Binh Nguyen Thanh

3D Visualization

3D Model Visualization in Samsung 3D TV Using 3D Glasses

Information Technology
2016

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Nguyen Thanh Binh
Title	3D Visualization
Year	2016
Language	English
Pages	55
Name of Supervisor	Smail Menani

A native 3D visualization application to visualize Prima's 3D models on 3D TV using 3D glasses with a basic mouse controlling system and menu controlling system is developed using HTML5, Javascript and PHP. The application's result fulfilled its purpose to be viewed on a Samsung 3D TV and controlled with provided controlling systems. The application could be improved by providing advance control functions, interaction with a user. The application could be improved with data connectivity fetched from databases if required.

Keywords 3D Visualization, WebGL, HTML, Javascript, ThreeJS

CONTENTS

LIST OF FIGURES AND TABLES

LIST OF ABBREVIATIONS

VR	Virtual Reality
AR	Augmented Reality
HTML	Hypertext Markup Language
API	Application Programming Interface
IDE	Integrated Development Environment
UI	User Interface
PHP	Hypertext Preprocessor
JSON	Javascript Object Notation

LIST OF APPENDICES

1 INTRODUCTION

3D technology and applications using 3D technology are continuously increasing. In the world of 3D, Virtual Reality (VR), takes the biggest shared. Prima Oy has the motivation to apply 3D technology to its product visualization but in a slightly different way. Native 3D visualization on Samsung 3D TV with 3D glasses to visualize Prima 3D models is the goal to be achieved and the result is to make the way of viewing the products similar to watching 3D movies together with user interaction. To fulfill the goal of the customer, 3D visualization a web-based application is developed using HTML5 and Javascript with WebGL. Controlling systems using a mouse and an external menu are provided in the application to control the rotation, zooming and moving, as well as changing the properties of the model. Administrator tools are provided to upload new models to the webserver. A special implementation is made for a special case given by the customer with a very detailed and a big model.

1.1 Requirements

There are specific requirements that need to be achieved. They are:

- The application must be visualizable on a Samsung 3D TV with 3D glasses.
- Models has automatic rotation animations.
- Mouse control functions.
- File selection or object selection menu.
- 3D objects received from Prima must be possible to be display.

The following advantage features would be nice to be implemented further:

- The application has controlling menu to control object size, scale and rotation speed.
- The application is able to open any 3D object extension without converting.
- The application is able to get object information and display it.

1.2 Specifications

- An admin function is used to upload models to a web server.
- Mouse control functions are used to provide a basic controlling function with mouse buttons and movement.
- Menu control functions are used to provide changing the properties of the model.
- A model selection function is used to select model to be shown on the output display.

2 ANALYSIS

The requirements are investigated, several technologies are tested and the analysis results are shown to be selected for the implementation.

2.1 Requirement Analysis

The technology should provide side-by-side rendering in order to make the model rendered viewable on the hardware in native 3D.

The technology should have the possibility to add a control method or provide a built-in controlling system using a mouse in order to control the model with a mouse.

The technology should be capable of providing menu control to control the model's variables and properties.

The technology should provide loading the drawing javascript files for detailed and big models.

2.2 Technology Analysis

2.2.1 Unity

Unity is a game development platform. It can be used to build 2D and 3D games, applications and deployed on multiple platforms such as desktop applications, Androids, iOS and Web application /1/. Unity supports VR and AR technologies. Numbers of demo applications are made and the results shows that Unity cannot fully fulfill the requirements. Demo applications failed on rendering side-by-side 3D for the hardware to display in native 3D.

2.2.2 Blender

Blender is an open source 3D modelling, animation, simulation and rendering software /2/. One demo application is made by Blender and the result shows that Blender can fulfill side-by-side 3D rendering. However, Blender is not able to deploy the result to any installable program or any platforms. The result can only be exported to 3D object files or can be viewed with Blender.

2.2.3 ThreeJS

ThreeJS is a javascript library that allows creating a 3D scene on basic HTML page using WebGL technology.

WebGL is a javascript API for rendering interactive 3D computer graphics and 2D graphics within any compatible web browser without the use of plug-in /4/.

ThreeJS fulfills the requirements and is chosen to be implemented.

2.3 Analysis result and Technology used

The analysis results show that ThreeJS fulfilled all requirements. It is a javascript library and can be viewed in any web browser that supports WebGL. No installation software needs to be exported. Together with ThreeJS, dat.GUI is used to implement controlling menu. dat.GUI is a lightweight graphical UI for changing variables in javascript /5/.The programming IDE used is Bracket. However, any text editing software could be used. Some IDE that could be considered are Notepad++ and Sublime Text.

3 DESIGN

The application contains four main functions which are shown in Table 1:

Function name	Function Description
Upload new model	Use administrator page to upload new model to web server.
Select model from model list	Select the model on the model list on dat.GUI menu to view.
Control model with mouse	Use mouse to control the model. Mouse function include zooming, rotating, moving object.
Control model with controller menu	Use dat.GUI menu to control the model. Function include changing scale of model and change self-rotation speed of model.

Table 1: List of functions

3.1 Use Case

The use case diagram of the application is shown in Figure 1. The use case diagram describes the list of functions and interaction of the system with user.

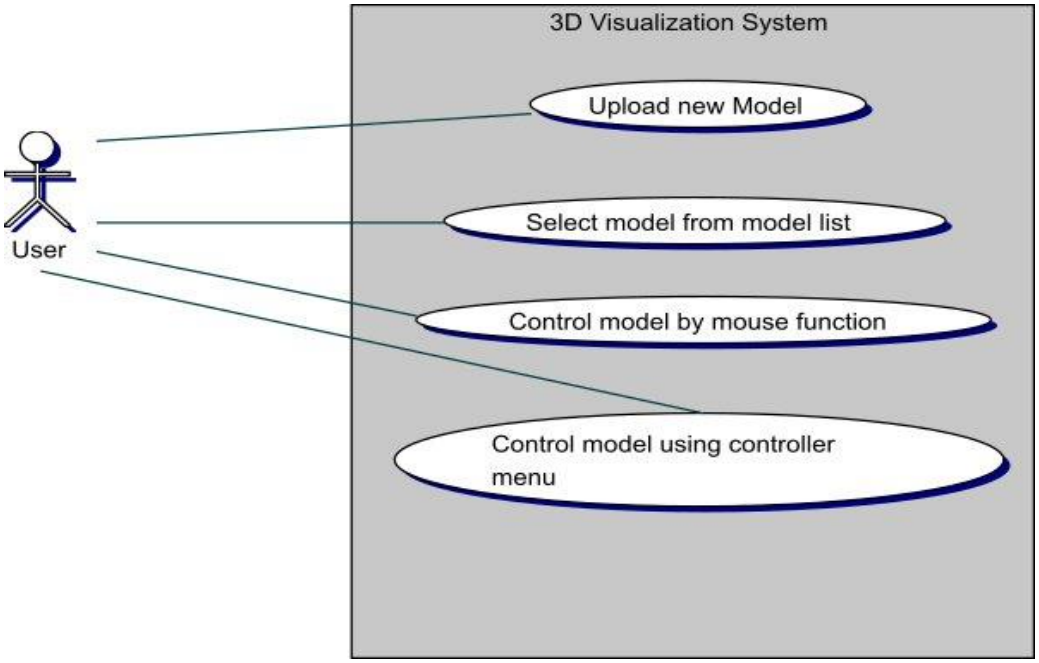

Figure 1. Use Case diagram

3.2 Function design

3.2.1 Uploading function

PHP script is used to develop the uploading function. A HTML page is used to provide a component to open the file selection dialog.A PHP function is used to get the directory on a webserver to be uploaded to, to get the file that selected from the HTML file dialog, upload the file with correspond name to selected directory or to use built-in upload function in PHP to upload the file to the directory.

3.2.2 Select model function

PHP script is used to get all the file names exist on the model directory with the constrain that the extension should be JSON. A Javascript function is used to get the return of PHP script in JSON format, read the response and paste it to an array variable. A dropdown list component on dat.GUI menu set the values which get from the array variable. When a file is selected, the file name is passed to the loading object function of ThreeJS and the loading function is used to load model to the scene.

3.2.3 Load model function

Load model function is the built-in ObjectLoader class of ThreeJS. An instance of ObjectLoader is created on the initiation state with the name is selected from the dropdown list. It could also provide the path to the model directory. ObjectLoader has a load function to load the model and add the model to the scene.

3.2.4 Mouse control function

The mouse control function is a built-in TrackballControls class of ThreeJS. Two instances of TrackballControls are created to control the camera and the light source separately so that the light is moving corresponding to the movement of the camera. Two controls will be updated on render function which call the requestAnimation-Frame() function to re-render the scene every time it changed.

3.2.5 Menu control function

A menu control function is developed to control some properties of the model such as self-rotation speed and the scale of the models. dat.GUI menu is used to provide the UI of the menu. When a change is made, the onChange function is called to set the variable of the models. It will then call the render function to render the scene and apply the change.

3.2.6 Special implementation function

The purpose the special implementation is to import the drawing files of the model into the scene. A PHP script is used to get all the drawing file names from the directory. A Javascript function is used to call the PHP script, get the response in JSON and pass it to an array variable. A javascript function is used to dynamically import drawing javascript files into the HTML page in order to call each drawing function inside. It also executes the function inside the external javascript files using eval(). The mouse control function remains the same but the menu control is limited to model selection only. The new menu control function is giving the name of the directory that contains drawing files of the model. The scaling and self-rotating is disabled on the special implementation.

3.3 Sequence diagram

The sequence diagram is used to describe the steps of using the function and also describes the flow of the function. The sequence diagrams for the main functions are shown in Figure 2, 3, 4 and 5.

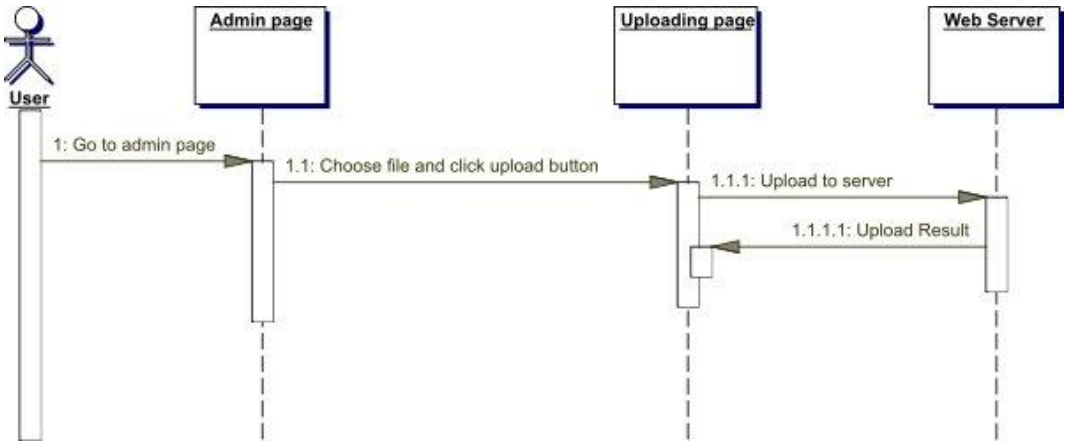

Figure 2. Uploading Sequence diagram

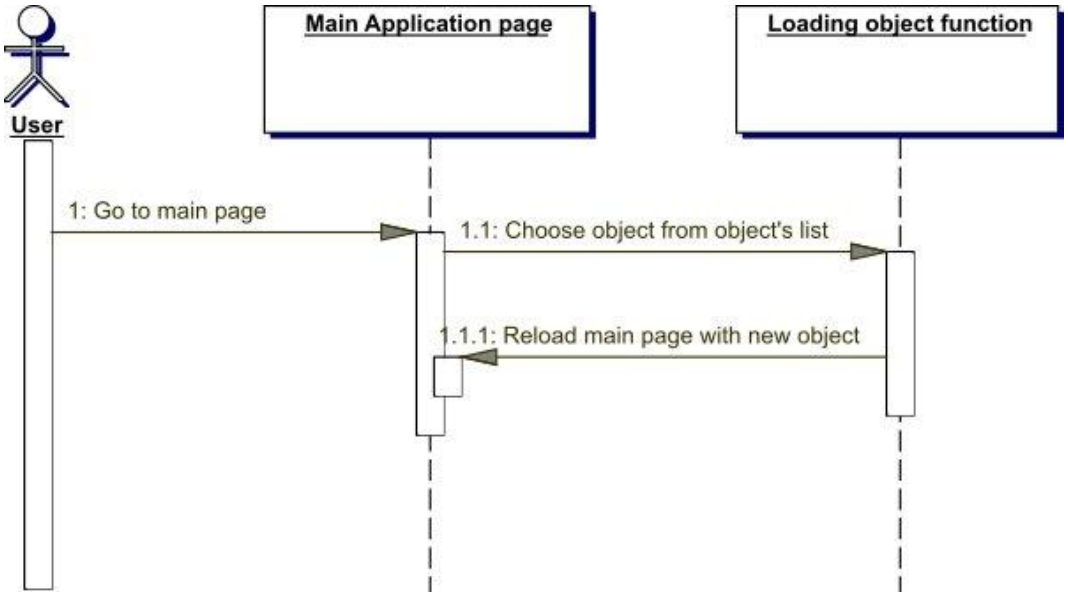

Figure 3. Model Selection Sequence diagram

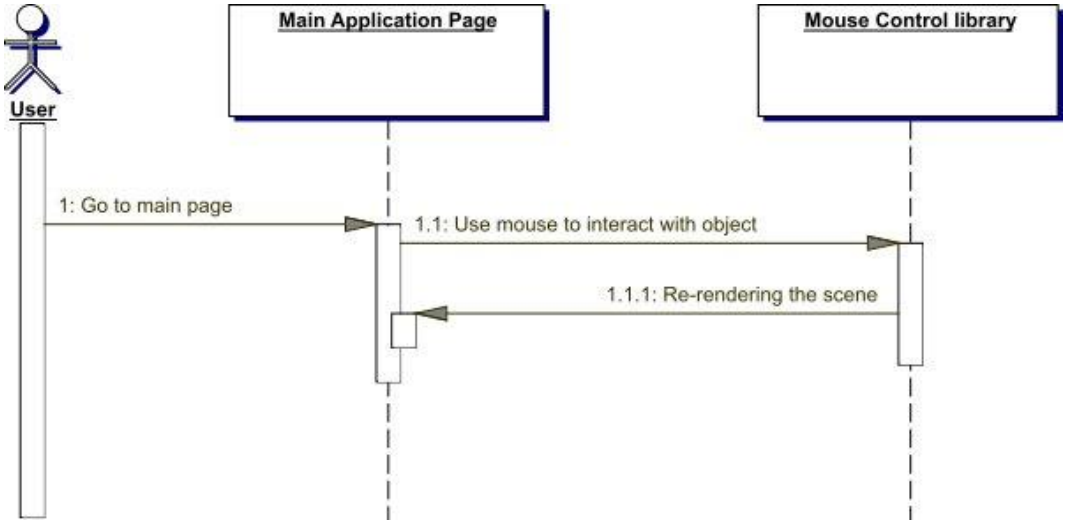

Figure 4. Mouse Control Sequence diagram

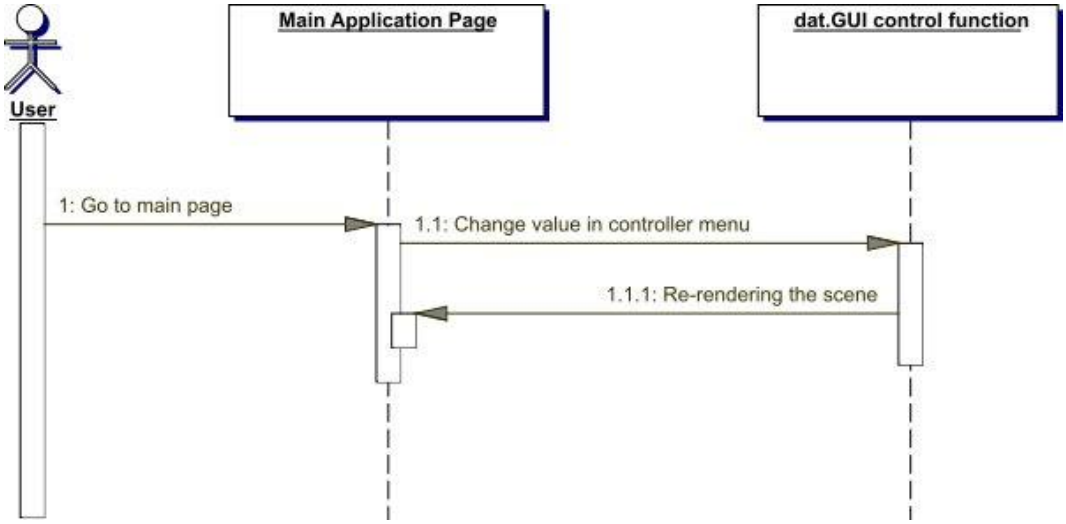

Figure 5. Menu Control Sequence diagram

3.4 UI Design

The UI designs are shown in Figure 6, 7 and 8.

The design for the main page of the application is shown in Figure 6. All pages are display the scene. The controller menu is on the top right corner and can be toggled show/hide.

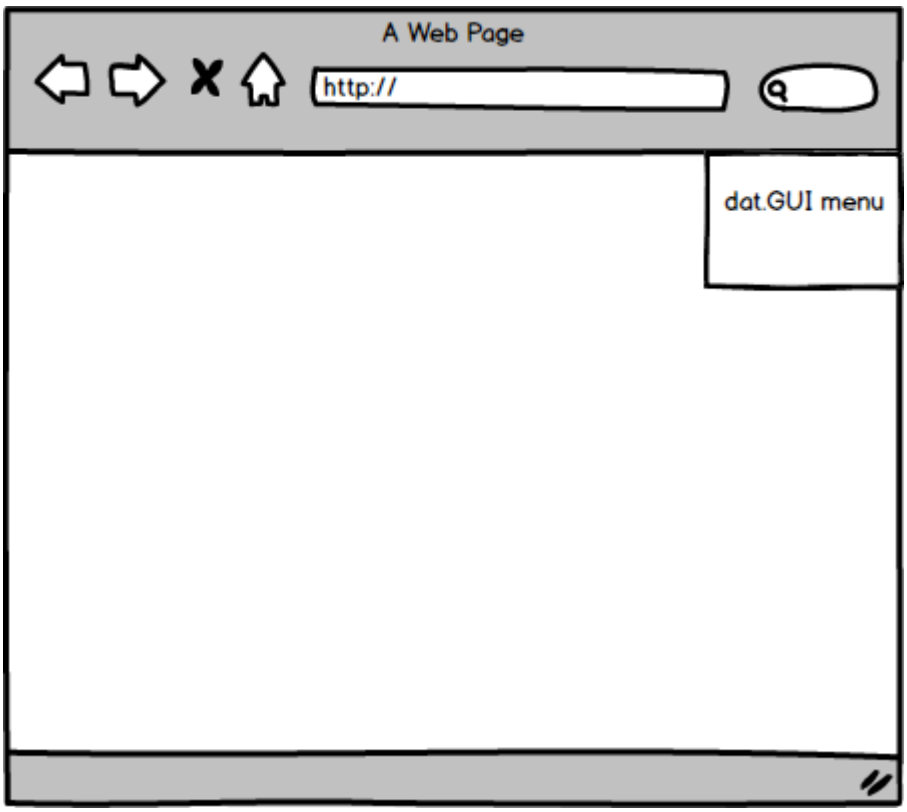

Figure 6. Main page design

The design for the admin page is simple. It contains a ”Choose file” button and a ”Upload” button as shown in Figure 7.

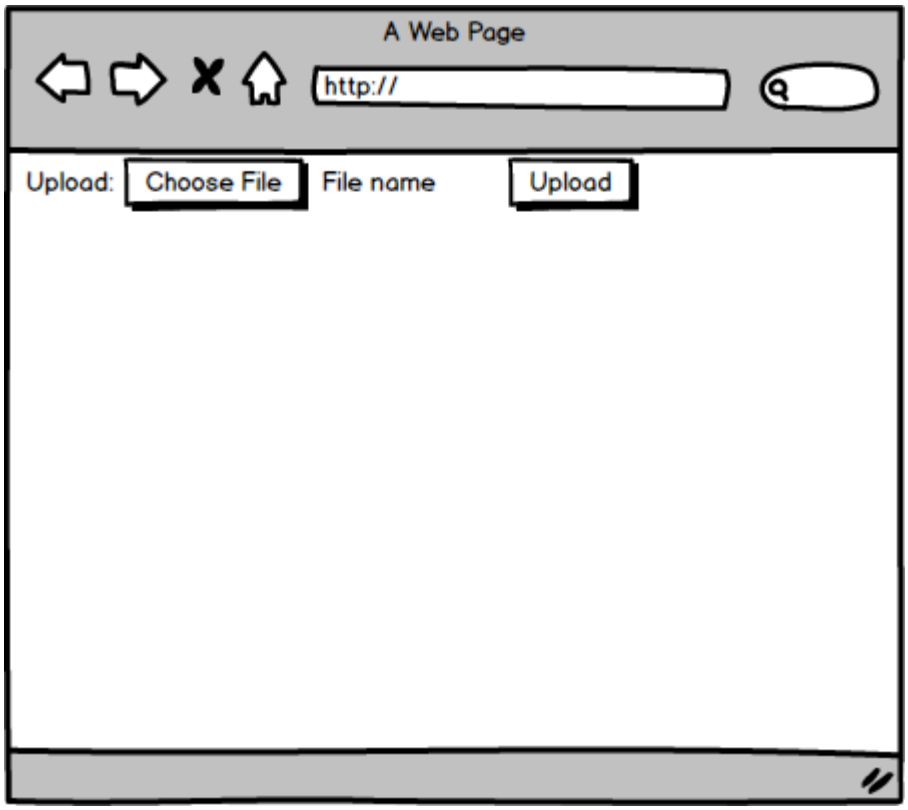

Figure 7. Admin page design

The design for the controller menu is shown in Figure 8.

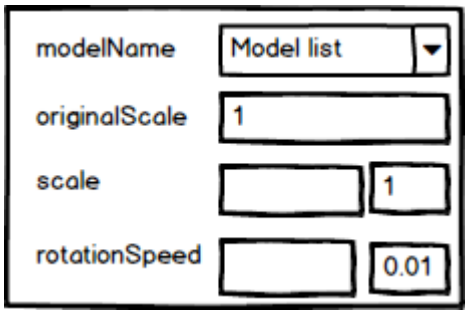

Figure 8. Control Menu design

4 IMPLEMENTATION

4.1 Implementation of admin page

The admin page is a simple HTML page which contains two buttons. The code of the main HTML page is shown in Figure 9.

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload" name="submit">
</form>

</body>
</html>
```

Figure 9. admin.html

The admin page contains a form that called ”upload.php” file to upload the selected object onto a web server.

4.2 Implementation of uploading function

The uploading function is called by the admin page. The purpose of it is to upload a selected object from the admin page to the web server. The function is writen in PHP as shown in Figure 10.

```php
<?php
//Target directory is set to current directory of the upload.php file.
$target_dir = "./";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$fileType = pathinfo($target_file,PATHINFO_EXTENSION);

// Check if file already exists
if (file_exists($target_file)) {
    echo "File already exists.";
    $uploadOk = 0;
}
// Allow certain file formats
if($fileType != "json" ) {
    echo "Sorry, only JSON Objects are allowed.";
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        //print success message
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
        //redirect to admin page after uploading in 5 seconds.
        header( "refresh:5;url=http://www.cc.puv.fi/~e1200649/3js/admin.html" );
        exit();
    } else {
        echo "There was an error uploading your file.";
    }
}
?>
```

Figure 10. upload.php

The upload function first checks if the file name already exists in the directory. If the file exists, the function will not upload the file and return message. If the file does not exist, the function will check if the file format is in JSON. If the file is not in JSON format, the function will not upload and return message. If the file passes all the checking criteria, the function will upload the file and return a success message. The function will redirect to admin page after 5 second when uploading succeeds.

4.3 Implementation of the main application

4.3.1 Create new Scene, WebGL renderer and container

All the components in the main application have to be added to a scene. The scene will be rendered by a WebGL renderer. The renderer will then be appended to a container. Container is a ”div” element of HTML page.

The container is created and added to the ”body” of the main HTML page.

```
container = document.createElement( 'div' );
document.body.appendChild( container );
```

Figure 11. Create container

Renderer is created as shown in Figure 12.

```
renderer = new THREE.WebGLRenderer();
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight );
container.appendChild( renderer.domElement );
```

Figure 12. Create renderer

The scene is created as shown in Figure 13.

```
scene = new THREE.Scene();
```

Figure 13. Create scene

4.3.2 Create side-by-side effect for the scene.

The application needs to be side-by-side in order to be viewed as native 3D by the hardware.

Side-by-side means that the scene is divided into two parts. Each parts represents either the left eye or the right eye of a human being. After dividing the scene, when viewing on the hardware, the hardware will merge two parts into one with different frequency and the 3D glasses will be able to view as native 3D. Native 3D means you view the object exactly the same as seeing by human eyes.

The effect is created as shown in Figure 14.

```
effect = new THREE.StereoEffect( renderer );
effect.setSize( window.innerWidth, window.innerHeight );
```

Figure 14. Create Effect

4.3.3 Add Object and components to scene

a. Load 3D object using Loader and add to scene

The object file needed to be open using Object Loader.

```
var objectLoader = new THREE.ObjectLoader();
```

Figure 15. Create object loader

```
objectLoader.load(model.modelName, function ( obj ) {
    cube= obj;
    scene.add( cube );
    animate();
} );
```

Figure 16. Load model

b. Create and add camera

```
camera = new THREE.PerspectiveCamera( 45, window.innerWidth / window.innerHeight, 1,4000 );
camera.position.z = 3;
camera.position.x = 0;
camera.position.y = 0;
```

Figure 17. Create camera

Camera is created as PerspectiveCamera of ThreeJS.

c. Create and add light to scene

```
var ambient = new THREE.AmbientLight( 0x444444 );
scene.add( ambient );

var directionalLight = new THREE.DirectionalLight( 0xffeedd );
directionalLight.position.set( 0, 0, 1 ).normalize();
scene.add( directionalLight );
```

Figure 18. Create light

d. Create and add object mouse controller

```
controls = new THREE.TrackballControls( camera, renderer.domElement );
controls.rotateSpeed = 0.5;
controls.addEventListener( 'change', render );
```

Figure 19. Create object mouse control

e. Create and add light mouse controller

The light controller has to be implemented with the object controller in order to move the light source the same way with the object movement.

```
controlLight = new THREE.TrackballControls( directionalLight, renderer.domElement );
controlLight.rotateSpeed = 0.5;
controlLight.addEventListener( 'change', render );
```

Figure 20. Create light mouse control

4.3.4 Render the scene

After adding an object and components to the scene, the scene needs to be rendered using a WebGL renderer

When the object is added, the ”animate” function is called. The ”animate” function will call ThreeJS built-in function ”requestAnimationFrame” to render the every new frame of the scene and also call ”render” function. The ”Render” function is used to render scene with WebGL as shown in Figure 21.

```
function animate() {

    requestAnimationFrame( animate );
    render();
    TWEEN.update();
    controls.update();
    controlLight.update();

}

function render() {
    cube.rotation.y -= model.rotationSpeed;
    cube.scale.x = model.scale;
    cube.scale.y = model.scale;
    cube.scale.z = model.scale;
    effect.render( scene, camera );

}
```

Figure 21. Render

The ”animate” function also calls the update function of the object controller and light controller that is created.

4.3.5 Menu control implementation

The menu control is implemented using dat.GUI library. The menu could be toggled show/hide by pressing “H” button on the keyboard.

The initial value of the menu’s properties are created as shown in Figure 22.

```
var model = new function() {
        this.modelName = 'model.json';
        this.originalScale = 1;
         this.scale = this.originalScale;
         this.rotationSpeed = 0.01;
        };
```

Figure 22. Initialize menu properties

Create GUI and add components to the GUI as shown in Figure 23.

```
var gui = new dat.GUI();
var mName = gui.add(model, 'modelName', modelList);
var oScale = gui.add(model, 'originalScale');
var scaleRange = gui.add(model, 'scale', model.originalScale/2,model.originalScale);
var rotationSpeed = gui.add(model, 'rotationSpeed', 0,0.1);
```

Figure 23. Create menu GUI

a. Load object list

When added ”mName” property to the GUI, a variable ”modelList” is defined. This variable is a list of models that exists in web server directory. The list is fetched from:

```
$.getJSON(link, function (json) {
    modelList = json;
    init();
});
```

Figure 24. Get model list

The get model list code uses the ”getJSON” function which is provided by jQuery. jQuery is a fast, small and feature-rich Javascript library /6/.

The ”link” variable could be a JSON file path or an URL that return a JSON string. In this case, the ”link” variable is defined as shown in Figure 25.

```
var link = "http://www.cc.puv.fi/~e1200649/3js/getfilename.php";
```

Figure 25. Link

The ”link” is pointed to an URL which returns the JSON string containing all the models names existing in a web server directory as the function to get name list is shown in Figure 26.

```php
<?php
$dir = "./";
$emparray = array();
// Open a directory, and read its contents
if (is_dir($dir)){
  if ($dh = opendir($dir)){
    while (($file = readdir($dh)) !== false){
        //check all the json file in the directory
        if(strpos($file, '.json') !== false)
        {
            //append file name to name list
            $emparray[] = $file;
        }
    }
    closedir($dh);
  }
}
  echo json_encode($emparray);
?>
```

Figure 26. getfilename.php

b. Change model function

When a model is chosen from the list, a change model function is called as shown in Figure 27.

```
mName.onChange(function(newValue){
    model.modelName = newValue;
    var c = getCookie(newValue);
    var v= 0.001;
    if(c!=""){
        v = parseFloat(c);
    }
    model.originalScale = v;
    model.scale = v;
    gui.__controllers[3].remove();
    gui.__controllers[2].remove();
    gui.__controllers[1].updateDisplay();
    gui.add(model, 'scale', v/2,v);
    gui.add(model, 'rotationSpeed', 0,0.1);

    scene.remove(cube);
    objectLoader.load(newValue, function ( obj ) {
        cube= obj;
        scene.add( cube );
        animate();
    });
});
```

Figure 27. Change model

The change model function basically removes the old model and loads the new selected one.

c. Change original scale property

The change original scale function is created for the reason that the input model could be in any size. It could be too large or too small for the camera to capture. Changing the scale means that the size of the input model will change with a given ratio. The initial value is 0.001.

```
oScale.onFinishChange(function(newValue){
    setCookie(model.modelName, newValue);
    model.scale = newValue;
    model.originalScale = newValue;
    gui.__controllers[3].remove();
    gui.__controllers[2].remove();
    gui.add(model, 'scale', newValue/2,newValue);
    gui.add(model, 'rotationSpeed', 0,0.1);
});
```

Figure 28. Change original scale

The ”change original scale” function also changes the ”scale” property of the GUI. The scale properties can be simply understood as zooming. The range of zooming is from its half scale to its full scale value.

d. Change zoom and rotation speed

When changing the value of the ”scale” and ”rotationSpeed” properties of the GUI, the value is automatically updated because the ”render” frame is called and refreshed by the ”animate” function that has been mentioned above.

4.3.6 Store originalScale value using cookies

There exist various model sizes. The “originalScale” value of each object could be stored to cookies if it is given, otherwise the model will have initial value which is 0.001.

```
function setCookie(cname, cvalue) {

    document.cookie = cname + "=" + cvalue;
}
```

Figure 29. Set Cookies function

```
function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for(var i = 0; i <ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') {
            c = c.substring(1);
        }
        if (c.indexOf(name) == 0) {
            return c.substring(name.length,c.length);
        }
    }
    return "";
}
```

Figure 30. Get Cookies function

The first time application is run on a browser, the initial cookies will be set to the initial object shown on the scene.

```
var cScale = getCookie("model.json");
if(cScale==""){
    setCookie("model.json", 0.001);
}
```

Figure 31. Initial Cookies

When a new object is chosen from the list, the cookies value of the selected object will be got if it exists, otherwise, the initial value or the latest value that was set to the previous object will be applied for the selected object. If the value is not set by the user, the value will not be saved to cookies.

```
mName.onChange(function(newValue){
    model.modelName = newValue;
    var c = getCookie(newValue);
    var v= 0.001;
    if(c!=""){
        v = parseFloat(c);
    }
    model.originalScale = v;
    model.scale = v;
    gui.__controllers[3].remove();
    gui.__controllers[2].remove();
    gui.__controllers[1].updateDisplay();
    gui.add(model, 'scale', v/2,v);
    gui.add(model, 'rotationSpeed', 0,0.1);

    scene.remove(cube);
    objectLoader.load(newValue, function ( obj ) {
        cube= obj;
        scene.add( cube );
        animate();
    });
});
```

Figure 32. Get Cookies value

When a new value is set by a user, the value will be saved to cookies under the name of the object that is selected.

```
oScale.onFinishChange(function(newValue){
    setCookie(model.modelName, newValue);
    model.scale = newValue;
    model.originalScale = newValue;
    gui.__controllers[3].remove();
    gui.__controllers[2].remove();
    gui.add(model, 'scale', newValue/2,newValue);
    gui.add(model, 'rotationSpeed', 0,0.1);
});
```

Figure 33. Set Cookies Value

4.3.7 Mouse control implementation

ThreeJS library comes with a fully functional controller using mouse. This application uses built-in mouse control that comes with ThreeJS. The functions contain:

a. Zooming using the scrolling button of the mouse.
b. Rotating using the left mouse button and mouse movement.
c. Moving an object using the right mouse button and mouse movement.

4.3.8 Other functions

a. Window resize function

A function is called when window size is changed.

```
function onWindowResize() {

    windowHalfX = window.innerWidth / 2;
    windowHalfY = window.innerHeight / 2;

    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();

    effect.setSize( window.innerWidth, window.innerHeight );

}
```

Figure 34. Window resize

b. Transform function

```
function transform( targets, duration ) {

    TWEEN.removeAll();

    for ( var i = 0; i < objects.length; i ++ ) {

        var object = objects[ i ];
        var target = targets[ i ];

        new TWEEN.Tween( object.position )
            .to( { x: target.position.x, y: target.position.y, z: target.position.z }, Math.random() * duration + duration )
            .easing( TWEEN.Easing.Exponential.InOut )
            .start();

        new TWEEN.Tween( object.rotation )
            .to( { x: target.rotation.x, y: target.rotation.y, z: target.rotation.z }, Math.random() * duration + duration )
            .easing( TWEEN.Easing.Exponential.InOut )
            .start();

    }

    new TWEEN.Tween( this )
        .to( {}, duration * 2 )
        .onUpdate( render )
        .start();

}
```

Figure 35. Transform

4.4 Special Implementation

Special Implementation is implemented separately from the main implementation. The reason to implement this feature is so that Prima could give the drawing files of the object in multiple javascript files. Also a special implementation is needed to load drawing JS files.

The following code (Figure 36) is used to get all the drawing js files from the directory.

```
<?php
function getjs($folder){
$dir ="./".$folder ;
$emparray = array();
// Open a directory, and read its contents
if (is_dir($dir)){
  if ($dh = opendir($dir)){
    while (($file = readdir($dh)) !== false){
        //check all the json file in the directory
        if(strpos($file, '.js') !== false && strpos($file, $folder) !== false && strpos($file, 'main') === false)
        {
            //append file name to name list
            $emparray[] = $file;
        }
    }
    closedir($dh);
  }
}
    return $emparray;
}

  echo json_encode(getjs($_POST['folder']));
?>
```

Figure 36. Get all JS drawing files name

The following code (Figure 37) is used to call the previous php function to get and store all the name of the javascript files.

```
function loadInit(param){
    $.ajax({
    url: 'getjs.php',
    type: 'post',
    dataType:'json',
    data: { "folder": param},
    success: function(response) {console.log(response.length); fileList = response;

    init(fileList); }
});
}
```

Figure 37. Get and store all JS files name

The following code (Figure 38) is used to load the model part by part

```
function loadObj(name,fileList){

    var i=0;
    setInterval(function(){
        if(i<fileList.length){
            scriptName = name + "/" +name + "_" + i + ".js";
            functionName = "create_geometry_" + i + "(scene)";
            console.log(scriptName);
             console.log(functionName);
            $.loadScript(scriptName, function(){
                eval(functionName);
            });
        }
        else{

            return;
        }
        i++;
    }, 500);
}
```

Figure 38. Load Model

5 TESTING

The application is tested manually on a web browser. The Javascript errors can be shown on a browser's developer tools console. In Chrome, the developer tools can be opened by pressing "F12" and go to "Console" tab of the developer tools to see javascript errors.

Appearance testing is made by viewing the application on the hardware. In a real case, Samsung 3D TV and 3D glasses are used as testing hardware.

The server for storing web application is the server at VAMK, Vaasa University of Applied Sciences..

5.1 Uploading Test

5.1.1 Preparing models

To use the application, models in JSON format have to be prepared. To have models in JSON format, the user could either export from the modelling software they use or converts from 3D object to JSON. In this application, the converting method is used and a converting website is chosen to convert from 3D object to JSON.

When having 3D objects, go to https://clara.io, creates an account and login to Clara.io.

Click on upload to upload3D objects into Clara.io (Figure 39).

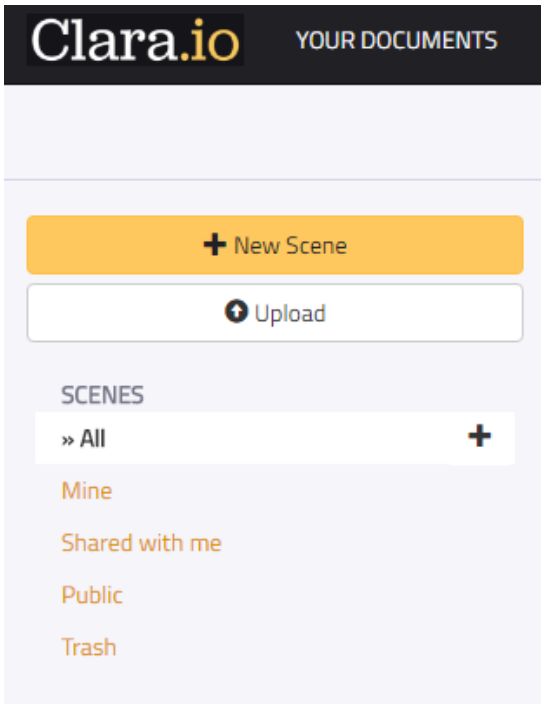

Figure 39. Upload 3D object to Clara.io

When uploading, it is possible to choose on the target renderer whether WebGL or V-Ray or both are used. WebGL should be selected (Figure 40).

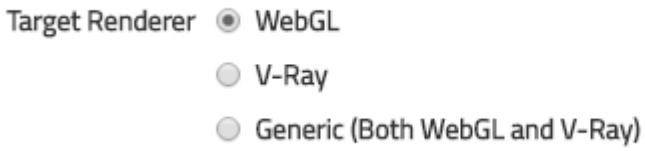

Figure 40. Set target to WebGL

After uploading the 3D model successfully, Clara.io will create a scene of the object. After that, select object needed to be exported to JSON, point to “Download” button and select “ThreeJS(json)” (Figure 41).

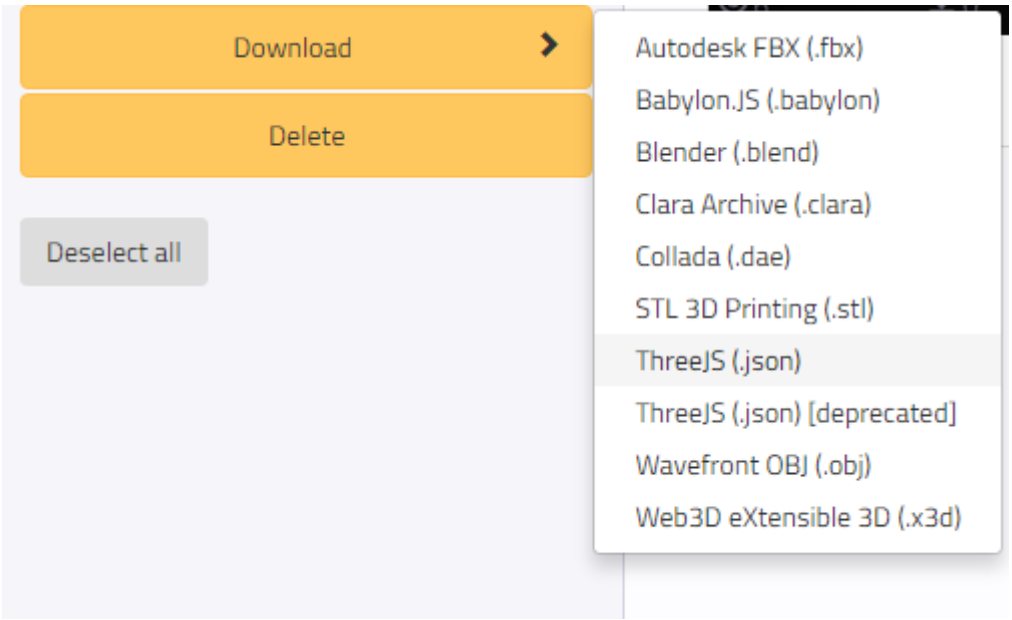

Figure 41. Download JSON

After choosing the download type, clara.io will export the object in the selected format and a zip folder containing a JSON object is ready for download. Download the folder and unzip it and the object is well prepared.

5.1.2 Upload object using admin tools

The admin tools is provided at http://www.cc.puv.fi/~e1200649/3js/admin.html

Click on ”Choose File” button and select the JSON object to upload (Figure 42)

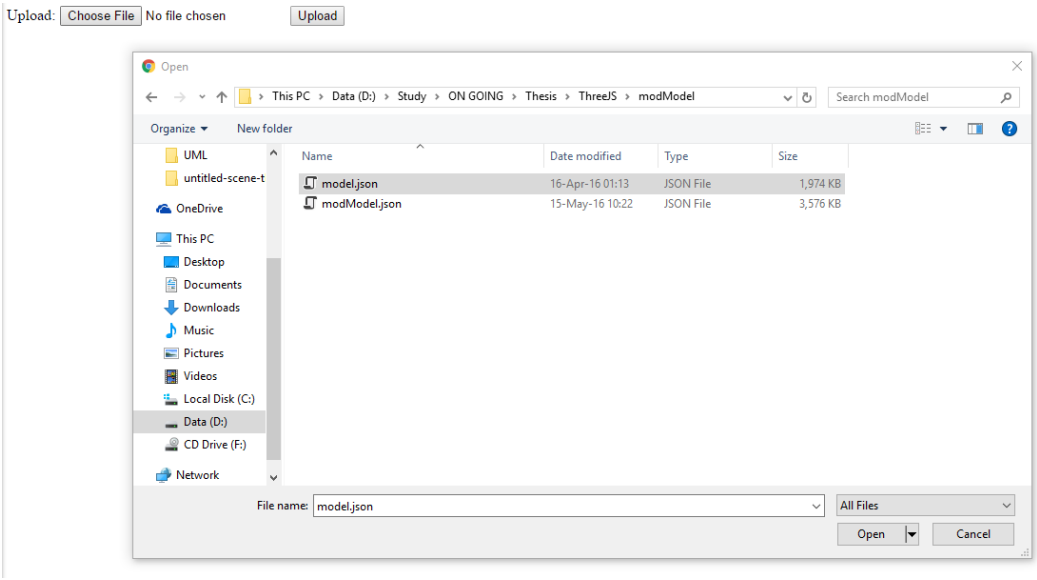

Figure 42. Choose file to upload

Click ”Upload” button to upload to web server.

If uploading succeeds, a message will be shown and the page will redirect to ”admin.html” after five seconds.

The file model.json has been uploaded.

Figure 43. Success message

If the file already exists, a message will be shown (Figure 44).

File already exists. Your file was not uploaded.

Figure 44. Exist message

There is a restriction on the university server that does not allow uploading files more than 3Mb to the server:

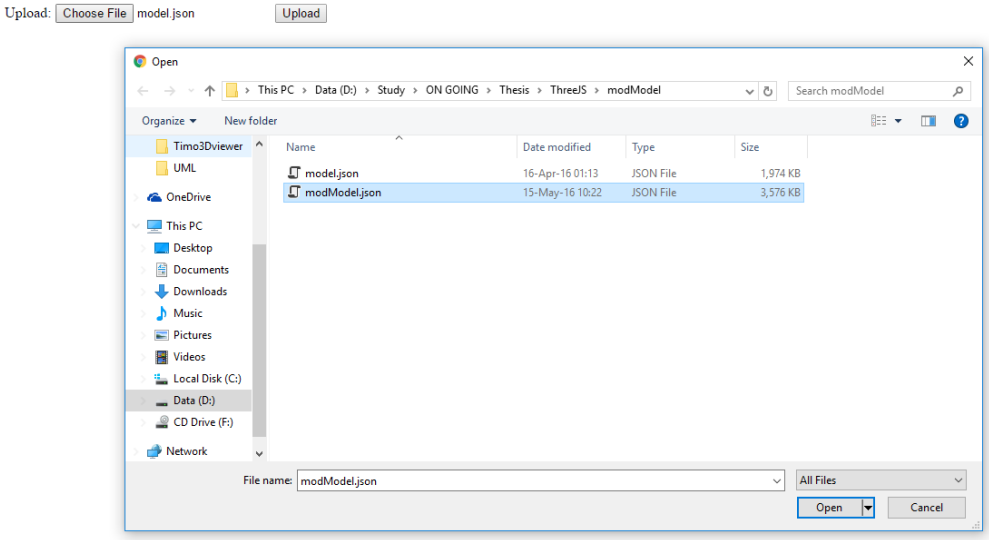

Figure 45. Choose file larger than 3Mb

When choosing file ”model.json” which is less than 3Mb, uploading succeeds but with ”modModel.json” which is 3.5Mb, uploading return error message as shown in Figure 46.

There was an error uploading your file.

Figure 46. Upload Error message

5.2 Main Application Test

The main application is provided at http://www.cc.puv.fi/~e1200649/3js/dat-gui.html

5.2.1 Menu Control Test

Figure 47. Initial State

The object is too large for the camera to view. A change in “originalScale” property has to be made from “1” to “0.001” (Figure 48).

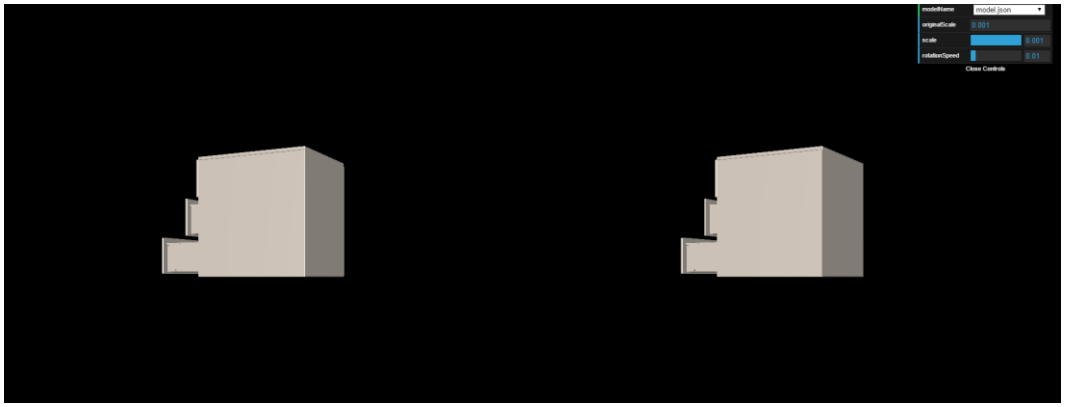

Figure 48. Change originalScale

Change “scale” property:

Figure 49. Change scale

The scale range in this case when the “originalScale” is set to ”0.001” will be from ”0.0005” to ”0.001”.

When choosing another model from the list, the setting of the menu control remain unchanged.

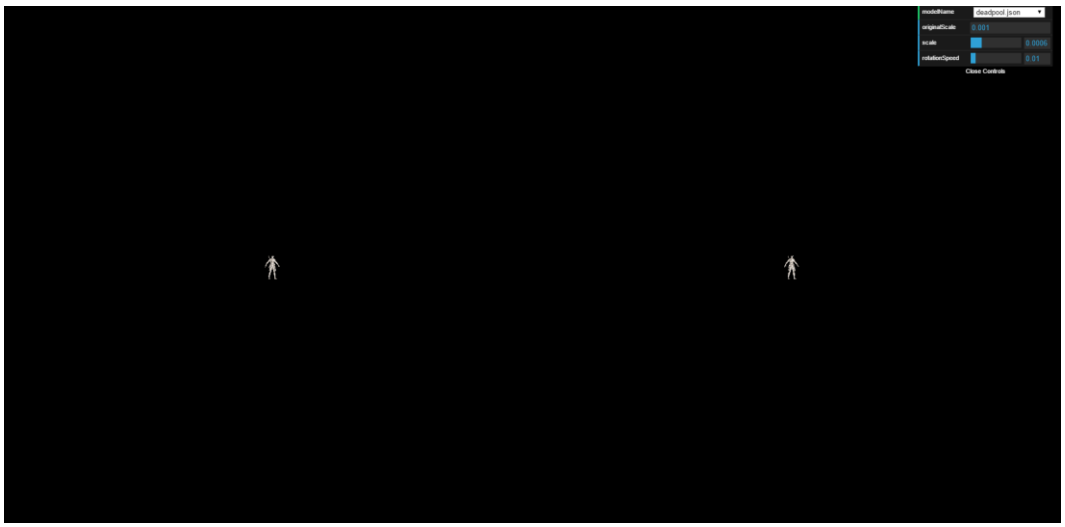

Figure 50. Change model

It is shown clearly by the test that model sizes are different from each other. That is the reason for changing the “originalScale” property function.

When ”rotationSpeed” changes, the object rotates faster. The automatic rotation speed ranges from ”0” to ”0.1”.

5.2.2 Mouse Control Test

Hold left mouse button and move the mouse to rotate the object (Figure 51).

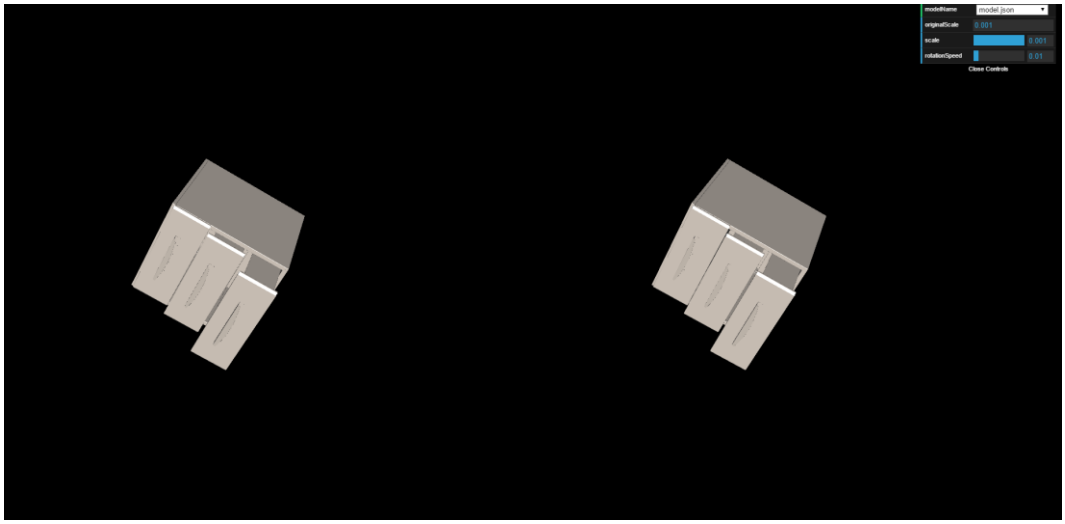

Figure 51. Mouse rotate

Use mouse scroll button to zoom in and out (Figure 52).

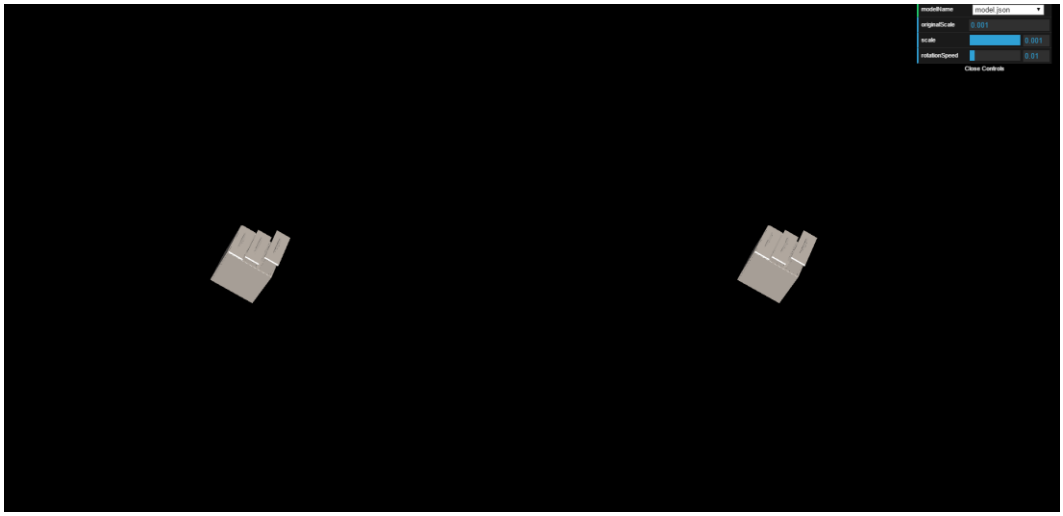

Figure 52. Mouse zoom

6 IMPLEMENTATION ANALYSIS

The main requirements are successfully implemented and tested. However, the advance features are not implemented fully. The application could not open any 3D objects without convertion to JSON. The information connection feature is not necessary at the moment.

In the implementation phase, the most challenging part is to debug the application. The application is mainly developed in Javascript and the only thing that could be shown is Javascript code fully broken. The next challenging is merging ThreeJS components together, controlling light with an object, using effect with the renderer. Beside that, getting dat.GUI library working with ThreeJS also has some challenge. The GUI component has to update one property itself when changing to another property. The GUI has to reload the object when choosing the object from the list. Every change has to be made and tested without any references or resources from the internet.

The application is developed and finished on time with an acceptable result. All basic requirements are fulfilled; one advance feature is fulfilled. However, the analysis phase took more time than expected. All the research technologies are learned, made a demo and tested.

The application could be improved by investigating multiple 3D objects importing without pre-converting process. If needed, the application could get the object's specifications and show the specifications somehow on the display.

7 CONCLUSION

The result of the application is viewable on a Samsung 3D TV using 3D glasses. The controlling system with mouse works as required and expected. The menu control works as required and expected. However, the application could not load different types of 3D extensions, the application only accepts 3D models in JSON format. It could be improved to accepts multiple 3D extensions without the preconverting step. The application performance for detailed and large models is quite low, it also depends on the hardware graphic card of the computer. The special implementation is made to load drawing javascript files. The drawing files draw the model partly. However, the special implementation is not totally automated and the camera information is hard-coded. It could be improved so that the special implementation could automatically get the information of the camera such as position and the view angle for the generated drawing files from the 3D models.

The thesis result satisfied the requirements of the customer and the result was usable for demonstration purposes. Advanced implementation could be implemented regards to the needs and purposes of the customer.

REFERENCES

Electronic publications

Github dat.GUI library.

https://github.com/dataarts/dat.gui

ThreeJS documentation

http://threejs.org/docs/index.html#Manual/Introduction/Creating_a_scene

ThreeJS examples

http://threejs.org/examples/

PHP File Upload

http://www.w3schools.com/php/php_file_upload.asp

jQuery Read JSON

http://api.jquery.com/jquery.getjson/

NUMBERED REFERENCES

In the theses written at Unit for Technology and Communication numbered references can be used. When referring to a source as a whole, no page numbers are required and the reference is marked like this: /1/. In situations where certain pages are referred to the reference is written like this: /1, 120-121/. If the number of sources is under 10, they are written in numbered order in the list of references, otherwise in alphabetical order. In other respects, the general instructions concerning references are the same as above.

MODEL OF A LIST OF REFERENCES FOR NUMBERED REFERENCES

The list formulated as above but the sources are in numerical order:

/1/ https://unity3d.com/

/2/ https://www.blender.org/about/

/3/ http://www.autodesk.com/products/maya/overview

/4/ https://en.wikipedia.org/wiki/WebGL

/5/ https://github.com/dataarts/dat.gui/blob/master/README.md

/6/ https://jquery.com/

APPEARANCE

1. **Figures,** drawings and photos are numbered with a running number. The word "**Figure**" with its number are in bold but the name of the figure is normal. The caption ends in a full stop and it is placed under the figure. The figures are separated with a blank line before and after from the text. The figures have to be referred to in the text.
2. **Appendices** are placed after the list of references. All figures, tables and lists that demonstrate the contents of the thesis but are too big in size or lesser in importance to be integrated in the text are marked as appendices. The questionnaires or interview questions used are also placed in the appendices. They are numbered with a running number (e.g. APPENDIX 1). If the number of appendices is big, a list of appendices is drawn up, placed after the contents page.
3. **A new chapter** is started on a new page. Each chapter should have at least two subtitles. In addition, each chapter should contain at least two paragraphs. A paragraph should contain at least three sentences.
4. **To emphasize** the text **bolding**, *italics* etc. can be used but sparingly and consistently. Underlining is not recommended.
5. **Dash (–)** is used between limits expressed either in numbers or in words. E.g. between 1941–1944, items 4–7, September – October, pages 12–15. In Word, the dash is produced with the combination [ctrl] [minus].
6. **Subheadings** are spelled so that important words are written with a capital initial. Prepositions and articles are written in small case. The title of the thesis in the abstract is spelled the same way. E.g. This is How a Good Title Looks Like.