vlab Tools
Reference Manual

Last updated: November 30, 2021

vlab was developed in the labs of Przemyslaw Prusinkiewicz at the University of Regina and the University of Calgary, Canada

Contents

1 Introduction

This manual describes the *tools* that exist within *vlab*. Each tool is used to interactively manipulate a specific type of data file. The following tools are currently available:

- a panel manager - to control parameters in a text file
- a colormap editor - to edit the 256 color triplets in a colormap file (`.map`)
- a material editor - to edit the material properties of colors in a material file (`.mat`)
- two surface editors - to define bicubic surfaces consisting of Bézier patches: *bezieredit* provides functionality to join patches with various levels of continuity, while *stedit* can display and edit a texture on the surface patch. Both use the same file format to describe the surface.
- a contour editor - to define B-spline contours in 2D space.
- a function editor - to define a function as a spline curve in 2D space.
- a utility to create a set of contours or functions in a single file (a *gallery*), and call the appropriate editor.
- a timeline editor - to define a set of functions over time

All of these tools produce files that can be used by the *vlab* modeling programs *cpfg* and *lpfg*.

See the *Vlab Framework* manual for a description of the components of *vlab*, including the tools and how they are used within the framework.

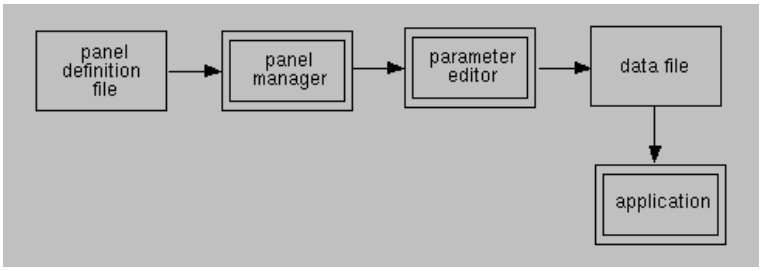


Figure 1: Communication flow from the Panel Manager to an application.

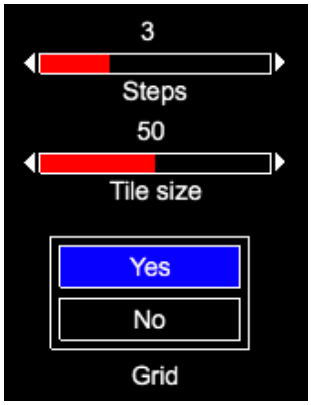
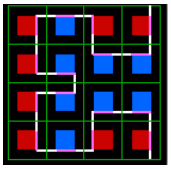


Figure 2: Example of a panel with two sliders, two buttons, and a label defining the button group.

2 Panel manager (*panel*)



See object: HilbertPanels

The panel manager is used to control parameters in a text file. This makes it possible to graphically manipulate any application that reads its parameters from a file. The panel manager inputs a predefined panel specification file, and outputs panel change messages to *stdout* where they can be piped to an editor that will update the data file (Section 2.3.2). The communication flow is shown in Figure 1.

The command line for the panel manager is:

`panel` *panelfile*

2.1 Using a panel

2.1.1 Manipulating sliders and buttons

The parameter values are manipulated using sliders and buttons (Figure 2).

Sliders. To manipulating a slider, left-click and hold within the slider outline, and move the mouse left (to decrease) or right (to increase) the value. Alternatively, click on the arrow at either end of the slider to decrease (left arrow) or increase (right arrow) the value by 1.

Buttons. Left-click on a button to activate it. There are several types of buttons:

- A *monostable button* will change colour when pressed, and return to the original colour when released.

- A *toggle button* will switch between on and off states (and colours) each time it is clicked.
- A *button group* is a set of mutually exclusive toggle buttons (as in the example in Figure 2). When any button is clicked, it is switched on, and all other buttons in the group are switched off.

2.1.2 The panel menu

The panel menu is activated by right-clicking anywhere within the panel. The menu items are:

Menu item	Description
Pages	Select a specific panel page, if there are multiple pages. Otherwise this item will not be present.
Reset	Reread the panel definition file to restore all controls to their default values, and output all the values (Section 2.3).
Broadcast	Output all current values (Section 2.3).
Edit	Open the panel editor window and switches to edit mode (Section 2.2).
Save as default	Set the default value of each control to its current value.
Refresh mode	Set the mode for outputting slider messages: Triggered = when the mouse is released Continuous = as a slider is moved (every slider value)
Exit	Output the null character (end-of-file) to terminate the file editor (Section 2.3.2), and close the panel.

2.2 Panel edit mode

The interactive editor is invoked by selecting Edit on the pop-up menu. It provides:

- a separate Panel Editor window that interacts with the panel, and
- a grid over the panel to aid with alignment.

The Panel Editor window is divided into three main sections (Figure 3): the upper section contains the items that can be added to a panel; the middle section is for aligning items on the panel; and the lower section defines characteristics of the panel itself.

There are also new options on the pop-up menu.

Main item	Sub-item	Description
Edit options	Edit item	Edit the selected item on the panel by envoking its editor (Sections 2.2.2 to 2.2.4). This can also be done by double clicking on the item.
Edit options	Clone item	Copy and paste the selected item on the same panel page.
Edit options	Copy item Paste item	Copy and paste the selected item in two steps. This allows the copied item to be pasted on another page on the panel.
Edit options	Delete item	Delete the selected item. There is no warning, and no un-do, for this function.
Edit options	Edit page	Edit the page characteristics.
Edit options	Delete page	Delete the page.
Edit options	Snap to grid	Align the bottom left corner of the selected item with the nearest grid intersection. Buttons within a group will not snap to the grid, but the button group itself will.
Execute		Return to normal panel functionality.
File	Load	Open another panel file.

Main item	Sub-item	Description
File	Reread	Reread the current panel from the panel file, overwriting any changes made since the panel was last saved.
File	Save	Save all changes to the panel file. This can also be done using the Save button at the bottom of the Panel Editor window.
File	Save as...	Save all changes to a new file. This new file will be used in all subsequent Save commands.

See Section 2.2.1 for more information on selecting and editing items, and Section 2.4 for details of the panel specification file.

2.2.1 Editing panel items

It is possible to move, edit, copy, and delete existing items on the panel, either graphically or using the Edit options on the pop-up menu.

To select items Left click on an item to select it: a dashed yellow rectangle indicates the selected item. To select several items at once, click on the first item, then hold the Shift key down and click on each of the remaining items. This will extend the dashed yellow rectangle around all of the items. Note that clicking on the first and last item does NOT select the items in between.

To move items Once selected, items can be moved by left-clicking anywhere within the yellow rectangle and dragging.

To align items The Snap to grid menu item aligns a selected item with the nearest grid intersection. When several items are selected, the middle section of the Panel Editor is also enabled (see Figure 3). Use these functions to align the selected group of items horizontally or vertically. Note that the Flip items vertically function is always available. It flips ALL the items on the panel page, regardless of the items selected.

To add a new slider, button, or label Click on the appropriate button in the Panel Editor window. The new item will appear in the middle of the panel, and its Editor window will pop up (see the slider example in Figure 4). In addition to the dashed yellow rectangle indicating that the new item is selected, there will also be a solid yellow rectangle around the item indicating that this is the item whose parameters are seen in the associated Editor window. Note that since the new item is positioned in the middle of the panel, other items may need adjusting to make room for it.

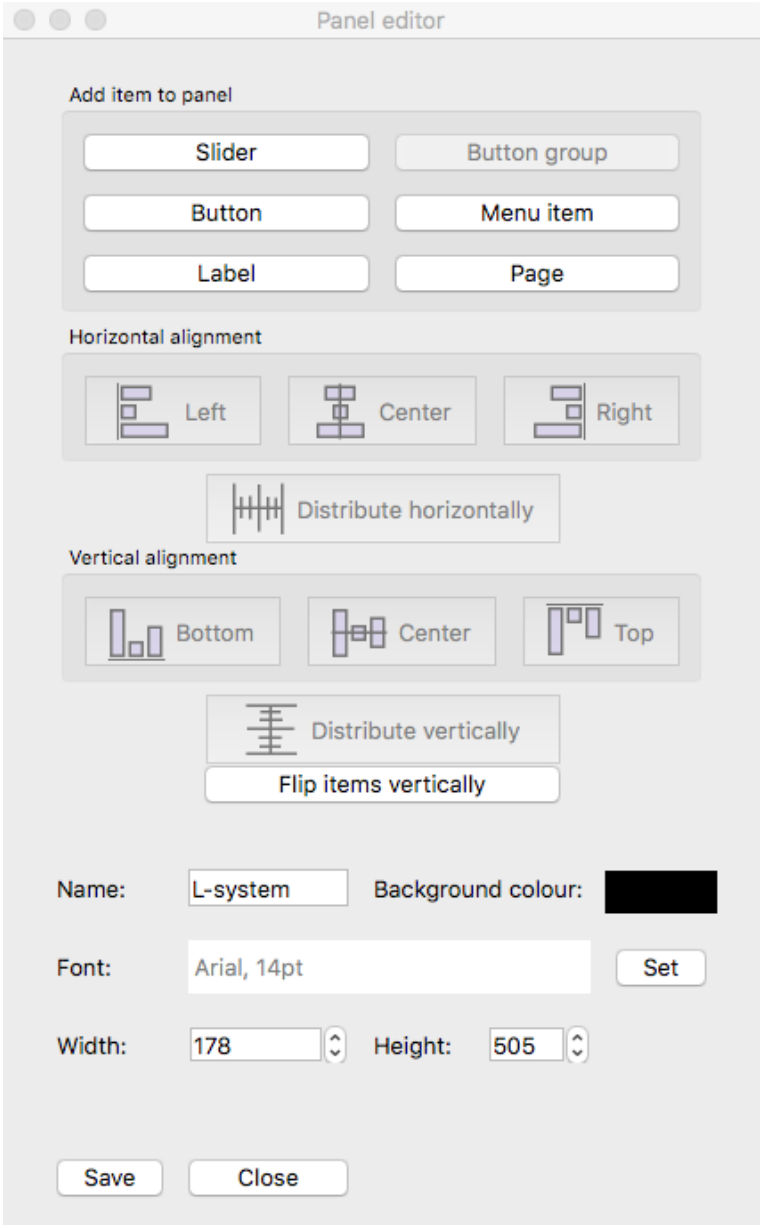


Figure 3: The Panel Editor.

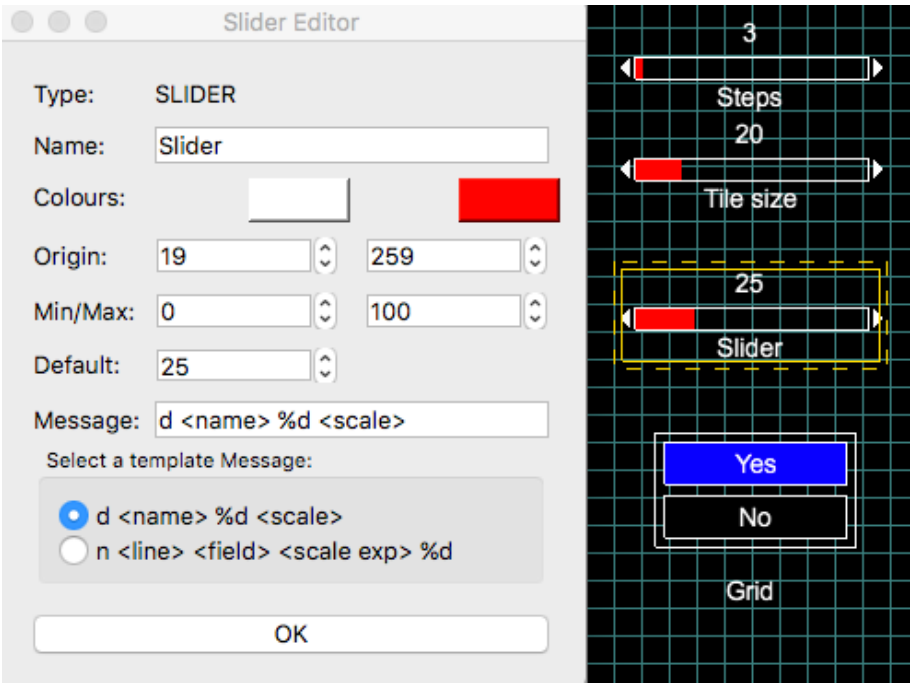


Figure 4: The Slider Editor and the new slider on the panel. The buttons have been moved to make room for the new slider.

2.2.2 Sliders

Sliders are used for data parameters that can range between a set of numbers. The fields on the Slider Editor (Figure 4) are:

Fields	Description	Default
Name	The label below the slider	Slider
Colours	The outline and name colour (left) and the slide colour (right)	White / Red
Origin	The position of the slider with respect to the top left corner of the panel. Normally the slider is positioned graphically - manual adjustments are only needed for very precise positioning.	
Min/Max	The minimum and maximum values for the slider.	0 - 10
Default	The default value for the slider, to be used when the panel is initially invoked and when Reset is selected from the menu. It will also be set to this value when the OK button is clicked.	5
Message	The message that is output when this slider is activated. The template messages follow the standards used by the data file editor *awkped* (Section 2.3.2). Selecting a template populates this field, which can then be edited to replace the <> parameters as required.	d <name> %d <scale>

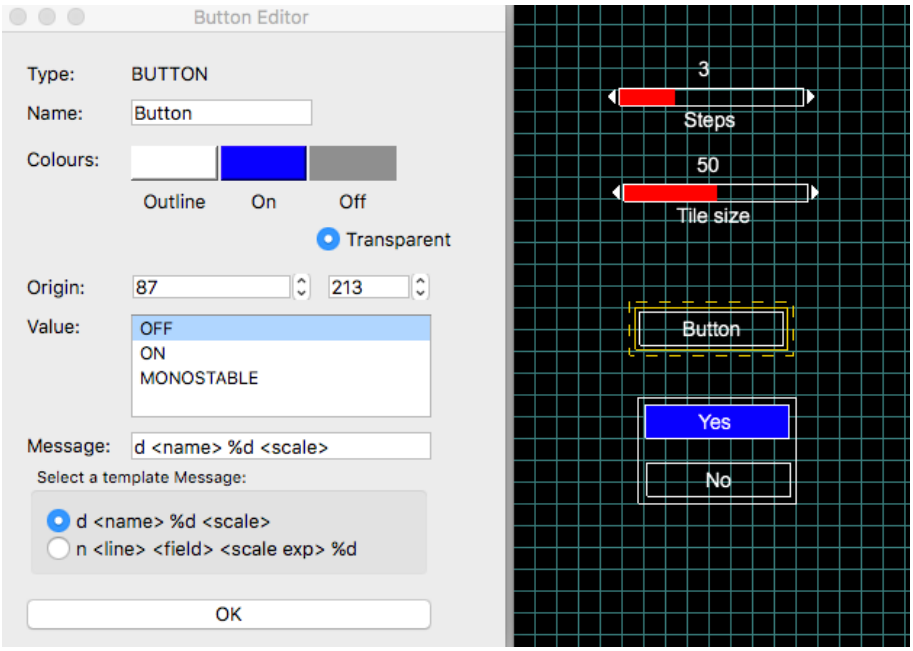


Figure 5: The Button Editor and a new button on the panel. The Yes/No buttons have been moved to make room for the new button.

2.2.3 Buttons

Buttons can be used for parameters with two values (On and Off), or a single value (Monostable). The fields on the Button Editor (Figure 5) are:

Fields	Description	Default
Name	The label within the button.	Button
Colours	The Outline, On, and Off colours of the button.	white / blue / black
Transparent	When this radio button is set, there is no Off colour: the button is transparent. (The color in the editor is set to grey.)	
Origin	The position of the button with respect to the top left corner of the panel. Normally items are positioned graphically - manual adjustments are only needed for very precise positioning.	
Value	The default setting for the button. A MONOSTABLE button is always off: it outputs the same value each time it is activated.	OFF
Message	The message that is output when the button is activated. The template messages follow the standards used by the data file editor *awkped* (Section 2.3.2). Selecting a template populates this field, which can then be edited as required.	d <name> %d <scale>

If there are multiple buttons on the panel with the same label, the Panel Manager will automatically add `%Copy1`, `%Copy2`, etc to the end of the name (Figure 6). The percent sign and all text following it are not displayed, but make the Name unique. The text may be edited.

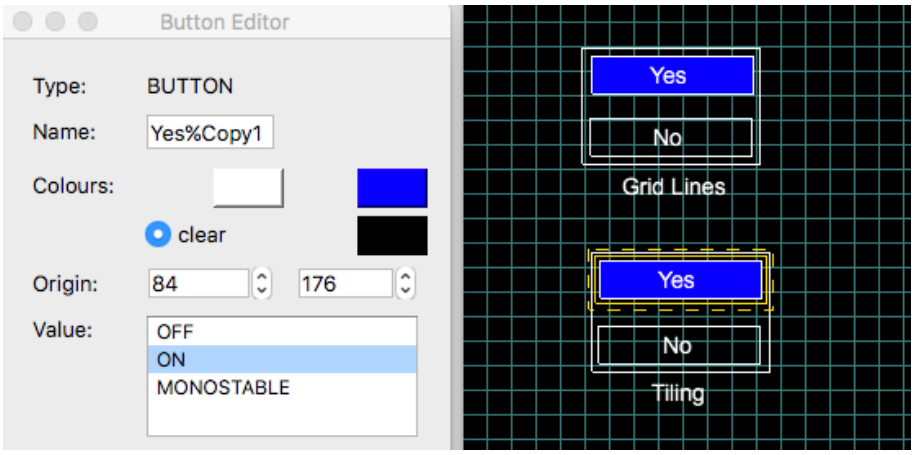


Figure 6: An example of non-unique button labels on a panel. The second Yes button has %Copy1 appended to its Name field in the Button Editor. This is not displayed on the button, but is used to distinguish this button from the one above.

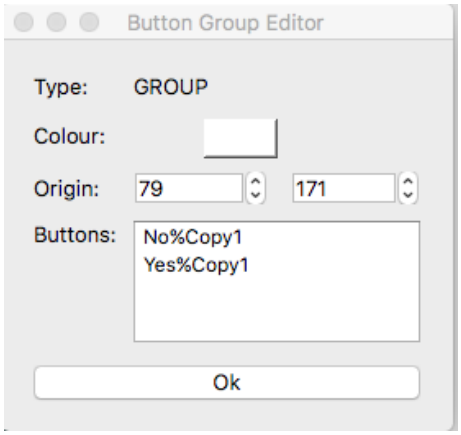


Figure 7: The Button Group Editor for the second set of buttons in Figure 6.

2.2.4 Button groups

To create a set of mutually exclusive toggle buttons (where only one button can be "on"), select a group of adjacent buttons: click on the first button, then hold down the Shift key and click on each of the other buttons in the group. Note that clicking on the first and last buttons will NOT include the buttons in between.

Selecting multiple buttons will enable the Button group function on the Panel Editor. Click it to open the Button Group Editor (Figure 7). To use the currently selected buttons, simply click Ok. To edit the group later, double click on the group to open the editor. The fields on the Button Group Editor are:

Parameter	Description	Default
Colour	The outline colour for the group	white
Origin	The position of the group with respect to the top left corner of the panel. Normally items are positioned graphically - manual adjustments are only needed for very precise positioning.	
Buttons	The names of the buttons that are part of the group.	

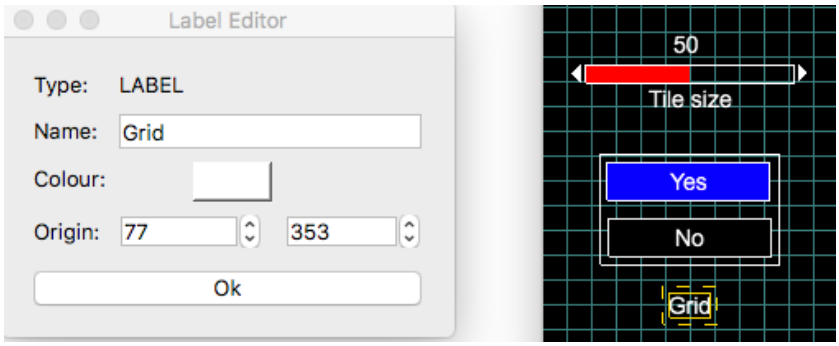


Figure 8: The Label Editor for a label defining a button group.

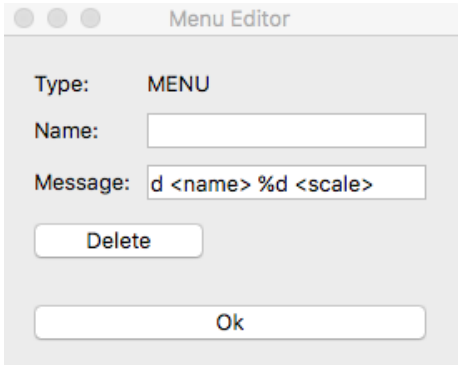


Figure 9: The Menu Editor used to add, update, and delete items for the panel's menu.

2.2.5 Labels

Labels can be used to add text anywhere on the panel. They are often used to label a Button Group. The fields on the Label Editor (Figure 8) are:

Field	Description	Default
Name	The text of the label	Label
Colour	The colour of the text	white
Origin	The position of the label with respect to the top left corner of the panel. Normally items are positioned graphically - manual adjustments are only needed for very precise positioning.	

2.2.6 Menu items

New menu items can be added to the panel menu under a Custom messages option at the top of the menu. (This option is only displayed if new menu items have been added.)

Items are added using the Menu item function on the Panel Editor. To edit an existing custom menu item, select it from the pop-up menu (Figure 10). There are only two fields on the editor, as well as a Delete button to remove an existing menu item:

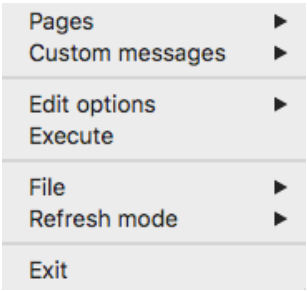


Figure 10: If the panel has multiple pages, they are listed at the top of the menu under Pages. Additional menu items are listed under Custom messages.

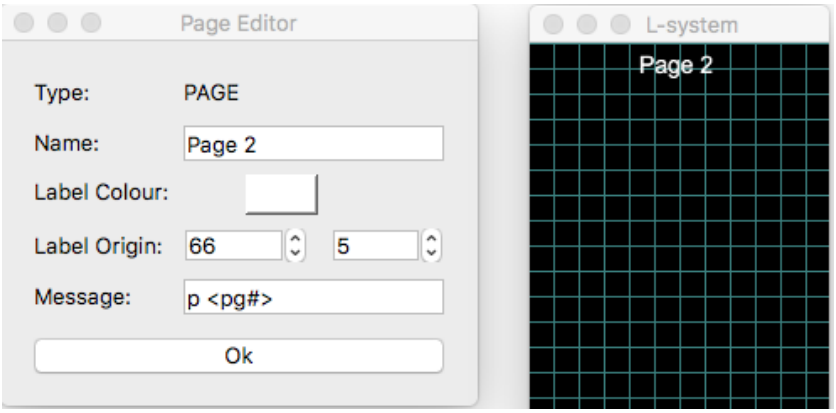


Figure 11: The default page and Page Editor when a second page is added to the panel.

Menu item	Description	Default
Name	The menu item name	
Message	The message that is output when the menu item is activated. The default message follows the standard used by the data file editor *awkped* (Section 2.3.2). It can be edited as required.	d <name> %d <scale>
Delete	Delete this menu item. If it is the last user-defined menu item, Custom messages will be removed from the pop-up menu as well.	

2.2.7 Pages

A panel may consist of several pages, only one of which is displayed at any given time. If there are multiple pages, the list of pages will appear at the top of the pop-up menu (Figure 10).

To add a new page Click the Page function on the Panel Editor to open the Page Editor (Figure 11). The fields in the editor are:

Menu item	Description	Default
Name	The label at the top of the page.	Page n
Label Colour	The colour of the page label	
Label Origin	The location of the page label with respect to the upper left corner of the page	
Message	The message that is output when this page is selected.	p <pg#>

To edit a page Use the Pages menu item to navigate to the desired page. Either double click on the page label, or select Edit page from the pop-up menu under Edit options. This will display the Page Editor.

To delete a page Select Delete page from the pop-up menu under Edit options. This will return to the first page of the panel. Note there is no warning: the page is immediately deleted, and there is no un-do.

2.2.8 Panel characteristics

The general characteristics of the panel are defined in the bottom section of the Panel Editor. The fields are:

Fields	Description
Name	The name of the panel window.
Background colour	The colour of the panel background.
Font	The font to be used for labels on the panel. To change the font, click the `Set` button.
Width Height	The width and height of the panel. These can also be adjusted graphically by clicking and dragging the edges of the window.

2.3 Panel output

The message associated with an item is output when the item is activated on the panel, or when Broadcast is selected from the pop-up menu (Section 2.1.2). The message is sent to *stdout*, where it can be accessed by a datafile editor (Section 2.3.2).

2.3.1 Message format

A message is output verbatim from the Message field associated with the item, with the exception of %d, which is used as follows:

- For sliders, %d is replaced by the displayed value.
- For buttons, the following %d values are output depending on the button type:
  - *Toggle button* - The %d is replaced with 1 when button is on, and 0 when off.
  - *Monostable button* - Two messages are output: in the first message %d is replaced with 1; in the second message with 0.
  - *Button group* - The message associated with the button that has been turned on is output, replacing %d with 1. There is no message for the button that was turned off.
- For all other items (pages and menu items), there is no value to associate with the %d parameter, so it is always replaced by zero if used.

2.3.2 Vlab datafile editor (*awkped*)

Vlab includes an *awk* program used to edit data files based on specific message formats. The editor, *awkped*, takes commands from *stdin* and uses them to edit the specified data file. Therefore, the output from a tool such as the panel manager, or the timeline editor (Section 10), can be piped into *awkped* which will perform the actual edits. For example:

`panel` *panelfile* `|` `awkped` *datafile*
`timeline` *functions*`.tset |` `awkped` *datafile*

Awkped can edit data files that contain:

- `#define` statements; or
- lines consisting of a parameter/field name followed by a colon, and value(s).

Define statements The command to edit `#define` statements begins with the letter `d`, and has the format:

`d` *fieldname value scale*

The editor will search for a line with the format

`#define` *fieldname n*

and will replace *n* with *value* divided by *scale*.

Field name and values The command to edit a value associated with a field name requires the exact line number, as well as the order number of the field to be updated. The editor expects lines that consist of a field name followed by a colon, and field values:

field name: value(s)

The command to the editor begins with the letter `n`, and has the format:

`n` *linenum fieldnum scaleexp value*

The editor replaces the $fieldnum^{th}$ value on line *linenum*, beginning after the field name (i.e. after the colon). The field content is replaced by $value \times 10^{scaleexp}$. For example, if the first two lines of the data file are:

```
color increment: 1
initial line width: 7 pixels
```

the first value (*fieldnum* = 1) in the second line is 7. Therefore, the `initial line width` can be changed to 5 pixels with the command:

```
n 2 1 0 5
```

This will replace the current value of the field (7) with $5 \times 10^0 = 5$.

Scaling The scale factor is useful for sliders, which can only have integer values. For example, if a field can have a value between 0.1 and 10.0, a slider can be created with a range of 1-100, with one of the following messages (depending on the data file format):

`d` *fieldname value* `10`
`n` *line field* `-1` *value*

In the first case (`d`), the current value is replaced with *value*/10. In the second case (`n`), it is replaced with $value \times 10^{-1} = value \times 0.1$.

2.4 The panel file

The panel is read and saved from/to a text file using the File commands on the popup menu in edit mode (Section 2.2). The *vlab* convention for panel files is `panel.`x, where x is the same extension as the data file the panel is editing.

The specification file itself contains a header, followed by groups of lines defining each of the components of the panel, beginning with the page and then each of the components on the page. The majority of the parameters are fields in the panel's Editor windows. Each line of the file consists of a keyword, followed by a colon, and the associated value(s) (see the panel header example below).

2.4.1 Panel header

Parameter	Description	Panel Editor field
`target`	The name of the data file to be edited with this panel. This is for information only.	
`panel name`	The name of the panel, as seen in the panel window's title bar.	Name
`background`	The RGB components making up the background colour of the panel, separated by commas.	Background colour
`size`	The width and height of the panel, specified as two numbers separated by a space.	Width Height
`font`	The font to be used for labels throughout the panel.	Font

An example of a panel header is:

```
target: View file
panel name: Viewing
background: 0,0,0
size: 178 505
font: Arial,14,-1,5,50,0,0,0,0,0
```

2.4.2 Page

Parameter	Description	Page Editor field
`type`	`PAGE`	Type
`name`	The page label.	Name
`color`	The RGB components of the label colour, separated by commas	Label Colour
`origin`	The x and y coordinates of the label with respect to the upper left corner of the panel, specified as two numbers separated by a space.	Label Origin
`message`	The message to be sent when this page is activated.	Message

2.4.3 Slider

Parameter	Description	Slider Editor field
`type`	`SLIDER`	Type
`name`	The label below the slider.	Name
`colors`	Two sets of RGB colours: one for the outline and name, and the second for the slider. The RGB components are separated by commas, with a space between the two sets of colours.	Colours
`origin`	The x and y coordinates of the slider with respect to the upper left corner of the panel, specified as two numbers separated by a space.	Origin
`min/max`	The minimum and maximum values of the slider.	Min/Max
`value`	The value of the slider when the panel opens.	Default
`message`	The message to be sent when the slider is activated.	Message

2.4.4 Button

Parameter	Description	Button Editor field
`type`	`BUTTON`	Type
`name`	The label within the button.	Name
`tricolor`	Three sets of RGB colours: the button outline, the "on" colour, and the "off" colour. The RGB components are separated by commas, with a space between each set.	Colours
`origin`	The x and y coordinates of the button with respect to the upper left corner of the panel, specified as two numbers separated by a space.	Origin
`value`	The setting of the button when the panel is opened: 0 = OFF 1 = ON 2 = MONSTABLE	Value
`message`	The message to be sent when the button is activated.	Message

2.4.5 Button group

Parameter	Description	Button Group Editor field
`type`	`GROUP`	Type
`color`	The RGB components of the group outline colour, separated by commas.	Colour
Button name 1 *Button name 2* ...	The name of each button in the group, one per line. The buttons should be defined before the group itself.	Buttons
`ENDGROUP`	To end the list of buttons.	

2.4.6 Label

Parameter	Description	Label Editor field
`type`	`LABEL`	Type
`name`	The text of the label.	Name
`color`	The RGB components of the label colour, separated by commas.	Colour
`origin`	The x and y coordinates of the label with respect to the upper left corner of the panel, specified as two numbers separated by a space.	Origin

2.4.7 Menu item

Parameter	Description	Menu Editor field
`type`	`MENU`	Type
`name`	The text of the menu item.	Name
`message`	The message to be sent when the menu item is selected.	Message

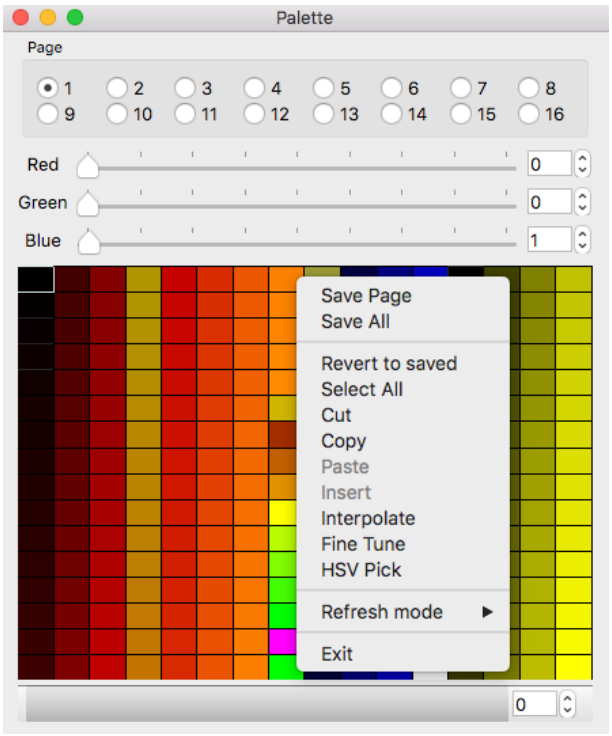


Figure 12: Example of the *palette* window and its pop-up menu. The selected color is outlined in white and its color number is indicated in the bottom right corner. The pop-up menu is displayed by right-clicking within the grid of colors.

3 Colormap editor (*palette*)

The *cpfg* and *lpfg* programs use either a colormap file, or a materials file (Section 4) to specify the colors in a simulation. A colormap is a binary file that defines 256 colors in triplets of three bytes, one for each of the R, G, and B components of the color. Thus the file is exactly 768 bytes.

The colormap file is created and edited with the *palette* program, which is invoked from the command line:

`palette` *filename*`.map`

See object: Mycelis Schematic

This command will display a window of colors such as in Figure 12.

Additional "pages" of colors can be created, with each page saved as a separate 768 byte file. To re-open a set of colormap files, use the `-m`*n* command line argument, where *n* is the page number for a specified file. For example, to open three colormap files as three separate pages within *palette*, the command line would be:

`palette -m1` *filename1*`.map -m2` *filename2*`.map -m3` *filename3*`.map`

3.1 The color grid

The 256 colors in the *palette* grid are numbered column-wise from 0 to 255 (i.e. the first column represents colors numbered 0-15, the second column 16-31, and so on).

To select a color, left-click on its cell. Its current RGB values will be displayed on the sliders above the grid, as well as in the field to the right of each slider. The color can be adjusted by

- manipulating the sliders,
- entering new numbers in the associated fields,
- using the up and down arrows next to a field to increase or decrease its value, or
- selecting the Fine Tune or HSV Pick menu item (see below).

It is also possible to select a range of colors by left-clicking on the first color, and holding the button down while dragging the mouse.

3.2 The *palette* menu

The menu items on the pop-up menu are defined as:

Menu item	Description
Save Page	Save the current page. If the page was not specified on the command line, it will be saved as `default`n`.map`, where n is the page number.
Save All	Save all pages. If any page was not specified on the command line, it will be saved as `default`n`.map`, where n is the page number.
Revert to saved	Reread the file(s) specified on the command line, and update the grid with the values in the file(s).
Select All	Select all the colors on the grid.
Cut	Cut the selected color(s). The remaining colors will be moved up, leaving blank (white) cells at the bottom of the rightmost column.
Copy	Copy the selected color(s).
Paste	Paste the previously cut or copied color(s) at the selected cell. The current color(s) will be overwritten with the cut/copied values.
Insert	Insert the previously cut or copied color(s) at the selected cell. The current color(s) will be moved down; the color(s) at the bottom of the rightmost column will be removed.
Interpolate	Create a gradient of colors across the selected cells, by interpolating between the color in the first cell and the color in the last cell.
Fine Tune	Display a dialog box (Figure 13) for finely adjusting the red, green, and blue components of the selected color(s), as well as the brightness.
HSV Pick	Dispay a dialog box (Figure 14) for manipulating all components of the color.
Refresh mode	Set the mode for refreshing the colormap file: Explicit = only when the Save command is selected. This is the default. Triggered = when a color value is changed, and the mouse is released. Continuous = as a color value is changed (all values of a slider as it is moved with the mouse).
Exit	Close the *palette* application.

Figure 13: The *palette* dialog box for fine-tuning a color, or range of colors.

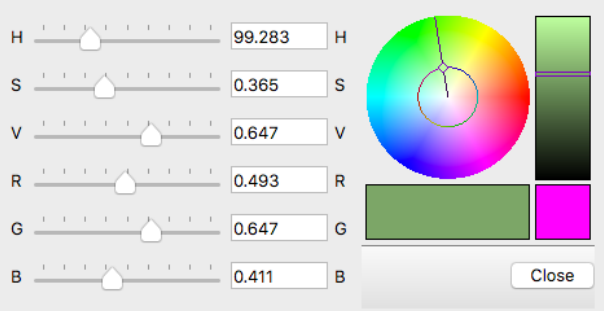


Figure 14: The *palette* dialog box for defining all the components of a color. Use the sliders or update the values for each component, or use the wheel to select a color and the rectangle to the right to select the gradient. The rectangle below the wheel displays the selected color, while the square beside it displays the original color. Return to the original color by clicking on the square.

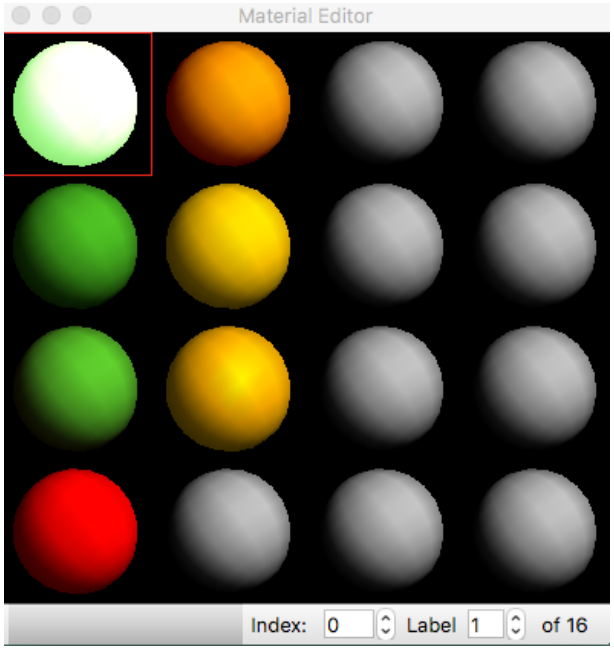


Figure 15: An example of the initial *medit* window with 4 colors defined. The currently selected color is outlined in red.

4 Material editor (*medit*)

The *cpfg* and *lpfg* programs use either a colormap file (Section 3) or a materials file to specify the colors in a simulation. The material editor defines the properties of colors in a materials file: the ambient, diffuse, specular, and emissions attributes, as well as transparency and shininess. It is a binary file that is created and edited by the *medit* program. The command line syntax of *medit* is:

See object: Monopodial Spiral

`medit` *filename*`.mat`

It is possible to open several materials files simultaneously by listing multiple filenames on the command line. Each file is opened in a separate window.

4.1 The *medit* window

When *medit* is invoked a window is displayed containing 16 spheres (Figure 15). This is the default number of colors defined on each page. The page number is specified at the bottom of the window in the Label field. The number of colors displayed on a single page can be changed on the View menu (Section 4.2.4).

The colors are numbered down the columns, beginning with zero. For example, in the 4x4 window the first column of colors are numbered 0-3, the second column 4-7, etc. The selected color is outlined in red, and its color number is displayed in the Index field at the bottom of the window.

A color can be selected by clicking on it with the mouse, or by changing the value in the Index field. Ranges of colors can also be selected with the mouse, by clicking on the first color and dragging the mouse to the last color in the range. There are also options on the Edit menu (Section 4.2.3) for selecting ranges of colors.

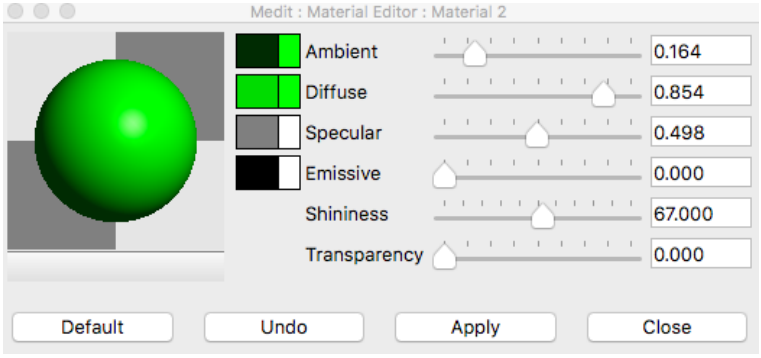


Figure 16: An example of the dialog window displayed by the M-Edit menu item. To change colors, click the rectangle to the left of Ambient, Diffuse, Specular, or Emissive. This will open the Color Pick dialog box (Figure 17) for the attribute.

4.2 The *medit* menus

The main functions of *medit* are found on the pop-up menu which is invoked by right-clicking in the *medit* window. These functions, along with less commonly used functions, are also available on the menu bar.

4.2.1 The pop-up menu

The *medit* pop-up menu contains the following options:

Menu item	Description
Save	Save the current color definitions to the file. This is only required when Refresh mode = Explicit (see below).
Save As...	Save the current color definitions to a new file.
Revert to Saved	Re-read the color definitions from the file, overwriting any changes made within *medit* since the last time the file was saved.
M-Edit	Open a dialog box (Figure 16) to edit the selected color.
Cut	Cut the selected color(s). The remaining colors will be moved up, leaving blank colors at the end of the last page.
Copy	Copy the selected color(s).
Paste	Paste the previously cut or copied color(s) at the selected cell. The current color(s) will be overwritten with the cut/copied values.
Insert	Insert the previously cut or copied color(s) at the selected cell. The current color(s) will be moved down; the color(s) at the end of the last page will be removed.
Interpolate	Create a gradient of colors across the selected cells, by interpolating between the color in the first cell and the color in the last cell.
Set to Default	Set the current color to the default values: grey with Ambient = 0.2. This is the same as clicking the Default button in the dialog box.
Refresh mode	Set the mode for refreshing the materials file: Explicit = only when the Save command is selected. This is the default. Triggered = when a color value is changed, and the mouse is released. Continuous = as a color value is changed (all values of a slider as it is moved with the mouse).

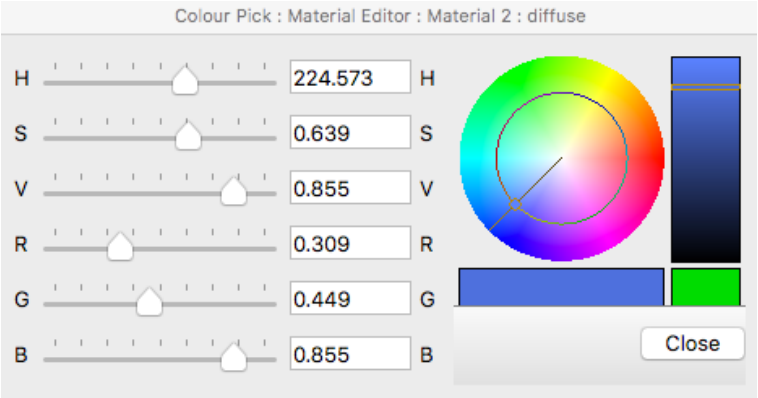


Figure 17: An example of the Color Pick dialog box which is displayed when the rectangle to the left of the Ambient, Diffuse, Specular, or Emissive attribute is selected in the M-Edit dialog box (Figure 16). The original color is shown in the small rectangle above the Close button. Changes made here are immediately reflected in the M-Edit dialog box.

4.2.2 The File menu

The functions available on the File menu are:

Menu item	Description
New	Close the current file (a dialog box will be displayed if changes have been made), and open a new *medit* window with default values for all materials.
New Window	Open a new *medit* window, without closing the old one. The default file name for the new window is noname.mat.
Load...	Close the current file, and open the file selected in the dialog box.
Load New Window...	Open a new *medit* window for the file selected in the dialog box, without closing the old one.
Save Save As... Revert to Saved	See the pop-up menu (Section 4.2.1).

4.2.3 The Edit menu

The functions available on the Edit menu are:

Menu item	Description
M-Edit Copy Paste Insert Interpolate Set to Default	See the pop-up menu (Section 4.2.1).
Select Range...	Select a range of colors by providing the index of the first color and either the number of colors, or the index of the last color.
Select All in Page	Select all the colors displayed in the window.
Select All in Material	Select all 256 colors available in the material file, including colors on pages not currently displayed.

4.2.4 The View menu

The functions available on the View menu are:

Menu item	Description
New Page Prev Page First Page	Move between pages of colors, similar to changing the Label field at the bottom of the *medit* window.
xs - 1x1 sm - 4x4 md - 8x8 lg - 16x16	Change the number of colors displayed on each page.
Enter /Exit Full Screen	Enter and exit full screen mode.

4.3 The materials file

The file produced by the material editor has the extension `.mat`. It is a binary file that contains one or more 15-byte records of the form:

```
struct materialrecord
{
unsigned char id;
unsigned char transparency;
unsigned char ambient[3];
unsigned char diffuse[3];
unsigned char emission[3];
unsigned char specular[3];
unsigned char shininess;
};
```

The parameters are defined as:

Parameter	Description
`id`	The material number
`transparency`	The transparency applied to all material components, where 0 = no transparency, and 255 = full transparency.

`ambient`
`diffuse`
`emission`
`specular`
	The RGB values of the respective material components.

`shininess`	A value in the range [0,128] that defines the shininess of the material.

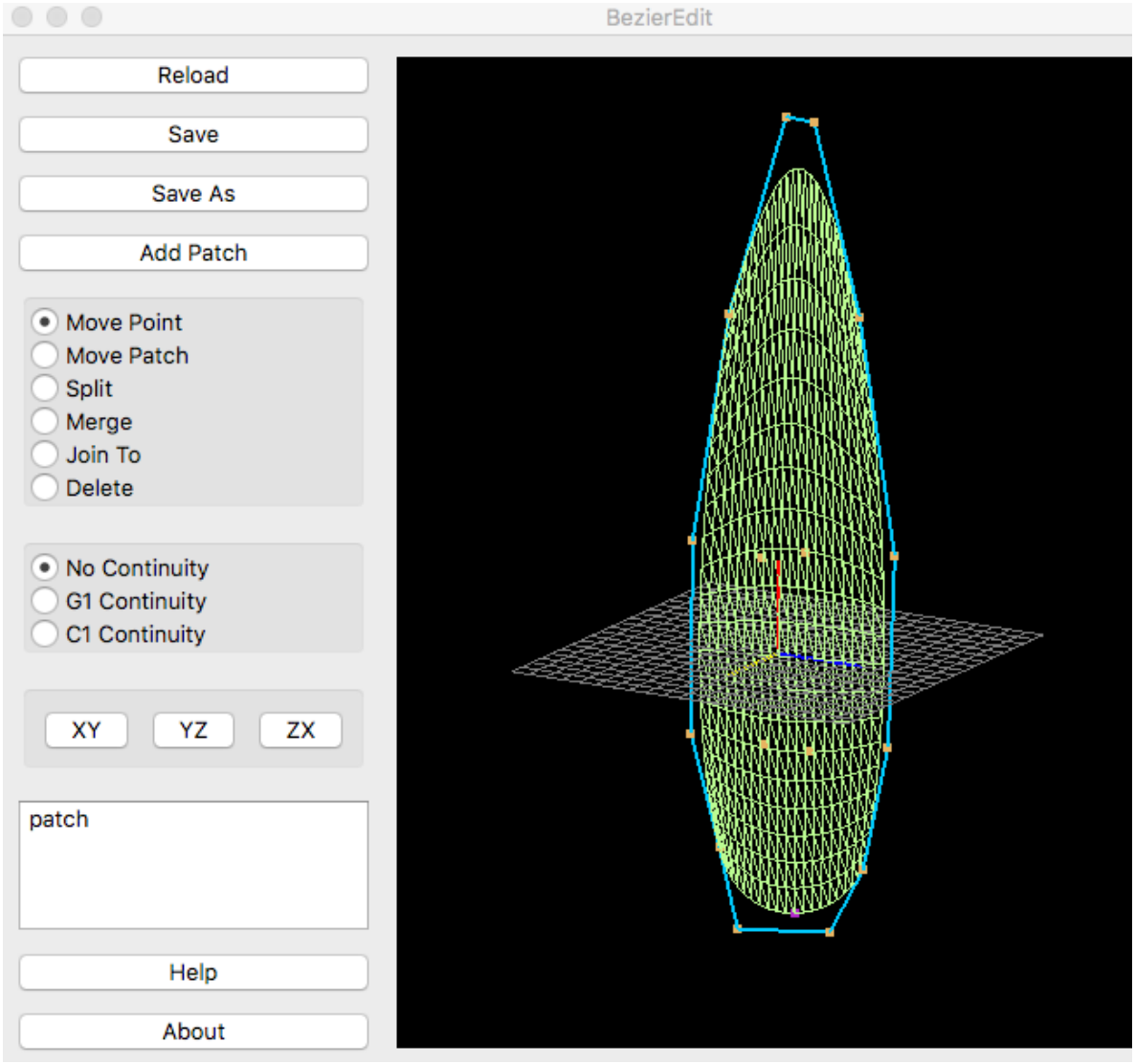
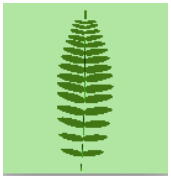


Figure 18: An example of the *bezieredit* window.

5 Surface editor (*bezieredit*)



See object:
LeafPosition

The surface editor defines a bicubic surface consisting of Bézier patches. The patches can exist separately, or can be joined with various levels of continuity. All work is done in a single view which can be rotated around the x and y axis to edit the surface from any angle. Manipulation of the surface is done through the control points of the Bézier patches.

The command line syntax is:

```
bezieredit [filename]
```

When invoked, the *bezieredit* window is displayed with a view of the surface on the right, and a control panel on the left (Figure 18).

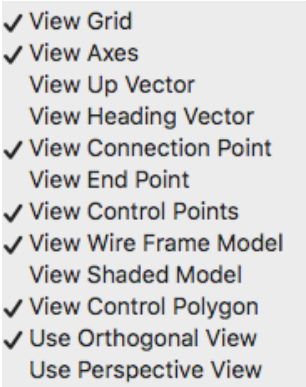


Figure 19: An example of the *bezieredit* pop-up menu.

5.1 Viewing the surface

5.1.1 Manipulating the view

The image can be manipulated with the left mouse button and a control key. To:

- rotate the image (Shift key)
- zoom in and out (Command key)
- pan left/right/up/down (Alt key)

The three buttons labeled XY, YZ and ZW on the control panel will snap the view to the respective planes. These planes correspond to the front, side, and top views.

5.1.2 View elements

Use the pop-up menu (Figure 19), invoked by clicking the right mouse button in the surface view, to show/hide various elements of the view. Note that:

- Up Vector and Heading Vector are attach to the connection point.
- View Wire Frame Model and View Shaded Model are mutually exclusive; selecting one, disables the other.
- Use Orthogonal View and Use Perspective View are also mutually exclusive.

5.1.3 Highlighting patches

Each patch is assigned a name, which is displayed in the box above the Help button on the control panel. When a surface has multiple patches, it is often convenient to highlight a particular patch. Clicking on a patch name will change its color, making it more visible. The highlighting does not affect the underlying data.

5.2 Manipulating the surface

5.2.1 Adding a patch

Use the Add Patch button on the control panel to insert a new patch into the view. The patch will be in the xy-plane, scaled to the current surface. A default patch name will be inserted into the list on the left. To change the name, double click on it. Note that the name must NOT contain any spaces.

5.2.2 Editing modes

The functionality of the left mouse button for manipulating the surface is dependent on the editing mode selected on the control panel, using the set of radio checkboxes below the Add Patch button:

Button	Description
Move Point	Move a single point by clicking and dragging it. The point is moved in the plane parallel to the current view. See Section 5.2.4 for details on manipulating a point that is occluded.
Move Patch	Move a patch by clicking and dragging on one of its control points.
Split	Split patches by selecting a point on an edge or corner shared by the patches. The patches can then be moved apart using Move Patch.
Merge	Merge two patches by selecting a control point on the corner or edge of one patch and then selecting any point on the second patch. Be sure to set the Continuity constraints (Section 5.2.3) before merging. The two patches will be merged based on the point selected on the first patch only: the opposite edge or corner of the second patch will be used. For example, if the first selected point is on the right edge of the patch, the left edge of the second patch will be merged to it. Likewise the upper-left corner of the first patch will always join to the lower-right corner of the second patch. The points that become the shared edge or corner are the interpolated values of the original values of the points from the first and second patches.
Join	Join two patches by selecting a control point on the corner or edge of one patch and then selecting any point on the second patch, similar to Merge. However, instead of interpolating the values of the points along the shared edge or corner, the values of the second patch are used. This will maintain the shape of the second patch when no continuity is enforced (Section 5.2.3). If continuity is enabled, the second patch's interior points may be modified somewhat, but the edges will be preserved.
Delete	Delete a patch by selecting any control point on it.

5.2.3 Continuity constraints

When continuity is enabled, all patch manipulation occurs under the specified continuity constraints. Control points will be modified automatically as needed to ensure that the constraints are retained around the edited points. The continuity constraints are:

Button	Description
No Continuity	No continuity constraints are enforced.
G1 Continuity	Geometric continuity constraints are enforced.
C1 Continuity	Algebraic continuity to the first derivative is enforced.

5.2.4 Sticky points

In some cases a point to be manipulated may be occluded by the image. The solution is to define it as a "sticky point" by double-clicking on it in any view where it is visible. The point will change color to indicate that it is now defined as a sticky point. When the sticky point is occluded by other points, it can be selected by clicking on the area where it is known to be. It can then be dragged as required. The point can be returned to normal by double-clicking on it again, or by selecting a new sticky point.

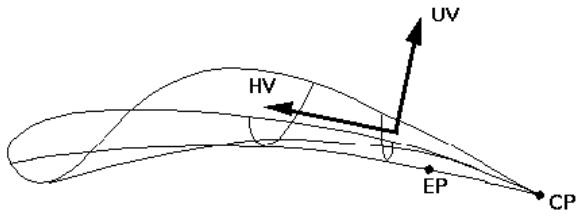


Figure 20: The geometric parameters in the header of a surface file: CP = contact point; EP = end point; HV = heading vector; UV = up vector.

5.3 Updating the surface specifications

The three buttons at the top of the control panel are used to update the surface specifications in the editor, or in the file:

Button	Description
Reload	Re-read the surface specification file, overwriting the current surface.
Save	Save the current characteristics of the surface to the file.
Save As	Save the current characteristics of the surface to a new file. A dialog box will be displayed to enter the new file name.

5.4 The *bezieredit* specification file

The file output from *bezieredit* has the following format, where x, y and z are real values, i and j are integers, and the remaining variables are text strings.

The header section of the file contains parameters that apply to the surface as a whole. See Figure 20 for a visual representation of the geometric parameters.

Parameter	Description
x_{min} x_{max} y_{min} y_{max} z_{min} z_{max}	The minimum and maximum values of x, y, and z for the surface as a whole.
`CONTACT POINT X:`x `Y:`y `Z:`z	The point at which the turtle connects to the surface.
`END POINT X:`x `Y:`y `Z:`z	The point where the turtle is positioned after drawing the surface.
`HEADING X:`x `Y:`y `Z:`z `UP X:`x `Y:`y `Z:`z	The heading and up vectors of the surface, matched to the corresponding vectors of the turtle to determine the orientation of the surface.
`SIZE:` x	The scaling parameter giving the size in surface units to be considered as equivalent to the default unit length associated with the `F` symbol/module in *cpfg* and *lpfg*.

The header is followed by groups of 10 lines, each describing one component patch. There should be one group of 10 lines for each patch making up the surface. Each group contains the following parameters:

Parameter	Description
patchname	The name used to identify the patch in the neighbourhood information below.

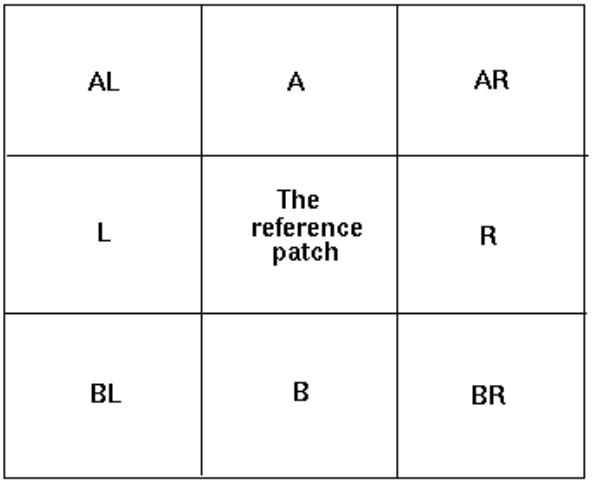


Figure 21: The abbreviations used to define the neighbours of a specified (reference) patch.

Parameter	Description
`TOP COLOR:` i `DIFFUSE:` x `BOTTOM COLOR:` j `DIFFUSE:` y	The color and diffuse lighting coefficients for either side of the patch. If the values are zero, the current turtle parameters are used.
`AL:` *patch1* `A:` *patch2* `AR:` *patch3* `L:` *patch4* `R:` *patch5* `BL:` *patch6* `B:` *patch7* `BR:` *patch8*	The patch neighbourhood information. The parameters are illustrated in Figure 21: `AL` = above left; `A` = above; `AR` = above right; `L` = left; `R` = right; `BL` = bottom left; `B` = bottom; and `BR` = bottom right. If there is no neighbouring patch, the tilde ($\sim$) symbol is used.
$x_{11}\ y_{11}\ z_{11}\ x_{12}\ y_{12}\ z_{12}\ x_{13}\ y_{13}\ z_{13}\ x_{14}\ y_{14}\ z_{14}$ $x_{21}\ y_{21}\ z_{21}\ x_{22}\ y_{22}\ z_{22}\ x_{23}\ y_{23}\ z_{23}\ x_{24}\ y_{24}\ z_{24}$ $x_{31}\ y_{31}\ z_{31}\ x_{32}\ y_{32}\ z_{32}\ x_{33}\ y_{33}\ z_{33}\ x_{34}\ y_{34}\ z_{34}$ $x_{41}\ y_{41}\ z_{41}\ x_{42}\ y_{42}\ z_{42}\ x_{43}\ y_{43}\ z_{43}\ x_{44}\ y_{44}\ z_{44}$	The patch control points. Each line represents one row of four points.

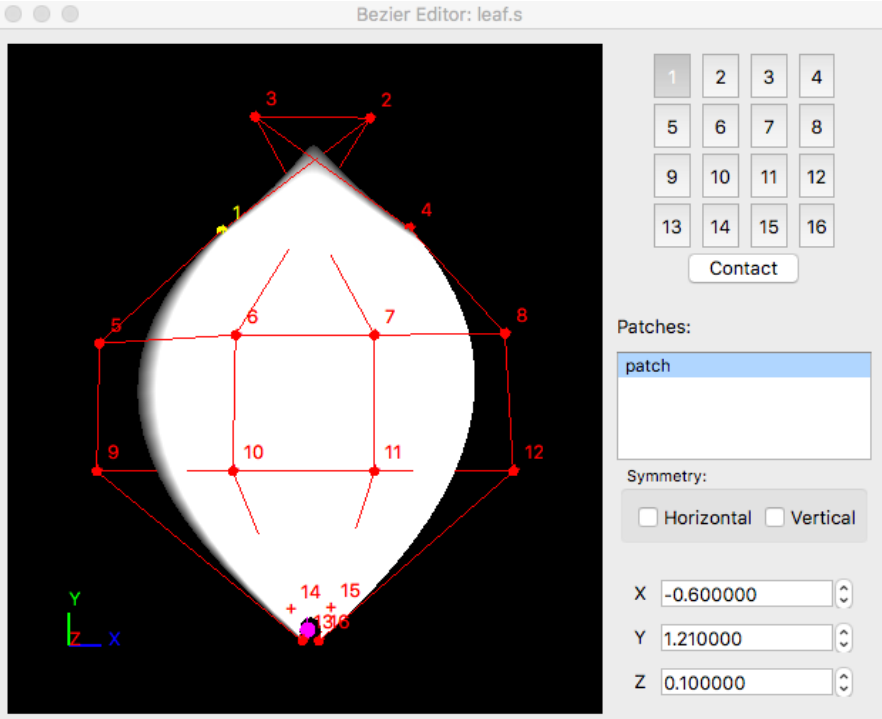


Figure 22: An example of the Bezier Editor window when only a surface file is included on the command line with the `-bezier` option.

6 Surface & texture editor (*stedit*)

See object: Monopodial-Maple

The surface and texture editor, *stedit*, provides slightly different functionality than *bezieredit* (Section 5). It uses the same file format to define a bicubic surface, but is designed for a single Bézier patch[1] with functionality to distort and place a texture on the patch.

The patch is defined in the Bezier Editor window; the texture in the Texture Distorter window. The texture can also be displayed on the patch in the Bezier Editor window.

There are several command line options, depending on whether the surface and/or texture should be displayed:

Command line	Description
`stedit -bezier` *surfacefile*	Open the Bezier Editor, and display the patch defined in *surfacefile*.
`stedit -bezier` *surfacefile texturefile*	Open the Bezier Editor, and display the patch defined in *surfacefile* superimposed with the texture image in *texturefile*.
`stedit -warp` *texturefile*	Open the Texture Distorter window, and display the texture defined in *texturefile*.
`stedit -both` *surfacefile texturefile*	Open both the Bezier Editor window and the Texture Distorter window, with the texture superimposed on the patch in the Bezier Editor window.

[1] *stedit* can display multiple patches (with the texture placed separately on each patch), but it does not include the *bezieredit* functionality to manage the connections between the patches.

6.1 The Bezier Editor window

The main *stedit* window, Bezier Editor, can be invoked with or without a texture. When *stedit* is invoked with the surface file only (using the `-bezier` option), the Bezier Editor window will display the surface and its control points (Figure 22).

6.1.1 Manipulating the view

The mouse can be used for most basic manipulations of the view:

- Rotate - click the left mouse button and drag anywhere in the window
- Pan - hold the Shift key down, click the left mouse button and drag anywhere in the window.
- Zoom - Use the scroll wheel on the mouse to zoom in and out.

Other view operations are available on the menus (see Section 6.1.3).

6.1.2 Editing points

There are several options for selecting and editing the control points and contact point:

- Hold the Command key down, and click and drag with the left mouse button. (See the Lock Rotations menu item in Section 6.1.3, to click and drag without using the Command key.)
- Select the point using the buttons on the side bar.
- Adjust the x, y and z coordinates of the point using the X, Y and Z fields at the bottom of the side bar.
- Click the Horizontal or Vertical checkbox under Symmetry on the side bar, to symmetrically move opposite control points with respect to the contact point.
- Select Edit > Contact Point on the menu bar (Section 6.1.3) for more detailed editing of the contact point.

6.1.3 The Bezier Editor menus

Pop-up menu The main functions associated with the surface are available on the pop-up menu by right-clicking anywhere in the window. Some of these functions are also available on the menu bar, as indicated:

Menu item	Description	Menu bar
Lock Rotation	Locks the current rotation in place, allowing the left mouse button to be used to select control points directly, without the use of the Command key.	View
Transform > Flip	Turn the surface 180° around a Horizontal or a Vertical axis at the contact point, or flip the Z coordinate of each point (Depthwise) to the opposite side of the contact point.	Edit
Transform > Translate...	Open a dialog box to enter the x, y and z increments to apply to all control points. Note that the contact point is NOT translated.	Edit
Transform > Rotate...	Open a dialog box to enter the x, y and z coordinates of the point around which to rotate the surface, and the angle of rotation.	Edit

Transform > Scale...	Open a dialog box to enter the x, y and z factors for scaling the surface. Clicking the Uniform checkbox will sync the scaling across all three axes.	Edit
Reset view	Return the view to the XY plane.	View
Center	Center the surface in the window.	View
Center at Contact	Move the surface such that the contact point is in the centre of the window.	View
Reload	Reread the surface from the surface file. A dialog box will be displayed to confirm that the surface should be reverted to the most recently saved version.	File
Refresh mode	Set the refresh mode to Explicit, Triggered or Continuous. The default is Explicit.	File
Quit	Exit the program. A dialog box will be displayed if changes have been made but not saved.	Stedit

Menu bar There are several functions available on the menu bar, in addition to the ones noted above:

Menu bar	Menu item	Description
File	New	Close the current window (after checking whether to save changes), and open a new Bezier Edit window with a simple square surface.
File	Save	Save the current characteristics of the surface to the surface file. This is only required when the refresh mode is set to Explicit.
File	Save As...	Save the current characteristics of the surface to a new file. A dialog box will be displayed to enter the new file name, which will be used with subsequent Save commands.
File	Open	Close the current window (after checking whether to save changes), and open a new surface file. A dialog box will be displayed to select the file.
Edit	Undo / Redo	Undo / redo the previous action taken.
Edit	Contact Point...	Open a dialog box to adjust all the characteristics of the contact point: its position (Contact Point), End Point, Heading Vector, Up Vector, and Size.
View	Wireframe	Toggle between a wireframe representation of the surface and either the default solid surface or a texture (Section 6.1.4).
View	Resize	Change the size of the control points, the lines between control points, the wireframe lines, and/or the number of wireframe subdivisions.
View	Color	Change the color of each component in the window, or Reset all colors to the default values.

6.1.4 Viewing a texture

There are two options for displaying a texture on the surface:

- Include the texture file on the command line with the `-bezier` option. The control points can be used to manipulate the surface and, therefore, the texture, but the texture itself cannot be manipulated.
- Use the command line option `-both` to open both the Bezier Editor window and the Texture Distorter window (Section 6.2). Changes to the image in the Texture Distorter window will be reflected in the Bezier Editor window.

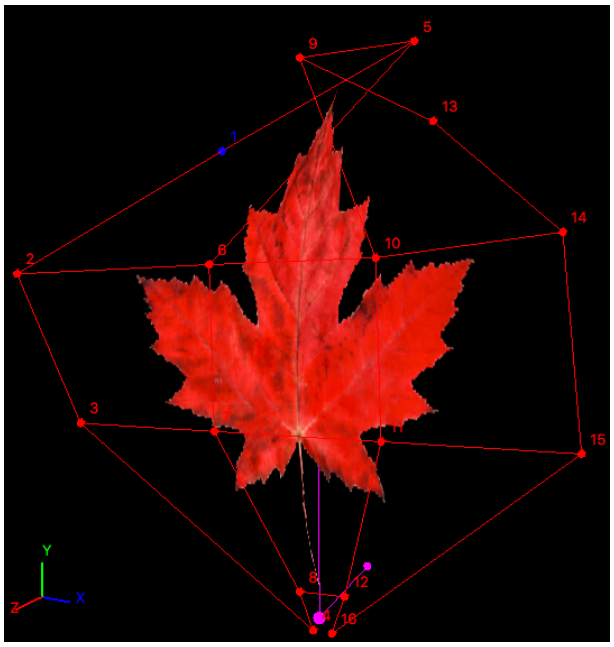


Figure 23: A view of a Bezier Editor surface with a texture superimposed.

In both cases the texture is displayed in the Bezier Editor rather than the "solid" surface (Figure 23). Note that the texture is not displayed in Wireframe mode.

6.2 The Texture Distorter window

A texture is an image file in PNG, JPG, or TIFF format that can be superimposed on a surface defined in the Bezier Editor window. The Texture Distorter provides functionality to distort the image to better fit the surface. It can be invoked separately with the `-warp` command line option, or in conjunction with the Bezier Editor window using `-both`.

6.2.1 Distorting the image

When the image is first displayed, there are default control points in each corner (Figure 24). The image is distorted by:

- Clicking and dragging on the existing control points with the left mouse button.
- Adding more control points by holding the Command key down, and clicking on the location for the new control point.
- Loading a set of predefined control points from a DTX file.

If the texture file is closed and re-opened, it will be displayed with the default four control points again. To save the current control point configuration in a DTX file, see the File > Control points menu (Section 6.2.2).

6.2.2 The Texture Distorter menus

Pop-up menu The main functions associated with the texture image file are available on the pop-up menu by right-clicking anywhere in the window. Some of these functions are also available on the menu bar, as indicated:

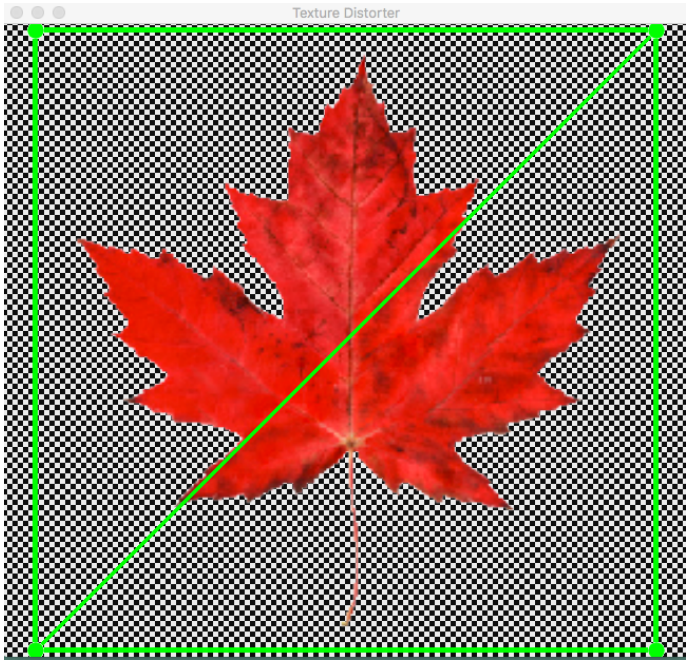


Figure 24: The Texture Distorter window displaying the same texture as Figure 23. Note that many images used as textures, such as this one, do not have a background, in which case a checkerboard pattern is displayed.

Menu item	Description	Menu bar
Resize image	Distort the image by altering its height or width. A dialog box is displayed to enter the new values.	Edit
Rotate CW	Rotate the image clockwise.	Edit
Rotate CCW	Rotate the image counter-clockwise.	Edit
Flip Horizontal	Display a mirror image of the texture.	Edit
Flip Vertical	Flip the texture upside down.	Edit
Save Texture	Save the current view of the texture to *texturefile*.	File > Texture
Reload	Reread the image from the texture file.	File
Refresh mode	Set the refresh mode to Explicit, Triggered or Continuous. The default is Explicit.	File
Quit	Exit the program. A dialog box will be displayed if changes have been made but not saved.	Stedit

Menu bar There are several functions available on the menu bar in addition to the ones noted above:

Menu bar	Menu item	Description
File	Texture > Save As...	Save the displayed image to a new file. A dialog box will be displayed to enter the new file name, and this new file will be used with subsequent Save commands.
File	Texture > Open	Close the current window (after checking whether to save changes), and open a new image file. A dialog box will be displayed to select the file.

File	Control points > Save	Save the current control points into a DTX file. This provides a means of setting the control points the next time the texture file is opened (rather than using the default control points). A dialog box will be displayed if a control point file has not previously been opened or saved.
File	Control points > Save As...	Save the current control points to a new DTX file. A dialog box will be displayed to enter the new file name.
File	Control points > Open	Set the control points based on the information in a DTX file. A dialog box will be displayed to select the file. This file will be used for subsequent Control points > Save commands.
File	Control points > Reset	Reset the control points to the default four, one in each corner of the image.
Edit	Undo / Redo	Undo or redo control point operations.
Edit	Capture	Temporarily save the current state of the image. Note that no visible change is made, and the image file is not updated.
Edit	Reset to capture	Return the image to the version saved with the Capture command.
View	Reset	Resets the view to display the control points and connecting lines (if they were turned off with the View > Show option).
View	Show	Turn on/off the display of the control points and connecting lines.
View	Resize	Change the size of the control points and/or the width of the connecting lines. A dialog box will be displayed to enter the new value.
View	Color	Change the color of components in the window, or reset to the default values.

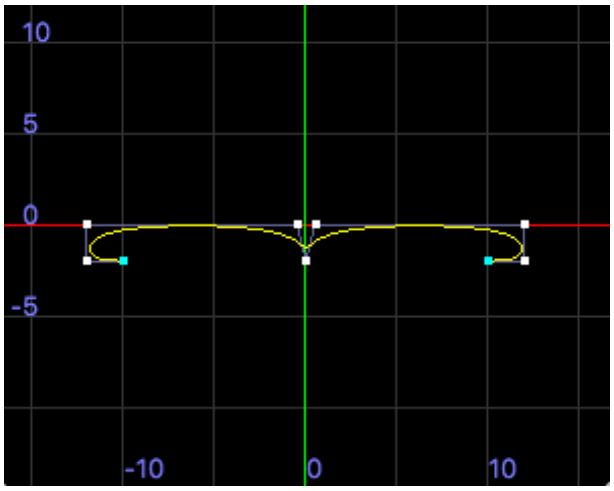
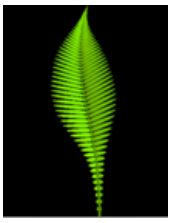


Figure 25: An example of the *cuspy* window.

7 Contour editor (*cuspy*)



See object: FernLeaf

The contour editor, *cuspy*, is a utility for editing B-spline contours in 2D space. The curve may be open or closed.

The command line for invoking the editor is:

```
cuspy [filename]
```

When invoked, the window in Figure 25 is displayed. If *filename* is not specified, a default contour of four control points is created, defining a closed circle about the origin.

Contours defined by *cuspy* can also be grouped using *gallery* (Section 9).

7.1 Manipulating the view

Use the menu (Section 7.3) to show/hide elements of the view (points, line segments, curve, limits), and the mouse and keyboard to:

- Zoom in and out (Alt key). It is also possible to zoom with the scroll wheel on the mouse.
- Pan (Shift key)

7.2 Manipulating the contour

The shape of the contour is manipulated by moving the control points. The actions related to control points all use the left mouse button. They are:

Action	Description
Move	Select and drag the control point.
Add	Hold down the Command button, and click at the desired location.
Change multiplicity	Double click on the control point. The color of the point will change. Double-clicking on a point with a multiplicity of 3 (the maximum) will reset it to multiplicity 1.
Translate all	Hold down the Command button, and use the arrow keys to move the entire contour. For larger steps, hold the Shift key down as well.
Delete	Hold down the Shift and Command buttons, and click on the point to be deleted.

7.3 The *cuspy* menus

Cuspy has both a pop-up menu, invoked by clicking on the window with the right mouse button, and a menu bar. Both contain the same actions. The following table lists the actions in the order given on the pop-up menu, indicating where they can be found on the menu bar.

Action	Description	Menu Bar
Save	Save the current contour to the file. This is only needed when Refresh mode = Explicit (see below).	File
Save as...	Save the current contour to a new file. A dialog box will be displayed to enter the new file name.	File
Revert to saved	Reload the contour from the file, overwriting any changes made since the contour was last saved.	File
Load background image	Load an image to be displayed in the background. This makes it possible to define a contour to match an image. By default the image is displayed with the lower left corner at the origin. It can be moved using the arrow keys.	File
Closed	Create a closed contour from the last to the first control points.	View
View points	Display/hide the control points.	View
View segments	Display/hide the line segments connecting the control points.	View
View curve	Display/hide the contour curve.	View
View axes	Display/hide the lines representing the x and y axes.	View
Edit name...	Update the name of the contour in the specification file.	File
Edit number of samples...	Change the number of samples that will be precomputed.	File
Refresh mode	Set the mode used to refresh the file: Explicit = only with the Save command (default). Triggered = when a change is made and the mouse is released. Continuous = as a change is being made (all values as the mouse is moved).	

7.4 The contour specification file

The contour editor has been enhanced several times resulting in several versions of the specification file. (See earlier versions in Section 7.5.) The editor will read all versions of the file, but will update the file to the latest version when saving.

The latest version of the contour specification file has the following format:

Parameter	Description
`cver 1 3`	The latest version of the file.
`name:` *name*	The name of the contour.
`points:` *n1 n2*	The number of distinct control points, *n1*, and the sum of their multiplicities, *n2*.
$x_1\ y_1\ z_1\ m_1$ $x_2\ y_2\ z_2\ m_2$ $x_3\ y_3\ z_3\ m_3$ ... $x_{n1}\ y_{n1}\ z_{n1}\ m_{n1}$	Each distinct control point, giving its x, y, and z coordinates, and its multiplicity, m.

Parameter	Description
`type:` *type*	The type of contour. A contour can be open (`o`) or closed (`c`) with regular (`r`) or endpoint (`e`) interpolation. Therefore, there are four distinct values for *type*: `or` = open with regular interpolation `oe` = open with endpoint interpolation `cr` = closed with regular interpolation `ce` = closed with endpoint interpolation
`samples:` n	The number of sides, n, a generalized cylinder will have when drawn with this contour.

It is recommended that control points be specified in counter-clockwise order (with respect to point [0,0,0]), because interpolation between clockwise and counter-clockwise contours results in a twisted generalized cylinder.

Note that if a contour includes some singularity (e.g. a sharp edge created by having three control points at the same location), the normals are not correct.

7.5 Obsolete Contour Editor formats

The following file formats may exist in old objects within *vlab*. The files can still be read by the contour editor, but will be saved using the latest format.

Version 1.2 does not include the `samples` parameter. Its format is:

`cver 1 2`
`name:` *name*
`points:` *n1 n2*
$x_1\ y_1\ z_1\ m_1$
$x_2\ y_2\ z_2\ m_2$
$x_3\ y_3\ z_3\ m_3$
...
$x_{n1}\ y_{n1}\ z_{n1}\ m_{n1}$
`type:` *type*

where all parameters are defined as in the latest version.

Version 1.1 has a simpler form of the `type` parameter, and does not include the `samples` parameter. Its format is:

`cver 1 1`
`name:` *name*
`points:` *n1 n2*
$x_1\ y_1\ z_1\ m_1$
$x_2\ y_2\ z_2\ m_2$
$x_3\ y_3\ z_3\ m_3$
...
$x_{n1}\ y_{n1}\ z_{n1}\ m_{n1}$
`type:` *type*

where all parameters are defined as in the latest version, except *type* which is either `open` or `closed`.

Version 1.0 has an entirely different format. It begins with a single header line followed by the x, y, and z coordinates of each point:

npts dim type
x_1 y_1 z_1
x_2 y_2 z_2
x_3 y_3 z_3
...
x_{npts} y_{npts} z_{npts}

where *npts* is the number of points, *dim* is the dimension (`2` or `3`), and *type* is `open` or `closed`. For example, a closed 3-dimensional contour with 12 control points could be defined as:

```
 12 3 closed
 0.16 -1.12 2.0
 0.41 -1.04 1.0
 0.58 -0.33 0.5
 1.08 -0.04 0.2
 1.08  0.49 0.0
 0.49  0.54 0.0
 0.33  0.91 0.1
-0.37  1.04 0.3
-0.70  0.62 0.2
-1.12  0.16 0.1
-0.87 -0.74 0.3
-0.41 -0.66 1.0
```

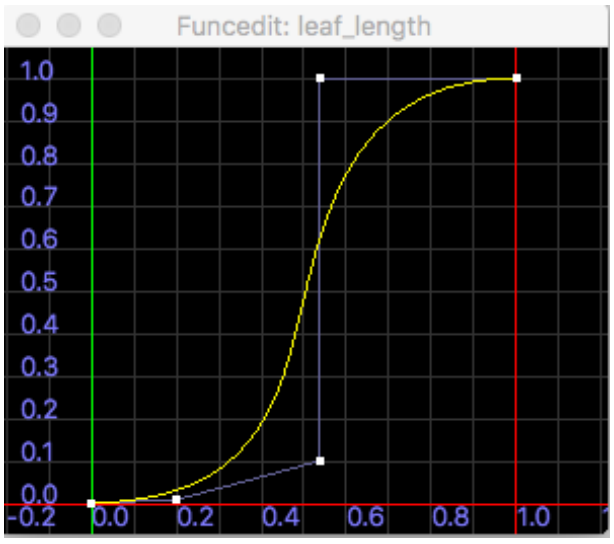


Figure 26: An example of the *funcedit* window.

8 Function editor (*funcedit*)

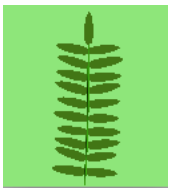
The function editor defines a function as a spline curve in 2D space where the first control point's x coordinate is equal to 0, and the last point's x coordinate is equal to 1. In addition, for any two control points, p_i and p_{i+1}, $x_{p_i} \leq x_{p_{i+1}}$.

The command line for invoking the editor is:

```
funcedit [filename]
```

When invoked the window in Figure 26 is displayed.

Functions defined by *funcedit* can also be grouped using *gallery* (Section 9).



See object: LeafRandom

8.1 Manipulating the view

Use the menu (Section 8.3) to show/hide elements of the view (points, line segments, curve, limits), and the mouse and keyboard to:

- Zoom in and out (Alt key). It is also possible to zoom with a scrolling wheel on the mouse.
- Pan (Shift key)

8.2 Manipulating the function

The shape of the function is manipulated by moving the control points. The actions related to control points all use the left mouse button. They are:

Action	Description
Move	Select and drag the control point. The first and last control points are constrained to the limits of the range.
Add	Hold down the Command key, and click on the location for a new control point.
Translate all	Hold down the Command key, and use the arrow keys to move the entire contour. For larger steps, hold the Shift key down as well.
Delete	Hold down the Shift and Command keys, and click on the point to be deleted.

8.3 The *funcedit* menus

Funcedit has both a pop-up menu, invoked by clicking on the window with the right mouse button, and a menu bar. Both contain the same actions. The following table lists the actions in the order given on the pop-up menu, indicating where they can be found on the menu bar.

Action	Description	Menu Bar
Save	Save the current function to the file. This is only needed when Refresh mode = Explicit (see below).	File
Save as...	Save the current function to a new file. A dialog box will be displayed to enter the new file name.	File
Revert to saved	Reload the function from the file, overwriting changes made since the function was last saved.	File
Load background image	Load an image to be displayed in the background. This makes it possible to define the function to match the image. By default the image is displayed with the lower left corner at the origin. It can be moved using the arrow keys.	File
Flip view	Flip the x and y axes.	View
View points	Display/hide the control points.	View
View segments	Display/hide the line segments connecting the control points.	View
View curve	Display/hide the function curve.	View
View limits	Display/hide the lines representing the range constraints of the function.	View
Edit name...	Update the name of the function within the file.	File
Edit number of samples...	Change the number of samples that will be precomputed.	File
Refresh mode	Set the mode used to refresh the file: Explicit = only with the Save command (default). Triggered = when a change is made and the mouse is released. Continuous = as a change is being made (all values as the mouse is moved).	

8.4 Function specification file

The current version of the function file has the following format. For earlier versions of the file see Section 8.5.

Parameter	Description
`fver 1 1`	The current version of the specification file.
`name:` *name*	The name of the function.
`samples:` n	The number of samples, n, to precompute (rather than calculated each time the function is accessed).
`flip:` *value*	Whether the independent variable, x, is displayed horizontally or vertically by the function editor. When *value* = `no`, x is on the horizontal axis; when *value* = `yes`, x is on the vertical axis.
`points:` n	The number of control points, n, where $n \geq 4$.

Parameter	Description
x_1 y_1 x_2 y_2 x_3 y_3 ... x_n y_n	The x and y coordinates of each control point.

8.5 Obsolete Function Editor formats

The following file format may exist in old objects within *vlab*. Files with this format can still be read by the function editor, but will be saved using the latest format.

Version 1.0 does not include the `name`, `samples`, or `flip` parameters, and begins with a `range` parameter. The complete format is:

`range:` `0.0 1.0`
`points:` n
x_1 y_1
x_2 y_2
x_3 y_3
...
x_n y_n

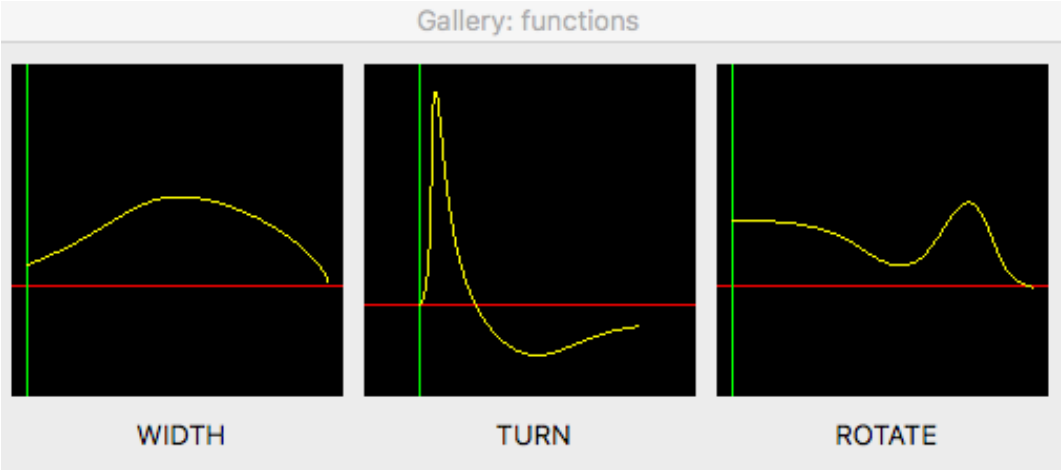


Figure 27: An example of a *gallery* window displaying three functions. Double-click on one of the icons to open the function in *funcedit*.

9 The *gallery* utility

The *gallery* utility can be used to display a group of contours created by *cuspy* (Section 7), or a goup of functions created by *funcedit* (Section 8).

The command line for invoking the editor is:

```
gallery [filename.xset]
```


See object: LilyLeaf

where *filename* has one of the following extensions:

`.cset` for a set of contours
`.fset` for a set of functions

When invoked a window such as the one in Figure 27 is displayed. To run *cuspy* or *funcedit*, simply double-click on the icon representing the contour/function.

9.1 The *gallery* menu

Right-click in the *gallery* window to display the pop-up menu. The items on the menu are:

Menu item	Description
Save gallery	Save changes made to the gallery.
Save gallery as...	Save a copy of the gallery to another file. A dialog box will be displayed to enter the new file name.
Update all views	Update the icons to display the current view of each contour/function.
Create new item...	Add a new contour/function to the gallery.
Duplicate item...	Create a duplicate of the contour/function that was right-clicked. A dialog box will be displayed to enter a name for the new item.
Load existing item...	Add an existing contour/function to the gallery from a `.con` or `.func` file.
Remove item	Remove the contour/function that was right-clicked.
Refresh mode	Set the mode for saving changes to the gallery file. The modes are: Explicit = with the Save gallery menu item; or Continuous/Triggered = when a change is made with *cuspy* or *funcedit* (also depends on the Refresh mode of these tools).
Exit	Quit *gallery*. If there are unsaved changed, a dialog box will be displayed.

9.2 Gallery specification file

The filename extension and first line of a gallery specification file defines whether the file contains contours or functions:

Set of	Filename extension	First line
Contours	`.cset`	`contourgalleryver 1 1`
Functions	`.fset`	`funcgalleryver 1 1`

The format of the remainder of the file is:

`items:` *n*
item1
item2
...
itemn

where each *item* is a complete definition of a *cuspy* contour (Section 7.4) or a *funcedit* function (Section 8.4).

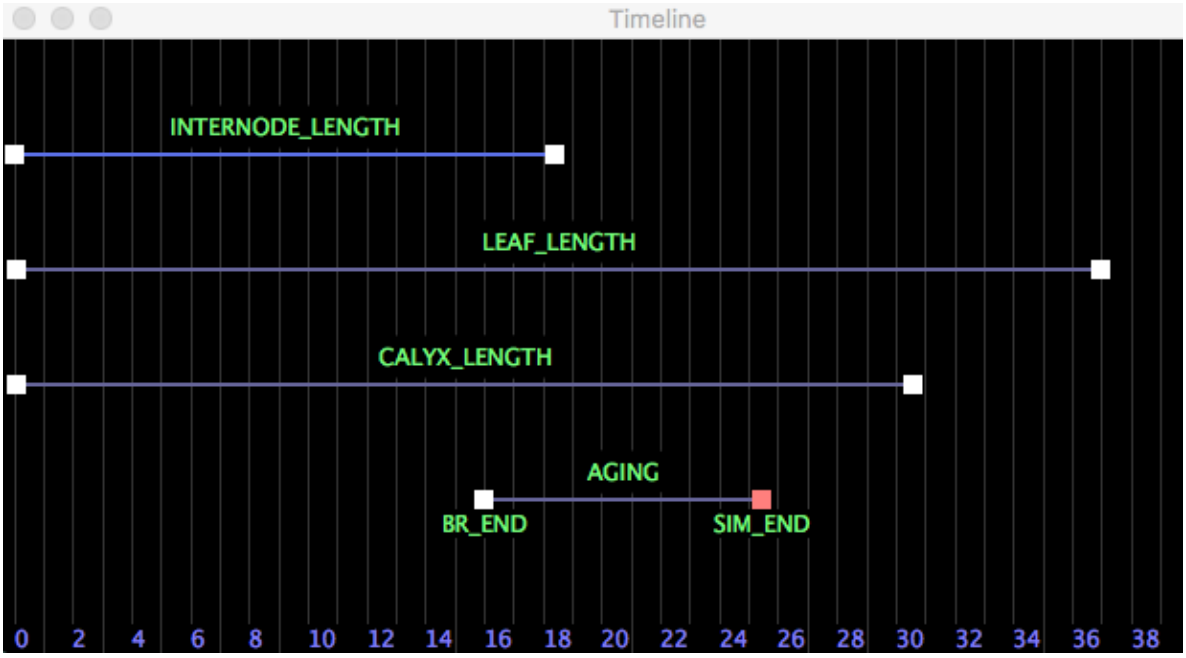


Figure 28: An example of a *timeline* editor window displaying four functions. Update a timeline by clicking and dragging on one of its endpoints.

10 Timeline editor (*timeline*)

The *timeline* editor provides another method of defining a set of functions. Each function is created using the *funcedit* tool over the range [0,1], but is evaluated using the range specified by its associated timeline (using the `tfunc()` function within *lpfg*).

The command line for invoking the editor is:

`timeline [-rmode` *rmode*`]` *filename*`.tset`

See object: Lychnis

where *filename*`.tset` contains a set of functions each with a defined time period (see Section 10.4), and *rmode* specifies the refresh mode: `explicit, triggered,` or `continuous` (Section 10.3). When invoked, a window such as the one in Figure 28 is displayed.

10.1 Using timelines

10.1.1 Manipulating a timeline

The timeline endpoints can be adjusted by clicking and dragging on them. To move multiple endpoints in unison, hold the Shift key down and select each of them, then click and drag on any one of the selected endpoints to move them together. Use Deselect all on the menu (see below) when done.

To edit the function associated with a timeline, double-click on its name. This will open *funcedit* (see Section 8).

To pan left and right, click and drag anywhere in the window except on an endpoint.

10.1.2 General menu items

The main menu is available by right-clicking anywhere in the *timeline* window. Some of the options are also available on the menu bar, under the menu indicated.

Action	Description	Menu bar
Deselect all	Deselect any endpoints that have been selected.	
Edit	Enter edit mode to add/change/delete timelines and their associated functions. See Section 10.2.	
Save	Save all changes to the current `.tset` file.	File
Save as	Save all changes to a different `.tset` file. A dialog box will be displayed to select the new file. All subsequent changes will also be saved to this new `.tset` file.	File
Refresh mode	Set the refresh mode to Explicit, Triggered or Continuous (Section 10.3). This will override the mode set using the `-rmode` argument on the command line.	File
Quit	Exit the program. If changes have been made but not saved, a dialog box will be displayed.	Timeline

10.2 Timeline edit mode

To add, change, or delete a timeline and/or its associated function, select Edit from the pop-up menu to enter edit mode. The background colour will change to indicate the new mode.

Select a specific timeline by clicking on its name.

Double clicking on the timeline name will open a dialog box for editing the characteristics of timeline (see Section 10.2.2).

10.2.1 Timeline Edit Menu

Right-click anywhere in the window to display the pop-up menu. It contains the following items:

Menu item	Description
Add timeline	Add a new timeline at the bottom of the window. A dialog box will be displayed to enter the characteristics of the timeline (Section 10.2.2), followed by *funcedit* (Section 8) to define the associated function.
Move selected timeline up	Switch the selected timeline with the one above it.
Move selected timeline down	Switch the selected timeline with the one below it.
Delete selected timeline	Display a dialog box to confirm that the selected timeline should be removed. The timelines below will be moved upwards.
Open function	Open the *funcedit* tool to edit the associated function. This is similar to double-clicking on the timeline name when in Execute mode.
Execute	Return to Execute mode to manipulate the timelines.
Save	Save the updated timelines to the current `.tset` file.
Save as	Save the updated timelines to a different `.tset` file. A dialog box will be displayed to select the new file. All subsequent changes will also be saved to this new `.tset` file.

10.2.2 Timeline Characteristics

The characteristics of a timeline can be edited by double-clicking on the timeline name to display the dialog box (Figure 29). This dialog box is also displayed when adding a new timeline with the Add timeline menu item.

The fields in the dialog box are:

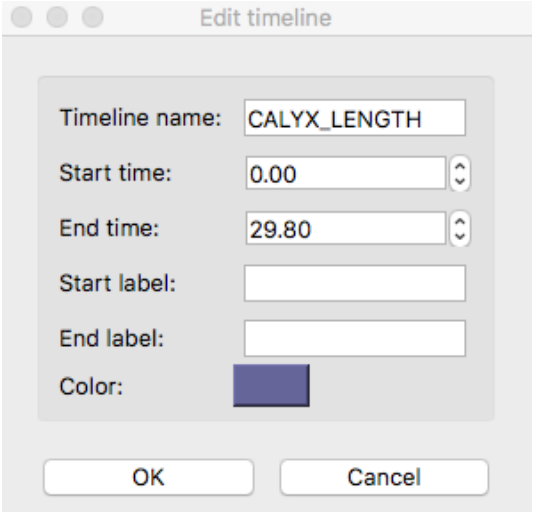


Figure 29: An example of the dialog box that is displayed to edit the characteristics of a timeline.

Field	Description
Timeline name	The name of the timeline, displayed above it.
Start time	The bottom of the range over which the function will be evaluated.
End time	The top of the range over which the function will be evaluated.
Start label	The name of the left endpoint of the timeline, displayed below the point.
End label	The name of the right endpoint of the timeline, displayed below the point.
Color	The color of the line. Click on the colored box to open a color wheel dialog.

The Timeline name, as well as the Start label and End label values, should be a single word (no spaces) for use with the `tfunc` function within *lpfg*.

The Start time and End time values are floating point numbers with two decimal places. The arrows at the side of the fields increase/decrease the value by 1.00.

10.3 Timeline editor output

In addition to updating *filename*`.tset` (Section 10.4), the timeline editor also outputs changes to labelled endpoints to *stdout*. The output depends on the Refresh mode:

Rmode	Output to *stdout*
Explicit	Output all labelled endpoints on Save.
Trigger	Output the final value of a labelled endpoint when the mouse button is released.
Continuous	Output all values of a labelled endpoint as the point is dragged.

The output has the form:

`d` *endpoint-name value* `1`

This format is standard for the *vlab* datafile editor, *awkped* (Section 2.3.2), and is used to update file parameters of the type:

`#define` *fieldname value*

This mechanism can be used to graphically update parameters within *lpfg*. For example, the named endpoints in Figure 28 could be associated with a data file, `aging.pset`, containing:

```
#define BR_END 15
#define SIM_END 25
```

By piping the timeline editor output through *awkped* with the command:

`timeline` *filename*`.tset | awkped aging.pset`

endpoint changes to the AGING timeline would update the corresponding parameters in `aging.pset`. If this file is included on the *lpfg* command line, the parameters can be read within the L-system using the `val()` function:

```
branching_ends = val(BR_END);
simulations_ends = val(SIM_END);
```

In this case, only the endpoints of the AGING timeline are of interest; the function associated with the timeline is not used.

10.4 Timeline specification file

The timeline editor reads and saves from/to a text file, *filename*`.tset`, which has a short header containing:

Parameter	Description
`timeEdit 1 2`	The current version of the specification file.
`items:` *n*	The number of timelines in the file.

The header is followed by a description of each timeline:

Parameter	Description
`start:` *startpt*	The left endpoint of the timeline defined as a 2 decimal number.
`end:` *endpt*	The right endpoint of the timeline defined as a 2 decimal number.
`name:` *linename*	The name of the timeline. This should be a single word (no spaces) for use with the `tfunc` function within *lpfg*.
`color:` *R G B*	The RGB components of the timeline color.
`startlabel:` *slabel*	The label name associated with the left endpoint of the timeline. This is optional.
`endlabel:` *elabel*	The label name associated with the right endpoint of the timeline. This is optional.
`timelineFunceditFileStart:`	The start of a *funcedit* description of the timeline's function, using the standard *funcedit* format (Section 8.4) which spans several lines.
`timelineFunceditFileEnd:`	The end of the function, and the timeline, description.

11 Credits

The *panel* editor was included in the first release of *vlab*, developed by Lynn Mercer as part of her Master's thesis [1]. The latest version, with graphical editing, was developed by Alejandro Garcia and Pascal Ferraro.

The *palette* program was developed by Przemyslaw Prusinkiewicz.

The material editor, *medit*, was developed by Joanne Penner.

The tools *bezieredit*, *cuspy*, and *gallery* were developed by Colin Smith, as well as the current QT implementation of *funcedit*. Radek Karwowski developed the original version of *funcedit* using Motif.

The *stedit* program was developed by Mark Koleszar.

The file format used by both *stedit* and *bezieredit* is based on the surface editor, *ise*, developed by Jim Hanan.

The original version of the *timeline* editor was developed by John Hall. The latest version was implemented by Pascal Ferraro.

All tools have been kept up-to-date within *vlab* by Pascal Ferraro.

12 Document History

This is the first manual to combine all the *vlab* tools into a single document. However, each tool was originally documented by the developer(s).

Date	Description	By
2021	First version combining all tools in a single manual	Lynn Mercer Pascal Ferraro Przemyslaw Prusinkiewicz

References

[1] Lynn Mercer. The virtual laboratory. Master's thesis, University of Regina, 1991.