3D Environment: WebGL and Three.js

Detailed report about technological choices, code and user functionalities

The Environment

3D Experience was developed with **Three.js** that allows the creation of GPU-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins. This is possible thanks to the advent of WebGL. High-level libraries such as Three.js or GLGE, SceneJS, PhiloGL or a number of other libraries make it possible to author complex 3D computer animations that display in the browser without the effort required for a traditional standalone application or a plugin.
Three.js is characterized by the following features:

Effects: Anaglyph, cross-eyed and parallax barrier.
Scenes: add and remove objects at run-time; fog
Cameras: perspective and orthographic; controllers: trackball, FPS, path and more
Animation: armatures, forward kinematics, inverse kinematics, morph and keyframe
Lights: ambient, direction, point and spot lights; shadows: cast and receive
Materials: Lambert, Phong, smooth shading, textures and more
Shaders: access to full OpenGL Shading Language (GLSL) capabilities: lens flare, depth pass and extensive post-processing library
Objects: meshes, particles, sprites, lines, ribbons, bones and more - all with Level of detail
Geometry: plane, cube, sphere, torus, 3D text and more; modifiers: lathe, extrude and tube
Data loaders: binary, image, JSON and scene
Utilities: full set of time and 3D math functions including frustum, matrix, quaternion, UVs and more
Export and import: utilities to create Three.js-compatible JSON files from within: Blender, openCTM, FBX, Max, and OBJ
Support: API documentation is under construction, public forum and wiki in full operation
Examples: Over 150 files of coding examples plus fonts, models, textures, sounds and other support files
Debugging: Stats.js, WebGL Inspector, Three.js Inspector
Three.js runs in all browsers supported by WebGL 1.0.

This library is based on **WebGL** is a JavaScript API for rendering 3D graphics within any compatible web browser without the use of plug-ins. WebGL is integrated completely into all the web standards of the browser allowing GPU accelerated usage of physics and image processing and effects as part of the web page canvas. Its elements can be mixed with other HTML elements and composited with other parts of the page or page background. WebGL programs consist of control code written in JavaScript and shader code that is written in OpenGL Shading Language (GLSL), a language similar to C or C++, and is executed on a computer's graphics processing unit (GPU).

Libraries, Tools and Models

The list the follows contains information about libraries and tools are used to develop the 3D Experience that will be showed and explained in detail in the next section.

Generals:

jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Bootstrap

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

Bootstrap Checkbox and Radio Switch:

It is a bootstrap plugins that as the aim to manage checkboxes and radio switchers. It permits to retrieve in easily way information, change interface representation and behaviours of radio and switcher components.

Chartist

Chartist.js is the product of a community that was disappointed about the abilities provided by other charting libraries. Of course there are hundreds of other great charting libraries but after using them there were always tweaks you would have wished for that were not included.

Bootstrap notify

Bootstrap Notify is a jQuery plugin to provide simple yet fully customisable notifications. The javascript code snippets in this documentation with the green edge are runnable by clicking them.

Light Bootstrap Dashboard

Light Bootstrap Dashboard is an admin dashboard template designed to be beautiful and simple. It is built on top of Bootstrap 3 and it is fully responsive. It comes with a big collections of elements that will offer you multiple possibilities to create the app that best fits your needs. It can be used to create admin panels, project management systems, web applications backend, CMS or CRM.

WebGL:

Three.js

See previous section: The **Environment**

Drag Controls

Drag Controls is a WebGL extension to manage drag controls inside the scene. Through this library you can drag object in the scene by the catching of specified events with Javascript Event Listeners.

Trackball Controls

It permits you to use Trackball controls to pan and move the camera around the scene, moreover it can be used to move the object around with the support of drag controls extension. With the trackball controls, you can very easily use your mouse to move the camera around the scene.

Stats

It permits to render statistics connect to the GPU and the scene renderization.

Renderer:

Projector

It is an extention to make vectors projection, it also useful to calculate intersections between mouse directrix and objects in the scene.

Technical Aspects and Implemented Interactions

To create **Scene**it was used a simple HTML div in which 3D object are renderized through the core render() function that deals in updating **controls** and to call the **render** function.
The render is accomplished by **THREE.WebGLRenderer** that is created and instantiated in the initialization function.
In the Initialization function all precalculated objects are created and inserted inside the **Scene**.
To create the complex tree structure a complex javascript object was created, it has the aim to mantain the whole informations about objects' dependecies and physical connections. Each complex object represent a **Database** structure, that is created as an objects composed by children with **autonomous** behaviour. Each **Database** is composed by a set of **Tables** and each Table is composed by a set of **Columns**. The Hierarchy between objects is underlined by positions, size, connections and even animations.
The object **DatabaseStruct** is composed by an identifier (multiple complex objects are inserted inside the Scene) , a base size used to define the difference between objects' size in relation to the explained hierarchy, the default informations about first object's position, default material for objects inside the "class", the default material for lines that connects each object to his direct father, what kind of controller the user wats to use for the specific complex object, and the texture is selected for a specified object inside the set of objects in the structure.

Follow the complete code of the function that initialize the Scene:

```
function init() {

  container = document.getElementById("canvas-container");
  camera = new THREE.PerspectiveCamera( camera_fov, window.innerWidth / window.innerHeight, 1, 10000 );
  setCameraPostion(
    initial_camera_position["x"],
    initial_camera_position["y"],
    initial_camera_position["z"]
  );

  controls = new THREE.TrackballControls( camera );
  controls.rotateSpeed = 1.0;
  controls.zoomSpeed = 1.2;
  controls.panSpeed = 0.8;
  controls.noZoom = false;
  controls.noPan = false;
  controls.staticMoving = true;
  controls.dynamicDampingFactor = 0.3;

  scene = new THREE.Scene();

  scene.add( new THREE.AmbientLight( 0x505050) );


  var light = new THREE.SpotLight( 0xffffff, 1.5 );
  light.position.set( 500, 800, 2000 );
  light.castShadow = true;

  light.shadow = new THREE.LightShadow( new THREE.PerspectiveCamera( 50, 1, 200, 10000 ) );
  light.shadow.bias = - 0.00022;

  light.shadow.mapSize.width = 2048;
  light.shadow.mapSize.height = 2048;
  scene.add( light );



  renderer = new THREE.WebGLRenderer( { antialias: true, alpha: true} );
  setRendererTheme(scene_theme);

  renderer.setPixelRatio( window.devicePixelRatio );
  renderer.setSize( window.innerWidth, window.innerHeight );
  renderer.sortObjects = false;

  renderer.shadowMap.enabled = true;
  renderer.shadowMap.type = THREE.PCFShadowMap;

  container.appendChild( renderer.domElement );


  <?php
    for($i=0; $i<count($udbs); $i++){
      $obj3d = $udbs[$i]->getObject3D();
      ?>
      <?php
      if($obj3d != null){
        ?>
        ds<?php echo($i); ?>= new DatabaseStruct("<?php echo($udbs[$i]->getName()); ?>", <?php echo($obj3d->getPositionX()); ?>, <?php echo($obj3d->getPositionY()); ?>, <?php

        var o<?php echo($i); ?> = ds<?php echo($i); ?>.createObject(
          "<?php echo($udbs[$i]->getId()); ?>",
          "Sphere",
          <?php echo($obj3d->getPositionX());?>,
          <?php echo($obj3d->getPositionY());?>,
          <?php echo($obj3d->getPositionZ());?>,
          <?php echo($obj3d->getSize());?>,
          "<?php echo($obj3d->getMaterial());?>",
          "<?php echo($obj3d->getColor());?>",
          "<?php echo($obj3d->getTexture());?>"
        );
        <?php

      }else{
        ?>
        ds<?php echo($i); ?>= new DatabaseStruct("<?php echo($udbs[$i]->getName()); ?>", <?php echo($i*800); ?>,0,0,null,null,3,2,"Drag");
        var o<?php echo($i); ?> = ds<?php echo($i); ?>.createObject(
          "<?php echo($udbs[$i]->getId()); ?>",
          "Sphere",
          <?php echo($i*800); ?>,
          0, 0, 60,
          "<?php
            $obj3d = $udbs[$i]->getObject3D();
            if($obj3d != null){
                echo($obj3d->getMaterial());
            }
            ?>",
          "<?php
          if($obj3d != null){
              echo($obj3d->getColor());
          }
          ?>","<?php
          if($obj3d != null){
              echo($obj3d->getTexture());
          }
          ?>");
        <?php
      }
      ?>
      <?php
        $tbs = $udbs[$i]->getTables();
        for($y=0; $y<count($tbs); $y++){
          $tbo = $tbs[$y]->getObject3D();
          if($tbo != null){
            ?>
            var tb<?php echo($y); ?> = ds<?php echo($i); ?>.createObject(
              "<?php echo($tbs[$y]->getId()); ?>",
              "Cube",
              <?php echo($tbo->getPositionX());?>,
              <?php echo($tbo->getPositionY());?>,
              <?php echo($tbo->getPositionZ());?>,
              <?php echo($tbo->getSize());?>,
              "<?php echo($tbo->getMaterial());?>",
              "<?php echo($tbo->getColor());?>",
              "<?php echo($tbo->getTexture());?>");

            <?php
          }else{
            ?>
            var tb<?php echo($y); ?> = ds<?php echo($i); ?>.createObject(
              "<?php echo($tbs[$y]->getId()); ?>",
              "Cube",
              null,
              null,
              null,
              40, 4);
            <?php
          }
          ?>

          ds<?php echo($i); ?>.connect(o<?php echo($i); ?>,tb<?php echo($y); ?>, true);
          <?php

          $cols = $tbs[$y]->getColumns();
          for($z =0 ; $z< count($cols); $z++){
            $clo = $cols[$z]->getObject3D();
            if($clo != null){
              ?>
              var cl<?php echo($z); ?> = ds<?php echo($i); ?>.createObject(
                "<?php echo($cols[$z]->getId()); ?>",
                "Sphere",
                <?php echo($clo->getPositionX());?>,
                <?php echo($clo->getPositionY());?>,
                <?php echo($clo->getPositionZ());?>,
                <?php echo($clo->getSize());?>,
                "<?php echo($clo->getMaterial());?>",
                "<?php echo($clo->getColor());?>",
              "<?php echo($clo->getTexture());?>");
              <?php
            }else{
              ?>
              var cl<?php echo($z); ?> = ds<?php echo($i); ?>.createObject(
                "<?php echo($cols[$z]->getId()); ?>",
                "Sphere",
                null,
                null,
                null,
                20, 4);
              <?php
            }
            ?>
            ds<?php echo($i); ?>.connect(tb<?php echo($y); ?>,cl<?php echo($z); ?>, true);
            <?php
          }
        }
        ?>
      ds<?php echo($i); ?>.addObject(o<?php echo($i); ?>, true);


      <?php
      echo("ds".$i.".initializeControls();");
      echo("ds".$i.".render();");
    }
  ?>

  function initializeGravity(reset){
    if(reset){
      resetAllConnections();
    }else{
      <?php
      for($i=0; $i<count($udbs); $i++){
        ?>
        animated_objects.push(ds<?php echo($i);?>.objects[0]);
        ds<?php echo($i);?>.lines.forEach(function(l){
          l.visible = false;
        });
        <?php
      }
      ?>
    }
  }

  var loaded = 0;
  var texture_number = 15;
  for(var i=0; i<texture_number;i++){
    texture_loader.load("assets/img/textures/texture"+i+".jpg", function(texture){
      texture.image.isMap = true;
      createTextureMaterial(texture);
      updateLoaded();
    });
  }

  function updateLoaded(){
    loaded = loaded+1;
    if(loaded == texture_number){
      console.log("now i can apply textures");
      <?php
      for($i=0; $i<count($udbs); $i++){
        echo("ds".$i.".applyTextureMaterial();");
      }
      ?>
    }
  }

  function setDisplayType(type){
    switch(type){
      case "graph":
        display_type = "graph";
        initializeGravity(true);
      break;
      case "gravity":
        display_type = "gravity";
        initializeGravity(false);
      break;
      default:
        display_type = "graph";
        initializeGravity(true);
      break;
    }
  }

  $('.change-display-type').click(function(){
    var type = $(this).attr("data-type");
    $('.change-display-type').parent().removeClass("li-selected");
    $(this).parent().addClass("li-selected");
    setDisplayType(type);
  });

  $('.change-animation-type').click(function(){
    $('.change-animation-type').parent().removeClass("li-selected");
    $(this).parent().addClass("li-selected");
    animation_type = $(this).attr("data-type");
  });

  clock = new THREE.Clock();
  window.addEventListener( 'resize', onWindowResize, false );

  setTimeout(function(){$('#loader').hide()}, 3000);

}
```