Group 1 Final Project

Frankel Zhao, Young, Kevin Zhen, Jack Zhang

March 1, 2024

Course PACSSR-301005: Computer Graphics

1 Introduction

The divine Chinese dragon guarding the Sun in the universe is what we expected for the project's final work, and the rendering indeed reached a similar effect. After the main theme is settled down, the work is divided into modeling different parts, finding texture, and implementing animation for each group member. The rendering quality turns out to be better than we expected as we continue to explore OPENJSCAD, THREE.JS, and shaders.

2 Approach

2.1 Model

The dragon model is divided into four parts: head, body section, feet, and tail.

The dragon head contains many rounded cube, sphere, cube, and cylinder. Cutting, moving, rotating, piecing is applied to different extent. Because of the mostly irregular shape of dragon head, no formula is involved in head model. The fundamention of the head is the brain, two long mouth and horn. Horn is implemented by cylinders and sphere, and take mirrored to get another. Brain and mouth are basically achieved by cube and rounded cube, some cutting is applied on them. After accomplishing the basic shapes, detail, hair, is also added to the model. Hair use cylinder and solution parameter is set to 3 to be the triangular prism, adjusting each of prism to cover the brain, the head is finished.

The body model is created for only a section of an entire body because the following animation step requires a string of body sections to move along a stereo curve. Each body section consists of the union of two spiral pillars to mimic the dragon's scales and a line of sharp protrusions for the dragon's dorsal fins.

The center of the dragon's foot is a sphere, the body of the four claws and the "legs" extending to the body are composed of two or three composite cylinders, the joints between the bodies are the sphere, and the nails of the dragon's claws are six cones. The six pyramid is obtained from the deformation of the sphere with the surface number parameter set to 3. The composite cylinder of the paw body is composed of a relatively thin cylinder and a thicker cylinder with a distance set outside the cylinder at a certain distance, which can form a visual effect similar to the dragon scale

For the tail, it is the combination of six curved tail bones plus the spiral extension from the body. A function was written for the tailbone growth where a tailbone part model could be built by specifying the start point, height, and direction. With this function, A tailbone part could continue to grow at the end of the previous tailbone part to form one curved tailbone. After adjusting the position, length, and direction of each bone part into six tailbones, combining them with the body results in the final appearance of the dragon tail.

2.2 Texture

After modeling the Chinese dragon's head, body, claw, and tail, the texture would be applied to them. The model would be purely black if only the .stl file were loaded, and for more photo-realistic scenes being animated, each part of our Chinese dragon would have a different texture to be applied. Since the .stl file generated by OpenJSCAD does not map the vertex and textures' uv coordinate, the difference between pixel and pixel cannot show well, texture is implemented by fragment shader to demonstrate triangles unit pictures.

$$\vec{finalColor} = ambient * kA + diffuse * kD + specular * kS \tag{1}$$

For obviously observing the position of the dragon in the scene, BRDF (Bidirectional Reflectance Distribution Function) is applied on the fragment shader. Calculating ambient, specular, and diffuse light separately and finally summing them up as the shader output, without using any parameter passed by the vertex shader, the fragment shader can be done finely.

2.3 Screen Design

To create a cosmic background, we adopted the bounding box instead of commonly-seen HDR. This assists us in utilizing some magnificent shaders in ShaderFrog or other existing GLSL-type beauty. Detailedly, this is achieved by a cubic geometry which enables inner face rendering. A star-filed shader is imposed on it, which is purely mathematically designed and equation-formulated for both nebula at the base and shining spots by noise generation. At the center, we placed a moon-like or sun-like celestial body, which switches between a flowing mysterious purple fluid and a halo sun. This is also achieved by mathematical formulating rather than texture mapping. Additionally, hundreds of interstellar dashes are simulated by a deformable sphere, which is achieved by a varying vertex shader. They all have random positions, random orientations, random sizes, and random colors, updated by every web refresh.

2.4 Animation

To animate the objects, we designed two types of motion, where the first is for the camera and the second is for the dragon. We have considered a stereo curve comprised of some sin and cos functions, along which each section of the dragon body can move and orientate itself to be tangential to this curve. In

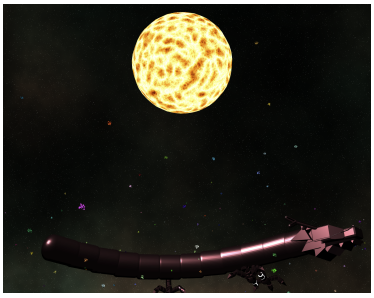

Figure 1: Scene with star-filed background, cosmic dust, and central star, and texture

general, it can give a sense of wavy soar, for such a creature living in mythology. A graphic illustration is given in Fig.2.

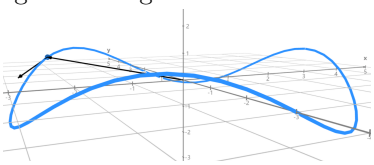

Figure 2: Parameterized path for dragon motion

For the camera motion, it can be as complex as an orbit and VOF variation, or simpler in our project as a circular motion. Such basic design, nonetheless, can meet a problem in our case. Because the star field shader adopts surface UV to generate noise, if the bounding box moves too fast relative to the camera, such a noise would become true noise in the background, but not smoothly shining spots without violent spatial movement. To tackle this issue, we designed to also adjust the bonding box position and rotation so that it is always static to the camera. This is also achieved by some math calculation, denoting p_c as the camera position, P_0 as the point the camera is looking at, l as their distance, and l_0 as the reference distance. θ_y is rotation around y axis and θ_z is rotation around z axis. The equations below describe all coding details.

$$\vec{p} = \frac{l - l_0}{l}\vec{p_c},\ \theta_y = -tan^{-1}(\frac{|p_{c,z} - p_{0,z}|}{|p_{c,x} - p_{0,x}|}),\ \theta_y = -tan^{-1}(\frac{|p_{c,y} - p_{0,y}|}{|p_{c,xz} - p_{0,xz}|}) \quad (2)$$

3 Result

Given all above processes, we can ultimately achieve a 3D Chinese dragon animation scene through three.js. Detailed demonstrations are provided in the below link:

https://furkathertaha.github.io/posts/threeJS-sf/