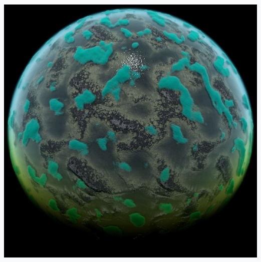
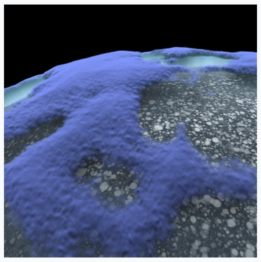
The ultimate planet rendering system.

This asset is designed to work in Unity 2021 LTS, Unity 2022 LTS, Unity 2023, and Unity 6, with the Built-In Render Pipeline, URP, and HDRP.

This asset includes a custom planet LOD system with fully configurable biomes, a volumetric atmosphere and cloud system, and an ocean system with underwater effects.

All example scenes are in the **"Plugins/CW/Planet Forge/Scenes"** folder, which show you what fully configured planets look like.


Seamlessly Fly From Space


Through The Atmosphere

Through The Ocean

And Go Underwater

Required Packages

Your project must have the **Burst** and **Mathematics** and **Collections** packages installed.

> Package Installation Guide

Making Your Own Planet

To make your own planet from scratch, you can either:

- Right click in the Hierarchy tab, and select the **"CW / Planet Forge / Planet (Radius = 500)"** option.
- Go to the menu bar, and select the **"GameObject / CW / Planet Forge / Planet (Radius = 500)"** option.

There are also options for 5,000 meter and 5,000,000 meter (similar to Earth) planets if you wish.

If you make a large planet, your main **Camera** component's **Clipping Planes / Far** setting must be large enough to render the whole planet.

If you make a large planet, it may spawn on top of the camera. If so, increase the planet's **Transform** component's **Position Z** value.

This will create a new **GameObject** called **"Planet"** with 4 child GameObjects:

1. **"Landscape"** - This handles rendering of the planet surface.
2. **"Sky"** - This handles rendering of the atmosphere.
3. **"Cloud"** - This handles rendering of the clouds.
4. **"Ocean"** - This handles rendering of the ocean, and underwater effects.

If your planet doesn't need an ocean/clouds/etc, then you can delete it. However, cloud rendering requires the sky (just like in real life).

You can hover the mouse over any inspector setting, and it will tell you what it does.

Your main scene light must have the **SgtLight** component to calculate lighting on the atmosphere and clouds. When you create a new gas giant, this will automatically be added.

Your scene must have the **SgtVolumeManager** component to render the gas giants. When you create a new gas giant, this will automatically be added.

Your main camera must have the **SgtVolumeCamera** component to render the gas giants. When you create a new gas giant, this will automatically be added.

Landscape

The **"Landscape"** GameObject has the SgtSphereLandscape component, which handles LOD mesh generation for your planet.

The LOD is driven by the **Detail** setting relative to the **Main Camera**'s position. However, if you want to override this, or have multiple cameras altering the LOD, then you can drag and drop them into the **Observers** list.

The landscape is generated using height and color textures, which are defined in the **Bundle** setting. If you want to create your own bundle, then you can add the **SgtLandscapeBundle** component to this GameObject (or in a prefab), and drag and drop it into the **Bundle** setting. By default, the **“Example Bundle”** is used, which contains a few example textures.

The **“Add Collider”** button will add the SgtLandscapeCollider component. This will generate colliders for the whole planet down to the specified MinimumTriangleSize in this component.

ⓘ If you have a large planet, then Unity may output warnings that there are large colliders in your scene, but there doesn’t seem to be a way to disable this...

The **“Add Detail”** button will add a child GameObject with the SgtLandscapeDetail component. This can apply a layer of detail around the whole planet, or to a specific region.

The **“Add Flatten”** button will add a child GameObject with the SgtLandscapeFlatten component. This can flatten the landscape in specific regions.

The **“Add Color”** button will add a child GameObject with the SgtLandscapeColor component. This can color the whole planet based on height and slope data, or to a specific region.

The **“Add Biome”** button will add a child GameObject with the SgtLandscapeBiome component. This combines the features of SgtLandscapeDetail and SgtLandscapeColor into one component, simplifying configuration.

The **“Add Prefab Spawner”** button will add a child GameObject with the SgtLandscapePrefabSpawner component. This will spawn prefabs on the surface of your planet as you approach.

The **“Add Static Spawner”** button will add a child GameObject with the SgtLandscapeStaticSpawner component. This will spawn static meshes on the surface of your planet as you approach.

> ⓘ By default, your planet is given one biome, which is in the **“SgtLandscapeBiome”** child GameObject.

If you want the planet to be based on a pre-generated albedo or height texture, then you can set it in the **AlbedoTex** or **HeightTex** setting.

> ⓘ These must use cylindrical (equirectangular) projection, use the **Single Channel Red** format, and have **Read/Write** enabled.

Sky

The **"Sky"** GameObject has the SgtSky component, which handles rendering of the sky and (optional) clouds.

You can adjust the **UpperColor** and **LowerColor** settings to change its look.

The **InnerMeshRadius** setting should match the radius of your ocean, or your landscape radius if your planet doesn't have an ocean.

If you enable the **Lighting** setting, then the atmosphere can receive light from one **SgtLight** component.

> ⓘ When you create a planet, the **SgtLight** component will automatically be added to the Sun or brightest light in your scene. Otherwise you must manually add this component.

Cloud

The **"Cloud"** GameObject has the SgtCloud component, which handles LOD mesh generation for your planet.

If you want the overall cloud coverage to have a specific shape, you can set it in the CoverageTex setting.

> This texture must use cylindrical (equirectangular) projection. The R/G/B/A channels can all be used with the CloudLayers setting.

You can add up to 4 cloud layers, and control settings like the Height, Thickness and Density of each layer.

You can also erode each layer with a detail texture using the **SgtCloudDetail** component.

Ocean

The **"Ocean"** GameObject has the **SgtOcean**, **SgtOceanRays**, and **SgtOceanDebris** components, which handle rendering of the ocean surface and underwater effects.

This component has the Radius setting, which should be greater than your landscape's radius.

Last updated 5 minutes ago