# ThinkingSketch Cookbook

Version 2.00

ThinkingSketch Unit www.sketch.jp

2002年8月27日



🛛 1: ThinkingSketch Unit Members

# はじめに

ThinkingSketch は人間が絵をかくことを支援するソフ トウエアです。その一番の特徴は、コンピュータに図 形を発生させることです。

ThinkingSketchには、ただ単に、絵をコンピュータに 勝手に発生させるのではありません。描き出す図形の 内容に対して人間が注文をつけることにより、「自分 で描きたい図形に、より近いもの」を生成することが できる仕組みが備わっています。

この「User's Guide」はThinkingSketch をはじめて使 う方々に、ThinkingSketchの基本的な機能を紹介する ためのものです。

さあこれから、みなさんを ThinikingSketch の世界へ ご案内しましょう。

ThinkingSketch Unit 一同

# 第1章 準備しよう

4

ThinkingSketch を利用するためには Java 言語で書い たプログラムを実行できる環境が必要です。一番簡単 なのは、JRE という Java の実行環境をインストール することです<sup>1</sup>。ほかにも JDK をインストールするこ とでもできます<sup>2</sup>。

あなたの環境にまだ、Java がインストールされていな いのであれば、java.sun.com というウエッブサイトか ら最新のJRE またはJDK をダウンロードし、指示に したがってインストールを行ってください。

Java をインストールするために必要なデータのサイズは数メガバイトあります。インターネットからダウンロードされる際はそのことに注意してください。また、JRE は雑誌の付録として提供されていることもあります。

Java の実行が可能になったら、ThinkingSketch Unit が提供するCD-ROM もしくは http://www.sketch.jp/ から ThinkingSketch をダウンロードしてインストー ルしてください。

そのあと本書の指示にしたがって ThinkingSketch を 実行してください<sup>3</sup>。

<sup>&</sup>lt;sup>1</sup>この環境はプログラムは作れませんが Java の実行が可能です。

<sup>&</sup>lt;sup>2</sup>この環境は少し大きくなりますが、Javaのプログラム作成と実行がともに可能です。

 $<sup>^3</sup>$ run、ts といったコマンドを実行することによって ThinkingSketch は実行されるように なっているはずです。

# 第2章 試してみよう

2.1 スタートアップ

6

パッケージの解説したがってThinkingSketch を実行 すると、コンピュータのスクリーンは全面がThinkingSketchのウィンドウになります。

ウインドウの下の部分に細長い、長方形の部分があり ます。ここにキーボードからテキストを入力します。

キーボードから、文字列を入力してエンターキーを押 すとコマンドが実行されます。



図 2.1: 最初の画面

#### 2.2 モンドリアンをまねてみる

20世紀前半に活躍したオランダの画家、ピエト・モン ドリアン (1872-1944) をご存知ですか?ThinkingSketch を最初に使うにあたって、ひとまずモンドリアンが描 いたような抽象的なパターンを生成してみましょう。

ウィンドウ下の長方形の部分<sup>1</sup>にマウスポインタを移動し、キーボードから、

mondrian

という文字列を入力してみましょう。エンターキーを 押すと、長方形で構成された絵が画面に現れます。



#### 図 2.2: モンドリアンの作品に似させてつくってみた図

2.3 mondrian コマンド

図形が出たら、マウスカーソルでコマンド入力エリ アを選択した状態であることを確認し、続けてエン ターキーを何回か押してみてください<sup>2</sup>。

長方形が書き加えられ、画面が少しずつ変化してゆく のがわかりますか?

ここでは、mondrian というコマンドが実行され、赤や 青,緑などの色のついた長方形が書き加えられます<sup>3</sup>。

さらに短い mond というコマンド<sup>4</sup>を用いると、長方 形とともに、メッシュが現れます。ThinkingStechで は、この縦横の線を2本ずつ選ぶことで、色をつける 長方形の大きさや場所を決めているのです。

<sup>&</sup>lt;sup>2</sup>ThinkingSketch にはヒストリという機能があります。一度コマンドを実行していたなら、 文字を再入力せずにコマンド入力を行う領域に上矢印()や下矢印())キーを用いて過去の 入力履歴を呼び出してエンターキーを押すことにより再実行を行うことができます。

<sup>&</sup>lt;sup>3</sup>mondrian コマンドは ThinkingSketch 独自のプラグイン機能で、実現されています。画家 Piet Mondrian は緑を使わなかったようですが、そういうことにこだわらず、ここでは緑色の 長方形も表示するようなプログラムにしました。

<sup>&</sup>lt;sup>4</sup>mond は mondrian コマンドの説明のために用意されたコマンドです。



図 2.3: さらに描き加える



図 2.4: 補助線 (メッシュ)を描く

2.4 画面のクリア

一度描いた画面をクリアしてみましょう。

clear

というコマンドをキーボードから入力します。 画面が 初期の状態に戻ります。



図 2.5: 画面のクリア直後

## 2.5 部品の読み込み

ThinkingStetch では、絵を描くのに利用する部品を オブジェクトとして保管することができます。

コマンド入力エリアに以下のようなコマンドを入力 し、

import samples/birdfish.drw

を実行してみましょう。

この操作では、あらかじめ作成されファイルに保管されていたオブジェクト(絵の部品)がThinkingSketch に読み込まれます $^{5}$ 。

 $<sup>^{5}</sup>$ ThinkingSketch が読み込めないファイルであった場合には単に無視され、ThinkingSketch には何もおこりません。



図 2.6: samples/birdfish.drw を読み込んだところ

2.6 部品庫への保存

画面上にいくつかのオブジェクトが存在することを 確認してください。ファイルを読み込んだあとであれ ば、画面上にオブジェクトが見えます。

コマンド

storage

を実行します。このコマンドの実行で画面から図形は 見えなくなります。

この操作で画面に表示されていたいくつかの図形が、 個々の部品として「部品庫」に保管されます。



#### 図 2.7: 部品庫に部品を移動

# 2.7 コマンド \*stella の使用

部品庫においた部品を画面に配置することができま す。

\*stella

を実行してみましょう。

さきほど読み込まれた図形が大きさを変えて、ひとつ 画面上に現れます。\*stella コマンドは、部品庫にあっ た部品をランダムに選択した二本の垂直な線、二本の 水平な線で囲まれた長方形の領域に収まるように変形 して配置します。



#### 図 2.8: \*stella コマンドで部品庫から部品を移動

2.8 部品庫から複数のオブジェクトをコピーする

\* で始まる名前をもつコマンドは、一度の操作で部 品庫から複数のオブジェクトを移動させるように指定 できます。一度に何個のオブジェクトを移動させるか を repeat のパラメータとして指定します。たとえば、

repeat 5

を実行すると、それ以降 \*stella で生成する図形は一度に5個作られることになります。



図 2.9: repeat のパラメータを設定



図 2.10: 再度 \*stella を実行する

## 2.9 その他のコマンド、\*pollockの使用

部品庫においた部品を画面に配置することができます。これに関連したコマンドはいくつかあります。

\*stella

\*pollock

\*imai

\*mattise

などです。コマンドをクリア(clear)を実行しながら、 順に実行してみましょう。その違いがわかるはずです。

はじめに紹介した\*stella はオブジェクトの高さや幅に 変化がありました。\*pollock は、オブジェクトの位置 だけが乱数に基づいて配置され、サイズの変化はあり ません。



図 2.11: \*stella の実行例



図 2.12: \*pollock の実行例

# 2.10 \*imai, \*matisseの使用

\*imai

\*matisse

を実行してみました (図 2.13, 2.14)。\*imai は\*stella よりさらに大胆な配置を行うようにみえます。これは 変換する矩形の一辺を、枠の一辺上に配置するためで す。\*matisse は、配置する先の矩形の大きさを限定し ます。そのため派手さは感じられませんが繊細な印象 を与えます。



図 2.13: \*imai の実行例



図 2.14: \*matisse の実行例

## 2.11 \*matisseの繰り返し使用

\*matisse

を何度か実行してみました。さびしい感じの画面が、 デザインされたパターンに変化していきます。

コマンド実行の結果は乱数によって変化するので、繰 り返し使用やrepeatの変数を変えることにより、画面 が変化していくことを確認してみましょう。



図 2.15: \*matisse の一度だけの実行例



図 2.16: \*matisse を繰り返し実行した例

2.12 mask の使用

これまで紹介したコマンドは画面全体にオブジェクトを配置していました。

オブジェクトを画面全体ではなく、特定の矩形に対し て配置することが可能です。

mask 100 200 300 400

を\*matisse実行前に実行すると、2点(100,200)、(300,400)で囲まれた矩形の中にオブジェクトを配置します。



図 2.17: mask と\* matisse の実行例

# 2.13 画面と部品庫のクリア

一度描いた画面と部品庫を同時に初期化します。

reset

というコマンドをキーボードから入力します。画面が 初期の状態に戻ります<sup>6</sup>。

<sup>&</sup>lt;sup>6</sup>reset コマンドに対し、clear コマンドは画面の初期化のみを行い、部品庫は変化しません。



図 2.18: reset で画面と部品庫を空にする

# 第3章 部品を作ろう

3.1 プリミティブ

前章では、あらかじめ作った部品を配置しました。

今度は、オリジナル部品を作ってみましょう。

まずは、

line 100 200 300 250

を実行してみましょう。端点の座標が(100, 200)、(300, 250)の直線からなる部品が画面に現れます。実は、部 品をつくるのは、コマンドラインでの入力だけが唯一 の方法ではありません。マウスのようなポインティン グデバイスを使うことも可能です。右ボタンを押すと メニューが表示されます。



図 3.1: 直線をひく



図 3.2: ポップアップメニューからプリミティブを選択

3.2 壁紙の読み込み

壁紙として画像ファイルを読み込んで、ウィンドウ の背面に表示する機能が提供されます。この画像をな ぞり、図形を作り出すことができます<sup>1</sup>。まずは、

loadimage samples/photo.jpg

を実行してみましょう。画面上は何も変化しませんが、 次に、

wall

を実行することにより、画面上にファイルから読み込 まれた写真が現れます。

<sup>&</sup>lt;sup>1</sup> 画像としては JPG, GIF の形式のファイルのみが読み込めます。読み込みが出来ない場合 は上記の二つの形式に変更してください。



図 3.3: JPG イメージを読み込む



図 3.4: 壁紙として表示

3.3 トレースする

トレースに適したプリミティブはDrawCurv です。 curv

をキー入力するか、ポップアップメニューから curv を 選択します。次に、壁紙をなぞることで自由な形のプ リミティブを作成することが可能です。

wall clear

を行うと壁紙が見えなくなるので、どのようなオブ ジェクトが出来たのかを確認することができます。


図 3.5: トレースの実行



図 3.6: 壁紙を除くと作り出した部品が見える

3.4 色をつける

オブジェクトに色をつけることができます。

color red

color コマンドを実行することにより、色を指定するこ とができます<sup>2</sup>。色を指定したあとで、描くオブジェク トは指定の色になります。また、カラーパレットを使 うことも可能です。ポップアップメニューの「colorpalette」を選択することにより、標準のカラーパレッ トを表示することが可能です。

<sup>&</sup>lt;sup>2</sup>black, blue, cyan, darkGray, green, lightGray, magenta, orange, pink, white, red, yellow の各色が指定できます。三原色の RGB を 255 段階の明るさで指定することも可能で、 たとえば color 255 0 0 とすると赤が選択できます。



図 3.7: 色の変更



図 3.8: カラーパレット

## 3.5 壁紙からパレットを作る

壁紙から色を抽出してパレットを作ることが可能で す。ポップアップメニューの「update-palette」を選択 したあと、ポップアップメニューの「image-palette」 を選択することにより、標準のカラーパレットを表示 することが可能です。

この機能を用いて、古典の絵画をスキャナやデジカメから読み込み、巨匠の色使いを追体験することもできます。



図 3.9: image-palette の更新



図 3.10: 壁紙から作ったパレット

3.6 色以外の属性

fill属性を指定することにより、オブジェクトの中を 塗りつぶすか枠だけにするかの決定が行えます。

fill fill

または、

fill outline

のように指定します。



図 3.11: fill 属性の図形と outline 属性の図形

## 第4章 マクロを使おう

## 4.1 マクロとは

ThinkingSketch ではテキストファイルに書いたコマンドをまとめて実行する機能をマクロと呼びます。

こうしておくと1行ずつコマンドを入力する手間を省 くことができます。自分の気に入った手順(コマンド 列)が決まったら、マクロにしておくと便利です。

コマンド行で実行したいコマンドをテキストエディタ でまとめて書いて、ファイルにSAVEします一つのコ マンドは1行にまとめて書きます。<sup>1</sup>。

マクロは、「メモ帳」など好みのテキストエディタで 作成しておきます。右にあるのはこれから実行しよう とする「マクロ howa1.mac」の内容をメモ帳でみたも のです。

<sup>&</sup>lt;sup>1</sup>マクロはどんな拡張子をつけても実行可能になっていますが、マクロであることがわかるように拡張子を".mac"とすることをお勧めします。

✓ howa1.mac - メモ帳 フ= (川(C) (日本(C)) (日本(C))	a (1-901)).	
プデイルビー 編集ビー 書式(U) Ifclear		
colord 150 150 150		
repeat 400 repeat 1 1		
mask 60 50 360 250		
color red *rain		
movemask 300 0 color blue ≭rain		
movemask color green ≭rain		
mask 60 250 360 450 color yellow *rain		
movemask 300 0 color cyan *rain		
movemask color red *rain		
mask 60 450 360 650 color red *rain		
movemask 300 0 color black ≭rain		
movemask color green		
1		

図 4.1: howa1.mac マクロをメモ帳で見る

4.2 マクロ howa1.mac

右にあるのが今回実行するマクロの全体です<sup>2</sup>。空行 をも含めて、全体で40行あります。これから、この マクロについての解説を行います。

<sup>&</sup>lt;sup>2</sup>プログラムの中の一行の先頭にある数字はマクロにかかれているものではなく、解説のために加えたものです。

```
1: colord 150 150 150
 2:
 3: repeat 400
 4: mirror 1 1
 5:
 6: mask 60 50 360 250
 7: color red
 8: *rain
9:
10: movemask 300 0
11: color blue
12: *rain
13:
14: movemask
15: color green
16: *rain
17:
18: mask 60 250 360 450
19: color yellow
20: *rain
21:
22: movemask 300 0
23: color cyan
24: *rain
25:
26: movemask
27: color red
28: *rain
29:
30: mask 60 450 360 650
31: color red
32: *rain
33:
34: movemask 300 0
35: color black
36: *rain
37:
38: movemask
39: color green
40: *rain
```

マクロ "samples/howa1.mac"

1行目の colord は、繰り返しオブジェクトを生成し たり、部品個からオブジェクトをコピーしたりする場 合、色彩に変化を与えるためのものです<sup>3</sup>。

赤、緑、青の加色混合の3原色に対応する3つの数字 を0から255までの値で指定します。数字が多くなる ほど変化のばらつきが大きくなります。

colord <赤の変化量> <緑の変化量> <青の変化量>

3行目はrepeatにより一度に生成するオブジェクト の数を400個に指定しています4。

4行目は初期化手続きの一種です5。

<sup>&</sup>lt;sup>3</sup>これらは、乱数によって色調に関連したパラメータに変化を与えることによって実現されま

す。
 <sup>4</sup>指定されているパラメータが1つであることに注意してください。

<sup>&</sup>lt;sup>5</sup>多くの場合は必要ありませんが、安定した動作のために使用しています。簡略化のために、

ここでは説明を省略します。

```
1: colord 150 150 150
2:
3: repeat 400
4: mirror 1 1
5:
```

マクロ "samples/howa1.mac line 1-5"

6行目のmaskは、\*stella、\*pollock、\*imai、\*matisse、 などで部品庫から部品をコピーする際にコピー先の大 きさを矩形領域<sup>6</sup>の指定によって指定ができます。

mask <左> <上> <右> <下>

7行目のcolorは8行目の\*rainコマンドで生成する図 形の色を指定しています。redを指定することにより、 赤を中心に描画を行うという指定となります。

8行目の\*rainはデモンストレーション用のプラグインに含まれているコマンドで、左上方向から右下方向に直線を引きます。

10行目のmovemaskは現在指定されているマスクの矩形領域を平行移動させるためのものです。

movemask <左右方向の移動> <上下方向の移動>

11 行目、12 行目は今度は青系の色で直線を描くこと を指定します。

14行目のmovemaskも現在指定されているマスクの矩 形領域を平行移動させるためのものです<sup>7</sup>。

movemask

このあと、15行目、16行目は今度は緑系の色で直線 を描くことを指定します。

<sup>&</sup>lt;sup>6</sup>矩形というのは「なが四角」ということです。

<sup>&</sup>lt;sup>7</sup>パラメータとして移動量の指定がない場合は前回指定したパラメータがそのまま使われます。

```
6: mask 60 50 360 250
7: color red
8: *rain
9:
10: movemask 300 0
11: color blue
12: *rain
13:
14: movemask
15: color green
16: *rain
17:
```

18-28 行目は表示する位置が異なりますが、6-17 行 目のくり返しです。

これらの部分では3つの矩形領域への描画を行いま す。 18: mask 60 250 360 450
19: color yellow
20: \*rain
21:
22: movemask 300 0
23: color cyan
24: \*rain
25:
26: movemask
27: color red
28: \*rain

マクロの実行は exec または! コマンドを用います。 コマンドの後にファイル名を書きます。

! samples/howa1.mac

右ページはコマンドの実行例です。

画面の中に矩形領域を設定し、中心となる色を指定します。描画時には色彩を変化させながら、400本ずつ線を引くという作業を繰り返し9つの領域を塗りつぶします。



図 4.2: マクロ samples/howa1.mac の実行例

マクロの実行前にlinewidthコマンドを用いて、描 画される線の太さを変更します。

linewidth 10

この指定により、今後描かれる線の太さが10となります。



図 4.3: 線幅を変更する

再度、同一のマクロを実行してみます。

! samples/howa1.mac

実行結果が異なり、類似の図柄ながら画面の雰囲気も 大きく変わるのがわかります。

このように場合によってはコマンドとマクロを交互に 使って新しい図柄の検討を行うことが可能です。

このほかにも \*rain を \*boxer に変更したり、\*bubble に変更したらどうなるかを調べてみると生成される図 柄が変化します。



図 4.4: マクロ samples/howa1.mac の実行

4.3 マクロ igo.mac

乱数による配置をおこなっていても、規則性を感じ させる図形を生成することは可能です。

今回はその例の一つ samples/igo.mac を紹介します。 このプログラムは16 行で記述されています。

1: color black 2: 3: wmax 50 4: wmin 50 5: hmax 50 6: hmin 50 7: 8: gridw 50 9: gridh 50 10: 11: repeat 25 12: 13: fill fill 14: \*bubble 15: fill outline 16: \*bubble

マクロ "samples/igo.mac"

1行目は、色の指定です。黒を指定しています。

wmax、wmin、は\*bubble<sup>8</sup>で生成されるオブジェクト の横幅の最大値、最小値を指定するためのコマンドで す。最大、最小を50と指定しますので生成されるオ ブジェクトの幅は50となります。

hmax、hmin、は同様にオブジェクトの高さを指定する ことができます。オブジェクトの高さは50と指定し たことになります。

gridw、gridx、でオブジェクトの配置をおこなう格子 の大きさを指定することができます。ここではx軸方 向の格子、y軸方向の格子の大きさを50とします。

<sup>&</sup>lt;sup>8</sup>\*matisse、\*rain、\*boxer コマンドにも wmax、wmin、は影響を与えます。これは hmax, hmin, gridw, gridh についても同様です。

4.3. マクロ igo.mac

1: color black 2: 3: wmax 50 4: wmin 50 5: hmax 50 6: hmin 50 7: 8: gridw 50 9: gridh 50 10:

マクロ "samples/igo.mac line 1-10"

\*bubble は repeat で指定した数だけ、楕円を生成し ます。14 行目と16 行目で実行が行われます。

11 行目では一度に生成するオブジェクトの数を 25 個 とします。

13行目、15行目のfillコマンドはそれぞれ図形の中 を塗りつぶす、塗りつぶさないという指定変更を行い ます。

このプログラムが実行されたあとは 50 個のオブジェ クトが存在します。

4.3. マクロ igo.mac

11: repeat 25
12:
13: fill fill
14: \*bubble
15: fill outline
16: \*bubble

egreen all and the second state of the secon

## マクロを実行してみます。

! samples/igo.mac

右の図のように、一種のデザイン画が表示されました。



図 4.5: マクロ sampels/igo.macの実行

4.4 マクロ yanagi\_style.mac

ー定のテイストをイメージしてマクロを作った例を 示します。

右の作品は、ThinkingSketch Unitのメンバーが過去に作成したものです。



図 4.6: ThinkingSketch Unit メンバーの作品

前ページの作品のイメージを参考にマクロを作成しました。

このマクロは samples/yanagi\_style.mac として実 行可能です。右の図は、作成した yanagi\_style.mac の実行結果の例です。





マクロの内容は右のとおりです。

第1行目から第5行目までは、画面に表示すべき部品 を設定しています。

ここでは、もとのイメージをスキャンしてJPG形式と したファイルyanagi\_style.jpgを読み込み、colormap コマンドによってこのイメージを構成している色彩か らなる矩形を生成します。こうやって生成した図形は storageの実行によって部品庫に移されます。

7行目は部品を50個コピーすることを指定し、8行目 ではマスク(対象となる描画領域)の指定を行います。 9行目以降はマスクをずらしては\*stellaコマンドの 実行をおこなうという作業を繰り返しています。
```
1: // select color
 2: loadimage samples/yanagi_style.jpg
 3: colormap
 4:
 5: storage
 6: //
 7: repeat 50
 8: mask 100 50 150 660
9: *stella
10: movemask 60 0
11: *stella
12: movemask
13: *stella
14: movemask
15: *stella
16: movemask
17: *stella
18: movemask
19: *stella
20: movemask
21: *stella
22: movemask
23: *stella
24: movemask
25: *stella
26: movemask
27: *stella
28: movemask
29: *stella
30: movemask
31: *stella
32: movemask
33: *stella
```

マクロ "samples/yanagi\_style.mac"

この段階ではまだ、最終形とは言えません。

これから先は好みに合わせてプログラムを変化させ、 さらに、画面に深みをつけていくことになります。

また、終盤には画面を手で編集して微調整をおこなう ことも有効です。



## 図 4.8: さらに変化をつけた画面

## 第5章 よいこの作品

ここでは、いくつか作品例を紹介します。これらの作 品を参考にしながら、新たな作品作りを試みてみてく ださい。

右の作品は、東京都在住のTさんがThinkingSketch をはじめて用いた頃のものです。

単純ですが、力強いオブジェクトが画面に広がってい ます。



図 5.1: T さんの作品たち

次の作品は、「赤い森」です。作者は同じTさん。昔、 キャンバスにむかって描いた絵をイメージしてThinkingSketchを利用しました。



図 5.2: T さんの作品「赤い森」

右の作品は、matisseの絵をイメージした別の作者の ものです。matisseの絵をイメージしたものです。マ チス風の部品を作ったあと、\*pollock コマンドで配置 し、微調整をおこなったものです。

80



図 5.3: マチス風

春の景色をイメージしました。\*imaiと\*pollockを 使っています<sup>1</sup>。

<sup>&</sup>lt;sup>1</sup>ここでは、一様にオブジェクトを配置するのではなく、一定の場所に配置を行うためのパラ メータの指定を行っています。



図 5.4: 春の景色

## Update History

2002 年 4 月 30 日: repeat コマンドを変更。repeat コマンドが一つ引数 を持つときと二つ引数を持つときで異なる意味を持たせていたが、あらた に mirror というコマンドをつくり repeat <x-dup> <y-dup> という機能を mirror <x-dup> <y-dup> という機能として実現した。

2002 年 8 月 26 日: Version 2.0 Alpha へとバージョンアップを行った。今回の大きな変更点は(1) 複数のページにデータを置けるようになり、互いのページを部品庫として指定できるようにしたこと。(2) 半透明のオブジェクトを指定できるようにしたこと。ページ間の移り変わりを行うときに、デゾルビングの効果をつけたこと。の2つの機能拡張を行ったことにある。