Nombre: Néstor Steven Aguilar garzón. Código: U6000423

Redactar (en sus propias palabras) un documento sobre las vistas perspectiva y ortográfica en computación gráfica.

Responder entre otras cuestiones:

1. ¿Qué es la vista perspectiva y en qué situaciones se aplica?

   La vista perspectiva se utiliza para representar los objetos en la disposición y forma que están a la vista, se emplea para crear ilusión de profundidad y espacio, una forma de crear imágenes reales y detalladas mediante nuestros ojos.

   Uso:

   Se aplica en diversas situaciones en la creación de video juegos, realidad virtual, dibujo técnico también se utiliza para la creación de planos o maquetas para una construcción, si se necesita representar objetos y/o espacios se aplica para poderlos ver de una forma más real.

2. ¿Qué es la vista ortográfica y en qué situaciones se aplica?

   Se utiliza para representar objetos en dos dimensiones, en la vista ortográfica los objetos se dibujan sin perspectiva, se dibujan con las proyecciones de sus lados.

   Uso:

   Se aplica en situaciones para mostrar los detalles de los objetos con una mayor precision, se utiliza para diagramas técnicos diseños de modas e ingeniería, esta permite visualizar con exactitud la forma y el tamaño de los objetos.

3. ¿Cómo se calcula una vista en perspectiva en la computación gráfica y qué parámetros se utilizan en su cálculo?

   La vista perspectiva en computación se calcula por medio del uso de una cámara, esta se coloca en punto específico y después se orienta hacia el objeto que se va a representar.

   Parámetros que se utilizan en su cálculo.

   posición de la cámara: se define la posición de la cámara en el espacio, es su ubicación en los ejes x, y y z.

   La dirección de la cámara: en esta se define la dirección hacia la cual la cámara va a apuntar, es el ángulo en cual se sitúa la cámara.

La distancia focal: esta es la distancia entre el objeto y la camara.

El campo de visión: es el rango de apertura que va tener la cámara.

4. ¿Cuáles elementos intervienen en la configuración de las vistas referidas y qué significado tiene cada uno de ellos en THREE.js?

   La cámara: define posición y perspectiva en la cual se renderiza la escena.
   El escenario: se definen los objetos que se van a renderizar.
   El renderizador: la información de la cámara y el escenario se convierte y esto se visualiza en una imagen.
   El lienzo: en el lienzo se renderiza la imagen generada por el renderizador.

5. Crear dos ejemplos (*perspectiva.htm* y *ortografica.htm*) para THREE.js en que se visualice el modelo (no renderizado) de un mismo escenario (una figura cualquiera, cubo, esfera, pirámide, o cualquiera otra generado a partir de los puntos vértices y no con la geometrías básicas predefinidas). Incluir como mínimo ejes principales XYZ, mall de plano XZ y un componente OrbitControls.

   - Vista Perspectiva

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Perspectiva</title>
  <style>
   body { margin: 0; }
   canvas { display: block; }
  </style>
 </head>
 <body>
  <script src="https://threejs.org/build/three.min.js"></script>
  <script src="https://threejs.org/examples/js/controls/OrbitControls.js"></script>
  <script>
   var scene = new THREE.Scene();

   var camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
   camera.position.z = 5;

   var renderer = new THREE.WebGLRenderer();
   renderer.setSize(window.innerWidth, window.innerHeight);
```

```
    document.body.appendChild(renderer.domElement);

    var controls = new THREE.OrbitControls(camera, renderer.domElement);

    var axesHelper = new THREE.AxesHelper(2);
    scene.add(axesHelper);

    var plane = new THREE.Mesh(new THREE.PlaneGeometry(10, 10), new
THREE.MeshBasicMaterial({ color: 0xffffff }));
    plane.rotation.x = -Math.PI / 2;
    scene.add(plane);

    var geometry = new THREE.Geometry();
    geometry.vertices.push(
      new THREE.Vector3(-1, 0, 0),
      new THREE.Vector3(0, 1, 0),
      new THREE.Vector3(1, 0, 0),
      new THREE.Vector3(0, 0, 1)
    );
    geometry.faces.push(
      new THREE.Face3(0, 1, 2),
      new THREE.Face3(0, 2, 3),
      new THREE.Face3(0, 3, 1),
      new THREE.Face3(1, 3, 2)
    );
    geometry.computeFaceNormals();

    var material = new THREE.MeshBasicMaterial({ color: 0xff00ff, wireframe: true });
    var mesh = new THREE.Mesh(geometry, material);
    scene.add(mesh);

    function animate() {
      requestAnimationFrame(animate);
      renderer.render(scene, camera);
    }

    animate();
   </script>
  </body>
</html>
```

- Vista ortográfica

```
<!DOCTYPE html>
```

```
<html>
   <head>
      <meta charset="utf-8">
      <title>Ortografica</title>
      <style>
         body { margin: 0; }
      </style>
   </head>

   <body>
    <h1> </h1>
      <script src="https://threejs.org/build/three.js"></script>
      <script type="text/javascript" src="dat.gui.js"></script>
      <script>

         //Se llama la camara y la escena
          const scene = new THREE.Scene();
          const camera = new THREE.PerspectiveCamera(30, window.innerWidth /
window.innerHeight);

         //Se llama el renderizador
          const renderer = new THREE.WebGLRenderer();
          renderer.setSize( window.innerWidth, window.innerHeight );
          document.body.appendChild( renderer.domElement );

         //figura
          var n=0;
          const geometry = new THREE.ConeGeometry( 1, 1, 10);
          const material = new THREE.MeshNormalMaterial( {color: 0x808000 } );
          const cone = new THREE.Mesh( geometry, material );
          scene.add( cone );
          cone.rotation.z=180
          cone.rotation.y=180
          cone.rotation.x=90


         //cuadricula
          const size = 20;
          const divisions =20 ;
          const gridHelper = new THREE.GridHelper( size, divisions );
          scene.add( gridHelper );

         //  Se ajusta la camara
          camera.position.z = -2;
```

```
        camera.position.y = 0.3;
        camera.position.x = 5;
        camera.lookAt(scene.position);
        camera.rotation.set(0,360,0);

      //ejes X,Y,Z
        const axesHelper = new THREE.AxesHelper(50);
        const negativoaxesHelper = new THREE.AxesHelper(-50);
        scene.add(axesHelper,negativoaxesHelper);

        function pieza1(){
          cube.position.y = 0.0;
        }

        function animate() {
          requestAnimationFrame( animate );
          renderer.render( scene, camera );
          }
        animate();

    </script>
  </body>
</html>
```

6. Relacionar las fuentes bibliográficas y/o webgrafía utilizadas en el desarrollo del presente trabajo.


*LA PERSPECTIVA*. (s. f.). Informática y Medios AudioVisuales Denian16.

*https://denian16.jimdofree.com/exposiciones/perspectiva/

*Perspectiva/Ortográfica — Blender Manual*. (s. f.).

*https://docs.blender.org/manual/es/2.91/editors/3dview/navigate/projections.html

*http://www.cs.uns.edu.ar/cg/clasespdf/3-Pipe3D.pdf

School, C. 3. A. (2019). ¿Vista en Perspectiva u Ortográfica al esculpir? — THE CUBE.

**THE CUBE*. https://www.thecube3danimation.com/blog-1/2019/2/6/vista-en-perspectiva-u-ortogrfica-al-esculpir

7. Publicar los tres archivos referidos anteriormente en un repositorio de nombre ***CG-231-B-301***