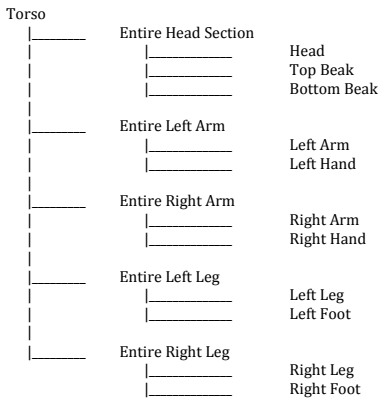
CSC418 – Assignment 2 – Part B – Report

Tanzeem Chowdhury
997574726
Nov. 09, 2014

Like A1, the first thing to complete for part B of the project was to determine how each of the penguin components was going to interact and transform in relation to one another. The objects are hierarchal, and we need 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 nested glPushMatrix() and glPopMatrix() calls, which contain transformations that are only meant for that specific component, such as scaling, different for each type of component. In addition, each component's translation and rotation transformations, relies and builds upon the translations already achieved via parental nodes earlier on the tree.

Once the basic kinematic skeleton was built, each component was coloured differently according to type of body component. The colouring was important to display the rendering styles later on. In addition to designing the kinematic tree for nested body part leafs, the joint locations were added. Animated rotations and translations were applied to all joints in the kinematic tree where required. The joints rotations were added by using the angle predefined in the starter code:

joint_ui_data->getDOF(Keyframe::VAR_BODY_PART)

Once the joints were complete, the predefined angle maximum and minimum constants were edited for appropriate rotations. Once the component rotations were completed, the root translates and root rotates were setup.

Rather than hard-code the vertices, each component's vertices were determined by its relational distances based on its length, width and depth, and scaled later by scaling constants. Depth, length and width constants were defined for the Torso, Head, Arm, Hand, Leg and Foot at the top of the display function.
The components are scaled after the shapes were drawn first in unit sized version by the following functions:

- drawCube(…) – used for rectangular prism components (i.e. leg, hand)
- drawTrapPrism(…) – used for trapezoid pyramid/prism components (i.e., head, torso, beak, arm)
- drawTriPrism(…) - used to for triangular prism components, (i.e. feet)

Once the main penguin is drawn, the render styling panel was worked on. At the top of the display function, a switch statement was placed do determine whether the penguin should be displayed in wireframe, solid, solid with outline, metallic or matte. For wireframe, glPolygonMode was set to (GL_FRONT_AND_BACK, GL_LINE). For solid, glPolygonMode was set to GL_FRONT_AND_BACK, GL_FILL. For solid with outline, the draw functions were altered, so that they call helper functions that do the drawing. In the original draw functions, if outline was determined to be the render style, the faces are first drawn as they would be with solid, calling the helper function. Then, glPolygonMode is set to GL_LINE, POLYGON_OFFSET_FILL is enabled, the face is redrawn again for the outline, and the polygon mode and offset are reset to solid and disabled respectively.

To achieve Metallic and Matte rendering, the phong model was used, where we needed to apply diffusion, ambience, and specular. In the switch statement that handled render style at the top of the display function, glMaterialfv was used to apply these properties for both metallic and matte renders. Prior to the switch statement, a function is called to initialize the light. This function (initializLight()) enables the gl lighting, colour material and light, and configures the light position, if metallic and matte are detected as the render style chosen. If it is one of the other three render styles, the lighting is disabled. The specific values for configuring the ambience, diffusion and specular were declared near the angles (metD, metA, metS, matD, matA, matS), and these values were tweaked until a satisfactory matte and metallic render was achieved.

For metallic and matte to render properly, the draw functions that create the unit shapes need to be edited, so that for each face that was created, a normal needed to be created aswell. The normal are required by openGL to calculate the lighting equations.

Finally, a control was added for adjusting the lighting in the render control box, under the panel named lighting.

For key frames, the function updateKeyframeButton was edited so that maxValidKeyframe is updated and keyframe entries are appropriately updated. Once that was complete, the key frame animation was added using the GUI controls.