# Procedural Celestial Rendering for 3D Navigation

## A Technique to Render a Parametric Celestial Skybox


Alain Galvan
agalvan@fiu.edu

Francisco Ortega
fortega@fiu.edu

Naphtali Rishe
ndr@acm.org


## Introduction

3D navigation is an important pillar of user interaction. We believe that the cosmos provides such an environment to derive generic gestures (or actions) that may serve in other 3D visualization environments.

We present a novel technique to render a parametric celestial skybox with the ability to light environments similar to natural color corrected images from telescopes.

We first pre-compute a spherical ray map that corresponds to the cubemap coordinates, then generate stars and dust through a combination of different noise generation shaders.

## Related Work

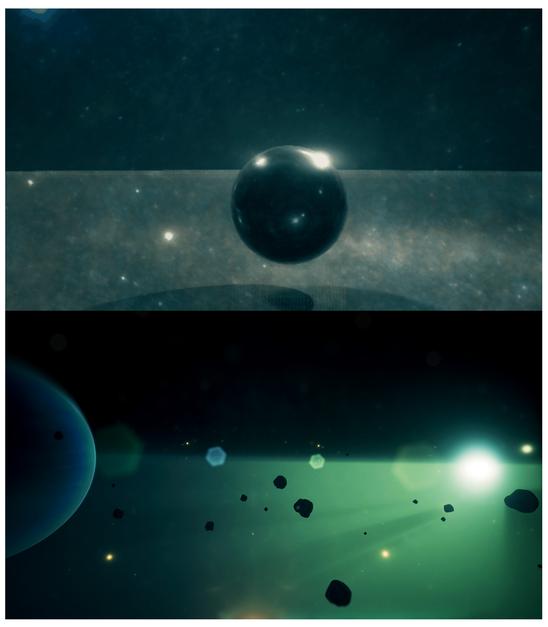
- **Elcott et al.** for Final Fantasy XV generates a sky with raymarching, and used light probes for lighting the scene.
- **Limberger et al.** attempt to render stars using billboards.
- **Jensen et al.** provided a model for physically accurate night skies.
- **Elek and Kmoch** provided a model for spectral scattering used in Unreal.
- **Trindade et al.** used cubemaps to rendering multi-scale 3D navigation environments.

## Approach

- We first formulate a physically-based model for starlight and stardust based on the user's origin position, star size, and temperature.
- We combine this with **volumetric raymarching** (a volumetric form of ray-casting) techniques for clouds and dust.
- The result of the cubemap is a real-time, efficient, and realistic environment that can be used either as a background, a reflection map, or an ambient cubemap light source.
- A more rich animated scene can be created because of the lights in the scene are controlled by a shader.

## Algorithm

- The Algorithm works by processing a cubemap render target. A cube based spherical directional map is generated by the shader, as shown in Figure 3, which is then used as an input to a four dimensional noise generation algorithm based on the work of Perlin et al. to create volumetric diffuse effects.
- The output of the noise generation algorithms is composited with a white noise function mapped to sharp changes in luminosity values.
- Due to the intense processing power required to render an entire cubemap for every frame, we employ update throttling and dynamically change the skybox resolution. Our system is implemented as a plugin for Unreal Engine 4. Our source code is easy to use and requires minimal changes to existing scenes.



## Mapings

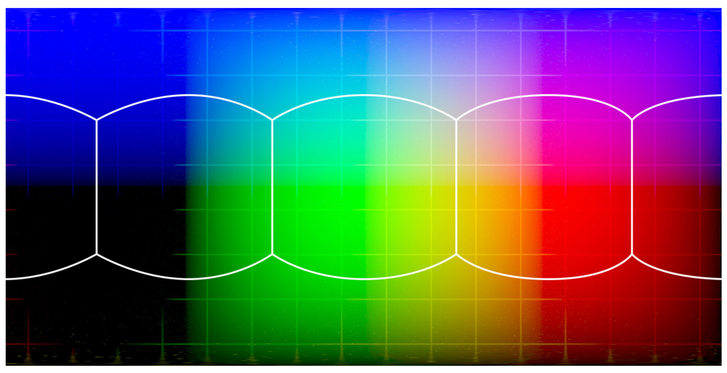


**Figure 3**: A spherical direction map used as an input for noise generation. The latitude/longitude and round lines were added to help distinguish cube faces. This map provides a continuous domain to sample points for voronoi noise.

## Elicitation

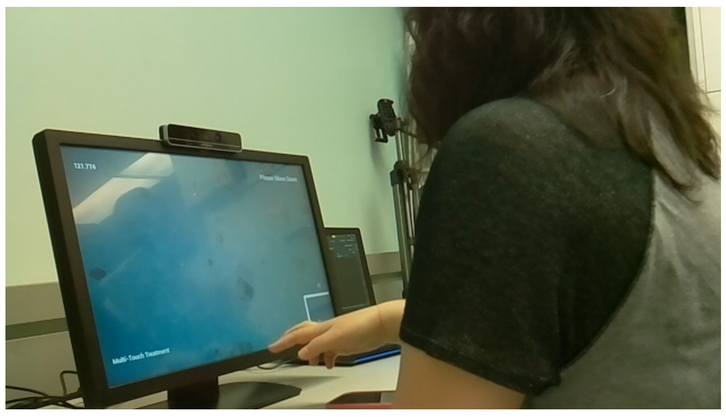



**Figure 4**: Our first evaluation of the system was in a gesture elicitation study. Users navigated based on instructions (e.g., move left, rotate, etc.)

## Evaluation

- During the trials, we were able to constantly refresh the generated sky at 60 frames per second on an **Nvidia 980 GTX** & **980m** (circa 2014) with texels of 1024 pixels per cube face.
- On an **Nvidia 650m** (circa 2012) the effect was too taxing, running at 22 frames per second at 1024 texel size, however 256 texel size ran smoothly at 60 frames per second. Currently, the fastest consumer Nvidia graphics cards include Titan X and 1080.
- The system performed successfully with the load of a multi-touch and Intel Real-Time sense camera.

## Conclusion

- Our approach provides other researchers with the ability to create large expanses of space for user interaction, in particular 3D navigation.
- The use of cubemap lighting looks similar to color corrected photographs provided by NASA.
- In the future we would like to provide a Vulkan version of our method to compare performance with the DirectX API as well as to further optimize the algorithm by taking advantage of CPU concurrency on each face of the cubemap.