Final Essay - Web Developers

Tresha Desai, Shweta Jones, Anushka Sinha, Vratin Srivastava

Spider web construction is an evolutionarily conserved behavior that has evolved over hundreds of millions of years. Now each spider has evolved according to its natural and environmental cues for certain tasks. For example, some may have adapted to have stickier webs, or tighter bounded webs for capturing prey. These small modifications and adaptations that have developed over years is one of the prime reasons why we have chosen to replicate a simulation discussed by a paper titled "Analysing Spider Web-building Behaviour with Rule-based Simulations and Genetic Algorithms" by two scientists: Thiemo Krink and Fritz Vollrath. To the normal person, spider web construction appears to be random, with no organization or rules, however, the researchers within this paper were able to identify certain rules that spiders rely on in order to weave their webs. This paper focused on simulating the web-construction behavior found in spiders by modeling it into a strict set of rules they observed among orb-weaving spiders. Spider webs are essentially complex animal architecture that is a result of "dynamic interactions of reasonably simple behavior patterns".

The species of spider that are heavily studied under the scientific lenses are Orb-weavers like Araneus diadematus and Zygiella x-notata. Orb-weavers are known for building extremely particular webs that are highly suited for a specific environment and various other local conditions such as climate, vegetation, and prey. Their webs have continued to improve or adapt based on how their preys have evolved. For example, the silk used to build their webs produce a bit of stickiness which helps them to keep their prey still and also stuck at one place on the web so they can completely conquer them. Web-building behavior of Orb-weavers is studied scientifically to understand how evolution and short-term adaptations influence such complex animal behavior that results in highly intricate patterns in the form of a web.

The major sensory modality used by these spiders are kinaesthetic senses which are the senses involved in the perception of body movement, position, and control. They use certain mechanoreceptors such as stretch receptors which convert mechanical stretch into action potentials so these movements get “sensed” by their nervous system. Using their kinaesthetic senses, they build webs that typically move inside-to-out and feature concentric circle strands. To replicate such a complex animal architecture as a computational problem, authors relied on geometrical representation of spider webs. Therefore, authors ignored the impact of spatial cues and other sensory modalities while building their mathematical models to computationally build spider webs.

The way we approached the mathematical formulation of the problem was to convert the sensory inputs that spiders use to a geometric form that would emulate the closest form to how they actually build their webs. To start, we initialized a cartesian coordinate system, and determined 4 random basis points, one on each part of the axes. The goal was to determine a conic function, unique to each quadrant that would pass through the aforementioned base points. These curves would provide us the limiting length of any radii in that quadrant.

The center of the conic, in our case a circle, was determined by randomizing a point (a, b) that lies on the perpendicular bisector of the line joining the two base points for the quadrant in question. The quadrant would be determined by the angle made by a given radii with the positive X-axis. The point (a, b) was randomized by randomizing b and determining ‘a’ from the formula:

$$a = \frac{x_1^{\,2} - y_1^{\,2} + 2by_1}{2x_1}$$

The above was obtained by expanding for the equation of a circle that is centered at (a, b) and passes through the points (x_1, y_1). Once we had obtained the equation for our circle using this randomized center and the base points, we would need to determine the end point of the radii in question. The equation of the circle was given by:

$$x^2 + y^2 - 2ax - 2by = y_1^2 - 2by_1$$

If we were to use a parametric form of the end point of the radii, where we know the angle α and have to determine the length of the line λ, we would have the end point of the radii be $(\lambda cos(\alpha), \lambda sin(\alpha))$, and we can convert the equation above to the form:

$$\lambda^2 - 2(acos(\alpha) + bsin(\alpha))\lambda + (2by_1 - y_1^2) = 0$$

This equation is of a standard quadratic form, and after solving for λ, we can obtain the end point of the radii. Given that all radii originate from the origin, we now know both points and can draw the line. The frame of the web can be made by joining the endpoints of the radii. The points, to be drawn, were converted back into canvas coordinates.

For the spiral, we take an initial distance d that would be the distance on the initial radii from the origin that would be the point where our spiral building starts. We can obtain a point (x, y) for any particular spiral point on the radii once we know the distance from the origin on the radii, and the angle the radii makes with the positive x-axis. The latter is already computed as part of our line object. The distance d_i can be given by:

$$d_i = d_{i-1} + \alpha_o$$

$$where\ \alpha_o = variability\ factor$$

$$d_0 = d$$

The variability factor is randomized for every radii, accounting for the imperfect web-building in nature that may occur due to external factors such as wind or sensory signals the spider may pick up. This factor ensures that the distance on the next radii is always greater than the previous radii, preventing any spiral overlap. The point on the radii (x, y) can be given by:

$$x = d_i cos(\alpha)$$

$$y = mx,\ where\ m = \frac{\lambda sin(\alpha)}{\lambda cos(\alpha)}$$

The slope is essentially the slope of the radii passing through the origin, and hence just a ratio of the y to x coordinates of the radii, that we have calculated in the previous section. Now that we can collect all spiral points for the web, the question becomes how to terminate the web building. Here we use the rule of there being enough space for the spider to build the web. If the distance between the calculated next point and the frame is less than a specific $minDistance$, the spider reverses direction and build its web in the opposite direction. This process of reversing directions when "there is no room to build" continues until there is no room to build in either direction, and the spiral is joined to the end point of the next radii in that direction, and this terminates the building process.

When approaching our code, we used certain data-types to store specific information to help construct the web. The main datatype we have is the "web" consisting of an array of lines along with the width of the web itself. Another datatype we have is "line", which basically stores the start and end point of the line itself, its angle, and a lineType defining the role it plays within the web which could be any of the following three: frame, radii, or spiral. And our final data-type used is OrderedPair which is primarily used to store the start and end points.

As for the code hierarchy, we established a top-down programming approach to build the web in a similar manner to the orb weaving spider. To begin drawing out the radii of the web, we ask the user for a particular angle. Using that angle, we then check which quadrant it belongs to. We also randomize four base points which then need to be converted from canvas coordinate system to cartesian coordinate system. We use another function to calculate the length of each radii. In this way, we calculate the start and end points of all the radii in the cartesian coordinate system. To create a framework for our web, we use a function that takes radii as a list of lines and the number of radii as inputs and joins the endpoints of each radius. Lastly for the spirals of our web, we initiate the startpoint of the spiral on the first radius and then check if the distance between the endpoint of the next radius and the startpoint of the next spiral is within a threshold distance. After doing all the arithmetic operations on the cartesian coordinate system, we convert back to the canvas coordinate system to use canvas.go to draw out our web.

We presented our final web construction through a GIF because it gives us the opportunity to follow the general web construction pattern. We began with drawing out the radii, followed by the frame, and then finally the spirals while creating our GIF. As mentioned above, we converted our start and end points for radii, frame, and spirals from cartesian coordinate to canvas coordinate system so we can utilize the Canvas package. We used the functions MoveTo and LineTo to draw the lines associated with radii, spirals, and the framework. We created each line as an image and appended them to an image list that gets converted to a GIF using the gifhelper method.

We also incorporated the use of a webpage by utilizing the “gowut" golang package. Here, users will be able to interact with a web application and manipulate the construction of the web. The web page will allow the users to play around with the parameters such as the number of radii, the initial angle, the minimum distance, initial distance and the alpha value to observe how such parameters impact the web construction. Based on those parameters, the user input values will be directed into our highest level function called CreateWeb which will then generate a GIF accordingly. Finally, the GIF will then be shown on the web page. We aimed to build a web page for our web simulation so we can allow the users to gain insight about how such simple-looking natural architecture actually stems from complex rules and parameters. It also permits the users to observe the changes that appear on the web when certain parameters are manipulated.

As mentioned in our final project proposal, we aimed to simulate a spider web construction for our final project. We hoped to follow the Rule System put forth by the authors of the paper we selected to replicate such a natural behavior in a computational manner. Authors started building the web by mathematically modeling the behavior of orb-weaving spiders. They built a Rule System that comprises EqualSpacingRule, LengthCrossRule, LastIntDistRule, MeshSizeRule, NoRoomRule, and NoEndRule. This Rule System allowed them to construct the radii, frame, and the spirals of the web. By taking inspiration from their Rule System while also incorporating conversion between cartesian and canvas coordinate systems, we were able to implement a top-down programming approach by coming up with our own functions and subroutines. In this way, we were able to successfully build a project where we implement an artificial web construction that can be represented through a gif and also allows users to interact with the web through a web application.

References

Analysing spider web-building behaviour with rule-based simulations and ... (n.d.). Retrieved October 14 , 2022, from https://www.cs.bham.ac.uk/~jer/spider.pdf

Wessel, L. (2018, October 31). *Sticky science: The evolution of spider webs*. Scientific American. Retrieved November 16, 2022, from https://www.scientificamerican.com/article/sticky-science-the-evolution-of-spider-webs/

Tew, N., & Hesselberg, T. (2017). *The effect of wind exposure on the web characteristics of a tetragnathid Orb Spider*. Journal of insect behavior. Retrieved November 10, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5488162/

Wu, Emma (2022, October 7). *Web Servers In Go* [PowerPoint slides]. Computational Biology, Carnegie Mellon University. https://canvas.cmu.edu/courses/29392/files/8741700?module_item_id=5303530

Jahoda, P. (2021, August 29). *Go service as a web server*. Medium. Retrieved November 15, 2022, from https://medium.com/swlh/create-go-service-the-easy-way-iii-c84489cc1ee0