1. Introduction

StellarGen can procedurally generate planetary systems containing a number of elements present in real systems, such as stars, rocky planets, gas planets, dwarf planets, asteroids, meteoroids and satellites, as well as other elements that help to build these artifacts like barycenters, orbits, belts, rings, atmospheres and oceans.

These are some of the main features of the asset:

- Generates stars, planets and small bodies
- Procedurally generated textures, normal maps and meshes
- Can generate hundreds of thousands unique planetary systems
- Integrated LOD system
- Control over the quality of texture generation
- Multithreaded generation with control over the maximum simultaneous threads

StellarGen currently only supports PC's running 64-bit version of Windows.

2. Demo scenes

The asset has two sample scenes that can be found at:

- *Assets\StellarGen\DemoScenes\AutoStart*
- *Assets\StellarGen\DemoScenes\ManualStart*

AutoStart contains an example of how to automatically start the planetary system generator, while *ManualStart* contains an example of how to manually start and stop the generation system.

3. Configuration interface

The object *StellarGenController* is responsible for linking Unity to the procedural generation system of *StellarGen*, and additionally contains a settings interface, which allows you to control some parameters of the artifacts generation, as shown in Figure 1.

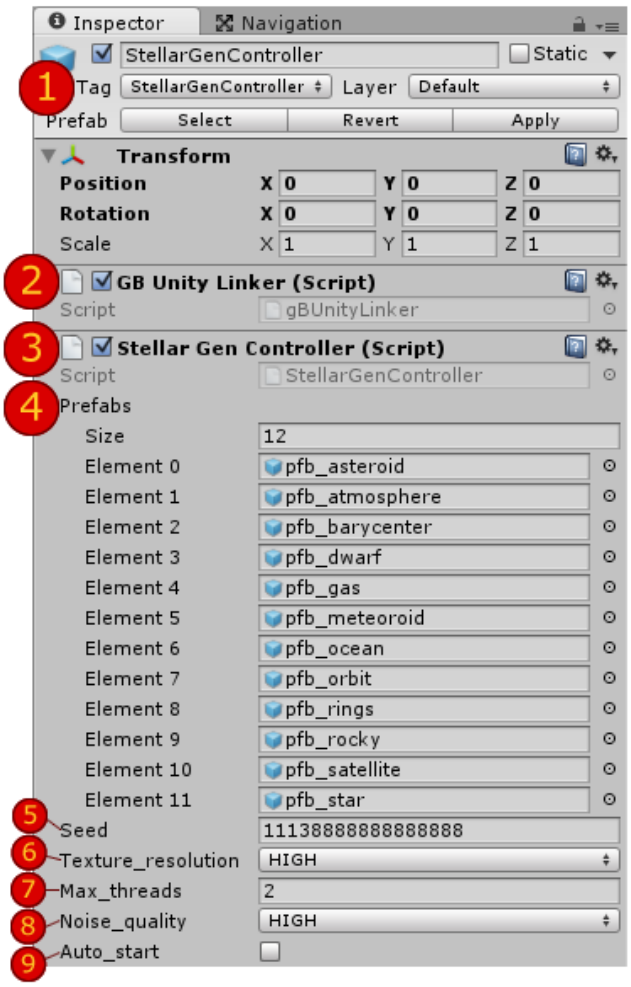

Figure 1. *StellarGenController* configuration interface.

1) Tag *StellarGenController*, should not be changed since it is used internally to locate the object.
2) Script *gBUnityLinker* is used to connect Unity to *StellarGen* internal system, so should not be removed.
3) Script *StellarGenController* contains the prefabs of the objects that compose the planetary system generated and the settings of the procedural generation.
4) Prefabs corresponding to objects to be generated. You can customize them by changing their materials or adding other components to them, however you must keep their names unchanged and should not remove the other existing components. Some of these prefabs have a script called *CustomGlow*, which is responsible for the customization of some aspects of their shader, remove or disable them if you want to use a material with a different shader.
5) The initial seed for the procedural generation system. The value is an unsigned 64-bit integer (*ulong*). It is recommended to change it via script, but if you want to change it via interface you should do it directly through the prefab.
6) The resolution setting of textures, it can be set to low, medium or high. On low quality, the maximum resolution of the textures will be 512x256 pixels. On medium quality, the maximum resolution of the textures will be 1024x512 pixels.

On high quality, the maximum resolution of the textures will be 2048x1024 pixels. It is important to note that not all objects have the same texture resolution settings, such as meteorites, for example, that have lower resolution settings.

7) Maximum number of simultaneous threads. Must be greater or equal to 1. This parameter is used to control the number of threads responsible for the parallel generation of the scene elements. If you do not know what is a thread, just set the value to 2.
8) Quality of coherent noise generation, can be set to low, medium or high. Generally the medium quality is more appropriate.
9) Auto start makes the system to be automatically generated at the beginning of the application. Check the demo scenes accompanying the package in case of doubt.

4. What is in the package?

Below you can check out a brief description of the files contained in the package.

- *Assets**Editor* and *Assets**StandardAssets*
  - Only a few image effects scripts provided by Unity. They are used in the demo scenes, so you can remove them if you want.
- *Assets**Plugins*
  - It contains the plugins of pseudo-random numbers generation and coherent noise generation used for the generation of the planetary system elements.
- *Assets**StellarGen*
  - *DemoScenes*
    - The demo scenes.
  - *Dll*
    - Encapsulates the procedural generation system logic. Where the magic happens.
  - *Resources*
    - *Camera*
      - Contains only a camera with some image effects to enhance the visual quality.
    - *Controller*
      - Contains the StarGenController, responsible for managing the access and settings of the generation system.
    - *Materials*
      - Here are the default materials used in the elements of planetary systems.
    - *Prefabs*
      - Prefabs of the planetary systems elements.
    - *Shaders*
      - It contains the default shader used in the materials.
    - *Skybox*

        - A simple skybox used in the demo scenes.
    - *Textures*
        - A normal map used as a placeholder in the materials that have option for normal maps.

5. Warnings

Only a few warnings which you should be aware.

- You may need to add the tag *StellarGenController* manually, otherwise when you run your game, after building it, the object *StellarGenController* will not be found and the game will not work properly. The same behavior does not occur in the Editor.
- For some reason it is necessary to keep a placeholder in the normal map slot, on the material, for it to work properly. You can find one in *Assets**StellarGen**Resources**Textures*. If you modify any of the default materials provided in the package, do not forget to set the placeholder, otherwise the normal map will not work.
- You can have only one planetary system at a time.

6. Contact info

If you have any other questions, you can contact me via email *mav.jed@gmail.com*.