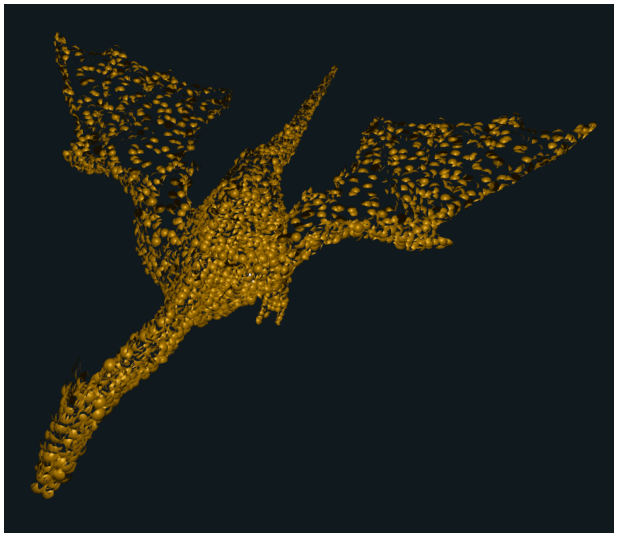
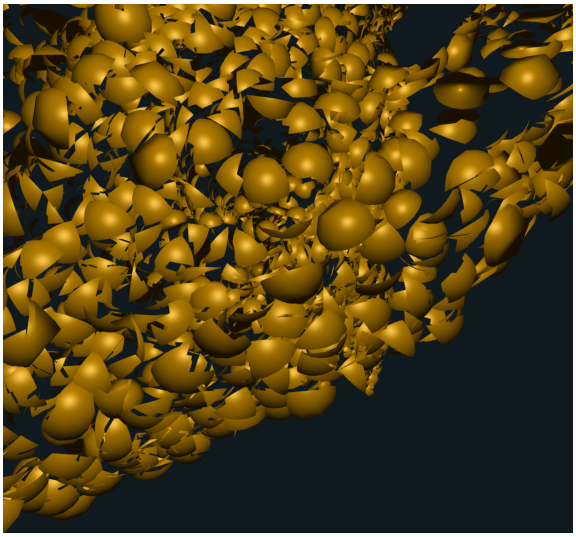
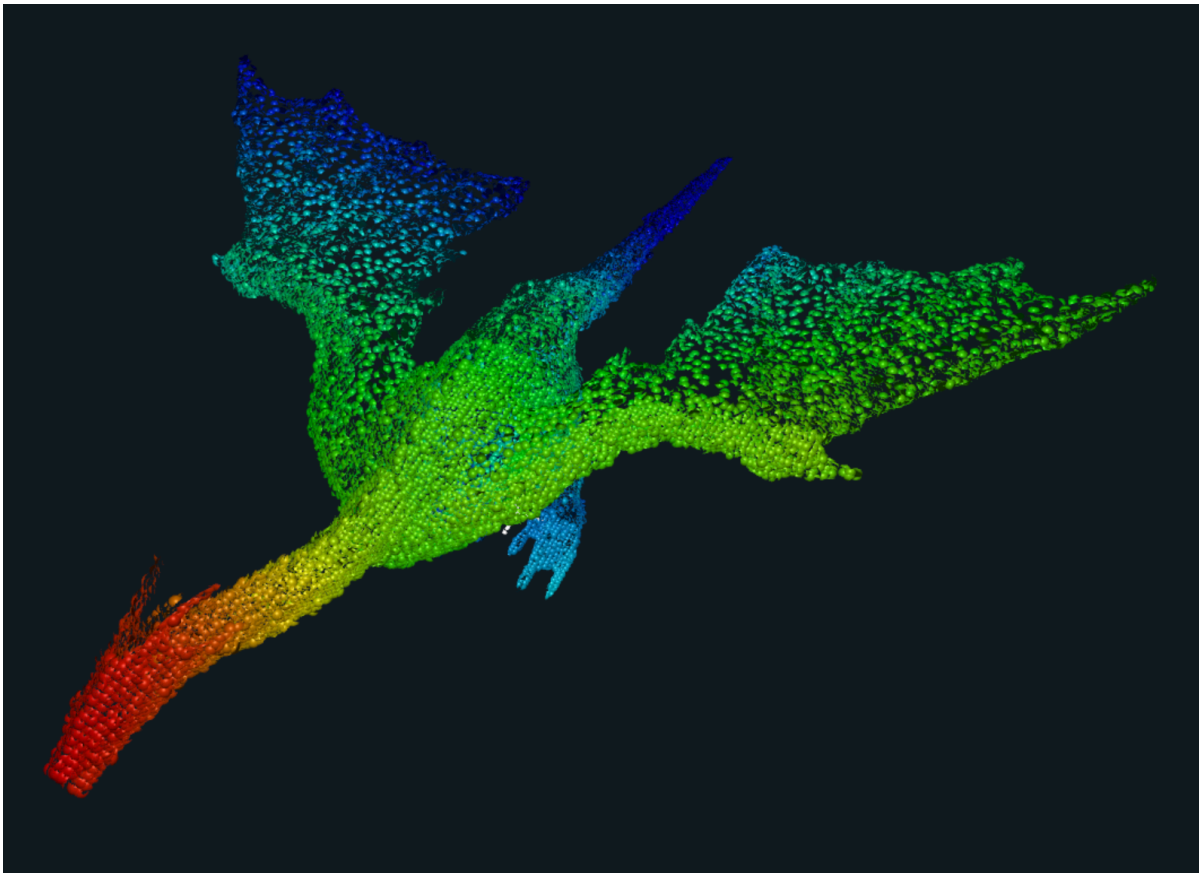
**CS 457 Project #7**

**Geometry Shaders: Turning a Polygonal Model into a Collection of Spheres**

Annette Tongsak
tongsaan@oregonstate.edu







Video link

To create this display, I:

**quantsph.vert**

1. Set gl_Position to gl_Vertex since we're doing matrix multiplication in the geometry shader

**proj7.glib**

1. Implemented calls to quantsph.vert, quantsph.geom, and quantsph.frag
2. Set parameters for the following variables so they appear as sliders in GLman:
    a. uLevel — how many levels to subdivide the triangle
    b. uDiam — diameter of the spheres
    c. uQuantize — quantization multiplier
    d. uKa — ambient lighting
    e. uKd — diffuse reflection
    f. uKs — specular reflection
    g. uShininess — specular exponent
    h. uLightX — x-coordinate of light position
    i. uLightY — y-coordinate of light position
    j. uLightZ — z-coordinate of light position
    k. uColor — object color
    l. uSpecularColor — color of light used for specular reflection
    m. uRedDepth — Z depth in eye coordinates at which ChromaDepth starts for red
    n. uBlueDepth — Z depth in eye coordinates at which ChromaDepth starts for blue
    o. uUseChromaDepth — boolean to toggle ChromaDepth
3. Specified the dragon OBJ

**quantsph.geom**

1. Brought in GLman variables as uniform int/floats
    a. uLevel
    b. uQuantize
    c. uDiam
    d. uLightX
    e. uLightY
    f. uLightZ
2. Specified output variables:
    a. gN — normal vector
    b. gL — vector from point to light
    c. gE — vector from point to eye
    d. gZ — eye coordinate depth of a vertex
3. Specified variables:
    a. V0, V1, V2 — 3 vertices of the original triangle
    b. V01, V02 — differences between the vertices
        i. V01 = V1 - V0

        ii. V02 = V2 - V0
    c. CG centroid of the original triangle
    d. LIGHTPOS light position utilizing uLightX, uLightY, uLightZ
4. Brought in Quantize() function
    a. Used to quantize a single float
5. ProduceVertex() function
    a. Calculate the interpolated vertex coordinates using (s,t)
    b. Translate the vertex coordinates relative to the centroid (CG)
    c. Scale the vertex coordinates to create a spherical shape
    d. Translate the vertex coordinates back to the global space
    e. Transform the vertex to eye coordinates by multiplying by the model-view matrix
    f. Calculate normal vector in eye coordinates
    g. Set output variables for use in fragment shader
    h. Transform the vertex to clip coordinates and emit it
6. main()
    a. Define variables V0, V1, V2, V01, V02, and CG
    b. Determine how many layers are needed based on the level specified by uLevel
    c. Calculate parameters for the current layer
        i. dt step size for subdivision of layers
        ii. t_top top of the layer
        iii. t_bot bottom of the layer
        iv. smax_top max value for s coordinate at top of layer
        v. smax_bot max value for s coordinate at bottom of layer
    d. Calculate subdivision
        i. Determine the number of vertices (nums) needed for the current layer
        ii. Calculate step size for s coordinate at top and bottom of the layer (ds_top and ds_bot)
    e. Produce vertices by calling ProduceVertex() function in a nested loop
        i. Create a horizontal strip of vertices within the current layer
    f. Close layer
    g. Update parameters for next layer by shifting t_top and t_bot

**quantsph.frag**

1. Brought in GLman variables as uniform int/floats
    a. uKa
    b. uKd
    c. uKs
    d. uColor
    e. uShininess
    f. uSpecularColor
    g. uLightX
    h. uLightY
    i. uLightZ
    j. uRedDepth

    k. uBlueDepth
    l. uUseChromaDepth
2. Specified input variables:
    a. gN
    b. gL
    c. gE
    d. gZ
3. Brought in Rainbow() function
    a. Color mapping function that take a single input parameter t and returns corresponding RGB color value
4. main()
    a. Normalize input vectors
    b. Determine colors
        i. If uUseChromaDepth is enabled, calculate a color based on the depth gZ using the Rainbow() function
    c. Do per-fragment lighting