FINAL PROJECT

Bonifazi Emanuele
Mat. 1815292

Project idea

The project consists in a flight simulator in which the player can drive a helicopter by using keyboard and available commands on the screen (engine and light controls). The goal is to take off from the starting base and to land on the final red base.

The usar will be able to control the altitude and the direction of the helicopter and by using the mouse it can control the camera in order to find the red base.

Frameworks

The two frameworks used to develop the project are Three.js and Tween.js that combined allow to build a complete project containing models, ambient details, and animations.

Tween.js

This framework was used to develop model and ambient details. It was implemented by the using of the three.js library.

This library contains functions that allow the developer to easy build some basic geometry like boxes, cone sphere etc… Through these basic structures it is possible to build more complex models in a hierarchical way.

It is also possible to use Tween.js to implement ambient details like lights sources. It makes available different kinds of lights that allow the developer to make more realistic the scene.

Another important feature of this library is that it provides some postprocessing functions like anti-aliasing that is very useful for making edges smoother.

Tween.js

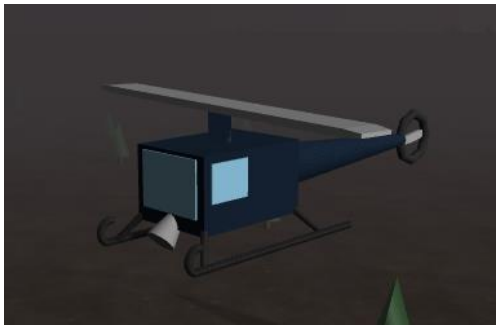
This framework was used to develop the animations. In particular, it provides a set of functions that help to make the animation smoother. By this way we will obtain a more realistic feeling, without the presence of jerky movements.

Also this framework was implemented by a library: tween.js.

Models

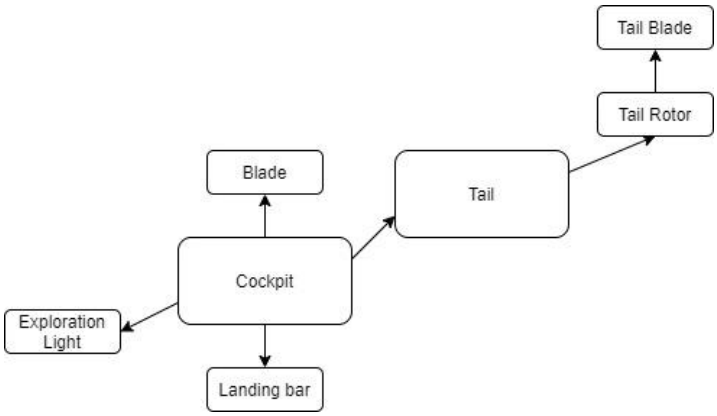
It is possible to identify two different main models that compose the scene: helicopter models and natural ambient model. Both of them were built in a hierarchical way and are not imported.

Helicopter



This is the most important model, and it is composed by different parts, each of which is a basic structure of three.js library.

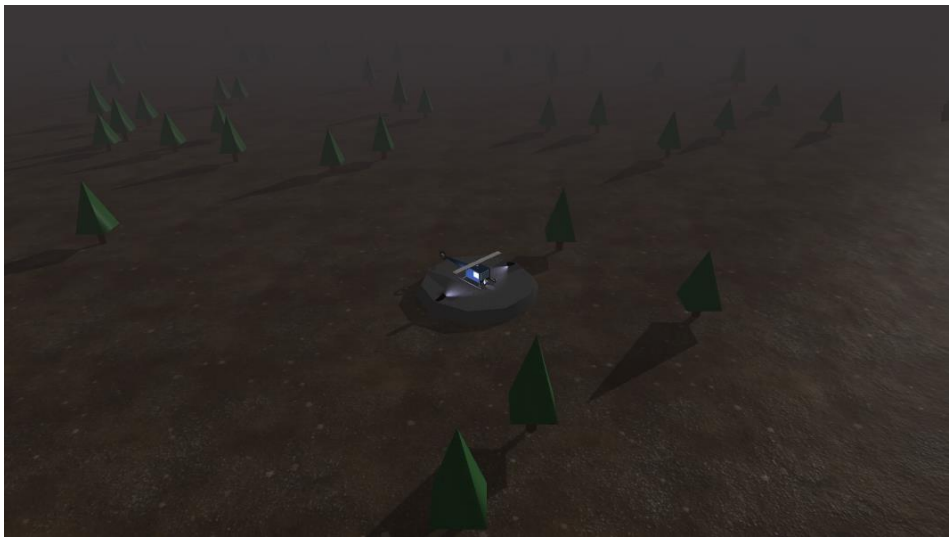
The following scheme represent the structure of the model:



It is possible to see that is a hierarchical model where the main node is the cockpit.

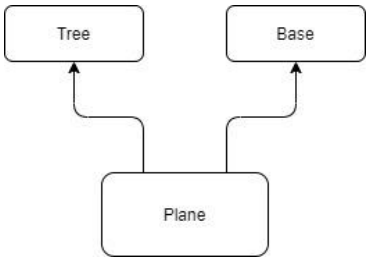
N.B. Some secondary elements are not in the scheme to preserve a clear view of the main structure.

Natural ambient



It consists in the world structure, composed by terrain, trees and the two bases.

The following scheme represent the structure of the model:



Please note that in the scheme are represented only the entities that compose the hierarchical model, not the real number of the instances of these entities. In particular we have two bases and there are many trees that are placed in a random way along all the plane.

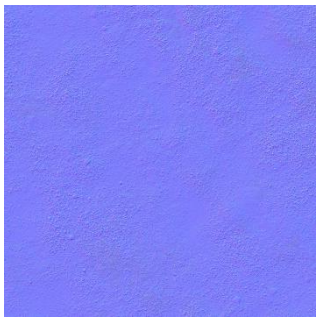
Trees

Also trees are very simple hierarchical model composed by trunk and crown. In this case it is important to highlight that to make the scene a little bit more realistic the size of trees is randomic in a given interval.

Textures

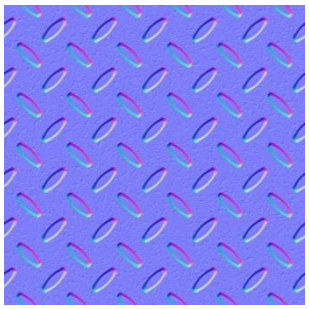
Textures are used in the project to obtain “realistic” surfaces. Three.js allows to use different kind of textures. In this project are used: colour, image and normal textures. The most important are the last two, these are in fact used together to produce a fake 3D surface feeling by working over light reflection.

An example that is possible to see at the beginning of the game is the following:



In this case the first image is the image texture, and the second image is the normal texture. Combining these two images it is possible to obtain a realistic surface on the plane.

It is also possible to use the normal texture without an image like in the case of the base:



It is possible to observe by playing the game that the surface of the base is light responsive and give the feeling of a 3D metal surface.

Animations

Animations are implemented by the using of tween.js library. This library provides functions that help to create smooth animation.

The following is an example of the blade animation:

```
var target = { y: 16 * Math.PI - 0.00001 };
var mainBladeRotation = new TWEEN.Tween(blade.rotation).to(target, 8000);
```

This animation starts when the button “Start Engine” is pressed and consist in the initial acceleration of the main blade. Blade starting from the initial position will rotate along the axes 8 times $(16 * \pi)$ with increasing speed.

To make it more realistic the acceleration is not linear but Quintic:

```
mainBladeRotation.easing(TWEEN.Easing.Quintic.In);
```

After the acceleration animation the static rotation start and in order to make this the function chain() allow the developer to attach two animation sequentially. Then the static animation is repeated infinitely times:

```
mainBladeRotation.chain(mainBladeRotation2);
```

```
mainBladeRotation2.repeat(Infinity);
```

The helicopter movement animations starts when controls button is fired (controls paragraph) and consist in change the position and the orientation of the cockpit. The other parts that compose the helicopter will follow the cockpit due to hierarchical structure.

Camera animations

To obtain a 3rd person view that follow the helicopter when this last one changes position with an animation also the camera changes position in the same direction. It is important to highlight that when the helicopter rotates along the axes the movement direction is no more along the world axes but along the cockpit relative axes. To allow these movement we will use the relative position coordinates:

```
var position = cockpit.localToWorld(new THREE.Vector3(1,0,0));
```

This function is used to specify that we want a position that is one unit forward along the cockpit x axes and by localToWorld() function we convert from relative coordinates to world coordinates. By this way we can move the helicopter forward independently from the orientation.

Controls (interactions):

The project provides three different kinds of controls: keyboard commands, screen buttons command and camera controls.

Keyboard commands

These are used to control altitude, direction, and rotation of the plane. Each time a key is pressed the relative movement animation is fired and the helicopter will move id desired way.

The following is an example that show how to raise the helicopter to altitude:

```
case 'KeyW':
    var auxAn = new TWEEN.Tween(cockpit.position).to({y: cockpit.position.y+0.5}, 50).start();
    var cameraAnimation = new TWEEN.Tween(camera.position).to({y: camera.position.y+0.5}, 50).start();
    var lightAnimation = new TWEEN.Tween(spotLight.target.position).to({y: spotLight.target.position.y+0.5}, 50).start();
break;
```

It is possible to see that commands are managed by a switch case that check the key passed by onKeyDown event.

Screen buttons commands

These commands are placed under the scene and by using them it is possible to start the starting engine procedure and to manage the exploration light placed in front of the helicopter.

By pressing the start engine button, it is possible to start the animation seen in the animation paragraph.

Light controls are used to change the colour and the intensity of the exploration light, it is also possible to turn off/on it:

```
document.getElementById("buttonLightOnOff").onclick = function () {
    if(flagLight){
        spotLight.intensity = document.getElementById("intensitySlider").disabled = true;
        spotLight.intensity = 0;
        flagLight = !flagLight;
    }
    else{
        spotLight.intensity = document.getElementById("intensitySlider").disabled = false;
        spotLight.intensity = document.getElementById("intensitySlider").value;
        flagLight = !flagLight;
    }
};
document.getElementById("buttonLightRed").onclick = function () {
    spotLight.color.setHex( 0xff0000 )
};
document.getElementById("buttonLightBlue").onclick = function () {
    spotLight.color.setHex( 0x0000ff )
};
document.getElementById("buttonLightWhite").onclick = function () {
    spotLight.color.setHex( 0xffffff )
};
document.getElementById("intensitySlider").oninput = function(){
    spotLight.intensity = this.value;
}
```

Camera controls

Camera controls are managed by OrbitControls library provided by three.js. These controls allow the user to use the mouse to move the camera around an object, in this case the helicopter. The control is based on the classical click and drag:

```
const controls = new THREE.OrbitControls(camera, renderer.domElement);
controls.target = cockpit.position;
controls.maxPolarAngle = Math.PI/2;
controls.maxDistance = 100;
controls.minDistance = 3;

controls.enableDamping = true;
controls.dampingFactor = 0.05;
```

Damping is used to make camera movements smooth during the transition (drag).