Interactive Graphics Homework 2

A.Y. 2019/2020

SAPIENZA
UNIVERSITÀ DI ROMA

Mauro Ficorella

Introduction

This relation describes the choices made for developing the Interactive Graphics' homework2; this project is written using HTML, JavaScript and WebGL.

I've to implement a hierarchical model of a grizzly bear made of a body, 4 legs (each one composed of 2 independent components, upper and lower leg), head and tail; all those components have to be implemented using cubes. I've to implement also a tree near the grizzly; finally, I chose to implement a meadow too, placed below the grizzly.

Then I've to implement an animation of this grizzly, starting in a walking mode, and then, when he arrives near the tree, starts to scratch his back against this tree.

This model was implemented adding several features to it, as described below in the solution.

Solution

Aesthetic aspect:

Regarding the aesthetic aspect of the homework, I've used different textures for grizzly's body, grizzly's head, grizzly's muzzle, tree's log, tree's leaves and meadow.

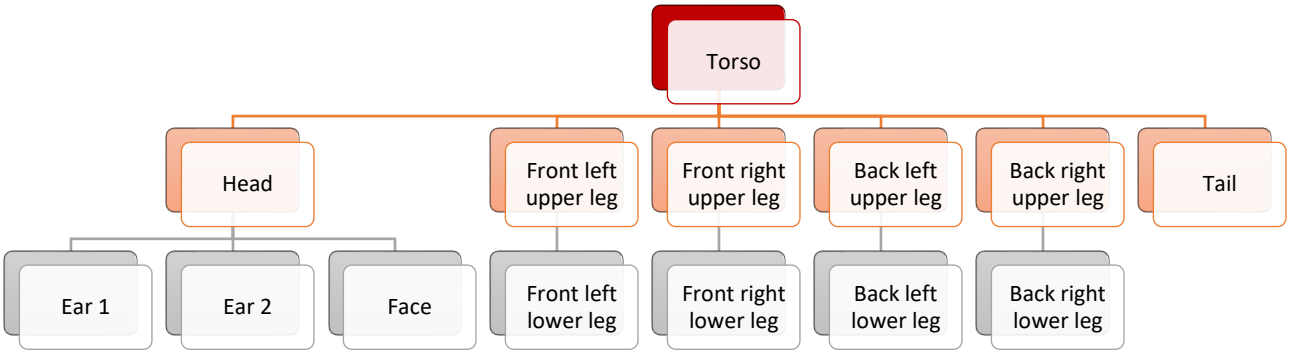
I've implemented a button that allows to start/stop the animation manually.

Requirement 1:

Regarding the first requirement, I've to create a hierarchical model of a grizzly bear, composed of a body, 4 legs (each one composed of 2 independent components, upper and lower leg), a head and a tail. All these components had to be formed by cubes.

In the case of the bear, I've added to the model also two ears and a muzzle, in order to make the grizzly more realistic.

Below there is represented grizzly's hierarchical model that I've implemented:



Requirement 2:

Regarding how I've implemented the textures, I used a different texture for grizzly's body, grizzly's head (with eyes), grizzly's muzzle (with nose), tree's log, tree's foliage and meadow.

I've used different texture coordinates in order to make eyes and nose correctly visible on the head's and muzzle's textures respectively.

Precisely, beyond standard texture coordinates used for all the other components on the screen, I made extra coordinates for the front part of the head that includes eyes, for the side parts of the head without eyes, for the front part of the muzzle that includes nose and for the side parts of the muzzle without nose. Said that, I've stored (pushed) those extra texture coordinates in two extra arrays of texture coordinates, as explained below, in order to apply them in the correct way.

More precisely, regarding all the texture coordinates that I've used, I've stored them in three different arrays, depending on which parts they regarded, in order to make them correctly visible only where needed.

Instead of only one quad function as in the original code, I created four extra functions, respectively for front head's side (side with eyes), head's sides without eyes, front muzzle's side (side with nose) and muzzle's sides without nose, in which I pushed points, colors and texture coordinates to the respective arrays; then I've used those functions into the cube function.

In order to apply different textures between body, head and the other parts (including also the tree and the meadow), I've configured those textures using six different versions of the configureTexture function, one for each texture that I've used. Then, in each function that I've created for each node of my grizzly's and tree's hierarchical model, I've used various flags in order to apply different textures only where it was needed and not everywhere; for example, in grizzly's torso function, I wrote that only texture regarding grizzly's fur had to be applied setting only "flagTexGrizzly" flag to true and all the other flags to false.

More in details, I've applied, within these functions, the following reasoning: I've specified the texture to initialize only in father and, unless texture was different from father's, in the first son (starting from left in the hierarchical model tree); then in the other sons, if the texture was the same of other brothers', I've only specified flags to set which texture shouldn't be applied; otherwise, if texture was different from other brothers', I've reinitialized it in the proper way.

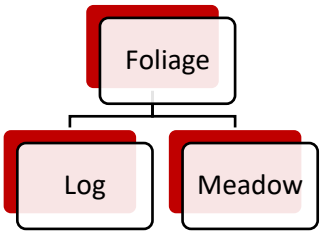
Those flags, then, are used in the html file to set each figure's color in the fragment shader as it follows in this fragment of code regarding grizzly's fur:

```
vec4 grizzlyColorTex = texture(texture0, vTexCoord);
if (flagTexGrizzly == true)   {
        fColor = grizzlyColorTex;
}
```

Given the fact that I've not found any texture on the internet that suited well with head and muzzle that included eyes and nose, I've decided to draw them on my own, using an iPad with Apple Pencil, in order to manually match them good with my geometries, taking inspiration from the famous game "Minecraft".

Requirement 3:

Regarding the tree, I've made a very simplified model as it follows:



I've also created a different matrix, t matrix, to differentiate grizzly's and tree's models, as it follows:

```
var m = mat4(); //grizzly's model matrix
var t = mat4(); //tree's model matrix
```

Indeed, also for the tree, I've used three different textures, one for the log, one for the foliage and one for the meadow, implemented in the same way as specified in the previous requirement regarding the grizzly.

Requirement 4:

In order to make the required animation, I've added a button that starts/stops animation by a click.

Regarding the animation, I've used different functions, in which I've modified, by hand, using increments, various nodes' theta and position variables, in order to obtain the required movement and translation of legs and body.

More precisely, I've used four different functions in order to make the whole animation, as described below.

One function is "walkFunction", in which I increased/decreased leg's angles in order to make them move as coherently as possible; in this function I've made a subdivision into two phases, one for legs' ascent and one for legs' descent; indeed, when the grizzly has arrived to a position in which step variable is major of 10, I've started increasing torso's theta until reaches 90 degrees, in order to let the grizzly rotate, horizontally, before it stands up, and, in order to stop the rotation, I've used a flag set to false when torso's angle reaches 90 degrees.

Another function is "posizioneArrivoFunction" that starts when grizzly arrives near tree's log and starts standing up to scratch against it; here I've modified grizzly's torso1 angle in order to let the grizzly stand up next to the tree until it's parallel to tree's log, and, simultaneously, I've modified also head angle to let it move coherently without making it fixed during movement; I've also increased step value in order to let grizzly go near the trunk during the stand up phase; indeed I've made grizzly's legs angulation vary coherently with the fact that it stands on its rear legs. In order to stop this phase, also here, I've used two flags, one to stop modifying legs' angles, and one to start scratching phase, described in the below function.

Then there is the function "scratchFunction", in which I let grizzly start scratching against the tree. Until count is equal to 10, I've let the bear start scratching against the tree by increasing/decreasing grizzly's torso height; I've also decreased/increased grizzly's legs height in order to let them stay fixed in a coherent position during the scratching phase; indeed, after count reached value 10, I've let grizzly go back to its four-legged position, and when its torso angle reached value 0, I've set "flagGoBack" to true in order to make it start returning to its starting position and "startScratch" to false in order to stop scratching phase.

Indeed, there is "posizioneFinaleRitornoFunction" that starts when grizzly returns to its starting position; more deeply, I made grizzly's legs return to its starting angulation; after that I've set flagArrivato to false in order to stop legs' animation.

For example, in order to make grizzly's torso stand up, I've created another torso node (torso1Id) in which I've modified the axis of rotation, as it follows:

```
m = mult(m, rotate(theta[torsoId], vec3(0, 1, 0) ));
m = mult(m, rotate(theta[torso1Id], vec3(1, 0, 0) )); //there is torso's
rotation axis change
```

As I've wrote before, I've also used some flags in order to start/stop and keep track of different animations without the need to add other buttons. This method has the main advantage to control every single movement in the expected way and synchronism, without issues.

Indeed, I've invoked all these functions into the "render" function; more precisely, I invoke all these functions if "flagAnimazione" is true, in order to start and stop animation in every moment by only a click on the screen; then, if step is lower than 22 and "walkFlag" is true, I start "walkFunction" in order to make grizzly start walking; then I increase grizzly's position (step variable) by 0.35 at a time; when the step variable reaches value 22, I set "walkFlag" to false in order to stop "walkFunction" and, if "legFlag" is true, I invoke "posizioneArrivoFunction" in order to let the bear start standing up, if "startScratch" is true, I invoke "scratchFunction" in order to let the bear start scratching in the proper way against the tree; then until the step variable doesn't reach the value -20 (starting value) and, at the same time, "flagGoBack" is true (it means that the grizzly has finished scratching against the tree), the grizzly starts returning back (using "walkFunction" to modify legs' angulation as before) to its start position decreasing its position by a 0.35 decrease at time; finally, when grizzly reaches final position, "flagArrivato" becomes true and, if "flagArrivato" is true, I let grizzly's leg angles return to their starting values and, now, I can set "flagGoBack" to false in order to finally stop the whole animation.