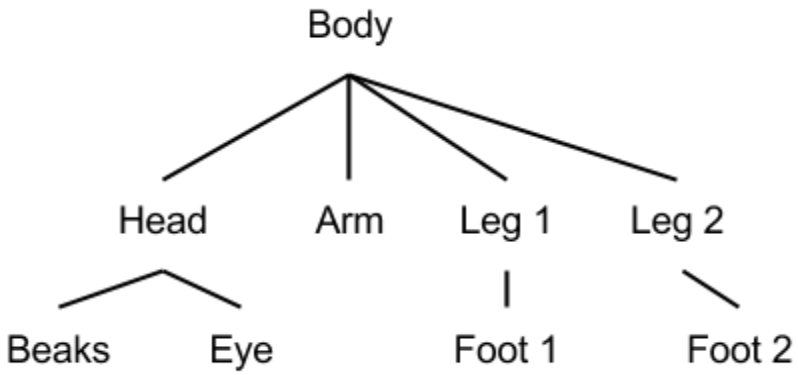
Assignment 1 Part B

Report

Tanzim Mokammel
9966155208

I started off by exploring the helper code. To grasp an understanding of the flow of the code, I modified the helper code to add in another box, and observe how the three boxes behaved when drawing, transforming and animating with respect to each other, and the world coordinates. Once I achieved transforming and animating the three boxes in the desired order, I started to expand the helper code to complete the project in the following steps: design the kinematic tree, draw, and transform the individual shapes, and finally define the functions their for animations.

The first step was to design the kinematic tree, so that I may draw some shapes in relation to each other, and ensure parts transformed together as needed. My designed kinematic tree is as follows:



I first drew the head to become more familiar with the hierarchy system. Then I drew the body, attaching the head, and then connecting the arm and the legs.

In designing the shapes, I attempted to mimic the provided drawing. Therefore, I used the UnitSquare class in GLWidget.h to make classes for a symmetrical trapezoid, an asymmetrical trapezoid (beak), and a triangle. using these new classes, I drew, scaled and translated shapes in order to construct the penguin in the order of the hierarchy. The order of action in all of the shapes is as follows:

- Draw the shape
- Translate the rotation joint to the desired spot (if needed)
    - Draw the joint circle at this spot

- Scale the shape
- Rotate/translate about the joint

For any shapes that require animation for itself, or animation for any objects under its hierarchy, its scaling has to be separated through a push and pop, as rotation after scaling causes shearing. To account for this, some shapes in the hierarchy were drawn to its world coordinates as opposed to what was drawn before it. Examples of this can be seen in the beak, legs and the arm.

It is extremely important to note that hierarchical objects must be drawn in a specific order, and surrounded by accurate push/pop clauses such that transformations are applied accurately to the kinematic tree. For example, the rotation on the neck must also apply to the beak. This is why designing the kinematic tree is an important first step.

During the phase of the initial drawing, I used the given sine function to animate the shapes around their joints in sync (using the joint_rot_t variable). This step helped me better understand that the scale matrices need to isolated to prevent shearing of anything else down the kinematic tree. Once I had all of the joints animating in sync, I constructed the sliders in MainWindow.h. In doing so, I added the required functions and variables in GLwidget.h (public slots). Following that, I modified GlWidget::timerEvent to define the functions for the animation. For the animation, I used variations of the provided sine function, modifying the min and max values in GLWidget.h to provide realistic rotations and translations. The arm, legs, and the head use rotation functions, while the beak, and the body use translation instead.

Once the above parts were tested to be working, I put a few final touches such as scaling the body parts, and changing colors to achieve a more aesthetically pleasing penguin.