Interactive Graphics
Homework 2

Flavia Monti, 1632488

1. **Create a hierarchical model of a (simplified) Grizzly bear, composed of the following parts;**
   a. body
   b. 4 legs, each one composed of 2 independent components (upper and lower leg)
   c. head
   d. tail
   All components are cubes, use the cube function present in the file.
   To create the model of the Grizzly bear it is used the code of figure.js file that contains all the procedures to compute a humanoid. This humanoid is then transformed into a bear.
   All the components of the humanoid are manipulated to be as more as possible similar to all the components of a bear. With respect to the humanoid it is added a new component representing the tail, this component is sibling of `rightUpperLeg` (a child of the torso) and has not child.
   All the values assigned to the heights and widths of the components of bear are divided by 3 to make the bear smaller than the size assumed with the original values. In this way there is more space in the scene to add also the tree.
   The orientation of the components is managed using `theta` vector. Some values are added to this vector: an additional value for the torso and value for the tail. The values in the theta vector are initialized in a way to put the bear like at the beginning of a walking.

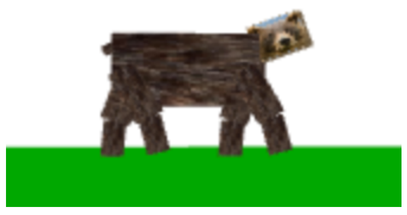

Figure 1: Walking position

   The fact that some components of the bear are overlapped is intentional, in this way the bear seems more realistic.

2. **Add a texture to the all the faces of the bear, except the head. The head has a separate texture.**
   Two textures are used and are managed with flags. In all the render functions of each components (the functions that are called when traverse function is called) there is a flag to specify if the component is the head or the body. The flag is set to true at the beginning of the function and then set to false at the end of the function, in-between there are the functions `drawArrays`, `activeTexture` and `bindTexture` that are called with the relative texture depending on what we need to render. An important aspect is the

configuration of the two textures in order to make them well separated. In particular is important the distinction between `gl.TEXTURE0` and `gl.TEXTURE1` and between the `textureMap` variables sent to the fragment shader.

3. **Create a (very simplified) model of a tree and position it near the bear.**
   To create the tree it is used the same technique used for the bear. The tree is composed by two "cubes" (a rectangle for the trunk and another rectangle for the leaves) that are initialized and added to the `figure` array. The trunk is the root component having leaves as child.
   In the render function it is called the traverse function with the id relative to the trunk and the tree is rendered.
   Also in this case there are some flags to manage the color of the trunk and of the leaves.
4. **Add a button that starts an animation of the bear so that, starting from an initial position where it is in a walking mode, it walks towards the tree by moving (alternatively back and forth) the legs, then stands up and starts scratching its back against the tree.**
   Before to implement the animation a floor (grass) is added to the scene so that the bear can walk on it. To realize the grass it is used again the approach used before.
   Looking at the technique used in the examples of the rotating cube seen during lectures, it is possible to implement the animation of the bear.
   The animation is "split" into 3 phases, a first phase of walking in front of the tree, a second phase of standing up and a third phase of scratching:
   1) Walking to the tree: in this case the values of the legs in the `theta` array are incremented or decremented to simulate the walking while the `y` coordinate and `x` coordinate of the torso are modified to reach the position in front of the trunk. Only the coordinates relative to the torso are modified because it is the root element of bear, so modifying them all the components of the bear are translated with the torso.
   2) Stand up: when the bear needs to stand up, the `theta` value of the torso and the `theta` values of the back legs of the bear are modified. During this phase the head also is rotated and also the `y` and `z` coordinates of the head are modified so that it translates to reach a great position during the standing up.
   3) Scratching: at the end of the standing up, back legs are set to the same be on equal position of bent legs. This is the starting point of the scratching.
      An issue encountered during this phase is the loop of the movement of the legs. At the beginning of the implementation was set a control based on equality (==) but then this was changed to a module (`%`) control. This because `theta` values can get floating point values so when the condition is based on equality to an integer, like 150, the condition is never satisfied.
      Then `theta` values of legs and `y` coordinate of the torso of the bear are modified to simulate the scratching.

   Below are reported two images representing the position at the top of the scratching and at the bottom of the scratching.

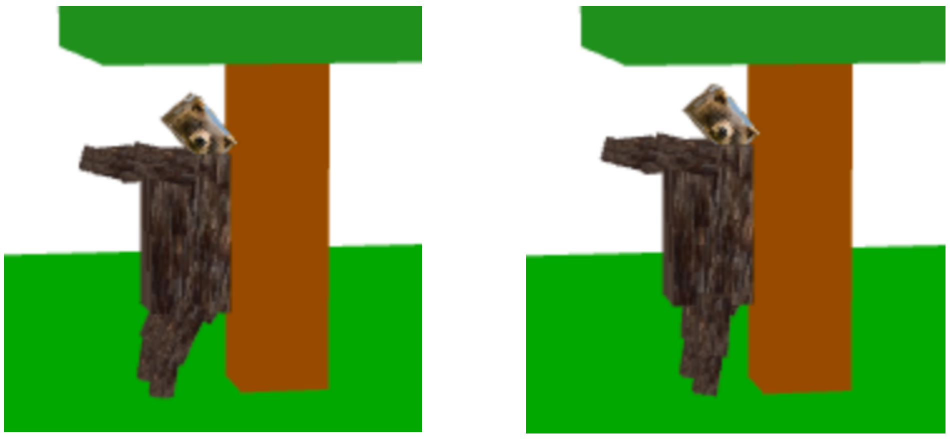
Figure 2: Bottom (left) and top (right) position of the scratching

References:

- Angel E., Shreiner D., Interactive Computer Graphics. A Top-Down Approach with WebGL 7th edition, Pearson, 2014
- https://www.khronos.org/registry/OpenGL-Refpages/