Interactive Graphics Project

SEPTEMBER 2020

VIRGINIA SANFILIPPO, 1652696

Contents

1- Introduction

My final project for the Interactive Graphics course consists of a simple web game, with a rabbit that has to collect as many Easter eggs as possible, jumping to avoid falling into the wells.

2- User Manual

2.1 COMMANDS

In the home page there are instructions and credits, along with a preview of the rabbit model.

Starting from the home the user can choose to pick the scenario and lights to be either in a day or in a night setting, then start the game with the *Start* button.

In the game, on the upper left side of the screen it'll show the current score, which is the number of eggs that have been collected by the rabbit this far.

If the user wants to end the game, he can either click on the *Stop* button, which will show the final score and end the game, or directly on the *Home* button, which will reload the initial page.

2.2 HOW TO PLAY

Once the game is started, the user can jump using the arrow up key. The goal of the game is to collect as many eggs as possible, jumping to avoid falling into the wells

3- Environment

3.1 LANGUAGES

The project is written in JavaScript and HTML.

3.2 LIBRARIES

- *Three.js*: a library that simplifies creating 3D content for a webpage, that was used to create the scene, add the objects and create their geometries, add lights and textures.
- *Tween.js*: a tweening library for JavaScript use, that was used to help smoothing the animations.

3.3 OTHER IMPORTED TOOLS

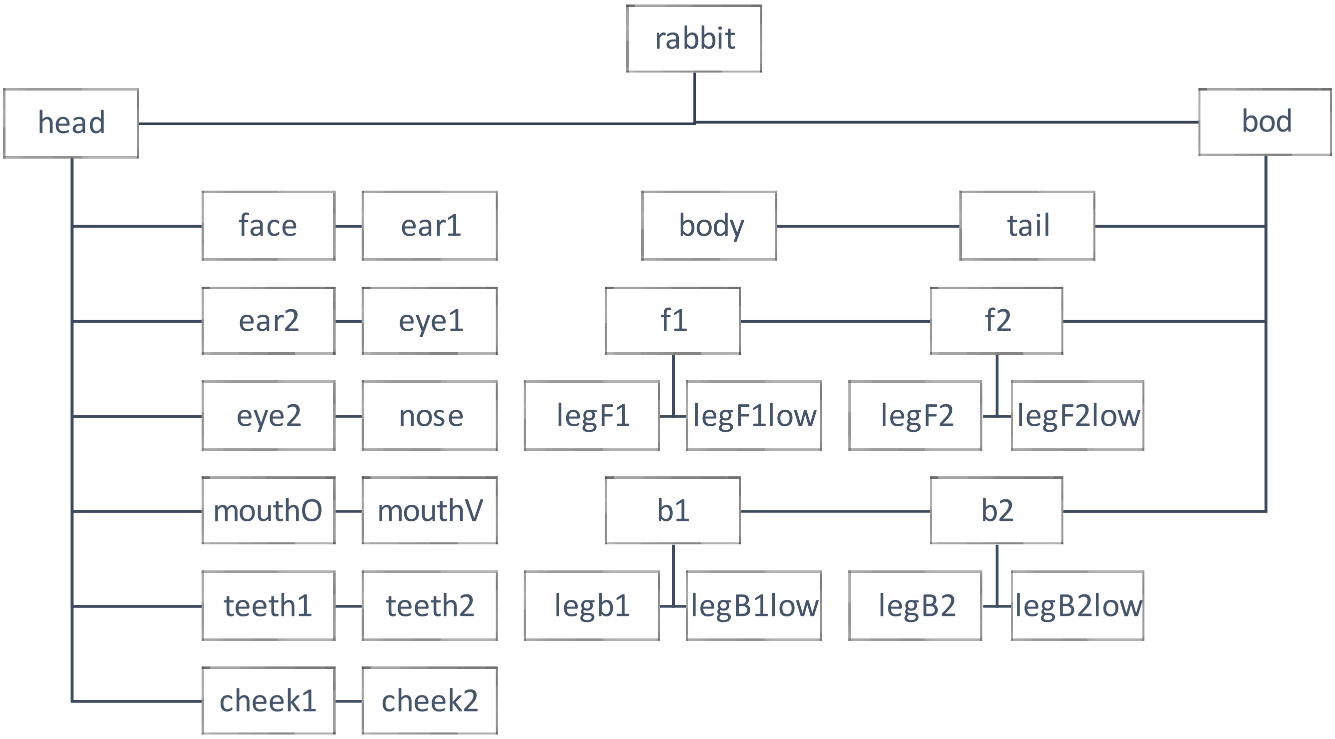
The project includes an imported equation for an egg-shaped geometry:

```
var points = [];

for (var deg =0; deg<=180; deg+=6 ) {

  var rad = Math.PI * deg / 180;

  var point = new THREE.Vector2((0.7 + 0.1 * Math.cos( rad)) * Math.sin(rad), -Math.cos(rad) );

  points.push( point );

}

eggGeometry = new THREE.LatheBufferGeometry(points, 32);
```

4- Technical Aspects

4.1 HIERARCHICAL MODELS

The hierarchical model approach used to create the rabbit made it easier to position and move the rabbit through the game.



The hierarchy is implemented through Three.js groups. The “rabbit” group includes all the elements and allows to change the position and rotation of the whole model.

```
function drawRabbit(){

   rabbit= new THREE.Group();

   drawHead();

   drawBody();

   rabbit.position.set(-4.5, -0.8, -0.2);

   rabbit.rotation.y= -0.2;

   rabbit.add(head);

   rabbit.add(bod);

   scene.add(rabbit);

}
```

Each element is implemented with Three.js's various geometries, for example the body and the face are 3D icosahedrons, while the cheeks are 2D circles.

```
  …
  faceGeometry= new THREE.IcosahedronGeometry(0.5, 0);
  face= new THREE.Mesh(faceGeometry, material);
  face.position.set(head.position.x, head.position.y, head.position.z);
…
```

4.2 OTHER SIMPLE ELEMENTS

Other elements featured in the project are the eggs and the wells.

4.3 LIGHTS

The project includes two lights, a point light and a basic dim ambient light.

The point light is used to simulate a sunlight and is the one modified when changing to night mode.

For the standard display mode (day) these are the lighting settings:

```
var light= new THREE.PointLight(0xffffff, 1, 100);
light.position.set(10,  20, 20);
light.power=14;
scene.add(light);


var lightAmbient = new THREE.AmbientLight( 0x606060 );
scene.add(lightAmbient);
```

4.4 MATERIALS AND TEXTURES

The materials used are both of MeshLambertMaterial type, which doesn't show specular highlights, for the rabbit's body and of MeshPhongMaterial type for other elements.

The textures used are 7 different patterns for the eggs, picked randomly and a white fur for the body of the rabbit (including the cheeks, where the texture is mixed with a pink color).

4.5 ANIMATIONS

Each animation has been implemented by hand over the hierarchical model using Tween.js.

4.5.1 RABBIT ANIMATIONS

The first animation showing up is in the home page, where the rabbit rotates its head back and forth. This was easily executed with tween.js's infinite chain.

```
…
tween_hi= new TWEEN.Tween(head.rotation)
            .to({x: -0.3}, 1100);
var tween_hi2= new TWEEN.Tween(head.rotation)
            .to({x: 0.3}, 1100);
tween_hi.chain(tween_hi2);
tween_hi2.chain(tween_hi);
tween_hi.start();
…
```

When starting the game, the rabbit first rotates its front legs up and moves along the x axis until it arrives in the middle of the scene.

This was implemented with a for loop and the setTimeout method (with increasing multiples of a timeout variables) calling the *single_run* function:

```
function single_run(){
  …
  rabbitx+=runxup;
  tween_run= new TWEEN.Tween(rabbit.position, groupRun).to({x: rabbitx}, upspeed);
  rabbitx+=runxdown;
  var tween_run2= new TWEEN.Tween(rabbit.position, groupRun).to({x: rabbitx}, downspeed);
  tween_run.chain(tween_run2);
  if(!stop_check)           tween_run.start();
}
```

Simultaneously, the rabbit does little jumps along the y axis, including the back legs groups b1 and b2 going up and down.

When the rabbit arrives in the middle of the scene, it stops moving along the x axis, but keeps “walking in place” according to the *run_fixed* function, performed from the start with an infinite tween chain.

The jump animation pauses the running, so that the movements are clearer. When jumping the rabbit moves faster and higher along the y axis once, also its ears, tail and back legs move.

The various elements of the jump are synchronized with tween.js’s chains and starts sequences.

When the rabbit doesn’t jump over a well, it falls. The fall animation is composed by a rotation on the x and y axis, followed by a fall along the y axis and a “scaling” effect using the z axis.

Once completed, the rabbit is removed from the scene, as the game is over.

```
function fall(){
   fall_check=true;
   ...
   var tween_fall= new TWEEN.Tween(rabbit.rotation)
                .to({x:-1, y:0.1}, rotatespeed);
   var tween_fall2= new TWEEN.Tween(rabbit.position)
                .to({y:rabbity-fally, z:-4}, downspeed)
                .onComplete(function dltrabbit(){
                   (scene.remove(rabbit));});
   tween_fall.chain(tween_fall2);
   tween_fall.start();
}
```

4.5.2 OTHER ELEMENTS ANIMATIONS

Both the eggs and the well animations are similar simple movements along the x axis.

The egg starts on the right end of the path, when it arrives in the middle of the scene (where the rabbit is) it’s either “grabbed” and deleted from the scene, or, if the rabbit jumps, the egg goes through with its pact until it’s out of view on the left end, then deleted.

Comparably, the well does the same, except that when the user makes the rabbit jump, it goes on with its pact, otherwise the rabbit falls and it’s game over.

5- User Interactions

Pressing the arrow up key [↑] the user can make the rabbit jump.

```
document.getElementById("body").onkeydown=function(event){
    var x= event.key;
    if(x=="ArrowUp"){
      if(!stop_check){
        jump_check=true;
        jump_check_egg=true;
        jump_check_well=true;

        pause();
        jump();

        setTimeout(function jmpchk(){
          jump_check_well=false;
          jump_check=false;
        }, 350);

        setTimeout(function jmpchk(){
          jump_check_egg=false;
        }, 250);

        setTimeout(restart, 280);
      }
    }
```

The implementation includes three Boolean variables: *jump_check*, for the animation update, *jump_check_egg* and *jump_check_well* for respectively checking if the rabbit got the egg or if he fell in the well.

The jump_check_egg timeout is set shorter to make catching the egg easier.

The user can also set the lights to be day or night from the Home page.

To switch to the night setting:

```
document.getElementById("night").onclick= function(event){
                light.power=6;
                night_check= true;
                scene.background= bgn;
 }
```