Title: Rigid Interpolation

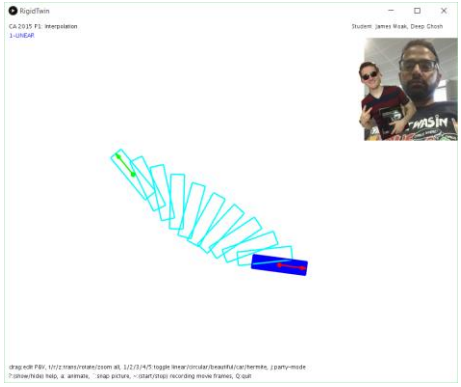
Team Members: James Moak, Deep Ghosh

Project Description: Build linear and circular interpolation between user provided keyframes (2 keyframes). Define a beautifying interpolation between the two keyframes which has an S like motion. Define a third keyframe and define an interpolation between the 3 frames which creates an S like motion. Add texture to the keyframe and animate the interpolation.

User Manual:

- t – translates the 2 keyframes and interpolating boxes between them on the screen.
- r – rotates the 2 keyframes and the interpolating boxes between them on screen.
- z – moves the 2 keyframes closer or further.
- drag – to change the individual keyframe location and orientation.
- 1 – Linearly interpolates between the 2 keyframes.
- 2 – Interpolates circularly between the 2 keyframes.
- 3 – Defines a 3rd keyframe based on the provided 2 keyframes and interpolates between the 3 keyframes using Neville's interpolation.
- 4 – Uses a car texture to interpolate between the 2 frames using a Hermite interpolation.
- 5 – Interpolates between the 2 frames using a Hermite interpolation but flips the vectors on the 2 keyframes when calculating the Hermite to create a different interpolation than 4.
- j – Applies a texture to all the keyframes on screen as well as the interpolating boxes.
- a – Animates all the active interpolations between the 2 keyframes.

Outline of Solution:



We will break down our solution into 5 parts as per the 5 modes defined in the application.

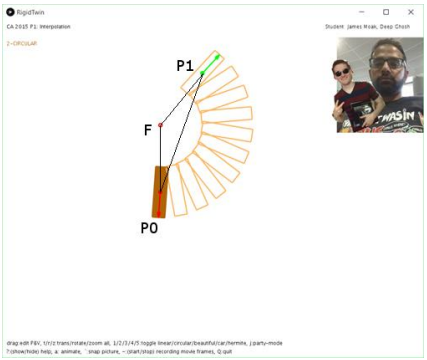
1. Linear – Already implemented in the given solution.

2. Circular – We compute a point (marked by the red dot in the following image). This point is the center of rotation for the circular interpolation. We solve for the point using the following steps.

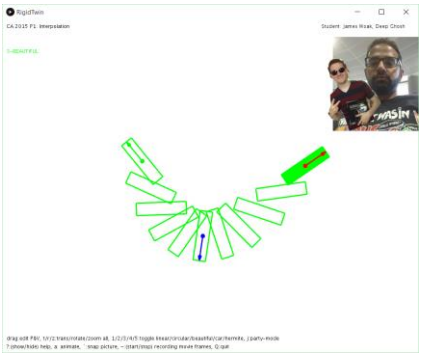
a. Let H be the midpoint between P0 and P1.We can find the midpoint by scaling the vector P0P1 by ½ and then applying the vector to point P0.

b. Find the distance between F and H by calculating the ratio of the distance of P1H and FP1. Let this distance be g.

c. Now scale the rotated vector P0P1 by distance g from H to find the center.



3. Beautiful (Neville's) – First we define a third frame based on the 2 provided keyframes. We check if there is an intersection between the 2 vectors that are provided. If there is we take the intersection point as the point of the 3rd frame. We define the vector at the third point by rotating one of the intersecting vectors by half of the angle between the vectors and flipping the resultant vector by 180 degrees.

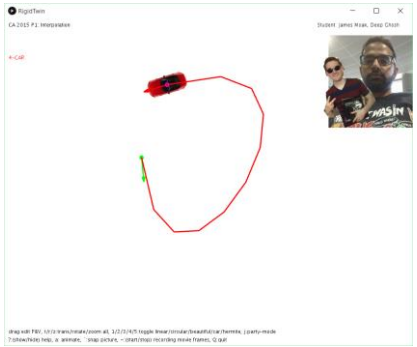


Next, define a Neville curve across our three points, where the formulation for P(t) on the curve for time t is L(L(P0,P1,t*2),L(P1,P2,t*2-1),t). To interpolate between the angles, we linearly interpolate the angle between our first frame and our calculated middle frame. Instead of going from 0 to ½t we go from 0 to t, and between the middle frame and the last one, we interpolate their angles from 0 to t again to ensure smoothness.

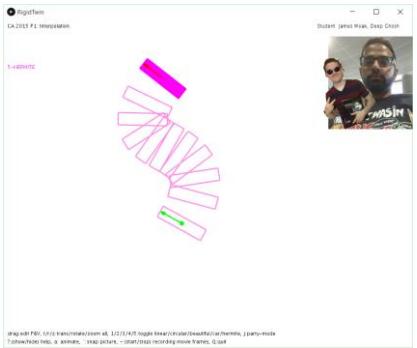
4. Car – This implementation uses the standard cubic Hermite interpolation. The following equation provides the basis for calculating the point during the interpolation.

```
float h1 = 2.0*pow(s,3.0) - 3.0*pow(s,2.0) + 1.0;
float h2 = -2.0*pow(s,3.0) + 3.0*pow(s,2.0);
float h3 = pow(s,3.0) - 2.0*pow(s,2.0) + s;
float h4 = pow(s,3.0) - pow(s,2.0);
// pt p = P(P(h1, P0, h2, P1), W(h3, V0, h4, V1));
pt p = P(h1*P0.x + h2*P1.x + h3*U0.x + h4*U1.x,
         h1*P0.y + h2*P1.y + h3*U0.y + h4*U1.y);
```

Where U0 and U1 are the original vectors from the 2 keyframes.



5. Hermite (rotated vectors) – This implementation uses the same logic as the 4th mode except the vectors for the 2 keyframes are rotated by 90 degrees to ensure a smooth curve. We rotate the vectors to create a better smoother curve when drawing the bounding boxes.



Observations:

We found that none of the 5 defined interpolations are universally beautiful in all cases. We also noticed that some of the interpolation's paths were beautiful/pleasing but their animations not so much, and vice versa.

Justification:

We added a total of 3 curves (not counting circular) because we thought they may be beautiful.

The first we added was Hermite. We wanted the interesting snake like motion it brings, taking advantage of the vectors in a dramatic way. Sadly, its animation was too un-smooth, causing it to look bad in most situations.

The next curve was the Car curve. It was simple – remove the rotation of the vectors we applied to Hermite to work with the rectangles more, and see the animated point as a car driving around. The changes in velocity ended up perfectly suiting it, and with the texture of a car it looks amazing. It goes to show how context can define how beautiful the animation looks, since it acts like a car and looks like one, despite the same animation looking terrible on the rectangles.

The final curve we added was the Beautiful curve, or the "calculate a third frame and use Neville's interpolation to generate a smooth curve between them" curve. This was beautiful because it was similar to Circular, if not as robust, while also being more interesting.

Limitations:

Our curves have many limitations. Beautiful and Hermite have keyframe combos that can cause them to look very poor. Beautiful's animation is almost always nice, but its path can look jarring. Hermite is the exact opposite – it has a pleasing path in many cases, but its motion is off-putting just as often.

Car looks great for virtually all keyframes except for ones that generate paths with very jagged turns.

Validation:

We conducted a user study on some people. The study consisted of two fixed key frames, and running through each curve's animation. We asked for their thoughts/reactions/feelings. For the most part, they thought the Circular curve was beautiful, but preferred the Beautiful curve (Neville's Curve) saying it was wider and more interesting. They also all enjoyed the animation of the Car, saying it looked, "like a real car driving". The Hermite curve with the rectangles was disliked overall for not animating smoothly.

Our test frames, with all interpolations at once: