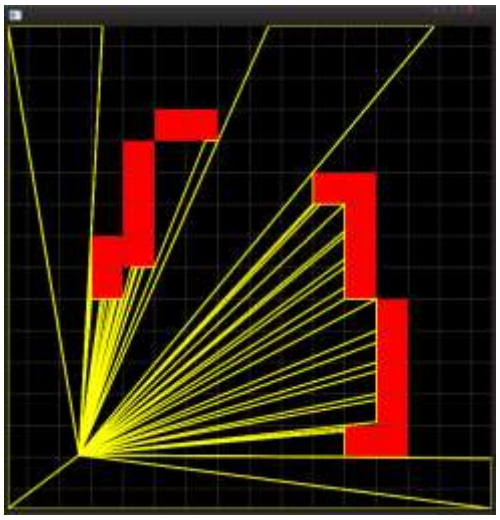
Sustentación

Primero que todo se buscó interiorizar el concepto y funcionalidad de raycasting y consta en un algoritmo que lanza rayos desde un punto de vista determinado con algunas propiedades, en este caso al ser una cámara de perspectiva todos los rayos saldrán del punto inicial hasta el far clip plane o un objeto. Al suceder esto solo se renderizara la parte de los objetos que choque con estos rayos.



El programa para hacer raytracing no me funciono, pero aquí está la explicación del intento:

- Primero busque ejemplos existentes de raytracing en la pagina de threejs y dentro de su documentación me salió esto:

```
var raycaster = new THREE.Raycaster();
var mouse = new THREE.Vector2();

function onMouseMove( event ) {

    // calculate mouse position in normalized device coordinates
    // (-1 to +1) for both components

    mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
    mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;

}

function render() {

    // update the picking ray with the camera and mouse position
    raycaster.setFromCamera( mouse, camera );

    // calculate objects intersecting the picking ray
    var intersects = raycaster.intersectObjects( scene.children );

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

        intersects[ i ].object.material.color.set( 0xff0000 );

    }

    renderer.render( scene, camera );

}

window.addEventListener( 'mousemove', onMouseMove, false );

window.requestAnimationFrame(render);
```

- Implementando esto a mi programa y tomando como ejemplo otros programas exitentes llegue a obtener esto:

```
var rollOverGeo  = new THREE.SphereGeometry(3, 8, 8);
var geometry2 = new THREE.CylinderGeometry(2, 2, 5, 32);
var geometry3 = new THREE.DodecahedronGeometry(3, 0);
var geometry4 = new THREE.TorusBufferGeometry(6, 3, 16, 100);
var rollOverMaterial = new THREE.MeshStandardMaterial({ color: 0x00ff00 });
var meshMaterial2 = new THREE.MeshStandardMaterial({ color: 0xB931E8 });
var meshMaterial3 = new THREE.MeshStandardMaterial({ color: 0x0046EB });
var meshMaterial4 = new THREE.MeshStandardMaterial({ color: 0x17BFE8 });
rollOverMaterial.vertexColors = THREE.FaceColors;
meshMaterial2.vertexColors = THREE.FaceColors;
meshMaterial3.vertexColors = THREE.FaceColors;

var lineMaterial = new THREE.LineBasicMaterial( { color: 0xffffff, transparent: true, opacity: 0.5 } );


var rollOverMesh = new THREE.Mesh(rollOverGeo, rollOverMaterial);
var cylindre = new THREE.Mesh(geometry2, meshMaterial2);
var dode = new THREE.Mesh(geometry3, meshMaterial3);

var lines = new THREE.LineSegments(rollOverGeo, lineMaterial);
var lines2 = new THREE.LineSegments(geometry2, lineMaterial);
var lines3 = new THREE.LineSegments(geometry3, lineMaterial);
raycaster = new THREE.Raycaster();
raycaster.setFromCamera(  camera );
```

Cree las geometrías y materiales, luego pase las geometrías de vertexColors a faceColors y por ultimo cree el raycaster con la posición de la cámara.

```
var group = new THREE.Group();
var group2 = new THREE.Group();
var group3 = new THREE.Group();
group.add(rollOverMesh);
group2.add(cylindre);
group3.add(dode);
group.add(lines);
group2.add(lines2);
group3.add(lines3);
group.translateZ(-10);
group2.translateZ(-15);
group2.translateX(5);
group3.translateX(-5);
group3.translateZ(-15);
scene.add(group);
scene.add(group2);
scene.add(group3);
objects.push(group);
objects.push(group2);
objects.push(group3);


camera.position.x = 5;
camera.position.y = 5;
camera.position.z = 10;
```

Después cree un grupo por cada mesh para posicionarlo y agregarle las líneas que muestran la forma del mesh, los agregue a la escena y les di un .push para llamarlos como objeto en otra función.

```
function ray (){
    raycaster.setFromCamera(camera);
    var intersects = raycaster.intersectObjects(objects);
    if(intersects.length>0){
        var intersect = intersects[0];

        rollOverMesh.position.copy( intersect.point ).add( intersect.face.normal );
            rollOverMesh.position.divideScalar( 50 ).floor().multiplyScalar( 50 ).addScalar( 25 );
    }
    render();
```

Aquí calculamos la intersección de los objetos y lo enviamos a la función render().

Al no funcionar esto y otros múltiples intentos le agregue mas mesh al código de backfaceculling, pero este no es muy optimo ya que toca mesh por mesh.