CS 557 Computer Graphics Shaders

Project #6

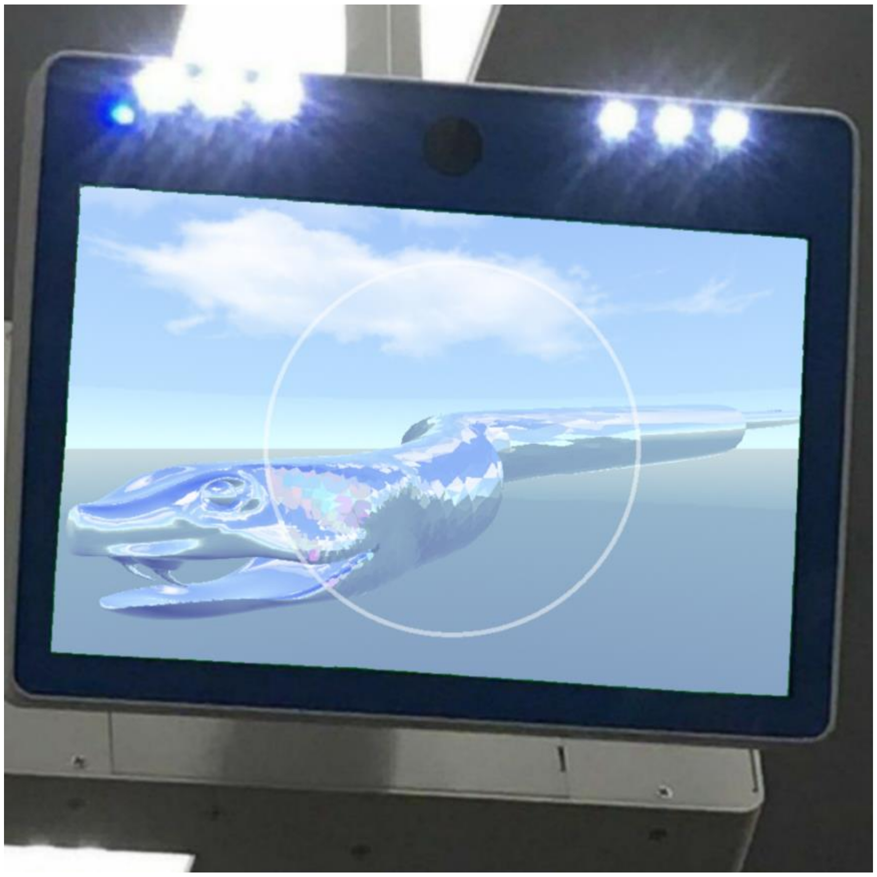
The Snake Menagerie Project

The Magic Lens on a Digital Screen: A Sidewinding Snake with Metallic and Diamond Textures

Name: Rachel Xing (#934229189)

Email Address: xingru@oregonstate.edu

Video Link: https://youtu.be/88Y_H_YXPBw



Description:

This project implements a shader that renders a sidewinding snake with a metallic texture in a blue sky scene using cube mapping. It features a magic lens that can transform the snake's texture to a diamond-like appearance within the lens, allowing users to switch between diamond and metallic textures. The entire view of the snake is displayed on a digital screen using a threepass rendering and image manipulation techniques. This project runs in glman using the "**shader.glib**" file.

The cubemap for the blue sky is retrieved from opengameart.org and loaded into the texture unit by the "**texture.vert**" and "**texture.frag**" shaders.

The cube mapping of the snake model into the blue sky and rendering it with metallic texture is implemented by the "**metal.vert**" and "**metal.frag**" shaders. The sidewinding transformation of the snake is generated in the vertex shader using a sine wave equation, with the x-coordinate of each point contracted to preserve the model's original length. In the fragment shader, the reflection is computed to simulate the metallic texture to the model's surface. In addition, the bump mapping technique, which uses a noise texture generated by glman to perturb the normal vectors and create crinkles on the model's surface, is applied to add more texture details. The following parameters can be adjusted for this part of rendering:

Parameter	**Description**
uAmp	Adjust the amplitude of the snake's sidewinding transformation.
uFreq	Adjust the frequency of the snake's sidewinding transformation.
uNoiseAmp	Control the amplitude of noisy crinkles on the model's surface.
uNoiseFreq	Control the frequency of noisy crinkles on the model's surface.

The cube mapping of the snake model into the blue sky and rendering it with diamond texture are implemented by the "**diamond.vert**" and "**diamond.frag**" shaders, and there is a file "**diamond.glib**" for this that can be run in glman. The vertex shader is similar to **"metal.vert"**, but the key difference is that it outputs all variables using "flat out", applying flat shading to create a faceted look that simulates the appearance of crystal or diamond. In the fragment shader, both reflection and refraction are computed to simulate the crystal or diamond texture on the model's surface. The crystal effect is achieved through reflection, while the diamond effect is created using refraction and dispersion. The dispersion is implemented through chromatic aberration, following the tutorial in this article written by Maxime Heckel. This method simulates real-world dispersion by assigning different refraction indices (eta) to each color channel. The following parameters can be adjusted for this part of rendering:

Parameter	**Description**
uAmp	Adjust the amplitude of the snake's sidewinding transformation.
uFreq	Adjust the frequency of the snake's sidewinding transformation.
uMix	Control the blending between crystal and diamond effects
uEtaRed	Control the index of refraction for the red light channel
uEtaBlue	Control the index of refraction for the blue light channel
uEtaGreen	Control the index of refraction for the green light channel

The cube mapping results of the metallic and diamond snakes are loaded into separate texture units and mapped onto a 2D quad using "**image.vert**" and "**image.frag**" for further image manipulation. In the fragment shader, a magic lens is created to render the portion of the snake inside the lens with a diamond/crystal texture while keeping the portion outside in a metallic texture. Within the magic lens, an image difference technique is applied, allowing users to adjust uTransit to change the snake's texture between diamond and metallic. Next, a digital green screen background is overlaid, and the chroma key technique is used to replace the green screen portion with the cube mapping results. Users can rotate the snake by selecting its model coordinates. However, since different vertex shaders are used for rendering the metallic (smoothing shading) and diamond snakes (flat shading), the view has two different model coordinate systems. The following parameters can be adjusted for this part of rendering:

Parameter	**Description**
uChromaKey	Adjust the chroma range for compositing the screen with the cube mapping results.
uTransit	Control the blending between diamond and metallic effects
uSc	Adjust the T center of the magic lens
uTc	Adjust the S center of the magic lens
uRad	Adjust the radius of the magic lens

Screenshots:

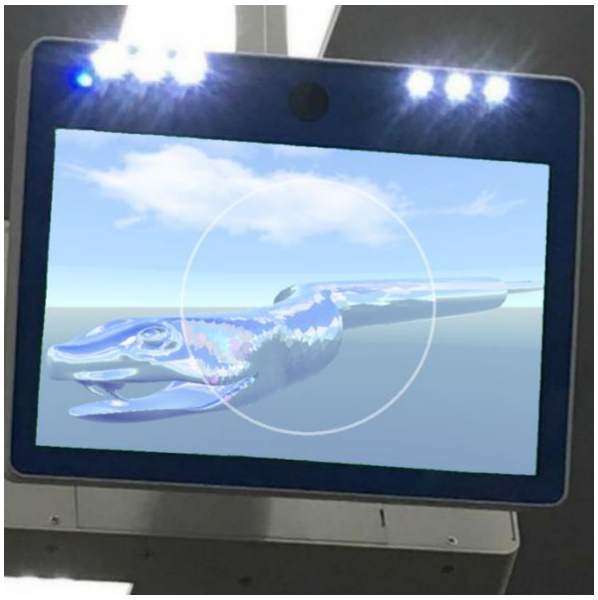


Figure 1. Default rendering of a snake with the magic lens effect within a digital screen. The portion of the snake within the lens is rendered with a **diamond effect**, while the portion outside the lens is rendered with a **metallic effect**.

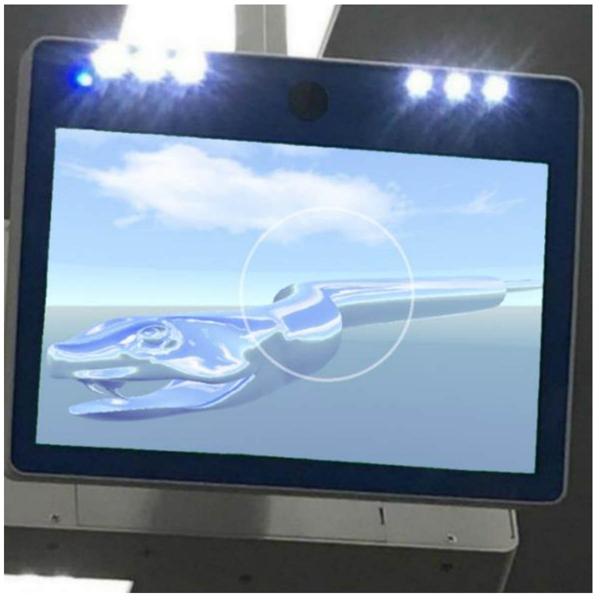


Figure 2. When **uTransit = 0.0**, the portion of the snake within the lens is rendered with the same metallic effect as the portion outside the lens.

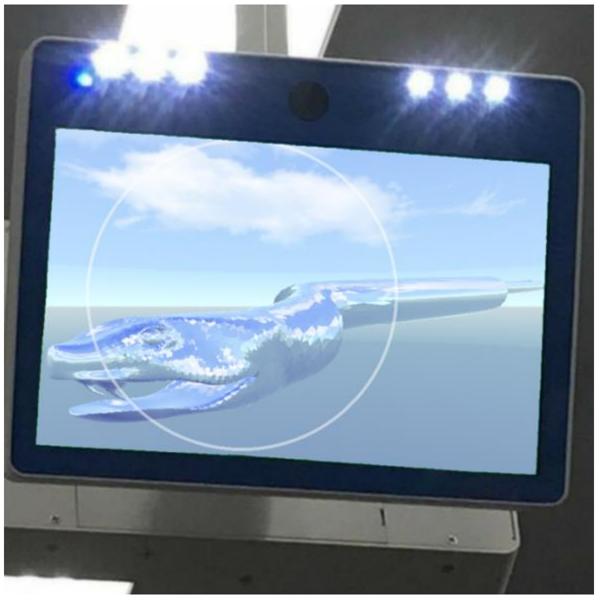


Figure 3. When **uMix = 1.0**, the portion of the snake within the lens is rendered with a crystal effect, where only reflections are counted.

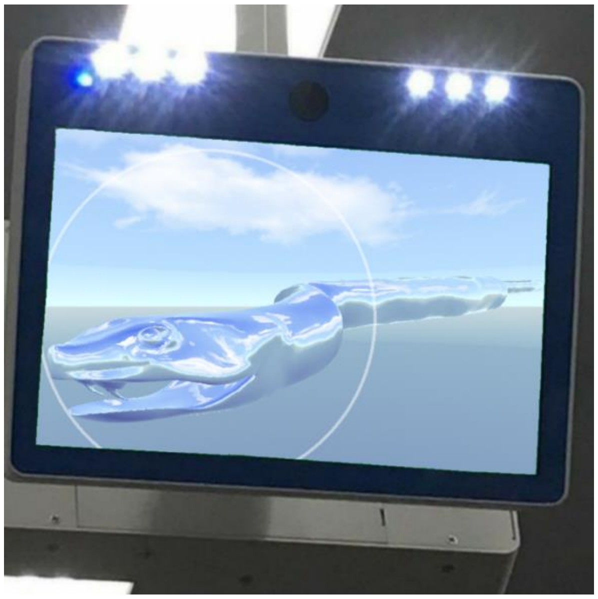


Figure 4. When **uNoiseAmp and uNoiseFreq increase**, the model surface visually develops crinkles due to bump mapping, and reflections on the metallic surface become distorted.

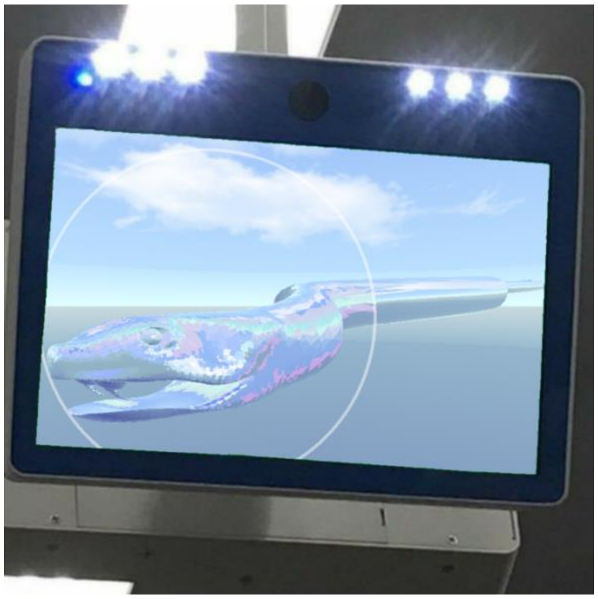


Figure 5. By **adjusting uEtaRed, uEtaBlue, and uEtaGreen**, you can achieve various chromatic aberration effects, which allows you to simulate different texture effects.

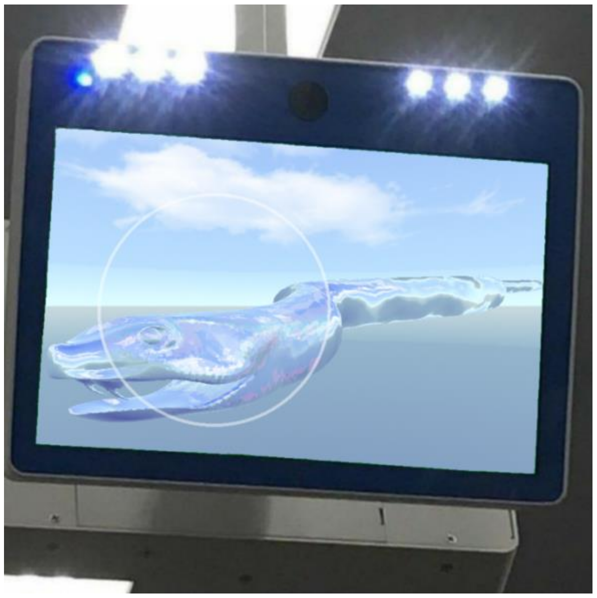


Figure 6. By **adjusting uTransit**, you can achieve more texture effects by blending the diamond/crystal and metallic effects together.