Interactive Graphics
Homework 2

Lorenzo Nicoletti - 1797464

May 26, 2021

Remark: the code has been tested only on Mozilla Firefox and on Microsoft Edge. Please, execute the command *python -m http.server* on a terminal to load the textures in the JPG format.

Exercise 1

The hierarchical model of the sheep is represented by the graph tree in *Figure 1* below. It is composed by a total of thirteen components, eleven nodes are the mandatory ones according to the requirements. Two more nodes, namely 'Left Eye' and 'Right eye', have been added to give realism to the animal. The root is the torso of the sheep that has six children, grouped in the four upper legs, the head (father of the two eyes) and the tail. Each upper leg is the father of the correspondent lower leg. The notation I have used for the legs is the following:

- Left/Right Upper/Lower **Arm** correspond to the front leg.
- Left/Right Upper/Lower **Leg** correspond to the rear leg.

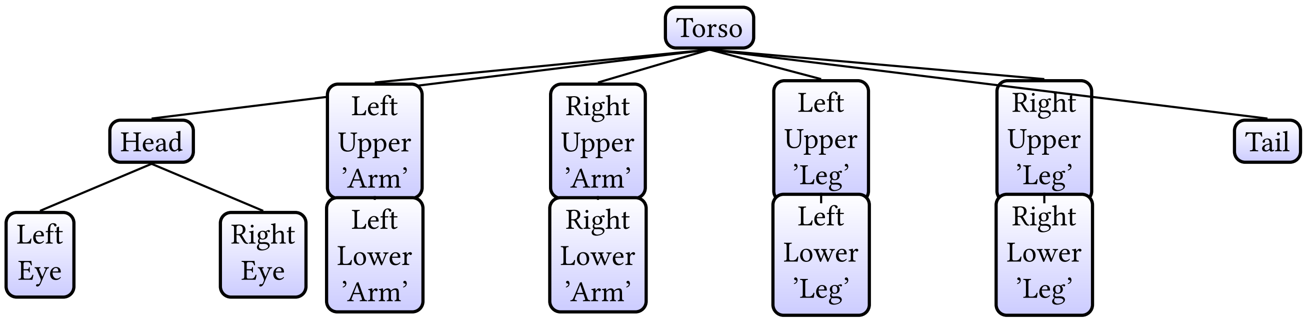

Figure 1: Hierarchical model of the sheep

The color of these components has been temporarily set to white and its final version will be later outputted in combination with the color/bump textures.

Exercise 2

The surface of the grass field is rendered as a separated object in the hierarchical model, namely it is just a single-node tree with no child or sibling. This choice has required an additional call of the *traverse* function while rendering, specifying the 'id' of the grass.

The grass texture is loaded as a JPG image in the html file and is simply read by the JavaScript through its identifier. The picture is then associated to the corresponding texture, specifying all the required parameters. Finally, the texture is bind and the 2D sampler is passed as a uniform to the fragment shader. Moreover, this shader has another uniform boolean flag that is activated in the *grass()* function when the rendering of the grass is performed in order to be able to distinguish among all the several textures and samplers involved in the application.

Exercise 3

The handling of the two textures of the sheep (body and head) has been done very similarly to the one of the grass field: the two associated 2D samplers and activation flags are set in their rendering functions (*head()*, *torso()*, *leftUpperArm()*, etc.) in order to make the shaders 'understand' which part of the scene they are creating.

On the other hand, the main aspect is the creation of the two textures of the sheep. The head texture is a very simple color texture that is created by specifying its RGB components. This generates a (*1 x 1*)-dimensional texture that is applied to every fragment composing the head of the sheep. To give a realistic effect, the texture is attached only to the front face of the head. This is easily done by properly setting the texture flags during the rendering, in particular during the execution of the *drawArrays()* invocations.

The introduction of the bump texture for the wool has required the definition of a light that consequently affects the other objects in the scene too. In order to not stress the lighting too much, I have decided to apply some simplifications to the standard lights definition.
While the texture is created as usual, namely with a pseudo-random generator to initialize the *data* array and give the 'wool effect' and a conversion to the normal vector to create the (*texSize x texSize*)-dimensional bump texture, the simplification lies in how the light is modeled: I have assigned only an ambient and a diffuse property to the light (neglecting specular and shininess terms for the sake of simplicity). In particular, this is replicated for the two identical lights I have introduced located in two extreme corners of the scene to avoid shadows while varying the viewer position (please, refer to *Exercise 6* section). Therefore, for each specific object, I have defined also a material ambient and a material diffuse that will be multiplied by the light parameters to define the ambient and the diffuse products to be passed to the shaders as uniforms to determine the final amount of illumination for each fragment.

Finally, if the rendering of the body of the sheep follows the bump routine, the remaining parts of the scene are rendered through a standard Per Fragment lighting model (with only ambient and diffuse terms again).

Exercise 4

The fence is modeled again as a separated hierarchical object composed of six components, namely its six posts. Here, the hierarchy is very simple since there are only siblings. As done for the grass, another call of the *traverse* function is required in the rendering, specifying the 'id' of the first post of the fence. The texture is created from a JPG image and is activated, bind and handled by the shaders exactly as done for the other textures, namely with a personalized 2D sampler and a flag activated in the *fence()* rendering function that is shared among all the posts.

Exercise 5

The animation is the crucial part of the homework. It starts at the click on the button that calls the *animate()* function, the routine to proceed in the animation. It is inspired by the traditional *render* function because of the invocation of *requestAnimationFrame(animate)* in its last line. The animation exploits the advantages of the hierarchical model when moving a single part of it: since the torso is the root, then translating it along the x-axis will imply a motion of the whole body in the scene. Therefore, I have divided the animation in stages according to the x-position of the torso that is slightly increased for each animated frame:

- in the first stage, the sheep needs to arrive close to the fence. The motion of the legs is quite tricky: looking to some videos of walking sheeps, I have observed that the front left and rear right legs move quite simultaneously as for the front right and rear left legs. So, I have implemented the walk accordingly, with the coupled legs moving back and forth. The head and the tail move/rotate too.
- the jump step is divided in an ascending and a descending phase, recreating the appearance of the jump: the torso rotates a bit and during the elevation the legs are fixed. The landing phase is just the complementary of the first phase.
- once landed, the sheep does some other steps until a point on the x-axis is reached, then it stops and the animation is completed.

For each execution of *animate()*, a for loop over the tree nodes is required to visualize the displacements between consecutive frames, by calling again the *initNodes()* method.

To finally recreate the initial scene before the animation, the user just needs to re-click on the same button, that simply reloads the window in its starting configuration.

Exercise 6

The user is allowed to move the camera whenever he/she wants by interacting with the provided sliders that set the values of the camera angles, width, height and depth. The angles are involved in the computation of the viewer position that is used in the *lookAt(viewerPos, at, up)* function to initialize the model view matrix. Moreover, the projection matrix is kept in its starting orthogonal viewing with varying values of the parameters according to the desired configuration of the user. This configuration can vary before, during and after the animation.