1. **Creating a hierarchical model of a cat**
   Starting from the initial hierarchical model given by the homework, which was providing head,torso, two arms and two legs. Arms and legs were composed by two parts an upper and lower part. To implement the cat I started from modifying the functions torso() and head(): these two use a cube and applying a translation and in particular a scaling to it I gave the correct shape. An easy way to modify the dimension of this parts was to simply add some dimensional parameters like width and weight. Then to implement the four legs i decided to move the arms of the initial robot, using a translation inside of the switch case to move them under the torso. Instead for the tail I simply implemented 3 pieces. So the hierarchical model was composed by : torso was the root , all the pieces like head, front upper legs, and back upper legs and the first piece of the tail were siblings and children of torso. Instead front lower legs and back lower legs were children of the respective legs.The second and third pieces of the tail were children of the first one. It was also necessary to add other positions for the array theta, corresponding the number of nodes. This array is needed because it gives the orientation for the pieces of the cat.

2. **Creating a carpet with a color and bump texture**
   For the carpet I simply created a function Carpet(), which was using in particular, the scaling that was stretching the cube to create a cube with a greater depth and width. To attach the textures I simply found on internet a texture of a carpet and create a bump version of it. To implement bumping, I used a light and Calculated the eye-space position of the vertex (eyePosition) by multiplying the vertex position with the model-view matrix and extracting the XYZ components. Calculated the eye-space position of the light, the normalized normal vector (N), normalized tangent vector(T), the bitangent vector (B), the light vector in texture-space coordinates (L) and view vector in texture-space coordinates.

3. **Creating a bump textures for the body and a color textures for the face.**
   I have applied the same technique used for the carpet, but in every draw function I send a parameter "useTexture" to the fragment shader to decide which to apply to each part of the body in order to apply the bump texture to the body and a color texture for the face.

4. **Creating the table.**
   I used a hierarchical model to implement the table , the root of the tree was the

plane and then I implemented the four legs like the children. To implement this I created an appropriate variable called figure1 which contains all the nodes of the table and I traversed it with traverse1() function calling always the function planeid()

5. **Animation: cat start to walking, jump on the table and walk to the center.**
   Creating the animation I decide to divided it in different in step: first the cat started to walk until it arrived to a certain position. To implement this first walking phase, I used the mix() function which use as parameters respectively : initial position of the cat , desired final position which is spaced from the table. Last parameter which regulates the interpolation, is used to regulate the pace of the cat. This function will modify the function that translates the torso, and during this translation I controlled also the angle of the legs using an if. This if control if the angle of the legs is less than 150 degrees then i have to increase the angle until 220 and go down again. Once it is arrived to this first final position, legs returned to a position of 180 degrees, the cat incline torso and the back legs to take the momentum for jumping. All of this movement of the legs are done using mix() from the actual angle to the final angle of the legs which are contained in theta[], instead the jumping is done by increasing Y-position and X-position. After this I repeated the first phase until it arrives to the center of the table. Once it is arrived at a certain height starts a new phase: I used again mix to take down the cat modifying the Y-position. Once it is arrived to the surface of the table starts a new phase of the animation of the cat, in which again we translate the torso of the cat and varying the angles, back and forth of the legs.
   This animation is controlled by a button and its relative event in the javascript. Whenever the button is pressed it calls an animation inside the render.

6. **Adding a camera position and a perspective**
   Adding the camera was about to create a ModelViewMatrix and setting all the classical parameters like eye,at,up defining it using the lookat(). To move the camera I simply added some sliders which control theta,phi and radius to modify the eye parameter of the lookat(). Instead for projection i decided to use a perspective projection, adding fovy, aspect, near and then a respective slider to control the fov parameter.

7. **Applying motion blur based on a button**
The idea behind the motion blur was to use a framebuffer and then a render buffer. Doing this allowed me to render first the entire scene inside a buffer which is not visible(so called off-screen). More precisely I rendered my scene inside a texture, the so called render to a texture. Indeed i created a texture "texture5" I attached it to the frame buffer as a color attachment for the framebuffer. I had also to define a render buffer to handle the problem of the depth. After this I created the usual buffer for textures and vertices. Now I was able to call DrawScene(), which draws the entire scene inside the framebuffer. One of the most important things to implement correctly was to create 3 past frame which allows me to the compute the switching between the frames. To implement the motion blur, I have to create a vertex and fragment shader dedicated for this effect. In the vertex shader, which was very simply takes as input the position of the canvas and texture coordinate. The real application of it was inside the fragment shader, it controls the values of the 3 past frame and one current frame and computes the average between the old frames. Then I mixed the average with the new image to create this effect. When you launch the application, it starts without the motion blur so if we start the animation you will not see any effect. To activate it you have to push the button "activate motion blur". In order to see better this post-processing effect i suggest to : before starting the animation move the camera near to the cat and behind it. Then activate the button motion blur and start the animation , in this way the cat starts walking and you will see a blurred effect on the movement of the cat. Another way to see better the motion blur, is to active the motion blur and even with the cat still, moving fast the camera.

P.S. : **I programmed all the application in a Mac OS and there was no problem, but I also tried to run my application using a Windows OS and sometimes I get this error : WEBGL-LOSE-CONTEXT. To handle with this I added an event function to reload this page once this error occurs.**