
Making crosswords with Qxw

Mark Owen
qxw@quinapalus.com
<http://www.quinapalus.com/qxw.html>

August 17, 2011

Contents

I	Examples	7
1	A simple blocked grid	9
1.1	Automatic fill	11
2	A more advanced blocked grid	13
2.1	Guided fill	14
2.2	Saving and exporting	16
3	A simple barred grid	17
3.1	Symmetry	18
3.2	Adding bars	18
4	Other grid shapes	19
4.1	Cutouts	19
4.2	Circular grids	19
4.3	Hexagonal grids	21
5	A 'letters latent' puzzle	27
6	A customised answer treatment	31
7	A numerical puzzle	35
II	Reference	37
8	Dictionaries	39
8.1	Using multiple dictionaries	39

8.2	Customising the dictionaries	40
8.3	Making dictionaries using external tools	41
9	Preferences and statistics	43
9.1	Preferences	43
9.2	Statistics	44
10	Selecting cells and lights	47
10.1	Selecting cells	47
10.2	Selecting lights	48
10.3	Switching selection mode	48
11	Cell and light properties	49
11.1	Cell properties	49
11.2	Light properties	50
12	Answer treatments	51
12.1	Built-in answer treatments	51
12.1.1	Playfair cipher	51
12.1.2	Substitution cipher	51
12.1.3	Fixed Caesar/Vigenère cipher	52
12.1.4	Variable Caesar cipher (clue order)	52
12.1.5	Misprint (clue order)	52
12.1.6	Delete single occurrence of letter (clue order)	52
12.1.7	Letters latent: delete all occurrences of letter (clue order)	53
12.1.8	Insert single letter (clue order)	53
12.2	Plug-in answer treatments	53
13	Keyboard and mouse command summary	55
13.1	Keyboard commands	56
13.2	Mouse commands	57

Introduction

Qxw is a program to help you design and publish crosswords, from the simplest blocked grid to the most sophisticated thematic puzzle. It can make rectangular-, hexagonal- or circular-format grids with blocks, bars or both. It has an automatic grid-filling feature that can handle a wide range of answer treatments—you can even add your own answer treatment methods. Grids can be filled using letters, digits, or a mixture of both. Qxw produces output in a form ready for professional publication.

This guide is in two parts: the first part gives various informal examples of what you can do with Qxw and how you do it, while the second part is a more comprehensive and structured description of the facilities available.

Qxw is free software, licensed under version 2 of the GPL (GNU General Public License). Qxw runs under the Linux operating system, and is tested on version 10.04 of the Ubuntu distribution.

Part I

Examples

Chapter 1

A simple blocked grid

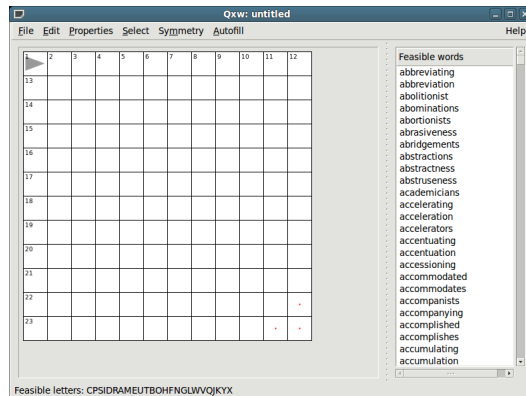


Figure: 1.1: How Qxw appears when it starts

The first thing to do when constructing a crossword with Qxw is to tell it the type and size of the grid you want. (You can change these later if necessary.) Select the menu item *Properties-Grid properties*. In the dialogue that appears, set the grid size to 7 columns by 5 rows. Leave the grid type as 'Plain rectangular'; you can fill in a title and author name if you wish. The dialogue should now appear as shown in Figure 1.2. Click on 'OK' and the grid size will change as requested.

The grey triangle in the top left-hand corner of the grid is the cursor. You can move it using the arrow keys on the keyboard or by left-clicking in the middle of a grid cell with the mouse.

You can change the direction in which the cursor points using the 'Page Up' and 'Page Down' keys or, using the mouse, by clicking on top of the cursor. You can move it forward in the current direction by pressing the spacebar and backwards by pressing 'Backspace' (sometimes labelled with a left-pointing arrow).

You can now start to add blocks to the grid. Move the cursor down one cell and to the right one cell. Now press the 'Insert' key (labelled 'Ins' on some keyboards) and a black block should appear

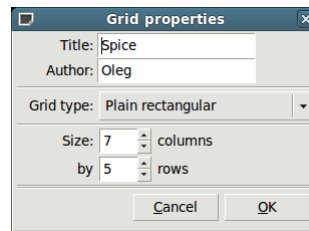


Figure 1.2: The Grid properties dialogue

under the cursor. There is a menu item *Edit-Solid block* that does the same thing as pressing 'Insert'. You can also arrange things so that clicking the mouse in the corner of a square creates or destroys a block there: see Section 9.1.

You will see that another block has also appeared in the grid, diagonally opposite the one you created. Qxw will try to maintain the symmetry of the grid as you construct it.

Continue adding blocks to the grid until it looks like Figure 1.3. If you make a mistake you can delete a block using the 'Delete' key (sometimes 'Del'); there is again a corresponding menu item *Edit-Empty*.

You can also use *Edit-Undo* ('Control-Z') to correct mistakes and *Edit-Redo* ('Control-Y') to repeat mistakes when you realise they weren't mistakes after all.

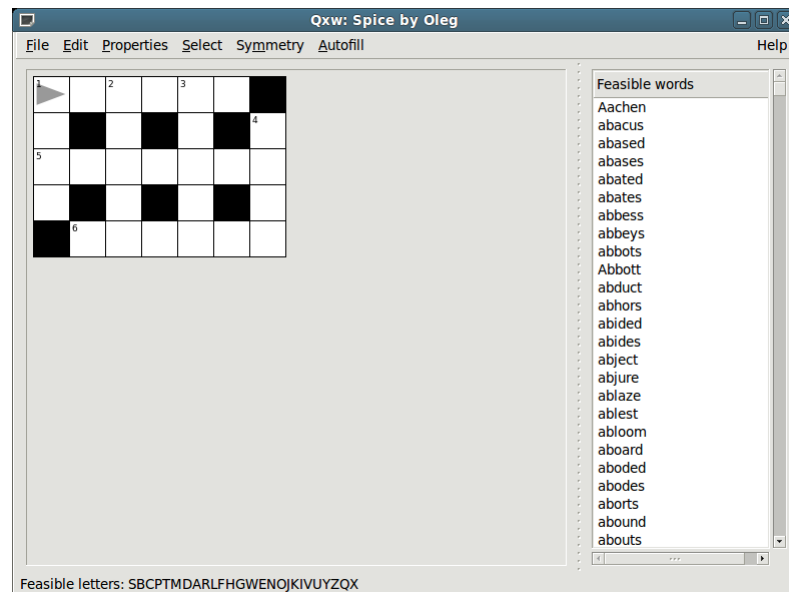


Figure 1.3: A simple blocked grid

You can now proceed to fill the grid with words. Qxw can help with this process to various degrees, from suggesting individual letters and words to fully automated filling.

1.1 Automatic fill

When Qxw starts it will look for a suitable dictionary in one of the standard places on your computer. You will need a dictionary to use Qxw's automatic filling features: see Chapter 8 for more information.

For a completely automatic fill, select the menu item *Autofill-Autofill* (or press 'Control-G'). Assuming that Qxw managed to find a suitable dictionary when it started up, it will fill the grid with words. At this point the words are only Qxw's suggestions, and so are shown in grey; select the menu item *Autofill-Accept hints* (or 'Control-A') to accept these suggestions, turning the letters black: see Figure 1.4.

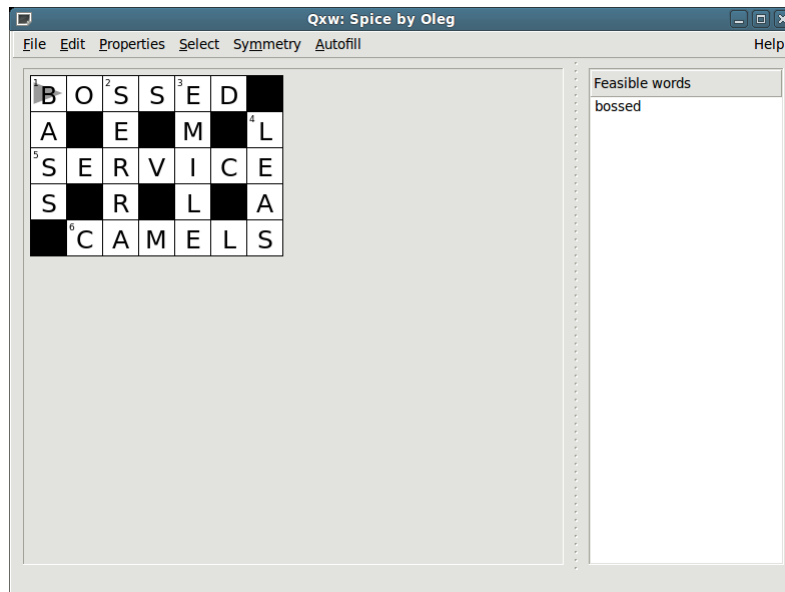


Figure: 1.4: An automatic fill of the simple blocked grid

Congratulations! You have just made your first crossword using Qxw.

Chapter 2

A more advanced blocked grid

In this chapter we will see how to construct a 15-by-15 blocked grid (a size used by many newspapers). We also have a few words that we want to include in the puzzle.

Begin, as before, by using the Grid properties dialogue to set the grid size to 15 columns by 15 rows. You may want to make the window bigger to avoid scrolling if the grid doesn't fit; also, you can move the dividing bar between the grid and the panel to the right to make more space. The display zoom factor can be adjusted using the menu item *Edit-Zoom*; as usual, there are keyboard equivalents.

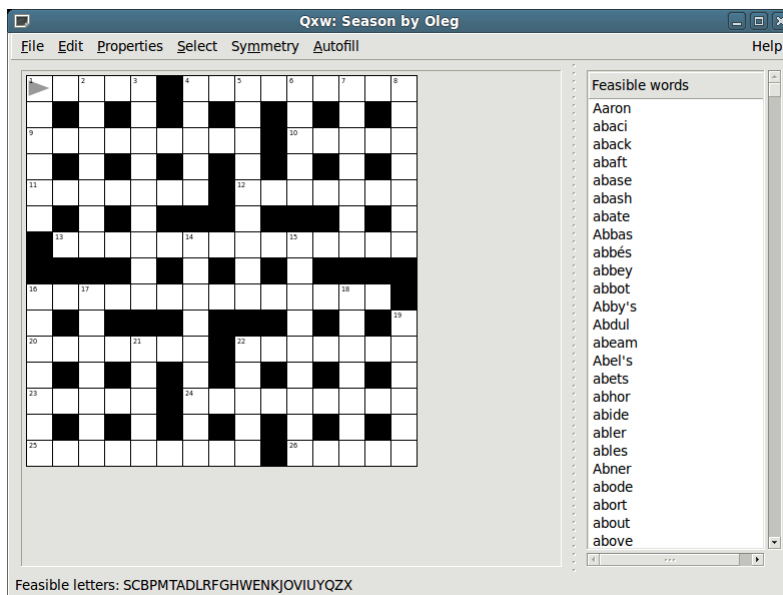


Figure: 2.1: Blocked grid

Use the cursor keys and the 'Insert' key as before to create the blocked diagram shown in Fig-

ure 2.1.

You can save your work using the *File-Save As* menu item: you need to choose a filename for it, which should normally end '.qxw', although this isn't compulsory. Use *File-Open* to load a saved crossword back at a later date.

You can fill the grid manually by simply typing letters. As you type the cursor automatically advances in the current direction. You can delete the letter under the cursor using the 'Delete' key, or you can use 'Tab', which also automatically advances the cursor. Unlike 'Delete', 'Tab' will not delete blocks.

Enter the thematic words for this crossword as shown in Figure 2.2.

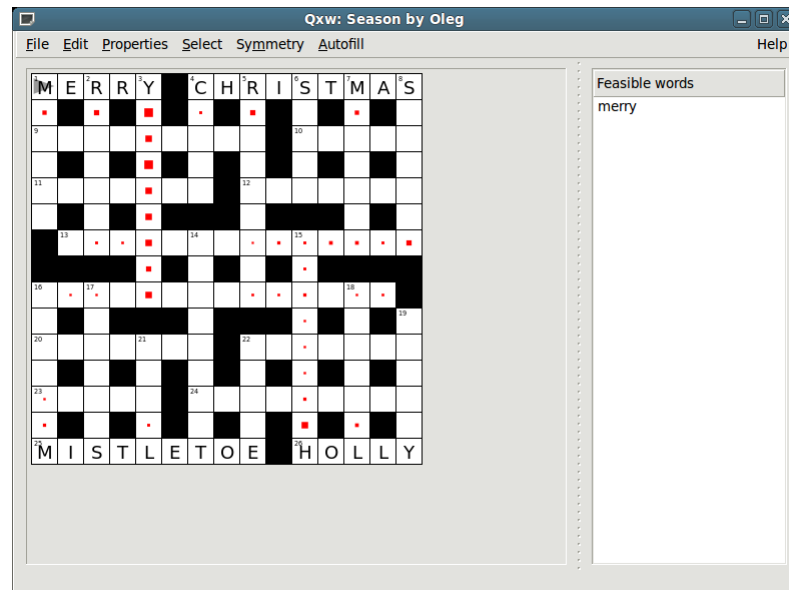


Figure 2.2: Blocked grid with theme words added

At this point we could use the automatic filling feature to complete the grid; however, we will continue this example in a more interactive way to demonstrate how Qxw can gently guide you in choosing suitable words.

2.1 Guided fill

You may have noticed as you constructed the grid that the feasible letter list at the bottom of Qxw's window and the list of words in the right-hand panel were continuously being updated. The feasible letter list shows what letters can be used to fill the cell under the cursor (the 'current cell'), in order from most promising to least promising. The right-hand panel shows the words that can be used to fill the grid entry that runs through the cursor in the direction in which it points: this grid entry is called the 'current light'.

Also, you will see small red dots start to appear in the diagram. These are ‘hotspots’, where there are relatively few feasible letters. The bigger the red dot, the fewer possibilities there are in that cell. Qxw takes into account the possible combinations of crossing words when computing which letters are feasible in each cell—it uses a small amount of ‘look-ahead’—and so can often see situations where a fill will be difficult or impossible before they become apparent to the user.

If only a single possibility remains for a cell, the forced letter will be shown in grey. To see this effect, try deleting any one of the letters of ‘MISTLETOE’: unless you are using a very odd dictionary Qxw will supply the missing letter. Using the menu item *Autofill-Accept hints* (or ‘Control-A’) you can make any letters shown in grey in the grid into a permanent part of the crossword as if you had typed them in.

When no possibilities remain to fill a cell, a grey question mark is displayed. Qxw looks ahead far enough that grey question marks will propagate over the entire grid when a fill becomes impossible.

In the grid shown in Figure 2.2 the red dots tell us that the third down light (the nine-letter word starting with ‘Y’) is likely to be the most constrained. Move the cursor over to that light and press ‘Page Up’ or ‘Page Down’ until it points in the Down direction. On the right you will see a list of feasible words: see Figure 2.3.

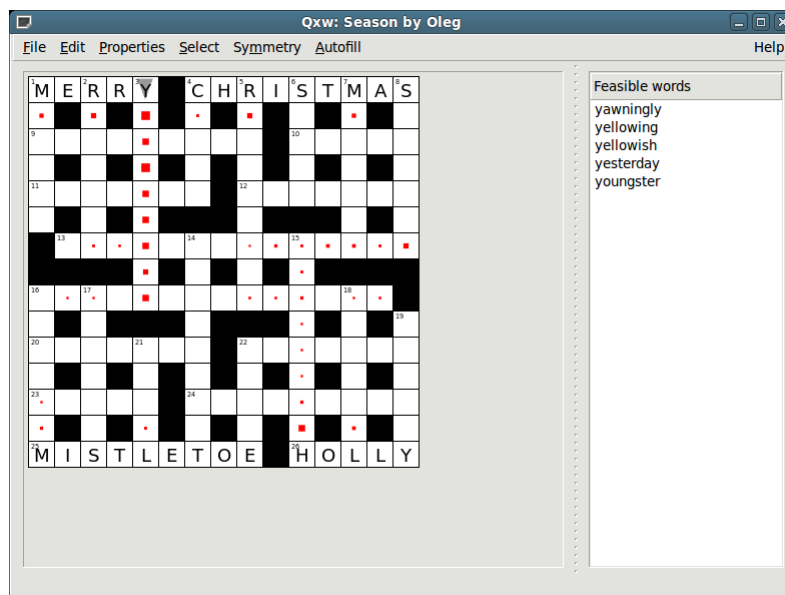


Figure: 2.3: Using the feasible word list

You can choose one of these words by clicking on it. The word will be entered as the current light, and the rest of the grid will update to show you where the new hotspots are. If you are unlucky, the remainder of the grid will turn to question marks. This means that there is no possible fill; delete the entry and try again. At this point you may find the *Edit-Undo* (‘Control-Z’) and *Edit-Redo* (‘Control-Y’) commands useful to save a lot of tedious deleting and re-entering of lights.

In this way you can complete the process of filling the grid manually, with just a little assistance

from Qxw. You don't need to enter whole words at a time if you don't want to: you can always type individual letters wherever you like in the grid. You also always have the option to use the automatic filling feature to complete the grid at any point.

2.2 Saving and exporting

You should of course save your work regularly. (Qxw does not make automatic backups.) Saving a crossword in Qxw's native format saves the grid contents and size, the title and author, and the names of the dictionary files. It does not make a copy of the dictionaries themselves.

When you have completed your grid you will want to save it in a form suitable for publication in print or on the Internet: this is called 'exporting'. Qxw can export your puzzle in various ways in a range of file formats. The various export options are listed under the *File* menu.

You can export the **blank grid image** (i.e., without answers) or the **filled grid image** (i.e., with answers) in EPS, HTML or PNG formats. EPS ('Encapsulated PostScript') is a format for drawings widely used in professional publishing; the HTML format makes your crossword into a web page using CSS ('cascading style sheets') to render the grid; and PNG is a bit-mapped graphics format also suitable for use on the Internet. The EPS and HTML formats can be scaled up after export without loss of image quality; PNG format images look blocky when scaled up. Qxw cannot export non-rectangular grids in HTML format.

You can also export just the **text of the answers** either in simple HTML or in plain text: this output can be used as a skeleton for writing clues using your favourite word processor or HTML editor.

Finally the **puzzle as a whole** or the **solution** can be output either as pure HTML (rectangular grids only) or as a combination of HTML with a PNG image to represent the grid. You can use an HTML editor to add clues, solution notes, or any other text.

Warning: Qxw cannot recover a crossword from its exported form. You must save your work using the *File-Save As* menu option.

Chapter 3

A simple barred grid

Now we will look at how to create a barred grid in Qxw. Barred grids are popular for advanced crosswords that use more obscure vocabulary; this makes the choice of dictionary more critical, and it is a good idea to read Chapter 8 before embarking on the construction of such a crossword.

Qxw does not particularly distinguish between barred and blocked grids: you can even mix bars and blocks within a grid if you like. So we start in the same way as before, using the *Properties-Grid properties* menu item to set the overall size. For this example we will use a 12-by-12 grid, which is popular for this type of crossword.

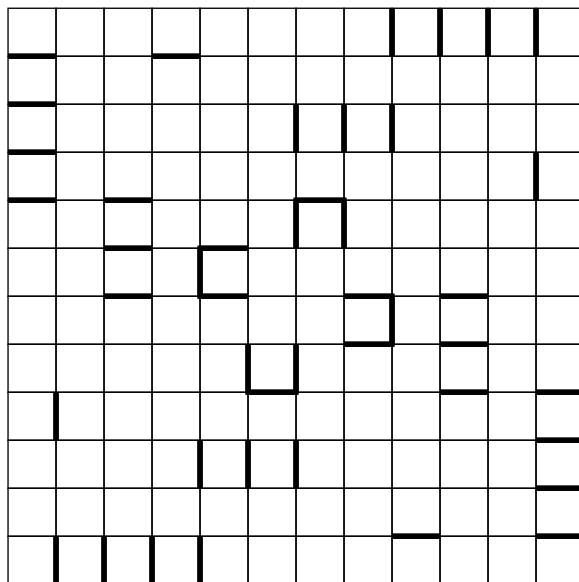


Figure 3.1: Simple barred grid

3.1 Symmetry

The previous example grids we looked at had two-fold (i.e., 180-degree) rotational symmetry automatically provided by Qxw. This is the default, but you can change it: here we will insist on four-fold (i.e., 90-degree) rotational symmetry. To do this, select the menu item *Symmetry-Fourfold rotational*. As you can see from the menu, Qxw offers a wide range of other symmetry types, including mirror symmetries and ‘up-down’ and ‘left-right’ symmetries, in which the bar or block pattern in one half of the grid matches that in the other. The various types of symmetry may be combined at will.

If you change the symmetry setting in the middle of grid construction, any subsequent operations will respect the new setting, but the pattern will not otherwise change.

3.2 Adding bars

With the symmetry specified, you can proceed to add bars to the grid. Place the cursor *after* the point where you wish to insert a bar, pointing *away* from the bar position, and press ‘Return’. Alternatively, use the Edit *Edit-Bar before* menu item.

You can also arrange things so that clicking the mouse on the edge of a square creates or destroys a bar there: see Section 9.1.

Add bars to the grid until it appears as in Figure 3.1.

The grid can now be filled manually or automatically as before. The result might look like Figure 3.2.

O	B	S	E	S	S	I	V	E	E	C	F
B	R	U	N	E	L	L	E	S	C	H	I
B	O	V	I	N	E	K	S	C	L	A	N
S	T	A	I	D	E	S	T	A	A	R	I
C	H	E	S	S	P	I	A	L	I	A	S
R	E	N	T	R	E	S	E	A	R	C	H
A	R	B	I	T	R	O	N	T	A	T	E
W	H	I	G	S	S	T	S	E	D	E	R
N	O	G	M	P	L	O	T	T	E	R	S
I	O	W	A	O	I	P	E	R	M	I	T
E	D	I	T	O	R	I	A	L	I	Z	E
R	S	G	A	N	E	C	D	O	T	E	S

Figure: 3.2: Simple barred grid with example automatic fill

Chapter 4

Other grid shapes

Qxw lets you create grids in a variety of shapes beyond the conventional rectangle or square.

4.1 Cutouts

The simplest way to customise the shape of the grid is to use cutouts. You start with a conventional rectangular grid and remove unwanted cells. An unwanted cell is removed by placing the cursor on it and pressing 'control-C' (or selecting the menu item *Edit-Cutout*). The removed cell is shown shaded on the screen and is not shown in exported versions of the grid. To return a cell to the grid, press 'Delete' (to turn it into an empty cell) or 'Insert' (to turn it into a block). Figure 4.1 shows a simple example of custom grid shape created in this way.

4.2 Circular grids

Qxw can create crosswords with circular grids. First select the menu item *Properties-Grid properties*. In the dialogue that appears set the grid type to 'Circular' and the size to 20 radii and 8 annuli. The resulting grid template appears as shown in Figure 4.2.

You can move the cursor using the arrow keys: the left and right arrow keys move the cursor forwards and backwards within an annulus, while the up and down arrow keys move the cursor radially away from and towards the centre. The 'Page Up' and 'Page Down' change the direction of the cursor as before.

Although Qxw will happily let you fill them, the cells near the centre of the grid are too small to fit a letter in comfortably. There are two approaches to solving this problem, which can be used in combination.

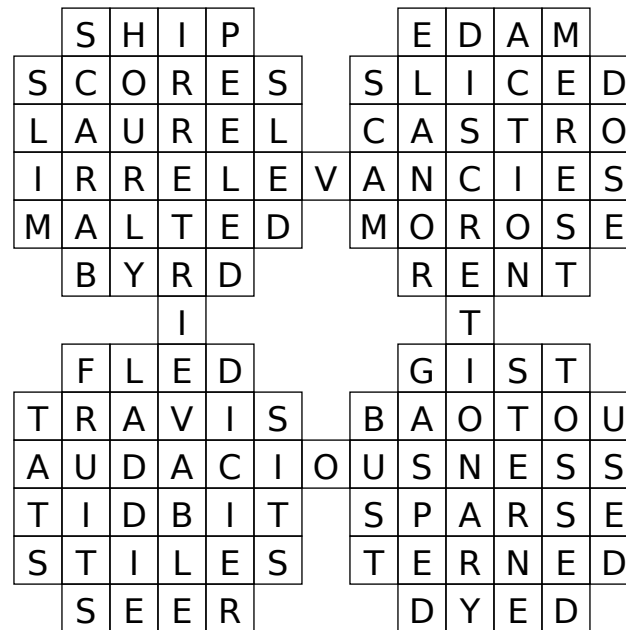


Figure 4.1: Custom grid shape created using cutouts

First, you can use cutouts to remove the centre cells from the grid as described above. (The job can be made a bit quicker by temporarily increasing the rotational symmetry setting—note that with 20 radii Qxw offers you, for example, the possibility of fivefold rotational symmetry.) The result might look like Figure 4.3.

Second, Qxw gives you the option to *merge* two or more cells into one by deleting the grid line that divides them. With the cursor pointing at the grid line to be deleted, press ‘control-M’ (or select the menu item *Edit-Merge with next*). The grid line ahead of the cursor will disappear: two cells have been merged into one. The operation respects the settings selected under the *Symmetry* menu. An example of the kind of grid you can create using merged cells is shown in Figure 4.4.

When filling the grid, a single character is assigned to each merged cell.

Pressing ‘control-M’ within a merged cell with no grid line immediately ahead of the cursor will restore that grid line, undoing (although only partially if more than two cells have been merged into one) the merge operation.

All the sub-cells comprising a merged group must lie in a line in one direction; in other words, for a circular grid, they must lie consecutively within one annulus or radius. Qxw will enforce this restriction by demerging cells as necessary.

Figure 4.5 shows an example of a more complex circular grid, with bars added to create lights within the annuli. (If there are no bars within a given annulus, Qxw treats it as if all the letters in that annulus are unchecked. To obtain the effect of a single light occupying the whole annulus, add a bar: the innermost annulus in the figure illustrates this.)

It is common in circular grids for words to be entered either forwards or backwards. To achieve

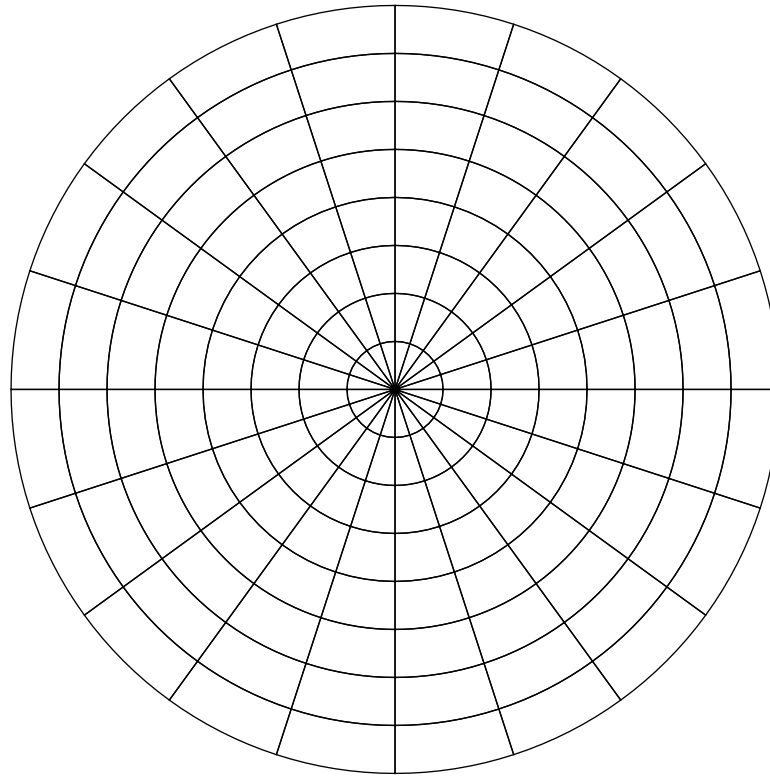


Figure 4.2: Circular grid template

this effect, select *Properties-Default light properties* and tick the box 'Allow light to be entered backwards'; then click 'OK'. Figure 4.5 was made with this setting. It is possible to allow reverse entry for only certain lights, for example only radial lights, using 'light properties': see Chapter 11.

Cell merging can be used in conjunction with Qxw's other grid types, although this is less commonly wanted. Figure 4.6 shows a simple example of a rectangular grid with merged cells.

Note that in general using merged cells increases the degree of checking between the entries in the grid, and thus can make the grid harder to fill.

4.3 Hexagonal grids

Qxw can also create crosswords based on hexagonal grids. In a hexagonal grid lights can run in three different directions. Two types of hexagonal grid are provided: one with lights running northeast, southeast and south, and the other with lights running east, southeast and southwest. As usual, you specify the grid type and size using the *Grid-Grid properties* menu item and you can cycle the cursor direction through the available options using the Page Up and Page Down keys. In other respects, constructing a hexagonal grid is just like constructing a rectangular or

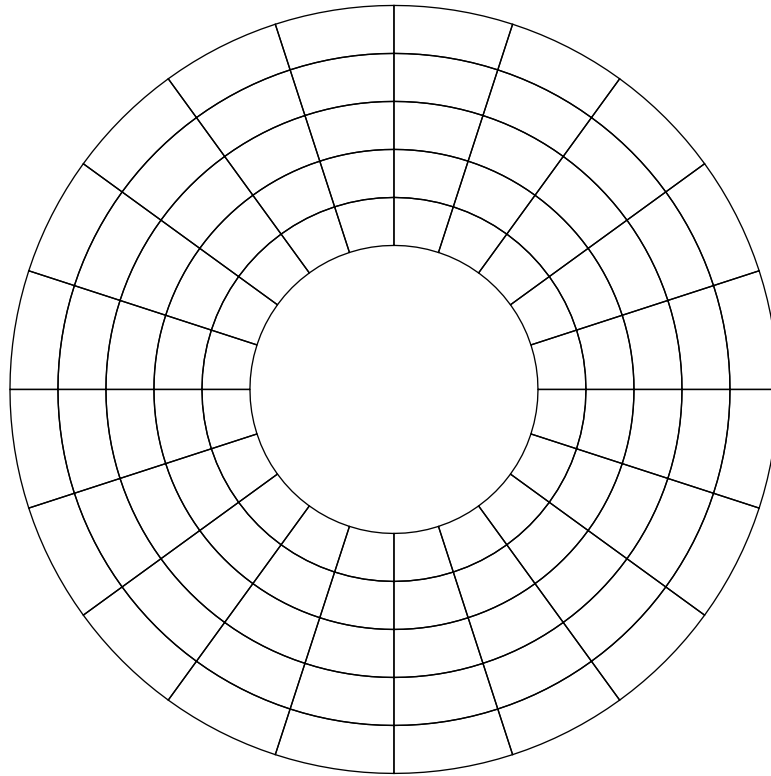


Figure: 4.3: Circular grid with central cutouts

circular grid. Figure 4.7 shows a simple example of a hexagonal grid.

Because most cells can have lights running through them in three different directions, it is possible for cells to be triply checked. This means that care is needed to ensure that overall the degree of checking is not so high that the grid cannot be filled.

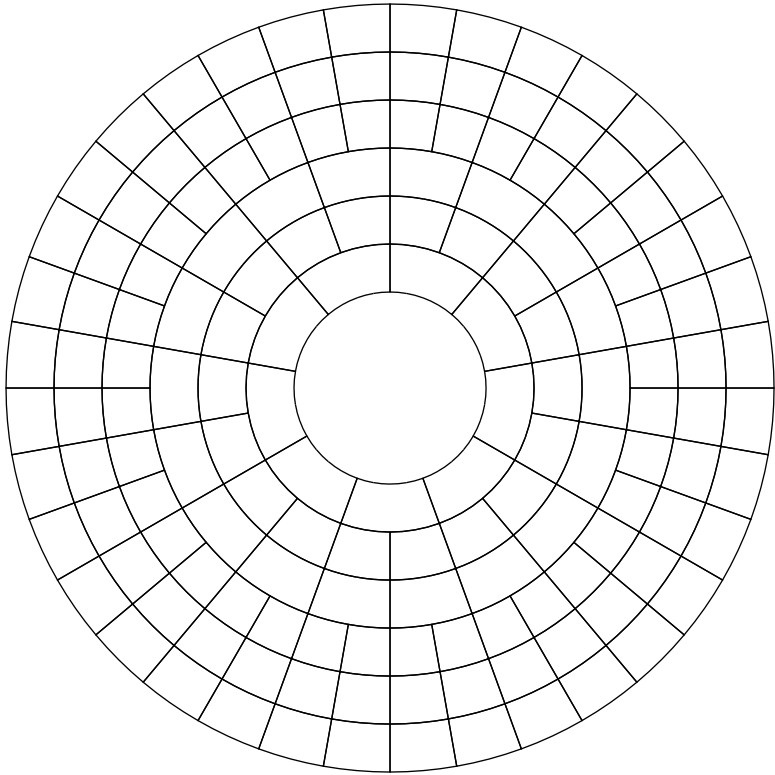


Figure: 4.4: Circular grid with cutouts and merged cells

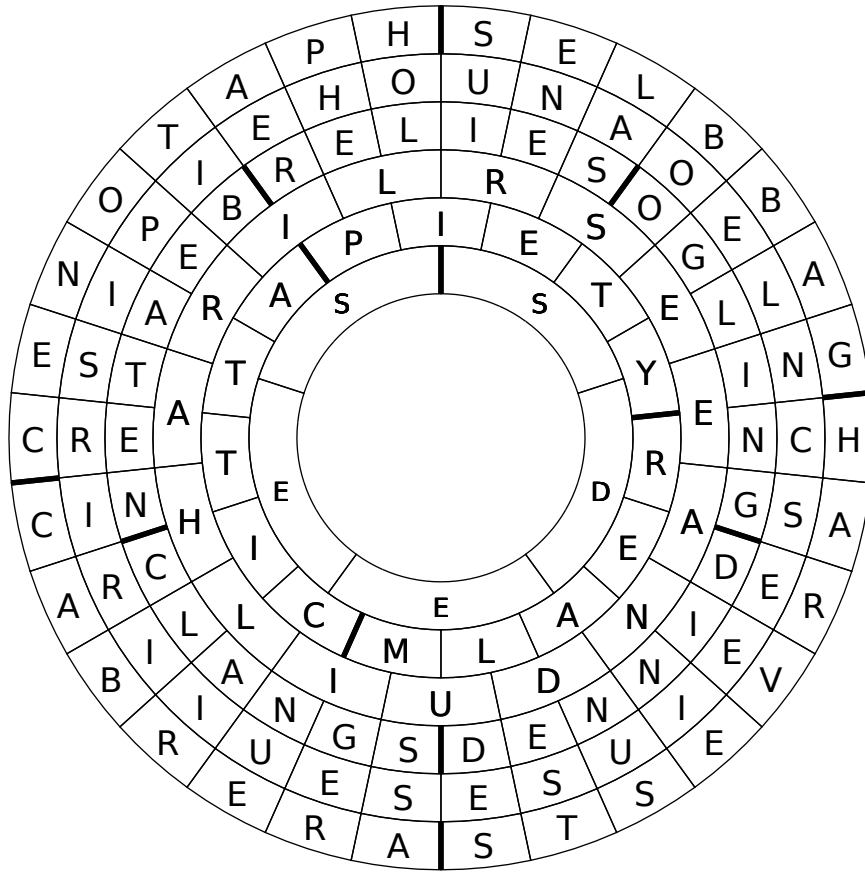


Figure 4.5: Filled circular grid with some words entered backwards

S	A	T	E	D
P	E	R	R	A
M	A	O	R	I
O	S	B	E	T
A	S	S	E	S

Figure 4.6: Filled rectangular grid with merged cells

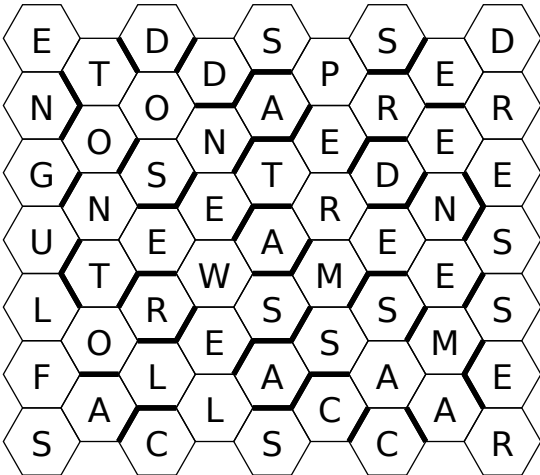


Figure 4.7: Filled hexagonal grid

Chapter 5

A 'letters latent' puzzle

In some advanced cryptic crossword puzzles, one or more of the clue answers are modified in some way before entry in the grid. Qxw calls such modification 'answer treatment'. Qxw will help you construct grids with a wide range of such treatments, including misprints, 'letters latent', various enciphering schemes, and many others. And, as we will see in the next chapter, you can even construct your own answer treatments.

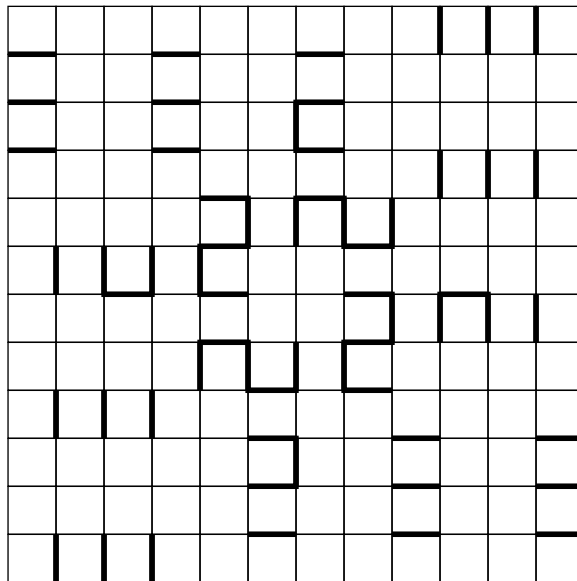


Figure: 5.1: Blank grid for 'letters latent' puzzle

In this chapter we will construct a puzzle using the 'letters latent' answer treatment. Such puzzles usually come with a preamble along the following lines:

One letter is to be omitted from the answer to each clue, wherever it occurs, before entry in the grid. In each clue the subsidiary indication leads to the grid entry.

Thus a clue answer of 'QUINQUEREME' might give rise to a grid entry of 'UINUEREME', the 'Q' being latent. It is not usually a requirement that the grid entry must itself be a word, but often the sequence of latent letters, taken in clue order, provides a helpful or thematic message to the solver.

The puzzle we will construct will apply the 'letters latent' treatment to the across answers only.

Start from a fresh 12-by-12 rectangular grid (as provided by Qxw when it starts up). Add bars to produce the diagram shown in Figure 5.1. Now tell Qxw which lights are to be treated. First select them: use the menu option `Select-Lights-in current direction` with the cursor pointing in the across direction. This will select all the across lights. Now choose the menu item `Properties-Selected light properties`, tick the boxes 'Override default light properties' and 'Enable answer treatment', and click 'OK'.

You can deselect the lights now if you wish, either by pressing 'shift-N' or using the menu item `Select-Nothing`.

For more information on how to select lights see Chapter 10; for more about what you can do with light properties, see Chapter 11.

Finally we need to specify the details of the answer treatment. Bring up the Answer treatment dialogue by choosing the menu item `Autofill-Answer treatment`. At the top of the dialogue you can choose the desired method of answer treatment: select 'Letters latent: delete all occurrences of letter (clue order)'. In the box marked 'Letters to delete' enter the message 'better a witty fool'. The result should look like Figure 5.2.

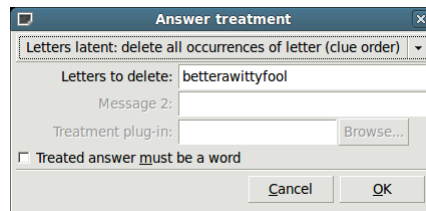


Figure 5.2: Answer treatment dialogue

You can now proceed to fill the grid as normal. Figure 5.3 shows an example automatic fill (the first and last across lights were chosen manually).

A	L	I	N	G	R	O	O	K	S	T	U
M	I	D	D	L	A	M	R	I	C	A	N
F	L	O	R	U	I	C	A	N	A	R	S
B	L	I	T	T	L	I	N	G	T	A	T
I	I	S	O	Y	H	S	G	S	H	M	E
O	M	T	C	B	E	A	C	H	E	A	R
M	A	G	C	W	A	N	D	I	T	S	I
E	R	R	A	M	D	G	A	P	E	A	L
T	L	E	T	E	T	R	A	S	T	L	E
R	E	E	I	N	G	A	L	C	R	A	N
I	N	C	N	S	I	D	E	R	A	T	E
C	E	E	A	H	A	O	W	S	D	A	Y

Figure: 5.3: 'Letters latent' automatic fill

Chapter 6

A customised answer treatment

Qxw's range of answer treatments includes most of those commonly found in advanced puzzles, but you are not limited to the built-in selection. In this chapter we will create a puzzle where answers are 'beheaded'—in other words, where an answer has its first letter removed to make the light.

Expressing a new answer treatment method to Qxw involves writing a program, called a 'plug-in'. The job of the program is to generate the possible lights that can arise from a candidate answer. To create the plug-in you will need to have a little experience with using the command line and you will need to make sure that you have the gcc C compiler installed. But don't worry if you have not programmed in C before: for most answer treatments the program will be very short indeed and easy to understand, and writing a plug-in makes an ideal gentle introduction to C programming.

```
#include "qxwplugin.h"

int treat(const char*answer) {
    strcpy(light,answer+1);
    return treatedanswer(light);
}
```

Figure 6.1: Plug-in code to remove the first letter of an answer to make a light

Figure 6.1 shows a program that 'beheads' answers. It consists of a single function, called `treat`. The work is done by the `strcpy()` command, which copies the answer string with an offset of one character (hence '`answer+1`') to the light.

Using an ordinary text editor create a file `behead.c` containing the program code shown. Compile it at the command line as follows:

```
gcc behead.c -o behead.so -shared
```

This creates the compiled plug-in `behead.so`, which Qxw can use.

Now, in Qxw, recreate the simple blank grid of Figure 1.3. There are two further steps to make Qxw use your new plug-in.

First, select the menu item *Autofill-Answer treatment*, bringing up the Answer treatment dialogue. At the top, select 'Custom plug-in' (Figure 6.2). Now click 'Browse' next to 'Treatment plug-in' to locate the plug-in file `behead.so`, and then click 'OK'.

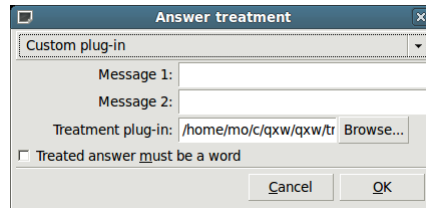


Figure 6.2: Selecting a custom plug-in in the Answer treatment dialogue

Second, as with the built-in answer treatments, you need to tell Qxw which lights are to be treated. For this example, we will have all lights subject to treatment: select *Properties-Default Light Properties* and tick the 'Enable answer treatment' box.

Now, if you select *Autofill-Autofill*, you should get a result similar to that in Figure 6.3.

A	T	E	R	E	D	
T		T		C		A
E	A	T	H	E	R	S
D		L		N		E
	R	E	A	T	H	S

Figure 6.3: Grid with all lights being 'beheaded' words

You can force all lights to be words: select *Autofill-Answer treatment* and tick the 'Treated answer must be a word' box. (This option is available for all answer treatments.) A fill might then look like Figure 6.4.

It is possible to write answer treatment plug-ins that make use of messages, just like the built-in 'Letters latent' or 'Misprint' plug-ins. Figure 6.5 shows a simple example.

This example checks that the letter that is about to be deleted matches the appropriate letter of the message (using `clueorder.index`) before allowing it as a treated answer. Figures 6.6 and 6.7 show grids filled using this plug-in.

A	C	T	O	R	S	
B		R		A		A
L	E	A	D	I	N	G
E		I		S		E
	O	T	H	E	R	S

Figure 6.4: Grid with all lights being 'beheaded' words that are themselves words

```

#include "qxwplugin.h"

int treat(const char*answer) {
    if(clueorderindex>=strlen(treatmessageAZ[0]))
        return 0;
    if(answer[0]!=treatmessageAZ[0][clueorderindex])
        return 0;
    strcpy(light,answer+1);
    return treatedanswer(light);
}

```

Figure 6.5: Plug-in code to behead answers so that deleted letters in clue order yield a message

Section 12.2 gives a full description of what is possible using answer treatment plug-ins, including a discussion of how to write a plug-in where two or more different lights can arise from a single answer.

U	T	R	A	C	E	S
A	E	E	L	E	R	S
R	O	N	T	A	G	E
A	B	E	E	R	E	X
C	E	C	R	E	A	M
H	A	N	N	A	H	A
E	R	D	S	M	A	N

Figure: 6.6: Grid with all lights being 'beheaded' words, with deleted letters in clue order spelling out 'OFF WITH HIS HEAD'

R	E	E	C	H	O	
A		B		I		P
S	A	B	E	L	L	A
E		E		L		H
	O	D	I	S	T	S

Figure: 6.7: Grid with all lights being 'beheaded' words that are themselves words, with deleted letters in clue order spelling out 'PIMENTO'

Chapter 7

A numerical puzzle

Qxw can fill grids with digits instead of, or as well as, letters. Figure 7.1 shows an automatically-filled example, constructed using two custom dictionaries and setting light properties to allow reverse entry.

6	4	1	4	1	9
3	2	0	0	0	2
1	7	0	7	4	9
7	2	4	4	4	8
9	4	8	0	0	7
1	8	9	3	9	1

Figure 7.1: Grid with each across light being a prime or the reverse of a prime, and each down light being a square or the reverse of a square

Digits can be used in a conventional word-based puzzle as proxies for special characters such as accented letters or for other thematic purposes. Figure 7.2 shows a simple example of what can be done. Again, a special-purpose dictionary was used to create this grid.

P	O	S	T	P	1	D
A	■	H	■	H	■	I
R	E	A	S	1	R	S
A	■	R	■	R	■	H
G	R	E	Y	S	T	1
1	■	B	■	I	■	S
D	O	1	S	N	U	T

Figure 7.2: Grid containing a mixture of letters and digits

Part II

Reference

Chapter 8

Dictionaries

When Qxw starts it looks for lists of words in various places on your system. Normally it will find a suitable list and load it to use as its dictionary for automatic filling. Often the first word list it finds is designed for use with a spell checker and this is not always ideal for constructing cross-words: in particular it may contain many abbreviations, proprietary names, and words ending in apostrophe-s.

You can override Qxw's default choice of dictionary, either from the command line or from the menu system. Let us suppose that you have a word list called `/usr/share/dict/ukacd18` that you want Qxw to use.

From the command line, when running Qxw type

```
./qxw -d /usr/share/dict/ukacd18
```

Or, using the menu system, go to *Autofill-Dictionaries*. This will open the Dictionaries dialogue (see Figure 8.1). Ignore everything except the top-left corner for the moment: either enter the full filename in the topmost box under 'File', or click on the topmost 'Browse' button and navigate to the word list file you want to use. Then click 'OK'. (You will get an error message at this point if, for example, the specified word list file does not exist.) Qxw will automatically construct a dictionary by removing all punctuation, accents and spaces from the words in the list.

When you now use the autofill feature you should see that the new dictionary is being used.

8.1 Using multiple dictionaries

Qxw can handle up to nine dictionaries at once. Each light in the grid can be drawn from any combination of these dictionaries (see Chapter 11). The word list files used can be specified at the command line:

```
./qxw -d <wordlist1> -d <wordlist2> -d <wordlist3> ...
```

or using the Dictionaries dialogue as above, entering one filename in the leftmost box of each row.

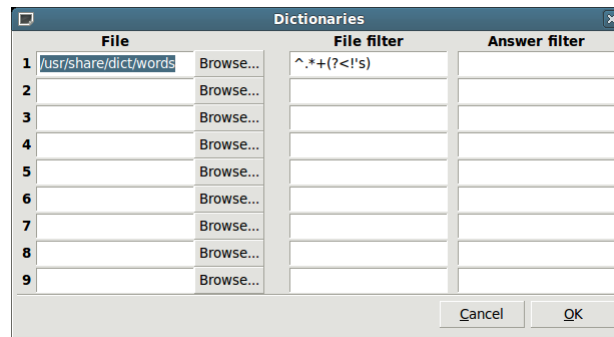


Figure 8.1: The Dictionaries dialogue

8.2 Customising the dictionaries

The contents of a dictionary can be customised using the filters provided in the Dictionaries dialogue. There are two filters associated with each dictionary. By default these filters are blank, which means that all the entries in the original word list are accepted. However, by entering a simple matching pattern, a custom dictionary can be created containing only words with a certain property. The filter syntax is that of ‘Perl-compatible regular expressions’ or PCREs. See <http://en.wikipedia.org/wiki/PCRE> for more details, and there are some useful examples at http://en.wikipedia.org/wiki/Regular_expression; The most authoritative source of information is at <http://www.pcre.org/>.

The first filter, the ‘File filter’, acts on the lines of text in the original word list file. Its primary use is to help eliminate undesirable entries from the dictionary. For example, the list found on many systems at `/usr/share/dict/words` often includes many entries ending apostrophe-s, which are not suitable for use in a crossword grid. They can be removed by specifying a ‘File filter’ that reads

```
^.*+(?!'s)
```

Qxw automatically applies such a filter when trying to load words from a default system dictionary (but not if you specify this dictionary’s name explicitly on the command line).

The second filter is called the ‘Answer filter’. This acts on the string of characters obtained from the line in the word list file after punctuation, accents and spaces have been removed. This provides an easy way to create a crossword with a simple theme. For example, to create a grid where no entry contains the letter ‘e’ set the answer filter to

```
^[^e]*$
```

Both the File filter and the Answer filter are insensitive to case.

The same source word list can be used in conjunction with different filters to make two or more different dictionaries. Using this in conjunction with ‘Light properties’ (see Chapter 11) you can construct a grid where (for example) no across answer contains the letter ‘e’ and each down answer contains the letter ‘q’.

8.3 Making dictionaries using external tools

Since Qxw's dictionaries are simple plain text files with one entry per line, you can create your own using any ordinary text editor (be sure to save the result as 'plain text'), or using any of the standard Linux command-line text processing utilities such as `grep`, `awk` and `perl`. For example, to create a dictionary `vowel` containing just those entries in `ukacd18` that start with a vowel, type:

```
grep "^[aeiouAEIOU]" <ukacd18 >vowel
```


Chapter 9

Preferences and statistics

9.1 Preferences

The Preferences dialogue (Figure 9.1) is accessed via the menu item *Edit-Preferences*. It allows you to configure a number of details of the way Qxw behaves.



Figure 9.1: The Preferences dialogue

In the **Export preferences** section you can specify the size of cells in exported grids. For square grids, this is the length of the side of the square; for hex grids, the approximate distance between two parallel sides of a cell; and for circular grids, the height of the cell in the radial direction.

Normally, clicking the mouse in the grid moves the cursor, or, if you click on top of the cursor, changes the current direction. Under **Editing preferences** you can configure Qxw so that when you click near an edge of a cell a bar is added or removed, or so that when you click near a corner a block is added or removed. You can have both of these behaviours active at once if you like.

Also under **Editing preferences** you can arrange for light numbers to be displayed in the grid as you edit.

You can configure Qxw's idea of 'underchecked' and 'overchecked' with the **Statistics preferences**.

The minimum checking ratio (before a light is deemed 'underchecked') is expressed as the percentage of checked cells in the light. The default value of 66% means that one unch is allowed in lights at least 3 cells long, two unches in lights of at least 6 cells, and so on. This is reasonable for typical barred grids, but you may wish to reduce the ratio to 50% for blocked grids.

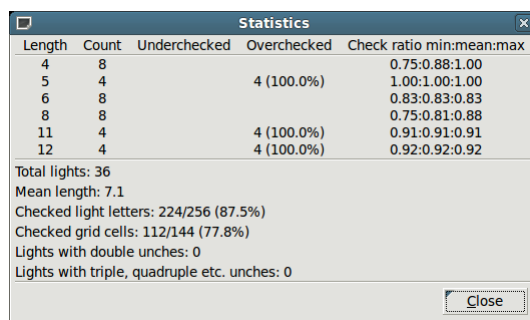
The maximum checking ratio (before a light is deemed 'overchecked') is expressed as the percentage of checked cells in the light plus one cell. This permits fully-checked entries of up to a certain maximum length. The default value of 75% means that lights of up to 4 cells may be fully checked, lights of up to 8 cells must have one unch, lights of up to 12 cells must have two unches, and so on.

Under **Autofill preferences** you can say whether the autofill function should always give the same result ('Deterministic') or potentially a different result every time it is invoked ('Slightly randomised' or 'Highly randomised').

By default the autofill function checks that no (pre-treatment) answer or light occurs twice in the grid. Untick 'Prevent duplicate answers and lights' to remove this check.

9.2 Statistics

Selecting the menu item *Edit-Statistics* brings up the Statistics dialogue: see Figure 9.2.



Length	Count	Underchecked	Overchecked	Check ratio min:mean:max
4	8			0.75:0.88:1.00
5	4		4 (100.0%)	1.00:1.00:1.00
6	8			0.83:0.83:0.83
8	8			0.75:0.81:0.88
11	4		4 (100.0%)	0.91:0.91:0.91
12	4		4 (100.0%)	0.92:0.92:0.92

Total lights: 36
Mean length: 7.1
Checked light letters: 224/256 (87.5%)
Checked grid cells: 112/144 (77.8%)
Lights with double unches: 0
Lights with triple, quadruple etc. unches: 0

Figure 9.2: The Statistics dialogue

At the top of the dialogue is a table analysing the lights in the grid by length. For each length it shows the number of lights of that length, the number of these (and percentage) that are under- or over-checked, the average checking ratio (proportion of checked letters in a light) and the minimum and maximum checking ratios.

Below the table some more general statistics appear, including the total light count, the mean light length, the number of letters checked across all lights, the number of checked grid cells, and a count of lights with double unches and triple-and-above unches.

You can adjust Qxw's idea of what 'underchecked' and 'overchecked' mean in the Preferences

dialogue: see Section 9.1.

Unlike Qxw's other dialogues, you can leave the Statistics dialogue active while you edit the grid. The information it displays is continuously updated as you work.

Chapter 10

Selecting cells and lights

Several of Qxw's functions operate on a selected set of cells or lights in the grid. For example, the menu item *Edit-Clear selected cells* ('shift-control-X') erases the letters that have been entered in the selected cells. Commands to do with selection involve the shift key.

Qxw can be in one of two selection modes: 'cell mode' and 'light mode'. The cell selection commands switch Qxw to cell mode; the light selection commands switch Qxw to light mode. Blocks and cutouts cannot be selected.

Selected cells are shown with a solid highlight; selected lights are shown highlighted by a thick line.

10.1 Selecting cells

Cells can be selected and deselected by holding down the shift key and the left mouse button while moving the mouse over the grid.

The command *Select-Current cell* ('shift-C') switches Qxw to cell selection mode (if it is not already in that mode) and adds the current cell to or removes the current cell from the cell selection.

When in cell selection mode, the command *Select-Nothing* ('shift-N') deselects all cells, leaving Qxw in cell selection mode; the command *Select-All* ('shift-A') selects all cells; and the command *Select-Invert* ('shift-I') selects all cells previously not selected, and deselects all cells previously selected.

The command *Select-Cells-overriding default properties* switches Qxw to cell selection mode (if it is not already in that mode) and selects those cells which have been set to override the default cell properties (see Chapter 11).

10.2 Selecting lights

Lights running in the direction in which the cursor is pointing can be selected and deselected by holding down the shift key and the right mouse button while moving the mouse over the grid.

The command *Select-Current light* ('shift-L') switches Qxw to light selection mode (if it is not already in that mode) and adds the current light to or removes the current light from the light selection.

When in light selection mode, the command *Select-Nothing* ('shift-N') deselects all lights, leaving Qxw in light selection mode; the command *Select-All* ('shift-A') selects all lights; and the command *Select-Invert* ('shift-I') selects all lights previously not selected, and deselects all lights previously selected.

The command *Select-Lights-in current direction* switches Qxw to light selection mode (if it is not already in that mode) and adds to or removes from the selection those lights which run in the direction the cursor is currently pointing.

The command *Select-Lights-overriding default properties* switches Qxw to light selection mode (if it is not already in that mode) and selects those lights which have been set to override the default light properties (see Chapter 11).

The command *Select-Lights-with answer treatment enabled* switches Qxw to light selection mode (if it is not already in that mode) and selects those lights whose properties have been set to enable answer treatment (see Chapter 11).

Select-Lights-with double or more unches, *Select-Lights-with triple or more unches*, *Select-Lights-that are underchecked* and *Select-Lights-that are overchecked* are commands that allow you to locate those lights flagged as described in the statistics dialogue (see Chapter 9.2).

10.3 Switching selection mode

In general when Qxw switches between cell and light selection mode as a consequence of one of the above commands the selection is reset. However, the command *Select-Cell mode <> light mode* ('shift-M') switches Qxw between selection modes without clearing the selection: if Qxw is in cell selection mode, it switches to light mode, selecting all lights incident with any selected cell; and if it is in light selection mode, it switches to cell mode, selecting all cells that form part of any selected light.

Chapter 11

Cell and light properties

Qxw lets you customise your grid and how it is filled by assigning properties to cells and lights.

In each case there is a set of default properties which can be overridden as required. For example, you would normally have the default cell properties set to give black text on a white background, but you could override this default to give white text on a red background in cells whose letters spell out a theme word.

Usually you would first set the default properties (using the command *Properties-Default cell properties* or *Properties-Default light properties*); then select the cells or lights where you wish to override the defaults using the selection commands described in Chapter 10; and finally use the *Properties-Selected cell properties* or *Properties-Selected light properties* commands to set the new properties.

11.1 Cell properties

Figure 11.1 shows the ‘Selected cell properties’ dialogue. (The ‘Default cell properties’ dialogue is the same except that it lacks the ‘Override default cell properties’ option.)

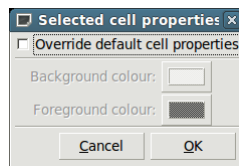


Figure: 11.1: The Selected cell properties dialogue

The dialogue allows you to change the foreground (i.e., text) and background colours used in the cell.

When the dialogue is called up, it shows the properties of the first selected cell in the grid (in normal reading order). When you click on the ‘OK’ button, the chosen properties are applied to

all selected cells.

11.2 Light properties

Figure 11.2 shows the ‘Selected light properties’ dialogue. (As before, the ‘Default light properties’ dialogue is the same except that it lacks the ‘Override default light properties’ option.)

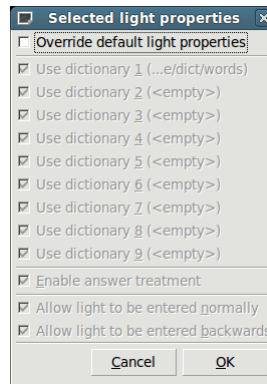


Figure: 11.2: The Selected light properties dialogue

The dialogue allows you to choose: which dictionaries are to be used for filling the light; whether answer treatment is enabled; and whether answers can be entered forwards and/or backwards.

When the dialogue is called up, it shows the properties of the first selected light in the grid (in clue order). When you click on the ‘OK’ button, the chosen properties are applied to all selected lights.

If you make changes to the grid after having set some light properties, Qxw will try to make an intelligent decision about which lights should have which properties. Although it does its best, it is possible that Qxw’s view on this will disagree with yours. The general rule is that Qxw attaches light properties to the cell containing the first letter of a light. Check (with the help of *Select-Lights-overriding default properties*) that things are as you expect.

Chapter 12

Answer treatments

12.1 Built-in answer treatments

Qxw contains eight built-in answer treatments. The following sections describe in detail how each behaves. In each case, the treatment is thought of as transforming a word (which will have been obtained from a dictionary file) into a light for entry in the grid. The answer treatments make use of up to two 'messages', which in a thematic crossword would typically be hidden quotations or further instructions to the solver.

Except where otherwise stated below, the messages you give in the answer treatment dialogue are stripped of non-alphabetic characters before being used.

12.1.1 Playfair cipher

The first message is used to make the keyword for a Playfair square. All 'J's in the keyword are replaced by 'I's. All but the first occurrence of each letter is deleted. The letters that remain, followed by the other letters of the alphabet (except 'J') in alphabetical order, are written to the Playfair square in normal reading order.

An answer of odd length is rejected. The answer is split into pairs of letters; if there are any double-letter pairs, the answer is rejected. All 'J's in the answer are replaced by 'I's. Finally, the answer is encoded using the Playfair square.

12.1.2 Substitution cipher

Each 'A' in the answer is replaced by the first letter of the first message; each 'B' by the second letter; each 'C' by the third letter; and so on. If the message is less than 26 characters long, letters without a specified replacement are left unchanged.

12.1.3 Fixed Caesar/Vigenère cipher

The first message is used as a keyword for a Vigenère cipher. If the message is empty, the answer is left unchanged.

Corresponding letters of answer and keyword are added using a system where $A = 0$, $B = 1$, $C = 2$ etc.; if a total is 26 or more, 26 is subtracted. Hence $A + A = A$, $C + F = H$, $N + N = A$ and $Y + Z = X$.

If the keyword is shorter than the answer it is repeated as necessary. In particular, if the keyword consists of one character, the result is a fixed Caesar cipher. Using the keyword 'N' results in the rot13 cipher.

12.1.4 Variable Caesar cipher (clue order)

The characters of the first message are used, one per light (with answer treatment enabled) in clue order, to provide offsets for a Caesar cipher. If the message is empty, answers are left unchanged; if the length of the message is less than the total number of lights with answer treatment enabled, the message is repeated as necessary.

A message character of 'A' leaves an answer unchanged; a message character of 'B' advances the letters of an answer by one (taking e.g. 'HAZY' to 'IBAZ'); a message character of 'C' advances the letters of an answer by two (taking 'HAZY' to 'JCBA'); and so on.

12.1.5 Misprint (clue order)

This answer treatment uses both messages in their raw form, before punctuation is removed.

Each light with answer treatment enabled, in clue order, uses one character from each message. Lights are produced from answers by changing an occurrence of the character from the first message to the character from the second message. Note that this means that a single answer word can give rise to two or more different lights, or, equally, may give rise to no lights at all.

If the character from the first message is '.' then any letter in the answer can be changed to the character from the second message to make the light. If the character from the second message is '.' then an occurrence in the answer of the character from the first message can be changed to any letter to make the light. If the characters from both messages are '.' the answer is left unchanged.

Other than when specifically instructed (i.e., when the letters from the two messages are the same), Qxw will not 'change' a letter to itself.

A message whose length is less than the total number of lights with answer treatment enabled is considered to be extended with '.' characters.

12.1.6 Delete single occurrence of letter (clue order)

Each light with answer treatment enabled, in clue order, uses one letter from the first message. For each occurrence of that letter in a candidate answer word, a light is constructed by deleting that occurrence. A single answer word can thus give rise to two or more different lights.

If the length of the message is less than the number of lights with answer treatment enabled, lights without a specified letter to be deleted are left unchanged.

12.1.7 Letters latent: delete all occurrences of letter (clue order)

Each light with answer treatment enabled, in clue order, uses one letter from the first message. If a candidate answer word does not contain that letter, it is discarded; otherwise a light is constructed by deleting every occurrence of the letter from the answer.

If the length of the message is less than the number of lights with answer treatment enabled, lights without a specified letter to be deleted are left unchanged.

12.1.8 Insert single letter (clue order)

Each light with answer treatment enabled, in clue order, uses one letter from the first message. A series of lights is constructed from each candidate answer word by inserting that letter at each possible position within the word, including at either end.

If the length of the message is less than the number of lights with answer treatment enabled, lights without a specified letter to be inserted are left unchanged.

12.2 Plug-in answer treatments

One of Qxw's most powerful features is its ability to use customised answer treatments in the form of plug-ins. This section describes the plug-in system in detail.

If 'Custom plug-in' has been chosen as the answer treatment method, Qxw will call the plug-in when building the feasible light list for each light with answer treatment enabled. This happens every time the user makes a change to the grid. Qxw calls the plug-in once for each word in the set of dictionaries selected for the light in question, passing in the word to be treated along with other information that the plug-in might need.

As you can see, a plug-in might get called hundreds of thousands or even millions of times whenever the user makes a change to the grid. Although Qxw will interrupt its feasible light list building to respond to a user action, it is nevertheless important that plug-in code execute as quickly as possible.

The plug-in takes the form of a C program, compiled using gcc with its `-shared` option. The resulting object file (which conventionally has a `.so` suffix) is dynamically loaded and unloaded by Qxw as necessary.

The program must provide a function called `treat()`. The function is passed a candidate answer word (as a `char*`), entirely in capitals and with no punctuation or spaces. It is expected to make a local modified version of this string (the 'treated answer'), again entirely in capitals and with no punctuation or spaces. The function must then call `treatedanswer()` with the modified string, which will be considered as a candidate for addition to the feasible light list. The function `treatedanswer()` returns a non-zero value if an error occurs, and this value should be passed back

as the return value of `treat()`.

In many cases a `treat()` function will call `treatedanswer()` exactly once, in which case the function can simply end `return treatedanswer(...)`; If the treatment is such that an answer can give rise to several different lights, `treat()` will call `treatedanswer()` more than once, and the returned value must be checked each time and returned if non-zero.

In addition, the plug-in may provide a function called `init()`, which is called immediately after the plug-in is loaded, and a function called `fini()`, which is called immediately before the plug-in is unloaded. The plug-in is reloaded whenever the user clicks 'OK' on the answer treatment dialogue, and so the `init()` function is a suitable place to do any relatively time-consuming pre-processing (such as building encoding tables) that the plug-in requires.

By including the header file `qxwplugin.h` a plug-in can access the following variables.

`int clueorderindex`, the sequence number of the current light in clue order, counting from zero. Only lights with answer treatment enabled are counted.

`int lightlength`, the length of the current light in characters.

`int lightx`, the x -coordinate of the start position of the current light (or angular position for circular grids), counting from zero.

`int lighty`, the y -coordinate of the start position of the current light (or radial position for circular grids, measured inwards from the perimeter), counting from zero.

`int lightdir`, the direction of the current light. For plain rectangular grids, the directions (in order counting from 0) are 'Across' and 'Down'; for hex grids with vertical lights 'Northeast', 'Southeast' and 'South'; for hex grids with horizontal lights 'East', 'Southeast' and 'Southwest'; and for circular grids 'Ring' and 'Radial'.

`char*treatmessage[]`, an array of two strings containing the messages as specified by the user in the answer treatment dialogue.

`char*treatmessageAZ[]`, an array of two strings containing the messages specified by the user in the answer treatment dialogue with all punctuation and spaces removed and letters converted to capitals 'A' to 'Z'.

`char light[]`, a temporary storage area enough space for the longest possible light (MXSZ characters) plus a zero termination byte. Plug-ins can use this area to construct the treated answer before passing it back to Qxw.

Plug-ins also have access to a function `int isword(const char*light)`; which checks whether a given string (entirely in capitals and with no punctuation or spaces) is a word in any of the dictionaries the user has selected for that light.

Chapter 13

Keyboard and mouse command summary

13.1 Keyboard commands

Keystroke	Menu item	Effect
A...Z, 0...9		enter character in grid
Space		advance cursor one position in current direction
Backspace		retreat cursor one position in current direction
Tab		delete letter from square and advance cursor one position in current direction
Return	<i>Edit-Bar before</i>	add bar before cursor position in current direction
Insert	<i>Edit-Solid block</i>	add block at cursor position
Delete	<i>Edit-Empty</i>	make empty square at cursor position
PageUp		change current direction one step clockwise
PageDown		change current direction one step anticlockwise
←, →, ↑, ↓		move cursor left, right, up, down
control-A	<i>Autofill-Accept hints</i>	accept hints (enter suggested letters in grey into grid)
control-C	<i>Edit-Cutout</i>	make cutout in grid
control-G	<i>Autofill-Autofill</i>	run automatic filler
control-M	<i>Edit-Merge with next</i>	merge current cell with next cell in current direction
control-N	<i>File-New</i>	start new grid
control-Q	<i>File-Open</i>	open a previously-saved file
control-S	<i>File-Save as</i>	save grid
control-Y	<i>Edit-Redo</i>	redo last undo
control-Z	<i>Edit-Undo</i>	undo last change

Keystroke	Menu item	Effect
shift-A	<i>Select-All</i>	select all lights or cells
shift-C	<i>Select-Current cell</i>	add cell under cursor to selection
shift-I	<i>Select-Invert</i>	invert current selection
shift-L	<i>Select-Current light</i>	add light going through cursor in current direction to selection
shift-M	<i>Select-Cell mode</i> <> <i>light mode</i>	switch between selecting cells and selecting lights
shift-N	<i>Select-Nothing</i>	reset selection
shift-control-G	<i>Autofill-Autofill</i> se- <i>lected cells</i>	run automatic filler on selected cells only
shift-control-X	<i>Edit-Clear</i> selected <i>cells</i>	clear selected cells

13.2 Mouse commands

Keystroke	Effect
left-click on cell edge	add/remove bar*
left-click on cell corner	add/remove block*
left-click on cursor	change current direction
other left-click on grid	move cursor
left-click in feasible word list	enter word in grid at cursor position
shift left-click	select/deselect cell
left-click and drag, holding shift	continue selecting/deselecting cells
shift right-click	select light in current direction
right-click and drag, holding shift	continue selecting/deselecting lights in current direction

* If function is enabled: see Section 9.1.