

K_ET Cindy の概要

K_ET Cindy Project Team

2018 年 8 月 18 日

- 第 3.2 版 -

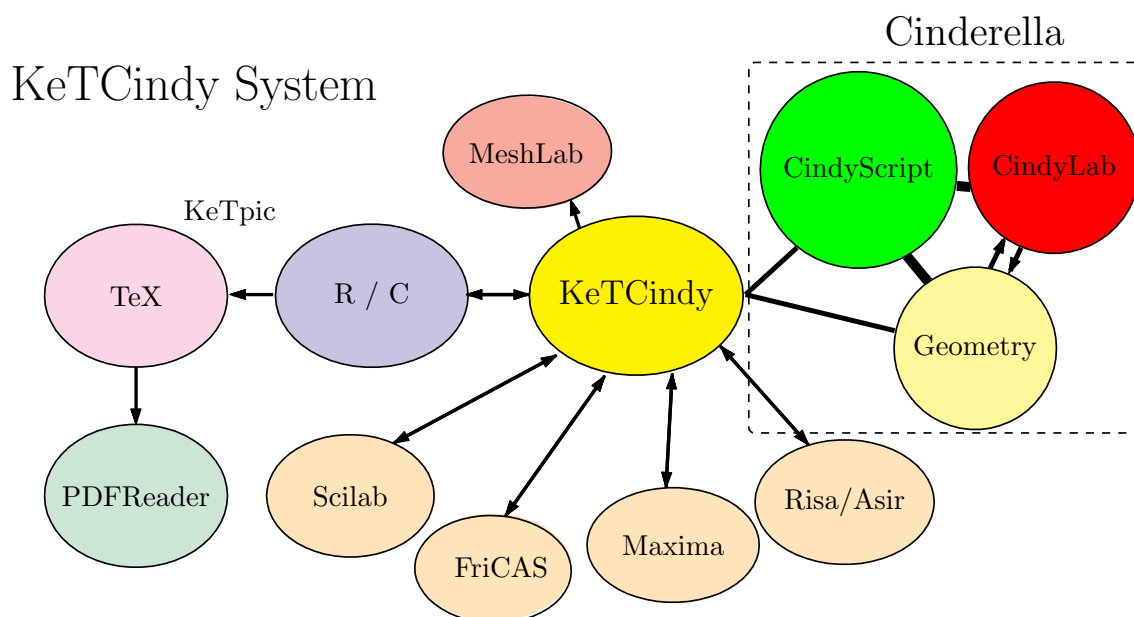
目次

1	K _E T Cindy について	2
1.1	システムの構成	2
1.2	K _E T Cindy による作図手順	3
1.2.1	平面幾何	3
1.2.2	関数のグラフ	5
1.2.3	空間図形	7
1.2.4	作表	8
1.2.5	他のソフトとの連携	8
2	プロットデータ	9
3	Cindyscript	10
3.1	Cindyscript エディタ	10
3.2	スクリプトの記述	11
3.3	変数と定数	12
3.4	よく使う Cindyscript のコマンド	14

1 KeTCindy について

1.1 システムの構成

KeTCindy は、Cinderella の作図機能を利用して、作図データの \LaTeX ファイルを作成するためのスクリプトライブラリである。KeTCindy は Cinderella のプログラミング言語 Cindyscript で記述されており、ユーザーは Cinderella によるインタラクティブな作図機能と、CindyScript によるプログラミングを用いて、 \LaTeX 文書の挿入図を効率よく作成することができる。また、各種数式処理ソフトと連携して計算を行うことができる。



Cinderella で作図した図のデータは、KeTCindy により、いったん R のファイルに書き出される。これを R で処理して \LaTeX ファイルを作成する。できた \LaTeX ファイルを、本文中に `input` コマンド で挿入すれば図が表示される。(KeTCindy の初期の版ではこのデータ処理に Scilab を用いていた。)

Cinderella と R やその他のソフトウェアとの連携には、バッチファイル (Mac ではシェルフファイル) を用いている。(概念図の両方向矢印) バッチファイルは `kc.bat`, シェルフファイルは `kc.sh` で、KeTCindy が目的に応じてこれらのファイルを書き出して実行するようになっている。

したがって、KeTCindy での図ファイルの作成手順は次のようになる。

- (1) 必要に応じて Cinderella の作図ツールで、点や線を作図しておく。
- (2) Cindyscript エディタでプログラムを書く。
- (3) 出力する。

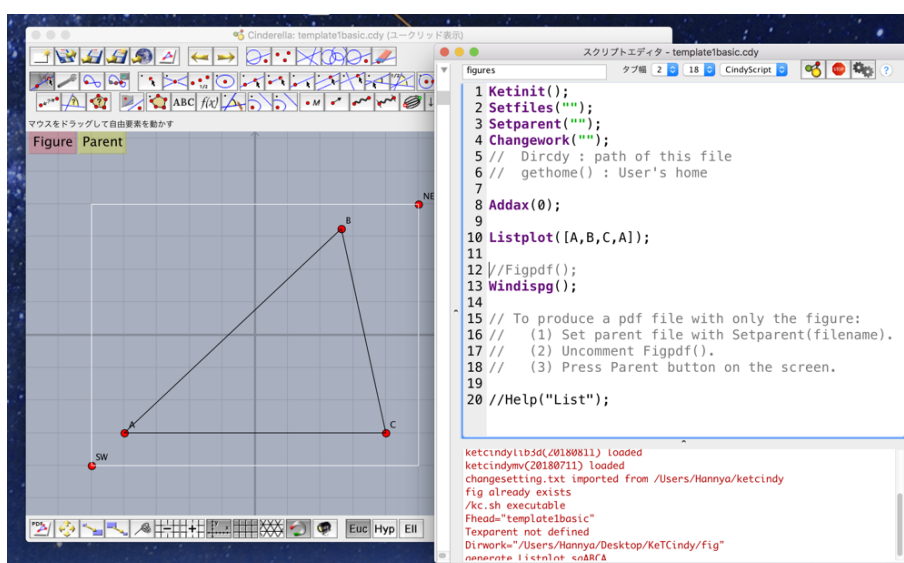
1.2 K_ETCindy による作図手順

K_ETCindy で作図し、TeX のファイルを作図する手順をチュートリアル形式で示す。

1.2.1 平面幾何


K_ETCindy のシステムに同梱されている、template1basic.cdy をひな形として用いる。適当な場所に複製を作り、名前を変えておこう。ここでは、単に template.cdy とする。

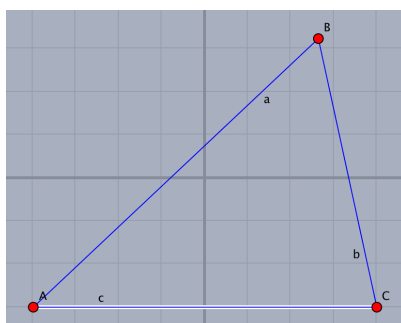
template.cdy をダブルクリックして開き、図が表示されたら、Ctrl+9 (Windows) / ⌘+9 (Mac) でスクリプトエディタを開く。2つの画面はマルチウィンドウにしておくのがよい。




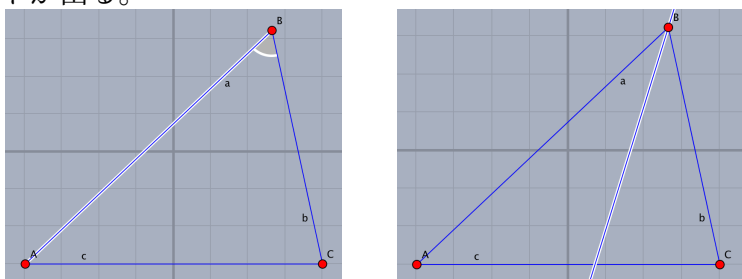
この三角形で Cinderella の作図機能を用いて内心を求め、内接円を描く。


まず、スクリプトエディタの Listplot([A,B,C,A]); の行頭にスラッシュを2本書き入れ、Shift+Enter で実行する。こうすると、この行はコメント行となり実行されない行になる。その結果、三角形が消えて点だけ残る。

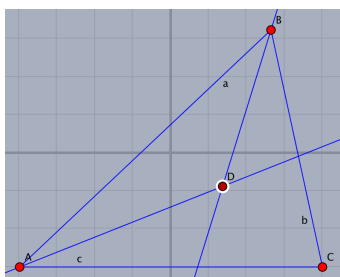
画面上部の作図ツールから「線分を加える」 をクリックして選択し、点 A から点 B へドラッグすると線分が描かれる。同様にして、BC, CA を引く。




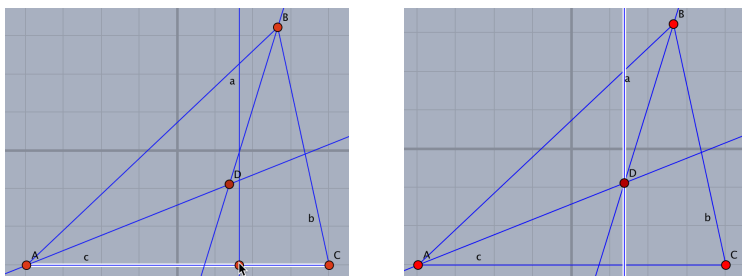
次に、角の二等分線を引く。「角の二等分線を加える」ツール  を選択し、辺 BA, BC を順にクリックすると角 ABC の二等分線が引かれる。このとき、次図左のように、該当する角を表すガイドが出る。



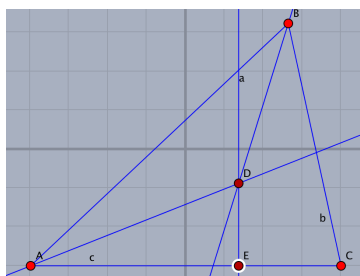
同様にして、角 A の二等分線を引き、「交点を求める」ツール  をクリックして、2本の二等分線を順にクリックすると交点が求められる。(角 A の二等分線を引いた直後はこれが選択状態にあるので、角 B の二等分線をクリックすればよい)




内接円の半径を決めるために、辺 AC に垂直で点 D を通る直線を描く。「垂線を加える」ツール  を選択し、辺 AC 上でマウスボタンを押し、そのまま点 D へドラッグすると垂線が引ける。



最後に、垂線と辺 AC の交点を求める。

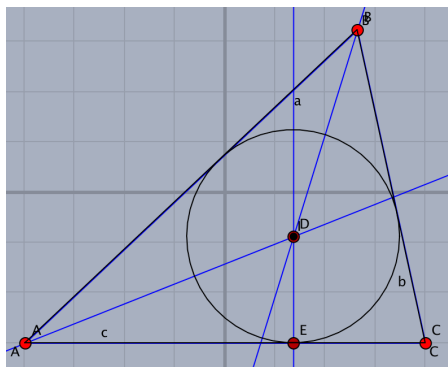


図が描かれたら、「要素を動かす」ツール  を選択して、始めの状態（動かすモード）に戻しておく。

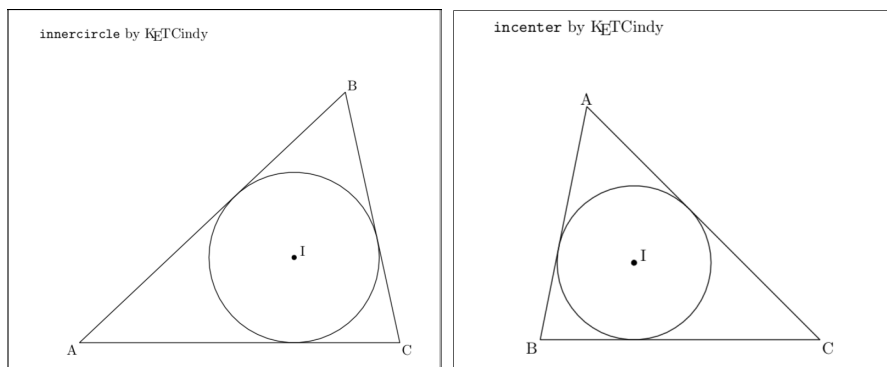
これで内心の作図と半径の作図ができた。内心円は描かなくてよい。

スクリプトエディタに戻り、先ほど書いた `//` を消して `Listplot([A,B,C,A]);` を有効にし、次のスクリプトを追加し、`Shift+Enter` で実行すると、内接円が描かれる。なお、2行目を `Setfiles("innercircle")` として、ファイル名も設定しておく。

```
Circledata([D,E]);
Pointdata("1",[D],["size=3"]);
Letter([A,"sw","A",B,"ne","B",C,"se","C",D,"ne2","I"]);
```



画面左上の「Figure」ボタンをクリックすると、プレビュー用のPDFができて表示される(下図左)。描画面で点Bをドラッグして三角形の形を変えれば、それに応じて出力する図も変えることができる(下図右)。



KfTCindy が TikZ などの作図支援ツールと異なるのは、Cinderella の作図機能を用いてインタラクティブに図の調整ができる点である。簡単な図であれば、座標の計算は不要で、Cinderella の作図画面を見ながら KfTCindy の作図関数でプログラムを書くだけでよい。

なお、Cinderella の作図機能については、付録の [作図ツール](#) を参照されたい。

1.2.2 関数のグラフ

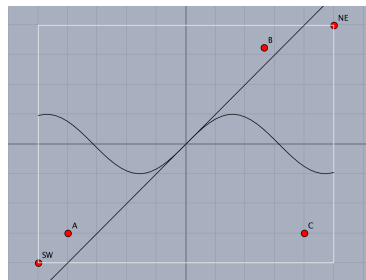
例として、 $y = \sin x$ と $y = x$ のグラフを描く。

template.cdy をダブルクリックして開き、`Ctrl+9` (Windows) / `⌘+9` (Mac) でスクリプトエディタを開く。

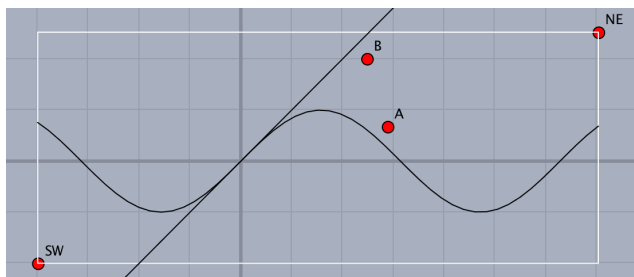
座標軸を描くので、`Addax(0);` を `Addax(1);` に変え、`Listplot([A,B,C,A]);` は使わないので削除し、かわりに次のスクリプトを書く。

```
Plotdata("1","y=sin(x)","x");
Plotdata("2","y=x","x");
```

これだけで右のようにグラフが描かれる。



描画範囲は点 NE と SW をドラッグして適当に決めよう。また、点 A,B,C が残ったままだが、これを関数名を表示する場所として利用するために適当な位置にドラッグして移動する。



関数名と x 軸上の交点を表示するために、関数 `Expr()` を使って次のように書く。

```
Expr([A,"e","y=\sin x",B,"e","y=x",[-pi,0],"s2","-\pi",[pi,0],"s2","\pi",
[2*pi,0],"s2","2\pi"]);
```

注) 改行せず1行に書いてよい。

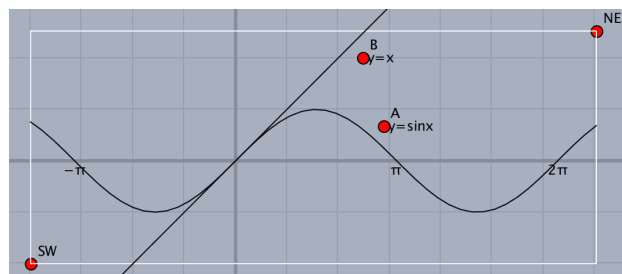
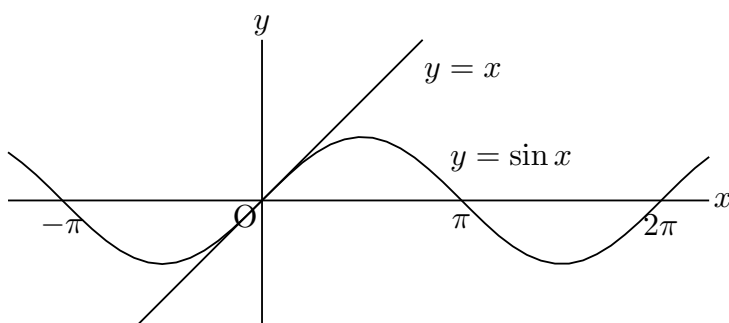


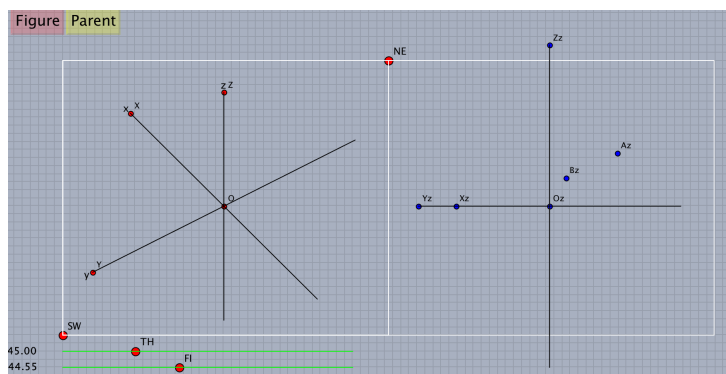
Figure ボタンをクリックすると、次の図が描かれる。



1.2.3 空間図形

KETCindy のシステムに同梱されている samples フォルダから, s05spacefigure フォルダを開き, s0501basic.cdy をひな形として使う。適当な場所に複製を作り, 名前を変えておくとよい。ここでは, 3Dbasic.cdy として進める。

3Dbasic.cdy を開くと次のような画面になる。



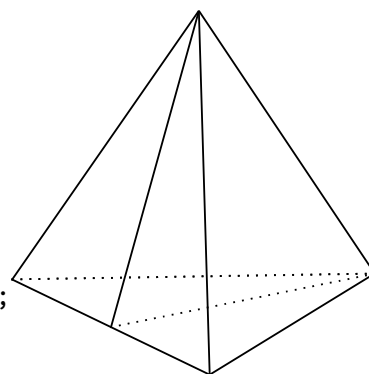
下のスライダで点 TH,FI をドラッグすると, 空間での視点の位置が変わる。(座標軸が回転する)

ここでは, 正四面体と, 高さを求めるのによく使われる補助線を描いてみよう。スクリプトエディタを開き, 次の3行を消す。

```
Ch=[1];
if(contains(Ch,1),
    Skeletonparadata("1",[1.5]);
);
```





かわりに次のスクリプトを書いて Shift+Enter で実行する。

```
Putpoint3d("A",2*[-1,-1/sqrt(3),0],"fix");
Putpoint3d("B",2*[1,-1/sqrt(3),0],"fix");
Putpoint3d("C",2*[0,sqrt(3)-1/sqrt(3),0],"fix");
Putpoint3d("D",2*[0,0,sqrt(3)],"fix");
Putpoint3d("M",(B3d+C3d)/2,"fix");
phd=Concatobj([ [A,B,C], [A,B,D], [A,C,D], [B,C,D] ]);
Spaceline("1",[D,M,A]);
VertexEdgeFace("1",phd,["Edg=nogeo"]);
Nohiddenbyfaces("1","phf3d1");
```



1.2.4 作表

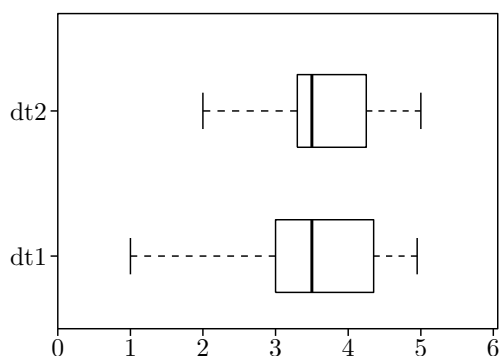
$\text{T}_{\text{E}}\text{X}$ で表を作るのはかなり面倒だが、 $\text{K}_{\text{E}}\text{T}_{\text{C}}\text{indy}$ を使えば比較的簡単に作表ができる。次の図は関数の増減・凹凸表である。以下は紹介にとどめる。関数リファレンスに例を掲載しているので参照されたい。

x	\cdots	-1	\cdots	0	\cdots	1	\cdots
y'	$+$	$+$	$+$	0	$-$	$-$	$-$
y''	$+$	0	$-$	$-$	$-$	0	$+$
y		$\frac{1}{\sqrt{e}}$		1		$\frac{1}{\sqrt{e}}$	

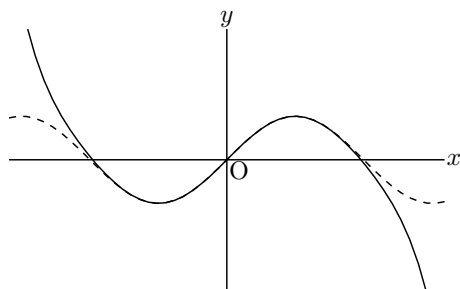
1.2.5 他のソフトとの連携

$\text{K}_{\text{E}}\text{T}_{\text{C}}\text{indy}$ は Cindyscript で記述されているが、Cindyscript はプログラミング言語であり、数式処理ソフトではない。そこで、R や Maxima などと連携することにより、機能を拡張することができるようになっている。統計計算は R、数式処理を用いた計算は Maxima を利用すると便利である。

【例】R を用いて箱ひげ図を描く



【例】Maxima を用いて $\sin x$ の 7 次のテイラー展開を行う。



2 プロットデータ

プロットデータ (Plot Data) とは、関数のグラフや幾何要素を描くデータのことである。K_{ET}Cindy では PD と略すことがある。

たとえば、線分は端点の座標 2 つからなるリストで表現できる。曲線は、描画範囲を分割して線分の集まりとして描画しており、このときのプロットデータはそれらの線分の端点のリストである。

プロットデータの名称は K_{ET}Cindy が次の規則により命名する。

- ・ 名称の頭部は、プロットデータを作成する関数ごとに決まっている。
- ・ 第 1 引数に name が与えられる場合、name を頭部に付加する。

【例】Listplot("1",[[0,0],[1,2]]); のとき, sg1

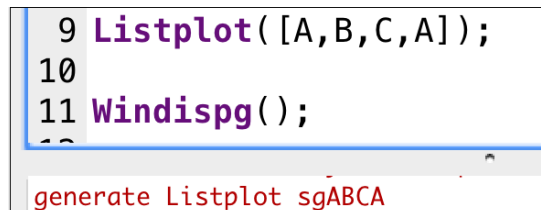
- ・ 第 1 引数の name を省略できる場合、引数で用いられた点の名前を頭部に付加する。

【例】Listplot([A,B,C]); のとき, sgABC

プロットデータを生成したときは、Cindyscript エディタのコンソールにその名称を表示する。たとえば、Listplot([A,B,C,A]) で三角形 ABC を描くと、コンソールに

```
generate Listplot sgABCA
```

と表示される。プロットデータを操作する関数では、この名称を用いる。



The screenshot shows a code editor with the following text: Line 9: Listplot([A,B,C,A]); Line 10: Line 11: Windisp(); Below the editor, the console output shows: generate Listplot sgABCA

プロットデータの内容は、Cindyscript の println() 関数を用いてコンソールに表示することができる。たとえば上記の場合、

```
println(sgABCA)
```

とすると、

```
[[1,3],[-1,0],[3,0],[1,3]]
```

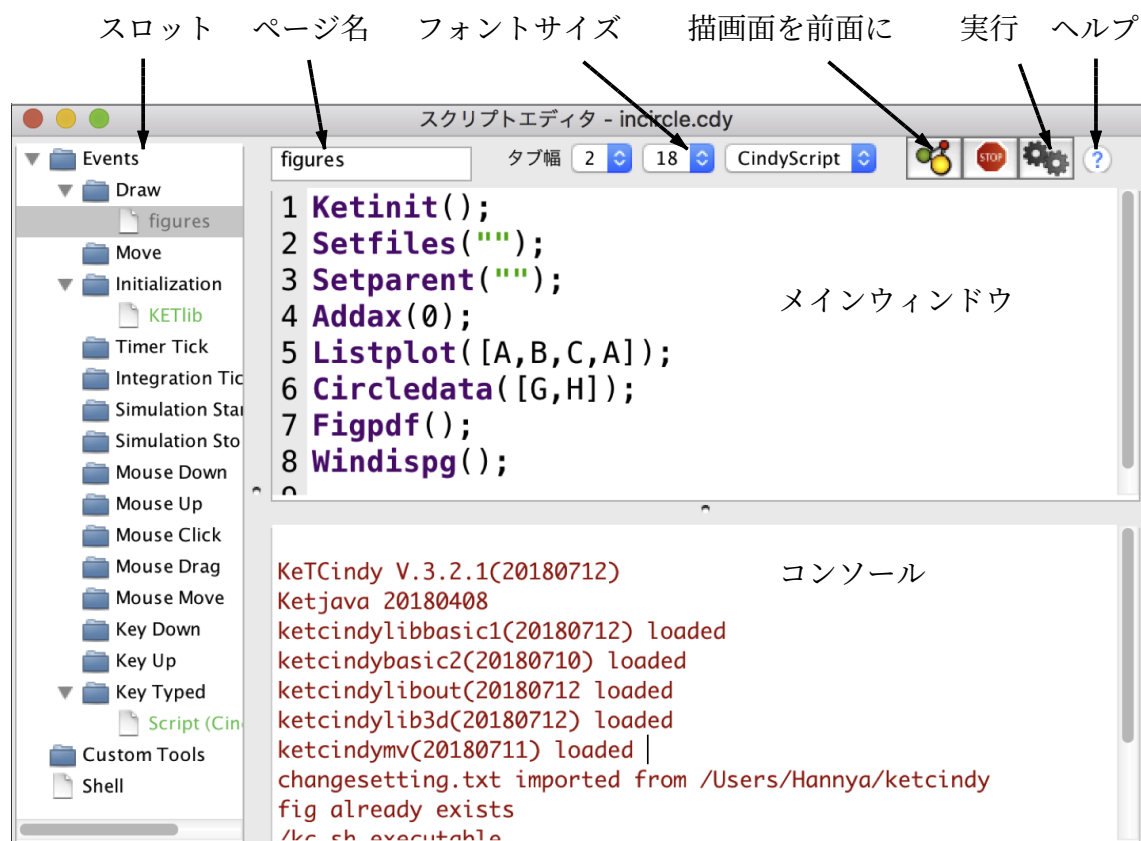
と表示される。A,B,C,A のそれぞれの座標からなるリストである。

プロットデータは、Cindyscript によるプログラムで作成してそれを K_{ET}Cindy で利用することもできる。(Listplot() の例 を参照) ただし、要素の数が大きいとエラーとなるので、1 つのプロットデータの要素は 200 程度とするのがよい。これより多い場合は分割する。

3 Cindyscript

3.1 Cindyscript エディタ

Cindyscript は Cinderella のプログラミング言語で、Cinderella 上のスクリプトエディタで記述する。スクリプトエディタは、「スクリプト」メニューの「Cindyscript」を選択するか、Ctrl+9 (Windows) / ⌘+9 (Mac) で開く。



スロットはそれぞれの実行タイミングでスクリプトを実行するものであり、他のプログラミング言語にはない特徴である。(スロットが隠れているときは境界線をドラッグする)

よく使うスロットは次の通り。

- | | |
|----------------|--|
| Draw | 描画面になにか変化が起きる（点を動かすなど）と実行される。
通常はここにスクリプトを書く。ひな形の templatebasic1.cdy では、Ketinit() などが記述された figures ページが用意されている。 |
| Initialization | スクリプトを実行すると、始めに 1 度 だけ実行される。
したがって、関数定義や変数の初期設定などを書く。
ひな形の templatebasic1.cdy ではここに KETlib というページがあり、 |

KeTCindy の初期設定に関する記述がある。

Key Typed

キーボードが押されたとき実行される。

KeTCindy では、ボタンによらずキーボードで出力を行うためのスクリプトが書かれている。

1つのスロットに複数のページを作ることができる。たとえば、KETlib 以外に初期設定のスクリプトを書く場合は、Initialization スロットのフォルダアイコンをクリックすることで新しいページができる。

KeTCindy の描画コマンドは Draw スロットに書く。

ページ名

各スロットでは、ページを分けて記述することができる。各ページの名前はスクリプトエディタの上の欄に書くことができる。

フォントサイズ

編集エリアのフォントサイズを変更する。

実行ボタン

プログラムを実行する。プログラムの実行は Shift+Enter でもできる。

ヘルプボタン

ブラウザを開いてマニュアルを表示する。

コンソール

print() 関数の結果やエラーメッセージが表示する。エラーメッセージは、「WARNING:」または「syntax error」に続いてその内容と該当する行番号が示される。これを読んでスクリプトの書き間違いをチェックする。

3.2 スクリプトの記述

編集エリアにプログラムを書くと、文字が色分けされて表示される。組み込み関数は青、ユーザー定義関数は紫、定義されていない関数は赤、文字列は緑で表示される。KeTCindy の関数はユーザー定義関数なので紫色で表示される。

編集エリアでは、Ctrl+C と Ctrl+V によるコピーアンドペースト、Ctrl+X と Ctrl+V によるカットアンドペーストができる。他のテキストエディタなどとの間でのコピーも同様にできる。

文字列の選択はマウスドラッグまたは Shift+ カーソルキーでおこなえる。

Ctrl+F による検索はできない。

スクリプトを記述するときの基本的なルールは次の通り。

- ・基本的に小文字で書く。大文字と小文字は区別される。

KeTCindy では、Cindyscript に組み込みの変数名・関数名と区別しやすいように、次の規則により名前を付けている。

- ・グローバルな変数はすべて大文字か、大文字で始まるものとする。
- ・局所変数は小文字で、関数定義の冒頭で `regional()` により局所変数として宣言する。
- ・関数名は大文字で始まる。
- ・複数の半角スペースは無視され、一つの半角スペースと見なされる。
- ・行末にはセミコロンを書く。改行だけでは命令文の終わりにならない。

3.3 変数と定数

変数

Cindyscript では、変数の型の宣言は不要。使用されたときに何が代入されたかで自動的に型が決まり、さらに、宣言し直さなくても異なる型の値を代入することができる。

【例】

```
a=10;
b=2;
c=a+sqrt(b);
a="の平方根";
println("10 に "+b+a+" を加えると"+c);
```

この例では、始めに `a` は整数型であるが、4 行目で文字列に変わる。

文字列はダブルクォートでくくる。異なる型の演算には注意を要するが、例外的に、5 行目のように、文字列と数を `+` 演算子で結ぶと、数は文字列化されて結合される。

予約定数

Cindyscript では、円周率 (`pi`) と虚数単位 (`i`) が定数として予約されている。`i` は、変数として使用することもでき、そのような場合、虚数単位に戻すには `i=complex(0,1)` を実行する。

KeTCindy の予約変数

KeTCindy が内部的に使用する予約変数がある。そのうち次のものはユーザーが値を変更または設定することができる。

Fhead 書き出されるファイル名の頭部。Setfiles() によって設定できる。
Texparent 親プロセスのファイル名。Setparent() によって設定できる。
Dirhead パスの頭部
Dirlib ライブラリ ketlib のパス
Dirbin ketbin のパス
Dirwork 作業ディレクトリのパス。Changework() によって設定できる。
Shellfile シェルファイル名

以下の予約変数は、ライブラリが使用するグローバル変数であるので、ユーザーはこれらの変数に値を代入してはいけない。なお、変数は大文字小文字を区別するので、すべて小文字で書く分には支障はない。ユーザーが作るプログラムでは、すべて小文字か、先頭だけが
大文字の変数を使うことを勧める。

ADDAXES, ArrowlineNumber, ArrowheadNumber, BezierNumber, COM0thlist, COM1stlist, COM2ndlist, Dq, FUNLIST, Fnamesc , Fnamescibody, Fnameout, Fnametex, GDATAList, GLIST, GCLIST, GOUTLIST, KCOLOR, KETPIC-COUNT, KETPICLAYER, LETTERlist, LFmark, MilliIn, PenThick, PenThickInit, POUTLIST, SCALEX, SCALEY, SCIRELIST, SCIWRLIST, TenSize, TenSizeInit, ULEN, XMAX, XMIN, YaSize, YaThick, YMAX, YMIN, VLIST

リスト

リストとは、数や文字などを集めたもので、それぞれのものを「要素」といい、[] の中にコンマで区切って記述する。要素は型を問わない。K_ETCindy では、曲線を描くプロットデータが座標のリストであり、リスト処理をうまく使えば K_ETCindy で効率的に作図ができる。

リストの n 番目の要素にアクセスするのに、アンダーバー_ を使う。

```
list=[1,2,3,4,5];  
println(list_2);
```

とすると、list の中の 2 番目の要素 2 が表示される。

```
list=[1,2,3,4,5];  
list_2="a";
```

とすると、list の中の 2 番目の要素が文字 a になる。

たとえば、曲線の交点を求める [Intersectcrvs\(\)](#) の戻り値から交点の座標を求めるにはアンダーバーを使う。使用例は、Intersectcrvs() の例を参照されたい。

3.4 よく使う Cindyscript のコマンド

値の表示

print(値) : コンソールに値を表示する。改行しない。

println(値) : コンソールに値を表示する。改行する。

【例】関数 Intersectcrvs() の戻り値を表示する。

```
tmp=intersectcrvs("sgAB","crCD");  
println(tmp);
```

条件判断

if(A,B,C) : もし A が真なら (成り立てば) B を, 偽なら C を実行する。

A の条件判断には次のものがよく使われる。

a が b より大きい	if(a>b,...
a が b より小さい	if(a<b,...
a が b 以上	if(a>=b,... (>と=の順序に注意)
a が b 以下	if(a<=b,... (<と=の順序に注意)
a と b が等しい	if(a==b,... (等号を 2 つ)
a と b が異なる	if(a!=b,...

if 文はネストして使うことができる。

【例】n が正, 負, ゼロのいずれかを判断して, コンソールに表示する。

```
if(n>0,print("正"),if(n==0,print("0"),print("負")));
```

繰り返し

repeat(n, 操作) : 操作を n 回繰り返す。

repeat(n,s, 操作) : 操作を n 回繰り返す。カウンタとして s を使う。(文字は他でも可)

【例】A を 4 個並べて描画面に表示する。

```
repeat(4,s,drawtext([s,0],4));
```

ここで、s の値は 4 回繰り返すうちに、1,2,3,4 と変化する。

リストによる繰り返し

forall(list, 処理) : リストの要素すべてに渡るように繰り返す。

【例】 点のペアをリストとし、線分を描く。

```
sglist=[[A,B],[C,D],[E,F]];
forall(sglist,Listplot(#));
```

これは、

```
Listplot([A,B]);
Listplot([C,D]);
Listplot([E,F]);
```

とするのと同じ。ここで、#は実行変数と呼ばれ、リストのそれぞれの要素を表す。

ユーザー定義関数

ユーザー定義関数は次の書式で定義する。

関数名 (引数):=(処理)

【例】 引数の値の正負を表示する関数 sign(n) を定義する。

```
sign(n):=(
  if(n>0,print("正"),print("0または負"));
);
```

定義した関数は

```
n=3;
println(sign(n));
```

のようにして使う。

KeTCindy では、アニメーション PDF を作成するときに、フレームを定義するのに使う。

幾何要素へのアクセス

Cinderella では、点の座標は同次座標で表されており、点の名称でそのまま座標を取得できることが多い。そのため、たとえば Listplot() 関数では、点を指定するのに、座標 [a,b] の代わりに点名を使うことができる。

Listplot() の書式 1 `Listplot("1",[1,1],[4,5])`

Listplot() の書式 2 `Listplot("1",[A,B])`

しかし、明確に直交座標で取得したい場合は `A.xy` (x,y 座標) `A.x` (x 座標) `A.y` (y 座標) として取得する。

リスト処理

Cindyscript のリスト処理のうち、よく使うものを挙げる。

`a` から `b` までの整数のリストは `a..b` (ドット 2 つ) で生成できる。このリストの各要素を番号代わりに使って、`apply(list,expr)` を用いると座標のリストを作ることができる。`apply(list,expr)` は、`list` の各要素に、処理 `expr` を施したリストを作る関数である。

【例】星形五角形を描く

```
r=2;
pt=apply(0..5,r*[cos(pi/2+#*4*pi/5),sin(pi/2+#*4*pi/5)]);
repeat(5,s,Listplot(text(s),[pt_s,pt_(s+1)]));
```

ここで、`text(s)` は、数を文字列に変換する Cindyscript の組み込み関数。

よくあるエラーメッセージ

Index out of range	リストの要素の個数外の値を指定した。
String Index out of range	文字列のインデックスが範囲外。
Potential type mismatch unexpected)	変数の型が合わない。文字と実数をかけ算したときなど。 括弧の種類が前後で合っていない。
close without open	閉じ括弧に対応する開き括弧がない。
open without close	開き括弧に対応する閉じ括弧がない。
Unknown function	関数が定義されていない。

その他、Cindyscript については、スクリプトエディタからヘルプを参照されたい。