CS 557 Computer Graphics Shaders

Project #7

Geometry Shaders: Turning a Triangle Model into a Collection of 3D Crosses

Name: Rachel Xing (#934229189)

Email Address: xingru@oregonstate.edu

Video Link: https://youtu.be/k_fCK2yUn2c

Description:

This project implements a geometry shader that can transform the triangle mesh of an input model into a collection of 3D crosses with a fragment shader that can color the crosses based on its eye-coordinate depth using a rainbow color scale (**extra credit**). The input model used in this project is a 3D print dachshund dog model retrieved from the CGTrader website. This project runs in glman using the "**quantcrosses.glib**" file.

The vertex shader "**quantcrosses.vert**" only passes the normal vector and vertex position to the geometry shader without applying any transformations. Since the geometry shader generates new vertices and normals for the newly generated crosses, all necessary transformations are performed at that stage.

The geometry shader "**quantcrosses.geom**" receives a collection of triangles with normal vector and vertex position from the vertex shader stage. For each triangle, it applies the triangle subdivision algorithm. The **uLevel** parameter controls the number of subdivision levels, and the number of subdivided layers of triangles is 2^{uLevel}. For each generated (s, t) pair in each iteration of the algorithm, barycentric interpolation is applied to compute the vertex position of the center of the newly generated cross with its normal vector and other needed information, such as z depth in the eye coordinate, light vector, and eye vector. Then, the vector position of the center is translated along each axis in both the positive and negative directions to generate line strips for the newly created cross at the current position within the subdivided triangle. The translating distance from the cross's center, which defines the length of one side of the cross, is controlled by the parameter **uSize**. To adjust the precision of each newly generated cross's center position, a quantization function is applied for this 3-vector, with the quantization multiplier controlled by the **uQuantize** parameter.

In the fragment shader, per-fragment lighting is implemented for the converted collection of crosses for the model as in previous projects, with adjustable parameters (**uKa, uKd, uKs, uShininess, uLightX, uLightY, uLightZ**) that can be modified through the glman user

interface. In addition, the parameter **uColor** allows users to change the cross color using glman's color palette. Besides, the shader features the ability to color the crosses based on eye coordinate depth using a red-to-blue rainbow color scale, where red represents close to the eye, blue represents distancing from the eye, and green represents the middle range. This can be toggled with the checkbox **uUseChromaDepth** in glman, and the eye coordinate depths for red and blue are controlled by the parameters **uRedDepth** and **uBlueDepth**. Based on these preset depths for the two colors, the fragment shader interpolates the color for each fragment.

Screenshots:

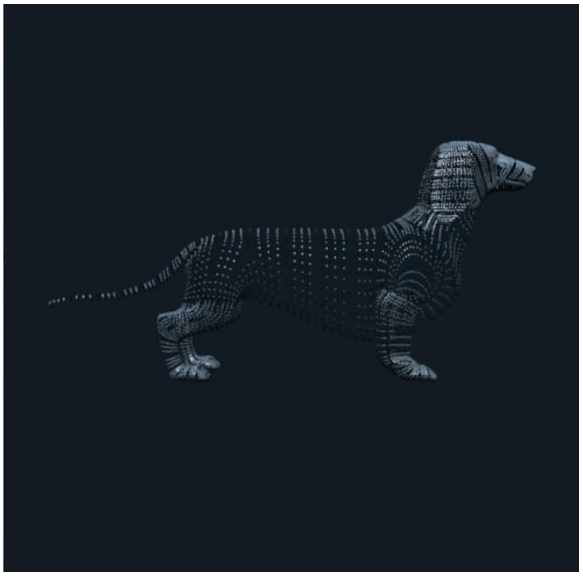

Figure 1. Converted collection of 3D crosses for the input dachshund dog model with **uLevel = 0 and large uQuantize**.

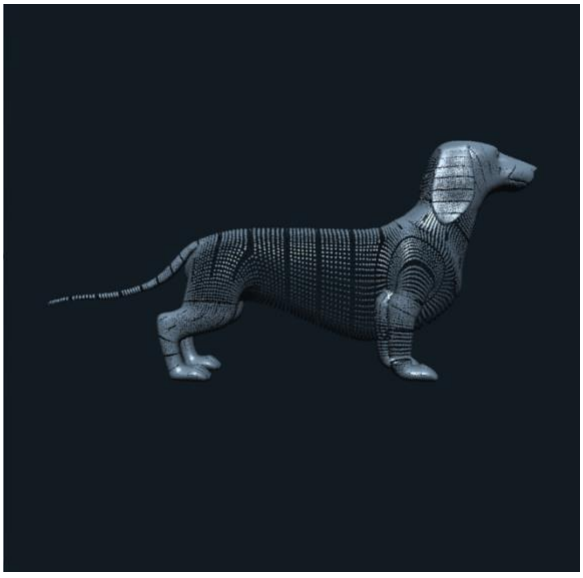

Figure 2. Converted collection of 3D crosses for the model with **uLevel = 1 and large uQuantize**.

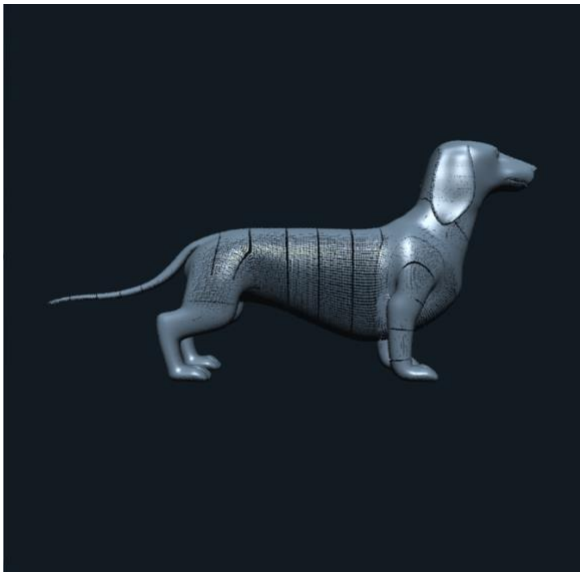

Figure 3. Converted collection of 3D crosses for the model with **uLevel = 2 and large uQuantize**.

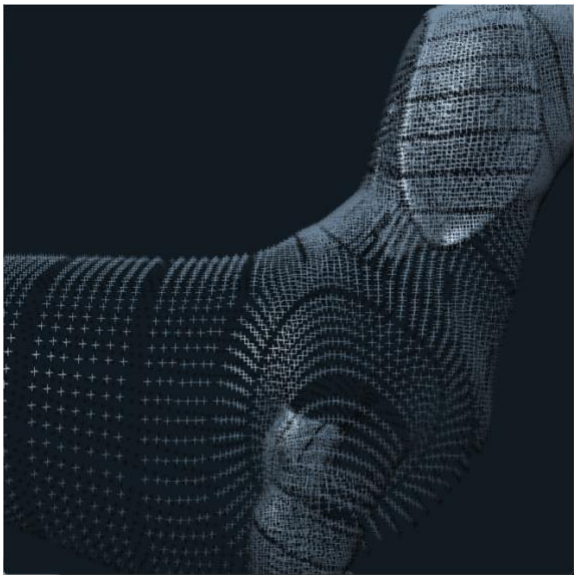

Figure 4. Zoom-in details for Figure 2 (**large uQuantize**).

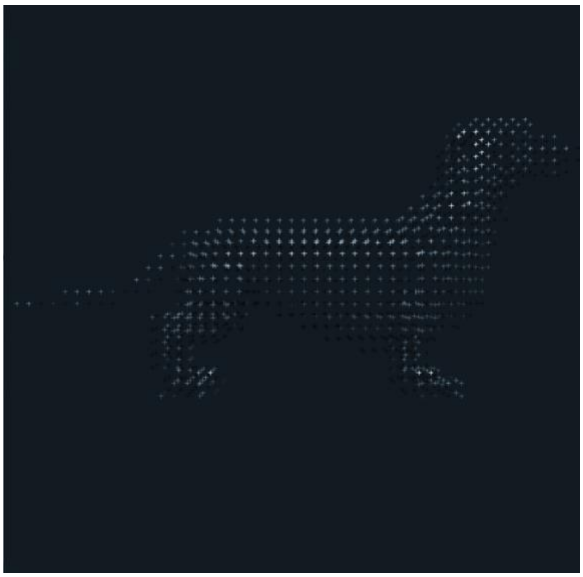

Figure 5. Converted collection of 3D crosses for the model with **uLevel = 1** and **small uQuantize**, where some crosses overlap due to the reduced precision of their positions.

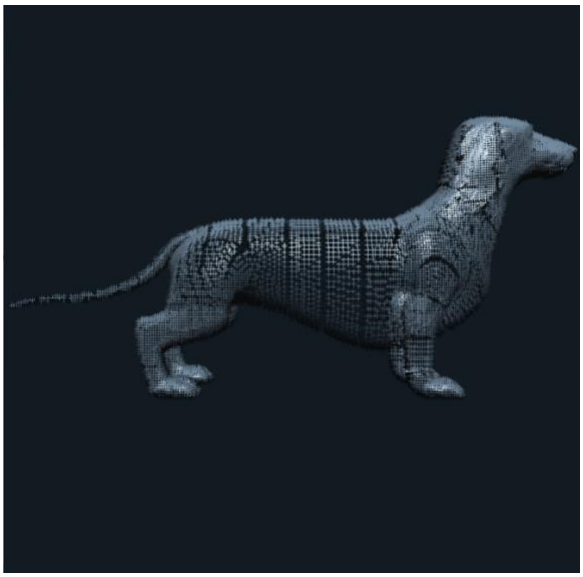

Figure 6. Converted collection of 3D crosses for the model with **uLevel = 1** and **moderate uQuantize**, where crosses are less evenly distributed compared to Figure 2.

Figure 7. Zoom-in details for Figure 6.

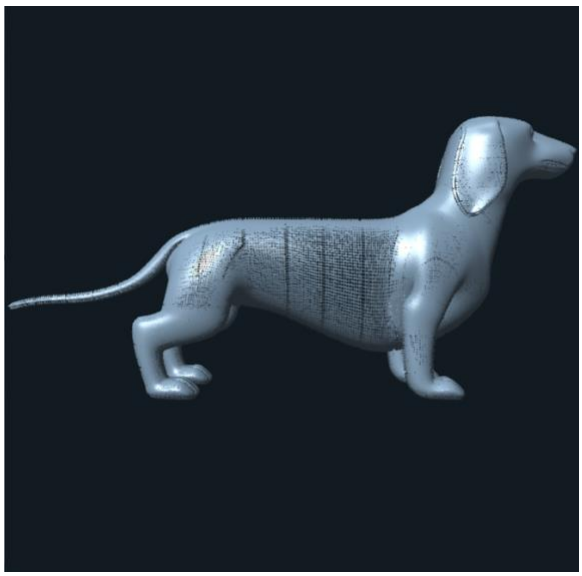

Figure 8. Evidence of functional per-fragment lighting by adjusting the **light position** and increasing **uKA**.

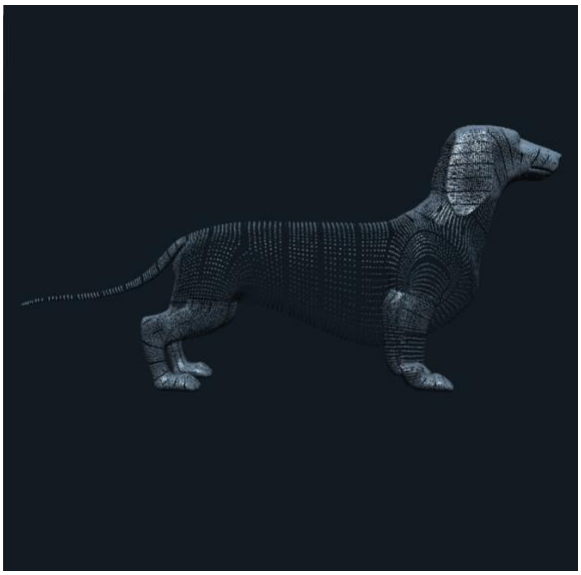

Figure 9. Converted collection of 3D crosses for the model with small **uSize**.

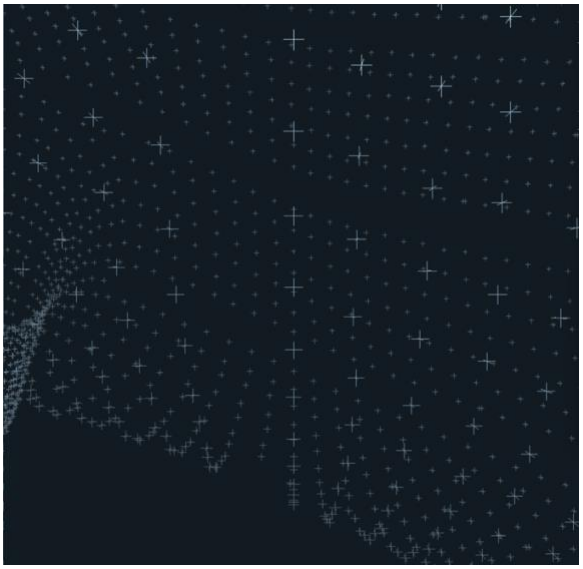

Figure 10. Zoom-in details for Figure 9.

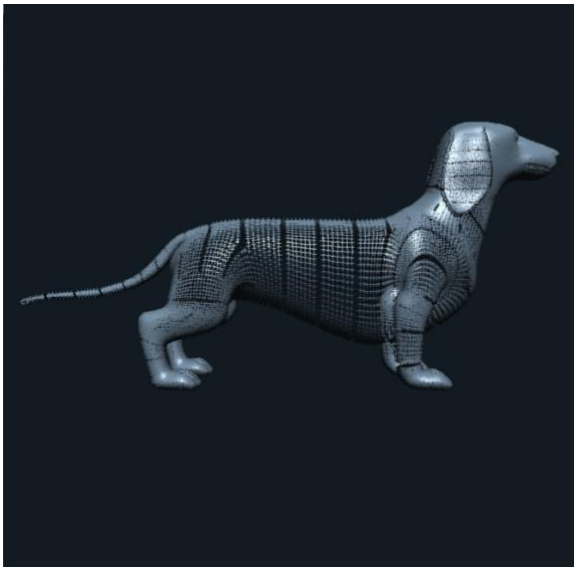

Figure 11. Converted collection of 3D crosses for the model with large **uSize**.

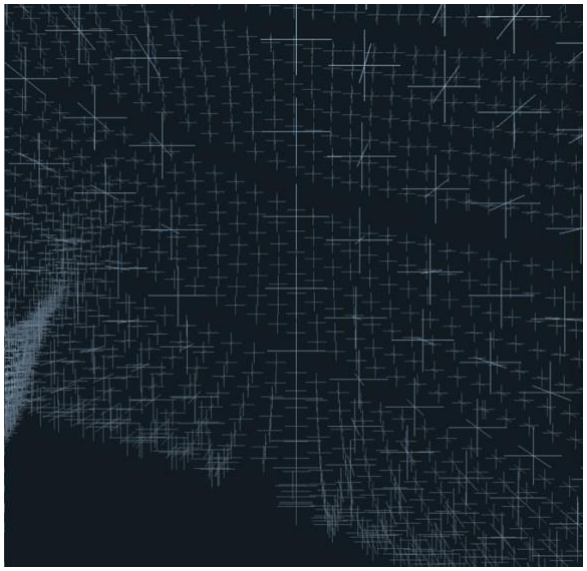

Figure 12. Zoom-in details for Figure 12.

(Extra Credits)

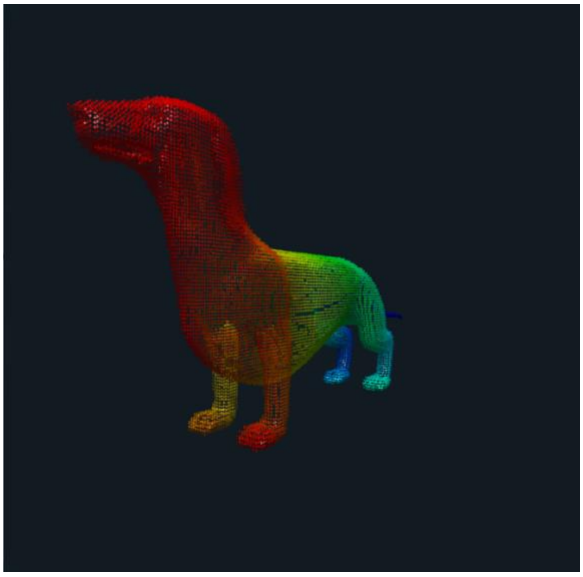

Figure 13. When **uUseChromaDepth** is enabled, the crosses at the front of the model are colored red, those at the middle are colored green, and those at the back are colored blue in eye coordinates.

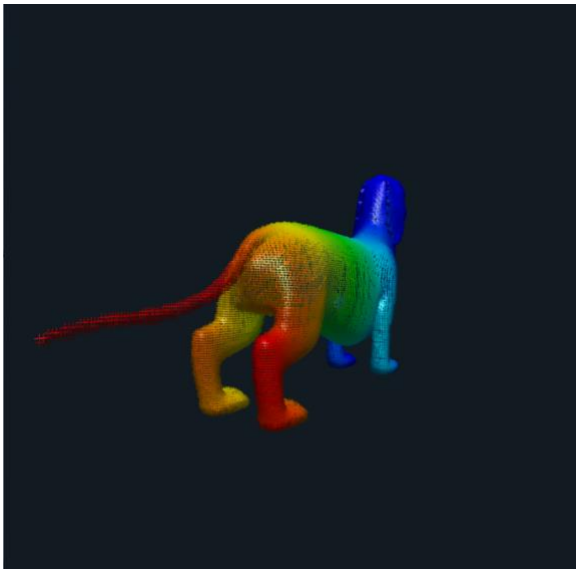

Figure 14. Evidence of red-to-blue color pattern is always in **eye coordinates** when rotating the model.