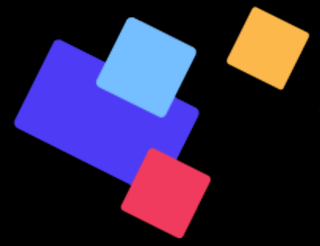


WebSpatial API

The boarding pass for the mainstream Web to multimodal AI devices

Dexter Yang (PICO OS, ByteDance)

Server-side (Cloud) AI Agent

Replacing humans

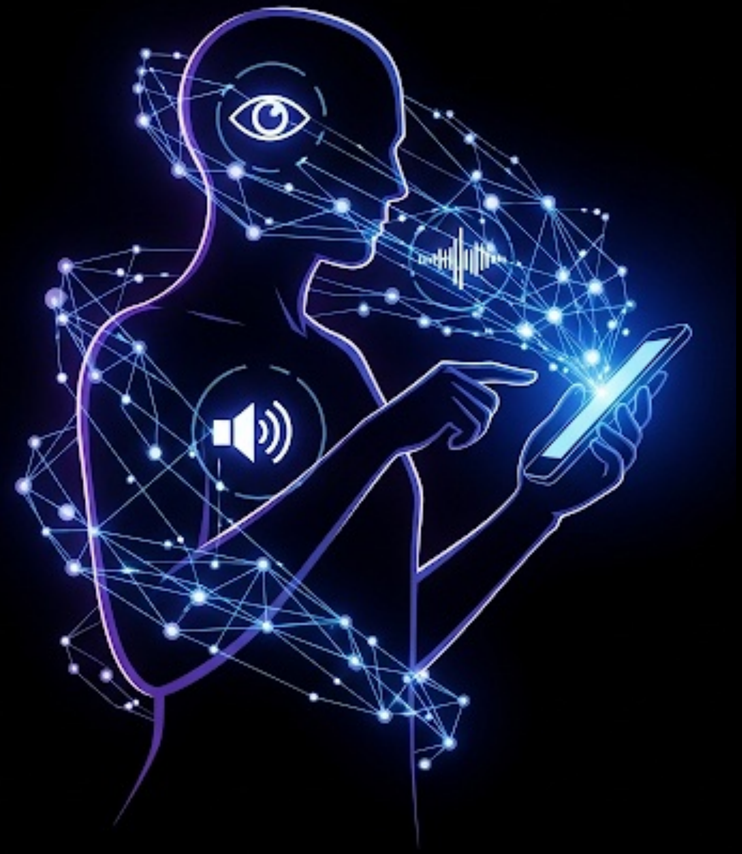
(Supply of human resources)

VS

Client-side (Edge) AI Agent

Augmenting humans

(Excluding robots)

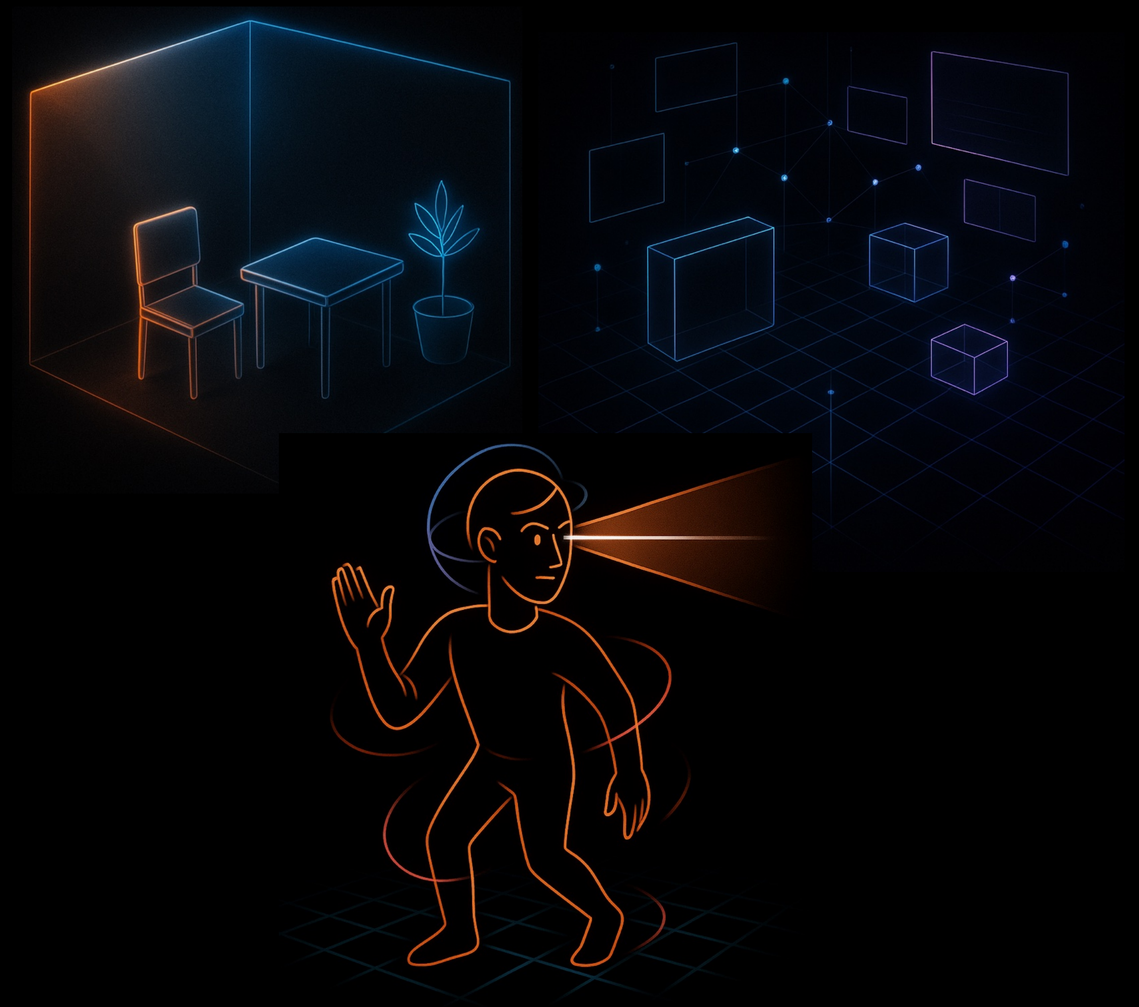


What you can see/hear,
AI should also see/hear.

Client-side (Edge)
AI Agent

You (Human)

Multimodal
Interactive Devices

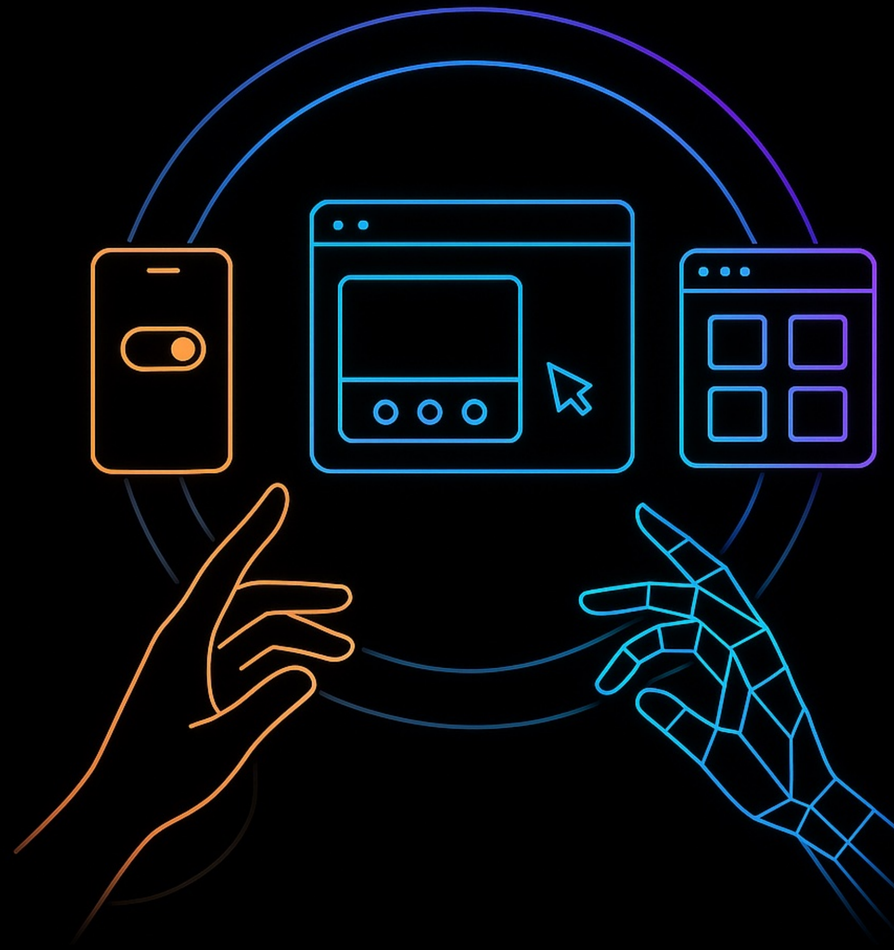


What you can use,
AI should also use.

Client-side (Edge)
AI Agent

You (Human)

Multimodal
Interactive Devices

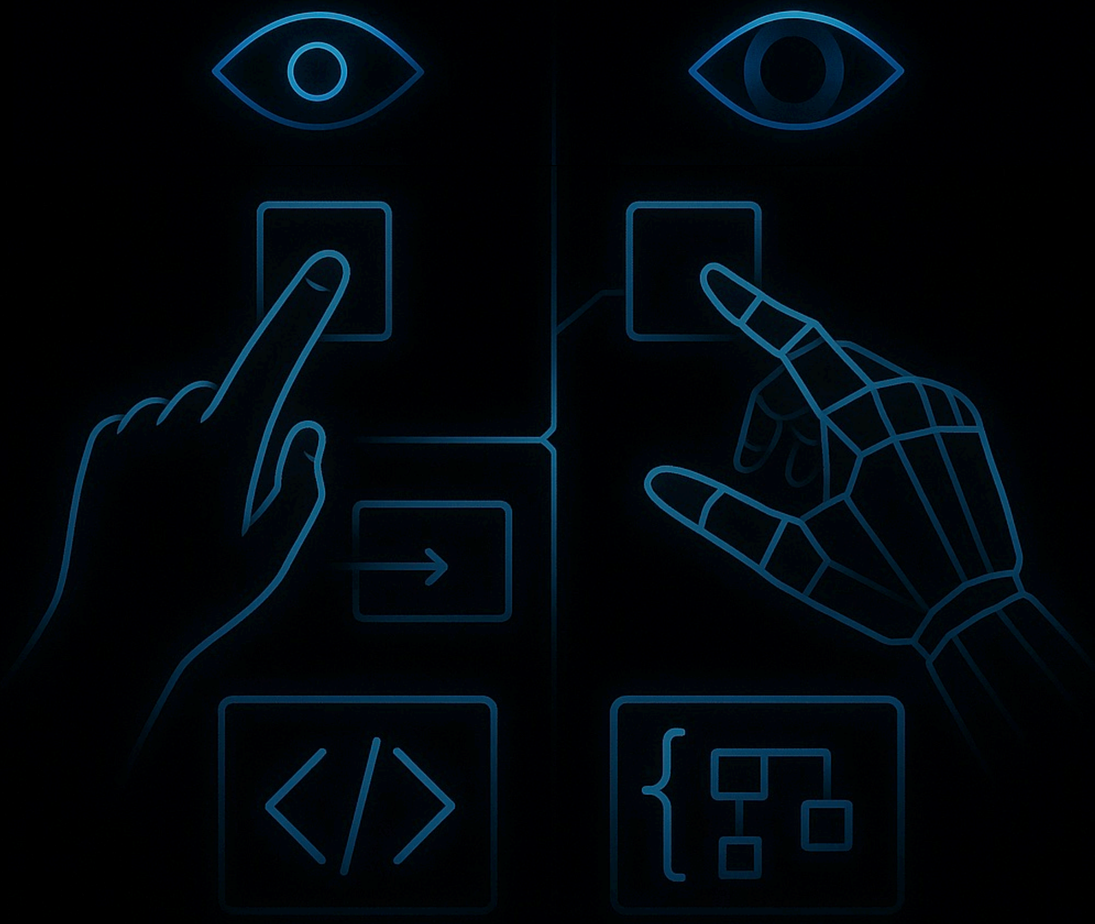


The way you use it,
AI should use it the same way.

Client-side (Edge)
AI Agent

You (Human)

Multimodal
Interactive Devices

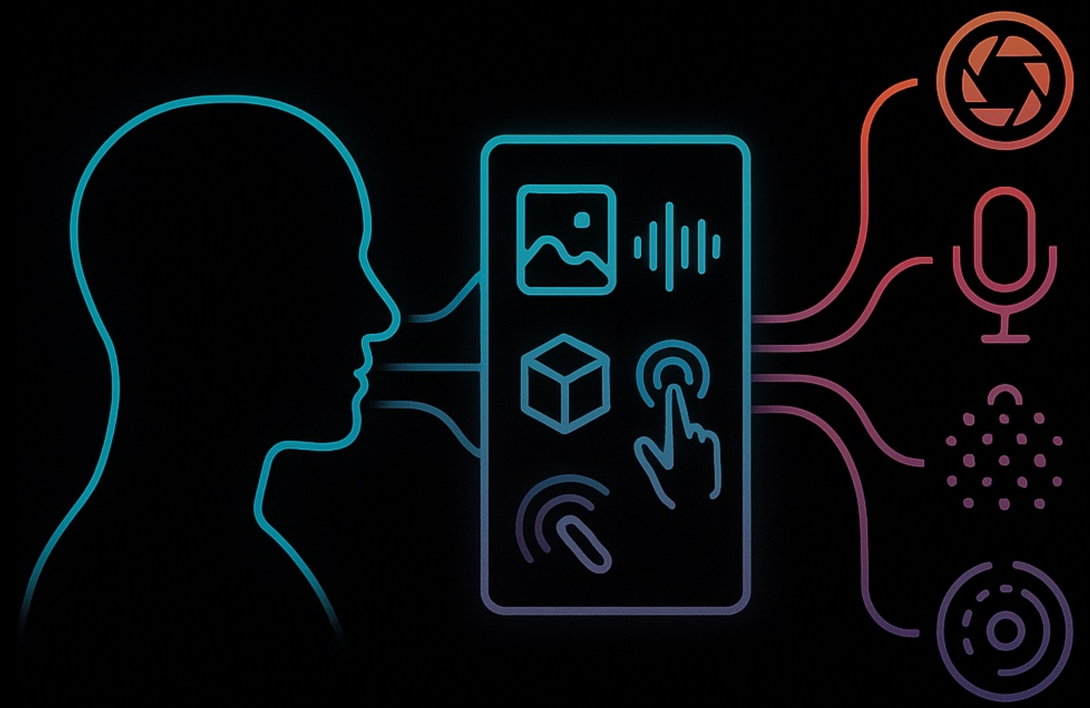


What prevents your errors,
AI needs.
What gives you clarity,
AI provides.

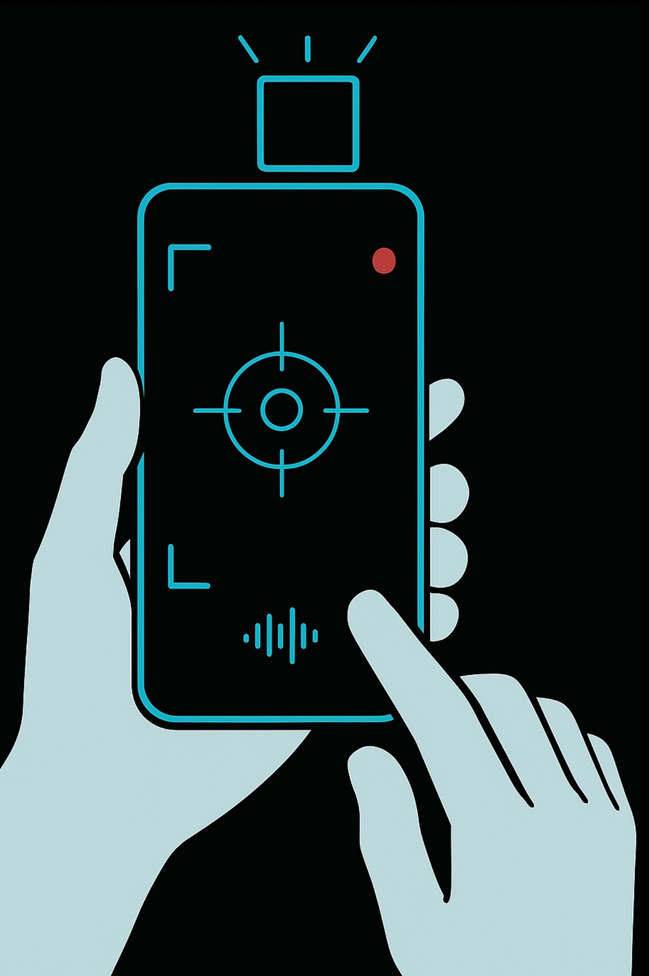
Client-side (Edge)
AI Agent

You (Human)

Multimodal
Interactive Devices



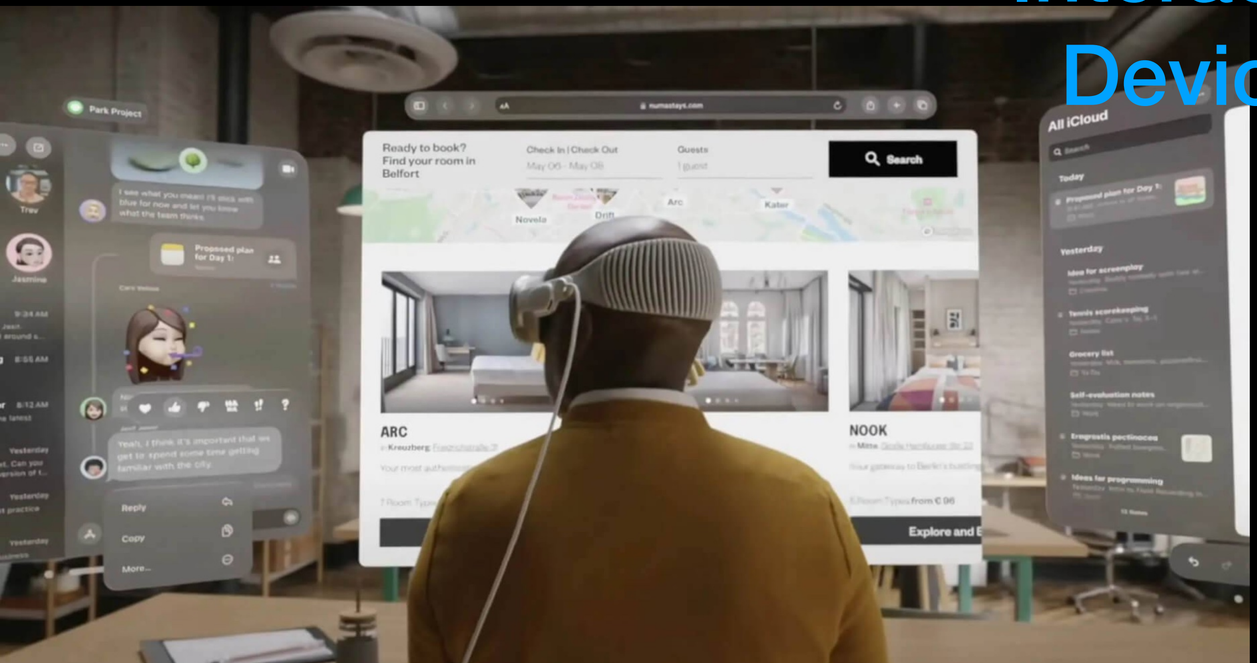
Handheld Devices



Wearable Devices

Hands-free Wearable (Head-mounted)

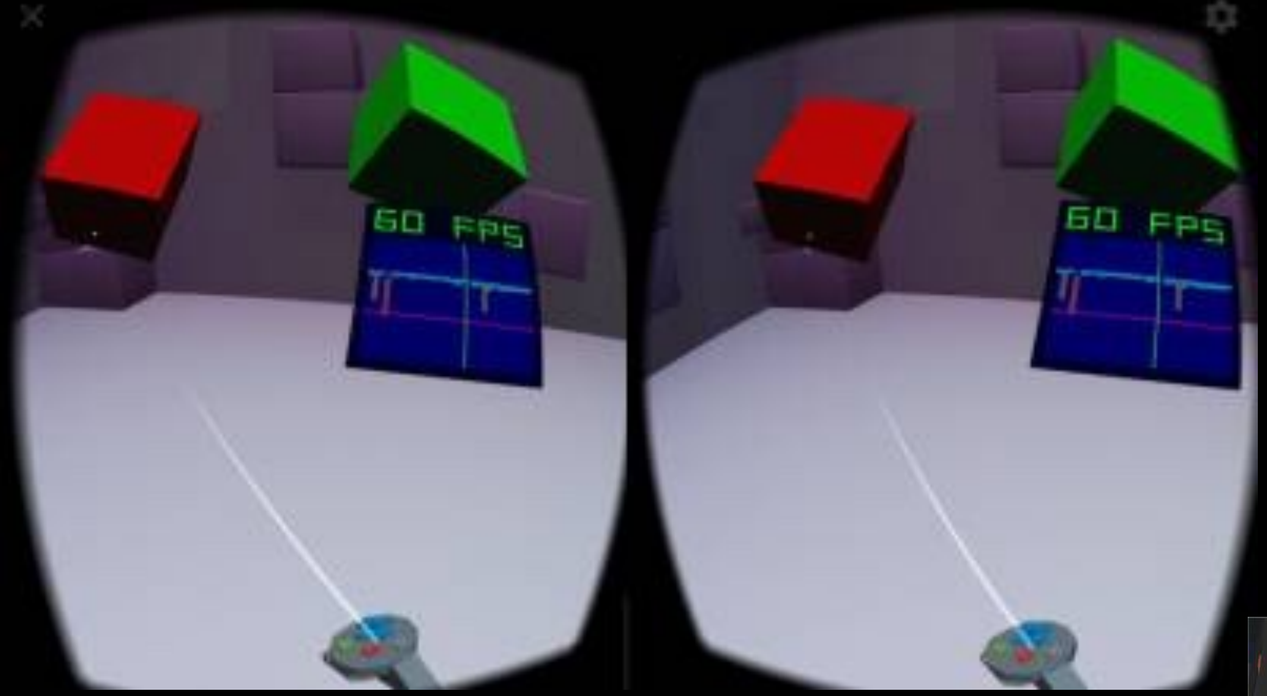
Apps on Multimodal Interactive Devices



Apps on Multimodal Interactive Devices

Apps on Multimodal Interactive Devices

Apps on Multimodal Interactive Devices



Previous-paradigm XR Apps

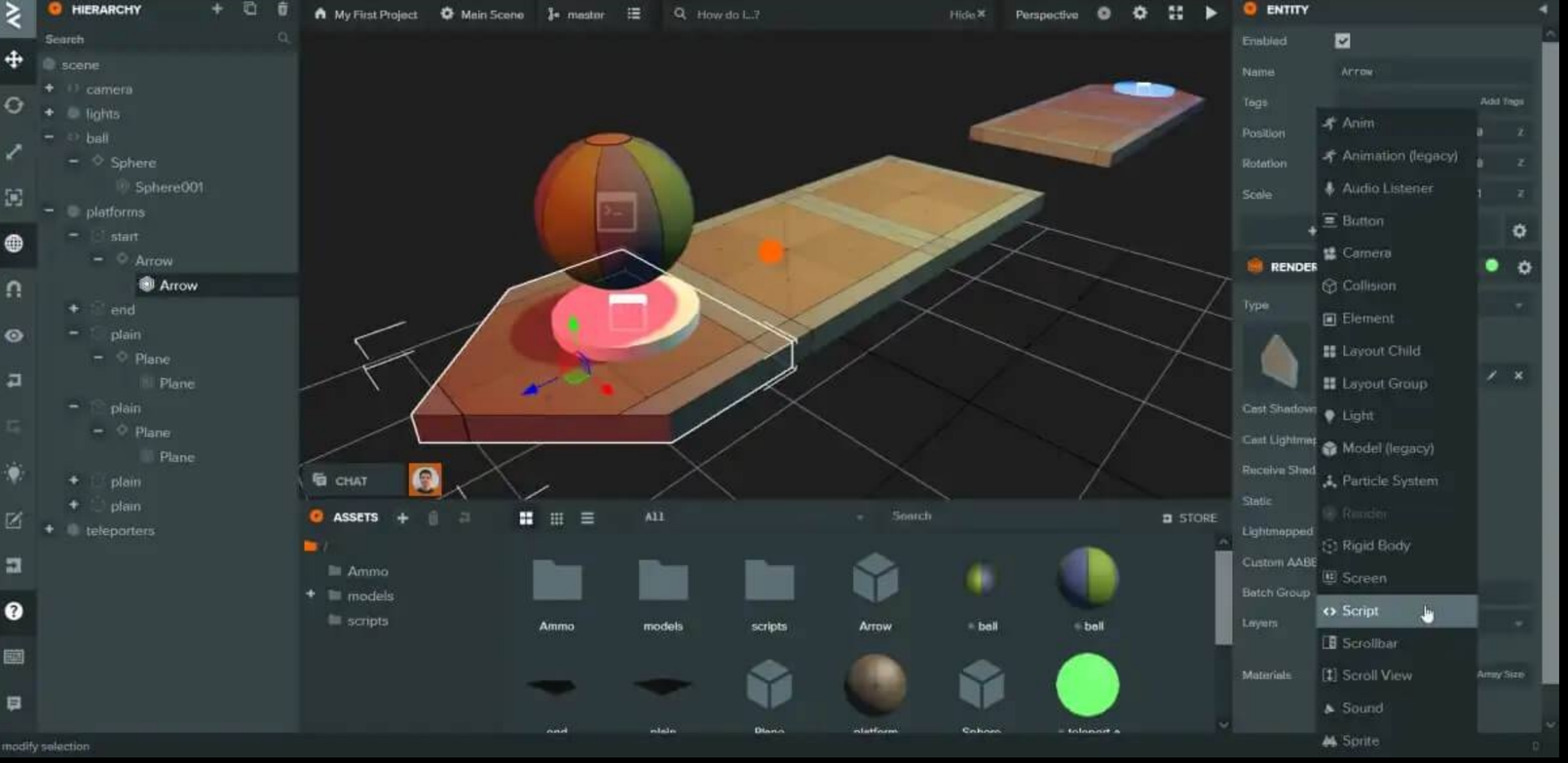
WebXR

WebGL

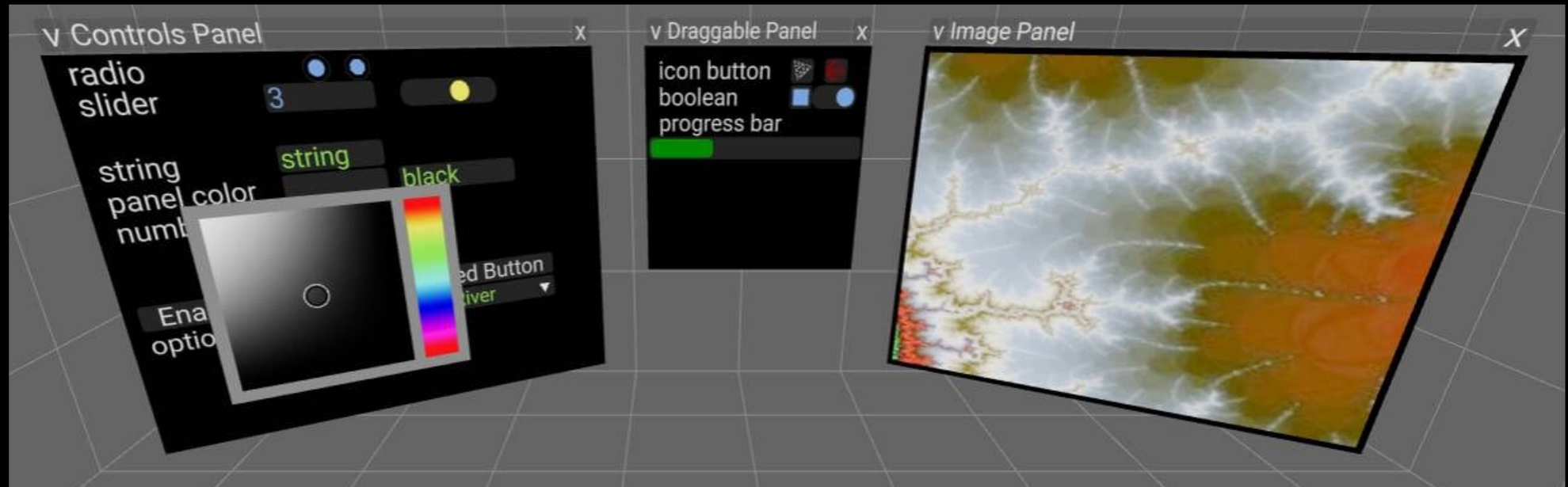
WebGPU

OpenXR

Vulkan

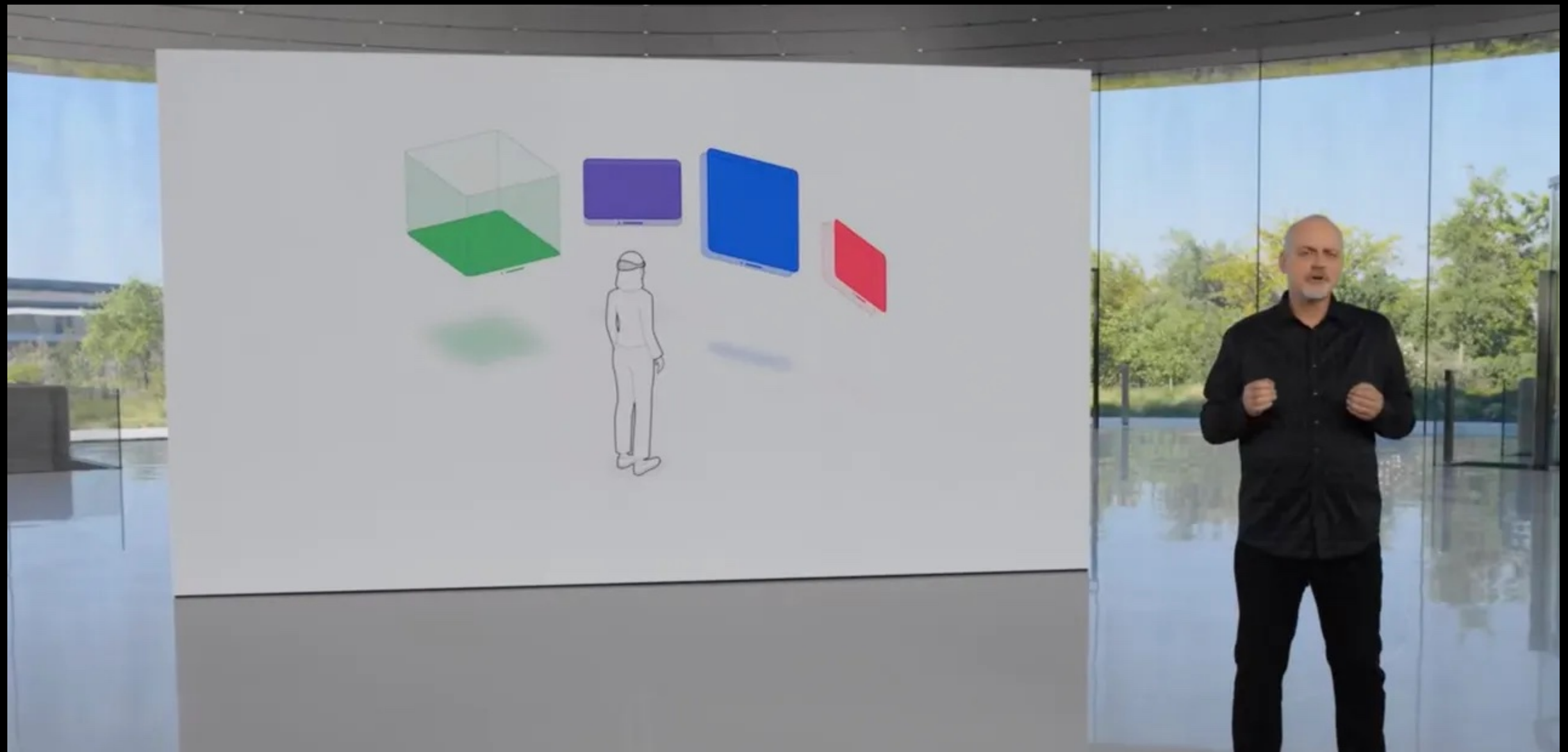


Previous-paradigm XR Apps



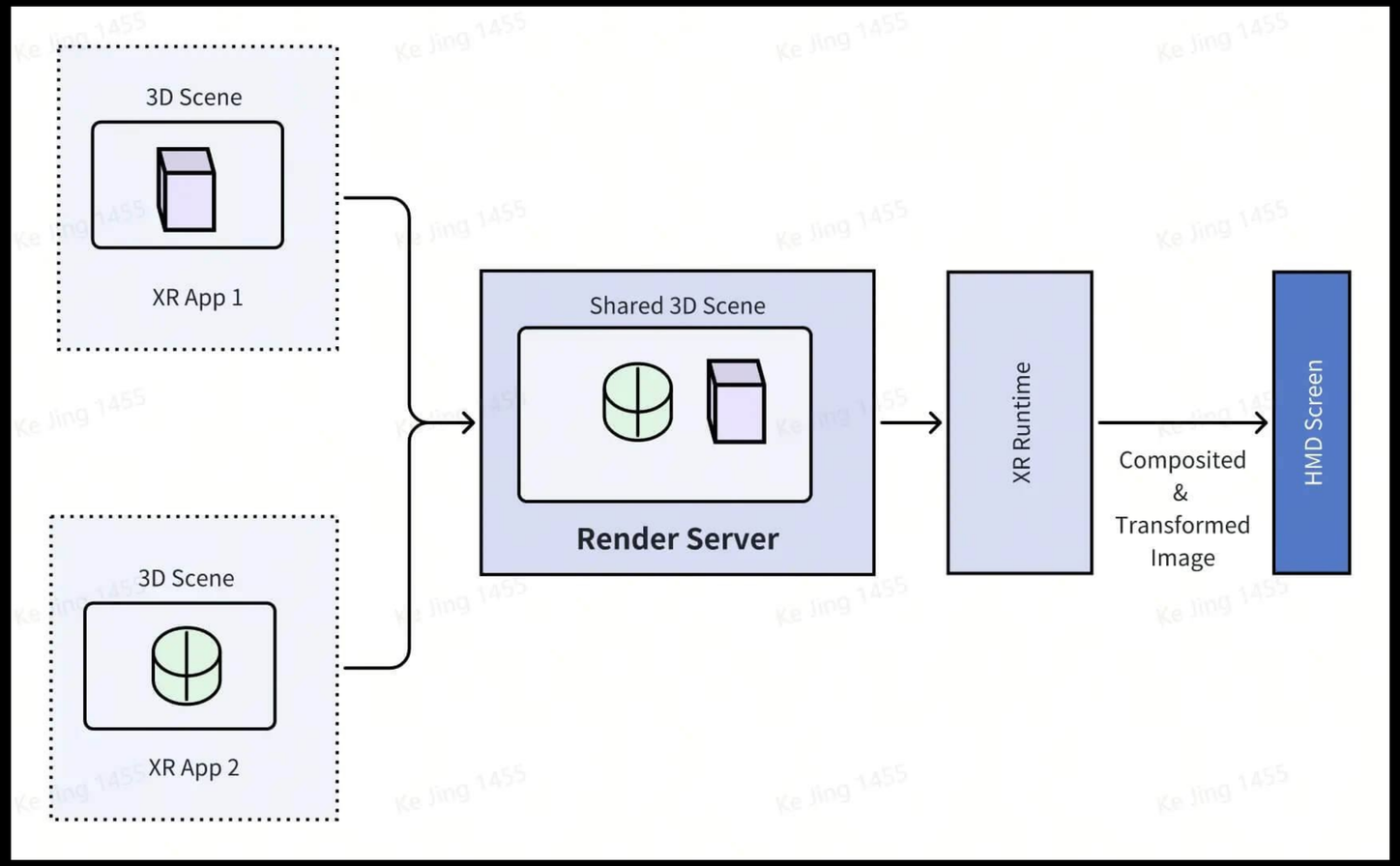
2D Apps in Previous-paradigm XR OS

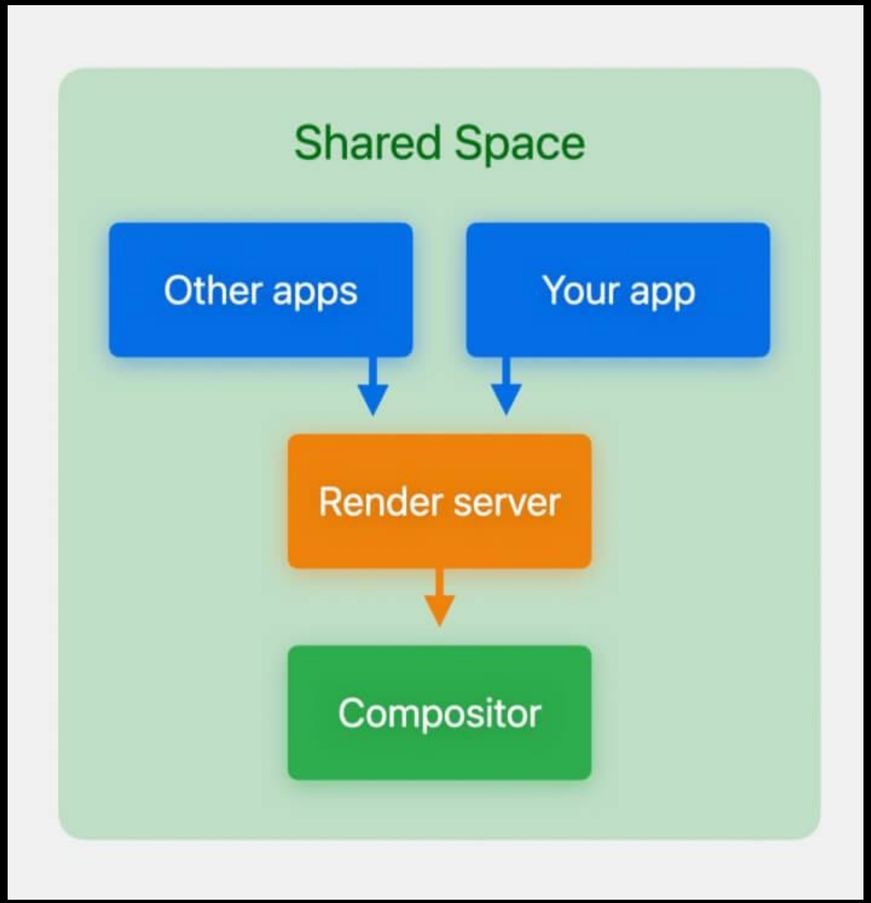
Shared Space



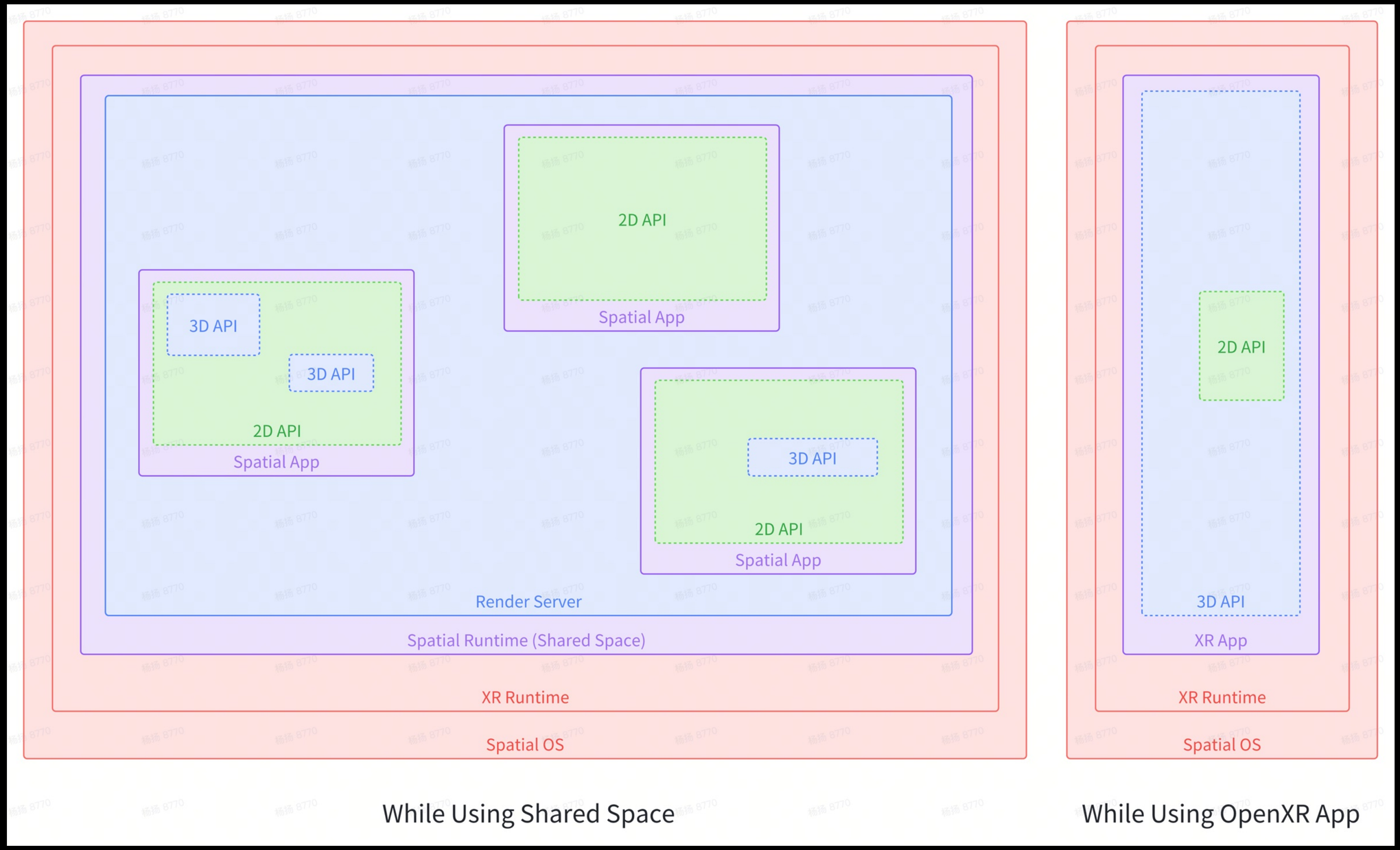
Shared Space

Unified Rendering

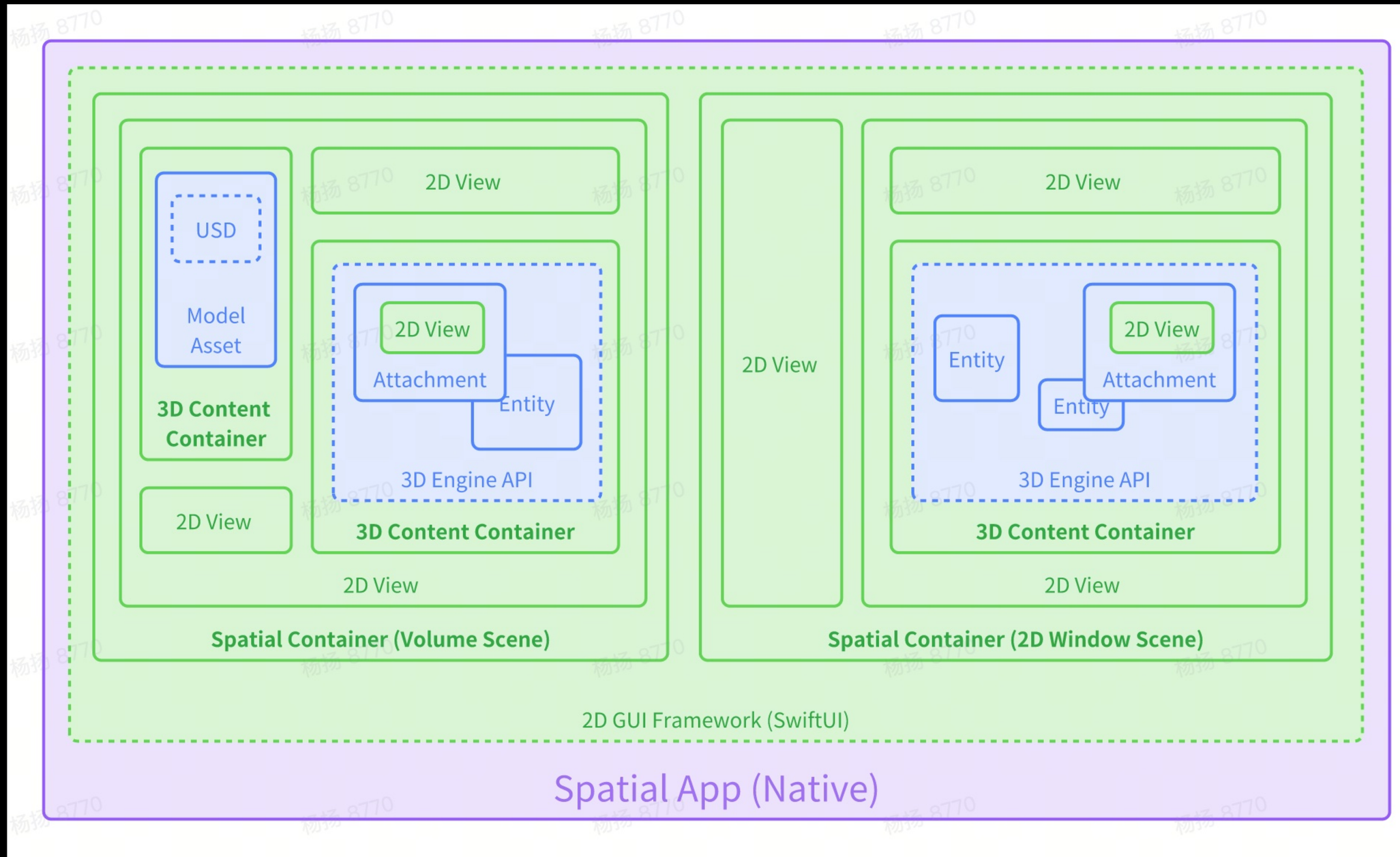




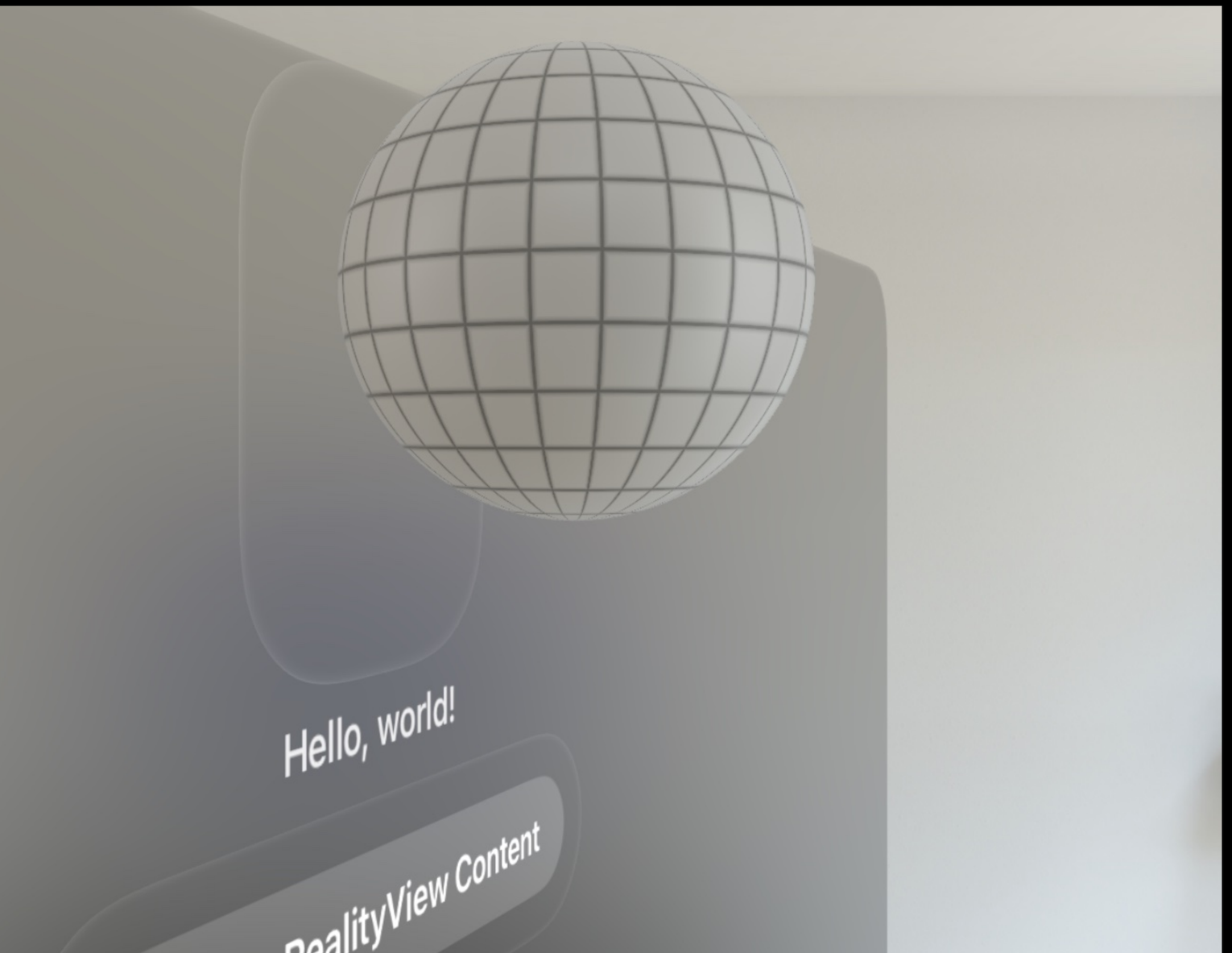
Spatial Apps - 2D-first Development Paradigm



Spatial Apps - 2D-first Development Paradigm

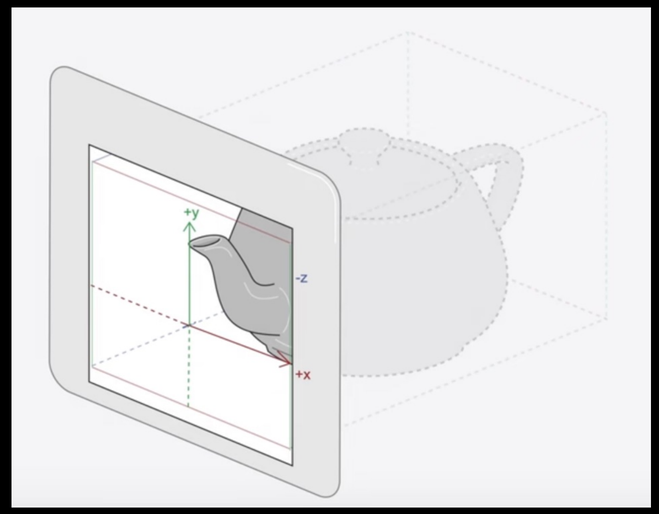


Spatial Apps - 2D-first Development Paradigm

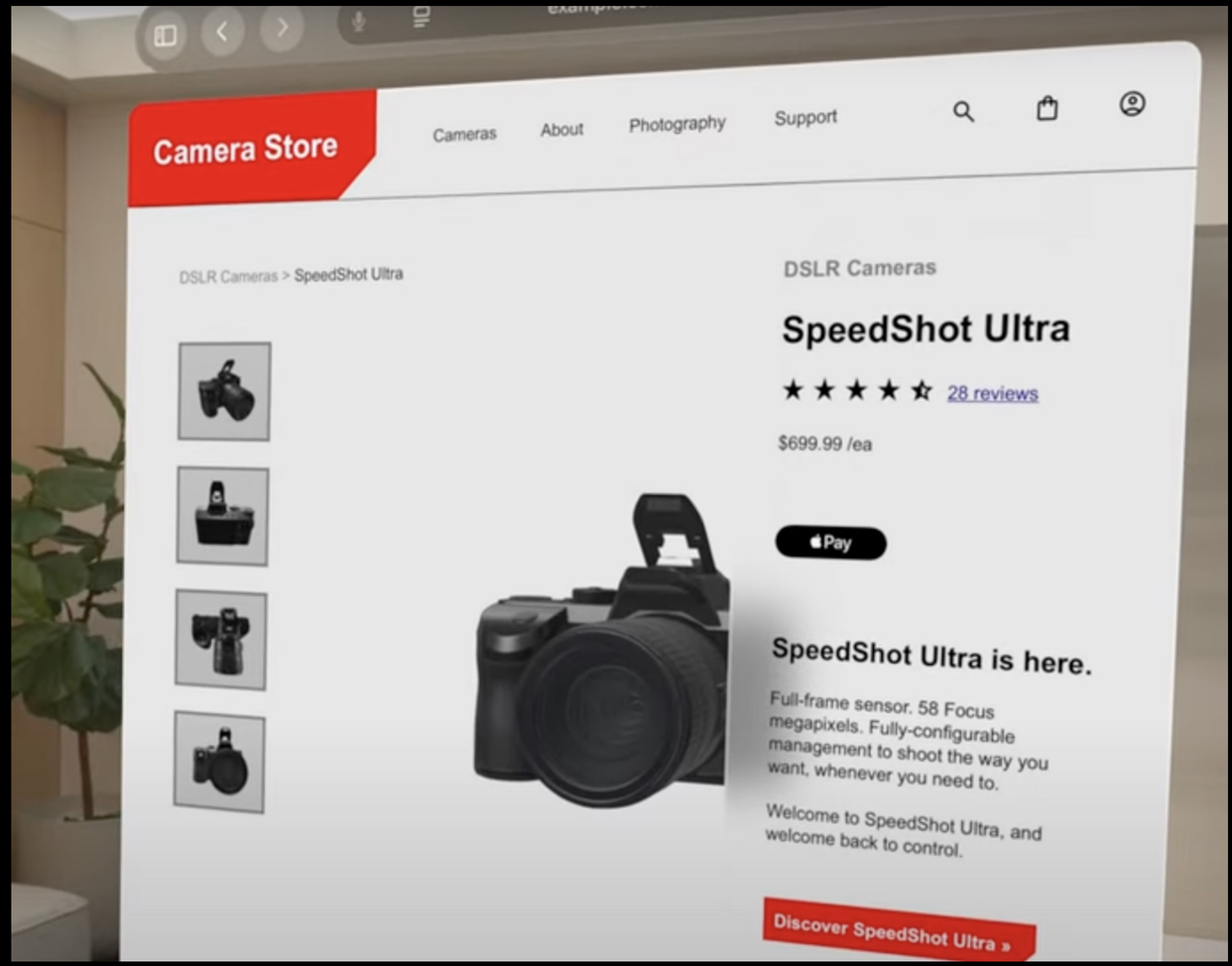


Existing Web Standards & Browser Implementations

- Model Element



HTML <model> Element APIs / width / height / src / currentSrc / complete / ready / boundingBoxCenter / boundingBoxExtents / entityTransform / autoplay / loop / playbackRate / duration / paused / play() / currentTime / stageMode / environmentMap / pause() / environmentMapReady



Existing Web Standards & Browser Implementations

- Model Element
- Immersive Media - Fullscreen API

HTML

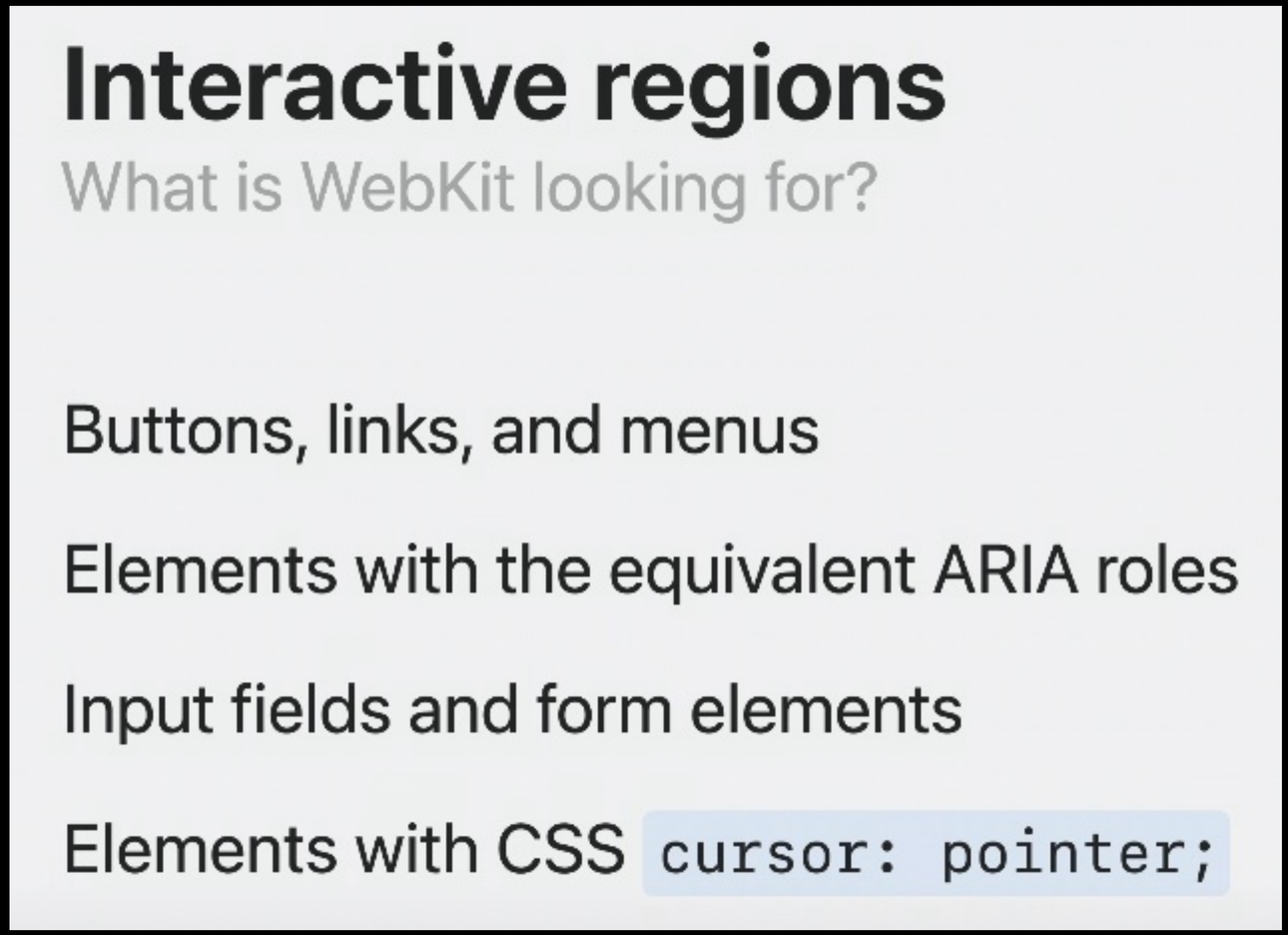
```
<picture>
  <source srcset="images/spatial/bikerack1.heic" type="image/heic">
  <img src="images/spatial/bikerack1.png"
    id="fullscreenTarget"
    alt="View from the bike rack"
  >
</picture>
```

Javascript

```
fullscreenTarget.addEventListener('click', event=>{
  try {
   fullscreenTarget.requestFullscreen();
  } catch(error) {
   //handle error gracefully
  }
});
```

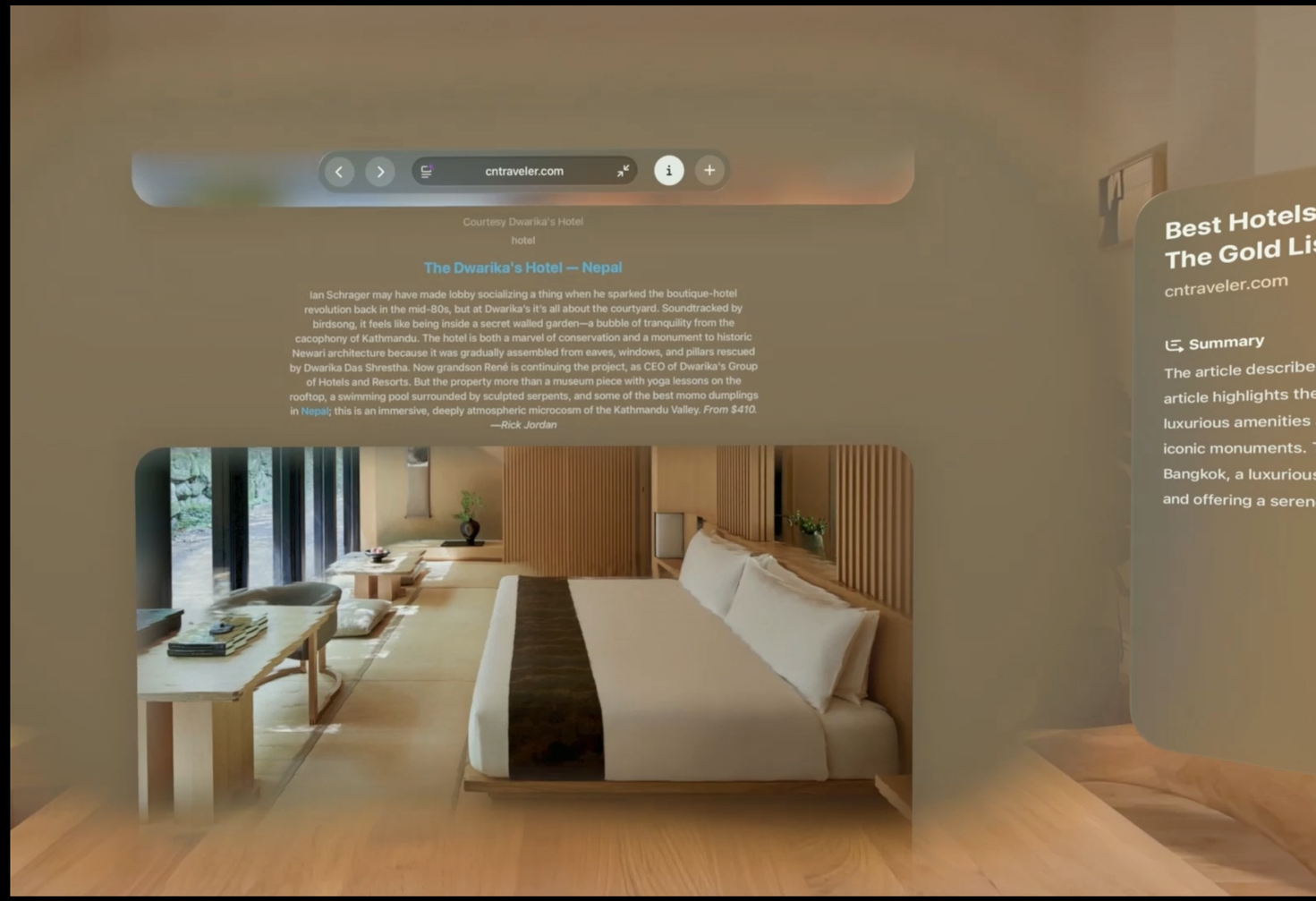
Existing Web Standards & Browser Implementations

- Model Element
- Immersive Media - Fullscreen API
- Hover Effect - Interaction Region
- Spatial Browsing



Existing Web Standards & Browser Implementations

- Model Element
- Immersive Media - Fullscreen API
- Hover Effect - Interaction Region
- Spatial Browsing



https://webspatial.dev/

webspatial / webspatial-sdk ★ 62

Type / to search

Code | Issues 70 | Pull requests 3 | Discussions | Actions | Security 8 | Insights | Settings

main | webspatial-sdk / README.md | Go to file t

dexteryy Replace repo docs with official site link; remove docs from the repo — 4f99359 · 3 months ago | Previous | His

Preview | Code | Blame | Raw

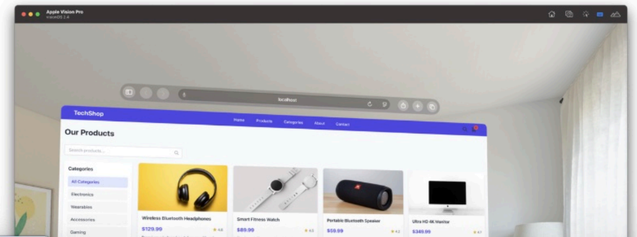
WebSpatial

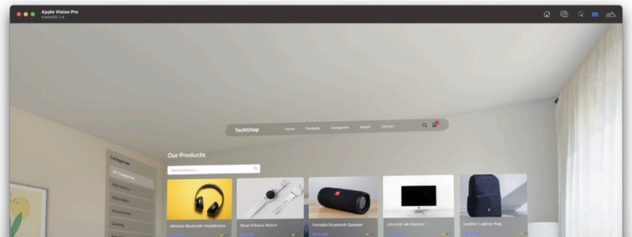
[Video] What if the web could be spatialized too?

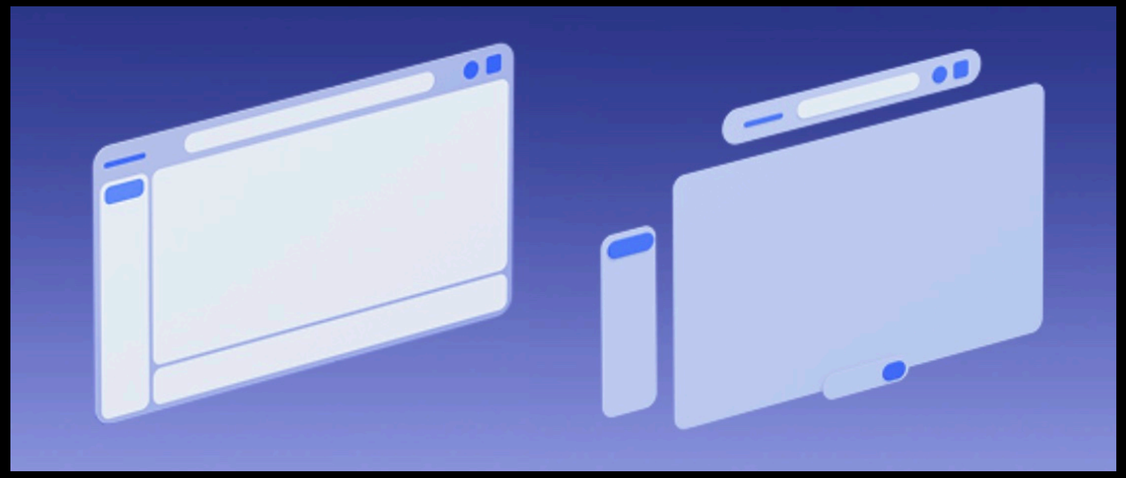
WebSpatial SDK

WebSpatial is a set of spatial APIs and ready-to-use SDK that extend the standard 2D Web ecosystem to support spatial computing across platforms. It enables the entire HTML/CSS-based Web world to step into the spatial era, gaining spatial power on par with native apps (like visionOS apps) while keeping the advantages they already have.

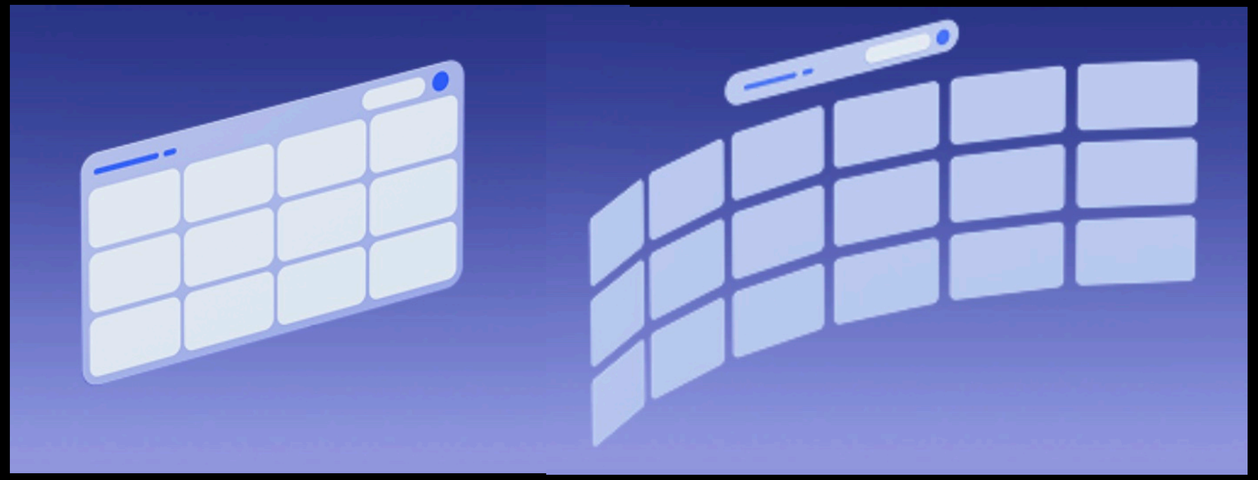
- **Open standard vision**: Extends existing HTML/CSS/JS with minimal new spatial APIs
- **2D Developer friendly**: Extending the existing web development ecosystem and 2D web development mindset
- **Zero-rewrite adoption** – Drop the SDK into an existing React project and with zero intrusion or additional cost
- **Cross-platform** – desktops, mobiles, and spatial-computing platfroms share one codebase.







Split the Web Page, Free the UI



Elevate HTML Elements, Unlock Depth

Multiple Scene Containers, Native Power

Add True 3D Content, Blend Dimensions

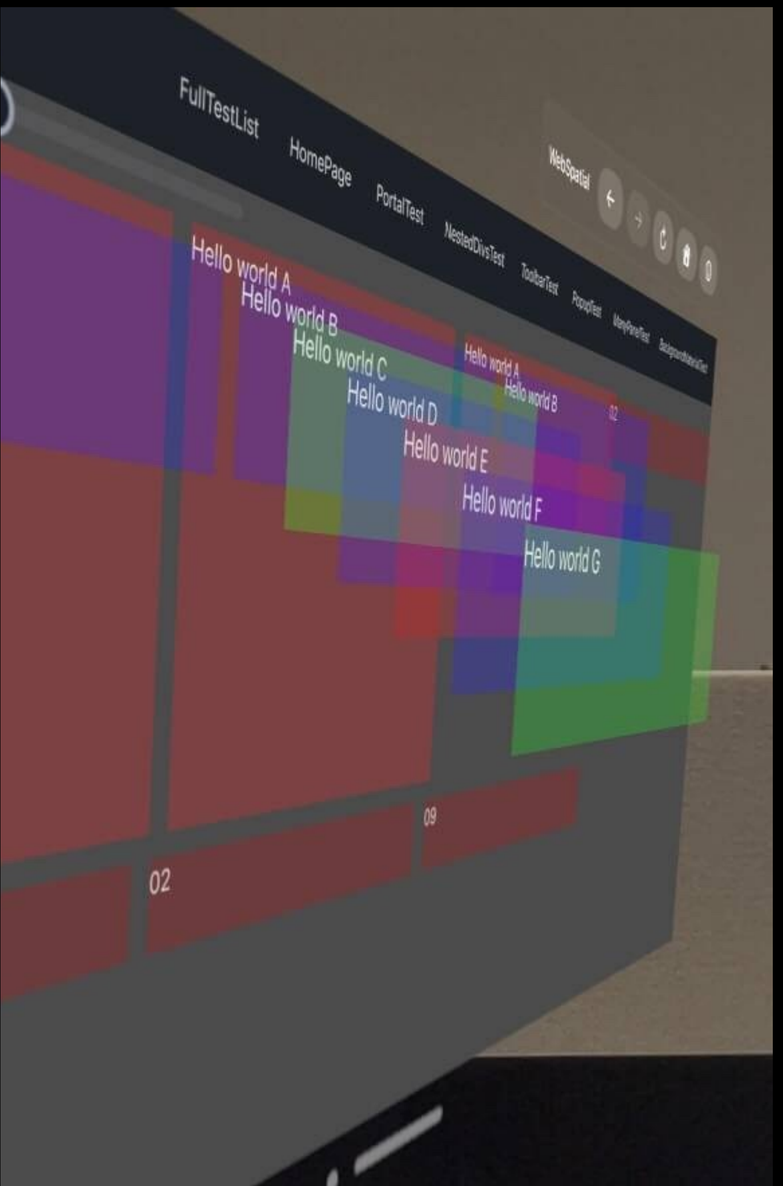
WebSpatial API: minimal new APIs extending HTML/CSS/DOM

- Layout/Style System: X/Y -> X/Y/Z, width/height -> width/height/depth
- 2D Content: Spatialized HTML Elements
- 3D Content: Extended Model Element, Dynamic 3D Container Element
- Web App Manifest / New Window -> Spatial App / Spatial Scene

WebSpatial API: Material Background

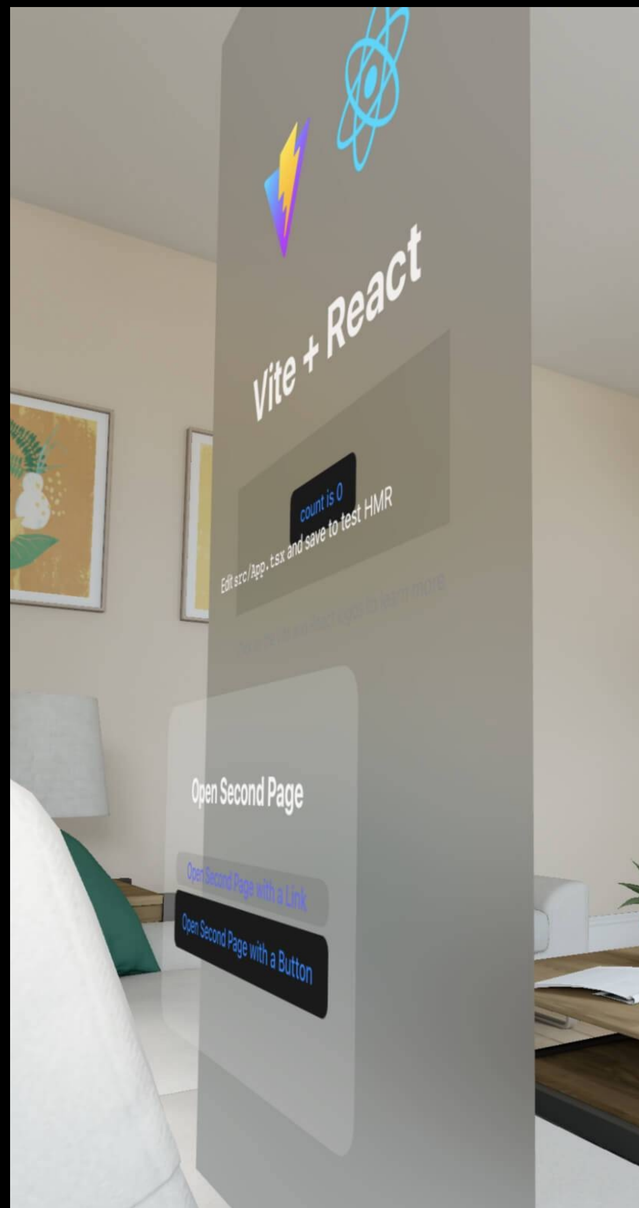
```
html.is-spatial {
  background-color: transparent;
  --xr-background-material: transparent;

  .count-card {
    --xr-background-material: thick;
    position: relative;
  }

  .link-card {
    --xr-background-material: translucent;
    border-radius: 20px;
    position: relative;
    top: 20px;

    a {
      display: block;
      --xr-background-material: thick;
      border-radius: 10px;
    }
  }
}
```

WebSpatial API: Spatialized HTML Elements



Depth Layout

```css
p {
  --xr-background-material: transparent;
  position: absolute;
  bottom: -10px;
  left: 0;
  right: 0;
  --xr-back: 20;
}
```

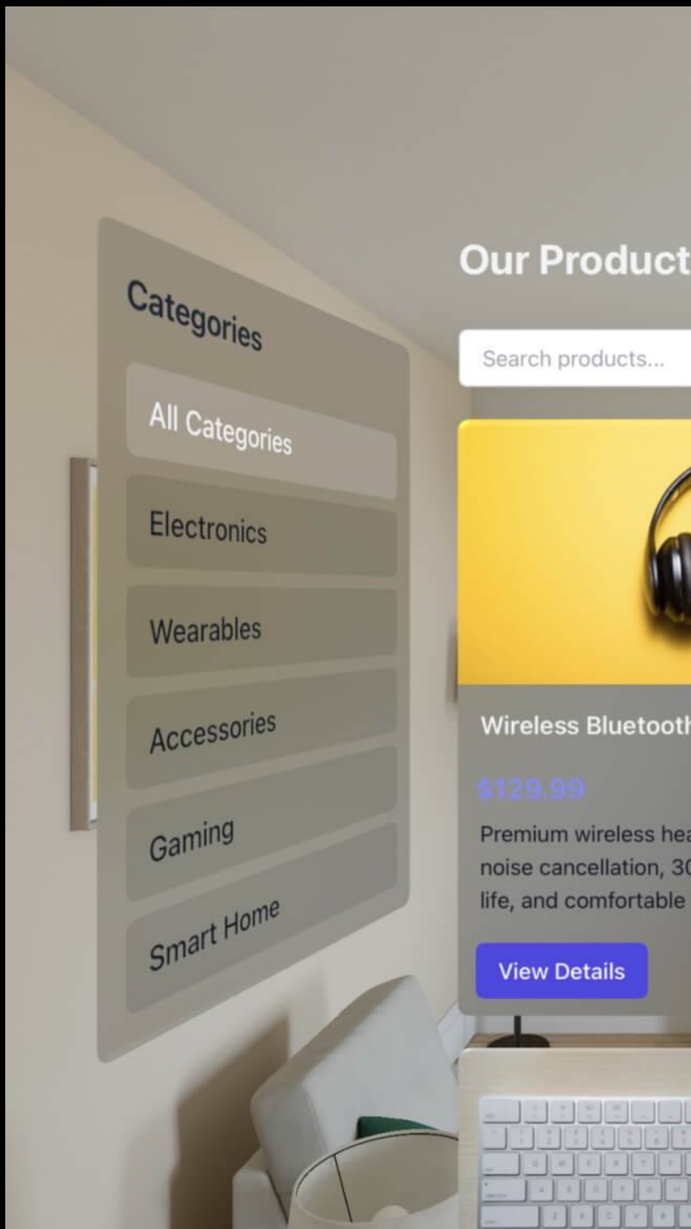


WebSpatial API: Spatialized HTML Elements

Spatial Transform

`transform-style: spatial;`

```
.list-meun {
  position: fixed;
  top: 200px;
  left: 0;
  transform-origin: top left;
  transform: translateZ(320px) rotateY(80deg);
}
```

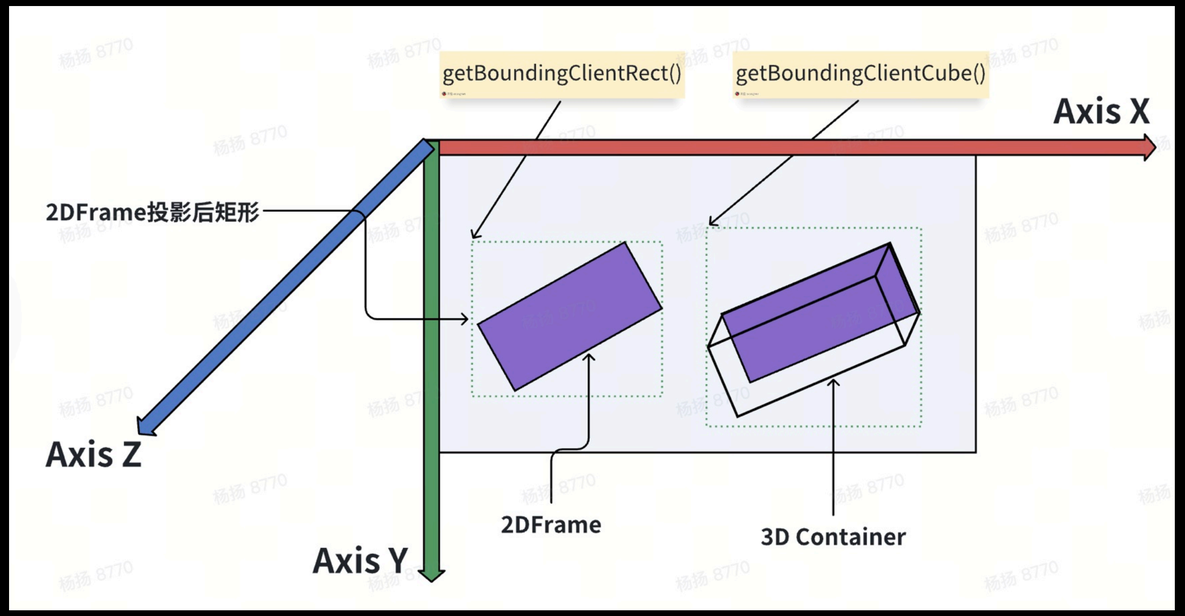


WebSpatial API: Static 3D Container Element

```
<model style="width: 400px; height: 300px" autoplay stagemode="orbit">
    <source src="assets/example.usdz" type="model/vnd.usd+zip">
    <source src="assets/example.glb" type="model/gltf-binary">
</model>
```

WebSpatial API: Static 3D Container Element

- `depth`
- `clientDepth`
- `getBoundingClientRect()`
- `getBoundingClientCube()`



WebSpatial API: Dynamic 3D Container Element

```
<reality style="width: 100%; height: 50px; --xr-depth: 200px; position: relative; --xr-back: 50px;">
  <asset id="exampleModel">
    <source type="model/vnd.usdz+zip" src="/models/example.usdz" />
    <source  type="model/gltf-binary" src="/models/example.glb" />
  </asset>
  <asset id="albedoTexture">
    <source type="image/avif" srcset="/textures/wood/albedo.avif" />
    <source type="image/webp" srcset="/textures/wood/albedo.webp" />
  </asset>
  <material id="exampleMat" type="unlit" texture="albedoTexture" />
  <div attachment="info-panel"><p>Hello, World!</p></div>
  <button attachment="my-button">Click me</button>
  <scenegraph>
    <entity>
      <entity model="exampleModel" material="exampleMat" position="0 0 0" rotation="0 0 0" />
      <box position="0 0 0" rotation="0 0 0">
        <box position="0 0 0" rotation="0 0 0" />
        <sphere position="0 0 0" rotation="0 0 0" />
      </box>
    </entity>
    <plane position="0 0 0" rotation="0 0 0" />
    <cone position="0 0 0" rotation="0 0 0" />
    <cylinder position="0 0 0" rotation="0 0 0" />
    <attachment name="info-panel" position="0 0 0" rotation="0 0 0" />
  </scenegraph>
</reality>
```

WebSpatial API: Spatial Events

- `spatialclick` - event.location3D
- `spatialdrag`, `spatialdragstart`, `spatialdragend`
- `spatialrotate`, `spatialrotatestart`, `spatialrotateend`
- `spatialmagnify`, `spatialmagnifystart`, `spatialmagnifyend`
- `spatialevent` - event.location3D

WebSpatial API: Spatial Scene (Window)

Starting Spatial Scene

Web App Manifest

New Spatial Scene

- `<a href={newSceneUrl} target="_blank">`
- `<a href={newSceneUrl} target="newSceneName">`
- `window.open(newSceneUrl);`
- `window.open(newSceneUrl, "newSceneName");`

WebSpatial API: Spatial Scene Initialization

- `type` - 'window' | 'volume' | ‘stage'
- `defaultSize` - { width, height, depth }
  - `window.innerDepth`, `window.outerDepth`
- `resizability`
- `worldScaling` - 'dynamic' | 'dynamic'
- `worldAlignment` - 'dynamic' |’gravityAligned'

WebSpatial SDK: Real-world Practice in HTML/CSS-based Declarative 2D Web Frameworks

```
  import { defineConfig } from "vite";
  import react from "@vitejs/plugin-react";
+ import webSpatial from "@webspatial/vite-plugin";
+ import { createHtmlPlugin } from "vite-plugin-html";

  // https://vite.dev/config/
  export default defineConfig({
    plugins: [
      react(),
+     webSpatial(),
+     createHtmlPlugin({
+       inject: {
+         data: {
+           XR_ENV: process.env.XR_ENV,
+         },
+       },
+     }),
    ],
```

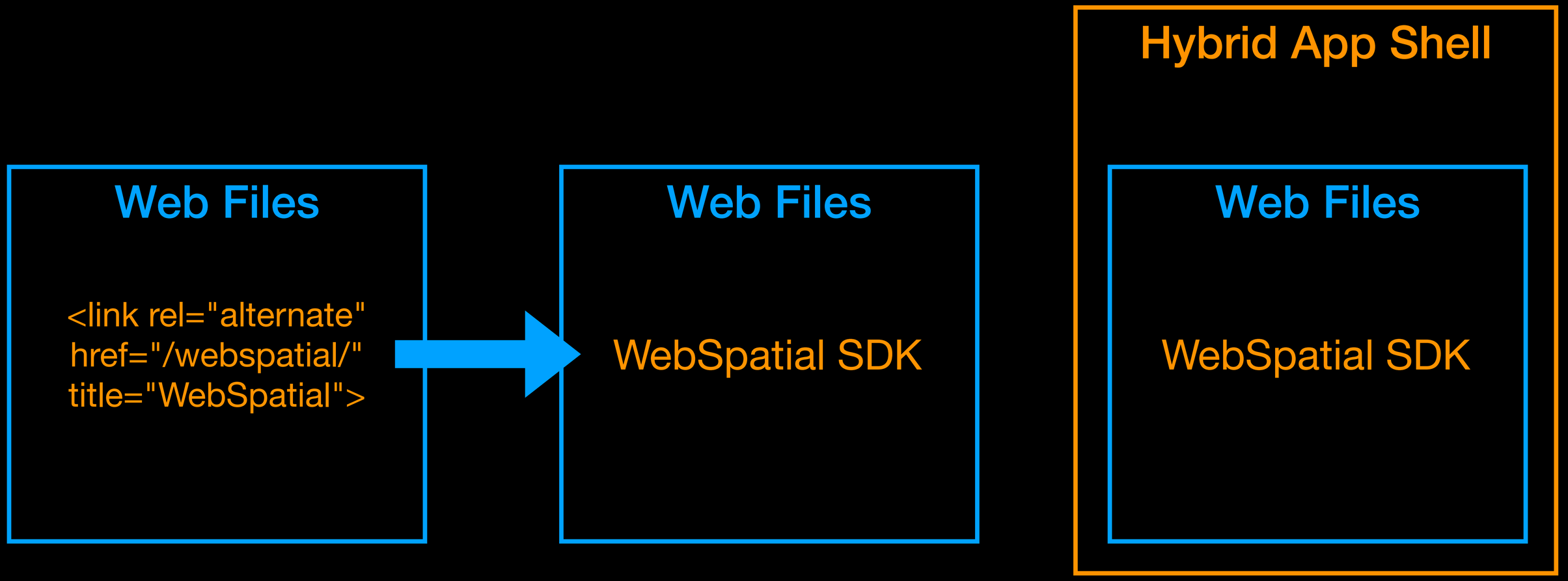
npm install @webspatial/react-sdk @webspatial/core-sdk
npm install -D @webspatial/vite-plugin
npm install -D @webspatial/builder @webspatial/platform-visionos

WebSpatial SDK: Build Output

Website/PWA
for Desktop/Mobile

Website/PWA
for WebSpatial

Packaged
WebSpatial App



WebSpatial Runtime: What's Available

- Hybrid App Shell: visionOS ✅, Android XR ⌛, …
- Spatial Web Runtime: PICO ⌛, Rokid ⌛, …
- Simulator: Desktop Chrome⌛

https://webspatial.dev/