CSCI 385: Written Part of *Walk Thru It*

Due: TBD

Several parts of the *Flip Book* application rely on geometric calculations to perform ray casting and hidden line removal. Below we ask you to work out those calculations on paper so as to prepare for the coding of them in `walk-thru.js` and `walk-thru-library.js`. In doing so, I strongly encourage you to take a "coordinate-free" approach. This has you work to calculate with point and vector operations, rather than directly work with their coordinate data.

This is a "low impact" assignment to get you thinking about the project right away. Make an attempt and hand in what you figure out. I'll share solutions of these in lecture.

1. Suppose we are given a center point C for projection, and a direction t into the scene that we are projecting. Suppose also we are given an additional direction u that represents a general upward direction for orienting the projection. You can assume that the two vectors are not parallel.

   Give an orthonormal frame centered at C and with basis directions e_1, e_2, e_3. These correspond to the x, y, z directions for a coordinate system for specifying the projection. It should have the property that $u \cdot e_1 = 0$. In devising this, tell me whether your frame is left- or right-handed.

   Note: The directions e_1, e_2, e_3 correspond to the vectors' `right`, `up`, and `into` for the `SceneCamera` constructor in `walk-thru.js`. The point C corresponds to `center` and the directions t and u correspond to `towards` and `upward`.

2. Now imagine a plane $\mathcal{P}$ that contains the point $\mathcal{O} = C + e_3$ and has the normal e_3. Within that plane, we have a 2-D orthonormal frame with $\mathcal{O}$ as the origin and e_1 and e_2 as the basis vectors. Let's consider projecting elements of a 3-D scene onto that plane using perspective projection.

   To do so, consider a scene point Q where $(Q - C) \cdot e_3 > 0$. Compute the coordinates x, y that give the projection of that point onto the plane $\mathcal{P}$. So let Q be a point in the scene and let its projection onto $\mathcal{P}$ be the point Q'.

   This would mean that $Q' = \mathcal{O} + xe_1 + ye_2$.

   Note: For the project, this is what's needed by the method `SceneCamera.project`. We have a plane sitting one unit away from the camera at `center` shooting in the direction of `into`. We are given a scene point as `location`, the code's analogue to P. We find the location of the `projection` as a `Point2d` object, the code's analogue to P' with coordinates x and y. In doing this perspective calculation, you'll also compute the `depth` information, the distance along direction e_3 (along `into`) where P sits in 3-space.

3. Consider two non-parallel line segments $\overline{P_0 P_1}$ and $\overline{Q_0 Q_1}$ that sit in Euclidean 2-space. Assume they intesect, and not at their endpoints. Find their intersection point S. In doing so, you can assume for any vector v that $v^\perp$ is a vector perpendicular to v and whose direction is rotated 90 degrees (counter-clockwise) from the direction of v.

   What are the conditions that these two segments intersect in this way?

   For what scalar value s does $S = (1 - s)P_0 + sP_1$?

   Note: For the project this is needed to code up `segmentIntesect` in `walk-thru-library.js`. And the value of s is just the "breakpoint value" that is collected by `SceneEdge.breakpoints` to break a projected edge into segments that may or may not be drawn in the PDF.

4. Let ray r emanate from a point R in a direction so that it passes through a point R' in Euclidean 3-space. Let Q_1, Q_2, and Q_3 be the vertex locations of a trianglular facet $\mathcal{T}$. Suppose the ray r intersects $\mathcal{T}$. Find the point of intersection S.

   Thinking more generally, what are the conditions for which the ray r intersects $\mathcal{T}$?

   Note: You'll need to figure out these conditions to write the code for `rayFacetIntersect` that is used by the method `SceneEdge.isSegmentVisible`. You'll be shooting a ray from the camera's center of projection to some point along an edge in the scene. Those will be the code analogues to R and R'. And then you'll want to know whether a triangular facet of a scene object, given by its three vertex points, is hit by the ray at a closer distance than $||R' - R||$. If it is, then that point R' will be hidden by that facet, at least from that camera's perspective.