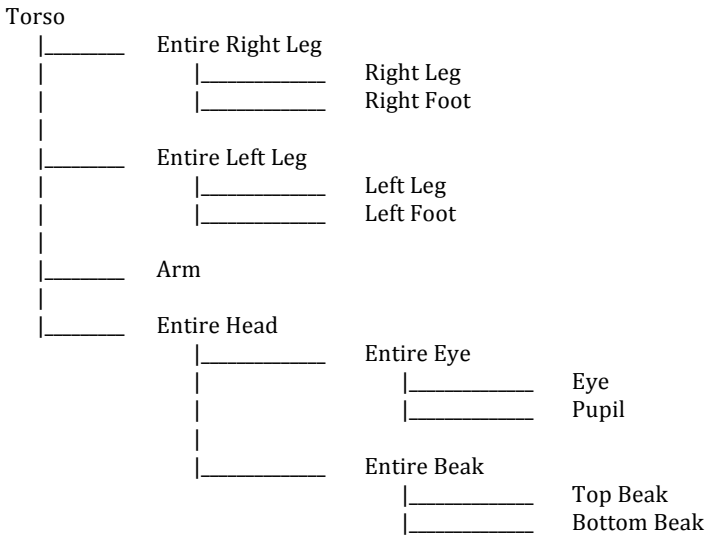
CSC418 – Assignment 1 – Part B – Report

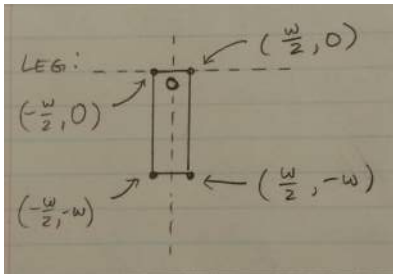
Tanzeem Chowdhury
997574726
Oct. 10, 2014

The first thing to complete for part B of the project was to completely understand and process how each of the penguin components were going to interact and transform in relation to one another. As the handout suggested, the objects are hierarchal, and we needed to organize the transformations in a kinematic tree. The tree would be comprised of a root, which is the body/torso, and the remaining components of the penguin would either be leaves of the body, or further leaves of descendants of the body. For my display function, the kinematic tree was structured in the following way:



Each component lies within glPushMatrix() and glPopMatrix() calls, which also contain transformations that are only meant for that specific component, such as scaling, which is pretty much different for each component. In addition, each component's translation and sometimes rotation transformations, relies and builds upon the translations already achieved via parental nodes earlier on the tree.

Rather than hard-code the vertices, each component's vertices were determined by its relational distances based on its height and width, and scaled later by height and width constants. The x-axis and y-axis were mapped based on where the circular joint is located. For example, for the leg, attached to the torso, the vertices were determined like so:



The advantage of this is that if we want to scale a component, instead having to edit every single vertex, we can just modify the length and width variables that are used to scale the component. [Attached at the end of this report are the remaining diagrams I drew.]

For the structural design of the project, every component was created using the following three created functions:

- drawPolygon(…)
  - used to create the outline of any sized polygon by sending in arrays of x and y vertices, as well as the number of vertices for parameters
  - arrays of x and y vertices are defined as constants in the display function before building the kinematic tree

- drawCircle(…)
  - used to create the outline of any sized circle by sending in radius of circle as a parameter

- drawFilledCircle(…)
  - used to create any sized filled circle by sending in radius of circle as a parameter

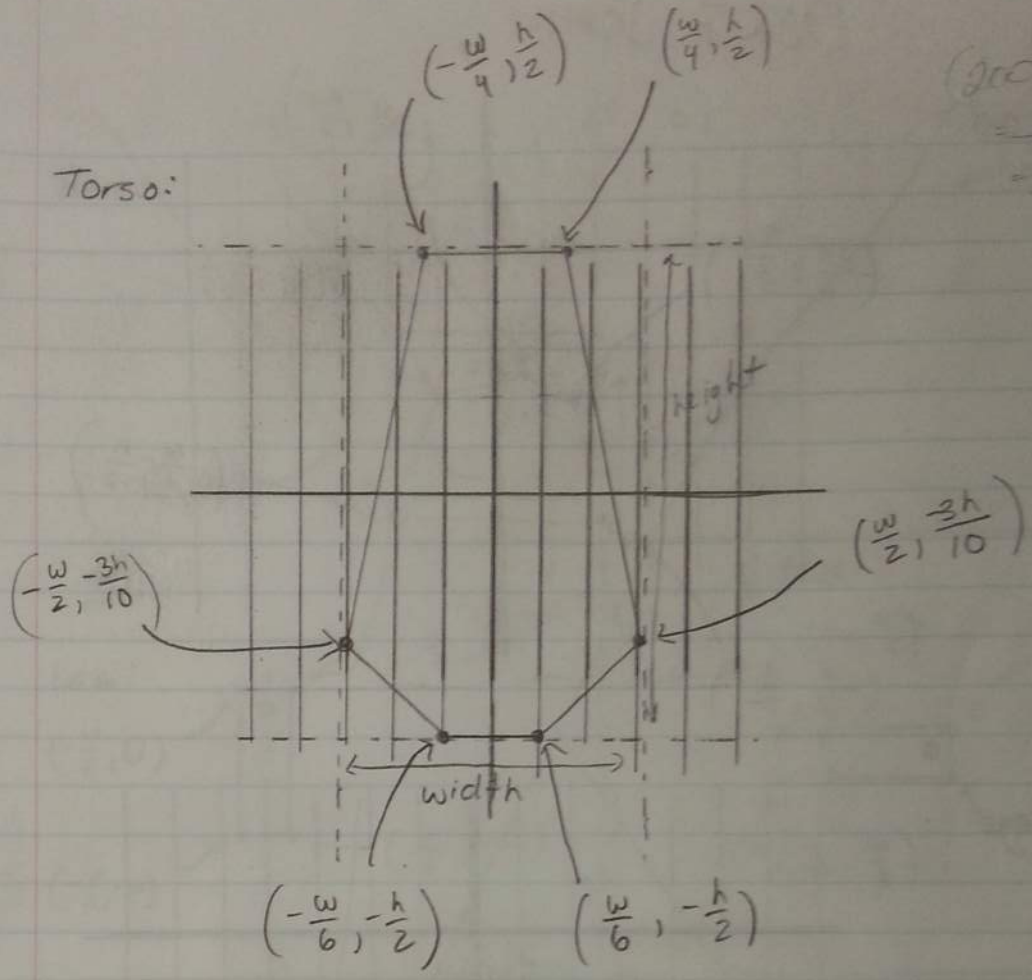
Once translations and scaling were added to the kinematic tree, rotations were added in next. Each rotation transformation added for a joint applied to the component connected to the joint, and any child components. For each component that rotated, additional min, max and joint parameters were defined. These parameters were used under the animate() function to update the component geometry, based on the speed, limits and rotation factors of the component. The angle used in glRotatef() call for one leg was the negative angle of the other leg, so the legs can swing in the opposite directions. In addition to the leg, the feet, arm, and head rotations are set.

For the beak, only the bottom component has movement. The geometry was defined similarly to the other components under the animate() function using the same sin function, however, instead of a glRotatef() call, a translation call is used for the animation, allowing the beak to be animated vertically up and down continuously. In addition to the individual components, the entire body itself is also animated. Animation of the body in the y direction is accomplished similarly to other components using the sin function. The entire body however is translated in the x direction at the same time, starting from the right side of the screen towards the left. Once the body reaches a certain point on the left side, the body is reset to the right. The following pseudo code sets up the structure of this functionality inside the animate() function:

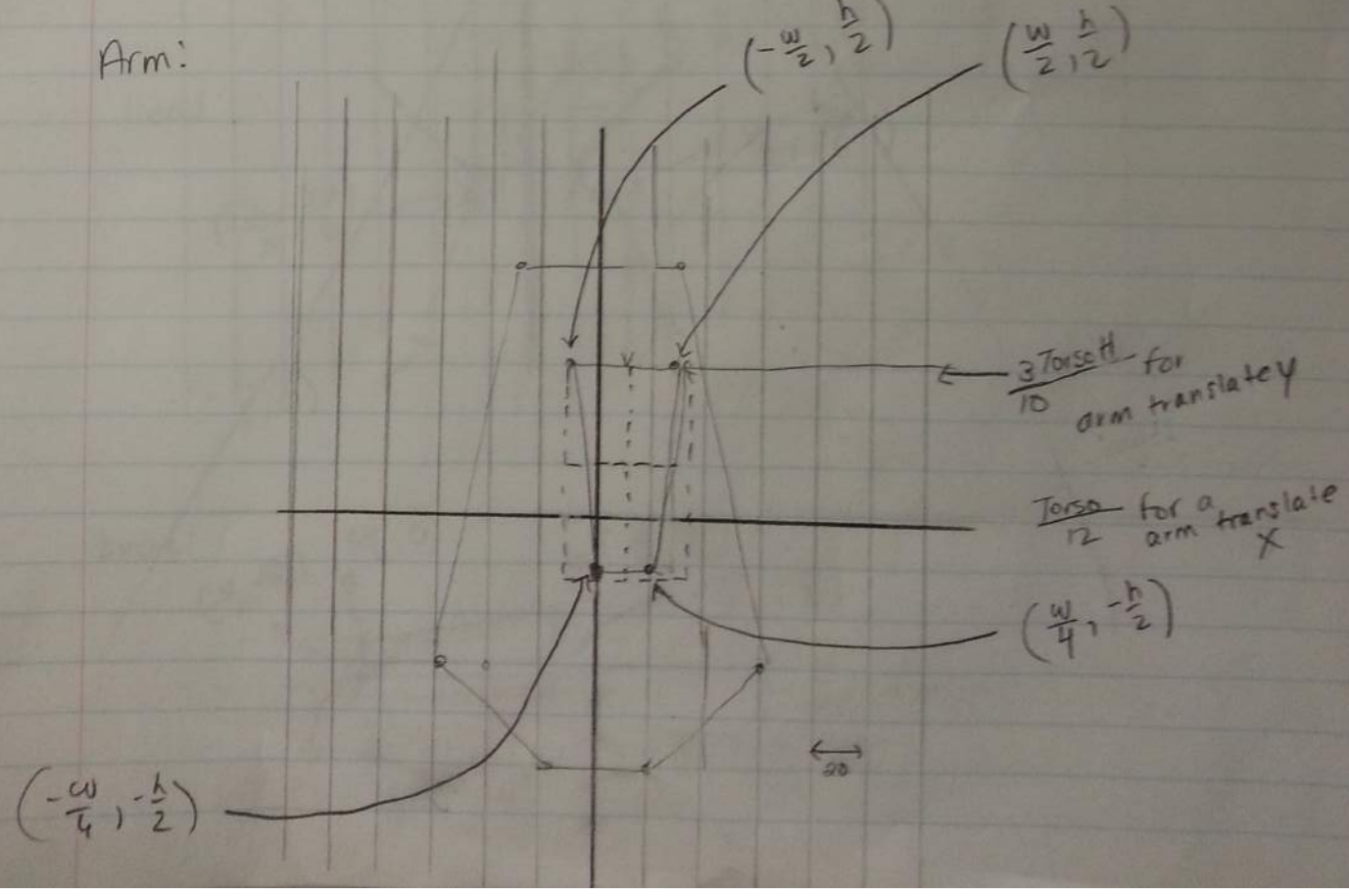
```
if (the body reaches -350 i.e. left side of screen) {
    reset position of the body to the right side
        (x_body_animate = 350.0)
    also need to reset the animation frame to zero
} else {
    animate the body in the x direction
}
```

The final step was to add a control for each additional joint in the initGlui() function, enabling individual controls in the graphical user interface.

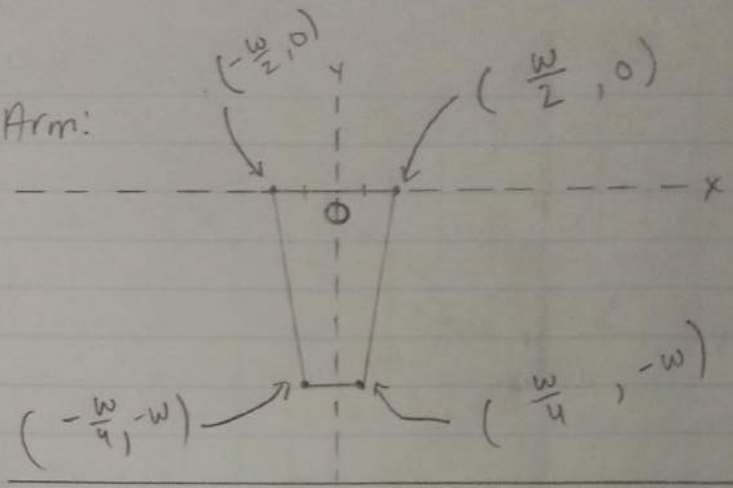
Torso:


$(-\frac{w}{4}, \frac{h}{2})$
$(\frac{w}{4}, \frac{h}{2})$
height
$(\frac{w}{2}, \frac{3h}{10})$
$(-\frac{w}{2}, -\frac{3h}{10})$
width
$(-\frac{w}{6}, -\frac{h}{2})$
$(\frac{w}{6}, -\frac{h}{2})$

Arm:

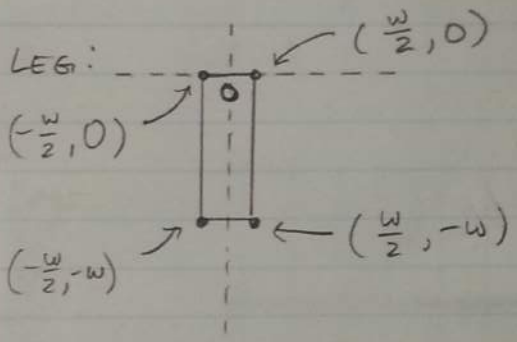

$(-\frac{w}{2}, \frac{h}{2})$
$(\frac{w}{2}, \frac{h}{2})$
$\frac{3\,Torso\,H}{10}$ for arm translate Y
$\frac{Torso}{12}$ for a arm translate X
$(\frac{w}{4}, -\frac{h}{2})$
$(-\frac{w}{4}, -\frac{h}{2})$

Arm:

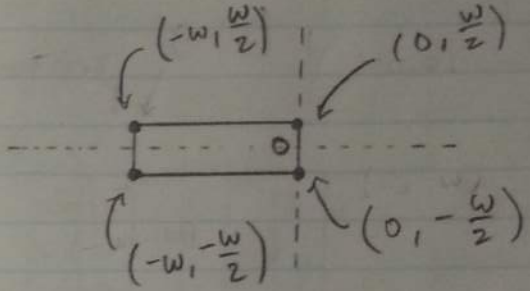


w = width or length

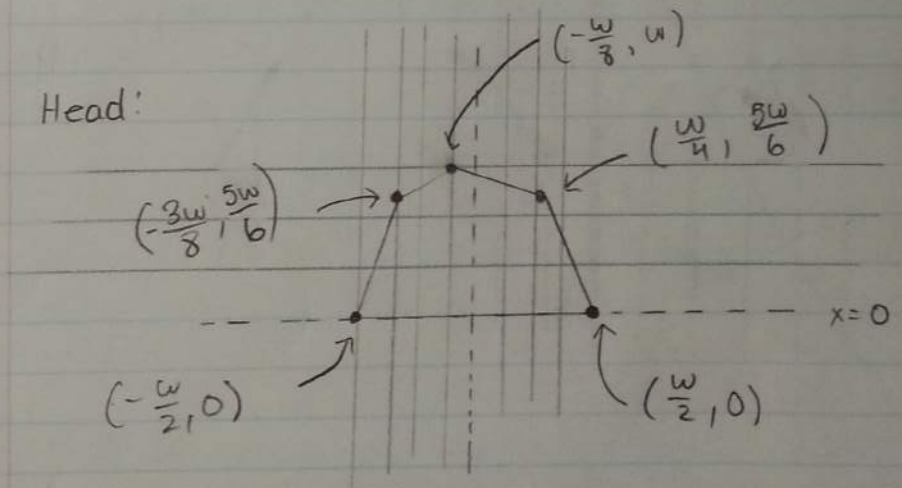
LEG:



Foot:



Head:



Beak: