

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**Інститут КНІТ
Кафедра ПЗ**

ЗВІТ

До лабораторної роботи № 2

З дисципліни: *“Віртуальна реальність”*

На тему: *“Імплементация тесту зіткнень API WEBXR”*

Лектор:

асист. каф. ПЗ
Задорожний І.М.

Виконав:

ст. гр. ПЗ-43
Лесневич Є.Є.

Прийняв:

асист. каф. ПЗ
Бауск О.Є.

« ____ » _____ 2025 р.

Σ= ____

Тема роботи: Імплементация тесту зіткнень (hit test) у WebXR додатку.

Мета роботи: Розширити функціональність WebXR додатку, створеного в попередній лабораторній роботі, додавши можливість тесту зіткнень для розміщення 3D об'єктів на реальних поверхнях.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Тест зіткнень (Hit Test) у WebXR

Тест зіткнень (hit test) у контексті доповненої реальності - це процес визначення, де промінь, що виходить з певної точки у певному напрямку, перетинається з реальними об'єктами у фізичному світі. У WebXR API цей функціонал дозволяє визначити, де віртуальні об'єкти можуть бути розміщені на реальних поверхнях.

Основні компоненти тесту зіткнень у WebXR:

1. `XRSession.requestHitTestSource()` - метод для створення джерела тесту зіткнень
2. `XRHitTestSource` - об'єкт, що представляє джерело тесту зіткнень
3. `XRFrame.getHitTestResults()` - метод для отримання результатів тесту зіткнень
4. `XRHitTestResult` - об'єкт, що містить інформацію про результат тесту зіткнень

Принцип роботи тесту зіткнень

1. Створюється джерело тесту зіткнень, яке визначає, звідки буде виходити промінь
2. У кожному кадрі анімації отримуються результати тесту зіткнень
3. Результати містять інформацію про точки перетину променя з реальними поверхнями
4. На основі цієї інформації можна розміщувати віртуальні об'єкти на реальних поверхнях

Типи джерел тесту зіткнень

WebXR підтримує два типи джерел тесту зіткнень:

1. Транзйєнтне джерело (Transient Hit Test Source) - промйнь виходить з певної точки у просторй вйдстеження, наприклад, з центру екрану
2. Постййне джерело (Persistent Hit Test Source) - промйнь виходить з певної точки у просторй вйдлйку, наприклад, з контролера// додайте: У цйй лабораторній роботй ми будемо використовувати транзйєнтне джерело тесту зіткнень, щоб визначити, де користувач "вказує" на реальнй поверхнй.

Реалйзаця тесту зіткнень у WebXR

Для реалйзацйй тесту зіткнень у WebXR необхідно:

1. Перевйрити пйдтримку функцйоналу тесту зіткнень у браузерй
2. Запросити необхіднй функцйй при створеннй сесйй WebXR
3. Створити джерело тесту зіткнень
4. Обробляти результати тесту зіткнень у кожному кадрй анімачйй
5. Використовувати отриманй результати для розмйщення вйртуальних об'єктйв

ЗАВДАННЯ

Розширити функцйональнйсть WebXR додатку, створеного в попереднйй лабораторній роботй, додавши можливйсть тесту зіткнень для розмйщення 3D об'єктйв на реальних поверхнях.

ХЙД ВИКОНАННЯ

Вмйст файлу package.json

```
{  
  "name": "webxr-project",  
  "version": "1.0.0",  
  "description": "WebXR demo project",  
  "scripts": {
```

```

    "start": "concurrently \"pnpm build --watch=forever\" \"pnpm exec
serve .\"",
    "build": "esbuild main.ts --bundle --outfile=dist/main.js --
format=esm --platform=browser --external:three/examples/jsm/loaders/GLTFLoader.js"
  },
  "devDependencies": {
    "@types/three": "^0.172.0",
    "@types/webxr": "^0.5.10",
    "concurrently": "^9.1.2",
    "esbuild": "^0.20.2",
    "serve": "^14.2.4",
    "typescript": "^5.3.3"
  },
  "packageManager":
"pnpm@10.5.2+sha512.da9dc28cd3ff40d0592188235ab25d3202add8a207afbedc682220e4a0029f
fbff4562102b9e6e46b4e3f9e8bd53e6d05de48544b0c57d4b0179e22c76d1199b",
  "dependencies": {
    "three": "^0.172.0"
  }
}

```

Вміст файлу main.ts

```

import * as THREE from 'three';
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader.js';
import type { GLTF } from 'three/examples/jsm/loaders/GLTFLoader.js';

const MODE = 'immersive-ar';

async function activateXR(): Promise<void> {
  const canvas = document.createElement("canvas");
  document.body.appendChild(canvas);

  const gl = canvas.getContext("webgl2", {xrCompatible: true});
  if (!gl) throw new Error("WebGL not supported");

  // FIX THIS:
  const scene = new THREE.Scene();

  const redMaterial = new THREE.MeshBasicMaterial({ color: 0xff0000 }); //
red for bottom face
  const greenMaterial = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
// green for top face
  const blueMaterial = new THREE.MeshBasicMaterial({ color: 0x0000ff });
// blue for other faces

  // initialize materials
  const materials = [
    blueMaterial, // front face
    blueMaterial, // back face
    greenMaterial, // top face
    redMaterial,  // bottom face
    blueMaterial, // left face

```

```

        blueMaterial // right face
    ];

    const cube = new THREE.Mesh(new THREE.BoxGeometry(0.5, 0.5, 0.5),
materials);
    // set cube position
    cube.position.set(1, 0, 1);
    // add cube to scene
    scene.add(cube);

    const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);
    scene.add(ambientLight);

    const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);
    directionalLight.position.set(10, 15, 10);
    scene.add(directionalLight);

    const renderer = new THREE.WebGLRenderer({
        alpha: true,
        preserveDrawingBuffer: true,
        canvas: canvas,
        context: gl
    });
    renderer.autoClear = false;

    // FIX THIS:
    const camera = new THREE.PerspectiveCamera();
    camera.matrixAutoUpdate = false;

    if (!navigator.xr) {
        throw new Error("WebXR is not supported by your browser");
    }

    try {
        const supported = await navigator.xr.isSessionSupported(MODE);
        if (!supported) {
            throw new Error(`${MODE} mode is not supported by your
browser/device`);
        }
    } catch (e) {
        throw new Error('Error checking WebXR support: ' + e);
    }

    const session = await navigator.xr.requestSession(
        MODE,
        {
            requiredFeatures: ['local'],
            optionalFeatures: ['hit-test']
        }
    );

    // FIX THIS:
    const baseLayer = new XRWebGLLayer(session, gl);
    session.updateRenderState({
        baseLayer
    });

    const referenceSpaceTypes: XRReferenceSpaceType[] = [
        'local'
    ];

    let referenceSpace: XRReferenceSpace | null = null;
    let hitTestSource: XRHitTestSource | undefined = undefined;

```

```

    // observe how reference space types and request reference space
    // are applied to the scene
    for (const spaceType of referenceSpaceTypes) {
        try {
            referenceSpace = await session.requestReferenceSpace(spaceType);
            const viewerSpace = await
session.requestReferenceSpace('viewer');
            if (session.requestHitTestSource) {
                hitTestSource = await session.requestHitTestSource({ space:
viewerSpace });
            }
            console.log('Reference space established:', spaceType);
            break;
        } catch(e) {
            console.log(e);
            console.log('Reference space failed:', spaceType);
            continue;
        }
    }

    if (!referenceSpace) {
        throw new Error('No reference space could be established');
    }

    const loader = new GLTFLoader();
    let reticle: THREE.Group;
    loader.load(
        "https://immersive-web.github.io/webxr-
samples/media/gltf/reticle/reticle.gltf",
        (gltf: GLTF) => {
            reticle = gltf.scene;
            reticle.visible = false;
            scene.add(reticle);
        }
    );

    session.addEventListener("select", (event) => {
        if (flower) {
            const clone = flower.clone();
            clone.position.copy(reticle.position);
            scene.add(clone);
        }
    });

    let flower: any;
    loader.load(
        "https://immersive-web.github.io/webxr-
samples/media/gltf/sunflower/sunflower.gltf",
        (gltf) => {
            flower = gltf.scene;
        }
    );

    // Create a render loop that allows us to draw on the AR view.
    const onXRFrame = (time: number, frame: XRFrame) => {
        // Queue up the next draw request.
        session.requestAnimationFrame(onXRFrame);

        const baseLayer = session.renderState.baseLayer;
        if (!baseLayer) return;

        // Bind the framebuffer and clear it

```

```

gl.bindFramebuffer(gl.FRAMEBUFFER, baseLayer.framebuffer);
gl.clearColor(0, 0, 0, 0);
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

const pose = frame.getViewerPose(referenceSpace);
if (pose) {
  const view = pose.views[0];
  const viewport = baseLayer.getViewport(view);
  if (!viewport) return;
  renderer.setSize(viewport.width, viewport.height);

  if (!hitTestSource) return;
  const hitTestResults = frame.getHitTestResults(hitTestSource);
  if (hitTestResults.length > 0 && reticle) {
    const hitPose = hitTestResults[0].getPose(referenceSpace);
    if (!hitPose) return;
    reticle.visible = true;
    reticle.position.set(hitPose.transform.position.x,
hitPose.transform.position.y, hitPose.transform.position.z);
    reticle.updateMatrixWorld(true);
  }

  // Update the camera with the XR view's transform and projection
  camera.matrix.fromArray(view.transform.matrix);
  camera.projectionMatrix.fromArray(view.projectionMatrix);
  camera.updateMatrixWorld(true);

  // Render the scene into the cleared framebuffer
  renderer.render(scene, camera);
}
};

session.requestAnimationFrame(onXRFrame);
}

// Make the function available globally
(window as any).activateXR = activateXR;

```

РЕЗУЛЬТАТИ



Рис. 1. Результат виконання програми – відмальований на сцені приціл



Рис. 2. Результат виконання програми – відмальований на координатах прицілу соняшник



Рис. 3. Результат виконання програми – вигляд соняшника з іншого боку



Рис. 4. Результат виконання програми – відмальовані соняшники на різних поверхнях

ВИСНОВКИ

Отже, під час виконання даної лабораторної роботи було розширено функціональність WebXR додатку, створеного в попередній лабораторній роботі, додавши можливість тесту зіткнень для розміщення 3D об'єктів на реальних поверхнях. Імплементовано тест зіткнень (hit test) у WebXR додатку.