Homework2 IG

Flavia Ferranti 1707897

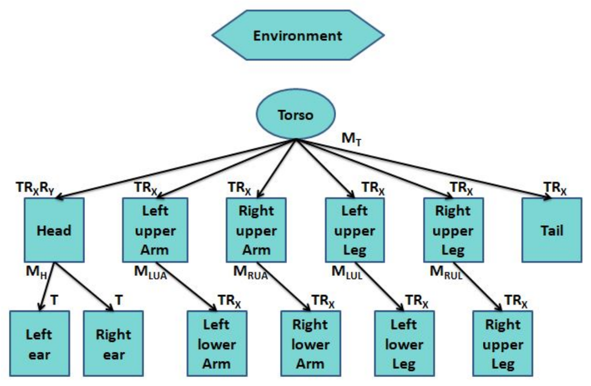
June 2020

1 Introduction

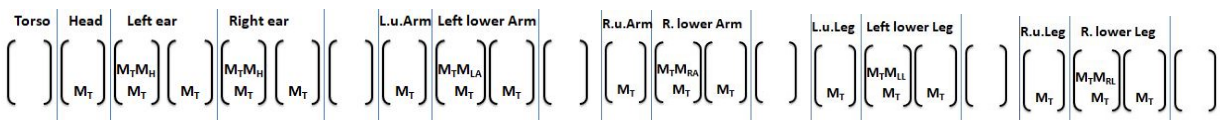
According to the request of this assignment, I had to realize a scene of a Grizzly bear that walks toward a tree and starts scratching its back.
The realization of this project was done using WebGl. In order to compute the animation I had to construct models that enables movement in a specified moment. Let's see these models in details.

2 Hierarchical model

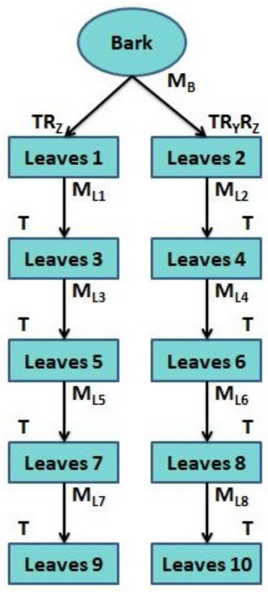
The hierarchical model is used to build a complex figure in such a way that it's made of several smaller elements related together in order to enable a "chain" reaction, extremely useful when the movement come to action. The relationship among these parts are represented through graphs, more precisely trough a directed graph so that each element is related to others trough *"siblings"* and *"children"*-like relation. More precisely the graph can be seen as an acyclic tree composed by root and internal or leaves nodes.



For what concern the Bear structure, the root is represented by the torso; its children, that are all siblings, are the head, tail and upper part of the paws. The head has two additive children: the ears, and the paws have as children their correspondent lower part. With this structure the incremental approach is extremely clear: the torso can rotate about the y-axis and the x-axis (in order to be put in a walk or stand position) according to a certain angle; the leg are translated with respect to the torso position, in order to be placed in the right spot, and can rotate about the x-axis (allowing the walking mode), the head is translated at the top of the torso and can rotate about both x and y-axis and the same holds for the tail, positioned at the bottom of the torso.
Now let's assume that the torso rotate according to the y-axis, in the same way the children as to rotate as well. So the tree structure allows for each child to inherit the model-view matrix of the parent plus its own matrix, used to orient it and locate it in the space. This means that between each part of the model there is a joint angle. This angle specifies the orientation of a component with respect to the one to which it is attached to (in the case of the root the orientation is respect to the surrounding environment). In my case all these joint angles are inside the *theta* array.
So after the creation of the 13 empty nodes that belongs to the bear and after the specifying of the various rotation's angles and translation's positions, I used the function *InitNode* to actually compute the model-view matrix of each component and save it as information inside the node. Furthermore the function establish each relation, through pointers, and it also point to the function that actually draw the image on the screen. The shape of each part are specified independently by this later function in an instance matrix that scale the original cube, used for every component of the bear, to the desired size. Later on the product between the component's model-view and the instance is send to the vertex shader.
The final result of all these operations can be seen on screen thanks to the use of the *traversal* function. The *traversal* visit every node composing the bear starting from the left and going as deep as possible before restarting from the first node on the right branch. This means that we will start by torso going to head, left ear and right ear and then continuing to the paws till the tail. The basic idea on which the function operate in order to apply the model-view matrix only to the children is using a stack. Here is the evolution of the bear stack at each call starting from the torso:



Once the *traverse* function is called in the render we can visualize our Bear.
In order to apply a certain animation to the scene, I used the same approach of the hierarchical model to construct the tree.



In this case the root is constituted by the bark of the tree and its children are the the first layer of leaves on the left and the other on the right. These leaves has other children that inherit their same orientation and so the new layers just need to be translated on top of the previous one. This procedure is repeated until arriving to a total of 5 layers. So even for this new model it was necessary to specify the *InitNode* function and the traverse one.

3 Animation

Once the main model of the scene was completed it was possible to start the animation. The main idea on which I based my animation was to define a set of keyframes and make the computer define the in-between frames. So the important thing was to identify the main positions and split the movement in a sequence of simpler movements. In order to generate the rest I used a linear interpolation with constant velocity that can easily be increased though a button. The linear interpolation is made of the following function:

$$x = x_0 + \frac{t - t_0}{t_1 - t_0}(x_1 - x_0)$$

where x_0 and x_1 represent respectively the starting and ending position and t_0 and t_1 the starting and ending time of the action.
In order to make the bear arrive near the tree and scratch upon it, I divided the movement in 5 sub-movements. Here we analyze each of them and each function that generates them.

3.1 Walk till reaching tree - Move forward function

This function was used to make the bear walk linearly from it's initial position to a final one that is near the tree and in front of the viewer.
The translation of the walk in the x and z axes was calculated with the interpolation function in a period of time going from $t_0 = 0$ to $t_1 = 10$. The result of each call of the function updated the precedence value of the bear position. After arriving to the desired new position, the next moving function was called.

3.2 Positioning with back to the tree - Rotate to tree function

Once the bear did all the walking. it had to adjust its position near to the tree in order to allow the standing up and scratching. For this reason, in a time going from 0 to 3, the bear improve its position by rotating of a certain angle *theta* equal to 100 degrees. In the same time it walks backward in order to attach the posterior paws to the tree. Also this function has been realized thanks to the interpolation, that allow us to change the joint angle of the torso and to translate back according to the x-axis. At the end of the movement the standing up starts its action.

3.3 Stand up function

This movement it's quite complex because many part of the body has to be managed. The first thing, of course, was to handle the movement of the torso by changing its joint angle and introducing a rotation according to the x-axis. Since the change in the model-view of the torso leads to a change in the model view of the paws, I had to managed also their related movement by rotating according to the same angle but in the opposite direction, so that when the torso stand up the paws remains down. Furthermore also the head was handled in such a way that when the bear stand up it looks straight in front of him. This function that last till $t_1 = 1$ leads directly to the scratching.

3.4 Scratch function

At the moment that the scratching starts, it has to be considered the translation up and down that the bear has to do alternatively in order to scratch properly.
So in a time going from 0 to 0.5 the bear goes up and in a time from 0.5 to 1 the bear goes down and this whole sequence is repeated several times.

While the bear moves up and down the tree oscillates backward and forward by changing the joint angle of the bark and so of the leaves. Once the action ends the user can make his choice: a message is displayed and the user is asked if he wants the animation to continue or to stop. If he chose to go on then the bear continue to scratch until the tree falls. So in the $fall_tree$ function the tree oscillates more and more while the bear scratch, repeating these change of the tree joint angle for 6 times, then, reached this time, the tree complete falls reaching the ground. This choice of action by the user was made through a flag that call the proper function.

3.5 Back to paws function

In either case, if the animation goes on or not, the movement of bear is concluded by the returning of the bear in the walking mode. So the function, opposite to the stand up, lower the angle of the torso and restore the correct position of the head and the paws.

3.6 Move paws

The most challenging movement, that can be seen whenever the bear change its position, is the movement of the paws. Studying the bear walk, I discovered that he moves alternatively the left and right paws, starting from an anterior one going to the opposite posterior and continuing with the remaining anterior and posterior. I replicated this exact same movement starting with the left anterior paw going forward and then backward, this movement is divided into upper and lower sub-movements. In the middle of this first movement, when the paw is coming backward, it starts the same for the right anterior paws. For what concern the posteriors, the right one starts right after the left anterior and the left posterior starts in the middle of the right posterior. Finally the angle of bending between upper and lower part of the paw is opposite in the posterior walk with respect to the one of the anterior.

4 Style

After that the model and the animation was handled, the last thing to complete was the style. First of all, the movement of the bear is managed by a button that when clicked starts the animation. This animation can be stopped by the same button in any time and it can restart from where it was paused by clicking again on it. As said the velocity of the animation can be increased or decreased when needed, of course only when the animation is active, and the movement of the bear can restart from the initial position in any time of the animation. Once the bear ends all his actions the only possible thing is to reset the bear to the initial position. The *reset* button can also be used when the animation is off but the bear angles has been moved thanks to the sliders, visible only when the movement is stopped. So the reset button put everything back to the initial position and orientation. Furthermore the user can change point of view in any moment by modifying the slider of *theta* and *phi*.
In order to make everything more realistic I also added, in the same way of the other models, a background that consist of a plane where the tree and the bear lye and the sky. This background, the tree and also the bear are all characterize by a realistic texture in order to make everything more lifelike.
The textures were created in the *init* function and I used three different texture for the bear: one for the body, one for the face and one for the ears; plus two texture for the tree and two for the background. Each different texture has its own image and size, while the coordinate are the same and are sent to the vertex through the *init* function.
Each texture was then called inside each drawing function where it was binded and sent to the fragment shader that, along with the coordinates, gives us back the final result.