

Guide to K_ETCindy

K_ETCindy Project Team

August 20, 2018

- ver.3.2 -

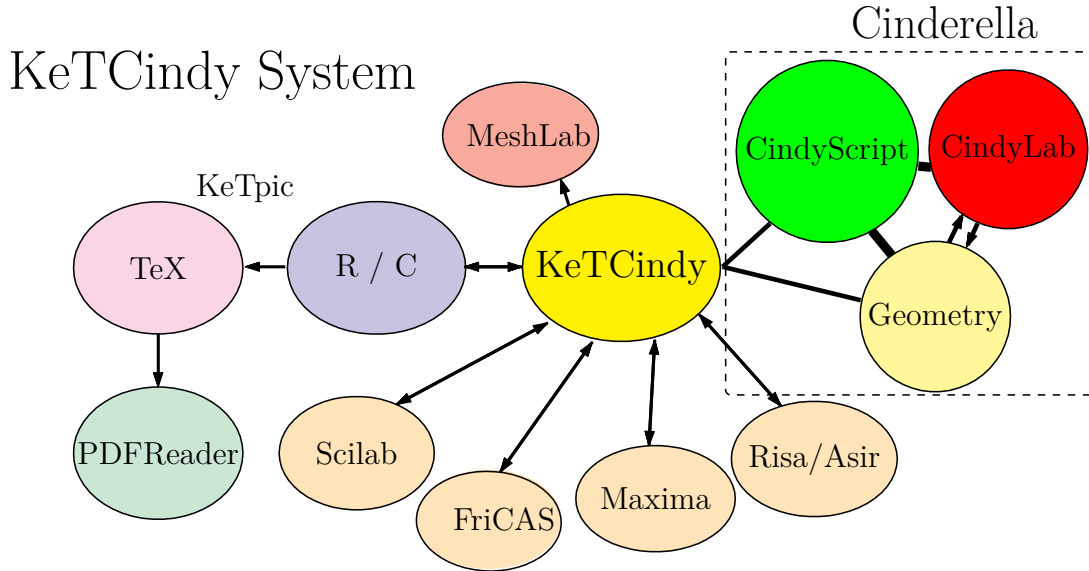
Contents

1	About K_ETCindy	2
1.1	Overview	2
1.2	The drawing procedure of K _E TCindy	4
1.2.1	Geometric figure	4
1.2.2	Graph of function	4
1.2.3	Spatial figure	4
1.2.4	Table	4
1.2.5	Collaboration with other software	4
1.3	Plotting data	5
2	Cindyscript	6
2.1	Cindyscript editor	6
2.2	Input	7
2.3	Variables and constants	7
2.4	Frequently used commands	7
3	Making Slides	8
3.1	Overall Flow	8
3.2	Editing Text File	8
3.3	Display of Page step by step	9
3.4	Making Flip Animation	9
3.5	Making Animation	9
3.6	Changing Style	10

1 About KeTCindy

1.1 Overview

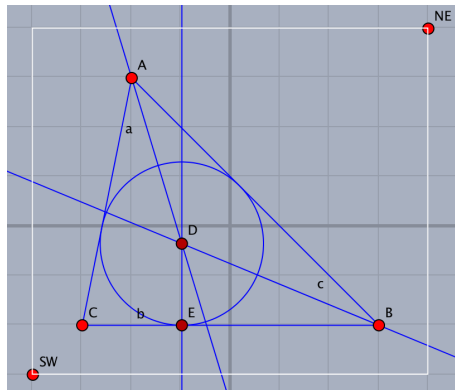
KeTCindy is a library of Cindyscript which is a programming language of Cinderella. It converts the data computed for generating dynamic graphics on Cinderella into \TeX graphical codes. Synchronized use of interactive graphics capabilities of Cinderella and well-structured programming capabilities of 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.



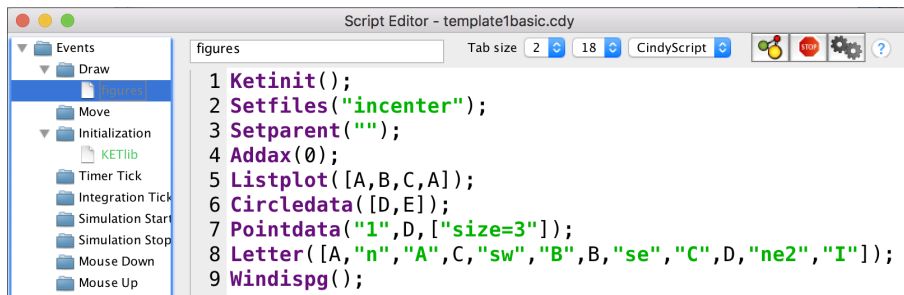
Firstly, dynamic figure is generated on Cinderella. Secondly, KeTCindy generates a source file of R and makes R execute it for the generation of \TeX graphical codes. Thirdly, those codes are formatted into \TeX file which is input in the targeting \TeX document via the command `\input`. Finally, usual 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 $\text{T}_\text{E}\text{X}$ final output. Also KETCindy codes are used to generate supplementary graphical elements and handle them.

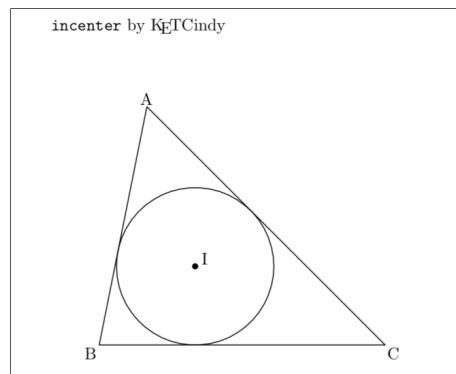


In this stage, the programming capabilities inherently implemented to Cindyscript can be used simultaneously. Execute the whole program by clicking the "Run" button. For more details, see section 3.

- (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).

<code>kc.sh</code> or <code>kc.bat</code>	shell script file(Mac) or batch file(Windows)
<code>incenter.r</code>	
<code>incenter.tex</code>	$\text{T}_\text{E}\text{X}$ file composed of graphical codes
<code>incentermain.aux</code>	
<code>incentermain.log</code>	
<code>incentermain.pdf</code>	PDF file to display the resulting graphical image
<code>incentermain.tex</code>	$\text{T}_\text{E}\text{X}$ file temporarily used to generate the file <code>incentermain.pdf</code>

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 $\text{T}_\text{E}\text{X}$, `incenter.tex` can be read into the targeting $\text{T}_\text{E}\text{X}$ document via the command

$$\backslash\text{input}\{\text{incenter}\}$$

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

1.2 The drawing procedure of K_ET Cindy

1.2.1 Geometric figure

1.2.2 Graph of function

1.2.3 Spatial figure

1.2.4 Table

1.2.5 Collaboration with other software

1.3 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.

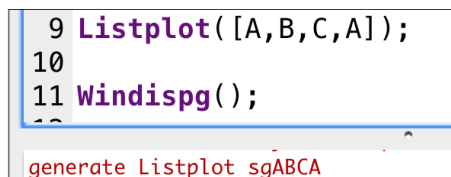
- The beginning part of the PD's name depends on the kind of the corresponding graphical element. For instance, `sg` is associated to segments and `cr` is associated to circles.
- When some extra name is specified as the first argument in the definition of PD, it is added to the beginning part given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



```
9 Listplot([A,B,C,A]);
10
11 Windisp();
12
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

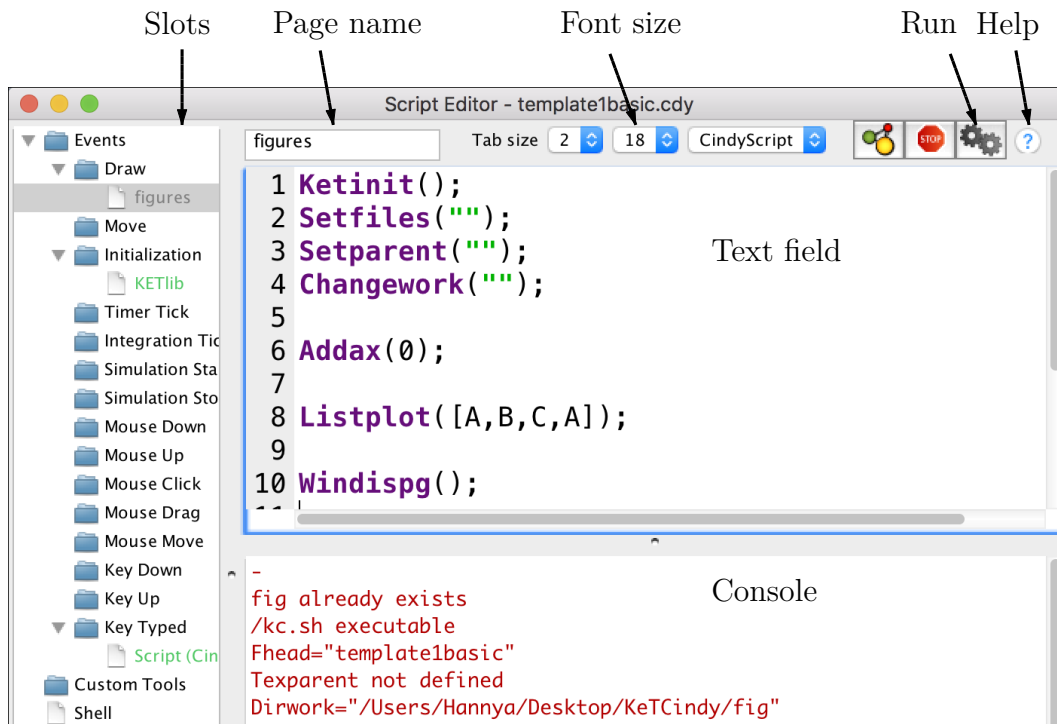
```
Translatedata("1","sgAB",[2,3]);
```

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.



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 commands in this slot are executed when some change, like movement of point, occurs in the Euclidean view. In `templatebasic1.cdy`, the prototype page named `figure` including the KETCindy commands 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 executed 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 prototype page named `KETlib` including the default setting of KETCindy has been prepared.

- Key Typed

The commands 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.



2.2 Input

2.3 Variables and constants

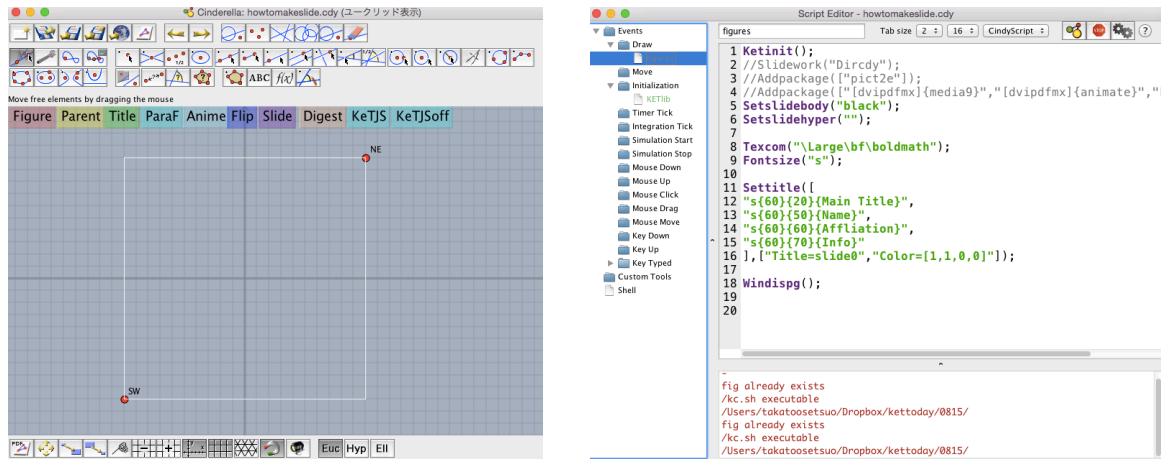
2.4 Frequently used commands

3 Making Slides

3.1 Overall Flow

KE_TCindy has functions to make slides for presentation. For this, follow steps below:

- 1) Copy and rename `template2slide.cdy` in `ketcindy` folder to a work folder and double-click the file (here, we call `sample.cdy`). Then the followings will appear.



- 2) Edit "Settitle", for example,


```
Settitle([
    "s{60}{20}{How To Use}",
    "s{60}{50}{a \ketcindy\ member}",
    "s{60}{60}{\ketcindy\ project}",
    "s{60}{70}{Aug. 20th}"
], ["Color=[1,1,0,0]"]);
```
- 3) Press button "Title", then the title page will be displayed. At the same time, text file "sample.txt" will be created if it does not exist. This `sample.txt` is a template file for making slides.
- 4) Press button "Slide", then KE_TCindy will make `sample.tex` from `sample.txt`, typeset it, and display `sample.pdf` which contains slides for presentation.

3.2 Editing Text File

- 1) Put `//` at the last of each line.
Rm) Use `||||` for `//`.
- 2) Commands are


```
title::titleslidenamename(::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.
Rem) Any other TeX command is available.

```

3.3 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,-]::sentence`
 display at all steps from 2
`%[-,2]::sentence`
 display at all steps until 2
`%[1..3,5]::sentence`
 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`.

3.4 Making Flip Animation

- 1) Define function `Mf(s)`, the state at `s`.
- 2) Put command `Setpara` in the script editor as
`Setpara(subfolder,functionstr(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`.

3.5 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.

3.6 Changing Style

The default styles such as size and color of letters can be changed. See KeTCindyReferenceE.