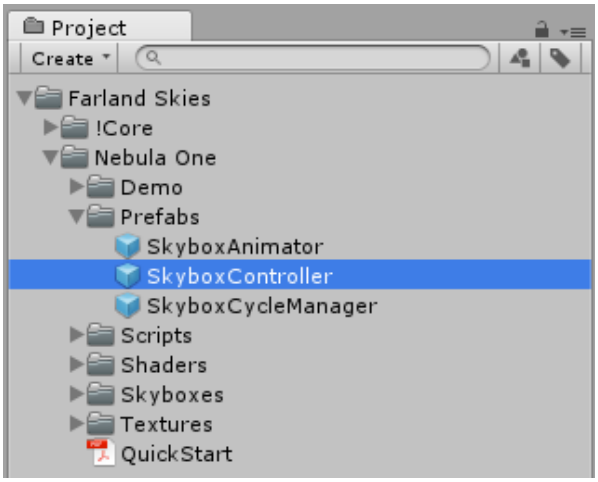

NEBULA ONE
QUICK START

Integrating and configuring skybox

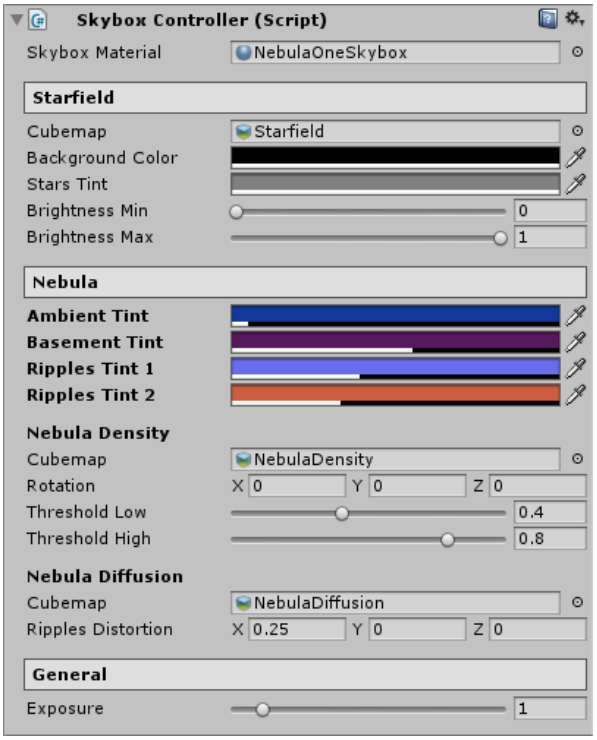


It's pretty easy to integrate skybox to your scene. All you need to do it's just drag the ***SkyboxController*** prefab to the hierarchy window.

Skybox will be assigned automatically for this scene, and now you can use ***SkyboxController*** object to manage its settings, either from scripts or directly in the editor.

Editor properties

Starfield



This section allows you to customize background color and stars tint. You can adjust the number and brightness of the stars with min/max threshold sliders.

Skybox controller comes with ready-to-use starfield cubemap, but you can use a custom one if you want to. It can be grayscale or in color, but only red channel will be used to draw starfield.

Nebula Colors

Ambient Tint applies the color filter to mild ripples across the whole skybox, regardless of nebula density.

Basement Tint refers to nebula basement haze, which is not affected by ripple distortion.

Ripples Tints 1-2 apply to two layers of nebula cloudy ripples. Usually, most dominant nebula colors.

Nebula Density

Density cubemap defines the shape of the nebula. The brighter the area on the cubemap, the denser and opaque nebula is in this area. ***Low/High Threshold*** sliders allow adjusting overall nebula thickness without modifying the cubemap.

Skybox controller comes with ready-to-use density cubemap, but you can use a custom one if you want to. It can be grayscale or in color, but only red channel will be used for nebula density. Also, it does not have to be large; 128px is good enough.

Nebula Diffusion

Diffusion cubemap determines the appearance of the nebula in the details, as well as final ripple distortion values. If you're familiar with ShaderLab, take a look at NebulaOneSkybox.shader to understand how it's used exactly. Otherwise, just leave it as is, the default value should work quite well.

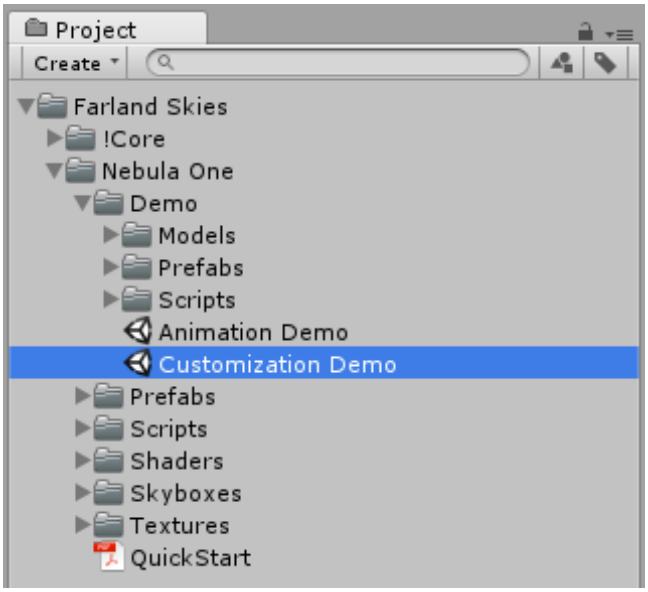
Ripples Distortion is a vector with distortion coefficient for each direction. It affects two layers of nebula cloudy ripples and allows to diversify nebula appearance. Not recommended to use values greater than 0.25f.

Access From Scripts

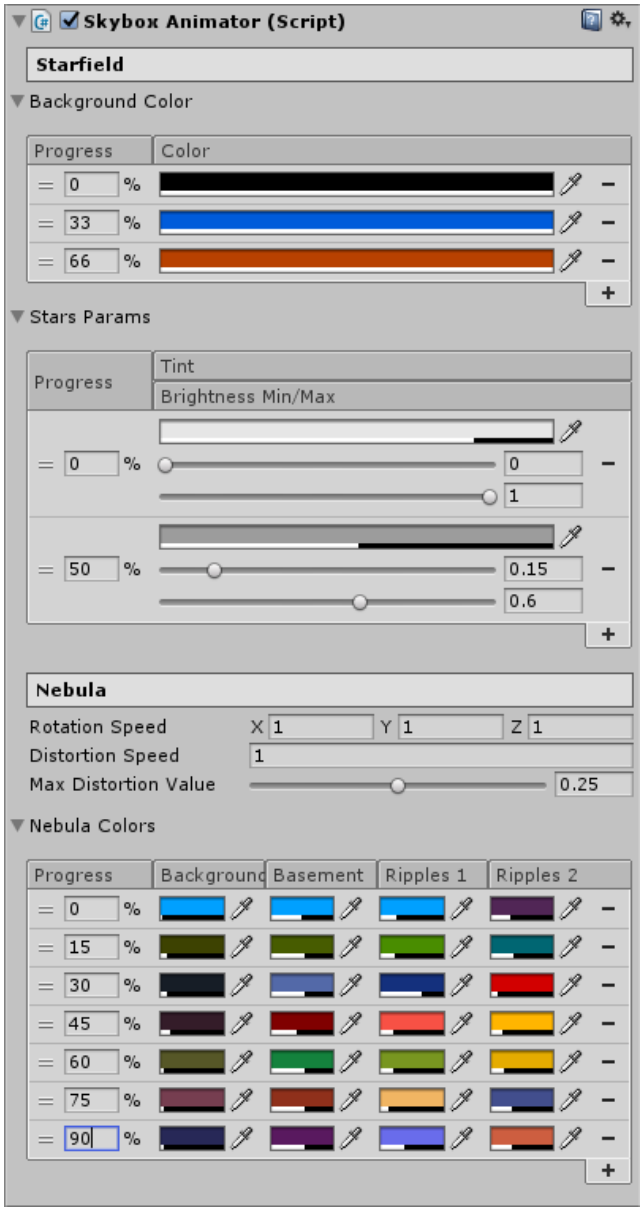
SkyboxController implements the "singleton" pattern, so you can easily access all its properties from anywhere in your scripts:

```
SkyboxController.Instance.BackgroundColor = Color.blue;
SkyboxController.Instance.StarsBrightnessMax = 0.75f;
SkyboxController.Instance.DensityRotation = Vector.zero;
```

Please take a look at the ***Customization Demo*** scene to get more details.



Skybox Animator



This package comes with a handy script that allows to animate skybox by changing starfield and nebula parameters over time.

At first, you need to drag the ***SkyboxAnimator*** prefab somewhere to your scene.

Then adjust prefab properties to fit your needs:

- ***Background Color*** foldout contains a list of starfield background colors for one animation cycle. Each list item includes corresponding cycle *"progress"* filed that should be specified in percents (0-100).
- ***Stars Params*** list allows you manage stars tint color and brightness over time. It works in the same way - just specify a progress in percents and parameter values for each element.
- ***Nebula Colors*** *list* uses the same approach as above for setting it up.
- ***Rotation Speed*** is a vector that specifies the angular speed coefficients of the nebula around corresponding axes.
- ***Distortion Speed*** is a coefficient that determines how fast nebula distortion will be changing over time.
- ***Max Distortion Value*** allows to specify maximum nebula ripples distortion during an animation cycle. Not recommended to use values greater than 0.25f.

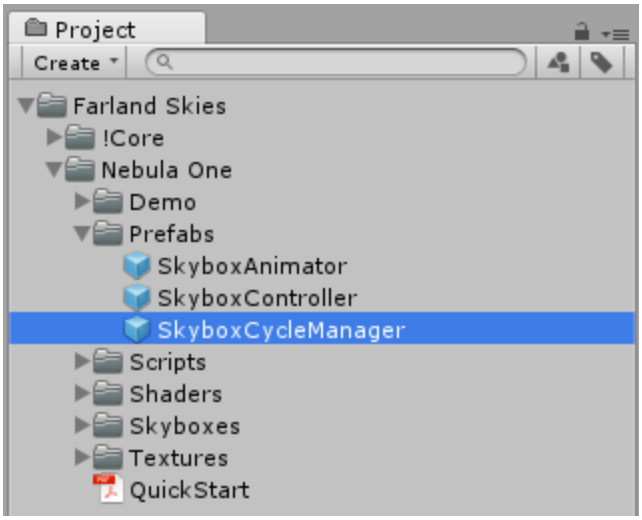
SkyboxAnimator script will automatically interpolate values between list items and update skybox parameters depending on its ***CycleProgress*** property.

However, ***SkyboxAnimator*** doesn't run through animation cycle automatically. It requires few lines of code by yourself, but in this way you can fully control your animation cycle duration, make your time flow non-linear, etc.

The simplest example to do so:

```
public void Update()
{
     SkyboxAnimator.Instance.CycleProgress
          += (Time.deltaTime / CycleDuration) * 100f;
}
```

If you don't want to bother with scripts by yourself, then you can just grab ***SkyboxCycleManager*** prefab and add it to your scene. It will manage animation cycle progress automatically.



Please take a look at the ***Animation Demo*** scene to get more details.