Guide to KETCindy

KETCindy Project Team

September 16, 2018

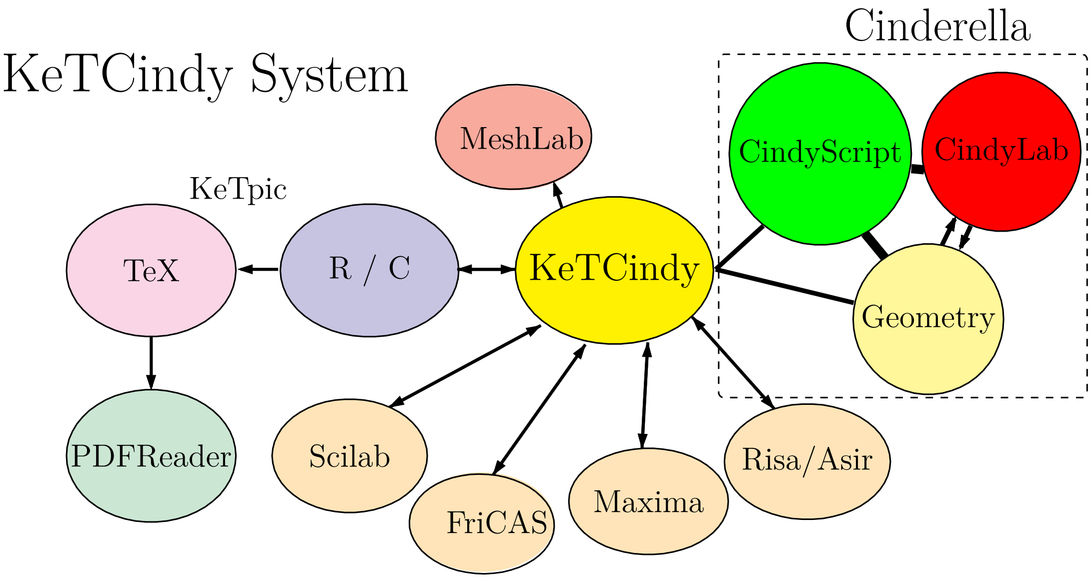
- ver.3.2 -

Contents

1 About KETCindy

1.1 Overview

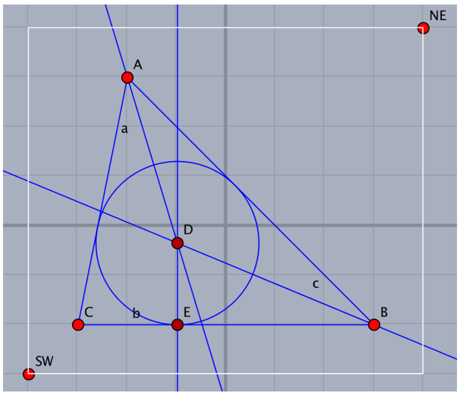
KETCindy is a library of Cindyscript which is a programming language of Cinderella. It converts the data computed for generating dynamic graphics on Cinderella into TeX graphical codes. Synchronized use of interactive graphics capabilities of Cinderella and well-structured programming capabilities of Cindyscript enables ordinary TeX users to efficiently embed high-quality graphics into TeX documents. Moreover, the collaborative use of KETCindy and other software such as R, Maxima and C has been enabled.



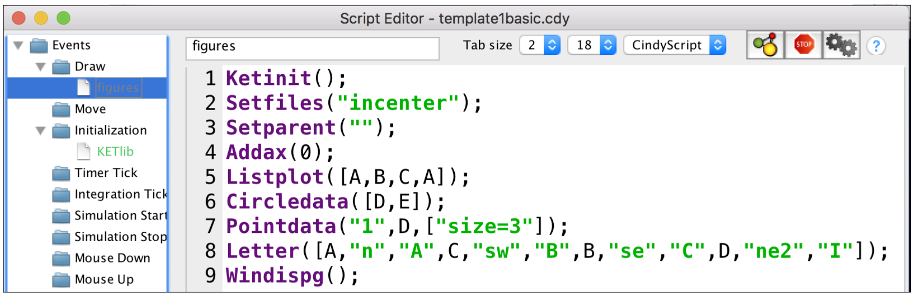
Firstly, dynamic figure is generated on Cinderella. Secondly, KETCindy generates a source file of R and makes R execute it for the generation of TeX graphical codes. Thirdly, those codes are formatted into TeX file which is input in the targetting TeX document via the command `\input`. Finally, usual compilation procedure of TeX results in the generation of final PDF output including the corresponding figure. A batch file `kc.bat` for Windows or a shell file `kc.sh` for Mac or Linux is generated via KETCindy in order to batch-process all the steps from the second to the last. Also by using these files, collaboration of Cinderella and other software as shown in the schematic diagram above is processed.

Summarizingly, specific steps to generate a TeX figure are listed as follows.

(1) Generate the needed geometric elements on the Euclidean view of Cinderella using its drawing tools. These elements can be moved interactively.



(2) Input the KETCindy codes into Cindyscript editor to specify the graphical elements to be displayed in TeX final output. Also KETCindy codes are used to generate supplementary graphical elements and handle them.



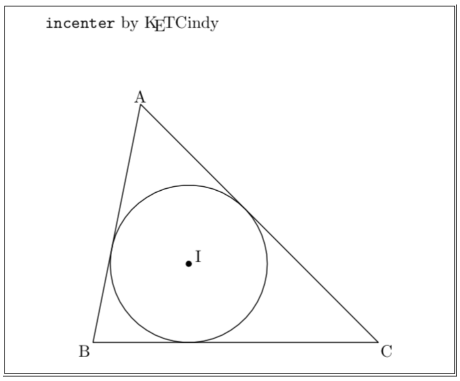
```
Ketinit();
Setfiles("incenter");
Setparent("");
Addax(0);
Listplot([A,B,C,A]);
Circledata([D,E]);
Pointdata("1",D,["size=3"]);
Letter([A,"n","A",C,"sw","B",B,"se","C",D,"ne2","I"]);
Windispg();
```

In this stage, the programming capabilities inherently implemented to Cindyscript can be used simultaneously. Execute the whole program by clicking the "Run" button. For more details, see section 3.

(3) Click the button named `Figures` in Euclidean view to automatically generate the following files in the folder named "fig". Here, "incenter" is the name specified via the command `Setfiles("incenter")` in step (2).

`kc`.sh or `kc.bat`	shell script file(Mac) or batch file(Windows)
`incenter.r`	
`incenter.tex`	TeX file composed of graphical codes
`incentermain.aux`	
`incentermain.log`	
`incentermain.pdf`	PDF file to display the resulting graphical image
`incentermain.tex`	TeX file temporarily used to generate the file `incentermain.pdf`

Subsequently, the file `incentermain.pdf` is automatically displayed as shown below.



We can manipulate this final output by modifying the inputs in steps (1) and (2) before processing the step (3) again.

(4) Using KETpic package of TeX, `incenter.tex` can be read into the targetting TeX document via the command

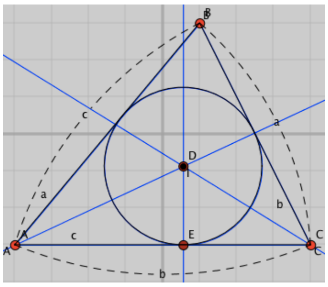
```
\input{incenter}
```

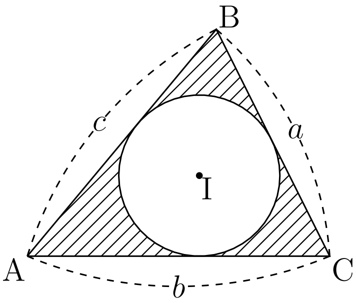
Then the same figure is embedded in the targetting PDF output.

1.2 Geometric Figures

Producing geometric figures in the plane is easy. Moreover, we can add hatchings in some areas, which is better than shading for monochrome printing. The following are the main parts of the script.

```
Listplot([A,B,C,A]);
Circledata([D,E]);
Bowdata([B,A],[1,0.5,"Expr=c","da"]);
Bowdata([C,B],[1,0.5,"Expr=a","da"]);
Bowdata([A,C],[1,0.5,"Expr=b","da"]);
Hatchdata("2",["oi"],[["crDE"],["sgABCA"]],["dr,0.7",""]);
Pointdata("I",D,["size=4"]);
Letter([A,"sw","A",B,"ne","B",C,"se","C",D,"se","I"]);
```





1.3 Graphs of Functions

KETCindy can produce graphs of functions with

```
Plotdata("1","x^2","x");
```

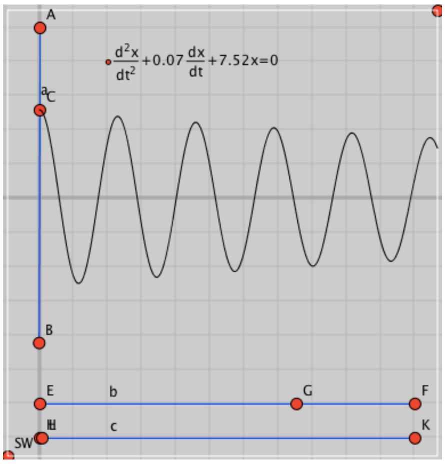
or parametrically with

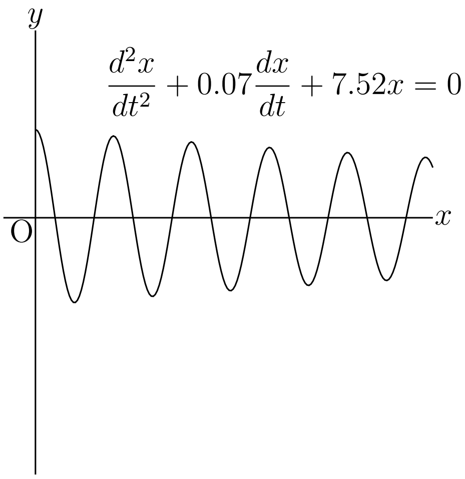
```
Paramplot("1","[2*cos(t),sin(t)]", "t=[0,2*pi]");
```

Here we give an example of the solution curve of a differential equation. The script is:

```
Deqplot("1","y``=-L.x*y`-G.x*y","t=[0,XMAX]",0,[C.y,0]);
// the equation is y''=-ay'-by (a=L.x, b=G.x).
// C.y,0 are initial values of y and y' at t=0.
Expr(M,"e","\displaystyle\frac{d^2 x}{dt^2}+"
      +"+L.x+"\frac{dx}{dt}+"+G.x+"x=0");
```

Note that points C, G, L on segments AB, EF, HK are movable, and are used to decide the coefficients and the initial value as you can see in the above scripts.

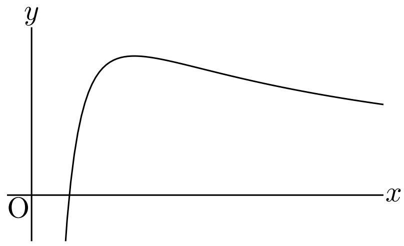




1.4 Drawing Tables

Writing the code for tables to be inserted into the TEX documents is sometimes troublesome. However, it is not a hard job for KETCindy (see the output in Figure.

```
xLst=apply(1..7,15);
yLst=[10,10,10,10,80];
rmvL=apply(1..6,"c"+text(#)+"r4r5");
rmvL=concat(rmvL,["r2c1c2","r3c1c2"]);
Tabledata("",xLst,yLst,rmvL);
Tlistplot(["c1r1","c2r4"]);
Tlistplot(["c2r1","c1r4"]);
Putrowexpr(1,"c",
    ["x","0","\cdots","e","\cdots","e\sqrt{e}","\cdots"]);
Putrowexpr(2,"c",["y`","","+","0","-","-","-"]);
Putrowexpr(3,"c",["y``","","-","-","-","0","+"]);
Putrowexpr(4,"c",["y","","","10/e","","15/e\sqrt{e}",""]);
Putcell("c0r4","c7r5","c","\input{fig/graph}");
```

<table>
<tr><td>x</td><td>0</td><td>$\cdots$</td><td>e</td><td>$\cdots$</td><td>$e\sqrt{e}$</td><td>$\cdots$</td></tr>
<tr><td>y'</td><td rowspan="3"></td><td>$+$</td><td>0</td><td>$-$</td><td>$-$</td><td>$-$</td></tr>
<tr><td>y''</td><td>$-$</td><td>$-$</td><td>$-$</td><td>0</td><td>$+$</td></tr>
<tr><td>y</td><td></td><td>$\frac{10}{e}$</td><td></td><td>$\frac{15}{e\sqrt{e}}$</td><td></td></tr>
<tr><td colspan="7">
</td></tr>
</table>

1.5 Plotting data

Here we call the data computed to generate the graphs of functions and geometric elements "Plotting data" which is abbreviated as PD. The PD to draw segment is the list of coordinates of its two endpoints. For example, when the coordinates of the points A and B are (1, 1) and (3, 2) respectively, PD of the segment AB named `Listplot ([A,B])` is stored in the form `[[1,1],[3,2]]`. Also the PD to draw a curve is the collection of those for drawing small segments which connect contiguous dividing points of the curve. PD are automatically given names via KETCindy following the rules below.

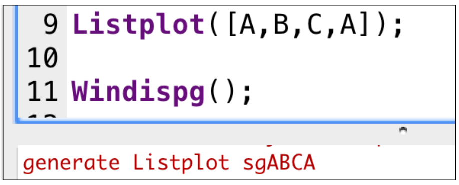
- The beginning part of the PD's name depends on the kind of the corresponding graphical element. For instance, `sg` is associated to segments and `cr` is associated to circles.
- When some extra name is specified as the first argument in the definition of PD, it is added to the beginning part given above. For instance, the PD defined below is given the name `sg1`.

```
Listplot("1",[[0,0],[1,2]]);
```

- When the extra name is not needed, the names of the points are added to the beginning part given above. For instance, the PD defined below is given the name `sgABC`.

```
Listplot([A,B,C]);
```

Once PD are generated, their names are displayed on the console view of Cinderella. For instance, when the PD named `sgABCA` is generated, the corresponding message is displayed as shown below.



```
Listplot([A,B,C,A]);

Windispg();
```

```
generate Listplot sgABCA
```

Also the content of PD is displayed via the function `println()` of Cindyscript. For instance, inputting the command `println(sgABCA)` makes the following list displayed.

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

This list is composed of the coordinates of the points A, B, and C.

These names of PD are used when the corresponding PD need to be transformed. For instance, PD to draw the parallel transport of the segment AB is generated via the KETCindy command

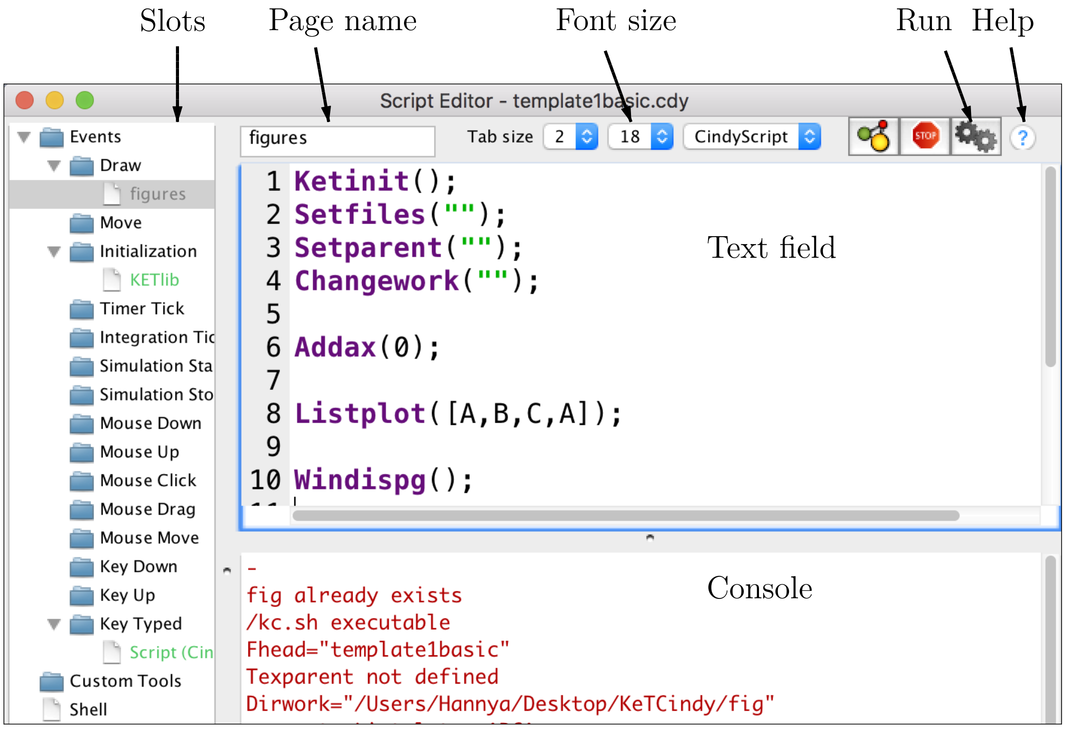
```
Translatedata("1","sgAB",[2,3]);
```

PD can be generated also by using the programming capability of Cindyscript which can be subsequently used in KETCindy. For more details, see the example of `Listplot()` in the command reference. Inclusion of too much elements into a single PD may cause some error. To prevent such error, PD should be divided into several PD each of which is composed of 200 elements or so.

2 Cindyscript

2.1 Cindyscript editor

Choose "Cindyscript" in the "Scripting" menu or push keybuttons Ctrl+9 (Windows) / Command+9 (Mac), then Cindyscript editor opens as shown below.



figures

Tab size 2

18

```
Ketinit();
Setfiles("");
Setparent("");
Changework("");

Addax(0);

Listplot([A,B,C,A]);

Windispg();
```

```
-
fig already exists
/kc.sh executable
Fhead="template1basic"
Texparent not defined
Dirwork="/Users/Hannya/Desktop/KeTCindy/fig"
```

Commands can be input into preferred "slot". Specific timing for execution of commands is assigned to each slot. The slot for current work can be chosen only by clicking the corresponding tab in the menu. Users can add extra pages to each slot. For instance, when some initialization other than those included in `KETlib` is needed, clicking the folder icon of "Initialization" makes a new page open in which extra commands can be input. The name of each page can be given by directly inputting it into the "Page name" column. The font size of the scripts can be tuned by changing the number in the "Font size" column. Frequently used slots are listed below.

- Draw

  The commnds in this slot are executed when some change, like movement of point, occurs in the Euclidean view. In `templatebasic1.cdy`, the protoype page named `figure` including the KETCindy commnads like `Ketinit();` and `Windispg();` which are unconditionally necessary has been prepared. The KETCindy commands for drawing should be input into this slot.

- Initialization

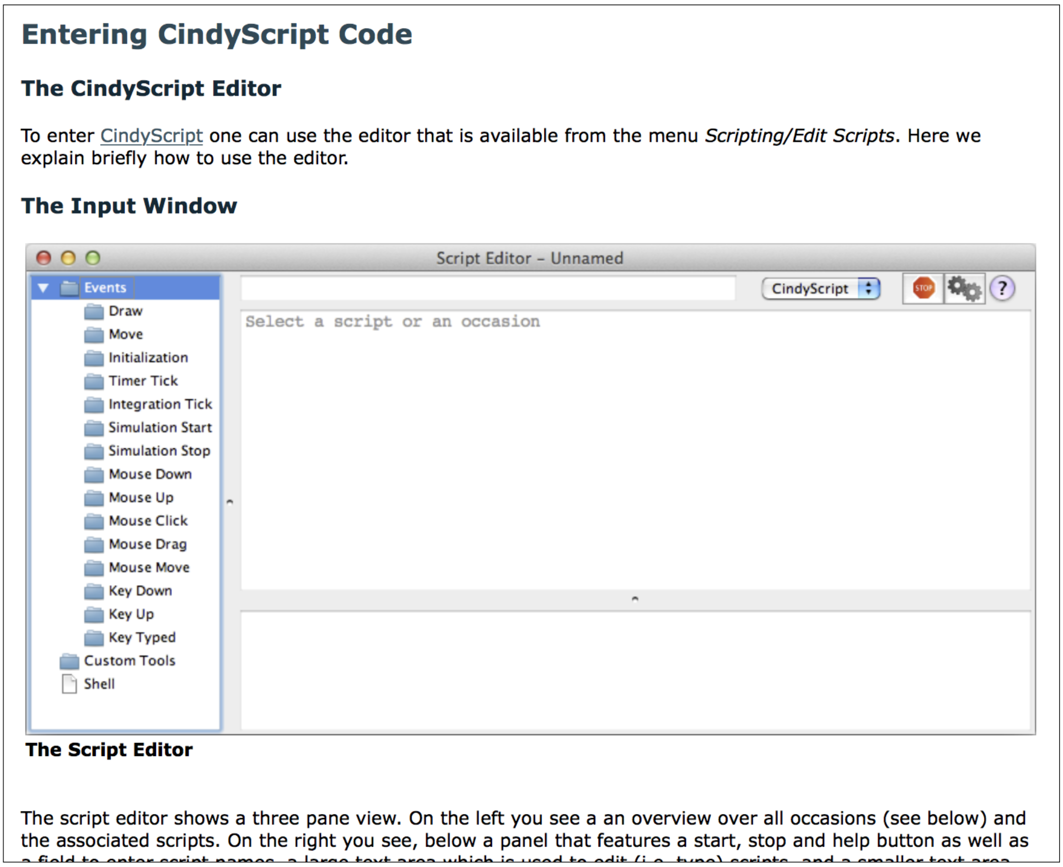
  The definitions of functions and the initial values of variables are input here. The commands in this slot are exected only once just after the "Run" button is clicked. Thus, the initial data in this slot is changed when some modifications are made in other slots. In `templatebasic1.cdy`, the protoype page named `KETlib` including the default setting of KETCindy has been prepared.

- Key Typed

  The commnds in this slot are executed when some key is pushed.

Clicking ”Run” button or pushing the keybuttons Shift+Enter makes the whole program be executed. The results derived from executing the function `print()` and error messages are displayed on the console view which is put at the bottom part of Cindyscript editor. Each error and its location is displayed together with the message ”WARNING” or ”syntax error”. The outputs displayed on the console can be copied to other usual text editors.

Click the ”Help” button, then reference manual of Cinderella opens as shown below.



Entering CindyScript Code

The CindyScript Editor

To enter CindyScript one can use the editor that is available from the menu *Scripting/Edit Scripts*. Here we explain briefly how to use the editor.

The Input Window

The Script Editor

The script editor shows a three pane view. On the left you see a an overview over all occasions (see below) and the associated scripts. On the right you see, below a panel that features a start, stop and help button as well as

2.2 Input

The attribute of each input into Cindyscript is specified via the color of the corresponding letters as listed below.

- The functions which are inherently implemented to Cinderella are displayed via blue color.
- The functions which are defined by user, including those of KETCindy, are displayed via purple color.
- The functions which are not yet defined are displayed via red color.
- Strings are displayed via green color.

As in the console view, copying and pasting to the other usual editing software via pushing the keybuttons Ctrl+C and Ctrl+V is accessible. Cutting and pasting via Ctrl+X and Ctrl+V is also possible. Also as in the other editing software, preferred strings can be specified via dragging mouse or pushing the keybutton Shift and moving the sursor. Serching for words via pushing Ctrl+F has not been enabled.

The fundamental rule of describing scripts on Cindyscript editor are listed below.

- Upper- and lowercase letters are distinguished. Using lowercase letters is preferable.

- As in TeX, several blanks are regarded as a single blank.
- A semicolon should be located at the end of each row. Starting a new paragraph does not result in the ending of commnds.

Particularly, in case of KETCindy, the input of commands are controlled by the following rules.

- The names of global variables begin with uppercase letters.
- The names of local variables begin with lowercase letters. Local variables are declared at the beginning part of the definitions of functions along with the Cinderella command `regional()`.
- The names of functions begin with uppercase letters.

2.3 Variables and constants

The declaration of the attribute of each variable is not needed in Cindyscript since it is automatically decided according to the input. Moreover, the different kind of value can be input without any declaration.

Example

```
a=10;
b=2;
c=a+sqrt(b);
a="the square root of"
println("The sum of"+a+b+''and 10 is''+c);
```

In this example, the attribute of variable `a` was firstly integer, and then changed to string at the fourth row.

The strings should be input with double quotation marks. The mathematical operations which involve several kind of variables must be taken much care. Exceptionally, connecting string and number with `+` results in the generation of one single string.

The variable `pi` is reserved in Cindyscript as the ratio of the circumference of a circle to its diameter. Also the variable `i` is reserved as the imaginary unit. When `i` is used as variable once, it is changed to the imaginary unit via the command

```
i=complex(0,1);
```

There are also some reserved variables in KETCindy. Among them, the following ones can be changed by users.

`Fhead`	the beginning part of the file name which can be set by `Setfiles()`
`Texparent`	the name of parent file which can be set by `Setparent()`
`Dirhead`	the beginning part of the path
`Dirlib`	the path to the library ketlib
`Dirbin`	the path to ketbin
`Dirwork`	the path to the working directory which can be set by `Changework()`
`Shellfile`	the name of shell file

Contrarily, the reserved variables listed below are the global variables usend in the library of KETCindy, whence cannot be changed by users.

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

A list can be defined by putting its elements in a square bracket with commas separating each other. The attribute of each element does not matter. The n-th element can be referred by using an underbar. For instance, the commands

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

make the second element be substituted by the letter `"a"`.

2.4 Frequently used commands

Displaying the computed output

The following commands make the current value of the variable on the console view.

`print(the name of variable);`	without a line break
`println(the name of variable);`	with a line break

Conditional branching

The commnad `if(A,B,C)` executes `B` if `A` is true and `C` otherwise. The followings are frequently used. Nested conditions can be interpreted.

`if(a>b,...);`	
`if(a<b,...);`	
`if(a>=b,...);`	$a \geqq b$
`if(a<=b,...);`	$a \leqq b$
`if(a=b,...);`	
`if(a!=b,...);`	$a \neq b$

Loop program

The commnad `for(n,operation)` executes the operation n times. If the counter should be specified, modify the command as `for(n,s,operation)`. where the value of s is successively changed. Loop program with respect to some list instead of counter is also possible via the command as `forall(list,operation)`. For example, the commands

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

have the same output as

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

User's definition of functions

The format of definition is `function name(argument):=(operation)`. For example, if we define the function `sign(n)` by

```
sign(n):=(
  if(n>0,print("positive"),print("0 or negative"));
);
```

it can be used as

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

Reference to geometric elements

The position of a point can be specified with both its name and the list of its x, y-coordinates. Thus, both of the following formats are allowed.

```
Listplot("1",[[1,1],[4,5]]);
Listplot("1",[A,B]);
```

Also we can get the coordinate of a point explicitly via the commands like `A.xy`, `A.x`, and `A.y`.

List processing

The list of integers between a and b is generated via the command `a..b`. For instance, the synchronized use with the command `apply` as below gives the shape of pentagram.

```
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)]));
```

Here the Cindyscript command `text` is used to convert the number into string.

3 Collaboration with other softwares

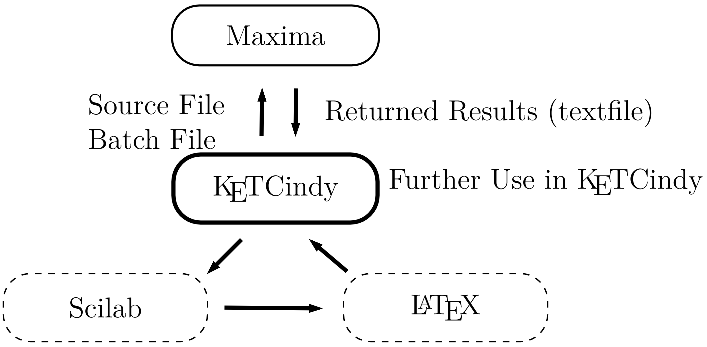
3.1 Overview

KETCindy has functionalies to call other softwares such as Maxima, Risa/Asir, R and C. Here, we introduce how to call Maxima.

The steps are as follows.

1. Generate the shell file to call a CAS.
2. Execute the file.
3. Return the result as text.
4. Use the result in KETCindy.
5. Produce the PDF file.

And the flowchart is as follows:



When interfacing with Maxima, commands `Mxfun`, `CalcbyM` and `Mxtex` are all we need to complete the task. `Mxfun` and `CalcbyM` are for calling single command and multi commands of Maxima respectively. `Mxtex` is used for code conversion to LaTeX. The output of Maxima is returned to KETCindy as a string or a list of strings for further processing.

The options of these commands are:

`"m/r"`	To decide whether the result file will be made again or not. If these options are not given, KETCindy decides automatically.
`"Disp=y/n"`	To decide whether the result will be displayed in the console or not. It is only availabe for `Mxfun` and `Mxtex`. The default is ”y”.

3.2 Commands related to Maxima

Mxfun

The arguments are name of variable in KETCindy, name of a function of Maxima, and a list of arguments of the function.

`Mxfun("1","diff",["sin(x)","x"]);` // The return is ”cos(x)”, assgined to mx1.

The above is equivallent to

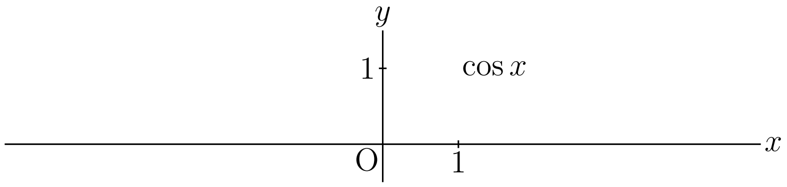
`Mxfun("1","diff(sin(x),x)",[]);`

Mxtex

The arguments are name of variable in KETCindy, an expression in Maxima format.

`Mxtex("1",mx1);` // The return is `"\cos x"`, assgined to tx1.

`Expr([0,1],"e",tx1]);`



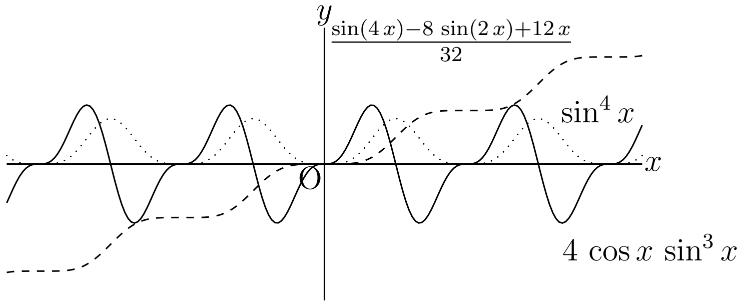
CalcbyM

The arguments are name of variable in KETCindy, a list of commands and the arguments of Maxima.

```
fn="sin(x)^4";
cmdL=[
  "df:diff",[fn,"x"],
  "df:ratsimp",["df"],
  "F:integrate",[fn,"x"],
  "F","ratsimp",["F"],
  "df::F",[]
];
CalcbyM("ans",cmdL);
```

The returned value is a list of df and F as strings, though these are not displayed in the console. They can be used, for example,

```
Plotdata("1",fn,"x",["Num=200","do"]);
Plotdata("2",ans_1,"x",["Num=200","dr"]);
Plotdata("3",ans_2,"x",["Num=200","da"]);
Mxtex("1",fn);
Mxtex("1",ans_1);
Mxtex("2",ans_2);
Expr([A,"e",tx1,B,"e",tx2,C,"w",tx3]);
```



Remark See KeTCindyreferenceE.pdf for more information.

3.3 Commands related to R

`Rfun` and `CalcbyR` are simillar to `Mxfun` and `CalcbyM`.
See `KeTCindyreferenceE.pdf` or `samples/s08R` for more information.

4 Three Dimentional figures of KETCindy

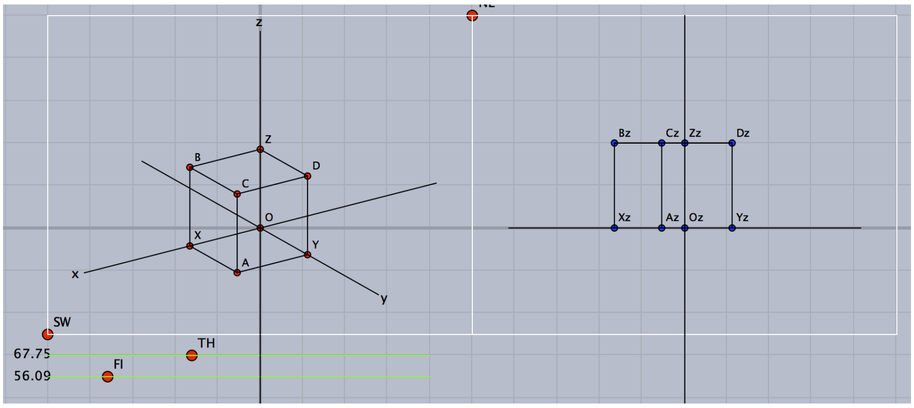
4.1 Summary and Geometric Elements

In KeTCindy's 3D-mode, there are two rectangular areas surrounded by a white frame on the Euclidean view.

The main area on the left side of the screen is simillar to that of two dimentional figures. Figures in this area will be drawn to the TEX document. The view direction can be moved with sliders under the main area. `TH` and `FI` mean angles θ and φ respectively, which are polar cocordinates of the view direction.

Figures from the view direction $(0,\ \varphi)$ are displayed in the sub area on the right side.

mainarea subarea



With internal command `Ptseg3data` which is called in `Start3d`, a point put to the main area with the drawing tool of Cinderella is regarded as a 3D point by KETCindy, and a correspoinding point is put in the sub area. Though the initial coordinate of z is 0, we can change it moving the point in the sub area.

For example, if we put point `A` on the main area, point `Az` will be put in the sub area and the 3D coordinates calculated from `A` and `Az` are assigned to varible `A3d`.

Remark Note that point `Az` will not be deleted automatically even if point `A` is deleted. We should delete it manually.

Geometric segment in the main area generates the corresponding geometric segment in the sub area as well.

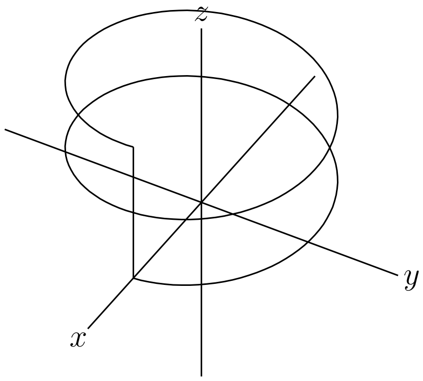
4.2 Lines and Curves

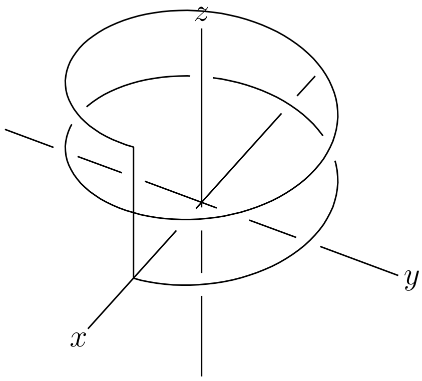
KETCindy commands `Spaceline` and `Spacecurve` are used do draw lines and curves in the space. Additionally, `Xyzax3data` is used to draw axis.

Examples

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
Spacecurve("1","3*[cos(t),sin(t),0.1*t]","t=[0,4*pi]",["Num=200"]);
pt1=[3,0,0]; pt2=[3,0,3*0.1*4*pi];
Spaceline("1",[pt1,pt2]);
Skeletonparadata("1");
```

Remark The last command skeleton elimination is for skeleton elimination. Compare two figures below. The right one is with skeleton elimination.





4.3 Two Dimensional Figures

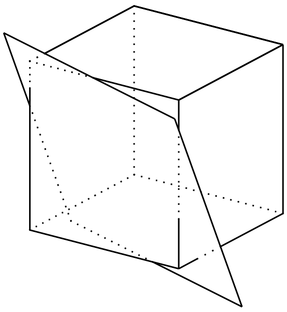
Data of two dimensional figures such as polyhedra or planes are given in obj format.

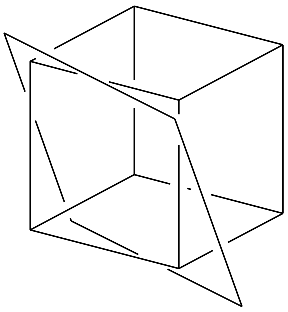
Examples

```
Start3d();
vertex=[[2,2,-2],[2,-2,-2],[-2,-2,-2],[-2,2,-2]];
Reflect3d1(``1'',vertex,[[0,0,0],[1,0,0],[0,1,0]];
vertex=concat(vertex,ref3d1);
edge=[[1,2,6,5],[1,5,8,4],[1,4,3,2],[2,3,7,6],[3,4,8,7],[5,6,7,8]];
cube=[vertex,edge];
plane=[[[-3,1,-3],[3,-1,-3],[-4,5,3],[2,3,3]],[[1,2,4,3]]];
tmp=Concatobj([cubic,plane]);
VertexEdgeFace("1",tmp,["Vtx=nogeo","Edg=nogeo"]);
Nohiddenbyfaces("1","phf3d1"); // for the figure on the left
```

Remark

Command `Concatobj` combines data in obj format.
Command `VertexEdgeFace` assigns vertices to `phv`, edges to `phe` and faces to `phf`.
Command `Nohiddenbyfaces` is for hiddenline elimination.
Use `Skeletonparadata("1")` if the figure on the right is desirable.





Remark See `KeTCindyreferenceE.pdf` or `samples/s05spacefigure` for more information.

4.4 Surfaces

Two variable function is defined as a list of one of the followings.

1. `["z=f(x,y)","x=[a,b]","y=[c,d]"]`

2. `["z=f(x,y)","x=x(u,v)","y(u,v)","u=[a,b]","v=[c,d]"]`

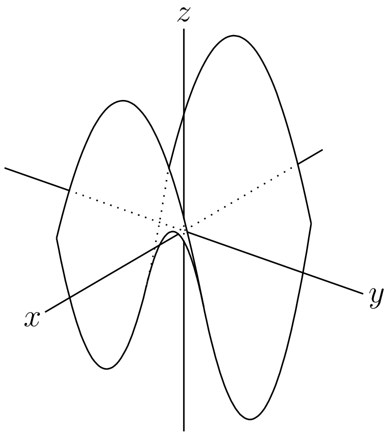
3. `["p","x=x(u,v)","y=y(u,v)","z=z(u,v)","u=[a,b]","v=[c,d]"]`

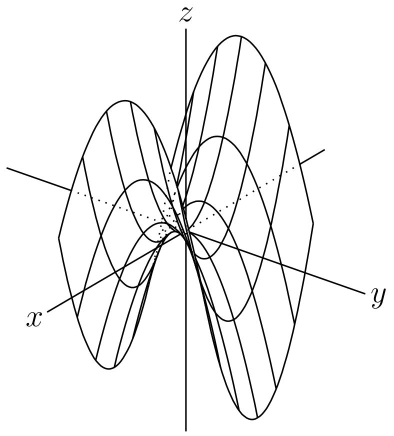
Optionally, you can add what boundaries should be drawn. The default is "wesn". Here, for example, "w" means the boundary defined by $[a,t]$ ($c \leqq t \leqq d$).

KETCindy calls C to speed up the calculation of hidden lines elimination.

Example

```
Start3d();
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
fd=["z=x^2-y^2","x=[-2,2]","y=[-2,2]","senw"];
Startsurf();
Sfbdparadata("1",fd);
Crvsfparadata("1","ax3d","sfbd3d1",fd);
ExeccmdC("1"); Windispg();
```





Remark Wires can be added if necessary with command `Wireparadata` as seen in the upper right side figure. The line-style also can be changed.

See `KeTCindyreferenceE.pdf` or `samples/s09surfaceC` for more information.

4.5 Generating Files in Obj Format

KETCindy can generate files of 3D figures in obj format. Moreover, KETCindy also can call Meshlab which is a 3D viewer.

Examples

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
fd=["p","x=4*sin(V)*cos(U)","y=4*sin(V)*sin(U)","z=4*cos(V)",
      "U=[pi/2,4*pi/2]","V=[0,pi]","we"];
Mkobjcmd("1",fd,[40,40,"-"]);
Mkobjcrvcmd("2","ax3d",[0.05,"xy"]);
Mkobjsymbcmd("x",0.2,0,[0,1,0],[5.2,0,0]);
Mkobjsymbcmd("y",0.2,0,[1,0,0],[0,5.2,0]);
Mkobjsymbcmd("z",0.2,0,[0,1,0],[0,0,5.2]);
SetObj();
```

Remark See `KeTCindyreferenceE.pdf` or `samples/s13meshlab` for more information.

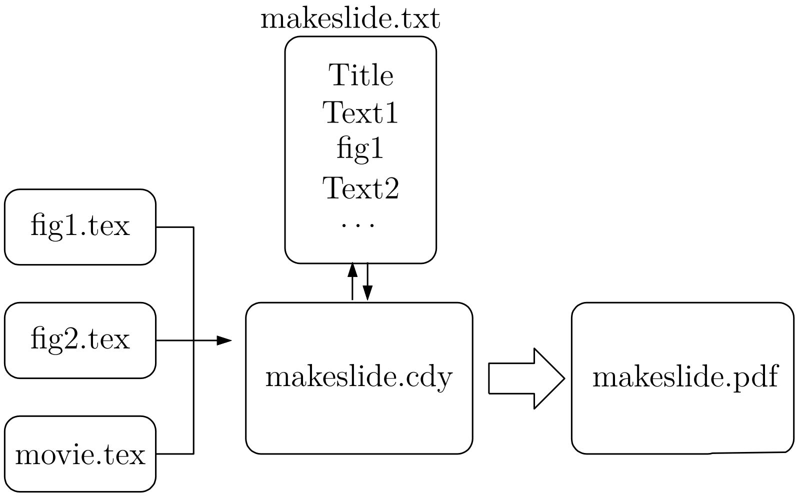
5 Making Slides

5.1 Outline

KETCindy has functions to make slides for presentation with the help of `layer` environment which is defined in `ketlayer.sty`. See `KeTpicStyleE.pdf` for details about `layer`.

You need both a cindy file and a text file with the same header. For simple preparation, copy `template2slide.cdy` to your work folder, rename the name, for example `makeslide.cdy`, double click the file and press the button `title` in the screen, then `makeslide.txt` will be generated.

The following chart shows the relation between them.

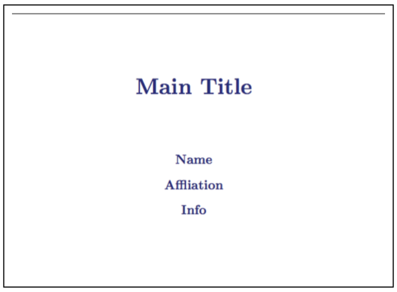


If necessary, edit `Settitle` in CindyScript editor, and press the gear mark. Open the text file, and write commands of KETCindy Slide and TeX as follows.

```
title::slide0//
main::Introduction//
\slidepage[m]//
new::Programming Language//
%repeat=6,para//
\slidepage//
itemize//
item::Python//
%thin[2,-]::item::Ruby//
%thin[3,-]::item::Java//
%thin[4,-]::item::JavaScript//
%thin[5,-]::item::CindyScript//
%thin[6,-]::item::C//
end//
```

Remark “//” should be added to the end of all lines. Use “||||” when you want to write `//`.

Press the button `Slide` in Cindy Screen, then KETCindy will make TeX file `makeslide.tex`, typeset it and displays the pdf file as follows. If there occurs an error, check the text file or the TeX file.

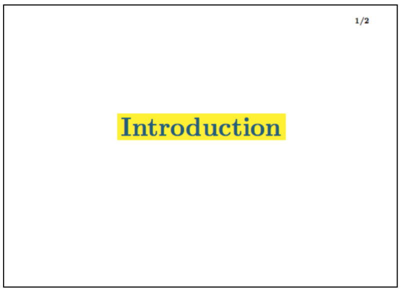


Main Title

Name

Affliation

Info



1/2

Introduction

2/2

Programming Language

- Ruby
- Java
- JavaScript
- CindyScript
- C

5.2 Commands of KETCindy slide

You can use the following commands.

```
    title::slide0 (::wallpaper)//
        Rem) Put only once at the first line.
    main::(main title)//
    new::(page title)//
    enumerate//
           =\begin{enumerate}
        Rem) Add the option such as [(1)] using :: .
    itemize//
           =\begin{itemize}
    layer::{xsize}{ysize}//
           =\begin{layer}{xsize}{ysize}
         Rem) "layer" is an environment defined in ketlayer.sty.
    item::sentence//
           =\item sentence
    putnote::dir{xpos}{ypos}::filename(,scale)//
           =putnotedir{xpos}{ypos}{\input{fig/filename}}|
         Rem) "putnote" is a command defined in ketlayer.sty
    end//
           =\end{itemize,enumerate,layer}
    ...//
          To insert a blank line.
```

You can also use the following TEX mcores added by KETCindy.

```
    \slidepage,\slidepage[m]//
          To display the number each page.
            \slidepage[m] is used for the \verbmain| page.|
    \setthin{thickness}
          To change the thickness of thin letters temporarily.
    \inputsound, \inputmovie
          To insert mp3/mp4 files.
```

Remark Any other TEX macroes are available. Put `%%` instead of `%` to comment out .

5.3 Display of Page step by step

1) Put just after new,

```
    %repeat=number of steps//
```

2) Put at the head of each line as

```
    %[2,-]::sentences
           display at all steps from 2
```

```
%[-,2]::sentences
        display at all steps until 2
%[1..3,5]::sentences
        display at steps of 1,2,3 and 5
```

3) Use `%thin` to display with thin letters.

```
%thin::[2,-]::sentence
```

4) The dencity can be changed with Setslidebody or `\setthin`.

5.4 Making Flip Animation

1) Define function `Mf(s)`, the state at s.

2) Put command `Setpara` in the script editor as

```
Setpara(subfolder,funcitonstr(mf(s)),range,options);
    options=["m/r", "Div=25"];
```

3) Describe in the text file as

```
%repeat=, para=subfolder:{0}:s{60}{10}:input(:scale)//
```

4) Press buttons `ParaF` and `Flip`, then `subfolder` will be generated.

5) Press button `Slide`.

5.5 Making Animation

1) Add the following in the script editor

```
Addpackage(["[dvipdfmx]{animate}"]);
```

2) Add in the second option of Setpara,

```
"Frate=num of frame in the second,"Scale=scale,"OpA=option of animation"
```

3) Press buttons `ParaF` and `Anime`, then `subfolder` will be generated.

4) Use `\input`, not layer, to display.

5.6 Commands to Insert a mp3/mp4 file

To insert a mp3 or mp4 file, change the first line to

```
title::slide0
::\usepackage{ketmedia}
::\usepackage[dvipdfmx]{media9}//
```

Use `\inputsound` or `\inputsoundclick` for mp3 files.

```
\inputsountclik[90]{folder/}{mp3file} %starts when the button clicked
```

The arguments are horizontal position(default is 90) of buttons, the folder, the file name.

Use `\inputmovie` for mp4 files.

```
\inputsountclik[90]{1}{0.4}{folder/}{mp4file} %starts when the button clicked
```

The 2nd and 3rd arguments are width and height as the coefficiients of `\linewidth`

5.7 Changing Style

The default styles such as size and color of letters can be changed. See `KeTCindyReferenceE,pdf` or `samples/s07slides`.