Maverick

An Intamin "Blitz Coaster"

Maverick - Cedar Point

Type	Steel - Sit-Down
Status	Operating
Opened	2007
Mfr.	Intamin AG
Height	105 ft
Length	4450 ft
Speed	70 mph

Maverick

Maverick

Heartlining; Track Shaping; FVDs and Designing with Newton

Jordan Zink

Carnegie Mellon University

jzink@andrew.cmu.edu

98-186: Spring 2015

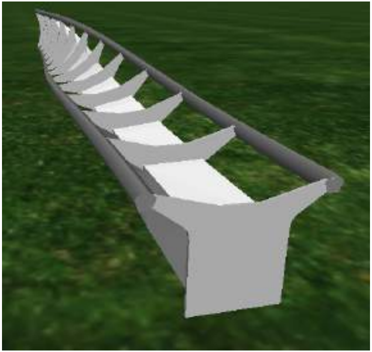
Outline

- Heartlining
- Track Shaping
  - "I know what element I want; how do I form it?"
  - Classic Track Shaping - Equations and a heavy reliance on circles
  - Modern Track Shaping - Exactly what forces you want with calculus (FVDs)

Heartlining

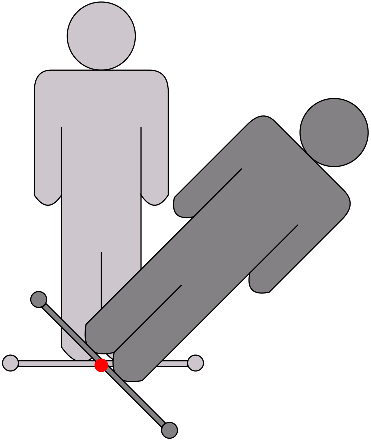
Heartlining

- Banking is important to transfer uncomfortable lateral G's to vertical
- Question: How to rotate track?
- Easiest to rotate around spine or between rails
    - On woodies, whatever keeps the support structure the same



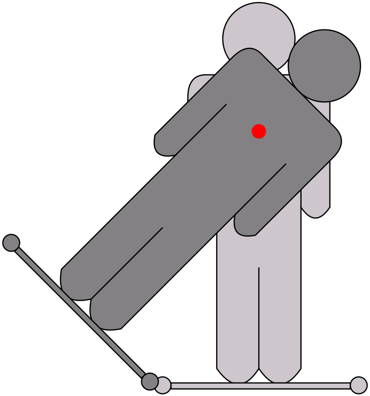
Heartlining

- Why might this be bad?
- What areas of the body are most sensitive
    - Head (where all inertial sensing is)
    - Center of gravity
    - Definitely not the feet
- Which parts travel most in this arrangement?

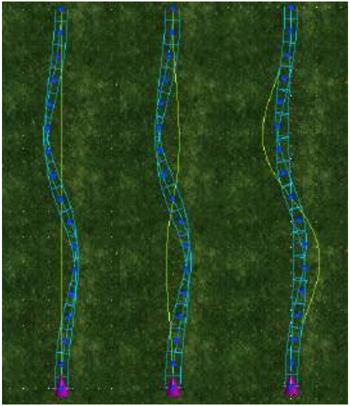


Heartlining

- Idea attributed to Stengel in the 70s
- Rotate track around rider center of gravity (the heart)
- Makes quick transitions more comfortable
- Track appears to swing under riders



Heartlining



- Heartline varies based on seating arrangement
  - Example left is same roll for sit down, stand up, and inverted
  - About 1m for sit down
- Track is designed without heartline and then it's added later
- All steel coasters today use heartlining
- Wooden coasters sorta do
  - Want feeling of being thrown around
  - Still helps smooth out transitions

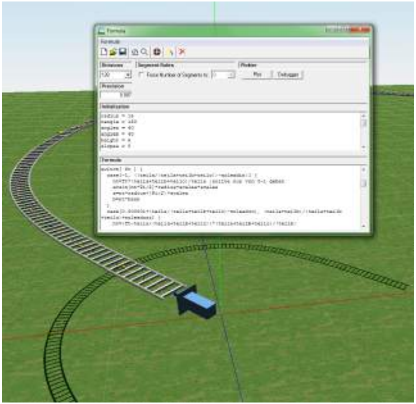
Classic Track Shaping

Representing the Track

- Get ready for equations!
- We will use parametric equations
  - We have a variable t which varies over an interval (we use 0 to 1)
  - All other coordinates are written in terms of t
  - Allows for complex shapes
- A single element, like a turn, might be one equation
- The entire coaster is a giant piece-wise equation
  - There are other ways to represent the track we will discuss later

No Limits Elementary

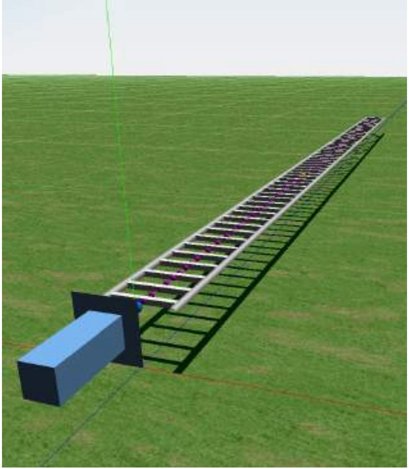
- Free tool to make track segments based on parametric equations
- Two parts to equation: Initialization and Formula
  - In formula, t is already set for you
  - You need to set x, y, and z (and optionally b for banking)
- Kinda hard to do really complicated things in
- Also Windows only and a bit out-dated



Example - Line

length = 30

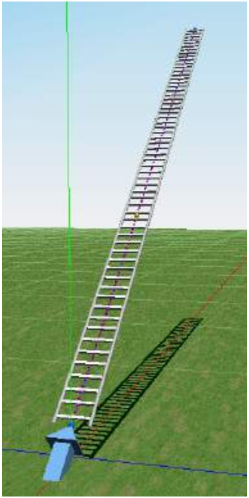
x = length * t
y = 1
z = 0



Example - Sloped Line

length = 30
angle = 40
angleInRads = angle * Pi / 180
dx = 30 * cos[angleInRads]
dy = 30 * sin[angleInRads]

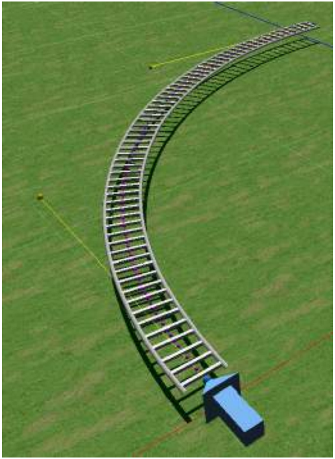
x = dx * t
y = 1 + dy * t
z = 0



Example - Flat Turn

radius = 20
angle = 90

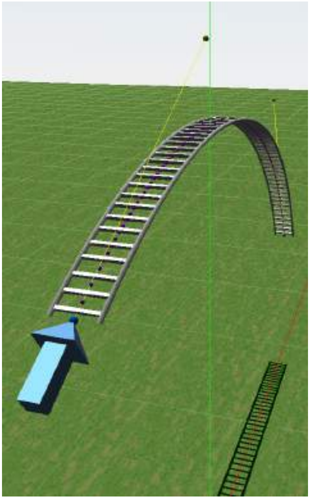
curAngle = (angle * Pi / 180) * t
x = radius * cos[curAngle]
y = 1
z = radius * sin[curAngle]



Example - Hill Crest

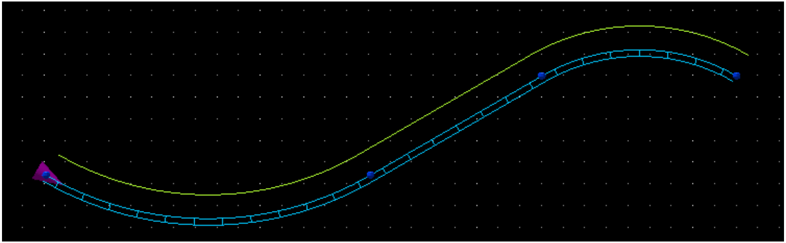
radius = 20
angleIn = 30
angleOut = -60
angle = angleOut - angleIn

curAngle = (angle * t + angleIn) * Pi / 180
x = radius * cos[curAngle]
y = radius * sin[curAngle]
z = 0



Lines and Semi-Cirlces

- Can do quite a bit with just lines and semi circles
- Example - Simple hills (I call them “Magnum Hills”)
  - Circle radius based on desired forces
  - Line length based on desired height



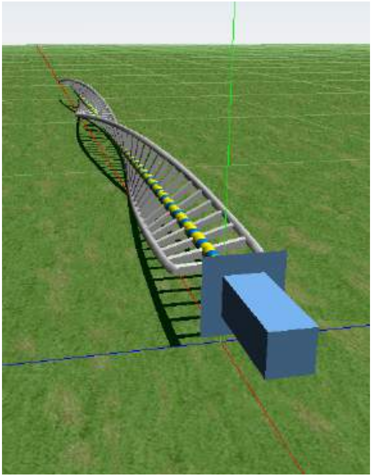
Example - Heartline Roll

- A flat 360 degree roll
- The riders' heartline is a straight line
    - So track will appear to curve around
- If not heartlined, would be an in-line twist
    - Rare, and usually uncomfortable

Heartline Roll - Without Heartline

length = 30

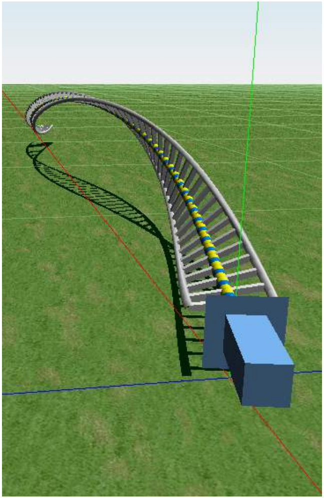
x = length * t
y = 1
z = 0
b = 2 * Pi * t



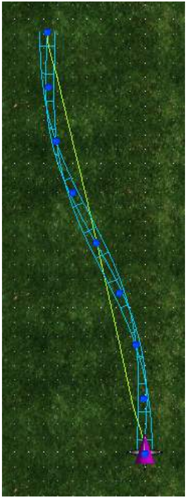
Heartline Roll - With Heartline

length = 30
hline = 1.1

b = 2 * Pi * t
x = length * t
y = 1 + **hline** - **hline * cos[b]**
z = **hline * sin[b]**



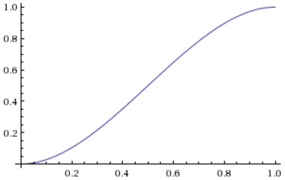
Heartline Roll - Not Quite Right



- The entry is angled a bit
- Results in roll shifting to the side
  - But the heartline is straight
- Must transition into banking change
  - Will do this with transition functions

Transition Functions

- A function $f(x)$ that satisfies the following conditions
  - $f(0) = 0$
  - $f(1) = 1$
  - $f'(0) = f'(1) = 0$
- Allows smooth motion from 0 to 1
- Can vary how steep middle of curve is for snappier transitions
  - Could also not be symmetric, but we won't go into that



Transition Functions - Some Examples

Bernstein

$$f(x) = 3x^2 - 2x^3$$

Quintic

$$f(x) = 6x^5 - 15x^4 + 10x^3$$

Using Sin

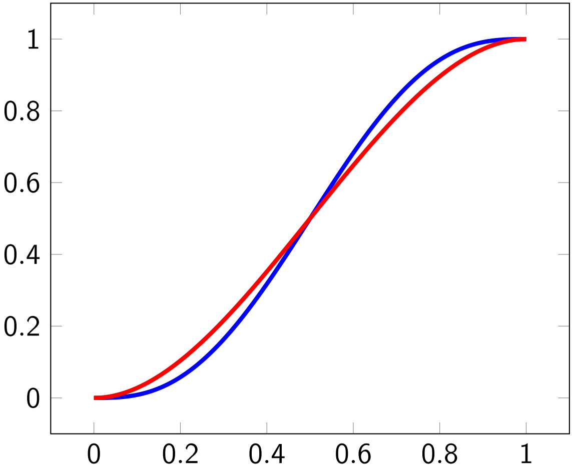
$$f(x) = \frac{1 + \sin\left(\pi t + \frac{\pi}{2}\right)}{2}$$

Using Cos

$$f(x) = \frac{1 - \cos(\pi t)}{2}$$

Transition Functions - Some Examples

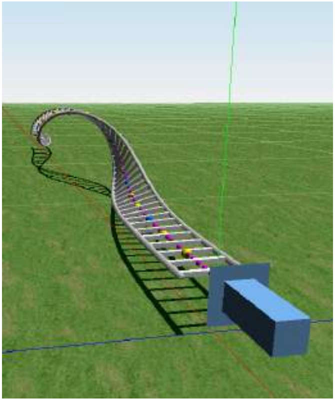
Red is Bernstein, Blue is Quintic



Heartline Roll - With Transition

length = 30
hline = 1.1

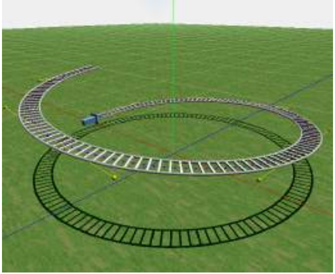
trans = 3*Pwr[t,2] - 2*Pwr[t,3]
b = 2 * Pi * **trans**
x = length * t
y = 1 + hline - hline * cos[b]
z = hline * sin[b]



Helix - Constant Slope

radius = 10
angle = 360
height = 5

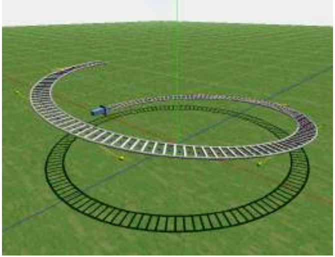
curAngle = (angle * Pi / 180) * t
x = radius * cos[curAngle]
y = 1 + **height * t**
z = radius * sin[curAngle]



Helix - Flat Entry and Exit

radius = 10
angle = 360
height = 5

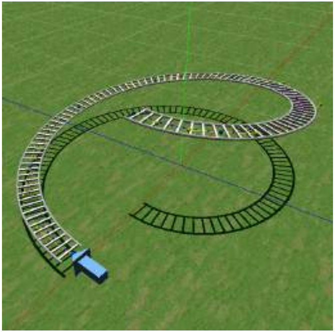
trans = 3*Pwr[t,2] - 2*Pwr[t,3]
curAngle = (angle * Pi / 180) * t
x = radius * cos[curAngle]
y = 1 + height * **trans**
z = radius * sin[curAngle]



Helix - Changing Radius (Incorrect)

rIn = 10
rOut = 5
angle = 360
height = 5

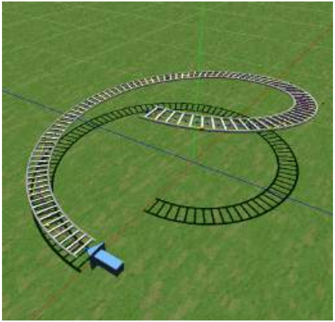
trans = 3*Pwr[t,2] - 2*Pwr[t,3]
curAngle = (angle * Pi / 180) * t
curRadius = (rOut - rIn) * t + rIn
x = curRadius * cos[curAngle]
y = 1 + height * trans
z = curRadius * sin[curAngle]



Helix - Changing Radius (Correct)

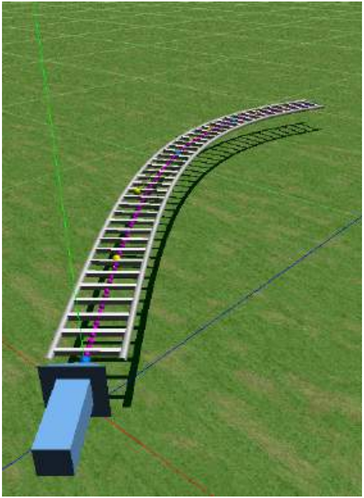
rIn = 10
rOut = 5
angle = 360
height = 5

trans = 3*Pwr[t,2] - 2*Pwr[t,3]
curAngle = (angle * Pi / 180) * t
curRadius = (rOut - rIn) * **trans** + rIn
x = curRadius * cos[curAngle]
y = 1 + height * trans
z = curRadius * sin[curAngle]



Lead In and Lead Outs

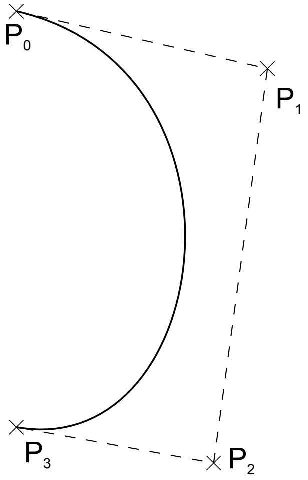
- Particular important to turns and hills
- Need to transition from no curve to curve
  - Go from an infinite radius (straight) to desired radius
  - Turns out to be a little tricky
- We won't cover the math



Two Brief Asides

Bezier Curves

- Alternative track representation
- A series of points (P_0, P_3) with control points (P_1, P_2)
  - Curve goes through main points
  - Curve is tangent to line formed by point and its control point
  - Smooth* in-between
- No Limits uses these
  - Elementary approximates the equations into bezier curves



Bezier Curves

Well, turns out bezier curves are just a graphical description of a parametric cubic

$$\begin{bmatrix} t^3 & t^2 & t & t \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Plug in the points, multiply out, and you have your curve

Heartlining

- We explicitly added the heartlining in the heartline roll
  - Was pretty simple
- Can be much more complicated when not a line
- Normally, designers ignore heartlining
  - Treat track as the curve of the heartline
  - Apply it later

Heartlining

Just use this "simple" set of transformation matrices

$$T = T_W R_H R_P R_B T_H$$

where

$$\begin{aligned} T_W &= \text{Equations for track without heartline (all axes)} \\ R_H &= \text{Rotation for heading (y-axis)} \\ R_P &= \text{Rotation for pitch (z-axis)} \\ R_B &= \text{Rotation for banking (x-axis)} \\ T_H &= \text{Translation for heartline (y-axis)} \end{aligned}$$

Yeah, you don't need to know any of that...

Modern Track Shaping

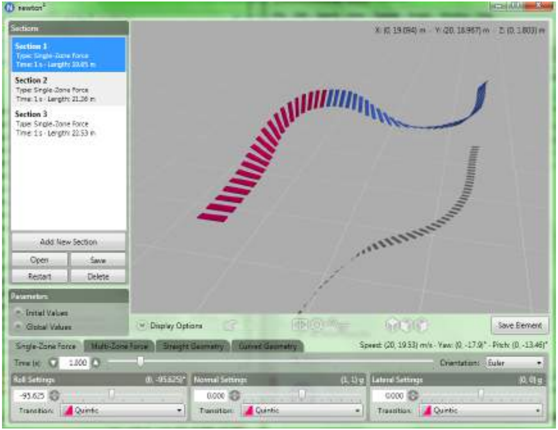
Downsides of Classic Track Shaping

- Lots of force and energy calculations
  - Often times need to repeat equations to get what you want
  - Can quickly get complicated
- Hard to get exactly what you want
  - Forces are usually the main goal
  - The exact shape is usually flexible
  - e.g. turns don't need to be perfect circles
- Smooth transitions difficult
  - What we discussed actually isn't good enough in some cases

FVDs

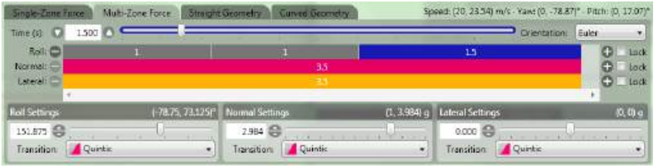
- Ideally, we would just want to specify forces the rider feels, in addition to banking
- Force Vector Design, or FVDs, allow us to do just that
    - Designer only specifies vertical force, banking, and optionally lateral force
    - Complicated calculus turns these into track equations
    - No need to mess with custom equations!
- Can make some really cool elements without lateral forces
    - Many would be impossible with classic track shaping

Newton



We will use Newton, a easy to use implementation of FVDs

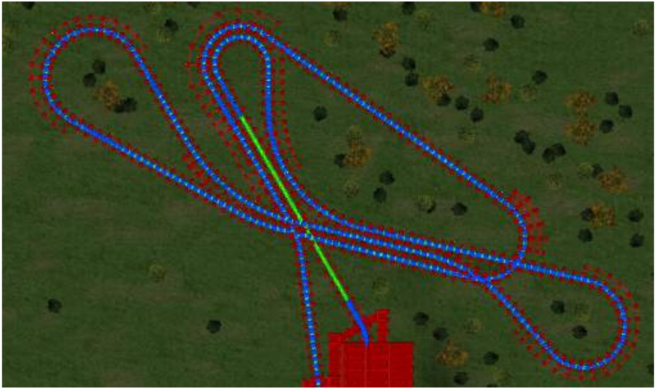
Newton



- Enter banking and forces as zones
    - Goes from the current value to a target value
    - Utilizes a transition
- Zones can overlap with each other

Examples of Using Newton

Issues with Newton



What about this track would be difficult in Newton?

Issues with Newton

- Hard to get exact geometries
    - Sometimes is a good thing
    - Some woodies do it to be artistically pleasing
- Theoretically, one could define heading instead of banking and things would still work
- Actually, as long as three of these are defined, I believe you can define the track
    - Banking, heading, pitch, vertical force, lateral force

FVDs in Real Life

- FVDs first showed up around 2006 in the No Limits community
- In 2011, Extreme Rusher at Happy Valley showed evidence of the technique
  - S&S launch coaster
- Most Rocky Mountain Construction coasters appear to use FVDs
  - e.g. wave turn on Outlaw Run

Next Week

- Next week will be a Newton activity
  - Different Location: WEH 5202 (Windows Cluster)
  - Different Time: 4:30pm
  - Email me if you have a conflict
- Back here in two weeks for Amusement Parks Part 1 (History, Chains, and Design)