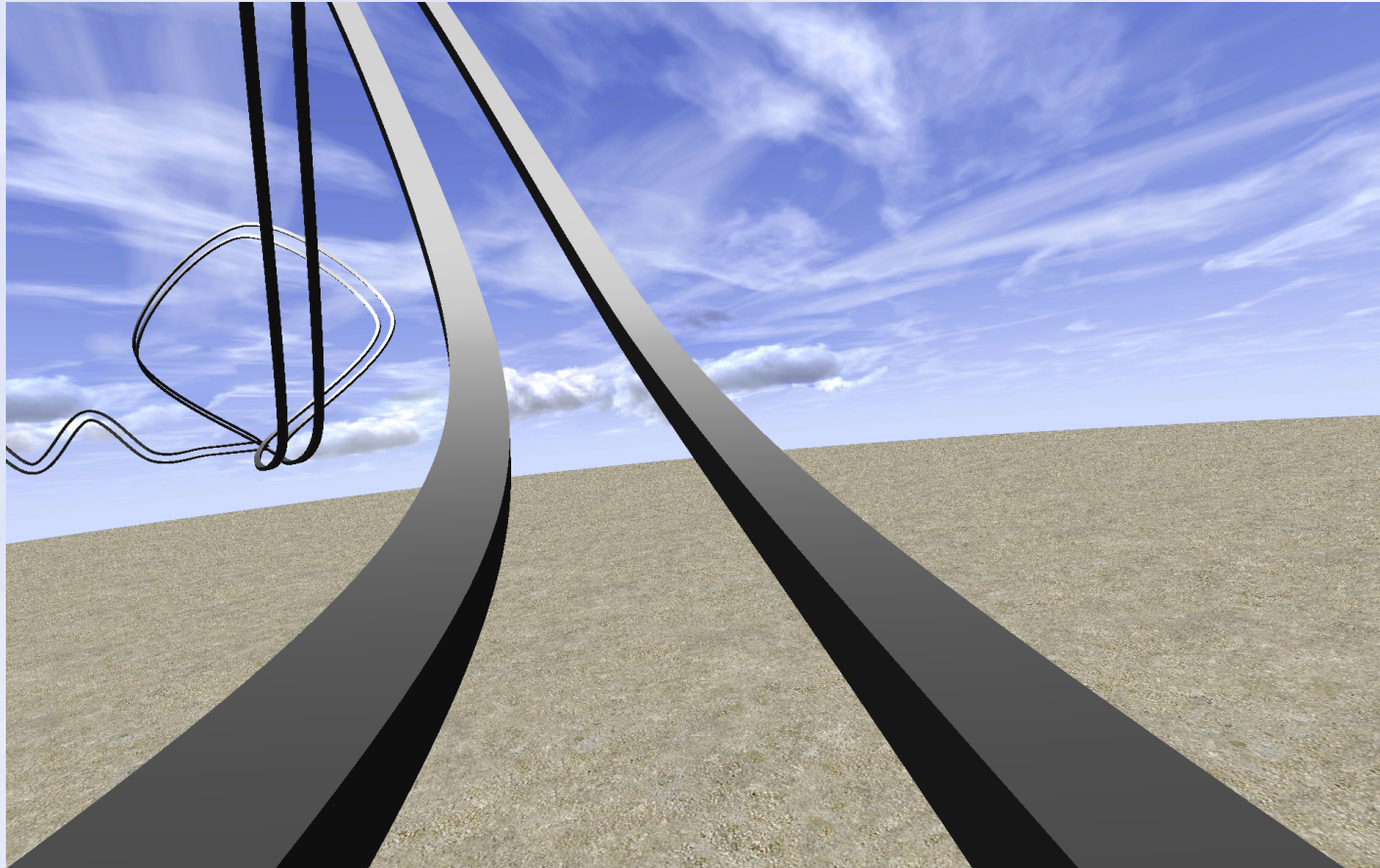


CS420 Assignment 2 Hints

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

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

basis

control matrix

Display splines in OpenGL

- Method 1(basic): brute force
 - $u = 0, 0.01, 0.02, 0.03, \dots, 1$
 - 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

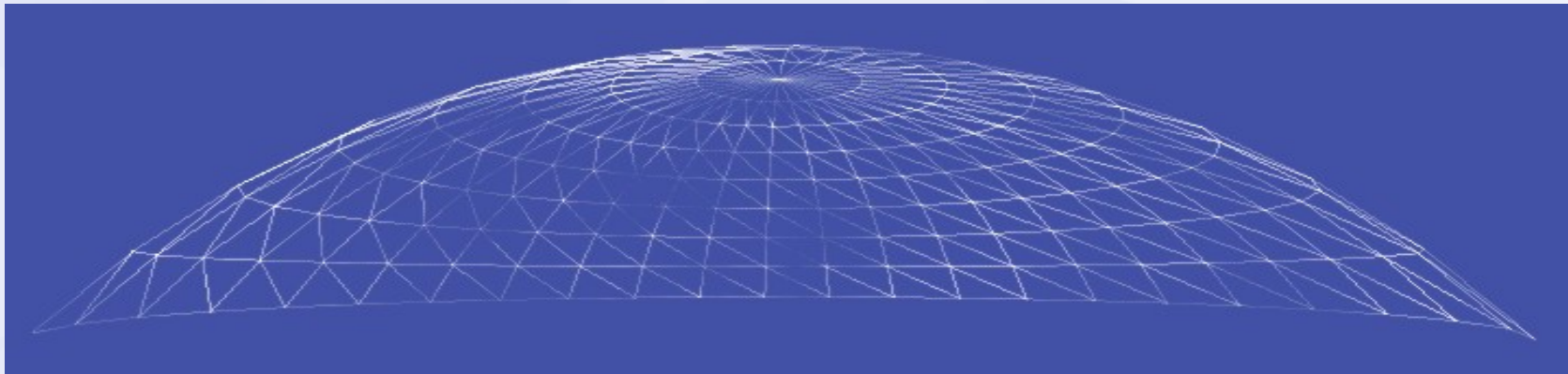
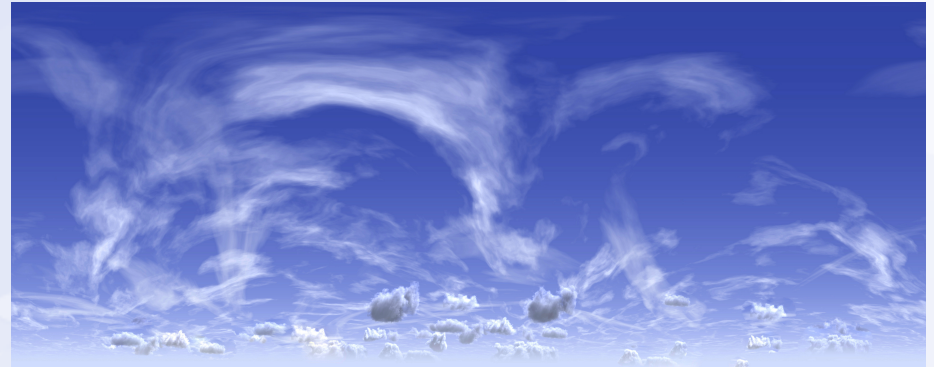
Ground

- 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



http://www.flipcode.com/archives/Sky_Domes.shtml

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: T_0, N_0, B_0
 - $u = 0 \Rightarrow T_0$
 - Pick an arbitrary V . $N_0 = \text{unit}(T_0 \times V)$ and $B_0 = \text{unit}(T_0 \times N_0)$. This guarantees T_0, N_0, B_0 perpendicular to each other.

Coordinate Transitions

- Next view: T1, N1, B1
 - Move u or x ahead, compute T1 based on new u
 - $N1 = \text{unit}(B0 \times T1)$ and $B1 = \text{unit}(T1 \times N1)$
- T2, N2, B2, ...
- Make camera “up” vector to be N or B
- Guarantee camera orientation changes continuously

Speed of the camera

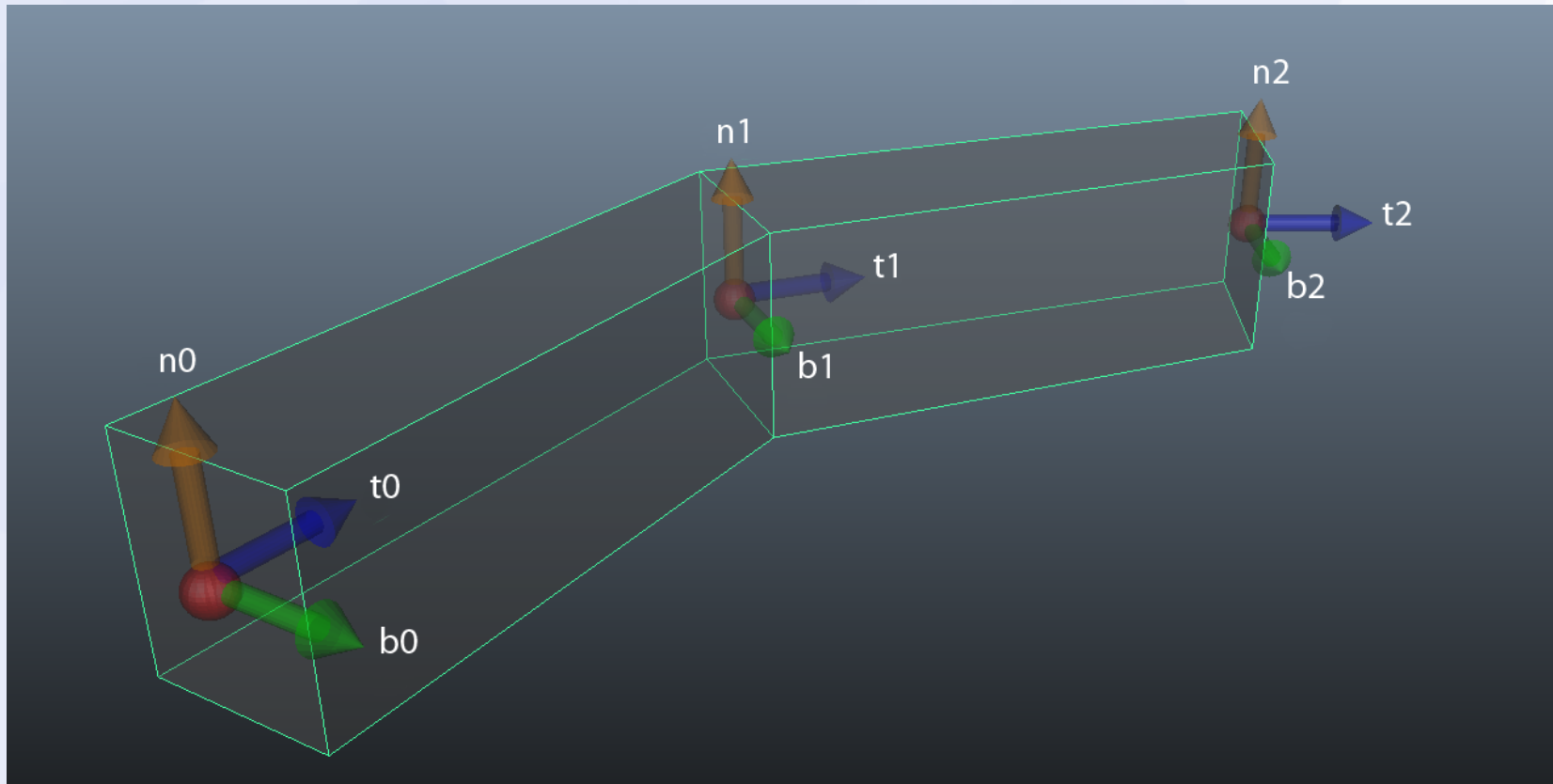
- Basic: fixed steps of u
- Extra: realistic in terms of gravity

-

$$u_{new} = u_{current} + (\Delta t) \frac{\sqrt{2g(h_{max} - h)}}{\left\| \frac{dp}{du} \right\|}$$

- Derive the equation above for extra credit

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!