Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

ingeniería en Ciencias de la Computación

Graficación - Primavera 2023

Evaluación 4: realidad Aumentada

Fernando José Gil Uribe

fernando.gilu@alumno.buap.mx

12 de Mayo 2023

Resumen

En este proyecto se exploró la integración de modelos tridimensionales en una escena virtual utilizando la biblioteca de JavaScript ThreeJS. Además, se implementó la tecnología de Realidad Aumentada utilizando AR.js para mostrar la escena tridimensional en un plano de la vida real. Se logró animar los objetos en Realidad Aumentada utilizando funciones seno y coseno relacionadas al tiempo y se relacionó el movimiento y escala de un objeto a la frecuencia promedio de la música que reproduce la página. Los modelos tridimensionales utilizados se obtuvieron de repositorios web de modelos 3D gratuitos y se integraron a la escena utilizando el OBJLoader nativo de ThreeJS. Uno de los principales logros del proyecto fue mantener el mismo principio de animación que se usó en proyectos anteriores y la posibilidad de reutilizar código.

Introducción

El presente informe técnico tiene como objetivo documentar el proyecto desarrollado en ThreeJS y AR.js, en el cual se creó una escena tridimensional que puede ser visualizada a través de Realidad Aumentada. En este informe se describirá el uso de modelos tridimensionales, la integración de la Realidad Aumentada y la animación de objetos en la escena 3D.

El proyecto tiene como objetivo explorar la capacidad de ThreeJS para crear escenas tridimensionales interactivas y su integración con la Realidad Aumentada utilizando AR.js. Además, se buscó dar vida a la escena mediante la animación de objetos, lo que permitió una experiencia más inmersiva para el usuario.

Este informe está dirigido a desarrolladores y diseñadores interesados en la creación de escenas tridimensionales interactivas y su integración con la Realidad Aumentada. También puede ser de interés para aquellos que quieran aprender sobre el uso de ThreeJS y AR.js en proyectos de desarrollo web.

Uso de modelos tridimensionales

En este proyecto, se utilizaron modelos tridimensionales para crear una escena en ThreeJS. Para poder cargar estos modelos, se usó el OBJLoader nativo de ThreeJS y se cargaron archivos de tipo OBJ. El OBJLoader se configuró para que reciba una función de carga que permitiera modificar el modelo cargado según las necesidades del proyecto. También se trasladó la variable del objeto creado a una variable global del programa para poder manipularla posteriormente en las animaciones. A pesar de que la carga de los modelos puede ser un poco compleja al principio, gracias a la documentación y la experiencia previa con ThreeJS, se facilitó su integración en el proyecto.

Explicación del código

Una función para dividir un texto en líneas y centrar cada línea:

```
function multiline(list){
    var max = 0
    for(const s of list){
        if(s.length > max) max = s.length
    }
    var temp = ""
    for(var s of list){
        while (s.length < max) {
            s = " " + s
        }
        temp += s + "\n"
    }
    return temp
}
```

Creación y carga del texto en la escena de RA

```
// load a font resource
fontLoader.load('fonts/helvetiker_regular.typeface.json', function (font) {
const nl = "\n"
const textGeometry = new THREE.TextGeometry(multiline([
    "4a Evaluacion",
    "Proyecto final RA",
    "Modelos del Proyecto 3 llevados a un",
    "Ambiente de Realidad Aumentada",
    "por Fernando Gil"
]), {
        font: font,
        size: .1,
        height: 0.01,
        curveSegments: 12,
        bevelEnabled: false,
        bevelThickness: 0.05,
        bevelSize: 0.05,
        bevelSegments: 5
    });
textGeometry.center()
const textMaterial = new THREE.MeshPhongMaterial({ color: 0x00ffff });
textMesh = new THREE.Mesh(textGeometry, textMaterial);
group1.add(textMesh);
});
```

Utilidades para trabajar con RA

```
//////////////////////////////////////////////////////////////////
// setup arToolkitSource
//////////////////////////////////////////////////////////////////

arToolkitSource = new THREEx.ArToolkitSource({
    sourceType : 'webcam',
});

function onResize()
{
    arToolkitSource.onResize()
    arToolkitSource.copySizeTo(renderer.domElement)
    if ( arToolkitContext.arController !== null )
    {
        arToolkitSource.copySizeTo(arToolkitContext.arController.canvas)
    }
}

arToolkitSource.init(function onReady(){
    onResize()
});

// handle resize event
window.addEventListener('resize', function(){
    onResize()
});

//////////////////////////////////////////////////////////////////
// setup arToolkitContext
//////////////////////////////////////////////////////////////////

// create atToolkitContext
arToolkitContext = new THREEx.ArToolkitContext({
    cameraParametersUrl: 'data/camera_para.dat',
    detectionMode: 'mono'
});

// copy projection matrix to camera when initialization complete
arToolkitContext.init( function onCompleted(){
    camera.projectionMatrix.copy( arToolkitContext.getProjectionMatrix() );
});
```

Grupos de objetos para trabajar con la escena de RA

```
//////////////////////////////////////////////////////////
// setup markerRoots
//////////////////////////////////////////////////////////

// build markerControls
markerRoot1 = new THREE.Group();
group1 = new THREE.Group();
group2 = new THREE.Group();
markerRoot1.add(group1)
markerRoot1.add(group2)
group2.visible = false
scene.add(markerRoot1);
let markerControls1 = new THREEx.ArMarkerControls(arToolkitContext, markerRoot1, {
    type: 'pattern', patternUrl: "data/hiro.patt",
})
```

Creación de una curva como trayecto de animación:

```
// Curve
curve = new THREE.CatmullRomCurve3( [
    new THREE.Vector3( .5, .5, .5 ),
    new THREE.Vector3( -.5, 0.5, .5 ),
    new THREE.Vector3( -.5, .5, -.5 ),
    new THREE.Vector3( .5, 0.5, -.5 ),
    new THREE.Vector3( .5, .5, .5 ),
], true);

curvePoints = curve.getPoints( 50 );
const geometry = new THREE.Geometry();
geometry.vertices = curvePoints;
const material = new THREE.LineBasicMaterial( { color: 0xff0000 } );
const curveObject = new THREE.Line( geometry, material );
```

Carga de un modelo y sus respectivas funciones de carga y error

```
function onModelLoadProgress(xhr) {
// console.log((xhr.loaded / xhr.total * 100) + '% loaded');
}

function onModelLoadError(error) {
// console.log("Model error: " + error);
}

// instantiate a loader
const model_loader = new THREE.OBJLoader();
// load a resource
model_loader.load(
    'models/objs/capybara.obj',
    function ( obj ) {
        obj.position.set(0, 1, 0)
        obj.rotation.x = Math.PI * 0.5;
        obj.rotation.y = Math.PI * 1;
        obj.rotation.z = Math.PI * 1;
        obj.scale.multiplyScalar(.05);
        group2.add(obj)
        capybara = obj;
    },onModelLoadProgress,onModelLoadError
);
```

Carga de las utilidades de audio

```
// Music
audio_listener = new THREE.AudioListener();
camera.add( audio_listener );
sound = new THREE.Audio( audio_listener );
audio_analyser = new THREE.AudioAnalyser( sound, 32 );
audio_loader = new THREE.AudioLoader();
audio_loader.load( 'audio/'+songs[0]+'.mp3', function( buffer ) {
    sound.setBuffer( buffer );
    sound.setLoop(true);
    sound.setVolume(0.5);
});
```

Creación y manejo del GUI de la página

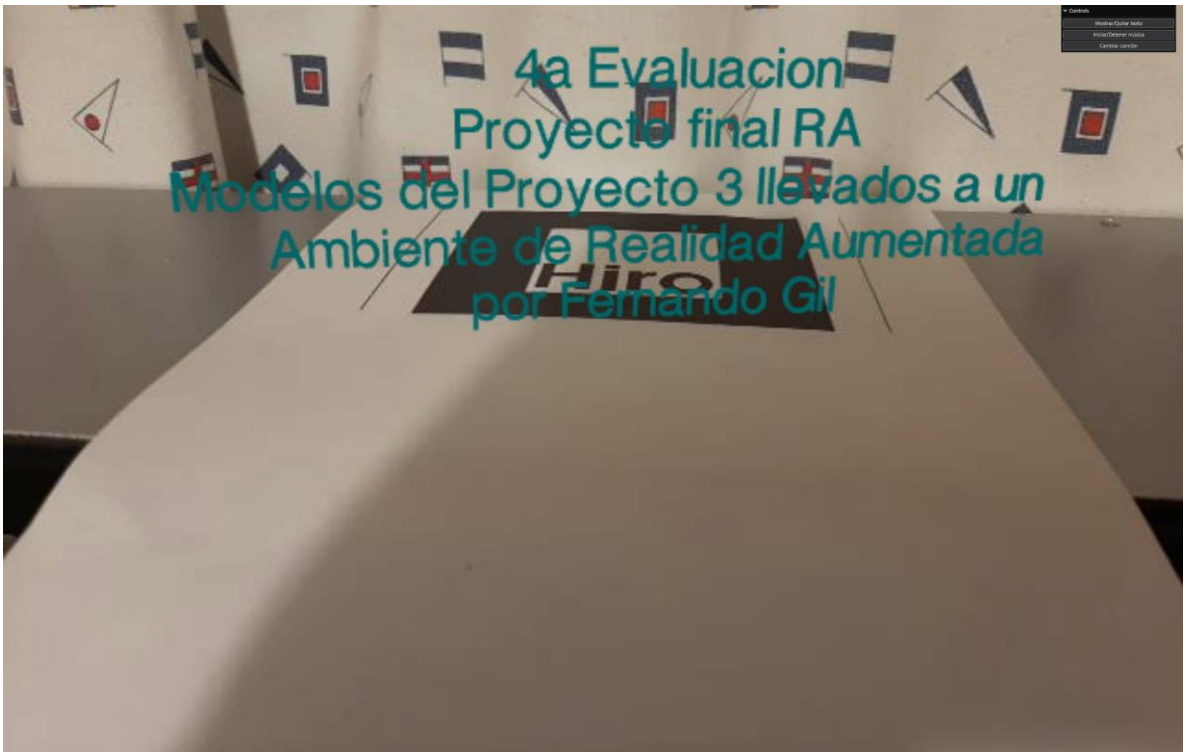
```
});
// Funciones para el GUI que muestran a cada planeta.
const params = {
    Credits: function () {
        group1.visible = !group1.visible
        group2.visible = !group2.visible
    },
    PlayPause: function () {
        if(sound.isPlaying) sound.pause();
        else sound.play();
    },
    ChangeSong: function () {
        songIndex = songIndex + 1 > songs.length-1 ? 0 : songIndex+1;
        sound.stop()
        audio_loader.load('audio/'+songs[songIndex]+'.mp3', function( buffer ) {
            sound.setBuffer( buffer );
            sound.setLoop(true);
            sound.setVolume(0.5);
            sound.play();
        });
    },
};
const gui = new GUI();
gui.add( params, 'Credits' ).name( 'Mostrar/Quitar texto' );
gui.add( params, 'PlayPause' ).name( 'Iniciar/Detener música' );
gui.add( params, 'ChangeSong' ).name( 'Cambiar canción' );
gui.open();
```

Todas las animaciones del proyecto

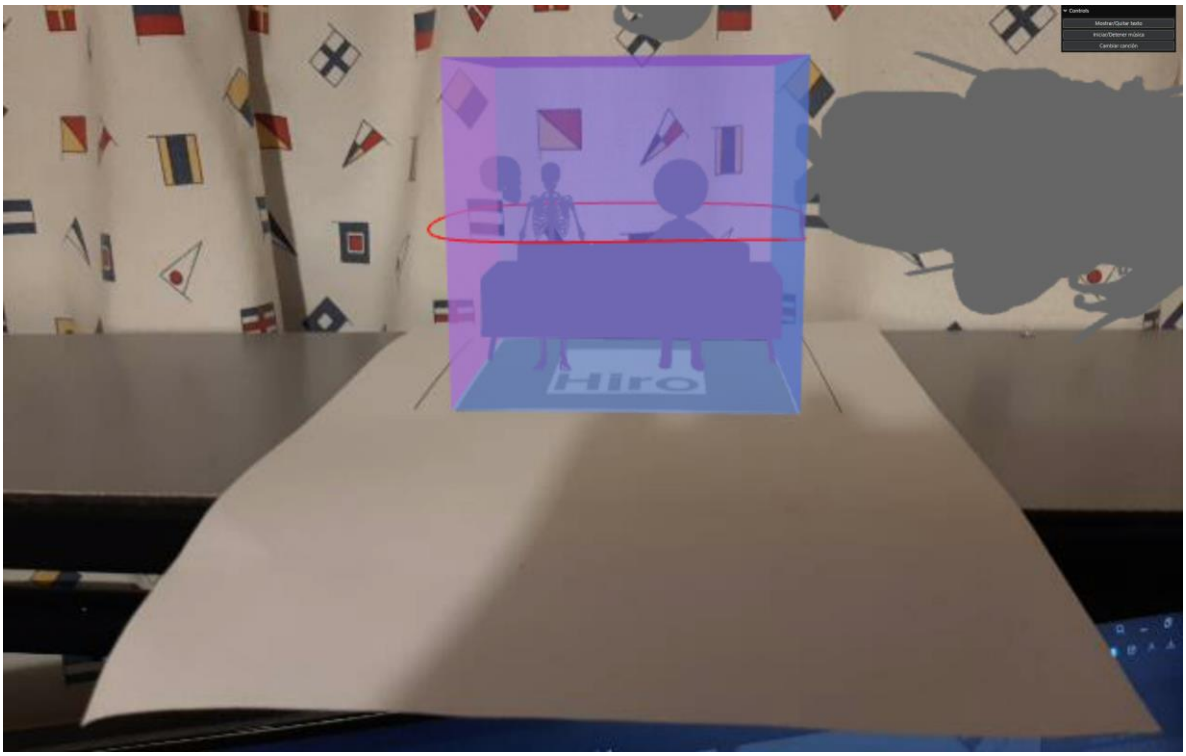
```
function update()
{
    time = .00025* performance.now();
    t = time % 1;
    varsize = Math.abs(Math.cos(time*10/2) * 150)
    if(group2.visible){
        if(cat){
            cat.position.x = -Math.cos(time*10/2) * animationRadius;
            cat.position.z = Math.sin(time*10/2) * animationRadius;
            cat.rotation.y = Math.PI * (time*1.5);
            cat.rotation.z = Math.PI * (time*2);
            cat.rotation.x = Math.PI * (time*3);
            cat.scale.multiplyScalar((Math.cos(time*10)>0 ? 1.01 : 0.99))
        }
        if(capybara){
            capybara.position.y = -Math.cos(time*10/2+20) * animationRadius;
            capybara.position.z = Math.sin(time*10/2+20) * animationRadius;
            capybara.rotation.y = Math.PI * (time*1.5);
            capybara.rotation.z = Math.PI * (time*2);
            capybara.rotation.x = Math.PI * (time*3);
            capybara.scale.multiplyScalar((-Math.cos(time*10)>0 ? 1.01 : 0.99))
        }
        if(skull1 && audio_analyser){
            if(curveInd > curvePoints.length-1)  curveInd = 0
            const freq = audio_analyser.getAverageFrequency()
            const pointOnCurve = curvePoints[curveInd];
            skull1.position.x = pointOnCurve.x
            skull1.position.z = pointOnCurve.z
            const setscale = freq ? freq/1000 : 0.025;
            skull1.scale.set(setscale,setscale,setscale)
            skull1.position.y = pointOnCurve.y + freq/200;
            currentRot +=1
            if(currentRot == rotationDelay) {
                currentRot = 0
                curveInd += 1
            }
            skull1.rotation.y = Math.PI * (time*1.5);

        }
        if(spaceship){
            spaceship.position.y = (Math.cos(time*10)>0 ?
            spaceship.position.y + spaceShipDeltaY : spaceship.position.y-spaceShipDeltaY)
            spaceship.rotation.x = -Math.PI * (Math.cos(time*10)*.01);
        }
    }
    if(group1.visible){
        if(textMesh){
            textMesh.position.y = Math.cos(time*10/2) * 0.2;
            textMesh.rotation.y = Math.cos(time*10/4) * 0.2;
        }
    }
    iter++
    // update artoolkit on every frame
    if ( arToolkitSource.ready !== false )
        arToolkitContext.update( arToolkitSource.domElement );
}
```

Resultados

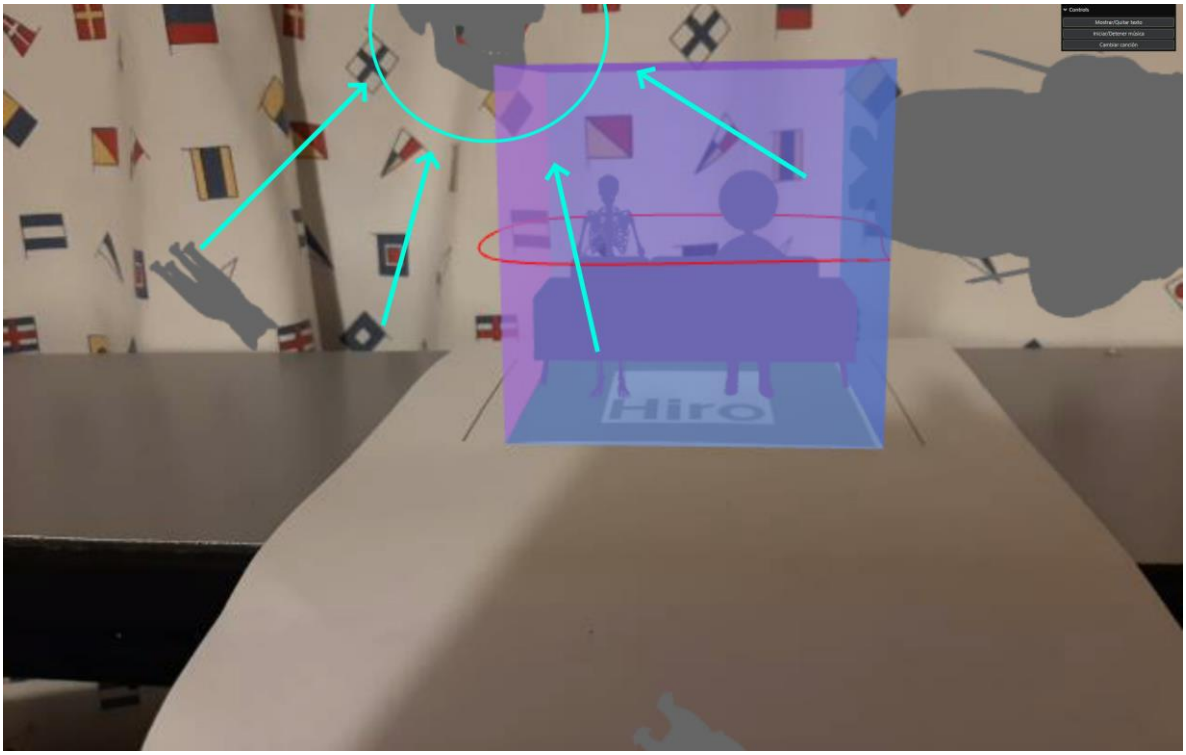
Texto inicial



Escena después de seleccionar la opción de quitar el texto



Escena con música activada relacionando las frecuencias a la animación de un objeto.



Conclusión

En conclusión, el desarrollo de este proyecto de Realidad Aumentada permitió explorar y comprender las posibilidades de ThreeJS en la carga y manipulación de modelos tridimensionales, así como la integración con AR.js para crear una experiencia interactiva en un plano de realidad aumentada. Aunque se presentaron algunos desafíos en la manipulación y escalado de los modelos, la biblioteca de ThreeJS demostró ser útil en la administración de estos. Además, la integración de la animación de los objetos con la frecuencia promedio de la música que reproduce la página resultó en una experiencia más inmersiva. En cuanto a la función de análisis de audio, aunque puede no tener un uso práctico en un proyecto formal, puede tener un potencial en animaciones similares que dependen del audio para su funcionamiento. En general, el proyecto permitió aprender y experimentar con las posibilidades de ThreeJS y AR.js en la creación de experiencias interactivas y tridimensionales.

Referencias

https://github.com/stemkoski/AR-Examples

https://stemkoski.github.io/AR-Examples/

https://threejs.org/docs/index.html?q=textgeo#examples/en/geometries/TextGeometry.parameters

https://github.com/stemkoski/AR-Examples/blob/master/hello-cube.html