Welcome to Screen Saver Construction Set 's Help. The information in the help selection is a subset of what is available in the Manual installed with your software. If you don't find what you need here, or if you require additional clarification, please consult the Manual and its dependent documents.

IMPORTANT: The Screen Saver Construction Set Help function consists of a subset of the complete Screen Saver Construction Set documentation. For a more thorough discussion of the features of this application, please see the Manual document and its dependent documents, installed with Screen Saver Construction Set . Double-click on the Manual icon, or select Manual from the Screen Saver Construction Set section of the Windows Programs menu.

- Help with Configuring Screen Saver Construction Set
- Help with Using Screen Saver Construction Set
- Help with Registering Screen Saver Construction Set
- Help with Contacting Alchemy Mindworks

Configuration and Setup

The Screen Saver Construction Set Setup dialog can be accessed by clicking on the Setup button in the tool bar or by selecting Setup from the File menu. Because there are a lot of format variations and other things you're likely to want to change periodically, you should become familiar with Setup.

Applet Stack Size: This item sets the initial stack size for new applets. See the discussion of stacks in the ANIMAL Language Reference document.

Default View Mode: This item sets the default view mode for new document windows.

Image Compression: This item defines the amount of compression used when saving Image blocks and the images stored with applets. Setting this value to zero will cause Screen Saver Construction Set to store images with no compression. Setting it to nine will result in the most aggressive possible compression. See the discussion of image compression in the Reference document.

Maximize Paint: Enable this item to cause Windows Paint to be maximized when it's called from an Image block editor. Disable it to allow Windows Paint to open normally.

Optimum Vertical Windows: Enable this item to have Screen Saver Construction Set's document windows adjusted to use as much of the vertical window space as possible. Disable it to have them created at the default size for a document window. Note that if this item is enabled, the default vertical window size will leave a gap between it and the bottom of the Screen Saver Construction Set application window to allow for minimized documents.

Registration Code: Your registration code will be provided when you register Screen Saver Construction Set Professional. If you received Screen Saver Construction Set Professional on an Alchemy Mindworks CD-ROM, it will be on your invoice in the line item for Screen Saver Construction Set Professional. It will look like this:

```
Screen Saver Construction Set Pro          REG # 12345-12-67890-88          $30.00
```

If you have received a confirmation e-mail message after registering Screen Saver Construction Set, it will appear in this message, like this:

```
1 Screen Saver Construction Set Pro          REG CODE 12345-12-67890-88
```

Screen Saver Construction Set registration codes are all 17 characters long, in the form of one five-digit group of numbers, a dash, one two-digit group of numbers, a dash, a second five-digit group on numbers, a dash and a final two-digit group of numbers. You must enter your code exactly in the Registration Code field to successfully register Screen Saver Construction Set. The dashes separating the digit groups are not entered.

NOTE: Once it has been entered, your registration code will not be displayed by Screen Saver Construction Set. If you have cause to re-install Screen Saver Construction Set in the future, you will require your registration name and code. Write them down now and store them in a safe place.

Registration Name: Your registration code is a complex checksum based on the characters of your name. If your name is not entered correctly in the Registration Name field, your registration code will not be accepted. Your registration name is printed on your invoice and is included in the confirmation e-mail message you received when you registered Screen Saver Construction Set. It must be entered exactly - all the characters, spaces and punctuation in your registration name as it is provided to you must match what you enter in the Registration Name field.

Saver Timeout: This item sets the screen saver time out in seconds. It's used when Screen Saver Construction Set makes a screen saver active, should you save it to your \WINDOWS directory.

Search for Paintbrush: If this item is enabled and Windows Paint – the MSPAINT.EXE file – is not found when Screen Saver Construction Set boots up, the software will search your had drive for it. Disable this item if Windows Paint is not on your system. Note that if you have Windows Paint on your system, and Screen Saver Construction Set has located it, this item will be ignored.

Show Startup Window: Enable this item to have the startup logo window displayed when Screen Saver Construction Set boots up.

Show Tip of the Day: Enable this item to have the tip of the day displayed when Screen Saver Construction Set boots up.

Warn Before Deleting Multiple Blocks: Enable this item to be warned if you click on the Delete button or hit Del when you have multiple blocks selected in a document window.

Default View Mode: This item sets the default view mode for new document windows.

Maximize Paint: Enable this item to cause the Windows Paint to be maximized when it's called from an Image block editor. Disable it to allow Windows Paint to open normally.

Optimum Vertical Windows: Enable this item to have Screen Saver Construction Set's document windows adjusted to use as much of the vertical window space as possible. Disable it to have them created at the default size for a document window. Note that if this item is enabled, the default vertical window size will leave a gap between it and the bottom of the Screen Saver Construction Set application window to allow for minimized documents.

Show Startup Window: Enable this item to have the startup logo window displayed when Screen Saver Construction Set boots up.

Show Tip of the Day: Enable this item to have the tip of the day displayed when Screen Saver Construction Set boots up.

Warn Before Deleting Multiple Blocks: Enable this item to be warned if you click on the Delete button or hit Del when you have multiple blocks selected in a document window.

Warn Before Displaying Manual: Enable this item to have Screen Saver Construction Set display a prompt before its on-line manual appears.

Search for Paintbrush: If this item is enabled and Windows Paint – the MSPAINT.EXE file – is not found when Screen Saver Construction Set boots up, the software will search your had drive for it. Disable this item if Windows Paint is not on your system. Note that if you have Windows Paint on your system, and Screen Saver Construction Set has located it, this item will be ignored.

Registration Code: Your registration code is provided when you register Screen Saver Construction Set. If you received Screen Saver Construction Set on an Alchemy Mindworks CD-ROM, it's on your invoice in the line item for Screen Saver Construction Set . It will look like this:

```
Screen Saver Construction Set Pro          REG # 12345-12-67890-88          $30.00
```

If you have received a confirmation e-mail message after registering Screen Saver Construction Set, it will appear in this message, like this:

```
1 Screen Saver Construction Set Pro        REG CODE 12345-12-67890-88
```

Screen Saver Construction Set   registration codes are all 17 characters long, in the form of one five-digit group of numbers, a dash, one two-digit group of numbers, a dash, a second five-digit group on numbers, a dash and a final two-digit group of numbers. You must enter your code exactly in the Registration Code field to successfully register Screen Saver Construction Set. The dashes separating the digit groups are not entered.

Note: Once it has been entered, your registration code will not be displayed by Screen Saver Construction Set. If you have cause to re-install Screen Saver Construction Set in the future, you will require your registration name and code. Write them down now and store them in a safe place.

Registration Name: Your registration code is a complex checksum based on the characters of your name. If your name is not entered correctly in the Registration Name field, your registration code will not be accepted. Your registration name is printed on your invoice and is included in the confirmation e-mail message you received when you registered Screen Saver Construction Set. It must be entered exactly – all the characters, spaces and punctuation in your registration name as it is provided to you must match what you enter in the Registration Name field.

No help is available for this item.

Reference

Screen Saver Construction Set embodies a great deal of functionality for managing and manipulating graphic images. It's unlikely that you'll find uses for absolutely everything Screen Saver Construction Set knows how to do. It's more unlikely still that you'll want to try to learn everything there is to know about Screen Saver Construction Set in a single session.

If you have not already done so, please read the Introduction and Tutorial document installed with Screen Saver Construction Set. It will walk you through the basic operation of the software and provide you with step by step procedures for performing the most common tasks that Screen Saver Construction Set is called upon to do.

- ANIMAL function reference
- Create Installers
- Header block editor
- Image block editor
- System information
- Sound block editor
- Text block editor

ANIMAL Language Function Reference

Abs(integer a)

The **Abs** function returns the absolute value of its argument. If the argument is less than zero, a positive value will be returned.

Add(integer a1, integer a2)

The **Add** function returns the sum of its two arguments

And(integer a1, integer a2)

The **And** function returns the logical AND of its two arguments. If both arguments are TRUE, the return value will be TRUE – otherwise it will be FALSE.

Arc(integer left, integer top, integer right, integer bottom, integer startangle, integer endangle)

The **Arc** function draws an arc of the ellipse bounded by the rectangle defined by its first four arguments. The arc begins at the angle defined by *startangle* and ends at the angle defined by *endangle*. The three o'clock position is represented by zero degrees, and increasing angles rotate counterclockwise. Angles are specified in degrees, ranging from zero through 359.

Beep([integer type])

The **Beep** function will cause your system's speaker to beep. If you call **Beep** with no argument, your system's default sound will be generated. There are a number of other sounds available to **Beep**, which can be selected by the argument passed to it. These sounds correspond to the sounds generated when system messages appear under Windows.

- 0 – The default sound.
- 16 – The sound associated with the hand icon.
- 32 – The sound associated with the question mark icon.
- 48 – The sound associated with the exclamation point icon.
- 64 – The sound associated with the asterisk icon.
- -1 – A short click through the internal speaker, rather than your sound card.

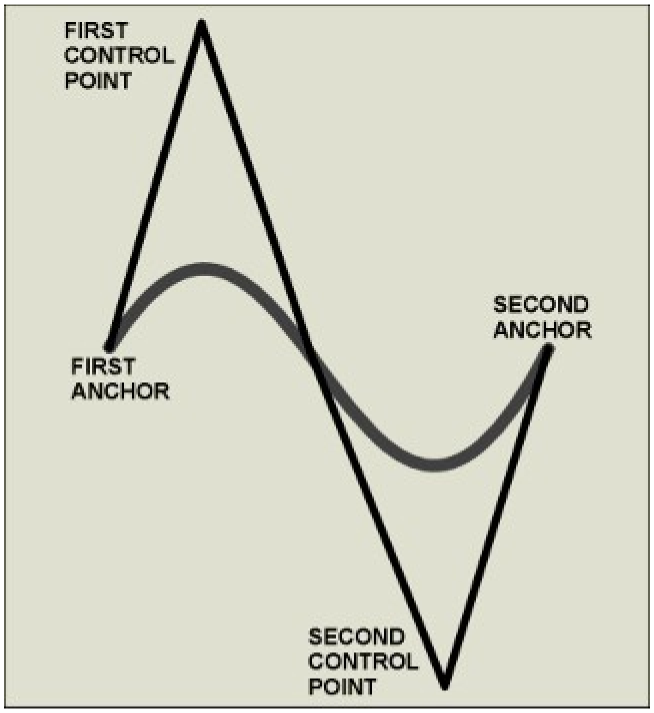
Other values passed to **Beep** may have unpredictable results.

Note that the actual sounds associated with these arguments can be configured by the users of Windows machines through the Sounds applet of the Control Panel. As such, **Beep** may make different noises on different computers.

BezierCurve(array a1, integer count)

The **BezierCurve** function draws complex Bézier curves. The array passed as its first argument defines the control points for the curve, and the value of the second argument specifies the number of elements in the array.

The following illustrates the control points of a Bézier curve:



A Bézier curve must be defined by a minimum of four points. Each point is stored as two consecutive elements in the array passed to **BezierCurve**. The first element is the horizontal coordinate and the second element is the vertical coordinate. The first point is the starting anchor; the second point is the first control; the third point is the second control and the fourth point is the ending anchor. A second curve can be added to the first by adding three additional points – six array entries – to the list of points. The ending anchor for the first curve will become the starting anchor for the second curve.

You can generate as many contiguous Bézier curves as you like with a single call to **BezierCurve**. Note that the control points of a Bézier curve may reside outside the boundaries of your screen.

BitAnd(integer a, integer mask)

The **BitAnd** function performs a bitwise AND between its two arguments and returns the result.

BitOr(integer a, integer mask)

The **BitOr** function performs a bitwise OR between its two arguments and returns the result.

BitXOr(integer a, integer mask)

The **BitXOr** function performs a bitwise Exclusive OR between its two arguments and returns the result.

ClearScreen([integer color])

The **ClearScreen** function clears the entire screen. If you don't pass an argument to

ClearScreen, the screen will be cleared to the background color defined in the Header block of your screen saver. Alternately, you can pass it a color to clear the screen to. See the **MakeColor** function for more about generating colors.

CopyScreenArea(integer destleft, integer desttop, integer left, integer top, integer right, integer bottom)

The **CopyScreenArea** function copies the rectangular area defined by the *left*, *top*, *right* and *bottom* arguments to the location specified by the *destleft* and *desttop* arguments. If the source rectangle for **CopyScreenArea** extends beyond the edges of your screen, the actual area copied will be adjusted to only include pixels which reside within your screen area. The **CopyScreenArea** function creates a copy of the area to be duplicated before it paints new pixels to your screen, so applying it to overlapping areas will work correctly.

Dec(integer variable a, [integer value])

The **Dec** function will decrement the value of the integer variable passed as its first argument. By default, it will subtract 1 from the integer variable. If you pass a second argument to **Dec**, it will subtract the value of this argument.

DeclareArray(array name, integer count)

The **DeclareArray** declares an array defined by the name argument and allocates count integer values. All the values are initially set to zero. Once an array has been defined, it cannot be destroyed or its size changed. Array variable names are identified as beginning with the & character. The value of *count* must be greater than zero and less than 32768.

DeclareInt(integer variable name, [integer init])

The **DeclareInt** function declares an integer variable. By default, new integer variables are assigned the value zero. If the *init* argument is present, it will be stored in the newly created integer variable. Integer variable names are identified as beginning with the @ character.

DeclareString(string variable name, [string init])

The **DeclareString** function declares a string variable. By default, new string variables contain zero-length strings. If the *init* argument is present, it will be stored in the newly created string variable. String variable names are identified as beginning with the $ character.

Div(integer a, integer b)

The **Div** function returns *a* divided by *b*.

Do()

The **Do** function indicates the beginning of a **Do / While** loop.

DrawEdge(integer left, integer top, integer right, integer bottom, integer edgeflags, integer borderflags)

The **DrawEdge** function allows you to access Windows' internal logic for drawing the edges of three-dimensional objects, such as buttons and frames. It's useful for simulating bits of Windows in your screen savers. The *left*, *top*, *right* and *bottom* arguments to **DrawEdge** define the rectangular area for the edge to be drawn. The *edgeflags* argument specifies the edge characteristics and the *borderflags* argument specifies which borders the edges will apply to. The latter two values can be a bit tricky to work with.

The *edgeflags* argument is formed by bitwise ANDing the following values together:

- Raised outer edge – 1
- Sunken outer edge – 2
- Raised inner edge – 4
- Sunken inner edge – 8

Useful combinations of these flags include:

- Raised edge – BitOr(1,4)
- Sunken edge – BitOr(2,8)
- Etched edge – BitOr(4,8)
- Bump edge – BitOr(1,8)

The *borderflags* argument is formed by bitwise ANDing the following values together:

- Left border – 1
- Top border – 2
- Right border – 4
- Bottom border – 8

Useful combinations of these flags include:

- Top left – BitOr(2,1)
- Top right – BitOr(2,4)
- Bottom left – BitOr(8,1)
- Bottom right – BitOr(8,4)
- Rectangle – BitOr(BitOr(1,2),BitOr(4,8))

The most commonly used *borderflags* argument will be Rectangle – all four sides. You can save yourself some typing by keeping in mind that the result of all those calls to **BitOr** will, in fact, be 15. If *borderflags* is set to 16, a diagonal line will be drawn.

The **DrawEdge** function draws using the colours defined in the colour scheme selected through the Windows control panel.

DrawGradient(integer startcolor, integer endcolor, integer left, integer top, integer right, integer bottom, integer vertical)

The **DrawGradient** function will fill a rectangular area with a color gradient. The gradient will be a smooth transition between the *startcolor* value and the *endcolor* value. The color values will typically be generated by **MakeColor**.

Set the *vertical* argument to 0 if you want to create a gradient which runs horizontally, or to 1 to create a gradient which runs vertically.

Gradients are significantly less attractive in a 256-color environment.

DrawPicture(integer picturenumber, integer left, integer top,[integer drawn,

integer transparentcolor])

The **DrawPicture** function will draw a picture from the picture list of the current applet. The picture to be drawn is specified by the *picturenumber* argument. Picture numbers are assigned when pictures are imported into an applet. The upper left corner of the picture to be drawn will appear at the coordinates specified by the *left* and *top* arguments.

On a system with a screen driver capable of displaying more than 256 colors, pictures will be drawn using the colors of their sources images. On systems with limited color screen drivers, pictures will be dithered or remapped to the common screen saver palette. Set the *drawn* value to 0 for photorealistic pictures, or to 1 for drawn images.

If the *transparentcolor* argument is included in a call to **DrawPictures**, any pixels in your source images which have the color specified in *transparentcolor* will not be drawn, and whatever is behind them will show through. The colors must match exactly for transparency to work. Displaying picture with transparency is somewhat slower than displaying a solid image. In addition, note that transparency may not work as you expect on systems with screen drivers capable of displaying 256 or fewer colors.

End()

The End function terminates a running applet.

FillRect(integer left, integer top, integer right, integer bottom)

The **FillRect** function draws a solid rectangle in the area defined by its *left*, *top*, *right* and *bottom* arguments, using the current brush color.

FillRoundRect(integer left, integer top, integer right, integer bottom, integer xround, integer yround)

The **FillRoundRect** function draws a solid rectangle in the area defined by its *left*, *top*, *right* and *bottom* arguments, using the current brush colour. The rectangle will have rounded corners. The corner will be drawn as arcs *xround* pixels wide by *yround* pixels deep.

FillEllipse(integer left, integer top, integer right, integer bottom)

The **FillEllipse** function draws a solid ellipse in the area defined by its *left*, *top*, *right* and *bottom* arguments, using the current brush colour.

FillPie(integer left, integer top, integer right, integer bottom, integer startangle, integer endangle)

The **FillPie** function draws a solid pie-shaped segment of the ellipse bounded by the rectangle defined by its first four arguments, using the current brush colour. The arc begins at the angle defined by *startangle* and ends at the angle defined by *endangle*. The three o'clock position is represented by zero degrees, and increasing angles rotate counterclockwise. Angles are specified in degrees, ranging from zero through 359.

FillPolygon(array points, integer count)

The **FillPolygon** draws a solid filled polygon using the current brush colour. The points of the polygon are defined by the array passed as the first argument. Each point is stored

as two consecutive elements in the array. The first element is the horizontal coordinate and the second element is the vertical coordinate. The *count* argument defines the number of elements in the array, that is, it's twice the number of points in the polygon. If the first and last points in the array are not identical, the polygon will draw an additional line to close itself.
The value of *count* must be 4 or greater.

FrameEllipse(integer left, integer top, integer right, integer bottom)

The **FrameEllipse** function draws a hollow ellipse in the area defined by its *left*, *top*, *right* and *bottom* arguments, using the current pen.

FramePie(integer left, integer top, integer right, integer bottom, integer startangle, integer endangle)

The **FramePie** function draws as hollow pie-shaped segment of the ellipse bounded by the rectangle defined by its first four arguments, using the current pen. The arc begins at the angle defined by *startangle* and ends at the angle defined by *endangle*. The three o'clock position is represented by zero degrees, and increasing angles rotate counterclockwise. Angles are specified in degrees, ranging from zero through 359.

FramePolygon(array points,integer count)

The **FramePolygon** draws a hollow polygon using the current pen. The points of the polygon are defined by the array passed as the first argument. Each point is stored as two consecutive elements in the array. The first element is the horizontal coordinate and the second element is the vertical coordinate. The *count* argument defines the number of elements in the array, that is, it's twice the number of points in the polygon. If the first and last points in the array are not identical, the polygon will draw an additional line to close itself.

FrameRect(integer left, integer top, integer right, integer bottom)

The **FrameRect** function draws a hollow rectangle in the area defined by its *left*, *top*, *right* and *bottom* arguments, using the current pen.

FrameRoundRect(integer left, integer top, integer right, integer bottom, integer xround, integer yround)

The **FrameRoundRect** function draws a hollow rectangle in the area defined by its *left*, *top*, *right* and *bottom* arguments, using the current pen. The rectangle will have rounded corners. The corner will be drawn as arcs *xround* pixels wide by *yround* pixels deep.

GetArrayValue(array name, integer index)

The **GetArrayValue** function will return the value stored in the array element specified by *index*.

GetFixedColor(integer number)

The **GetFixedColor** function will return the color of the entry of the standard screen saver palette specified by *number*. The value of number must be between 0 and 255. The colors returned by **GetFixedColor** will match the colors used in the Header, Image and Text blocks of Screen Saver Construction Set.

GetHorizontalPos()

The **GetHorizontalPos** function returns the horizontal value of the current drawing position.

GetPictureDimensions(integer picturenumber, integer variable xvariable, integer variable yvariable, integer variable bitsvariable)

The **GetPictureDimensions** function will store the dimensions in pixels and the bit depth of a picture in the variables passed to it. The picture to be measured is specified by the *picturenumber* argument. Picture numbers are assigned when pictures are imported into an applet. The value stored in *bitsvariable* will be 1, 4, 8 or 24.

GetPixel(integer left integer top)

The **GetPixel** function returns the color of the pixel on your screen at the location defined by *left* and *top*.

GetSaverColor()

The **GetSaverColor** returns the screen saver background color, as specified in the Header block of your screen saver.

GetScreenBits()

The **GetScreenBits** function returns the color depth in bits of your screen driver.

GetScreenDepth()

The **GetScreenDepth** function returns the depth in pixels of your screen.

GetScreenWidth()

The **GetScreenWidth** function returns the width in pixels of your screen.

GetSysColor(integer index)

The **GetSysColor** function returns the color of a Windows system element. System colours are defined through the display applet of the Windows Control Panel. The color returned is defined by the argument passed to **GetSysColor**, as follows:

· Scroll bar	0
· Background	1
· Active caption	2
· Inactive caption	3
· Menu	4
· Window	5
· Window frame	6
· Menu text	7
· Window text	8
· Caption text	9
· Active border	10
· Inactive border	11
· Application work space	12
· Highlight	13

·	Highlighted text	14
·	Button face	15
·	Button shadow	16
·	Grey text	17
·	Button text	18
·	Inactive caption text	19
·	Button highlight	20
·	3D dark shadow	21
·	3D light	22
·	Information text	23
·	Information background	24

Passing values to GetSysColor other than those specified here may produce unpredictable results.

GetSystemTime()

The **GetSystemTime** function returns the number of seconds which have elapsed since January 1, 1970, based on the current setting of the system clock. This value is not affected by year-2000 issues, but the maximally-paranoid should note that it will wrap back to January 1, 1970 some time in July of the year 2116. If you write any screen savers which you feel will still be popular a century from now, be sure to add a warning to this effect to their documentation.

GetTextDepth(string text)

The **GetTextDepth** function will return the depth in pixels of the text passed to it, based on the current font.

GetTextLength(string text)

The **GetTextLength** function will return the width in pixels of the text passed to it, based on the current font.

GetVerticalPos()

The **GetVerticalPos** function returns the vertical value of the current drawing position.

GoSub(label subroutine)

The **Gosub** function will suspend execution of the current applet and call the subroutine specified by the argument passed to it. When the subroutine terminates by calling the Return function, execution of the applet will resume with the line immediately following the **GoSub**.

GoTo(label jump)

The **GoTo** function will jump to the label passed as its argument and continue executing the current applet from there. **GoTo'**s are considered to be the tools of unclean spirits, and will lead you into the hell of funky programmers, from whence no soul has ever returned – something worth keeping in mind.

If(integer state, label jump)

The **If** function will jump to the label passed as its second argument if its first argument is TRUE, that is, non-zero.

Inc(integer variable a, [integer value])

The **Inc** function will increment the value of the integer variable passed as its first argument. By default, it will add 1 to the integer variable. If you pass a second argument to **Inc**, it will add the value of this argument.

IsEqual(integer a, integer b)

The **IsEqual** function will return TRUE if *a* equals *b*, or FALSE otherwise.

IsGreaterThan(integer a, integer b)

The **IsGreaterThan** function will return TRUE if *a* is greater than *b*, or FALSE otherwise.

IsGreaterThanOrEqualTo(integer a, integer b)

The **IsGreaterThanOrEqualTo** function will return TRUE if *a* is greater than or equal to *b*, or FALSE otherwise.

IsLessThan(integer a, integer b)

The **IsLessThan** function will return TRUE if *a* is less than *b*, or FALSE otherwise.

IsLessThanOrEqualTo(integer a, integer b)

The **IsLessThanOrEqualTo** function will return TRUE if *a* is less than or equal to *b*, or FALSE otherwise.

IsNotEqual(integer a, integer b)

The **IsNotEqual** function will return TRUE if *a* does not equal *b*, or FALSE otherwise.

IsSoundPlaying()

The **IsSoundPlaying** will return TRUE if a sound is currently playing, or FALSE otherwise.

KillSound()

The **KillSound** function will stop the currently playing sound, if there is one. If no sound is playing, it does nothing.

Line(integer left, integer top, integer right, integer bottom)

The **Line** function draws a line from the point defined by *left*, *top* to the point defined by *right*, *bottom*, using the current pen. The current drawing position is set to *right*, *bottom*.

MakeColor(integer red, integer green, integer blue)

The **MakeColor** function returns a color value defined by the red, green and blue values passed to it. The arguments must be between 0 and 255. The value returned by **MakeColor** can be used as an argument for any ANIMAL function which expects a color.

Max(integer a, integer b)

The **Max** function returns the greater of its two arguments.

Min(integer a, integer b)

The **Min** function returns the lesser of its two arguments.

Mod(integer a, integer b)

The **Mod** function returns the remainder of *a* divided by *b*. Rumors to the effect that **Mod** actually draws bell-bottom jeans on your screen and says "groovy" a lot are spurious.

Mul(integer a, integer b)

The **Mul** function returns *a* multiplied by *b*.

Not(integer state)

The **Not** function returns the opposite state of its argument. If *state* is TRUE, **Not** returns FALSE. If *state* is FALSE, **Not** returns TRUE.

Or(integer a1, integer a2)

The **Or** function returns the logical OR of its two arguments. If either argument is TRUE, the return value will be TRUE – otherwise it will be FALSE.

ParagraphText(string text, integer left, integer top, integer width, integer justification, [integer margin])

The **ParagraphText** function draws formatted paragraph text using the current font and text colour. The text passed to **ParagraphText** can include carriage returns. It will be drawn in a rectangle having its upper left corner in the location specified by the left and top arguments, and a width defined by the width argument. The depth of the rectangle will be whatever is required to draw all the text.

ParagraphText can use one of the following justification types:

Left	0
Center	1
Right	2

The margin argument defines the additional margin area around the the text, in pixels.

The **ParagraphText** function returns the depth of the rectangle which contains the text.

PlaySound(integer soundnumber)

The **PlaySound** function will play a sound. The sound to be played is specified by the *soundnumber* argument. Sound numbers are assigned when sounds are imported into an applet. Only one sound can play at a time – if you call **PlaySound** while a sound is playing, the current sound will be terminated and the new sound played.

Pow(integer a1, integer a2)

The **Pow** function returns *a1* raised to the power of *a2*.

Print([variable arguments])

The **Print** function displays text on your screen at the current drawing position. It updates the current drawing to the end of the displayed text position when it's done. It uses the current font and text color.

You can pass any number of integer and string arguments to **Print**. The arguments will be printed in the order they're passed.

The **Print** function only prints single lines of text, and it ignores carriage returns.

Random(integer limit)

The **Random** function returns a random number between 0 and *limit*-1.

RestoreGraphicState()

The **RestoreGraphicState** function restores the text and drawing parameters saved by a previous call to **SaveGraphicState**. It's essential that every call to **SaveGraphicState** have a corresponding call to **RestoreGraphicState**.

RestoreScreenArea(integer left, integer top)

The **RestoreScreenArea** function paints a previously saved screen fragment to the screen at the location defined by left and top. **RestoreScreenArea** must be called after a corresponding call to **SaveScreenArea**. To balance a call to **SaveScreenArea** without actually painting anything to your screen, call **RestoreScreenArea** with arguments that will cause it to paint an area beyond the edge of your screen.

Return()

The **Return** function returns from a subroutine called through **GoSub**.

SaveGraphicState()

The **SaveGraphicState** function stores the current drawing position, text colour, back color, text background transparency mode, brush, pen and font. You can subsequently change some or all of these values. Upon calling **RestoreGraphicState**, all the save values will be restored. You can nest calls to **SaveGraphicState**, but it's essential that every call to **SaveGraphicState** be balanced by a corresponding call to **RestoreGraphicState**.

SaveScreenArea(integer left, integer top, integer right, integer bottom)

The **SaveScreenArea** function copies the screen area defined by its arguments and stores it. The **RestoreScreenArea** function can be used to paint a saved area to the screen. Every call to **SaveScreenArea** must be balanced by a call to **RestoreScreenArea.**

SetArrayValue(array name, integer index, integer value)

The **SetArrayValue** function will set the element of an array specified by *index* to *value*.

SetBrushColor(integer color)

The **SetBrushColor** function will set the current brush to the color specified by its argument.

SetDrawingOffset(integer left, integer top)

The **SetDrawingOffset** function will set the current drawing offset to *left* and *top*. The drawing offset values area added to the coordinates of everything ANIMAL draws to your screen. The default values area zero.

SetFont(string name, integer size, integer bold, integer italic, integer underline)

The **SetFont** function sets the current text font. The current font will be used in subsequent calls to **Print**, **GetTextDepth**, **GetTextLength**, and **ParagraphText**.

The *name* argument can be the name of any font installed in your system. However, keep in mind, if you plan to distribute your screen savers, that the fonts you use will also have to have been installed on any other machines that will run your screen savers. Restricting your selection of fonts to Times New Roman, Arial, WingDings, Courier New and other common Windows fonts is less likely to have your screen savers behave oddly when they display text on someone else's computer. If *name* specifies a font that doesn't exist, Windows will choose a default font, usually Arial or Courier.

The *size* argument defines the size of the font in points. For practical purposes, one point is equivalent to one pixel on your monitor. Professional typographers will usually get all red in the face and pop a few minor blood vessels if you suggest this within their hearing, but it's nearly true. The value of *size* should not be less than 4, and probably several points larger still if you actually expect people to be able to read your text without a microscope.

Set *bold*, *italic* and *underline* to 1 to enable these effects, or to 0 to disable them.

SetPenColor(integer color, integer thickness)

The **SetPenColor** function will set the current drawing pen to the color and thickness specified in its arguments. The value of *thickness* should be one or greater.

SetPixel(integer left, integer top, integer color)

The **SetPixel** function sets the pixel on your screen at the location specified by *left* and *top* to *color*.

SetPosition(integer left, integer top)

The **SetPosition** function sets the current drawing position to *left* and *top*.

SetTextBackColor(integer color)

The **SetTextBackColor** function sets the current text background color.

SetTextForeColor(integer color)

The **SetTextForeColor** function sets the current text foreground color.

SetTextTransparent(integer transparent)

The **SetTextTransparent** function sets the text transparency mode. Pass it an argument of 1 to have subsequent calls to **Print** and **ParagraphText** draw text with no background color, or an argument of 0 to have the background of text drawn in the current text background color.

ShiftLeft(integer a1, integer a2)

The **ShiftLeft** function returns *a1* bitwise shifted left by *a2*.

ShiftRight(integer a1, integer a2)

The **ShiftLeft** function returns *a1* bitwise shifted right by *a2*.

Sleep(integer milliseconds)

The **Sleep** function will cause your applet to do nothing for the number of milliseconds passed to it. The **Sleep** function checks the state of your screen saver while it's sleeping – if the screen saver closes while it's sleeping, it won't stay asleep.

StrCat(string variable, string text)

The **StrCat** function adds the text in its second argument to the text in the string variable passed as its first argument. Having nothing much to do with felines, this process is properly called "concatenation."

StrCmp(string s1, string s2)

The **StrCmp** function returns TRUE if the two strings passed to it are identical, or FALSE otherwise.

StrCmpI(string s1, string s2)

The **StrCmpI** function performs a string comparison like **StrCmp**, except that it ignores the case of the strings in question.

StrCpy(string variable, string text)

The **StrCpy** function copies the text in its second argument to the string variable passed as its first argument. Any text previously stored in the string variable will be discarded.

StrFromChar(string variable, integer c)

The **StrFromChar** function creates a one-character long string which holds a character having the ASCII value *c*, and stores the string in the string variable passed as its first argument.

StrFromInt(string variable text, integer c)

The **StrFromInt** function creates a string which holds a text respresentation of the number *c*, and stores the string in the string variable passed as its first argument.

StrGetChar(string text, integer index)

The **StrGetChar** function returns the ASCII value of the character at location *index* in the string passed as its first argument. If *index* is greater than the length of the string, it returns the ASCII value of the last character in the string.

StrGetTime(string variable text)

The **StrGetTime** function will get the current system time and date and store it as text in the string variable passed to it. The string will always be 24 characters long, and time will be formatted as follows:

- Thu Jul 15 01:10:30 1999

StrLen(string text)

The **StrLen** function returns the number of characters in the string passed to it.

StrLwr(string variable text)

The **StrLwr** function will set all the alphabetic characters in the string variable passed to it to lower case.

StrSub(string variable destination, string initialstring, integer start, integer end)

The **StrSub** function will copy a section from the text passed to it as its *initialstring* argument beginning with the character position specified by *start* until the character position specified by *end*, and place the new string in *destination*.

StrUpr(string variable text)

The **StrUpr** function will set all the alphabetic characters in the string variable passed to it to upper case.

Sub(integer a1, integer a2)

The **Sub** function will return *a1* minus *a2*.

While(integer state)

The **While** function designates the end of a Do / While loop. As long as the value of the argument passed to **While** is TRUE – non-zero – the loop will repeat. Every call to **Do** must be balanced by a call to **While**.

NOTE: In addition to whichever string variables you create, two "magic" strings exist for any running applet. The string variable *$_OWN* contains the name of the owner or user of the computer upon which the screen saver in question is running. The string *$_ORG* contains the name of the company that owns the computer. These names are supplied when Windows is installed.

Header

Every screen saver created by Screen Saver Construction Set has a header block as the first block in its list. You can edit the Header block by double-clicking on it. The Header block editor provides access to the following items:

Background: This is the colour used to blank your screen. In most cases, you'll want to leave this at the default colour, black. While you can choose any colour for a screen saver background, the brighter ones could be said to defeat the purpose of saving your monitor from unnecessary damage.

Password Protect on Open: Enable this item to password protect your screen saver files. Password-protected screen savers will display correctly, but they cannot be opened in Screen Saver Construction Set without the password you assign to them. This will prevent other users from opening and modifying your screen savers. The password you assign to a screen saver Header block has nothing to do with the screen saver passwords assigned by users of screen savers under Windows.

Password: Enter a password in this field if Password Protect on Open is enabled.

Preview: When a screen saver is selected in the Windows Control Panel Display applet, a small version of the screen saver will appear. For the most part, Screen Saver Construction Set screen savers are too complex to make this practical. You can specify what you'd like to appear in the preview – either a scrolling line of text or a graphic. An example of whichever you choose, similar to what will appear in the Windows Control Panel, will be displayed.

If you select the Picture preview option, click on Select to import a picture.

Screen Saver Title: This is the text which will be displayed in the Control Panel Display applet if you select the Text Scroll preview option, above. It will also appear as the title of the Setup dialog for your screen saver in the Control Panel Display applet, should someone click on Setup.

About Dialog Text: Text entered into this field will appear in the Setup dialog for your screen saver. You can enter up to 1024 characters in this field. This is a good place to put copyright information about your screen saver, and other credits and details.

The Header block editor displays several bits of information about your screen savers, including who created them. Note that in a password-protected screen saver, this information will not be accessible, as no one else will be able to open your screen saver.

Image

A screen saver can include graphics as Image blocks. Graphics can also be displayed from within Applets, which are discussed elsewhere in this document.

The images for Image blocks are imported from image files. Screen Saver Construction Set will import images from JPEG, BMP, PCX, TGA and PNG files. In answer to the obvious question, Screen Saver Construction Set will not import files in the GIF format, nor is it likely to in the future. As GIF is an antiquated format with poor image quality and some serious patent issues, Screen Saver Construction Set does not include GIF support. If you have source images in the GIF format which you wish to import into a Screen Saver Construction Set document, we recommend that you use our Graphic Workshop Professional software to convert your pictures to a supported format, such as PNG or BMP.

Image blocks under Screen Saver Construction Set can display still images, and they can display them with animated transitions. Animated transitions can be used to make your graphics wipe in and out, appear as a sequence of tiles, split and otherwise jump around a bit.

The image in an Image block is displayed in three sequential processes – it transitions in, waits for a while and then transitions out. You can select a transition that doesn't do anything to make an image appear and disappear with no fanfare.

The following are the available effects for transitions.

None: No animation – your picture will appear, wait for a while and then vanish.

Chaos and Order: You picture will appear as a mosaic of random tiles which will gradually arrange themselves correctly. This effect does not have a corresponding transition out.

Horizontal Split: Your picture will split horizontally.

No Clear: You picture will appear, but will not be erased from the screen when its display time has elapsed.

Plummet: Your picture will appear as a sequence of vertical stripes, which will drop into the image area.

Quadrants: Your picture will animate into the image area as four sections. This effect does not have a corresponding transition out.

Random Effect: One of the other effects will be chosen at random.

Sandstorm: Your picture will appear as a cloud of pixels.

Tile: Your picture will appear as a mosaic of tiles.

Vertical Split: Your picture will split vertically.

Wipe in from Left: Your picture will wipe in from the left. That was pretty obvious…

Wipe in from Right. Your picture will wipe in from the right.

The Effect control group in the Image block editor controls the image transitions. The Transition In combo box selects the transition that precedes your image, and the Transition Out combo box selects the transition that follows your image. Other controls include:

Speed: The speed of the animation of your transitions.

Pause: The number of seconds your image will remain motionless between transitions.

Duration: The number of seconds that the image block being edited will remain active before it moves along to the next block in your screen saver. If this value is greater than the time required for your image to transition in, pause and transition out, your image will cycle through its transitions more than once. If you set the Duration field to one, your Image block will perform one cycle of its transition in, pause and transition one and then move on to the next block.

As discussed in greater detail in the Introduction and Tutorial document installed with Screen Saver Construction Set, it's important to keep in mind that your screen savers may run on computers having screen dimensions different from those of your computer. For this reason, images displayed on your screen area positioned in a screen saver are located relative to the center of your screen or to one of the corners, rather than at absolute coordinates. The Origin selector in the Image block editor allows you to determine what your image will be positioned relative to.

The Horizontal Offset and Vertical Offset fields will position your image. You can also position your image interactively. Click on Preview and click and drag your image around the Preview screen.

The Preview button will allow you to see your image, with its transitions, as it will appear in your completed screen saver. If the Close Preview after Duration item is selected in the Image block editor dialog, the Preview window will automatically close when the specified duration has expired. If this item is not selected, a beep will sound when the duration has expired, but your image will continue to cycle through its transitions.

The Paint button will open your image in a paint window, allowing you to edit it. By default, this will be Windows Paint. When you exist the paint window, your modified image will be replaced in your screen saver. See the discussion of Paint Applications, elsewhere in this document, for more about the Paint function. Note that the Paint function will change the image in your screen saver – it will not affect the source image file from which it was imported.

The Drawn item in the Image block editor dialog won't do much of anything it your computer has a screen driver capable of supporting more than 256 colours. It tells your screen savers how to handle images on machines with 256-colour drivers, should your screen savers find themselves on one. See the discussion of Palettes in the Reference document for more about this.

NOTE: Screen Saver Construction Set saves images with no compression, or with lossless ZLIB compression, as configured in its Setup dialog. Lossless compression will not degrade the quality of your images, and as such you can open and save a Screen Saver Construction Set screen saver as often as you like without adversely affecting the pictures therein. Lossless compression, however, is nowhere near as effective at compressing images as is the lossy compression used in JPEG files. If you import graphics from files in the JPG format for use in a screen saver, your resulting screen saver may become much larger than your source JPEG files. There's not much that can be done about this – it's just the way the universe works. Should you have occasion to contact us and inquire why Screen Saver Construction Set's files are so much larger than the JPEG files you imported pictures from, we will with the greatest of regret be compelled to dispatch our leather-winged demon of the night to rip out your lungs. Be warned.

Text

A screen saver can include text as Text blocks. Text can also be displayed from within Applets, which are discussed elsewhere in this document. The text in Text blocks can consist of single lines, single lines that scroll and formatted paragraph text.

Text blocks can include up to 65530 characters of text each – although if you use anything approaching this much text, it's unlikely that anyone looking at your screen savers will read it. You can set text in any font you like, with one minor catch. While Screen Saver Construction Set will use any font you tell it to, the font in question will have to be installed on whichever machines your screen savers run on as well. If you design a screen saver to use a font called Nuclear Accident Sanserif, and Nuclear Accident Sanserif doesn't happen to be available on the machines that ultimately run your screen saver, all your text will appear in another font – most likely Courier.

It's a really good idea to restrict your use of fonts to the ones which area installed with Windows – Times New Roman, Arial, WingDings and so on. If you need text in unusual fonts, consider setting it as a graphic using Windows Paint or GIF Construction Set Professional, and then displaying it with as an Image block, or from within an applet. If you use GIF Construction Set Professional's Banner function to handle display text, you'll have access to its various special effects, anti-aliasing and so on.

A Text block can display text using the following effects:

Single Line, Width of Text: Your text will displayed in a single line against a background the size of the text.

Single Line, Width of Screen: Your text will displayed in a single line against a background the height of the text and the width of your screen.

Scrolling Line, Width of Text: Your text will displayed in a single line against a background the size of the text. The text will scroll.

Scrolling Line, Width of Screen: Your text will displayed in a single line against a background the height of the text and the width of your screen. The text will scroll.

Multiple Line, Depth Set Numerically: Your text will be displayed in multiple lines, in a rectangle in which both the width and depth are defined.

Multiple Line, Depth of Text: Your text will be displayed in multiple lines, in a rectangle in which the width is defined and the depth is whatever it needs to be to enclose your text.

You can select the text effect for a Text block through the **Text Effect** combo box.

The **Speed** control below the Text Effect combo box controls the scrolling speed for those text effects which scroll. Text always scrolls from right to left.

The **Duration** control defines the number of seconds which the Text block in question will be active. After the number of seconds in the Duration field have elapsed, the Text block will terminate and the next block in your screen saver will become active. The text drawn by your Text block may be erased from your screen, depending on the setting of the Clear Text After Display option.

Text is drawn in a **Foreground Colour** against a **Background Colour**, both of which can be selected through the appropriate colour buttons in the Text block editor. If you select the **Use Saver Background** option, your text will be drawn against a background of whatever colour the screen saver background was set to in the Header block for your screen saver.

The background area of your text can be made larger than the text itself. Set the **Margin** control to the number of pixels by which you'd like it increased.

The **Justification** control will allow you to set the justification for the text displayed by your text blocks – you probably guessed that. Justification is not applicable to single line text set in an area the width of the text.

As discussed in greater detail in the Introduction and Tutorial document installed with Screen Saver Construction Set, it's important to keep in mind that your screen savers may run on computers having screen dimensions different from those of your computer. For this reason, text blocks displayed on your screen area positioned in a screen saver are located relative to the center of your screen or to one of the corners, rather than at absolute coordinates. The **Origin** selector in the Text block editor allows you to determine what your text will be positioned relative to.

The **Horizontal Offset** and **Vertical Offset** fields will position your text. You can also position your image interactively. Click on Preview and click and drag your text around the Preview screen.

The **Preview** button will allow you to see your text, with its effects, as it will appear in your completed screen saver. If the Close Preview after Duration item is selected in the Text block editor dialog, the Preview window will automatically close when the specified duration has expired. If this item is not selected, a beep will sound when the duration has expired, but your text will continue to cycle through its effect.

Sound

The Sound block editor is something of a misnomer, as it doesn't actually edit anything. It will tell you a bit about your sound blocks and let you preview them.

Sounds can be imported from WAV and MIDI files. WAV files store sampled sounds, and MIDI files store sequenced music. You can find archives of WAVE and MIDI files on the Internet, and there are a few examples stored in the \WINDOWS\MEDIA directory on your computer.

If your computer has a sound card and a microphone, you can record your own WAV files using the Sound Recorder application in the Programs / Accessories / Entertainment menu of the Windows Start button.

A screen saver can only play one sound at a time. This is true whether the sound is played through a sound block or an applet. If you start a new sound while an old sound is still playing, the old sound will be terminated.

The Pause While Sound Plays option in the Sound block editor will cause your screen saver to suspend operation until the sound in question has finished playing. This can take quite a while for MIDI files. Disable this option if you'd like to start a sound and then display whatever is in the next block of your screen saver – for example, to start some background music and then display a picture.

The Description field of the Sound block editor will allow you to assign some descriptive text to your sound blocks. This text appears in the document windows for Screen Saver Construction Set.

System Information

The System Information dialog will provide you with an overview of the configuration of your computer. This can be useful in unraveling configuration issues. It includes the following:

Screen Width and Screen Depth: These are the dimensions in pixels of your current screen display mode.

Network Present: If this field is set to Yes, Windows is of the opinion that your computer is connected to a network of some sort.

Slow Machine: This field indicates Windows' assessment of the speed of your computer.

Windows Version: This is the internal version number for your installation of Windows.

Machine Owner: This is the owner name used when you installed Windows.

Organization: This is the organization or company name used when you installed Windows.

Colour Depth: The last item in the System Information dialog indicates the number of colours which your current Windows screen driver can display without dithering. If this number is 256 or less, Screen Saver Construction Set is running a lot slower and looking a lot uglier than it needs to. See the Drivers document for help in reconfiguring your Windows screen driver.

Quick Colour Selector

The Quick Colour Selector function in the Applet editor will let you enter the RGB values for colour into your applet text without having to resort to colour tables or intuitive guessing. Click on the Quick Colour Selector button, select a colour and the three numbers which represent its RGB components will be pasted into your applet text at the current cursor location. Typically, the current cursor location should be within the brackets of a call to the MakeColor function.

In addition to selecting colours interactively, you can pick colours from the list of named colours in the Colour dialog. If you create a new colour, it can be given a name and added to this list -- click on Add.

By default, the Quick Colour Selector opens a file called SCSPRO.CCL to fetch its colour list. If you click on Open in the Colour dialog, you can use an alternate colour list. The ORTHO256.CCL colour list, also provided with Screen Saver Construction Set, includes all the colours in the default Screen Saver Construction Set palette.

CCL files are actually text documents -- you can open them with Windows NotePad. Be careful editing them.

Create Installer

Once you've created a completed screen saver, you'll probably need a way to distribute it. Screen saver SCR files can be installed by copying them to the \WINDOWS directory of the machine that will be using them, and thereafter configuring them through the Windows Control Panel. This can prove a bit difficult for some Windows users, however. It's also not as slick and professional as it might be. To this end, Screen Saver Construction Set can create clickable installers for your screen savers.

An installer is a "wrapper" for your SCR file. A complete installer will contain your screen saver, an optional document to describe what it does, an optional logo bitmap for the installer screen and the smarts required to unpack your screen saver, store it where it belongs and configure it automatically, without requiring that your users resort to the Windows Control Panel.

You can distribute the installers created by Screen Saver Construction Set as you would any other executable file. They lend themselves to being uploaded to software archives and other resources on the Internet, as they require virtually no instructions to operate them.

To create an installer, select Create Installer from the Screen Saver Construction Set File menu. Here's what the fields do:

Saver Background: When your installer is run, it will paint a gradient background in your choice of colours. Blue is traditional, magenta is only appropriate if your screen saver is entitled "Courtesan's Boudoir" and white looks unusual. Experiment with all of them.

Screen Saver: This is a path to the screen saver SCR file you wish to create an installer for. Click on Select to browse for a file. Screen Saver Construction Set is only able to create installers for SCR files it has created.

Document File: This is a path to a WordPad .DOC document file. This field is optional. If you use it, a document describing your screen saver will be displayed at the conclusion of the installation process. You can use this document to describe yourself, to include a shareware beg notice for your screen saver, to provide copyright information and so on. Be sure you use WordPad, not Microsoft Word, to create this document.

You might want to include the following items in your document:

- The document file will not be available once the installation is complete. Users who wish to refer to it in the future should select Save As from the WordPad File menu to save a copy.

- The screen saver installation does not include an uninstall function – a screen saver can be completely uninstalled by deleting it from the \WINDOWS or \WINNT directory of the machine it was installed on.

- If there are any image or sound credits or copyright restrictions applicable to the contents of your screen saver, this is the place to mention them.

- Should your screen saver have screen size or colour depth considerations, state them in this document. Something along the lines of "don't even think about running this screen saver on a funky 640 by 480 pixel, 16-colour garage-sale antique of a computer" might be appropriate.

Logo file: This is a path to an image file. This field is optional. If you use it, a logo or other graphic will appear in the upper left corner of your installer screen. The source image file can be in any format supported by Screen Saver Construction Set. It's a really good idea to keep your logo graphics small, and preferably using no more than 256 colours.

Description: This field can store a brief description of your screen saver. It will be displayed at the bottom of your installer screen. It will accept up to 260 characters. Note that by default, it will include your name and the name of your company or organization – you can edit this.

Save As: When you have the fields of the Create Installer dialog completed, click on Save As to save the completed installer to disk.

Preview: The Preview button will run your completed installer and show you what it looks like. An installer running in Preview mode is fully functional – if you click on Install in the first installer window, your screen saver will be installed on your machine.

NOTE: Like the screen savers created by Screen Saver Construction Set, installers created by an unregistered shareware version of Screen Saver Construction Set will only run on the system which created them.

The installers created by Screen Saver Construction Set are stand-alone EXE files, and they can be distributed without any ancillary DLLs, run-times or libraries. They'll run correctly under Windows 95, 98, NT 4.0 or better and 2000 – and presumably on later descendents of these as well.

The main screen of the installer will consist of a gradient of the colour you selected with the Saver Background button. The title of your screen saver, as defined in its header block, will appear near the top of the installer screen in embossed type. The Description message will appear near the bottom. The initial install dialog includes the following items:

Wait Time: This is the default wait time in minutes before the screen saver blanks your screen. The initial value is defined by the Saver Timeout value in the Setup dialog of Screen Saver Construction Set. Users are free to change this as they see fit, and it can be configured after installation through the Windows Control Panel.

Make This My Default Screen Saver: If this box is enabled, the screen saver will be installed and will become the current default screen saver. If it's not enabled, the screen saver will be installed, but the current screen saver settings in the Windows Control Panel will not be changed.

Preview: If you click on this button, the screen saver about to be installed will run just as it would were it the default screen saver on your system. The Preview button does not actually install anything, however.

Installers created with a registered copy of Screen Saver Construction Set are freely distributable under the same conditions as its screen savers are.

Registering Screen Saver Construction Set

Screen Saver Construction Set and all its ancillary applications are shareware. We invite you to evaluate the Screen Saver Construction Set package to see if it's suitable for your needs. If you like it, please register it. If it turns out not to be what you need, or if you don't feel it's worth registering, please delete it from your hard drive and accept our thanks for trying it out. If you do productive work of any sort with this software, you are no longer evaluating it.

If you like Screen Saver Construction Set and you find it useful, you are requested to register it. The current registration fee is $30.00 – plus $5.00 shipping if you'd like the latest version sent to you on CD-ROM. This will entitle you to technical support, notification of updates, a free copy of the latest version of this software and other worthwhile things. It will also avail you of a registration number to shut off the closing beg notice. More to the point, though, it'll make you feel good. We've not infested Screen Saver Construction Set with excessive beg notices, crippled it or had it verbally insult you after ten days. We trust you to support this software if you like it.

When you register Screen Saver Construction Set, you will be e-mailed your registration code as soon as your order is processed. Enter your code into a downloaded copy of Screen Saver Construction Set and it will become a registered copy, identical to what you would receive on CD-ROM. If you want to save the $5.00 shipping fee, select the NO SHIPPING option when you order. Note that absolutely nothing is shipped for NO SHIPPING orders – not even a receipt.

If you want to see additional features in Screen Saver Construction Set, register it. If we had an Arcturian mega-dollar for everyone who has said they'd most certainly register their copy if we'd add just one more thing to it, we could buy ourselves a universe and retire.

Oh yes – should you fail to support this program and continue to use it, a leather winged demon of the night will tear itself, shrieking blood and fury, from the endless caverns of the nether world, hurl itself into the darkness with a thirst for blood on its slavering fangs and search the very threads of time for the throbbing of your heartbeat. Just thought you'd want to know that.

Order On Line

ORDER IT ON LINE

You can order on line at our web page http://www.mindworkshop.com/alchemy/alchemy.html – **click on any of the blue key icons to access the secure server.**

Order by Phone

Call our **toll-free order desk** at **1-800-263-1138** from Canada and the United States. Call **0800-89-7355** from Great Britain. Call **1-800-554-082** from Australia. Callers from other countries can reach the order desk at **1-905-936-9500**. The order desk is available 24 hours a day, seven days a week.

Please note: the order desk only takes credit card orders. They cannot provide you with order status, modify or cancel previously submitted orders, provide product information or help you out with technical support. They cannot transfer you to someone who can.

Order by FAX

You can FAX your order to **1-905-936-9502**. Please use the order form included with this software.

Order by Snail-Mail

You can mail your order to:

Alchemy Mindworks Inc.
P.O. Box 500
Beeton, Ontario
L0G 1A0
CANADA

Please use the order form included with this software.

If you're paying by a cheque drawn on a bank outside North America, please be sure to make your cheque payable in US dollars. Your cheque must have the address of a North American clearing office and a bank transit number printed on it. Please do not send us Eurocheques – they cannot be cleared outside Europe.

Note that when you send us your order – and then when we send you your software – two distinct post offices get to deal with the ensuing mail. It can take a few weeks for things to get through this system – we ask that you be patient. We don't fully understand why it takes less time for a package to get to Australia than it does to send one to Cleveland. Some things are best left as mysteries.

Paying by Plastic

We can accept credit card payments by:

Registration Fee

Canadian users: The registration fee for Screen Saver Construction Set is $30.00 (CDN) plus seven percent GST, plus $5.00 shipping, or $37.10. We sincerely regret collecting this tax on behalf of several levels of government which will only squander it. If you sincerely regret having to pay it, we urge you to express your regret by voting in the next federal and provincial elections.

American users: The registration fee for Screen Saver Construction Set is $30.00 (US) plus $5.00 shipping.

Other users: The registration fee for Screen Saver Construction Set is $30.00 (US) plus $5.00 shipping.

Payment from countries outside Canada must be in US dollars. Please note that we are not able to accept purchase orders. We cannot ship software COD.

All orders are processed immediately. Please see the Terms, Conditions and Legal Dogma document included with this software for complete ordering details and all the stuff the lawyers made us say.

Note that Screen Saver Construction Set Classic registration codes will not work in Screen Saver Construction Set .

Contacting Alchemy Mindworks Inc.

Screen Saver Construction Set and all its ancillary applications are shareware. We invite you to evaluate the Screen Saver Construction Set package to see if it's suitable for your needs. If you like it, please register it. If it turns out not to be what you need, or if you don't feel it's worth registering, please delete it from your hard drive and accept our thanks for trying it out. If you do productive work of any sort with this software, you are no longer evaluating it.

The best way to get in touch with us is by e-mail to alchemy@mindworkshop.com. Except in conditions of flood, famine or mental-health days, our e-mail is answered within a few hours.

Technical support and customer service are available at 1-905-936-9501, 10:00am to 5:00pm EST most business days. If you encounter our answering machine – it does happen occasionally – please try back later. **One of the considerations in offering very low cost software is that we are unable to return calls for technical support.**

If you are an unregistered user of this software, we will at our discretion assist you to the extent required for you to ascertain whether this software is suitable for your application.

Callers who are rude, abusive or pig-headed with our technical support staff will wind up as demon-chow. See the section on Registration elsewhere in this document.

Please do not call our 800 number for technical support. The people who answer our sales line can put a credit card machine into warp drive, but they know less about software than most cats know about quantum mechanics. They are unable to transfer your call.

You can contact us by snail-mail at:

Alchemy Mindworks Inc.
P.O. Box 500
Beeton, Ontario
L0G 1A0
Canada

Our FAX number is **1-905-936-9502**.

We are unable to provide technical support by FAX or snail-mail.

The fabled Alchemy Mindworks BBS - poor, barnacle-encrusted antique that it was - has finally bit the dust. Alchemy Mindworks no longer maintains a BBS.

We ask that in contacting us you appreciate that we are a small company with limited resources. If you have not registered this software we will not tell you to go to hell, but please don't ask us for half an hour of free technical support. We have not built the price of technical support into the cost of Screen Saver Construction Set, as few users require it. We believe very strongly in not making everyone pay for something that only a small group needs. If our various governments felt the same way, our various economies wouldn't be in the midst of melting down.