Assignment 3 for CS6491

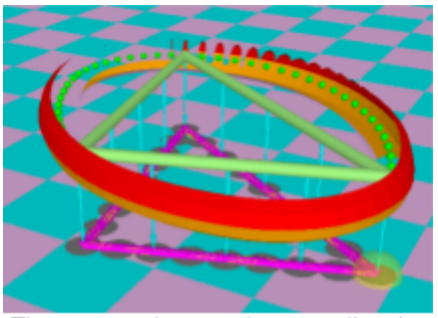
Abdurrahmane Rikli
Linh Hoang
Ruth Petit - Bois

Abstract

This paper goes through the procedure of rendering braids twisting around a series of concatenated elbows following a PCC (Piecewise-circular curve).

Problem Statement

Given a series of points, draw a PCC along those points, render some braids revolving around the PCC, then twist the braids.



The green points are the centerline that is surrounded by the rendered elbows

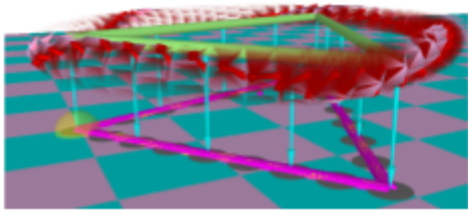
Implementation

Modules

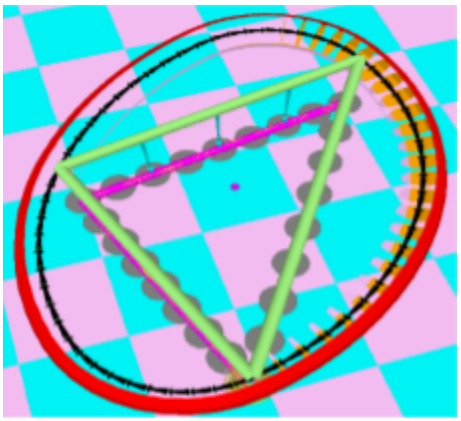
Elbow

Depiction of an elbow is the first and primary step. The main elements are:

- **Center line**: The line starts and ends at designated points and is denoted by the function C(u), which ranges between 0-1. It is generated by algorithm A1 below.
- **Vector K**: A vector normal to the plane generated by two vectors: OA, OC. O being the origin point, and A, C being the start and end points of the elbow.
- **Vector C0_C1**: An “Euler” unit vector pointing from the current C(u) to C(u+1).



The rotating K vector is depicted by the red arrows within the elbows (colored lattice not rendered)

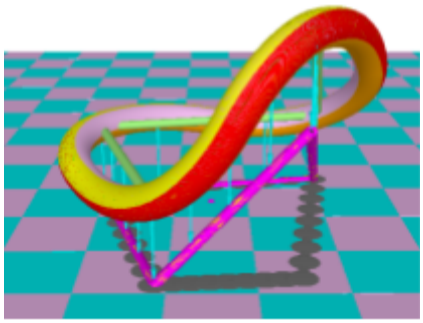


The vector C0_C1 (Cu_C(u+1)) is depicted by the black arrows

Using these elements, a lattice of quad meshes is constructed using the points S(u, v), where ‘u’ stands for the period along the center line, and ‘v’ stands for an angle by which K is rotated around the axis C0_C1.
Furthermore, the lattice is colored dependent on the ‘v’ values, where 0-0.25 is colored yellow, 0.25-0.5 is colored red, 0.5-0.75 is colored orange, and 0.75-1 is colored pink.

Performer

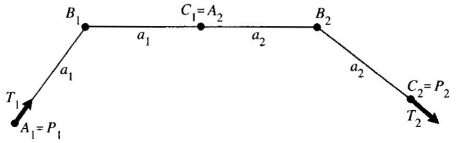
This module was a collaborative effort of the team.

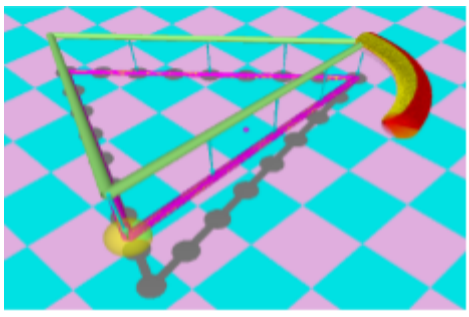


The rendered elbows showcasing the red, orange, pink, and yellow lattices

Piecewise Circular Curve (PCC)

Given an ordered set of points, and tangential information for each point (expanded on in algorithm A2), an equisided control polygon is constructed for each consecutive pair of points. The polygon is used to construct two circumcircles (A1-C1-B2, C1-B2-C).. These circumcircles’ perimeters are concatenated in such a way that the velocity remains G1 continuous. The line generated is used as the centerline for the shape.





Each edge contains 2 biarcs, depicted above is one of the two biarcs for that edge

Performer

Linh Houang exclusively worked on this module.

10/20/2018

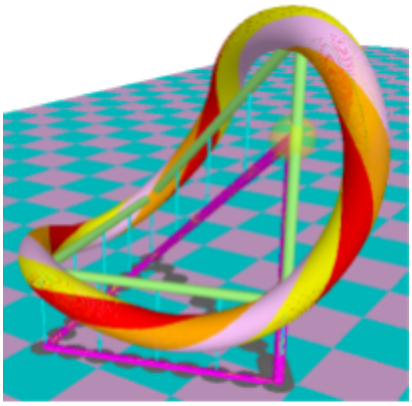
Assignment 3 for CS6491

Abdurrahmane Rikli
Linh Hoang
Ruth Petit - Bois

Twist

The twist goes through several steps:

1. Add a value to the global twist variable (multiple of 2 PI).
2. Divide the global twist value by the number of elbows in the shape (double the number of points). This provides the local twist for each elbow.
3. Run a preliminary loop over the PCC to calculate the difference between the start rotating vector and the end rotating vector.
4. Divide the difference between the two vectors by the number of elbows, and propagate the reverse extra value to each elbow's local twist to compensate.
5. Per elbow:
    a. Inherit the previous rotating vector's value (or just the normal for the starting elbow).
    b. Per C(u), rotate the rotating vector around the C0_C1 vector (calculated each C(u)) by the total twist.



The rendered elbows' lattices have been twisted but remain consistent

Performer

The Twist was a collaboration of Abdurrahmane Rikli, and Linh Houang.

Braids

The braids piggyback off of the rotating vector for each elbow, using it to generate their path: For each C(u), a vector is calculated from the rotating vector and the twist value assigned for the braids, then displaced by the tightness value of the braids (radius). This generates a separate vector for the braids to follow.
The different braid algorithms are expanded upon in the algorithm section.

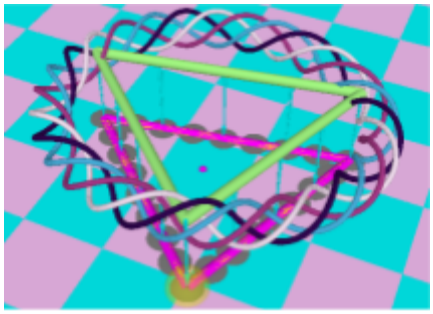
Performer

The Braids were a collaboration of Abdurrahmane Rikli and Ruth Petit - Bois.

Algorithms

A1:

```
float centerLineLeft = radius*cos(t*alpha);
float centerLineRight = radius*sin(t*alpha);
vec centerLineCalcLeft = V(centerLineLeft, I);
vec centerLineCalcRight = V(IxK).mul(centerLineRight);
vec centerLineVec = centerLineCalcLeft.add(centerLineCalcRight);
pt centerLinePoint = P(O,centerLineVec);
return centerLinePoint;
```



Example of a braid pattern produced by the program

To get the centerline point, take in the parameter *t* (being an arbitrary distance between the center points). Multiply the radius of the elbows by the cosine of the parameter *t* by the angle (now referenced as ***a***) and multiply the radius of the elbows by the sine of the parameter *t* by the angle (now referenced as ***b***) . Multiply the result of ***a*** by the vector *I* (I being the vector containing the point of the origin to the start point). Multiply the result ***b*** by the cross product of I and K (being the normal). Add these vectors together and the center point is the addition of the origin and the new calculated vector.

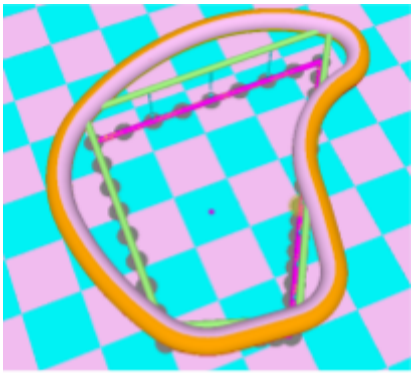


Image depicting the First Method of

A2:

Tangent estimation:

In the project Linh implemented three tangents estimations algorithms:

First Method: By using the previous edge

Second Method: Using the method in the PCC paper

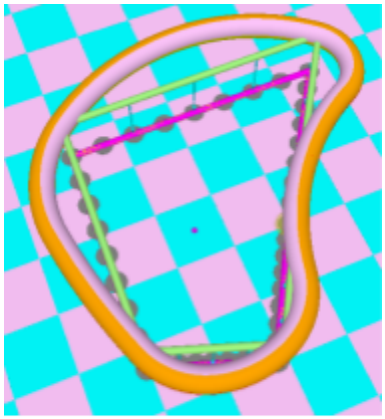


Image depicting the Second

Abdurrahmane Rikli
Linh Hoang
Ruth Petit - Bois

Third Method: By interpolating the tangent information obtained by Algorithm One using LPM morph. The pseudo code for Three is as followed:

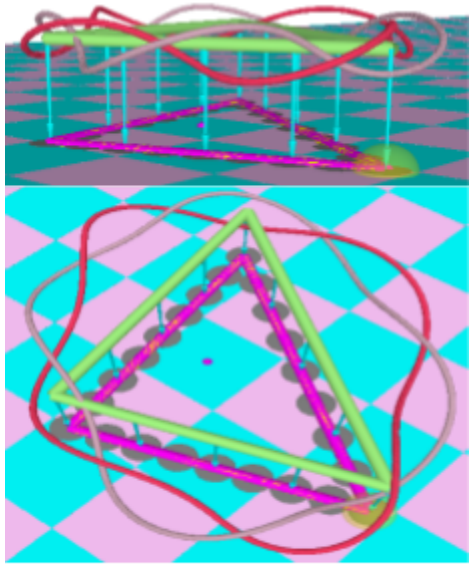
```
Tangents = Using algorithm One to get tangent information for all vertices.
For all vertices, tangent at vertex i:
        InterpolatedTangent[i] = LPM(Tangents[i - 1], Tangents[i + 1])
```

Image depicting the Third Method of Tangent Estimation (curves upon itself, but still connected)

Braids

```
pt[] points = new pt[numOfThreads];
float delta = 2 * PI / numOfThreads;
vec vector_displace = V(reference);
pt point_displaced = new pt(); point_displaced.setTo(origin);
vec angle;
```

Initiate an array of points to return. Get an angle magnitude to iterate over for each sector. Initiate a reference vector that aligns with the rotating vector for the elbow. Initiate a point that aligns with C(u).

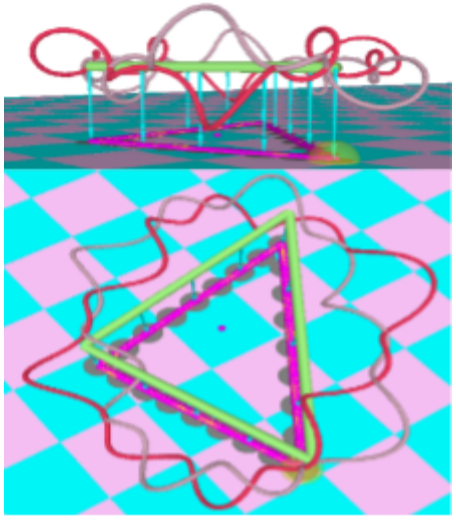


Examples of Double Helix Pattern (Side & Top)

DOUBLE_HELIX

```
For i to numOfBraids{
vector_displace = lrot(reference, axis, i * delta);
point_displaced.add(vector_displace);
points[i] = P(point_displaced);
vector_displace = V(reference);
point_displaced = point_displaced.setTo(origin);
}
```

Subdivide the circle into n sectors, then animate a path for each sector. The “weaving” only becomes apparent after twisting.

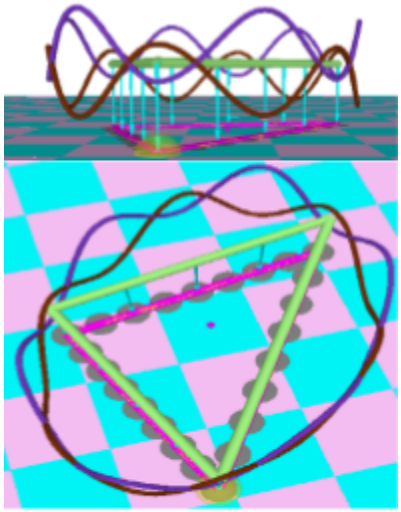


Examples of Sine Pattern (Side & Top)

SINE

```
For i to numOfBraids{
vector_displace = lrot(vector_displace, axis, i * delta); // Vector to start point
angle = vector_displace;
        vector_displace = lrot(vector_displace, axis, delta); // Vector to end point (and next sector's start point)
        point_displaced.add(angle); // Move point to start point
        vector_displace.sub(angle); // Vector for point to oscillate on
        point_displaced.add(vector_displace.mul(sin(period * 2 * PI ))); // Displace point by oscillated value
        points[i] = P(point_displaced);
        vector_displace = V(reference); // Reset vector
        point_displaced = point_displaced.setTo(origin); // Reset point
}
```

Subdivide the circle into n sectors, then have each braid oscillate at the edge of the section in a Sine arc.



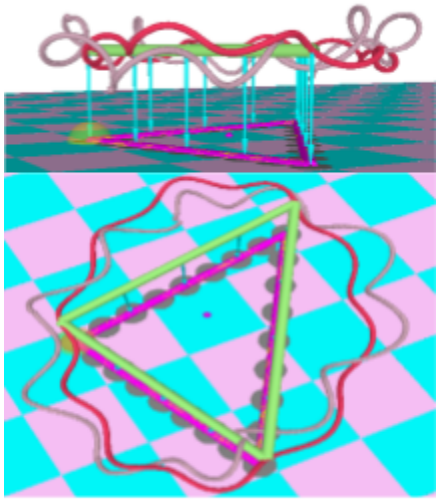
Examples of Cosine Single Pattern

COSINE

```
For i to numOfBraids{
        angle = vector_displace.rotate(i * delta, reference, referenceOrthogonal); // Vector to start point
```

```
                        vector_displace.rotate(delta, reference, referenceOrthogonal); // Vector to end point (and
        next sector's start point)
                        point_displaced.add(angle); // Move point to start point
                        vector_displace.sub(angle); // Vector for point to oscillate on
                        point_displaced.add(vector_displace.mul(cos(period * 2 * PI))); // Displace point by
        oscillated value
                        points[i] = point_displaced;
                        vector_displace = V(reference); // Reset vector
                        point_displaced = point_displaced.setTo(origin); // Reset point
}
```

Subdivide the circle into n sectors, then have each braid oscillate at the edge of the section in a Cosine arc.



Examples of SineCosine Single Pattern (Side & Top)

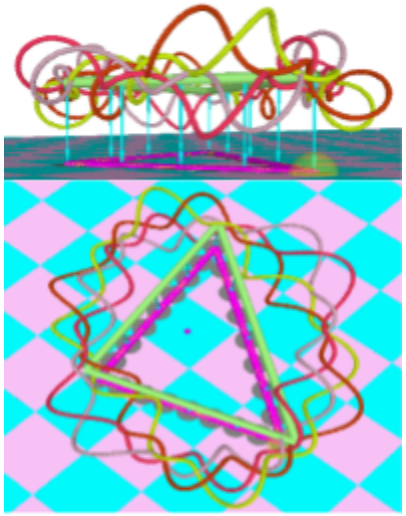
SINECOSINESINGLE

```
        points = new pt[2];
        points[0] = P(point_displaced.add(vector_displace.mul(sin(period * 2 *
PI))));
        point_displaced = point_displaced.setTo(origin);
        points[1] = P(point_displaced.add(referenceOrthogonal.mul(cos(period * 2 *
PI))));
```

Draw a Sine arc vertically and a cosine arc horizontally.

SINECOSINESERIES

```
delta = PI / numOfThreads; // Redistribute the delta for half the circle instead
        for(int i = 0; i < numOfThreads*2; i+=2){
        // Sine sector
                vector_displace = lrot(vector_displace, axis, i * delta); //
        Vector to start point
                angle = vector_displace;
                vector_displace = lrot(vector_displace, axis, numOfThreads * delta); // Vector to end point
                point_displaced.add(angle); // Move point to start point
                vector_displace.sub(angle); // Vector for point to oscillate on
                point_displaced.add(vector_displace.mul(sin(period * 2 * PI))); //
        Displace point by oscillated value
                points[i] = P(point_displaced);
                vector_displace = V(reference); // Reset vector
                point_displaced = point_displaced.setTo(origin); // Reset point
                // Cosine sector
                vector_displace = lrot(vector_displace, axis, (i+1) * delta); //
        Vector to start point
                angle = vector_displace;
                vector_displace = lrot(vector_displace, axis, numOfThreads * delta);
        // Vector to end point
                point_displaced.add(angle); // Move point to start point
                vector_displace.sub(angle); // Vector for point to oscillate on
                point_displaced.add(vector_displace.mul(cos(period * 2 * PI))); //
        Displace point by oscillated value
                points[i+1] = P(point_displaced);
                vector_displace = V(reference); // Reset vector
                point_displaced = point_displaced.setTo(origin); // Reset point
        }
```



Examples of SineCosine Series Pattern (Side & Top)

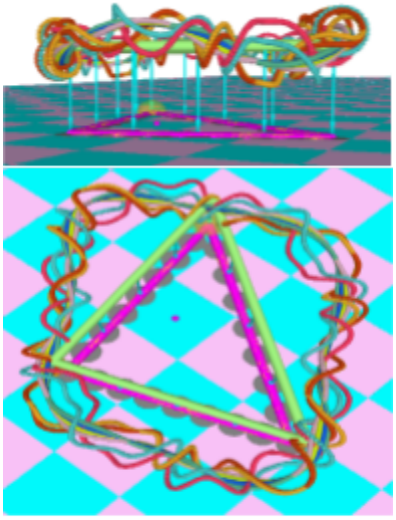
Oscillates the path from one sector to the next-opposite sector, alternating between a sine wave and a cosine wave.

SINEALT

```
        points = new pt[8];
        for(int i = 0; i < 8; i+=2){
                vector_displace = lrot(reference, axis, (i * PI / 2) + sin(period * 2 * PI) * PI);
```

Abdurrahmane Rikli
Linh Hoang
Ruth Petit - Bois

```
            points[i] = P(point_displaced.add(vector_displace));
            point_displaced = point_displaced.setTo(origin); // Reset point
            vector_displace = V(reference); // Reset vector
            vector_displace = lrot(reference, axis, PI / 6);
            point_displaced.add(vector_displace);
            angle = lrot(reference, axis, (i * PI / 2) * 4 * PI / 6);
            angle.sub(vector_displace);
            points[i+1] = P(point_displaced.add(angle.mul(cos(period * 2 * PI))));
    }
```



Examples of SineAlt Pattern (Side & Top)

Extra Functionality

Keyboard Key	Action	Keyboard Key	Action
< || >	Twist Elbow Clockwise || Counter Clockwise	y	Turn the Radius of the Elbows to 5 instead of 20
[||]	Twist Original Black Path P Clockwise || Counterclockwise	1	Add / Remove the rendering buffer for increased performance
/	Reset Elbow Twists to 0	-	TIghten the twists/braids
h	Switch Tangent Methods	2	Increase the number of paths
g	Render Original Black path P	s	Decrease the number of paths
j	Render the Biarc (Or anything to do with the Elbows)	;	Show colored quads/Lattice Structure
\	Show Rotating K (Red Arrows)	o	Show CenterLine (green points)
v	Switch between path patterns		

Credits

Abdurrahmane Rikli wrote the first draft. Linh Hoang & Ruth Petit - Bois edited. The code was written by all, with notation indicating who wrote what. Screenshots and video taken by Ruth Petit - Bois.