CSCI 420 ASSIGNMENT 2 README FILE

Subject : CSCI420 - Computer Graphics
Assignment 2 : Simulating a Roller Coaster
Author : Stuti R. Rastogi
USC ID : 8006687469

Description : In this assignment, we use Catmull-Rom splines along with OpenGL texture mapping to create a roller coaster simulation.

Core Credit Features:

1. Uses OpenGL core profile, version 3.2 or higher - YES

2. Completed all Levels:
   Level 1: YES
   Level 2: YES
   Level 3: YES
   Level 4: YES
   Level 5: YES

3. Used Catmull-Rom Splines to render the Track: YES

4. Rendered a Rail Cross Section - YES

5. Rendered the camera at a reasonable speed in a continuous path/orientation - YES

6. Run at interactive frame rate (>15fps at 1280 x 720) - YES

7. Understandably written, well commented code - YES

8. Attached an Animation folder containing not more than 1000 screenshots - YES

9. Attached this ReadMe File - YES

Extra Credit Features:

1. Render a T-shaped rail cross section - NO

2. Render a Double Rail - YES

3. Made the track circular and closed it with C1 continuity - NO

4. Added OpenGl lighting - NO

5. Any Additional Scene Elements? (list them here) - NO

6. Generate track from several sequences of splines - NO

7. Draw splines using recursive subdivision - NO

8. Modify velocity with which the camera moves - NO

9. Create tracks that mimic a real world coaster - NO

10. Render environment in a better manner - NO

Additional Features: (Please document any additional features you may have implemented other than the ones described above)

1. Attached derivation of the equation to physically accurately update u (Extra Credit)

2. Tracks plane texture mapping, texture mapping for double rails (Extra Credit)

Open-Ended Problems: (Please document approaches to any open-ended problems that you have tackled)

1. Level 1 - For the Catmull-Rom spline basis functions, s (tension parameter) was taken to be 0.5, and increments of u are in steps of 0.01 to get a decent speed of the roller coaster.

2. Level 2 - The ground is at the plane z = 20.0. This was to ensure negative Z was up (above the ground) for the roller coasters (as was told that is required by input files) and to have the entire spline above the ground.

3. Level 3 - The sky goes from z = 25.0 to z = -200.0. It starts below the ground to not see a gap at the horizon. The same texture was used for the 5 faces of the sky box, to avoid loading multiple texture images.

4. Level 4 - In order to calculate the Frenet frame, the arbitrary vector v was chosen to be $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$. This was to have unit length and to reduce the chances of T and v being parallel. The camera position is updated in the idle function, making use of the Frenet frame calculated at each point of the spline.

5. Level 5 - Calculated the 4 points for the cross-section at every point of the spline. However, ignored the above 2 points (v0, v1 and v4, v5 in Fig. 1) so as to have only the track of the roller coaster (more realistic).

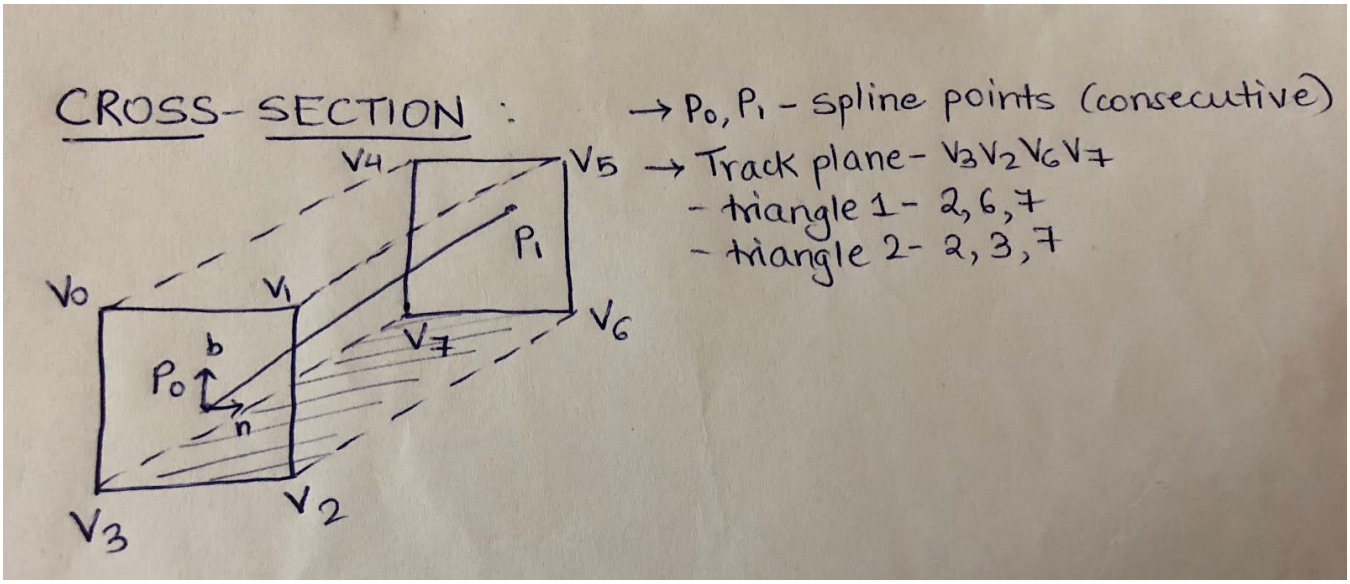

Fig. 1

6. Rendering double rail - Calculated 4 more points per the bottom 2 points of the cross section as shown in Fig. 2. This was to create a tube with 4 faces for each of the rails. Then the faces of these rails were texture mapped with a metal like texture to look like rails.

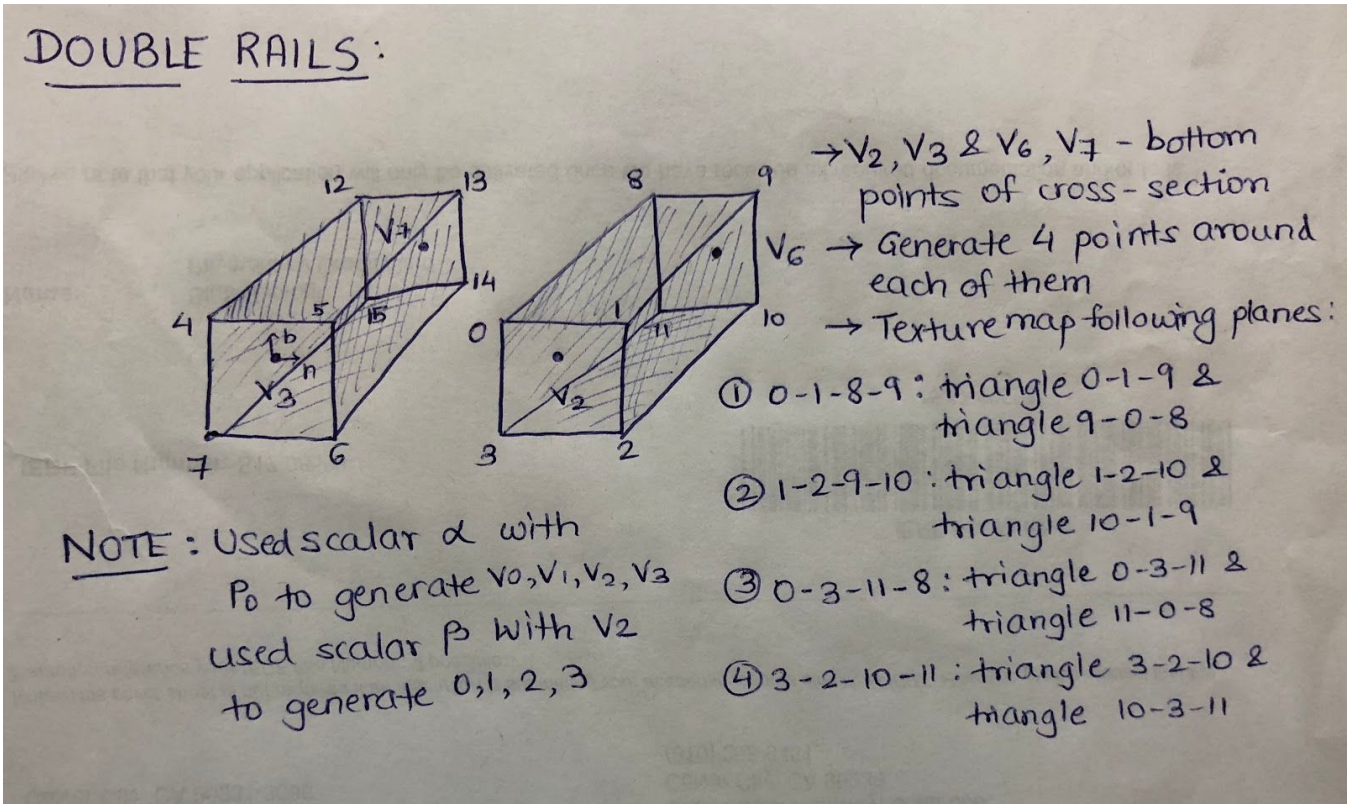

Fig. 2

7. Rendering track plane - In order to not have the track empty, a plane was rendered between the double rails and texture mapped to look like the railroads of a roller coaster. (Plane given by v2, v3, v6, v7 in Fig. 1).

8. Derivation of the physically realistic equation for u - The derivation was worked out using conservation of energy principles. The derivation is shown in Fig. 3 as well as uploaded as the file derivation.jpg in the assignment folder (hw2).

* Derivation of physically realistic equation for u:

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

→ Conservation of energy - total energy throughout the motion remains constant

∴ Kinetic energy (K.E) + Potential Energy (P.E) = constant

- At any point : K.E. $= \frac{1}{2}mv^2$ (m: mass, v: speed)
  & P.E. $= mgh$ (g: gravitational constant, h: height at current point)
- At top of the roller coaster : $h = h_{max}$ & $v = 0$
  ∴ Energy $= \frac{1}{2}mv^{2^{0}} + mgh_{max} = \underline{mgh_{max}}$
- Applying conservation of energy between any point (height h & velocity v) & top point of roller coaster, we get:

$$\frac{1}{2}mv^2 + mgh = mgh_{max} \qquad (mass \neq 0)$$

$$\therefore v^2 + 2gh = 2gh_{max}$$

$$\therefore v = \sqrt{2g(h_{max}-h)} \qquad \ldots\ldots (1)$$

→ v is speed, which is magnitude of rate of change of position (p) with time (t).

$$\therefore v = \left\|\frac{dp}{dt}\right\| = \left\|\frac{dp}{du}\right\| \cdot \frac{du}{dt} \qquad \begin{matrix}(u \text{ is scalar}) \\ (\text{chain rule})\end{matrix}$$

$$= \left\|\frac{dp}{du}\right\| \cdot \frac{\Delta u}{\Delta t} = \left\|\frac{dp}{du}\right\| \frac{(u_{new}-u_{curr})}{\Delta t} \quad \ldots (2)$$

→ Equating (1) & (2),

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

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

Fig. 3

Keyboard/Mouse controls:

1. ‘x’ - Save screenshot

2. ‘Esc’ - Exit program

NOTE: The mouse controls to scale, rotate, translate from HW1 were removed so as to keep camera motion on the roller coaster.

Names of the .cpp files you made changes to:

1. hw2.cpp
2. basicPipelineProgram.cpp - To handle multiple shaders
3. textureVS.glsl - Vertex Shader for textures
4. textureFS.glsl - Fragment Shader for textures

References:

1. http://www.hao-li.com/cs420-fs2015/exercises/Exercise02.pdf