CPSC 414
Assignment 1

Due February 7, 2003

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: 

Student Number: 

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>/ 3</td>
</tr>
<tr>
<td>Question 2</td>
<td>/ 3</td>
</tr>
<tr>
<td>Question 3</td>
<td>/ 9</td>
</tr>
<tr>
<td>Question 4</td>
<td>/ 2</td>
</tr>
<tr>
<td>Question 5</td>
<td>/ 2</td>
</tr>
<tr>
<td>Question 6</td>
<td>/ 19</td>
</tr>
<tr>
<td>TOTAL</td>
<td>/ 38</td>
</tr>
</tbody>
</table>
1. Composition of Transformations
   
   (a) (1 point) Show that the transformations $\text{Trans}(3, 2, 0) \text{Rot}(z, 90^\circ)$ and $\text{Rot}(z, 90^\circ) \text{Trans}(2, -3, 0)$ are equivalent by comparing the resulting transformation matrices.

   (b) (2 points) Show that they are equivalent by sketching two figures that illustrate the interpretation of the above transformations in a left-to-right order, i.e., in local coordinates, and which illustrate the intermediate and final coordinate frames for each of the two compound transformations.

2. (3 points) A rigid body transformation is one which consists of only rotations and translations, and so it does not change the shape or scale of the object. Which (if any) of the following correspond to rigid body transformations? Why or why not?

\[
\begin{bmatrix}
1 & 0 & 0 & 10 \\
0 & -1 & 0 & 2 \\
0 & 0 & 1 & -6 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0.5 & 0.5 & -0.5 & -1 \\
-0.5 & 0.5 & 0 & 3 \\
0 & 0 & 1 & 2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 0.5 & -0.866 & 1 \\
0 & 0.866 & 0.5 & -1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
3. Transformation as a Change of Coordinate Frame

(a) (4 points) Specify the coordinates of point \( P \) with respect to coordinate frames \( A, B, C, \) and \( D \).

(b) (2 points) Derive a transformation that takes a point expressed with respect to frame \( D \) and re-expresses it in terms of frame \( B \), i.e., determine \( M_{D\rightarrow B} \), where \( P_B = M_{D\rightarrow B} P_D \). Verify your solution using your answer to part (a).

(c) (3 points) Derive a transformation that takes a point from frame \( C \) to frame \( B \), i.e., determine \( M_{C\rightarrow B} \), where \( P_B = M_{C\rightarrow B} P_C \). This is easiest if you pass through an intermediate coordinate system, such as frame \( A \). Verify your solution using your
answer to part (a).

4. (2 points) Determine graphically what the following transformation does. Sketch an object that has the transformation applied to it. Describe in words what the transformation involves.

\[
\begin{bmatrix}
0 & -1 & 0 & -1 \\
-1 & 0.5 & 0 & 2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

5. (2 points) For the following simple lens, use graphical techniques to determine where the image plane should be placed such that the image of the arrow will be in focus.

![Lens Diagram](image)

6. Coding

In this question, you will be creating an animated 3D human figure that will walk across the screen. The following is a suggested ordering of steps.
(a) (3 points) Create a simple model of the torso as a rectangular block of appropriate dimensions. Use GL_POLYGON or GL_QUADS to draw each face, and assign each face a slightly different colour. Setup a suitable perspective or orthographic projection, and render the torso. You will need to allocate a Z-buffer and enable the depth test in order to ensure proper results in terms of visibility, i.e., all faces behind the immediately visible faces are hidden. This is done using: `glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH)` and `glEnable(GL_DEPTH_TEST)`, respectively.

(b) (1 point) Add a glut key binding such that ESC exits your program.

(c) (3 points) Build the rest of your articulated figure. Use an appropriate hierarchy of transformations. Implement it using appropriately structured code. You may want to define separate geometry for each type of link, or you may want to use scaled versions of something like a cube. Draw your human figure in an appropriate rest pose — typically this is a simple standing pose.

(d) (2 points) Add the ability of your character to assume different poses by adding joints to all the links. To keep things simple to start with, assume that each joint only has one degree of freedom and rotates about the same axis as the knee joint, as shown below.

(c) (3 points) Add the capability to have your character be displayed in different poses, each pose being defined in a file, `keyframe.txt`. This file will consist of a specification of all of the degrees of freedom of the character. Each line in this file should have a format something like the following:

```
keyframe 6.34 2.0 1.0 10 20 10 10 -20 10 10 40 50 -40 50 10 20
```

where:

- parameter 1: time stamp (ignore this for now)
- parameter 2: torso location, x
- parameter 3: torso location, y
- parameter 4: torso lean
- parameter 5: right hip angle
- parameter 6: right knee angle
- parameter 7: right ankle angle
- parameter 8: left hip angle
- parameter 9: left knee angle
parameter 10: left ankle angle
parameter 11: right shoulder angle
parameter 12: right elbow angle
parameter 13: left shoulder angle
parameter 14: left elbow angle
parameter 15: neck forward tilt
parameter 16: head forward tilt

Add a keybinding such that hitting the 'k' key reads in the next line of the keyframe file and displays the next pose. Once all poses are displayed, cycle back to the first pose again.

(f) (3 points) Now you will be animating the figure by using linear interpolation between the keyframes in order to produce 'in-between' poses of your character. Choose a fixed time step and add an animation loop. At first, debug this by simply continuing to display the current pose until it is time to display the next pose. Next, implement the linear interpolation in order to compute an updated pose for each time step. Create a keybinding that lets the spacebar start and stop the animation. When you get to the end of the animation, loop back to the beginning. You will need to experiment in order to set reasonable keyframes (animation isn’t easy!).

(g) (4 points) To improve your animation you can have your character perform a variety of motions, populate the environment with interesting objects, generate an improved model for the body, etc. The marking here will be necessarily subjective. The best 3 animations will be entered into the 414 hall of fame.

Hand-in Instructions

- Hand in a printed copy of your code and README file.
- Create a root directory for our course in your account, called cs414. Later all the assignment handin files should be put in this directory.
- For assignment 1, create a folder called assn1 under cs414 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, c, and txt.
- In your README file, please describe how to run your program and 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 to what extent you have gone to, what problems you are having and what might be the promising solutions as you think.
- The assignment should be handed in with the exact command:

  handin cs414 assn1

This will handin your entire assn1 directory tree by making a copy of your assn1 directory, and deleting all subdirectories! (If you want to know more about this handin command, use: man handin)