Sapienza University of Rome

Interactive Graphics

Project Report

LegoMan

Shend Osmanaj

ID: 1783335

osmanaj.1783335@studenti.uniroma1.it

July 2018

Table of Contents

Abstract

For Interactive Graphics project I decided to do a human-like figure Lego style(or Minecraft) which is placed on an environment.

The project contains a figure which was realized in hierarchical mode, and textures for the ground and the sky. Also, there are implemented different kinds of lights like AmbientLight and SpotLight.

The environment can be managed with the mouse, which allows us to rotate around the environment.

In the figure are performed some animations which can be seen by clicking the corresponding buttons. They allow the figure to walk, run, wave, shake head.

Part 1

Tools and libraries

1.1 ThreeJS

ThreeJs is a browser JavaScript library which is used to create and display animated 3D computer graphics in a web browser. Three.js uses WebGL.

ThreeJS has a lot of cool features which make our work easier, like:

Effects, scenes, cameras, animation, lights, materials, shaders, geometry and much more. All we have to do to use it, is to download it and include it as a source file in our code.

1.2 Visual Studio Code

By doing this project I used VS Code, which is a code editor developed by Microsoft.

It includes support for programming in JavaScript and ThreeJS, among other programming languages. It is lightweight, customizable, free and open-source.

Part 2

Technical aspect of the Project

2.1 Scene

In threeJS we need three things to display anything: scene, camera and renderer, so that we can render the scene with the camera. In the project is used a perspective camera which has a field of view 55 , an aspect ratio of window width by the window height, and the near and far elements which tell that elements closer than near and elements farther than far will not be displayed. The camera z position is changed to 4 , so the figure can be seen in an appropriate size. I included OrbitalControls.js which is included in the official ThreeJS folder. It allows us to navigate in the environment by moving our mouse and by zooming in and out.

There is sky dome defined in the project, which uses an image of sky as a texture and sphereGeometry as a geometry. Inside of it are found the other elements, including ground which is a PlaneGeometry and uses a texture of grass.

2.2 Hierarchical model

The figure is created in hierarchical model. Where the body is the parent, and head, arms and legs are its children. By using hierarchical models, it is easier to do translations and rotation. When we apply transformation on the parent, then it automatically affects its children. In this case, when I want to move the figure, I apply the translation on body(the parent) instead of all the parts one by one.

All of the body parts are created using BoxGeometry and use a checkerboard texture as a material.

2.3 Lights and textures

There are two different lights applied in the project: AmbientLight and SpotLight. AmbientLight is already applied and SpotLight can be activated by clicking the SpotLight button.

AmbientLight has a color of 0x00ffff and is set in the position (0,1,0), meaning on top of the figure. SpotLight is of color 0xffffff with an intensity of 0.5 and position of (0,1,0). After defined, they have to be added in the scene in order to see their effects.

There are different kind of textures which are applied to the body, the ground and the sky. Body uses an image called checkerboard as a texture which is used as a parameter to the material that is used for the body.

For the head of the figure is used an array called materialArray which takes six images that will be applied as texture on each face of the head (which is a cube).

Ground uses an image of grass as a texture which is set to be repeated by:

```
grassTexture.wrapS = grassTexture.wrapT = THREE.RepeatWrapping;

grassTexture.offset.set(0,0);

grassTexture.repeat.set(2,2);
```

A grassMaterial which is a MeshPhongMaterial takes this texture as a parameter.

Sky uses an image of sky as a texture. SkyMaterial which is a MeshPhongMaterial takes the texture as parameter. By setting:

```
skyMaterial.side = THREE.BackSide
```

we decide which side of faces will be rendered, in this case the backside.

2.4 User Interaction

In the project is included OrbitControls.js which allow the user to control the environment by mouse clicking and by zooming in and out. Also, the user can start and stop different animations by clicking the corresponding buttons.

2.5 Animations

There are some animations performed in the human-like figure.

These animations can be performed by clicking the buttons Yes, No, Walk, Run, Walk Right.

The yes button executes a function called sayYes(). This function rotates the component head among its x axis, like this:

```
head.rotation.x += direction * 0.01;
```

It will continue to rotate on one direction, until it reaches a certain value, and then it will change its direction of rotation. This is done by using if statements, where as long as the rotation has not reached a certain value then rotate positively, then after the rotation has reached a value then rotate in the opposite direction. This is needed to simulate the nodding of the head, so the cube won't rotate always on one direction.

The no button executes a function called sayNo(). This function rotates the head among its y axis, like this:

```
head.rotation.y += direction * 0.01;
```

Here too, are used some boundary values which make it look realistic while the figure is saying no. There are two certain values, when reached, will change the direction of the rotation of head among y.

The wave button makes the figure to wave his left hand by executing a function called waveHand(). At first the position of the hand is changed by using:

```
leftArm.position.set(sizes.bodyW/2 + sizes.armW/2,sizes.bodyH/100 + 0.25,0);
```

which will change the position of left Arm by its y axis. Then the hand is rotated among its z axis:

```
leftArm.rotation.z += direction * 0.01;
```

We have two if statements that make sure that after the hand reaches a certain rotation value, it should change its direction.

The walk button executes the function moveFigure(). This function will make the figure move, simulating its walk. We apply translation on the body on y and z to make it move like this:

```
body.translateY(-0.003);
body.translateZ(0.003);
```

which means that every time the screen is refreshed, move body along its y and z axis.

Since body is the parent of all the other body elements, translation on it will affect all the other elements, so we do not have to apply translation on other elements.

While the body is moving, the arms and the legs have to move too, this is achieved by rotating them among their x axis, like this:

```
leftArm.rotation.x += direction *  0.01;
rightArm.rotation.x -=  direction * 0.01;
leftLeg.rotation.x += direction * 0.01;
rightLeg.rotation.x -= direction * 0.01;
```

As we can see, the leftArm goes in opposite direction of rightArm, and leftLeg goes in opposite direction of rightLeg, to simulate walking.

The run button will execute a function called moveFaster() which is similar to function moveFigure() but here to simulated running, the rotation and the translation are done in higher values.

The button Walk Right will execute a function called walkRight(). This function will make the figure move along x axis until a certain position is reached.

The body is translated among x and z, like this:

```
body.translateX(0.01);
body.translateZ(0.01);
```

And the arms and legs are rotated among their x axis, same as in the previous functions explained above.

All of these methods are called on a function called render() which starts and stops the animations.

In case of walking, it checks if the figure has arrived a certain point, in this case -1.5, if not keep calling moveFigure(), otherwise stop the figure.

In case of walking among x, it will check if body.rotation. y < 0.7, if yes, rotate the body by 0.5. This will make the figure rotate in a certain position and then walk among x.

Also it checks if body.position.x has reached a certain value, in this case 5. As long as it hasn't keep walking, otherwise stop.