

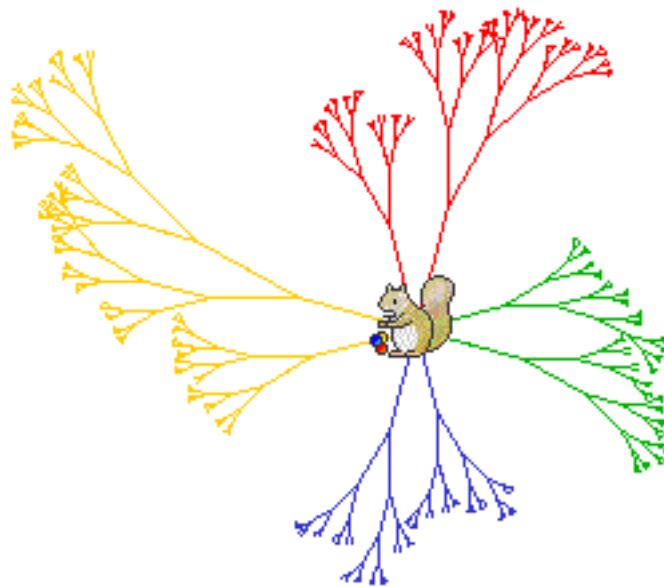
---

# SKIPPY Developer's Guide

Programming Reference

---

**Version 0.14**  
March 10, 2001





*SKIPPY Developer's Guide*

Copyright ©1999-2001 *Kirilla* . All Rights Reserved

No part of this manual may be reproduced or transmitted in any form, electronic or mechanical, for any purpose without the prior written agreement of *Kirilla* .

The contents of this document are furnished for informational use only; they are subject to change without notice and should not be understood as a commitment by *Kirilla* . *Kirilla* has tried to make the information in this document as accurate and reliable as possible, but assume no liability for errors or omissions.

*Kirilla* will revise often the software described in this document and reserves the right to make such changes without notification.

Author: Jean-Louis Villecroze

Email: [jl@kirilla.com](mailto:jl@kirilla.com)

Web site: <http://www.kirilla.com>

This document was prepared with  $\text{\LaTeX}$  2 $\epsilon$  .

$\text{\TeX}$  is a trademark of the American Mathematical Society

# Contents

<b>About this Developer's Guide</b>	<b>1</b>
<b>1 What is SKIPPY ?</b>	<b>2</b>
<b>2 SKIPPY Basics</b>	<b>3</b>
<b>3 The objects</b>	<b>7</b>
3.1 The Sheet widget . . . . .	7
3.2 The Skippy object . . . . .	8
<b>4 Release notes</b>	<b>16</b>
4.1 Version 0.14 . . . . .	16
4.1.1 Notes . . . . .	16
4.1.2 Changes . . . . .	16
4.1.3 Additions . . . . .	16
4.1.4 Bugs fixed . . . . .	16
4.2 Version 0.9 . . . . .	16
4.2.1 Notes . . . . .	16
4.2.2 Changes . . . . .	16
4.2.3 Additions . . . . .	17
4.2.4 Bugs fixed . . . . .	17
4.3 Version 0.8 . . . . .	17
4.3.1 Notes . . . . .	17
4.3.2 Changes . . . . .	17
4.3.3 Additions . . . . .	17
4.3.4 Bugs fixed . . . . .	17
4.4 Version 0.7 . . . . .	17
4.4.1 Notes . . . . .	17
4.4.2 Changes . . . . .	17
4.4.3 Additions . . . . .	17
4.4.4 Bugs fixed . . . . .	17

**Index**

**17**

# List of Figures

2.1	A Sheet widget and a Skippy . . . . .	4
2.2	Moving forward . . . . .	4
2.3	Goto . . . . .	5
2.4	Pen up/down . . . . .	6
2.5	Coordinate and heading . . . . .	6
3.1	Drawing an arc . . . . .	9
3.2	Drawing an ellipse . . . . .	10
3.3	Drawing a rectangle . . . . .	12
3.4	Drawing a round rectangle . . . . .	13
3.5	Drawing a string . . . . .	14
3.6	Drawing a triangle . . . . .	15

# About this Developer's Guide

This document covers the Add-on SKIPPY from the SQUIRREL programming language. This Add-On allows one to draw in a special widget.

At this time, neither SKIPPY nor this document is perfect. We would appreciate notification of any errors found.

This guide is divided into four parts:

**Getting started** introduces SKIPPY

**Squirrel basics** shows, mostly by examples, how to use SKIPPY

**Primitives and Methods** lists and describes all the primitives and methods

**Release notes** contains pertinent information on the releases

It should be understood that several additional features will be included in upcoming releases.

We have included several documentation conventions in this document. These are:

- All code elements are presented in a distinct font like `print "foo"`
- Primitive syntax is often a combination of code element and italic font. The part in italic is always the input to the primitive.
- Primitive inputs use special kinds of symbols :
  - **(word)** indicate that the input is optional
  - **word | number** indicate that the input can be either a word or a number
  - **(word)+** indicate that the primitive can take on several words as input, but at least one is required.
  - **(word)\*** indicate that the primitive can take on several words as input, but one is optional.

A big *Mahalo*<sup>1</sup> to Susan Banh<sup>2</sup>, for her much appreciated contributions towards rewriting and editing this document !

Please enjoy reading this manual and have fun with SKIPPY !

Jean-Louis, March 10, 2001

---

<sup>1</sup>"Thank you" in Hawaiian

<sup>2</sup>and all my love

# Chapter 1

## What is SKIPPY ?

SKIPPY to SQUIRREL is what *Turtle* is to Logo. SKIPPY is a simple tool which offers the possibility of drawing on a screen. The programming language SQUIRREL is used to program the drawing.

SKIPPY could help kids discover mathematics and computers in a way that is more entertaining than what is offered in a typical classroom setting. It can of course be used also to create images.

## Chapter 2

# SKIPPY Basics

SKIPPY is composed of two elements:

- a special widget called *Sheet*, in which the drawings are made on
- an object called *Skippy* which can draw on a *Sheet* widget or in an *Image* object.

Before we start drawing, we need to first create a *Sheet* widget and place it inside a window:

*Example 1*

```
1  make "win Window "titled 'Skippy' [100 100] "not.zoomable
2
3  make "sheet Sheet [200 200]
4
5  Glue :win "top [] :sheet
6
7  $win~show
8
9  make "sk Skippy :sheet
10 $sk~show true
```

A *Sheet* widget is created by calling the primitive *Sheet*. A mandatory input of a single list, stating its width and height is required for this primitive. The *Skippy* object is then created by the primitive *Skippy*. The only input to this primitive is the *Sheet* in which the *Skippy* will draw on.

On line 10, we use the *Skippy* method *show* to make it visible. A small triangle will then appear where the *Skippy* is located on the *Sheet*. The heading is indicated in this triangle. In our example, the heading is 0.

To draw on the *Sheet* we created, we need to use one of the methods from the *Skippy* object. For example, the *hop* method moves the *Skippy* forward by a certain distance:

*Example 2*

```
1  $sk~hop 20
```

A *Sheet* widget accepts more than one *Skippy* object and supports multi-threading. The distance 20 corresponds to 20 pixels.





Figure 2.1: A Sheet widget and a Skippy



Figure 2.2: Moving forward

It is possible to move a *Skippy* without actually drawing:

*Example 3*

```
1  $sk~goto 20 30
```

Or by instructing *Skippy* to not draw when moving:



Figure 2.3: Goto

*Example 4*

```
1  $sk~pen "up
2  $sk~hop 20
3  $sk~left 90 20
4  $sk~pen "down
```



Figure 2.4: Pen up/down

A *Skippy* could move to any coordinate and acquire any heading within a *Sheet*. The following image shows the coordinate system and heading:

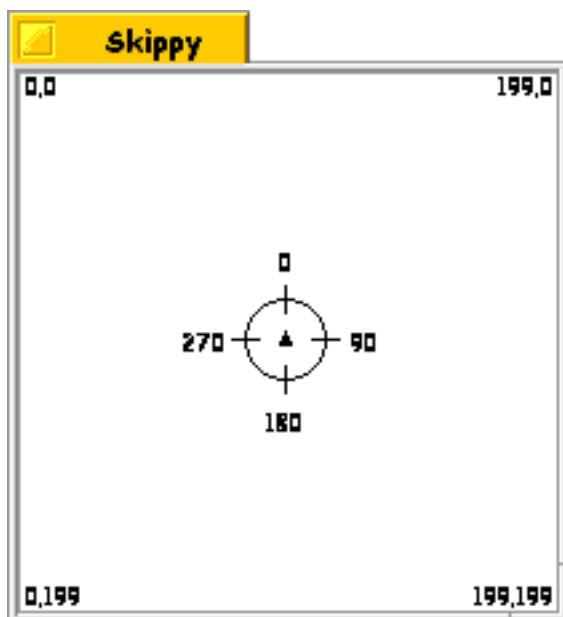


Figure 2.5: Coordinate and heading

## Chapter 3

# The objects

### 3.1 The Sheet widget

A *Sheet* is like any other widget of the GUI Add-On. It has the same configuration, hooks, and methods as common widgets. In addition, it has four specific methods:

#### **clear**

```
$sheet~clear (list list)
```

Clear the sheet widget if no input is given. If two lists are given as input, they describe the part of the widget we want to clear (left-top and right-bottom coordinate)

```
@> $sheet~clear [23 45] [67 95]
```

#### **export**

```
$sheet~export (word string) | image
```

Save a snapshot of the widget in an image file. The inputs can be either a word and a string, or an image object. The method outputs `false` if the export fails.

```
@> $sheet~export "GIF 'snap.gif'
```

#### **origin**

```
$sheet~origin word (list)
```

Get or set the origin of the coordinates. The first input is "get or "set. The second input (if given) is a list of 2 numbers.

```
@> $sheet~origin "set [200 200]
```

### skippies

```
$sheet~skippies
```

Output a list of all the *Skippy* objects currently attached to the *Sheet*.

```
@> print llength $sheet~skippies  
1
```

## 3.2 The Skippy object

A *Skippy* object is like any other SQUIRREL object. It could be cloned, deleted, or made into a variable.

Several methods are available:

### arc

```
$sk~arc word number number number number
```

If the first input is "fill", the method draws an arc (a portion of an ellipse) and then fills it in. If the first input is "stroke", the method just draws an arc. The center of the ellipse is the current position of the skippy. The second and third inputs specify the x and y radius, respectively. The arc will start at the angle given as the fourth input and will stretch along the ellipse for the angle given as the fifth input. The position of the skippy is not modified.

```
@> $sk~arc "stroke 20 20 0 180
```

### config

```
$sk~config word word (thing)
```

Set or get the configuration of the skippy. The first input must be set or get. The second input is the name of the configuration and the third input is specified only when setting a value to the configuration. When the first input is "get", the method outputs the value of the configuration.

```
@> $sk~config "set "size 5
```

### c.string

```
$sk~c.string (thing)+
```

Draw a string centered on the current position using the current heading as the orientation of the string.

```
@> $sk~c.string 'HELLO'
```

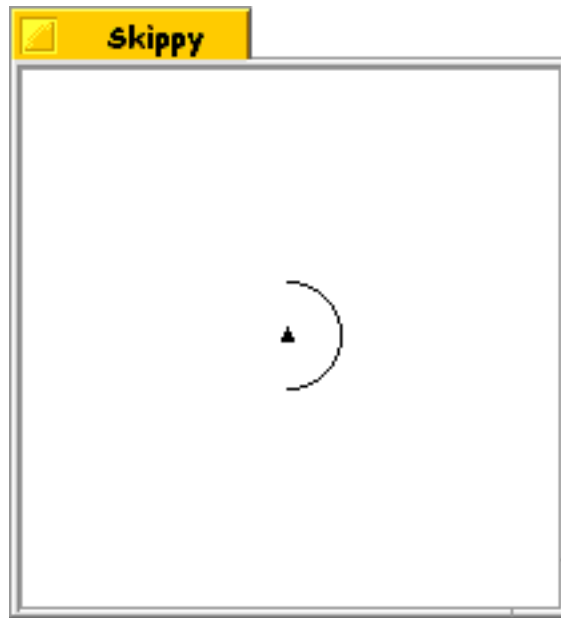


Figure 3.1: Drawing an arc

**c.draw**

```
$sk~c.draw image
```

Draw the image given as input, centered on the current position.

```
@> make "img Image 'an image.gif'
@> $sk~c.draw :img
```

**draw**

```
$sk~draw image
```

Draw the image given as input at the current position.

```
@> make "img Image 'an image.gif'
@> $sk~draw :img
```

**ellipse**

```
$sk~arc word number number
```

If the first input is "fill, the method draws an ellipse and then fills it. If the first input is "stroke, the method just draws an ellipse. The center of the ellipse is the current position of the skippy. The second and third inputs are the x and y radius, respectively. The position of the skippy is not modified.

```
@> $sk~ellipse "fill 20 50
```



Figure 3.2: Drawing an ellipse

**gohome**

```
$sk~gohome
```

Move the skippy back to its home position.

```
@> $sk~gohome
```

**goto**

```
$sk~goto ((number number) | list)
```

Move the skippy to a new location. The input could be a list of two elements (x and y) or two numbers (x and y). If no input is given, the method output the current position (in a list).

```
@> $sk~goto [10 10]
```

**hback**

```
$sk~hback number
```

Move the skippy backwards for a given distance. If the pen is active, this primitive draws a line from the current position to the new position.

```
@> $sk~hback 20
```

**heading**

```
$sk~heading (number)
```

Set the heading of the skippy if an input is given. If there is no input given, output the current heading.

```
@> print $sk~heading  
256
```

**hop**

```
$sk~hop number
```

Move the skippy forward a given distance. If the pen is active, this primitive draws a line from the current position to the new position.

```
@> $sk~hop 5
```

**left**

```
$sk~left number (number)
```

Turn the skippy left by the angle given as the first input. If a second input is specified, the skippy will also go forward the distance given by the second input.

```
@> $sk~left 90 5
```

**pen**

```
$sk~pen (word)
```

If the input is "up, the skippy will not draw when it moves. If the input is "down, the skippy will draw when it moves. If no input is given, the method outputs "up if the pen is up and "down if the pen is down.

```
@> $sk~pen "up
```

**rect**

```
$sk~rect number number
```

If the first input is "fill, the method draws a rectangle and then fills it in. If the first input is "stroke, the method just draws a rectangle. The center of the rectangle is the current position of the skippy. The second and third inputs are the width and height, respectively. The position of the skippy is not modified.

```
@> $sk~rect "stroke 30 40
```



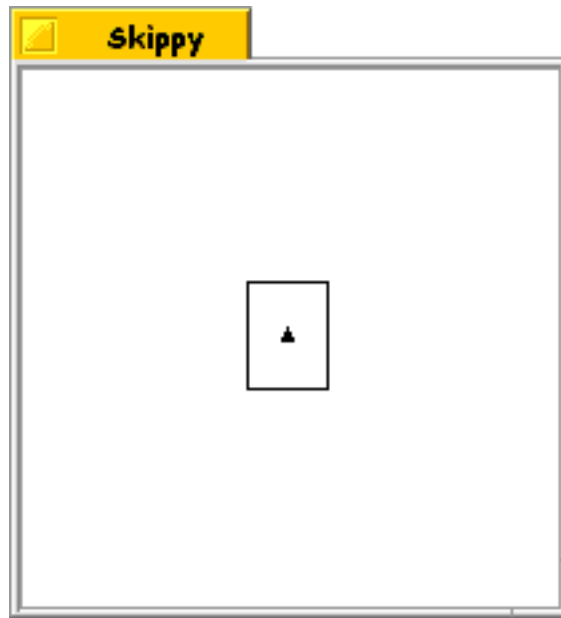


Figure 3.3: Drawing a rectangle

**right**

```
$sk~right number (number)
```

Turn the skippy right by the angle given as the first input. If a second input is specified, the skippy will also go forward the distance given by the second input.

```
@> $sk~right 30 10
```

**round.rect**

```
$sk~round.rect number number
```

If the first input is "fill", the method draws a rectangle and then fills it in. If the first input is "stroke", the method just draws a rectangle. This rectangle will have its corner rounded using the third and fourth inputs as x and y radius (huh???). The center of the rectangle is the current position of the skippy. The second and third inputs are the width and height, respectively. The position of the skippy is not modified.

```
@> $sk~round.rect "stroke 30 30 5 5
```

**show**

```
$sk~round.show boolean
```

If the input is true, the skippy will always appear. If the input is false, the skippy will not appear.

```
@> $sk~show false
```

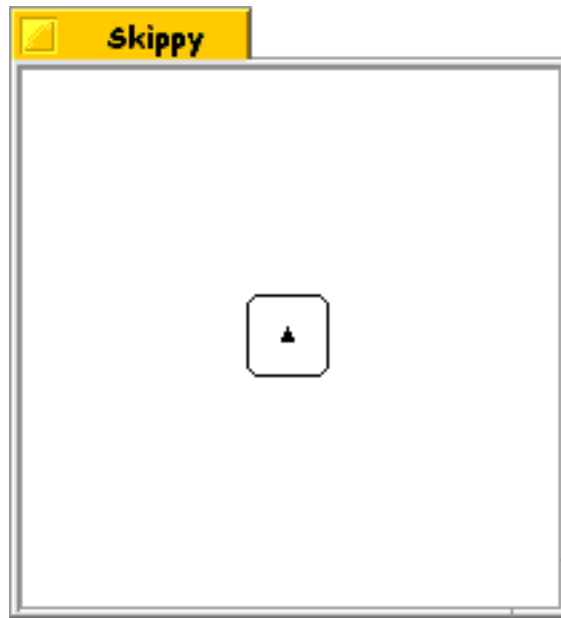


Figure 3.4: Drawing a round rectangle

**string**

```
$sk~string (thing)+
```

Draw a string from the current position using the current heading as the orientation of the string.

```
@> $sk~string 'HELLO'
```

**triangle**

```
$sk~triangle word number angle number angle number angle
```

If the first input is "fill", the method draws a triangle and then fills it in. If the first input is "stroke", the method just draws a triangle. The three corners of the triangle are specified by giving its angle and distance from the current position. The position of the skippy is not modified.

```
@> $sk~triangle "fill 0 30 90 30 0 0
```

**w.string**

```
$sk~w.string string
```

Compute and output the width (in pixel) used to draw the input with the current font.

```
@> $sk~w.string 'HELLO'
```

```
13
```



Figure 3.5: Drawing a string

**warp**

```
$sk~warp number number
```

Move to a new position like with the method `goto`, but draw a line from the previous position to the new one.

```
@> $sk~warp 40 23
```

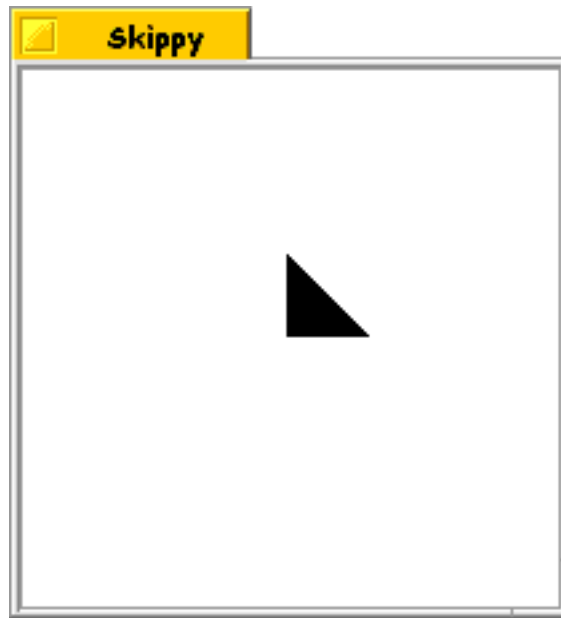


Figure 3.6: Drawing a triangle

The configuration of a *Skippy* allows one to set the color, font, size and home position.

Configuration	Purpose	Value
"color	Set or get the pen color	a color list.
"font	Set or get the font to use	a font object.
"high.color	Set or get the high color of the pen	a color list
"home	Set or get the home position	2 numbers or a list of 2 numbers
"low.color	Set or get the low color of the pen	a color list
"size	Set or get the size of the pen	a number

Table 3.1: Skippy's configuration

# Chapter 4

## Release notes

### 4.1 Version 0.14

#### 4.1.1 Notes

Minor update, support SQUIRREL 5.3.

#### 4.1.2 Changes

None.

#### 4.1.3 Additions

- Skippy can draw directly to an *Image* object.
- Methods `draw` and `c.draw` to Skippy.
- Method `origin` to the Sheet.

#### 4.1.4 Bugs fixed

None.

### 4.2 Version 0.9

#### 4.2.1 Notes

A small update.

#### 4.2.2 Changes

None.

### 4.2.3 Additions

- Method `warp`
- Method `w.string`

### 4.2.4 Bugs fixed

None.

## 4.3 Version 0.8

### 4.3.1 Notes

This version support the application *beSpotLight*.

### 4.3.2 Changes

- When used with no input, the method `goto` of *Skippy* output the current position.

### 4.3.3 Additions

- *Font* configuration to *Skippy* object.
- Method `c.string` that draw a string centered on the current position

### 4.3.4 Bugs fixed

None.

## 4.4 Version 0.7

### 4.4.1 Notes

This document is the first release of the rewritten *Skippy* in *SQUIRREL* . Most of the commands in the older version have been re-implemented and several new features have been added.

### 4.4.2 Changes

None.

### 4.4.3 Additions

None.

### 4.4.4 Bugs fixed

None.

# Index

## Sheet

### Methods

- clear, 7
- export, 7
- origin, 7
- skippies, 8

## Skippy

### Configuration

- color, 15
- font, 15
- high.color, 15
- home, 15
- low.color, 15
- size, 15

### Methods

- arc, 8
- c.draw, 9
- c.string, 8
- config, 8
- draw, 9
- ellipse, 9
- gohome, 10
- goto, 10
- hback, 10
- heading, 11
- hop, 11
- left, 11
- pen, 11
- rect, 11
- right, 12
- round.rect, 12
- show, 12
- string, 13
- triangle, 13
- w.string, 13
- warp, 14