The Riven Graph, 1: The 1997 Game as a Graph

From individual cards to the world between them

Guillaume Lethuillier
glthr.com

November 15, 2026 · part of the series “The Riven Graph”

Abstract

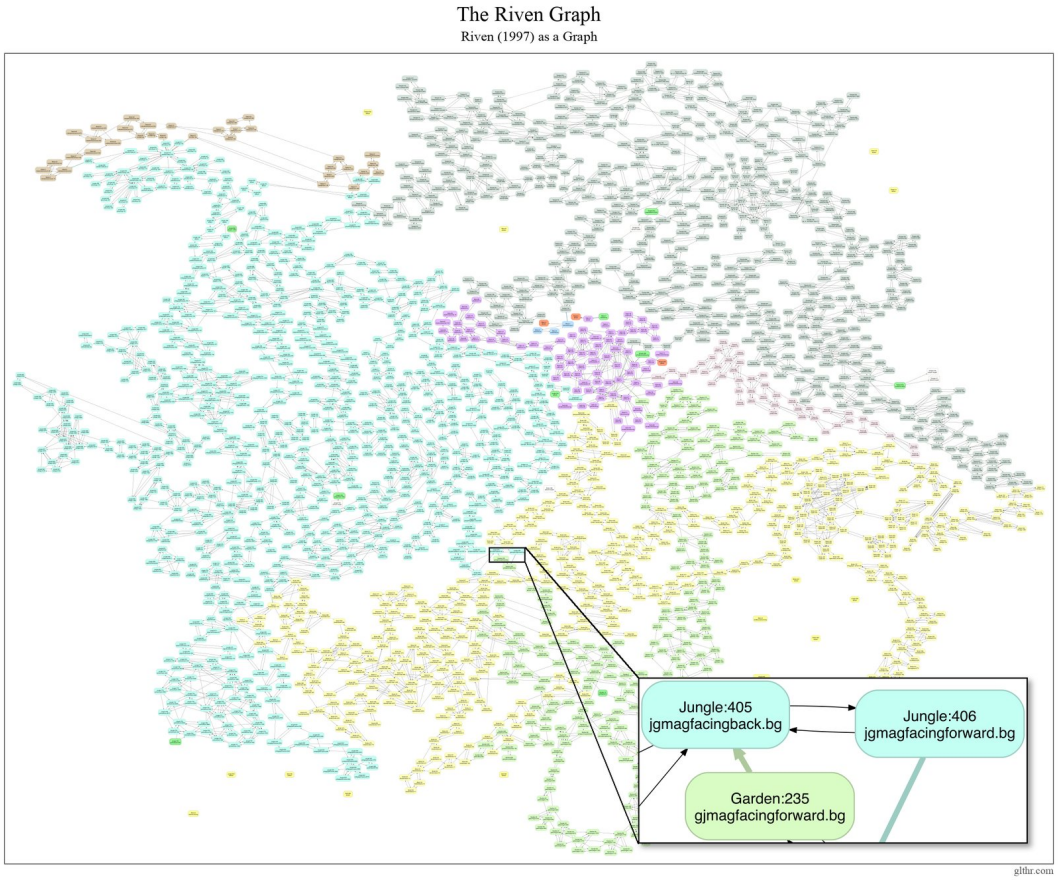
I represent the original Riven (1997) as a directed graph of 2,307 cards and five virtual targets. Its 6,722 typed connections preserve script references, separately attributed native transitions and caller-owned continuations. The default view hides Zip shortcuts and ending resets. This first article explains the graph and its browser view. It describes structural relationships, without executing the puzzle state that determines which ones a player can follow.

When I introduced the Riven Navigator (now the first part of The Riven Diffs), I could inspect the game one card at a time: its images, its hotspots and the binary scripts attached to them. The missing part was the relationship between those cards. I had the places, but not yet the network that joined them.

Riven can now be viewed as a graph, extending the approach I first applied to Myst:

Version of the game: Riven 1.0, the original English five-CD release, September 1997. The card numbers are those of this release.

Cite as: Guillaume Lethuillier, “The Riven Graph, 1: The 1997 Game as a Graph”, glthr.com, November 15, 2026. https://glthr.com/riven-graph-1



The Riven Graph can be **explored in the browser**, where any card can be found by its stack and number.

Disclaimer

This is an unofficial, non-commercial study of Riven (1997), independent of Cyan Worlds. The game artwork remains the work of its creators. The analysis contains spoilers and reads the released data without executing every possible world state. Structural references, inferred engine behaviour and directly compared images are identified separately.

Introduction

Roadmap

(1) In this article, I introduce the Riven Graph and explain its nodes, arrows and colours. How the Graph Was Created then describes how an AI coding agent built the extractor from a short brief and the earlier Myst work.

(2) In the second article, I compare the graph with the game's pictures and scripts: theoretical paths, disconnected routes, an earlier way of linking, a maglev station that is never empty, and the variables behind the easter eggs. The figures use the same card labels, graph excerpts and eye indicators as the Myst findings.

A Card Can Hold Several Views

In the Navigator, I showed cards with many alternative images: a door opened or closed, a bridge raised or lowered, a room turned to face another direction. The Riven Diffs, written with Twitch, examines the visual differences between such renders.

Those alternatives usually remain inside the same graph node. Opening a door may change the image without changing cards. Walking through it can lead to another node. Journal pages and other picture substitutions can likewise change the display without adding a navigation vertex.

The graph answers a particular question: **which cards refer to which other cards?** The scripts, pictures and state changes of a location form a richer structure inside its node.

What Is in This Graph?

Its eight stacks contain the following cards:

Stack	Internal name	Shipped cards
Menu and books	`aspit`	8
Boiler Island	`bspit`	498
Garden Island	`gspit`	267
Jungle Island	`jspit`	814
Gehn's Age	`ospit`	89
Prison Island	`pspit`	47
Tay / Rebel Age	`rspit`	37
Temple Island	`tspit`	547
Total		**2,307**

Five virtual targets bring the total to **2,312 nodes**. Stack membership is an attribute used for colours and filters. Stacks have no separate vertices here.

Basic Properties of the Riven Graph

On the web, card labels such as Temple:155 have a thick underline, in the color of their stack, and open that card in the graph browser.

Nodes and Edges

The graph has 2,312 nodes: the 2,307 cards of the game and the five virtual targets described below. Its 6,722 typed connections are summarized in the table of connections. The default drawing shows 6,112 of them.

Paths

From the first view of the game (Temple:155), the telescope that ends it (Temple:137) is 3 connections away. In Myst, the same distance was 24. The two most distant cards of the graph are 125 connections apart (130 in Myst). Zip shortcuts, ending resets and automatic next steps are excluded from these distances. As in Myst, a path is a distance between views and says nothing about the puzzles that stand on it. The second article gives the routes.

Clusters

Stacks are not clusters here, and yet the drawing groups cards by island, as it did for Myst's Ages. This is explainable: *only 27 pairs of cards belong to two different stacks.* Counting the isolated nodes, the graph has 24 weakly connected components, but one of them holds 2,287 of the 2,312 nodes, and 2,258 cards can all reach one another. The first view reaches 2,262 nodes. The 50 that it does not reach are 46 cards, mostly the menu, the journals and the black cards that the engine skips, and four of the five virtual targets.

Legend: How to Read the Graph

The browser graph uses a card's stack for its colour, its references for its connections, and distinct borders or arrowheads for special cases. The following legend explains those conventions before the findings put them to use.

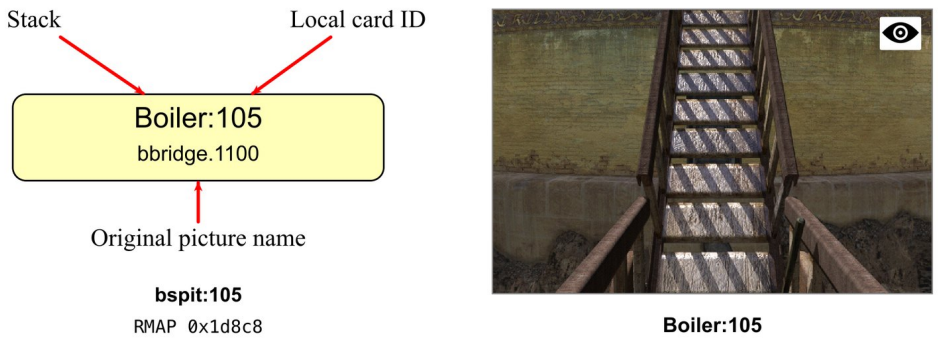
Reading a Node

Labels and Colours

Each stack has its own colour. A label such as **Temple:155** means local card 155 in the Temple stack, whose internal name is `tspit`. This is the gameplay entry reached when a new game begins in this edition. The internal form, `tspit:155`, identifies the same card.

A local number needs its stack: Temple:155 and Jungle:155 are different cards. The browser accepts the displayed name or internal form. Under the label, each box gives the original name of the card's first picture, as the archive stores it without the number of the card: every card has such a picture, whereas **only 202 of the 2,307 cards also have a name of their own**, which the browser exposes.

An `RMAP` table relates local card numbers to identifiers used by stack changes and engine routines. Temple:155 has the RMAP value `0x6e9a`[1], kept in the exported data so that a label can be traced back to the data.



The label of Boiler:105, its stored picture filename and one of its views. The card also has a name of its own, `Ωbsuper2Ω`. The browser key `bspit:105` and RMAP value `0x1d8c8` identify the same card through different parts of the data. The open eye identifies an ordinary card view, following the figure convention in the findings.

Side note: The Ω characters belong to the game's naming convention for Zip places: **all 194 cards flagged as Zip places have names enclosed by Ω**. Visiting

[1] RMAP is the name of a resource type in the game's data files, the Mohawk archives, in which every resource has a four-letter type. Each stack holds one RMAP resource: a table with one 32-bit identifier per card, indexed by the local number of the card. An instruction that changes stack, and some routines of the engine, designate a card by this identifier rather than by its number in the stack.

one registers its full name. When Zip mode is enabled, a shortcut matches its hotspot name against those visited places. The engine uses the Zip flag and complete-name matching. The 606 extracted shortcuts target 189 of these places. Five flagged cards have no matching shortcut reference. No other Greek letter or symbol appears in the extracted card and hotspot names.[2]

Sources, Sinks and Isolated Nodes

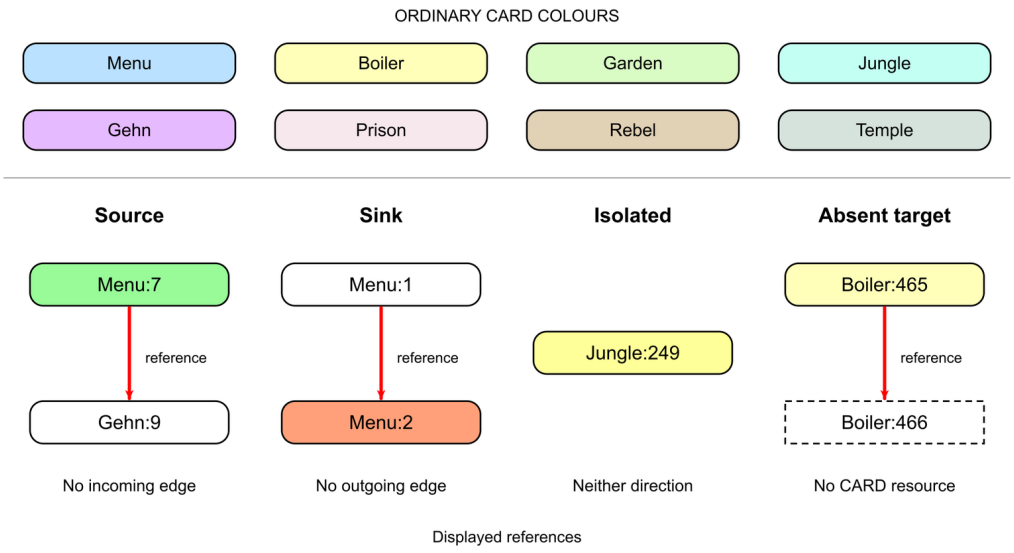
The usual graph terms apply. A **source** has no incoming connection from another node; a **sink** has no outgoing connection to another node; an **isolated** node has neither.

These properties describe a graph projection, without establishing access in the game. A journal can be opened through the inventory and return to its caller. Leaving that mechanism unexpanded in the graph does not make the journal inaccessible.

Virtual Targets

Eight navigation instructions refer to five local card numbers for which this extraction finds no `CARD` resource: **Boiler:466**, **Jungle:740**, **Temple:361**, **Temple:477** and **Temple:478**. I keep those targets as virtual nodes, with a distinct border, rather than discard the references.

The files establish that a script names an absent target, without showing what it once contained, whether it was finished, or why the reference remains. A virtual node preserves a question in the data.

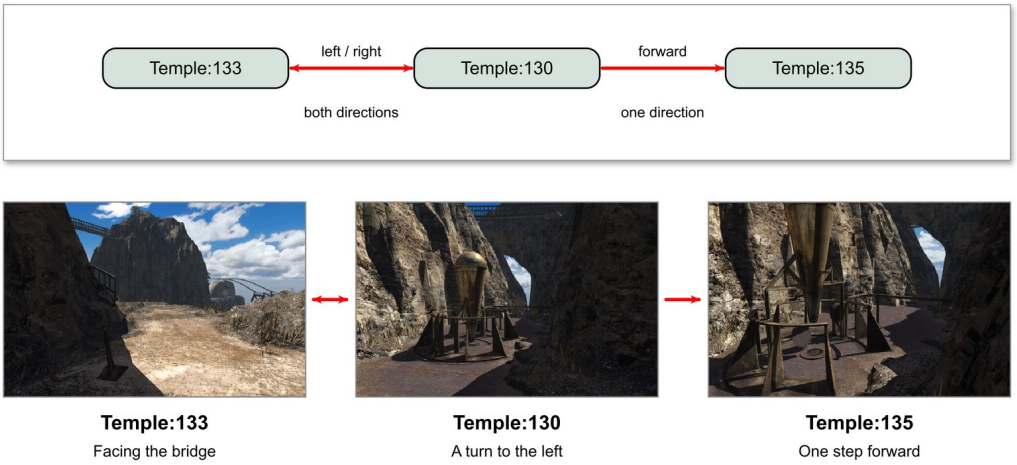


Reading a Connection

An arrow has a source and a destination. If both directions exist, the cards are mutually connected. If only one exists, following it does not imply that the player can simply reverse the move.

[2] RivenX already recognized these delimiters in its 2008 source.

The two directions do not mean the same thing. Turning to the left or to the right changes the point of view, and the game lets you turn back. Walking forward does not let you walk backward. In the graph, 91.4% of the 3,460 pairs of cards joined by a turn have their reverse pair, against 9.2% of the 1,584 pairs joined by a step forward. The Myst Graph does not tell a turn from a step, but over all its movements 55.5% of the pairs of cards have their reverse pair, against 66.4% for Riven.

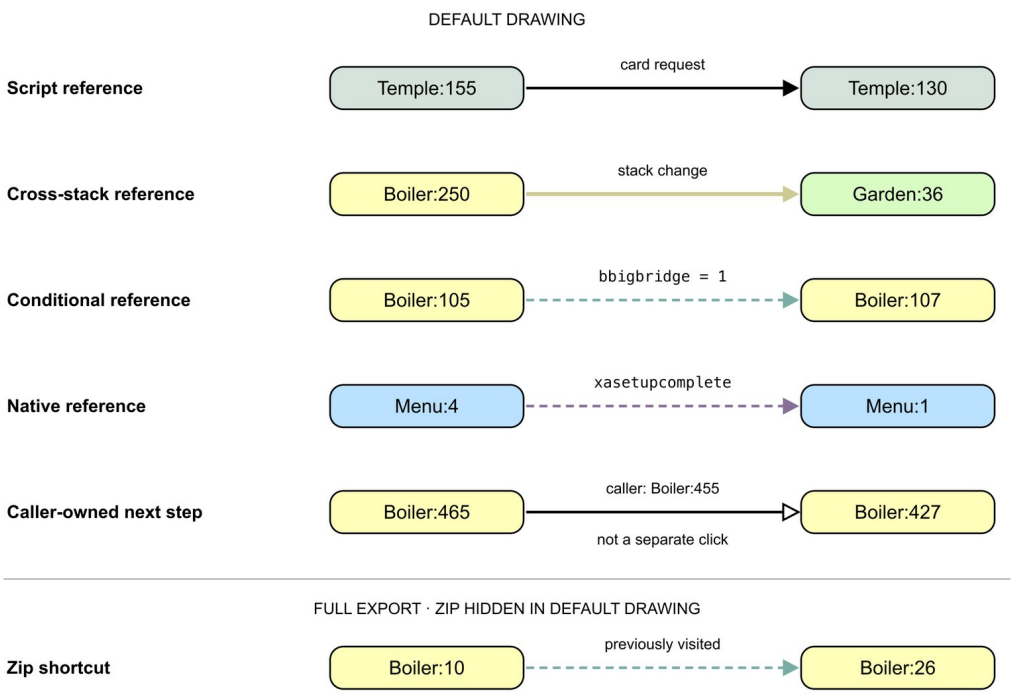


A turn between Temple:133 and Temple:130 goes both ways. The step from Temple:130 to Temple:135 goes one way.

There are several kinds of connection:

Kind	Typed connections	Meaning
Script reference	6,077	A card or hotspot script requests another card or stack.
Native movement or timer	13	Navigation handled by a reviewed engine routine.
Zip shortcut	606	A potential shortcut whose destination must have been visited.
Ending reset	4	A return to the menu after an ending sequence.
Caller-owned continuation	22	One handler enters an intermediate card, then requests the next one.
Total	**6,722**	All exported kinds together.

The default drawing shows **6,112 typed connections**: script references, native movement and caller-owned continuations, with Zip shortcuts and ending resets hidden. Path calculations also exclude the 22 continuations, leaving **6,090 movement records** joining **6,072 unique ordered pairs** of cards.



From top to bottom: Temple:155 → Temple:130, Boiler:250 → Garden:36, Boiler:105 → Boiler:107 (under `bbigbridge = 1`), Menu:4 → Menu:1 (through the engine), Boiler:465 → Boiler:427 (within the handler of Boiler:455), and Boiler:10 → Boiler:26. The hollow step is not an independent click. Zip shortcuts require Zip mode and a previously visited destination. They are hidden in the default drawing.

Several instructions can support the same edge. I merge references only when their source, resolved destination, kind and conditional status agree, while keeping every contributing instruction in its evidence. Fourteen pairs keep separate enabled and conditional records of the same kind. Collapsing those records would conceal a distinction in the controls that support them.

Typed records consequently differ from unique source–destination pairs: the same cards can be joined by different kinds of connection, or by both enabled and conditional references.

Local Conditions Remain Visible

A script can choose a destination by testing a variable. The extractor reads nested branches and keeps their tests beside the connection, with its event, hotspot and byte offset. Across all kinds, **835 connections are conditional**, including all **606 Zip shortcuts** and the **four ending resets**. The default drawing contains **225 conditional connections**.

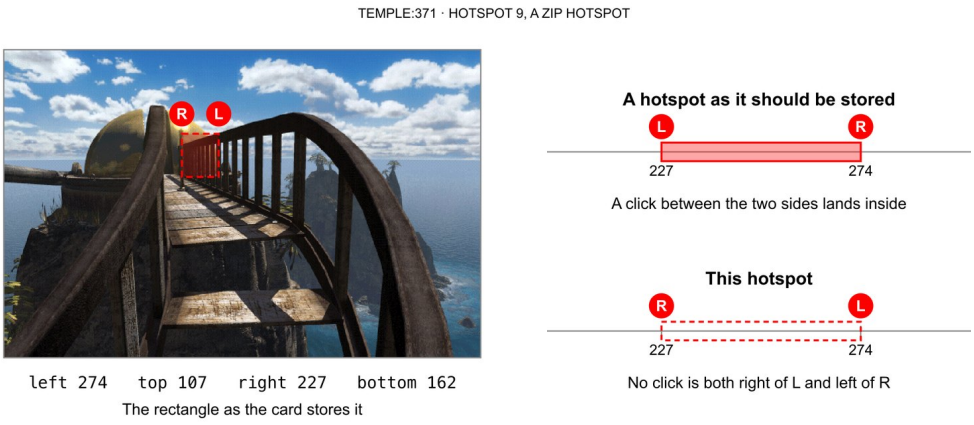
The hotspot audit follows stored activation lists and enabling or disabling instructions, including changes made by a caller after entering another card. It classifies **6,456 hotspots as enabled, 897 as conditional and 1,145 as unknown**. Unknown means that native, deferred or unresolved behaviour prevents a firm conclusion. It does not mean disabled. There are **zero hotspots proven permanently disabled by this activation analysis**. Unresolved activation is not sufficient to mark an edge disabled.

A Zip Hotspot with Reversed Coordinates

The geometry of the controls reveals a peculiar case. A Zip hotspot of Temple:371 has its left side at 274 and its right side at 227: its rectangle is inverted. *It is the*

only one in the game.

Its script leads to Temple:374, the same destination as the ordinary forward hotspot. The reversed coordinates are already in the original 1997 game files. ScummVM replaces them with an empty rectangle when it loads the hotspot.[3] To interpret an arrow, both its script and the geometry of its control matter.



Hollow Arrows for a Continuing Handler

Sometimes one handler requests a card and then, straight away, another: the player clicks, the game shows an intermediate view, then moves on to the next card. The graph keeps both requests as ordinary arrows from the card that owns the handler. It adds a third one, drawn with a **hollow arrowhead**, from the intermediate view to the next card, and records which handler it comes from. There are 22 such continuations. They explain why a view that seems to lead nowhere, an apparent sink, or a view that is only passed through, still has a way out.

An arrow is added only when nothing can happen between the two requests. They must belong to the same handler and the same branch, and only harmless instructions may lie between them: changing the picture, playing a sound, waiting. A stored movie instruction, which delays a command until a movie reaches a given time, is harmless when the delayed command cannot change the card. The 49 such instructions of the game all start a sound. The card that owns the handler and the intermediate card must not run anything that could change the card or call native code that the extractor cannot read, when they are loaded or left. Five pairs of requests are left out for that last reason: those of the handlers of the five dome cards (Boiler:443, Garden:173, Jungle:339, Prison:42 and Temple:386), which each call a routine of the engine.

Path search ignores these arrows: a continuation belongs to its handler and cannot become a shortcut for another card. Saved returns from the inventory remain separate metadata.

[3] Whether Cyan did this deliberately is unknown. In 2010, Matthew Hoops suggested intentional disabling because the Zip hotspot leads to the same card as the ordinary forward movement. That remains a hypothesis. His original explanation.

Some Links Belong to the Engine

Reading the scripts alone misses part of Riven's navigation. For example, the engine has a table of special destination changes: requesting particular cards redirects the player to another stack.

The graph resolves these changes at the requested destination. It keeps the literal target as evidence and records the resolved endpoint, instead of pretending that the placeholder card contains an ordinary outgoing script. Reviewed native routines also supply a few movement and timer connections. These additions are attributed to a pinned ScummVM implementation. They are separate from the bytecode read from the discs.

Inventory returns receive different treatment. Opening a book saves the caller, and closing it returns to that saved place. Connecting the book to every possible caller would introduce shortcuts through the whole world. I keep this mechanism as explicit metadata rather than invent such edges.

How the Graph Was Created

The technical article about the Myst Graph describes a fairly long journey: converting HyperCard stacks, parsing scripts, identifying references, choosing a graph representation, then making a drawing large enough to read.

For Riven, I began by pointing an AI coding agent to that article and DeMystify, together with the other Myst Graph articles, the Riven Diffs, which opens with the Riven Navigator. The five CD images were in the working folder. My request was simply to use those precedents to decompile Riven and generate its graph.

That was enough to get the project moving. The agent inspected the files, researched their format, wrote an extractor, ran it and produced a graph. A follow-up request asked for a local web page in the same style as the Myst one. It could reuse the earlier viewer rather than begin its visual design again.

I could describe the intended result by referring to existing work, provide the original material, and let the agent work through the implementation. The short brief delegated a substantial investigation.

What the Agent Had to Work Out

Riven's files are different from Myst's HyperCard stacks. The extractor needed to read the ISO images, find the Mohawk archives, and identify the resources holding cards, hotspots, names, picture lists and scripts. It also needed to combine patch archives in the order used by the game: selecting the wrong version of a resource can change the data before any graph analysis begins.

The scripts required a binary parser. It reads command counts, arguments and nested switch branches, preserving their resource-relative byte offsets. A stack change needs both a name lookup and RMAP resolution. Treating every number as a local card ID would produce plausible-looking mistakes.

Some navigation lives in native routines rather than in the disc scripts. The agent used ScummVM's Mohawk implementation as a reference for these cases, including special destination redirects and a small set of movement and timer transitions. Those additions are attributed separately in the graph.

The word "decompile" therefore needs a precise scope here. The project decodes the game's resource data and script bytecode into readable structures. It does not recover Cyan's original interpreter source. ScummVM supplies documented engine behaviour for the native portion. It can also contain compatibility fixes that differ from the original executable.

The implementation is integrated into DeMystify. A new Riven importer supplies the existing graph analysis and shared DOT, PDF and browser renderers. The change contains source code, tests and documentation. The original archives and extracted images stay out of the repository.

Conclusion

I find the graph most useful at two scales: the complete network, where clusters and outliers suggest questions, and the individual card, where its scripts can support or reject an interpretation.

In the next article, I begin with the graph's less obvious connections and absences. They provide another way of examining Riven alongside the visual comparisons of the Navigator: studying both the contents of a place and its connections to other places.

How to cite this article

> Guillaume Lethuillier, "The Riven Graph, 1: The 1997 Game as a Graph", glthr.com, November 15, 2026.
> https://glthr.com/riven-graph-1

For BibTeX:

```
@misc{lethuillier2026riven,
  author = {Lethuillier, Guillaume},
  title  = {The Riven Graph, 1: The 1997 Game as a Graph},
  year   = {2026},
  month  = nov,
  url    = {https://glthr.com/riven-graph-1},
  note   = {glthr.com}
}
```

This document was typeset with LaTeX from the article published at https://glthr.com/riven-graph-1, which remains the reference version.