3D Starfields for Unity

Thank you for your purchase! This document will guide you through using 3D starfields for the first time and tweaking your stars.

Overview

3D Starfields for Unity combines a procedural star shader with Unity's particle systems to render starfields. Each star is a billboard particle, which is drawn using the star shader, according to various parameters. The shader uses no textures, so you will not need to bother with texture compression or resolution to get the best visual quality. The shader also eliminates flickering problems by ensuring that distant stars are always rendered with at least one pixel.

Several scripts are provided to handle the generation and edition of the star particles, so that the particle system, material and particle colors are all properly set up:

- The RandomStarfield script is likely the one you will want to use for your project. It places stars at random positions within the range you define, with the colors of your choice. This is what you should use for background starfields.
- The ObjectStarfield script places stars based on specific game object positions. This is what you would use to create a 3D star map, in which you can travel from star to star, or more generally to display stars at specific positions in the scene.
- The InfiniteStarfieldEffect script is for a more specific use, if you need a "travelling-through-the-stars" visual effect in your game.

Three example scenes are also provided to demonstrate how the Starfield scripts can be used. The scenes and all the scripts and assets they depend on are located in the "Examples" folder, so if you would like to integrate 3D Starfields in your own project, you can skip importing this folder.

Getting started

Quick-start prefab

The simplest way to add 3D Starfields to your project is to import everything except the "Examples" folder, and drop the "Random Starfield" prefab into your scene. Make sure your camera is set to render as far as the starfield's Max Distance. That's it, you're done! Press play and look at the stars.

Examples

If you would like to start with the examples, create a new empty project, and import everything from the package into it. The examples scenes are located in the relevant Examples sub-folders.

The example starfields were made in Linear color space, so go to your Player settings and switch to Linear instead of Gamma. If you are using Unity 4.6 rather than Unity 5+, Linear color

space is not available, so in this case the examples already use the gamma version of the star shader, with tweaked colors.

Proper use

Now that you have your starfield, you might want to check a few things:

- When you tweak material parameters in the editor or ingame, the material will be modified permanently, thus affecting every starfield that uses it. It would be wise to create a new material for every new starfield that you make. To do so, you can copy paste the provided StarParticleMaterial, and set a reference to it on the starfield script in your scene. If you intend to have only one starfield in your project, you can ignore this.
- In Unity 5+, all the materials in the package use the linear version of the shader by default. If your project uses gamma color space, you should switch the shader to the gamma version. You can do so in the material properties. Color blending is a bit strange in gamma mode, so if you have a choice I recommend switching to linear.
- The starfield scripts have a star size parameter. It is set to a big enough value by default, but if you encounter flickering of distant stars, you will want to increase it until the flickering disappears.
- If you would like to live-preview the starfields in the editor, it would be a good idea to set your editor's scene background color to something dark, or to display your skybox in the scene if you intend to use one. Ideally your scene view should look like your game view when you play. Position your scene camera correctly to look at the stars, preferably from the center.
- Most projects will either want to navigate inside a 3D starfield, or display a background starfield of distant stars. For background starfields, it is best to render the starfield in a separate background camera, so that the 3D stars don't move as much (or at all) when you move your main camera. This is what the "BackgroundStarfield" example does - check the Examples section of this documentation for more details.

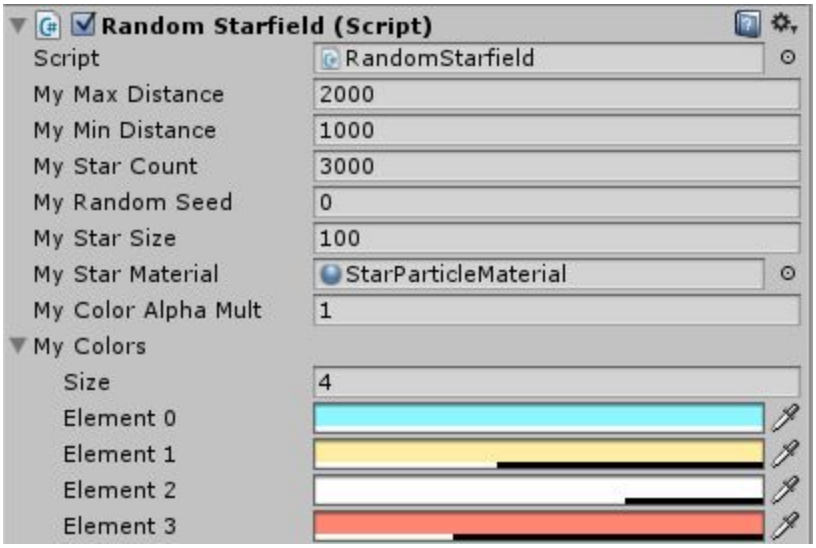
Tweaking

Once you have a Random Starfield in your scene, you can change the parameters and view the result instantly in the scene view. Every time you change a value on the script, the starfield will be regenerated with the new properties. A detailed explanation of the script parameters is provided in a later section of this document.

In addition to the starfield generation parameters and colors that can be changed on the script, you can change how the stars are rendered by modifying the material properties. You will find those either on the starfield object, or by directly inspecting the material.

Starfield Scripts

Random Starfield



This is the script for generating 3D starfields. It will place stars at random positions between Min Distance and Max Distance. Colors are picked randomly within the set of colors provided, and if that is empty they will be white.

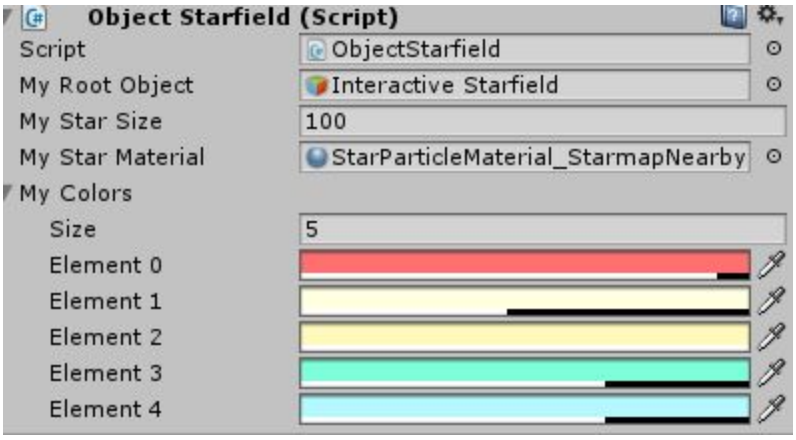
The star size is the particle billboard size in unity units. It has to be big enough so that the most distant billboard has at least 1 pixel visible. This is to avoid flickering issues.

When choosing colors for your stars, the alpha will be taken into account. Lower the alpha to make fainter stars for a given color. Color Alpha Mult lets you change the alpha for all colors at the same time, without having to change the material variables.

All the random choices made by the script (color and position of each star) are initialized with Random Seed. This means that as long as you keep the same seed, and the same parameters, your starfield will be the same every time. This allows you to tweak the starfield in the editor, then get the same exact result ingame. Change this value to generate a new version of the starfield.

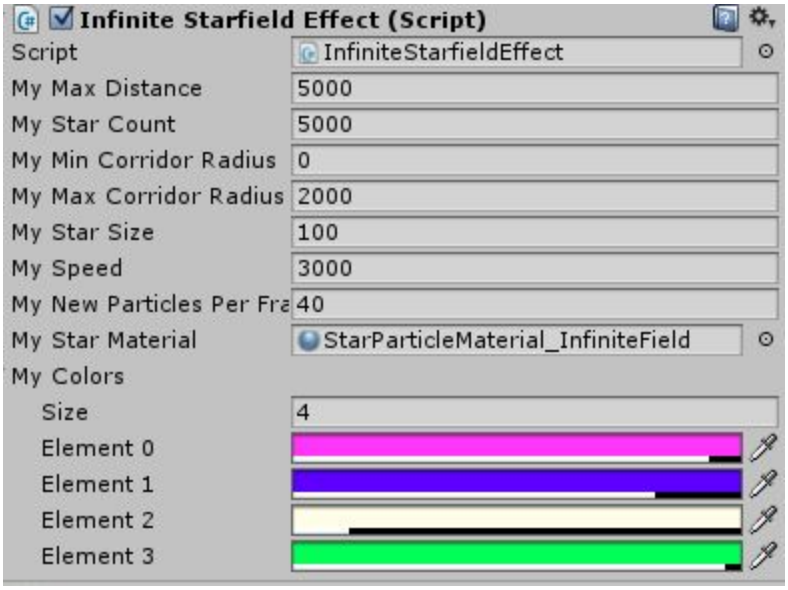
New in 1.1.1: the “Center Around Zero” option has been added to provide retrocompatibility for existing users. RandomStarfield will now generate stars around the object’s position, rather than around (0,0,0), unless “Center Around Zero” is used.

Object Starfield



If you want your starfield to match the positions of a specific set of objects, this is the script you will want to use. The Root Object is the transform parent to all the stars you want to use. Check the Starmap example scene for an example. Another possible application of this script is to manually place stars in your background starfields, in order to form a constellation.

Infinite Starfield Effect



This is a more specific script but it might be useful in some games. Stars are generated in a circle away from the camera, at Max Distance. The stars then move towards the camera at the specified speed, thus forming a cylinder/corridor. The particles are timed to disappear right after they pass the camera, so no stars are rendered behind it.

New Particles Per Frame is a value set to limit the number of new particles spawned so that not all the stars are spawned at the same time at the beginning of the play session. You may have to adjust it if you change the maximum distance or speed of the particles.

Making your own

If you would like to generate stars your own way, just have a look at what the scripts above do. They both derive from StarfieldBase, which has a couple of functions to take care of setting up a particle system and important hidden properties on the material.

Material Tweaks

The shader used to render the stars provides many variables that you can tweak in the material properties.

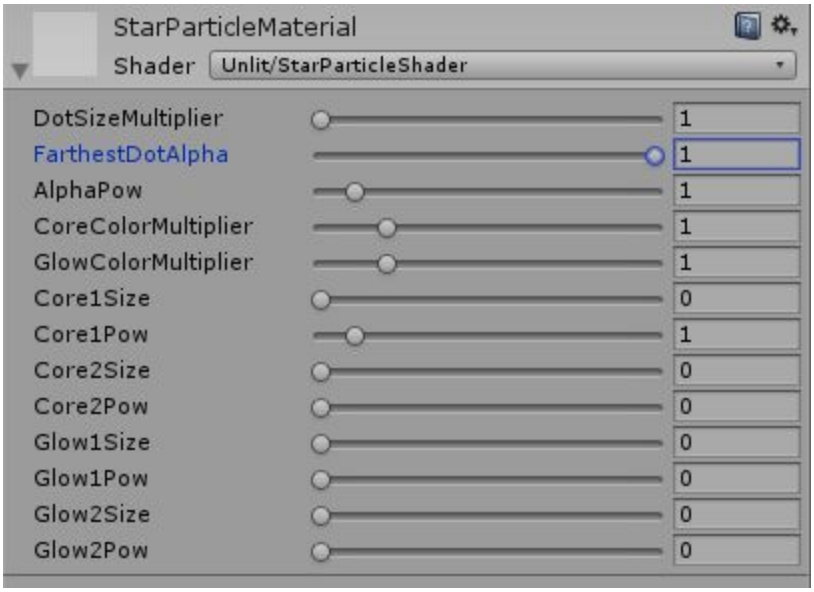
Main controls

DotSizeMultiplier	To avoid flickering and improve the look of starfields, distant stars are always rendered with at least one pixel. You can use DotSizeMultiplier to increase this.
FarthestDotAlpha	Changing this value modifies the alpha of the minimal dot for distant stars, to make them seem fainter. The distance is based on whatever the starfield scripts are using as a maximum distance.
AlphaPow	This is a master control for the alpha of all the star components (except the minimal dot), which comes in handy to tweak the glow of the starfield easily. The internal calculation sets the alpha to pow(alpha, AlphaPow), and since alpha is between 0 and 1, a bigger AlphaPow makes fainter stars.

Core and Glow

There’s 4 components to the stars, aside from the minimal dot - Core1&2, Glow1&2. Not all 4 are needed to make a star, but having 4 gives more tweaking options.

Each of them has a size and power variable, which allow you to tweak how the star color fades when going farther from the center. It’s easier to understand how it works if you first set up your material like this:

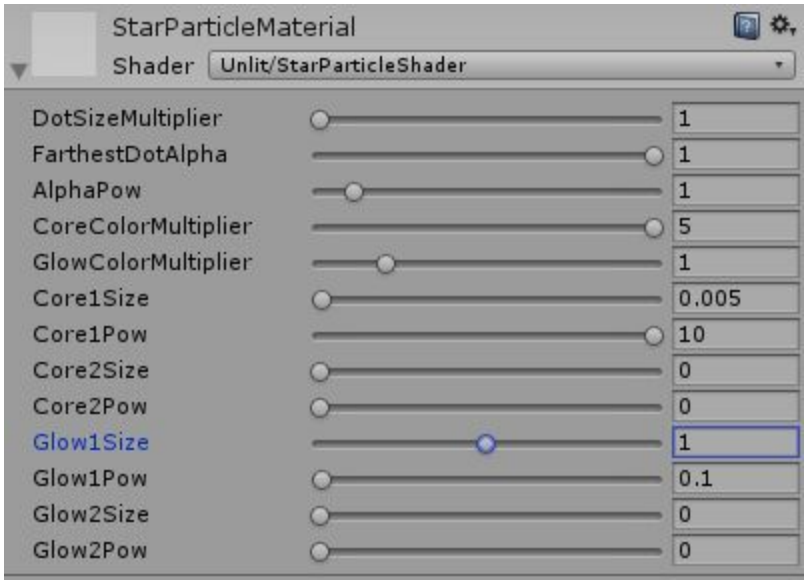


In this configuration, the stars appear only as one white dot. If you increase Core1Size to 1, the star color will cover all of the billboard surface of the particle, by fading from the center to the border.

If you increase Core1Pow, the star color will fade more slowly, so the color is stronger overall, especially in the center. If you decrease Core1Pow below 1, the fading goes faster, and the color is fainter overall.

Now use Core1Size = 0.5 and Core1Pow = 1, and tweak the CoreColorMultiplier. If you go above 1, the color becomes whiter in the center, like a bloom effect.

Core and Glow are separated so that the color multiplier can affect them separately. The idea is to use small core sizes with high power, combined with large glow sizes with low power. Try this for example:



This is a simple star that uses only Core1 and Glow1. From there you can tweak the other variables to see how they modify the effect, or you can try using Core2 and Glow2 to add more color layers on the star.

Notice also that increasing the color multiplier for the glow makes a very bright star, to the point that the entire billboard is covered. You will need to decrease the glow size accordingly to make sure no coloring reaches the border.

Scintillation

Star scintillation is available from version 1.1, but disabled by default. To activate star scintillation, check the boolean on the StarfieldBase script, or in your own scripts call EnableKeyword("ENABLE_SCINTILLATION") on the material. Scintillation is visible only when playing.

You can tweak the scintillation speed and strength on the material. The speed simply multiplies the time used by the shader, while the strength is the percentage of the star's alpha that can be affected by twinkling: 0.1 means that the star oscillates between 90% and 100% visibility, while 0.7 makes it oscillate between 30% and 100% visibility.

Examples

Background Starfield

Please make sure to read the “Random Starfields” script explanation before reading about this example.

In this demo scene, the random starfield is used as a distant background for the stars. As mentioned previously, it is best to render such a starfield in a separate background camera, so that the 3D stars don't move as much (or at all) when you move your main camera.

In order to do that, the background starfield is set to a "Background" layer, and a background camera is configured to only render objects in the "Background" layer. In addition, this camera is placed at a lower depth than the main camera, and it uses the Solid Color clear flags, whereas the main camera uses the "depth only" clear flags.

Finally, if the main camera rotates around the rocky planet, then it rotates at the megameter scale, whereas stars would be light years away, a light year being 10 000 000 000 megameters. In order to convey that difference of scale, when the main camera moves one unit, the background camera should move 1/10000000000 units - that's pretty much zero, so I didn't even bother scripting it :) However the background camera should rotate as normal, which means rotating the same way as the main camera (there's a script for that).

Starmap

The Starmap demo also has a background starfield, but since we are moving a ship in the stars, everything is at the same scale, and therefore uses the same camera.

The main starfield in this demo, however, is the one with brighter stars that is generated from a list of game objects. Check the "Interactive Starfield" game object in the scene - it uses an Object Field Generator for the purpose of this demo, which simply spawns game objects in a large area. Then the "Object Starfield" script, as described in the Starfield Scripts section, generates the star particles to match the game object positions.

The prefab for the game objects contains a Sphere collider, so the objects serve as physics for our star particles. The Starmap Controls object and script performs raycasts every frame to detect if a star has been hovered or clicked, and move the ship accordingly.

Infinite Starfield Effect

This is pretty much explained in the section Starfield Scripts section.

Known issues

- When you use Undo/Redo, or sometimes when you select a different object in the hierarchy, then select the Starfield again, Unity will clear the starfield. I have found no way to prevent that, but all you need to do to refresh the starfield is make sure it is selected, then change one of the variables or click the Refresh button on the component.

Final thoughts

If you have any questions regarding this asset, feel free to get in touch with me at unity@timecraftgames.com.

Release notes

Version 1.2

Added background space skybox to show how 3D starfields look when combined with skyboxes; Fixed Unity 5.6 warnings.

Version 1.1.1

Fixed “deprecated” warnings introduced with Unity 5.3; RandomStarfield can now generates stars around its transform position, rather than zero. Added a “center around zero” option so that nothing changes for existing users.

Version 1.1

Added star scintillation