CS 6323 Computer Animation & Gaming

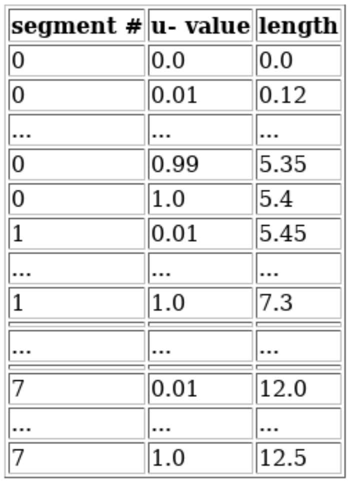
Assignment 4 (Grade: 10 points)

Path Following with Orientation Interpolation

Now you already have a program to move the shuttle object along the curve path through eight points, please write a program to rotate the shuttle object. The orientation of the shuttle object is required to interpolate the orientation of the eight points, which are given by quaternions. When the shuttle object is moving along the curved path, use Slerp to interpolate the orientation between two quaternions along the corresponding line segment.

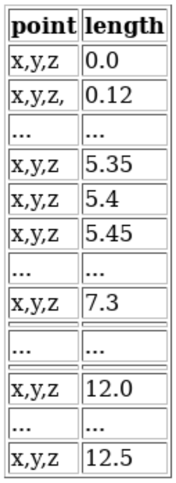
Initialization:

1. Set coefficient matrices for each segment using Catmull-Rom spline formulation. (Already finished in your Assignment 2)
2. For each segment, loop through points, summing linear distances to create a table of parametric values and summed linear distances to approximate arc length. Because we are using multiple segments, the table might look like:



segment #	u- value	length
0	0.0	0.0
0	0.01	0.12
...	...	...
0	0.99	5.35
0	1.0	5.4
1	0.01	5.45
...	...	...
1	1.0	7.3
...	...	...
7	0.01	12.0
...	...	...
7	1.0	12.5

OR



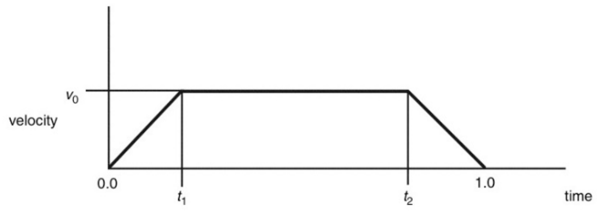
point	length
x,y,z	0.0
x,y,z,	0.12
...	...
x,y,z	5.35
x,y,z	5.4
x,y,z	5.45
...	...
x,y,z	7.3
...	...
x,y,z	12.0
...	...
x,y,z	12.5

   Compute at least 200 point per segment. If you normalize the lengths in the table so the total length is 1.0 and if the ease-in/ease-out function i/o is also normalized to go from ***0.0*** to ***1.0***, then the code is easier to reuse with other ease-in/ease-out procedures.
3. Draw the cubic curve (Already finished in your Assignment 2).
   Please keep your assignment 2 implementation in this assignment. The cubic curve is still rendered.

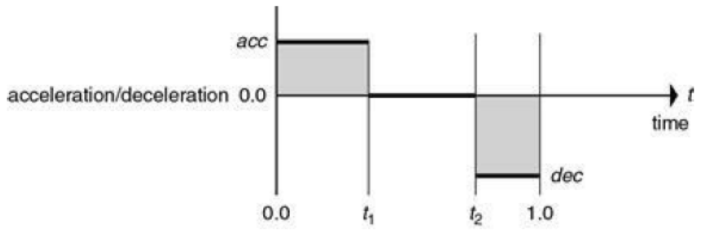
   Keep your assignment 3 ease-in/ease-out implementation.

Simulation:

1. Increment a time value ***t***, that goes from zero to one as the curve is traversed
2. Apply an ease function, ***s* = *ease(t)***, using constant acceleration assumption. Support interactive change of t_1 and t_2.



$v = v_0 \cdot \frac{t}{t_1}$	$0.0 < t < t_1$
$v = v_0$	$t_1 < t < t_2$
$v = v_0 \cdot \left(1.0 - \frac{t - t_2}{1 - t_2}\right)$	$t_2 < t < 1.0$



$a = acc$	$0 < t < t_1$
$a = 0.0$	$t_1 < t < t_2$
$a = dec$	$t_2 < t < 1.0$

3. Search the table created by the initialization routine for the entries ***s*** in between
4. Compute the fraction that ***s*** is between two entries
5. Use the computed fraction to interpolate between the points recorded in the table, ***u = table(s)***
6. Evaluate the interpolation function to produce a point along the curve, ***p(x,y,z) = P(u(s(t)))***. More specifically, use current time to calculate current distance using ease function. Use current distance to get the current interpolation position based on searching. Do transformation based on current position and previous position.
7. Based on the computed fraction u and two quaternions of end points of the curve segment, interpolate the quaternion $q(t)$ using Slerp.
8. Rotate the object according to $q(t)$ and move the object according to $p(x, y, z)$.

Note: because *t* monotonically increases, so does *s* and, therefore, so do the indices of the entries retrieved from the table. This should make your search more efficient since you can start the next search from the point of the previous search.

Requirements:

- The orientation quaternion $q = s + ix + jy + kz$ for the eight points are given in the starting code as (s, x, y, z). The starting code draws a local coordinate axis for each point, which are all identical to world coordinate system. Please rotate the cubes and their local coordinate axis at each point according to the given orientation quaternion q. (Grade: 2 points)
- Please place the moving bunny object and its axis initially at the first point and their orientation being identical to the orientation of the first point. (Grade: 1 points)
- Add a button to GUI to enable/disable the ease-in/ease-out control. Add a button to GUI to enable/disable the rotation control. (Grade: 1 points)

- When rotation control is enabled, the bunny object adjusts its orientation based on Slerp interpolation. The local coordinate axis should be moving and rotating with it. (Grade: 6 points)

Note: Quaternion to Rotation matrix:

Suppose the quaternion is $q = s + ix + jy + kz$, then the corresponding rotation matrix $\boldsymbol{M}_q$ is:

$$\boldsymbol{M}_q = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2sz & 2xz + 2sy \\ 2xy + 2sz & 1 - 2x^2 - 2z^2 & 2yz - 2sx \\ 2xz - 2sy & 2yz + 2sx & 1 - 2x^2 - 2y^2 \end{bmatrix}$$