2IV10 Assignment

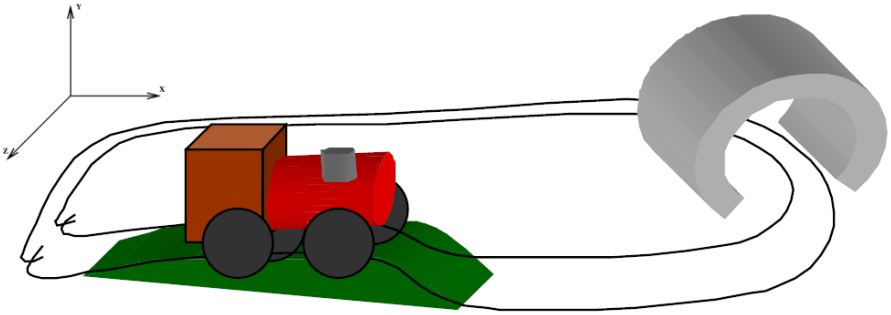
2011-2012

1 Introduction

In this assignment you will be creating a virtual complete train set, with a model train, tracks, hill and tunnel. The train will be animated such that it traverses the tracks.

2 Deliverables

The final deliverables for this assignment are a report on paper together with the source code and the executable on a CD.



3 Creating and animating the train set

The best method for building the objects is by placing the drawing code for each object in a separate function. Each object is defined using a coordinate system that makes modelling convenient. Then, when placing the objects in the world, you can translate them to the appropriate location. Start by making a menu called "Toggle Animation". If the train is not moving, and this menu is selected, the train should start going around the track. If the train is moving and this item is selected, the train should stop. Some suggestions for the animation of the train will be provided to you. The camera should be placed such that the entire scene is visible and you can see the train going around the track (except when it is in the tunnel).

3.1 Making and moving the train model

The third step is building the model of the train. The train is composed of a 3D box next to a cylinder on one side. The smoke stack is another cylinder protruding from the body cylinder. There are four circular wheels on the sides of the train. The train should be constrained to

be high enough above the tracks such that the bottom of the wheels touch the track, but not the train body. As the train moves along the track, it is translated and rotated. On level ground the train will need to be rotated around the Y-axis. To climb the hill, it will also need to be rotated around the Z or X-axis . *Hint*: Draw the box, cylinder, smoke stack and wheels with respect to one object coordinate system. When the train needs to be moved, the coordinate system is simply translated and/or rotated. Make sure to place the origin of this object coordinate system in a location that is convenient to rotate around.

3.2 Drawing the track

The track is defined by a closed Bézier curve composed of multiple joined segments (at least 6). You are free to make the track any shape you want given one constraint: at the location where you place the hill, it has to elevate off the ground plane. *Hint*: The Bézier curve defines the track shape, but the actual two tracks are offset on each side of the curve.

3.3 Building the hill and tunnel

The tunnel is constructed from simple Bézier surfaces that are placed such that the track goes under it. Make sure that the tunnel bottom is on the ground plane. The hill needs to coincide with the elevated part of the track. It should also be flat along the path of the track. Use multiple Bézier surfaces to build the hill model. You may have multiple Bézier surfaces which define each side of the hill with another surface joining them that is flat along the track.

3.4 Lights and materials

Initially, you should build the object models using simple color attributes (glColor3f). Once all above functionality is in place, introduce lights and material properties. The different surfaces in the train set (hill, tunnel, train) have different material properties. The hill is a diffuse green surface, the train should look metallic with a non-zero shininess component, and the tunnel should look like concrete or wood-made. Introduce a light that illuminates the entire scene, the coloring of which is your own choice. *Hint*: You'll have to define your material properties and lights such that they work well together. For example, if you define a surface to have a particular specular color (say blue), make sure the light illuminating the scene actually has non-zero blue component, otherwise you won't see the specular highlight.

Exercise 1 Implement the modelling and animation of the train set, as described above. Your report should include descriptions of the modeling choices which you made, along with an explanation of the animation procedure. For details on Bézier curves and surfaces, read the course book and/or slides. For this exercise you can earn 7 points.

Your animation should resemble the one provided to you at the website of the assignment.

4 Add-ons

4.1 Spotlights at night

Add a menu item called "Day/Night". When this menu is selected and the normal lights (described above) are illuminating the scene, change the light model such that the scene is

illuminated with a soft diffuse blue light. Also, during the night illumination, add a spotlight in the front of the train. The animation should not be affected by the change of lighting. This menu item should also allow you to switch back to normal lights.

4.2 Bird's eye view

Add a menu item called "Bird's eye view". This menu item should allow you to switch to a view where the camera is placed on top of the tunnel (it can be anywhere on top of the tunnel) looking down at the scene. You should still be able to animate the scene, but now the train might not be visible when it is out of the camera field of view. Also add a menu item 'Normal view' to switch back to the view described above.

4.3 Wagon

Add one or more wagons to the train and animate them while they are being pulled along the track.

4.4 Terrain

Add additional features to the terrain where the train is moving. For instance, add procedurally defined objects (trees, houses, etc), add billboarding trees, etcetera.

4.5 Animated camera

Add a menu item called 'View from train'. This menu item should allow you to switch to a view from the moving train. Make sure that you still see a part of the train. Also add a menu item 'Normal view' to switch back to the view described above.

Exercise 2 Implement some of the add-ons as described above and document them in the report. For each correctly implemented add-on you earn one point, but no more than 3 points in total will be granted.

5 Suggestions for documentation

The report should be a self-contained document that can be read without the assignment text. The report should contain enough detail to allow reproduction of your results. For this mathematical formulas with suitable explanations are the preferred method of communication. The report should not contain code snippets from your source code, but, if absolutely necessary, pseudo-code may be used. A concise user manual of your program must be given.

6 Suggestions for implementation

Start with a simple opengl program which sets up a viewing and 3D perspective transforms. Then, implement classes for each object which you should draw in this animation. Simple opengl skeleton programs are provided for C, java, and Delphi.

6.1 Modeling the tunnel

The tunnel can be constructed using four Bézier surfaces, $S1$, $S2$, $S3$ and $S4$, as in Fig. 1. Then, the tunnel can be described by three numbers, L the length of the tunnel, $r1$ the inner

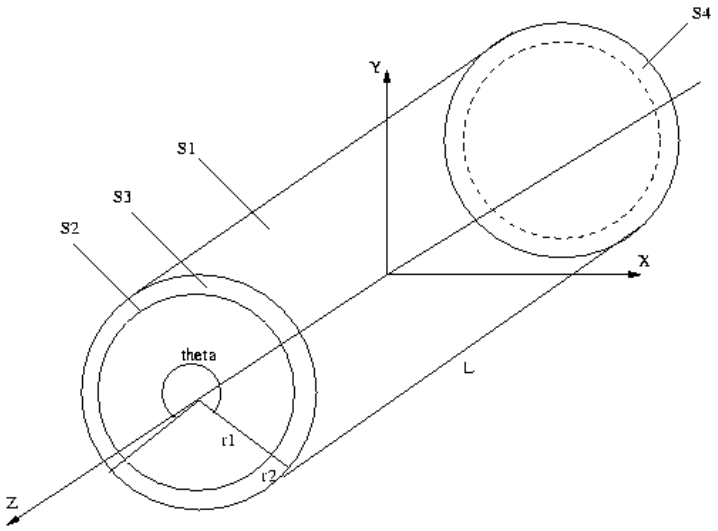


Figure 1: The tunnel.

radius and $r2$ the outer radius. Points (knots) on the surfaces $S1$ and $S2$ can be generated using the parametric equations for the circle ($x = r\cos(\theta)$, $y = r\sin(\theta)$). Once you generated the knots for all surfaces, you can use the OpenGL functions `glMap2f` to define 2D evaluators for each surface. Note that the maximum order of the Bernstein polynomials should be less than 8. Then, you should define the 2D meshes using `glMapGrid2f`. Finally, you evaluate the meshes (generate vertex positions) using `glEvalMesh2`. Since you will have to use OpenGL lighting, you also need to generate normals to the surfaces. Store the generated vertices and normals in an OpenGL display list, see Chapter 7 from the red book. *Hint*: you can use `GL_AUTO_NORMAL` to compute the normal vectors of a Bezier surface automatically.

6.2 Modeling the hill

The hill can be modelled as depicted in Fig. 2. It is described also by three parameters, W the width, L the length and H the maximum height which is also the radius of the quarter of the circle used to construct the rounded surface patch, see Fig. 2.
Similarly use an OpenGL display list to store the vertices and normals of the three surfaces composing the hill.

6.3 Modeling the train engine

The train engine can be modelled using cubes, and (capped) cylinders. Use the glut/glu functions `glutSolidCube`, and `gluCylinder` to render cubes and (capped) cylinders.

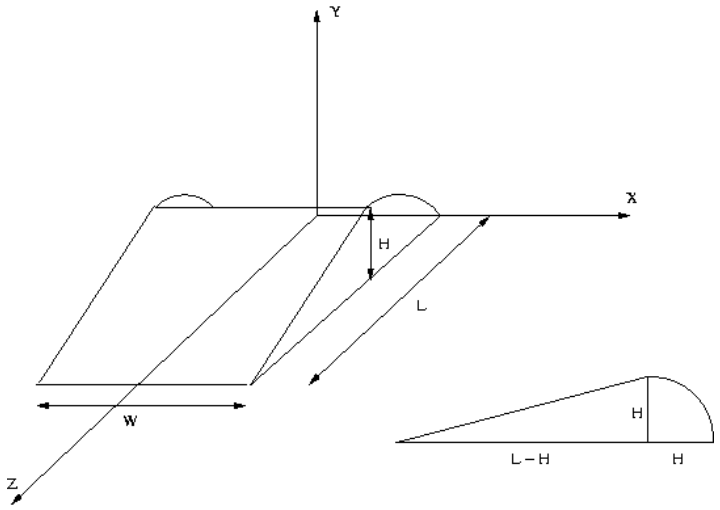


Figure 2: The hill.

6.4 Modeling the tracks. Animation

Though modeling the tracks is conceptually simpler than modeling the other objects, it poses some difficulties since the tracks should follow the hill model. Therefore, we will model this in an iterative process, involving re-evaluations and re-sampling of Bézier curves. Since the support offered by OpenGL is limited, below is a function which you can use to evaluate a given Bézier curve at an arbitrary parameter u, with $u \in [0, 1]$.

```
void eval_bezier_curve(float u, std::vector<point3d> &knots, point3d &p)
{
 int order=(int)knots.size()-1;

 if(u<=0.0f)
  p=knots[0];
 else if(u>=1.0f)
  p=knots[order];
 else {
  double b=pow(1-u, order);
  p=point3d(b*knots[0].x, b*knots[0].y, b*knots[0].z);

  for(int i=1;i<=order;i++) {
   b*=(order-i+1)*u/i/(1-u);

   p.x+=b*knots[i].x;
   p.y+=b*knots[i].y;
   p.z+=b*knots[i].z;
  }
 }
}
```

First, start with a curve enclosed in a rectangle, obtained by generating 8 knot points along the edges of the rectangle. Now, animate the train with the hill and tunnel in their places. (Of course the curve describing the tracks will be underneath the hill.) To animate the train you will need to evaluate the Bézier curve and compute the tangent and normal vectors. From these vectors you can extract two rotation angles which you should use in your animation.

Once you have the animation running, find the value of the parameter u for which the train engine reaches the hill. Also find the one immediately after the hill. Let these two values be u_1 and u_2. Now you will have to generate a new set of knot points and a new curve, by constraining the curve to go up hill between u_1 and u_2. For the values of u which are not in the interval $[u_1, u_2]$ you should resample the old curve. Generate for the new curve a number of about 200 knot points, until you are satisfied with the results. Then, you can resample once again the new curve and generate OpenGL vertices (which you should again keep in a display list) for the two tracks offset by a given parameter O along each side of the curve. Note that this time you need to store the knots since you will need to evaluate the curve for animation.