AltspaceVR

Synchronizing a World browser

Pavilion presentation guide with complete script attached

This guide describes the approach used in the Pavilion. The Host can choose a page, scroll, reload, or blank the screen. Each visitor has a separate BS Browser, so the script shares presentation state through the World and applies it to each visitor's browser.

What the World needs

In Unity, place a BS Browser on the screen and give its GameObject a known name. Add a screen anchor GameObject for the controls. An optional image surface can cover the screen when no page is selected. Put the JavaScript in the World's `WebRoot/script.js` and load it from `WebRoot/index.html`. The control panel is created at runtime under the screen anchor.

Replace the bracketed values

`[WORLD_SLUG]` - a unique short identifier for this World's state key and messages.
`[SCREEN_ANCHOR_NAME]` - the Unity screen anchor's exact name.
`[BROWSER_OBJECT_NAME]` - the exact name of the GameObject with the BS Browser.
`[OPTIONAL_PLACEHOLDER_NAME]` - an image surface name; use a unique unused name if there is none.

Adjust the panel's local position and scale for your screen. Set a starting URL on the BS Browser component if desired. The bracketed values are in quoted strings, so the example parses before replacement, but it will not find your World objects until you replace them.

How the sync works

1. Host request. Open validates the typed URL and writes the requested page, scroll position, blank state, and revision into protected space state. The Host must pass the SDK's `isSpaceAdmin` permission check and have permission to write protected state.

2. Confirmed state. The panel shows a pending message, then every client listens for `space-state-changed`. Only the confirmed state is applied to the screen. A late visitor reads the current saved state on entry.

3. Browser interaction. The Host's browser reports page navigation and scrolling. The script publishes those changes so followers can update their own browsers. Click coordinates are sent as one-shot events and replayed when the follower is on the same URL.

What has been tested

Several Altspace participants joined the public Pavilion World and confirmed that the browser presentation synced live. The Host opened a new URL, saw Presenting, and navigated the site; Clear URL, Reload, Scroll, Top, and Blank or Show worked. Interaction replay and visitor control restrictions were not separately recorded. Each browser loads separately; some sites may require sign-in or refuse embedding.



AI with Christine Jump | Practical AI
christinejump.com
/in/christinejump
@aiwithchristinejump
christinefj
AI with Christine Jump
@AIwithChristineJump

Download

AltspaceVR

The script sections that make it work

These excerpts explain the flow; the attached JavaScript is the complete copyable script.

Find the World objects and check the Host

The script looks up the named screen anchor and BS Browser. The panel is created under that anchor at runtime. The Host check uses the SDK's exact `isSpaceAdmin` property.

```
const anchor = await scene.Find("[SCREEN_ANCHOR_NAME]");
const browserObject = await scene.Find("[BROWSER_OBJECT_NAME]");
const browser = browserObject?.GetComponent(BS.CT.Browser);
const host = () => scene.localUser?.isSpaceAdmin === true;
```

Write the requested presentation state

Open validates the address, then `publish` writes protected space state. It does not treat the Host's click as proof that the World accepted the change.

```
const KEY = "[WORLD_SLUG].presentation.v1";
pendingState = { ...(pendingState || state), ...patch };
statusText("Syncing presentation...");
scene.SetProtectedSpaceProps({
    [KEY]: JSON.stringify(pendingState)
});
```

Apply only confirmed updates

Each visitor listens for the protected state change. The full script validates the payload, updates that visitor's BS Browser URL or scroll position, and displays errors when a write is rejected.

```
scene.On("space-state-changed", event => {
    for (const change of event.detail.changes || []) {
        if (change.property !== KEY || change.isPublic === true ||
            change.type === "public") continue;
        const next = JSON.parse(change.newValue);
        apply(next);
    }
});
```

Connect the panel and browser interaction

The text field tracks what the Host types. Open publishes that address. The full script also relays page navigation and scrolling, and sends clicks as one-shot events for followers on the same URL.

```
field.OnChange(event => {
    draftUrl = String(event.value ?? field.value ?? "");
});
button("Open", () => publish({
    url: webUrl(draftUrl), scroll: 0, blank: false,
    revision: String(Date.now())
}));
```




AI with Christine Jump | Practical AI
christinejump.com
/in/christinejump
@aiwithchristinejump
christinefj
AI with Christine Jump
@AIwithChristineJump

Download