CPSC 314: Assignment 2

Due 4pm, Friday, October 12 2012

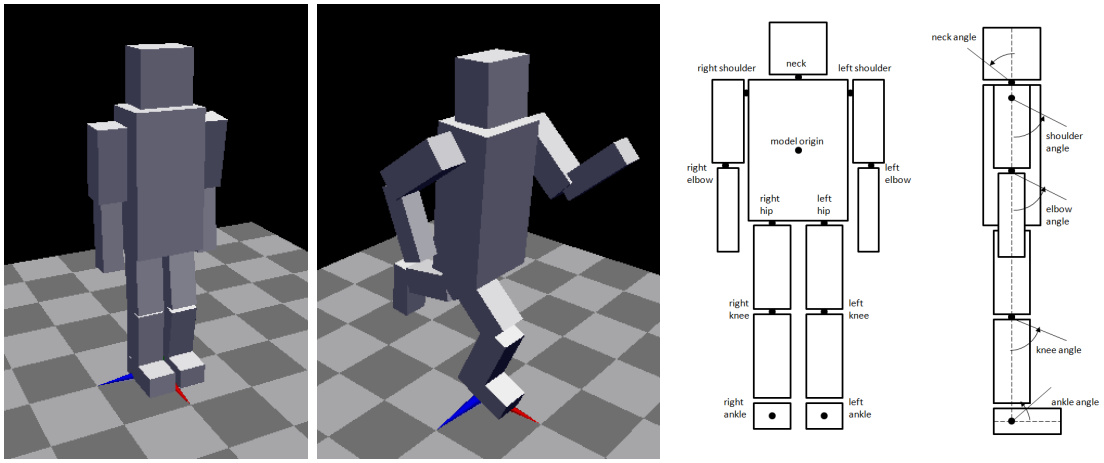

Figure 1: Left: Cuboid robot in two poses, Right: degrees of freedom (position + angles)

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 robot (Figure 1) which is modeled from cuboid 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/Vsep2012/a2/a2.tar.gz`

and compile it

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

The provided solution draws and animates a robot. 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 all 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 drawCuboid() function (utils.cpp) that draws a cuboid centered at the origin with sides of the given lengths (x, y, z). You can call the glutSolidCube function in the process.

(c) (50 points) Draw your articulated figure - fill the missing code in the Robot::draw() function. You should only use the drawCuboid() function for the actual drawing. Do not use any glut or GL geometric primitives. Model the robot using the set of links shown in Figure 1. 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 1), stored in a Pose structure (retrieved with Robot::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 Robot::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 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 changes to all the DOFs. Example animations could be: get the robot to jump in the air; have the robot sit or lie on the ground; have the robot dance.

(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.

- Add at least two extra links to your models (e.g. hands, antennae).
- Add extra degrees of freedom to the robot: activate the 'z' coordinate for the center of mass and add out of plane rotation for a couple of joints.

For each of those extensions you would need to modify the keyframe reading mechanism to support extra degrees of freedom. You would need to provide a keyframe file which

showcases the changes you made (the easiest way to do that is to edit the provided examples via some small external script). Your code should **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 add more links or DOFs to your character, use additional shapes when drawing the robot, populate the environment with other interesting objects, have multiple robots roam the world, etc... 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.
- Use the root directory cs314 that you created for assignment 1. For assignment 2, create a folder called assn2 under cs314 and put all the source files that you want to handin in it, including your "Makefile" and your README file. Don't use subdirectories – these will be deleted. NOTE: we only accept README, Makefile and files ending in cpp, hpp, c, h, and txt.
- 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 (e.g. 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 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 7 minutes. The demo sessions times will be determined closer to submission date.

- 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.

- 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 005 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
```

  You will then 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.