Glowy Space V1.0

By Ricardo Concha – Nemoris Games

"Thank you for getting this pack. Let's check how it can help you improve your game or make things easier"

Glowy Space is an asset pack that allows you to create scenes set in outer space with just a few images and a lot of customization. The focus of this pack are the scripts that control the behavior of all the elements in screen, including a fully customizable 4-Direction parallax effect.

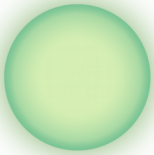
Art Assets

The art assets were created with efficiency in mind, consuming a minimal amount of memory and using as few sprites as possible to get **full quality** without increasing the draw calls.

All art assets allow scaling without losing image quality (the maximum scale amount allowed depends on the target screen resolution)

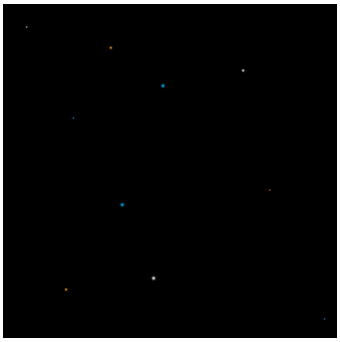
This pack allows the user to compose complex assets for their project. It includes the following sprites:

1. Planet: This asset is the most used and important in the pack. It represents a generic planet with a green base color that create a great effect when other colors are applied to it.



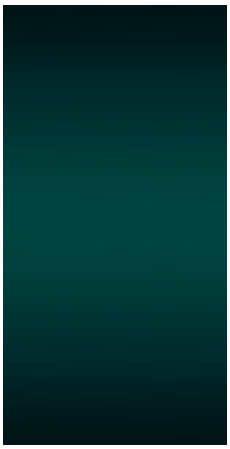
2. Star: This asset is made to attract attention with its own glow. Any color can be applied to it, and it can even be used in GUI.

3. Small Star Group: This asset solves the problem of needing to use many stars moving in parallax, while avoiding a high memory use. Also, using too many stars in a single image makes the parallax look strange. The Small Star Group asset provides a middle ground solution that works optimally and looks good.



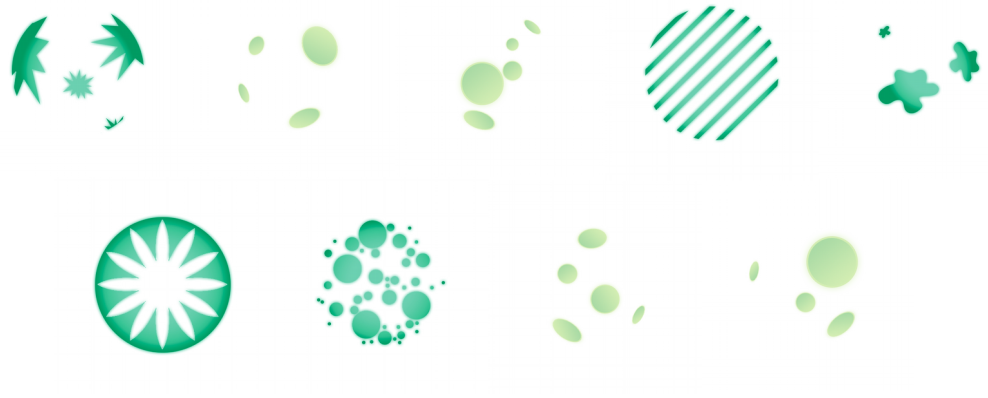
4. Big Planet: This asset is a very big planet that can be used as part of the background. It has a fade effect to reduce its image size without losing quality when is scaled too high.

5. Space Background: This asset is a very long image, vertical wise. It was made to correctly scale horizontally and fill any background needed.



6. External Glow: This asset is an effect that creates something similar to an energy bubble. Ideal for shiny objects.

7. Planet Layers: These layers are applied over planets to take advantage of their glow effects, and can be used in both the normal or big planet assets.



8. Planet Ring: This ring also works as a layer to apply over a planet. It's mainly used to add an extra layer to the big planets.

Scripts, Behaviors and Components

As said, the key points in this asset pack are the behaviors controlled by simple, customizable scripts. These scripts were written to be used in any way that the user may need.

1. **SpaceManager.cs**
   This script is used to store important information regarding a scene. It defines and stores the **scroll direction** for all space assets.

2. **ParallaxMovement.cs**
   This is a very important script, it manages the parallax effect in each object using the scroll direction given by the selection in the SpaceManager object (included in the Prefabs folder).

   The parallax effect uses a random speed given by **minSpeed** and **maxSpeed**. A 0f value will make the object to move along with the camera, a value of less than 0f will make the object to move ahead of the camera

   To determine if the object is completely out of screen the script, it adds a **limitOffScreen** parameter that reads the viewport's value. For example, if the user want that the object's limit is a half size of the screen, then the limitOffScreen value should be 0.5f

   The **behaviourOnExit** determines what the object should do when it exits the screen (including the limitOffScreen value). **Destroy** will destroy the object and **Regenerate** take the object, reposition it according to the scroll direction and its children have any random type component, it gets randomized again.

3. **CameraMovement.cs**
   This script moves the camera according to the **speed** set by the user and in the scroll direction determined by the **SpaceManager** object (included in the Prefabs folder).

4. **ShootingStar.cs**
   This script controls the shooting star and meteor prefabs in the assets. It positions them according to the scroll direction, sets their direction pointing to the center of the screen and puts the object to 'sleep' until the **spawnTime** value is reached.

5. **RandomColor.cs**
   This script randomly picks a color from a list given by the user and applies it to the **<SpriteRenderer>** component in the same object.

   It also has a **invisibleProbability** value that calculates a random number between 0f to 100f and, if that number is less than the given probability value, makes the sprite invisible.

6. **RandomSprite.cs**
   This script randomly chooses a sprite from a list given by the user and applies it to the **<SpriteRenderer>** component in the same object.

7. **RandomRotation.cs**
   Make the object rotate at the speed given by **rotationSpeedMax**. If the **randomize** bool is true, the script will choose a speed between 0f and rotationSpeedMax.

8. **RandomSize.cs**
   This script applies the **multiplierMax** value to the current objects localScale, multipliying it. For example, if an object has a localScale of (1f, 2f, 3f) and the multiplier is 4f, the object's new size will be (4f, 8f, 12f). The multiplier used is a random number generated between 1f and **multiplierMax**.

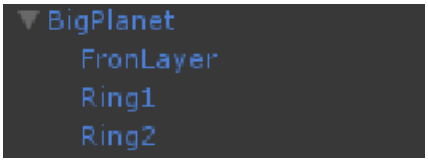
Prefabs

To have all the components working correctly so you can just pick the Prefabs and incorporate them into your project, we have prepared a scene called **ExampleScene** and the following Prefabs:

1. **SceneManager**
   This prefab determines the scroll direction of the scene. Other prefabs use this value to change their behavior. The user can set this value to any of the four given directions (up to down, down to up, left to right, right to left) before playing the scene.

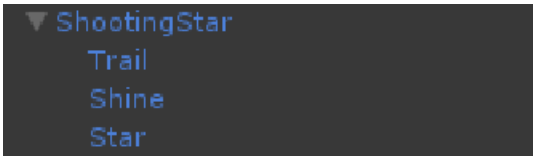
2. **Planet**
   This prefab is a planet with random behaviors such as RandomColor, RandomSize, RandomRotation and the ParallaxMovement value to move in the scene. It contains the following children objects:
   - **FrontLayer**: A layer that contains different sprites in the RandomSprite component that are located over the planet sprite.
   - **Atmosphere**: Based on the planet sprite, this object simulates a visual effect over the planet.
   - **PivotMoon** and **Moon**: The pivot creates the orbit rotation for a smaller moon object that uses the planet sprite.

3. **BigPlanet**
   This prefab works just like the Planet prefab, except that it does not have any Moon objects as children. Instead, it has two rings in the planet with a high probability of been hidden.



4. **Star**
   This prefab has random behaviors such as RandomColor, RandomSize, RandomRotation and the ParallaxMovement to move in the scene.

5. ShootingStar
   This prefab uses the ShootingStar component. Its children objects draw the trail behind the star, its shine (represented by the Glow sprite) and the star that is the main sprite.



6. **Comet**
   This prefab behaves just like the ShootingStar, with the difference that instead of a star, it uses a comet body that is represented by a planet sprite.

7. **SmallStarGroup**
   This prefab represents a group of stars that are perfect for background, creating a depth effect using parallax. It has been optimized to use the less resources possible because it needs to be replicated several times to create a perfect effect. A proof of this is that in the ExampleScene, this prefab is replicated 20 times, which constitutes the main consumption of memory and since the memory use remains consistently low after that, it does not represent a problem in runtime.

Final Note

Thanks for using this library. I hope you find it useful. I'll try my best to provide support regarding any issues, so you can contact me directly at rconcha@nemorisgames.com and I'll reply as soon as possible.

I definitely hope to expand and add more features in the future, so any ideas and feedback are welcome.

Please rate this asset and comment so more people can be aware and use it to improve their games.

Best regards,
Ricardo Concha – Nemoris Games