Procedural Celestial Rendering for 3D Navigation

Alain Galvan* Francisco Ortega† Naphtali Rishe‡

School of Computing and Information Sciences
Florida International University

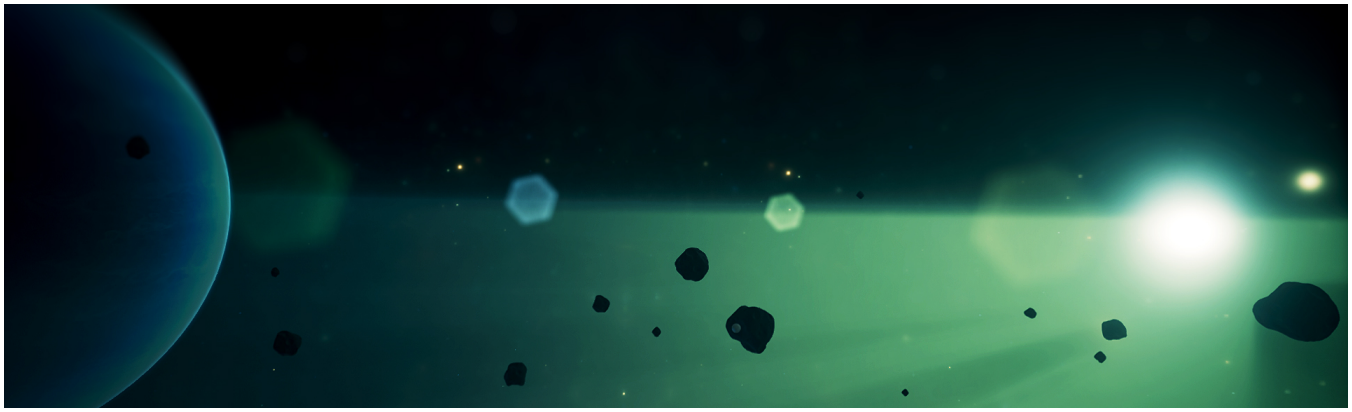

Figure 1: *An example 3D environment using our method of skybox generation.*

Abstract

Finding the best suitable environment for 3D navigation that utilizes at least six degrees-of-freedom is still difficult. Furthermore, creating a system to procedurally generate a large virtual environment provides an opportunity for researchers to understand this problem further. Therefore, 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.

Keywords: natural phenomena, 3d navigation, physically-based rendering, image-based lighting

Index Terms: I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture—Fractals; H.5.2 [User Interfaces]: 3D Navigation—User Interaction

1 Introduction

3D navigation is an important pillar of user interaction. Navigation includes travel (the engine that takes us from point A to point B) and wayfinding (our cognitive spatial understanding of the environment) [**?**]. 3D travel may include six degrees-of-freedom (DOF) depending on the domain requirements; however, in many instances, a user may require less than 6 DOF. This leads to a question on what type of environment is best suited to test 6-DOF interactions (translations and rotations on the x,y, and z axes)? We believe that the cosmos provides such environment to derive generic gestures (or actions) that may serve in other 3D visualization environments. The primary reason for this observation is that users may find it necessary to perform all the translations and rotations for this environment.

This paper presents one approach that may allow other researchers to easily create large sets of data for interaction studies. This will be valuable to the 3D User Interfaces (3DUI) community. Instead of manually creating spaces, there are methods to automate this process (see Section 2). We propose a newer method that provides an efficient real-time algorithm to render celestial skies as the problem of large expanses of space still poses a problem for real-time rendering. For example, some common techniques include the use of pseudo-random noise clamped to specific intervals or the use of static measured data. However, these techniques fail to accurately represent space or fail to offer parametric control.

2 Related Work

There has been plenty of research in rendering physically-accurate day and night skies, but ambient light from stars and space has not been given much attention. For Final Fantasy XV, Elcott et al. attempted a similar technique of procedurally generating a sky with raymarching, but used light probes rather than an ambient cubemap to apply lighting effects to the environment [**?**]. Limberger et al. attempted to render stars using billboards that are processed with a pixel shader to adjust their brightness and color [**?**].

Jensen et al. provided a model for physically accurate night skies but opted to use measured data to diffuse galactic and cosmic light [**?**]. They enumerated sources of night illumination that we've opted to use in this paper, namely diffuse galactic/cosmic light. Elek and Kmoch provided a model for spectral scattering but focused on the effect this had on planetary atmospheres [**?**]. Similarly, other researchers have focused on a physically accurate atmospheric scattering model [**?**, **?**]. We make use of the technique Bruneton and Neyret provided for planetary bodies in our example, but opt to focus on outer space [**?**].

Our approach uses a cubemap (a 3D texture consisting of 6

*e-mail:agalvan@fiu.edu
†e-mail:fortega@fiu.edu
‡e-mail:ndr@acm.org

```
float3 rayDirection = normalize(mul(uniforms.
    ViewMatrix, float4(UV-float2(0.5, 0.5), .5,
    0.))).xyz;

OutColor.a = 1.0;
OutColor.rgb = stars(rayDirection, uniforms.
    elapsedTime);
OutColor.rgb += voronoi(rayDirection, uniforms.
    elapsedTime);
```

Listing 1: *Fragment Shader for Celestial Rendering*

faces), which has been used as a fast way of providing ambient lighting for mobile devices, and has been by shown by Trindade et al. to also serve in the use of rendering multi-scale 3D navigation environments [**?**].

3 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 model is applied via a shader (see Listing 1) to generate a high-dynamic range cubemap. 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. What is most interesting about our approach is that if we applied our technique to a ambient cubemap, a more rich animated interface would be created because of the described dynamic lighting.

Algorithm 1 works by processing a cubemap render target. A cube based spherical directional map is generated by the shader, as shown in Figure 2, 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. We utilized Varonoi noise because Zaninetti observed that the distribution of galaxies in the universe closely resembles Voronoi diagrams [**?**]. Finally, 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. It is currently available for download at `https://github.com/OpenHID/realtime-celestial-rendering`.

Algorithm 1 General Algorithm For Celestial Rendering

Ensure: all input/output buffers must have been initialized
1: **for** $cubeFace = 1$ to 6 **do**
2: $runShader(Texture[cubeFace], RotationMatrix[cubeFace])$
3: **end for**

4 Evaluation and Results

Our first evaluation of the system was during a gesture elicitation study, where users navigated based on instructions (e.g., move left, move right, rotate, etc.), as shown in Figure 3. The system performed successfully with the load of a multi-touch and Intel Real-Time sense camera. 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

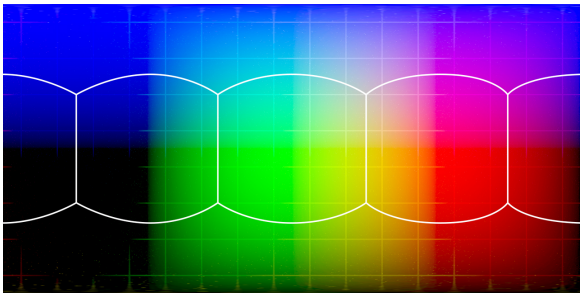

Figure 2: *A cube based directional map, overlaid with both lines of latitude and a cubemap adapted from [**?**].*

at 22 frames per second at 1024 texel size; however, 256 texel size ran smoothly at 60 frames per second.

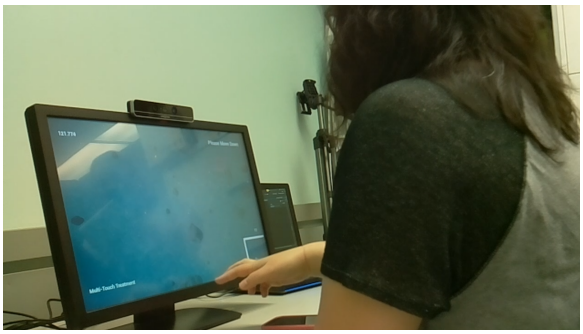

Figure 3: *Participant using Celestial Sky*

5 Conclusions and Future Work

We have presented a method for a physically-based, real-time rendering of celestial skies, building off of work from [**?**]. Our approach provides other researchers with the ability to create large expanses of space for user interaction, in particular 3D navigation. Our method results in a celestial environment that is easier to manipulate in situations where the sky transforms, such as in a space ship or speeding up time. In addition, the use of cubemap lighting that looks similar to color corrected photographs provided by NASA provides an optional light source for dim environments such as night time.

In the future, we would like to provide a Vulkan implementation of the algorithm to compare performance with directX devices as well as to further optimize the algorithm by taking advantage of CPU concurrency on each face of the cubemap.

References

[1] D. A. Bowman, E. Kruijff, J. J. LaViola, Jr, and I. Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2004.

[2] E. Bruneton and F. Neyret. Precomputed Atmospheric Scattering. *Computer Graphics Forum*, 27(4):1079, 2008.

[3] S. Elcott, K. Chang, M. Miyamoto, and N. Metaaphanon. Rendering techniques of Final Fantasy XV. *ACM SIGGRAPH 2016 Talks on - SIGGRAPH '16*, 2016.

[4] O. Elek and P. Kmoch. Real-time spectral scattering in large-scale natural participating media. 2010.

[5] Epic. Creating cubemaps, 2016. [Online; accessed 20-Dec-2016].

[6] J. Haber, M. Magnor, and H.-P. Seidel. Physically-based simulation of twilight phenomena. *ACM Transactions on Graphics*, 24(4):1353, 2005.

[7] H. W. Jensen, F. Durand, J. Dorsey, M. M. Stark, P. Shirley, and S. Premože. A physically-based night sky model. *Proceedings of the*

28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01, 2001.

[8] D. Limberger, J. Engel, and J. Döllner. Single-pass rendering of day and night sky phenomena. In *Proceedings of the Vision, Modeling, and Visualization Workshop 2012*, pages 55–62. Eurographics Association, 11 2012.

[9] K. Perlin. Improving noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 681–682, New York, NY, USA, 2002. ACM.

[10] D. R. Trindade and A. B. Raposo. Improving 3d navigation in multiscale environments using cubemap-based techniques. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 1215–1221, New York, NY, USA, 2011. ACM.

[11] L. Zaninetti. Photometric Effects and Voronoi-Diagrams as a Mixed Model for the Spatial Distribution of Galaxies. *The Open Astronomy Journal*, 6(1):48, 2013.