Exercise 1

The following hierarchical structure has been created (includes the sheep for exercise 1, the field for exercise 2 and the fence for exercise 4):

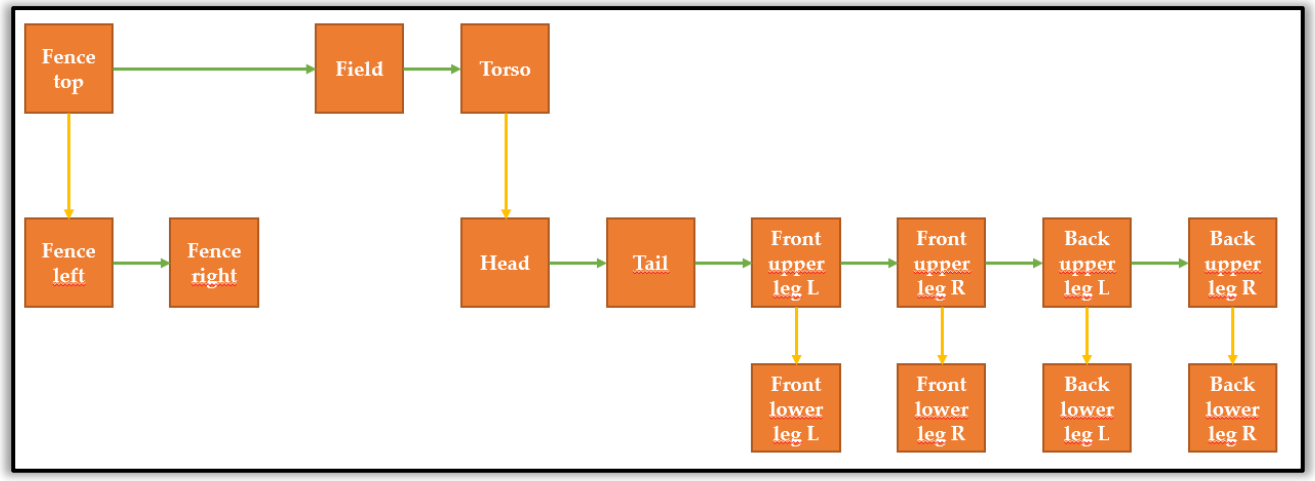

Figure 1. The green arrows represent the relationship between siblings, the yellow ones between father and son.

The initNodes() function also generates the MV matrixes for the individual objects. These are the main transformations calculated:

- The torso can rotate with respect to the y-axis and the z-axis. However, the rotation with respect to the z-axis takes place by taking as a pivot the centre between the joints of the legs with the torsoleg (forward or backward depending on the value of a variable "jumpPhase" which assumes 1 or -1). This allowed a good animation of the jump as in the first phase it soars on the back legs and in the second phase it lands on the front ones.
- All upper legs were positioned 25% of the torso, i.e., in the middle of the half. (the position in front, back, right, and left changes depending on the leg).
- All upper and lower legs and the tail rotate around the z-axis with a pivot positioned on their joint.
- The head rotates around its centre and is positioned in front of the torso in such a way that half the head protrudes upwards.

All the upper and lower legs and the tail are interpenetrated a little in their parent object (i.e., the tail in the torso, the lower legs in the upper legs and so on). This is so that when they rotate you do not see that they are detached. The interpenetration value is calculated based on a borderline case, i.e., when the child object is at 90 degrees to the parent. In this situation the object will protrude by half its width. To avoid this, simply interpenetrate by half its width.

The material of the sheep was defined by taking an exaggerated white as ambient material and a dark grey as diffuse (almost no specular component). This material was attached to all the faces of the sheep except for the lower legs and the face (Exercise 3). For the lower legs, a material tending towards light brown was used.

Exercise 2

A plane z = 0 of size 100x100 represents the field. This field has a bright green material with very few reflections to which a 512x512 bump map has been attached. The normal map that defines it takes as its initial texture a texture generated by the function:

$$data(i,j) = \frac{1}{Math.random() * Math.sin(i+j) * Math.cos(i+j)}.$$

In the vertex shader the tangent coordinates TBN have been calculated with which in the fragment shader the final colour has been calculated taking as normal the normal map passed as texture.

Exercise 3

I generated an 8x8 texture for the face by hand using the sheep from the game "minecraft" as a reference point. For the wool parts, a 64x64 bump map was made taking as initial texture:

$$data(i,j) = Math.random() * Math.cos(j)$$

(striped effect with random component to give a wool-like effect).

For the sheep's feet a 32x32 bump map with 0.3 in columns multiples of 3, 0 otherwise.

To distinguish the face with the rest of the cube representing the head we made a drawarray() with a bindtexture(face texture) and then a drawarray() of the rest with bindtexture(wool texture) and passed as uniform uface, respectively 1 and 0. In the fragment shader if uFace is 1 normal is the normal of the object and the position of the light is the position of the light, otherwise it is a bumpmap and so for the normal we use the passed texture and for the light the coordinates calculated in TBN in the vertex shader. For the bump map the final colour is the sum of ambient, diffuse, and specular, for the face, this sum is multiplied to the texture producing the lightning effect in shadow. However, as the resulting colour is very dark, this result has been multiplied by 1.2.

Exercise 4

To create the fence, three "sticks" were used, two vertical and one horizontal (father of the vertical ones). To give three-dimensionality, the horizontal one was given a slightly higher width. A bump map was also applied to the fence, taking the following as the initial texture:

$$data(i,j) = Math.cos(i+j)$$ (oblique wood grain).

Exercise 5

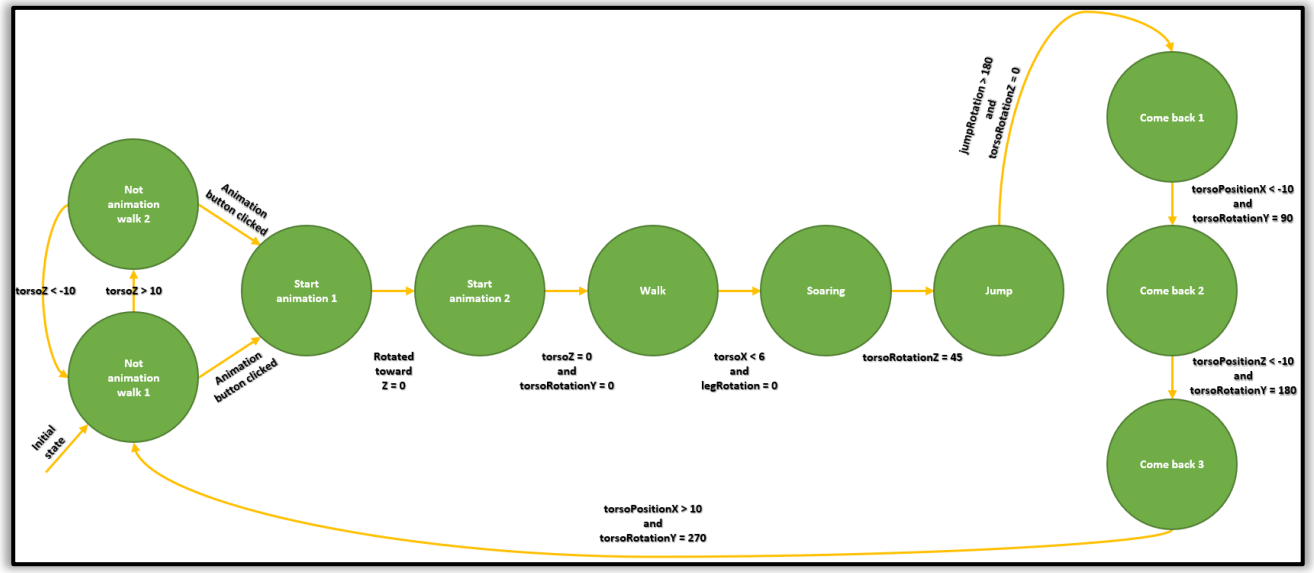

Figure 2. Finite state automata representing animation. The circles are the animation steps and the arrows are the transaction functions.

The animation was done using a finite state automaton as shown in Figure 2. Transaction functions are used to switch from one type of animation to another through an event or a certain verified condition. Before talking about the main animations, I would like to talk about the walking

animation that is used in several animations. This is done by moving the legs (upper and lower) in a crossed way (front left with back right and front right with back left): two forwards and the other two backwards and vice versa after a certain angle of rotation. This reasoning was also applied at different speeds to the head and tail to make the sheep seem more animated.
There are five main animations:

1. **Waiting**. It includes the states of notAnimationWalk 1 and 2 and in this animation the sheep is in walk mode and walks back and forth at coordinate x = 10. The NotAnimationWalk1 state is the initial state.
2. **Animation started**. It includes the startAnimation 1 and 2 states and in this state the sheep turns towards z = 0 (with the optimal rotation), walks towards that point and turns towards the fence. This animation is triggered by the click of the animation start button.
3. **Approach the fence**. It includes the walk state, and with the walk animation the sheep approaches the fence. This animation starts from **Animation started** when the sheep is in position z = 0 and rotated towards the fence.
4. **Jump**! Includes soaring and jump states. The sheep stands on its back legs rotating around the axis above the back legs, jumps and halfway through the jump repositions itself upright, in the second half, rotating around the axis above the front legs, it lands on the front legs and finally repositions straight on the other side of the fence. This animation starts when the sheep is close to the fence.
5. **Come back**. Includes comeBack states 1, 2 and 3. in these states with the walk animation after landing it makes a square turn until it returns to x = 10 where it starts again with the Waiting animation. This animation starts when the sheep has landed upright after the fence.

Exercise 6

The user will be able to move the camera during any phase of the animations through sliders that will change the viewer's phi and theta angle. The user can also change the fovy, again using a slider.

Extras

I added as an extra a slider to increase and decrease the fps from 60 to 144 (default 90) so that you can increase or decrease the speed of the animations. Another reason is because on FireFox the frame rate is much lower than on chrome and this slider allows a clean display in both browsers.
Instead of having a fixed light with a constant colour, I switched to a day-night cycle. The light represents the sun and is positioned at coordinates (sunR*cos(sunT), sunR*cos(sunT), 0) where sunR is the distance from the origin and sunT is the angle, i.e., "the time of day/night". If sunT is ≤ 20 we are in the period when light begins to rise from the darkness of night, vice versa if >160 it is becoming night and therefore the period when everything gets darker. From 20 to 50 there is sunrise, from 130 to 160 sunset. From 50 to 130 is daytime. Every time the angle drops below 0 it is set to 180 and we go from day to night or vice versa. Night has few changes and therefore passes 10 times faster. The colour of the ambient light is dark grey during the day and is halved at night, making everything darker. Diffuse, specular light is orange for sunrise and sunset, night blue at night and white during the day. The shading from one phase to another has been done by interpolating from one colour to another knowing the number of change steps (the change step for this interpolation will therefore be different for each RGB channel). The same reasoning has been applied to the clear color with the only difference being that the colour of the day is light blue. This day-night cycle has been made to move and change the light constantly to better see the changes taking place on the bump maps.