Interactive Graphics homework 2

Pietro Manganelli Conforti 1754825

1. The hierarchical model of a Grizzly bear
2. Other models, textures and buttons
3. The animation

1. The hierarchical model of a Grizzly bear

The subject used in this project is a Grizzly bear, which performs a task where it walks up to a tree and stand up next to it in order to scratch its back. The model of the bear is created following a hierarchical model, where each element(torso, head, ears and so on) is a node eventually connected to a son and a sibling. The root of this hierarchical tree is the torso of the bear, which, for example, has as son the head, which in turn has the left upper arm as sibling and the left ear as son, and so on for the other nodes. Each node is initialized and associated by an id to an angle in the theta array, used to rotate it. At the end of the *init* function, those nodes are associated to their values(scale, position, relatives..) using the initNodes function, where a switch identifies the respective id passed and the node is created and placed in the figure array. During this creation are passed: the transformation matrix, the one which considers the rotation and translation of the element, and the render function, where starting from the modelViewMatrix the IstanceMatrix is created, taking into account the scale and traslation of the element. Next it passed to the shader as a 4vf matrix and drawn by drawArrays function, which draws the six faces of the parallelepiped, one at time. Together with those two values are also eventually passed the sibling's id and the son's id to the node that is being created. After the init those values are used during the render function, where this figure array is called by the traverse function, which starting from a given nodeId(the torso in our case) creates all the node. To consider the movement of different part of the same object the parent modelViewMatrix is pushed into a stack variable, which keeps the information to the sons and pops it before the traverse of the parent's sibling.

By doing this the bear can be viewed as a single element without the need to move each part coherently each time it moves. Moreover, tail and ears have been added to the hierarchical bear model, so are moving coherently to it while there is a movement. Next, each leg has two independent components, upper and lower part, and lower ones are scaled to fit them as paws. Those nodes are created following the described system, where the tail is a son of the torso(being the sibling of the rightUpperLeg) and the ears are sons of the head.

2. The other models, textures and controllers

The other models are inserted in the image after the bear during the render. They are all created by their respective functions and are a small tree, a big one, a bush and the plane where the bear walks. The small tree contains three parallelepipeds, one for the log and two for the crown. The bigger tree has a bigger log (chosen by the bear to scratch its back) and a crown with five elements that are built in a for cycle where each of them is smaller than the previous and placed on the top of it. Each function uses a similar syntax to the one in the render functions in the bear's nodes, so starts from the

actual modelViewMatrix and multiplies it by a scaling and a translating factor, in order to resize and locate its model. Next it is drawn by drawArrays as before, this one time for each parallelepiped inside the object.

To apply the textures, first are loaded in the html and next passed in the javascript file, where each one is configured with by the *configureTexture* function, which takes as parameter the pattern and define the property and filter modes of its application. Lastly this texture is added into the textureArray, to be picked when needed. Next, to use multiple textures at the same time, is utilized the function *useTexture* to switch the current texture with another one (from the textureArray). This function binds the new texture which has to be used before the drawArray function. For example, to change the head texture of the bear from the brown hairy one in the head function it is bonded before drawing the face and next switched back. With this solution textures are easily appliable when needed, and can be added with the same procedure, thus loading, configuring, and adding it in the array used by the useTexture function.

To control the camera inside the scene we can use six buttons: four control the radius and theta value, used to rotate the camera around the scene, and two buttons control the view volume of the orthogonal matrix, in order to zoom in or out of the scene.

3.The animation

For what concerns the animation it is a walk done by the bear to scratch its back on a tree. It can be decomposed into three principal parts: the walk, which is at first linear, next rotates a bit (to orientate the bear) and lastly becomes only a rotation until is in the proper position to stand up and put its back on the tree. Next, after a small interval, the bear stands up and starts to move its torso, legs, and arms, simulating the scratching. Finally, after it scratched for a while a button "sit down" is enabled and if pressed the bear sits and rests. All this movement are in respect to a global variable (*counter* in the code) which is incremented at each iteration, so different movement are related to different intervals of this counter. If the specific part of the animation ended is because the counter got over its interval, thus another set of commands have to be followed. For the first part of the movement the bear walks, so the arms and legs need to move forward and backward, and this is done by modifying the angle in the theta array described before. Theta values of arms, legs, torso, and head is updated at each movement by a small amount, to simulate all the movement of a walk. To make possible the alternating pattern of the movements (back and forth) a flag is used in such a way that when the angle of the legs goes over or past certain values the animation is inverted. Together with these movements (almost half of the times they move, but is scalable) a small translation and/or rotation is effectuated by multiplying the rotation and translation matrices. This information is stored into a matrix called *AnimationMatrix*. This matrix is a global variable initialized to an identity matrix and

is multiplied inside the initNode to the transform matrix of the torso, thus is inside the torso node created. In this way referencing the root of the bear each time a movement is realized it can be seen as a single element, as described above. After the scratching, the counter return inside the scratching-move interval, so the bear continues to perform this action except if the button for sitting gets pressed. In that case it at first gets back on its feet, then retrocedes a bit and next sit down, following the same methods as before.