**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.

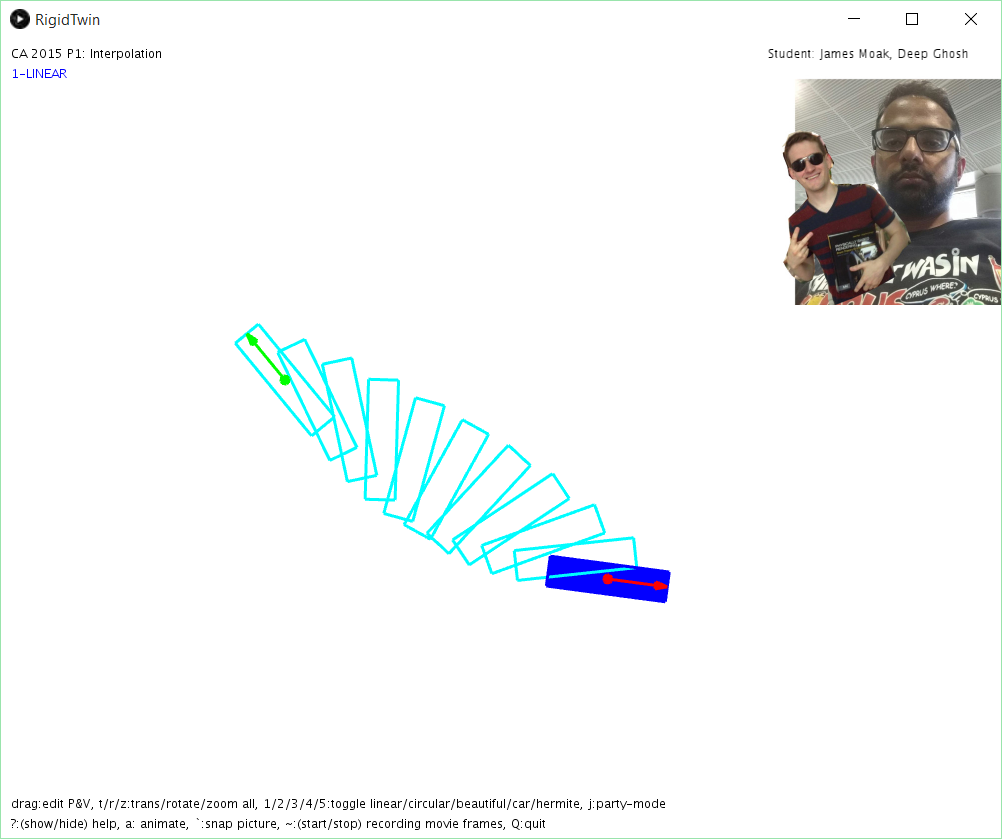
**User Manual:**

* 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.

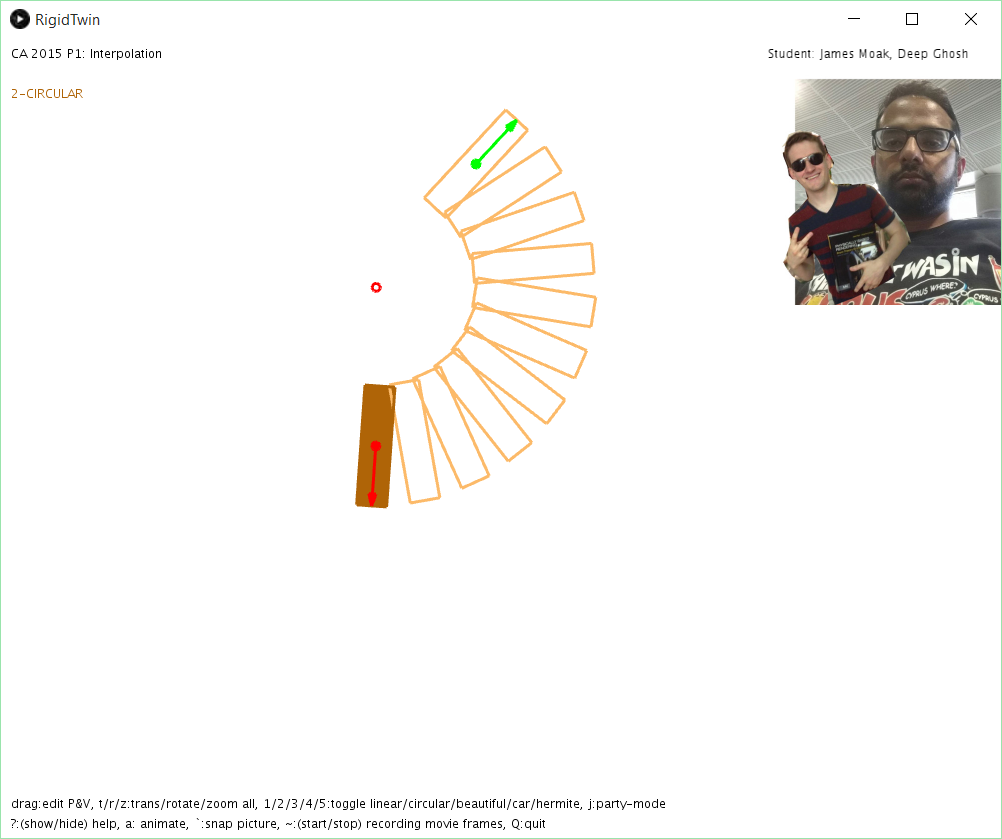


2. Circular – We compute a point (marked by the red dot in the following image). This point is the center of rotation for the circular interpolation. We solve for the point using the following steps.

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.



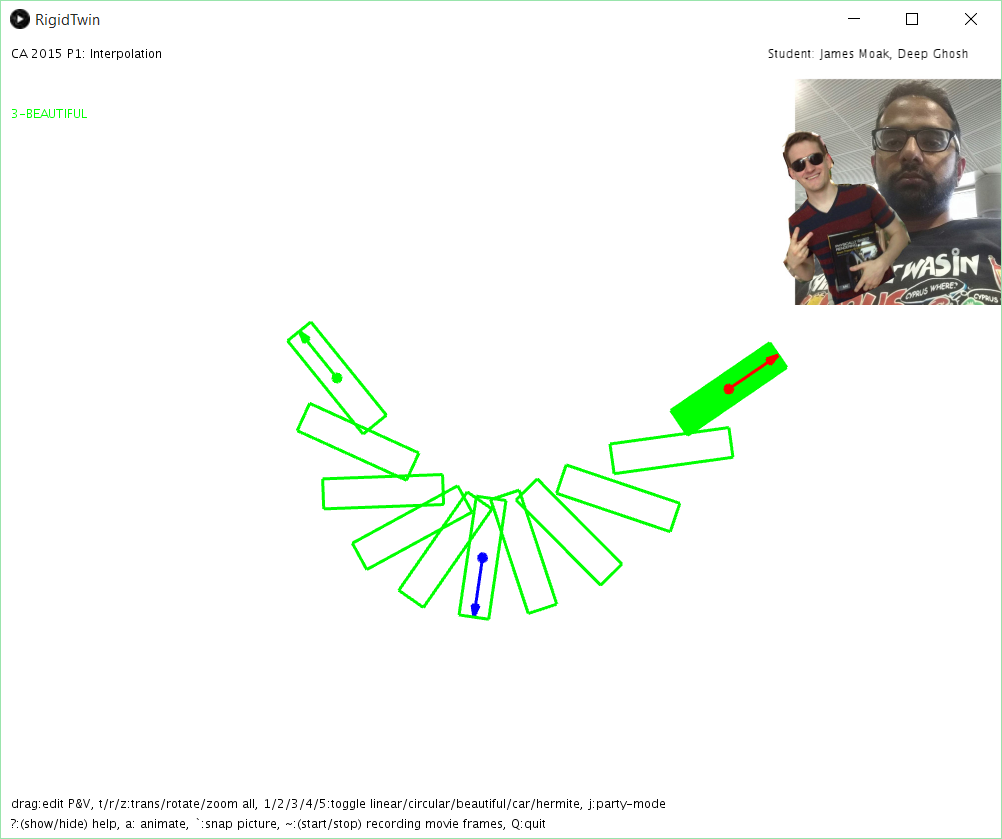
F

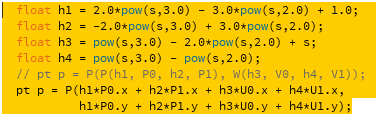
P1

P0

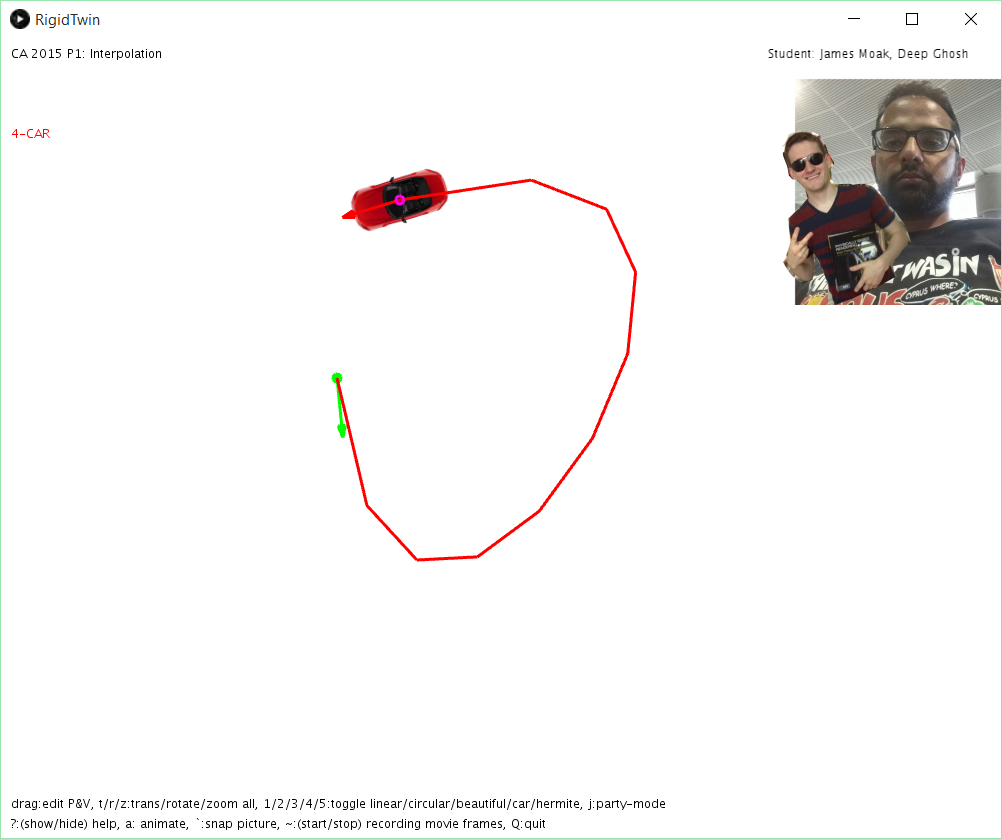
3. Neville’s – First we define a third frame based on the 2 provided keyframes. We check if there is an intersection between the 2 vectors that are provided. If there is we take the intersection point as the point of the 3rd frame. We define the vector at the third point by rotating one of the intersecting 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.

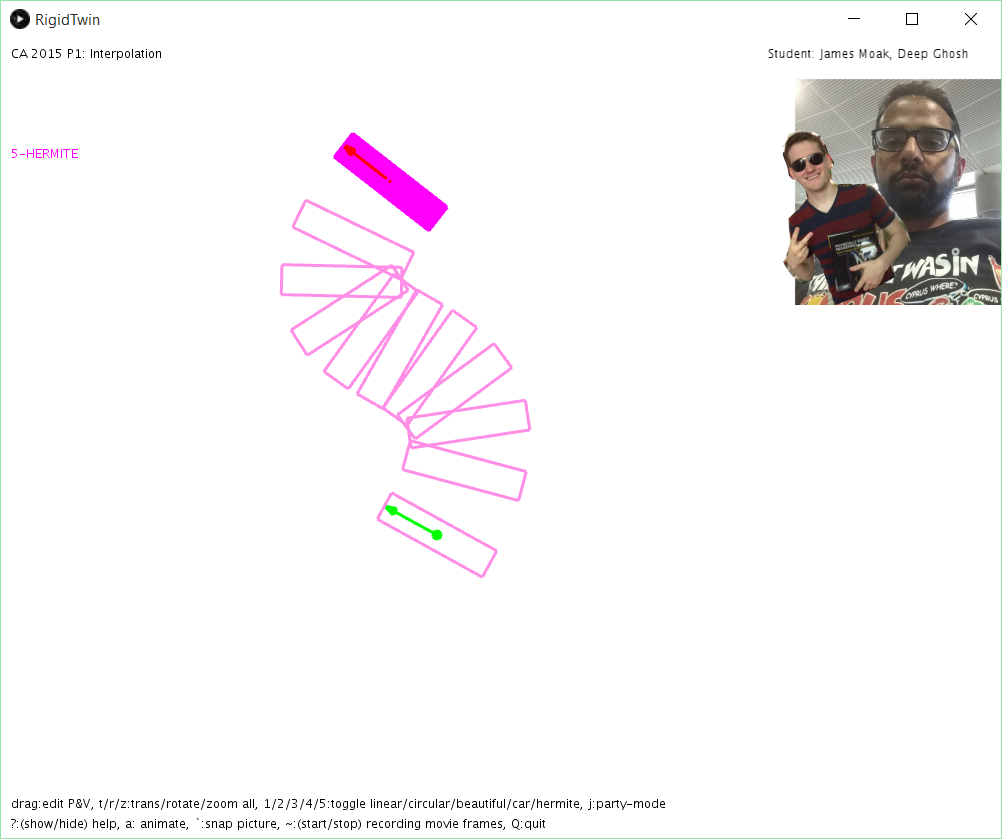


4. Hermite – This implementation uses the standard cubic Hermite interpolation. The following equation provides the basis for calculating the point during the interpolation.  
 

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



5. Hermite (flipped 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. In different poses/positions of the keyframe and depending on the end objective the beauty of the interpolations can be defined as beautiful. When we say objectives we mean the form of the curve being generated, like the interpolation with the car looks nicer at times as there is no need to draw the intermediate boxes. While some interpolations look nicer while animating but not so much when they are drawn out without animation.

**Justification:**

**Limitations:**

**Validation:**