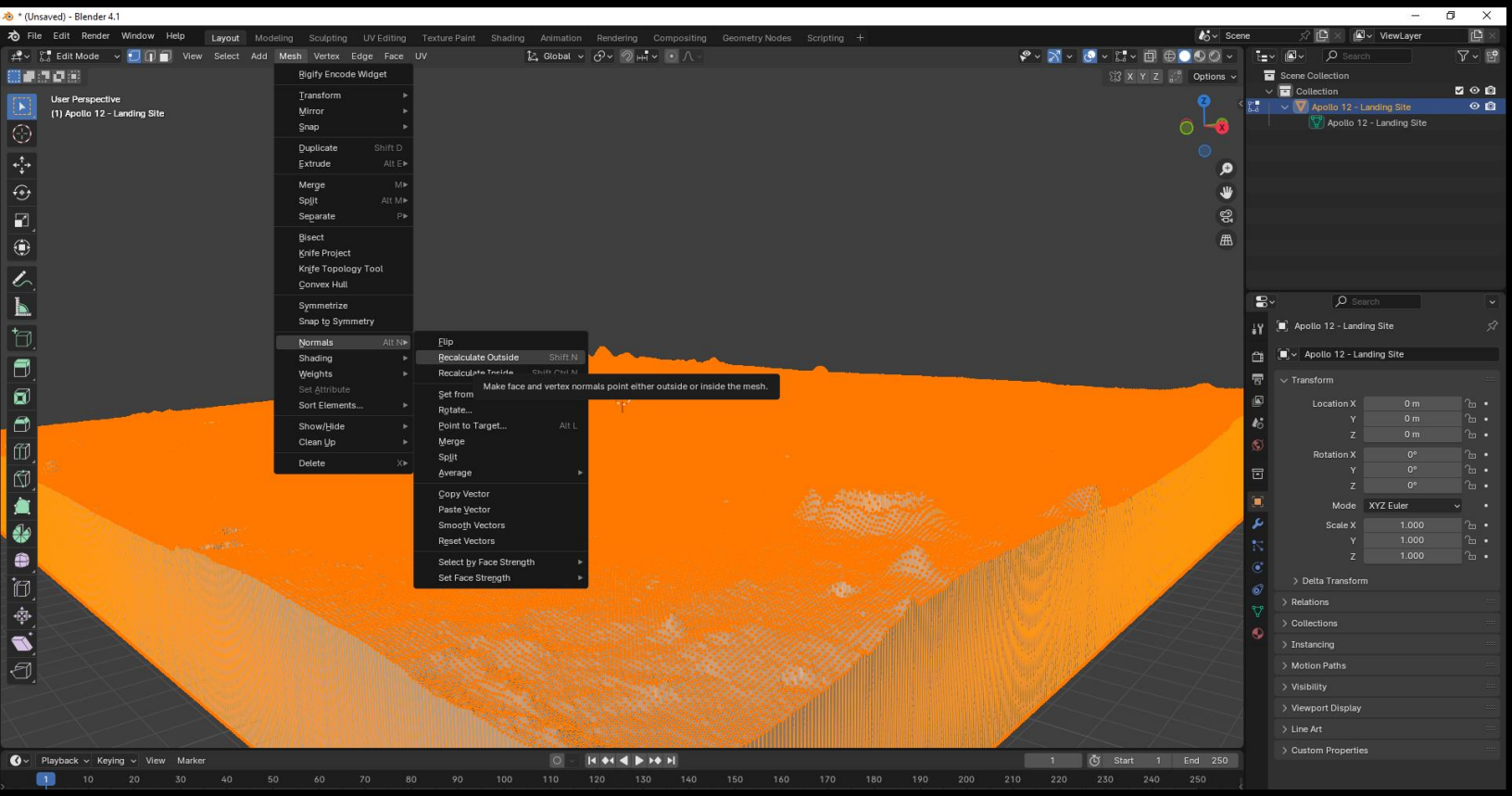
2. Importing of each individual STL topology into Blender



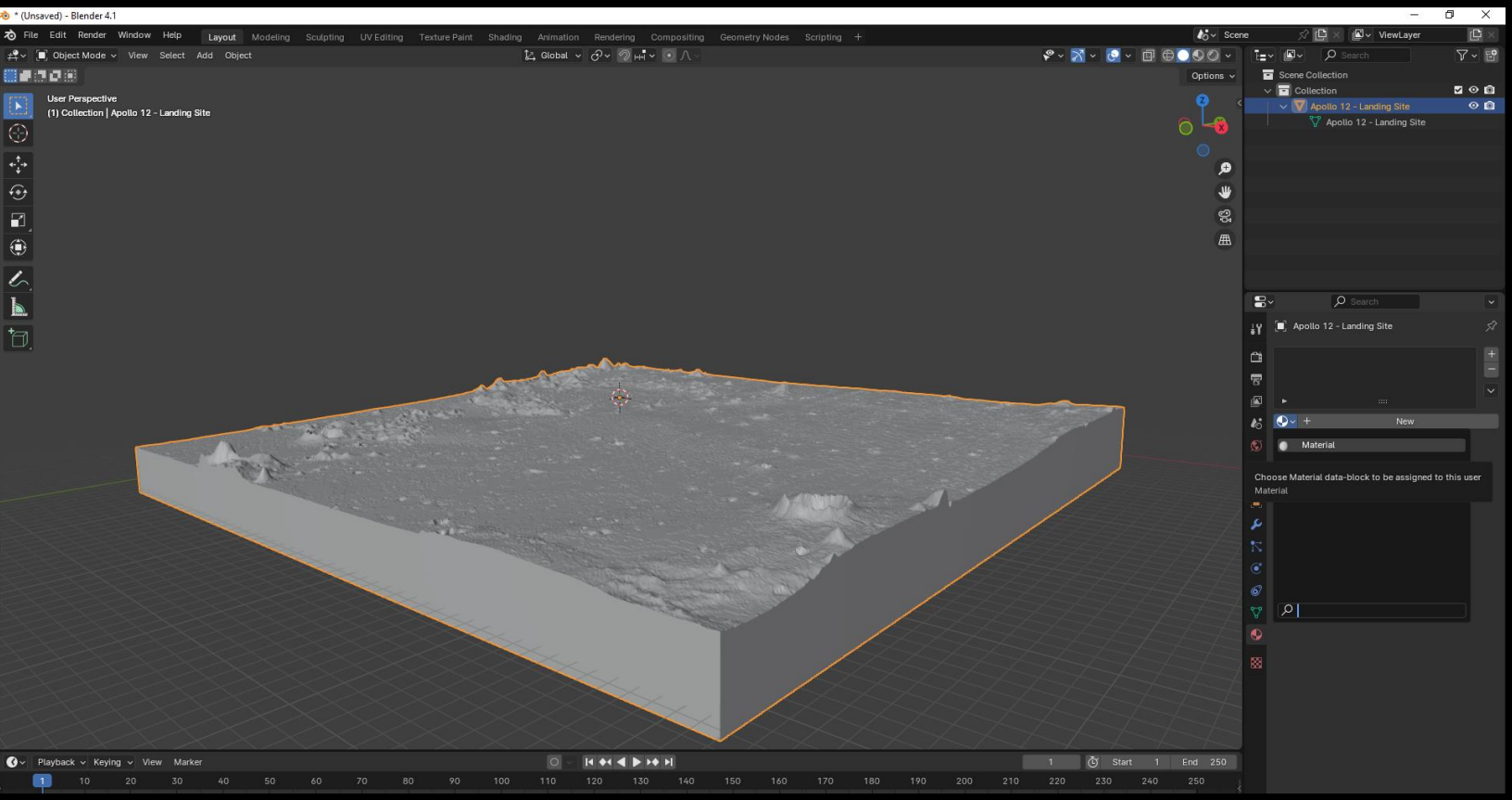
3. UV Cleanup of Topology



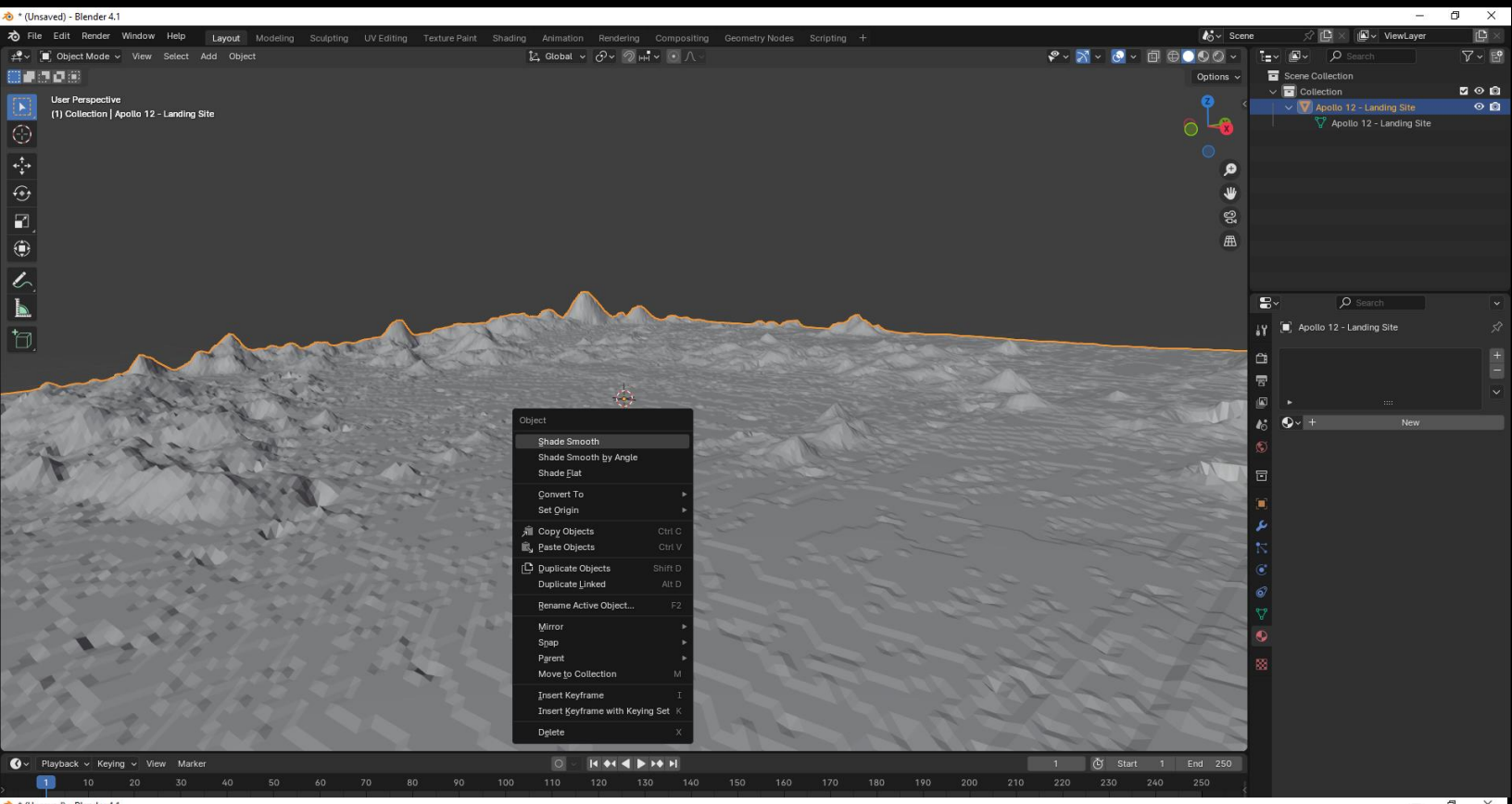
4. Normal recalculation on the exterior

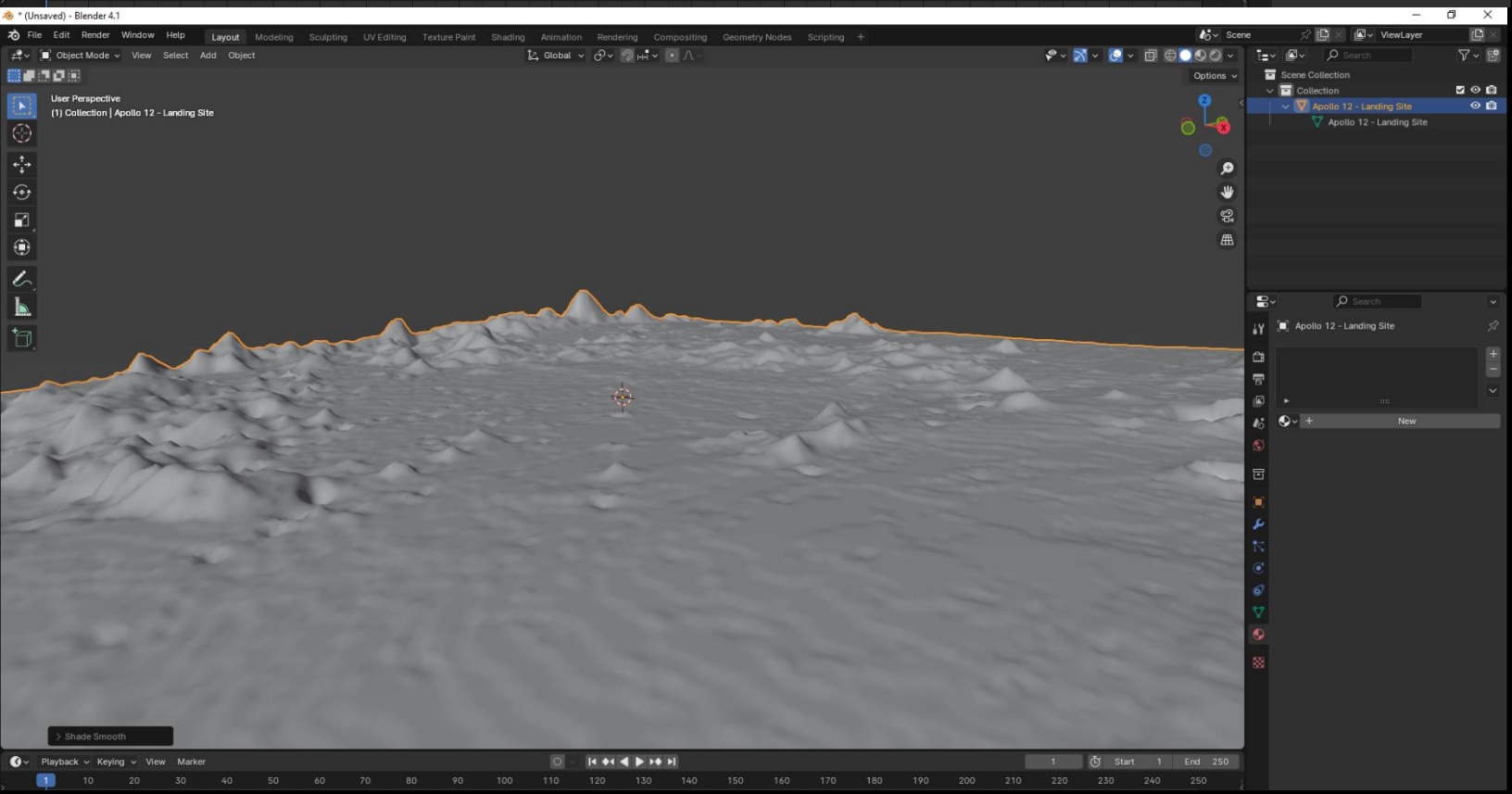


5. Material Assignment

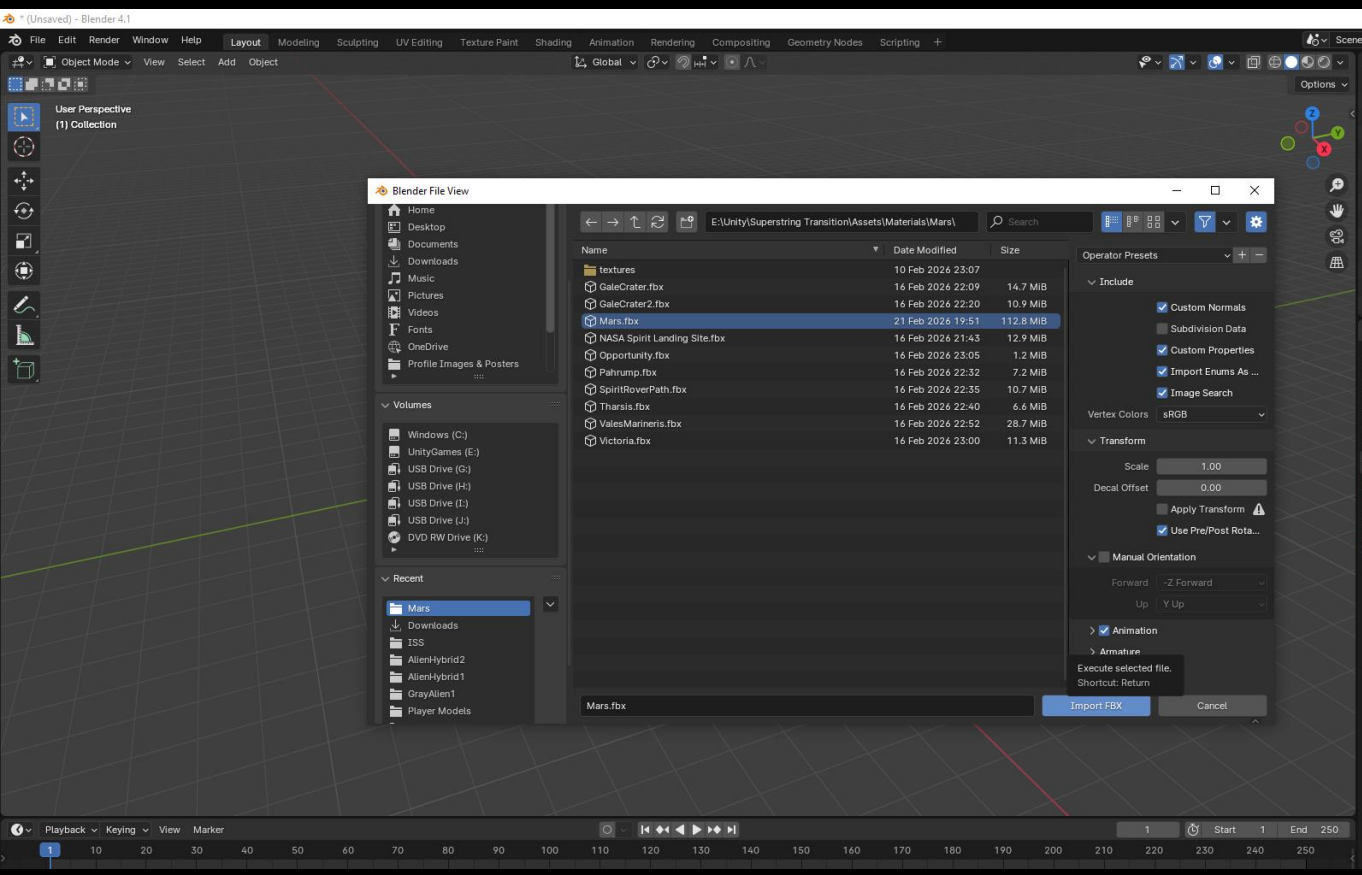


6. The smoothing out of polygons for High Definition Material rendering

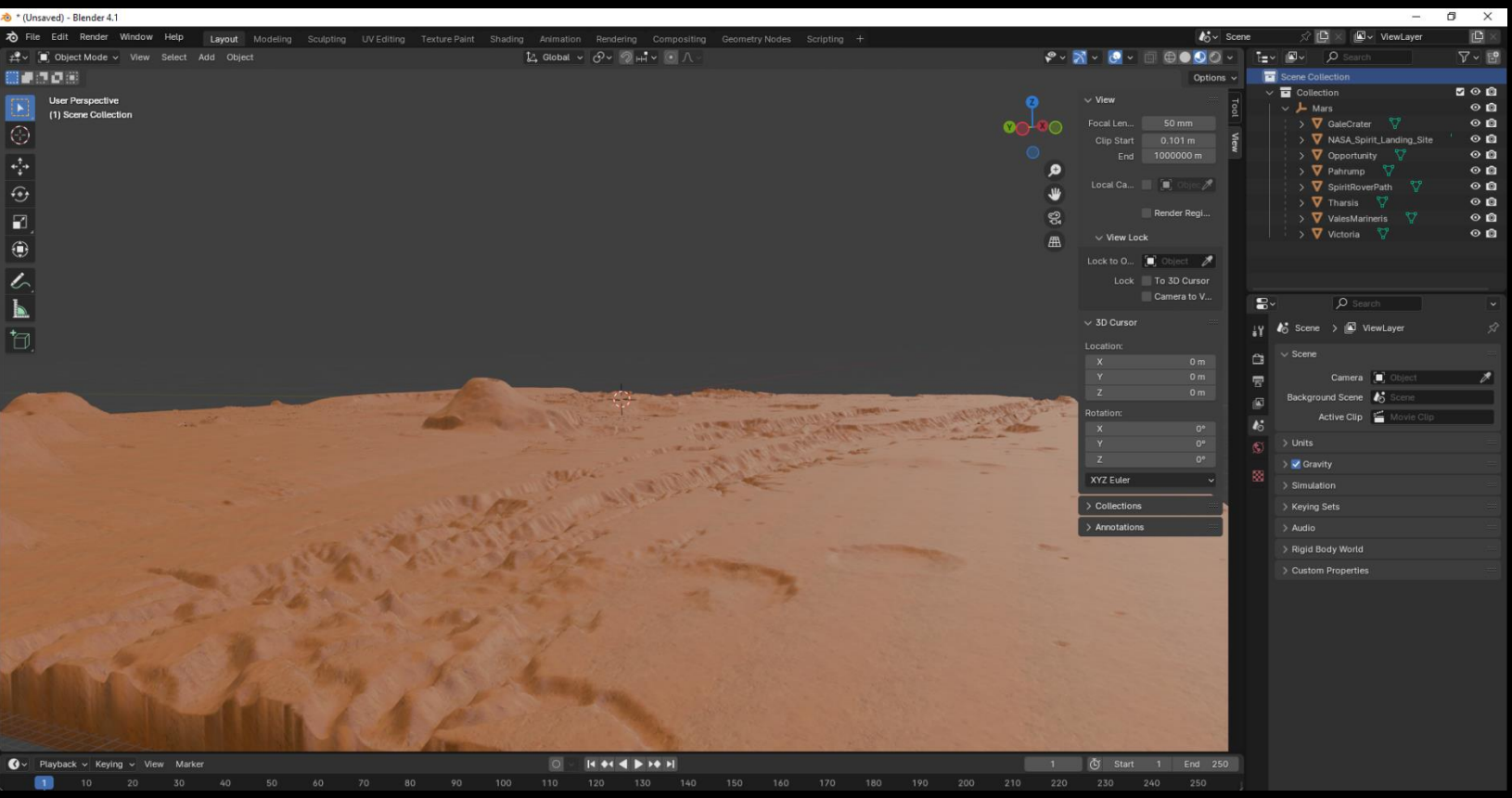




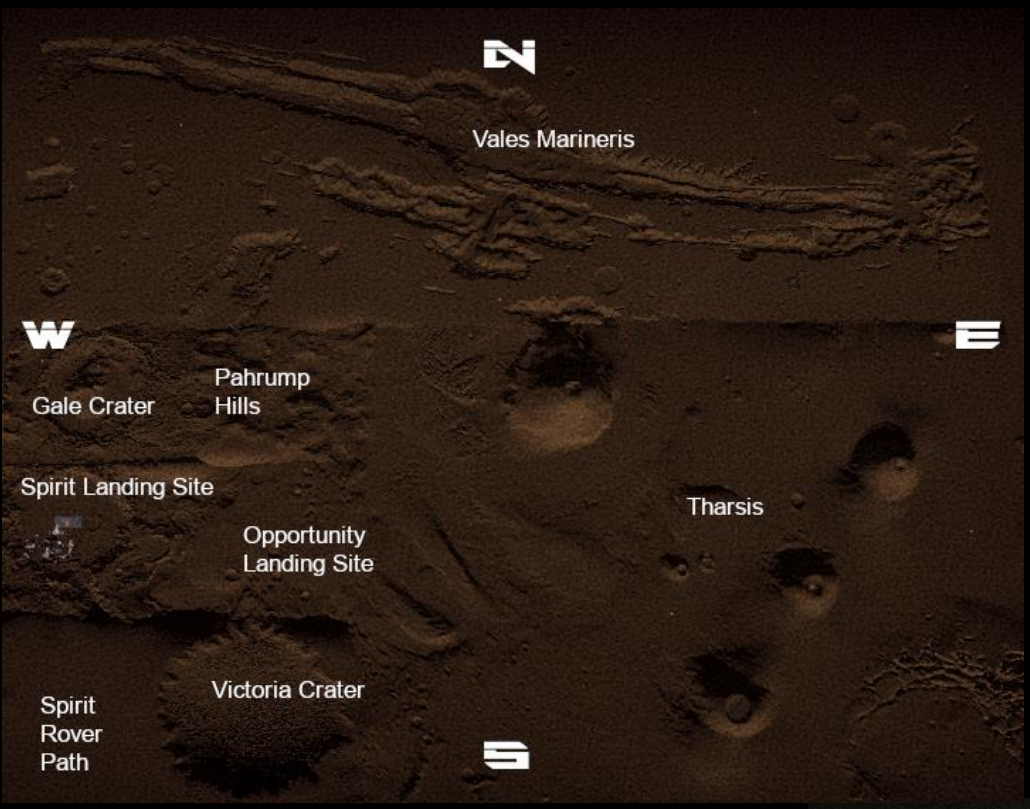
7. The repetition of this process for each STL inside Blender in preparation for unification as FBX (Unity's native 3D model types).

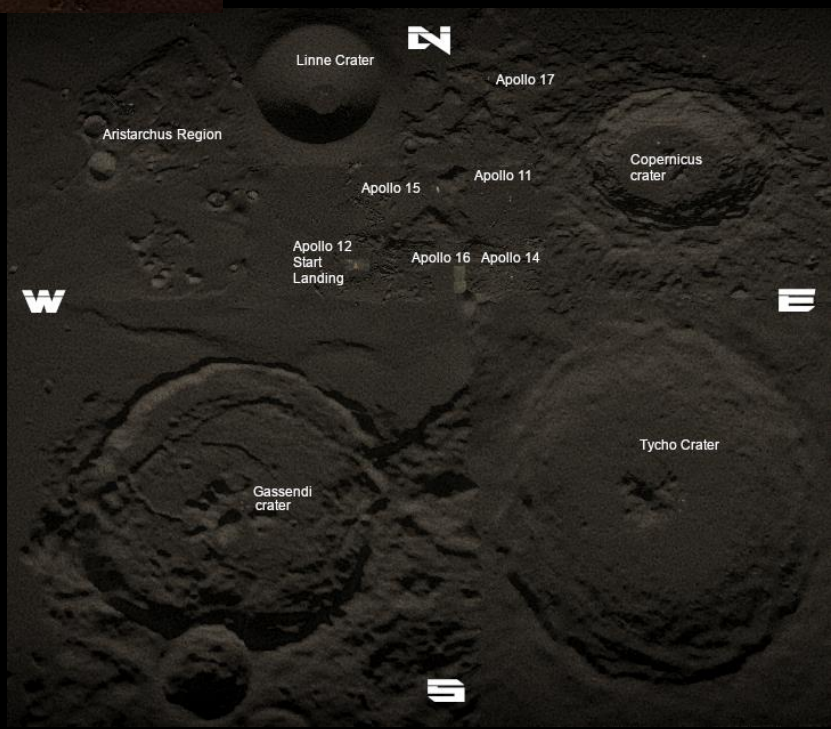


8. Example of Unified and sculpted at the seams Terrain for the Mars 8000+ km expanse.



9. The unification of the imports after cleanup to match cardinal points that are mapped for navigational convenience on maps in-game, oriented to each STLs relative position to each other on the actual body. The regions are unified and scaled reasonably based on actual size in relevance to each other.





10. Similar processes involving dozens of station, lander, satellite and probe .glbs.