CPSC 414, Project 1: Articulated Elephant

Out: Wed 17 Sep 2003
Due: Thu 2 Oct 2003 5pm PST
Value: 10% of final grade

In this project, you will create an articulated elephant. There are three required parts (45 points), and up to 8 more extra credit points can be earned.

Modelling: [15 points total] Model an elephant out of transformed spheres and cubes.

- (2 pts) Create drawSphere() and drawCube() functions that draw a unit sphere and a unit cube, respectively. This is the only place in your program that you should call OpenGL geometry commands.
- (13 pts) Model your elephant out of transformed spheres and cubes. Remember that you can do nonuniform scaling to create ellipsoids or long skinny boxes. You should orient your elephant so that you see a side view from the default camera position. You should create your elephant using a hierarchical scene graph structure, so your placements should be relative to the coordinate system of the larger structure it's attached to rather than as an absolute transformation from the origin. For example, the head should be placed relative to the neck coordinate system. Your elephant should have at least the following parts:
  - (1 pt each:) Body, Tail, Neck, Head
  - (2 pts) Ears
  - (3 pts) 4 Legs: upper leg, lower leg, foot.
  - (4 pts) Trunk (should make out of several segments, so it can curl)
  - (extra credit: up to 3 pts) Add more detail, for instance tusks, a tail tuft, eyes, etc.

  You should build your elephant in a 'rest' pose where all the joint angles are set to 0, because you will be moving the joints as below.

Animation: [25 points total] Animate the joints of your elephant. Specifically

- (2 pts) Simple tail wag: Rotate tail up wrt body.
- (2 pts) Head/neck nod: Rotate neck/head piece down wrt body.
- (4 pts) Leg raise: This is a two-stage process. Rotate at hip between body and hip, then rotate at knee between upper leg and lower leg to bend lower leg under until it's closer to parallel with the ground.
- (7 pts) Trunk roll: Make trunk curl under. This is a multi-stage process, where each segment needs be moved.
- (extra credit: up to 4 pts) Add more motions. For instance, an ear wiggle where the ears do some interesting and vaguely physically plausible rotation(s), or a complex tail wag where the tail rotates around in a circular arc, or standing upright on the hind legs, or whatever strikes your fancy.
- (10 pts) Transition: Show the motion as smoothly animated transition instead of a jumpcut. You should linearly interpolate between the old parameter and the new one in a loop, and explicitly redisplay the image between each invocation.

Interaction: [5 points total] Interactively control your elephant.

- (5 pts) Keys: Add the following GLUT key bindings for the animation: 't' for Tail, 'h' for Head, 'l' for Leg (and 'm', 'n', 'o' for the other three legs), 'k' for trunK roll. These keys should act as toggles: on a click, move from rest position to new position, or vice versa. Add the binding of the 'q' key for a clean exit of the program. Have the spacebar toggle between "jump" mode and "transition" mode. Pick unused letters to trigger your extra credit motions, document them in your writeup.

Suggested Strategy

Clearly, there are dependencies here: if you don't model a tail, you can't get credit for any of the tail animation parts. I recommend that you interleave the modelling, animation, and interaction. For instance, start with a body and neck. Place the neck with respect to the body. This is a good time to do the camera movement interaction, so that you can check whether things are placed correctly. Sometimes a view from one side can look right, but you can see from the top or front that it's in the wrong spot along one of the other axes. You can get far with 3 camera placements: default side, top, and front. Now go onto the nod keyboard interaction, then implement the nodding animation. Once that all seems to work, go on to another body part. Do interpolation after you've done all the body parts. Finish doing the entire required functionality before starting on any extra credit.

You might find it easier to debug your code if you use a separate transformation for the joint animation then the one you use for the modelling.

Template

The template code is available from http://www.ugrad.cs.ubc.ca/ cs414/Downloads/proj1.tar, which contains the two files `p1.cpp` and `Makefile`.
The template code allows you to change the viewpoint to look at the elephant from any of six directions. Consider the elephant to be at the center of a cube. In the default you're looking at it from one face of the cube, for a side view. You can move the camera so that you can see the elephant from any of the other 5 faces of the cube: other side, front, back, above, under. Trigger this action with the 's', 'f', 'b', 'a', 'u', respectively. The 'r' key resets to the original view. You should construct your elephant so that the default view is indeed the side view.

Handin

- Create a root directory for our course in your account, called cs414. Later all the assingment handint files should be put in this directory.
- For project 1, create a folder called proj1 under cs414 and copy to there all the files you want to handin, including your source, makefile, README. Do not use subdirectories, these will be deleted. We only accept README, makefile, and files ending in cpp and txt.
- In your README please describe what functionality you have implemented, as well as any information you would like to give us for getting credit for partial implementation. If you don't complete all the requirements, please state clearly what you have successfully implemented, what problems you are having, and what you currently think might be a promising solution. If you did extra credit work, say what you did and how many extra credit points you think the work is worth.

  Please be clear and concise. The README should only be a few pages at most.
- The assignment should be handed in with the exact command:

  ```
  handin cs414 proj1
  ```

  This will handin your entire proj1 directory tree by making a copy of your proj1 directory. Note that any subdirectories in that directory will be deleted! For more information about the `handin` command, see `man handin`.

Grading

- This project will be graded with face-to-face demos: you will demo your program for the TA. If need be, you can concisely summarize the arguments in your README about incomplete work. You can also explain any extra credit features you implemented.
- We will circulate a signup sheet for demo slots in class. Each slot will be 10 minutes. The demo sessions will be Friday Oct 4 (except during the scheduled lab from 10-11), Monday Oct 6, and Tue Oct 7.
- You *must* ensure that your program compiles and runs on the lab machines. If you worked on this assignment elsewhere, it is your responsibility to test it in the lab. The face to face grading time slots are short, you will not have time to do any 'quick fixes'! If your code as handed in does not run during the grading session, you will fail the assignment.
- Bring a printed copy of your code and README file to the face-to-face grading session to give to the TA. This printout must match exactly what you submitted electronically. The code that you demo must match exactly what you submitted electronically: you will show the TA a long listing of the files that you're using, so that he can quickly verify that the file timestamps are before the submission deadline. Arrive at CICSR 011 at least 10 minutes before your scheduled session. Log into a machine, and double-check that your code compiles and runs properly. Then delete the executable.

  When the TA comes to your computer. you will type the following:

  ```
  % ls -l
  % make
  % p1
  ```

Hall of Fame

The best work will be posted on the course web site in the Hall of Fame.