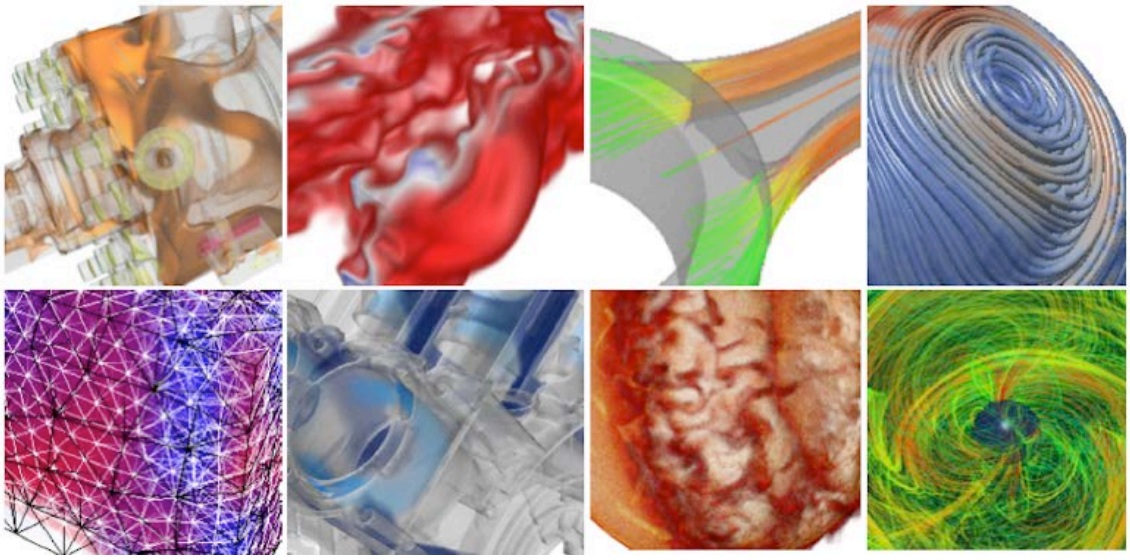
Information Visualization 2018



Documentation of the Workshop

Name: DENG XUEJIAO

Student Number:188x123x

Date:2018.6.13

KOBE UNIVERSITY

Contents

Description

Implementation Code

Description

1. Firstly, open the URL below (Chrome is recommended.):

https://hzlajojo.github.io/InfoVis2018/workshop/visualization.html,

then you will see this page on your Web browser.

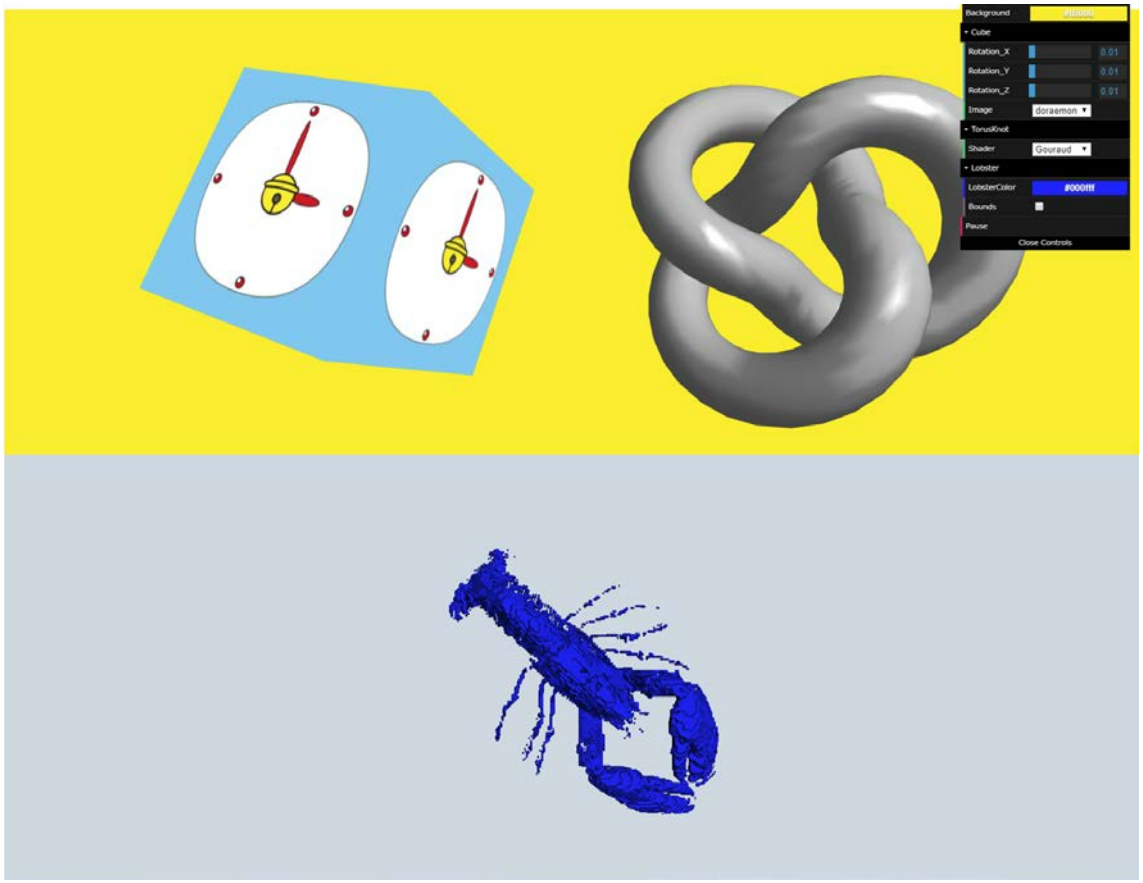


Fig.1 There are three objects(Cube, Torus Knot, Lobster) and one GUI controller, with which you can control some attributes (such as rotation speed, color and material) of each object by dragging or clicking it.

2. Components of the controller (Fig.2):

A) Background: change the color of the upper part's background;

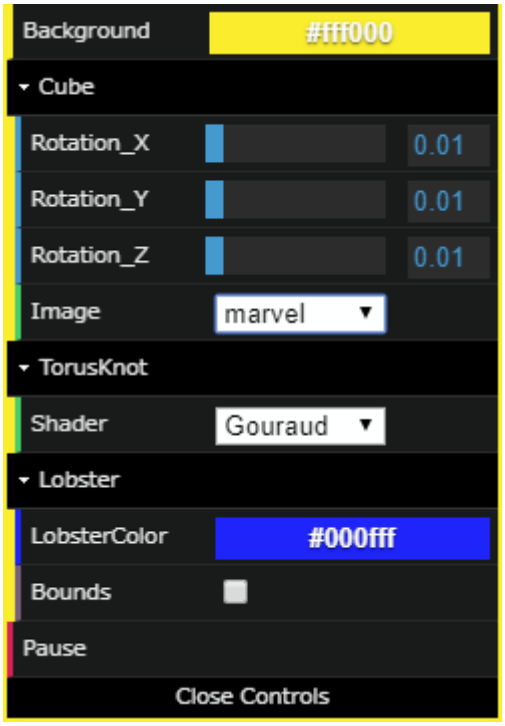
B) Cube(folder):

① Rotation_X: change the rotation speed of the cube in the x-axis direction;

② Rotation_Y: change the rotation speed of the cube in they y-axis direction;

③ Rotation_Z: change the rotation speed of the cube in the z-axis direction;

④ Image: switch the texture applied to the surface of the cube;



C) TorusKnot(folder):

Shader: switch the shader of the rotating torus knot.

D) Lobster(folder):

① LobsterColor: change the color of the lobster.

② Bounds: turn on/off the bounding box.

E) Pause: pop up the dialogue box with words “Pause!” to pause.

Fig.2 GUI controller

Fig.3 Comparison of two textures on the surface of cubes.

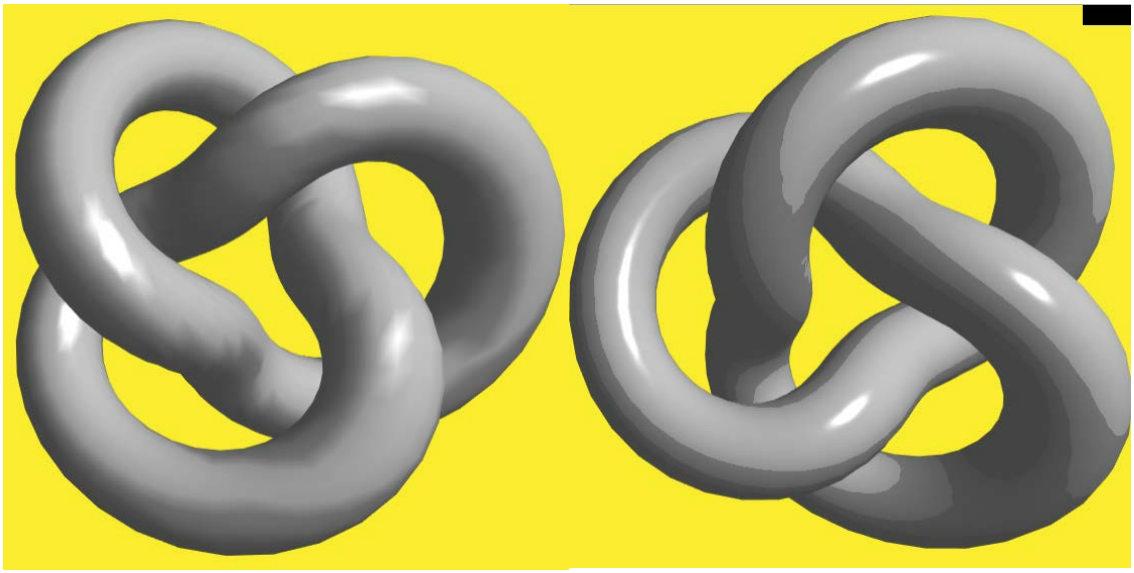


Fig.4 Comparison of two shaders applied to the rotating torus knots.

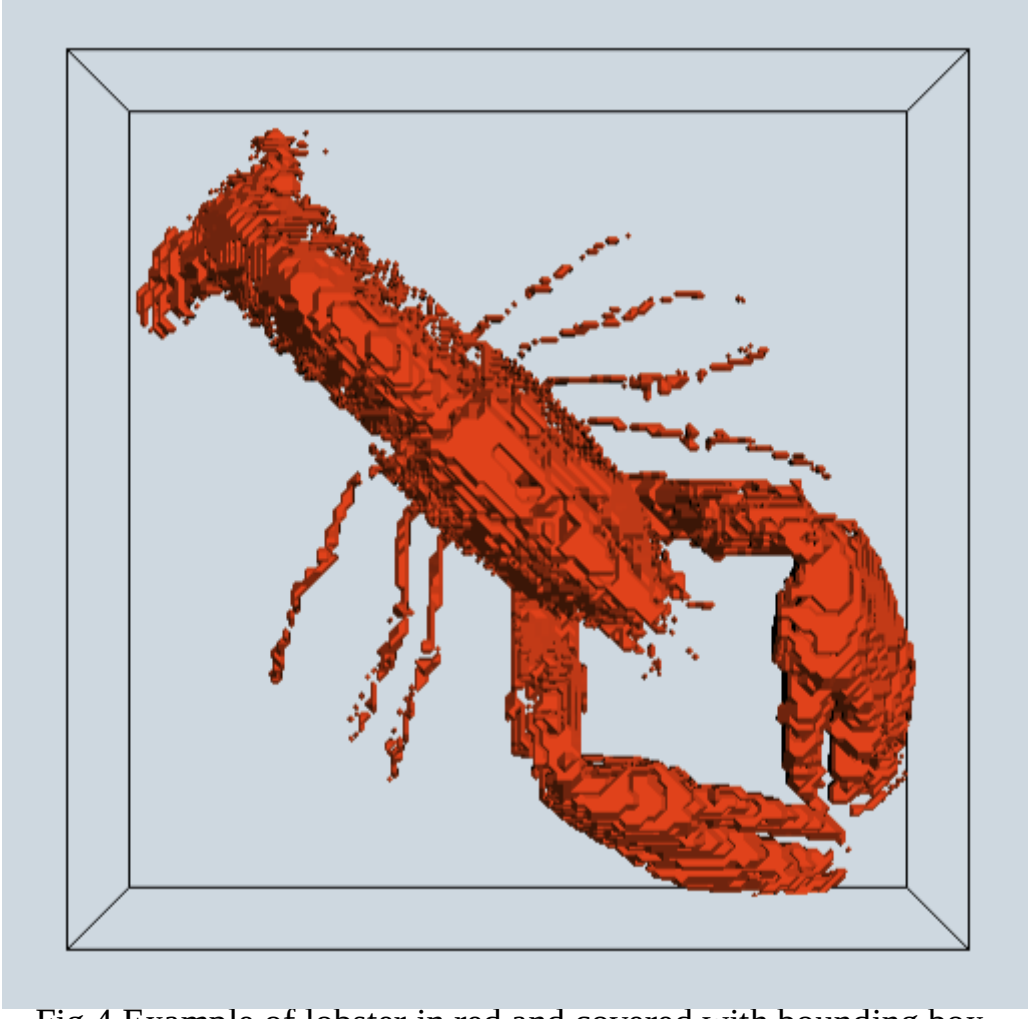


Fig.4 Example of lobster in red and covered with bounding box.

Implementation Code

```
▪<html> == $0
  <script type="text/javascript" charset="utf-8" id="zm-extension" src="chrome-extension://fdcgdnkidjaadafnichfpabhfomcebme/scripts/webrtc-patch.js" async>
  </script>
 ▶<head>…</head>
 ▼<body>
    <script src="three.min.js"></script>
    <script src="three.js"></script>
    <script src="cube.js"></script>
    <script src="three.min.js"></script>
    <script src="cube_torusknot.js"></script>
    <script src="TrackballControls.js"></script>
    <script src="Lut.js"></script>
    <script src="KVS.min.js"></script>
    <script src="KVS2THREE.min.js"></script>
    <script src="KVSLobsterData.js"></script>
    <script src="Bounds.js"></script>
    <script src="Isosurfaces.js"></script>
    <script src="lobster.js"></script>
    <script type="text/javascript" src="dat.gui.js"></script>
   ▶<script type="text/javascript">…</script>
   ▶<div class="dg ac">…</div>
    <!-- Gouraud Shading -->
   ▶<script type="x-shader/x-vertex" id="gouraud.vert">…</script>
   ▶<script type="x-shader/x-fragment" id="gouraud.frag">…</script>
    <!-- Phong Shading -->
   ▶<script type="x-shader/x-vertex" id="phong.vert">…</script>
   ▶<script type="x-shader/x-fragment" id="phong.frag">…</script>
    <script>
       cube_torusknot();
       lobster();
     </script>
    <canvas width="1665" height="642" style="width: 1665px; height: 642.5px;">
    <canvas width="1665" height="642" style="width: 1665px; height: 642.5px;">
  </body>
</html>
```

Declarations

Fig.5 This is the HTML file, in which all the code below goes into the <script> tag.

1. This HTML file consists of lots of declarations in the form of <script> tags, and there are some js files especially needed in order to call the functions to create the cube, the torus knot, the lobster and the GUI controller.More specifically, we need “three.js”, “three.min.js” for the cube, “three.min.js” for the torus knot, “three.min.js”, “Lut.js”, “KVS.min.js”, “KVS2THREE.min.js”, “KVSLobsterData.js”,

"Bounds.js", "Isosurfaces.js", "TrackballControls.js" for the lobster, and "data.gui.js" for the GUI controller.

2. Define the controller:

```
<script type="text/javascript"> == $0
            var obj = {
            Rotation_X: 0.01,
            Rotation_Y: 0.01,
            Rotation_Z: 0.01,
        Image:'doraemon',
        Shader:'Gouraud',
        Bounds: false,
            Pause: function () {
              alert('Pause!');
            },
            color: "#fff000",
        lobstercolor: "#000fff"
        };
      var Gui;
        var gui = new dat.gui.GUI();
      gui.addColor(obj, 'color').name("Background");
      Gui = gui.addFolder( "Cube" );
        Gui.add(obj, 'Rotation_X').min(0).max(0.1).step(0.01);
        Gui.add(obj, 'Rotation_Y').min(0).max(0.1).step(0.01);
        Gui.add(obj, 'Rotation_Z').min(0).max(0.1).step(0.01);
        Gui.add(obj, 'Image',['doraemon','marvel']);
      Gui.open();

      Gui = gui.addFolder( "TorusKnot" );
        Gui.add(obj, 'Shader', ['Gouraud' ,'Phong','NoShader'] );
      Gui.open();

      Gui = gui.addFolder( "Lobster" );
        Gui.addColor(obj, 'lobstercolor').name("LobsterColor").onChange(
  function (){
                var color = surfaces.material;
                if( color != obj.lobstercolor )
                screen.scene.remove(surfaces);
                surfaces.material.color = obj.lobstercolor;
                surfaces = Isosurfaces( volume, isovalue );
                screen.scene.add( surfaces );

          });
          Gui.add(obj, 'Bounds').name("Bounds").onChange(  function (value){
                if( obj.Bounds ) {
                    bounds =  Bounds( volume );
                    screen.scene.add( bounds  );
                }

                else {
                  screen.scene.remove( bounds );
                }

            });
      Gui.open();
      gui.add(obj, 'Pause');

</script>
```

Initialize the elements of the controller

3. Define the Gouraud Shading for the torus knot:

```
<!-- Gouraud Shading -->
<script type="x-shader/x-vertex" id="gouraud.vert">
    varying vec3 point_color;
    varying vec4 point_position;
    varying vec3 normal_vector;
    uniform vec3 light_position;
    uniform vec3 camera_position;
    vec3 ToonReflection(vec3 C,vec3 L,vec3 N,vec3 V){
      float ka = 0.3;
      float kd = 0.5;
      float ks = 0.8;
      float n = 50.0;
      vec3 H = normalize(L+V);
      vec3 R = reflect(-L,N);
      float dd = max(dot(N,L),0.0);
      float ds = pow(max(dot(R,V),0.0),n);
      if(dd <= 0.0){
        ds=0.0;
      }
      float Ia = ka;
      float Id = kd * dd;
      float Is = ks * ds;
      const float A = 0.1;
      const float B = 0.3;
      const float c = 0.6;
      const float D = 1.0;
      float df = max(0.0,dot(N,L));
      if(df<A) df = 0.0;
      else if(df<B) df = B;
      else if(df<c) df = c;
      else df = D;
      float sf = max(0.0,dot(N,H));
      sf = pow(sf,n);
      return vec3(Ia + df * Id + sf * Is);
    }
    void main() {
      point_position = modelViewMatrix * vec4(position,1.0);
      normal_vector = normalMatrix*normal;
      vec3 C = color;
      vec3 L = normalize(light_position - point_position.xyz);
      vec3 N = normalize(normal_vector);
      vec3 V = normalize(camera_position - point_position.xyz);
      point_color = ToonReflection(C,L,N,V);
      gl_Position = projectionMatrix * point_position;
    }

</script>
<script type="x-shader/x-fragment" id="gouraud.frag">
    varying vec3 point_color;
    void main() {
      gl_FragColor = vec4(point_color,1.0);
    }

</script>
```

4. Define the Phong Shading for the torus knot:

```
<!-- Phong Shading -->
<script type="x-shader/x-vertex" id="phong.vert">
    varying vec3 point_color;
    varying vec4 point_position;
    varying vec3 normal_vector;
    void main() {
      point_color = color;
      point_position = modelViewMatrix * vec4(position, 1.0);
      normal_vector = normalMatrix * normal;
      gl_Position = projectionMatrix * point_position;
    }

</script>
<script type="x-shader/x-fragment" id="phong.frag">
    varying vec3 point_color;
    varying vec4 point_position;
    varying vec3 normal_vector;
    uniform vec3 light_position;
    uniform vec3 camera_position;
    vec3 ToonReflection(vec3 C,vec3 L,vec3 N,vec3 V){
      float ka = 0.3;
      float kd = 0.5;
      float ks = 0.8;
      float n = 50.0;
      vec3 H = normalize(L+V);
      vec3 R = reflect(-L,N);
      float dd = max(dot(N,L),0.0);
      float ds = pow(max(dot(R,V),0.0),n);
      if(dd <= 0.0){
        ds=0.0;
      }
      float Ia = ka;
      float Id = kd * dd;
      float Is = ks * ds;
      const float A = 0.1;
      const float B = 0.3;
      const float c = 0.6;
      const float D = 1.0;
      float df = max(0.0,dot(N,L));
      if(df<A) df = 0.0;
      else if(df<B) df = B;
      else if(df<c) df = c;
      else df = D;
      float sf = max(0.0,dot(N,H));
      sf = pow(sf,n);
      return vec3(Ia + df * Id + sf * Is);
    }
    void main() {
      vec3 C = point_color;
      vec3 L = normalize(light_position - point_position.xyz);
      vec3 N = normalize(normal_vector);
      vec3 V = normalize(camera_position - point_position.xyz);
      vec3 shaded_color = ToonReflection(C,L,N,V);
      gl_FragColor = vec4(shaded_color,1.0);
    }

</script>
```

5. The function for the cube and the torus knot (it will be easier to read by combing them into one function “cube_torusknot”):

```
function cube_torusknot() {
  var width = window.innerWidth  ;
  var height = window.innerHeight/2 ;

  var scene = new THREE.Scene();

  var camera = new THREE.PerspectiveCamera( 45, width / height, 1, 1000 );
  camera.position.set( 0, 0, 5 );

  var light = new THREE.PointLight();
  light.position.set( 5, 5, 5 );
  scene.add( light );

  var renderer = new THREE.WebGLRenderer( { antialias: true }  );
  renderer.setClearColor(obj.color , 1.0);// Color of renderer
  renderer.setPixelRatio( window.devicePixelRatio );
  renderer.setSize( width, height );
  document.body.appendChild( renderer.domElement );

  var texture1 = new THREE.TextureLoader().load( 'doraemon.jpg' );
  var texture2 = new THREE.TextureLoader().load( 'marvel.jpg' );
  var geometry1 = new THREE.BoxBufferGeometry( 2, 2, 2 );
  var material1 = new THREE.MeshBasicMaterial( { map: texture1 } );
  var mesh = new THREE.Mesh( geometry1, material1 );
  mesh.position.x = -2;
  mesh.position.y =0;
  mesh.position.z = 0;
  scene.add( mesh );

  var geometry2 = new THREE.TorusKnotGeometry( 1, 0.3, 100, 20 );
  var Phong = new THREE.ShaderMaterial ({
    vertexColors: THREE.VertexColors,
    vertexShader: document.getElementById('phong.vert').text,
    fragmentShader: document.getElementById('phong.frag').text,
    uniforms: {
      light_position: { type: 'v3', value: light.position }
    }
  });
  var Gouraud = new THREE.ShaderMaterial ({
    vertexColors: THREE.VertexColors,
    vertexShader: document.getElementById('gouraud.vert').text,
    fragmentShader: document.getElementById('gouraud.frag').text,
    uniforms: {
      light_position: { type: 'v3', value: light.position }
    }
  });
  var material2 = Gouraud;
  var torus_knot = new THREE.Mesh( geometry2, material2 );
  torus_knot.position.x = 2;
  torus_knot.position.y = 0 ;
  torus_knot.position.z = 0 ;
  scene.add( torus_knot );

  window.addEventListener( 'resize', onWindowResize, false );

      function onWindowResize() {
              camera.aspect = width / height;
              camera.updateProjectionMatrix();
              renderer.setSize( width , height );
      }
```

Create a cube with texture covering its surface

Create a torus knot with shader on its surface

```
  loop();
  function loop() {
    requestAnimationFrame( loop );
    renderer.setClearColor(obj.color,1.0);// Color of renderer
    switch (obj.Image) {                          //switch the texture for the cube
      case 'doraemon': mesh.material = new THREE.MeshBasicMaterial( { map: texture1 } );break;
      case 'marvel':  mesh.material = new THREE.MeshBasicMaterial( { map: texture2 } );break;
    }
        mesh.rotation.x += obj.Rotation_X;
        mesh.rotation.y += obj.Rotation_Y;
        mesh.rotation.z += obj.Rotation_Z;

    var material;
  switch (obj.Shader) {                          //switch the shader for the torus knot
    case 'NoShader': material = new THREE.ShaderMaterial ();break;
    case 'Gouraud':  material = Gouraud;break;
    case 'Phong': material = Phong;break;
  }
    torus_knot.material = material;
    torus_knot.rotation.x += 0.01;
    torus_knot.rotation.y += 0.01;

    renderer.render( scene, camera );
  }
}
```

6. The function of the lobster:

```
var surfaces;
var screen;
var volume = new KVS.LobsterData();
var isovalue = 128;
var bounds;

function lobster()
{
    screen= new KVS.THREEScreen();

    screen.init( volume, {
        width:  window.innerWidth  ,
        height: window.innerHeight/2,
        enableAutoResize: false
    });

    surfaces = Isosurfaces( volume, isovalue );
    screen.scene.add( surfaces );

    document.addEventListener( 'mousemove', function() {
        screen.light.position.copy( screen.camera.position );
    });

    window.addEventListener( 'resize', function() {
        screen.resize( [ width, height ] );
    });

    screen.loop();
}
```