Name: Cecilia Aponte		Matricola: 1822225
Interactive Graphics		Homework 2 – Report

This second homework was given to create a simplified dog using a hierarchical model, adding a procedural texture of a checkboard with a decrease in intensity from the front to back, and finally adding a button for the dog to walk along the x-axis. This report shows the results of my implementation to solve the questions requested.

The first task was to create a hierarchical model of a simplified dog. I started by creating variables for the height and width of each body part: body, tail, head, 4 upper legs, and 4 lower legs. They were each set as rectangle proportional to what the actual body part would be like in a real dog. For example, the tail is a thinner rectangle than the head is as seen in *Figure 1*. To create the hierarchical model, a function is used to create a node in the model, each node signifying a body part and its link as the root, sibling or child as seen in *Figure 2*. This will determine how a body part will

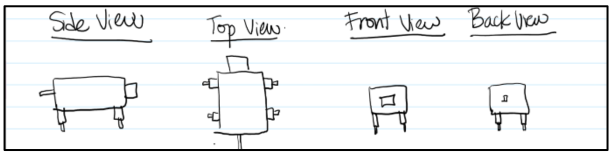

Figure 3: Various views of the simplified to draw.

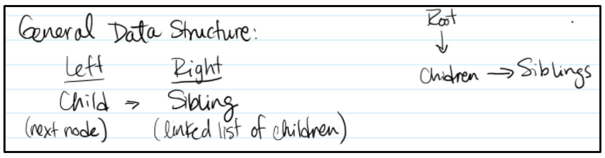

Figure 4: General data structure for hierarchical model.

move relative to the rest. As seen in *Figure 3*, the dog's body movement is the root and its movement will affect every other part in the body (they will move with it); the head, tail and 4 upper legs will be moving with the body but can rotate independently since these are all children of the root. The 4 lower legs on the other hand will be move with either respective upper leg, since these are their siblings. Another function is later created to initiate each node that will translate or rotate the body part and will specify its matrix to be transformed, the function that will be rendered, its sibling and child. The features for each body part are as follows: the torso will be placed on the x-axis to the far left of the window, so that it starts walking from the beginning, and can rotate it necessary in the y-axis (like if the dog would be rolling); the head as seen in *Figure 4* was translated in the x and y axis to be on the top, right center of the torso and can rotate in the y and z axis like looking to the sides and up; all the

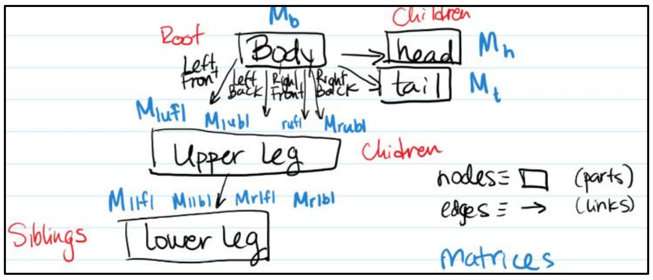

Figure 1: Hierarchical (DAG) model for a simplified dog.

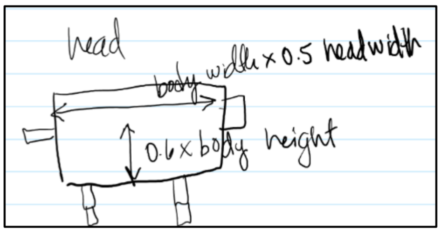

Figure 2: Translation of the head to be placed in the correct location relative to the dog's torso.

upper legs are shifted down to the bottom of the torso, front/back legs are located on opposite corners in the x-axis of the torso, and left/right legs are located on opposite corners in the z-axis. Each leg can rotate about the z-axis to simulate its movement along the x-axis; the lower legs are just translated to the bottom of each upper leg's height and can rotate also along the z-axis; lastly, the tail was translated to the left, top center of the dog and is rotated to 40 degrees along the x-axis to simulate a normal dog tail. For each node, the body parts were translated into the torso, to overlap and decrease any gap that might be visible due to rotation later. Each node has its own function where the instance matrix is multiplied by the Modelview matrix with a translating to half of the body part's height and scaling based on the size of the part. This is later flattened, and the triangles are drawn. Every node is later traversed meaning the Modelview matrix is pushed and pop in order to save the information of the sibling and child in order to transfer the same movements to these. The disadvantage of using this method is that it is not using the DAG model where instead of doing every leg separate, you would use just one for the upper leg and another for the lower leg as shown on *Figure 3*. This way you would be taking advantage of the fact that all the upper and lower wheels are identical, therefore decreasing not only the code but also the computing needed to produce the animation. However, an advantage is that by keeping things separate, you can play with any differences that are present, like the fact that the front and rear legs face the opposite angle and also in case you might want to sketch an animation with one leg longer or shorter to see the result of the walk in comparison.

I added an extra feature to my implementation and this is to add a 3D view of my dog. Initially I did this because I wanted to check and make sure that all the locations of the body parts were placed in the correct location since in 2D it is hard to tell. Therefore, once I created my 3D view I noticed that the legs were located too near to the center of the dog and naturally the dog has its legs in the outside of its body. As I continued to construct my animation, this 3D view was incredibly useful to check the animation of the walk, and the turning of the head, and any other feature I was working on. Thus I consider this feature both important for me to implement the tasks and useful for the viewer to view the dog in a more realistic way and judge its movements. My implementation for this view was that I moved my projection and modelview matrix to the render function so that it would be possible to change it with the button I added. I used the same projection matrix for both 2D and 3D (although I did try to use separate ones to improve the 2D view and the issues I was having as described in my last paragraph in this report. I implemented a button for the change therefore in the modelview matrix where using the 2D view I used the regular mat4 matrix with no special parameters (although I also tested this to see if it would improve the 2D view) and for the 3D view I changed the lookAt parameters of eye, at, and up so that the final view is as portrayed. This implementation has been very gratifying to build and use as it shows a more realistic dog and makes me prouder of the creation!

The second task was to add a procedural texture to add a checkerboard pattern with a linear decrease of intensity from the front to the back of the body. I started this by creating the two images, where one will have the checkboard pattern in both colors and the other image has a decrease in intensity of the checkboard image. At the front of the dog the checkboard image is clear and bright, but as it moves to the end of the dog it starts getting darker until the checkboard pattern is almost completely black. For the second image I implemented a linear decrease by using the length of the body part in the horizontal (“j” in the code) to apply the decrease. For it to be linear, it must be in the form of y = m*x + b. Therefore, I created the simplest method and tested many combinations to see the optimal method which is 1+j. If the intensity wants to be to the opposite side of the body it would be in the form of 1-j; if more than one decrease is needed, then j would be multiplied by a quantity; and lastly if the intensity would commence or end in a different location than at the beginning, then you would add a number higher than 1 (ex. 100 + j would give you a patter that starts with intensity decrease at almost the middle of the dog and moves along the back of the dog and then the intensity continues at the front of the dog to conclude at the same point it started. This disadvantage of using this linear method is that you cannot control where the intensity starts to decrease (maybe you want the decrease to happen quicker or slower). After this is done, the texture is configured. The first texture is given a nearest mipmap linear so that the it’s sharp, while the second texture is given a linear mipmap linear so to make it blur a little and blend in better and produces the smoothest result; both of which linearly interpolate the texel values from the two nearest best choices of mipmaps. The advantage is that this produces the optimal resulting texture, although a disadvantage is that using linear mipmap linear is smoother and therefore takes longer to compute. Another disadvantage is that when looking at the dog in 3D, the checkboard boxes look distorted, but if not used the dog would look too blurry. Also, for the first texture I used a magnification filter that was nearest, instead of linear because it produced the sharpest image while for the second texture I used linear to use the opposite effect. Later both textures are activated and blended separately to be later sent to the fragment shader and combined with also the color to become one single 2D texture. The texture coordinates are sent also to both the vertex and fragment shaders.

The third task was to start an animation of the dog so that is walking along the x-axis. My implementation was to add a button that would activate my dog walking from the left of the screen to the right and would repeat itself by reappearing at the left side again. I did this by keeping a variable move_count that would check the position of the dog as it adds 1 every time it renders the animation. This way if the dog's torso position is at the end (greater than position 55), it would restart itself (location -55). My first object implementation was to make the head shift during walking progressively by 2 degrees until it arrives at 90 degrees to watch the viewer. Then I moved the tail by delaying the rendering by 5 counts, so only every 5 render the tail angle would change directions from 40 to -40 degrees. For the legs, I took care of observing and studying the actual movement of dogs walking to imitate as close as possible their walk to my implementation as seen in *Figure 5*. Therefore, the walk I

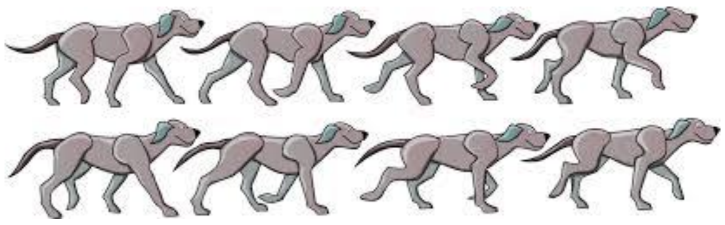

Figure 5: The sequence of dog walking.

implemented was to start with the left from leg moving, then after 20 rendering/movement counts the dog torso would move and every leg except the right leg which would wait 10 more counts before moving and the left back leg which would wait 20 more seconds. This is because any movement initially starts with just a single leg movement, followed by the rest of the body then the opposite leg of the same side which starts to move and finally the back leg on the same side would move. This is done to make sure that at least 2 legs are on the floor at the same time and sometimes 3 to keep the dog in balance, and for my implementation for my dog to look like it has a normal walk. This walk can be better appreciated when in the 3D view. For all the upper legs, the maximum angle on flexion is 30 and -30. When it arrives at either of these angles, it reverses its direction of motion and starts to move to the opposite direction incrementally by 1. The front and back legs also move in motions opposite to each other as they try to keep balance, so the front upper legs move forward by adding one degree at a time until it reaches its maximum and the back upper legs move initially backwards by subtracting instead in the same manner. The lower legs, on the other hand, move only to a maximum of 20 or -20 degrees. The front lower legs decrease by half a degree where the front lower legs increase or decrease in the same direction as their respective upper leg while the back lower legs move in the direction opposite of their corresponding upper legs. The result of this is a syncretized motion very similar to an actual dog's walk, which can be better appreciated in the 3D view. I also added information where when the 3D view button is pressed again to stop walking, the dog's head moves back to its original position looking ahead and its tail to its initial position. Then all the nodes are initiated so that the dog moves, and the torso is traversed and the entire image animated. The advantages of this method are that the dog has a normal walk similar to an actual dog, also that you can control each leg separately to fine-tune the walking method. The disadvantage of it is that due to the projection matrix near and far locations, I encountered some complications since my near was located at 0 and my far at 50 the tail and two of my dog's legs disappeared due to the location I had located it. I tried many different methods to solve this and I did many times, however when doing so it caused other issues. My current implementation now has near as -50 and far as 50 and the entire dog is being seen however it has caused the head to turn in the incorrect direction from the back to the side instead of the front to the side. However, the implementation is correct because if the 3D view is seen, the head is turning in the correct direction! Also, I know that is the correct projection of the dog because my dog is showing that the left leg starts to move first in the animation. Therefore, this might just be a minor issue since the 3D shows a more complete and accurate interpretation of the code and implementation I created.