Interactive Graphics

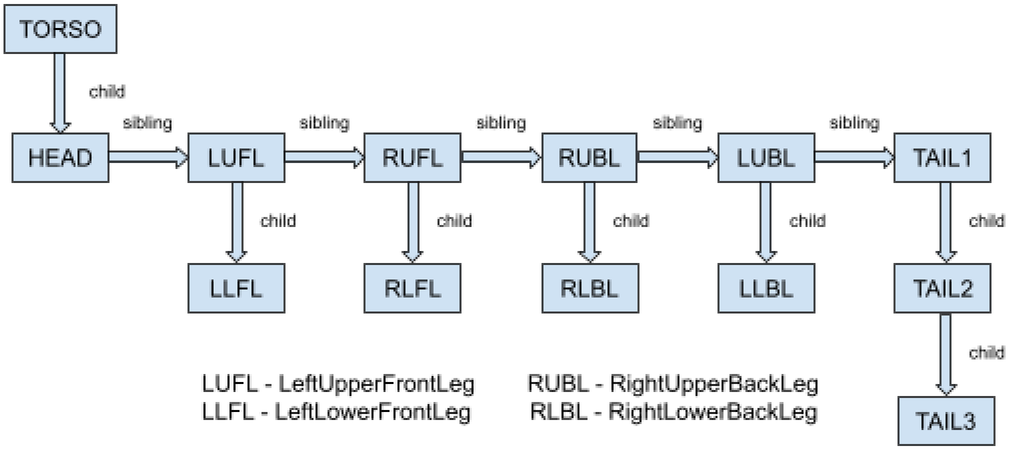
Small project - 2

1. Create a hierarchical model of a (very simplified) cat, composed of the following parts;

a. body
b. 2 upper legs, each one composed of 2 independent components (upper and lower part)
c. 2 lower legs, each one composed of 2 independent components (upper and lower part)
d. head
e. tail (composed of at least 3 components)

All components are cubes, use the cube function present in the file. The cat has a light gray color.

The hierarchical model of a cat is composed in the following hierarchical model:



LUFL - LeftUpperFrontLeg
LLFL - LeftLowerFrontLeg

RUFL - RightUpperFrontLeg
RLFL - RightLowerFrontLeg

RUBL - RightUpperBackLeg
RLBL - RightLowerBackLeg

LUBL - LeftUpperBackLeg
LLBL - LeftLowerBackLeg

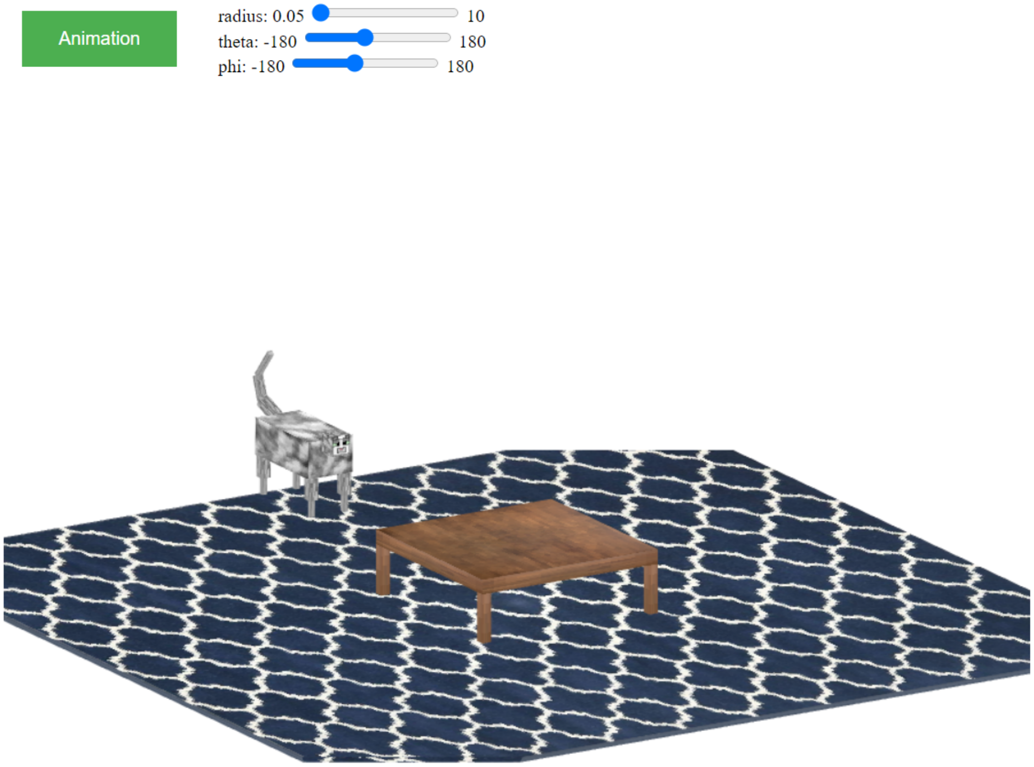
There is the main part of the model - body, i.e., torso. Each part of the model is a node and each node is linked to at least one other node. The Torso is a parent node and has no sibling, a Head node is linked to the Torso as a child and has no child. The LeftUpperFrontLeg node is linked to the Head as a sibling and has a LeftLowerFrontLeg as a child node, and so on. The Tail nodes consists of three parts: Tail1 node is linked to one of the legs as a sibling node, in this case it's a sibling to the LeftUpperBackLeg, the Tail2 is the second part of the tail, it's just an extension of the Tail1, but a little slimmer, it is linked to Tail1 as a child, the same is for third part of the tail-Tail3, it is linked to Tail2 as a child.

We need to first change the position and orientation of the parent node Torso by specifying values in translation and rotation functions according to the surface's position and cat's height. The sibling nodes change their positions and orientations according to their parent and sibling nodes width/heights and orientations and child nodes according to their parent nodes.

The color of cat is specified as a light gray.

2. Add a surface on which you position the cat that corresponds to a carpet. Attach to it a bump and color texture to give the appearance of a carpet.

The surface is composed of a single 3D cuboid figure. The bump texture is generated as a data of random bumps of range from [0, 100] and applied on the surface in fragment-shader along with a color texture as an uploaded image of a carpet. The texture data is random bumps of range from 0 to 100.

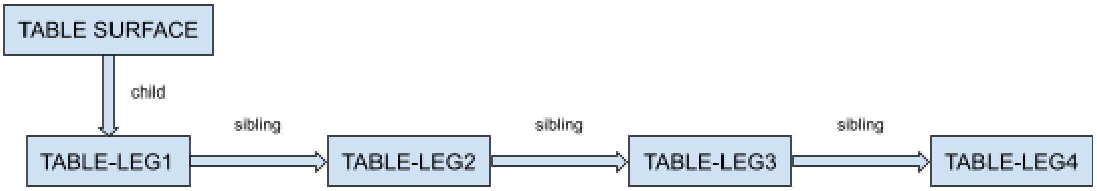


3. Load or generate at least two more textures. A color texture is attached to the front face of the head and a bump texture is applied to the sides of the body.

The same generated bump texture which is used for the surface is applied on the cat model along with a loaded image which represents the fur of the cat in mixed gray-white colors. The color texture is applied on the front face of the cat.

4. Add a (simple) table on top of the central part of the carpet.

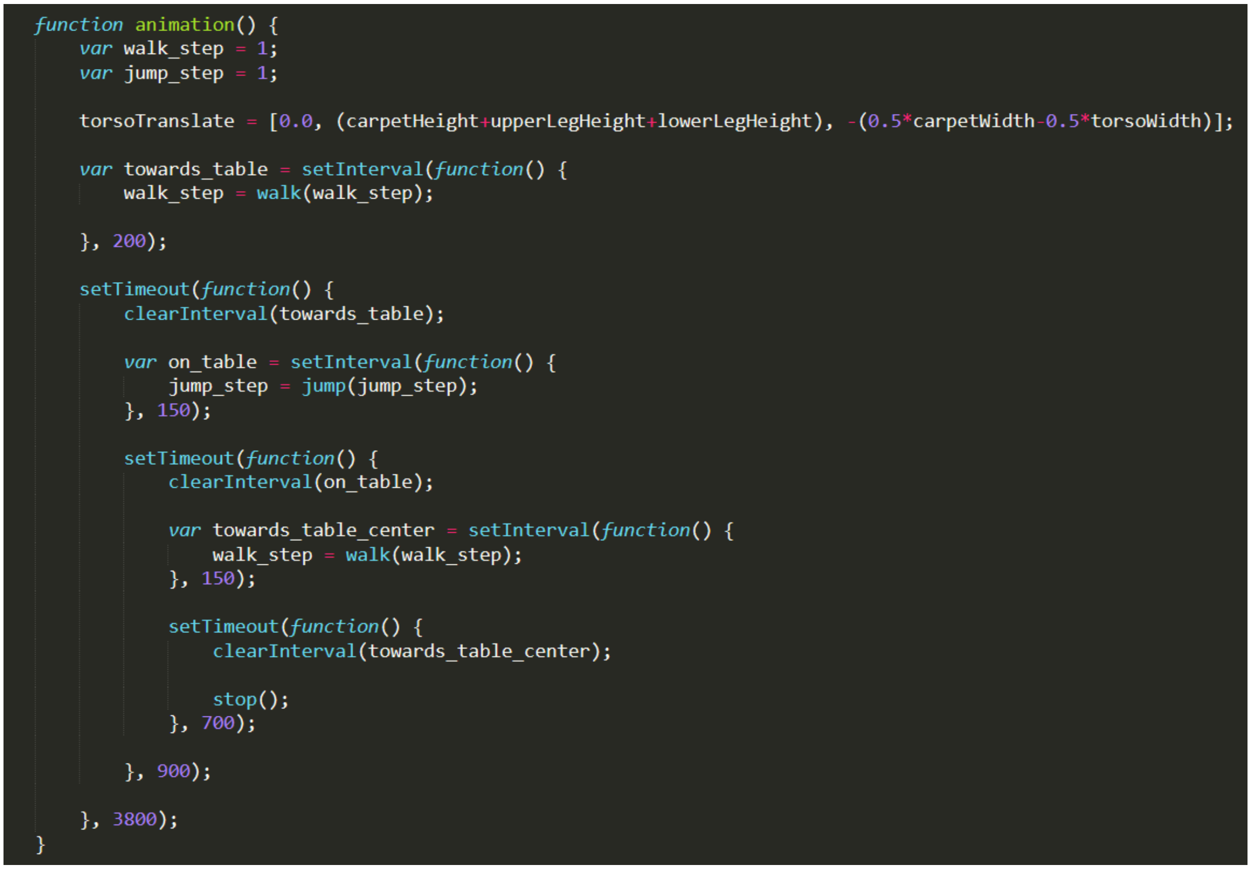
The table is added on top of the central part of the carpet. The (simple) table is composed as another hierarchical model in following way:



The loaded texture of the wooden image is applied on the table.

5. Add a button that starts an animation of the cat so that, starting from an initial position it walks towards the table, then jumps on top of the table, and then moves to the center of the table and stops there.

The animation is a set of consequent scenes, which means if we change each scene a bit and render them consequently we will get an animation.

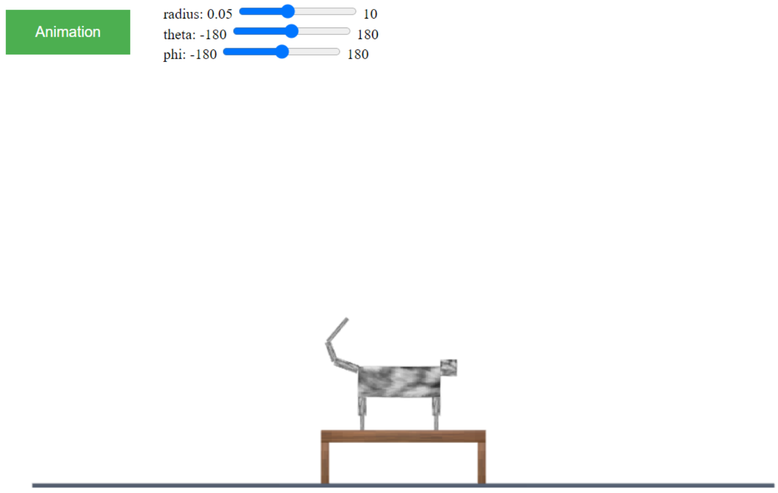


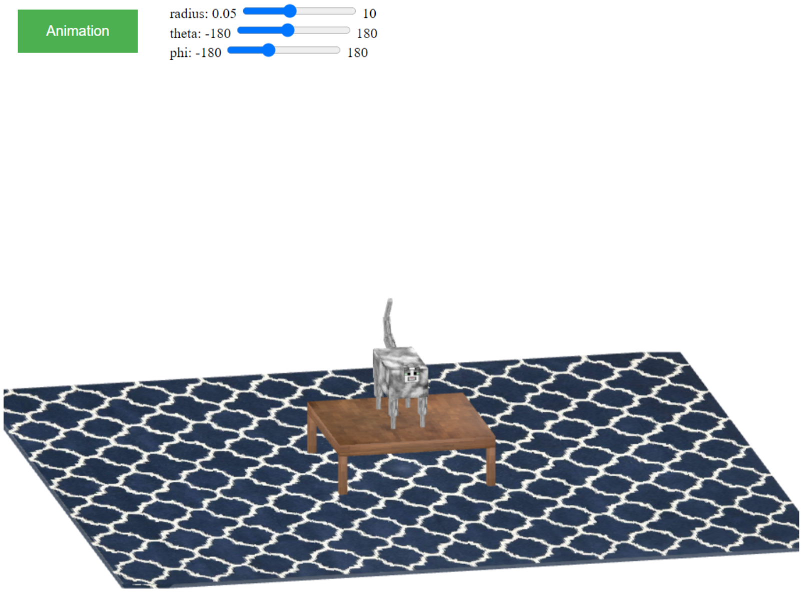
```
function animation() {
    var walk_step = 1;
    var jump_step = 1;

    torsoTranslate = [0.0, (carpetHeight+upperLegHeight+lowerLegHeight), -(0.5*carpetWidth-0.5*torsoWidth)];

    var towards_table = setInterval(function() {
        walk_step = walk(walk_step);

    }, 200);

    setTimeout(function() {
        clearInterval(towards_table);

        var on_table = setInterval(function() {
            jump_step = jump(jump_step);
        }, 150);

        setTimeout(function() {
            clearInterval(on_table);

            var towards_table_center = setInterval(function() {
                walk_step = walk(walk_step);
            }, 150);

            setTimeout(function() {
                clearInterval(towards_table_center);

                stop();
            }, 700);

        }, 900);

    }, 3800);
}
```

The animation is done by changing the values of the translation and rotation functions of the cat model and rendering each change consequently. The time control is done by using setInterval() and setTimeout() functions. These functions allow us to change the movement of the cat: from walking towards the table to jumping after specific time, and then after jumping on table to walking to the table center and after timeout stop there. setInterval() allows re-rendering each change, and setTimeout() allows to change the animation movement.

6. Allow the user to move the camera before and during the animation.

For viewer position lookAt() function is added in order to allow the user to move the camera before and during the animation. lookAt() function consists of eye, at and up vectors: lookAt(eye, at, up). The eye vector consists of radius (distance of the camera from the origin) and angles theta and phi to move the camera. By changing the values of theta and phi the viewer can see the scene from different sides.


Animation
radius: 0.05
10
theta: -180
180
phi: -180
180


Animation
radius: 0.05
10
theta: -180
180
phi: -180
180