CPSC 587/687: Fundamentals of Computer Animation

Andrew Owens, Jeremy Hart, Adam Runions and Przemyslaw Prusinkiewicz

University of Calgary

NOTE: These tutorial notes are a work in progress. If you find errors within this document, please relay them to either TA (Andrew or Jeremy), and we will make the appropriate corrections. Thank you!

Table of Contents

1 Roller Coaster Assignment

The roller coaster assignment is comprised of two parts, with the second building on the first. The first part of the assignment is to simulate a *bead on a wire* that moves along the wire dictated by the kinematics of the conservation of mechanical energy. There are several components which require careful consideration and implementation in order to complete this assignment (and its bonuses!), with the first being "how to define the wire/track?"

1.1 Creating the wire

There are many candidate techniques for designing a curve in three-dimensional space, we will present two that are particularly apt for the scope of the assignment (i.e. easy to understand and implemented if very few lines of code). Both methods produce a piece-wise linear curve for visualization, but their approaches to calculating arc-length parameterizations and creating coordinate frames along the curve differ slightly. So you may choose either, but they will affect your implementation choices later in the assignment. Luckily, both require the same amount of work in the end, so there is no wrong choice.

B-spline subdivision curve, *"The chasing game"* Given a small ordered set of n three-dimensional points, $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$, where $\boldsymbol{p}_n \in \mathbb{R}^3$, a closed, piece-wise linear curve can be created by connecting the consecutive points with line segments in their prescribed order.

The essence of the *"The chasing game"* is to successively refine the initially coarse piece-wise linear curve into a finer curve. This iterative approach requires two steps per iteration, with each iteration producing another piece-wise linear curve that is finer than the previous.

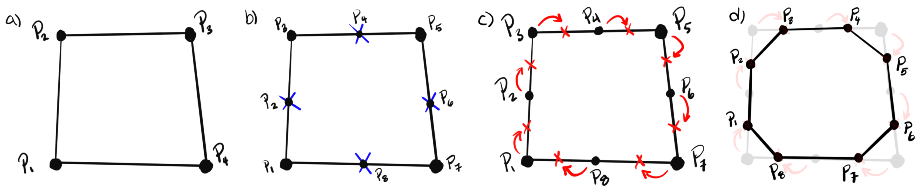


Fig. 1. *"The chasing game"* closed curve subdivision scheme.

i) Given the current piece-wise linear curve $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$ (e.g. Figure 1a), we insert new points at the mid-points of each line segment (e.g. Figure 1b), introducing a new piece-wise linear curve with twice as many points as before, $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{2n})$

ii Given the new set of $2n$ points, $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{2n})$, we consecutively reposition each point $\boldsymbol{p}_i$ to the average of the following point and itself, i.e.

$$\boldsymbol{p}_i^{new} = \frac{1}{2}\left(\boldsymbol{p}_i + \boldsymbol{p}_{i+1}\right) \quad \text{(e.g. Figure 1c)} \tag{1}$$

thus producing another piece-wise linear curve with twice as many points, and is smoother than previously defined (e.g. Figure 1d).

The above formulation assumes a closed piece-wise linear curve. Applying the *"The chasing game"* to an open curve will progressively shorten the front of the curve (e.g. Figure 2a,b). To properly produce an open curve, without shortening the curve with every subdivision step, simply create an additional copy of the point $\boldsymbol{p}_1$ before in step *i)*, and ignore re-positioning it in step *ii)*, therefore fixing it in space (e.g. Figure 2c).

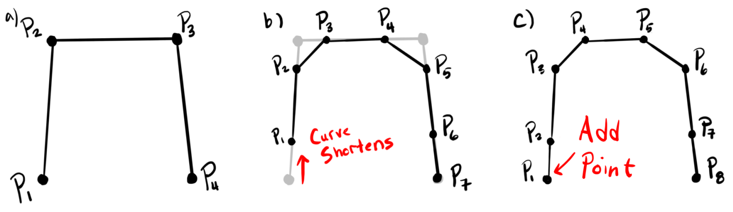


Fig. 2. *"The chasing game"* open curve subdivision scheme.

Most roller coasters are cyclical (closed curves), as roller coaster carts typically return to where they started, so this may not be appropriate for your design; however, an open curve is an option if you are inclined to create an unconventional roller coaster. After a number of iterations, you will obtain a piece-wise linear curve that is sufficiently smooth enough for the remainder of the assignment.

Parametric composite Bézier curve A cubic Bézier parametric curve $B(t)$ with $0 \leq t \leq 1$, is defined by four control points $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4 \in \mathbb{R}^3$, as defines the set of all points such that

$$B(t) = (1-t)^3 \boldsymbol{p}_1 + 3(1-t)^2 t\boldsymbol{p}_2 + 3(1-t)t^2\boldsymbol{p}_3 + t^3\boldsymbol{p}_4 \quad \text{s.t.} \quad 0 \leq t \leq 1 \qquad (2)$$

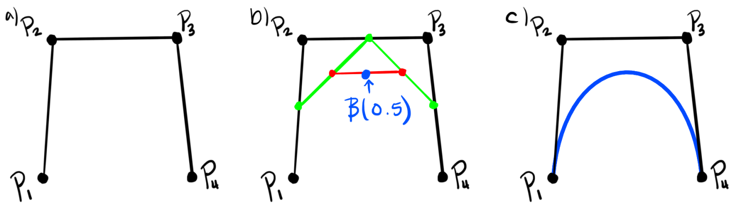


Fig. 3. Cubic Bézier parametric curve. a) A configuration of the 4 control points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$ and $\boldsymbol{p}_4 \in \mathbb{R}^3$. b) A point evaluated on the curve at $B(0.5)$ where $t = 0.5$ (using the *de Casteljau* algorithm). c) The set of all points evaluated at each $0 \leq t \leq 1$ creates a smooth curve that interpolates the endpoints.

Cubic Bézier parametric curves of only four control points are easy to evaluate, (e.g Figure 3b, using the *de Casteljau* algorithm, or directly evaluating points using Equation 2), however they not all that expressive. Simply adding more control points necessarily increases the order of Bézier curves, so an alternative is required to have more expressive curves, and thus roller coasters tracks. An option is to compose multiple cubic Bézier parametric curves end-to-end into a single curve $C(t)$. (see Figure 4).

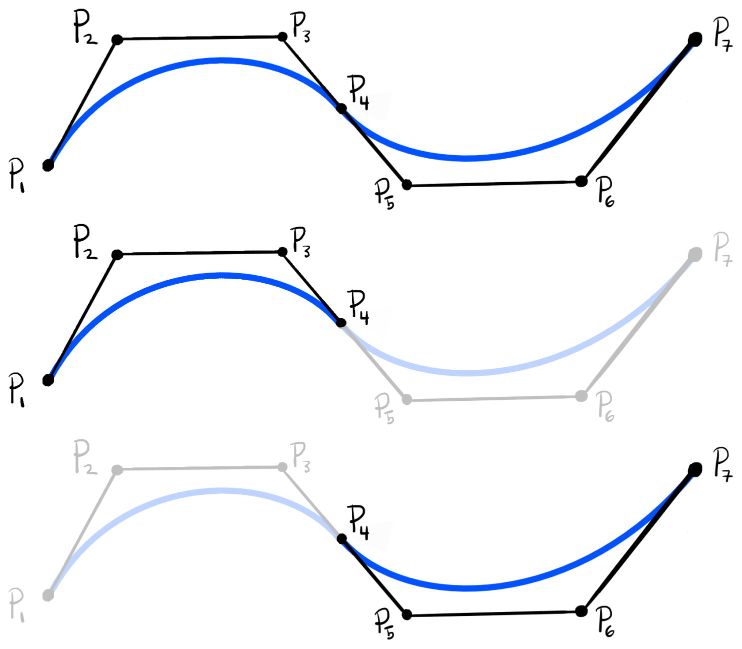


Fig. 4. Composite cubic Bézier parametric curve. (Top) The composite Bézier curve $C(t)$ comprised of seven control points. (Middle) The first cubic Bézier curve $B_1(t)$, with $0 \leq t \leq 1$, is comprised of the points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$ and $\boldsymbol{p}_4 \in \mathbb{R}^3$. (Bottom) The second cubic Bézier curve $B_2(t)$, with $0 \leq t \leq 1$, is comprised of the last point of the previous curve, $\boldsymbol{p}_4$, and the following points $\boldsymbol{p}_5$, $\boldsymbol{p}_6$ and $\boldsymbol{p}_7 \in \mathbb{R}^3$.

As we see, each cubic Bézier curve $B_i(t)$ requires 4 control points to be appropriately defined, while sharing the end point with neighboring curves, i.e. $B_i(t)$ and $B_{i+1}(t)$ share a single control point. A simple way to guarantee you have the appropriate number of control points for n cubic Bézier curves, you require $3n+1$ control points for an open curve, and $3n$ for a closed curve (requiring at least two curves, $n=2$).

Each cubic Bézier curve $B_i(t)$ composing the final curve are individually paramterized as $0 \leq t \leq 1$. It is convenient to parameterize the entire composite curve (or spline) $C(t)$ with $0 \leq t \leq 1$ (e.g. Figure 5). An easy way to achieve this parameterization of the curve $C(t)$, comprised of n cubic Bézier curves, is to partition the parameter space of $C(t)$ into n equal disjoint subsets.

Example 1. With $n = 2$ curves, the first Bézier curve $B_1(t)$ produces the points of the curve $C(t)$ for $0 \leq t < \frac{1}{n}$, and the second Bézier curve $B_2(t)$ produces the points of the curve $C(t)$ for $\frac{1}{n} \leq t < 1$ (see Figure 5).

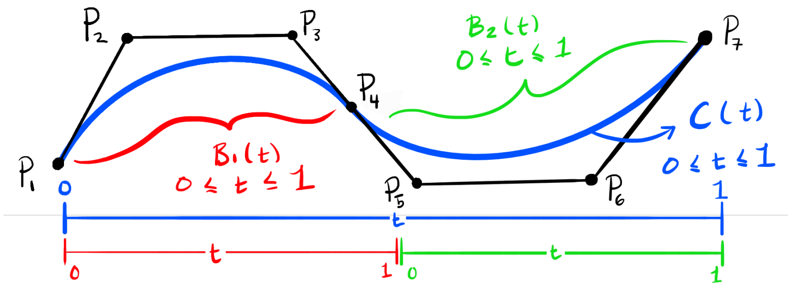


Fig. 5. Partitioning of the parameter space $0 \leq t \leq 1$ of the composite curve $C(t)$, comprised of cubic Bézier curves $B_1(t)$ and $B_2(t)$.

We note that each composing curve $B_i(t)$ interpolates its endpoints (e.g. Figure 4b,c), and that neighboring curves $B_i(t)$ and $B_{i+1}(t)$ share a control point $\boldsymbol{p}_j$. In order for the curve $C(t)$ to be (C^1) smooth at all points, we must guarantee that the *tangents* of $B_i(t)$ and $B_{i+1}(t)$ are the same at point $\boldsymbol{p}_j$. This is accomplished by ensuring that control points $\boldsymbol{p}_{j-1}$, $\boldsymbol{p}_j$ and $\boldsymbol{p}_{j+1}$ are collinear (e.g. $\boldsymbol{p}_3$, $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$ in Figure 5).

1.2 Moving along the curve with unit speed

As we will see later in this assignment, the bead/cart will move along the curve based on a calculated *speed*. We know that speed $v \in \mathbb{R}$ is equivalently defined as

$$v = \frac{\Delta s}{\Delta t} \tag{3}$$

where $\Delta s \in \mathbb{R}$ is the distance traveled, and $\Delta t \in \mathbb{R}$ is the change in time. Therefore, for a user defined Δt (e.g. $\Delta t = 0.0001$), the distance traveled for a given speed s is calculated as

$$\Delta s = v \Delta t. \tag{4}$$

Thus, which ever method that you have chosen to create your roller coaster curve (see above), you need a method to move along the curve, from any given point, by a given distance Δs (see Figure 6).

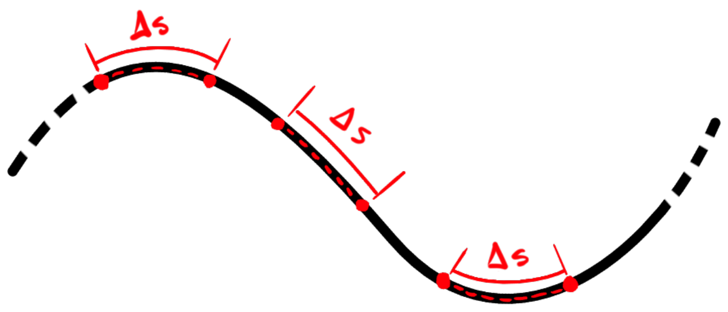


Fig. 6. Points on a curve separated by a distance Δs.

Moving along a piece-wise linear curve Given a piece-wise linear curve, exclusively defined as sequence of points $\{\boldsymbol{p}_i\}$ in $\mathbb{R}^3$ (as produced by B-spline subdivision), there are several methods to move along the curve by a given distance $\Delta s \in \mathbb{R}$. One that is particularly suitable for this assignment is to incrementally "march" along the curve until you have moved a distance of Δs (see Algorithm 1).

Example 2. Let the bead's (cart's) current position along the curve at time t be $b_t \in \mathbb{R}^3$. Its current position is along one of the curve's line segments with endpoints, for example, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ (see Figure 7).

Given the bead's (cart's) current speed v at time t, the distance (Δs) it travels in Δt seconds is $\Delta s = v\Delta t$. For example, let $\Delta s = 1.8$, then $\boldsymbol{b}_t$ must move along the curve by this amount. The line segment of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ has length 0.5, and $\boldsymbol{b}_1$ only has 0.3 left to move along the line segment. If $\Delta s <= 0.3$, we would have enough room to place it on this line segment, however Δs is larger than 0.3, so we must check the following line segments.

The next line segment from $\boldsymbol{p}_2$ to $\boldsymbol{p}_3$ has length 0.8, which means that total distance traversed by $\boldsymbol{b}_t$ along now is $0.3 + 0.8 = 1.1$, which is still less than $\Delta s = 1.8$. The bead (cart) $\boldsymbol{b}_t$ must then move to the next line segment of $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$, and so forth. Once $\boldsymbol{b}_t$ traverses to the line segment of $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$, the distance traveled is $0.3+0.8+0.5 = 1.6$, which is still less than $\Delta s = 1.8$, however the line segment has length 0.4 and $0.3 + 0.8 + 0.5 + 0.4 = 2.0$ is greater than $\Delta s = 1.8$. Therefore, $\boldsymbol{b}_t$ will be positioned on this line segment at time $t + \Delta t$ after moving a distance of Δs.

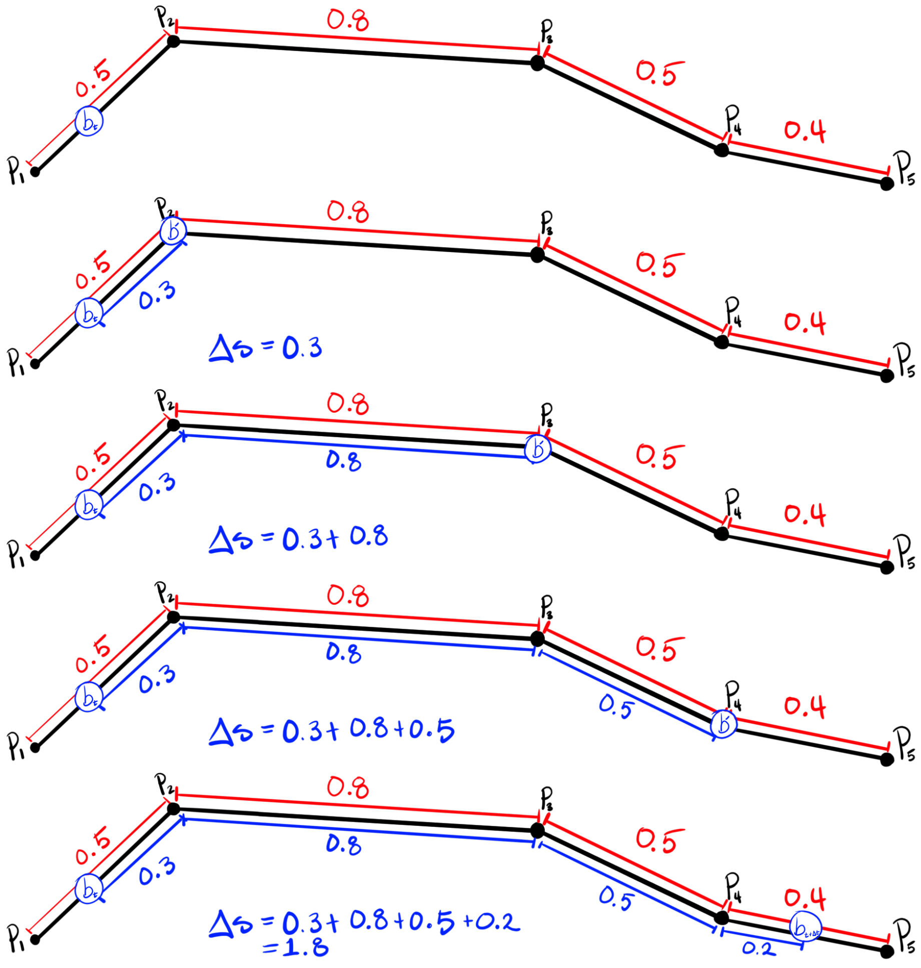


Fig. 7. Bead (cart) $\boldsymbol{b}_t$ is moved along the piece-wise liner curve, defined by $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5\}$, by a distance of Δs.

Algorithm 1 Arc length parameterization: piece-wise linear curve

```
function ArcLengthParameterization($\boldsymbol{b}_t$, $i$, $\{\boldsymbol{p}_i\}$, $\Delta s$)
    if $\|\boldsymbol{p}_{i+1} - \boldsymbol{b}_t\| > \Delta s$ then
        $\boldsymbol{b}_{t+\Delta t} \leftarrow \boldsymbol{b}_t + \Delta s \frac{\boldsymbol{p}_{i+1} - \boldsymbol{b}_t}{\|\boldsymbol{p}_{i+1} - \boldsymbol{b}_t\|}$
        return $\boldsymbol{b}_{t+\Delta t}$
    else
        $\Delta s' \leftarrow \|\boldsymbol{p}_{i+1} - \boldsymbol{b}_t\|$
        $i \leftarrow i + 1$
        while $\Delta s' + \|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\| < \Delta s$ do
            $\Delta s' \leftarrow \Delta s' + \|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|$
            $i \leftarrow i + 1$
        end while
        return $(\Delta s - \Delta s') \frac{\boldsymbol{p}_{i+1} - \boldsymbol{p}_i}{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|}$
    end if
end function
```

Moving along a parametric curve Given a parametric curve $C(u)$, where $0 \leq u \leq 1$, which defines our track, we can sample along the curve by a constant Δu, to create a piece-wise linear curve, and arc-length parameterize it as we did above. However, if you wish to arc-length parameterize the parametric curve as is, then another method is required. As described in class and in the course note, solving for the arc-length parameterization of a general curve is quite difficult (if not impossible). Instead, we precalculate *samples* of the parametric curve at constant arc lengths Δs, which are calculated by marching along the curve by constant Δu. We do this (i) calculating the total length of the curve L (see Algorithm 2), (ii) dividing the length up of the curve up into the desired number (N) of segments $\Delta s = \frac{l}{N}$, then (iii) march along curve in increments of Δs and map the associated u values with integer increments of Δs, i.e. $uValues[i] = u$ implies that arc length of $C(u)$ is $i * \Delta s$ (see Algorithm 3). Marching along the curve in increments of Δu is likely to overshoot the desired arc length of Δs (see Figure 8), so Algorithm 4 is used to refine the u value to give the appropriate arc length Δs.

Algorithm 2 Total arc length (L) of parametric $C(u)$ curve

```
function TotalArcLength($C(u)$, $\Delta u$)
    $L \leftarrow 0$
    $u' \leftarrow 0$
    $\boldsymbol{p} \leftarrow C(u')$
    while $u' \leq 1$ do
        $u' \leftarrow u' + \Delta u$
        $L \leftarrow L + \|\boldsymbol{p} - C(u')\|$
        $\boldsymbol{p} \leftarrow C(u')$
    end while
    return $L$
end function
```

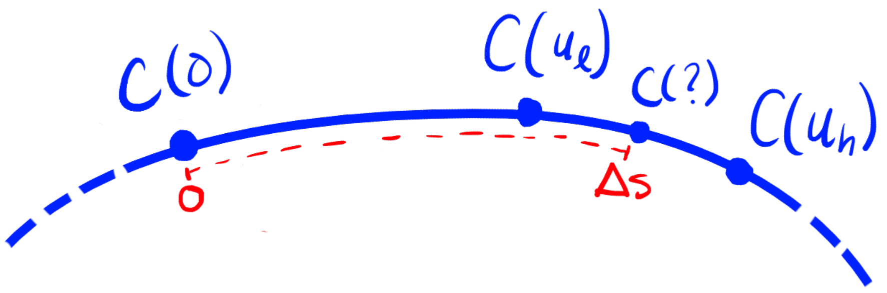


Fig. 8. The u_h value must be refined to somewhere between u_l and u_h to fine the appropriate u value that has arc length of Δs.

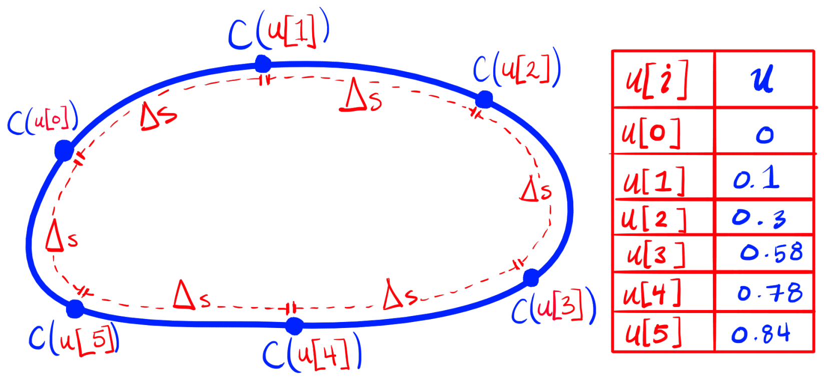
Algorithm 3 Arc length reparameterization of the parametric curve $C(u)$

```
function ArcLengthParameterization($C(u)$, $\Delta u$, $L$, $N$)
    $uValues[N]$
    $i \leftarrow 0$
    $\Delta s \leftarrow \frac{L}{N}$
    $\Delta s_{curr} \leftarrow 0$
    $u_h \leftarrow 0$
    while $u_h \leq 1$ do
        $\boldsymbol{p}_{curr} \leftarrow C(u_h)$
        $u_h \leftarrow u_h + \Delta u$
        if $\Delta s_{curr} + \|C(u_h) - \boldsymbol{p}_{curr}\| > \Delta s$ then
            $u_l \leftarrow u_h - \Delta u$
            $uValues[i] \leftarrow$ BisectionRefinementUValue($C(u)$, $u_l$, $u_h$, $\Delta s$, $\Delta s_{curr}$, $\boldsymbol{p}_{curr}$)
            $u_h \leftarrow uValues[i]$
            $i \leftarrow i + 1$
            $\Delta s_{curr} \leftarrow 0$
        else
            $\Delta s_{curr} \leftarrow \Delta s_{curr} + \|C(u_h) - \boldsymbol{p}_{curr}\|$
        end if
    end while
    return $uValues$
end function
```

Algorithm 4 Refinement of u_m value such that $\Delta s = \Delta s_{curr} + \|C(u_m) - \boldsymbol{p}_{curr}\|$

```
function BisectionRefinementUValue($C(u)$, $u_l$, $u_h$, $\Delta s$, $\Delta s_{curr}$, $\boldsymbol{p}_{curr}$)
    $iter \leftarrow 0$
    while $iter \le MAX\_ITER$ do
        $u_m \leftarrow \frac{u_h + u_l}{2}$
        $\boldsymbol{p}_m \leftarrow C(u_m)$
        $\Delta s_m \leftarrow \Delta s_{curr} + \|\boldsymbol{p}_m - \boldsymbol{p}_{curr}\|$
        if $|\Delta s_m - \Delta s| < TOLERANCE$ or $\frac{u_h - u_l}{2} < TOLERANCE$ then
            return $u_m$
        end if
        if $\Delta s_m < \Delta s$ then
            $u_l \leftarrow u_m$
        else
            $u_h \leftarrow u_m$
        end if
    end while
end function
```

Example 3. Let $L = 9$ be the length of the curve $C(u)$, and we wish to have $N = 6$ partitions[1] of $C(u)$ each with an arc length of $\Delta s = \frac{L}{N} = 1.5$. After running Algorithms 1, 3 we will have a table of u values, whereby consecutive u values correspond to consecutive points on the curve $C(u)$ with an arc length of $\Delta s = 1.5$ between them.



u[i]	u
u[0]	0
u[1]	0.1
u[2]	0.3
u[3]	0.58
u[4]	0.78
u[5]	0.84

Fig. 9. Table mapping integer multiples of distances Δs with the corresponding parameter values in $C(u)$.

Assume the bead is currently at the position $C(0)$ along the curve, meaning it is within the segment of $i_{old} = 0$. We wish to move the bead along the curve

[1] This number is purely for demonstration, in your program you should have many, *many* more partitions.

by a distance of 3.5. The curve is partitioned into equal segments of $\Delta s = 1.5$, therefore the offset to the new segment will be $i_{new} = \lfloor \frac{3.5}{1.5} \rfloor = \lfloor 2.\overline{333} \rfloor = 2$. This segment is $2 * 1.5 = 3$ away from the beginning of the curve, thus the bead will be within segment 2 by $3.5 - 3 = 0.5$. The ratio $\frac{0.5}{1.5} = 0.\overline{333}$ gives amount to linearly interpolate the u values of segment 2, i.e. the new bead position is at $C(u_{new})$, where

$$u_{new} = \left(1 - \frac{0.5}{1.5}\right) * u[2] + \left(\frac{0.5}{1.5}\right) * u[2+1] \tag{5}$$

$$u_{new} = \left(1 - 0.\overline{333}\right) * 0.3 + \left(0.\overline{333}\right) * 0.58 \tag{6}$$

$$u_{new} = 0.39\overline{333} \tag{7}$$

1.3 Calculating speed of bead (cart) from curve

As we saw above, moving the bead along the curve requires us to evaluate an instantaneous *speed* ($v \in \mathbb{R}$), and thereafter a distance along the curve s. To calculate this speed, we us single particle kinematics and the law of conservation of energy which describe the motion of a particle on a curve over time.

Equations of motions First, we establish that a particle's *position* along a curve at time t is defined by a vector $\boldsymbol{s}(t) \in \mathbb{R}^3$. We know that the change in position over some time is related to the speed of the particle. Using calculus, the instantaneous change in position is *velocity* $\boldsymbol{v}(t) \in \mathbb{R}^3$. The magnitude (length) of the vector $\boldsymbol{v}$ is the speed v of the particle. Likewise, the instantaneous change in velocity is *acceleration* $\boldsymbol{a}(t) \in \mathbb{R}^3$.

$$\boldsymbol{s} \qquad \text{position} \tag{8}$$

$$\boldsymbol{v} = \frac{d\boldsymbol{s}}{dt} \qquad \text{velocity} \tag{9}$$

$$\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt} = \frac{d^2\boldsymbol{s}}{dt^2} \qquad \text{acceleration} \tag{10}$$

We are able to evaluate the position of a particle at time t by 'adding up' the instantaneous changes in position (i.e. integrating the velocity) over the amount of time t.

$$\boldsymbol{s}(t) = \int_0^t \boldsymbol{v}(t)dt \tag{11}$$

If a particle is moving along a straight line with constant speed, the particle undergoes no acceleration (as neither the velocity's direction, nor magnitude change). Therefore, the speed v is a constant value (i.e. not a function of t), and Equation 15 evaluates a distance s traveled along the line

$$v = \text{constant} \quad \Rightarrow \quad s(t) = \int_0^t v dt = vt. \tag{12}$$

Similarly, if acceleration is constant in direction and magnitude, then

$$a = \text{constant} \quad \Rightarrow \quad v(t) = \int_0^t a dt = at. \tag{13}$$

Substituting Equation 13 into Equation 15, we get the equation of motion for a particle moving along a straight line with constant acceleration, and the duration of that motion.

$$s(t) = \int_0^t at dt \tag{14}$$

$$= \frac{1}{2}at^2 \Rightarrow t = \sqrt{\frac{2s(t)}{a}} \tag{15}$$

An example of a particle moving with constant acceleration is an object free-falling under gravity (ignoring air resistance and other external forces or impulses). The acceleration of gravity g on Earth's surface is approximately constant, $g \approx 9.81\frac{m}{s^2}$. Therefore, the equation of motions of an object free-falling under gravity is

$$s(t) = \frac{1}{2}gt^2 \Rightarrow t = \sqrt{\frac{2s(t)}{g}} \tag{16}$$

$$v(t) = gt \Rightarrow v = \sqrt{2gs(t)} \tag{17}$$

This directly relates velocity to the distance traveled along the straight line. Since the object is falling towards the earth, the distance is the difference between the object's starting height (H), and the height (h) after a duration of time t. Therefore, the general equation for velocity is $v = \sqrt{2g\,(H-h)}$. This is directly related to the law of conservation of energy, stating that energy cannot be created nor destroyed, only taking different forms and summing to a constant. In this system, there are only two forms: potential energy $E_{potential}$, and kinetic energy $E_{kinetic}$. The object at its highest point is not moving, and exhibits only potential energy (the potential of falling). Once it starts to fall, its potential energy it gradually converted into kinetic energy.

$$v = \sqrt{2g\,(H-h)} \tag{18}$$

$$\Rightarrow \frac{v^2}{2} = gH - gh \tag{19}$$

$$\Rightarrow \frac{v^2}{2} + gh = gH \tag{20}$$

$$\Rightarrow \frac{mv^2}{2} + mgh = mgH \tag{21}$$

$$\Rightarrow E_{kinetic} + E_{potential} = E_{constant} \tag{22}$$

Movement along an inclined surface Now, that we know how to describe motion of an object falling towards the Earth under a constant acceleration of gravity (g), we inspect motion along an inclined surface (see Figure 10). Let the object slide along the inclined surface (without friction) under the acceleration of gravity.

As the object is bound to the surface, and cannot fall through it, and only undergoes acceleration along the surface. With an incline of α radians, the constant acceleration of the object under gravity is decomposed into two components, acceleration along the surface a and the remaining acceleration normal to the surface (see Figure 10,left). Therefore, $a = g \sin\alpha$, and the speed v of the object after traveling a distance of $s = \frac{H}{\sin\alpha}$ along the incline is

$$\begin{aligned} v &= \sqrt{2as} && (23)\\ &= \sqrt{2\,(g\sin\alpha)\left(\frac{H}{\sin\alpha}\right)} && (24)\\ &= \sqrt{2gH} && (25)\end{aligned}$$

We see here, that the velocity of the object is independent of the angle of incline, and this holds for all instantaneous inclines, i.e. all smooth paths (see Figure 10,right). Therefore, for any sufficiently smooth curve, (as produced by

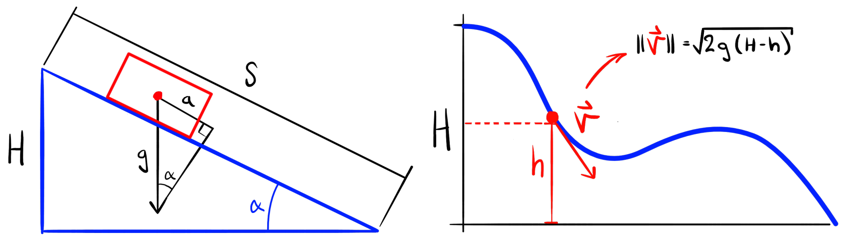


Fig. 10. Object sliding along an inclined surface under gravity g only experiences acceleration a tangent to the surface.

the method described for creating your roller coaster track), you can calculate the instantaneous speed (v) of the bead/cart depending on difference of its current height (h) from its height at the start of free fall (H). Then, calculating the distance traveled (Δs) at the speed v, given a user defined Δt, $\Delta s = v\Delta t$, you can move the bead/cart along the curve by the calculated Δs.

Three phases of the roller coaster Now that we know how to calculate the speed of the bead/cart anywhere along the curve accelerating under gravity, we can now define the three phases of the roller coaster.

Lifting stage. First, the roller coaster should start at ground level, and be "mechanically" brought to the highest point in the track H. Here, mechanically means that the bead/cart should rise to the apex at a constant and controlled speed. Move the bead/cart along the track using a minimum speed,

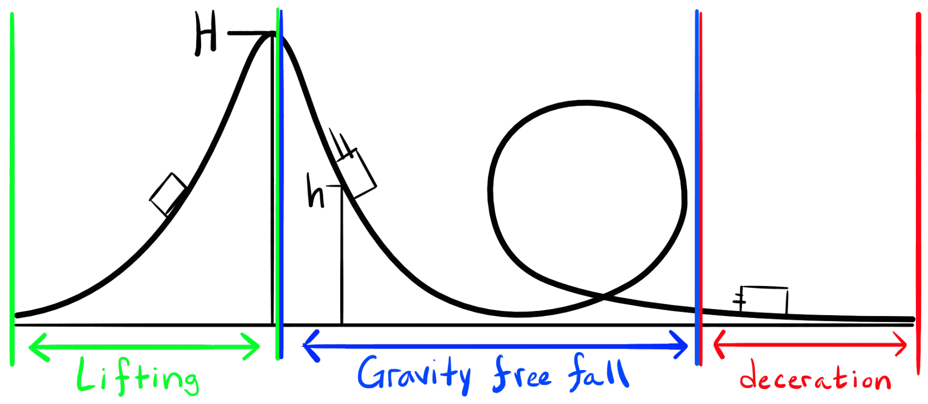
$$v = v_{min}. \tag{26}$$

Gravity free fall stage. Once the bead/cart is just past the highest point of the track, it should commence in free fall, with its speed being calculated by the law of conservation of energy,

$$v = \sqrt{2g(H-h)}. \tag{27}$$

Deceleration stage. As the bead/cart nears the end of the track, it should begin to decelerate in a smooth way. Realistically, you would want it to come to a standstill by the end of the track, simulating the passenger (un)loading. You should be able to calculate the speed of the bead/cart as it enters the deceleration stage (v_{dec}), the current distance of the bead/cart from the end of the track (d_{dec}) and the length of the deceleration stage (L_{dec}), so you can linearly decelerate to zero,

$$v = v_{dec}\frac{d_{dec}}{L_{dec}}. \tag{28}$$



1.4 Framing the curve

Circular motion Thus far, we've discussed motion along a straight line, however we need to address the accelerations felt under circular motion. When motion is restricted to a circular path with constant radius r, the arc length s can be defined as a function of the angle $\alpha \in [0, 2\pi)$, just as the total distance around the circle (circumference) is $C = 2\pi r$.

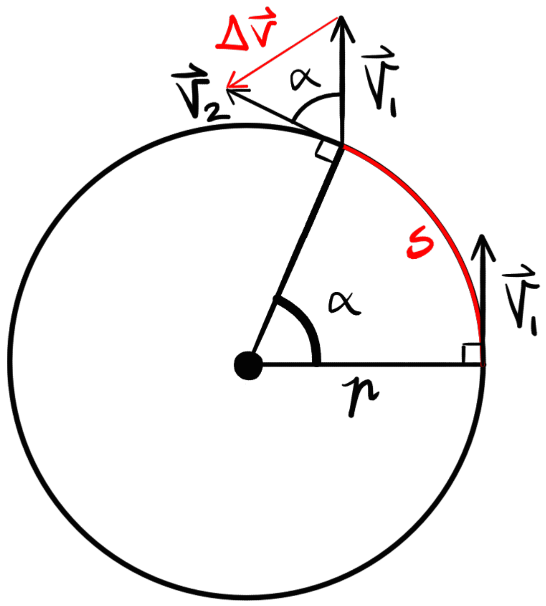


Fig. 11. We want the track to be oriented such that the passenger will only experience accelerations *normal* to the track.

$$s = \alpha r \tag{29}$$
$$\tag{30}$$

We can take the derivatives of this relation with respect to time t

$$\frac{ds}{dt} = \frac{d\alpha}{dt} r \quad (r \text{ is constant}) \tag{31}$$
$$\Rightarrow v = \omega r \quad (\omega \text{ is } \textit{angular} \text{ speed}) \tag{32}$$

where v is the *tangential* speed, which makes sense as its dependent on the distance from the center of rotation. Similarly, we define the scalar magnitudes *angular* acceleration ϵ and *tangential* acceleration $a_{\|}$.

$$\frac{d^2 s}{dt^2} = \frac{d^2 \alpha}{dt^2} r \tag{33}$$
$$\Rightarrow a_{\|} = \epsilon r \tag{34}$$

This implies that moving along a circular path with radius r and constant angular speed ω gives a constant tangential speed (magnitude) v. However, the direction of tangential motion to the circle necessarily changes over time (see $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in Figure 11). The direction of this change is always towards the center of rotation $\Delta\boldsymbol{v}$. The magnitude of this change is approximated for very small angles α (just as we calculated the arc length s or circumference of a circle),

$$\Delta v \approx v\alpha \tag{35}$$

The centripetal change of velocity over time t is centripetal acceleration $a_\perp$

$$\begin{aligned} a_\perp &= \frac{d\Delta v}{dt} && (36)\\ &= \frac{dv\omega}{dt} \quad (v \text{ and } \omega \text{ are constant}) && (37)\\ &= v\omega \quad (v = \omega r) && (38)\\ &= \omega^2 r \equiv \frac{v^2}{r} && (39) \end{aligned}$$

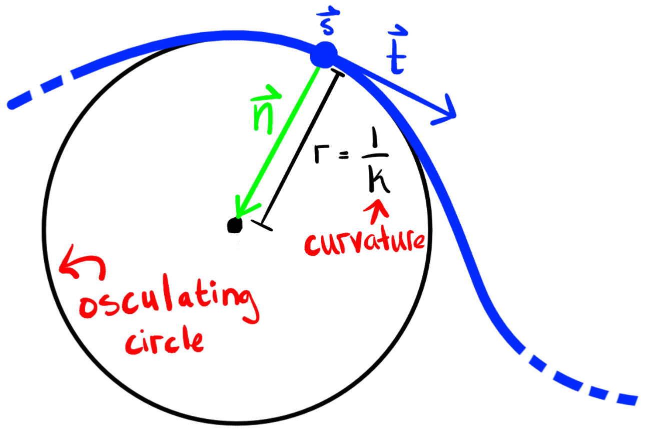


Fig. 12. Osculating circle: best fit circle that approximates a curve at a point.

Centripetal acceleration along general curves Just as we used a vector tangent to the curve to linearly approximate the local change at a point on the curve (e.g. velocity), we use an *osculating circle* to approximate the local centripetal changes (i.e. acceleration) at a point as well. The osculating circle is the circle of best fit to the curve a point on the curve (see Figure 12). Let $\boldsymbol{s}$ be a arc-length parameterized curve,

$$\begin{aligned} &\boldsymbol{s}(s) && (40)\\ \boldsymbol{t}(s) &= \frac{d\boldsymbol{s}}{ds} \quad (\boldsymbol{t} \text{ tangent to curve}) && (41)\\ \boldsymbol{n}(s) &= \frac{d\boldsymbol{t}}{ds} \quad (\boldsymbol{n} \text{ normal to curve}) && (42)\\ & && (43) \end{aligned}$$

then the first and second derivative of the curve, produce orthogonal vectors $\boldsymbol{t}$ and $\boldsymbol{n}$ where $\boldsymbol{n}$ points towards the center of the osculating circle with length r. The radius defines a special quantity called *curvature* $k = \frac{1}{r}$, which measures how much the curve locally deviates from being a straight line. Note, that as $r \to \infty$, $k \to 0$ and vise versa. It is always possible to fit an osculating circle to any point on the curve, even along straight line, however these circle are technically degenerate (i.e. having zero curvature $k = 0$). As this osculating circle locally defines the curve at any point, we use it to define the instantaneous circular motion of a particle at any point using the tools discussed above. If the particle is moving along at unit speed $v = 1$, then the osculating circle is uniquely defined and the centripetal acceleration is

$$a_\perp = \frac{v^2}{r} \tag{44}$$

$$= \frac{1}{r} \tag{45}$$

$$= k \tag{46}$$

Therefore, centripetal acceleration (in vector notation) of a particle anywhere along the curve is

$$\boldsymbol{a}_\perp = k\hat{\boldsymbol{n}} \tag{47}$$

and to calculate it, we need the direction $\hat{\boldsymbol{n}}$ and curvature k.

For parametric curves that are arc length parameterized (as above), you can approximate the centripetal acceleration directly at position t along the curve (for some small Δt)

$$\boldsymbol{a}_\perp = \frac{C(t + \Delta t) - 2C(t) + C(t - \Delta t)}{\Delta t^2} \tag{48}$$

For the piece-wise linear curve it may be desirable to take a more geometric 2 step approach (see Figure 13). The direction $\hat{\boldsymbol{n}}$ of curvature is

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{p'}_{i+1} - 2\boldsymbol{p}_i + \boldsymbol{p'}_{i-1}}{\|\boldsymbol{p'}_{i+1} - 2\boldsymbol{p}_i + \boldsymbol{p'}_{i-1}\|} \tag{49}$$

while the radius of the osculating circle requires a little geometry and algebra

$$x = \frac{1}{2}\|\boldsymbol{p'}_{i+1} - 2\boldsymbol{p}_i + \boldsymbol{p'}_{i-1}\| \tag{50}$$

$$c = \frac{1}{2}\|\boldsymbol{p'}_{i+1} - \boldsymbol{p'}_{i-1}\| \tag{51}$$

then the curvature is calculated

$$r^2 = (r - x)^2 + c^2 \tag{52}$$

$$r^2 = r^2 - 2rx + x^2 + c^2 \tag{53}$$

$$r = \frac{x^2 + c^2}{2x} \Rightarrow k = \frac{2x}{x^2 + c^2} \tag{54}$$

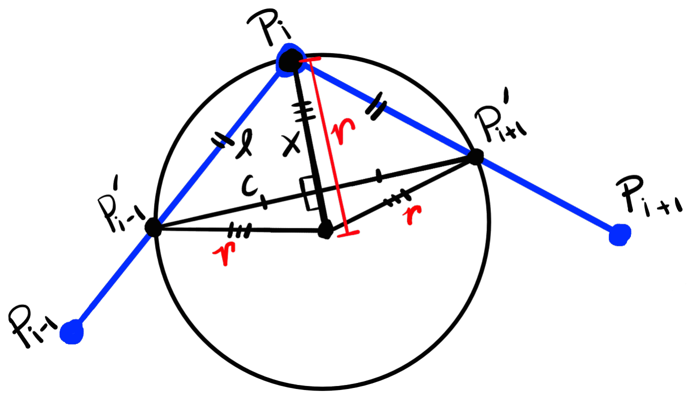


Fig. 13. Calculating the curvature $k = \frac{1}{r}$ for the osculating circle at point $\boldsymbol{p}_i$.

Pulling it all together, the centripetal acceleration for a piece wise linear curve at point $\boldsymbol{p}_i$ is

$$\begin{aligned}\boldsymbol{a}_\perp &= k\hat{\boldsymbol{n}} && (55)\\ &= \frac{2x}{x^2+c^2}\frac{\boldsymbol{p'}_{i+1} - 2\boldsymbol{p}_i + \boldsymbol{p'}_{i-1}}{\|\boldsymbol{p'}_{i+1} - 2\boldsymbol{p}_i + \boldsymbol{p'}_{i-1}\|} && (56)\\ &= \frac{1}{x^2+c^2}\left(\boldsymbol{p'}_{i+1} - 2\boldsymbol{p}_i + \boldsymbol{p'}_{i-1}\right) && (57)\end{aligned}$$

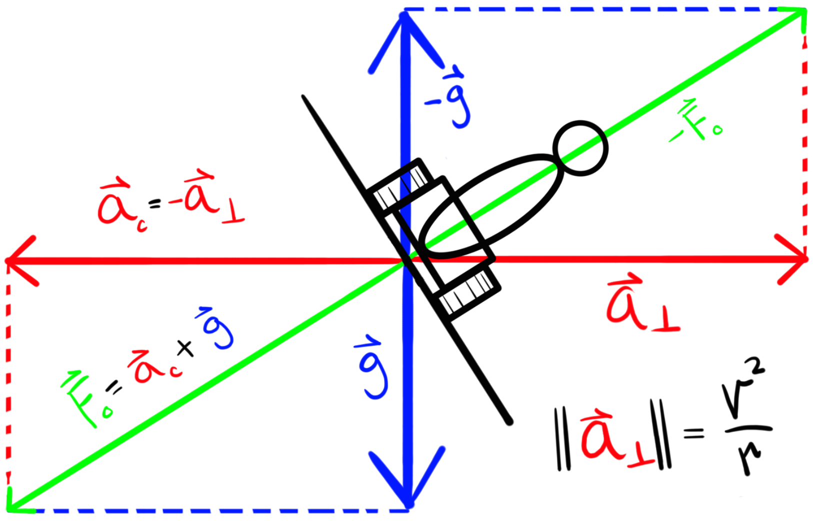
Calculating the *normal* force With the previous discussion on centripetal acceleration, we saw that at every point along the track, there is a centripetal *force* $m\boldsymbol{a}_\perp$ acting on the passenger as they travel in a cart around a sharp bend (see Figure). This force is directed inwards to the center of the osculating circle that best fits the carts local position on the track. This is so even when the track is straight where $k \to \infty$, then $\|m\boldsymbol{a}_\perp\| = \|k\hat{\boldsymbol{n}}\| \to 0$.

However, if this centripetal force is directed *towards* the center of rotation, why is it that when you are in a car that takes a sharp turn do you feel a force pulling you *away* from the center? This apparent outward force is described by Newton's Laws of Motion. Newton's First Law states that

"A body at rest will remain at rest, and a body in motion will remain in motion unless it is acted upon by an external force."

For now, this means that an object moving in straight line will continue in a straight line unless an outside force causes it to speed up, slow down or alter is direction. As we saw, the instantaneous velocity of an object bound to a circular motion is tangent to the circular path. Without centripetal force acting on the object towards the center of rotation, the object would leave the circular path and carry on in the straight line defined by the velocity and it current position. This continuous redirection of the velocity towards the center of curvature is experienced as an outward force equal in magnitude, and opposite in direction to the centripetal force. This *pseudo*-force is centrifugal force $m\boldsymbol{a}_c = -m\boldsymbol{a}_\perp$. Real or not, this is the exactly the 'experienced' force by the user that we wish to align with the passenger's head and feet so that the forces are felt directly through the feet (i.e. there are no tangential forces acting on the passenger) as they go around a sharp turn. However, in our simulation, centrifugal (centripetal) force is not the only force at play: gravity is the constant force $m\boldsymbol{g} = m\,(0, -9.80665, 0)$ towards the ground that is added to the centrifugal force, and the total force experienced by the passenger is $\boldsymbol{F}_o$ (see Figure 14). To ensure this force is only felt along the line passing through the passenger's head and feet, we must orient the passenger's 'up' or normal direction $\boldsymbol{N}$ exactly opposite to this force

$$\boldsymbol{N} = -\boldsymbol{F}_o \tag{58}$$
$$= -\left(\boldsymbol{a}_c + \boldsymbol{g}\right) \tag{59}$$
$$= \boldsymbol{a}_\perp - \boldsymbol{g} \tag{60}$$



$-\vec{g}$
$-\vec{F}_o$
$\vec{a}_c = -\vec{a}_\perp$
$\vec{a}_\perp$
$\vec{F}_o = \vec{a}_c + \vec{g}$
$\vec{g}$
$\|\vec{a}_\perp\| = \frac{v^2}{r}$

Fig. 14. We want the track to be oriented such that the passenger will only experience accelerations *normal* to the track (i.e. $\boldsymbol{N} = -\boldsymbol{F}_o$).

2 Putting it all together: Placing your cart and framing the curve

As discussed in Tutorials 1 and 2 (see Math review notes), to create a coordinate transform from one coordinate space to another, you require a set of mutually orthogonal, unit length vectors, e.g. the standard basis

$$\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\} = \left\{ \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix} \right\} \tag{61}$$

The linear transformation describing this coordinate system change is the matrix with columns belonging to this set

$$\begin{bmatrix} | & | & | \\ \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 \\ | & | & | \end{bmatrix} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \tag{62}$$

Obviously this is just the *Identity* transformation, which when multiplied by a point $\boldsymbol{p} = (x_p, y_p, z_p)$ transforms it into the same coordinate space.

$$\begin{bmatrix} | & | & | \\ \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix}x_p\\y_p\\z_p\end{bmatrix} = x_p \begin{bmatrix}1\\0\\0\end{bmatrix} + y_p \begin{bmatrix}0\\1\\0\end{bmatrix} + z_p \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}x_p\\y_p\\z_p\end{bmatrix} \tag{63}$$

If we want translate where the origin of this new coordinate system to e.g. $\boldsymbol{o} = (x_o, y_o, z_o)$, we must apply an *affine* transformation to the point defined in homogeneous coordinates

$$\begin{bmatrix}1&0&0&x_o\\0&1&0&y_o\\0&0&1&z_o\\0&0&0&1\end{bmatrix} \begin{bmatrix}x_p\\y_p\\z_p\\1\end{bmatrix} = x_p \begin{bmatrix}1\\0\\0\\0\end{bmatrix} + y_p \begin{bmatrix}0\\1\\0\\0\end{bmatrix} + z_p \begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \begin{bmatrix}x_o\\y_o\\z_o\\1\end{bmatrix} = \begin{bmatrix}x_p + x_o\\y_p + y_o\\z_p + z_o\\1\end{bmatrix} \tag{64}$$

However, with another set of mutually orthogonal, unit length vectors, this point would be transformed into the coordinate space defined by the set of vectors.

Let's assume that we have a polygonized *cart* mesh object with vertices defined in its own object space. We wish to reposition this cart onto the track, and orient it so that it follows the track and its 'up' direction points opposite to the computed centrifugal and gravitational forces. Let's the new position of the cart (at time t) along the track is $\boldsymbol{p} = (x_p, y_p, z_p)$. Then we know that the *affine* transformation (model matrix) M looks something like the following

$$M = \begin{bmatrix}?&?&?&x_p\\?&?&?&y_p\\?&?&?&z_p\\0&0&0&1\end{bmatrix} \tag{65}$$

To reorient the cart, we transform it so that its 'up' or y-axis points in the direction of the computed normal direction of the track $\boldsymbol{N}$

$$\boldsymbol{N} = \frac{v^2}{r}\hat{\boldsymbol{n}} + \begin{bmatrix}0\\g\\0\end{bmatrix} \tag{66}$$

where v is the computed *speed* of the cart at position $\boldsymbol{p}$ along the track, and r is the computed *radius* of the osculating circle at the same point (remember that if you calculate curvature k instead, then $r = \frac{1}{k}$). This vector points in the correct direction of 'up' for the cart, but its magnitude is not unit length, i.e. $\|\boldsymbol{N}\| \neq 1$. So if we normalize $\boldsymbol{N}$, we get the unit length vector $\hat{\boldsymbol{N}} = \frac{\boldsymbol{N}}{\|\boldsymbol{N}\|} = (x_{\hat{\boldsymbol{N}}}, y_{\hat{\boldsymbol{N}}}, z_{\hat{\boldsymbol{N}}})$, which is suitable for our the y-axis of the new coordinate system

$$M = \begin{bmatrix} ? & x_{\hat{\boldsymbol{N}}} & ? & x_p \\ ? & y_{\hat{\boldsymbol{N}}} & ? & y_p \\ ? & z_{\hat{\boldsymbol{N}}} & ? & z_p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{67}$$

Depending on how you design choice, either the x-axis or z-axis will define the 'forward' direction of your cart in object space. This 'forward' direction should naturally follow the curve/track, implying that it should point in the direction of the *tangent* vector to the curve at position $\boldsymbol{p}$. Depending on your choice of defining the curve, the tangent can be computed in a number of ways.

If you are using a parametric definition of the curve, you can approximate the tangent in a similar manner to how we calculated the curvature, with a *forward* finite difference

$$\boldsymbol{T} = \frac{C(t + \Delta t) - C(t)}{\Delta t} \tag{68}$$

If you using a piece-wise linear curve for your track, can take similar finite differences around a point $\boldsymbol{p_i}$

$$\boldsymbol{T} = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i \quad \text{or} \quad \boldsymbol{T} = \boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1} \tag{69}$$

Either, approximate the local tangent around $\boldsymbol{p_i}$ if the distance to its neighboring points is sufficiently small. Again, this vector (probably) points in the correct direction, but needs to be normalized, $\hat{\boldsymbol{T}}_{tmp} = \frac{\boldsymbol{T}}{\|\boldsymbol{T}\|}$. We preface this with 'probably' as this tangent is an approximation, dependent on your sampling of your curve. However, fret not, $\hat{\boldsymbol{T}}_{tmp}$ is sufficiently different from $\hat{\boldsymbol{N}}$ to create a set of mutually orthogonal, unit length vectors from the two.

Let $\boldsymbol{B}$ be the *bi-normal* to the (temporary) tangent vector $\hat{\boldsymbol{T}}_{tmp}$ and normal vector $\hat{\boldsymbol{N}}$ defined as $\boldsymbol{B} = \hat{\boldsymbol{T}}_{tmp} \times \hat{\boldsymbol{N}}$ (the cross product). $\boldsymbol{B}$ is now guaranteed to be orthogonal to the plane defined by vectors $\hat{\boldsymbol{T}}_{tmp}$ and $\hat{\boldsymbol{N}}$, but its length, again must be normalized, $\hat{\boldsymbol{B}} = \frac{\boldsymbol{B}}{\|\boldsymbol{B}\|}$. Now with $\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{N}}$ being mutually orthogonal, and unit length, all that is left is to correct the tangent vector. $\boldsymbol{T} = \hat{\boldsymbol{N}} \times \hat{\boldsymbol{B}}$ (remember right hand rule). Normalizing as before, and packing into the coordinate transformation (model matrix) M, assuming that the cart is deigned with the 'forward' direction pointing along the z-axis, then

$$M = \begin{bmatrix} x_{\hat{\boldsymbol{B}}} & x_{\hat{\boldsymbol{N}}} & x_{\hat{\boldsymbol{T}}} & x_p \\ y_{\hat{\boldsymbol{B}}} & y_{\hat{\boldsymbol{N}}} & y_{\hat{\boldsymbol{T}}} & y_p \\ z_{\hat{\boldsymbol{B}}} & z_{\hat{\boldsymbol{N}}} & z_{\hat{\boldsymbol{T}}} & z_p \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ \hat{\boldsymbol{B}} & \hat{\boldsymbol{N}} & \hat{\boldsymbol{T}} & \boldsymbol{p} \\ | & | & | & | \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{70}$$

This transformation will move the cart to the appropriate point on the track, and orient it accordingly. At this point, to frame the curve (i.e. draw your track), you actually only need the bi-normal vector $\hat{\boldsymbol{B}}$.