CS420 Assignment 2 Hints
Simulating a Roller Coaster
Starter Code

- The Mac and Linux starter codes require the "pic" image library in the first assignment.
- "pic" directory has to be located one level above your assignment 2 directory
Spline

Catmull-Rom Spline Matrix

\[
\begin{bmatrix}
  x & y & z \\
\end{bmatrix} =
\begin{bmatrix}
  u^3 & u^2 & u & 1 \\
\end{bmatrix}
\begin{bmatrix}
  -s & 2 - s & s - 2 & s \\
  2s & s - 3 & 3 - 2s & -s \\
  -s & 0 & s & 0 \\
  0 & 1 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  x_1 & y_1 & z_1 \\
  x_2 & y_2 & z_2 \\
  x_3 & y_3 & z_3 \\
  x_4 & y_4 & z_4 \\
\end{bmatrix}
\]

basis control matrix
Display splines in OpenGL

- **Method 1(basic): brute force**
  - $u = 0, 0.01, 0.02, 0.03, \ldots, 1$
  - Fixed even steps of $u$ does not mean even steps of $x$
  - Line length will vary over the curve

- **Method 2(extra): recursive subdivision**
Recursive Subdivision

- **Method 2: recursive subdivision** - vary step size to draw short lines

  Subdivide(u0, u1, maxlinelength)
  umid = (u0 + u1)/2
  x0 = F(u0)
  x1 = F(u1)
  if |x1 - x0| > maxlinelength
      Subdivide(u0, umid, maxlinelength)
      Subdivide(umid, u1, maxlinelength)
  else drawline(x0, x1)

- **Variant on Method 2** - subdivide based on curvature
  - replace condition in “if” statement with straightness criterion
  - draws fewer lines in flatter regions of the curve
Ground

- A large plane
- Texture-mapped

“pic” library has some functions to load texture images
Sky

- Texture-mapped
- Basic: Use a cube
  - Not realistic
- Extra: e.g. A dome

Move Camera

- Camera points along the tangent vector of the curve
  - \( t(u) = \text{unit}(p'(u)) = \text{unit}([3u^2 2u 1 0] M C) \).
- How to find an “up” vector for the camera?
  - Camera orientation should be continuous
Coordinate Transitions

- Establish a local coordinate system for each point on the curve
  - T, N, B
- Initialization: T0, N0, B0
  - u = 0 => T0
  - Pick an arbitrary V. N0 = unit(T0 x V) and B0 = unit(T0 x N0). This guarantees T0, N0, B0 perpendicular to each other.
Coordinate Transitions

- Next view: T1, N1, B1
  - Move u or x ahead, compute T1 based on new u
  - N1 = unit(B0 x T1) and B1 = unit(T1 x N1)

- T2, N2, B2, ...

- Make camera “up” vector to be N or B

- Guarantee camera orientation changes continuously
Speed of the camera

- Basic: fixed steps of \( u \)
- Extra: realistic in terms of gravity

- Derive the equation above for extra credit

\[
\begin{align*}
    u_{\text{new}} &= u_{\text{current}} + (\Delta t) \frac{\sqrt{2g(h_{\text{max}} - h)}}{\frac{dp}{du}}
\end{align*}
\]
Rail Cross-section

Other shapes are also allowed. e.g. A circle or ellipse.
Animation Requirement

- Do not exceed 1000 frames
- Frame rate: 15fps
- Start as soon as possible
- Have fun!
Thanks!