The Revelation Graph, 1: The 2004 Game as a Graph

From a graph that is drawn to a graph that has a shape

Guillaume Lethuillier
glthr.com

November 21, 2026 · part of the series “The Revelation Graph”

> **Work in progress — November 21, 2026**
> I am still working in **Myst IV: Revelation**. I will update these articles over the coming weeks, so please feel free to come back later to see what’s new.

Abstract

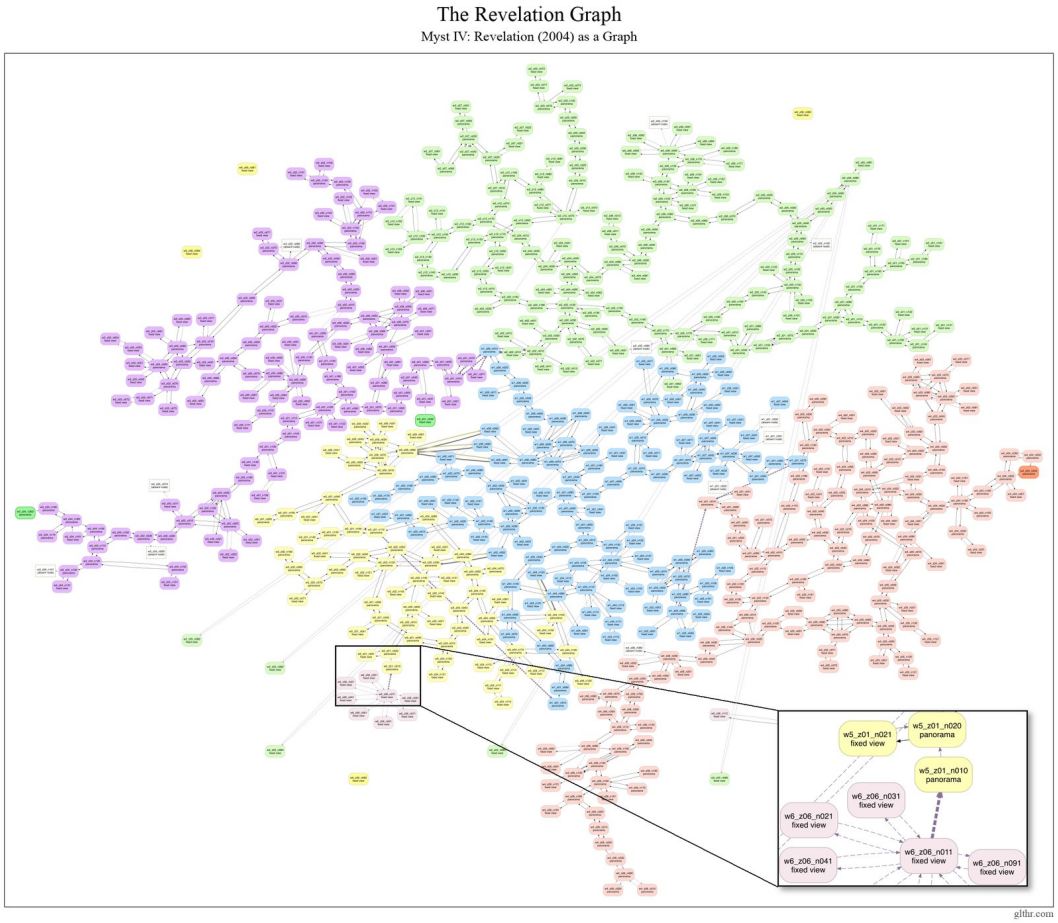
I represent the original Myst IV: Revelation (2004) as a directed graph of 618 stored nodes and 11 virtual targets. Its 1,355 typed connections combine readable commands, class parameters, constants in compiled code and 63 inferred connections, which are marked as such. For the first time in the series, the game also stores where each node stands, so the graph can be drawn as a map in three dimensions. This first article explains the graph, its browser view and its limits. It describes structural relationships, without executing the state that determines which ones a player can follow.

After Myst, Riven and Exile, the fourth game was not an obvious candidate. Revelation was made by Ubisoft Montreal, with an engine of its own.

Revelation can nevertheless be viewed as a graph:

Version of the game: Myst IV: Revelation 1.0, the original English release on two DVDs, September 2004. The node names are those of this release.

Cite as: Guillaume Lethuillier, “The Revelation Graph, 1: The 2004 Game as a Graph”, glthr.com, November 21, 2026. https://glthr.com/revelation-graph-1



The Revelation Graph can be **explored in the browser**, where any node can be found by its name, and where the same nodes can be seen as a map.

Disclaimer

This is an unofficial, non-commercial study of Myst IV: Revelation (2004), independent of Cyan Worlds and Ubisoft. The game artwork remains the work of its creators. The analysis contains spoilers and reads the released data without executing every possible world state. Stored references, parameters of compiled classes, constants read in compiled code and inferred connections are identified separately.

Introduction

Roadmap

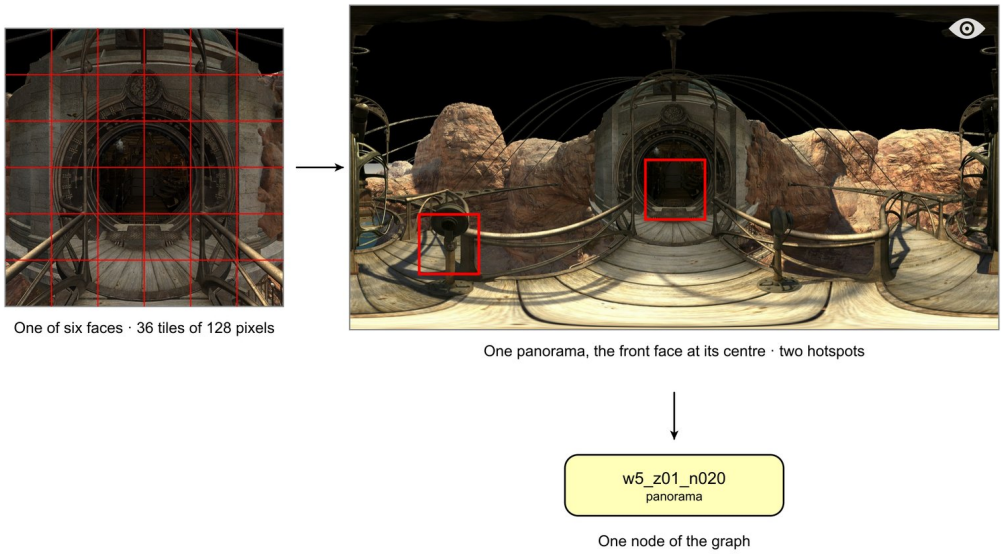
(1) In this article, I introduce the Revelation Graph and explain its nodes, arrows and colors. Archives Inside Archives describes where the game keeps its navigation.

(2) In the second article, I compare the graph with the game's pictures and positions: theoretical paths, the one passage from day to night, nodes that stand close and are far apart, eleven targets that are named and not stored, and thirty nodes without a picture. The figures use the same node labels, graph excerpts and eye indicators as the Myst, Riven and Exile findings.

One Archive per Node

Like Exile, Revelation is a world of places: the player stands in a **node** and looks all around. Turning loads nothing. A node is either a **panorama**, seen in every direction, or a **fixed view**, a close-up of a drawer, a lever or a page that the player cannot turn in.

The game stores each node as one archive. Inside it are the picture, the hotspots, the sounds, the movies and the instructions of that node, and nothing of another node. The picture of a panorama is a cube of six faces, and each face is cut into 36 tiles of 128 pixels. A tile that is entirely transparent is not stored: the sky is drawn behind the cube by other means.



A node of Tomahna as the game stores it (six faces of 36 tiles), as this study displays it (one panorama, the front face at its center) and as the graph draws it (one box).

Revelation has **618 nodes**: 382 panoramas and 236 fixed views. Exile has 820 places and Riven 2,307 cards. The figure is small for a game on two DVDs, and the reason is the same as in Exile: a node is a whole sphere, and here a sphere over which movies, water and moving light are laid.

What Is in This Graph?

The game names a node by three numbers: a world, a zone and a node. `w5_z01_n020` is node 20 of zone 1 of world 5. The worlds are the Ages, with two particularities. Tomahna is stored twice, by day and by night. And the menu is a world of its own, whose nodes are its pages.

World	Age	Zones	Nodes	Panoramas	Fixed views
`w5`	Tomahna, by day	4	67	43	24
`w1`	Tomahna, by night	8	126	72	54
`w2`	Haven	8	154	97	57
`w3`	Spire	6	120	69	51
`w4`	Serenia	6	136	101	35

World	Age	Zones	Nodes	Panoramas	Fixed views
`w6`	Menu	2	15	0	15
Total			**618**	**382**	**236**

Eleven virtual targets bring the total to 629 nodes. World and zone are attributes used for labels, colors and filters: they have no vertices of their own.

The node numbers carry a convention that the data never states and, outside the menu, always follows. Every panorama has a number that ends in 0, and no fixed view does: the fixed views of a panorama take the numbers that follow it, and `w1_z01_n021` is a close-up of `w1_z01_n020`.

The names are the game's own, and one of them even decides where it opens: changing the first line of the settings file `data.dsc` starts the game directly in any stored node, as shown in Visions the Amulet Never Showed.

Basic Properties of the Revelation Graph

Node labels such as w5_z01_n020 have a thick underline, in the color of their Age, and open that node in the graph browser.

Nodes and Edges

The graph has 629 nodes: 618 stored nodes and the 11 virtual targets described below. Its 1,355 typed connections are summarized in the table of connections.

Paths

A new game begins at w5_z01_n010, in Tomahna by day. The first node of the night, w1_z01_n310, is 18 connections away. The arrival in Serenia, w4_z01_n010, is 32 connections away, and the arrivals in Haven and in Spire are 40 away. The two most distant nodes of the graph are 87 connections apart (189 in Exile, 125 in Riven, 130 in Myst). As in the earlier games, a path is a distance between nodes and says nothing about the puzzles that stand on it. The second article gives the routes.

Clusters

The drawing groups nodes by Age, and the grouping is nearly exact: only 20 connections join two worlds. Tomahna by night sits in the middle. Haven and Spire are entered from one of its rooms and return to it, and Serenia is reached from it. Tomahna by day leads to the night once and is never returned to.

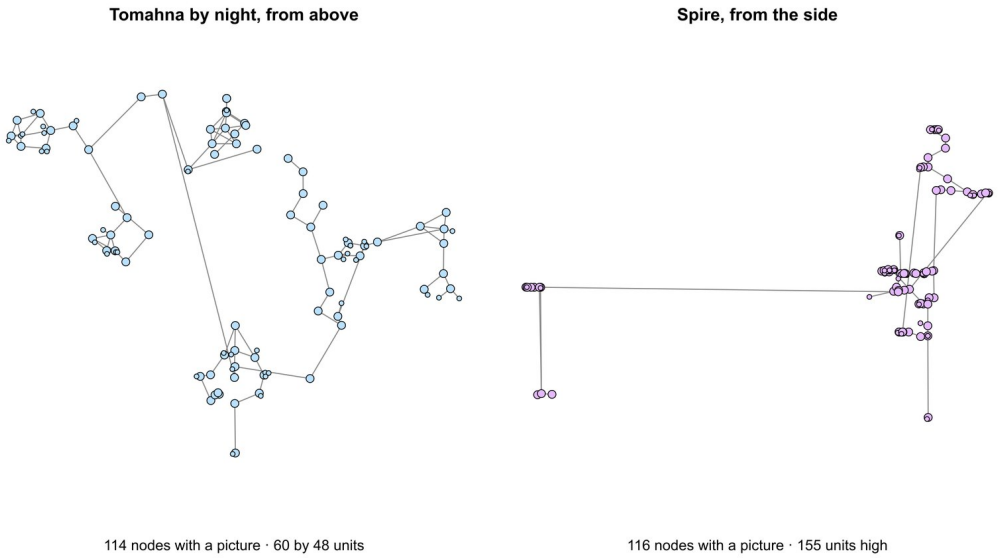
From the first node, 609 of the 629 nodes are reached. The 20 others are the 15 pages of the menu and 5 nodes that the second article examines.

A Graph That Has a Shape

The graphs of Myst, Riven and Exile have no shape of their own. Their drawings are computed: a program places the boxes so that the arrows are short, and another program would place them differently.

Revelation is different. **Each node stores three numbers that are its position**, in the axes of its Age, the third one upward. Two nodes that a connection

joins stand, in the median, 3.2 units apart. Two nodes of the same Age picked at random stand about 48 units apart. The numbers are therefore the place where the node was rendered, and the nodes of an Age, drawn at their positions, give its map. The 30 nodes that store no picture are the exception: many of their positions appear to be copies, so the maps leave all 30 out.



Two Ages drawn from the stored positions alone. Tomahna by night, from above, is 60 units by 48. Spire, from the side, is 155 units high.

The graph browser therefore has two drawings. The graph is the one of the earlier articles. The map places the nodes where the game stores them and can be turned and zoomed, one Age at a time or the five together. The game says nothing of where an Age lies from another, since they are different worlds: in the view of all five, their arrangement is a choice of the page.

The hotspots agree with the positions. A hotspot that leads to another node is stored with its own offset from the node, and that offset points at the destination: for 754 of the 872 hotspots that I could compare, the angle between the two directions is under 30 degrees.

Legend: How to Read the Graph

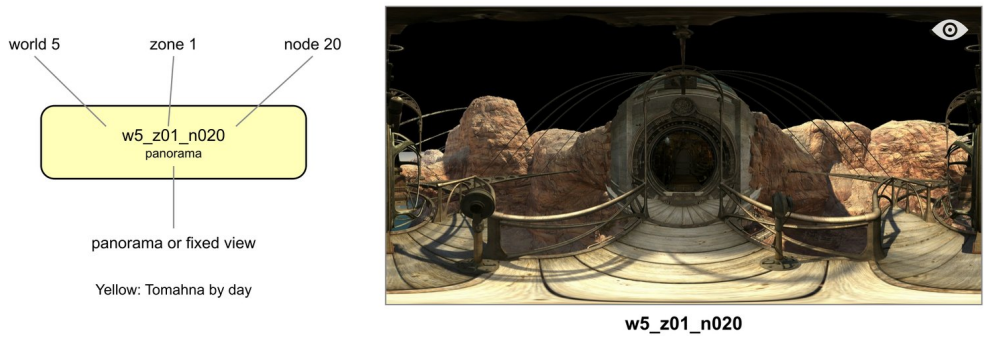
The browser graph uses a node's Age for its color, its references for its connections, and distinct borders or arrowheads for special cases. The following legend explains those conventions before the findings put them to use.

Reading a Node

Labels and Colors

Each Age has its own color. A label is the name that the game gives the node, unchanged: w5_z01_n020 is world 5, zone 1, node 20. The name is complete in itself: no two nodes share one.

Under the label, each box says whether the node is a panorama or a fixed view.



The label of w5_z01_n020 and its view. The open eye identifies a view that the game displays, following the figure convention of the findings.

Panoramas, Fixed Views and Layers

A **panorama** stores six faces. A **fixed view** stores one, and the game shows it flat. Most fixed views are entered from a panorama and go back to it.

What a node shows is not one picture. Besides its default cube, a node can hold masks that are drawn over it, and panels on which movies play: a door that opens, a lever that moves, a character who speaks. This study displays the default cube only. Thirty nodes have no cube at all, and the second article says what they are.

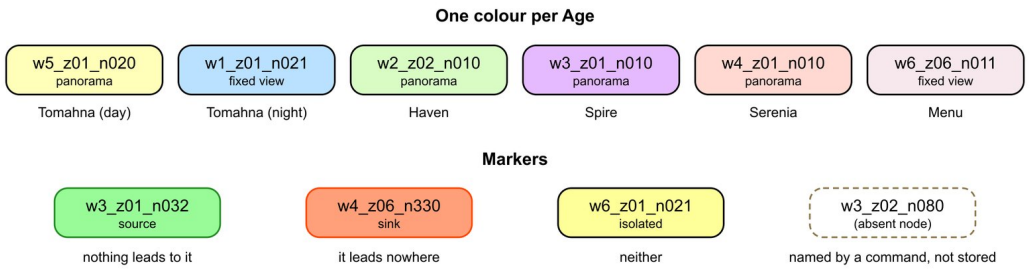
Sources, Sinks and Isolated Nodes

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. Some nodes are entered by compiled code whose origin I could not establish. They appear as isolated nodes, and the browser says so in their record.

Virtual Targets

Thirteen connections lead to eleven nodes that this release does not store, such as w3_z02_n080. I keep each target as a virtual node, with a distinct border, rather than discard the reference. The second article comes back to them.



Reading a Connection

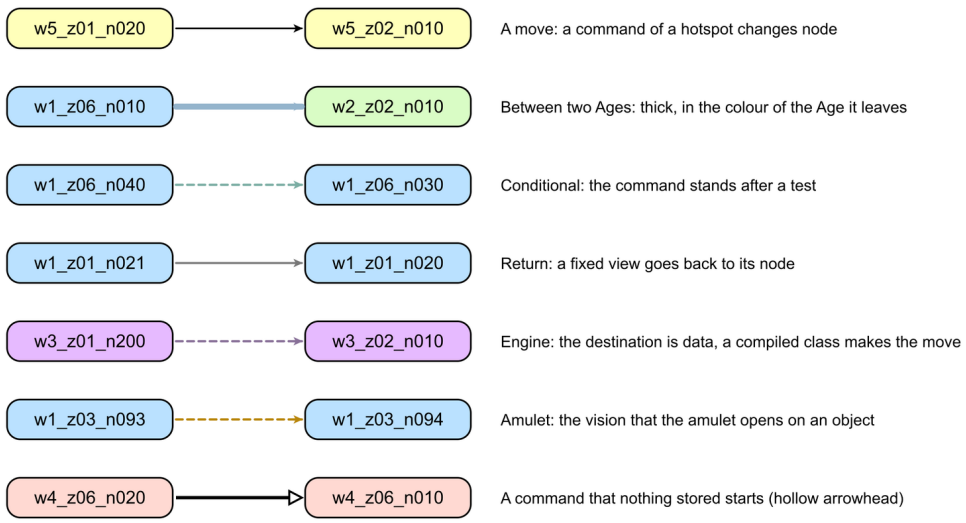
An arrow has a source and a destination. If both directions exist, the nodes are mutually connected. If only one exists, following it does not imply that the player can simply reverse the move.

As in Exile, the way back is usually behind the player: **95.1% of the 1,336 pairs of nodes have their reverse pair**, against 90.4% for Exile, 66.4% for Riven and 55.5% for Myst. The 66 one-way pairs are rides, falls, links and the passage from day to night.

There are several kinds of connection:

Kind	Typed connections	Meaning
Move	1,031	A command started by a hotspot changes node, or a transition plays a movie and arrives elsewhere.
Return	209	A fixed view goes back to a node.
Engine	95	The destination is stored as data, and a compiled class makes the move.
Amulet	9	The vision that the amulet opens on an object.
Automatic	2	A command that no click starts changes node.
Unreferenced command	9	A command changes node, and nothing stored starts it.
Total	**1,355**	All exported kinds together.

Path calculations leave out the 9 unreferenced commands and 10 of the returns, for a reason given below. They retain 1,336 unique ordered pairs of nodes.



From top to bottom: w5_z01_n020 → w5_z02_n010, w1_z06_n010 → w2_z02_n010, w1_z06_n040 → w1_z06_n030, w1_z01_n021 → w1_z01_n020, w3_z01_n200 → w3_z02_n010, w1_z03_n093 → w1_z03_n094, and w4_z06_n020 → w4_z06_n010.

Several pieces of evidence can support the same edge. I merge references only when their source, destination and kind agree, while keeping every contributing instruction in its evidence.

Commands Are Sentences

The instructions of Revelation are not numbered operations. They are lines of text, grouped in named blocks, and the line that moves the player reads `ContextChange(1,01,021)`: world 1, zone 1, node 21. The game stores 28,980 such lines in 6,913 blocks. Of them, 1,029 change node.

A block does nothing by itself. Something has to start it, and the node says what: a small parameter file ties a hotspot to three blocks, one for the pointer entering it, one for the pointer leaving it and one for a click. A move of the graph is a `ContextChange` in a block that a click starts.

Conditions Are Rare

A block can test a variable and jump. The extractor keeps these tests beside the connection and never evaluates them. Only **28 connections are conditional**, against 316 in Exile. This does not mean that Revelation has fewer locked doors. It means that the locks are elsewhere: in the compiled classes. Ten of the 28 are tests read in their code; among them, the passage from day to night waits for two variables of a puzzle.

What the Compiled Classes Do

Each node lists the **classes** that run in it, with their parameters. A class is compiled code, shipped in the libraries of the game. Most are general: a hotspot that plays a sound, a movie that loops. Others belong to one node, such as `w1_z01_n020`, the class of a door with an iris and a lever.

A class does what the commands cannot: wait for a lever to be pulled all the way, play a ride, open a linking book. When it moves the player, the destination is usually still data, and I found it in five forms:

- a hotspot named for its destination, such as `h_to_w4_z01_n010`, that no command serves;
- three parameters that give a world, a zone and a node, such as the `m_world_to_go`, `m_zone_to_go` and `m_node_to_go` of an elevator of Spire;
- a transition movie of the node named for its destination, such as `w1_z06_n050.p_tr_w1_z09_n030_default_transit`;
- the name of such a hotspot written as a constant in a library;
- on the side of the destination, the direction to face when arriving from a named node.

These are the **engine** connections: 95 in all. The links between the Ages are among them. In the room of Tomahna that holds the linking books, w1_z06_n010, the hotspots to Haven and to Spire are parameters of the class of the room.

A few destinations are written in the compiled code itself, as three constants pushed before the call that changes node. The libraries hold 62 of them. Some stand apart from their call, behind other instructions, and I read those by hand. To give a destination to a node, I follow its routine back to the class that owns it, through the method that calls the routine or hands it over, and to the node that lists the class. That names the node of 51 of the 62. The 11 others are recorded on their destination, without an arrow, since nothing says where they start. The passage

from day to night is one of the constants set apart from their call: it belongs to the class `w5_z01_n110`, which bears the name of its node.

Returns That Belong to the Caller

A fixed view goes back where the player came from. For 140 of them, the view records the node to go back to. For 59 others it records nothing, and the engine remembers the last node: I infer those returns from the nodes that enter the view, and mark them as inferred.

Five fixed views are entered from two nodes. They are close-ups stored once, in Tomahna by night, and used by day as well: the names of what opens them, in French, speak of a drawer (*tiroir*), a porthole (*hublot*) and a door (*porte*). Each has two inferred returns, one to the day and one to the night, and each return belongs to the node the player came from. Entering such a view by day and leaving it into the night would be a shortcut that the game does not have. The graph keeps these 10 returns, draws them with a hollow arrowhead and leaves them out of path search, as the Riven Graph does for its caller-owned continuations.

Commands That Nothing Starts

Nine blocks change node and are started by nothing that the data stores: no hotspot, no parameter of a class, no other block. Some are probably started by compiled code that builds their name. The graph draws them with a hollow arrowhead and path search ignores them.

Archives Inside Archives

The format had been described by players of the game who wanted to look at its pictures (myst4tools, the Tomahna project): an archive is a tree of names, sizes and offsets, followed by the files, uncompressed. Archives nest. The main one holds 618 node archives, and a node archive holds one archive per picture layer.

The names in the binary files are lightly scrambled: in every byte, each pair of neighboring bits is exchanged. Applying the same exchange again gives the text back. Comparing a scrambled name with the name it had to be revealed the rule.

No Engine to Lean On

For Myst, Riven and Exile, an open-source program that plays the game says what each instruction means and what the engine does on its own. Revelation has no such program. Two things made up for it.

The first is that the game explains itself. Commands are sentences, parameters have names, and the names were chosen by people who had to find their way in them: a hotspot that leads to a node is called `h_to_` followed by the name of that node.

The second is that the libraries can be read without being run. They import one function to change node. Where the three numbers of the destination are constants pushed just before the call, I read them in the disassembly. Where they are loaded from an object, they come from the parameters, which the node archives already hold.

What Is Inferred

Every connection keeps its evidence, and 63 of the 1,355 are inferred rather than read:

- 59 returns of fixed views that record no node to go back to;
- 3 entries into fixed views that record their node and that nothing stored opens;
- 1 connection set by hand, from the first node of a new game. w5_z01_n010 is a scene, not a place to walk in, and it stores no exit. Its library holds one compiled transition to another node, w5_z01_n020, and I attribute it to the first node.

The browser marks each of them, and their tooltips say so in the drawing.

Was Everything Read?

The node archives are listed by the main archive, so none can be missed: 618 archives, 618 nodes. Their small files were all decoded without an error. What can be missed is a move, in three cases: a destination computed in compiled code from something other than a parameter, a compiled constant that I could not attribute, and a compiled constant that the reading does not recognize. The third happened: a first reading missed eight of the 62 constants, the passage from day to night among them. w1_z01_n310, where the night begins, stood in the graph as a complete node without an entrance, until a wider reading of the libraries found its entrance, three constants of the library of the day set apart from their call, and with it the way to five fixed views of Haven and to two nodes of the Dream.

The word “decompile” needs a precise scope here. The project decodes the game’s archives, commands and parameters into readable structures, and reads constants in the compiled libraries. It does not recover the engine’s source code or fully model how its classes use those parameters.

Conclusion

Revelation keeps the world of places that Exile introduced and stores it differently: one archive per node, sentences for commands, and compiled classes for everything that is not a simple click. The graph that comes out of it is smaller than the earlier ones and nearly symmetric.

It is also the first of the series that has a shape. In the next article, I use that shape: to follow paths, to measure what stands close and is far apart, and to look at what the discs name and do not hold.

How to cite this article

> Guillaume Lethuillier, "The Revelation Graph, 1: The 2004 Game as a Graph", glthr.com, November 21, 2026.
> https://glthr.com/revelation-graph-1

For BibTeX:

```
@misc{lethuillier2026revelation,
  author = {Lethuillier, Guillaume},
  title  = {The Revelation Graph, 1: The 2004 Game as a Graph},
  year   = {2026},
  month  = nov,
  url    = {https://glthr.com/revelation-graph-1},
  note   = {glthr.com}
}
```

Your questions and leads are welcome: myst@glthr.com

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