Simple Planet Generator

Thank you so much for purchasing the Simple Planet Generator, which allows you to create procedural planets within Unity.

Firstly, it is important to note that this system does not currently have a draw tool, and purely makes use of algorithms which will generate planets within Unity. We do however feature a large amount of options which allow you to customize the planets to your preference.

We highly recommend that new users make use of the Prefabs included with the system. However all scripts will run independently and you are more than welcome to simply drag these scripts onto your own meshes. (We do include a default mesh for use in your projects)

The following Prefabs are included in the system:

- Planet Simple (Planet and Water Sphere only)
- Planet Atmosphere (Planet, Water, And Atmosphere)
- Planet Clouds (Planet, Water, Clouds Only)
- Planet Clouds Atmosphere (Planet, Water, Atmosphere, And Clouds)

All of these prefabs include the gravity script which will attract rigidbodies to the surface of your planet.

We also include a very simple cloud prefab which makes use of a custom texture and the Unity Particle System

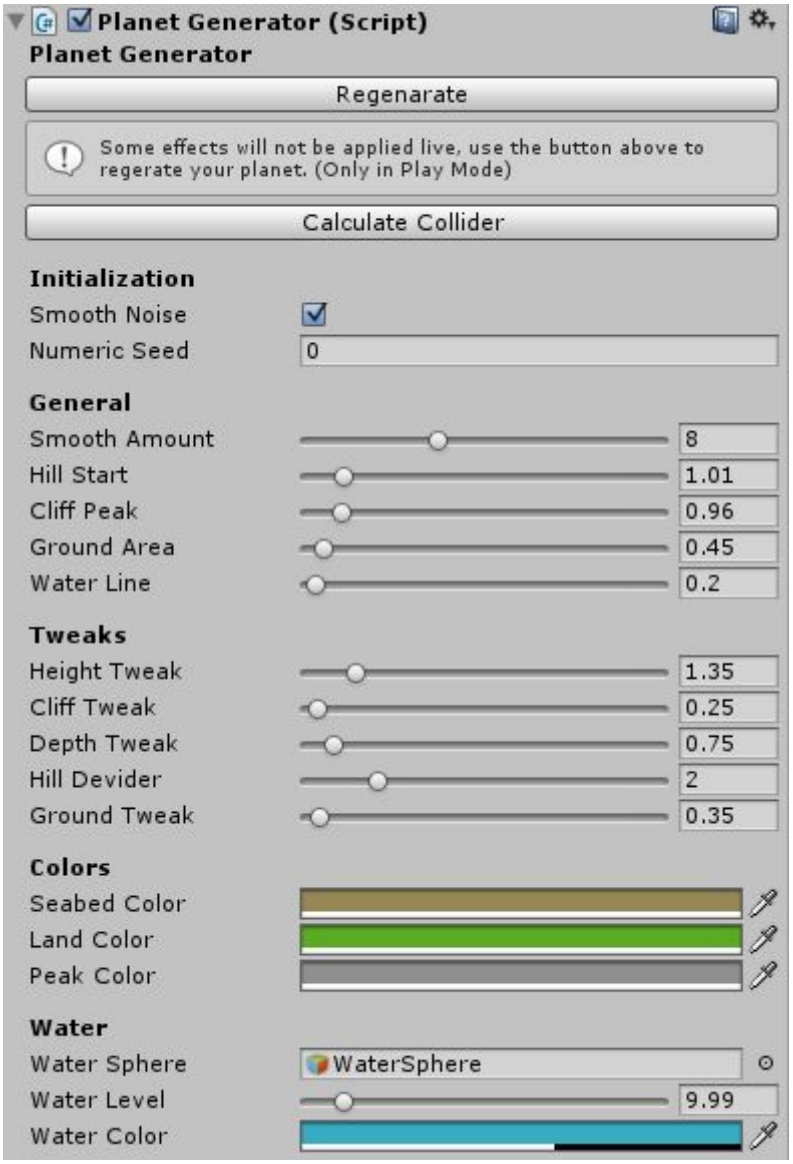
The rest of this overview, will assist you with getting familiar with the core inspectors, scripts, and showcase some public methods which could be used in game.

Planet Generator

The Planet Generator script controls the primary planet generation, by deforming the base mesh using one of two noise options, more on this later.

It allows you to setup heights, surface tweaks, colors, and control your water sphere (optional)

The custom inspector was developed to streamline this process and make it easy to use regardless of your development experience.



As you can see, there are quite a few options here, which we hope will allow you to achieve your goals with procedural planet generation.

It is important to note, that although most of the sliders will affect the planets live in play mode, some will require that you recalculate the mesh using the simple buttons in the top of the inspector. (We also supply a public method for this called ‘GenerateHeights’)

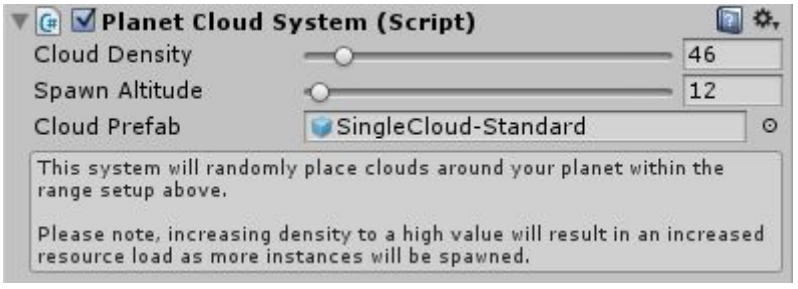
In addition to this, you will need to recalculate the mesh's collider once changes have been made. This is a required step if you wish to allow rigibody objects to collide accurately with your planet surface.

Once again we have a button within the custom inspector which handles this, and you simply need to click this to recalculate the mesh collider. (We supply a public method for this called 'UpdateCollider')

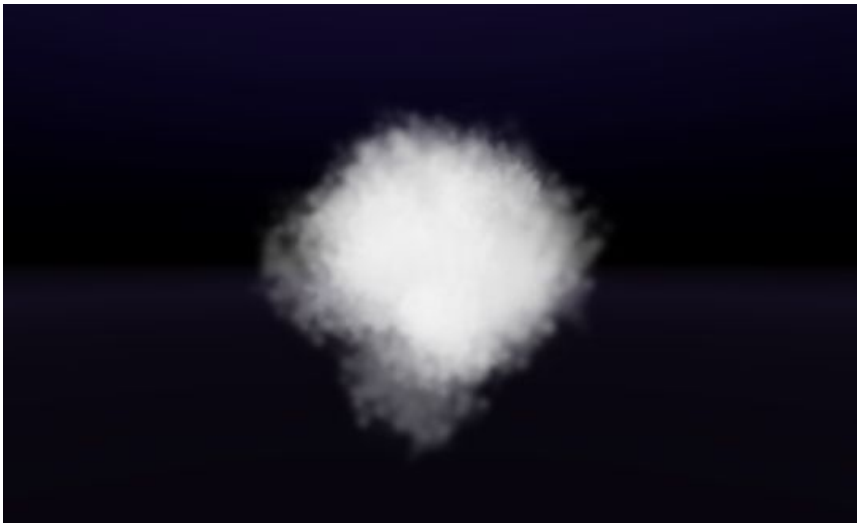
Planet Clouds

The planet clouds system handles generating clouds dynamically within the range of your planet. This is done by using a defined spawn altitude and spawning clouds (particle system in examples) randomly within range of the planet.

The inspector allows for simple control, however we do advise that you keep an eye on resource load, as increasing cloud density may cause issues with optimization.



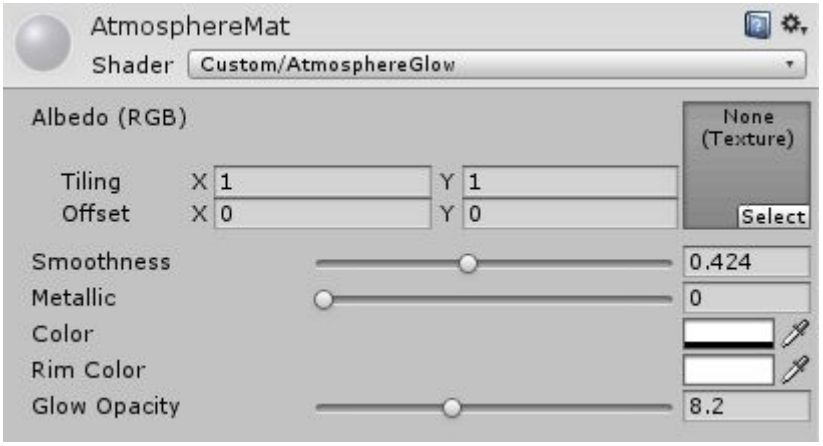
We include the following prefab in our demo for the cloud system which uses a custom texture and the Unity Particle System.



Planet Atmosphere

The system includes a custom shader which will add an atmospheric glow to the object which it is placed on.

We have included this in our demo system as a sphere with a material applied which uses this shader. You are welcome and encouraged to use this in your projects as well:

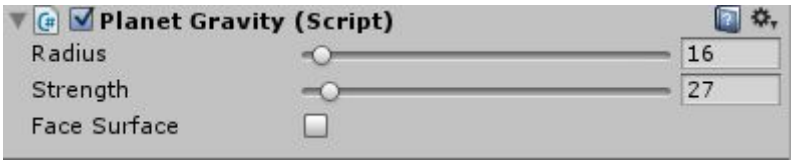


Planet Gravity

We also include a very simple planet gravity system which will find all colliders (with rigidbody attached) within range and apply force to them to attract them to your planet surface.

Force is applied dynamically based on distance, meaning objects will move faster the closer they get to your planet surface.

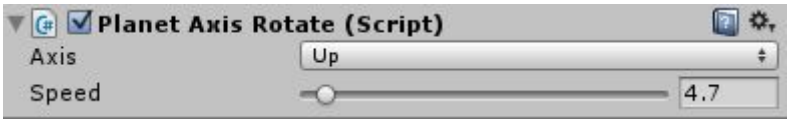
The inspector also allows you to force objects to face the planet as they approach:



Planet Axis Rotation

We noticed in our own projects that we sometimes wanted the planet to rotate on a specific axis, so that we could see all sides of the planet as we were making changes.

We therefore included a script which could also handle this for you:



Simply attach it to your planet, select an axis, and set your preferred speed.

For Developers:

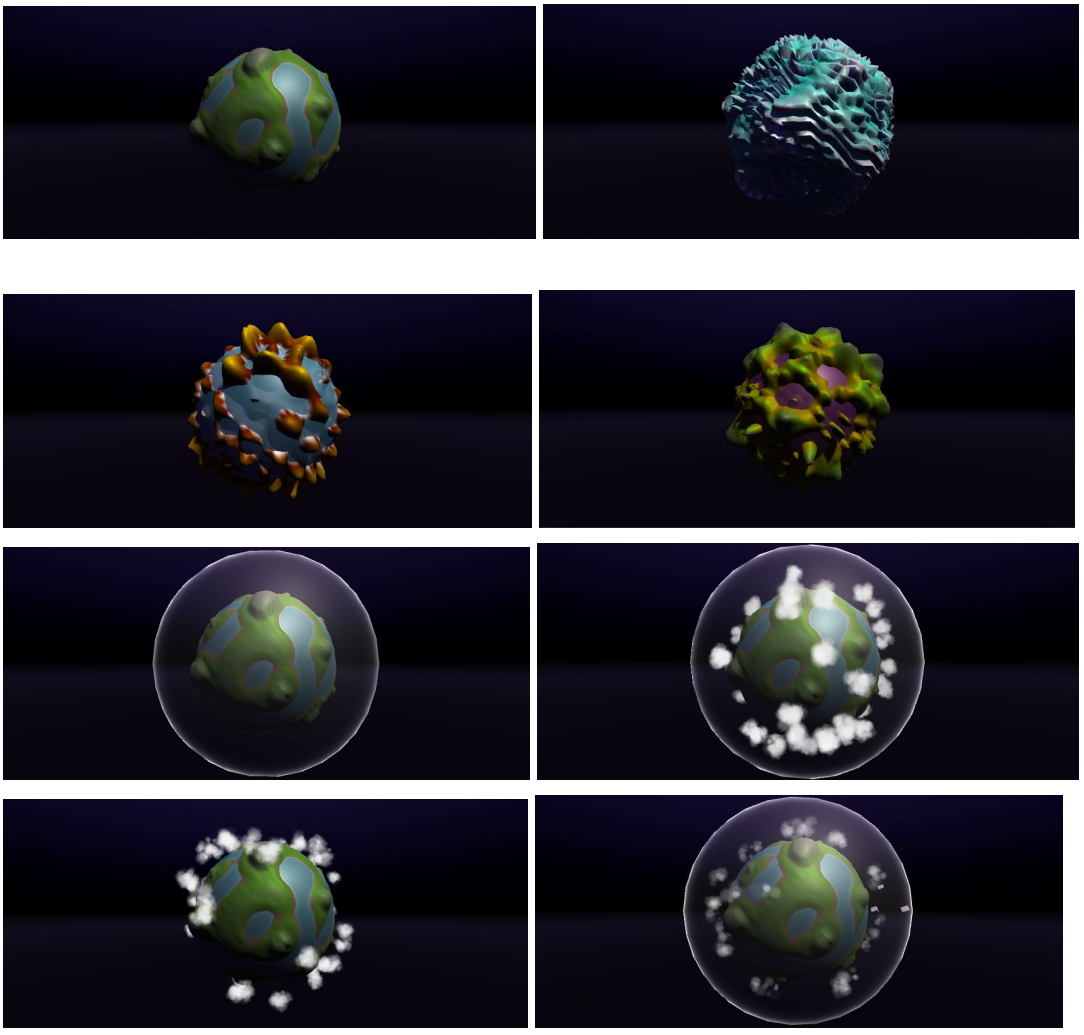
All of our scripts make use of public variables, which can adjusted by simply targeting the class and adjusting them directly.

We decided to leave this open ended and build custom inspector windows to limit user access to variables, as we thought it best to allow developers maximum control over the code.

If you have any questions please get in touch with us and we would be happy to assist.

Examples:

Let's have a look at some examples of what can be achieved.



A big thank you from our side once again for your purchase, we really hope you enjoy the system!