Immersive Web: WebGL, WebVR and Beyond

Looking back at WebVR I realized how important it's to keep paying attention to technologies that you're interested in or they may change from under your feet.

VR (and now AR) is one of these areas.

What it is

WebXR is an evolutionary step currently under Origin Trials, a way to run experiments for a subset of developers using a feature. This allows for rapid iteration and quick feedback but without running the risk of the feature becoming a defacto standard, particularly when the API or feature is not finished.

In this context I'll use WebXR to mean the WebXR Device API.

WebXR lets you create AR and VR web sites by providing access to input and output capabilities of AR/VR hardware.

Examples of hardware devices include:

- Google's Daydream
- Oculus Rift
- Samsung Gear VR
- HTC Vive
- Windows Mixed Reality headsets

For more details, watch Brandon Jones' presentation at Google I/O this year for more details on WebXR.

How it works?

We'll look at two examples of what we can do with WebXR: Using Magic Windows to place 3D content inside a regular web page and a second example of how to create old-school WebVR content to use with VR devices.

For these demos we'll use Three.js to actually build the content.

Playing WebGL content

There shouldn't be any difference in playing with WebGL content in browsers that support WebGL and WebVR/WebXR. WebGL is not part of this trial and shouldn't affect the way WebGL content look on the browser.

Playing WebXR content from a browser

The next step is how to render content using one codebase that will work well with all possible uses for WebXR.

Mozilla has released an extension to Three.js called three.xr.js that makes it easier to work creating WebXR content.

The example below assumes that you've imported `three.js` and `three.xr.js`.

What I like is that, while there's a lot of code, even in this simple example, it's mostly boilerplate for `three.xr` and can be abstracted to a separate file to be reused later.

This is the basic example from `three.xr.js` that will display the basic WebGL scene and detect if there is a VR or AR device available and, if there is one, present buttons to enter the Mixed Reality mode that the device supports.

```
var clock = new THREE.Clock();
var container;
var renderer, camera, scene;
var floor, cube, sky;

// ar, magicWindow, vr
var activeRealityType = 'magicWindow';
'magicWindow'// Query for available displays then run init
THREE.WebXRUtils.getDisplays().then(init);

function init(displays) {
  // Create a container div and append it to body= document.createElement(
  document.body.appendChild( container );

  // Create thr'div'// Create three.js scene and camera
  scene = new THREE.Scene();
  camera = new THREE.PerspectiveCamera();
  scene.add( camera );
  camera.position.set(0, 1, 0);
  renderer = new THREE.WebGLRenderer( { alpha: true } );
  renderer.autoClear = false;
  container.appendChild( renderer.domElement );

  // Add custom content
  // Define the floor
  floor =   new THREE.Mesh(new THREE.PlaneGeometry(8, 8),
            new THREE.MeshBasicMaterial({
              color:0xffffff,
              transparent: true,
              opacity: 0.3
            }));
```

```
floor.geometry.rotateX(- Math.PI / 2);
scene.add (floor);

// Define a cube
var geometry = new THREE.BoxGeometry(0.5,0.5,0.5);
var material = new THREE.MeshNormalMaterial();
cube = new THREE.Mesh(geometry, material);
cube.position.set(0, 1, -1.5);
cube.rotation.y = Math.PI/4;
scene.add (cube);

// Define the sky
var skyGeometry = new THREE.SphereGeometry(5);
var skyMaterial = new THREE.MeshNormalMaterial({
  side: THREE.BackSide
});
sky = new THREE.Mesh(skyGeometry, skyMaterial);
scene.add (sky);
// End custom content

// Define standard evnts and elementsListener( 'resize', onWindowResize
onWindowResize();
var options = {
  // Flag to start A'resize'// Flag to start AR if is the unique displa
  AR_AUTOSTART: false, // Default: trueHREE.WebXRManager(options, displ
renderer.xr.addEventListener('sessionStarted', sessionStarted);
renderer.xr.addEventListener('sessionEnded', sessionEnded);

if(!renderer.xr.autoStarted){
  var buttonsContainer = document.createElement( 'div' );
  buttonsContainer.id = 'buttonsContainer';
  buttonsContainer.style.position = 'absolute';
  buttonsContainer.style.bottom = '10%';
  buttonsContainer.style.width = '100%';
  buttonsContainer.style.textAlign = 'center';
  buttonsContainer.style.zIndex = '999';
  document.body.appendChild(buttonsContainer);
```

```
    addEnterButtons(displays);
  }
  renderer.animate(render);
}

function sessionStarted(data) {
  activeRealityType = data.session.realityType;
  if(data.session.realityType === 'ar'){
    sky.visible = false;
  }
}

function sessionEnded(data) {
  activeRealityType = 'magicWindow';
  if(data.session.realityType === 'ar'){
    sky.visible = true;
  }
}

function addEnterButtons(displays) {
  for (var i = 0; i < displays.length; i++) {
    var display = displays[i];
    if(display.supportedRealities.vr){
      buttonsContainer.appendChild(getEnterButton(display, 'vr'));
    }
    if(display.supportedRealities.ar){
      buttonsContainer.appendChild(getEnterButton(display, 'ar'));
    }
  }
}

function getEnterButton(display, reality) {
  // HMDs require the call t'sessionStarted'// HMDs require the call to st
  // occur due to a user input event, so make a
  // button to trigger thatutton' );
  button.style.display = 'inline-block';
  button.style.margin = '5px';
  button.style.width = '120px';
```

```
  button.style.border = '0';
  button.style.padding = '8px';
  button.style.cursor = 'pointer';
  button.style.backgroundColor = '#000';
  button.style.color = '#fff';
  button.style.fontFamily = 'sans-serif';
  button.style.fontSize = '13px';
  button.style.fontStyle = 'normal';
  button.style.textAlign = 'center';
  if(reality === 'vr'){
    button.textContent = 'ENTER VR';
  }else{
    button.textContent = 'ENTER AR';
  }

  button.addEventListener('click', ev => {
    if(reality === 'ar'){
      if(!renderer.xr.sessionActive){
        // Entering AR.
        button.textContent'inline-block'// Entering AR.r.xr.startSession(
      } else {
        // Exiting AR.
        button.textContent = 'ENTER AR';
'ENTER AR'// Exiting AR.ession();
        sky.visible = true;
      }
    } else {
      document.getElementById('buttonsContainer').style.display = 'none';
      renderer.xr.startPresenting();
    }
  });

  if(reality === 'vr'){
    window.addEventListener('vrdisplaypresentchange', (evt) => {
      // Polyfill places cameraActivateddisplay inside the detail proper'
        // Exiting VR.
        renderer.xr.endSession();
        document.getElementById('buttonsContainer').style.display = 'bloc
```

```
      }
    });
  }
  return button;
}

function onWindowResize() {
  camera.aspect = window.innerWidth / window.innerHeight;
  camera.updateProjectionMatrix();
}

// Called once per frame, before render, to give the app a chance to updat
    case 'vr':
      var delta = clock.getDelta() * 60;
      // cube.rotation.y += delta * 0.01;
      break;
  }

  if(!renderer.xr.sessionActive){
    renderer.s'vr'// cube.rotation.y += delta * 0.01;// cube.rotation.y +=
      break;
  }

  if(!renderer.xr.sessionActive){
    renderer.setSize( window.innerWidth, window.innerHeight );
    renderer.render(scene, camera);
  } else {}
}
```

Future Looking: Magic Window, glipmses of VR in a 2D context

One of the really cool things you can do with WebXR is to create “Magic Windows”, areas where we can view 3D content inside our regular web page.

So far all the examples I’ve see of WebXR are either full screen 3D on-device experiences or experiences that require an AR-enabled device to work. That’s awesome but wouldn’t it be nice if we could launch the content from a regular web

page?

At Google I/O this year I saw such a demo.

It included an AR object inserted as an image or iframe in a page that would only go into AR mode when viewed in an AR-capable device and the user actually clicked on the image to activate the the placement of the AR object in the physical space.

I'm not 100% certain that this was WebXR or if it was more on the ARCore/ARKit side of the AR/VR experience. Since it was on a webpage I expect the earlier but there was a big push on AR for the Android and iOS platforms so I'm not sure.

I'll continue researching and will post more when I have configmration either way.

Links and examples

- Mozilla Mixed Reality Blog
- webxr examples
- three.xr.js
- Progressive WebXR