CS 450 Final Project

Building and Animating a Cobweb

Annette Tongsak

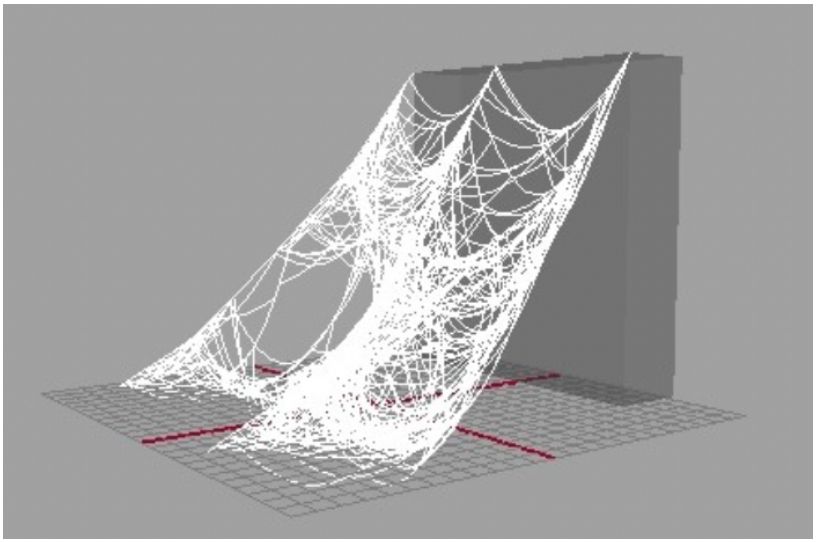
tongsaan@oregonstate.edu

Proposal

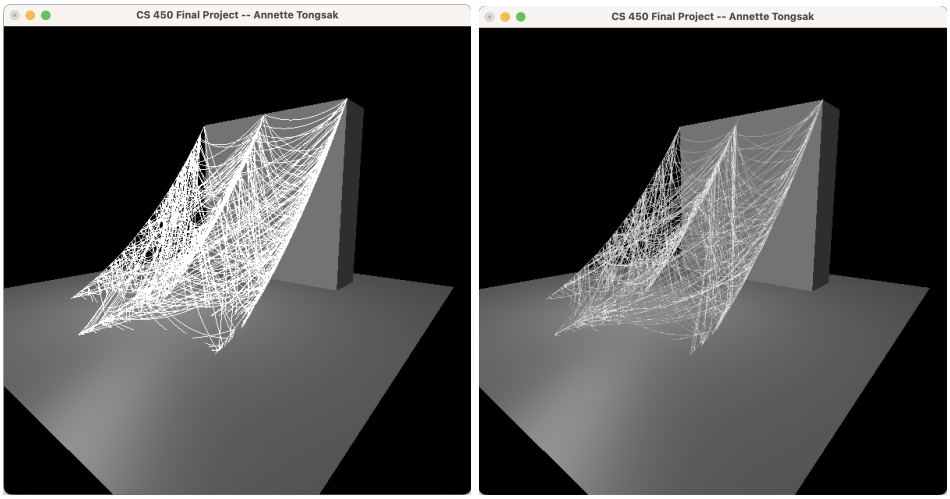
Inspired by Robert Russ's mention of Pixar's cobweb generation program, I want to propose the animation of a cobweb being generated in a scene as my final project. There will be other objects in the scene that the web is being built against, and I hope to involve shadows to make their silhouettes visible. I am also curious whether changing part of the cobweb's transparency will make it more realistic.

I plan to reference DreamWorks' 2011 paper, "Building and Animating Cobwebs for Antique Sets" as a guide to achieve similar results in OpenGL.

Here is a visual reference of what I am hoping to achieve for the cobweb:

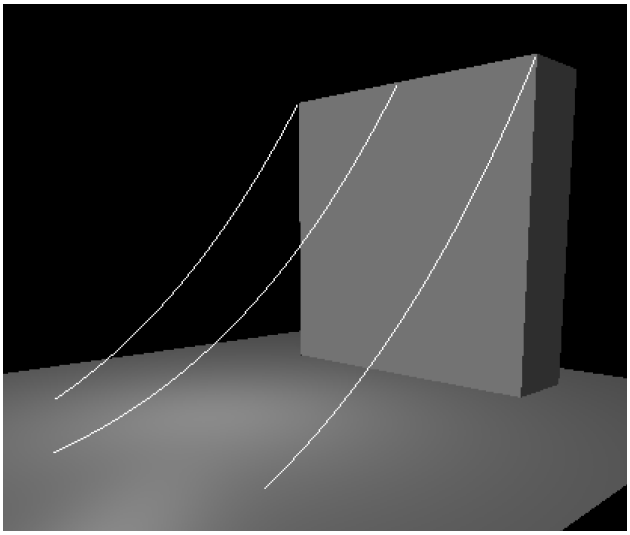


Project Execution - [Video link]



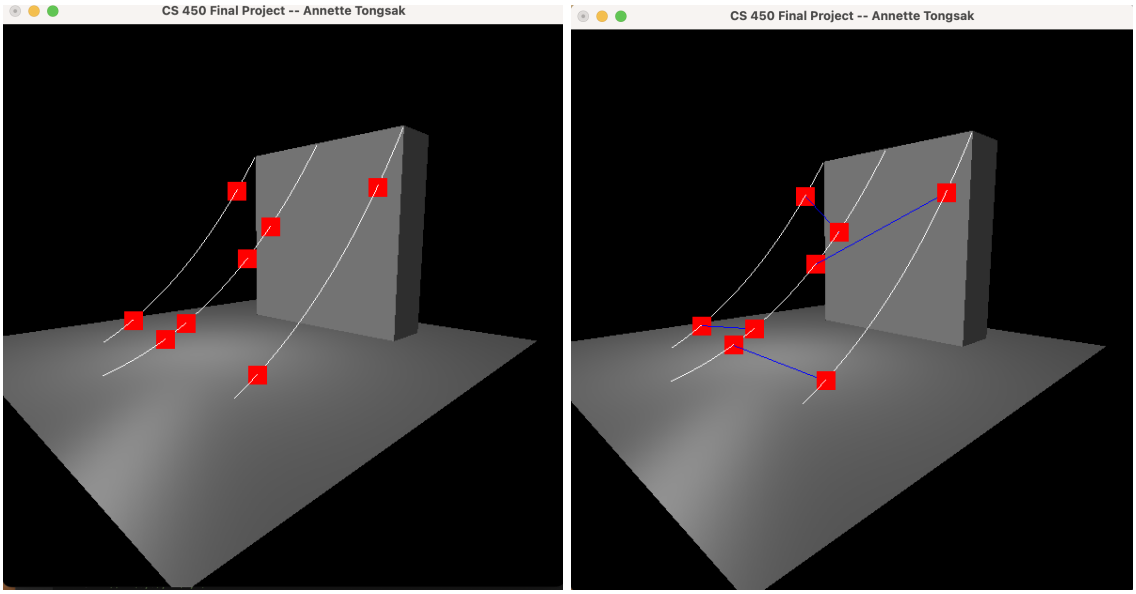
To create this display, I worked in this order:

1) Created the grid and wall
2) Added a point light
3) Drew base curves
    a) Referencing DreamWorks' catenary equation for approximating the shape of a hanging flexible chain supported at its ends, I created a function designed to generate such using GL_LINE_STRIP. This function incorporates parameters—x-axis, initial z-axis value, final z-axis value, and height—to allow for control over the shape and position of the curve.
    b) I subsequently drew three base curves attached to the wall using this function, with the middle curve a bit longer to imitate the realistic structure of a cobweb.

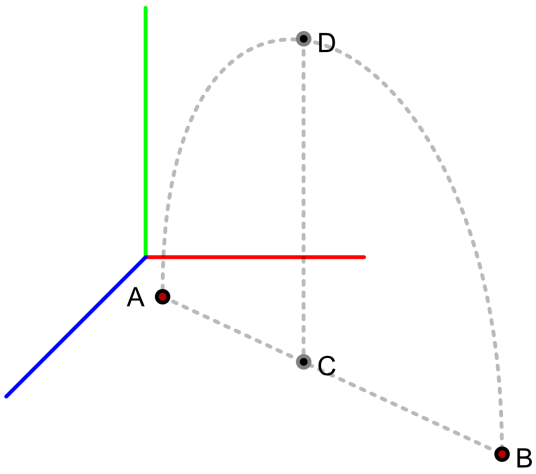


4) Drew connected base curves

a) Like in the DreamWorks paper, I added curves to connect the three base curves. This was done by first implementing and using a function to get a point on the catenary base curve that I wanted a curve to be attached to. I then tentatively added straight lines to connect the points.



b) To achieve a curved line between the base curves, I referenced the parametric form equations for a parabola in 3D space that I found here (funnily enough, the question came from another computer graphicser who was trying to implement 3D parabolas!) to create a function that takes three points: two connection points (ie: A and B) and a point that directs the curve (ie: D).



3

Let's first specify the line from A to B:

$$u(\lambda) = (1-\lambda)A + \lambda B$$

where $\lambda \in [0, 1]$.

Now we stack the parabola on top:

$$f(\lambda) = (\lambda - 0)(\lambda - 1)h$$

where h is such that $f(1/2) = \vec{CD}$, which is the vector from C to D:

$$f(1/2) = (1/2)(1/2 - 1)h = \vec{CD} \iff$$
$$h = -4 \cdot \vec{CD} \iff$$
$$f(\lambda) = 4\,\vec{CD}\,\lambda(1-\lambda)$$

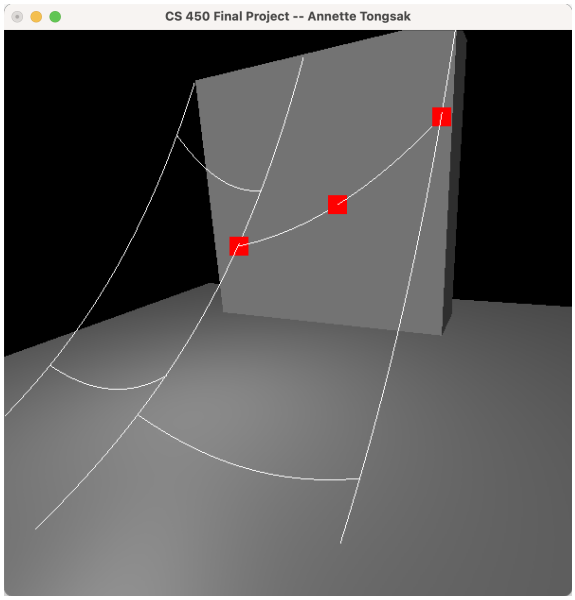
This gives

$$v(\lambda) = (1-\lambda)A + \lambda B + 4\,\vec{CD}\,\lambda(1-\lambda)$$

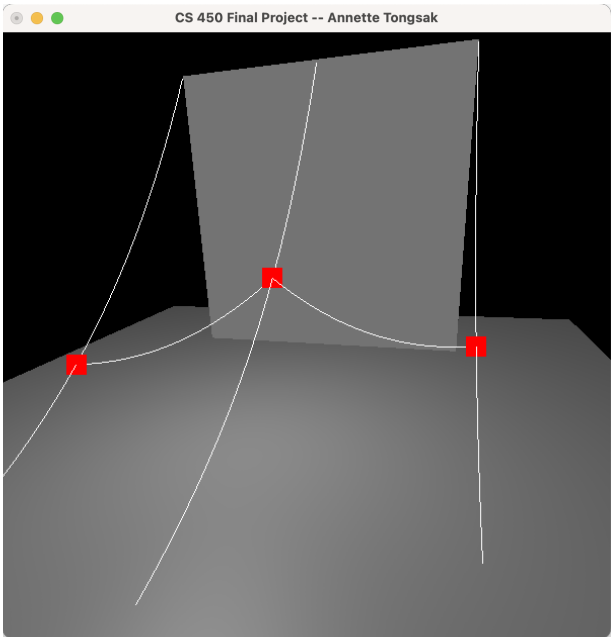
This requires $C = (A + B)/2$. That way A and B can be arbitrary points, they need not lie in the $z = 0$ plane.

Share Cite Follow edited Jun 19, 2021 at 16:25 answered Jun 19, 2021 at 10:23

mvw
34.4k 2 32 64

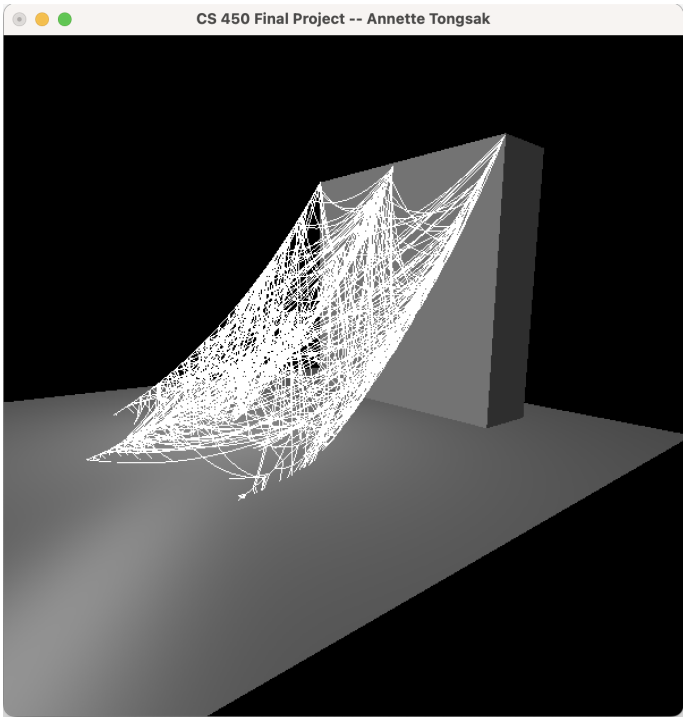


5) Knit filled curves
    a) Now that I knew how to draw a filled curve between two points on different lines, I wanted to create such filled curves at random spots on the base curves. I did this by implementing functions that allowed me to get random points on the base curves and connect them.



    b) Unlike the DreamWorks paper, I do not offset curve points with catenary equations to apply a deterministic gravity effect, which is why the filled curves are everywhere. Another flaw is that I put the calls to the function using rand() in

Display() instead of storing each filled curve outside and then calling them in main(), causing new random curves to be constantly generated and rendered.



6) Added toggles to show progression and turn on/off cobweb transparency
    a) Notably, I implemented a transparency toggle that affects the alpha value of the cobweb in an attempt to make it appear more realistic.

By doing this project, I learned a lot about the math behind implementing parabolas in 3D space. I had previously drawn parabolas on one plane and then translated or transformed it, but now I know how to create parabolas from one point to another without modifications. I also learned about blending in OpenGL to implement transparency within an object.