PlanetGen v1.0 Manual

What is PlanetGen?

PlanetGen is a toolset for making procedural planets.
The package contains a generator which create equirectangular projected heightmap texture based on random fractal noise.
The generated heightmap is used by a custom PG Planet shader. Colors, bumps, water specular mask calculated inside the shader. The heightmaps are temporary only, no texture asset need to be saved. Texture generation works both in editor and runtime. Currently a solid planet heightmap generator available in the package.

Quickstart

To make your own planet,

1. navigate to *Tools/Zololgo/PlanetGen* and select *Create New Solid Planet*. This will create you a Planet object in the Hierarchy with default properties.
2. Create a new material and select *Zololgo/PlanetGen | Planet/Standard Solid Planet* shader for it.
3. Assign the material to the *Planet Material* field in the new planet's *PG Solid Planet* component.
4. Play with the properties of *PG Solid Planet (dont forget to click the Generate button!)* and tweak the shader colors and levels to get the desired look of your planet!

Or you can start with the example scenes to find out the capabilities of PlanetGen.

Examples

The package contains several example scenes. You can use them to explore the features and possibilities of PlanetGen.

- **First Planet** example is just a simple scene with a single planet. The planet is the end result of the Quickstart steps.
- **Sample Planets** example is a small showcase of different planets. Planets in this scene also have halos.
- **Procedural planet** example shows how to make a planet programmatically. The scene itself doesn't have a planet object, it is created at runtime from script in the Planet Maker GameObject.
- **Planet Randomizer** is an example for runtime planet modification. You can randomize the the look of the planet by clicking the Randomize button runtime.
- **Custom texture in Planet Shader** example is the opposite of the previous example: instead a generated heightmap, a heightmap image from planet earth used in the planet shader.
- **Ramp Texture Planet Shader** example is a showcase that the generator can be paired with any shader, not with just the default one. In this example, a color ramp texture sampled for the albedo, based on the generated heightmap.

Create New Planets

In the *Tools/Zololgo/PlanetGen* submenu, you can find shortcuts for planet (and other PG object) generation.

- **Create New Solid Planet**
  Shortcut to create a basic Planet in the Hierarchy. It is a single GameObject with a *PG Solid Plane*t component, *MeshFilter*, *MeshRenderer* and *SphereCollider*.

- **Create New Solid Planet LOD Group**
  Shortcut to create a Planet LOD Group in the Hierarchy. This is a GameObject structure with different LOD levels as child Objects. *PG Solid Plane*t, *LODGroup* and *SphereCollider* component is assigned to the parent object. *MeshFilter* and *MeshRenderer* assigned to the child objects.
- **Create New Halo**
  This is an additional effect to the planets. It is basically a low poly geosphere with Planet Halo shader, resulting an inverted, faded sphere, which combined with a planet can achieve a nice and cheap glow-like effect.

Components and Shaders

PG Solid Planet component

- **Planet Material** - The used material by the Planet. The planet and Planet LOD groups automatically updates the material on the renderers with this Planet Material property.
- **Surface properties** - These properties controls the fractal noise. Results only updates if you click Generate, or the auto generation toggle is on.
    - *Noise scale* change the land size, low values creates bigger, fewer land forms, higher values results in more small islands.
    - *Noise persistence* controls the fuzziness of the noise. Low values produce smooth noise, higher values creates a more eroded look.
    - *Seed* value initialize the random generator. Noise results are relies on this value.

- **Heightmap texture properties** - These settings are identical to the texture importer settings in unity. PlanetGen heightmaps are procedurally created textures, but you can have control over the texture properties. For more information, please read about texture importer properties in Unity manual.
- **Auto Generate and Generate** - If you change the surface properties, you have to regenerate the heightmap to see the results. You can do this by hit the Generate button after a change, or set on the auto generation toggle. Please note, auto generation slows down the Planet inspector, especially if high heightmap texture resolution is used. A recommended workflow to set the resolution a low preset like 1024x512, and have auto ganarate toggle enabled during the tweaking the planet surface parameters. After the customizing is done, you can set a higher resolution for the texture.
- **Randomize** is works as generate button, with the addition of randomizing the seed value.

Solid Planet Standard shader

- **Atmosphere properties**
    - *Atmosphere Color* is an overlay color on the planet. The color's alpha channel controls the overlay opacity. If the alpha is 1, then only the pure color is visible. With low alpha values, you can use this property to unify the surface's different colors. Slightly higher values creates a foggy, atmospheric look. A bright, blueish Atmosphere Color with high alpha can create an icy, frozen look.
    - *Rim Color* controls the rim light effect around the planet. Alpha channel controls the effect's opacity.

- **Surface parameters**
  - *Heightmap* is the most important field in the shader. Luminance value of the Heightmap pixels controls the final color of the texture through Land and Water level settings, and also the source of the bump mapping. By default, the heightmap field is populated by the planet generator, but feel free to use your own textures too.
  - *Bump Strength* controls the effect's strength of the bump mapping. It adds great details to the surface and a more believable look.
- **Water settings**
  - *Sea Level* controls the amount of water on the planet. 0 means no water, 1 means everything is underwater.
  - *Sea Color* is the color of the planet above the sea level.
  - *Shininess* Controls the light's reflection on the water surface.
- **Land settings**
  - *Color Blending* controls the transition between the land colors in different land levels. Low values means hard edges between the levels, higher values give a more realistic look.
  - *Land Base Color* is the default land color over the sea level.
  - *Hill Level* adds an additional layer to the surface colorization. This is a sea level relative value, so 0 not equals to the heigthmap pixel's black, but to the actual sea level.
  - *Hill Color* is the color of the surface above the *Hill Level*.
  - *Mountain Level* also adds an additional layer to the surface colorization. This is a sea level relative value, so 0 equals to the to the actual sea level.
  - *Mountain Color* is the color of the surface above the *Mountain Level*.

- **Cloud settings**
  Clouds can help to achieve a more realistic look to your planet. It use a noise texture mapped around the planet using triplanar projection.
    - *Cloud Color* controls the color (RGB) and opacity (A) of the clouds.
    - *Cloud Texture* is the texture used for clouds. It must be a tilable texture. It can be a noise texture , or actual cloud photo.
    - *Uniform Scale* scales the cloud texture uniformly.
    - *Offset* parameter can scroll the clouds around the planet. You can use this value to pan the clouds to a desired position.

Planet Halo shader
This shader used in the Halo object to create a fake glow effect around your planet.
Tint property controls the Color of the effect. This is an additive blend shader, which means darker colors fade to the background more than brighter colors.
Falloff property controls the bluriness of the Halo.

Planet mesh

Planet meshes are not procedural. Single sphere mesh with equirectangular UV and a sphere LOD group included in the package.

Customize

You can use the Planet shader without the Generator by using your own heightmaps, or you can use your custom shader combined with the Generator. You can find examples in the package for both.

Platform support and performance

PlanetGen planets intended to use on any platform and both in Desktop and Mobile development.
Texture generation is procedural, no need to store huge amount of textures for your planets, which means smaller build sizes.
The Planet shader is a SM3 shader, and performs well in Mobile devices (but please note, gles 3 support required).
The Planet LOD groups helps to keep your poly count low and framerates high if you have many planets in your scene.

Under the hood

PlanetGen heightmap creation is a hidden rendering process.
By default it is happens when you hit the Generate button in a Planet component. It also automatically called when the scene loaded (both in editor and runtime), or when you enable a disabled Planet.
Rendering the heightmap is fast, It only takes a couple of frames. (depends on the resolution).
Please note, planet texture generation only works in default rendering pipeline at the moment, scriptable rendering pipelines like USRP or HSRP are not supported yet.