Interactive Graphics

Final Project

(26.07.2020)

Presented By:
Berrak Senses
Artsiom Sauchuk

1.Introduction

1.1. Idea

In this project we have simulated a dog ball fetching game in a house garden.

1.2. Libraries and technologies used

- **React:** *We have used this library to build user interfaces.*
- **create-react-app:** *It is a set of scripts which help to develop a front-end application using React. This library helps us to run the project without configuring.*
- **three.js:** *We have used three.js for creating animated 3D graphics. Three.js simplifies the WebGL API.*
- **tween.js:** *It is used for smooth animations.*
- **dat.gui:** *It is a lightweight graphical user interface which we used during the developing phace.*

We have also added some .obj files using three.js ObjLoader method to make our scene look more realistic.

2. Models

2.1. Light

We wanted our scene to simulate a real garden, so instead of putting a spotlight we have decided to use directional light(just like the sun) in order to give more realistic vision to our artificial garden. The color of the directional light is the same as the natural global light. We have also added another light which is an ambient light. This light is globally illuminating all objects in the scene equally. However, it doesn't have any direction.

2.2. Dog

For the dog, we have created a hierarchical model by using the three.js library and for each mesh of the model we have used the “phong material” to make them simulate the reflection of the light in the scene. The dog has a different type of geometry for the meshes. The hierarchy can be seen below;

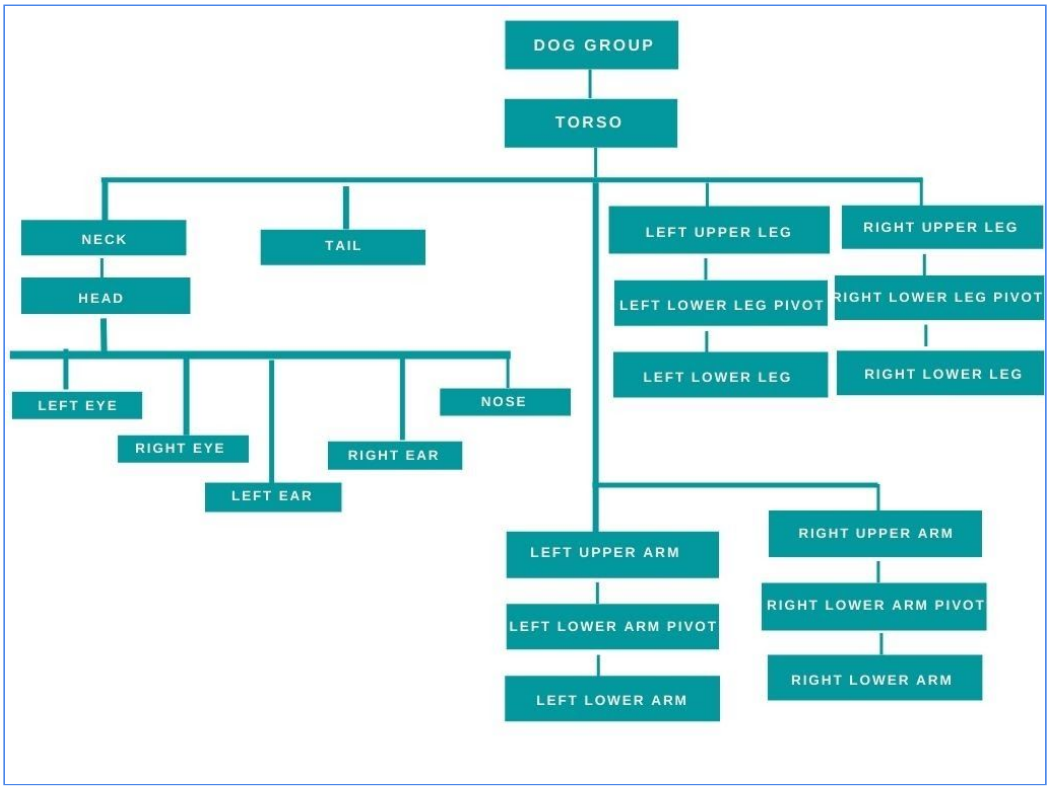

Figure 1: Dog hierarchy

2.3 Human

We have created the human hierarchical model by using the three.js library and we also used phong material for each of the elements of the model. The human has three different textures. The textures are: skin, shirt and the face. The skin texture is applied to each element of the human model except the “torso” and the “head”. The torso has the shirt texture, whereas the head mesh has the texture of a human face. The human hierarchy can be seen in below;

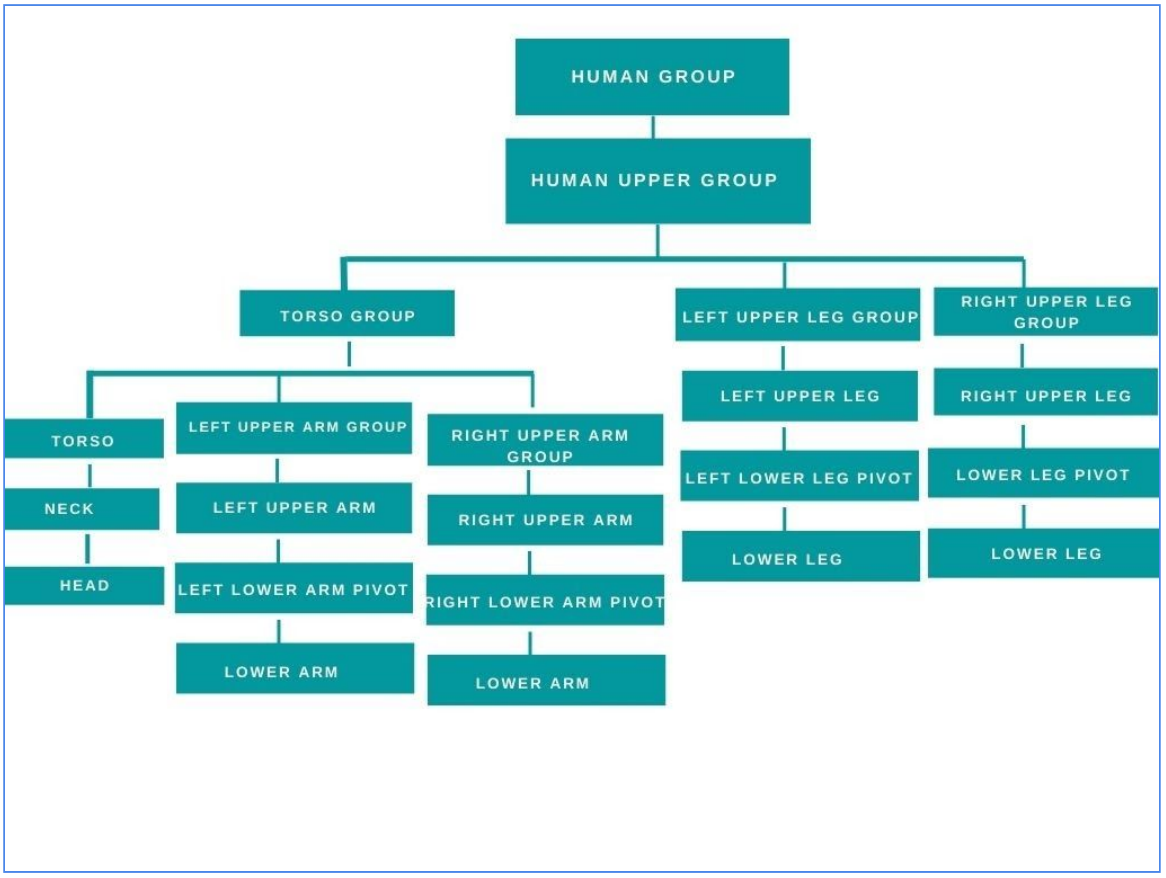

Figure 2: Human hierarchy

2.4 Decoration

To make the scene as realistic as possible, we have decided to use some models. Since we were trying to simulate a garden, we used four models in our scene: grass, fence, tree and a house.The ground is a model of grass and the object has the texture of a grass image. The house model is a wood house so we thought that it would be more accurate to use a wooden texture. The tree has two different textures, one for the leaves and one for the bark of the tree. The last model that we have used in our project was the fence object. We wanted to use the fence object to show the boundaries of our scene to the user. All of the models that are loaded have the phong material.

Since loading the objects is a quite slow process, we have decided to use a “loading” animation while the models are loading, whenever the models are loaded, the scene is presented to the user.

3. İmplementation

3.1. Animation

After we have created the hierarchy for the dog and the human, we tried to make some animations. However, when we were dealing with the animation of the arms and the legs we noticed that the animations were quite different than we were expected. After some experiments and some research we found that in the three.js library the rotations were made from the middle of the mesh.

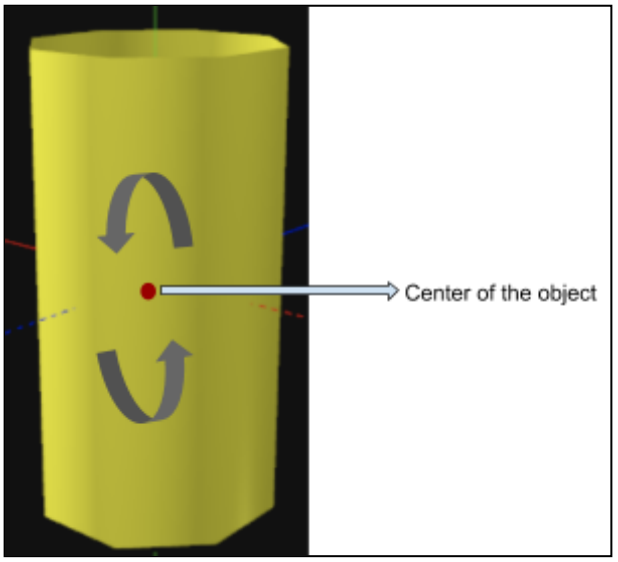

Figure 3: Middle of the mesh.

For solving this problem we have added an instance of Object3D class called *“pivot”* to make the animations look more realistic. With the help of the pivot object, the rotation point of the mesh can be relocated. We wanted to move the rotation point of the mesh. The change of the rotation point can be seen below;

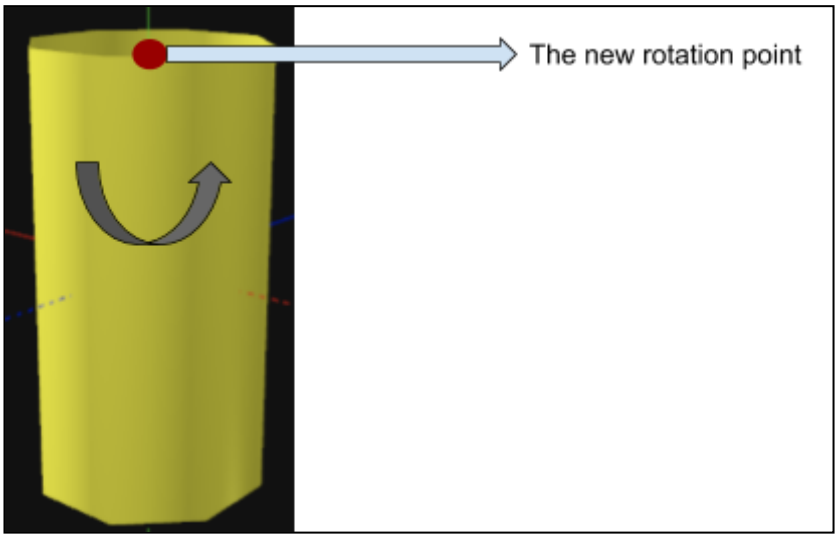

Figure 4: Change of the rotation point.

When we implemented all the models we started to work on animations. For implementing this, we decided to use Tween.js library which provides an API for smooth transition between different keyframes.

Each model has a state represented by the rotation angles of legs, arms and head as well as position of the body in world space. The ball model's state is just represented by the position in space. These states are called "keyframes". When we define an animation, we define a starting point (initial keyframe) and the target (final keyframe, the end of the animation). Then we use tween.js which provides us an intermediate keyframe between initial and final keyframes at each time interval. Tween.js also provides different forms of the interpolation. For the dog movement we used "Quartic.InOut" interpolation (see Figure3).

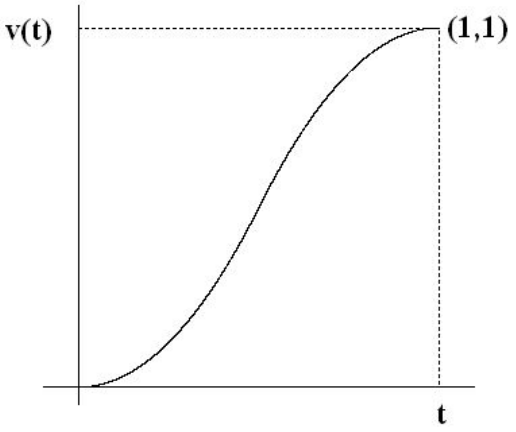

Figure 5: "Quartic.InOut" interpolation from tween.js

3.2. The Ball

The ball is a spherical object which the humanoid can throw. The ball flight path is calculated every time by using formula in figure 4 given throw angle, initial speed, and initial height at which ball is thrown. When trajectory is computed we divide it into ten keyframes. Those keyframes are represented by pairs of coordinates (x; y).

$$y = xtg\alpha - \frac{g}{2v_0^2 \cos^2 \alpha} x^2 \quad .$$

Figure 4. Formula for calculating the ball flight path, where α - throwing angle, v0 - initial speed, g - constant gravitational acceleration.

The trajectory is calculated in the 2D space. In order to translate the 2D trajectory to 3D we use a starting point of throw *(X_0, Y_0, Z_0)* and a direction unit vector *d*. The transformation happens according to formula in the figure 5.

$$\begin{aligned} X_i &= X_0 + d_x * x_i \\ Y_i &= y_i \\ Z_i &= Z_0 + d_z * x_i \end{aligned}$$

Figure 6. Transforming 2D flight path to 3D
(X_0, Y_0, Z_0) - initial 3D point of throw.
(Xi, Yi, Zi) - new 3D coordinates.
(xi, yi) - 2D coordinates of the trajectory
d - direction unit vector

After that transformation we have ten keyframes that represent a point in 3d space. Then we use tween.js for the smooth transition between those keyframes. For a smooth trajectory we could divide it not by ten but into twenty keyframes. But as we saw, there is no visual difference.

When transition upon trajectory is finished and the ball is on the ground, the dog starts moving in towards the final position of the ball to take it.

4. User interaction

The animations are started by pressing one of the three keys of "enter", "space" and "shift". The human can rotate using the left and right arrows on the keyboard. The mouse can be used for rotating the camera, as well as zoom-in and zoom-out using the mouse's wheel.

5. Further development

Since the trajectory of the ball is computed automatically based on the initial speed and throwing angle, it is possible to consider an ability to change that parameters by the user. For example, the user could fix a particular initial speed and an angle to hit the ball into some place. It also can be considered as a game where we score points which describe how precisely a user hit the target on the ground.

Used materials

1. The house model:
   https://www.turbosquid.com/3d-models/thai-culture-house-3ds-free/1060013
2. The tree model:
   https://www.turbosquid.com/3d-models/tree-tow-set-3d-model-1447788
3. The fence model:
   https://www.turbosquid.com/3d-models/fence-white-x-free/842342
4. The grass model:
   https://free3d.com/3d-model/-rectangular-grass-patch--205749.html
5. CSS styles for the loader:
   https://co depen.io/I-is-kevin/details/jqejXj
6. CSS styles for the keyboard buttons from the homepage:
   https://codepen.io/thebabydino/pen/zYrOQvr