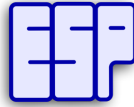


ESP File Specification for Puzzles

Created by SchwanSongs Software Products

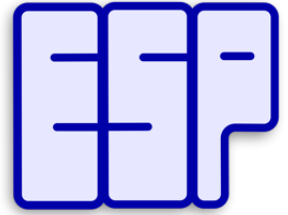
Version 1.0, Revision 2025.08.24



1. Overview	2
2. File Format Description	3
Header	3
File Header Information section	3
Puzzle Header Information section	4
Authoring section	5
Export Information section	5
Puzzle Title section	6
Details - Word Search (PuzzleClass 1)	8
Alphabet/Language section	8
Puzzle Settings section	10
Puzzle Letter Grid and Word List section	11
Puzzle Floor and Walls section	14
Puzzle Answers section	16
Puzzle "Look-up-tables" section	16
Details - Maze (PuzzleClass 2)	18
Puzzle Settings	18
Walls and Paths	20
Maze-Phrase Settings	21
Cell Walls	23
Solution	24
Puzzle "Look-up-tables" section	24
Details - Sudoku (PuzzleClass 3)	26
Look-Up Tables (LUTs)	27
File Checksum	27
3. File Version Changes	29
4. Example Files	30
Word Search (Puzzle Class 1)	30
Maze (Puzzle Class 2)	38
Sudoku (Puzzle Class 3)	42

1. Overview

This document describes the ESP¹ File, which captures all the SchwanSongs-generated puzzle information you need to re-create them in your own style for your publishing workflow or interactive "game-player" software. This document has sections describing each file element, any changes in different versions of the file format, and example files at the end.



An ESP file is generally meant to work with my grid-style puzzles, like word searches, mazes, and sudoku, but can be extended to other puzzle types. It contains the entire "puzzle" and "solution" information, normalized coordinates for wall lines, and the user's chosen colors and font names, allowing you to draw the puzzle in your own format/size.

This file format is created/exported alongside the puzzle and answer/solution image files, by several puzzle software applications that I publish. You are free to use the file format however you like, including adapting your own software to read or write to this format.

The file has the extension ".esp"² (upper or lower case), and is encoded in a standard JSON (UTF-8) file format.

There is a basic "reference web page" that describes this file, and offers this document for download. The page is hosted at:

www.schwansongs.com/espfile/

Note the intentional similarity to the "fileType" JSON element below.

¹ E.S.P. stands for "E.S.Puzzles" in this document, and probably not for "Extra-Sensory Perception" (despite the intentionally alluded pun.)

² My ESP logo, which I have signed my (programming, puzzles, photography, poetry, printing, etc.) works with since I was a teen, is an intentional nod to a certain world-famous Dutch graphic artist, MCE.

2. File Format Description

ESP Files are in standard JSON file format, which is a simple text file with an opening and closing curly brace surrounding a set of “key-value pairs” with the strings enclosed in double-quotes. The structure is called a dictionary. Here’s a simple tiny example:

```
{
  "fileType": "schwansongs/espfile",
  "fileVersion": 1,
}
```

In this document I will be replacing the unweildy phrase “key-value pairs” with “elements.” Each ESP File element in the file is detailed here, with “header” elements that are common to all *types* of puzzles, and elements unique each different type of puzzle. Each of these different puzzle *types* is identified in this document as a “puzzle class”.

Each ESP File element can appear in any order in the file.

All ESP File key names are camel-cased³, so for example, the “file type” key will be written “fileType”. Any actual string values, when they are items of an emumerated list and not user-displayable, will also be camel-cased.

All ESP File string values that have special characters will be encoded per the JSON spec, i.e., newline is replaced with \n, carriage return is replaced with \r, tab is replaced with \t, double quote is replaced with \", backslash is replaced with double-backslash \\.

All numeric indeces are zero-based. All coordinate positions are zero-based, and the origin of puzzle grids is in the upper left-hand corner, with x and y growing to the right and down. All raw 2-dimensional coordinate arrays are rows of columns, that is:

```
[ [x0,x1,x2...], // y0
  [x0,x1,x2...]  // y1
]
```

In this document, any element followed by an asterisk is optional, and is not required to be in the file.

Header

The file starts with some basic information common to all supported puzzle classes.

File Header Information section

fileType

This is a file type or format identifier that asserts that this file conforms to the ESP File format described in this document. It will always have this string value.

Example:

```
"fileType": "schwansongs/espfile"
```

³ Camel-case will leave all letters lower-case, but capitalize secondary attached words.

fileVersion

This is version 1 of the specification. If the format changes in a way incompatible with prior readers, the new version will be noted here, and the "File Version Changes" section below will describe what is different. Versioning will be a simple incrementing integer, starting at 1.

Example:

```
"fileVersion": 1
```

softwareName

This is a string name of the software that created this file, just for informational info.

Example:

```
"softwareName": "Whirlwind WordSearch"
```

softwareVersion

This is a string version number of the software that created this file, just for informational info.

Example:

```
"softwareVersion": "5.4.3"
```

Puzzle Header Information section

The following section holds things that describe puzzles in general.

puzzleClass

This integer value specifies the overall class (i.e. type) of puzzle. Depending on this value, the individual "Details" sections below will list the elements specific to each puzzle class.

The values are:

1. Word Search puzzles
2. Maze puzzles
3. Sudoku puzzles

Example:

```
"puzzleClass": 1
```

puzzleClassName

This is user-displayable text describing the overall class of puzzle.

Example:

```
"puzzleClassName": "Word Search"
```

puzzleGUID*

This is an optional unique identifier string for each puzzle of a class, a kind of serial number that insures puzzles are different from each other. This value and format is specific to each software generator. The only requirement is that any two otherwise-identical puzzles that have different solution layouts should have different PuzzleGUIDs. Note that this GUID does not take into account the user-specified settings for puzzle layout and display. This means it is only valid for comparing two puzzles that have all other settings in

common (e.g. same dimension, same shape, etc.) It is theoretically possible to have two different puzzles that have very different shapes/sizes to have the same GUID.

Example:

```
"puzzleGUID": "abra-guid"
```

Authoring section

The following section holds who authored the puzzle, and when.

authorName*

This is a string of the name of the puzzle author.

Example:

```
"authorName": "Jason Morasse"
```

authorCopyright*

This is a string of the copyright statement for the puzzle author.

Example:

```
"authorCopyright": "Copyright © 2025, Mr. A-Z"
```

datePublished

This is a string of the date of the puzzle publication/creation, in yyyyymmdd format.

Example:

```
"datePublished": "20250730"
```

Export Information section

The following section holds export details that may be of use.

exportInfo

When exporting a set of files, the user can set a specific file name prefix string as the file name. That then gets an export counter # added, then the suffix _Puz or _Ans, and finally the appropriate file extension (.png, .esp, etc.) If doing bulk-builds, then the design-counter and design-variation are also added.

There may be extra side-files exported, alongside the main puzzle/solution files, and they will have their full filenames listed below as well.

For more information on export file settings, see the user guide “Export Files” section. For more information on the bulk-builder “designCounter” and “designVariation”, see the user guide “Bulk Builder” section.

The dictionary keys:values are:

- "exportImageDimension": A pair of integers that give the exported image's requested width and height, in pixels.
- "exportFileFormat": A descriptive string for the exported format of the files (PNG, JPEG, etc.)
- "exportFileNameLayoutStyle": A string describing the order in which the parts of the filenames are put together, "String+Counter+Format", "Counter+String+Format", "Format+Counter+String".
- "exportFileNameBase": The user-specified base string to name the export files.

- “exportFileNameExt”: The file extension (thus file type) for the exported files.
- “exportFileNamePuzzle”: The full file name used to export the file for the puzzle.
- “exportFileNameSolution”: The full file name used to export the file for the solution.
- “designCounter”*: If doing a bulk-build, this is the incrementing “design counter” for the export.
- “designVariation”*: If doing a bulk-build, this is the incrementing “variation counter” for the export.
- “exportCounter”: The incrementing “export session” counter optionally added to the file name.

Example:

```
"exportInfo": {
  "exportImageDimension": {"width": 1200, "height": 1200},
  "exportImageDimension": "PNG",
  "exportImageDimension": "String+Counter+Format",
  "exportFileNameBase": "tsuki",
  "exportFileNameExt": "png",
  "exportFileNamePuzzle": "tsuki_2_Puz.png",
  "exportFileNameSolution": "tsuki_2_Ans.png",
  "exportCounter": 2
}
```

Puzzle Title section

The following optional section holds puzzle title/subtitle text, if the puzzle has them.

title*

Some puzzles have both a title and additionally a subtitle text field. There can be puzzle-macros embedded in the text, so the text will be listed with the macros expanded, and raw with the macros in-place.

The dictionary keys:values are:

- “titleText”: The final macro-expanded title text the user sees.
- “titleRaw”: The raw title text the user originally entered, with macros un-expanded.
- “subtitleText”: The final macro-expanded subtitle text the user sees.
- “subtitleRaw”: The raw subtitle text the user originally entered, with macros un-expanded.
- “justification”: the positioning of the text above the puzzle:, left, right, center.
- “font”: The fontLUT index to use for the title & subtitle.
- “color”: The colorLUT index to use for the title & subtitle.
- “relFontSize”: This gives a relative font size adjustment for the default title & subtitle sizes, from 0.5 to 2.0, with 1.0 being the default.

Example:

```
"title": {
  "titleText": "Puzzle Heart 20x20 – Where the Wild Things Are"
  "titleRaw": "Puzzle $$shape $$dimxy – Where the Wild Things Are"
  "justification": "center",
```

```
"font": 0,  
"color": 0,  
"relFontSize": 0.8  
}
```

Details - Word Search (PuzzleClass 1)

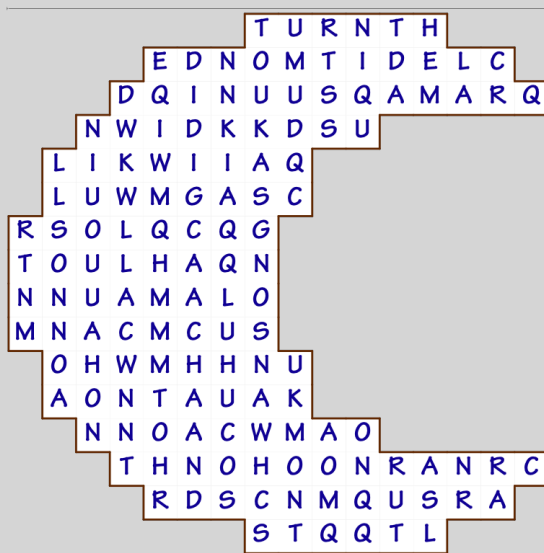
Word Search puzzles contain a 2-dimensional grid of square cells, each cell contains a character. The grid is oriented with the first cell (0,0) in the upper left corner, and x increases to the right, y increases down. Some cells may not exist in the grid, allowing for non-square shapes to be designed. There may be an outer wall around the entire shape, and there may be inner grid lines between each cell. There may also be a solid-colored square at each cell to create an opaque floor. The answer key can be displayed with a number of features.

There are many ways to configure the puzzles, and it is best to first read the "Whirlwind WordSearch User Guide" to become familiar with puzzle shapes, word lists, different languages, puzzle macros, hidden words, secret messages, etc.

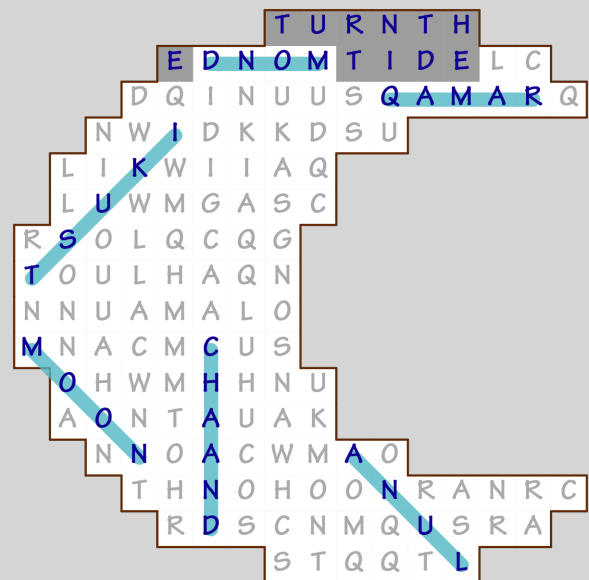
Note that the "puzzleGUID" for word searches is the "SPIN" as described in the Whirlwind WordSearch user guide.

Moon in many languages

- 16X16



- | | |
|-----------|----------|
| 1. Chaand | 4. Moon |
| 2. Luna | 5. Qamar |
| 3. Mond | 6. Tsuki |



Alphabet/Language section

langCode

This is a string that describes the language/alphabet used in this puzzle as an ISO 639 string. The full list of supported language codes (with unsupported languages like Klingon⁴ removed) is:

- 0. "en_Latn_US": English Latin
- 1. "cat": Catalan

⁴ Despite concerted efforts, the Klingon language has not been allowed to be added to the Unicode standard.

- 2. "cs": Czech
- 3. "da_DK": Danish
- 4. "nl": Dutch
- 5. "fr": French
- 6. "de": German
- 7. "el": Greek
- 9. "hu_HU": Hungarian
- 10. "isl": Icelandic
- 11. "ita": Italian
- 12. "ja_Hira": Japanese Hiragana
- 13. "ja_Kana": Japanese Katakana
- 14. "no": Norwegian
- 15. "por": Portuguese
- 17. "ro_RO": Romanian
- 18. "ru_Cyrl": Russian Cyrillic
- 19. "san_Latn": Sanskrit_Latin
- 20. "es": Spanish
- 21. "sv_SE": Swedish
- 22. "tr": Turkish
- 23. "uk-Cyrl": Ukrainian
- 24. "cym": Welsh
- 25. "numbers_Arabic": Numbers (Arabic)
- 26. "numbers_Kanji": Numbers (Japanese/Chinese)

Example:

```
"langCode": "en_Latn_US"
```

alphaLangName

This is a string that gives an English name for the language/alphabet used in this puzzle. See the “langCode” table above for these names.

Example:

```
"alphaLangName": "English Latin"
```

validCharacters

This is a string of characters that are valid for this puzzle (based on the "langCode" above.)

Example:

```
"validCharacters": "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

Puzzle Settings section

puzzleDimension

This is a dictionary of a pair of integers that gives the outer puzzle dimensions in # of cells wide and high. It can vary from 3 up to 80, depending on the puzzle class. For the two rectangle shapes, the puzzle dimensions will not be the same/square.

Example:

```
"puzzleDimension": {"width":16; "height":16}
```

puzzleShapeName

This is a user-displayable string that describes the shape of cells in the puzzle.

Example:

```
"puzzleShapeName": "Crescent"
```

difficulty

This is a dictionary containing an overall difficulty-setting for the puzzle, and a number of other difficulty-related settings.

The dictionary keys/values are:

- "value": An integer that goes from 1 (easy) to 8 (super-hard).
- "desc": Related string that gives a user-displayable text description of the difficulty level.
- "directionsAllowed": The final macro-expanded title text the user sees. The word directions in the array are: [Right,Down,RightDn, RightUp, Up, LeftUp, Left,LeftDn]. See the “Word Directions” section below.
- “allow1LetterWords”: true if single-letter words are allowed in the puzzle.
- “allowNumbersWithAlpha”: true if arabic/kanji numbers are allowed to be a part of a word, like “Route 66”.
- “allowEmbeddedSubwords”: true if WSS allows subwords to stay embedded in larger words, like allowing both “Bat” and “Bath”.
- "wordOverlapAmount": "no overlapping", "a little overlapping", "some overlapping", "lots of overlapping", "ALL overlapping"
- "decoyWordletsAmount": "no decoys", "a few decoys", "some decoys", "lots of decoys"
- "fillLetterSubset": This is the string of alphabetic letters that will be used to randomly fill the remaining puzzle spaces around words. It can be a smaller subset of the full alphabet.

Example:

```
"difficulty": {  
  "value": 8,  
  "desc": "8. Diabolically Difficult",  
  "directionsAllowed": [true,true,true,true,true,true,true],  
  "allow1LetterWords": false,  
  "allowNumbersWithAlpha": true,  
  "allowEmbeddedSubwords": true,  
  "wordOverlapAmount": "a little overlapping",  
  "decoyWordletsAmount": "a few decoys",
```

```
"fillLetterSubset": "abcdefghijklmnopqrstuvwxyz"
}
```

Puzzle Letter Grid and Word List section

gridLetterFormatting

This is a dictionary containing formatting settings for the letters in the puzzle and answer grids, font, size, color, positioning, capitalization, etc.

The dictionary keys:values are:

- "font": The fontLUT index to use for each letter.
- "color": The colorLUT index to use for each letter.
- "relFontSize": relative size of letter in cell, from 0.5 to 2.0, where 1.0 is default.
- "doOutlined": true if the letter should be drawn with an outline, or stroke around it.
- "outlineColor*": colorLUT to use for the outline/stroke of the letter. The letter's inside (fill) color will still be the "color" value above. This line will only be written if "doOutlined" is true.
- "outlineWidth*": relative thickness of the outlined font, from 1-10, where 4 is default. This line will only be written if "doOutlined" is true.
- "allLowerCase": true if the puzzle letters were all converted to lower case.
- "removeDiacriticals": true if the diacritical marks have been stripped from each letter of the words before they were placed in the puzzle grid.

Example:

```
"gridLetterFormatting": {
  "font":1,
  "color":1,
  "relFontSize":1.11867,
  "relVertAdjust":-0.417704,
  "doOutlined":false,
  "outlineColor":2,
  "outlineWidth":4,
  "allLowerCase":false,
  "removeDiacriticals":false
},
```

wordListFormatting

This is a dictionary containing formatting settings for the word list displayed under the puzzle, number of columns, capitalization, etc. It is drawn in the same color as the grid letters.

The dictionary keys:values are:

- "numColumns": 0-8 // 0=auto-calculated.
- "centeredColumns": true if the words should be center-justified within the columns (looks best if "numbered" is false).
- "alphaSorted": true if the words are listed alphabetically.

- "upperCased": true if the words are all-upper-case.
- "numbered": true if the word list is prefixed with ordinal numbering (1. 2. 3...)
- "addDirectionArrows": false,
- "addDirectionArrowsOnlyOnAnswer": true,
- "font": The fontLUT index to use for the word list.
- "color": The colorLUT index to use for the word list.
- "relFontSize": relative size of the word list, from 0.5 to 1.5, where 1.0 is default.

Example:

```
"wordListFormatting": {
  "numColumns": 2,
  "centeredColumns": false,
  "alphaSorted": true,
  "upperCased": false,
  "numbered": true,
  "addDirectionArrows": false,
  "addDirectionArrowsOnlyOnAnswer": true,
  "font": 2,
  "color": 3,
  "relFontSize": 1.15
}
```

wordList

This is an array of dictionaries, one for each word in the puzzle. Some words may be marked "hidden" by the user, meaning that the word is in the puzzle, but they do not want it listed in the final word list. This is useful for hiding "bonus words" or adding a company name as a watermark.

The dictionary keys:values are:

- "word": The word to find in the puzzle.
- "clue*": If this puzzle is built as a "word search with clues", then this is the text to display instead of the actual word, giving another level of indirection for the puzzle. Clues are optional, but if they are used, then every word⁵ in this list must have a clue field.
- "hidden": true/false whether it should be hidden from the displayed word list
- "startCellXY": the cell x,y position of the first letter of the word.
- "direction": A direction number from 1-8, indicating the direction the rest of the word letters are laid into the grid. See the section "Word Directions" below for values and meanings.

Example:

```
"wordList": [
  {"word": "dunlin", "hidden": false, "startCellXY": [5,13], "direction": 8},
  {"word": "grebe", "hidden": false, "startCellXY": [13,6], "direction": 6},
```

⁵ Since the world is full of exceptions, any word that is marked as "hidden" does **not** have to have a clue, since it is not shown in the word list.

```
[{"word": "munia", "hidden": false, "startCellXY": [0, 14], "direction": 4},
{"word": "phalarope", "hidden": false, "startCellXY": [15, 15], "direction": 7},
{"word": "phoebe", "hidden": false, "startCellXY": [18, 13], "direction": 5},
{"word": "snipe", "hidden": false, "startCellXY": [14, 0], "direction": 3},
{"word": "whimbrel", "hidden": false, "startCellXY": [5, 0], "direction": 2}
]
```

Word Directions

Word directions are indicated by a number, from 1 to 8, that indicate the following directions the word goes from its starting coordinates:

1. horizontal to the right ➡
2. vertical down ↓
3. diagonal down to the right ↘
4. diagonal up to the right ↗
5. vertical up ↑
6. diagonal up to the left ↖
7. horizontal to the left ←
8. diagonal down to the left ↙

secretMessage*

This is an optional dictionary that contains "secret message" information, if the user added one to the puzzle.

Secret messages are text that is laid into the puzzle after all the words are placed. The secret message can be at the top, bottom, center, or random position in the puzzle, and it can either take up some of the random letter positions, or take up every remaining position (called exact-match.)

Usually the cell floor color under the Secret Message letters is tinted a light gray in the answer key, to show where it is laid out.

See the Whirlwind User guide for more information on this feature.

The dictionary keys:values are:

- "original": The full displayable phrase or sentence, with all punctuation and spaces left intact.
- "stripped": The stripped-down version of the secret message, containing only the valid alphabetic characters included in the word search puzzle.
- "template": A version of the "original" string, but with all alphabetic characters replaced with underscores. Useful for displaying on the puzzle page as a clue for what to look for.
- "addTemplateSpaces": True if the template underscores should be separated by space characters to make them easier to read.
- "puzzlePromptText": A user-supplied text prompt to hint at what the secret message is.
- "answerPromptText": The full text prompt displayed on the answer page.
- "showInPrintedAnswer":

- "placement": a string that specifies where in the puzzle the user chose to place the secret message. It can be any of: top|middle|bottom|random|exact-fit.
- "cells": An array of [x,y] cell positions for each letter of the secret message in the puzzle.

Example:

```
"secretMessage": {
  "original": "Turn the Tide",
  "stripped": "turnthetide",
  "template": "_ _ _ _ _ _ _ _ _ _",
  "addTemplateSpaces": true,
  "puzzlePromptText": "What does the moon do?",
  "answerPromptText": "Secret Message is:",
  "showInPrintedAnswer": true,
  "placement": "bottom",
  "cells": [[6,14],[8,14],[9,14],[10,14],[12,14],[13,14],[14,14],[8,15],[9,15],[10,15],[11,15]]
}
```

puzzleGrid

This is a 2-dimensional array, a grid of all the letters in the puzzle, with ***null*** used to denote any cells that are not part of the puzzle shape.

Example:

```
"puzzleGrid": [
  [null,null,null,null,null,"K" ,"I" ,"M" ,"O" ,null,null,null],
  [null,null,null,"T" ,"M" ,"N" ,"O" ,"O" ,"M" ,"S" ,"U" ,"K" ],
  [null,null,"U" ,"S" ,"U" ,"N" ,"T" ,"U" ,null,null,null,null],
  [null,"S" ,"L" ,"U" ,"N" ,"A" ,"O" ,null,null,null,null,null],
  ["T" ,"T" ,"M" ,"D" ,"O" ,"L" ,null,null,null,null,null,null],
  ["T" ,"N" ,"N" ,"O" ,"K" ,"U" ,null,null,null,null,null,null],
  ["T" ,"O" ,"N" ,"I" ,"I" ,"N" ,null,null,null,null,null,null],
  ["O" ,"S" ,"N" ,"D" ,"A" ,"K" ,null,null,null,null,null,null],
  [null,"A" ,"U" ,"U" ,"A" ,"O" ,"D" ,null,null,null,null,null],
  [null,null,"D" ,"K" ,"L" ,"O" ,"N" ,"T" ,null,null,null,null],
  [null,null,null,"U" ,"I" ,"R" ,"O" ,"N" ,"T" ,"H" ,"E" ,"T" ],
  [null,null,null,null,null,"I" ,"M" ,"D" ,"E" ,null,null,null]
]
```

Puzzle Floor and Walls section

cellFloorColors

This is an optional 2-dimensional array if the user added a non-transparent "cell floor" color on the puzzle. It is a grid of all the cell positions in the puzzle, with ***null*** used to denote cells that are not part of the puzzle shape. The number will indicate the colorLUT index of the color to use in that cell. This table does not change the floor color for cells that are part of the secret message, since that is a feature only shown on the Answer key, so it will be up to you to read the secret message "cells" and color them differently if you wish.

Example:

```
"cellFloorColors": [
  [null,null,null,null,null, 5, 5, 5, 5,null,null,null],
  [null,null,null, 5, 5, 5, 5, 5, 5, 5, 5, 5],
  [null,null, 5, 5, 5, 5, 5, 5,null,null,null,null],
  [null, 5, 5, 5, 5, 5, 5, 5,null,null,null,null,null],
  [ 5, 5, 5, 5, 5, 5,null,null,null,null,null,null],
  [ 5, 5, 5, 5, 5, 5,null,null,null,null,null,null],
  [ 5, 5, 5, 5, 5, 5,null,null,null,null,null,null],
  [ 5, 5, 5, 5, 5, 5,null,null,null,null,null,null],
  [null, 5, 5, 5, 5, 5, 5, 5,null,null,null,null,null],
  [null,null, 5, 5, 5, 5, 5, 5,null,null,null,null,null],
  [null,null,null, 5, 5, 5, 5, 5, 5, 5, 5, 5],
  [null,null,null,null,null, 5, 5, 5, 5,null,null,null]
]
```

cellWalls

This is an optional dictionary if the user wanted cell walls (or "grid lines") on the puzzle. It has the color and wall line width of all the lines, and an array of wall line coordinates around all the cell positions in the puzzle.

The dictionary keys:values are:

- "color": An index into the colorLUT table for the user-chosen color for the wall lines.
- "width": A width of cell lines. 0 means no cell wall drawn. 0.5, 1.0 ... 8.0 is a relative width, scaled based on viewport size. Outline walls and inner grid walls are all the same size.
- "walls": Each element in the array is an array of 4 numbers, representing x0,y0 (line start) and x1,y1 (line end) in that order. Cell walls are not repeated, that is, since cell 1 right wall is the same as cell 2 left wall, that wall will only appear once, not twice. If the user wanted to outline *only* the shape, then only those outline walls will be listed here, none of the inner walls. The line start and end coordinates are relative to a puzzle grid with a unit cell size, i.e, 0.0 to 1.0. You must then scale this up based on the size and position of whatever viewport you are drawing into.

Example:

```
"cellWalls": {
  "color": 4,
  "width": 4,
  "walls": [
    [0,0,0,1], [1,1,0,1], [1,1,1,0], [0,0,1,0],
    [1,0,1,1], [2,1,1,1], [2,1,2,0], [1,0,2,0],
    [2,0,2,1], [3,1,2,1], [3,1,3,0], [2,0,3,0],
    [3,0,3,1], [4,1,3,1], [4,1,4,0], [3,0,4,0],
    [4,0,4,1], [5,1,4,1], [5,1,5,0], [4,0,5,0],
    [5,0,5,1], [6,1,5,1],
    [7,1,6,1],
    [8,1,7,1] ]
}
```

Puzzle Answers section

The answer key can be drawn in a number of ways.

answerLetters

This is a dictionary describing how the puzzle letters are drawn in the answer key.

The dictionary keys:values are:

- "showPuzzleLettersStyle": "none/dimmed/all".
- "dimPuzzleLetterColor": An index into the colorLUT table for the user-chosen color for the non-answer letters if the showPuzzleLettersStyle is "dimmed".

Example:

```
"answerLetters": {  
  "showPuzzleLettersStyle": "dimmed",  
  "dimPuzzleLetterColor": 5  
}
```

answerHighlights

This is a dictionary describing how the answer key words are highlighted.

The dictionary keys:values are:

- "style": "strikeThru/solidOval/openOval".
- "lineWidth": a relative size from 1.0 (thin) to 5.0 (thick), 3 is average.
- "openOvalWidth": (-1 .. 0 .. +1).
- "color": An index into the colorLUT table for the answer highlight line.

Example:

```
"answerHighlights": {  
  "style": "solidOval",  
  "lineWidth": 3  
  "openOvalWidth": 0  
  "color": 6  
}
```

Puzzle "Look-up-tables" section

colorLUT*

The colors are strings of hex digits, prefixed with a pound sign (#). The hex string consists of 2-digits each for red, green, blue, alpha components ("#rrggbbaa"), and the table indices refer to the following user-chosen puzzle colors:

- 0: titleFont Color
- 1: puzzleLetterFont Color
- 2: puzzleLetterOutline Color
- 3: wordList Color

- 4: dimmedPuzzleLetters Color
- 5: cellWall Color
- 6: cellFloor Color
- 7: answerHighlight Color

Example:

```
"colorLUT": ["#006200ff", "#0c0081ff", "#000000ff", "#000000ff", "#999999ff", "#1e2b70ff",
"#ffffffff", "#689ec8ff"]
```

widthLUT*

The wall elements may reference a widthIndex, which will be a number from 0 to n, used as an index into this table to refer to a user-chosen width. The 0 index indicates *no* line at all.

Each width value is a normalized line width relative to a unit-sized cell (i.e. a cell whose width and height are 1.0 unit.) You must then scale this up based on the size of the viewport you are drawing into.

Example:

```
"widthLUT": [0.00, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.08, 0.09, 0.15]
```

fontLUT*

The font indices refer to the following user-chosen font names (from the user's computer):

- 0: puzzleTitle Font
- 1: puzzleLetter Font
- 2: wordList Font
- 3: alphaNumericShapeLetter Font

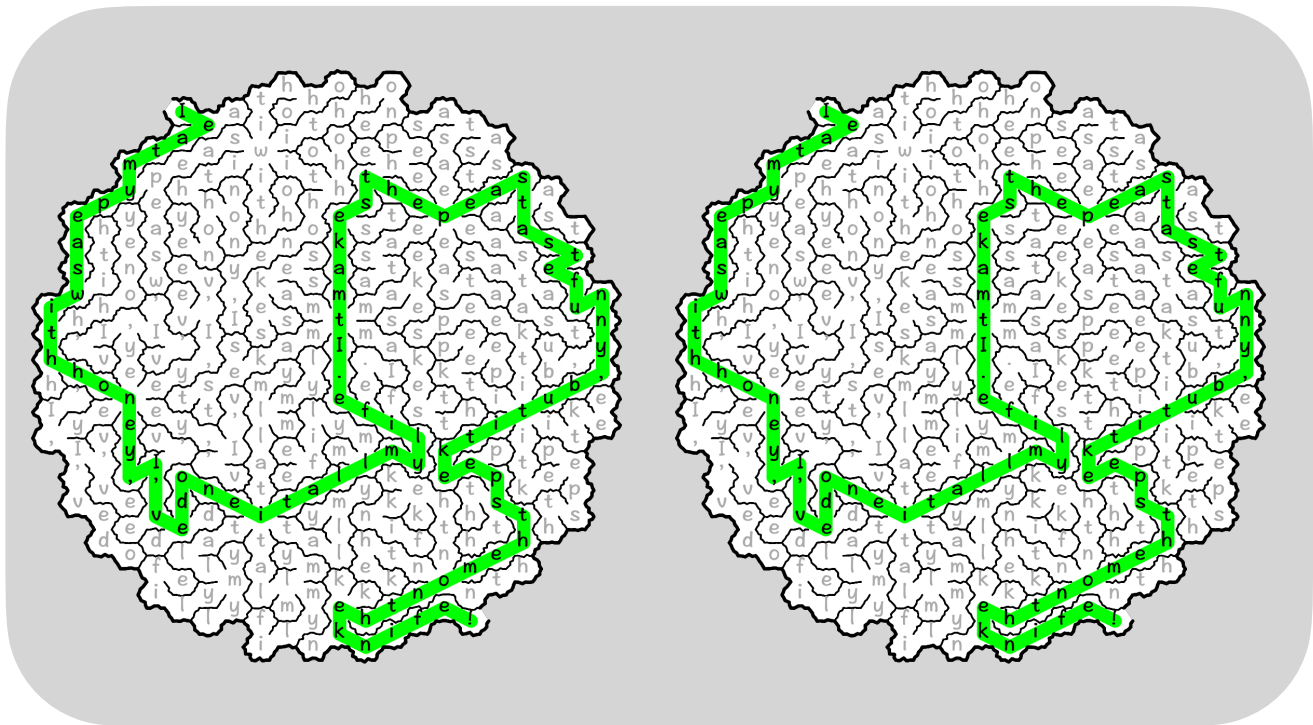
Example:

```
"fontLUT": ["Crillee", "Tekton Pro", "Hannotate SC", "Futura"]
```

Details - Maze (PuzzleClass 2)

Maze puzzles contain a 2-dimensional grid of cells. The cells can be square, hexagonal, iso-triangular, or "octo-grid". Despite the visual layout of the cells when they are hexagonal or triangular, I still have them mapped to a simple x,y grid, which I will describe below. The grid is oriented with the first cell (0,0) in the upper left corner, and x increases to the right, y increases down. Some cells may not exist in the grid, allowing for non-square maze shapes to be designed. There is an outer wall around the entire shape, and there are inner walls between each cell, with some walls missing to create the maze. There may also be a solid-colored "floor" shape under each cell to create an opaque floor.

There are many ways to configure the mazes, and it is best to first read the "Minos Maze Maker User Guide" to become familiar with maze shapes, cell shapes, cell labels, start/end markers, puzzle macros, maze-phrases, etc.



Puzzle Settings

puzzleDimension

This is a dictionary of a pair of integers that gives the outer puzzle dimensions in # of cells wide and high. It can vary from 3 up to 80, depending on the puzzle class. For the two rectangle shapes, the puzzle dimensions will not be the same/square.

Example:

```
"puzzleDimension": {"width":21; "height":21}
```

puzzleShapeInfo

This is a dictionary that describes the shape of the overall puzzle.

```
"puzzleShapeName": "xxx"
```

```
"puzzleANShapeFont": fontLUT2
```

Example:

```
"puzzleShapeInfo": {  
  "puzzleShapeName": "Windmill",  
  "puzzleANShapeFont": "Quixote"  
}
```

difficulty

This is a dictionary containing an overall difficulty-setting for the puzzle. It has a "value" integer that goes from 1 (easy) to 11 (super-hard), and a "desc" string that simply mentions that we only do basic settings around here.

Example:

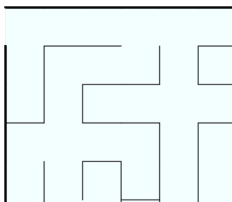
```
"difficulty": { "value":11, "desc":"Basic Settings" }
```

cellInfo

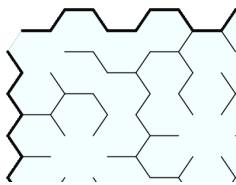
This is a dictionary containing a number of maze cell configuration items.

The dictionary keys/values are:

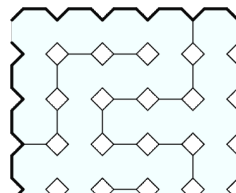
- "cellShape": The shape of the maze cells, which can be square, hex, octo-grid, iso-triangle.



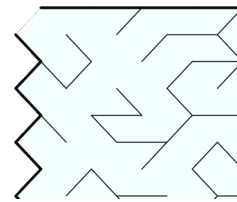
Square (Ortho)



Hex (Sigma)

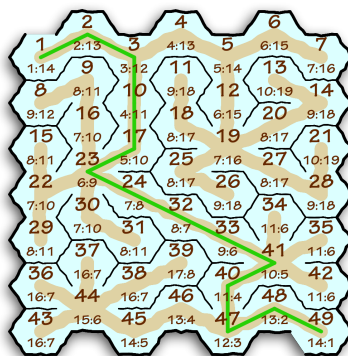
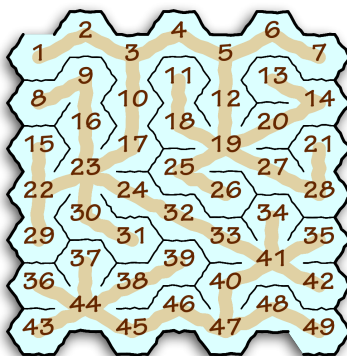


Octo-Grid (Ortho)



Iso-Triangle (Delta-Iso)

- "doCellLabeling": true if user turned on labeling inside each cell (see visual example below), which adds the following items too:



- "cellLabelStyle": How labels are displayed, decimal cell index, hexadecimal cell index, or decimal x,y pair.
- "cellLabelColor": colorLUT index of color for cell labels.
- "cellLabelFont": fontLUT index of color for cell labels.
- "cellLabelSize4Puzzle": Relative size (-1.0 to +1.0) to grow/shrink the *puzzle* cell labels within the cells.
- "cellLabelSize4Solution": Relative size (-1.0 to +1.0) to grow/shrink the *solution* cell labels within the cells.
- "doCellFloor": true if the "floor" of the cell is to be filled with a color.

- "cellFloorColor": colorLUT index of color for cell floor fill.

Example:

```
"cellInfo": {
  "cellShape": "Square (Ortho)",
  "doCellLabeling": true,
  "cellLabelStyle": "dec",
  "cellLabelColor": 2,
  "cellLabelFont": 2,
  "cellLabelSize4Puzzle": 0.117932,
  "cellLabelSize4Solution": -0.123594
  "doCellFloor": true,
  "cellFloorColor": 1
}
```

Walls and Paths

wallPathInfo

WIP... RSN...

This is a dictionary containing a number of configuration items for how maze walls and paths are drawn.

The dictionary keys:values are:

Example:

```
"wallPathInfo": {
  "style": "allWallsPath",
  "wallOuterThickness": 5,
  "pathPuzzleColor": 3,
  "wallInnerThickness": 2,
  "pathPuzzleColor": 4,
  "openWallsAtPathEnds": true,
  "wallWiggleAmount": 0,
  "wallWiggleOuter": false,
  "pathRelThickness": 0.0710371,
  "pathPuzzleColor": 5,
  "pathSolutionColor": 6
}
```

startEndMarks

This is a dictionary describing where the start and end of the maze are, and how the markers are displayed.

The dictionary keys:values are:

- "showMarkers": true if the start/end markers are displayed on the maze.
- // ---- "arrowDisplayStyle": "none" | "thick-flat" | "thick-pointy" | "thin-flat" | "thin-pointy"
- // ---- "markerRelSize": -1/+1,

- # WIP... RSN...

```
"startEndMarks": {
  "showMarkers": "none",
  "markerRelSize": 0,
  "startMarker": { "locationXY": [3, 0], "openingDirection": 1, "markerText": "S", "color": 7, "font": 3 },
  "endMarker": { "locationXY": [13, 16], "openingDirection": 2, "markerText": "E", "color": 8, "font": 3 },
  "openWallsAtPathEnds": true
}
```

```
// ---- "original": "A BC."
// ---- "stripped": "ABC"
// ---- "template": "_ _."
// ---- "prompt": "Find the XYZ"
// ---- "promptStyle": "none" | "template" | "prompt+template" | "prompt+text" | "text"
// ---- "exactFit": t/f
// ---- "addTemplateSpaces": t/f
// ---- "font": fontLUT // FontLUT2_MazePhraseFontName
// ---- "color": colorLUT // ColorLUT2_MazePhraseColor
// ---- "relFontSize": n ///< -1 to +1
// ---- "relVertAdjust": n ///< -1 to +1
// ---- "mazePhraseCellXY": [ [x,y,"Letter"], [x,y,"Letter"]... ]
// ---- "letterFillCellXY": [ [x,y,"Letter"], [x,y,"Letter"]... ]
```

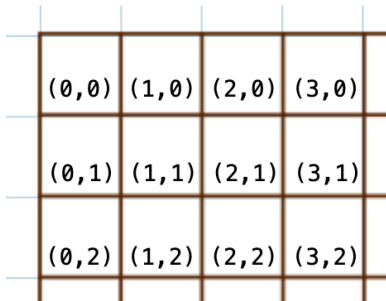
WIP... RSN...

```
"mazePhrase": {
  "original": "Where are the leprechauns today?",
  "stripped": "Wherearetheprechaunstoday?",
  "template": "
```

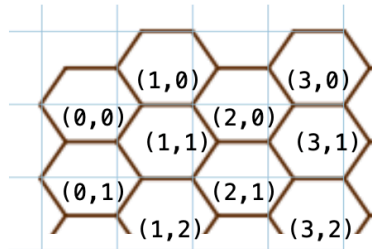

Cell Walls

cellWallsOuter

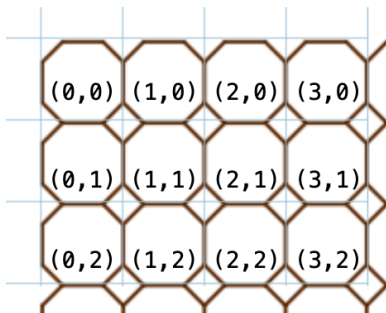
WIP... RSN...



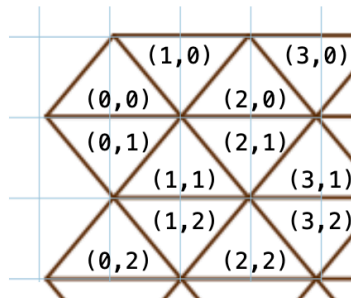
Square



Hex



Octo-Grid



Iso-Tri

The dictionary keys:values are:

Example:

```
// -- cellWallsOuter {  
// ---- "color": colorIndex,  
// ---- "width": widthIndex,  
// ---- "walls": [ [x0,y0,x1,y1,cellID1,cellID2], [x0,y0,x1,y1,cellID1,cellID2], ... ]  
//      walls are lines with an x0,y0 start point and an x1,y1 endpoint,  
//      where the points are based on each maze cell being unit-sized(0,0 to 1,1),  
//      and cellID1 & cellID2 are the cells the wall is between
```

cellWallsInner

WIP... RSN...

The dictionary keys:values are:

Example:

```
// -- cellWallsInner {  
// ---- "color": colorIndex,  
// ---- "width": widthIndex,
```

```
// ---- "walls": [ [x0,y0,x1,y1,cellID1,cellID2], [x0,y0,x1,y1,cellID1,cellID2], ... ]
//      walls are lines with an x0,y0 start point and an x1,y1 endpoint,
//      where the points are based on each maze cell being unit-sized(0,0 to 1,1),
//      and cellID1 & cellID2 are the cells the wall is between
```

Solution

solution

WIP... RSN...

The dictionary keys/values are:

```
// -- solution {
// ---- "pathStyle": "CircleDots"| "CellDots"| "Line"| "Arrows"| "Maze-Phrase Letters (Sparse)"|
//      "Maze-Phrase Letters (Dimmed)"| "Maze-Phrase Letters & Line"
// ---- "pathColor": colorIndex,
// ---- "relWidth": -1.0 to +1.0,
// ---- "cells": [ [x,y], [x,y]...]
```

Example:

```
"solution": {
  "pathColor": 9,
  "relWidth": 0.551188,
  "cells": [ [3,0],[3,1],[3,2],[4,2],[5,2],[6,2],[7,2],[7,3],[7,4],[7,5],[7,6],[6,6],[5,6],
[4,6],[3,6],[3,7],[4,7],[5,7],[5,8],[4,8],[3,8],[3,9],[3,10],[3,11],[2,11],[1,11],[0,11],[0,12],
[0,13],[1,13],[2,13],[2,14],[2,15],[3,15],[4,15],[4,14],[5,14],[6,14],[7,14],[7,13],[7,12],
[7,11],[8,11],[9,11],[10,11],[10,12],[10,13],[11,13],[11,14],[11,15],[11,16],[12,16],[13,16] ]
}
```

Puzzle "Look-up-tables" section

colorLUT*

The colors are strings of hex digits, prefixed with a pound sign (#). The hex string consists of 2-digits each for red, green, blue, alpha components ("#rrggbbaa"), and the table indices refer to the following user-chosen puzzle colors:

- 0: titleFont Color
- 1: cellFloor Color
- 2: cellLabel Color
- 3: outerWall Color
- 4: innerWall Color
- 5: mazePath Color
- 6: answerPath Color
- 7: startMarker Color
- 8: finishMarker Color
- 9: solutionMarker Color

- 10: mazePhraseMarker Color

widthLUT*

The wall elements reference a widthIndex, which will be a number from 0 to n, used as an index into this table to refer to a user-chosen width. The 0 index indicates *no* line at all.

Each width value is a normalized line width relative to a unit-sized cell (i.e. a cell whose width and height are 1.0 unit.) You must then scale this up based on the size of the viewport you are drawing into.

Example:

```
"widthLUT": [0.00, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.08, 0.09, 0.15]
```

FontLUT*

The font indices refer to the following user-chosen font names (from the user's computer):

- 0: Puzzle title font
- 1: AlphaNumeric Shape font
- 2: Cell Label font
- 3: Start/End Marker text font
- 4: Maze-Phrase font

Example:

```
"fontLUT": ["Curlz MT", "Futura", "Hannotate SC", "Hannotate SC", "Crillee"]
```

WIP... RSN...

Wait what? Wouldn't this be nice? Chicken soup with rice?

7								5
			7	4	2			
	4	2	6		1	3	8	
4	8		5		3		1	6
1		9				5		3
3								4
	7	8		6		1	3	
2		4	1		5	7		8
6		3				4		2

	8	2				6	3					16	10	
	16		12	4	5				9	7	2		1	
				15		10	16			12				
11	3				16	14	12	5	1	4			9	8
				5	6		16	11	3	14				
3		15		2	4	1	7	8	12		9			14
13	2	7										4	8	16
			5	8						13	7			
1				4		13	12		6					11
14		8		1					5			10		4
				16	14	6			4	13	11			
	11	13	7	10	12	2	14		1	3	16	5	15	
6	1			12	10		11	2		14	8			4
10		14	3		7	16	5	13	11	15		8	9	2
	4	9	8									11	14	16
7				14		4	6		9					3

	一	四	
一	二	三	四
	四	一	
一	二	三	四

Look-Up Tables (LUTs)

These "lookup tables" are used in some of the above puzzle classes, and present indexed lists of the user's choices of color, font name, and line width usage for different visual aspects of the puzzle display. You can use or remap these items when you display the puzzle, or ignore them.

See each puzzle class section above for its specific list of indices it uses.

ColorLUT*

This is an optional array of color values, in #rrggbbaa hex format. The rr, gg, bb values range from 00 (black) to ff (on), and the alpha (aa) value ranges from 00 (transparent) to ff (opaque.) Some of the puzzle drawing elements (floors, walls) may reference a color, which will be a number from 0 to n, used as an index into this table to refer to a color the user chose in the software. You can use this user's color, or come up with the colors a different way.

Example:

```
"ColorLUT": ["#rrggbbaa", ...]
```

WidthLUT*

This is an optional array of line-width values, relative to a unit cell size. Some of the puzzle drawing elements (walls) may reference a WidthIndex, which will be a number from 0 to n, used as an index into this table to refer to a width the user chose in the software.

Example:

```
"WidthLUT": [0.01, 0.05, 0.1]
```

FontLUT*

This is an optional array of font names from the user's computer where the puzzle was exported. Some of the puzzle drawing elements (titles, letters, etc.) reference a font, which will be a number from 0 to n, used as an index into this table to refer to a font the user chose in the software. You can use this user's font, or come up with the fonts a different way.

Example:

Example:

```
"FontLUT": ["SomeUserFontName", ...]
```

File Checksum

WIP... RSN...

The reason for a checksum is to insure the file was not damaged or meddled with in transit. It should be a relatively quick calculation that does not completely rely on the proper JSON formatting of the file, since the file could have become damaged in a non-JSON-compliant way.

So the checksum will relate to the overall UTF-8 character contents of the file up to (but not including) this last line in the file. The checksum line will be appended to the prior-checksummed file.

espFileChecksum*

This is an optional checksum of the prior portion of the file to insure it wasn't damaged or tampered with.

To calculate the checksum of the file, you must read it in as a UTF-8 string of characters, and perform the following simple checksumming of all the bytes up to and including the colon character just after the key text "espFileChecksum".

The algorithm for doing the checksum is:

```
uint32 checksum = 222 ///< A nice Valentine number to "salt" the accumulator
for bytePointer at each byte of file,
    up to and including the colon character just after the key text "espFileChecksum":
{
    uint8 byteValue = *bytepointer; (get the next byte from the file)
    checksum = checksum*8 + byteValue; (shift up the checksum and add the new value)
}

checksum = checksum & 0x7fffffff ///< remove top (sign) bit
return the checksum; (all done, there you go)
```

Given the simple file here,

```
{
    "fileType": "schwansongs/espfile"
    "fileVersion": 1,
    "espFileChecksum": 1234567
}
```

the calculated checksum would be: 1234567

Example:

```
"espFileChecksum": 1234567
```

3. File Version Changes

Version 1 - First Draft, so no changes from prior versions.


```

"exportFileNamePuzzle": "tsuki_1_Puz.png",
"exportFileNameSolution": "tsuki_1_Ans.png",
"exportCounter": 1
},
"title": {
  "titleText": "\"Moon\" in Other Languages - 16x16",
  "titleRaw": "\"Moon\" in Other Languages - $$dimxy",
  "justification": "center",
  "font": 0,
  "color": 0,
  "relFontSize": 1.5
},
"langCode": "fr",
"alphaLangName": "French",
"validCharacters": "ABCDEFGHIJKLMNOPQRSTUVWXYZÆĖİÖŮÂÊÎÔŰÁÉÍÓÚÀÈÌÒÙ",
"puzzleDimension": {"width": 16, "height": 16},
"puzzleShapeName": "Crescent",
"difficulty": {
  "value": 8,
  "desc": "8. Diabolically Difficult",
  "directionsAllowed": [true,true,true,true,true,true,true,true],
  "allow1LetterWords": false,
  "allowNumbersWithAlpha": true,
  "allowEmbeddedSubwords": true,
  "wordOverlapAmount": "a little overlapping",
  "decoyWordletsAmount": "a few decoys",
  "allowEmbeddedSubwords": "abcdefghijklmnopqrstuvwxyz"
},
"gridLetterFormatting": {
  "font": 1,
  "color": 1,
  "relFontSize": 1.11867,
  "relVertAdjust": -0.417704,
  "doOutlined": false,
  "allLowerCase": false,
  "removeDiacriticals": false
},
"wordListFormatting": {
  "numColumns": 2,
  "centeredColumns": false,
  "alphaSorted": true,
  "upperCased": false,
  "numbered": true,

```

```

"addDirectionArrows": false,
"addDirectionArrowsOnlyOnAnswer": true,
"font": 2,
"color": 3,
"relFontSize": 1.15
},
"wordList": [
  {"word": "chaand", "hidden": false, "startCellXY": [5,9], "direction": 2},
  {"word": "luna", "hidden": false, "startCellXY": [12,15], "direction": 6},
  {"word": "mond", "hidden": false, "startCellXY": [7,0], "direction": 8},
  {"word": "moon", "hidden": false, "startCellXY": [0,7], "direction": 3},
  {"word": "qamar", "hidden": false, "startCellXY": [14,1], "direction": 7},
  {"word": "schwansongs", "hidden": true, "startCellXY": [7,13], "direction": 5},
  {"word": "tsuki", "hidden": false, "startCellXY": [1,5], "direction": 1}
],
"secretMessage": {
  "original": "Turn the Tide",
  "stripped": "turnthetide",
  "template": "_ _ _ _ _ _ _ _ _ _",
  "addTemplateSpaces": true,
  "puzzlePromptText": "What does the moon do?",
  "answerPromptText": "Secret Message:",
  "showInPrintedAnswer": true,
  "placement": "bottom",
  "cells": [[8,14],[9,14],[10,14],[12,14],[13,14],[14,14],[7,15],[8,15],[9,15],[10,15],[11,15]]
},
"puzzleGrid": [
  [null,null,null,null,null,null,null,"M","D","A","L","I","L",null,null,null],
  [null,null,null,null,"A","N","O","Q","D","T","R","A","M","A","Q",null],
  [null,null,null,"A","U","N","S","N","O","N","U","I","D","N","S","M"],
  [null,null,"H","L","D","I","O","S","D","N","I",null,null,null,null,null],
  [null,"C","I","Q","K","A","I","G","W",null,null,null,null,null,null],
  [null,"T","S","U","K","I","M","N","R",null,null,null,null,null,null],
  ["T","H","I","A","M","A","O","O",null,null,null,null,null,null,null],
  ["M","K","N","A","Q","O","N","S",null,null,null,null,null,null,null],
  ["U","O","Q","A","Q","N","Q","N",null,null,null,null,null,null,null],
  ["O","O","O","A","K","C","I","A",null,null,null,null,null,null,null],
  [null,"O","H","N","N","H","N","W","I",null,null,null,null,null,null],
  [null,"C","H","U","Q","A","S","H","L",null,null,null,null,null,null],
  [null,null,"L","H","G","A","D","C","O","A","W",null,null,null,null,null],
  [null,null,null,"L","S","N","D","S","C","C","N","K","I","U","O","I"],
  [null,null,null,null,"O","D","T","L","T","U","R","U","N","T","H",null],
  [null,null,null,null,null,null,null,"E","T","I","D","E","L",null,null,null]
]

```



```

    ],
    "answerLetters": {
        "showPuzzleLettersStyle": "dimmed",
        "dimPuzzleLetterColor": 4
    },
    "answerHighlights": {
        "style": "openOval",
        "lineWidth": 3,
        "openOvalWidth": 0.233141,
        "color": 7
    },
    "cellFloorColors": [
        [null,null,null,null,null,null,null, 6, 6, 6, 6, 6, 6,null,null,null],
        [null,null,null,null, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,null],
        [null,null,null, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6],
        [null,null, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null],
        [null, 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [null, 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [ 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [ 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [ 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [null, 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [null, 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null,null,null,null,null],
        [null,null, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,null,null,null],
        [null,null,null, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6],
        [null,null,null,null, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,null],
        [null,null,null,null,null,null,null, 6, 6, 6, 6, 6, 6, 6,null,null,null]
    ],
    "cellWalls": {
        "color": 5,
        "width": 4,
        "walls": [
            [0,0,0,1],[1,1,0,1],[1,1,1,0],[0,0,1,0],
            [1,0,1,1],[2,1,1,1],[2,1,2,0],[1,0,2,0],
            [2,0,2,1],[3,1,2,1],[3,1,3,0],[2,0,3,0],
            [3,0,3,1],[4,1,3,1],[4,1,4,0],[3,0,4,0],
            [4,0,4,1],[5,1,4,1],[5,1,5,0],[4,0,5,0],
            [5,0,5,1],[6,1,5,1],[6,1,6,0],[5,0,6,0],
            [6,0,6,1],[7,1,6,1],[7,1,7,0],[6,0,7,0],
            [7,0,7,1],[8,1,7,1],
            [9,1,8,1],
            [10,1,9,1],

```

```

[11,1,10,1],
[12,1,11,1],
[13,1,12,1], [13,1,13,0],
[13,0,13,1], [14,1,13,1], [14,1,14,0], [13,0,14,0],
[14,0,14,1], [15,1,14,1], [15,1,15,0], [14,0,15,0],
[15,0,15,1], [16,1,15,1], [16,1,16,0], [15,0,16,0],
[0,1,0,2], [1,2,0,2], [1,2,1,1], [0,1,1,1],
[1,1,1,2], [2,2,1,2], [2,2,2,1], [1,1,2,1],
[2,1,2,2], [3,2,2,2], [3,2,3,1], [2,1,3,1],
[3,1,3,2], [4,2,3,2], [4,2,4,1], [3,1,4,1],
[4,1,4,2], [5,2,4,2],
[6,2,5,2],
[7,2,6,2],
[14,2,13,2],
[15,2,14,2], [15,2,15,1],
[15,1,15,2], [16,2,15,2], [16,2,16,1], [15,1,16,1],
[0,2,0,3], [1,3,0,3], [1,3,1,2], [0,2,1,2],
[1,2,1,3], [2,3,1,3], [2,3,2,2], [1,2,2,2],
[2,2,2,3], [3,3,2,3], [3,3,3,2], [2,2,3,2],
[3,2,3,3], [4,3,3,3],
[11,2,12,2],
[12,2,13,2],
[13,2,14,2],
[14,2,15,2],
[16,3,15,3], [16,3,16,2], [15,2,16,2],
[0,3,0,4], [1,4,0,4], [1,4,1,3], [0,3,1,3],
[1,3,1,4], [2,4,1,4], [2,4,2,3], [1,3,2,3],
[2,3,2,4], [3,4,2,4],
[9,3,10,3],
[11,4,11,3], [10,3,11,3],
[11,3,11,4], [12,4,11,4], [12,4,12,3], [11,3,12,3],
[12,3,12,4], [13,4,12,4], [13,4,13,3], [12,3,13,3],
[13,3,13,4], [14,4,13,4], [14,4,14,3], [13,3,14,3],
[14,3,14,4], [15,4,14,4], [15,4,15,3], [14,3,15,3],
[15,3,15,4], [16,4,15,4], [16,4,16,3], [15,3,16,3],
[0,4,0,5], [1,5,0,5], [1,5,1,4], [0,4,1,4],
[1,4,1,5], [2,5,1,5],
[9,5,9,4],
[9,4,9,5], [10,5,9,5], [10,5,10,4], [9,4,10,4],
[10,4,10,5], [11,5,10,5], [11,5,11,4], [10,4,11,4],
[11,4,11,5], [12,5,11,5], [12,5,12,4], [11,4,12,4],
[12,4,12,5], [13,5,12,5], [13,5,13,4], [12,4,13,4],
[13,4,13,5], [14,5,13,5], [14,5,14,4], [13,4,14,4],

```

[14,4,14,5], [15,5,14,5], [15,5,15,4], [14,4,15,4],
 [15,4,15,5], [16,5,15,5], [16,5,16,4], [15,4,16,4],
 [0,5,0,6], [1,6,0,6], [1,6,1,5], [0,5,1,5],
 [1,5,1,6],
 [9,6,9,5], [8,5,9,5],
 [9,5,9,6], [10,6,9,6], [10,6,10,5], [9,5,10,5],
 [10,5,10,6], [11,6,10,6], [11,6,11,5], [10,5,11,5],
 [11,5,11,6], [12,6,11,6], [12,6,12,5], [11,5,12,5],
 [12,5,12,6], [13,6,12,6], [13,6,13,5], [12,5,13,5],
 [13,5,13,6], [14,6,13,6], [14,6,14,5], [13,5,14,5],
 [14,5,14,6], [15,6,14,6], [15,6,15,5], [14,5,15,5],
 [15,5,15,6], [16,6,15,6], [16,6,16,5], [15,5,16,5],
 [0,6,0,7], [1,7,0,7],
 [8,7,8,6],
 [8,6,8,7], [9,7,8,7], [9,7,9,6], [8,6,9,6],
 [9,6,9,7], [10,7,9,7], [10,7,10,6], [9,6,10,6],
 [10,6,10,7], [11,7,10,7], [11,7,11,6], [10,6,11,6],
 [11,6,11,7], [12,7,11,7], [12,7,12,6], [11,6,12,6],
 [12,6,12,7], [13,7,12,7], [13,7,13,6], [12,6,13,6],
 [13,6,13,7], [14,7,13,7], [14,7,14,6], [13,6,14,6],
 [14,6,14,7], [15,7,14,7], [15,7,15,6], [14,6,15,6],
 [15,6,15,7], [16,7,15,7], [16,7,16,6], [15,6,16,6],
 [0,7,0,8],
 [8,8,8,7],
 [8,7,8,8], [9,8,8,8], [9,8,9,7], [8,7,9,7],
 [9,7,9,8], [10,8,9,8], [10,8,10,7], [9,7,10,7],
 [10,7,10,8], [11,8,10,8], [11,8,11,7], [10,7,11,7],
 [11,7,11,8], [12,8,11,8], [12,8,12,7], [11,7,12,7],
 [12,7,12,8], [13,8,12,8], [13,8,13,7], [12,7,13,7],
 [13,7,13,8], [14,8,13,8], [14,8,14,7], [13,7,14,7],
 [14,7,14,8], [15,8,14,8], [15,8,15,7], [14,7,15,7],
 [15,7,15,8], [16,8,15,8], [16,8,16,7], [15,7,16,7],
 [0,8,0,9],
 [8,9,8,8],
 [8,8,8,9], [9,9,8,9], [9,9,9,8], [8,8,9,8],
 [9,8,9,9], [10,9,9,9], [10,9,10,8], [9,8,10,8],
 [10,8,10,9], [11,9,10,9], [11,9,11,8], [10,8,11,8],
 [11,8,11,9], [12,9,11,9], [12,9,12,8], [11,8,12,8],
 [12,8,12,9], [13,9,12,9], [13,9,13,8], [12,8,13,8],
 [13,8,13,9], [14,9,13,9], [14,9,14,8], [13,8,14,8],
 [14,8,14,9], [15,9,14,9], [15,9,15,8], [14,8,15,8],
 [15,8,15,9], [16,9,15,9], [16,9,16,8], [15,8,16,8],
 [0,9,0,10], [0,9,1,9],

[8,10,8,9],
 [8,9,8,10], [9,10,8,10], [9,10,9,9], [8,9,9,9],
 [9,9,9,10], [10,10,9,10], [10,10,10,9], [9,9,10,9],
 [10,9,10,10], [11,10,10,10], [11,10,11,9], [10,9,11,9],
 [11,9,11,10], [12,10,11,10], [12,10,12,9], [11,9,12,9],
 [12,9,12,10], [13,10,12,10], [13,10,13,9], [12,9,13,9],
 [13,9,13,10], [14,10,13,10], [14,10,14,9], [13,9,14,9],
 [14,9,14,10], [15,10,14,10], [15,10,15,9], [14,9,15,9],
 [15,9,15,10], [16,10,15,10], [16,10,16,9], [15,9,16,9],
 [0,10,0,11], [1,11,0,11], [1,11,1,10], [0,10,1,10],
 [1,10,1,11],
 [9,11,8,11], [9,11,9,10],
 [9,10,9,11], [10,11,9,11], [10,11,10,10], [9,10,10,10],
 [10,10,10,11], [11,11,10,11], [11,11,11,10], [10,10,11,10],
 [11,10,11,11], [12,11,11,11], [12,11,12,10], [11,10,12,10],
 [12,10,12,11], [13,11,12,11], [13,11,13,10], [12,10,13,10],
 [13,10,13,11], [14,11,13,11], [14,11,14,10], [13,10,14,10],
 [14,10,14,11], [15,11,14,11], [15,11,15,10], [14,10,15,10],
 [15,10,15,11], [16,11,15,11], [16,11,16,10], [15,10,16,10],
 [0,11,0,12], [1,12,0,12], [1,12,1,11], [0,11,1,11],
 [1,11,1,12], [1,11,2,11],
 [9,12,9,11],
 [9,11,9,12], [10,12,9,12], [10,12,10,11], [9,11,10,11],
 [10,11,10,12], [11,12,10,12], [11,12,11,11], [10,11,11,11],
 [11,11,11,12], [12,12,11,12], [12,12,12,11], [11,11,12,11],
 [12,11,12,12], [13,12,12,12], [13,12,13,11], [12,11,13,11],
 [13,11,13,12], [14,12,13,12], [14,12,14,11], [13,11,14,11],
 [14,11,14,12], [15,12,14,12], [15,12,15,11], [14,11,15,11],
 [15,11,15,12], [16,12,15,12], [16,12,16,11], [15,11,16,11],
 [0,12,0,13], [1,13,0,13], [1,13,1,12], [0,12,1,12],
 [1,12,1,13], [2,13,1,13], [2,13,2,12], [1,12,2,12],
 [2,12,2,13], [2,12,3,12],
 [10,13,9,13],
 [11,13,10,13], [11,13,11,12],
 [11,12,11,13], [12,13,11,13], [12,13,12,12], [11,12,12,12],
 [12,12,12,13], [13,13,12,13], [13,13,13,12], [12,12,13,12],
 [13,12,13,13], [14,13,13,13], [14,13,14,12], [13,12,14,12],
 [14,12,14,13], [15,13,14,13], [15,13,15,12], [14,12,15,12],
 [15,12,15,13], [16,13,15,13], [16,13,16,12], [15,12,16,12],
 [0,13,0,14], [1,14,0,14], [1,14,1,13], [0,13,1,13],
 [1,13,1,14], [2,14,1,14], [2,14,2,13], [1,13,2,13],
 [2,13,2,14], [3,14,2,14], [3,14,3,13], [2,13,3,13],
 [3,13,3,14], [3,13,4,13],

```

[12,14,11,14],
[13,14,12,14],
[14,14,13,14],
[15,14,14,14],
[16,14,15,14],[16,14,16,13],[15,13,16,13],
[0,14,0,15],[1,15,0,15],[1,15,1,14],[0,14,1,14],
[1,14,1,15],[2,15,1,15],[2,15,2,14],[1,14,2,14],
[2,14,2,15],[3,15,2,15],[3,15,3,14],[2,14,3,14],
[3,14,3,15],[4,15,3,15],[4,15,4,14],[3,14,4,14],
[4,14,4,15],[4,14,5,14],
[5,14,6,14],
[6,14,7,14],
[13,14,14,14],
[15,15,15,14],[14,14,15,14],
[15,14,15,15],[16,15,15,15],[16,15,16,14],[15,14,16,14],
[0,15,0,16],[1,16,0,16],[1,16,1,15],[0,15,1,15],
[1,15,1,16],[2,16,1,16],[2,16,2,15],[1,15,2,15],
[2,15,2,16],[3,16,2,16],[3,16,3,15],[2,15,3,15],
[3,15,3,16],[4,16,3,16],[4,16,4,15],[3,15,4,15],
[4,15,4,16],[5,16,4,16],[5,16,5,15],[4,15,5,15],
[5,15,5,16],[6,16,5,16],[6,16,6,15],[5,15,6,15],
[6,15,6,16],[7,16,6,16],[7,16,7,15],[6,15,7,15],
[7,15,7,16],[7,15,8,15],
[8,15,9,15],
[9,15,10,15],
[10,15,11,15],
[11,15,12,15],
[13,16,13,15],[12,15,13,15],
[13,15,13,16],[14,16,13,16],[14,16,14,15],[13,15,14,15],
[14,15,14,16],[15,16,14,16],[15,16,15,15],[14,15,15,15],
[15,15,15,16],[16,16,15,16],[16,16,16,15],[15,15,16,15] ]
},
"colorLUT": ["#006200ff", "#0c0081ff", "#000000ff", "#000000ff", "#999999ff", "#1e2b70ff",
"#ffffff", "#689ec8ff"],
"widthLUT": [0.000, 0.007, 0.014, 0.028, 0.042, 0.056, 0.069, 0.083, 0.097, 0.111],
"fontLUT": ["Crillee", "Tekton Pro", "Hannotate SC", "Futura"]
}

```

Maze (Puzzle Class 2)

WIP... RSN...

Given the following maze and its answer:

Here is the ESP file for the maze puzzle:

```
{
  "fileType": "schwansongs/espfile",
  "fileVersion": 1,
  "softwareName": "Minos Maze Maker Pro",
  "softwareVersion": "Version 2.2.2 (25082401)",
  "puzzleClass": 2,
  "puzzleClassName": "Maze",
  "puzzleGUID": "cnbn-tezw",
  "authorName": "",
  "authorCopyright": "Copyright (c) 2025, Your Company Here",
  "datePublished": "20250824",
  "exportInfo": {
    "exportImageDimension": {"width": 1200, "height": 1200},
    "exportFileFormat": "PNG",
    "exportFileNameLayoutStyle": "String+Counter+Format",
    "exportFileNameBase": "walltemplate",
    "exportFileNameExt": "png",
    "exportFileNamePuzzle": "walltemplate_1_Puz.png",
    "exportFileNameSolution": "walltemplate_1_Ans.png",
    "exportCounter": 1
  },
  "title": {
    "justification": "center",
    "font": 0,
    "color": 0,
    "relFontSize": 1
  },
  "puzzleDimension": {"width": 5, "height": 5},
  "puzzleShapeInfo": {
    "puzzleShapeName": "Square",
    "puzzleANShapeFont": 1
  },
  "difficulty": { "value": 10, "desc": "Basic Settings"},
  "cellInfo": {
    "cellShape": "Octo-Grid (Ortho)",
    "doCellLabeling": false,
    "doCellFloor": true,
    "cellFloorColor": 1
  }
}
```

```

    },
    "wallPathInfo": {
        "style": "allWalls",
        "wallOuterThickness": 5,
        "pathPuzzleColor": 3,
        "wallInnerThickness": 2,
        "pathPuzzleColor": 4,
        "openWallsAtPathEnds": true,
        "wallWiggleAmount": 0,
        "wallWiggleOuter": false,
        "pathRelThickness": 0,
        "pathPuzzleColor": 5,
        "pathSolutionColor": 6
    },
    "startEndMarks": {
        "showMarkers": "none",
        "arrowDisplayStyle": "thick-pointy",
        "markerRelSize": 0,
        "startMarker": {"locationXY": [0,0], "openingDirection": 1, "markerText": "S", "color": 7,
"font": 3},
        "endMarker": {"locationXY": [4,4], "openingDirection": 2, "markerText": "E", "color": 8,
"font": 3},
        "openWallsAtPathEnds": true
    },
    "mazePhrase": {
        "original": "where?",
        "stripped": "where?",
        "template": "_ _ _ _ ?",
        "prompt": "Hidden Maze-Phrase:\n_ _ _ _ ?",
        "promptStyle": "prompt+template",
        "exactFit": false,
        "addTemplateSpaces": true,
        "font": 4,
        "color": 10,
        "relFontSize": 0.0840127,
        "relVertAdjust": 0.4,
        "mazePhraseCellXY": [[0,0,"w"],[2,0,"h"],[2,1,"e"],[1,3,"r"],[2,3,"e"],[4,4,"?"]],
        "letterFillCellXY": [[4,1,"e"],[4,2,"w"],[0,3,"w"],[4,3,"w"]]
    },
    "cells": [
        [ 0, 0, 0],[ 1, 0, 1],[ 2, 0, 2],[ 3, 0, 3],[ 4, 0, 4],
        [ 0, 1, 5],[ 1, 1, 6],[ 2, 1, 7],[ 3, 1, 8],[ 4, 1, 9],
        [ 0, 2, 10],[ 1, 2, 11],[ 2, 2, 12],[ 3, 2, 13],[ 4, 2, 14],

```

```

    [ 0, 3, 15],[ 1, 3, 16],[ 2, 3, 17],[ 3, 3, 18],[ 4, 3, 19],
    [ 0, 4, 20],[ 1, 4, 21],[ 2, 4, 22],[ 3, 4, 23],[ 4, 4, 24]
  ],
  "cellWallsOuter": {
    "color": 3,
    "width": 5,
    "walls":
      [[0,0.25,0.25,0,0,null],[0.75,0,1,0.25,0,null],[0.25,1,0,0.75,0,null]
        , [1,0.25,1.25,0,1,null],[1.75,0,2,0.25,1,null],[2,0.25,2.25,0,2,null],[2.75,0,3,0.25,2,null]
        , [3,0.25,3.25,0,3,null],[3.75,0,4,0.25,3,null],[4,0.25,4.25,0,4,null],[4.75,0,5,0.25,4,null]
        , [5,0.75,4.75,1,4,null],[0,1.25,0.25,1,5,null],[0.25,2,0,1.75,5,null],[4.75,1,5,1.25,9,null]
        , [5,1.75,4.75,2,9,null],[0,2.25,0.25,2,10,null],[0.25,3,0,2.75,10,null],
      [4.75,2,5,2.25,14,null]
        , [5,2.75,4.75,3,14,null],[0,3.25,0.25,3,15,null],[0.25,4,0,3.75,15,null],
      [4.75,3,5,3.25,19,null]
        , [5,3.75,4.75,4,19,null],[0,4.25,0.25,4,20,null],[1,4.75,0.75,5,20,null],
      [0.25,5,0,4.75,20,null]
        , [2,4.75,1.75,5,21,null],[1.25,5,1,4.75,21,null],[3,4.75,2.75,5,22,null],
      [2.25,5,2,4.75,22,null]
        , [4,4.75,3.75,5,23,null],[3.25,5,3,4.75,23,null],[4.75,4,5,4.25,24,null],
      [5,4.75,4.75,5,24,null]
        , [4.25,5,4,4.75,24,null],
      [0,0.75,0,0.25,0,0],[1.25,0,1.75,0,1,0],[2.25,0,2.75,0,2,0]
        , [3.25,0,3.75,0,3,0],[4.25,0,4.75,0,4,0],[5,0.25,5,0.75,4,0],[0,1.75,0,1.25,5,0]
        , [5,1.25,5,1.75,9,0],[0,2.75,0,2.25,10,0],[5,2.25,5,2.75,14,0],[0,3.75,0,3.25,15,0]
        , [5,3.25,5,3.75,19,0],[0,4.75,0,4.25,20,0],[0.75,5,0.25,5,20,0],[1.75,5,1.25,5,21,0]
        , [2.75,5,2.25,5,22,0],[3.75,5,3.25,5,23,0],[4.75,5,4.25,5,24,0]
      ]
  },
  "cellWallsInner": {
    "color": 4,
    "width": 2,
    "walls":
      [[1,0.75,0.75,1,0,null],[2,0.75,1.75,1,1,null],[1.25,1,1,0.75,1,null]
        , [3,0.75,2.75,1,2,null],[2.25,1,2,0.75,2,null],[4,0.75,3.75,1,3,null],[3.25,1,3,0.75,3,null]
        , [4.25,1,4,0.75,4,null],[0.75,1,1,1.25,5,null],[1,1.75,0.75,2,5,null],[1,1.25,1.25,1,6,null]
        , [1.75,1,2,1.25,6,null],[2,1.75,1.75,2,6,null],[1.25,2,1,1.75,6,null],[2,1.25,2.25,1,7,null]
        , [2.75,1,3,1.25,7,null],[3,1.75,2.75,2,7,null],[2.25,2,2,1.75,7,null],[3,1.25,3.25,1,8,null]
        , [3.75,1,4,1.25,8,null],[4,1.75,3.75,2,8,null],[3.25,2,3,1.75,8,null],[4,1.25,4.25,1,9,null]
        , [4.25,2,4,1.75,9,null],[0.75,2,1,2.25,10,null],[1,2.75,0.75,3,10,null],
      [1,2.25,1.25,2,11,null]
        , [1.75,2,2,2.25,11,null],[2,2.75,1.75,3,11,null],[1.25,3,1,2.75,11,null],
      [2,2.25,2.25,2,12,null]
        , [2.75,2,3,2.25,12,null],[3,2.75,2.75,3,12,null],[2.25,3,2,2.75,12,null],
      [3,2.25,3.25,2,13,null]
      ]
  }
}

```



```

    , [3.75, 2, 4, 2.25, 13, null], [4, 2.75, 3.75, 3, 13, null], [3.25, 3, 3, 2.75, 13, null],
    [4, 2.25, 4.25, 2, 14, null]
    , [4.25, 3, 4, 2.75, 14, null], [0.75, 3, 1, 3.25, 15, null], [1, 3.75, 0.75, 4, 15, null],
    [1, 3.25, 1.25, 3, 16, null]
    , [1.75, 3, 2, 3.25, 16, null], [2, 3.75, 1.75, 4, 16, null], [1.25, 4, 1, 3.75, 16, null],
    [2, 3.25, 2.25, 3, 17, null]
    , [2.75, 3, 3, 3.25, 17, null], [3, 3.75, 2.75, 4, 17, null], [2.25, 4, 2, 3.75, 17, null],
    [3, 3.25, 3.25, 3, 18, null]
    , [3.75, 3, 4, 3.25, 18, null], [4, 3.75, 3.75, 4, 18, null], [3.25, 4, 3, 3.75, 18, null],
    [4, 3.25, 4.25, 3, 19, null]
    , [4.25, 4, 4, 3.75, 19, null], [0.75, 4, 1, 4.25, 20, null], [1, 4.25, 1.25, 4, 21, null],
    [1.75, 4, 2, 4.25, 21, null]
    , [2, 4.25, 2.25, 4, 22, null], [2.75, 4, 3, 4.25, 22, null], [3, 4.25, 3.25, 4, 23, null],
    [3.75, 4, 4, 4.25, 23, null]
    , [4, 4.25, 4.25, 4, 24, null],
    [0.75, 1, 0.25, 1, 0, 5], [1.75, 1, 1.25, 1, 1, 6], [2.75, 1, 2.25, 1, 2, 7]
    , [4, 0.25, 4, 0.75, 3, 4], [1, 1.25, 1, 1.75, 5, 6], [2.75, 2, 2.25, 2, 7, 12], [3.75, 2, 3.25, 2, 8, 13]
    , [4.75, 2, 4.25, 2, 9, 14], [1, 2.25, 1, 2.75, 10, 11], [2, 2.25, 2, 2.75, 11, 12], [3, 2.25, 3, 2.75, 12, 13]
    , [1, 3.25, 1, 3.75, 15, 16], [2, 3.25, 2, 3.75, 16, 17], [4, 3.25, 4, 3.75, 18, 19], [4.75, 4, 4.25, 4, 19, 24]
    , [3, 4.25, 3, 4.75, 22, 23]
  ]
},
"solution": {
  "pathStyle": "Maze-Phrase Letters & Line",
  "pathColor": 9,
  "relWidth": -0.205695,
  "cells": [[0, 0], [1, 0], [2, 0], [3, 0], [3, 1], [2, 1], [1, 1], [1, 2], [1, 3], [1, 4], [2, 4], [2, 3], [3, 3],
    [3, 4], [4, 4]]
},
"colorLUT": ["#400047ff", "#c0ffffff", "#000000ff", "#ff0900ff", "#7400a1ff", "#434343ff",
  "#999999ff", "#0000feff", "#fb0006ff", "#ffc21cff", "#000000ff"],
"widthLUT": [0.050, 0.100, 0.200, 0.300, 0.400, 0.500, 0.600, 0.700, 0.800],
"fontLUT": ["HarlequinFLF", "Times New Roman Bold", "Menlo", "Short Stack", "Hannotate SC"]
}

```

Sudoku (Puzzle Class 3)

Here is an example JSON file for a sudoku puzzle: