DISCLAIMER

Following a brief Miscommunication among our Team Members ,
our project scope has undergone a subtle shift.
What initially began as "Building a Space Biology Knowledge Engine"
has now evolved into "Create Your Own Challenge."

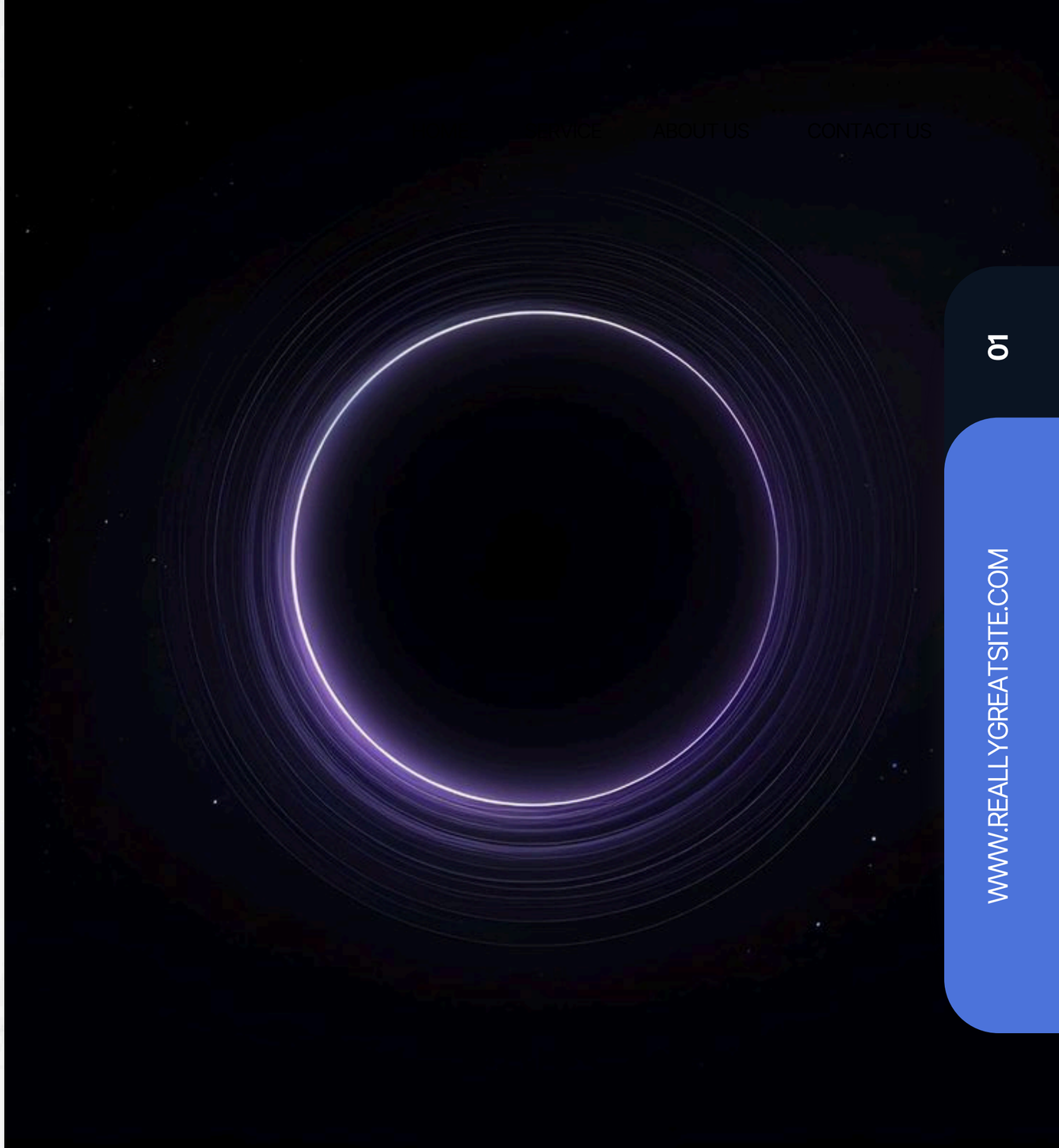

Borcelle Travel
HOME
SERVICE
ABOUT US
CONTACT US
01
ORIGIN
SPHERE

Borcelle Travel

INTRODUCTION

To create a scientifically-inspired procedural system that simulates how planets form from primordial gas clouds — gradually transitioning through nebula → accretion → proto-planet → mature planet, all within a dynamic, algorithmic, shader-driven 3D environment.

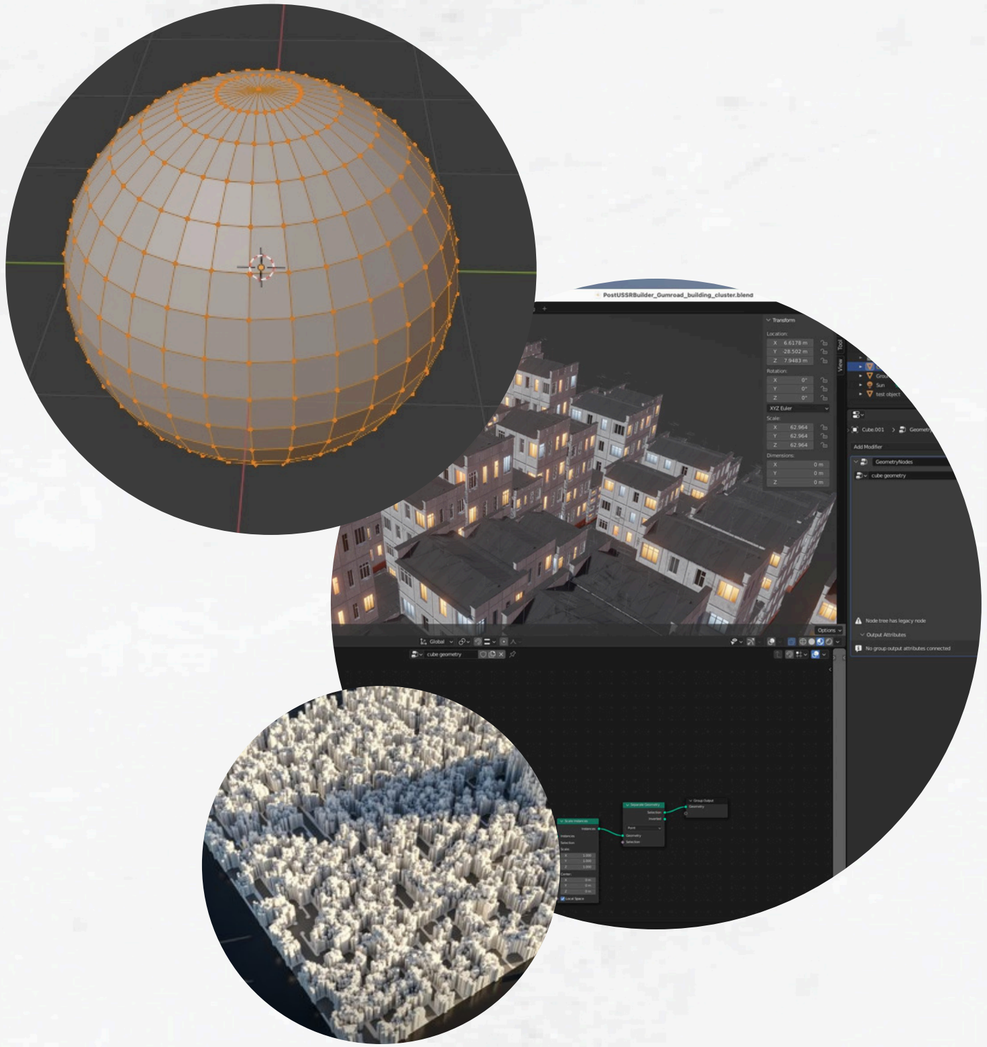
This includes both visual fidelity (particles, shaders, procedural geometry) and algorithmic formation logic (mass, density, collisions, layer formation, etc.).



CORE SYSTEMS **IMPLEMENTED-**

A. . PROCEDURAL SPHERE SYSTEM

B. PROCEDURAL MATERIAL LOGIC



Borcelle Travel

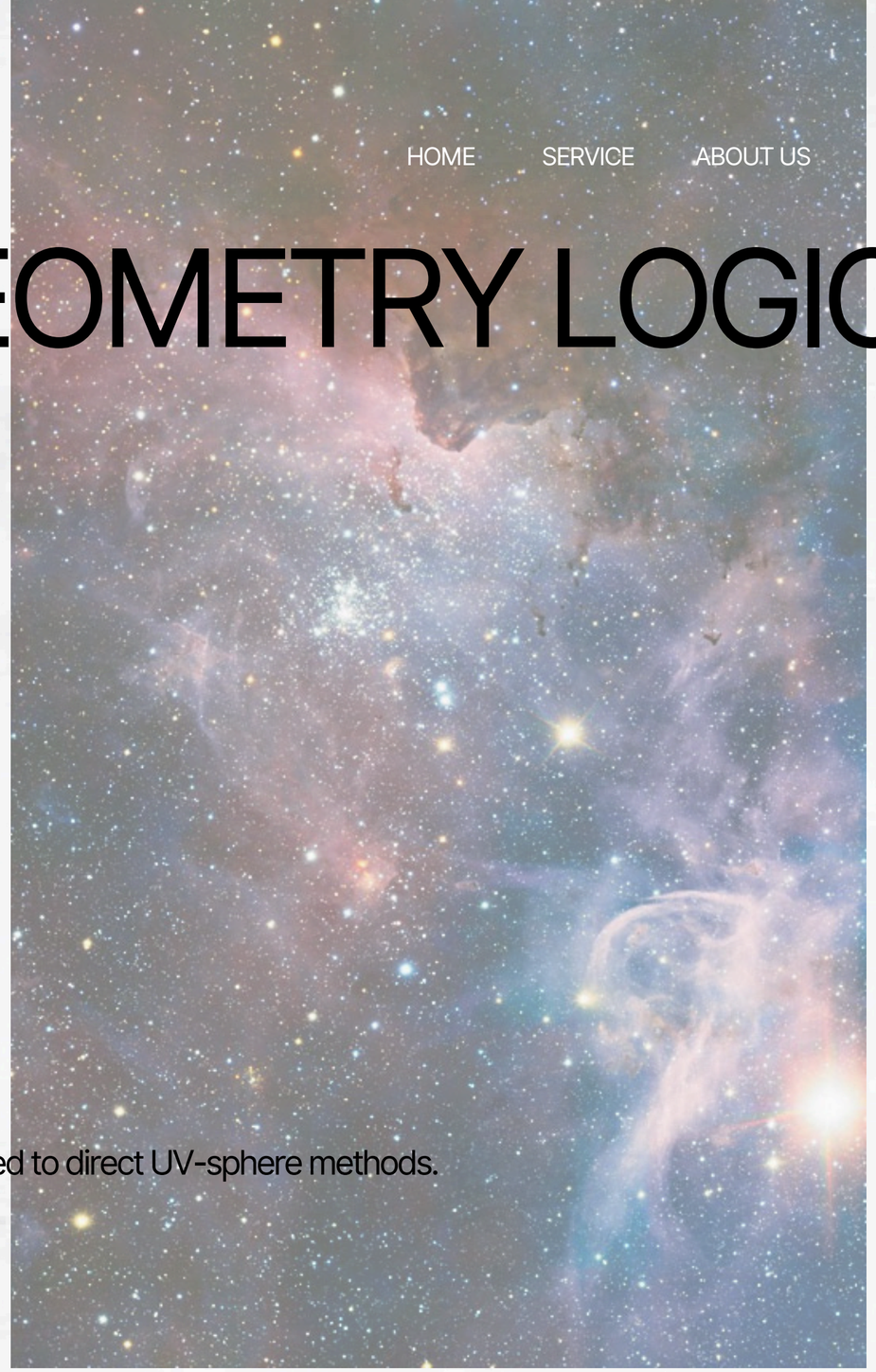
HOME SERVICE ABOUT US CONTACT US

A. PROCEDURAL SPHERE SYSTEM

FEATURES IMPLEMENTED

1. Fully Procedural Mesh Generation
2. Adaptive Per-Face Resolution
3. Global vs Adaptive Modes
4. Dynamic Auto-Update System
5. Smoothness Algorithm (Cube → Sphere Morph)
6. Mesh Quality Post-Processing

B. PROCEDURAL GEOMETRY LOGIC



The mesh is built algorithmically, not by any imported models.

Core Formulae:

Each vertex is defined as:

v = normalize(cubeDirection) * radius

Cube direction is computed as:

localUp + ((xPercent - 0.5) * 2 * axisA) + ((yPercent - 0.5) * 2 * axisB)

Where:

axisA, axisB are perpendicular to localUp

xPercent, yPercent ∈ [0,1] based on grid position

Each face = $(resolution+1)^2$ vertices and $resolution^2 * 2$ triangles

This gives uniform vertex distribution and near-perfect spherical topology compared to direct UV-sphere methods.

3. ALGORITHMIC PHILOSOPHY & DESIGN INTENT

The design is modular and data-driven:

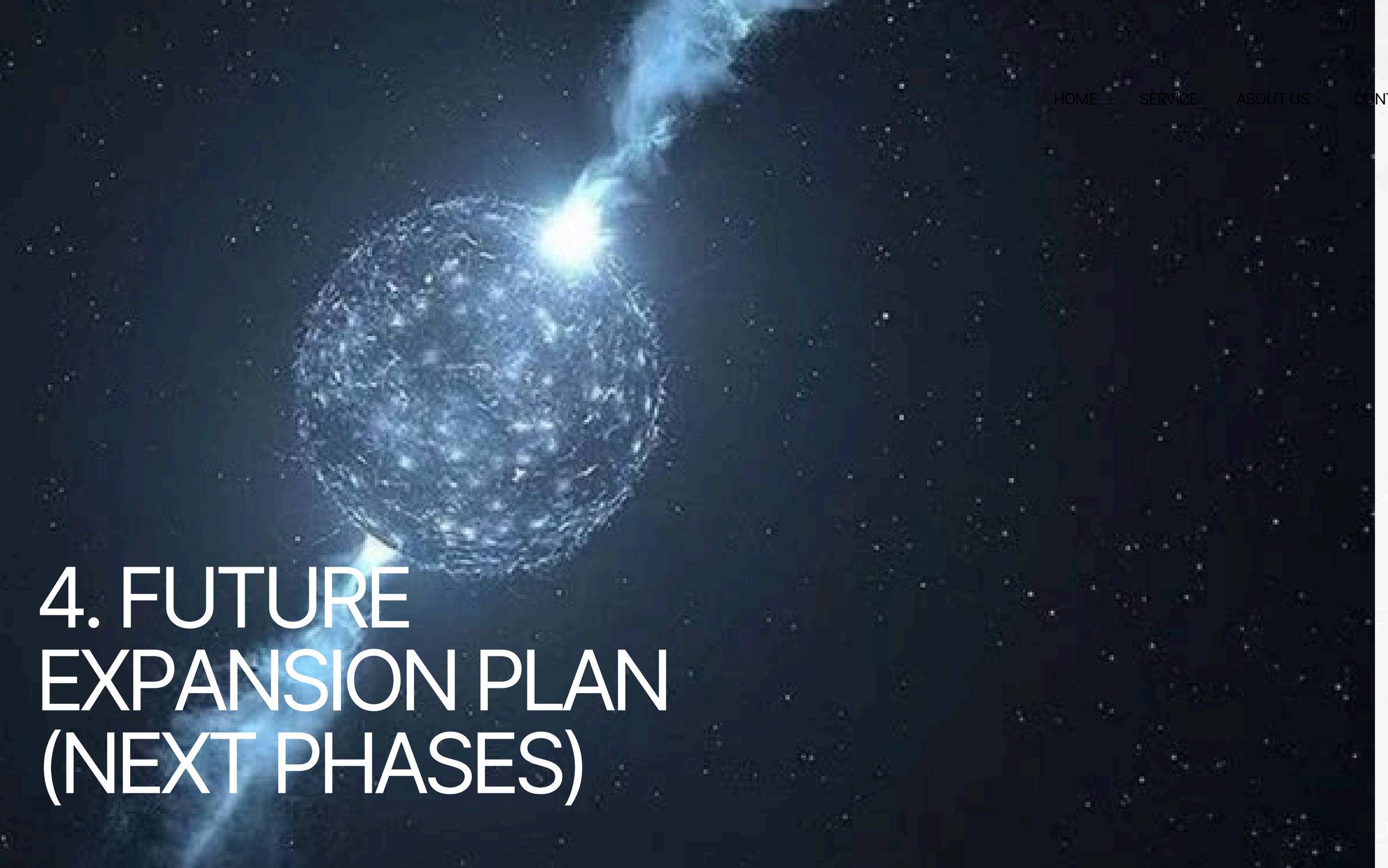
Every component (mesh, density, core, collision, atmosphere) will be algorithmically driven.

The planet isn't "modeled" — it's grown through physical simulation layers.

Core Conceptual Pipeline

Interstellar Medium (ISM)
↓
Rotating Nebula (Particle Swirl)
↓
Accretion Disk + Clumping
↓
Proto-Planet Mass Center
↓
Core Differentiation (Density / Heat / Composition)
↓
Crust Formation + Atmosphere + Magnetic Field

Each of these stages maps to one or more procedural systems (particle, shader, mesh deformation, etc.).



4. FUTURE EXPANSION PLAN (NEXT PHASES)

PHRASE 2

Dynamic Core & Mass Simulation

Simulate density layers (core, mantle, crust) inside the same procedural sphere.

Introduce mass distribution shader and core heat gradient.

Dynamically deform mesh based on gravitational pull + collision impact.

PHRASE 3

Collision & Accretion System

Implement particle collisions that:

Merge (mass + velocity conservation)

Generate craters or terrain bumps on impact.

Possibly use simplified SPH (Smoothed Particle Hydrodynamics) logic for impact visuals.

PHRASE 4

Atmospheric Shell & Material System

Procedural atmosphere sphere using GPU instancing or shell layers.

Shader-controlled density falloff, scattering, and color blending.

Adaptive transparency for gases or dust clouds.

PHRASE 5

Procedural Terrain Generation

Heightmap or noise-based surface variation.

Controlled via Perlin / Simplex / Worley noise.

Can reflect geological processes: lava cooling, erosion, or tectonic shaping.

PHRASE 6

Particle Cloud / Nebula Simulation

Before planet formation, simulate rotating dust/gas nebula with procedural motion.

Implement angular momentum-based swirl behavior.

Core formation from gradual condensation of center particles.

PHRASE 7

Interactive Parameter Framework

Real-time adjustable sliders for:

Radius, density, rotation speed, orbital tilt, etc.

Enable visual “planet formation timeline” playback.

Possibly connect with an external JSON or ScriptableObject data model for reproducible setups.

PHRASE 8

Rendering & Visualization Enhancements

GPU compute-based particle systems.

Shader Graph integration for heat maps (core temperature, density).

Custom lighting models (Lambertian + subsurface scattering for realism).

5. TECHNICAL HIGHLIGHTS

System Type Key Logic Current Status

Mesh Builder C# Procedural Cube → Sphere conversion with adaptive per-face ✅ Stable
Auto Update Unity Editor Safe delayed mesh rebuild ✅ Fixed
Adaptive Resolution Data-driven Face-wise grid control ✅ Stable
Smoothness Interpolation algorithm LerpUnclamped-based morph range ✅ Enhanced
Validation Safety patch Editor safe rebuild system ✅ Fixed
Expandability Modular Future collision, density, shader systems 🚀 In progress

6. ALGORITHMIC SUMMARY

Concept Mathematical Model Description

Cube to Sphere Projection v = normalize(cubeDir) * r

Ensures evenly spaced vertices on a spherical surface

Per-face Resolution Grid (x,y) → % of resolution Generates local quads per face

Triangle Indexing i, i+1, nextRow, nextRow+1 Defines mesh connectivity

Smoothness Interpolation LerpUnclamped(cube → normalized, s) Morph factor controlling curvature

Adaptive Sync Conditional resolution normalization Keeps consistent edge transitions

Mesh Recalculation Built-in Unity mesh recalculations Auto normal & tangent correction

7. PHILOSOPHICAL GOAL OF THE SYSTEM

This is not just a “planet generator.”
It’s a planet growth simulation framework — built from fundamental physical abstractions, not arbitrary art assets.

It’s capable of evolving from:

A dust cloud → forming solid core → gaining atmosphere → shaping surface terrain → hosting visual simulations of orbits and environment.

Essentially, a digital microcosm of cosmic creation.
This modular base allows both scientific visualization and game-like interaction.

8. LONG-TERM VISION

Once all modules connect, you'll have a complete planet formation lifecycle simulator:
Start with procedural nebula seeding
Observe mass aggregation
Watch planet birth and stabilization
Control parameters in real-time
Eventually expand into multi-planet system simulation
This will be a blend of science, mathematics, and procedural art, all evolving dynamically in a 3D environment.


THANK YOU