```
<!DOCTYPE html>

<head>
  <meta name="viewport" content="width=device-width, user-scalable=no,
minimum-scale=1.0, maximum-scale=1.0">
  <title>AR project template</title>
  <!-- Додаємо необхідні бібліотеки -->
  <script src="js/three.js"></script>
  <script src="js/tween.umd.js"></script>
  <script src='loaders/GLTFLoader.js'></script>
  <script src='loaders/GLTF2Loader.js'></script>
  <script src='loaders/MTLLoader.js'></script>
  <script src='loaders/OBJLoader.js'></script>
  <!-- Додаємо jsartookit -->
  <script src="jsartoolkit5/artoolkit.min.js"></script>
  <script src="jsartoolkit5/artoolkit.api.js"></script>
  <!-- Додаємо threex.artoolkit -->
  <script src="threex/threex-artoolkitsource.js"></script>
  <script src="threex/threex-artoolkitcontext.js"></script>
  <script src="threex/threex-arbasecontrols.js"></script>
  <script src="threex/threex-armarkercontrols.js"></script>
</head>

<body style='margin : 0px; overflow: hidden; font-family: Monospace;
user-select: none; pointer-events: none;'>

  <div id="access" style="top: 0; left: 0; right:0; bottom: 0; background:
#000; position: absolute; user-select: all; pointer-events: all;">
    <div id="text-wrapper" style="top: 50%; left: 50%; position: absolute;
color: #fff; transform: translate(-50%, -50%); text-align: center;
        text-transform: uppercase; font-family: Arial, Helvetica, sans-serif;
font-weight: 400; line-height: 1.5em; font-size: large; white-space: nowrap;
        user-select: none; pointer-events: none;
      ">
      Press here
      <br>
      to enter the experience
    </div>
  </div>
```

```
  <div id="loader" style="position: absolute; top: 0; left: 0; right: 0; bottom:
0; background: #fff; pointer-events: none; user-select: none;
      transition: all .2s linear; display: none;">
      <div style="left: 50%; top: 50%; position: absolute; transform:
translate(-50%, -50%); text-align: center; width: 130px; height: 165px;
          font-family: Arial, Helvetica, sans-serif; font-weight: 400; line-height:
1.5em; font-size: large;" class="spinner-wrapper">
          <img style="width: 130px; height: 130px; pointer-events: none;
user-select: none;" src="data/spin.gif" alt="spin gif">
          <br>
          Loading...
      </div>
  </div>

  <script>
      const access = document.getElementById('access');
      const loader = document.getElementById('loader');

      function initiateExperience() {
          // Оголошуємо глобальні змінні
          var scene, camera, renderer, clock, deltaTime, totalTime;

          var patternIdOffset = 10000000000

          // Змінні необхідні для роботи AR оточення
          var arToolkitSource, arToolkitContext;

          // Головний контейнер, до якого увійдуть всі 3D об'єкти для
програми
          var markerRoot, mainContainer;

          // Окремий масив для зберігання всього аудіо та відео контенту,
який буде
          // запущений натисканням на екран смартфона
          var audioContent = [];

          var contentPromises = [];

          let contentInitialized = false;
```

```
        let barcodesSound = new Map();
        let patternsSound = new Map();
        let barcodesID = [];
        let patternsID = [];

        let controller;

        // Ініціалізуємо сцену та запускаємо цикл анімації
        initialize();
        animate();

        function initialize() {
            // Оголошуємо сцену, в яку додамо головний контейнер з усіма
3D об'єктами.
            scene = new THREE.Scene();

            // Додаємо світло на сцену, інакше базові матеріали будуть
просто чорними.
            // т.к. їм нема чого відображати, зверніться до документації
бібліотеки three.js, щоб
            // прочитати про докладну роботу класу Material
            let ambientLight = new THREE.AmbientLight(0xffffff, 0.75);
            scene.add(ambientLight);

            // Додаємо камеру, яка буде пізніше перепризначена на
камеру смартфона
            camera = new THREE.Camera();
            scene.add(camera);
            const listener = new THREE.AudioListener();
            camera.add(listener);
            const audioLoader = new THREE.AudioLoader();

            // Оголошуємо стандартний рендерер і додаємо його до тега
body html документа
            renderer = new THREE.WebGLRenderer({
                antialias: true,
                alpha: true
            });
            renderer.setClearColor(new THREE.Color('lightgrey'), 0)
            renderer.setSize(640, 480);
```

```
        renderer.domElement.style.position = 'absolute'
        renderer.domElement.style.top = '0px'
        renderer.domElement.style.left = '0px'
        document.body.appendChild(renderer.domElement);

        clock = new THREE.Clock();
        deltaTime = 0;
        totalTime = 0;

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

        // Функція перерендерує AR сцену під поточний розмір canvas
        function onResize() {
            arToolkitSource.onResize()
            arToolkitSource.copySizeTo(renderer.domElement)
            if (arToolkitContext.arController !== null) {

arToolkitSource.copySizeTo(arToolkitContext.arController.canvas)
            }
        }

        // Викликаємо функцію під час ініціалізації AR
        arToolkitSource.init(function onReady() {
            onResize()
        });

        // Викликаємо функцію на resize івент веб-сторінки
        window.addEventListener('resize', function () {
            onResize()
        });

        // Ініціалізуємо AR контекст під камеру, патерни, баркод 3x3
        arToolkitContext = new THREEx.ArToolkitContext({
            cameraParametersUrl: 'data/camera_para.dat',
            detectionMode: 'mono_and_matrix',
            matrixCodeType: "3x3",
            maxDetectionRate: 60,
            canvasWidth: 640,
```

```
            canvasHeight: 480
        });

        // Відновлюємо матрицю проекції камери після закінчення ініціалізації
        arToolkitContext.init(function onCompleted() {

camera.projectionMatrix.copy(arToolkitContext.getProjectionMatrix());
        });

        // Створюємо головну групу для всіх 3D об'єктів
        mainContainer = new THREE.Group();

        // Масив назв файлів .patt. Масив заповнюється в порядку додавання маркерів
        // якщо замість .patt було додано баркод, на його місце в масив додається порожній рядок
        const patternNames = ["" ,"" ,""];
        // Масив баркодів, заповнюється одночасно з масивом патернів
        // якщо замість баркоду був доданий .patt, на його місце масив додається -1
        const patternBarcode = [1 ,2 ,3];
        // Масив типів контенту кожного маркера, заповнюється значеннями: зображення, модель, відео
        const modes = ["image" ,"image" ,"image"];
        // Масив файлів моделей, якщо немає моделі буде додано порожній рядок
        const modelFiles = ["" ,"" ,""];
        // Масив файлів зображень, якщо немає зображення буде додано також порожній рядок
        const imageFiles = ["файл 1.jpg" ,"файл 2.jpg" ,"файл 3.png"];
        // Масив файлів відео, якщо немає відео буде ще один порожній рядок
        const videoFiles = ["" ,"" ,""];
        // Масив файлів аудіо, якщо немає аудіо буде так само порожній рядок
        const audioFiles = ["Файл 1.2.m4a" ,"Файл 2.2.m4a" ,"Файл 3.2.m4a"];
```

```
        // Масив опцій повтору аудіо та відео контенту, по дефолту для всіх буде false
        const repeatOptions = ["false" ,"false" ,"false"];

        // Створюємо масив для всіх маркерів
        const markerRoots = [];
        for (let i = 0; i < 3; i++) {
            markerRoots[i] = new THREE.Group();
        }

        // Проходимо по кожному маркеру з масиву і додаємо його в головний контейнер
        for (let i = 0; i < 3; i++) {
            mainContainer.add(markerRoots[i]);

            // Якщо поточний маркер – це баркод, створюємо AR контролер під баркод
            // якщо поточний маркер це патерн, аналогічно створюємо AR контролер під патерн
            if (patternBarcode[i] === -1) {
                let markerControls1 = new THREEx.ArMarkerControls(arToolkitContext, markerRoots[i], {
                    type: 'pattern', patternUrl: patternNames[i], size: 1 + (i + 1) / patternIdOffset
                })
                patternsID.push(patternNames[i]);
            } else {
                let markerControls1 = new THREEx.ArMarkerControls(arToolkitContext, markerRoots[i], {
                    type: "barcode", barcodeValue: patternBarcode[i],
                })
                barcodesID.push(patternBarcode[i]);
            }

            // Використовуємо switch для роботи з кожним окремим випадком контенту
            switch (modes[i]) {
                // Якщо контент під маркер це модель
                case 'model':
```

```
                    function onProgress(xhr) { console.log((xhr.loaded /
xhr.total * 100) + '% loaded'); }
                    function onError(xhr) { console.log('An error happened'); }

                    contentPromises.push(new Promise((resolve) => {
                        const test = new
THREE.GLTF2Loader().load(`${modelFiles[i]}`, (response) => {
                            const scene = response.scene;
                            const object = scene.children[0];
                            // Іноді модель не можна побачити з кількох
причин, найчастіше варто збільшити чи зменшити у 100 разів.
                            // Читайте:
https://threejs.org/docs/index.html#manual/en/introduction/Loading-3D-mod
els
                            // Тут ми зменшуємо її, щоб точно побачити її на
сцені. Ви можете видалити цю шкалу, якщо потрібно
                            object.scale.set(0.01, 0.01, 0.01);
                            // Ви можете самостійно змінити поворот або
позицію моделі
                            // object.position.set(0, Math.PI / 2, Math.PI / 4);
                            // object.rotation.set(0, Math.PI / 2, Math.PI / 4);
                            // Додавання нашої моделі до контейнера групи
маркерів
                            markerRoots[i].add(object);
                            resolve(modelFiles[i])
                        }, onProgress, onError)
                    }).then((file) => {
                        console.log(`File ${file} loaded`)
                    }))
                    break;
                // Якщо контент під маркер це зображення
                case 'image':
                    if (imageFiles[i]) {
                        contentPromises.push(new Promise((resolve) => {
                            // Завантажуємо зображення
                            let loader = new THREE.TextureLoader();
                            loader.load(`${imageFiles[i]}`, (texture) => {
                                let geometry1, ratio = texture.image.naturalWidth /
texture.image.naturalHeight;
```

```
                            if (texture.image.naturalHeight <
texture.image.naturalWidth) {
                                geometry1 = new
THREE.PlaneBufferGeometry(ratio, 1);
                            } else {
                                geometry1 = new
THREE.PlaneBufferGeometry(1, 1 / ratio);
                            }
                            let material1 = new THREE.MeshBasicMaterial({
map: texture, side: THREE.DoubleSide });
                            mesh1 = new THREE.Mesh(geometry1,
material1);
                            // Повертаємо площину
                            mesh1.rotation.x = -Math.PI / 2;
                            // Додаємо площину у контейнер
                            markerRoots[i].add(mesh1);
                            resolve(imageFiles[i])
                        });
                    }).then(image => {
                        console.log(`File ${image} loaded`)
                    }))
                }
                break;
            // Якщо контент під маркер - це відео
            case 'video':
                // Оголошуємо площину під відео
                let geometry2 = new THREE.PlaneBufferGeometry(1.6 ,
0.9);
                // Оголошуємо та завантажуємо відео
                let video = document.createElement('video');
                video.src = `${videoFiles[i]}`;
                video.playsInline = true;
                // Встановлюємо відео на автоповтор залежно від
значення у масиві
                if (repeatOptions[i]) {
                    video.addEventListener('ended', () => {
                        video.play();
                    })
                }
```

```
                    // Додаємо відео до масиву аудіо контенту
                    if (patternBarcode[i] === -1) {
                        patternsSound.set(i, video);
                    } else {
                        barcodesSound.set(patternBarcode[i], video);
                    }
                    // Перенаправляємо текстуру з відео на матеріал для
площини.
                    let texture2 = new THREE.VideoTexture(video);
                    texture2.minFilter = THREE.LinearFilter;
                    texture2.magFilter = THREE.LinearFilter;
                    texture2.format = THREE.RGBFormat;
                    let material2 = new THREE.MeshBasicMaterial({ map:
texture2 });
                    mesh2 = new THREE.Mesh(geometry2, material2);
                    // Повертаємо площину
                    mesh2.rotation.x = -Math.PI / 2;
                    // Додаємо площину у контейнер
                    markerRoots[i].add(mesh2);
                    break;
                case 'controller':
                    controller = new THREE.Mesh(
                        new THREE.CubeGeometry(10, 0.15, 0.15),
                        new THREE.MeshBasicMaterial({ color: 'green' })
                    );
                    controller.rotation.y = Math.PI / 2;
                    controller.position.y = 0.125;
                    controller.position.z = -4.5;
                    markerRoots[i].add(controller);
                    break;
                default:
                    // Якщо жодного контенту не додано, додаємо білу
площину.
                    mesh11 = new THREE.Mesh(new
THREE.PlaneBufferGeometry(1, 1),
                        new THREE.MeshBasicMaterial({ color: '#fff' }));
                    // Повертаємо площину
                    mesh11.rotation.x = -Math.PI / 2;
                    // Додаємо площину у контейнер
```

```
                markerRoots[i].add(mesh11);
                break;
            }

            // Якщо є аудіо файли, налаштовуємо їх і додаємо в масив аудіо контенту.
            if (audioFiles[i]) {
                contentPromises.push(new Promise((resolve, reject) => {
                    audioLoader.load(`${audioFiles[i]}`, function (buffer) {
                        // Створюємо аудіо джерело
                        let sound = new THREE.Audio(listener);
                        sound.name = `${audioFiles[i]}`;
                        sound.setBuffer(buffer);
                        // Встановлюємо відео на автоповтор залежно від значення у масиві
                        if (repeatOptions[i]) {
                            sound.setLoop(true);
                        }
                        if (patternBarcode[i] === -1) {
                            patternsSound.set(i, sound);
                        } else {
                            barcodesSound.set(patternBarcode[i], sound);
                        }
                        resolve(sound)
                    });
                }).then((sound) => {
                    sound.play()
                    sound.stop()
                    console.log(`File ${sound.name} loaded`)
                }))
            }
        }

        // Ховаємо лоадер після завантаження компонентів
        Promise.all(contentPromises)
            .then(() => {
                console.log('Most of the content loaded')
                contentInitialized = true;
                loader.style.opacity = '0';
```

```
        });

    // Додаємо головний контейнер на сцену
    scene.add(mainContainer);
}

function checkController() {
    if (controller) {
        mainContainer.traverse((object) => {
            if (object.isMesh && object !== controller) {
                if (detectCollisionCubes(object, controller)) {
                    object.material.color.set('red')
                } else {
                    object.material.color.set('white')
                }
            }
        });
    }
}

// Функція пошуку перетинів між двома об'єктами сцени
function detectCollisionCubes(object1, object2) {
    object1.geometry.computeBoundingBox();
    object2.geometry.computeBoundingBox();
    object1.updateMatrixWorld();
    object2.updateMatrixWorld();

    const box1 = object1.geometry.boundingBox.clone();
    box1.applyMatrix4(object1.matrixWorld);

    const box2 = object2.geometry.boundingBox.clone();
    box2.applyMatrix4(object2.matrixWorld);

    return box1.intersectsBox(box2);
};

// Оновлюємо AR контент на кожен кадр
function update() {
    if (arToolkitSource.ready !== false) {
        arToolkitContext.update(arToolkitSource.domElement);
```

```
            if (contentInitialized) {
                if (barcodesID.length) {
                    barcodesID.forEach((elem, index) => {
                        if
(arToolkitContext.arController.barcodeMarkers[elem].inCurrent) {
                            let sound = barcodesSound.get(elem);
                            if (sound && !sound.isPlaying) sound.play();
                        } else {
                            let sound = barcodesSound.get(elem);
                            if (sound && sound.nodeName === 'VIDEO') {
                                if (!sound.paused) sound.pause()
                            }
                            if (sound && sound.isPlaying) sound.stop();
                        }
                    })
                }
                if (patternsID.length) {
                    for (let index = 0; index < patternsID.length; index++) {
                        if
(arToolkitContext.arController.patternMarkers[index].inCurrent) {
                            let patternID =
(arToolkitContext.arController.patternMarkers[index].markerWidth - 1) *
patternIdOffset - 1;
                            patternID = Math.round(patternID)
                            let sound = patternsSound.get(patternID);
                            if (sound && !sound.isPlaying) sound.play();
                        } else {
                            let patternID =
(arToolkitContext.arController.patternMarkers[index].markerWidth - 1) *
patternIdOffset - 1;
                            patternID = Math.round(patternID)
                            let sound = patternsSound.get(patternID);
                            if (sound && sound.nodeName === 'VIDEO') {
                                if (!sound.paused) sound.pause()
                            }
                            if (sound && sound.isPlaying) sound.stop();
                        }
                    }
                }
```

```
                }
            }
        }

        // Рендерім сцену на кожен кадр
        function render() {
            renderer.render(scene, camera);
        }

        // Запускаємо цикл анімації
        function animate(time) {
            // Прив'язуємо цикл анімації до рендеру браузера
            requestAnimationFrame(animate);
            deltaTime = clock.getDelta();
            totalTime += deltaTime;
            update();
            checkController();
            render();
        }
    }

    // Прибираємо блок після натискання на екран для дозволу аудіо програвання на iOS
    access.addEventListener('click', () => {
        initiateExperience();
        document.body.removeChild(access);
        loader.style.display = 'block';
    });
  </script>

</body>

</html>
```