

CS 175: Assignment 4

Robots and Part Picking

Due:

Monday, October 19th, 11:59pm

Assignment Objectives

This assignment will build on the previous assignment and extend it by implementing a system for drawing articulated bodies, as well as the ability to select objects on the screen (picking.)

Your program will draw two robots, instead of two cubes as done in the previous assignments, and allow the manipulation of robot parts in a way that preserves the hierarchical structure of the robot. You will still allow the user to rotate and translate the robots as well, and the sky-camera, using the same arcball interface as in the previous assignment.

Step 1: Drawing the Robots

We will want to support hierarchical models. You will need to create a model for your robot, which can be as simple as you like as long as it contains the following: a main body, a head, a neck, upper arms, lower arms, upper legs, and lower legs.

These robots (or whatever your hierarchical model looks like) should be drawn hierarchically. This means the lower left arm will not keep track of its overall position and orientation, but rather its position and orientation with respect to the upper left arm. It is only when computing the matrices for each hierarchical part for rendering that you will be multiplying out those long strings of matrices as done in class.

NOTE: It is recommended that you store rotations for the model's joints using Euler angles (r_x, r_y, r_z) —corresponding to rotations about the x, y, and z axes respectively—as opposed to using a rotation matrix. This will make it easier to do animation in a later assignment.

Computing the matrices: In your OpenGL application, you will be writing code to compute the matrices for each part and sending that data as uniform variables to the shader.

Step 2: Part Picking

We now turn to allowing the user to determine the pose of each robot. The user should be allowed to press “p” and then use the mouse to left click on a desired robot part. The determination of which part the user clicked on is called “picking.” After picking a part, the user should then be allowed to use the mouse to rotate the part about its joint (connecting it to parent).

Picking will be performed as follows: When “p” is pressed, the next left mouse click will trigger the following: The current scene is rendered, but this time instead of dumping the usual colors into the color (back) buffer, you will assign each part to be drawn a unique ID (color) that is drawn into the buffer instead. Drawing this ID tagged scene will perhaps involve writing/using different vertex and fragment shaders that produce a solid color. After rendering the scene (you may have to call `glFlush()` after issuing all your draw calls), read back the (back) buffer (or at least the region around the mouse click) and thereby determine which part was selected by the mouse.

- If the mouse click was not over any robot, then the sky-camera is selected (so it can be manipulated as is done in assignment 3.)

- If the mouse click was over a robot's main body, the entire robot is selected.
- If the mouse click was over a robot's part, then the chosen robot part is selected.

The robot clicked on should become the currently active robot.

Assuming a valid part is selected by the left mouse click, dragging the left mouse button in x and y directions should correspond to rotations about two of the part's axes (the choice of which axes is yours as long as they are distinct).

Also note that the object ID is an `int`, while the color used to draw an object is an RGB color with each component specified as a `float` in the range $[0, 1]$. For simplicity, we can simply use the first component (Red) of the color to represent object IDs. Assume an ID can take value $0, \dots, N-1$, to convert from ID to a float in $[0,1]$, one can use:

```
float f=(float)id/(float)(N-1);
```

To convert back from a float to an integer ID one can use:

```
int id=(int)(roundf(f * (N-1)));
```

We use `roundf` instead of simple cast-to-int to avoid possible errors introduced by floating point inaccuracy.