Nell Waliczek

Software Engineering Lead

TWITTER	@NellWaliczek
GITHUB	github.com/NellWaliczek

Lewis Weaver

Program Manager

TWITTER	@lew_weav
GITHUB	github.com/leweaver

#msedgesummit

Microsoft

Mixed Reality on the web using WebVR

#msedgesummit

Available October 17th

#msedgesummit

WebVR

“**WebVR** is an open **specification** that makes it possible to experience **VR in your browser**. The goal is to make it easier for **everyone** to get into VR experiences, no matter what device you have”

https://webvr.info

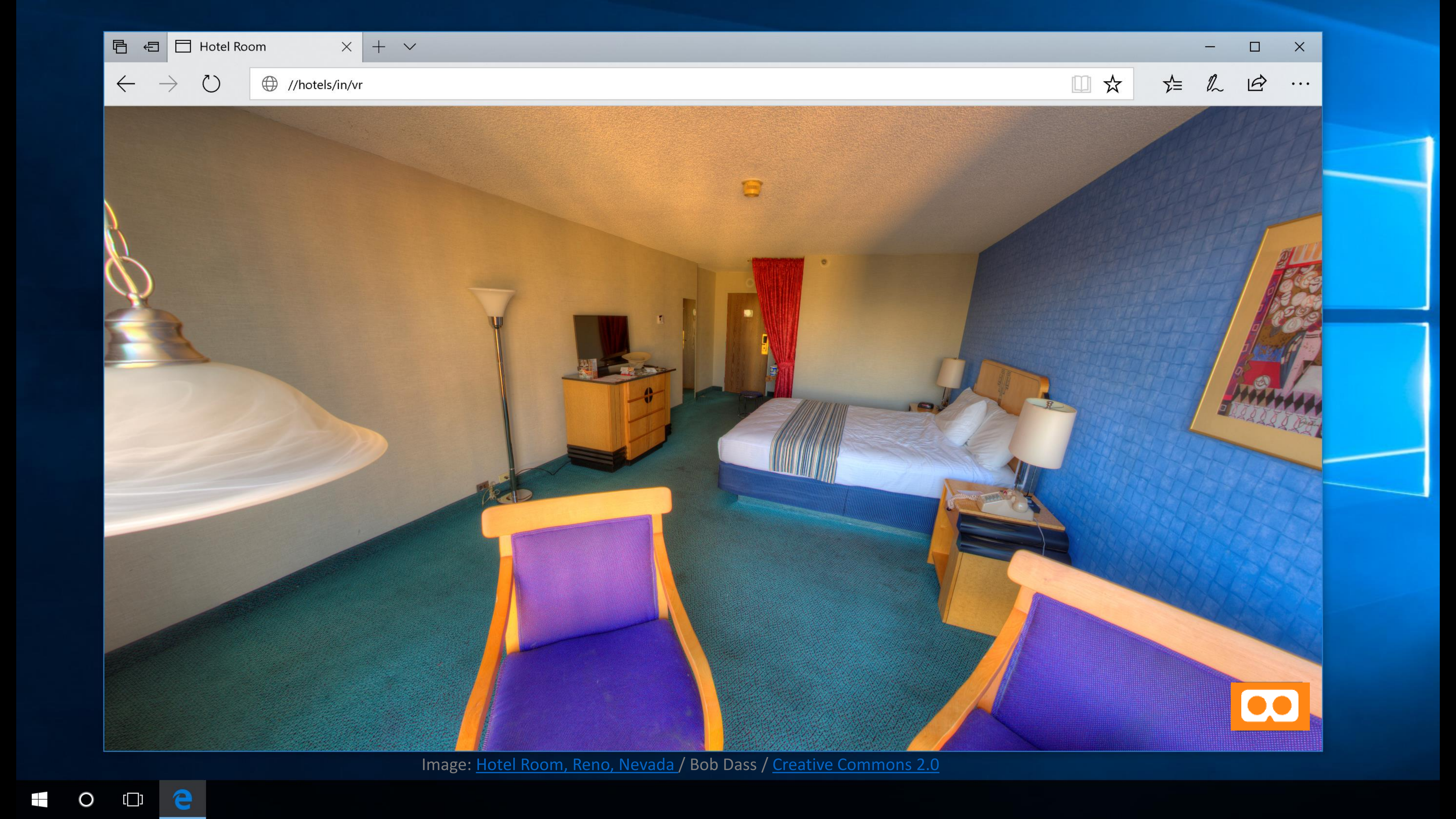
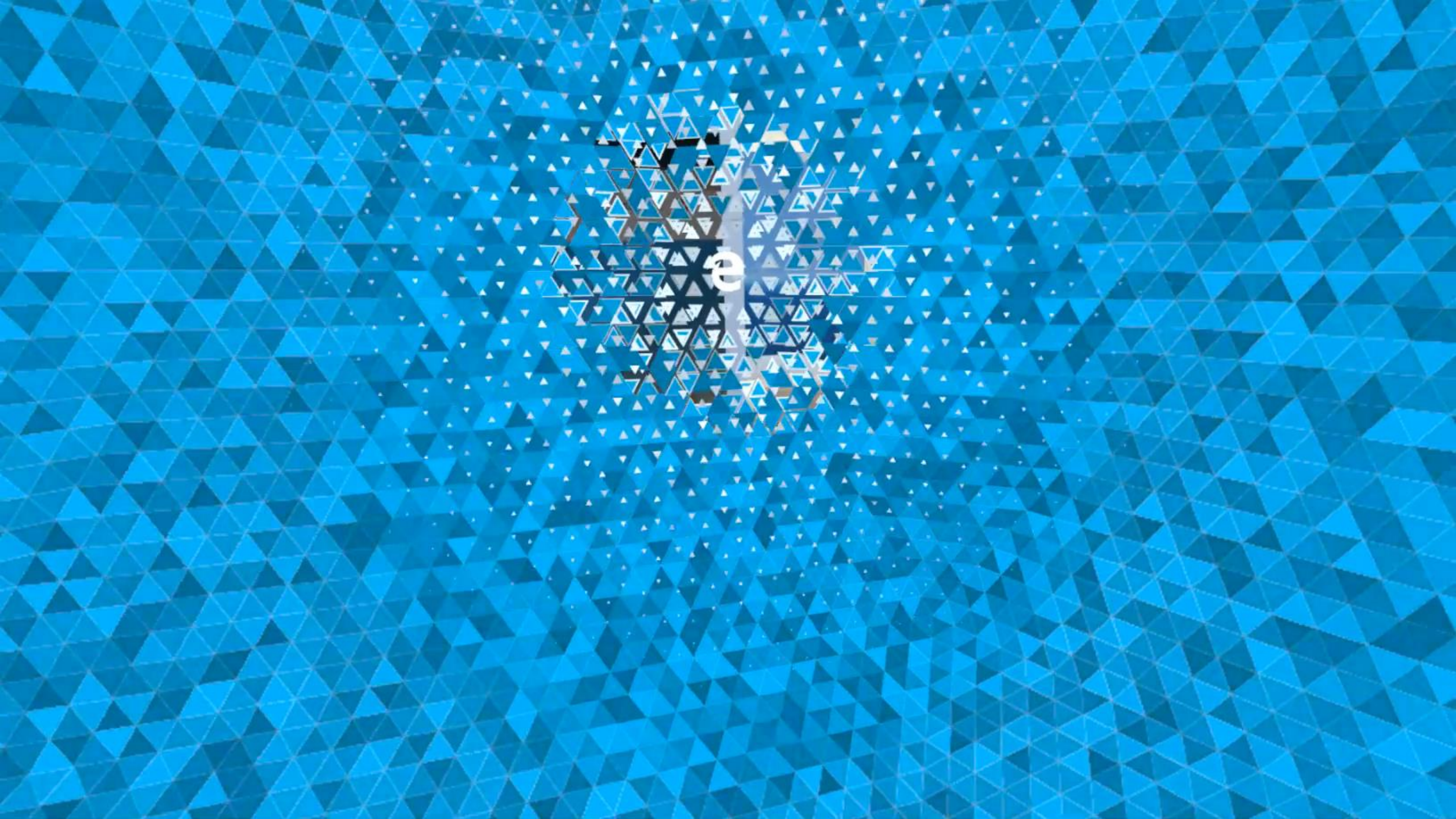
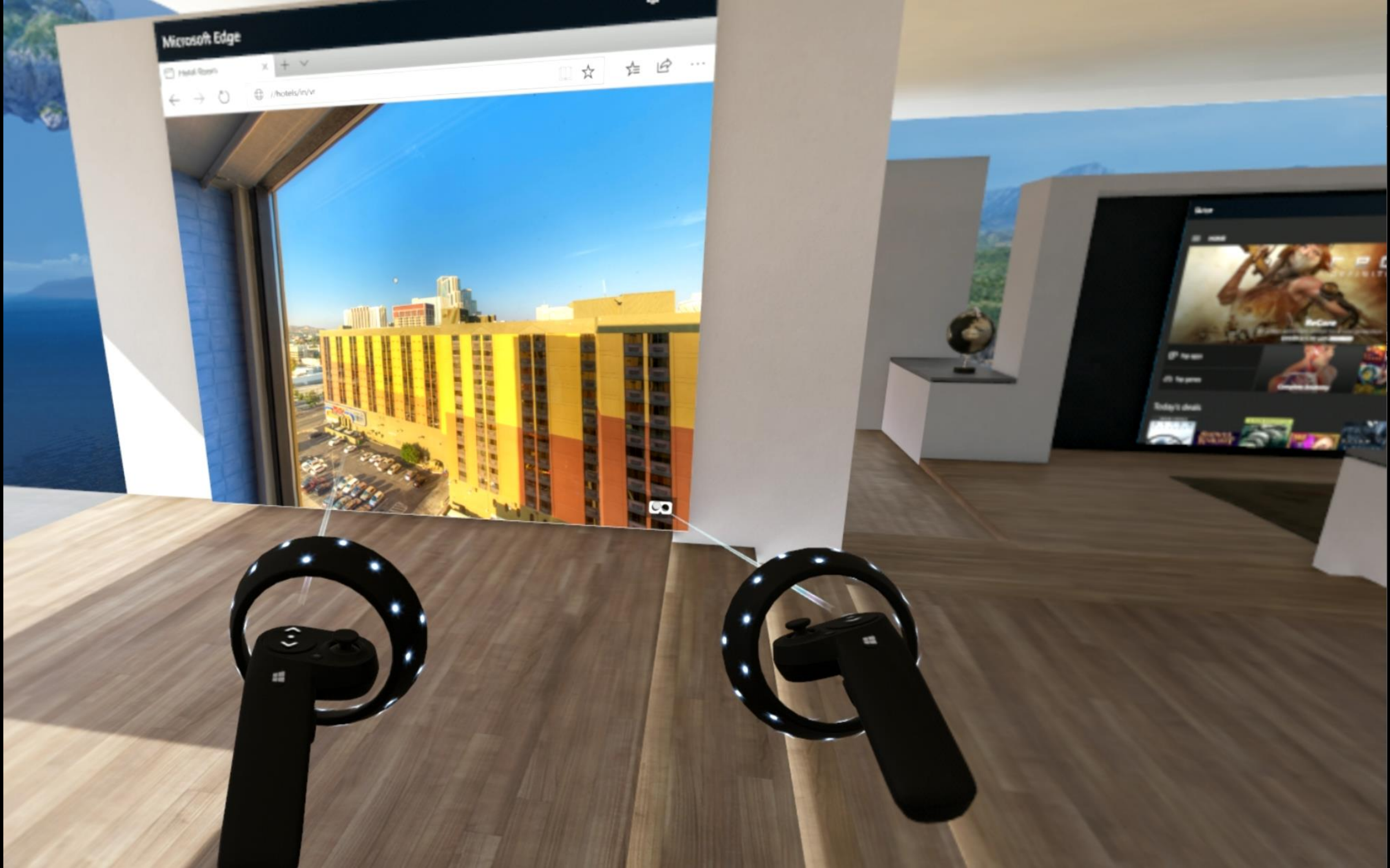


Image: Hotel Room, Reno, Nevada / Bob Dass / Creative Commons 2.0


e


Microsoft Edge
Hotel Room
://hotels/in/vr
Today's deals

Tourism

Real Estate

Online Shopping

360 photos and videos

Gaming

& More

Browser WebVR Support

Microsoft Edge	Windows Mixed Reality	
Firefox	HTC VIVE	oculus
Chromium *	HTC VIVE	oculus
Servo *	HTC VIVE	

Oculus Browser	Samsung GearVR	
Samsung Internet *	Samsung GearVR	
Chrome for Android *	daydream	cardboard
Chromium *	daydream	cardboard

#msedgesummit

* "experimental feature" or "origin trial"

HOW DOES IT WORK?

Headset	=	`VRDisplay`
Resolution	=	`VRDisplay.getEyeParameters()`
Frame callback	=	`VRDisplay.requestAnimationFrame()`
Pose, etc.	=	`VRDisplay.getFrameData()`

WebVR ❤ WebGL

WRITING CODE

#msedgesummit

Displaying WebVR content

1. Query for an available headset
2. Request access to use the headset
3. Draw to the headset using WebGL
4. Return to 2D

Displaying WebVR content

1. Query for an available headset
2. Request access to use the headset
3. Draw to the headset using WebGL
4. Return to 2D

```
var vrDisplay = null;

// Find connected displays
function findDisplays() {
  if (!navigator.getVRDisplays) {
    /* Fall back to orientation APIs */
    return;
  }

  navigator.getVRDisplays().then((vrDisplays)=> {
    vrDisplay = (vrDisplays.length > 0) ? vrDisplays[0] : null;
  }).catch( /* Fall back to orientation APIs */ );
}
```

```
var vrDisplay = null;

// Finds connected displays
function findDisplays() {
  if (!navigator.getVRDisplays) {
    /* Fall back to orientation APIs */
    return;
  }

  navigator.getVRDisplays().then((vrDisplays)=> {
    vrDisplay = (vrDisplays.length > 0) ? vrDisplays[0] : null;
  }).catch( /* Fall back to orientation APIs */ );
}
```

```
// Detect connected displays on initial load
findDisplays();

// Listen for headset connection
window.addEventListener('vrdisplayconnect', findDisplays);

// Listen for headset disconnection
window.addEventListener('vrdisplaydisconnect', findDisplays);
```

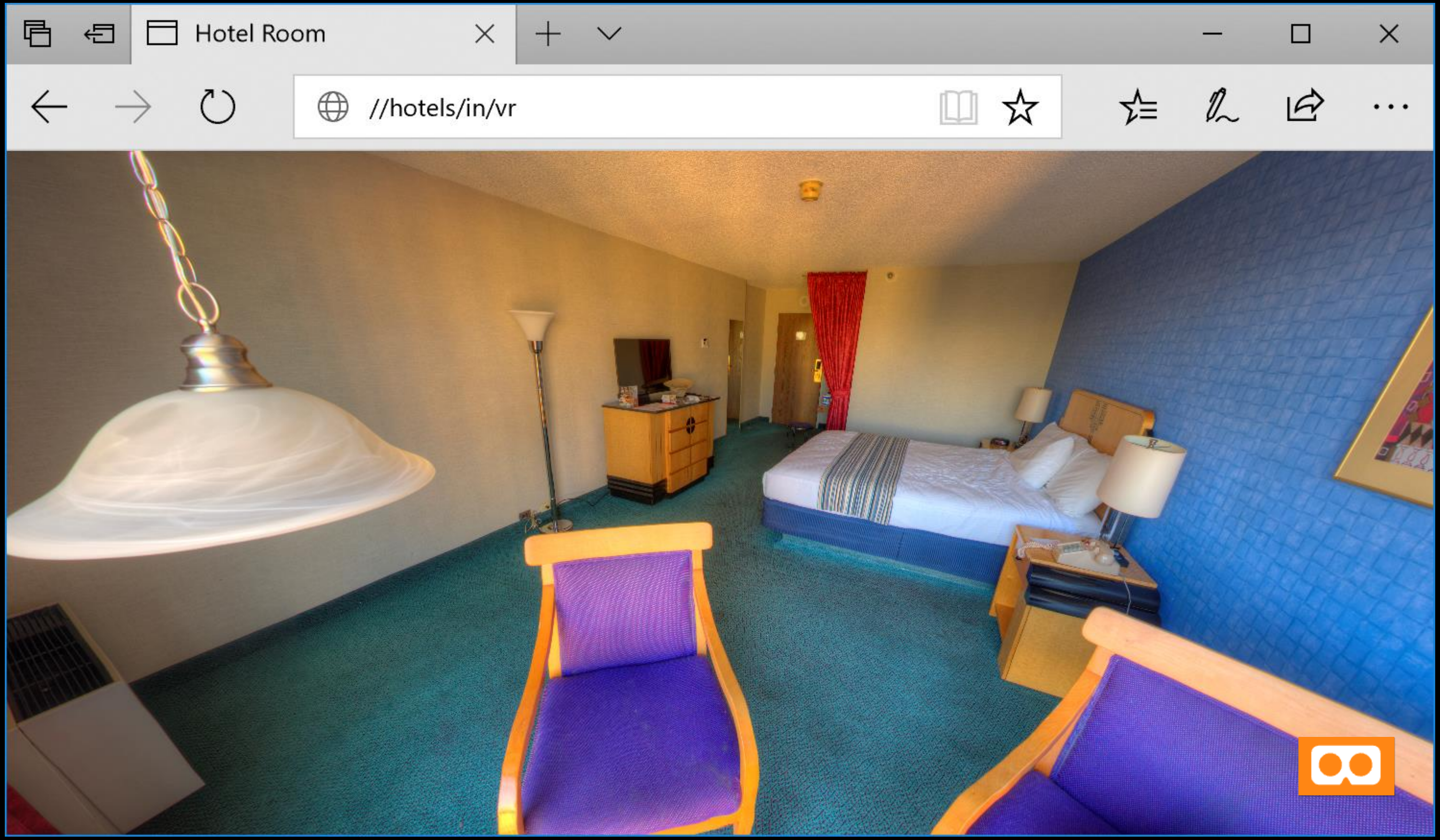
Displaying WebVR content

1. Query for an available headset
2. Request access to use the headset
3. Draw to the headset using WebGL
4. Return to 2D

```
var canvas = document.getElementById("webgl-canvas");
setupWebGLResources(canvas);

function enterVR() {
  // Request exclusive use of the headset for rendering
  vrDisplay.requestPresent([source:canvas]).then(()=>{

    // Queue animation callback
    queueAnimationFrameCallback();
  }).catch( /* Handle rejection */ );
});
```

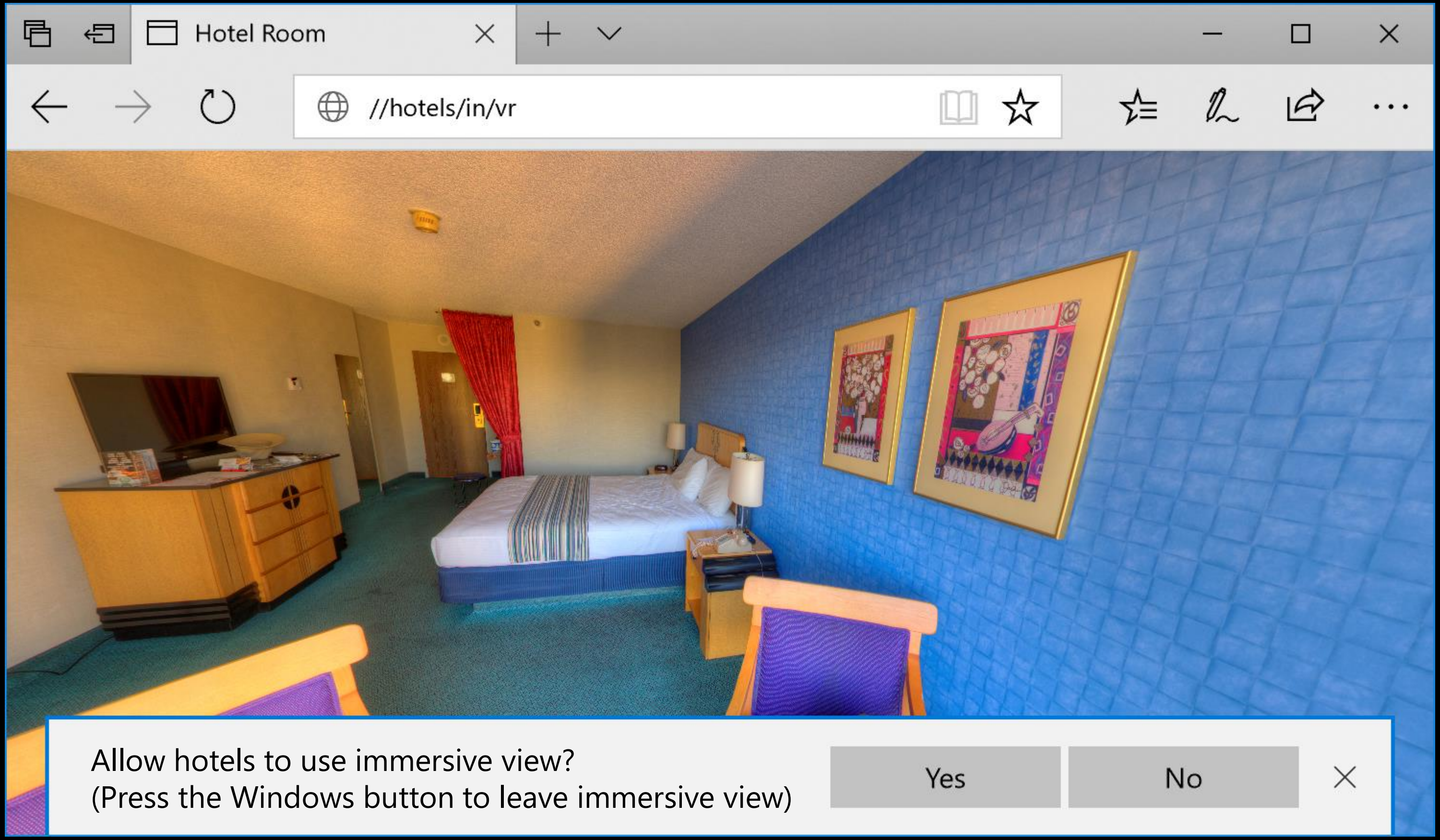
```
var callbackId;

function queueAnimationFrameCallback() {
  if (vrDisplay && vrDisplay.isPresenting)

    // Callback at HEADSET refresh rate
    callbackId = vrDisplay.requestAnimationFrame(
      onVrFrameCallback);
  } else {

    // Callback at WINDOW refresh rate
    callbackId = window.requestAnimationFrame(
      onWindowFrameCallback);
  }
}
```

```
var enterVRButton = document.getElementById('entervr');

// Handle user initiated button click
enterVRButton.addEventListener('click', enterVR);
```



#msedgesummit

```
var enterVRButton = document.getElementById('entervr');

// Handle user initiated button click
enterVRButton.addEventListener('click', enterVR);
```



Allow hotels to use immersive view?
(Press the Windows button to leave immersive view)

Displaying WebVR content

1. Query for an available headset
2. Request access to use the headset
3. Draw to the headset using WebGL
4. Return to 2D

```
var frameData = new VRFrameData();

function onVrFrameCallback() {

  // If the headset pose is available,
  // update the canvas, draw the pixels, and send to headset
  if (vrDisplay.getFrameData(frameData)) {
    updateCanvasSize();
    drawVRScene();
    vrDisplay.submitFrame();
  }

  // Queue the next frame
  queueAnimationFrameCallback();
}
```

```
var frameData = new VRFrameData();

function onVrFrameCallback() {

  // If the headset pose is available,
  // update the canvas, draw the pixels, and send to headset
  if (vrDisplay.getFrameData(frameData)) {
    updateCanvasSize();
    drawVRScene();
    vrDisplay.submitFrame();
  }

  // Queue the next frame
  queueAnimationFrameCallback();
}
```

```
// Update the canvas to be big enough for drawing both eyes
function updateCanvasSize() {

  // Get headset resolution
  var leftEye = vrDisplay.getEyeParameters("left");
  var rightEye = vrDisplay.getEyeParameters("right");

  // Update the canvas width
  canvas.width = leftEye.renderWidth + rightEye.renderWidth;

  // Update the canvas height
  canvas.height = Math.max(
    leftEye.renderHeight,
    rightEye.renderHeight);
}
```

```
function drawVRScene() {
  // Update 3D scene
  updateScene(frameData);

  // Render the left eye
  gl.setViewport(/* left half of canvas */);
  drawEye(
    frameData.leftViewMatrix,
    frameData.leftProjectionMatrix);

  // Render the right eye
  gl.setViewport(/* right half of canvas */);
  drawEye(
    frameData.rightViewMatrix,
    frameData.rightProjectionMatrix);
}
```

#msedgesummit

```
// Indicate intent to handle webglcontextrestored
function onContextLost( event ) {
  event.preventDefault();
}
canvas.addEventListener('webglcontextlost', onContextLost);

// Reload WebGL resources such as textures, etc
function onContextRestored() {
  setupWebGLResources(canvas);
}
canvas.addEventListener('webglcontextrestored', onContextRestored);
```

Displaying WebVR content

1. Query for an available headset
2. Request access to use the headset
3. Draw to the headset using WebGL
4. Return to 2D

```
var exitVRButton = document.getElementById('exitvr');

// Exit Present
function exitVR() {
  vrDisplay.exitPresent().catch( /* Handle rejection */ );
});

// Handle user initiated button click
exitVRButton.addEventListener('click', exitVR);
```

```
function onPresentChanged() {

  // Cancel outstanding callback
  if (vrDisplay.isPresenting) {
    window.cancelAnimationFrame(callbackID);
  } else {
    vrDisplay.cancelAnimationFrame(callbackID);
  }

  // Queue next frame
  queueAnimationFrameCallback();
}

// Register for presentation state change event
window.addEventListener(
  'vrdisplaypresentchange', onPresentChanged);
```

Displaying WebVR content

1. Query for an available headset
2. Request access to use the headset
3. Draw to the headset using WebGL
4. Return to 2D

Pairing Button
Clam Shell Select Click With Thumb Dish
Elastic Finger Loop
Micro USB for Charging

Interacting with WebVR content

- Targeting objects
- Providing user feedback
- APIs

Gaze-and-commit

- Gamepad button
- Mouse click
- Keyboard press
- Steady hover

Gaze-and-commit

- Gamepad button
- Mouse click
- Keyboard press
- Steady hover

Point-and-commit

- Motion controller button

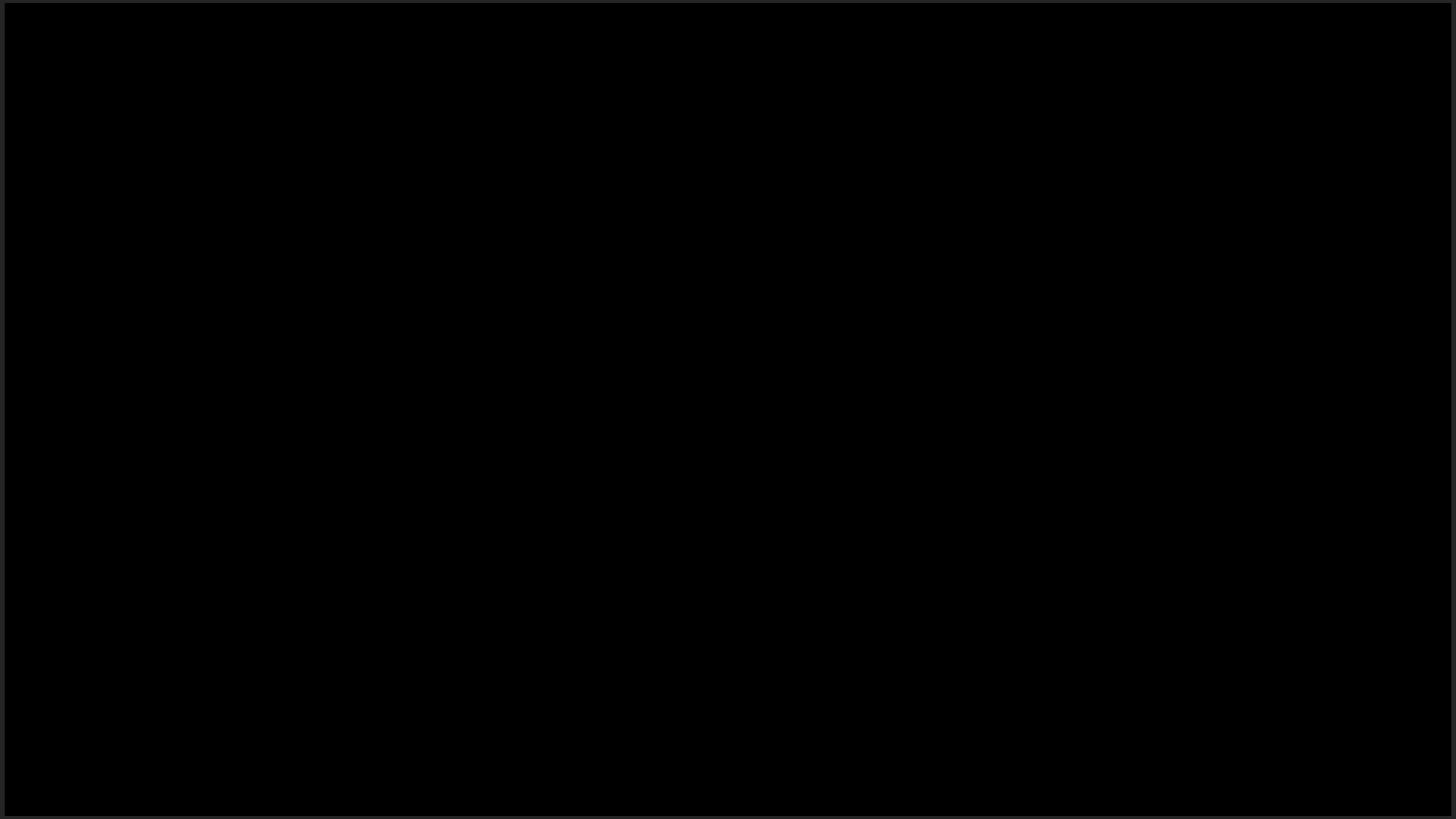
Interacting with WebVR content

- Targeting objects
- Providing user feedback
- APIs

Cursor

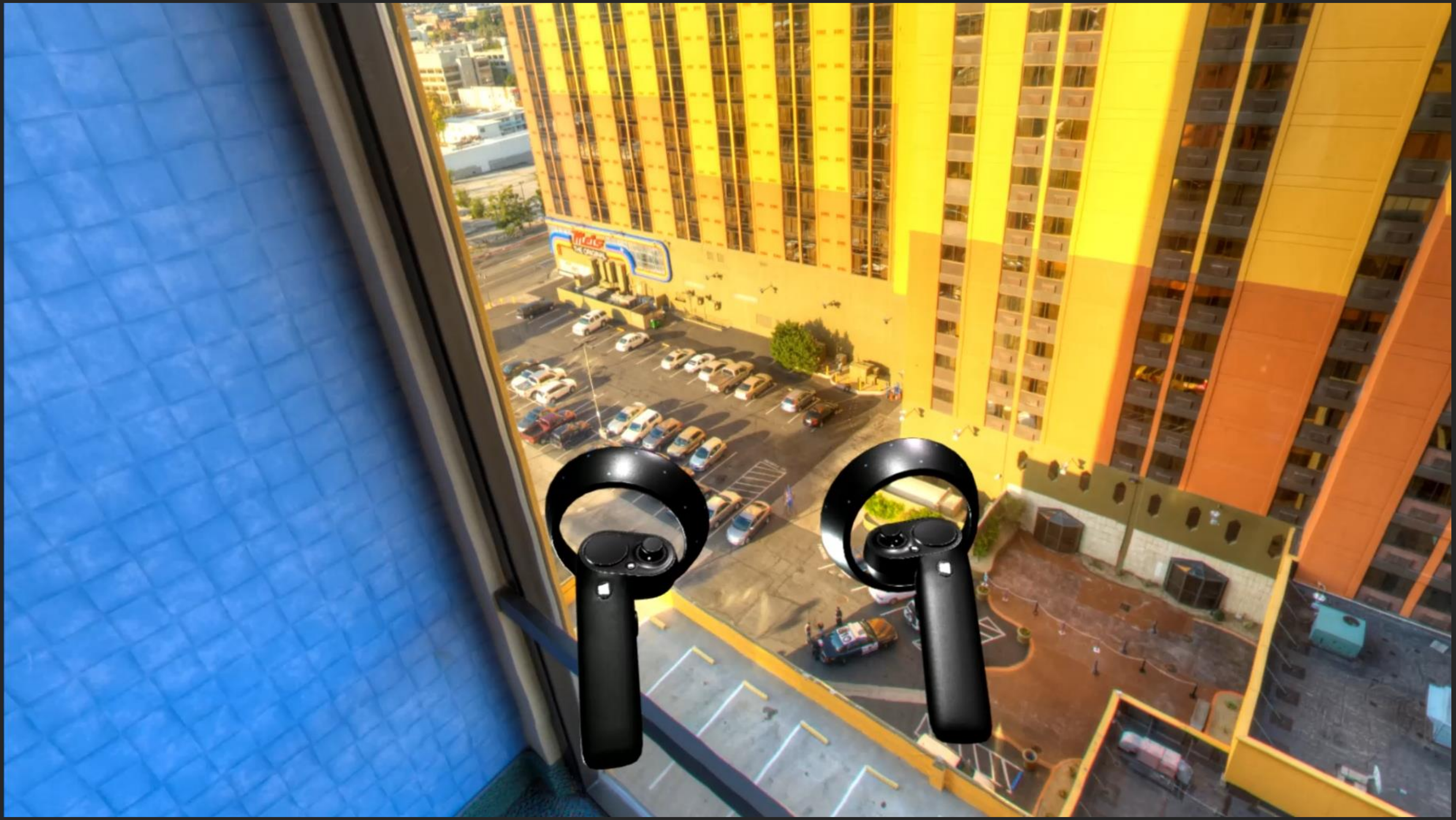
#msedgesummit

Pointing Ray



#msedgesummit

Motion Controllers and buttons



Interacting with WebVR content

- Targeting objects
- Providing user feedback
- APIs

Gamepads & controllers	=	`navigator.getGamepads()`
Motion controller pose	=	`Gamepad.pose`
Gaze ray origin	=	`VRFrameData.pose`
Mouse clicks	=	`element.requestPointerLock()`

Gamepads & controllers	=	`navigator.getGamepads()`
Motion controller pose	=	`Gamepad.pose`
Gaze ray origin	=	`VRFrameData.pose`
Mouse clicks	=	`element.requestPointerLock()`

Gamepads & controllers	=	`navigator.getGamepads()`
Motion controller pose	**=**	**`Gamepad.pose`**
Gaze ray origin	=	`VRFrameData.pose`
Mouse clicks	=	`element.requestPointerLock()`

Gamepads & controllers	= `navigator.getGamepads()`
Motion controller pose	= `Gamepad.pose`
Gaze ray origin	**= `VRFrameData.pose`**
Mouse clicks	= `element.requestPointerLock()`

Gamepads & controllers	=	`navigator.getGamepads()`
Motion controller pose	=	`Gamepad.pose`
Gaze ray origin	=	`VRFrameData.pose`
Mouse clicks	=	`element.requestPointerLock()`

```
// Event handler for vrdisplaypresentchange
function onPresentChanged() {
  …
  if (vrDisplay.isPresenting) {
    canvas.requestPointerLock();
  } else {
    document.exitPointerLock();
  }
  …
}
```

#msedgesummit

```
// Ensure pointerlock taken when restricted
window.addEventListener('vrdisplaypointerrestricted', () => {
  canvas.requestPointerLock();
});

// Ensure pointerlock release when unrestricted
window.addEventListener('vrdisplaypointerunrestricted', () => {
  document.exitPointerLock();
});
```

Interacting with WebVR content

- Targeting objects
- Providing user feedback
- Coding it up

LIBRARIES

babylon.JS

three.js

React VR

A-FRAME

Windows Mixed Reality support

	Basics	WebGL context switching	Motion Controllers
babylon.JS 3.1	✔	✔	✔
A-FRAME 0.7.0*	✔	✔	✔
three.js R88*	✔	✔	✔
React VR 2.0.0*	✔	✔	✓

#msedgesummit

three.js

- Lightweight 3D library
- Define scenes and geometry in JavaScript
- Fine grained control over rendering
- Provides WebVR built-in
- Motion controller support under development

React VR

- Familiar declarative style of React
- Use React components in VR
- React Libraries and Tools
- Motion controller example code available

A-FRAME

- Make WebVR using HTML
- Handles VR setup
- Entity/Component system
- Component Registry
- Gaze-and-commit support
- Point-and-commit support

babylon.JS

- JavaScript 3D engine
- High degree of control over rendering
- Add VR with 1 line of code
- Gaze-and-commit support
- Point-and-commit support
- Doc.BabylonJS.com
- BabylonJS-Playground.com

A-Frame Example

```html
<a-scene>
  <!-- VR Controllers -->
  <a-entity laser-controls="hand: left"></a-entity>
  <a-entity laser-controls="hand: right"></a-entity>

  <!-- Hotel Room -->
  <a-sky src="hotel-room.jpg"></a-sky>
</a-scene>
```

BabylonJS Example

```
var scene = new BABYLON.Scene(engine);
scene.createDefaultVRExperience();

var skybox = scene.createDefaultSkybox(new
BABYLON.Texture(
    "/assets/purple-room.jpg", scene, true), false);
skybox.material.reflectionTexture.coordinatesMode =
    BABYLON.Texture.FIXED_EQUIRECTANGULAR_MODE;
```

4 SUGGESTED PRACTICES

#msedgesummit

Prioritize frame rate over scene complexity

Start using a headset early on

Considerations for Maximum User Comfort

#msedgesummit

Test with diverse hardware

What’s next for WebVR

Thank you!

DECK	aka.ms/edgesummit-webvr
RESOURCES	aka.ms/edgesummit-webvr-docs
TWITTER	@NellWaliczek, @lew_weav
GITHUB	github.com/NellWaliczek, github.com/leweaver

#msedgesummit