Two Builds Plus a Done-For-You Path

Drop a Vibe-Coded Game Into Storyline or Rise

Rachel Weiss · E.M. Designs · emdesigns.ai *Canadian eLearning Conference 2026*

The Big Idea

There are two ways to **build** a portable web object game and drop it into Storyline or Rise, and one **done-for-you** path that skips the build entirely. Pick the one that fits where you are right now.

	Build 1: Lovable Folder	Build 2: The Deeper Path	Bonus: PWO Generator
Who does the work	You, in Lovable	You, in Figma/Base44 + Replit	I do it. You drop in questions
Output	A self-contained folder	A self-contained folder	A hosted link
Works offline	Yes	Yes	No (needs internet)
Edit questions later	Edit `questions.csv` in the folder	Edit `questions.csv` in the folder	Edit in the generator, link updates live
Best for	Fast, self-contained games	Maximum control, locked-down LMSs	"I just need a working game today"

The one distinction that explains all three: Storyline and Rise can take a web object as a **folder** (it copies the whole game *inside* your published course) or as a **URL** (it points to a live website at runtime). The two builds give you a folder. The Bonus gives you a hosted link, done right.

Why a Folder Usually Beats a Link

Both forms are valid. A folder wins when:

1. **Corporate LMS firewalls** block outside websites. A folder lives *inside* the course package, so there is nothing external to block.
2. **Permanence matters.** A folder is yours forever. A link depends on someone else's hosting.

3. **Pixel-perfect sizing.** When you own the files, you control how the game fits the frame.

A link wins when:

1. **You're not the one updating questions.** With the PWO Generator, anyone you trust can change questions through a friendly UI, and the link stays the same.
2. **You don't want to babysit a build.** A hosted link from a tool I maintain doesn't break when Lovable changes its defaults next week.

Build 1: The Lovable Folder *(Build with me)*

Best for: a fast, self-contained game built in the session, no Replit required.

The key thing: **Lovable can build and package the folder for you.** You don't need a separate build environment. You ask Lovable for the embeddable build and it hands you a folder with `index.html`, an `assets/` folder, and `questions.csv`, all using relative paths so it works inside a web object.

1. Build your game in **Lovable** using the Lovable Game Prompt from your toolkit.
2. The prompt tells Lovable to write the build to `web-object/`, remove that folder from `.gitignore`, and commit it. That's important: Lovable's default `dist/` is gitignored on the free plan, so it never makes it into the "Download codebase" ZIP.
3. Click **Download codebase** at the bottom-left of the Lovable file tree. You get a ZIP of the whole project.
4. Unzip it. The `web-object/` folder is your portable build. `web-object-rise.zip` is the Rise upload.
5. **Storyline:** Insert → Web Object → point at the unzipped `web-object/` folder.
6. **Rise:** add a **Code Block** → **Upload project**, upload `web-object-rise.zip` directly.
7. **Publish to test** (Storyline → Review 360; Rise → Preview/Review).

> **The Rise "red index" thing.** Rise's Code Block expects `index.html` at the very top of the zip. The Lovable Game Prompt produces `web-object-rise.zip` with `index.html` already at the root, so you won't see the red error. If you ever zip the *folder* instead of its *contents*, you will.

In short: *"A fast build that lives inside your course."*

> **One thing to know.** New Lovable projects default to server-side rendering, which is not a droppable static folder. The Lovable Game Prompt overrides that. It also tells Lovable to build to `web-object/` instead of `dist/`, and to commit it, so the free plan's "Download codebase" button actually includes it.

Build 2: The Deeper Path *(Reference)*

Best for: anything you're shipping to a real corporate LMS, or when you want to own and edit the source code directly.

1. Design or generate the game with **Figma Make** or **Base44** (use the Tetris Game Prompt as your starting point).
2. Import the project into **Replit** (`Create Repl → Import from GitHub`).
3. Paste the **Converter Prompt** to make it Storyline-ready (relative paths, single-file, external `questions.csv`).
4. Run `npm run build:storyline` and confirm the `dist-storyline/` folder appears.
5. **Storyline:** Insert → Web Object → point at the folder. **Rise: Code Block → Upload project**, upload the zip, keep `index.html` at the top level of the zip.
6. **Publish to test.**

In short: *"More code control. Same final folder."*

So why use this path at all, if Lovable can hand you a folder?

Two honest reasons, and neither is "the file is better." A folder from Lovable and one built in Replit are basically the same kind of thing. The difference is **who runs the build, and how much control you have:**

1. **Some tools only give you code, not a finished build.** Figma Make and Base44 generate the game's code but don't compile or package it. To turn that code into a droppable folder you need somewhere to build it. **Replit is a free, browser-based code workspace** where you run the build and out comes the `dist` folder, no installs.
2. **More control when you want to fine-tune.** With the source code in front of you, you can adjust things directly instead of re-prompting and hoping.

Bottom line: Lovable is faster because it builds for you. The Figma-to-Replit path gives you more hands-on control and works with tools that only hand you code. Pick the one that fits the job.

Bonus: The PWO Generator *(Done for you)*

Best for: you don't want to build anything. You want a working game in your course today.

The PWO Generator is the tool I built to give you the link method *done right*. You pick a game, drop in your questions, and get a hosted link that goes straight into Storyline's Web Object or Rise's Embed block. I host it. You don't touch code. When your questions need to change, you edit them in the generator and the link updates live.

1. Go to **emdesigns.ai/pwo-generator**.
2. Sign up for a free account.
3. Pick a game.
4. Paste your questions (Manual, AI Paste, CSV, or Word doc).
5. Hit Generate. Copy the hosted link.
6. **Storyline:** Insert → Web Object → Web Address → paste the link. **Rise:** Embed block → paste the link.
7. **Publish to test.**

Free accounts include one generation per month. Pro is $14.99/mo, removes the limit, and adds LMS scoring.

In short: *"The link method, done right. Hosted by me. Editable forever."*

Storyline vs. Rise: how each takes a game

All three paths work in both tools, but the mechanism differs and Rise has two very different blocks people mix up.

Storyline has one path: **Insert** → **Web Object** and point at the *unzipped* folder (for the two builds) or paste the URL (for the Bonus). Storyline packages folder-based games into the published course (offline-safe).

Rise has two different blocks, and the choice matters:

- **Embed block = the link path.** Use it for the Bonus (a hosted URL). It needs internet.
- **Code Block = the packaged path** (the real equivalent of a Storyline web object). Choose **Upload project** and give it the *zip*. This is the one for the folder builds.

Three Rise Code Block specifics worth knowing:

1. **Zip, not folder**, and `index.html` must be at the zip's top level (not inside `dist/`), or you get the red "index" error.

2. **Standard Articulate 360 AI license required.** Code Block isn't on every plan.
3. **It can report completion.** Add `window.parent.postMessage({ type: 'complete' }, '*')` when the game ends and Rise's continue blocks will mark the activity complete.

Three Things Everyone Forgets

- **Preview won't show web objects.** You must publish (Storyline → Review 360; Rise → Preview/Review) to test any of these paths.
- **For folder builds, `index.html` must be at the root** of the folder (Storyline) or the zip (Rise's Code Block).
- **The CSV format depends on how the game was built.** The Figma/Replit converter uses `id,question,option1...option4,correctAnswer` with 0-based answers. A Lovable-built game may use a different schema. Pick one format per game so the "edit the CSV" rules line up.

A Few Terms, in Plain English

- **Vibe coding:** describing what you want in plain language and letting an AI write the code. You're directing, not programming.
- **IDE (like Replit):** a workspace where you can run and edit a code project, right in your browser.
- **GitHub:** an online home for project files.
- **Build / the `dist` (or `web-object`) folder:** "building" turns raw code into a finished, packaged version. The result is the ready-to-use game you drop into Storyline or Rise.
- **Web object:** a self-contained mini web app embedded inside a course.

Quick FAQ

"Can I just make the game in Canva?" Not as a real web object. Canva is a design tool, not a code generator, and it has no true HTML/code export. For a real, ownable, droppable game you need a vibe-coding tool: Lovable, Figma Make, Base44, or the PWO Generator.

"Do I need to know how to code?" No. That's the whole point of vibe coding: you describe what you want, the AI writes it. You're directing, not programming. The PWO Generator skips even that step.

"Will the game's score report back to my LMS?" A web object displays the game; its internal score doesn't pass to the LMS gradebook on its own without extra wiring (xAPI / JavaScript). In Rise, a Code Block can at least mark the lesson complete. PWO Generator's Pro tier includes LMS scoring.

"Storyline or Rise, which should I use?" Both work. In Storyline you Insert → Web Object. In Rise, use the **Code Block** for a packaged game, not the Embed block. (Embed is for hosted links.)

"Do I have to pay for these tools?" Free tiers work for Lovable, Figma Make, Base44, and Replit. The PWO Generator has a free tier (one generation per month) and a Pro tier ($14.99/mo).

"Can I change the questions later without rebuilding?" Yes. With a folder, edit the `questions.csv` next to `index.html`. With the PWO Generator, edit the questions in the tool and the hosted link updates live.

Get all the materials, prompts, and the PWO Generator at: **emdesigns.ai/canadian-elearning-conference**