CPSC 314: Programming Assignment 2

Due 4pm, Friday, October 11, 2013

The purpose of this assignment is to practice geometric transformations and transformation hierarchies, and gain experience in modeling and animating articulated characters. Specifically, you will be modeling and animating a kitten (Figures 1 and 2) which is modeled from ellipsoidal links. You should use hierarchical transformations as described in class when drawing and animating it. The following is a suggested ordering of steps:

(a) (0 points) Download the template code from

`http://www.ugrad.cs.ubc.ca/~cs314/Vsep2013/a2/a2.tar.gz`

and compile it

```
mkdir assn2
cd assn2
tar xvzf a2.tar.gz
make
```

The provided solution draws and animates a kitten. The animation sequences are specified using keyframe files passed as command line arguments to the program. The program has a simple keyboard user interface, which lets you run the animation, traverse it pose by pose, and perform other basic operations. It also has mouse controls which allow you to move the camera. Run the solution both with and without command line arguments and with different keyboard inputs to better understand the expected behavior. The README file describes the set of keyboard commands and what each of them does. It also describes the structure of the keyframe files.

The template provides the basic OpenGL environment, keyframe file reading and parsing, and mouse/keyboard interface. Read the template files and understand what each function does.

(b) (5 points) Fill the missing code in the drawEllipse() function (utils.cpp) that draws an ellipse centered at the origin with three radii given by (x, y, z). You can call the glutSolidSphere function in the process. Remember that you can do nonuniform scaling to create ellipsoids from spheres.

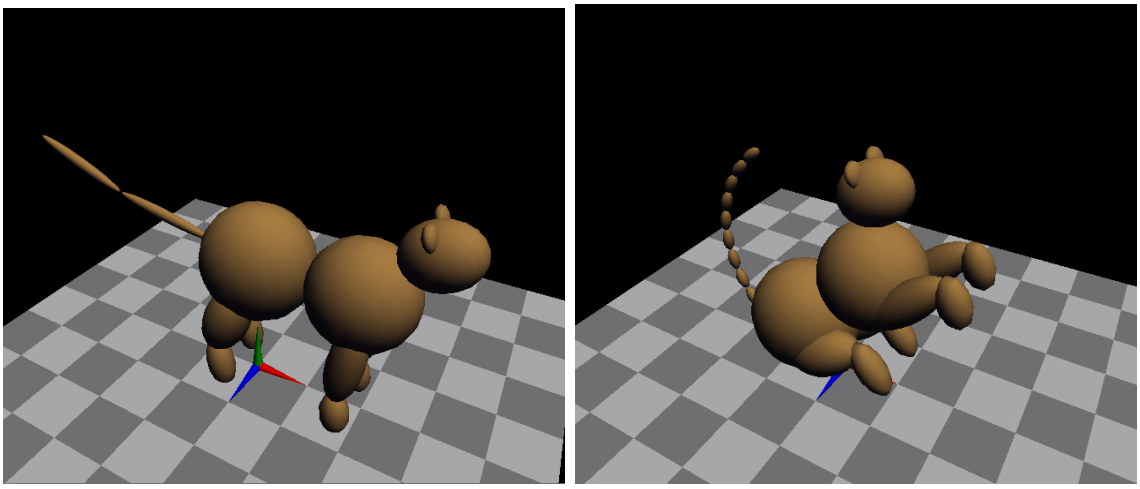
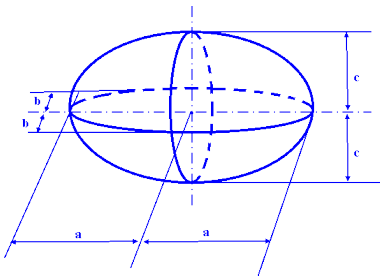


Figure 1: Kitten in two poses



(c) (50 points) Draw your articulated figure - fill the missing code in the Kitten::draw() function. You should only use the drawEllipse() function for the actual drawing. Do not use any glut or GL geometric primitives. Model the kitten using the set of links shown in Figure 2. Each link should have one degree of freedom (DOF), rotation around the Z-axis (The one exception is the torso, which should also have three additional degrees of freedom to specify its center position in 3D). Use the DOF values (positions and angles, examples shown in Figure 2), stored in a Pose structure (retrieved with Kitten::getPose()) to specify the connections between the links. Set the dimensions of the links and the joint positions yourself (I would recommend using roughly similar proportions to the ones in the example solution). Use an appropriate hierarchy of transformations. **Hint: do not write all the code at once, add one link at a time and test the code on the provided keyframe files. Your code should behave similarly to the solution when reading an individual keyframe.**

(d) (10 points) Add code to the Kitten::getPose() function to generate smooth transition (interpolate) between consecutive keyframes. Think how to achieve this given the DOFs (positions and angles) for the two frames. A basic linear interpolation (think line segment formula as shown in class) is sufficient for this part of the assignment.

(e) (15 points) Create a keyframe file for a non-trivial animation containing four frames (first frame can always be the rest pose). The animation should involve significant

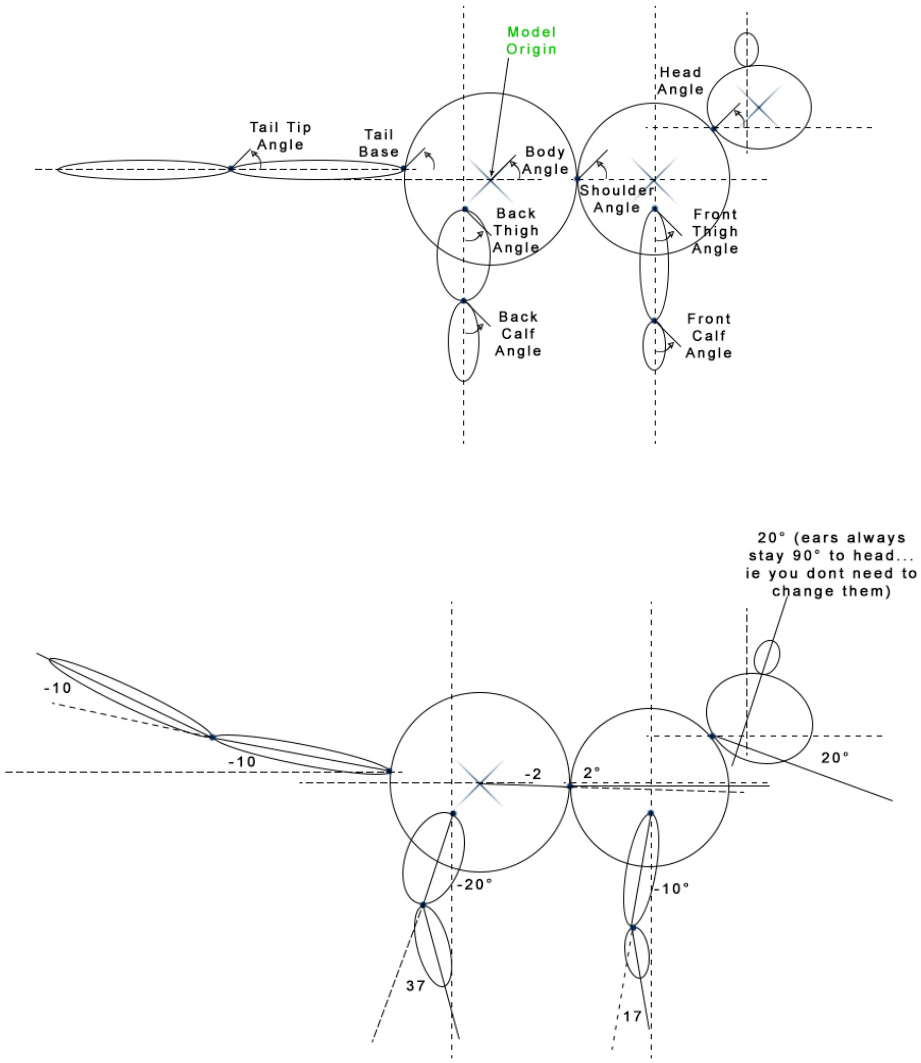


Figure 2: Degrees of freedom (position + angles)

changes to all the DOFs. Example animations could be: get the kitten to sit; have the kitten pounce; have the roll over.

(f) (20 points) **Do not start on this part until you completed all the tasks above.** For the (semi) free-form part of the assignment you should do one of the following extensions.

- Make a tail with ten or more links that bends smoothly. These links should be generated efficiently (i.e. not individually). You should implement an algorithm to distribute the "tip" angle (Figure 2) between all of the links. No extra elements should be added to the keyframe files for this part.(see Figure 1: Top-Right Pose)
- Add extra degrees of freedom to the kitten: activate out of plane (z axis) rotation along the major four cat spine joints or on the leg joints in order to achieve one of the following positions:
  - chase tail
  - sit without legs intersecting (Figure 2: right)
  - curl up into a sleeping ball (Figure 2: left)

  For this extensions you will need to modify the keyframe reading mechanism to support extra degrees of freedom. You will need to provide a keyframe file which showcases the changes you have made (the easiest way to do that is to edit the provided examples via some small external script). Your code must **still run** on all the keyframe files we provide as well as the animation you create in (e) above.

Implementing both extensions will, at the discretion of the grader, earn you extra bonus marks.

Bonus: To improve your animation you can:

(a) add more links, shapes, or DOFs to make your kitten more realistic

(b) add functionality to the advanced tail to allow for question mark and s-shaped tails

(c) add smooth interpolation of keyframes (e.g. use splines)

(d) etc...

For any new geometric features, keyframe files must be included to show them off.

The marking here will be necessarily subjective. In addition, the best animations will be entered into the 314 Hall of Fame.

Hand-in Instructions

- Hand in all the source files for the assignment, a README, and all the keyframe files you generated. Document in the README file what each keyframe file does. Note that all the keyframe files MUST have a .txt ending.

Figure 3: Left: Kitten curled up, Right: Kitten sitting with legs splayed
images from www.lovemeow.com, www.warrenphotographic.co.uk

- In your README file, please describe what functionalities you have implemented, as well as any kind of 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 tried, what problems you are having and what you think might be promising solutions. If you are using external sources (i.e. to populate the world with more objects), provide clear attribution.

- The assignment should be handed in with the exact command:

  ```
  handin cs314 assn2
  ```

  This will handin your entire assn2 directory tree by making a copy of your assn2 directory, and deleting all subdirectories! (If you want to know more about this handin command, use: man handin)

Assignment Grading

The assignment will be graded with face-to-face demos: you will demo your program for the TA. If your assignment is incomplete, you can concisely summarize your explanations and what you were trying to do in your README. You can also explain any extra credit features you implemented.

- We will circulate a sign-up sheet for demo slots in class. Each slot will be 7 minutes. The demo sessions times will be determined closer to submission date.
- You must ensure that your program compiles and runs on whatever medium you intend to demo on (laptop, lab computer, etc.). 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.
- 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 ICICS/CS 005 at least 10 minutes before your scheduled session. Double-check that your code compiles and runs properly.
- When the TA comes to your computer you will run your assignment with the keyframe files we provide and with your own keyframe animation file. If you are shooting for the bonus points, show the grader the extra features and/or scenarios you created.