

# Simon Tatham's Portable Puzzle Collection

---

This is a collection of small one-player puzzle games.

This manual is copyright 2004-2021 Simon Tatham. All rights reserved. You may distribute this documentation under the MIT licence. See appendix A for the licence text in full.

# Contents

---

Chapter 1: Introduction . . . . .	7
Chapter 2: Common features . . . . .	8
2.1 Common actions . . . . .	8
2.2 Specifying games with the game ID . . . . .	9
2.3 The ‘Type’ menu . . . . .	10
2.4 Specifying game parameters on the command line . . . . .	10
2.5 Unix command-line options . . . . .	11
Chapter 3: Net . . . . .	14
3.1 Net controls . . . . .	14
3.2 Net parameters . . . . .	15
Chapter 4: Cube . . . . .	16
4.1 Cube controls . . . . .	16
4.2 Cube parameters . . . . .	16
Chapter 5: Fifteen . . . . .	17
5.1 Fifteen controls . . . . .	17
5.2 Fifteen parameters . . . . .	17
Chapter 6: Sixteen . . . . .	18
6.1 Sixteen controls . . . . .	18
6.2 Sixteen parameters . . . . .	18
Chapter 7: Twiddle . . . . .	19
7.1 Twiddle controls . . . . .	19
7.2 Twiddle parameters . . . . .	19
Chapter 8: Rectangles . . . . .	21
8.1 Rectangles controls . . . . .	21
8.2 Rectangles parameters . . . . .	21

Chapter 9: Netslide . . . . .	23
Chapter 10: Pattern . . . . .	24
10.1 Pattern controls . . . . .	24
10.2 Pattern parameters . . . . .	24
Chapter 11: Solo . . . . .	25
11.1 Solo controls . . . . .	25
11.2 Solo parameters . . . . .	26
Chapter 12: Mines . . . . .	28
12.1 Mines controls . . . . .	28
12.2 Mines parameters . . . . .	29
Chapter 13: Same Game . . . . .	30
13.1 Same Game controls . . . . .	30
13.2 Same Game parameters . . . . .	30
Chapter 14: Flip . . . . .	32
14.1 Flip controls . . . . .	32
14.2 Flip parameters . . . . .	32
Chapter 15: Guess . . . . .	33
15.1 Guess controls . . . . .	33
15.2 Guess parameters . . . . .	33
Chapter 16: Pegs . . . . .	35
16.1 Pegs controls . . . . .	35
16.2 Pegs parameters . . . . .	35
Chapter 17: Dominosa . . . . .	36
17.1 Dominosa controls . . . . .	36
17.2 Dominosa parameters . . . . .	36
Chapter 18: Untangle . . . . .	37
18.1 Untangle controls . . . . .	37
18.2 Untangle parameters . . . . .	37
Chapter 19: Black Box . . . . .	38
19.1 Black Box controls . . . . .	39

19.2 Black Box parameters . . . . .	40
Chapter 20: Slant . . . . .	41
20.1 Slant controls . . . . .	41
20.2 Slant parameters . . . . .	41
Chapter 21: Light Up . . . . .	43
21.1 Light Up controls . . . . .	43
21.2 Light Up parameters . . . . .	43
Chapter 22: Map . . . . .	45
22.1 Map controls . . . . .	45
22.2 Map parameters . . . . .	46
Chapter 23: Loopy . . . . .	47
23.1 Loopy controls . . . . .	47
23.2 Loopy parameters . . . . .	47
Chapter 24: Inertia . . . . .	49
24.1 Inertia controls . . . . .	49
24.2 Inertia parameters . . . . .	49
Chapter 25: Tents . . . . .	50
25.1 Tents controls . . . . .	50
25.2 Tents parameters . . . . .	50
Chapter 26: Bridges . . . . .	52
26.1 Bridges controls . . . . .	52
26.2 Bridges parameters . . . . .	53
Chapter 27: Unequal . . . . .	55
27.1 Unequal controls . . . . .	55
27.2 Unequal parameters . . . . .	56
Chapter 28: Galaxies . . . . .	57
28.1 Galaxies controls . . . . .	57
28.2 Galaxies parameters . . . . .	57
Chapter 29: Filling . . . . .	59
29.1 Filling controls . . . . .	59

29.2 Filling parameters . . . . .	59
Chapter 30: Keen . . . . .	60
30.1 Keen controls . . . . .	60
30.2 Keen parameters . . . . .	61
Chapter 31: Towers . . . . .	62
31.1 Towers controls . . . . .	62
31.2 Towers parameters . . . . .	63
Chapter 32: Singles . . . . .	64
32.1 Singles controls . . . . .	64
32.2 Singles parameters . . . . .	64
Chapter 33: Magnets . . . . .	65
33.1 Magnets controls . . . . .	65
33.2 Magnets parameters . . . . .	65
Chapter 34: Signpost . . . . .	67
34.1 Signpost controls . . . . .	67
34.2 Signpost parameters . . . . .	68
Chapter 35: Range . . . . .	69
35.1 Range controls . . . . .	69
35.2 Range parameters . . . . .	69
Chapter 36: Pearl . . . . .	70
36.1 Pearl controls . . . . .	70
36.2 Pearl parameters . . . . .	71
Chapter 37: Undead . . . . .	72
37.1 Undead controls . . . . .	72
37.2 Undead parameters . . . . .	73
Chapter 38: Unruly . . . . .	74
38.1 Unruly controls . . . . .	74
38.2 Unruly parameters . . . . .	74
Chapter 39: Flood . . . . .	75
39.1 Flood controls . . . . .	75

39.2 Flood parameters . . . . .	75
Chapter 40: Tracks . . . . .	77
40.1 Tracks controls . . . . .	77
40.2 Tracks parameters . . . . .	77
Chapter 41: Palisade . . . . .	78
41.1 Palisade controls . . . . .	78
41.2 Palisade parameters . . . . .	78
Chapter 42: Mosaic . . . . .	79
42.1 Mosaic controls . . . . .	79
42.2 Mosaic parameters . . . . .	79
Appendix A: Licence . . . . .	80
Index . . . . .	81

# Chapter 1: Introduction

---

I wrote this collection because I thought there should be more small desktop toys available: little games you can pop up in a window and play for two or three minutes while you take a break from whatever else you were doing. And I was also annoyed that every time I found a good game on (say) Unix, it wasn't available the next time I was sitting at a Windows machine, or vice versa; so I arranged that everything in my personal puzzle collection will happily run on both, and have more recently done a port to Mac OS X as well. When I find (or perhaps invent) further puzzle games that I like, they'll be added to this collection and will immediately be available on both platforms. And if anyone feels like writing any other front ends – PocketPC, Mac OS pre-10, or whatever it might be – then all the games in this framework will immediately become available on another platform as well.

The actual games in this collection were mostly not my invention; they are re-implementations of existing game concepts within my portable puzzle framework. I do not claim credit, in general, for inventing the rules of any of these puzzles. (I don't even claim authorship of all the code; some of the puzzles have been submitted by other authors.)

This collection is distributed under the MIT licence (see appendix A). This means that you can do pretty much anything you like with the game binaries or the code, except pretending you wrote them yourself, or suing me if anything goes wrong.

The most recent versions, and source code, can be found at <https://www.chiark.greenend.org.uk/~sgtatham/puzzles/>.

Please report bugs to [anakin@pobox.com](mailto:anakin@pobox.com). You might find it helpful to read this article before reporting a bug:

<https://www.chiark.greenend.org.uk/~sgtatham/bugs.html>

Patches are welcome. Especially if they provide a new front end (to make all these games run on another platform), or a new game.

## Chapter 2: Common features

---

This chapter describes features that are common to all the games.

### 2.1 Common actions

These actions are all available from the ‘Game’ menu and via keyboard shortcuts, in addition to any game-specific actions.

(On Mac OS X, to conform with local user interface standards, these actions are situated on the ‘File’ and ‘Edit’ menus instead.)

*New game* (‘N’, Ctrl+‘N’)

Starts a new game, with a random initial state.

*Restart game*

Resets the current game to its initial state. (This can be undone.)

*Load*

Loads a saved game from a file on disk.

*Save*

Saves the current state of your game to a file on disk.

The Load and Save operations preserve your entire game history (so you can save, reload, and still Undo and Redo things you had done before saving).

*Print*

Where supported (currently only on Windows), brings up a dialog allowing you to print an arbitrary number of puzzles randomly generated from the current parameters, optionally including the current puzzle. (Only for puzzles which make sense to print, of course – it's hard to think of a sensible printable representation of Fifteen!)

*Undo* (‘U’, Ctrl+‘Z’, Ctrl+‘\_’)

Undoes a single move. (You can undo moves back to the start of the session.)

*Redo* (‘R’, Ctrl+‘R’)

Redoes a previously undone move.

*Copy*

Copies the current state of your game to the clipboard in text format, so that you can paste



it into (say) an e-mail client or a web message board if you're discussing the game with someone else. (Not all games support this feature.)

### *Solve*

Transforms the puzzle instantly into its solved state. For some games (Cube) this feature is not supported at all because it is of no particular use. For other games (such as Pattern), the solved state can be used to give you information, if you can't see how a solution can exist at all or you want to know where you made a mistake. For still other games (such as Sixteen), automatic solution tells you nothing about how to *get* to the solution, but it does provide a useful way to get there quickly so that you can experiment with set-piece moves and transformations.

Some games (such as Solo) are capable of solving a game ID you have typed in from elsewhere. Other games (such as Rectangles) cannot solve a game ID they didn't invent themselves, but when they did invent the game ID they know what the solution is already. Still other games (Pattern) can solve *some* external game IDs, but only if they aren't too difficult.

The 'Solve' command adds the solved state to the end of the undo chain for the puzzle. In other words, if you want to go back to solving it yourself after seeing the answer, you can just press Undo.

### *Quit* ('Q', Ctrl+'Q')

Closes the application entirely.

## 2.2 Specifying games with the game ID

There are two ways to save a game specification out of a puzzle and recreate it later, or recreate it in somebody else's copy of the same puzzle.

The 'Specific' and 'Random Seed' options from the 'Game' menu (or the 'File' menu, on Mac OS X) each show a piece of text (a 'game ID') which is sufficient to reconstruct precisely the same game at a later date.

You can enter either of these pieces of text back into the program (via the same 'Specific' or 'Random Seed' menu options) at a later point, and it will recreate the same game. You can also use either one as a command line argument (on Windows or Unix); see section 2.4 for more detail.

The difference between the two forms is that a descriptive game ID is a literal *description* of the initial state of the game, whereas a random seed is just a piece of arbitrary text which was provided as input to the random number generator used to create the puzzle. This means that:

- Descriptive game IDs tend to be longer in many puzzles (although some, such as Cube (chapter 4), only need very short descriptions). So a random seed is often a *quicker* way to note down the puzzle you're currently playing, or to tell it to somebody else so they can play the same one as you.
- Any text at all is a valid random seed. The automatically generated ones are fifteen-digit numbers, but anything will do; you can type in your full name, or a word you just made up, and a valid puzzle will be generated from it. This provides a way for two or more people to race to complete the same puzzle: you think of a random seed, then everybody types it

in at the same time, and nobody has an advantage due to having seen the generated puzzle before anybody else.

- It is often possible to convert puzzles from other sources (such as ‘nonograms’ or ‘sudoku’ from newspapers) into descriptive game IDs suitable for use with these programs.
- Random seeds are not guaranteed to produce the same result if you use them with a different *version* of the puzzle program. This is because the generation algorithm might have been improved or modified in later versions of the code, and will therefore produce a different result when given the same sequence of random numbers. Use a descriptive game ID if you aren't sure that it will be used on the same version of the program as yours.

(Use the ‘About’ menu option to find out the version number of the program. Programs with the same version number running on different platforms should still be random-seed compatible.)

A descriptive game ID starts with a piece of text which encodes the *parameters* of the current game (such as grid size). Then there is a colon, and after that is the description of the game's initial state. A random seed starts with a similar string of parameters, but then it contains a hash sign followed by arbitrary data.

If you enter a descriptive game ID, the program will not be able to show you the random seed which generated it, since it wasn't generated *from* a random seed. If you *enter* a random seed, however, the program will be able to show you the descriptive game ID derived from that random seed.

Note that the game parameter strings are not always identical between the two forms. For some games, there will be parameter data provided with the random seed which is not included in the descriptive game ID. This is because that parameter information is only relevant when *generating* puzzle grids, and is not important when playing them. Thus, for example, the difficulty level in Solo (chapter 11) is not mentioned in the descriptive game ID.

These additional parameters are also not set permanently if you type in a game ID. For example, suppose you have Solo set to ‘Advanced’ difficulty level, and then a friend wants your help with a ‘Trivial’ puzzle; so the friend reads out a random seed specifying ‘Trivial’ difficulty, and you type it in. The program will generate you the same ‘Trivial’ grid which your friend was having trouble with, but once you have finished playing it, when you ask for a new game it will automatically go back to the ‘Advanced’ difficulty which it was previously set on.

## 2.3 The ‘Type’ menu

The ‘Type’ menu, if present, may contain a list of preset game settings. Selecting one of these will start a new random game with the parameters specified.

The ‘Type’ menu may also contain a ‘Custom’ option which allows you to fine-tune game parameters. The parameters available are specific to each game and are described in the following sections.

## 2.4 Specifying game parameters on the command line

(This section does not apply to the Mac OS X version.)

The games in this collection deliberately do not ever save information on to the computer they run on: they have no high score tables and no saved preferences. (This is because I expect at

least some people to play them at work, and those people will probably appreciate leaving as little evidence as possible!)

However, if you do want to arrange for one of these games to default to a particular set of parameters, you can specify them on the command line.

The easiest way to do this is to set up the parameters you want using the ‘Type’ menu (see section 2.3), and then to select ‘Random Seed’ from the ‘Game’ or ‘File’ menu (see section 2.2). The text in the ‘Game ID’ box will be composed of two parts, separated by a hash. The first of these parts represents the game parameters (the size of the playing area, for example, and anything else you set using the ‘Type’ menu).

If you run the game with just that parameter text on the command line, it will start up with the settings you specified.

For example: if you run Cube (see chapter 4), select ‘Octahedron’ from the ‘Type’ menu, and then go to the game ID selection, you will see a string of the form ‘o2x2#338686542711620’. Take only the part before the hash (‘o2x2’), and start Cube with that text on the command line: ‘PREFIX-cube o2x2’.

If you copy the *entire* game ID on to the command line, the game will start up in the specific game that was described. This is occasionally a more convenient way to start a particular game ID than by pasting it into the game ID selection box.

(You could also retrieve the encoded game parameters using the ‘Specific’ menu option instead of ‘Random Seed’, but if you do then some options, such as the difficulty level in Solo, will be missing. See section 2.2 for more details on this.)

## 2.5 Unix command-line options

(This section only applies to the Unix port.)

In addition to being able to specify game parameters on the command line (see section 2.4), there are various other options:

`--game`

`--load`

These options respectively determine whether the command-line argument is treated as specifying game parameters or a save file to load. Only one should be specified. If neither of these options is specified, a guess is made based on the format of the argument.

`--generate n`

If this option is specified, instead of a puzzle being displayed, a number of descriptive game IDs will be invented and printed on standard output. This is useful for gaining access to the game generation algorithms without necessarily using the frontend.

If game parameters are specified on the command-line, they will be used to generate the game IDs; otherwise a default set of parameters will be used.

The most common use of this option is in conjunction with `--print`, in which case its behaviour is slightly different; see below.

`--print wxh`

If this option is specified, instead of a puzzle being displayed, a printed representation of one or more unsolved puzzles is sent to standard output, in PostScript format.

On each page of puzzles, there will be  $w$  across and  $h$  down. If there are more puzzles than  $w \times h$ , more than one page will be printed.

If `--generate` has also been specified, the invented game IDs will be used to generate the printed output. Otherwise, a list of game IDs is expected on standard input (which can be descriptive or random seeds; see section 2.2), in the same format produced by `--generate`.

For example:

```
PREFIX-net --generate 12 --print 2x3 7x7w | lpr
```

will generate two pages of printed Net puzzles (each of which will have a  $7 \times 7$  wrapping grid), and pipe the output to the `lpr` command, which on many systems will send them to an actual printer.

There are various other options which affect printing; see below.

`--save file-prefix [ --save-suffix file-suffix ]`

If this option is specified, instead of a puzzle being displayed, saved-game files for one or more unsolved puzzles are written to files constructed from the supplied prefix and/or suffix.

If `--generate` has also been specified, the invented game IDs will be used to generate the printed output. Otherwise, a list of game IDs is expected on standard input (which can be descriptive or random seeds; see section 2.2), in the same format produced by `--generate`.

For example:

```
PREFIX-net --generate 12 --save game --save-suffix .sav
```

will generate twelve Net saved-game files with the names `game0.sav` to `game11.sav`.

`--version`

Prints version information about the game, and then quits.

The following options are only meaningful if `--print` is also specified:

`--with-solutions`

The set of pages filled with unsolved puzzles will be followed by the solutions to those puzzles.

`--scale n`

Adjusts how big each puzzle is when printed. Larger numbers make puzzles bigger; the default is 1.0.

--colour

Puzzles will be printed in colour, rather than in black and white (if supported by the puzzle).

## Chapter 3: Net

---

(*Note:* the Windows version of this game is called `NETGAME.EXE` to avoid clashing with Windows's own `NET.EXE`.)

I originally saw this in the form of a Flash game called FreeNet [1], written by Pavils Jurjans; there are several other implementations under the name NetWalk. The computer prepares a network by connecting up the centres of squares in a grid, and then shuffles the network by rotating every tile randomly. Your job is to rotate it all back into place. The successful solution will be an entirely connected network, with no closed loops. As a visual aid, all tiles which are connected to the one in the middle are highlighted.

[1] <http://www.jurjans.lv/stuff/net/FreeNet.htm>

### 3.1 Net controls

This game can be played with either the keyboard or the mouse. The controls are:

*Select tile:* mouse pointer, arrow keys

*Rotate tile anticlockwise:* left mouse button, 'A' key

*Rotate tile clockwise:* right mouse button, 'D' key

*Rotate tile by 180 degrees:* 'F' key

*Lock (or unlock) tile:* middle mouse button, shift-click, 'S' key

You can lock a tile once you're sure of its orientation. You can also unlock it again, but while it's locked you can't accidentally turn it.

The following controls are not necessary to complete the game, but may be useful:

*Shift grid:* Shift + arrow keys

On grids that wrap, you can move the origin of the grid, so that tiles that were on opposite sides of the grid can be seen together.

*Move centre:* Ctrl + arrow keys

You can change which tile is used as the source of highlighting. (It doesn't ultimately matter which tile this is, as every tile will be connected to every other tile in a correct solution, but it may be helpful in the intermediate stages of solving the puzzle.)

*Jumble tiles:* 'J' key

This key turns all tiles that are not locked to random orientations.

(All the actions described in section 2.1 are also available.)

## 3.2 Net parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

### *Width, Height*

Size of grid in tiles.

### *Walls wrap around*

If checked, flow can pass from the left edge to the right edge, and from top to bottom, and vice versa.

### *Barrier probability*

A number between 0.0 and 1.0 controlling whether an immovable barrier is placed between two tiles to prevent flow between them (a higher number gives more barriers). Since barriers are immovable, they act as constraints on the solution (i.e., hints).

The grid generation in Net has been carefully arranged so that the barriers are independent of the rest of the grid. This means that if you note down the random seed used to generate the current puzzle (see section 2.2), change the *Barrier probability* parameter, and then re-enter the same random seed, you should see exactly the same starting grid, with the only change being the number of barriers. So if you're stuck on a particular grid and need a hint, you could start up another instance of Net, set up the same parameters but a higher barrier probability, and enter the game seed from the original Net window.

### *Ensure unique solution*

Normally, Net will make sure that the puzzles it presents have only one solution. Puzzles with ambiguous sections can be more difficult and more subtle, so if you like you can turn off this feature and risk having ambiguous puzzles. (Also, finding *all* the possible solutions can be an additional challenge for an advanced player.)

## Chapter 4: Cube

---

This is another one I originally saw as a web game. This one was a Java game [2], by Paul Scott. You have a grid of 16 squares, six of which are blue; on one square rests a cube. Your move is to use the arrow keys to roll the cube through 90 degrees so that it moves to an adjacent square. If you roll the cube on to a blue square, the blue square is picked up on one face of the cube; if you roll a blue face of the cube on to a non-blue square, the blueness is put down again. (In general, whenever you roll the cube, the two faces that come into contact swap colours.) Your job is to get all six blue squares on to the six faces of the cube at the same time. Count your moves and try to do it in as few as possible.

Unlike the original Java game, my version has an additional feature: once you've mastered the game with a cube rolling on a square grid, you can change to a triangular grid and roll any of a tetrahedron, an octahedron or an icosahedron.

[2] <http://www3.sympatico.ca/paulscott/cube/cube.htm>

### 4.1 Cube controls

This game can be played with either the keyboard or the mouse.

Left-clicking anywhere on the window will move the cube (or other solid) towards the mouse pointer.

The arrow keys can also be used to roll the cube on its square grid in the four cardinal directions. On the triangular grids, the mapping of arrow keys to directions is more approximate. Vertical movement is disallowed where it doesn't make sense. The four keys surrounding the arrow keys on the numeric keypad ('7', '9', '1', '3') can be used for diagonal movement.

(All the actions described in section 2.1 are also available.)

### 4.2 Cube parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

#### *Type of solid*

Selects the solid to roll (and hence the shape of the grid): tetrahedron, cube, octahedron, or icosahedron.

#### *Width / top, Height / bottom*

On a square grid, horizontal and vertical dimensions. On a triangular grid, the number of triangles on the top and bottom rows respectively.



## Chapter 5: Fifteen

---

The old ones are the best: this is the good old ‘15-puzzle’ with sliding tiles. You have a 4×4 square grid; 15 squares contain numbered tiles, and the sixteenth is empty. Your move is to choose a tile next to the empty space, and slide it into the space. The aim is to end up with the tiles in numerical order, with the space in the bottom right (so that the top row reads 1,2,3,4 and the bottom row reads 13,14,15,*space*).

### 5.1 Fifteen controls

This game can be controlled with the mouse or the keyboard.

A left-click with the mouse in the row or column containing the empty space will move as many tiles as necessary to move the space to the mouse pointer.

The arrow keys will move a tile adjacent to the space in the direction indicated (moving the space in the *opposite* direction).

Pressing ‘h’ will make a suggested move. Pressing ‘h’ enough times will solve the game, but it may scramble your progress while doing so.

(All the actions described in section 2.1 are also available.)

### 5.2 Fifteen parameters

The only options available from the ‘Custom...’ option on the ‘Type’ menu are *Width* and *Height*, which are self-explanatory. (Once you've changed these, it's not a ‘15-puzzle’ any more, of course!)

## Chapter 6: Sixteen

---

Another sliding tile puzzle, visually similar to Fifteen (see chapter 5) but with a different type of move. This time, there is no hole: all 16 squares on the grid contain numbered squares. Your move is to shift an entire row left or right, or shift an entire column up or down; every time you do that, the tile you shift off the grid re-appears at the other end of the same row, in the space you just vacated. To win, arrange the tiles into numerical order (1,2,3,4 on the top row, 13,14,15,16 on the bottom). When you've done that, try playing on different sizes of grid.

I *might* have invented this game myself, though only by accident if so (and I'm sure other people have independently invented it). I thought I was imitating a screensaver I'd seen, but I have a feeling that the screensaver might actually have been a Fifteen-type puzzle rather than this slightly different kind. So this might be the one thing in my puzzle collection which represents creativity on my part rather than just engineering.

### 6.1 Sixteen controls

Left-clicking on an arrow will move the appropriate row or column in the direction indicated. Right-clicking will move it in the opposite direction.

Alternatively, use the cursor keys to move the position indicator around the edge of the grid, and use the return key to move the row/column in the direction indicated.

You can also move the tiles directly. Move the cursor onto a tile, hold Control and press an arrow key to move the tile under the cursor and move the cursor along with the tile. Or, hold Shift to move only the tile. Pressing Enter simulates holding down Control (press Enter again to release), while pressing Space simulates holding down shift.

(All the actions described in section 2.1 are also available.)

### 6.2 Sixteen parameters

The parameters available from the 'Custom...' option on the 'Type' menu are:

- *Width* and *Height*, which are self-explanatory.
- You can ask for a limited shuffling operation to be performed on the grid. By default, Sixteen will shuffle the grid in such a way that any arrangement is about as probable as any other. You can override this by requesting a precise number of shuffling moves to be performed. Typically your aim is then to determine the precise set of shuffling moves and invert them exactly, so that you answer (say) a four-move shuffle with a four-move solution. Note that the more moves you ask for, the more likely it is that solutions shorter than the target length will turn out to be possible.

## Chapter 7: Twiddle

---

Twiddle is a tile-rearrangement puzzle, visually similar to Sixteen (see chapter 6): you are given a grid of square tiles, each containing a number, and your aim is to arrange the numbers into ascending order.

In basic Twiddle, your move is to rotate a square group of four tiles about their common centre. (Orientation is not significant in the basic puzzle, although you can select it.) On more advanced settings, you can rotate a larger square group of tiles.

I first saw this type of puzzle in the GameCube game ‘Metroid Prime 2’. In the Main Gyro Chamber in that game, there is a puzzle you solve to unlock a door, which is a special case of Twiddle. I developed this game as a generalisation of that puzzle.

### 7.1 Twiddle controls

To play Twiddle, click the mouse in the centre of the square group you wish to rotate. In the basic mode, you rotate a  $2 \times 2$  square, which means you have to click at a corner point where four tiles meet.

In more advanced modes you might be rotating  $3 \times 3$  or even more at a time; if the size of the square is odd then you simply click in the centre tile of the square you want to rotate.

Clicking with the left mouse button rotates the group anticlockwise. Clicking with the right button rotates it clockwise.

You can also move an outline square around the grid with the cursor keys; the square is the size above ( $2 \times 2$  by default, or larger). Pressing the return key or space bar will rotate the current square anticlockwise or clockwise respectively.

(All the actions described in section 2.1 are also available.)

### 7.2 Twiddle parameters

Twiddle provides several configuration options via the ‘Custom’ option on the ‘Type’ menu:

- You can configure the width and height of the puzzle grid.
- You can configure the size of square block that rotates at a time.
- You can ask for every square in the grid to be distinguishable (the default), or you can ask for a simplified puzzle in which there are groups of identical numbers. In the simplified puzzle your aim is just to arrange all the 1s into the first row, all the 2s into the second row, and so on.
- You can configure whether the orientation of tiles matters. If you ask for an orientable puzzle, each tile will have a triangle drawn in it. All the triangles must be pointing upwards

to complete the puzzle.

- You can ask for a limited shuffling operation to be performed on the grid. By default, Twiddle will shuffle the grid so much that any arrangement is about as probable as any other. You can override this by requesting a precise number of shuffling moves to be performed. Typically your aim is then to determine the precise set of shuffling moves and invert them exactly, so that you answer (say) a four-move shuffle with a four-move solution. Note that the more moves you ask for, the more likely it is that solutions shorter than the target length will turn out to be possible.

## Chapter 8: Rectangles

---

You have a grid of squares, with numbers written in some (but not all) of the squares. Your task is to subdivide the grid into rectangles of various sizes, such that (a) every rectangle contains exactly one numbered square, and (b) the area of each rectangle is equal to the number written in its numbered square.

Credit for this game goes to the Japanese puzzle magazine Nikoli [3]; I've also seen a Palm implementation at Puzzle Palace [4]. Unlike Puzzle Palace's implementation, my version automatically generates random grids of any size you like. The quality of puzzle design is therefore not quite as good as hand-crafted puzzles would be, but on the plus side you get an inexhaustible supply of puzzles tailored to your own specification.

[3] <http://www.nikoli.co.jp/en/puzzles/shikaku.html> (beware of Flash)

[4] <https://web.archive.org/web/20041024001459/http://www.puzzle.gr.jp/puz>

### 8.1 Rectangles controls

This game is played with the mouse or cursor keys.

Left-click any edge to toggle it on or off, or left-click and drag to draw an entire rectangle (or line) on the grid in one go (removing any existing edges within that rectangle). Right-clicking and dragging will allow you to erase the contents of a rectangle without affecting its edges.

Alternatively, use the cursor keys to move the position indicator around the board. Pressing the return key then allows you to use the cursor keys to drag a rectangle out from that position, and pressing the return key again completes the rectangle. Using the space bar instead of the return key allows you to erase the contents of a rectangle without affecting its edges, as above. Pressing escape cancels a drag.

When a rectangle of the correct size is completed, it will be shaded.

(All the actions described in section 2.1 are also available.)

### 8.2 Rectangles parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

*Width, Height*

Size of grid, in squares.

*Expansion factor*

This is a mechanism for changing the type of grids generated by the program. Some people prefer a grid containing a few large rectangles to one containing many small ones. So you

can ask Rectangles to essentially generate a *smaller* grid than the size you specified, and then to expand it by adding rows and columns.

The default expansion factor of zero means that Rectangles will simply generate a grid of the size you ask for, and do nothing further. If you set an expansion factor of (say) 0.5, it means that each dimension of the grid will be expanded to half again as big after generation. In other words, the initial grid will be  $\frac{2}{3}$  the size in each dimension, and will be expanded to its full size without adding any more rectangles.

Setting an expansion factor of around 0.5 tends to make the game more difficult, and also (in my experience) rewards a less deductive and more intuitive playing style. If you set it *too* high, though, the game simply cannot generate more than a few rectangles to cover the entire grid, and the game becomes trivial.

#### *Ensure unique solution*

Normally, Rectangles will make sure that the puzzles it presents have only one solution. Puzzles with ambiguous sections can be more difficult and more subtle, so if you like you can turn off this feature and risk having ambiguous puzzles. Also, finding *all* the possible solutions can be an additional challenge for an advanced player. Turning off this option can also speed up puzzle generation.

## Chapter 9: Netslide

---

This game combines the grid generation of Net (see chapter 3) with the movement of Sixteen (see chapter 6): you have a Net grid, but instead of rotating tiles back into place you have to slide them into place by moving a whole row at a time.

As in Sixteen, control is with the mouse or cursor keys. See section 6.1.

The available game parameters have similar meanings to those in Net (see section 3.2) and Sixteen (see section 6.2).

Netslide was contributed to this collection by Richard Boulton.

## Chapter 10: Pattern

---

You have a grid of squares, which must all be filled in either black or white. Beside each row of the grid are listed the lengths of the runs of black squares on that row; above each column are listed the lengths of the runs of black squares in that column. Your aim is to fill in the entire grid black or white.

I first saw this puzzle form around 1995, under the name ‘nonograms’. I’ve seen it in various places since then, under different names.

Normally, puzzles of this type turn out to be a meaningful picture of something once you’ve solved them. However, since this version generates the puzzles automatically, they will just look like random groupings of squares. (One user has suggested that this is actually a *good* thing, since it prevents you from guessing the colour of squares based on the picture, and forces you to use logic instead.) The advantage, though, is that you never run out of them.

### 10.1 Pattern controls

This game is played with the mouse.

Left-click in a square to colour it black. Right-click to colour it white. If you make a mistake, you can middle-click, or hold down Shift while clicking with any button, to colour the square in the default grey (meaning ‘undecided’) again.

You can click and drag with the left or right mouse button to colour a vertical or horizontal line of squares black or white at a time (respectively). If you click and drag with the middle button, or with Shift held down, you can colour a whole rectangle of squares grey.

You can also move around the grid with the cursor keys. Pressing the return key will cycle the current cell through empty, then black, then white, then empty, and the space bar does the same cycle in reverse.

Moving the cursor while holding Control will colour the moved-over squares black. Holding Shift will colour the moved-over squares white, and holding both will colour them grey.

(All the actions described in section 2.1 are also available.)

### 10.2 Pattern parameters

The only options available from the ‘Custom...’ option on the ‘Type’ menu are *Width* and *Height*, which are self-explanatory.



## Chapter 11: Solo

---

You have a square grid, which is divided into as many equally sized sub-blocks as the grid has rows. Each square must be filled in with a digit from 1 to the size of the grid, in such a way that

- every row contains only one occurrence of each digit
- every column contains only one occurrence of each digit
- every block contains only one occurrence of each digit.
- (optionally, by default off) each of the square's two main diagonals contains only one occurrence of each digit.

You are given some of the numbers as clues; your aim is to place the rest of the numbers correctly.

Under the default settings, the sub-blocks are square or rectangular. The default puzzle size is  $3 \times 3$  (a  $9 \times 9$  actual grid, divided into nine  $3 \times 3$  blocks). You can also select sizes with rectangular blocks instead of square ones, such as  $2 \times 3$  (a  $6 \times 6$  grid divided into six  $3 \times 2$  blocks). Alternatively, you can select 'jigsaw' mode, in which the sub-blocks are arbitrary shapes which differ between individual puzzles.

Another available mode is 'killer'. In this mode, clues are not given in the form of filled-in squares; instead, the grid is divided into 'cages' by coloured lines, and for each cage the game tells you what the sum of all the digits in that cage should be. Also, no digit may appear more than once within a cage, even if the cage crosses the boundaries of existing regions.

If you select a puzzle size which requires more than 9 digits, the additional digits will be letters of the alphabet. For example, if you select  $3 \times 4$  then the digits which go in your grid will be 1 to 9, plus 'a', 'b' and 'c'. This cannot be selected for killer puzzles.

I first saw this puzzle in Nikoli [5], although it's also been popularised by various newspapers under the name 'Sudoku' or 'Su Doku'. Howard Garns is considered the inventor of the modern form of the puzzle, and it was first published in *Dell Pencil Puzzles and Word Games*. A more elaborate treatment of the history of the puzzle can be found on Wikipedia [6].

[5] <http://www.nikoli.co.jp/en/puzzles/sudoku.html> (beware of Flash)

[6] <http://en.wikipedia.org/wiki/Sudoku>

### 11.1 Solo controls

To play Solo, simply click the mouse in any empty square and then type a digit or letter on the keyboard to fill that square. If you make a mistake, click the mouse in the incorrect square and press Space to clear it again (or use the Undo feature).

If you *right-click* in a square and then type a number, that number will be entered in the square

as a ‘pencil mark’. You can have pencil marks for multiple numbers in the same square. Squares containing filled-in numbers cannot also contain pencil marks.

The game pays no attention to pencil marks, so exactly what you use them for is up to you: you can use them as reminders that a particular square needs to be re-examined once you know more about a particular number, or you can use them as lists of the possible numbers in a given square, or anything else you feel like.

To erase a single pencil mark, right-click in the square and type the same number again.

All pencil marks in a square are erased when you left-click and type a number, or when you left-click and press space. Right-clicking and pressing space will also erase pencil marks.

Alternatively, use the cursor keys to move the mark around the grid. Pressing the return key toggles the mark (from a normal mark to a pencil mark), and typing a number in is entered in the square in the appropriate way; typing in a 0 or using the space bar will clear a filled square.

(All the actions described in section 2.1 are also available.)

## 11.2 Solo parameters

Solo allows you to configure two separate dimensions of the puzzle grid on the ‘Type’ menu: the number of columns, and the number of rows, into which the main grid is divided. (The size of a block is the inverse of this: for example, if you select 2 columns and 3 rows, each actual block will have 3 columns and 2 rows.)

If you tick the ‘X’ checkbox, Solo will apply the optional extra constraint that the two main diagonals of the grid also contain one of every digit. (This is sometimes known as ‘Sudoku-X’ in newspapers.) In this mode, the squares on the two main diagonals will be shaded slightly so that you know it's enabled.

If you tick the ‘Jigsaw’ checkbox, Solo will generate randomly shaped sub-blocks. In this mode, the actual grid size will be taken to be the product of the numbers entered in the ‘Columns’ and ‘Rows’ boxes. There is no reason why you have to enter a number greater than 1 in both boxes; Jigsaw mode has no constraint on the grid size, and it can even be a prime number if you feel like it.

If you tick the ‘Killer’ checkbox, Solo will generate a set of cages, which are randomly shaped and drawn in an outline of a different colour. Each of these regions contains a smaller clue which shows the digit sum of all the squares in this region.

You can also configure the type of symmetry shown in the generated puzzles. More symmetry makes the puzzles look prettier but may also make them easier, since the symmetry constraints can force more clues than necessary to be present. Completely asymmetric puzzles have the freedom to contain as few clues as possible.

Finally, you can configure the difficulty of the generated puzzles. Difficulty levels are judged by the complexity of the techniques of deduction required to solve the puzzle: each level requires a mode of reasoning which was not necessary in the previous one. In particular, on difficulty levels ‘Trivial’ and ‘Basic’ there will be a square you can fill in with a single number at all times, whereas at ‘Intermediate’ level and beyond you will have to make partial deductions about the *set* of squares a number could be in (or the set of numbers that could be in a square). At ‘Unreasonable’ level, even this is not enough, and you will eventually have to make a guess, and then backtrack if it turns out to be wrong.

Generating difficult puzzles is itself difficult: if you select one of the higher difficulty levels, Solo may have to make many attempts at generating a puzzle before it finds one hard enough for you. Be prepared to wait, especially if you have also configured a large puzzle size.

## Chapter 12: Mines

---

You have a grid of covered squares, some of which contain mines, but you don't know which. Your job is to uncover every square which does *not* contain a mine. If you uncover a square containing a mine, you lose. If you uncover a square which does not contain a mine, you are told how many mines are contained within the eight surrounding squares.

This game needs no introduction; popularised by Windows, it is perhaps the single best known desktop puzzle game in existence.

This version of it has an unusual property. By default, it will generate its mine positions in such a way as to ensure that you never need to *guess* where a mine is: you will always be able to deduce it somehow. So you will never, as can happen in other versions, get to the last four squares and discover that there are two mines left but you have no way of knowing for sure where they are.

### 12.1 Mines controls

This game is played with the mouse.

If you left-click in a covered square, it will be uncovered.

If you right-click in a covered square, it will place a flag which indicates that the square is believed to be a mine. Left-clicking in a marked square will not uncover it, for safety. You can right-click again to remove a mark placed in error.

If you left-click in an *uncovered* square, it will 'clear around' the square. This means: if the square has exactly as many flags surrounding it as it should have mines, then all the covered squares next to it which are *not* flagged will be uncovered. So once you think you know the location of all the mines around a square, you can use this function as a shortcut to avoid having to click on each of the remaining squares one by one.

If you uncover a square which has *no* mines in the surrounding eight squares, then it is obviously safe to uncover those squares in turn, and so on if any of them also has no surrounding mines. This will be done for you automatically; so sometimes when you uncover a square, a whole new area will open up to be explored.

You can also use the cursor keys to move around the minefield. Pressing the return key in a covered square uncovers it, and in an uncovered square will clear around it (so it acts as the left button), pressing the space bar in a covered square will place a flag (similarly, it acts as the right button).

All the actions described in section 2.1 are also available.

Even Undo is available, although you might consider it cheating to use it. If you step on a mine, the program will only reveal the mine in question (unlike most other implementations, which reveal all of them). You can then Undo your fatal move and continue playing if you like. The

program will track the number of times you died (and Undo will not reduce that counter), so when you get to the end of the game you know whether or not you did it without making any errors.

(If you really want to know the full layout of the grid, which other implementations will show you after you die, you can always use the Solve menu option.)

## 12.2 Mines parameters

The options available from the ‘Custom...’ option on the ‘Type’ menu are:

### *Width, Height*

Size of grid in squares.

### *Mines*

Number of mines in the grid. You can enter this as an absolute mine count, or alternatively you can put a % sign on the end in which case the game will arrange for that proportion of the squares in the grid to be mines.

Beware of setting the mine count too high. At very high densities, the program may spend forever searching for a solvable grid.

### *Ensure solubility*

When this option is enabled (as it is by default), Mines will ensure that the entire grid can be fully deduced starting from the initial open space. If you prefer the riskier grids generated by other implementations, you can switch off this option.

## Chapter 13: Same Game

---

You have a grid of coloured squares, which you have to clear by highlighting contiguous regions of more than one coloured square; the larger the region you highlight, the more points you get (and the faster you clear the arena).

If you clear the grid you win. If you end up with nothing but single squares (i.e., there are no more clickable regions left) you lose.

Removing a region causes the rest of the grid to shuffle up: blocks that are suspended will fall down (first), and then empty columns are filled from the right.

Same Game was contributed to this collection by James Harvey.

### 13.1 Same Game controls

This game can be played with either the keyboard or the mouse.

If you left-click an unselected region, it becomes selected (possibly clearing the current selection).

If you left-click the selected region, it will be removed (and the rest of the grid shuffled immediately).

If you right-click the selected region, it will be unselected.

The cursor keys move a cursor around the grid. Pressing the Space or Enter keys while the cursor is in an unselected region selects it; pressing Space or Enter again removes it as above.

(All the actions described in section 2.1 are also available.)

### 13.2 Same Game parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*No. of colours*

Number of different colours used to fill the grid; the more colours, the fewer large regions of colour and thus the more difficult it is to successfully clear the grid.

*Scoring system*

Controls the precise mechanism used for scoring. With the default system,  $(n-2)^2$ , only regions of three squares or more will score any points at all. With the alternative  $(n-1)^2$

system, regions of two squares score a point each, and larger regions score relatively more points.

### *Ensure solubility*

If this option is ticked (the default state), generated grids will be guaranteed to have at least one solution.

If you turn it off, the game generator will not try to guarantee soluble grids; it will, however, still ensure that there are at least 2 squares of each colour on the grid at the start (since a grid with exactly one square of a given colour is *definitely* insoluble). Grids generated with this option disabled may contain more large areas of contiguous colour, leading to opportunities for higher scores; they can also take less time to generate.

## Chapter 14: Flip

---

You have a grid of squares, some light and some dark. Your aim is to light all the squares up at the same time. You can choose any square and flip its state from light to dark or dark to light, but when you do so, other squares around it change state as well.

Each square contains a small diagram showing which other squares change when you flip it.

### 14.1 Flip controls

This game can be played with either the keyboard or the mouse.

Left-click in a square to flip it and its associated squares, or use the cursor keys to choose a square and the space bar or Enter key to flip.

If you use the ‘Solve’ function on this game, it will mark some of the squares in red. If you click once in every square with a red mark, the game should be solved. (If you click in a square *without* a red mark, a red mark will appear in it to indicate that you will need to reverse that operation to reach the solution.)

(All the actions described in section 2.1 are also available.)

### 14.2 Flip parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Shape type*

This control determines the shape of the region which is flipped by clicking in any given square. The default setting, ‘Crosses’, causes every square to flip itself and its four immediate neighbours (or three or two if it's at an edge or corner). The other setting, ‘Random’, causes a random shape to be chosen for every square, so the game is different every time.



## Chapter 15: Guess

---

You have a set of coloured pegs, and have to reproduce a predetermined sequence of them (chosen by the computer) within a certain number of guesses.

Each guess gets marked with the number of correctly-coloured pegs in the correct places (in black), and also the number of correctly-coloured pegs in the wrong places (in white).

This game is also known (and marketed, by Hasbro, mainly) as a board game ‘Mastermind’, with 6 colours, 4 pegs per row, and 10 guesses. However, this version allows custom settings of number of colours (up to 10), number of pegs per row, and number of guesses.

Guess was contributed to this collection by James Harvey.

### 15.1 Guess controls

This game can be played with either the keyboard or the mouse.

With the mouse, drag a coloured peg from the tray on the left-hand side to its required position in the current guess; pegs may also be dragged from current and past guesses to copy them elsewhere. To remove a peg, drag it off its current position to somewhere invalid.

Right-clicking in the current guess adds a ‘hold’ marker; pegs that have hold markers will be automatically added to the next guess after marking.

Alternatively, with the keyboard, the up and down cursor keys can be used to select a peg colour, the left and right keys to select a peg position, and the space bar or Enter key to place a peg of the selected colour in the chosen position. ‘D’ or Backspace removes a peg, and Space adds a hold marker.

Pressing ‘h’ or ‘?’ will fill the current guess with a suggested guess. Using this is not recommended for 10 or more pegs as it is slow.

When the guess is complete, the smaller feedback pegs will be highlighted; clicking on these (or moving the peg cursor to them with the arrow keys and pressing the space bar or Enter key) will mark the current guess, copy any held pegs to the next guess, and move the ‘current guess’ marker.

If you correctly position all the pegs the solution will be displayed below; if you run out of guesses (or select ‘Solve...’) the solution will also be revealed.

(All the actions described in section 2.1 are also available.)

### 15.2 Guess parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu. The default game matches the parameters for the board game ‘Mastermind’.

### *Colours*

Number of colours the solution is chosen from; from 2 to 10 (more is harder).

### *Pegs per guess*

Number of pegs per guess (more is harder).

### *Guesses*

Number of guesses you have to find the solution in (fewer is harder).

### *Allow blanks*

Allows blank pegs to be given as part of a guess (makes it easier, because you know that those will never be counted as part of the solution). This is turned off by default.

Note that this doesn't allow blank pegs in the solution; if you really wanted that, use one extra colour.

### *Allow duplicates*

Allows the solution (and the guesses) to contain colours more than once; this increases the search space (making things harder), and is turned on by default.

## Chapter 16: Pegs

---

A number of pegs are placed in holes on a board. You can remove a peg by jumping an adjacent peg over it (horizontally or vertically) to a vacant hole on the other side. Your aim is to remove all but one of the pegs initially present.

This game, best known as ‘Peg Solitaire’, is possibly one of the oldest puzzle games still commonly known.

### 16.1 Pegs controls

To move a peg, drag it with the mouse from its current position to its final position. If the final position is exactly two holes away from the initial position, is currently unoccupied by a peg, and there is a peg in the intervening square, the move will be permitted and the intervening peg will be removed.

Vacant spaces which you can move a peg into are marked with holes. A space with no peg and no hole is not available for moving at all: it is an obstacle which you must work around.

You can also use the cursor keys to move a position indicator around the board. Pressing the return key while over a peg, followed by a cursor key, will jump the peg in that direction (if that is a legal move).

(All the actions described in section 2.1 are also available.)

### 16.2 Pegs parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in holes.

*Board type*

Controls whether you are given a board of a standard shape or a randomly generated shape. The two standard shapes currently supported are ‘Cross’ and ‘Octagon’ (also commonly known as the English and European traditional board layouts respectively). Selecting ‘Random’ will give you a different board shape every time (but always one that is known to have a solution).

## Chapter 17: Dominosa

---

A normal set of dominoes – that is, one instance of every (unordered) pair of numbers from 0 to 6 – has been arranged irregularly into a rectangle; then the number in each square has been written down and the dominoes themselves removed. Your task is to reconstruct the pattern by arranging the set of dominoes to match the provided array of numbers.

This puzzle is widely credited to O. S. Adler, and takes part of its name from those initials.

### 17.1 Dominosa controls

Left-clicking between any two adjacent numbers places a domino covering them, or removes one if it is already present. Trying to place a domino which overlaps existing dominoes will remove the ones it overlaps.

Right-clicking between two adjacent numbers draws a line between them, which you can use to remind yourself that you know those two numbers are *not* covered by a single domino. Right-clicking again removes the line.

You can also use the cursor keys to move a cursor around the grid. When the cursor is half way between two adjacent numbers, pressing the return key will place a domino covering those numbers, or pressing the space bar will lay a line between the two squares. Repeating either action removes the domino or line.

Pressing a number key will highlight all occurrences of that number. Pressing that number again will clear the highlighting. Up to two different numbers can be highlighted at any given time.

(All the actions described in section 2.1 are also available.)

### 17.2 Dominosa parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

#### *Maximum number on dominoes*

Controls the size of the puzzle, by controlling the size of the set of dominoes used to make it. Dominoes with numbers going up to  $N$  will give rise to an  $(N+2) \times (N+1)$  rectangle; so, in particular, the default value of 6 gives an  $8 \times 7$  grid.

#### *Ensure unique solution*

Normally, Dominosa will make sure that the puzzles it presents have only one solution. Puzzles with ambiguous sections can be more difficult and sometimes more subtle, so if you like you can turn off this feature. Also, finding *all* the possible solutions can be an additional challenge for an advanced player. Turning off this option can also speed up puzzle generation.

## Chapter 18: Untangle

---

You are given a number of points, some of which have lines drawn between them. You can move the points about arbitrarily; your aim is to position the points so that no line crosses another.

I originally saw this in the form of a Flash game called Planarity [7], written by John Tantalo.

[7] <http://planarity.net>

### 18.1 Untangle controls

To move a point, click on it with the left mouse button and drag it into a new position.

(All the actions described in section 2.1 are also available.)

### 18.2 Untangle parameters

There is only one parameter available from the ‘Custom...’ option on the ‘Type’ menu:

*Number of points*

Controls the size of the puzzle, by specifying the number of points in the generated graph.

## Chapter 19: Black Box

---

A number of balls are hidden in a rectangular arena. You have to deduce the positions of the balls by firing lasers positioned at the edges of the arena and observing how their beams are deflected.

Beams will travel straight from their origin until they hit the opposite side of the arena (at which point they emerge), unless affected by balls in one of the following ways:

- A beam that hits a ball head-on is absorbed and will never re-emerge. This includes beams that meet a ball on the first rank of the arena.
- A beam with a ball in its front-left square and no ball ahead of it gets deflected 90 degrees to the right.
- A beam with a ball in its front-right square and no ball ahead of it gets similarly deflected to the left.
- A beam that would re-emerge from its entry location is considered to be ‘reflected’.
- A beam which would get deflected before entering the arena by a ball to the front-left or front-right of its entry point is also considered to be ‘reflected’.

Beams that are reflected appear as a ‘R’; beams that hit balls head-on appear as ‘H’. Otherwise, a number appears at the firing point and the location where the beam emerges (this number is unique to that shot).

You can place guesses as to the location of the balls, based on the entry and exit patterns of the beams; once you have placed enough balls a button appears enabling you to have your guesses checked.

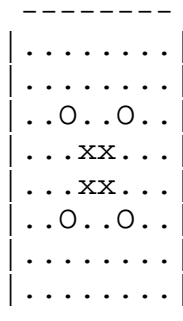
Here is a diagram showing how the positions of balls can create each of the beam behaviours shown above:

```
1RHR----
|..O.O...|
2.....3
|.....|
|.....|
3.....|
|.....O.|
H.....|
|.....O..|
12-RR---
```

As shown, it is possible for a beam to receive multiple reflections before re-emerging (see turn 3). Similarly, a beam may be reflected (possibly more than once) before receiving a hit (the ‘H’

on the left side of the example).

Note that any layout with more than 4 balls may have a non-unique solution. The following diagram illustrates this; if you know the board contains 5 balls, it is impossible to determine where the fifth ball is (possible positions marked with an x):



For this reason, when you have your guesses checked, the game will check that your solution *produces the same results* as the computer's, rather than that your solution is identical to the computer's. So in the above example, you could put the fifth ball at *any* of the locations marked with an x, and you would still win.

Black Box was contributed to this collection by James Harvey.

## 19.1 Black Box controls

To fire a laser beam, left-click in a square around the edge of the arena. The results will be displayed immediately. Clicking or holding the left button on one of these squares will highlight the current go (or a previous go) to confirm the exit point for that laser, if applicable.

To guess the location of a ball, left-click within the arena and a black circle will appear marking the guess; click again to remove the guessed ball.

Locations in the arena may be locked against modification by right-clicking; whole rows and columns may be similarly locked by right-clicking in the laser square above/below that column, or to the left/right of that row.

The cursor keys may also be used to move around the grid. Pressing the Enter key will fire a laser or add a new ball-location guess, and pressing Space will lock a cell, row, or column.

When an appropriate number of balls have been guessed, a button will appear at the top-left corner of the grid; clicking that (with mouse or cursor) will check your guesses.

If you click the 'check' button and your guesses are not correct, the game will show you the minimum information necessary to demonstrate this to you, so you can try again. If your ball positions are not consistent with the beam paths you already know about, one beam path will be circled to indicate that it proves you wrong. If your positions match all the existing beam paths but are still wrong, one new beam path will be revealed (written in red) which is not consistent with your current guesses.

If you decide to give up completely, you can select Solve to reveal the actual ball positions. At this point, correctly-placed balls will be displayed as filled black circles, incorrectly-placed balls as filled black circles with red crosses, and missing balls as filled red circles. In addition, a red circle marks any laser you had already fired which is not consistent with your ball layout

(just as when you press the ‘check’ button), and red text marks any laser you *could* have fired in order to distinguish your ball layout from the correct one.

(All the actions described in section 2.1 are also available.)

## 19.2 Black Box parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares. There are  $2 \times \text{Width} \times \text{Height}$  lasers per grid, two per row and two per column.

*No. of balls*

Number of balls to place in the grid. This can be a single number, or a range (separated with a hyphen, like ‘2-6’), and determines the number of balls to place on the grid. The ‘reveal’ button is only enabled if you have guessed an appropriate number of balls; a guess using a different number to the original solution is still acceptable, if all the beam inputs and outputs match.



## Chapter 20: Slant

---

You have a grid of squares. Your aim is to draw a diagonal line through each square, and choose which way each line slants so that the following conditions are met:

- The diagonal lines never form a loop.
- Any point with a circled number has precisely that many lines meeting at it. (Thus, a 4 is the centre of a cross shape, whereas a zero is the centre of a diamond shape – or rather, a partial diamond shape, because a zero can never appear in the middle of the grid because that would immediately cause a loop.)

Credit for this puzzle goes to Nikoli [8].

[8] [http://www.nikoli.co.jp/ja/puzzles/gokigen\\_naname](http://www.nikoli.co.jp/ja/puzzles/gokigen_naname) (in Japanese)

### 20.1 Slant controls

Left-clicking in a blank square will place a \ in it (a line leaning to the left, i.e. running from the top left of the square to the bottom right). Right-clicking in a blank square will place a / in it (leaning to the right, running from top right to bottom left).

Continuing to click either button will cycle between the three possible square contents. Thus, if you left-click repeatedly in a blank square it will change from blank to \ to / back to blank, and if you right-click repeatedly the square will change from blank to / to \ back to blank. (Therefore, you can play the game entirely with one button if you need to.)

You can also use the cursor keys to move around the grid. Pressing the return or space keys will place a \ or a /, respectively, and will then cycle them as above. You can also press / or \ to place a / or \, respectively, independent of what is already in the cursor square. Backspace removes any line from the cursor square.

(All the actions described in section 2.1 are also available.)

### 20.2 Slant parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Difficulty*

Controls the difficulty of the generated puzzle. At Hard level, you are required to do deductions based on knowledge of *relationships* between squares rather than always being able to deduce the exact contents of one square at a time. (For example, you might know

that two squares slant in the same direction, even if you don't yet know what that direction is, and this might enable you to deduce something about still other squares.) Even at Hard level, guesswork and backtracking should never be necessary.

## Chapter 21: Light Up

---

You have a grid of squares. Some are filled in black; some of the black squares are numbered. Your aim is to ‘light up’ all the empty squares by placing light bulbs in some of them.

Each light bulb illuminates the square it is on, plus all squares in line with it horizontally or vertically unless a black square is blocking the way.

To win the game, you must satisfy the following conditions:

- All non-black squares are lit.
- No light is lit by another light.
- All numbered black squares have exactly that number of lights adjacent to them (in the four squares above, below, and to the side).

Non-numbered black squares may have any number of lights adjacent to them.

Credit for this puzzle goes to Nikoli [9].

Light Up was contributed to this collection by James Harvey.

[9] <http://www.nikoli.co.jp/en/puzzles/akari.html> (beware of Flash)

### 21.1 Light Up controls

Left-clicking in a non-black square will toggle the presence of a light in that square. Right-clicking in a non-black square toggles a mark there to aid solving; it can be used to highlight squares that cannot be lit, for example.

You may not place a light in a marked square, nor place a mark in a lit square.

The game will highlight obvious errors in red. Lights lit by other lights are highlighted in this way, as are numbered squares which do not (or cannot) have the right number of lights next to them.

Thus, the grid is solved when all non-black squares have yellow highlights and there are no red lights.

(All the actions described in section 2.1 are also available.)

### 21.2 Light Up parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

### *%age of black squares*

Rough percentage of black squares in the grid.

This is a hint rather than an instruction. If the grid generator is unable to generate a puzzle to this precise specification, it will increase the proportion of black squares until it can.

### *Symmetry*

Allows you to specify the required symmetry of the black squares in the grid. (This does not affect the difficulty of the puzzles noticeably.)

### *Difficulty*

‘Easy’ means that the puzzles should be soluble without backtracking or guessing, ‘Hard’ means that some guesses will probably be necessary.

## Chapter 22: Map

---

You are given a map consisting of a number of regions. Your task is to colour each region with one of four colours, in such a way that no two regions sharing a boundary have the same colour. You are provided with some regions already coloured, sufficient to make the remainder of the solution unique.

Only regions which share a length of border are required to be different colours. Two regions which meet at only one *point* (i.e. are diagonally separated) may be the same colour.

I believe this puzzle is original; I've never seen an implementation of it anywhere else. The concept of a four-colouring puzzle was suggested by Owen Dunn; credit must also go to Nikoli and to Verity Allan for inspiring the train of thought that led to me realising Owen's suggestion was a viable puzzle. Thanks also to Gareth Taylor for many detailed suggestions.

### 22.1 Map controls

To colour a region, click the left mouse button on an existing region of the desired colour and drag that colour into the new region.

(The program will always ensure the starting puzzle has at least one region of each colour, so that this is always possible!)

If you need to clear a region, you can drag from an empty region, or from the puzzle boundary if there are no empty regions left.

Dragging a colour using the *right* mouse button will stipple the region in that colour, which you can use as a note to yourself that you think the region *might* be that colour. A region can contain stipples in multiple colours at once. (This is often useful at the harder difficulty levels.)

You can also use the cursor keys to move around the map: the colour of the cursor indicates the position of the colour you would drag (which is not obvious if you're on a region's boundary, since it depends on the direction from which you approached the boundary). Pressing the return key starts a drag of that colour, as above, which you control with the cursor keys; pressing the return key again finishes the drag. The space bar can be used similarly to create a stippled region. Double-pressing the return key (without moving the cursor) will clear the region, as a drag from an empty region does: this is useful with the cursor mode if you have filled the entire map in but need to correct the layout.

If you press L during play, the game will toggle display of a number in each region of the map. This is useful if you want to discuss a particular puzzle instance with a friend – having an unambiguous name for each region is much easier than trying to refer to them all by names such as ‘the one down and right of the brown one on the top border’.

(All the actions described in section 2.1 are also available.)

## 22.2 Map parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

### *Width, Height*

Size of grid in squares.

### *Regions*

Number of regions in the generated map.

### *Difficulty*

In ‘Easy’ mode, there should always be at least one region whose colour can be determined trivially. In ‘Normal’ and ‘Hard’ modes, you will have to use increasingly complex logic to deduce the colour of some regions. However, it will always be possible without having to guess or backtrack.

In ‘Unreasonable’ mode, the program will feel free to generate puzzles which are as hard as it can possibly make them: the only constraint is that they should still have a unique solution. Solving Unreasonable puzzles may require guessing and backtracking.

## Chapter 23: Loopy

---

You are given a grid of dots, marked with yellow lines to indicate which dots you are allowed to connect directly together. Your aim is to use some subset of those yellow lines to draw a single unbroken loop from dot to dot within the grid.

Some of the spaces between the lines contain numbers. These numbers indicate how many of the lines around that space form part of the loop. The loop you draw must correctly satisfy all of these clues to be considered a correct solution.

In the default mode, the dots are arranged in a grid of squares; however, you can also play on triangular or hexagonal grids, or even more exotic ones.

Credit for the basic puzzle idea goes to Nikoli [10].

Loopy was originally contributed to this collection by Mike Pinna, and subsequently enhanced to handle various types of non-square grid by Lambros Lambrou.

[10] <http://www.nikoli.co.jp/en/puzzles/slitherlink.html> (beware of Flash)

### 23.1 Loopy controls

Click the left mouse button on a yellow line to turn it black, indicating that you think it is part of the loop. Click again to turn the line yellow again (meaning you aren't sure yet).

If you are sure that a particular line segment is *not* part of the loop, you can click the right mouse button to remove it completely. Again, clicking a second time will turn the line back to yellow.

(All the actions described in section 2.1 are also available.)

### 23.2 Loopy parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

*Width, Height*

Size of grid, measured in number of regions across and down. For square grids, it's clear how this is counted; for other types of grid you may have to think a bit to see how the dimensions are measured.

*Grid type*

Allows you to choose between a selection of types of tiling. Some have all the faces the same but may have multiple different types of vertex (e.g. the *Cairo* or *Kites* mode); others have all the vertices the same but may have different types of face (e.g. the *Great Hexagonal*). The square, triangular and honeycomb grids are fully regular, and have all

their vertices *and* faces the same; this makes them the least confusing to play.

### *Difficulty*

Controls the difficulty of the generated puzzle.



## Chapter 24: Inertia

---

You are a small green ball sitting in a grid full of obstacles. Your aim is to collect all the gems without running into any mines.

You can move the ball in any orthogonal *or diagonal* direction. Once the ball starts moving, it will continue until something stops it. A wall directly in its path will stop it (but if it is moving diagonally, it will move through a diagonal gap between two other walls without stopping). Also, some of the squares are ‘stops’; when the ball moves on to a stop, it will stop moving no matter what direction it was going in. Gems do *not* stop the ball; it picks them up and keeps on going.

Running into a mine is fatal. Even if you picked up the last gem in the same move which then hit a mine, the game will count you as dead rather than victorious.

This game was originally implemented for Windows by Ben Olmstead [11], who was kind enough to release his source code on request so that it could be re-implemented for this collection.

[11] <http://xn13.com/>

### 24.1 Inertia controls

You can move the ball in any of the eight directions using the numeric keypad. Alternatively, if you click the left mouse button on the grid, the ball will begin a move in the general direction of where you clicked.

If you use the ‘Solve’ function on this game, the program will compute a path through the grid which collects all the remaining gems and returns to the current position. A hint arrow will appear on the ball indicating the direction in which you should move to begin on this path. If you then move in that direction, the arrow will update to indicate the next direction on the path. You can also press Space to automatically move in the direction of the hint arrow. If you move in a different direction from the one shown by the arrow, arrows will be shown only if the puzzle is still solvable.

All the actions described in section 2.1 are also available. In particular, if you do run into a mine and die, you can use the Undo function and resume playing from before the fatal move. The game will keep track of the number of times you have done this.

### 24.2 Inertia parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

## Chapter 25: Tents

---

You have a grid of squares, some of which contain trees. Your aim is to place tents in some of the remaining squares, in such a way that the following conditions are met:

- There are exactly as many tents as trees.
- The tents and trees can be matched up in such a way that each tent is directly adjacent (horizontally or vertically, but not diagonally) to its own tree. However, a tent may be adjacent to other trees as well as its own.
- No two tents are adjacent horizontally, vertically *or diagonally*.
- The number of tents in each row, and in each column, matches the numbers given round the sides of the grid.

This puzzle can be found in several places on the Internet, and was brought to my attention by e-mail. I don't know who I should credit for inventing it.

### 25.1 Tents controls

Left-clicking in a blank square will place a tent in it. Right-clicking in a blank square will colour it green, indicating that you are sure it *isn't* a tent. Clicking either button in an occupied square will clear it.

If you *drag* with the right button along a row or column, every blank square in the region you cover will be turned green, and no other squares will be affected. (This is useful for clearing the remainder of a row once you have placed all its tents.)

You can also use the cursor keys to move around the grid. Pressing the return key over an empty square will place a tent, and pressing the space bar over an empty square will colour it green; either key will clear an occupied square. Holding Shift and pressing the cursor keys will colour empty squares green. Holding Control and pressing the cursor keys will colour green both empty squares and squares with tents.

(All the actions described in section 2.1 are also available.)

### 25.2 Tents parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

*Width, Height*

Size of grid in squares.

*Difficulty*

Controls the difficulty of the generated puzzle. More difficult puzzles require more

complex deductions, but at present none of the available difficulty levels requires guesswork or backtracking.

## Chapter 26: Bridges

---

You have a set of islands distributed across the playing area. Each island contains a number. Your aim is to connect the islands together with bridges, in such a way that:

- Bridges run horizontally or vertically.
- The number of bridges terminating at any island is equal to the number written in that island.
- Two bridges may run in parallel between the same two islands, but no more than two may do so.
- No bridge crosses another bridge.
- All the islands are connected together.

There are some configurable alternative modes, which involve changing the parallel-bridge limit to something other than 2, and introducing the additional constraint that no sequence of bridges may form a loop from one island back to the same island. The rules stated above are the default ones.

Credit for this puzzle goes to Nikoli [12].

Bridges was contributed to this collection by James Harvey.

[12] <http://www.nikoli.co.jp/en/puzzles/hashiwokakero.html> (beware of Flash)

### 26.1 Bridges controls

To place a bridge between two islands, click the mouse down on one island and drag it towards the other. You do not need to drag all the way to the other island; you only need to move the mouse far enough for the intended bridge direction to be unambiguous. (So you can keep the mouse near the starting island and conveniently throw bridges out from it in many directions.)

Doing this again when a bridge is already present will add another parallel bridge. If there are already as many bridges between the two islands as permitted by the current game rules (i.e. two by default), the same dragging action will remove all of them.

If you want to remind yourself that two islands definitely *do not* have a bridge between them, you can right-drag between them in the same way to draw a ‘non-bridge’ marker.

If you think you have finished with an island (i.e. you have placed all its bridges and are confident that they are in the right places), you can mark the island as finished by left-clicking on it. This will highlight it and all the bridges connected to it, and you will be prevented from accidentally

modifying any of those bridges in future. Left-clicking again on a highlighted island will unmark it and restore your ability to modify it.

You can also use the cursor keys to move around the grid: if possible the cursor will always move orthogonally, otherwise it will move towards the nearest island to the indicated direction. Holding Control and pressing a cursor key will lay a bridge in that direction (if available); Shift and a cursor key will lay a 'non-bridge' marker. Pressing the return key followed by a cursor key will also lay a bridge in that direction.

You can mark an island as finished by pressing the space bar or by pressing the return key twice.

By pressing a number key, you can jump to the nearest island with that number. Letters 'a', ..., 'f' count as 10, ..., 15 and '0' as 16.

Violations of the puzzle rules will be marked in red:

- An island with too many bridges will be highlighted in red.
- An island with too few bridges will be highlighted in red if it is definitely an error (as opposed to merely not being finished yet): if adding enough bridges would involve having to cross another bridge or remove a non-bridge marker, or if the island has been highlighted as complete.
- A group of islands and bridges may be highlighted in red if it is a closed subset of the puzzle with no way to connect it to the rest of the islands. For example, if you directly connect two 1s together with a bridge and they are not the only two islands on the grid, they will light up red to indicate that such a group cannot be contained in any valid solution.
- If you have selected the (non-default) option to disallow loops in the solution, a group of bridges which forms a loop will be highlighted.

(All the actions described in section 2.1 are also available.)

## 26.2 Bridges parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

*Width, Height*

Size of grid in squares.

*Difficulty*

Difficulty level of puzzle.

*Allow loops*

This is set by default. If cleared, puzzles will be generated in such a way that they are always soluble without creating a loop, and solutions which do involve a loop will be disallowed.

*Max. bridges per direction*

Maximum number of bridges in any particular direction. The default is 2, but you can change it to 1, 3 or 4. In general, fewer is easier.

### *%age of island squares*

Gives a rough percentage of islands the generator will try and lay before finishing the puzzle. Certain layouts will not manage to lay enough islands; this is an upper bound.

### *Expansion factor (%age)*

The grid generator works by picking an existing island at random (after first creating an initial island somewhere). It then decides on a direction (at random), and then works out how far it could extend before creating another island. This parameter determines how likely it is to extend as far as it can, rather than choosing somewhere closer.

High expansion factors usually mean easier puzzles with fewer possible islands; low expansion factors can create lots of tightly-packed islands.

## Chapter 27: Unequal

---

You have a square grid; each square may contain a digit from 1 to the size of the grid, and some squares have clue signs between them. Your aim is to fully populate the grid with numbers such that:

- Each row contains only one occurrence of each digit
- Each column contains only one occurrence of each digit
- All the clue signs are satisfied.

There are two modes for this game, ‘Unequal’ and ‘Adjacent’.

In ‘Unequal’ mode, the clue signs are greater-than symbols indicating one square's value is greater than its neighbour's. In this mode not all clues may be visible, particularly at higher difficulty levels.

In ‘Adjacent’ mode, the clue signs are bars indicating one square's value is numerically adjacent (i.e. one higher or one lower) than its neighbour. In this mode all clues are always visible: absence of a bar thus means that a square's value is definitely not numerically adjacent to that neighbour's.

In ‘Trivial’ difficulty level (available via the ‘Custom’ game type selector), there are no greater-than signs in ‘Unequal’ mode; the puzzle is to solve the Latin square only.

At the time of writing, the ‘Unequal’ mode of this puzzle is appearing in the Guardian weekly under the name ‘Futoshiki’.

Unequal was contributed to this collection by James Harvey.

### 27.1 Unequal controls

Unequal shares much of its control system with Solo.

To play Unequal, simply click the mouse in any empty square and then type a digit or letter on the keyboard to fill that square. If you make a mistake, click the mouse in the incorrect square and press Space to clear it again (or use the Undo feature).

If you *right-click* in a square and then type a number, that number will be entered in the square as a ‘pencil mark’. You can have pencil marks for multiple numbers in the same square. Squares containing filled-in numbers cannot also contain pencil marks.

The game pays no attention to pencil marks, so exactly what you use them for is up to you: you can use them as reminders that a particular square needs to be re-examined once you know more about a particular number, or you can use them as lists of the possible numbers in a given square, or anything else you feel like.

To erase a single pencil mark, right-click in the square and type the same number again.

All pencil marks in a square are erased when you left-click and type a number, or when you left-click and press space. Right-clicking and pressing space will also erase pencil marks.

As for Solo, the cursor keys can be used in conjunction with the digit keys to set numbers or pencil marks. You can also use the 'M' key to auto-fill every numeric hint, ready for removal as required, or the 'H' key to do the same but also to remove all obvious hints.

Alternatively, use the cursor keys to move the mark around the grid. Pressing the return key toggles the mark (from a normal mark to a pencil mark), and typing a number in is entered in the square in the appropriate way; typing in a 0 or using the space bar will clear a filled square.

Left-clicking a clue will mark it as done (grey it out), or unmark it if it is already marked. Holding Control or Shift and pressing an arrow key likewise marks any clue adjacent to the cursor in the given direction.

(All the actions described in section 2.1 are also available.)

## 27.2 Unequal parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

### *Mode*

Mode of the puzzle ('Unequal' or 'Adjacent')

### *Size (s\*s)*

Size of grid.

### *Difficulty*

Controls the difficulty of the generated puzzle. At Trivial level, there are no greater-than signs; the puzzle is to solve the Latin square only. At Recursive level (only available via the 'Custom' game type selector) backtracking will be required, but the solution should still be unique. The levels in between require increasingly complex reasoning to avoid having to backtrack.



## Chapter 28: Galaxies

---

You have a rectangular grid containing a number of dots. Your aim is to partition the rectangle into connected regions of squares, in such a way that every region is  $180^\circ$  rotationally symmetric, and contains exactly one dot which is located at its centre of symmetry.

To enter your solution, you draw lines along the grid edges to mark the boundaries of the regions. The puzzle is complete when the marked lines on the grid are precisely those that separate two squares belonging to different regions.

This puzzle was invented by Nikoli [13], under the name ‘Tentai Show’; its name is commonly translated into English as ‘Spiral Galaxies’.

Galaxies was contributed to this collection by James Harvey.

[13] [http://www.nikoli.co.jp/en/puzzles/astromical\\_show.html](http://www.nikoli.co.jp/en/puzzles/astromical_show.html)

### 28.1 Galaxies controls

Left-click on any grid line to draw an edge if there isn't one already, or to remove one if there is. When you create a valid region (one which is closed, contains exactly one dot, is  $180^\circ$  symmetric about that dot, and contains no extraneous edges between two of its own squares), it will be highlighted automatically; so your aim is to have the whole grid highlighted in that way.

During solving, you might know that a particular grid square belongs to a specific dot, but not be sure of where the edges go and which other squares are connected to the dot. In order to mark this so you don't forget, you can right-click on the dot and drag, which will create an arrow marker pointing at the dot. Drop that in a square of your choice and it will remind you which dot it's associated with. You can also right-click on existing arrows to pick them up and move them, or destroy them by dropping them off the edge of the grid. (Also, if you're not sure which dot an arrow is pointing at, you can pick it up and move it around to make it clearer. It will swivel constantly as you drag it, to stay pointed at its parent dot.)

You can also use the cursor keys to move around the grid squares and lines. Pressing the return key when over a grid line will draw or clear its edge, as above. Pressing the return key when over a dot will pick up an arrow, to be dropped the next time the return key is pressed; this can also be used to move existing arrows around, removing them by dropping them on a dot or another arrow.

(All the actions described in section 2.1 are also available.)

### 28.2 Galaxies parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Difficulty*

Controls the difficulty of the generated puzzle. More difficult puzzles require more complex deductions, and the 'Unreasonable' difficulty level may require backtracking.

## Chapter 29: Filling

---

You have a grid of squares, some of which contain digits, and the rest of which are empty. Your job is to fill in digits in the empty squares, in such a way that each connected region of squares all containing the same digit has an area equal to that digit.

(‘Connected region’, for the purposes of this game, does not count diagonally separated squares as adjacent.)

For example, it follows that no square can contain a zero, and that two adjacent squares can not both contain a one. No region has an area greater than 9 (because then its area would not be a single digit).

Credit for this puzzle goes to Nikoli [14].

Filling was contributed to this collection by Jonas Kölker.

[14] <http://www.nikoli.co.jp/en/puzzles/fillomino.html>

### 29.1 Filling controls

To play Filling, simply click the mouse in any empty square and then type a digit on the keyboard to fill that square. By dragging the mouse, you can select multiple squares to fill with a single keypress. If you make a mistake, click the mouse in the incorrect square and press 0, Space, Backspace or Enter to clear it again (or use the Undo feature).

You can also move around the grid with the cursor keys; typing a digit will fill the square containing the cursor with that number; typing 0 will clear it. You can also select multiple squares for numbering or clearing with the return and arrow keys, before typing a digit to fill or clear the highlighted squares (as above). The space bar adds and removes single squares to and from the selection. Backspace and escape remove all squares from the selection.

(All the actions described in section 2.1 are also available.)

### 29.2 Filling parameters

Filling allows you to configure the number of rows and columns of the grid, through the ‘Type’ menu.

## Chapter 30: Keen

---

You have a square grid; each square may contain a digit from 1 to the size of the grid. The grid is divided into blocks of varying shape and size, with arithmetic clues written in them. Your aim is to fully populate the grid with digits such that:

- Each row contains only one occurrence of each digit
- Each column contains only one occurrence of each digit
- The digits in each block can be combined to form the number stated in the clue, using the arithmetic operation given in the clue. That is:
  - An addition clue means that the sum of the digits in the block must be the given number. For example, ‘15+’ means the contents of the block adds up to fifteen.
  - A multiplication clue (e.g. ‘60×’), similarly, means that the product of the digits in the block must be the given number.
  - A subtraction clue will always be written in a block of size two, and it means that one of the digits in the block is greater than the other by the given amount. For example, ‘2−’ means that one of the digits in the block is 2 more than the other, or equivalently that one digit minus the other one is 2. The two digits could be either way round, though.
  - A division clue (e.g. ‘3÷’), similarly, is always in a block of size two and means that one digit divided by the other is equal to the given amount.

Note that a block may contain the same digit more than once (provided the identical ones are not in the same row and column). This rule is precisely the opposite of the rule in Solo’s ‘Killer’ mode (see chapter 11).

This puzzle appears in the Times under the name ‘KenKen’.

### 30.1 Keen controls

Keen shares much of its control system with Solo (and Unequal).

To play Keen, simply click the mouse in any empty square and then type a digit on the keyboard to fill that square. If you make a mistake, click the mouse in the incorrect square and press Space to clear it again (or use the Undo feature).

If you *right*-click in a square and then type a number, that number will be entered in the square as a ‘pencil mark’. You can have pencil marks for multiple numbers in the same square. Squares containing filled-in numbers cannot also contain pencil marks.

The game pays no attention to pencil marks, so exactly what you use them for is up to you:

you can use them as reminders that a particular square needs to be re-examined once you know more about a particular number, or you can use them as lists of the possible numbers in a given square, or anything else you feel like.

To erase a single pencil mark, right-click in the square and type the same number again.

All pencil marks in a square are erased when you left-click and type a number, or when you left-click and press space. Right-clicking and pressing space will also erase pencil marks.

As for Solo, the cursor keys can be used in conjunction with the digit keys to set numbers or pencil marks. Use the cursor keys to move a highlight around the grid, and type a digit to enter it in the highlighted square. Pressing return toggles the highlight into a mode in which you can enter or remove pencil marks.

Pressing M will fill in a full set of pencil marks in every square that does not have a main digit in it.

(All the actions described in section 2.1 are also available.)

## 30.2 Keen parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

### *Grid size*

Specifies the size of the grid. Lower limit is 3; upper limit is 9 (because the user interface would become more difficult with ‘digits’ bigger than 9!).

### *Difficulty*

Controls the difficulty of the generated puzzle. At Unreasonable level, some backtracking will be required, but the solution should still be unique. The remaining levels require increasingly complex reasoning to avoid having to backtrack.

### *Multiplication only*

If this is enabled, all boxes will be multiplication boxes. With this rule, the puzzle is known as ‘Inshi No Heya’.

## Chapter 31: Towers

---

You have a square grid. On each square of the grid you can build a tower, with its height ranging from 1 to the size of the grid. Around the edge of the grid are some numeric clues.

Your task is to build a tower on every square, in such a way that:

- Each row contains every possible height of tower once
- Each column contains every possible height of tower once
- Each numeric clue describes the number of towers that can be seen if you look into the square from that direction, assuming that shorter towers are hidden behind taller ones. For example, in a 5×5 grid, a clue marked '5' indicates that the five tower heights must appear in increasing order (otherwise you would not be able to see all five towers), whereas a clue marked '1' indicates that the tallest tower (the one marked 5) must come first.

In harder or larger puzzles, some towers will be specified for you as well as the clues round the edge, and some edge clues may be missing.

This puzzle appears on the web under various names, particularly 'Skyscrapers', but I don't know who first invented it.

### 31.1 Towers controls

Towers shares much of its control system with Solo, Unequal and Keen.

To play Towers, simply click the mouse in any empty square and then type a digit on the keyboard to fill that square with a tower of the given height. If you make a mistake, click the mouse in the incorrect square and press Space to clear it again (or use the Undo feature).

If you *right*-click in a square and then type a number, that number will be entered in the square as a 'pencil mark'. You can have pencil marks for multiple numbers in the same square. A square containing a tower cannot also contain pencil marks.

The game pays no attention to pencil marks, so exactly what you use them for is up to you: you can use them as reminders that a particular square needs to be re-examined once you know more about a particular number, or you can use them as lists of the possible numbers in a given square, or anything else you feel like.

To erase a single pencil mark, right-click in the square and type the same number again.

All pencil marks in a square are erased when you left-click and type a number, or when you left-click and press space. Right-clicking and pressing space will also erase pencil marks.

As for Solo, the cursor keys can be used in conjunction with the digit keys to set numbers or pencil marks. Use the cursor keys to move a highlight around the grid, and type a digit to enter

it in the highlighted square. Pressing return toggles the highlight into a mode in which you can enter or remove pencil marks.

Pressing M will fill in a full set of pencil marks in every square that does not have a main digit in it.

Left-clicking a clue will mark it as done (grey it out), or unmark it if it is already marked. Holding Control or Shift and pressing an arrow key likewise marks any clue in the given direction.

(All the actions described in section 2.1 are also available.)

## 31.2 Towers parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

### *Grid size*

Specifies the size of the grid. Lower limit is 3; upper limit is 9 (because the user interface would become more difficult with ‘digits’ bigger than 9!).

### *Difficulty*

Controls the difficulty of the generated puzzle. At Unreasonable level, some backtracking will be required, but the solution should still be unique. The remaining levels require increasingly complex reasoning to avoid having to backtrack.

## Chapter 32: Singles

---

You have a grid of white squares, all of which contain numbers. Your task is to colour some of the squares black (removing the number) so as to satisfy all of the following conditions:

- No number occurs more than once in any row or column.
- No black square is horizontally or vertically adjacent to any other black square.
- The remaining white squares must all form one contiguous region (connected by edges, not just touching at corners).

Credit for this puzzle goes to Nikoli [15] who call it Hitori.

Singles was contributed to this collection by James Harvey.

[15] <http://www.nikoli.com/en/puzzles/hitori.html> (beware of Flash)

### 32.1 Singles controls

Left-clicking on an empty square will colour it black; left-clicking again will restore the number. Right-clicking will add a circle (useful for indicating that a cell is definitely not black).

You can also use the cursor keys to move around the grid. Pressing the return or space keys will turn a square black or add a circle respectively, and pressing the key again will restore the number or remove the circle.

(All the actions described in section 2.1 are also available.)

### 32.2 Singles parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Difficulty*

Controls the difficulty of the generated puzzle.



## Chapter 33: Magnets

---

A rectangular grid has been filled with a mixture of magnets (that is, dominoes with one positive end and one negative end) and blank dominoes (that is, dominoes with two neutral poles). These dominoes are initially only seen in silhouette. Around the grid are placed a number of clues indicating the number of positive and negative poles contained in certain columns and rows.

Your aim is to correctly place the magnets and blank dominoes such that all the clues are satisfied, with the additional constraint that no two similar magnetic poles may be orthogonally adjacent (since they repel). Neutral poles do not repel, and can be adjacent to any other pole.

Credit for this puzzle goes to Janko [16].

Magnets was contributed to this collection by James Harvey.

[16] <http://www.janko.at/Raetsel/Magnete/index.htm>

### 33.1 Magnets controls

Left-clicking on an empty square places a magnet at that position with the positive pole on the square and the negative pole on the other half of the magnet; left-clicking again reverses the polarity, and a third click removes the magnet.

Right-clicking on an empty square places a blank domino there. Right-clicking again places two question marks on the domino, signifying ‘this cannot be blank’ (which can be useful to note deductions while solving), and right-clicking again empties the domino.

Left-clicking a clue will mark it as done (grey it out), or unmark it if it is already marked.

You can also use the cursor keys to move a cursor around the grid. Pressing the return key will lay a domino with a positive pole at that position; pressing again reverses the polarity and then removes the domino, as with left-clicking. Using the space bar allows placement of blank dominoes and cannot-be-blank hints, as for right-clicking.

(All the actions described in section 2.1 are also available.)

### 33.2 Magnets parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares. There will be half  $Width \times Height$  dominoes in the grid: if this number is odd then one square will be blank.

(Grids with at least one odd dimension tend to be easier to solve.)

### *Difficulty*

Controls the difficulty of the generated puzzle. At Tricky level, you are required to make more deductions about empty dominoes and row/column counts.

### *Strip clues*

If true, some of the clues around the grid are removed at generation time, making the puzzle more difficult.

## Chapter 34: Signpost

---

You have a grid of squares; each square (except the last one) contains an arrow, and some squares also contain numbers. Your job is to connect the squares to form a continuous list of numbers starting at 1 and linked in the direction of the arrows – so the arrow inside the square with the number 1 will point to the square containing the number 2, which will point to the square containing the number 3, etc. Each square can be any distance away from the previous one, as long as it is somewhere in the direction of the arrow.

By convention the first and last numbers are shown; one or more interim numbers may also appear at the beginning.

Credit for this puzzle goes to Janko [17], who call it ‘Pfeilpfad’ (‘arrow path’).

Signpost was contributed to this collection by James Harvey.

[17] <http://janko.at/Raetsel/Pfeilpfad/index.htm>

### 34.1 Signpost controls

To play Signpost, you connect squares together by dragging from one square to another, indicating that they are adjacent in the sequence. Drag with the left button from a square to its successor, or with the right button from a square to its predecessor.

If you connect together two squares in this way and one of them has a number in it, the appropriate number will appear in the other square. If you connect two non-numbered squares, they will be assigned temporary algebraic labels: on the first occasion, they will be labelled ‘a’ and ‘a+1’, and then ‘b’ and ‘b+1’, and so on. Connecting more squares on to the ends of such a chain will cause them all to be labelled with the same letter.

When you left-click or right-click in a square, the legal squares to connect it to will be shown.

The arrow in each square starts off black, and goes grey once you connect the square to its successor. Also, each square which needs a predecessor has a small dot in the bottom left corner, which vanishes once you link a square to it. So your aim is always to connect a square with a black arrow to a square with a dot.

To remove any links for a particular square (both incoming and outgoing), left-drag it off the grid. To remove a whole chain, right-drag any square in the chain off the grid.

You can also use the cursor keys to move around the grid squares and lines. Pressing the return key when over a square starts a link operation, and pressing the return key again over a square will finish the link, if allowable. Pressing the space bar over a square will show the other squares pointing to it, and allow you to form a backward link, and pressing the space bar again cancels this.

(All the actions described in section 2.1 are also available.)

## 34.2 Signpost parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Force start/end to corners*

If true, the start and end squares are always placed in opposite corners (the start at the top left, and the end at the bottom right). If false the start and end squares are placed randomly (although always both shown).

## Chapter 35: Range

---

You have a grid of squares; some squares contain numbers. Your job is to colour some of the squares black, such that several criteria are satisfied:

- no square with a number is coloured black.
- no two black squares are adjacent (horizontally or vertically).
- for any two white squares, there is a path between them using only white squares.
- for each square with a number, that number denotes the total number of white squares reachable from that square going in a straight line in any horizontal or vertical direction until hitting a wall or a black square; the square with the number is included in the total (once).

For instance, a square containing the number one must have four black squares as its neighbours by the last criterion; but then it's impossible for it to be connected to any outside white square, which violates the second to last criterion. So no square will contain the number one.

Credit for this puzzle goes to Nikoli, who have variously called it ‘Kurodoko’, ‘Kuromasu’ or ‘Where is Black Cells’. [18].

Range was contributed to this collection by Jonas Kölker.

[18] [http://www.nikoli.co.jp/en/puzzles/where\\_is\\_black\\_cells.html](http://www.nikoli.co.jp/en/puzzles/where_is_black_cells.html)

### 35.1 Range controls

Click with the left button to paint a square black, or with the right button to mark a square with a dot to indicate that you are sure it should *not* be painted black. Repeated clicking with either button will cycle the square through the three possible states (filled, dotted or empty) in opposite directions.

You can also use the cursor keys to move around the grid squares. Pressing Return does the same as clicking with the left button, while pressing Space does the same as a right button click. Moving with the cursor keys while holding Shift will place dots in all squares that are moved through.

(All the actions described in section 2.1 are also available.)

### 35.2 Range parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

## Chapter 36: Pearl

---

You have a grid of squares. Your job is to draw lines between the centres of horizontally or vertically adjacent squares, so that the lines form a single closed loop. In the resulting grid, some of the squares that the loop passes through will contain corners, and some will be straight horizontal or vertical lines. (And some squares can be completely empty – the loop doesn't have to pass through every square.)

Some of the squares contain black and white circles, which are clues that the loop must satisfy.

A black circle in a square indicates that that square is a corner, but neither of the squares adjacent to it in the loop is also a corner.

A white circle indicates that the square is a straight edge, but *at least one* of the squares adjacent to it in the loop is a corner.

(In both cases, the clue only constrains the two squares adjacent *in the loop*, that is, the squares that the loop passes into after leaving the clue square. The squares that are only adjacent *in the grid* are not constrained.)

Credit for this puzzle goes to Nikoli, who call it 'Masyu'. [19]

Thanks to James Harvey for assistance with the implementation.

[19] <http://www.nikoli.co.jp/en/puzzles/masyu.html> (beware of Flash)

### 36.1 Pearl controls

Click with the left button on a grid edge to draw a segment of the loop through that edge, or to remove a segment once it is drawn.

Drag with the left button through a series of squares to draw more than one segment of the loop in one go. Alternatively, drag over an existing part of the loop to undraw it, or to undraw part of it and then go in a different direction.

Click with the right button on a grid edge to mark it with a cross, indicating that you are sure the loop does not go through that edge. (For instance, if you have decided which of the squares adjacent to a white clue has to be a corner, but don't yet know which way the corner turns, you might mark the one way it *can't* go with a cross.)

Alternatively, use the cursor keys to move the cursor. Use the Enter key to begin and end keyboard 'drag' operations. Use the Space, Escape or Backspace keys to cancel the drag. Or, hold Control while dragging with the cursor keys to toggle segments as you move between squares.

Pressing Control-Shift-arrowkey or Shift-arrowkey simulates a left or right click, respectively, on the edge in the direction of the key.

(All the actions described in section 2.1 are also available.)

## 36.2 Pearl parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

### *Width, Height*

Size of grid in squares.

### *Difficulty*

Controls the difficulty of the generated puzzle.

### *Allow unsoluble*

If this is set, then the game will be generated in the simplest way: every clue square that can possibly be provided will be shown, and the generator will not check whether the puzzle can be uniquely solved.

This speeds up game generation, and allows much larger grids to be played. At least one possible solution will still always exist, but there's no guarantee that it will be unique, or that it will be possible to deduce it step by step.

## Chapter 37: Undead

---

You are given a grid of squares, some of which contain diagonal mirrors. Every square which is not a mirror must be filled with one of three types of undead monster: a ghost, a vampire, or a zombie.

Vampires can be seen directly, but are invisible when reflected in mirrors. Ghosts are the opposite way round: they can be seen in mirrors, but are invisible when looked at directly. Zombies are visible by any means.

You are also told the total number of each type of monster in the grid. Also around the edge of the grid are written numbers, which indicate how many monsters can be seen if you look into the grid along a row or column starting from that position. (The diagonal mirrors are reflective on both sides. If your reflected line of sight crosses the same monster more than once, the number will count it each time it is visible, not just once.)

This puzzle type was invented by David Millar, under the name ‘Haunted Mirror Maze’. See [20] for more details.

Undead was contributed to this collection by Steffen Bauer.

[20] <http://www.janko.at/Raetsel/Spukschloss/index.htm>

### 37.1 Undead controls

Undead has a similar control system to Solo, Unequal and Keen.

To play Undead, click the mouse in any empty square and then type a letter on the keyboard indicating the type of monster: ‘G’ for a ghost, ‘V’ for a vampire, or ‘Z’ for a zombie. If you make a mistake, click the mouse in the incorrect square and press Space to clear it again (or use the Undo feature).

If you *right*-click in a square and then type a letter, the corresponding monster will be shown in reduced size in that square, as a ‘pencil mark’. You can have pencil marks for multiple monsters in the same square. A square containing a full-size monster cannot also contain pencil marks.

The game pays no attention to pencil marks, so exactly what you use them for is up to you: you can use them as reminders that a particular square needs to be re-examined once you know more about a particular monster, or you can use them as lists of the possible monster in a given square, or anything else you feel like.

To erase a single pencil mark, right-click in the square and type the same letter again.

All pencil marks in a square are erased when you left-click and type a monster letter, or when you left-click and press Space. Right-clicking and pressing space will also erase pencil marks.

As for Solo, the cursor keys can be used in conjunction with the letter keys to place monsters



or pencil marks. Use the cursor keys to move a highlight around the grid, and type a monster letter to enter it in the highlighted square. Pressing return toggles the highlight into a mode in which you can enter or remove pencil marks.

If you prefer plain letters of the alphabet to cute monster pictures, you can press ‘A’ to toggle between showing the monsters as monsters or showing them as letters.

Left-clicking a clue will mark it as done (grey it out), or unmark it if it is already marked.

(All the actions described in section 2.1 are also available.)

## **37.2 Undead parameters**

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Difficulty*

Controls the difficulty of the generated puzzle.

## Chapter 38: Unruly

---

You are given a grid of squares, which you must colour either black or white. Some squares are provided as clues; the rest are left for you to fill in. Each row and column must contain the same number of black and white squares, and no row or column may contain three consecutive squares of the same colour.

This puzzle type was invented by Adolfo Zanellati, under the name ‘Tohu wa Vohu’. See [21] for more details.

Unruly was contributed to this collection by Lennard Sprong.

[21] <http://www.janko.at/Raetsel/Tohu-Wa-Vohu/index.htm>

### 38.1 Unruly controls

To play Unruly, click the mouse in a square to change its colour. Left-clicking an empty square will turn it black, and right-clicking will turn it white. Keep clicking the same button to cycle through the three possible states for the square. If you middle-click in a square it will be reset to empty.

You can also use the cursor keys to move around the grid. Pressing the return or space keys will turn an empty square black or white respectively (and then cycle the colours in the same way as the mouse buttons), and pressing Backspace will reset a square to empty.

(All the actions described in section 2.1 are also available.)

### 38.2 Unruly parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares. (Note that the rules of the game require both the width and height to be even numbers.)

*Difficulty*

Controls the difficulty of the generated puzzle.

*Unique rows and columns*

If enabled, no two rows are permitted to have exactly the same pattern, and likewise columns. (A row and a column can match, though.)

## Chapter 39: Flood

---

You are given a grid of squares, coloured at random in multiple colours. In each move, you can flood-fill the top left square in a colour of your choice (i.e. every square reachable from the starting square by an orthogonally connected path of squares all the same colour will be filled in the new colour). As you do this, more and more of the grid becomes connected to the starting square.

Your aim is to make the whole grid the same colour, in as few moves as possible. The game will set a limit on the number of moves, based on running its own internal solver. You win if you can make the whole grid the same colour in that many moves or fewer.

I saw this game (with a fixed grid size, fixed number of colours, and fixed move limit) at <http://floodit.appspot.com> (no longer accessible).

### 39.1 Flood controls

To play Flood, click the mouse in a square. The top left corner and everything connected to it will be flood-filled with the colour of the square you clicked. Clicking a square the same colour as the top left corner has no effect, and therefore does not count as a move.

You can also use the cursor keys to move a cursor (outline black square) around the grid. Pressing the return key will fill the top left corner in the colour of the square under the cursor.

(All the actions described in section 2.1 are also available.)

### 39.2 Flood parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of the grid, in squares.

*Colours*

Number of colours used to fill the grid. Must be at least 3 (with two colours there would only be one legal move at any stage, hence no choice to make at all), and at most 10.

*Extra moves permitted*

Controls the difficulty of the puzzle, by increasing the move limit. In each new grid, Flood will run an internal solver to generate its own solution, and then the value in this field will be added to the length of Flood's solution to generate the game's move limit. So a value of 0 requires you to be just as efficient as Flood's automated solver, and a larger value makes it easier.

(Note that Flood's internal solver will not necessarily find the shortest possible solution, though I believe it's pretty close. For a real challenge, set this value to 0 and then try to solve a grid in *strictly fewer* moves than the limit you're given!)

## Chapter 40: Tracks

---

You are given a grid of squares, some of which are filled with train tracks. You need to complete the track from A to B so that the rows and columns contain the same number of track segments as are indicated in the clues to the top and right of the grid.

There are only straight and 90 degree curved rails, and the track may not cross itself.

Tracks was contributed to this collection by James Harvey.

### 40.1 Tracks controls

Left-clicking on an edge between two squares adds a track segment between the two squares. Right-clicking on an edge adds a cross on the edge, indicating no track is possible there.

Left-clicking in a square adds a colour indicator showing that you know the square must contain a track, even if you don't know which edges it crosses yet. Right-clicking in a square adds a cross indicating it contains no track segment.

Left- or right-dragging between squares allows you to lay a straight line of is-track or is-not-track indicators, useful for filling in rows or columns to match the clue.

(All the actions described in section 2.1 are also available.)

### 40.2 Tracks parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

*Width, Height*

Size of the grid, in squares.

*Difficulty*

Controls the difficulty of the generated puzzle: at Tricky level, you are required to make more deductions regarding disregarding moves that would lead to impossible crossings later.

*Disallow consecutive 1 clues*

Controls whether the Tracks game generation permits two adjacent rows or columns to have a 1 clue, or permits the row or column of the track's endpoint to have a 1 clue. By default this is not permitted, to avoid long straight boring segments of track and make the games more twiddly and interesting. If you want to restore the possibility, turn this option off.

# Chapter 41: Palisade

---

You're given a grid of squares, some of which contain numbers. Your goal is to subdivide the grid into contiguous regions, all of the same (given) size, such that each square containing a number is adjacent to exactly that many edges (including those between the inside and the outside of the grid).

Credit for this puzzle goes to Nikoli, who call it 'Five Cells'. [22].

Palisade was contributed to this collection by Jonas Kölker.

[22] [http://nikoli.co.jp/en/puzzles/five\\_cells.html](http://nikoli.co.jp/en/puzzles/five_cells.html)

## 41.1 Palisade controls

Left-click to place an edge. Right-click to indicate 'no edge'. Alternatively, the arrow keys will move a keyboard cursor. Holding Control while pressing an arrow key will place an edge. Press Shift-arrowkey to switch off an edge. Repeat an action to perform its inverse.

(All the actions described in section 2.1 are also available.)

## 41.2 Palisade parameters

These parameters are available from the 'Custom...' option on the 'Type' menu.

*Width, Height*

Size of grid in squares.

*Region size*

The size of the regions into which the grid must be subdivided.

## Chapter 42: Mosaic

---

You are given a grid of squares, which you must colour either black or white.

Some squares contain clue numbers. Each clue tells you the number of black squares in the 3×3 region surrounding the clue – *including* the clue square itself.

This game is variously known in other locations as: ArtMosaico, Count and Darken, Cuenta Y Sombrea, Fill-a-Pix, Fill-In, Komsu Karala, Magipic, Majipiku, Mosaico, Mosaik, Mozaiek, Nampre Puzzle, Nurie-Puzzle, Oekaki-Pix, Voisimage.

Mosaic was contributed to this collection by Didi Kohen. Colour design by Michal Shomer. The implementation is loosely based on [github.com/mordechaim/Mosaic](https://github.com/mordechaim/Mosaic).

### 42.1 Mosaic controls

To play Mosaic, click the mouse in a square to change its colour. Left-clicking an empty square will turn it black, and right-clicking will turn it white. Keep clicking the same button to cycle through the three possible states for the square.

If you hold down the mouse button and drag, you can colour multiple cells in a single action.

You can also use the cursor keys to move around the grid. Pressing the return or space keys will turn an empty square black or white respectively (and then cycle the colours in the same way as the mouse buttons), and pressing Backspace will reset a square to empty.

### 42.2 Mosaic parameters

These parameters are available from the ‘Custom...’ option on the ‘Type’ menu.

*Width, Height*

Size of grid in squares.

*Aggressive generation*

With this option set, the game generator will try harder to eliminate unnecessary clues on the board. This slows down generation, so it's not recommended for boards larger than, say, 30×30.

## Appendix A: Licence

---

This software is copyright 2004-2021 Simon Tatham.

Portions copyright Richard Boulton, James Harvey, Mike Pinna, Jonas Kölker, Dariusz Olszewski, Michael Schierl, Lambros Lambrou, Bernd Schmidt, Steffen Bauer, Lennard Sprong, Rogier Goossens, Michael Quevillon, Asher Gordon and Didi Kohen.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the ‘Software’), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED ‘AS IS’, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



# Index

---

Black Box	38	controls, for Towers	62
Bridges	52	controls, for Tracks	77
bugs	7	controls, for Twiddle	19
command line	9, 10, 11	controls, for Undead	72
common features	8	controls, for Unequal	55
controls	8	controls, for Unruly	74
controls, for Black Box	39	controls, for Untangle	37
controls, for Bridges	52	copy	8
controls, for Cube	16	copyright	80
controls, for Dominosa	36	Cube	16
controls, for Fifteen	17	‘Custom’, menu option	10
controls, for Filling	59	default parameters, specifying	11
controls, for Flip	32	Dominosa	36
controls, for Flood	75	Edit menu	8
controls, for Galaxies	57	exit	9
controls, for Guess	33	feedback	7
controls, for Inertia	49	Fifteen	17
controls, for Keen	60	File menu	8
controls, for Light Up	43	Filling	59
controls, for Loopy	47	Flip	32
controls, for Magnets	65	Flood	75
controls, for Map	45	format, ID	10
controls, for Mines	28	four-colouring	45
controls, for Mosaic	79	FreeNet	14
controls, for Net	14	Futoshiki	55
controls, for Netslide	23	Galaxies	57
controls, for Palisade	78	game ID	9
controls, for Pattern	24	game ID, format	10
controls, for Pearl	70	game ID, generating	11
controls, for Pegs	35	Game menu	8, 9
controls, for Range	69	generating game IDs	11
controls, for Rectangles	21	Guess	33
controls, for Same Game	30	Hitori	64
controls, for Signpost	67	ID format	10
controls, for Singles	64	ID, game	9
controls, for Sixteen	18	Inertia	49
controls, for Slant	41	initial state	9
controls, for Solo	25	Janko	65, 67
controls, for Tents	50	Keen	60

KenKen	60	parameters, for Loopy	47
keys	8	parameters, for Magnets	65
keys, for Black Box	39	parameters, for Map	46
keys, for Cube	16	parameters, for Mines	29
keys, for Fifteen	17	parameters, for Mosaic	79
keys, for Flip	32	parameters, for Net	15
keys, for Guess	33	parameters, for Netslide	23
keys, for Inertia	49	parameters, for Palisade	78
keys, for Net	14	parameters, for Pattern	24
keys, for Same Game	30	parameters, for Pearl	71
Latin square	55	parameters, for Pegs	35
licence	80	parameters, for Range	69
licence, MIT	7, 80	parameters, for Rectangles	21
Light Up	43	parameters, for Same Game	30
Linux	7, 11	parameters, for Signpost	68
load	8, 11	parameters, for Singles	64
Loopy	47	parameters, for Sixteen	18
Mac OS X	7, 8, 9, 10	parameters, for Slant	41
Magnets	65	parameters, for Solo	26
Map	45	parameters, for Tents	50
Mastermind	33	parameters, for Towers	63
Mines	28	parameters, for Tracks	77
MIT licence	7, 80	parameters, for Twiddle	19
Mosaic	79	parameters, for Undead	73
Net	14	parameters, for Unequal	56
NETGAME . EXE	14	parameters, for Unruly	74
Netslide	23	parameters, for Untangle	37
NetWalk	14	patches	7
new game	8	Pattern	24
Nikoli		Pearl	70
21, 25, 41, 43, 47, 52, 57, 59, 64, 69, 70, 78		Pegs	35
nonograms	24	Planarity	37
Palisade	78	PostScript	12
parameters	10	preferences, specifying default	11
parameters, for Black Box	40	preset	10
parameters, for Bridges	53	printing, on Unix	12
parameters, for Cube	16	printing, on Windows	8
parameters, for Dominosa	36	15-puzzle	17
parameters, for Fifteen	17	Puzzle Palace	21
parameters, for Filling	59	quit	9
parameters, for flip	32	Random Seed	9
parameters, for Flood	75	Range	69
parameters, for Galaxies	57	Rectangles	21
parameters, for Guess	33	redo	8
parameters, for Inertia	49	restart game	8
parameters, for Keen	61	Same Game	30
parameters, for Light Up	43	save	8, 11

shortcuts (keyboard)	8
shortcuts (keyboard), for Black Box	39
shortcuts (keyboard), for Cube	16
shortcuts (keyboard), for Fifteen	17
shortcuts (keyboard), for Flip	32
shortcuts (keyboard), for Guess	33
shortcuts (keyboard), for Inertia	49
shortcuts (keyboard), for Net	14
shortcuts (keyboard), for Same Game	30
Signpost	67
Singles	64
Sixteen	18
Skyscrapers	62
Slant	41
Solitaire, Peg	35
Solo	25
solve	9
source code	7
‘Specific’, menu option	9
state, initial	9
Tents	50
Towers	62
Tracks	77
Twiddle	19
Type menu	10
Undead	72
undo	8
Unequal	55
Unix	7, 11
Unruly	74
Untangle	37
version	10
website	7
Windows	7, 14