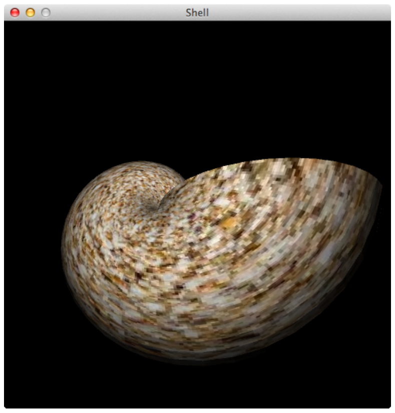
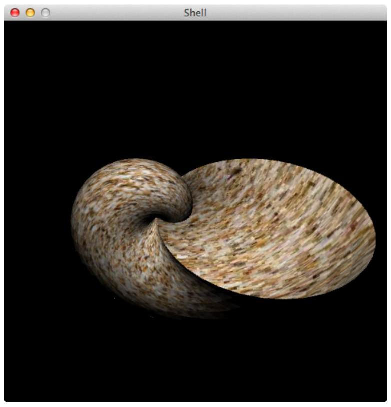
Final Exam (Begin Thursday 5/13; Due Saturday 5/15 by 11:59pm; 100 pts total)
CSCI 3353 - Hibbs
<u>Instructions:</u> There are 4 coding questions for this exam. Each question asks you to create a Processing sketch to achieve the goal specified. For each question, you should create a sketch/archive with the following naming convention. For each question, please name *BOTH* the sketch *AND* the archive with this convention: **final_q#_username.pde/.zip.** When done, bundle your sketch archives together into a single zip file named **final_username.zip**, upload this file to google docs, and share it with *only* me (mhibbs@trinity.edu).

<u>Q1 (20 pts)</u> - Create a sketch that displays a flat shaded, texture mapped 3D shell/cornucopia shape, similar to that in the images below. You must include the ability to rotate the scene based on mouse movements (this does NOT need to be a full glass ball interface). The 3D parametric equations for this shell shape are as follows, given 2 parameters *t* (in the range 0 to pi) and *s* (in the range 0 to 2pi):

$$x = (\frac{4}{3})^s (\sin t)^2 \cos s$$
$$y = (\frac{4}{3})^s (\sin t)^2 \sin s$$
$$z = (\frac{4}{3})^s \sin t \cos t$$

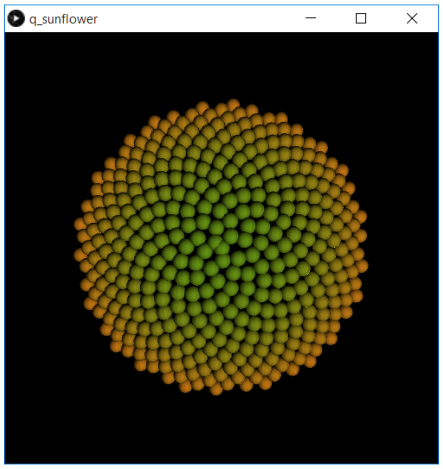


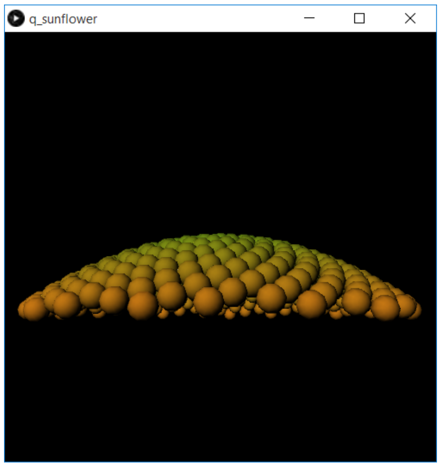


<u>Q2 (20 pts)</u> - Create a 3D sketch displaying spheres arranged in the form of the seeds of a sunflower. Sunflower seed arrangement is based on a variation of Fermat's spiral, called the Vogel spiral. The formula for the Vogel spiral in 2D ***polar coordinates (r,θ)*** is the following:

$(r = \sqrt{n},\ \theta = 2\pi\varphi n)$, where φ is a constant related to the golden ratio: $\varphi = \frac{1+\sqrt{5}}{2}$

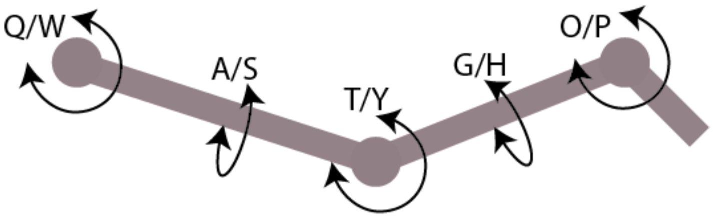
and *n* is an integer ranging from 0 to the total number of seeds. Following these formulas will create a "flat" spiral pattern. For full credit, your seed spheres should also form a dome shape, similar to a real sunflower. Your seed colors should transition between 2 colors as the indexes *n* vary from 0 to the number of seeds. You must also include the ability to rotate the scene based on mouse movements (this does NOT need to be a full glass ball interface).



← mouse movement →


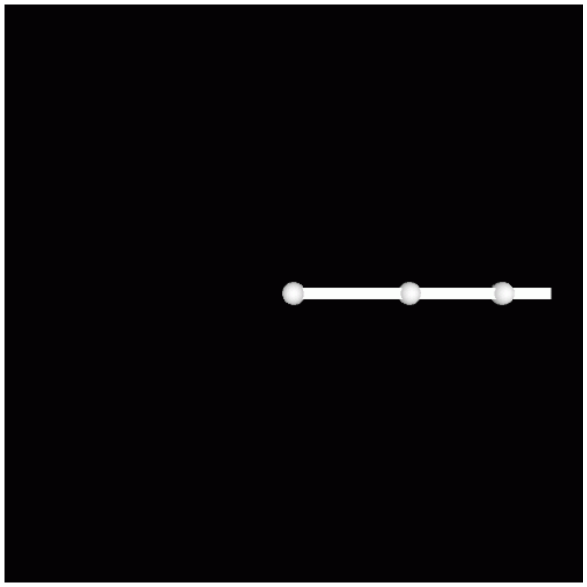
<u>Q3 (30 pts)</u> - Create a sketch that displays an animated, 3D particle system that uses a GLSL POINT shader to draw "billboarded" shapes or textures as the particles. This question is more open ended in that you may choose the structure, shape, paths, and other details of your "billboard" particle system. In this case, "animated" just means that the particles should be in motion similar to the systems created in class. Except in this case, the particles should be 3D points, that are rendered as more complex objects by a GLSL point shader. Intersections, collisions, or boundaries are not required.

<u>Q4 (30 pts)</u> - Create a sketch containing an interactive, articulated robot arm. The robot arm should be constructed of a shoulder joint, an upper arm, an elbow joint, a lower arm, a wrist joint, and a hand. Each joint should be displayed as a sphere, and each arm/hand should be shown as a box/quadrilateral. Subsequent arms/hands should be shorter along the path from shoulder to hand. Keyboard keys should change the rotational angles of the joints as specified:



Specifically, Q/W should change the shoulder joint's rotation, while A/S should change the shoulder's "twist" amount. This distinction is drawn between an angle determining where the upper arm extends out from the shoulder (Q/W value) versus the rotational orientation of the upper arm from the shoulder (A/S value).

Specifically, Q/W rotations affect the robot arm further down from the shoulder within the plane of the upper arm; while A/S rotates the lower arm and hand around the axis of the upper arm. Think of Q/W as moving your arm up/down with the arm in the same plane as your body; while A/S rotates your arm around an axis from the shoulder towards the elbow. A similar relationship should exist for the elbow, where T/Y are used to change the angle between the upper arm and lower arm; while G/H are used to twist the lower arm with respect to the upper arm (similar to rotating your wrist along the axis of your lower arm). Lastly, the O/P keys should bend the hand up/down in relation to the wrist. (There is no need to have a "twist" at the wrist joint). See the following example:



Full credit requires multiple key presses to occur simultaneously (consider using a HashSet of characters in conjunction with the keyPressed() and keyReleased() functions). You must also include the ability to rotate the scene based on mouse movements (this does NOT need to be a full glass ball interface).

HINT: Don't overthink this. Keeping track of angles and using the built in box() and sphere() primitives is entirely possible to solve this question, and limiting your code to these may restrict your thinking to a simpler implementation. Remember the modelview matrix and how it is used throughout a scene -- it can be changed between individual pieces of geometry. A solution of less than 80 lines of code is entirely possible.