

## **LES Map Editor V2**

COLLABORATORS			
	TITLE : LES Map Editor V2		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		August 9, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>LES Map Editor V2</b>	<b>1</b>
1.1	Main Menu . . . . .	1
1.2	les_introduction . . . . .	1
1.3	les_shareware . . . . .	2
1.4	les_docs . . . . .	3
1.5	les_screenlayout . . . . .	5
1.6	les_menustrip . . . . .	5
1.7	les_drawingcontrol . . . . .	8
1.8	les_informationarea . . . . .	9
1.9	les_controlarea . . . . .	9
1.10	les_shapescroller . . . . .	10
1.11	les_viewarea . . . . .	11
1.12	les_shapeselector . . . . .	12
1.13	les_groupselector . . . . .	13
1.14	les_rangeselector . . . . .	14
1.15	les_configuration . . . . .	14
1.16	les_surfacedata . . . . .	16
1.17	les_reflecttable . . . . .	16
1.18	les_shapenames . . . . .	18
1.19	les_mapshapeadjust . . . . .	18
1.20	les_loadingshapes . . . . .	19
1.21	les_loadingmaps . . . . .	20
1.22	les_savingmaps . . . . .	20
1.23	les_savingcustommaps . . . . .	21
1.24	les_saverawmap . . . . .	21
1.25	les_custommapfileformat . . . . .	22
1.26	les_tooltypesupport . . . . .	24
1.27	les_generatesource . . . . .	26
1.28	Arexx Support . . . . .	26
1.29	Arexx Loader/Saver . . . . .	28
1.30	les_credits . . . . .	30
1.31	les_index . . . . .	30

---

## Chapter 1

# LES Map Editor V2

### 1.1 Main Menu

```
LES Map Editor V2
=====

(formerly RI Map Editor)

LES Map Editor V2  (c)1994 Leading Edge Software

System Requirements:
-----

1Mb+ of memory
OS2.0 at least

Introduction
Shareware
Docs
Credits

Main Index
```

### 1.2 les\_introduction

```
Introduction
=====
```

```
Who are Leading Edge Software?
-----
```

Leading Edge Software is a team of university students (and staff) aiming to produce quality software for the Amiga range of computers.

The LES team:

```
Steven McNamara - Code,Documentation,Gfx ( 68000 expert :-) )
Steven Matty    - Code,Documentation,Gfx,Sampling
```

---

Steven Green - Code?, Documentation, Gfx, Sampling  
Steven Innell - Music, Documentation, 3D Gfx  
Robert Brennan - Gfx  
Nigel Hughes - Code, Documentation, Gfx  
Mike Richards - Gfx

The first project to be undertaken by the group was a user friendly Map Editor suitable not only for games but many other applications as well (this will become apparent further within these docs). Enjoy :-)

Contact us!  
=====

Note: If you don't know some of the gibberish in this section just ignore it - you don't have access to what I am talking about :-)

There are a number of ways to get in contact with Leading Edge Software, these being: EMail, Normal Mail, Phone or even Talk on the UNIX systems.

Normal Mail:     Leading Edge Software  
                  Stephen McNamara,  
                  17 Mayles Road,  
                  Southsea,  
                  Portsmouth,  
                  Hants.  
                  PO4 8NP.

Telephone: England

(01705) 781507 - Steven McNamara  
(01705) 611522 - Steven Green

EMail:           sis3149@sis.port.ac.uk - Stephen McNamara  
                  sis3099@sis.port.ac.uk - Steven Green  
                  sis3147@sis.port.ac.uk - Steven Matty  
                  spi@dcs.qmw.ac.uk       - Steven Innell

You can also (Y)Talk us at either  
                  sis3149@anubis.sis.port.ac.uk  
                  or     sis3099@anubis.sis.port.ac.uk

Please feel free to mail us if you want a chat about this or anything else we have done.

## 1.3 les\_shareware

Shareware  
-----

Please help support this program by registering for it. It is only by people registering that this program will continue to grow in the future. You'll also ensure that our group as a whole continues to release public domain and shareware software. If you intend to use this editor, as with all shareware software, you should really register.

---

The editor is not crippled in any way, nor has it been made awkward to use (for example, requesters popping up randomly whilst using it). There is nothing to make you register (except loads of extra stonking features!), we're just appealing to your sense of justice. Support the shareware ideal.

The registration fee for the LES Map Editor is five pounds sterling.

Things on the 'Will do if people register' are:

- o Shape editing - via an arexx link to ShapeZ (a shape grabber/editor written by Nigel Hughes)
- o Palette editing and saving.
- o Definable keyboard short-cuts.
- o Better memory/screen mode handling.
- o ARexx loader/saver list for automatic file format selection.
- o etc.....

Registered Version

-----

The registered version of this program already supports the following cool features:

\* Multiple map files in memory at once - maps can be any size but currently share the same shapes and palette. Multiple maps means though that you can easily cut 'n' paste between maps (using the Group functions of the Map Editor).

\* Loads more arexx support, including: shape requester, adding menu items (so an arexx program can add its own menu items/subitems to the Map Editor's menu strip).

\* And errrrmmmm wellllllll and more :).

Now, on with the docs .....

## 1.4 les\_docs

What is the LES Map Editor?

=====

The LES Map Editor is a program written is written in Blitz Basic 2 and 680x0 assembler. The editor is geared towards to Blitz Basic 2 developers, but this does not mean that it is restricted for use within Blitz Basic. It could be used for any type of map creation, for any language.

Basically a Map Editor simplifies the task of creating maps for your games, projects, designers, puzzles etc. etc. In fact it may be used for any such task in which a map is needed. Such an example for the type of maps that can be created would be an overhead view game such as pacman or dynablasters, even sideviewing games can be designed e.g.. the shareware game StarWoids had its maps created using the LES Map Editor.

This Map Editor is different to others however through its user friendliness and ease of use - you can even grab iff pictures of blocks with it and use them as blocks for use with your map. This means that the only program you need to create you map from scratch is the LES Map Editor.

The LES Map Editor uses Intuition fully, thus allowing anyone to use it quickly and easily. As long as you, the user, have a basic understanding of ideas like menu strips, windows and buttons (gadgets) then you should have no problems getting into, and using, this editor.

Some Jargon and references....

Some simple jargon used inside this doc file:

Blocks/shapes - these are the graphics that you're using to build up your map. They can be thought as of bricks that are built up into wall, where the wall would be your map.

Groups - groups are 'collections' of blocks. They are used to make repeated collections of blocks easier to use.

Maps - this is the actual data you are editing. Maps are built up from shapes and groups using the editor and saved out for inclusion in your game.

Blitz2 Map Editor - this is the original map editor supplied with the Blitz Basic 2 package. The LES Map Editor is partially based on this editor, and includes practically all of its features. Any Blitz2 map editor users should find the transition to this editor very painless.

#### Screen Layout

Editor Menustrip  
Drawing Control Area  
Information Area  
View Area  
Control Area  
Shape Scroller  
Group Selector  
Range Selector  
Configuration  
Surface Data

#### Reflect Table

Shape Names  
Map Shape Adjust  
Loading Shapes  
Loading Maps  
Saving Maps  
Saving Custom Maps  
Save Raw Map  
Custom Map File Format  
Tooltype Support  
Generate Source

Shape Selector

ARexx Support

## 1.5 les\_screenlayout

The screen layout  
=====

The LES Map Editor screen is laid out into a series of vertical sections. These are:

- o Menu Strip
- o Drawing Control Area
- o Shape Scroller [optional]
- o View Area

These areas will be discussed one after the other.

## 1.6 les\_menustrip

The Map Editor Menu Strip  
=====

Most of the features of the LES Map Editor are reached via the menu strip. Map editor has 3 menus: the Project menu, the Prefs menu and the operations menu. Following is a complete list of menu options available, with keyboard short-cuts and explanatory text. All on menus are shown with the standard 'Amiga' symbol next to their name to help you identify them.

On the Project menu:

The items in the project menu allow you to load and save different file types, as well as control the current map and shapes.

New map Amiga n - brings up the new map requester, from where you can create a new map or load one off disk. You can cancel this requester if you like, in which case nothing happens to your map. Otherwise, a new map will erase your old one.

Clear map Amiga c - opens a requester asking you if you really want to clear the current map. Clicking okay will erase everything in the map.

Resize map Amiga z - brings up a requester that looks like the New map requester except that here you can resize either the map size or the block size for shapes. Clicking Okay will change the size of the map without erasing it. You will, though,



loose data if you resize the map to smaller than its original size.

Changing the block size in this requester will cause all current shapes to be scanned to make sure they are no bigger than the new size. Shapes that are larger than the block size will be cleared.

Load map Amiga l - brings up a file requester from where you can select either a custom map or a raw file to load into the editor. When you select a file, the program will automatically try and identify the file type. If it doesn't manage to identify it, a requester will open up asking you to select the file type to load it as. This file requester remembers the path you last loaded from for easy map editing.

Load shapes Amiga a - opens a file requester asking you to select a shapes file to load. Selecting a file will cause the Shape Load window to open up. Here you select some options about how the shapes in the file should be loaded in or can cancel the load. See the section on Loading Shapes for more information.

Load palette Amiga p - brings up a file requester asking you to select an IFF file from which to load a palette from. The file can be a picture, anim or just a palette file on its own (e.g. saved out from DPaint's Save palette option).

Save map Amiga s - opens a file selector from where you select the filename to save the current map as. You will be warned if the file already exists and asked if you really want to overwrite it. Selecting a file will cause the Save Map window open where you can select what information you want saved in the map file. See the section on Saving Maps for more information.

Save raw Amiga S - opens a file requester from where you can select the name of the raw map file to save. Selecting a file will cause 2 requesters to open up asking you how you want the raw map to be saved. Select your options here and the map will be saved to disk. See the section on Saving Maps for more information.

Generate source Amiga G - brings up a window asking you to choose exactly how the Blitz source code will be generated. There are many options to choose from, you should look at the section on Generating Source for more information about them. Clicking cancel will abort this function whilst okay will open a file requester where you select where to save the source to.

About Amiga ? - opens up several requesters giving program information and details of the author.

Quit and die Amiga q - opens a requester asking you to confirm that

---

you really do want to quit. Selecting okay will cause the program to end and all current work to be deleted. Please save out any maps that you wish to keep before you select this option.

The Prefs (stands for Preferences) menu:

This menu allows you to configure the map editor environment.

Configuration Amiga / - brings up windows where you can configure almost everything inside the editor. There are three configuration windows, and each has a More button which allows you to jump to the next one.

See the section on Configuration to get more information about all the configurable options.

Iconify Amiga I - iconifies the program to either the Tools menu on the Workbench screen or a window on the Workbench screen depending on the iconify mode selected in the configuration. Iconfying the program causes its screen to close and some memory to be frozen up. It does not quit the program - instead it just puts the program to sleep until you need it again.

Screen mode Amiga m - opens up a ReqTools screen mode requester from where you can select your preferred mode (e.g. Lowres, hires etc.). Clicking on okay will cause the map editor to reopen its screen in the new resolution straight away. Only standard screen modes are available (e.g. no HAM modes). Note: when you save your configuration, the screen mode details are automatically saved with it so that the editor can open its screen in your preferred mode when next run.

Remap shapes Amiga e - whenever a palette is loaded into the editor, the bottom four colours are automatically altered by the editor to preserve the bottom four colours for its own use. This means that the editor will always look the same. This does also mean, though, that graphics that use the bottom four colours will be corrupted slightly. The remap option allows you to change all occurrences of the bottom four colours in the shapes to spare colours at the top of the palette. This means that all shapes will be viewed correctly inside the editor. Remapping can only be done if your shapes are in less than 256 colours and you have a palette loaded. Also note that the palette loaded isn't actually altered by this command. Thus you can save out the palette in a custom map without worrying about it being corrupted.

Surface data Amiga d - opens up a window from where you can edit the surface data for the currently loaded shapes. See the section on Surface Data for more information.

Reflect table - has a sub menu where you decide what you want to do to the reflect table. the submenus are: Load, Edit and Save. Selecting a submenu will cause the

desired function to run.

See the section on Reflect Table for more information.

Shape names - brings up a requester where you can select whether you want to load or save shape names. You can also cancel this requester.

See the section on Shape Names for more information.

The operations menu:

This menu allows you to perform several different, map related operations. You should note, though, that options on these menus *cannot* be undone. For example: once a Fill Map Range function has been performed, you cannot use the undone function to restore the original data.

Goto map x,y Amiga g - opens a window from where you can select whether or not the move is relative or absolute. This window has two gadgets in which you can enter the coordinate to go to, these gadgets are automatically loaded with the current screen position when the window is opened. Selecting Okay will cause the move to be performed, Cancel will abort.

Fill map range Amiga f - this first brings up the shapes requester for you to select a shape to fill with. It then opens a window from where you select the area of the map to fill. You type in the position and size of the fill into the gadgets and click Okay to perform the fill, Cancel will abort.

Replace shape Amiga r - this function replaces all occurrences of one shape in the current map with the current shape. It will first ask you to select the shape to replace from the shape requester. Then it will open a window from where you can select the area of the map to replace. Only shapes within this area will be replaced. Click Okay to perform the replace, or Cancel to abort.

Map flip Amiga h - brings up a requester asking you in which direction you want to flip. After selecting either X or Y it opens up a window from where you can select the area of the map to be flipped. Selecting Okay will perform the flip, else Cancel will abort.

Note: this operation uses the Reflect Table.

Map shape Amiga - Allows you to alter the map data directly by adjusting subtracting an integer value from all positions within a given map range.

See later in this doc for a full discussion of this function.

## 1.7 les\_drawingcontrol

The Drawing Control Area

=====

---

This area is located directly under the menu strip. It is split into the separate areas, these being the Information area and the Control area. Here you are given information about the current position on display in the map and given the chance to change the current drawing mode.

## 1.8 les\_informationarea

Information Area  
=====

The Information area is located on the left of the Drawing Control area and has two lines of text on it. The first is titled MAP POS, this stands for map position and tells you the map coordinates underneath the mouse pointer. It also shows the shape number at that position in the map, If the mouse is not inside the drawing area, this line will be blank. It will also remain blank, if you have not selected "Onscreen coordinates" from the configuration screen from the Prefs menu.

The information is shown in the form of:

MAP POS: x,y,shape number

although shape number will not appear if the map position is not defined (e.g. you have drawn anything there yet).

The second line of text in this area shows the current view coordinates of the view in relation to the map. It is shown in the form of:

VIEWING: x1 to x2,y1 to y2

where (x1,y1) are the coordinates, in the map, of the top left hand corner of the view area, and (x2,y2) are the coordinates of the bottom right hand side of view area. If the current map is smaller than the viewing area, (x2,y2) will be the bottom right corner of the map.

## 1.9 les\_controlarea

Control Area  
=====

The Control area is located on the right of the Drawing Control area. It contains a series of gadgets that allow you to control the map drawing process, these gadgets are:

Gadget	Shortcut	Description
Shape	<Space>	Brings up the shape selector window
Group	g	Brings up the group selector window
L(eft)	l	Asks you to click on a shape in the map to assign to the left mouse button. If you hold down <Shift> whilst pressing the shortcut, the shape under the mouse pointer

is automatically selected.

R(ight) r Asks you whether you want to select a map block to assign to the right mouse button or delete the current assign. If you click on select you'll be asked to select a shape from the map. As with L(ef), pressing <Shift> and the shortcut will cause the shape under the mouse pointer to be automatically selected.

The following are only available if you are using a screen that is wider than 320 pixels. If your screen is only 320 pixels wide (low res) then these functions can only be activated from the keyboard shortcuts.

Undo u Causes the last shape or group drawing operation to be undone. Clicking on undo again will cause the original data to return. Note that the undo buffer can only undo a maximum of 1000 shapes.

Full f Fills the view area with all the current shapes. From here you can then select a shape for either the left or right mouse button by clicking on one. If you have more shapes than can be viewed on screen at once, you can scroll the area up and down with the up and down cursor keys.

Name n brings up a window with a list of all currently defined names, from here you can select a shape to put on the left mouse button by name. Just click in the list of names and hit Okay.

Draw/Box/Line <none> selects the current drawing mode for shapes only - not groups. Changing the mode automatically changes back to shape drawing.

## 1.10 les\_shapescroller

### The Shape Scroller

=====

The shape scroller is located directly below the Drawing Control Area. It allows you to view a range of the currently loaded shapes and select different shapes without the need to open any windows. Just clicking on a shape in the scroller with either the right mouse button or the left mouse button will automatically select that shape for the mouse button pressed. Note that you must release the button to continue.

The scroller has two scroll buttons, these are located at opposite ends of the scroller and allow you to move it left and right through the shapes. Click on these buttons with the left mouse button to scroll one shape at a time in the selected direction, or click with the right mouse to move to the start or end of the shapes (depending on what button you click on).

The scroller automatically adjusts its size to fill the whole horizontal screen width. All shapes inside the scroller, are shown as 16x16 blocks,

this means that a shape larger than this will be clipped to fit inside this size. If you haven't got enough shapes loaded to fill the scroller, the extra slots will be meshed out and will be unselectable.

Last point about the scroller: if you do not like the scroller, you can switch it off from the configuration menu. Switching it off allows the view area to expand vertically by approximately 17 pixels. This can be useful if you're trying to get as much of a map onscreen as possible.

## 1.11 les\_viewarea

### The View Area

=====

This area is the actual view area for the map you're editing. It takes up most of the screen's height and always fills the full width of the screen.

Moving the mouse pointer into the scroll area will cause the current shape (or group) to be drawn under the mouse pointer. This draw is just to show you what the shape will look like when actually look like when drawn down - it is not a permanent change. Moving the mouse around the view area will cause the current drawing shape(s) to move with the pointer, giving you a preview of what it would look like if you drew them down.

Drawing with shapes is easy. Simply press the left mouse button and release to draw the current shape(s) under your mouse pointer, or press and move the mouse to draw the shape(s) at several different positions at once. Pressing the right mouse button will cause a different shape to be drawn on your map, depending on the current setting of the button. The default for the right mouse button is delete, this means that when you draw, instead of a shape being drawn onto your map, the current shape at the map position is deleted. You can, though, assign a shape to the right mouse button (by either selecting one from the shape scroller, pressing R on the keyboard or selecting the 'R' gadget on the screen). When drawing with this, any shapes you try and draw will be replaced with selected right mouse button shape.

It is fairly unlikely that your map is going to exactly fit into the view area. When your map is smaller than the view, the area inside the view where you cannot draw will be meshed off. You will be unable to draw anything inside this meshed area. You will, though, only get a mesh if you have the mesh option selected from the configuration windows, otherwise the area will be blank. If your map is too big for the view you are able to scroll around it with the cursor keys. This lets you create and edit a map that is much larger than the current view can show with ease. Pressing a cursor key will scroll the map in the given direction one block unit (the unit will be the block width for X scrolling or the block height for Y scrolling). If this isn't fast enough for you, you can press shift at the same time to jump a full views worth of map. Note, though, that you'll never be able to scroll outside of the map area. If you wish to move to a specific coordinate or wish to move large distances, you can use the Goto Map XY menu option to jump to a certain position.

## 1.12 les\_shapeselector

The following are additional requesters that are used to help you, amongst other things, select the current drawing shape/group.

### Shape Selector Window =====

This window allows you to select a shape from the currently loaded ones. It is called up by pressing either on the 'Shape' gadget on the Control Area or by pressing <SPACE> on the keyboard. When used in this way, it allows you to easily select a shape to place on the left mouse button.

Inside the window, you will find a set of gadgets that let you move up and down through the currently loaded shapes, as well as some info about the current shape. On the left and right sides of the window are two sets of two gadgets. They are for changing the current shape, the bottom top gadget of each set moves one shape at a time, whilst the bottom one moves ten shapes at a time. This gadgets can be activated from the keyboard, use '<' and '>' to decrease or increase the current shape by one, or press <Shift> and either of them to decrease/increase by ten shapes.

The number of the current shape is shown at the top of the window. Directly underneath this there is a display box in which the current shape is shown. If the shape is larger than the box, it will be clipped to fit inside it. The last bit of info that is shown about the shape is the name of it. This is displayed in a text gadget directly under the view area, you can type a new name for the shape directly into this gadget. The name should be in the form of a variable, e.g. Backgroundblock. You can have numbers in the name, but they should always appear at the end of it, e.g.:

```
Backgroundblock1  is allowed
Backgroundlblock  is not allowed
```

The name text gadget can be activated from the keyboard, just press [Return]. Note, though, that whilst the text gadget is active, you cannot use any of the keyboard shortcuts for the window.

An extra feature of this window is that you can select a shape from the map. All you have to do is click outside the window on a shape in the view area. This will cause the current shape to automatically switch to the selected shape.

To exit the window and select the current shape for the left mouse button, just press on the Select gadget at the bottom of the window or press the S key.

### Operations =====

Several of the functions on the Operations menu will call up the shape selector window. When this happens, you will not be selecting a shape to draw with, but rather a shape to perform the operation with. An example of this is the replace operation, in this you'll be asked to select a shape to replace with the current shape. From the selector window you'd click on

the shape you wanted replaced in your map.

## 1.13 les\_groupselector

### Group Selector Window =====

This window allows you to control groups inside the map editor. From it you can select, create or delete any groups available. The window can be invoked either by clicking the gadget Group in the Control Area, or by pressing the G key.

The group window is very similar to the shape selector window in that it shows you info about the current group and allows you to cycle through the groups. Where it differs, though, is in that it has a series of extra gadgets that are specific to groups, these are:

**Xflip/Yflip:** This allows you to flip the current group either horizontally (X) or vertically (Y) about its center.

**X0.5:** It is possible that your group may be too big to display in the view area of the window. If this happens you can select this gadget to enable viewing at half normal size. With this gadget 'checked' all groups will be shown at half their normal size inside the view area. This will not affect drawing with the groups.

**R.T.:** This stands for Reflect Table and says whether or not you want the reflect table to be used when groups are flipped horizontally and vertically. If this gadget is checked the reflect table is on for group flipping.

**Create:** Clicking on this will allow you to edit a group from scratch. The selector window will be hidden, and you'll be able to click with the left mouse button on parts of the map to add items to the group. The relative position and shape number of the map square you click on will be stored in the current group. You cannot add empty positions to a group, and a group can have a maximum of 100 shapes in it. You can also only select each position once, the position will be highlighted to show that it has been added okay. Press with the right mouse button to end creation of groups and return to the selector window. Note: the current group will be replaced by the new one when you perform this function.

**Delete:** Allows you to clear the current groups data. Note, though, that you cannot undo this delete.

**Select:** Exits the selector window and puts the map editor

---



into group drawing mode. If the group you have currently selected has no shapes in it, the mode will automatically be changed back to shape draw and will use the last selected shape.

Please note that all these gadgets have keyboard shortcuts. They will all have an underline character in their text showing the key to press to activate them.

## 1.14 les\_rangeselector

Range Selector Window  
=====

This window is used by all the functions that just act on a specific area of the map. In it, you select the area of the map that the function should use, or cancel if you wish to abort the function.

There are 8 gadgets in the window, and these are split up into 2 groups.

<String gadgets>

Start Map X: Set the start x position for the function.

Start Map Y: Set the start y position for the function.

Width to do: Set the width of the function (must be at least 1)

Height to do: Set the height of the function (must be at least 1)

<Button gadgets>

From View: Sets up the start positions, width and height to reflect the current map view area. This can be used to quickly select the visible area of the map.

From Map: Sets up the start positions, width and height so that the function will work on the entire map.

Okay: Exit the requester and perform the function.

Cancel: Abort the requester and cancel the function.

## 1.15 les\_configuration

Configuration  
=====

The map editor is very configurable. You can change a lot of options to customize it to just how you like it. You can also save all preferences so that it will automatically load into the correct mode when next run.

The configuration section consists of 3 windows, entitled Default Paths,

Misc 1 and Misc 2. All these requesters have 3 buttons at the bottom of their windows:

- o More - skips to the next window
- o Save - save the current configuration
- o Okay - exit configuration section

#### Default paths =====

From here you can select the default paths to load/save maps from, load shapes from and load palettes from. You can either enter a new path into the string gadgets, or you can click on the gadgets next to the paths to bring up a requester from where you can click a path with the mouse.

#### Misc 1 =====

This contains a series of different gadgets:

- o Buffersize: set the size of the buffer used for undrawing shapes and groups on the map. The size will be dependant on the depth of your screen and the number of shapes in the groups you're drawing with. If in doubt, set this value high (max is 500k) to ensure that a crash will not occur.
- o Iconify to: this cycle gadget lets you select whether to iconify to the Tools menu or a window on the Workbench screen.
- o Iconify end: this is used when you are iconifying to a window. It lets you select whether the right mouse button ends the iconify, or the zoom gadget of the window.
- o IconifyX: default x position of the iconify window.
- o IconifyY: default y position of the iconify window.
- o Use AppIcon: select whether or not an appicon appears on the Workbench screen. The appicon can have maps dropped on it to automatically load them in.
- o Interleaved: lets you select whether or not the screen is interleaved (OS3.0+ only). Interleaved screens flicker much less when they're scrolled and so are a lot nicer to work with.

#### Misc 2 =====

- o Shape remap: decide what happens when shapes are loaded whilst a palette is also loaded. Options are:
    - No - don't remap shapes
    - Yes- automatically remap shapes
    - Ask- open a requester asking whether or not to remap the shapes straight after
-

loading them

- o Use mesh: lets you select whether or not areas of the view are that you cannot draw on are meshed out. If this checkbox is 'checked' then the mesh is on.
- o Shape scroller: switch shape scroller on and off. If checked then the scroller is on.  
See the section named Shape Scroller for info about the scroller.
- o Quiet shape remap: decide whether remapping shapes gives a progress indicator or if it just sets the mouse to a busy pointer. Quiet remapping is slightly faster than using the progress indicator.
- o RIMP icon files: select whether to save icon files when custom maps are saved.
- o Onscreen co-ordinates: shows x,y coords, and shape number next to MAP POS: in the information area.

## 1.16 les\_surfacedata

### Surface Data =====

Surface data allows you to assign separate values to shapes. When you save custom maps, you'll be given an option to save out an extra chunk of data that will hold a copy of your map, except that all shape numbers are replaced by surface data. Thus you could build, for example, a tree out of 6 separate blocks, but assign them the same surface value. Your program could then use the main map data to draw an onscreen map and use the surface data map for something like collision detection.

The surface data window consists of two shape display boxes. Each has a slider so that the shape displayed in it can be changed to any of the currently loaded. Each also has a gadget for entering an integer value for the shapes surface data.

There are two display areas so that you can easily set the surface values for a range of shapes. In the center of the window there is a 3 integer gadget. This gadget holds the value to assign to either all shapes or a range of shapes. To set a range, set up one display area to point at the first shape in your range, and set the other to point at the last. This range can then have one value assigned to every shape in it by clicking the 'Set Range' button.

## 1.17 les\_reflecttable

## Reflect Table

=====

The reflect table allows you extra control over flipping maps and groups. In the reflect table, each shape loaded has a xflip and a yflip shapes number. What these means is that when a shape is flipped, it automatically changes to the alternate shape in the table. Thus xflipping a shape in a group or map would cause it to change into an alternate shape.

Let me give an example to illustrate this. Say you had some tunnel shapes. On represents a left exit from a tunnel, one a mid-section of tunnel and the last a right exit from a tunnel. These shapes in a map could represent a tunnel section in your map. They'd be drawn out with my left exit, then the mid and then the right exit, going horizontally across the screen. Now what would happen if you flipped just these 3 blocks? The left and right exit would be reversed, giving a corrupted tunnel that would need redrawing to make it look correct.

To get round this, you could use the reflect table. You'd set it up so that the left tunnel exit flipped horizontally into the right exit. Then when the area of map was flipped, the exits would automatically change into the correct exit. This is a fairly simple example, a larger one could involve a number of blocks in a shoot-em-up. It would be likely that a lot of the background blocks would just be the x or y flips (e.g. flipping the 'image' if a shape when drawing) of other blocks. Thus you could draw an area of map, and using the reflect table, easily flip this and paste it somewhere else for my variety in your level.

## Editing the reflect table

=====

The edit window consists of 3 shape display areas. The one in the top left shows the current shape being edited. The other two show what the shape currently flips into in the x and y directions. These can be changed using the sliders next to each area to change what the shape flips into.

There are several buttons in this window, these are:

- o **Correct:** this looks at the shapes that the current shape flips into. It then makes it so that those shapes will flip into the current shape. This can be explained from the tunnel example given earlier. When you flip the left tunnel exit horizontally, you want it to change into the right tunnel exit. Inversely, when you flip the right one, you want it to turn into the left one. To do this, all you'd have to do is set up the left to flip into the right and then press on 'Correct'. The right exit will then automatically be adjusted to flip into the left exit.

- o **Reset ALL:** resets all shapes so that they flip into themselves.

- o **Okay:** Exit the window and keep any changes.

- o **Cancel:** Exit the window but don't keep any changes.

## 1.18 les\_shapenames

### Shape Names =====

In the map editor, individual shapes can be named. This can be very helpful as in programs it is very useful to address individual shapes by a constant (e.g. #background) than an actual number. If you use constants, you can change orders of shapes and delete shapes in your files without having to go through your program making changes to all your blits etc. All you have to do is change your maps and change the constants at the top of your program.

Names are entered in the shape select window. Inside this window is a string gadget, inside which you can enter a name. Don't put a '#' at the start of the name - when you save names, they automatically have a '#' added to the front of them. See the section on the shape select window for more information about entering shape names (e.g. what names are allowed).

### The 'Shape Names' menu item =====

Clicking on this menu item will cause a requester to open. From here you can either Load or Save shape names. Shape names are stored in a standard ASCII file that can be inserted straight into any source code. They are saved out in the following form:

```
'#'+<shapename>+'='+'<shapenumber>
```

e.g. if the name for shape 20 is Backgroundblock, it would be saved as:

```
#Backgroundblock=20
```

Assembler programmers will need to change the shape name file after inserting it to replace all '=' with ' equ '.

If you are loading, you should make sure that the file you select is actually a shape names file. Selecting a different file could lead to a crash.

## 1.19 les\_mapshapeadjust

### Map Shape Adjust =====

This menu item allows you to add or subtract an integer value from a specific area of the current map. It is a fairly complex process, in that a series of requesters have to be replied to before the process actually takes place.

What actually happens is that you select firstly a shape range to perform the adjustment on. Only shapes within this range will actually be altered on the map, but the alteration may cause the final shape number to 'leave' this range. The adjusted value will never be more than the currently

loaded number of shapes.

Then you must enter an integer value to added to all the shape numbers in the current map. This value can be either positive (to increase the shape numbers) or negative (to decrease the shape numbers).

Next you select the map range to change. when the adjust is performed, it will only have any effect within this range.

After you have entered all the above information, you'll be asked to *\*finally\** confirm that you want to actually perform the change. Beware, though, the change cannot be undone.

Let me list out those steps to try and make it a bit clearer:

- o Select shape range. This is done with two visits to the Shape Select Window
- o Select adjust value. Done using a standard string requester
- o Select map range. This is done using the Range Select Window
- o Confirm that you want to actually perform the adjust

And that's all you have to do! Simple, huh? ;-)

## 1.20 les\_loadingshapes

### Loading Shapes =====

Shapes can be loaded in two ways. Either from a Blitz2 shapes file or from an IFF picture file. If you select an IFF from the file selector, the editor will automatically load and scan the picture for shapes. It finds the edges of all graphics in the iff and cuts individual shapes out. It will also ask you whether or not you want to grab the palette from the iff for use in the editor.

Once you've selected a shapes file from the file selector you'll be presented with a window from where you can select how to load the shapes in. Options will have checkboxes next to them, click these on or off affects the status of the options. Valid options are:

- o Reject smaller size - blocks that are smaller than the current block size will be replaced with a blank shape if this option is enabled. Note that shapes larger than the block size are always cleared.
- o Reset handles - if enabled automatically sets the shape handles to the top left corner. This option *\*should\** always be enabled since shapes with different handles could possibly cause a software failure.
- o Overwrite current - if enabled, this will load the new shapes

at position 0. If disabled, the last current shape will be calculated, and the shapes will be loaded after these. This is useful if your maps are made up off several different shapes files.

- o Quiet mode - if enabled, this will skip the status information for shape scanning. This speeds up loading but you loose out on information about whether shapes are being resized etc.

## 1.21 les\_loadingmaps

### Loading Maps =====

Clicking on the Load Map menu item will cause a file requester to open. From here you'll be asked to select a map file to load. This can either be a custom map, a raw map or a map from the original Blitz Basic 2 map editor.

When you select a map, the map editor will automatically try and identify it. It will look at the start of the file and try and find the custom header. If it finds this, it will perform a custom map load where it will look for information in the map, and ask you to confirm which chunks of data you want loaded from the map. The map data will always be loaded, but you may select whether or not to load extra info, for example palette, surface data and groups.

If the file is not identified, a requester will open asking you to identify the map as raw (in which case you'll be asked what form the data is held in) or as a Blitz2 map file. If you select Blitz2, the map editor will automatically look for the <mapname>.mapstuff file that is associated with Blitz2 maps. If this file cannot be found, your map may become corrupted as this file holds the map and block dimensions.

## 1.22 les\_savingmaps

### Saving Maps =====

The map editor supports saving in two formats: custom and raw. Both of these formats have there good and bad points and are suitable for different stages of a programs development.

Custom maps always come out at least slightly larger than raw maps. They do though hold a lot more info than raw maps. For example, you can save a palette, a pointer to a shapes file, group information and surface data inside one custom map file. Thus when you load this one file into the editor, you'll automatically have all your groups back, plus the palette and the shape file you originally loaded. This can take a lot of hassle

---

out of editing map files, as it simplifies the process of loading in a map.

On the negative side of custom maps is that fact that anybody could view your map files simply by loading them in. They could also rip out your maps as size info etc. is held all in one file. Depending on what sort of project you're working on, this could be a crucial point.

Raw maps are simply what they say: they are just a chunk of memory saved out that describes the map. They hold no size information at all, or any other type of info, except 'this shape goes here' info. They are always smaller than custom maps. They are thus perfect for a finished product as it anybody wanting to view them would have to have some info about their dimensions before viewing could be done. For development, though, they are not as good as custom maps because you have to manually load in extra data (like a palette) and cannot save group information.

It is recommended that you use the custom maps during development of your programs, and then use raw maps in a finished product is necessary (e.g. for security).

When saving either raw or custom maps, you'll first be asked to select the save filename from a file selector, and then asked how you want the data saved. When you select the name, you'll also be asked to confirm an overwrite of an existing file (if it does already exist).

## 1.23 les\_savingcustommaps

Saving custom maps  
=====

When you save these maps, a window will open from where you can select exactly what data to save with your map. There will be a series of checkboxes in this window, which you can click on or off depending on what you want saved. Valid options, at this time, are:

- o Save palette in map
- o Save group information
- o Create Icon file \*
- o Pointer to shape file (stores filename in map)
- o Surface data

\* Creates an icon file for the saved map, with the map editor as the default tool for it.

After selecting your options just click on the save gadget to go ahead with the save or click cancel to abort.

## 1.24 les\_saverawmap

---



```
Save raw map
=====
```

After selecting your filename, you'll be asked to select the size of data to save. The options given to you are: Byte, Word and Cancel. Byte should only be selected if you use less than 128 shapes in total (this takes into account the fact that a basic doing a Peek.b will sign extend the number). Byte files, though, will come out as half the size of word files. The only other difference is that slightly different code is needed to use each type of data in your own programs. The byte option is mainly of us to people who already use a map editor that saves out data in bytes (like the Blitz2 map editor).

A second option will be given to you if you don't cancel the first requester. This option asks you what value you want to identify a null position in the map when saved. Possibly answers are: 00, -1 and cancel. 00 is the default, and means that the first shape number available for use in the map is 1. -1 is available to help those who have previously used different editors that use this value for null slots. Note that custom maps use 00 for a null position.

The save will then go ahead as long as you didn't select cancel from either requester.

## 1.25 les\_custommapfileformat

## Custom Map File Format

Here is the structure of map files saved out using the custom map mode.  
Note for the uninitiated:

```
.l = longword value (4 bytes)
.w = word value (2 bytes)
.b = byte (errrrrrrrrr..... 1 byte!)

+4 = 4 byte offset from start of data
```

Map data is saved out in 'chunks'. These chunks have the general form:

```
4 byte header: identifier. e.g. CMAP = palette data
4 byte size  : byte size of chunk
...          : actual data
```

Custom Map Header: (always present)

```
+0.1      "RIMP"      ;File identifier
+4.w      block width ;pixels
+6.w      block height ;pixels
+8.w      map width  ;in blocks
+10.w     map height ;in blocks
```

Version chunk: (always present)

```
+0.1    "VERS"      ;Chunk identifier
+4.1    size of chunk ;in bytes
+8.1    "2.00"      ;4 byte string holding
           ; version number
```

Map chunks follow in the following order (if present in map file):

Palette:

```
+0.1    "CMAP"      ;Chunk identifier
+4.1    size of chunk ;in bytes (SAME AS IFF ILBM)
+8,9,10.b r,g,b    ;values for colour 0
+11,12,13.b r,g,b  ;values for colour 1
etc.... for all colours
```

Groups:

```
+0.1    "GRPS"      ;Chunk identifier
+4.1    byte size of chunk
+8.1    number of groups
```

for each group:

```
+0.w number of blocks
+2,+4,+6.w block 1,deltax,deltay,block number
+2,+4,+6.w block 2,deltax,deltay,block number
```

Shapes:

```
+0.1    "SHAP"      ; chunk identifier
+4.1    size of chunk ; in bytes
+8..... filename of shapes file
```

Surface data: (map data)

```
+0.1    "SURF"      ; chunk identifier
+4.1    size of chunk ; in bytes
+8....   map data ; (stored same as in MAPD chunk)
```

Surface data: (array of values)

```
+0.1    "SURD"      ; chunk identifier
+4.1    size of chunk ; in bytes (=2* number of
           ; shapes)
+8.w    data        ; surface value for shape 0
+10.w   data        ; surface value for shape 1
(repeats for all defined shapes)
```

Map Data:

```
+0.1    "MAPD"      ;Chunk identifier
+4.1    length of data ;bytes
+8....   data stored one row at a time
```

Each map position has a word value assigned to it which gives the block number at that square in the map. 0=no block in map position, 1 is therefore the first block number available for use.

## 1.26 les\_tooltypesupport

### Tooltype Support =====

The Map Editor supports a whole list of tooltypes to allow you to customise it to your own needs. These tooltypes can be edited directly by bringing up the information requester from WorkBench or can be changed by using the save configuration option in the configuration windows.

### ToolType list =====

MODEID - This is the viewmode id of the screen you want the editor to open on. It is recommended that you do not edit this tooltype directly - using the save configuration option from inside the program will automatically set this tooltype.

SWIDTH - This is the width of the screen to open. It must be at least 320 pixels and must be a multiple of 320 pixels wide.

SHEIGHT - This is the height of the screen to open. It must be at least 200 pixels tall.

MAPPATH  
PALPATH  
SHAPEPATH - These are the default paths for loading and saving. Respectively, they are for map loading and saving, palette loading and shape loading.

BUFFERSIZE - This is the size of the buffer that the program uses when it is moving shapes around the screen under the mouse pointer. A reasonable size for the buffer is 200k but be warned that if you are using a group with loads of large, colourful blocks then the buffer may need to be bigger.

ICONIFYTYPE - The program has an iconify feature that allows you to hide it away (e.g. close screen and free up loads of memory). When the program is iconified it uses up very little processor time and only memory associated with the actual program and the shapes loaded. This tooltype takes values 0 to 1 and lets you select the type of iconify that occurs. A value of 1 means that the program should create a menu item on WorkBench's Tools menu. Selecting this option uniconifies the program. A value of 0 means that the program should create a window on WorkBench. Closing this window or using the zoom gadget activates the program.

ICONIFYEND  
ICONIFYX

ICONIFYY - If you selected a window iconify then these tooltypes let you configure the iconify window. The ICONIFYEND tooltype lets you decide whether the iconify is ended by selecting the zoom gadget on the window or by pressing the right mouse button whilst the window is active.

APPICON - This says whether or not you want the program to automatically create an appicon onto which you can drop RIMP map files for fast loading. This tooltype has no value, if you do not want an appicon then just put brackets around it (" (APPICON) ").

QUIETREMAP - This allows you to select whether or not the shape remap option shows you a progress bar whilst remapping. Bracketing the tooltype out enables the progress bar.

CREATEICONS - This allows you to select whether or not icon files should be created for any RIMP files saved out. When icon files are created the default tool of the file is automatically set to the map editor allowing you to double click the file to load the map editor.

DOMESH - This tooltype allows you to switch on or off the mesh that covers areas of the current screen that go outside of the current map. You will see a mesh if this option is enabled and your map is smaller than the current screen.

AUTOREMAP - This option has 3 values: Yes, No and Ask. It applies to shape loading and lets you decide whether or not shapes should be remapped straight after they are loaded. Note: you can only remap if you have a palette already loaded. Set this tooltype to ask too bring up a requester asking you to decide whether or not to remap.

SHAPESCROLL - Select whether the shape scroller is on or off. Selecting this will mean that shape are displayed at the top of the view area for easy selection.

SHAPEBUTTONS - This is a binary number that holds the last saved status of the gadgets on the GetShapes window. You should not edit these in the tooltype. Select your preferred mode when doing Load Shapes and then select Save configuration.

SAVEBUTTONS - This is like SHAPEBUTTONS except that it applies to the statuses of the gadgets in the Save Custom Map window.

---

## 1.27 les\_generatesource

Generating Source  
=====

Selecting the Generate Source option is mostly of help to Blitz Basic 2 users, since it generates some ASCII text that will display your map on a given bitmap, once loaded into Blitz Basic.

Firstly, upon selecting this option, you are asked whether you are sure you want to generate the source code. If you do, then select Okay, otherwise click on cancel.

You are now faced with a screen containing several Cycle gadgets and text entry gadgets. I will list them below, and explain them further below:

Shortcut	Gadget	Use
f	file format	Select style in which map will be saved (RIMP, or RAW). See the sections on saving custom/raw maps for more details on the merits of either style.
s	raw size	Byte or Word. Sets the size of the data written out. (only used when map being saved is in raw format).
n	raw null	sets the value to use as null when saving as raw. Valid values are 00, or -1. Again, refer to saving raw map section for merits of either value.
l	Blitz2 Mode	Sets the mode the map will run in under Blitz 2. Options are AMIGA or BLITZ.
D	Data mode	sets how the data values will be written in the file. Options are VARIABLES or CONSTANTS.
o	shape offset	sets the offset that the first shape will be loaded from.
p	name prefix	sets a prefix for the name of the map you are saving.

You also have the option to Cancel from here, otherwise just click on Create and via the file-requester that pops up, enter a place to save your source to.

## 1.28 Arexx Support

## Arexx Support

-----

This program has an arexx port, named MapEditor. If you have several versions of running at once, though, they'll have port names MapEditor0, MapEditor1 etc for all versions. By sending messages to these ports functions can be performed and values returned.

To send a message to an arexx port from the cli you use the rx command (found in sys:rexxc/). This command accepts a command line like:

```
rx 'address "MapEditor" QUIT'
```

		Arexx port    Function
		to send to    to send

If the function returns a value, rx will print this value to the cli before exiting.

The Map Editor supports a fair number of arexx commands, all of which make it perform certain operations. The commands are (case insignificant):

```
QUIT           - exit the program and free up all memory allocated
BEQUIET        - reduce number of requesters to minimum (not fully
                  implemented)
BENOISY        - show all requesters (not fully implemented)
ICONIFY        - iconify the Map Editor (will iconify to the currently
                  defined mode, e.g. Tools menu or window)
REDRAWMAP      - Causes the map screen to be redrawn. You can
                  optionally provide a set of coordinates to redraw at,

                  e.g. REDRAWMAP 10,0   will redraw at x=10,y=0
ACTIVATE       - pops Map Editor screen to front and activates its
                  main window
ABOUT         - bring up the information requester
VERSION        - return a string holding the Map Editor version number
LOADMAP        - load a map into the editor - the editor will
                  automatically sense what format the map is in
GRABMAP        - grab a map from memory (see  Arexx Loader/Saver
                  for more information)
LOADSHAPE      - specify a filename to load shapes from, can be an IFF
                  or Blitz2 shape file. You can also optionally specify
                  an offset to load shapes at.
LOADPALETTE    - specify a palette file to load from
RETURNMAP      - return a data block describing the current map
                  (see  Arexx Loader/Saver  for more information)
KILLME         - Tells the Map Editor to send a message 'EXIT' to your
                  port before it exits
REQUEST        - Bring up a requester on the Map Editor's screen. The
                  title will be the name of your message port, you
                  supply the body text (the actual requester contents)
                  and optionally supply the buttons to display.
```

```
e.g. rx 'address "MapEditor" REQUEST "Hello|world" _Okay|_Cancel'
```

REQUEST returns the value of the button selected in the requester to you.

LOADREQUEST- open up a load requester on the Map Editor's screen.

You supply a title for the requester (which should have the word Load in it) and a default filename (e.g. Work:Temp/Testing). The return value will be the filename entered by the user or NULL. This routine makes sure that the file exists before returning.

SAVEREQUEST- same as LOADREQUEST except that it should have Save in the title. If the selected file already exists, this routine asks the user to confirm that they want to write over it.

STRINGREQUEST- gets a string from the user, returns "" if the user cancels the requester.

Note that this command list is being extended in functionality and usefulness in the registered version of this program. The additional commands are a necessity if you intend to use this program extensively via arexx.

#### Command Keys

-----

This program allows you to place Arexx/Dos commands on keyboard shortcuts. The commands are entered using the Arexx Commands... menu item on the Operations menu. From this menu item you can edit/load/save the different definitions. If you decide to edit, a window will open with ten string gadgets - enter your commands into these.

These command keys allow you to activate your own custom loaders and savers from the F1-10 keys on your keyboard. For example, if your loader/saver had an arexx port named 'GameMap' you could have the following functions set up:

```
F1: sys:rexxc/rx 'address "GameMap" LOAD'
F2: sys:rexxc/rx 'address "GameMap" SAVE'
F3: sys:rexxc/rx 'address "GameMap" ABOUT'
F4: sys:rexxc/rx 'address "GameMap" QUIT'
```

You could then select the relevant Fkey to activate your loader/saver.

Note about arexx commands: when the Map Editor first runs it looks for a file 'mapeditor\_arexx' in its program directory. If it finds this file, it loads it in as its default command set.

## 1.29 Arexx Loader/Saver

### Arexx Loader/Saver

-----

Using arexx and the MapEditor arexx command set it is possible to write custom loaders and savers for the Map Editor. These loaders would be activated by pressing a command key (see Arexx Support ) and then

work by calling the Map Editors functions.

As a rule, a loader/saver must support at least the following four functions:

```
ABOUT - open an information requester
LOAD   - load a map
SAVE   - save a map
QUIT   - quit the loader/saver
```

The loader/saver must decide on what function is being called and act appropriately. It is up to the writer of the loader/saver what other features it has.

An example loader/saver, written in pseudo-code, would look something like:

```
Open arexx port
waitloop:
  while function<>QUIT
    Wait for message at port
    If function=LOAD
      do_load
      message Map Editor, function GRABMAP
    endif
    If function=SAVE
      message Map Editor, function RETURNMAP
      do_save
    endif
    If function=ABOUT
      do_requester
    endif
  Wend
Close arexx port
End
```

It is up to you exactly what features your program has. For example it could have an AppIcon - when the user drops a file on it the file is automatically loaded into the Map Editor. Your program doesn't even have to load or save at all - it can be a check program for your maps, that generates specific details in them, or integrity checks them (for games that require a strict map layout). The possibilities are endless.

Loading/Saving a map into Map Editor

Both loading and saving use the following structure for passing information between programs when loading (using GRABMAP) and Saving (using RETURNMAP):

```
Newtype.mapeditormsg
width.w      ; Width of map
height.w     ; Height of map
bwidth.w     ; Block width
bheight.w    ; Block height
datasize.b   ; Datasize (0=byte, 1=word)
mapnull.b    ; Null indicator (0=00, 1=-1)
```

---



```

    address.1    ; pointer to map data
End Newtype

```

The null indicator byte identifies what value in the map data indicates a null position. If 0, then 0 in the map data means a null position and shapes count from 1 in the map. If 1, then -1 means a null position and shape usage starts from 0. This value must be either 0 or 1!

When getting the address of this structure from the Map Editor (using RETURNMAP) the datasize will always be 1, and the mapnull value will always be 0.

## 1.30 les\_credits

### Credits

-----

Design/Code: Stephen McNamara

Advice: Steven Matty  
Steven Green

Stuart Gray  
Martin Kift  
Jurgen Valks  
and others???

Thanx to: Nico Franco for the ReqTools library  
Neil O'Rourke for ReqTools support in  
Blitz  
Acid Software for Blitz Basic 2  
Everyone who mailed me about the  
original RI Map Editor release (yes I  
know it was bugged..... ;-) )  
Blitz-list mailers everywhere

Documentation: Stephen McNamara  
Steven Green  
Steven Innell

AmigaGuide conversion: Jurgen Valks

## 1.31 les\_index

### LES MAP EDITOR V2 Function Index

=====

Screen Layout	Reflect Table
Editor Menustrip	Shape Names
Drawing Control Area	Map Shape Adjust
Information Area	Loading Shapes
View Area	Loading Maps
Control Area	Saving Maps

---

Shape Scroller	Saving Custom Maps
Group Selector	Save Raw Map
Range Selector	Custom Map File Format
Configuration	Tooltype Support
Surface Data	Generate Source
Shape Selector	ARexx Support
	Introduction
	Shareware
	Docs
	Credits

---