

Gruesøme's Arcade – Embedding Guide (v1.4)

Built: 2026-01-07

sig(b64)=YnVpbHQgYnkgZ3J1ZXPDuG1l

## Goal

Embed any game inside the Arcade shell safely and consistently, while supporting:

- Wallet sync (credits + membership + avatar)
- Run request/grant + run result submission
- Metrics v3.x catalog + leaderboard v2 boards

Runtime files (already shipped in repo)

- /public/\_lib/arcadeGameEmbedAdapter.js
- /public/\_lib/runCoordinator.js



### 1) Iframe Embed Contract



Recommended iframe shape:

```
<iframe
  id="gameFrame"
  src="/games/<gameId>/index.html"
  sandbox="allow-scripts allow-pointer-lock"
  allow="fullscreen"
  referrerpolicy="no-referrer"
></iframe>
```

Sandbox guidance (important)

- Default: do NOT include allow-same-origin.
  - It removes Chrome warnings and reduces escape risk.
- Only add allow-same-origin if the game **\*must\*** use same-origin storage/cookies.
- Do not add allow-top-navigation unless explicitly needed.
- Use postMessage for all communication.

If you must allow popups (wallet sites, help pages):

- add allow-popups (and consider allow-popups-to-escape-sandbox only if needed)



### 2) postMessage: Types + Payloads



Standard message types (string):

- ARCADE:READY
- ARCADE:SYNC
- ARCADE:REQUEST\_RUN
- ARCADE:RUN\_GRANTED
- ARCADE:RUN\_DENIED
- ARCADE:RUN\_RESULT

Parent should validate origin:

- Maintain an allowList per game (recommended).
- Reject messages from unexpected origins.

Game should also validate origin where possible:

- If you embed cross-origin games later, lock targetOrigin.

3) SYNC State + the "getSync()" Compatibility Layer

A lot of older "overlay UI" code calls `getSync()`.

If you see:

ReferenceError: getSync is not defined

Fix in the game iframe:

- Define `window.getSync()` and store the last SYNC payload.

Minimal patch (safe):

```
window.__ARCADE_SYNC = null;
window.getSync = () => window.__ARCADE_SYNC;

window.addEventListener('message', (ev) => {
  const msg = ev?.data;
  if (msg?.type === 'ARCADE:SYNC') {
    window.__ARCADE_SYNC = msg.payload || null;
  }
});
```

4) Static Hosting + Vercel Rewrites (No "JS served as HTML")

If module scripts fail to load with MIME type text/html, your SPA fallback is catching asset requests.

Keep rewrite order:

- Specific prefixes first (/api, /games, /assets, /vendor, etc)
  - Then a "static file passthrough" rule for anything containing a dot
  - Then the SPA catch-all LAST → /index.html

Example static passthrough:

```
source: "/*.*\..*")  
destination: "/$1"
```

#### 5) CSP: Comet/Perplexity Font Noise (Optional)

If you use the Comet browser and it injects:

<https://r2cdn.perplexity.ai/fonts/FKGroteskNeue.woff2>

Your CSP may block it and log warnings. This does not break the app, but it's noisy.

If you want to allow it:

- Add <https://r2cdn.perplexity.ai> to font-src in vercel.json CSP.

### Example:

font-src 'self' https://fonts.gstatic.com https://cdn.jsdelivr.net https://r2cdn.perplexity.ai

## Recommendation

- Keep CSP strict in production if you do not need that font.
  - If you want a clean console while using Comet, allow it explicitly.

## 6) WebGL Stability

If you see:

- "Too many active WebGL contexts"
- "THREE.WebGLRenderer: Context Lost"
- "Framebuffer is incomplete: Attachment has zero size"

You are likely creating THREE renderers repeatedly (SPA view switches, multiple iframes) without

Requirements for any Three.js feature:

- Only one active renderer per page/iframe.
- On unmount:
  - cancelAnimationFrame
  - renderer.dispose()
  - renderer.forceContextLoss()
  - dispose geometries/materials/textures
- Do not render when canvas has 0x0 size; delay until visible, and handle resize.

End.