CS 6323 Computer Animation and Gaming

Assignment 3 (Grade: 10 points)

Path following with ease-in/ease-out control

Write a program to move a aircraft object that follows a curved path through eight points starting from the first point, accelerating to some maximum speed at t_1, keeps constant speed till t_2, and then decelerating to stop at the initial position.

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

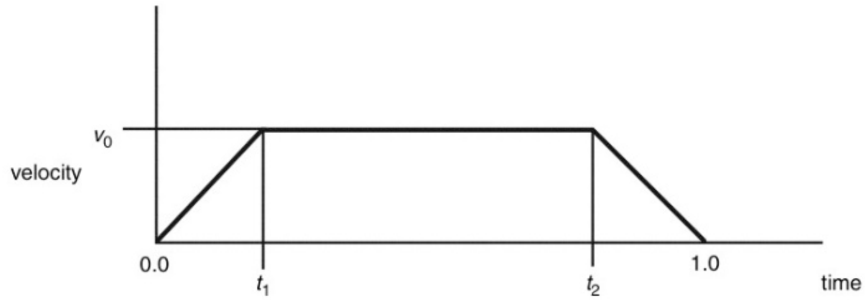
OR

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.

Simulation:

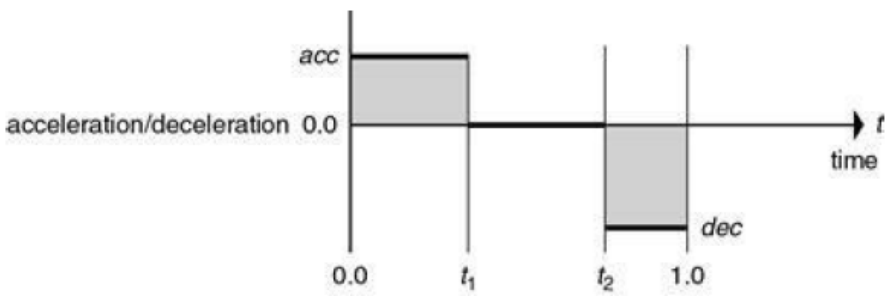
1. Increment a time value ***t***, that goes from zero to one as the curve is traversed
2. Apply an ease function, ***s = ease(t)***, using constant acceleration assumption. Support interactive change of t_1 and t_2.



$$v = v_0 \cdot \frac{t}{t_1} \qquad 0.0 < t < t_1$$

$$v = v_0 \qquad t_1 < t < t_2$$

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



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

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.

You do NOT need to control the orientation of the aircraft object in this assignment.

Important Hints:

1: Your input contains three variables: t_1, t_2 and total time t_{total}. t_1 and t_2 should be two variables **normalized** in the range (0,1) and t_{total} should be your total **actual moving time** in real world. The time is normalized to (0,1) in the equations. Make sure you convert it back them to the total time in real world. Hence, your actual accelerate time is (0, $t_1 * t_{total}$), constant speed time is ($t_1 * t_{total}, t_2 * t_{total}$) and deceleration time is ($t_2 * t_{total}, t_{total}$). Because you have your total distance fixed on your distance table, you need to calculate the acceleration value acc, deceleration value dec and the constant speed value v, based on some simple high school physics equations.

2: Note that, because $\boldsymbol{t}$ is monotonically increasing, so does $\boldsymbol{s}$ and, therefore, so do the indices of the entries retrieved from the table. It can be better if you start the next search from the point of the previous search(however, using binary search is also fine).