B481 / Fall 2022 – Homework 01

Zach Graber (zegraber)

September 9, 2022

1.

(a) **Q:** Given two arbitrary temperature scales, A and B, with freezing and boiling points of A_f, A_b, B_f, and B_b, what are the conversion formulas for $A \longrightarrow B$ and $B \longrightarrow A$?

A: We first compute the normalized "distance" of the temperature (t_A) between A_b and A_f (in other words, how far between boiling and freezing the temperature is): $t_n = \frac{t_A - A_f}{A_b - A_f}$. With that information, we can use standard linear interpolation; in this case, we are "interpolating" between the boiling and freezing points of scale B:
$t_B = B_b t_n + B_f(1 - t_n)$
$\implies$ to convert $A \longrightarrow B$: $t_B = B_b \cdot \dfrac{t_A - A_f}{A_b - A_f} + B_f \cdot \left(1 - \dfrac{t_A - A_f}{A_b - A_f}\right)$
We can easily sanity-check this result by noticing that when $t_A = A_b$, $t_B = B_b$, and when $t_A = A_f$, $t_B = B_f$.

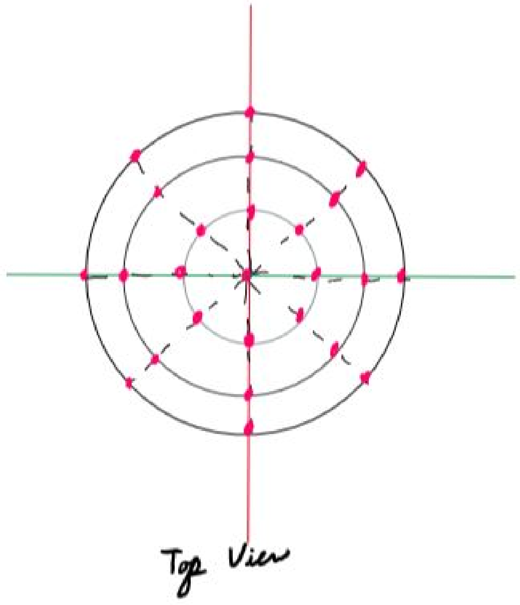
To convert in the opposite direction, just switch things out a bit:
$B \longrightarrow A$: $t_A = A_b \cdot \dfrac{t_B - B_f}{B_b - B_f} + A_f \cdot \left(1 - \dfrac{t_B - B_f}{B_b - B_f}\right)$

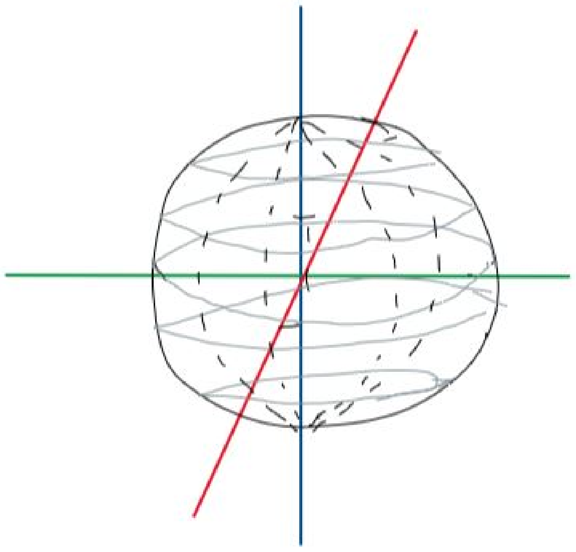
(b) **Q:** Write down the functions $F(t)$ and $C(t)$ from freezing to boiling temperatures for Fahrenheit and Celsius scales; t should range from 0 to 1.

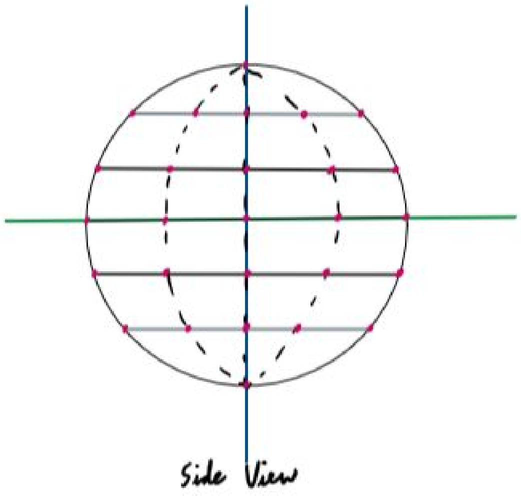
A: Since t is already normalized, we just plug into the generalized formula from the first part of the problem:
$F(t) = 212t + 32(1 - t) = 180t + 32$
$C(t) = 100t + 0(1 - t) = 100t$

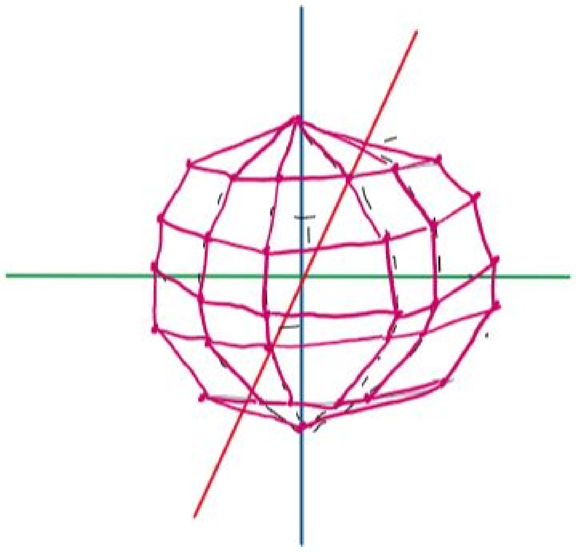
2. **Q:** In computer graphics, objects such as spheres are usually approximated by simpler objects constructed from flat polygons (polyhedra). Using lines of longitude and latitude, define a set of simple polygons that approximates a sphere centered at the origin. Can you use only quadrilaterals or only triangles?

A: First, let's examine the simplest way to approximate a sphere: draw some lines of longitude and latitude and draw a vertex at every intersection. This would yield a surface similar to the one pictured below. In fact, you might find that no matter the number of lines of longitude, closing the sphere with a single point will always yield triangles.

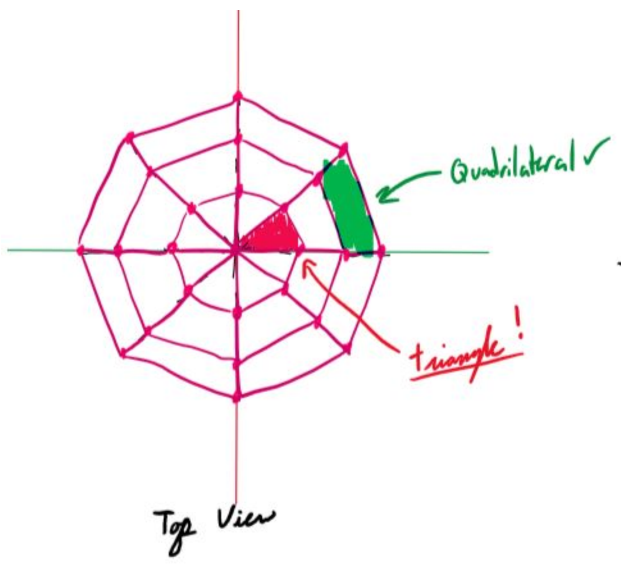


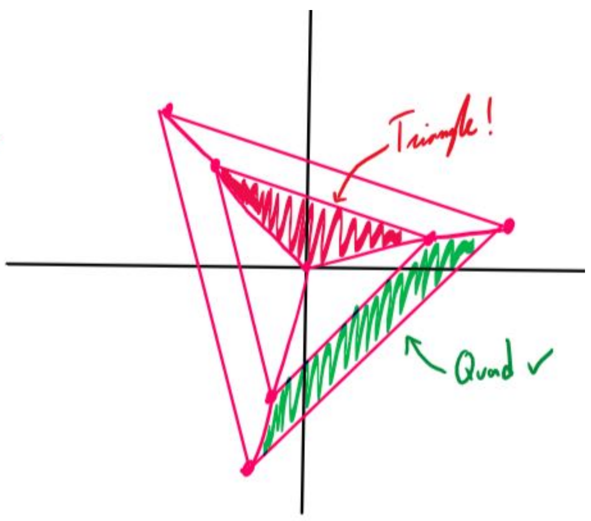




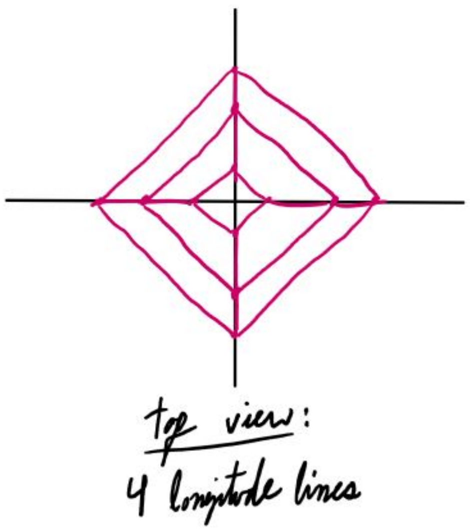


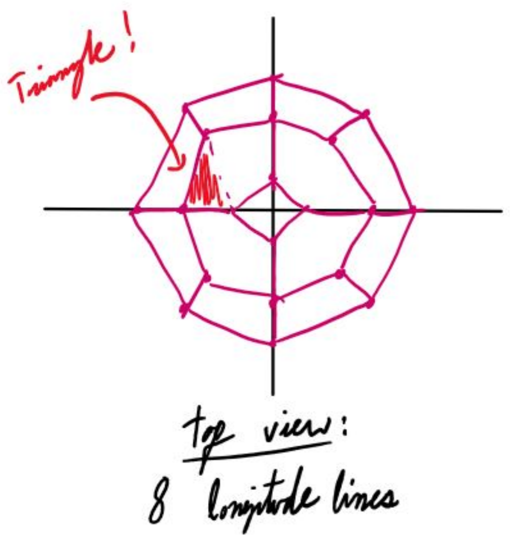
This surface is a mix of quadrilaterals and triangles. Let's seek to answer the two other questions presented. First, can you approximate a sphere with **only quadrilaterals**? The shape we already have won't work, since as the sphere closes at the top/bottom, the outermost ring of vertices must converge into a single point, thereby creating triangles, as illustrated below.





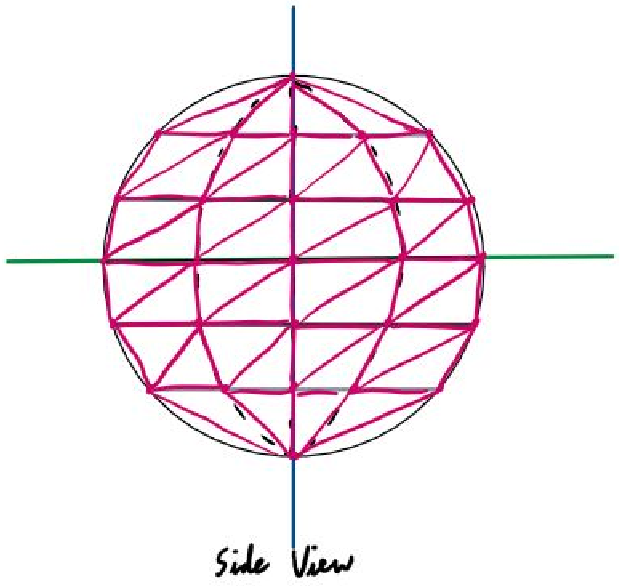
What if, instead of closing the sphere with a single point on top, we closed it with a polygon? Naturally, if we want the sphere to be made of only quads, the primitive on top would necessarily be a quad, too.





As it turns out, though, this approach *only* works for a “sphere” with 4 lines of longitude. As soon as you add more, it becomes impossible to join them without needing to form a triangle. So technically speaking, yes, you *can* approximate a sphere using only quadrilaterals, but it’s a pretty lousy excuse for a sphere.

What about using **only triangles**? This is much easier. Simply take the surface created in the beginning of this exercise (by placing vertices at every intersection of a line of longitude with a line of latitude) and divide each of its quadrilaterals into two triangles.



This solution hints at a larger truth, as well: any surface made of quadrilaterals can be described solely with triangles, since a quadrilateral is nothing but two triangles with two common vertices. Also interesting is that, between the first surface we described and this final one, the *geometry* (locations of the vertices) never changed–only the topology did.

3.

Q: In the OpenGL ES 3.0 pipeline, as shown at Lecture 02, there's a stage that performs Primitive Assembly operations. Explain in your own words:

(a) What is the stage that precedes Primitive Assembly, and what is the stage that follows Primitive Assembly?

(b) What are the primitives that get assembled in this stage? Describe at least two different primitives that may need getting assembled in this stage of the pipeline.

A:

(a) Before Primitive Assembly comes vertex processing (the vertex shader step). In this step, objects' vertices are modeled from ideal/object coordinates into a world coordinate system, then transformed into an eye coordinate space, and finally projected into normalized device coordinates. The stage following Primitive Assembly is rasterization, in which infinitely precise primitives are sampled (rasterized) into finite-precision fragments.

(b) The primitives that get assembled in this step are exactly what the name implies: they are the most primitive shapes/surfaces possible. Things like line segments and polygons (such as triangles) essentially have their raw vertex information turned into a more robust, "real" description of a shape.