CPSC 314, Project 1: Articulated Kangaroo

Out: Wed 19 Jan 2005
Due: Thu 27 Jan 2005 6pm
Value: 10% of final grade

In this project, you will create an articulated kangaroo. There are three required parts (100 points), and up to 10 extra credit points can be earned. The best work will be posted on the course web site in the Hall of Fame.

Modelling: [35 points total] Model a kangaroo out of transformed cubes, using only $4x4$ matrices to position and deform them.

- (1 pt) Create a drawCube() function that draws a unit cube. This is the only place in your program that you should call OpenGL geometry commands.
- (34 pts) Model your kangaroo out of transformed cubes. Remember that you can do nonuniform scaling to create long skinny boxes. You should orient your kangaroo so that you see a side view from the default camera position. You should create your kangaroo using a hierarchical scene graph structure. That is, instead of just using an absolute tranformation from the world origin for each individual part, you should use a relative transformation between an object and its parent in the hierarchy. For example, the head should be placed relative to the neck coordinate system. This hierarchical structure will make both modelling and animation much easier in the long run, even if it might seem like extra work at first! Your kangaroo should have at least the following parts:
  - (2 pts each:) Body, Neck, Head, Ears.
  - (4 pts each:) 2 Forearms: arm, paw.
  - (5 pts each:) 2 Rear Legs: upper leg, lower leg, paw.
  - (8 pts) Tail: should make out of several segments, so it can curl.
  - (extra credit: up to 4 pts) Add color. And/or add more detail to your kangaroo, for instance eyes, claws, etc. And/or make a pouch with a baby kangaroo in it. And/or make an interesting environment in which your kangaroo can hop around - again using only cubes and $4x4$ matrices.

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

Animation: [55 points total] Animate the joints of your kangaroo. Specifically

- (5 pts) Head/neck turn: Turn neck/head section wrt body so kangaroo looks sideways.
- (5 pts) Forearm lower: Rotate the whole arm down wrt body.
- (8 pts) Single leg raise: Rotate the leg assembly up wrt body, rotate the foot up wrt lower leg.
- (12 pts) Tail roll: Make tail curl up (or down, if you'd rather). This is a multi-stage process, where each segment needs be moved incrementally with respect to the previous one.
- (15 pts) Transitions: 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 redisplay the image between each step.
- (10 pts) Hop: Have both legs raise as above, then lower, and have the entire kangaroo jump into the air (translate up wrt ground), then come back down again. This feature is only interesting in transition mode.
- (extra credit: up to 6 pts) Add more motions. For instance, an ear wiggle where the ears do some interesting and vaguely physically plausible rotation(s), or a more complex tail wag, or having your kangaroo drop down to be on all fours. Or new camera positions or trajectories. Or whatever strikes your fancy.

Interaction: [10 points total] Interactively control your kangaroo.

- (10 pts) Keys: Add the following GLUT key bindings for the animation: 't' for head turn, 'a' for arm lowering, 'l' for leg raise (and 'm' for the other leg), 'c' for tail curl, and 'h' for hop. 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 "jumpcut" 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. If you don't interleave the modelling and animation, you might spend a lot of time creating an animal with the wrong kind of hierarchical structure to move correctly. 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 Download from `http://www.ugrad.cs.ubc.ca/˜cs314/Vjan2005/proj1.tar`, which contains the two files `p1.cpp` and `Makefile`.
The template code allows you to change the viewpoint to look at the central object from any of six directions. Consider the kangaroo 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 kangaroo from any of the other 5 faces of the cube: other side, front, back, over, under. Trigger this action with the 's', 'f', 'b', 'o', 'u', respectively. The 'r' key resets to the original view. You should construct your kangaroo so that the default view is indeed the side view.

Documentation 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.
You must include at least two images of your kangaroo. We'll post the best of these kangaroo images in the Hall of Fame. One good choice would be front and side views. On the Linux boxes in the lab, the easiest way to take snapshots is with the ImageMagick `import` utility. Just type `import myfile.png` and click on your program's window with the resulting X-shaped cursor. You can look at your images with the `display command`. If you want to show off your kangaroo in action through a loop of several images, you can make animated GIFs easily using the `convert` command, as described at `http://www.tjhsst.edu/ dhyatt/supercomp/n401a.html`.

Handin Create a root directory for our course in your account, called cs314. All the assignment handin subdirectories should be put in this directory.
For project 1, create a subdirectory of cs314 called proj1 and copy to there all the files you want to hand in, including your source, makefile, README. Do not use subdirectories, these will be deleted. We only accept README, makefile, source files ending in .cpp and .h, and image files ending in .png or .jpg or .gif.
The assignment should be handed in with the exact command: `handin cs314 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 Your grade will be partially based on correct performance and partially on clean coding practices (probably a 70%/30% split).
This project will be graded with face-to-face demos: you will demo your program for the grader. We will circulate a signup sheet for demo slots in class. Each slot will be 10 minutes. For the first part of your slot, you will demo your program to the grader. Then, the grader will briefly discuss the code with you. Finally, the grader will spend some time alone, looking at the code further and writing up notes.
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 grader. This printout, and the code that you demo, must match exactly what you submitted electronically: you will show the grader a long listing of the files that you're using, to 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 grader comes to your computer, you will type the following:

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