

nanDECK

Tips & Tricks

In the editor there is the same shortcut for changing upper-lower case of characters used in Microsoft Word, i.e., Shift-F3.

If you put .ttf and .otf files in the same folder with nanDECK executable (or in a subfolder named "fonts"), these fonts will be usable within nanDECK without need to install them.

A right click on the preview (the top-right image) rebuild only the current card, a double click open/close the preview window and a single click select the image (i.e., copy the image in the clipboard, ready to be pasted in another software).

Drag and drop a spreadsheet in nanDECK, it will add a LINK line with that file and ask you if you want to create an automatic layout with the data from the spreadsheet.

A right click on the "Validate deck" button does a validate and a build, a wheel click on the same button does a validate, build, and print preview.

F11 over a font name replaces it with the next font in alphabetical order, CTRL+F11 with the previous.

Right click on a keyword opens the window for modifying its parameter, right click on an empty line opens the menu for cut-copy-paste and adding a keyword.

If you save in the same folder with the executable a file named **shortcuts.txt** with lines of code you can recall them with a keyboard shortcut. Example:

```
r:RECTANGLE = 1, 0, 0, 100%, 100%, #0000FF
T:FONT = ARIAL, 32, , #000000
TEXT = 1, "Test", 0, 0, 100%, 100%
```

The r: means that the 1st line is recalled with CTRL+ALT+r, instead the T: means that the other lines can be recalled with CTRL+ALT+SHIFT+t, and these shortcuts are shown in the main window as buttons.

F2 on a keyword opens the window for inserting parameters. Instead F3 opens the same window, with also the window for inserting coordinates for the rectangle.

The "Auto build" option (on the right, under the card image), which you can also enable/disable with F5, rebuilds the current card every time you change the script (even by just one character), therefore you can see the result of your writing in real time.

When you have a script with TEXT (and FONT) lines and you need to convert them to HTMLTEXT (and HTMLFONT), you can do it automatically, with Insert → Convert TEXT in HTMLTEXT.

If you prefer a single button to do the validate and the build, the option is in Config → Main → "Validate & Build buttons": Two buttons / One button.

nanDECK is limited to ~600 standard cards (6x9cm at 300 DPI). If you need more, you can switch between RAM and disk usage, in Config → Main → Deck file location → to disk (remember to reboot). Note, on disk it is quite slower than on RAM.

Enabling the "Partial" option (on the right of the window, shortcut F10) means that the deck is built from the beginning of the script up to the line where the cursor is, in the editor.

The T flag in IMAGE means that the top-left pixel color in the image is treated as transparent when drawing an image, and therefore it works with every format, even those that do not support transparency. Instead, the N flag reads and uses the transparency info in a png image, and for this format it is way better to use N instead of T; currently the N flag is enabled by default (is the option Config → Language → Use the N flag with png image files).

If there are LINK line(s) in your script, the "Linked data" button opens the corresponding spreadsheet (using the default installed software).

If you want to change the default keyboard shortcuts, go to Config, and click on the "Save keys" button, then you can open the *keys.txt* file (in the same folder with the executable) and modify it. For example, the first line is:

```
actNew,Alt+N
```

And you can change it into:

```
actNew,Alt+A
```

You can use multiple VISUAL..ENDVISUAL structure, in this case when you click on the Visual Editor button, the section opened in the visual editor is the section closest to the editor cursor. Example:

```
VISUAL=, 10, 10  
RECTANGLE=1,0%,0%,100%,100%,#FF0000  
ENDVISUAL
```

```
VISUAL=, 10, 10  
ELLIPSE=1,0%,0%,100%,100%,#00FF00  
ENDVISUAL
```

In the visual editor you will see a rectangle or an ellipse, but both would be drawn in the resulting card.

To enable/disable blocks of text in the editor, you can select the lines and press the "Add" button to add the comment character at the start of the lines, "Rem" button to remove comment character, and "Com" button to switch between commented and uncommented status (or you can use SECTION..ENDSECTION structures).

When not specified by a CARD line, the number of cards in a deck is set equal to the number of lines in a linked spreadsheet. If there is more than one spreadsheet, the largest one is taken.

If you leave the range empty in a directive, that directive is drawn on all the deck, i.e., the null range equals the whole deck.

.txt is the default extension for script file; you can instead select .nde as the extension in Config, and if you click on the "Register extension", you will assign nanDECK as the program to open .nde files when you double click on them.

[label] or {label?} is the Nth element in a sequence (with N the number of a card), and [[label]] or {{label?}} is the content of a label with a name equal to the Nth element in a sequence.

When the cursor is on a parenthesis, pressing CTRL+SHIFT+B moves the cursor to the corresponding one.

Labels are evaluated only in the validation step; therefore, this script does not work:

```
[a]=0  
rectangle=1-10,0,[a]%,10%,10%,#FF0000  
[a]=[a]+10
```

If you need a true variable, use a counter (like A for an integer value, AA for a floating one).

```
counter=1,A,0  
rectangle=1-10,0,{A}%,10%,10%,#FF0000  
counter=1-10,A,A+10
```

The Nth element in a sequence is paired with the Nth card in a range, if you want instead to pair it with Nth card in a deck, add a BASERANGE=,ON to your script. Example, this prints "1" on third card:

```
[a]=1|2|3|4  
FONT=arial,32,,#000000  
TEXT=3,[a],0,0,100%,100%
```

Instead, this script prints "3" on third card:

```
BASERANGE=,ON  
[a]=1|2|3|4  
FONT=arial,32,,#000000  
TEXT=3,[a],0,0,100%,100%
```

Note that BASERANGE is ON by default (is the option Config → Language → Default ON for BASERANGE).

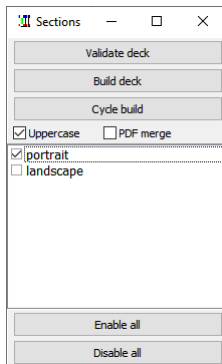
If you need a card with a different size from the others, to be saved separately or used to draw something to be copied to the standard cards, you can use card 0 (i.e., the "canvas").

Use example for SECTION:

```
SECTION=portrait,ON
    cardsize=6,9
ENDSECTION
SECTION=landscape,OFF
    cardsize=9,6
ENDSECTION

font=arial,32,,#000000
text=1,"This is 1st line of text",0,0,100%,50%,center,wwcenter
text=1,"This is 2nd line of text",0,50%,100%,50%,center,wwcenter
```

When you validate it, this window appears:



If the checkbox is on portrait the script is executed with CARDSIZE=6,9, if the checkbox is on landscape the script is executed with CARDSIZE=9,6

Another example:

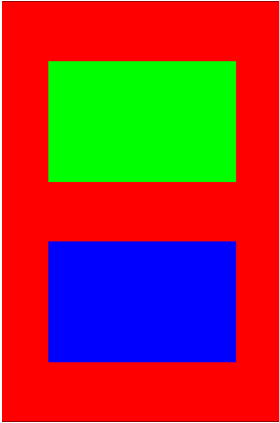
```
SECTION=portrait,ON,0
    cardsize=6,9$
    savepdf=portrait.pdf
ENDSECTION
SECTION=landscape,OFF,0
    cardsize=9,6
    savepdf=landscape.pdf
ENDSECTION

font=arial,32,,#000000
text=1,"This is 1st line of text",0,0,100%,50%,center,wwcenter
text=1,"This is 2nd line of text",0,50%,100%,50%,center,wwcenter
```

A FRAME...ENDFRAME structure is a fast way to create frames, naming them with a single character. Example:

```
FRAME
AAAAAA
BBBBBA
BBBBBA
AAAAAA
ACCCCA
ACCCCA
AAAAAA
ENDFRAME

rectangle=1,<A>,#FF0000
rectangle=1,<B>,#00FF00
rectangle=1,<C>,#0000FF
```



You can work simultaneously with different scripts on multiple tabs, with CTRL+N add a new tab, with CTRL+F4 close the current tab.

If you move the editor cursor to a label, color, or image file, you will see its respective content at the bottom of the screen.

If you need to add a flag to every keyword in one script, you can use a FLAGS line. Example, to add the E flag to all HTMLTEXT lines:

```
FLAGS=,HTMLTEXT,E
```

With CTRL+G you can go to a specific line in the script, with F6 you can go to a specific card in the deck.

In the main editor you can create a bookmark with CTRL+SHIFT+1 (or 2..9) and go to that bookmark with CTRL+1 (or 2..9).

If you need a range with only even, or odd cards, instead of writing all the numbers you can use a syntax like this:

```
RECTANGLE=1-100$ab>a, 1, 1, 4, 4, #FF0000
```

i.e., for each couple of card ab, only the card a go in the range.

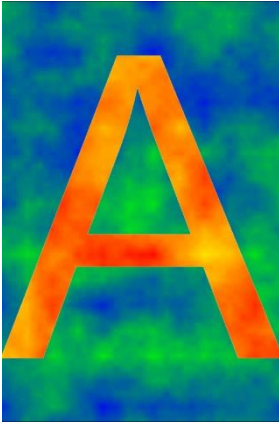
The exact boundaries of a text can be detected with a TEXTLIMIT line and TT, TL, TR, and TB variables. Example:

```
font=arial,16,,#000000
textlimit=1
text=1,"This is a test",0,0,100%,100%,center,center
rectangle=1,tl,tt,tr-tl,tb-tt,#FF0000,empty,1%
```

This is a test

A & in colors (with a number for the quantity of applications) create a plasma effect using specified colors, example:

```
font=arial,256,,#FF0000#FFFF00&8,#00FF00#0000FF&8
text=1,"A",0,0,100%,100%
```



If you want to randomize a sequence, add an N before the name. This example prints a random number from 1 to 6:

```
n[d]=1|2|3|4|5|6  
font=arial,128,,#000000  
text=1,[d],0,0,100%,100%
```

A[label]=... creates a sequence sorting the elements in ascending order, Z[label]=... in descending order.

The P flag in IMAGE keeps the proportions of an image, but it can leave empty zones.

```
image=1,The_Blue_Marble.jpg,0,0,100%,100%,0,P
```



Instead, the C flag keeps the proportions and cuts the image.

```
image=1,The_Blue_Marble.jpg,0,0,100%,100%,0,C
```



The C flag in IMAGE centers the cropped image, if you want you can align to the top edge (adding a U flag), right edge (E flag), bottom edge (S flag), or left edge (W flag). Example:

```
image=1,The_Blue_Marble.jpg,0,0,100%,100%,0,CW
```



With CTRL+SPACE you can insert one of the validated labels.

The Insert → Folder option creates a sequence with all the filenames contained in a folder. Example:

```
{[icons]="  
C:\Users\Nand\Desktop\icons\boar.png|  
C:\Users\Nand\Desktop\icons\deer.png|  
C:\Users\Nand\Desktop\icons\fox.png|  
C:\Users\Nand\Desktop\icons\test1.txt"}
```

If you need to use two (or more) different configurations, you can keep two nanDECK.exe executables in two different folders, each one will have its own configuration (for example, one that runs in RAM and one that runs on disk).

If you have multiple sheets in a spreadsheet, you can load a specific one with this syntax (without the sheet name, the first sheet is loaded):

```
LINK = spreadsheet.xlsx!sheet
```

If you select a color (i.e., the character # and six numbers), pressing the F9 key loads it into the color editor, to be able to edit it.

If you always use a basic template, you can set defaults in the "wiz" button. Example:

```
BORDER=NONE  
UNIT=INCH  
PAGE=8.5, 11, PORTRAIT, HV  
DPI=300  
CARDSIZE=2.5, 3.5  
BASERANGE=, ON
```

The 'New deck wizard' dialog box is shown with the 'Text' tab selected. The settings are as follows:

- Border:** ☒ None
- Border color & Thickness:** Choose (button), Thickness (cm):
- Card size:** Width: 2.5, Height: 3.5
- Page size:** ☒ Letter, Custom (button)
- Custom size:** Width: , Height:
- Page orientation:** ☒ Portrait, ☐ Landscape
- Units:** ☒ inch, ☐ cm, ☐ mm
- Page DPI:** ☒ 300, ☐ 600, ☐ 150, Custom (button)
- Number of cards:**
- Custom DPI:**
- Page Margins:** Left (cm): 1, Right (cm): 1, Top (cm): 1, Bottom (cm): 1
- Center page:** ☒
- Gap between cards:** Horizontal (cm): 0, Vertical (cm): 0
- BASERANGE:** ☒ ON (Nth sequence / Nth card), ☐ OFF (Nth sequence / Nth range)
- Link data file:** (text field), Browse... (button)
- Automatic layout from linked file:** ☐ **Visible grid:** ☐ **Use frames:** ☐
- Buttons:** Cancel, Confirm + Visual, Confirm (highlighted in blue)

CTRL+D makes a copy of the current script in a new tab, saving it with a new name (the old one + 1); useful if you want to keep two versions of a script, while modifying one.

The right column of the main window can be resized, drag left/right the border of the preview box.

In every alpha parameter instead of a single value can be specified an alpha gradient, with @ and an angle. Example:

```
rectangle=1,0,0,100%,100%,#FFFF00#00FFFF@90
font=arial,256,T,#000000
text=1,A,0,0,100%,100%,center,center,0,100@0
```



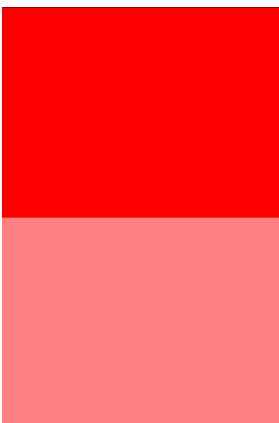
The letter H in a color result in a random value, so #HHHHHH is a random color, while #HH0000 (for example) is a random shade of red.

#ZZZZZZ, #YYYYYY, #XXXXXX, #WWWWWW, and #VVVVVV are variables that can store colors using the COLORS directive. Example:

```
colors=1,#FF0000
ellipse=1,0,0,100%,50%,#ZZZZZZ
rectangle=1,0,50%,100%,50%,#ZZZZZZ
```

In COLORS you can change the brightness or contrast of a color with the syntax: #FFFFFF+brightness+contrast, example:

```
COLORS=,#FF0000-50
rectangle=1,0,0,100%,50%,#FF0000
rectangle=1,0,50%,100%,50%,#ZZZZZZ
```



There are some default settings that are set if there are no commands that indicate otherwise, that is, an empty script looks like it contains these lines:

```
UNIT=CM
PAGE=21,29.7, PORTRAIT
CARDSIZE=6,9
DPI=300
```



```
BORDER=RECTANGLE, #000000
MARGINS=1, 1, 1, 1
```

In a spreadsheet, text in bold, italic, underline is automatically read by LINK and converted in tags for HTMLTEXT.
Example:

	A	B	C
1	Text		
2	This is bold , <i>italic</i> and <u>underline</u> .		
3			
4			
5			

Script:

```
link=data.xlsx
htmlfont=default,arial,16,, #000000
htmltext=1-{{(text)}}, [text], 0, 0, 100%, 100%, #FFFFFF, 0, BE, 100, default
```

Result:

This is **bold**, *italic* and
underline.

You can split a single long line of code into multiple ones using { }. Example:

```
{{[text]}="This is a long text  
divided on multiple lines  
instead of a single one."}
```

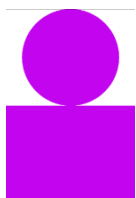
If you want to create only a subset of a deck, you can use a RENDER line (with start/stop cards, or a range as parameters).

If you have several RENDER lines, you will see a window summarizing all these lines upon validation, and you can build a deck by selecting one of them with a double click.

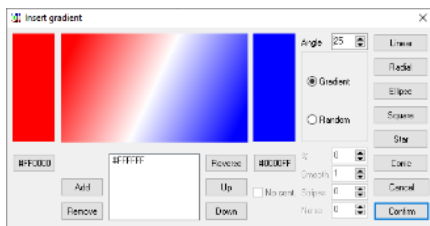
CTRL+ARROWS move the cursor to the next/previous word in the editor, CTRL+TAB select the next word, CTRL+SHIFT+TAB select the previous word.

#LLLLLL is the last color used. Useful for retrieving the same random color, for example:

```
circle=1,0,0,100%,50%, #HHHHHH  
rectangle=1,0,50%,100%,50%, #LLLLLL
```



To see how it looks first, you can blend a gradient in the gradient window by pressing CTRL+F9.



If the size of the font in the editor is too big or too small, you can change it in the Config button (left column "Editor text size").

To speed things up, if you use a LINKMULTI line, the multiple cards are not created individually but copied from the first one. If you need the cards to be created, add a LINKMULCOPY=OFF line before the LINK line.

In a deck with LINKMULTI, a right click on right / left arrows (under the preview) move the preview image to the next / previous different card.

In ICONS, you can add a "backspace" with the < symbol, therefore you can draw more than a single icon in the same space (providing it has a transparency layer). Example:

```
icon=1,a,crossed-bones.png
icon=1,b,star-prominences.png
```

```
icons=1,"ab",0,0,100%,50%,20%,20%,0,PN
icons=1,"bbb<<<aaa",0,50%,100%,50%,20%,20%,0,PN
```



In ICONS, you can add a "newline" with the > symbol. Example:

```
icon=1,a,crossed-bones.png
icon=1,b,star-prominences.png
```

```
icons=1,"b>aa>bbb",0,0,100%,100%,20%,20%,0,PN
```



if you need to know the coordinates of a specific icon, you can use the COOICON function. Example:

```
icon=1,a,crossed-bones.png
```

```
icon=1,b,crossed-bones.png
```

```
icons=1,"aba",0,0,100%,100%,20%,20%,0,PN
```

```
ellipse=1,cooicon(b),#00FF00,empty,3%
```



If there are multiple icons that correspond to the icon used in COOICON, its result is the combined area of all the icons. Example:

```
icon=1,a,crossed-bones.png
```

```
icon=1,b,crossed-bones.png
```

```
icons=1,"baa",0,0,100%,100%,20%,20%,0,PN
```

```
ellipse=1,cooicon(a),#00FF00,empty,3%
```

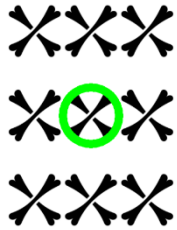


To get the coordinates of a specific icon among the same codes, use the 5th parameter in COOICON. Example:

```
icon=1,a,crossed-bones.png
```

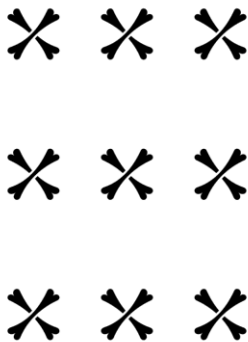
```
icons=1,"aaa>aaa>aaa",0,0,100%,100%,20%,20%,0,PN
```

```
ellipse=1,cooicon(a,,,5),#00FF00,empty,3%
```



Usually, the images in ICON are grouped at the center (or in a different position with the alignment parameters), if you add the WH flags the images are distributed horizontally (flag W) and/or vertically (flag H). Example:

```
icon=1,a,crossed-bones.png
icons=1,"aaaaaaaaa",0,0,100%,100%,20%,20%,0,PNHW
```



If you want a single pdf file with cards or tokens of different sizes, you can use more than one script, save all the cards with SAVE, and use a final script to merge all the image files with a MOSAIC line, or you can save the single pdfs with PDFSAVE lines and merge them with a MERGEPDF line.

The PDFMERGE function can be used not only to merge two or more pdf files, but also to split a pdf into individual pages. Example:

```
[pdf]%, (a), 1, PDFPAGES(source.pdf) = PDFMERGE(split(a).pdf, source.pdf, (a))
```

Of the four buttons “PDF”, “CP”, “GIFa”, and “TIFF”, only the last one (to save a multipage TIFF of the deck) does not have an equivalent directive, for the other three it is better to use the corresponding one (is more practical):

“PDF”, to save a PDF of the deck → SAVEPDF

“CP” to convert PDF to images → LOADPDF (and SAVE)

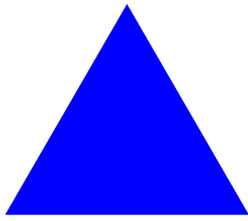
“GIFa” to save an animated GIF of the deck → SAVEGIFA

the elements in a sequence repeat themselves if you ask for an index not present, for example the output of a sequence of two elements in card #3 is the 1st element, in card #4 is the 2nd, in card #5 is the 1st and so on...

SHIFT+CTRL+D duplicates the current line in the editor, and it can be used also on multiple lines, selecting them before using this shortcut.

If you need an equilateral triangle, instead of using a TRIANGLE directive (which needs the coordinates of the three vertices) you can use a POLYGON directive. Example:

```
polygon=1,0,0,100%,100%,3,0,#0000FF
```



The D is an exception for the counters, as a default it gives a random number between 1 and 6 (i.e., 1d6). You can specify a different number of dice (for example 2d or 3d), or a different number of faces (d10 or d20), or both (2d10).

Window placement is saved on exit and restored on start; if something goes wrong (for example you disconnect a monitor) and the window appears offscreen you can use CTRL+T to restore it (or in extreme cases, you can delete the configuration file nandeck.ini, in the same folder as the executable).

Each object in the visual editor or the virtual table has eight handles all around to change its size, if you think that they are too big or too small, you can change their size with two options in Config → Main → “Handle size (Visual editor)” (or “Handle size (Table)” for the virtual table).

If you need a curved text, you can use a BEZIER (with EDGE=1,NULL to not show the line) and Z flag in FONT. Example:

```
edge=1,null
bezier=1,3,1,6,1,0,8,3,8,#000000,0.1
font=arial,32,TZ,#000000
text=1,"This is curved text",0,0,100%,100%
```



If you want to create a sequence without using a spreadsheet, you can write it directly in the script, using one of these methods:

```
[test]=one|two|three|four|five
```

--or--

```
sequence=test
one
two
three
four
five
endsequence
```

--or--

```
sequence
test|one
test|two
test|three
test|four
test|five
endsequence
```

First method for short sequences, second for longer ones, and third when you have more than one sequence (with the same length).

You can add expressions in texts, to be calculated when printed, enclosing them in { } (for TEXT) or {{ }} (for HTMLTEXT); and you can recall also the content of a sequence, with the syntax {label?}. Example:

```
[a]=1|2|3
font=arial,32,,#000000
text=1-{{a}},{{a?}*2},0,0,100%,100%
```

--or--

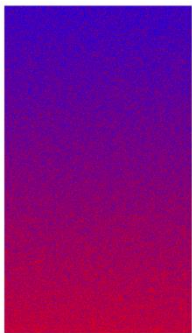
```
[a]=1|2|3
HTMLFONT=font1,arial,32,,#000000,CENTER
HTMLMARGINS=font1,0,0,0,0,CENTER
HTMLTEXT=1-{{a}},"{{{a?}*2}}",0,0,100%,100%,#FFFFFF,0,E,100,font1
```

If you need grids on your charts, note that GRID and HEXGRID (for rectangular and hexagonal grids, respectively) have only basic functionality, because frames created with FRAMEBOX and FRAMEHEX (later introduced) are much more flexible.

To simulate a higher resolution without changing it (or using higher DPI) you can add an OVERSAMPLE=2, which creates a card at twice the current resolution, then scales it to half (but note that render time quadruples).

To mix two kinds of gradients, draw the first and add the second with a partial transparent layer. Example:

```
rectangle=1,1,1,4,7,#FF0000#0000FF@90
layer=50
rectangle=1,1,1,4,7,#FF0000#0000FF
endlayer
```



By default, in ICONS the images are grouped in the center of the rectangle. Example:

```
icon=,F,fox.png
icon=,B,boar.png
icon=,D,deer.png

[symb]=FBD

rectangle=,0,0,100%,20%,#B8FFB8
icons=1-{{symb}},[symb],0,0,100%,20%,20%,20%,0,PN
```



With the W flag you can distribute them horizontally, and with H vertically:

```
icon=,F,fox.png
icon=,B,boar.png
icon=,D,deer.png
```

```
[symb]=FBD
```

```
rectangle=,0,0,100%,20%,#B8FFB8
icons=1-{(symb)},[symb],0,0,100%,20%,20%,20%,0,PNW
```



Unicode characters can be used in HTMLTEXT with codes:

```
htmlfont=default,arial,16,,#000000
```

```
htmltext=1,"Aries  &#9800;<br>Taurus  &#9801;<br>Gemini
&#9802;",0,0,100%,100%,#FFFFFF,0,BE,100,default
```

Aries ♈
Taurus ♉
Gemini ♊

But it is more convenient to use HTMLKEY to indicate correspondences between keys and codes, for example, (aries):

```
htmlfont=default,arial,16,,#000000

htmlkey=,"(aries)","&#9800;"
htmlkey=,"(taurus)","&#9801;"
htmlkey=,"(gemini)","&#9802;"

htmltext=1,"Aries (aries)<br>Taurus (taurus)<br>Gemini (gemini)",0,0,100%,100%,#FFFFFF,0,BE,100,default
```

This way you can also change the font:

```
htmlfont=default,arial,16,,#000000
htmlfont=emoji,segoe ui emoji,16,,#000000

htmlkey=,"(aries)","&#9800;",emoji
htmlkey=,"(taurus)","&#9801;",emoji
htmlkey=,"(gemini)","&#9802;",emoji

htmltext=1,"Aries (aries)<br>Taurus (taurus)<br>Gemini (gemini)",0,0,100%,100%,#FFFFFF,0,BE,100,default
```

Aries 🈶
Taurus 🈷
Gemini 🈸

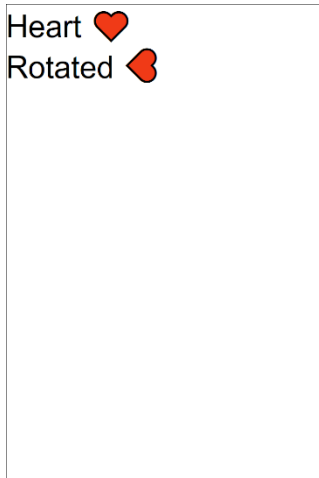
If you need a rotated emoji (or another character), there is an angle parameter in HTMLFONT. Example:

```
htmlfont=default,arial,16,,#000000
htmlfont=emoji,segoe ui emoji,16,,#000000
htmlfont=emoji90,segoe ui emoji,16,,#000000,,0,0,0,,,0,0,,0,90

htmlkey=,"(heart)","&#9829;",emoji
htmlkey=,"(heart90)","&#9829;",emoji90
```



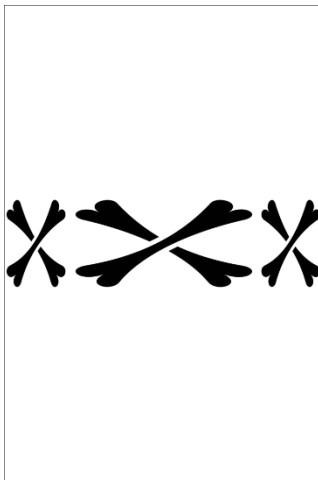
```
htmltext=1,"Heart (heart)<br>Rotated  
(heart90)",0,0,100%,100%,#FFFFFF,0,BE,100,default
```



When using DUPLEX or FOLD, it is normal to see more cards than those present in a spreadsheet, because the program needs to rearrange the cards for printing and does so by copying them to new higher positions (and the PRINT line hides the original cards). For this reason, when these instructions are used, a CARDS line should not be specified, to leave the program free to generate the necessary cards without limits.

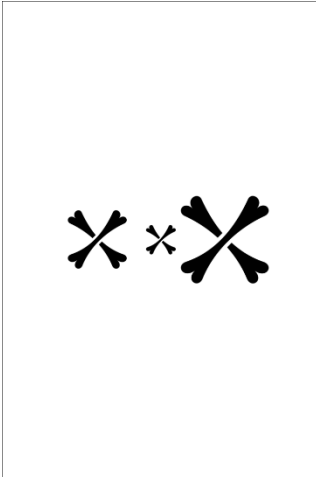
The size of the icons in ICONS is always the same, but you can concatenate more than one space (and therefore create larger spaces) by using the character _ before a character that identifies an icon. Example for a normal icon, a triple-width one, followed by a normal icon:

```
icon=1,a,crossed-bones.png  
icons=1,"a__aa",0,0,100%,100%,20%,20%,0,N
```



You can use the 4th and 5th parameters in ICON to change the size of the image (the default value is 100):

```
icon=1,a,crossed-bones.png,100,100  
icon=1,b,crossed-bones.png,50,50  
icon=1,c,crossed-bones.png,150,150  
icons=1,"abc",0,0,100%,100%,20%,20%,0,PN
```



If you see a card created in a random way (for example with MANDALA) that you want to save, but it's too late to add a SAVE line (because a build will destroy that specific result) you can save the card with the "Save as..." button in the "Card preview" option (or with the "Save images" button in the main window).

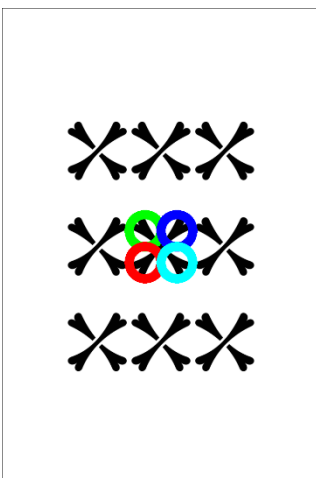
If you have a single LINK line in your script, the auto-layout feature (CTRL+F1) automatically creates a script with all the data in the spreadsheet, taken from 1st column (top of the card) to the last column (bottom of the card). If you want to try some variations, use CTRL+SHIFT+F1, and the data would be positioned each time in a different way.

In Windows, all the paths are defined with the backslash between the name of the various folders and the filename, in nanDECK you can use both the slash and the backslash, since they are converted internally into a backslash.

The SAVEPDF directive doesn't have a range parameter, but if you want to create a pdf consisting of only a few cards, you can add a PRINT=range line to the script.

The COOICON parameters allow you to choose only one section of an area of an icon, for example TL, TR, TB, and TR are the four quadrants:

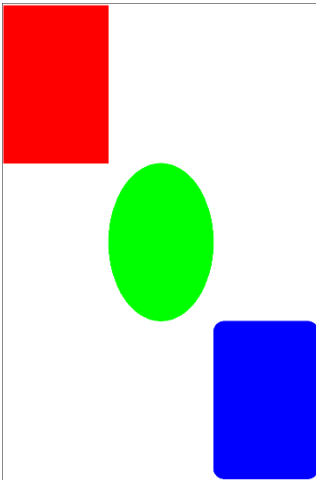
```
icon=1,a,crossed-bones.png
icons=1,"aaa>aaa>aaa",0,0,100%,100%,20%,20%,0,PN
ellipse=1,cooicon(a,tl,50,50,5),#00FF00,empty,3%
ellipse=1,cooicon(a,tr,50,50,5),#0000FF,empty,3%
ellipse=1,cooicon(a,bl,50,50,5),#FF0000,empty,3%
ellipse=1,cooicon(a,br,50,50,5),#00FFFF,empty,3%
```



An ICONS line does not have to draw something to have the coordinated calculated, therefore you can use the COOICON function without ICON lines. Example:

```
icons=1,abcdefghi,0,0,100%,100%,33%,33%
```

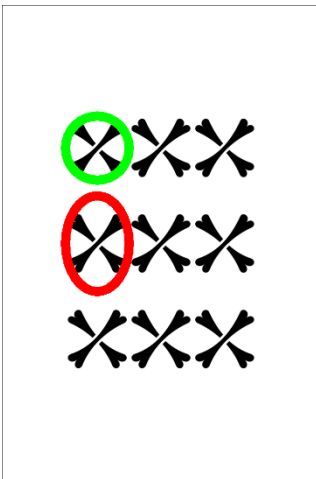
```
rectangle=1,cooicon(a),#FF0000
ellipse=1,cooicon(e),#00FF00
roundrect=1,cooicon(i),#0000FF
```



When there is a P flag in ICONS, COOICON returns the size of the current icon drawn, but if you add an O flag, it returns the size of the area before the icon is adjusted. Example:

```
icon=,a,crossed-bones.png
icon=,b,crossed-bones.png
icon=,c,crossed-bones.png
icons=1-2,"abc>abc>abc",0,0,100%,100%,20%,20%,0,PN
```

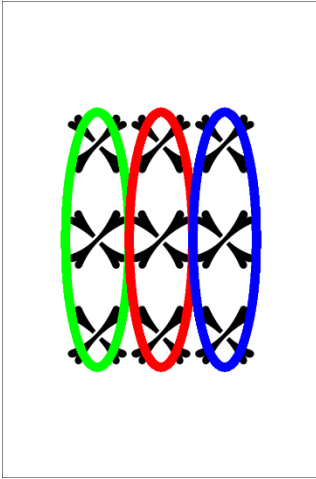
```
ellipse=1,cooicon(a,,,1),#00FF00,empty,3%
ellipse=1,cooicon(a,O,,2),#FF0000,empty,3%
```



If you have multiple icons with the same key and you don't specify an index, the COOICON function returns the sum of all icon areas. Example:

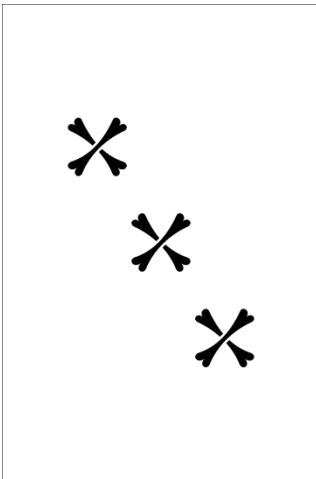
```
icon=,a,crossed-bones.png
icon=,b,crossed-bones.png
icon=,c,crossed-bones.png
icons=1-2,"abc>abc>abc",0,0,100%,100%,20%,20%,0,PN
```

```
ellipse=1,cooicon(a),#00FF00,empty,3%
ellipse=1,cooicon(b),#FF0000,empty,3%
ellipse=1,cooicon(c),#0000FF,empty,3%
```



If you use a code with ICONS that you have not defined with an ICON line, nothing is drawn in that position.
Example:

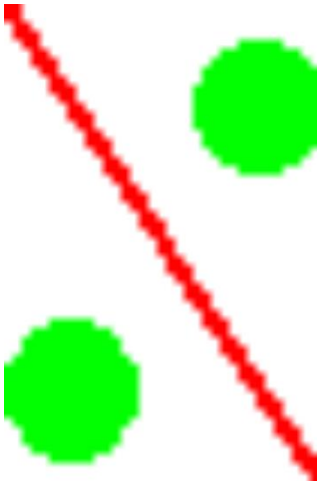
```
icon=,a,crossed-bones.png
icons=1,"axx>xax>xxa",0,0,100%,100%,20%,20%,0,PN
```



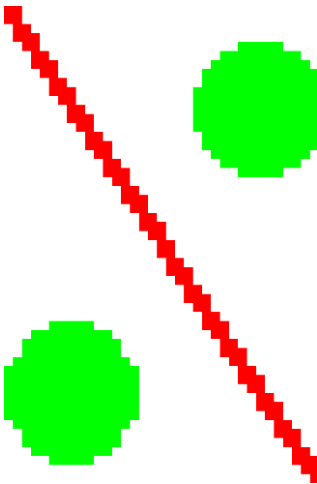
If you want to make sure you don't lose your changes, you can activate the auto-save option (in Config → Interface → Auto-save script on "Build deck", whereby the script is automatically saved on every build. And for added security, you can save your scripts in folders synced to a cloud service, like OneDrive or Dropbox.

A low value in OVERSAMPLE gives the card a pixelated effect. Example:

```
oversample=0.05
border=none
line=1,0,0,100%,100%,#FF0000,5%
circle=1,60%,0,40%,40%,#00FF00
circle=1,0,60%,40%,40%,#00FF00
```



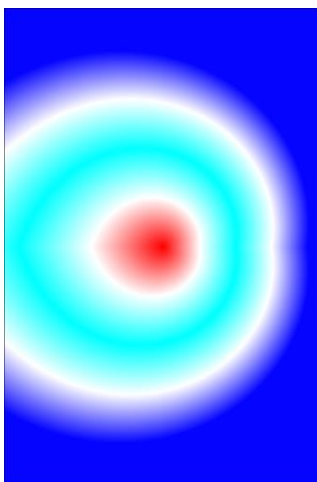
And if you want to remove the dithering, add an IMAGEFILTER=NEAREST line.



In the color window (F9) you can read a color from another point of the screen with the "Pick" button (or CTRL+P shortcut).

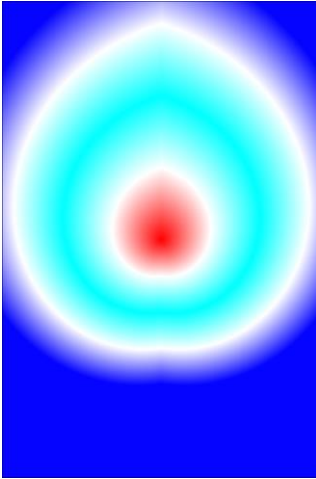
For star gradients you can have a single "point", or better, a single "lobe". Example:

```
rectangle=1,0,0,100%,100%,#FF0000#FFFFFF#00FFFF#FFFFFF#0000FF@401
```



And you can rotate the star adding another @angle parameter. Example:

```
rectangle=1,0,0,100%,100%,#FF0000#FFFFFF#00FFFF#FFFFFF#0000FF@401@270
```



By dragging and dropping a folder into nanDECK, a script is created to print the images contained in the folder.
Example:

```
BORDER=NONE
UNIT=INCH
PAGE=8.5,11, PORTRAIT, HV
DPI=300
CARDSIZE=2.5,3.5
BASERANGE=, ON
[img]=DIRFILES("G:\Temp\*. *")
IMAGE=1-{ (img) }, [img], 0,0,100%,100%
```

To draw an image in a shape different from a rectangle, you can use a BRUSH directive and a POLYGON (or another directive). Example:

```
cardsize=6,6
brush=1,custom,earth.jpg,6,6
star=1,0,0,100%,100%,8,0,40, #FFFFFF
```



If you need multiple copies of the same card and you have the data in a spreadsheet, simply add a column with the number of copies (e.g., with the name count) and insert a LINKMULTI=count line before the LINK line. Or, if you need the same number of copies for every card, just add a LINKMULTI=number line.

You can use a layer to draw a partial transparent rectangle but remember that by default the color of the top left pixel is treated as transparent, so a rectangle across the entire card is not visible, unless a CHROMAKEY line is used first. Example:

```
chromakey=null
layer=50
  rectangle=1,0,0,100%,100%, #FF0000
endlayer
```

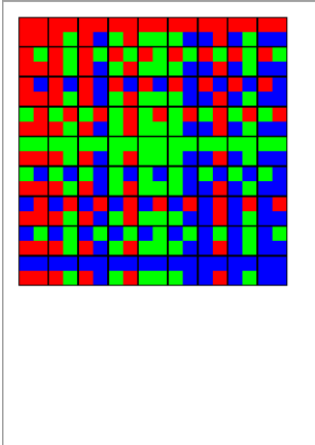
With the combinatorial engine it is possible to generate very quickly tiles with all possible color combinations. Example, a tile with four quadrants of three colors:

```
cardsize=2,2
pr[comb]4=a|b|c
```

```

[cola]=#FF0000
[colb]=#00FF00
[colc]=#0000FF
[all]=1-{(comb)}
[frame]=framebox(0,0,100%,100%,50%,50%,N)
rectangle=[all],<frame1>,[col[comb:1,1]]
rectangle=[all],<frame2>,[col[comb:2,1]]
rectangle=[all],<frame3>,[col[comb:4,1]]
rectangle=[all],<frame4>,[col[comb:3,1]]
rectangle=[all],0,0,100%,100%,#000000,empty,5%

```

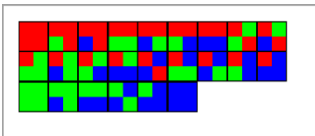


In the above example, if I want to remove duplicate tiles due to rotations, it is enough to put an x in the definition of permutations with repetition (i.e., prx instead of pr).

```

cardsize=2,2
prx[comb]4=a|b|c
[cola]=#FF0000
[colb]=#00FF00
[colc]=#0000FF
[all]=1-{(comb)}
[frame]=framebox(0,0,100%,100%,50%,50%,N)
rectangle=[all],<frame1>,[col[comb:1,1]]
rectangle=[all],<frame2>,[col[comb:2,1]]
rectangle=[all],<frame3>,[col[comb:4,1]]
rectangle=[all],<frame4>,[col[comb:3,1]]
rectangle=[all],0,0,100%,100%,#000000,empty,5%

```

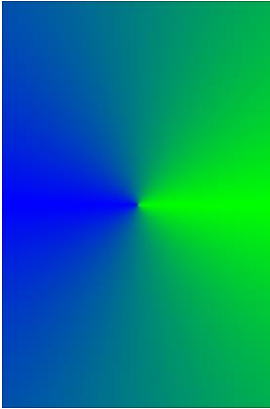


For speed, if a LINKMULTI line is specified, the cards are not created one by one, but the first card of a multiple is copied onto the others. In case the cards must be created one by one because there are lines that make modifications to the individual cards, you must add this line at the beginning of the script:

```
LINKMULCOPY=OFF
```

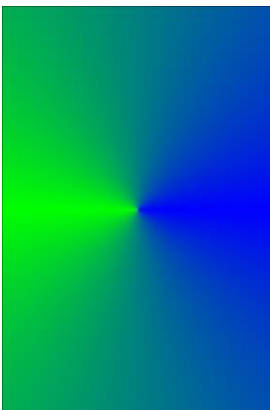
In a conical gradient (using angle @364) if you use the same color at the start and end of the sequence you will not see any jumps in the color sequence. Example:

```
rectangle=1,0,0,100%,100%,#00FF00#0000FF#00FF00@364
```



To change the orientation of a conic gradient, add a second angle to @364. Example:

```
rectangle=1,0,0,100%,100%,#00FF00#0000FF#00FF00@364@180
```



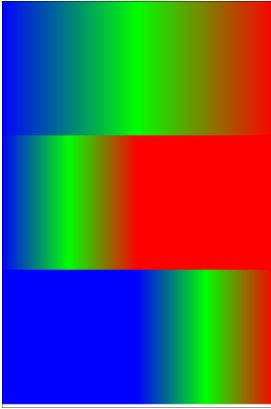
Hollowed text effect:

```
font=arial,64,,#FFFFFF,#A0A0A0
text=1,"Test",-0.1,0.1,6,9,center,center
font=arial,64,0,#000000,#FF0000
text=1,"Test",0,0,6,9,center,center
```



The colors in a gradient are distributed equally, if you need to modify these proportions, you can add more copies of the same color. Example:

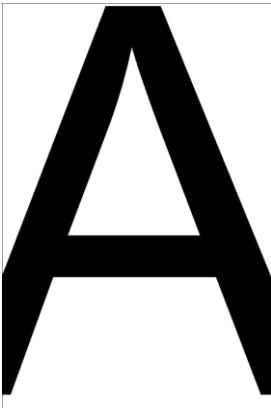
```
rectangle=1,0,0,100%,33%,#FF0000#00FF00#0000FF@0
rectangle=1,0,33%,100%,33%,#FF0000#FF0000#FF0000#00FF00#0000FF@0
rectangle=1,0,66%,100%,33%,#FF0000#00FF00#0000FF#0000FF#0000FF@0`
```

The nandeck.ini file contains all the configuration settings that you find in the “Config” button; if you need two different settings (example, one for cards in RAM, one for cards on disk), you can save two executables in two different folders, each one will use its nandeck.ini file, so with the possibility to use settings different.

For font size you can specify a % (of the whole card) instead of a value. Example:

```
font=arial,100%,,#000000
text=1,"A",0,0,100%,100%
```



The B flag for the transparency in HTMLTEXT works way better than the T flag because it does a different thing, i.e. with the T flag the text is drawn over the background color, and the result is drawn over the card considering the background color as transparent; instead with the B flag the text is directly drawn over the card.

If you use a LINK line with a spreadsheet name that does not exist, nanDECK creates an empty file at the validation step (and you can open it and fill the data with a click on “Linked data” button).

If you need to see the details of a card, in the Card preview there is a slider that allows you to increase the magnification, up to a factor of 8x.

If you realize you’ve made a change to the visual editor that you didn’t want to make, but you’ve already exited by pressing the "Confirm" button, you can always go back to the previous version of the source with the CTRL+Z key.

The command COLORS=,#50%ç50% stores in the variable #ZZZZZ the color contained in the pixel at the coordinates 50%,50% (i.e., at the center of the card). Useful if you want to draw an element with a color that depends on a previously drawn image in the card.

The BLEED command fills a rectangle outward with the colors it finds along the perimeter. Example:

```
cardsize=6.5,9.5
origin=,0.25,0.25,100,100,6,9
rectangle=1,0,0,100%,100%,#FF0000
circle=1,10%,10%,80%,80%,#00FF00
polygon=1,20%,20%,60%,60%,3,0,#0000FF
bleed=1,0,0,100%,100%
```

If you need to add a flag to all directives in a script, you can use a FLAGS line. For example, to add the B flag to all FONT lines after:

```
flags=,font,B
```

The shortcut CTRL+R comments on the current line (or the selected block of text), CTRL+U uncomments it.

F1 shows you the help page on the directive/function in current line in the editor.

For TEXT, the alignment is specified directly, with 7th and 8th parameters; instead, for HTMLTEXT, the horizontal alignment is specified in HTMLFONT (6th parameter), and the vertical alignment in HTMLMARGINS (6th parameter).

The text in HTMLTEXT is written adjacent to the rectangle specified as parameters, if you want to leave some space, instead of reducing the rectangle you can use an HTMLMARGINS line (note that this refers to the font, so you must use also a HTMLFONT line).

The eleventh parameter in HTMLTEXT specifies a font that is applied to the whole text, the twelfth instead specifies a font that is applied to each single paragraph of the text; the difference can be seen for example when drawing the borders for a HTMLBORDERS line, in the first case the border is around all the text, in the second there is a border around each single paragraph. Usually just the first is needed, do not use the same font in both parameters.

When using the twelfth parameter in HTMLTEXT to indicate which font must be applied to each text paragraph, the
 characters are used to understand where a paragraph begins and ends, but this separator can be changed by specifying it in the thirteenth parameter.

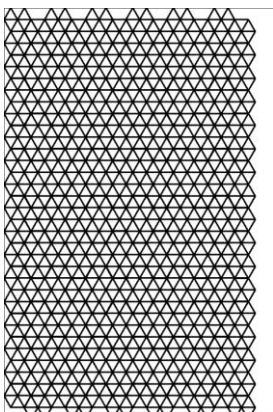
If you need a card with an exact size in pixels, use UNIT=INCH, DPI=1, and the size in pixels. Example for a card 1000x1000 pixels:

```
unit=inch  
dpi=1  
cardsize=1000,1000
```

```
ellipse=1,0,0,100%,100%,#FF0000#0000FF@360  
save=1,result.png
```

If you need a triangle map, you can obtain one drawing hexagons in frames, and add a star with factor 1:

```
[hex]=framehex(0,0,100%,100%,5%)  
polygon=1,<hex>,6,30,#000000,empty,0.5%  
star=1,<hex>,6,30,1,#000000,empty,0.5%
```



If you need to change saturation or brightness of a color, select it, press F9, and change these values with the last two sliders, then confirm the new color.

Remember that the default border is black, 1 pixel wide; if you do not want it, add a BORDER=NONE line (or a BORDER with another color/width).

An easy way to choose a value based on another value in a sequence is to use nested labels. Example:

```
[type]=animal|plant|animal|animal|plant|plant|water

[color_animal]=#FF0000
[color_plant]=#00FF00
[color_water]=#0000FF
```

```
rectangle=1-{ (type) },0,0,100%,100%,[color_[type]]
```

An alternative way to choose a color based on another value in a sequence is to use color variables:

```
[type]=animal|plant|animal|animal|plant|plant|water

if=[type]=animal
  colors=, #FF0000
elseif=[type]=plant
  colors=, #00FF00
else
  colors=, #0000FF
endif

rectangle=1-{ (type) },0,0,100%,100%,#ZZZZZZ
```

The PRINT directive shows you only the cards indicated in the parameters, but the whole deck is still built. If you only want to build some cards, you can use them together with the RENDER directive.

You can link multiple spreadsheets, but if they have the same column names, the latest one replaces the previous ones; instead, if you want to concatenate them, add a LABELMERGE=ON line before the LINK lines.

You can link a sheet (for example named "one") from a spreadsheet (for example named "data.xlsx") with this syntax:

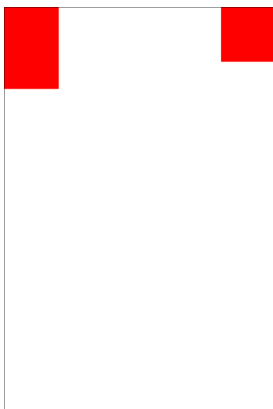
```
LINK=data.xlsx!one
```

To create an animated gif from an image, draw 360 cards (frames) each rotated by 1°, or 180 by 2°, 120 by 3° and so on. Example:

```
border=none
cardsize=5,5
image=1-120,image.png,0,0,100%,100%,$*3,NR
savegifa=image.gif,50,OP
```

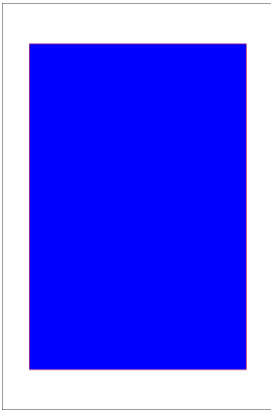
With % it is difficult to draw squares on rectangular papers (where 20% horizontally is different from 20% vertically); for that you can use ORIGIN to change what % means, to make them relative to the card aspect ratio. Example:

```
rectangle=1,0,0,20%,20%,#FF0000
origin=1,0,0,100,100*info(w)/info(h)
rectangle=1,80%,0,20%,20%,#FF0000
```



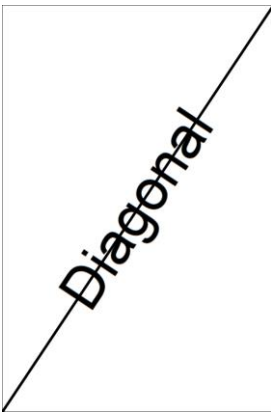
If you don't specify a border in directives like RECTANGLE, with one color is drawn a full rectangle, with two colors is drawn a one pixel-thick border. Example:

```
rectangle=1,10%,10%,80%,80%,#FF0000,#0000FF
```



To draw a text aligned to the diagonal of the rectangle can be done with a G (or D) flag in FONT. Example:

```
font=arial,36,G,#000000  
text=1,Diagonal,0,0,100%,100%  
line=1,0,100%,100%,0,#000000,1%
```



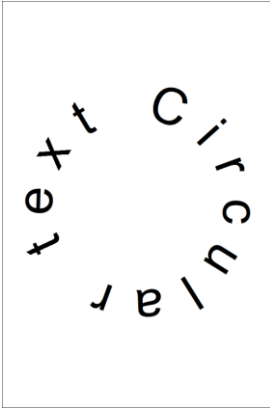
The O flag in font draws only the background, leaving the text empty. Example:

```
line=1,0,0,100%,100%,#000000,10%  
font=arial,256,O,#000000,#FF0000  
text=1,"A",0,0,100%,100%
```



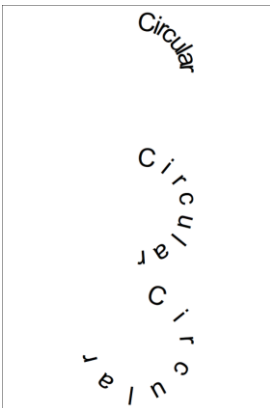
The C flag in FONT draws a text in a circle. Example:

```
font=arial,32,C,#000000  
text=1,"Circular text ",0,1.5,6,6
```



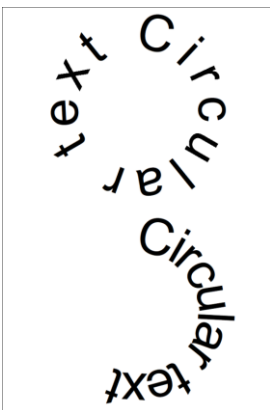
Adding Q/H/E to flag C writes text in a quarter/half/three-quarters of the circumference. Example:

```
font=arial,16,CQ,#000000
text=1,"Circular",1.5,0,3,3
font=arial,16,CH,#000000
text=1,"Circular",1.5,3,3,3
font=arial,16,CE,#000000
text=1,"Circular",1.5,6,3,3
```



By default, the circular text is extended to the entire circumference, however if you set the distance between characters with the 10th parameter in FONT, you can limit the text to an arc. Example:

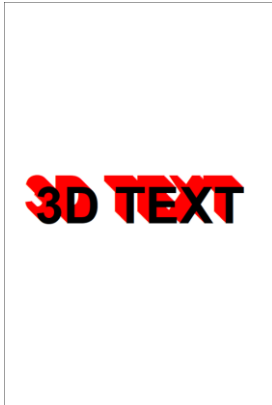
```
font=arial,32,C,#000000
text=1,"Circular text ",0.75,0,4.5,4.5,left
font=arial,32,C,#000000,#FFFFFF,0,0,0,0,0
text=1,"Circular text",0.75,4.5,4.5,4.5,left
```



You can simulate a 3D effect with a loop. Example:

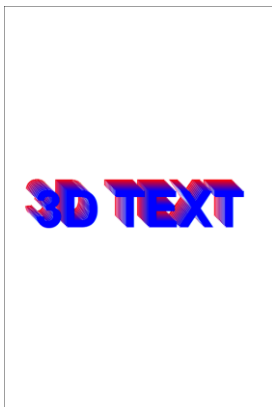
```
font=arial,32,BT,#FF0000
for=a,1,10
  text=1,"3D TEXT",0-a/40,0-a/40,100%,100%
```

```
next
font=arial,32,BT,#000000
text=1,"3D TEXT",0,0,100%,100%
```



You can add a gradient to the 3D effect with the GRADIENTSEQ function. Example:

```
[col]=gradientseq(#FF0000#0000FF@0,10)
for=a,10,1,-1
  font=arial,32,BT,{col?a}
  text=1,"3D TEXT",0-a/40,0-a/40,100%,100%
next
font=arial,32,BT,#0000FF
text=1,"3D TEXT",0,0,100%,100%
```



If you only need the card images as single files, you can disable storing the cards in RAM and thus create an unlimited number of cards with a STORE=,OFF line. Example:

```
store=,off
font=arial,96,,#HHHHHHH,#HHHHHHH
text=1-1000,{S},0,0,100%,100%
save=1-1000,"Card{S}.png",0,0,100%,100%
```

If you need to add year-month-day to the name of file you create, you can use the INFO function. Example:

```
DISPLAY=join("Result_",info(y),"_",info(m),"_",info(a),".png")
```

The size of the right column in the main window can be changed with the mouse, placing it to the right of the editor, where the cursor changes shape, the same can be done for the list with the logs, in the lower part of the window.

By default, expressions in HTMLTEXT are evaluated if delimited by {{ ... }}. If you want to use { ... } (like in TEXT) add this line:

```
expression={,}
```

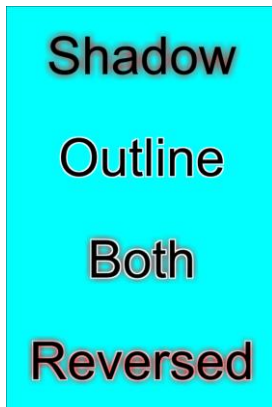
Normally the shadow is drawn below the outline, with the O flag in HTMLFONT the shadow is drawn above. Example:

```
rectangle=1,0,0,100%,100%,#00FFFF
```

```

htmlfont=font1,arial,32,,#000000,center,0,0,0.2,#FF0000
htmlmargins=font1,0,0,0,0,center
htmlfont=font2,arial,32,,#000000,center,0,0,0,#FF0000,#FFFFFF,0.03
htmlmargins=font2,0,0,0,0,center
htmlfont=font3,arial,32,,#000000,center,0,0,0.2,#FF0000,#FFFFFF,0.03
htmlmargins=font3,0,0,0,0,center
htmlfont=font4,arial,32,0,#000000,center,0,0,0.2,#FF0000,#FFFFFF,0.03
htmlmargins=font4,0,0,0,0,center
htmltext=1,"Shadow",0,0,100%,25%,#FFFFFF,0,EB,100,font1
htmltext=1,"Outline",0,25%,100%,25%,#FFFFFF,0,EB,100,font2
htmltext=1,"Both",0,50%,100%,25%,#FFFFFF,0,EB,100,font3
htmltext=1,"Reversed",0,75%,100%,25%,#FFFFFF,0,EB,100,font4

```



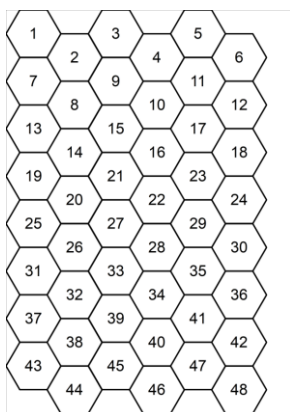
With the FOLD layout it's better to leave some space in the middle of the sheet, along the fold (with the GAP command), so the part with the spine of the fold is cut away; otherwise, it can be unsightly.

If you need hexagonal tokens, the default FRAMEHEX gives you a grid that is hard to cut. Example:

```

cardsize=21,29.7
[h]=framehex(0,0,21,29.7,2)
polygon=1,<h>,6,30,#000000,empty,0.1
font=arial,32,T,#000000
text=1,{°},<h>

```

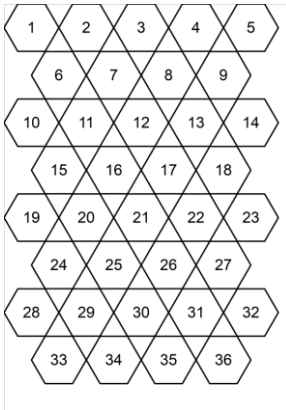


Adding an X flag gives you a layout easier to cut:

```

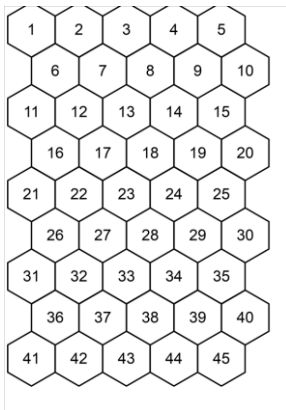
cardsize=21,29.7
[h]=framehex(0,0,21,29.7,2,X)
polygon=1,<h>,6,30,#000000,empty,0.1
font=arial,32,T,#000000
text=1,{°},<h>

```



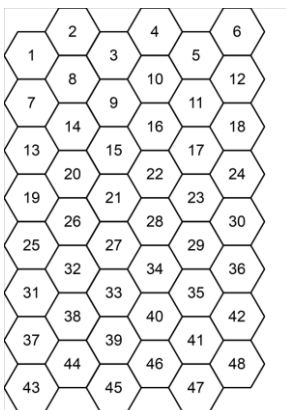
The A flag in FRAMEHEX changes the orientation of the hexagons. Example:

```
cardsize=21,29.7
[h]=framehex(0,0,21,29.7,2,A)
polygon=1,<h>,6,0,#000000,empty,0.1
font=arial,32,T,#000000
text=1,{°},<h>
```



The S flag in FRAMEHEX creates layouts where instead of having a starting hexagon there is half an empty space. Example:

```
cardsize=21,29.7
[h]=framehex(0,0,21,29.7,2,S)
polygon=1,<h>,6,30,#000000,empty,0.1
font=arial,32,T,#000000
text=1,{°},<h>
```

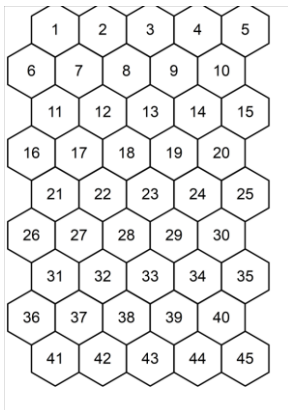


The same, with horizontal lines of hexagons instead of vertical ones:

```
cardsize=21,29.7
[h]=framehex(0,0,21,29.7,2,AS)
polygon=1,<h>,6,0,#000000,empty,0.1
```



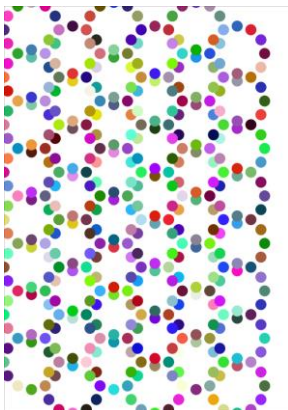
```
font=arial,32,T,#000000
text=1,{°},<h>
```



Frames can be used to create other frames. For example, in this script in every hexagonal frame there are twelve circular frames, used to draw circles:

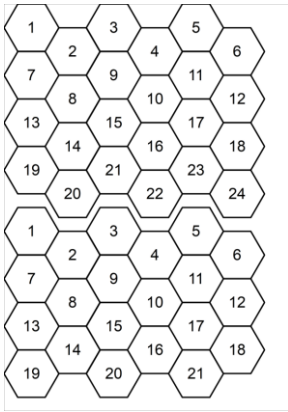
```
cardsize=21,29.7
[h]=framehex(0,0,21,29.7,2,N)
[s°]=frameclock(<h*>,0.8,0.8,12)
ellipse=1,<s*>,#HHHHHH
```

Note the [s°] used to create the sub-frames.



The T flag in FRAMEHEX is for not adding the last hexagon in shifted columns (or lines). Example:

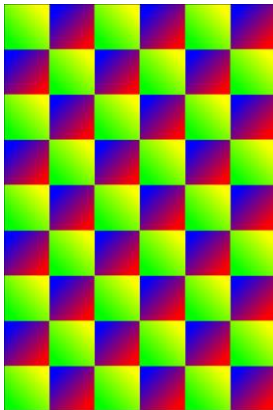
```
cardsize=21,29.7
[h1]=framehex(0,0,21,14.85,2)
[h2]=framehex(0,14.85,21,14.85,2,T)
polygon=1,<h1>,6,30,#000000,empty,0.1
polygon=1,<h2>,6,30,#000000,empty,0.1
font=arial,32,T,#000000
text=1,{°},<h1>
text=1,{°},<h2>
```



By default, the 5th parameter in FRAMEHEX specifies the length of a side of the hexagon, if you want instead to specify the diameter, add the M flag.

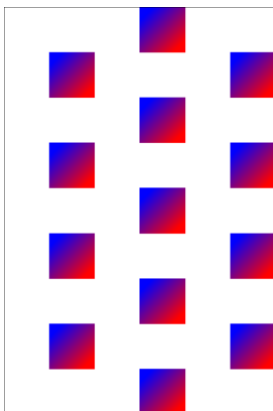
The W and B flags are used to create frames with only half the frames in a FRAMEBOX. Example:

```
[a]=framebox(0,0,6,9,1,1,w)
[b]=framebox(0,0,6,9,1,1,b)
rectangle=1,<a>,#FF0000#0000FF@45
rectangle=1,<b>,#00FF00#FFFF00@135
```



To select a frame every N, use FRAMELIST with & operator. Example (every 4 frames):

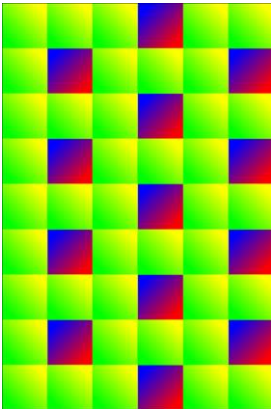
```
[a]=framebox(0,0,6,9,1,1)
[b]=framelist(a&4)
rectangle=1,<b>,#FF0000#0000FF@45
```



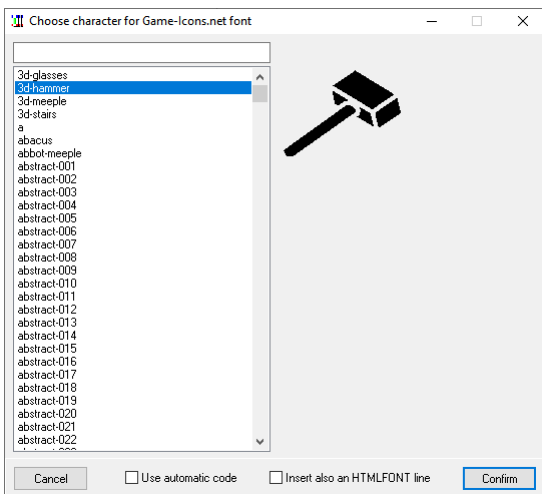
From the previous example, if I want to select frames that are not part of a group, I can use the FRAMESUB function. Example:

```
[a]=framebox(0,0,6,9,1,1)
[b]=framelist(a&4)
[c]=framesub(a,b)
```

```
rectangle=1,<b>,#FF0000#0000FF@45
rectangle=1,<c>,#00FF00#FFFF00@135
```



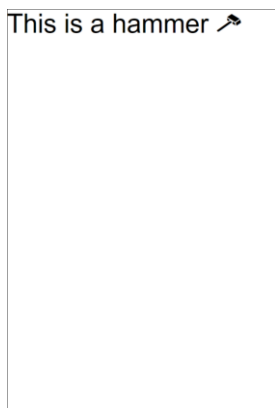
Instead of using single images from game-icons.net, you can use the font, and automate it all using the “Insert” → “Character for game-icons.net font” button, which shows you this window:



Select one character, click on “Use automatic code” and “Insert also an HTMLFONT line”, and you have the first two lines in this example script:

```
HTMLFONT=gameicons,game-icons-net,16,,#000000
HTMLKEY=,(3d-hammer),gameicons_auto,gameicons
```

```
htmlfont=default,arial,16,,#000000
htmltext=1,"This is a hammer (3d-hammer)",0,0,100%,100%,#FFFFFF,0,BE,100,default
```



In the previous feature, without the “Use automatic code” flag, the value of the character is used, i.e., `` for 3d-hammer:

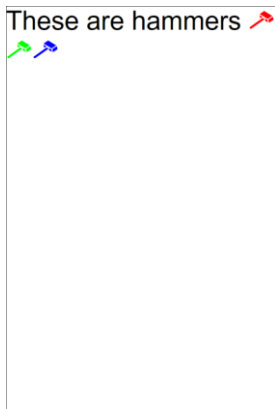
```
HTMLKEY=,(3d-hammer),"&#xe001;",gameicons
```

Instead, "gameicons_auto" in the 3rd parameter means that is used the character specified as name in the 2nd parameter.

An advantage of using game-icons.net fonts instead of images is that you don't need to save files in advance for each color, since you can change the color with an HTMLFONT. Example:

```
HTMLFONT=gameicons_r,game-icons-net,16,,#FF0000
HTMLFONT=gameicons_g,game-icons-net,16,,#00FF00
HTMLFONT=gameicons_b,game-icons-net,16,,#0000FF
HTMLKEY=(3d-hammer_r),"&#xe001;",gameicons_r
HTMLKEY=(3d-hammer_g),"&#xe001;",gameicons_g
HTMLKEY=(3d-hammer_b),"&#xe001;",gameicons_b

htmlfont=default,arial,16,,#000000
htmltext=1,"These are hammers (3d-hammer_r) (3d-hammer_g) (3d-hammer_b)",0,0,100%,100%,#FFFFFF,0,BE,100,default
```

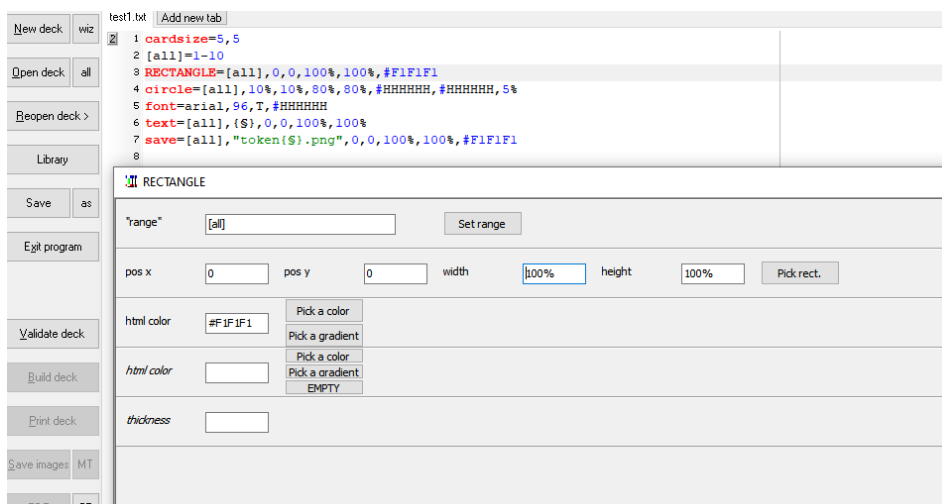


Following the previous script, if you want to use gameicons_auto to skip the actual code, you can use < > tags to differentiate multiple formats. Example:

```
HTMLFONT=gameicons_r,game-icons-net,16,,#FF0000
HTMLFONT=gameicons_g,game-icons-net,16,,#00FF00
HTMLFONT=gameicons_b,game-icons-net,16,,#0000FF
HTMLKEY=(3d-hammer<r>),gameicons_auto,gameicons_r
HTMLKEY=(3d-hammer<g>),gameicons_auto,gameicons_g
HTMLKEY=(3d-hammer<b>),gameicons_auto,gameicons_b

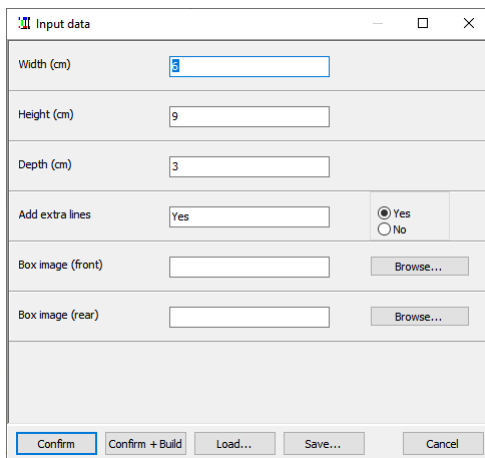
htmlfont=default,arial,16,,#000000
htmltext=1,"These are hammers (3d-hammer<r>) (3d-hammer<g>) (3d-hammer<b>)",0,0,100%,100%,#FFFFFF,0,BE,100,default
```

In a numeric field, in the F2 edit window, you can change the number value with the mouse wheel.



When you use the HV flags in PAGE, to center the result in the page, values in MARGINS are ignored, therefore you do not need to specify that line.

If you want to create a flexible script, to be configured by entering parameters from a window rather than from a file, you can use commands like INPUTTEXT or INPUTCHOICE, like the "Tuckbox" example in the reference:



The parameter values in the window generated by commands such as INPUTTEXT are saved (in a file with the same name as the script and .ini extension) to be re-proposed in the next execution.

If you want to test changing the font, you can do it without changing every single line but using FONTCHANGE. Example:

```
FONTCHANGE=arial,times new roman
```

Instead of reading the data from a spreadsheet, you can alternatively keep them in the script using a SEQUENCE...ENDSEQUENCE structure. Example:

```
SEQUENCE =
Title      |Earth
Image      |Earth.jpg
Description |Earth is the third planet from the Sun.
Radius     |6.371
Orbital Period|365

Title      |Moon
Image      |Moon.jpg
Description |The Moon is Earth's only natural satellite.
Radius     |1.737
Orbital Period|26
ENDSEQUENCE
```

If you click on the “Highlight” button, the element drawn from the current line in the editor is drawn with a red tint; to restore the item to its natural color, click the “Highlight” button again.

After the validation, if you press CTRL+Space you can select a label from a list of all the labels (which also shows you their content).

Aith the data in a SEQUENCE structure, you can duplicate the cards with a MULTI= line. Example:

```
SEQUENCE =
multi=2
Title      |Earth
Image      |Earth.jpg
Description |Earth is the third planet from the Sun.
Radius     |6.371
Orbital Period|365

multi=3
Title      |Moon
```

```
Image      |Moon.jpg
Description|The Moon is Earth's only natural satellite.
Radius     |1.737
Orbital Period|26
ENDSEQUENCE
```

Question: why is the 67 in bold compared to the 66 in this example?

```
[all]=1
rectangle=[all],0,0,100%,100%,#000000

font=arial,8,T,#FFFFFF
text=[all],"POW",75%,0%,15%,10%,center,center,-18
text=[all],"TGH",75%,10%,15%,10%,center,center,-18

font=arial,14,T,#FFFFFF
text=[all],67,85%,5%,15%,10%,center,center,-18
text=[all],66,85%,15%,15%,10%,center,center,-18
```

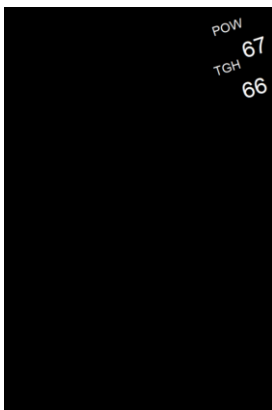


With the T flag, the default point read to determine the transparent color is top left; in this case is read some color different from black (from the “POW” text). To correct, you can read the color in the center with a CHROMAKEY=CENTER line:

```
[all]=1
rectangle=[all],0,0,100%,100%,#000000
chromakey=center

font=arial,8,T,#FFFFFF
text=[all],"POW",75%,0%,15%,10%,center,center,-18
text=[all],"TGH",75%,10%,15%,10%,center,center,-18

font=arial,14,T,#FFFFFF
text=[all],67,85%,5%,15%,10%,center,center,-18
text=[all],66,85%,15%,15%,10%,center,center,-18
```



In the above example, instead of specifying a position to read a color, in CHROMAKEY you can use directly a color the be treated as transparent. Example:

```
[all]=1
rectangle=[all],0,0,100%,100%,#000000
chromakey=#000000

font=arial,8,T,#FFFFFF
text=[all],"POW",75%,0%,15%,10%,center,center,-18
text=[all],"TGH",75%,10%,15%,10%,center,center,-18

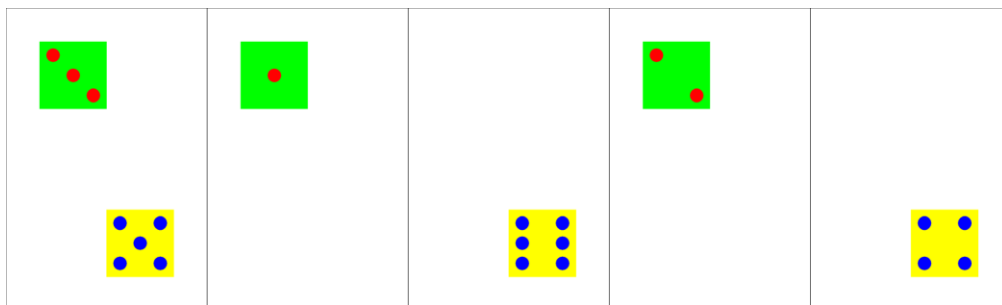
font=arial,14,T,#FFFFFF
text=[all],67,85%,5%,15%,10%,center,center,-18
text=[all],66,85%,15%,15%,10%,center,center,-18
```

The result is the same as above.

The CHROMAKEY command is used when in another command (like FONT or IMAGE) is present the T flag, and it treats the color specified (or the color found in a specific position) in CHROMAKEY as transparent (by default, is the color of the top-left pixel).

This is a macro that draws a face of a d6:

```
macro=face,(r),(x),(y),(w),(h),(d),(colf),(colb)
  rectangle=(r),(x),(y),(w),(h),(colb)
  if=(d)@23456
    ellipse=(r),(x)+(w)/10,(y)+(h)/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
  if=(d)@456
    ellipse=(r),(x)+(w)*7/10,(y)+(h)/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
  if=(d)@6
    ellipse=(r),(x)+(w)/10,(y)+(h)*4/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
  if=(d)@135
    ellipse=(r),(x)+(w)*4/10,(y)+(h)*4/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
  if=(d)@6
    ellipse=(r),(x)+(w)*7/10,(y)+(h)*4/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
  if=(d)@456
    ellipse=(r),(x)+(w)/10,(y)+(h)*7/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
  if=(d)@23456
    ellipse=(r),(x)+(w)*7/10,(y)+(h)*7/10,((w)/10)*2,((h)/10)*2,(colf)
  endif
end
```



The above macro, modified for values 1-9:

```
macro=face,(r),(x),(y),(w),(h),(d),(colf),(colb)
  roundrect=(r),(x),(y),(w),(h),(colb)
  if=(d)@23456789
    ellipse=(r),(x)+(w)/10,(y)+(h)/10,((w)/10)*2,((h)/10)*2,(colf)
```

```

endif
if=(d)@89
    ellipse=(r),(x)+(w)*4/10,(y)+(h)/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@456789
    ellipse=(r),(x)+(w)*7/10,(y)+(h)/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@6789
    ellipse=(r),(x)+(w)/10,(y)+(h)*4/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@13579
    ellipse=(r),(x)+(w)*4/10,(y)+(h)*4/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@6789
    ellipse=(r),(x)+(w)*7/10,(y)+(h)*4/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@456789
    ellipse=(r),(x)+(w)/10,(y)+(h)*7/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@89
    ellipse=(r),(x)+(w)*4/10,(y)+(h)*7/10,((w)/10)*2,((h)/10)*2,(colf)
endif
if=(d)@23456789
    ellipse=(r),(x)+(w)*7/10,(y)+(h)*7/10,((w)/10)*2,((h)/10)*2,(colf)
endif
end

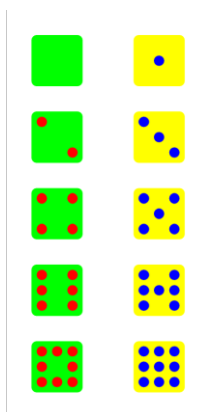
```

Example:

```

cardsize=8,16
face=1,1,1,2,2,0,#FF0000,#00FF00
face=1,5,1,2,2,1,#0000FF,#FFFF00
face=1,1,4,2,2,2,#FF0000,#00FF00
face=1,5,4,2,2,3,#0000FF,#FFFF00
face=1,1,7,2,2,4,#FF0000,#00FF00
face=1,5,7,2,2,5,#0000FF,#FFFF00
face=1,1,10,2,2,6,#FF0000,#00FF00
face=1,5,10,2,2,7,#0000FF,#FFFF00
face=1,1,13,2,2,8,#FF0000,#00FF00
face=1,5,13,2,2,9,#0000FF,#FFFF00

```

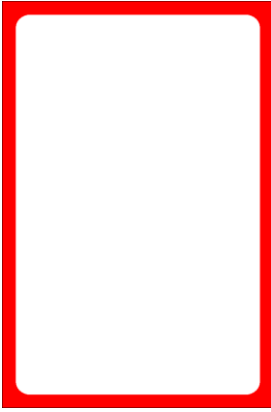


A rectangle (or a rounded rectangle) with a wide border is drawn half inside, half outside, example:

```

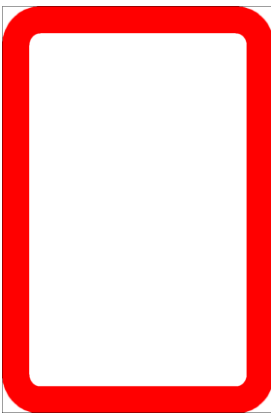
roundrect=1,0,0,100%,100%,#FF0000,empty,10%`

```

With an EDGE=,INSIDE line, all the shapes are drawn inside the coordinates specified:

```
edge=,inside
roundrect=1,0,0,100%,100%,#FF0000,empty,10%
```



If you click on “wiz” button → Miscellaneous → Include labels for HTML colors you will have a list of all the colors (with name and value) from the Insert color (F9) window.

Instead of white you can give a different color to the page (i.e., the margins and the gaps) you can use the 5th parameter in the PAGE command, and note that you can use a sequence, one color for each page.

In addition to the page color, a different color can be set for the page borders, with the 7th parameter in the PAGE command and the 8th to 11th parameters for the border dimensions (left, right, top, bottom).

Another way to draw a border (or other graphic element) on the page is to use PAGESHAPE. Example:

```
page=21,29.7,portrait,hv
font=arial,96,,#HHHHHH,#HHHHHH
text=1-9,{S},0,0,100%,100%

pageshape=0,0,21,1,A,#00FF00
pageshape=20,0,1,29.7,A,#00FF00
pageshape=0,28.7,21,1,A,#00FF00
pageshape=0,0,1,29.7,A,#00FF00
```

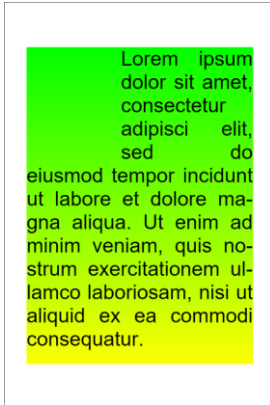
By default, the IMAGE command loads an image file adapting it to all the space indicated (the P and C flags maintain the proportions by enlarging it in a different way), instead with the D flag the file is loaded maintaining the original dimensions (reading the DPI), centering the image in the indicated space.

When an image does not occupy all the indicated space in the IMAGE command, as for example with the D flag, instead of the center it can be aligned to one of the sides of the rectangle, with the U (top), E (right), S (below), and W (left); using two aligns to the relative corner.

The L and R flags in HTMLIMAGE are used to place an image on the left or right side of the HTMLTEXT rectangle, if you use a transparent image you can make the text flow into a non-rectangular space. Example:

```
[text]="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrum exercitationem ullamco laboriosam, nisi ut aliquid ex ea commodi
consequatur."
rectangle=1,0.5,1,5,7,#FFFF00#00FF00@90
```

```
htmlimage=,(a),box_tra.png,2,2,L
htmlfont=default,arial,12,H,#000000,justify
[text_box]=join((a),[text])
htmltext=1,[text_box],0.5,1,5,7,#FFFFFF,0,BE,100,default
```



In the previous example, the H flag in HTMLFONT is used to indicate that words can be divided into syllables, so that the justified alignment is more homogeneous.

To use the small caps format, simply add the A flag to the HTMLFONT command.

The Z flag in HTMLFONT keeps all the text with that font on a single line, useful to be sure that texts (or linked images) are not split on different lines.

You can overlay two HTMLIMAGE images using a negative left margin with the second image. Example:

```
[text]="This is an (h) (d) example."

htmlfont=default,arial,12,,#000000
htmlimage=,(h),heart-wings.png,20%,20%,PM
htmlimage=,(d),drop.png,14%,14%,PM,0,0,-18%
htmltext=1-{(text)},[text],0,0,100%,100%,#FFFFFF,0,BE,100,default
```

This is an  example.

To concatenate labels (both sequences and single values) and constants it is always advisable to use the JOIN function; CONCAT and CONCAT1 also exist, but they are not as flexible (temporally they were added to the program before JOIN).

CONCAT syntax might seem strange, in fact the original purpose was not to deal with character strings or labels but was to create a color gradient by specifying a certain number of repetitions for each color.

The result of a JOIN between sequences of different lengths is a sequence of several elements equal to the longest sequence, and to find the elements of the shortest sequences to match, they are repeated.

JOINIF can be used when there are two alternatives in choosing the value to be concatenated. Example:

```
[num]=0|1|2

[a]=joinif(,[num],<>,1,"coins","coin")
[b]=join("Gain ",[num]," ",[a])
```

The result is that [b] equals “Gain 0 coins|Gain 1 coin|Gain 2 coins”

Functions can always be nested. Example:

```
[num]=0|1|2

[b]=join("Gain ",[num]," ",joinif(,[num],<>,1,"coins","coin"))
```

Normally an empty range indicates the whole deck, but in any case there must be an element that indicates the total number of cards, such as a link line (the number of cards is equal to the number of lines in the spreadsheet), the cards command, or a line with a range. Example, this script doesn't work (because there is an empty range and no other element):

```
[a]=#FF0000|#00FF00|#0000FF
rectangle=,0,0,100%,100%, [a]
```

An inline function is not calculated during validation, so for a script like this to work, it must also have a line indicating the number of cards to be created:

```
[a]=#FF0000|#00FF00|#0000FF
rectangle=autorange(3),0,0,100%,100%, [a]
```

In other words, since the number of cards is established during validation, an interval calculated during the build is not considered to determine how many cards are present.

About the above example, to make it work, you must remove the inline function and change it into a function that is evaluated during the validation step. Example:

```
[a]=#FF0000|#00FF00|#0000FF
[b]=autorange(3)
rectangle=[b],0,0,100%,100%, [a]
```

Spreadsheet column titles may contain formatting (it will be automatically removed in the name of the sequences), but non-standard characters will be converted to Unicode codes, so it is better to either use a LINKUNI=OFF (but this will remove all Unicode codes) or use only standard characters for the titles.

If you indicate in a LINK line a file that doesn't exist, nanDECK will ask you if it should create it.

Spaces between parameters and at the beginning of a command can be freely used. You can write either:

```
rectangle=1-10,0,0,100%,100%,#FF0000
```

--or--

```
rectangle = 1-10, 0, 0, 100%, 100%, #FF0000
```

Normally when a rotation is applied to an image, this is reduced so that it fits completely in the rectangle of the IMAGE command, if instead you want to maintain its dimensions the R flag must be added.

Question: how many permutations (combinations where order is important) are obtained with three colors repeated four times?

```
pr[result]4=A|B|C
```

[result] will contain 81 elements:

```
AAAA
AAAB
AAAC
```

AABA
AABB
AABC
AACA
AACB
AACC
ABAA
ABAB
ABAC
ABBA
ABBB
ABBC
ABCA
ABCB
ABCC
ACAA
ACAB
ACAC
ACBA
ACBB
ACBC
ACCA
ACCB
ACCC
BAAA
BAAB
BAAC
BABA
BABB
BABC
BACA
BACB
BACC
BBAA
BBAB
BBAC
BBBA
BBBB
BBBC
BBCA
BBCB
BBCC
BCAA
BCAB
BCAC
BCBA
BCBB
BCBC
BCCA
BCCB
BCCC
CAAA
CAAB
CAAC
CABA
CABB
CABC
CACA
CACB
CACC
CBAA
CBAB
CBAC

```
CBBA
CBBB
CBBC
CBCA
CBCB
CBCC
CCAA
CCAB
CCAC
CCBA
CCBB
CCBC
CCCA
CCCB
CCCC
```

From the previous example, if the order doesn't matter (combinations instead of permutations), the line becomes:

```
cr[result] 4=A|B|C
```

Result (15 elements):

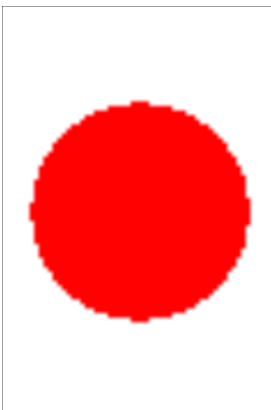
```
AAAA
AAAB
AAAC
AABB
AABC
AACC
ABBB
ABBC
ABCC
ACCC
BBBB
BBBC
BBCC
BCCC
CCCC
```

The default image filter (LINEAR) does a smoothing that is noticeable on low resolution or pixel art images. Example, this is an image at low resolution:



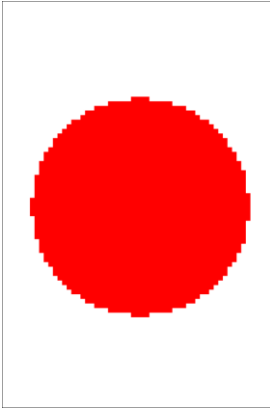
Loaded with an IMAGE:

```
image=1,dot.png,0,0,100%,100%,0,P
```



If you want to remove the blur, use the NEAREST image filter:

```
imagefilter=nearest
image=1,dot.png,0,0,100%,100%,0,P
```



If you need a card of a precise pixel size, you can also use the 10th and 11th parameters of SAVE. Example:

```
ellipse=1,0,0,100%,100%,#FF0000#00FF00#0000FF@360
save="card{$}.png",0,0,100%,100%,1000,1000
```

When you are editing the parameters of a directive in the F2 window, if you double click on a field you can insert one of the labels that have been defined.

If you need to merge two scripts, instead of changing all ranges of the second script, you can use a RANGE line to move all ranges of the second script forward by the number of cards in the first script.

Usually, it is not possible to modify values stored in labels, but if you use the {label?} syntax, you can change its value with the SET directive.

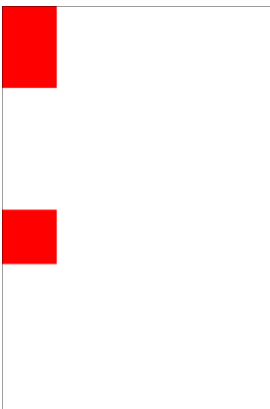
If you need a sequence of numbers, you can use AUTOLABEL to generate it, and if you need the actual sequence as a constant, you can copy and paste it from the F8 window after validation.

AUTOLABEL can be used to create intervals, indicating the comma as the separator of the sequence. Example:

```
[a]=autolabel(1,100,5,",")
```

Drawing a square using percentages isn't easy because equal percentages usually apply to different dimensions. To compensate, a percentage must be multiplied by the aspect ratio. Example:

```
rectangle=1,0,0,20%,20%,#FF0000
rectangle=1,0,50%,20%,{20/9*6}%,#FF0000
```



This script can be used to read a pdf and save the cards as individual images (the initial lines contain the parameters):

```
[pdf]=test.pdf
[page_w]=8.5
[page_h]=11
[card_w]=2.5
[card_h]=3.5

unit=inch
```

```

cardsize=[page_w],[page_h]
[pages]=PDFPAGES([pdf])
[num_w]=round([page_w]/[card_w],0,down)
[num_h]=round([page_h]/[card_h],0,down)
[mar_w]=([page_w]-[card_w]*[num_w])/2
[mar_h]=([page_h]-[card_h]*[num_h])/2

loadpdf=1-[pages],[pdf],0,0,[page_w],[page_h],{$}
[grid]=framebox([mar_w],[mar_h],[card_w]*[num_w],[card_h]*[num_h],[card_w],[card_h])

save=,"card{{{${-1}*[num_w]*[num_h]}+°}.png",<grid>

```

Like the previous script, this one reads a pdf and divides it into cards, this time dividing them into fronts and backs:

```

[pdf]=test.pdf
[page_w]=8.5
[page_h]=11
[card_w]=2.5
[card_h]=3.5

unit=inch
cardsize=[page_w],[page_h]
[pages]=PDFPAGES([pdf])
[num_w]=round([page_w]/[card_w],0,down)
[num_h]=round([page_h]/[card_h],0,down)
[mar_w]=([page_w]-[card_w]*[num_w])/2
[mar_h]=([page_h]-[card_h]*[num_h])/2

loadpdf=1-[pages],[pdf],0,0,[page_w],[page_h],{$}
[grid]=framebox([mar_w],[mar_h],[card_w]*[num_w],[card_h]*[num_h],[card_w],[card_h])

save=1-[pages]$ab>a,"front{{{${-1}/2*[num_w]*[num_h]}+°}.png",<grid>
save=1-[pages]$ab>b,"back{{{${-1}£2*[num_w]*[num_h]}+°}.png",<grid>

```

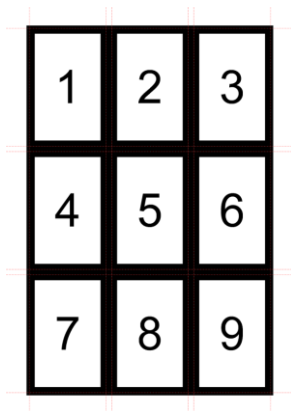
Guidelines are drawn above the edges by default but can be moved with 6th and 7th parameters in BORDER.

Example:

```

page=21,29.7,portrait,hv
border=none,#000000,0,dashed,#FF0000,0.2,0.2
font=arial,96,,#000000
text=1-9,{$},0,0,100%,100%
rectangle=,0,0,100%,100%,#000000,empty,20%

```

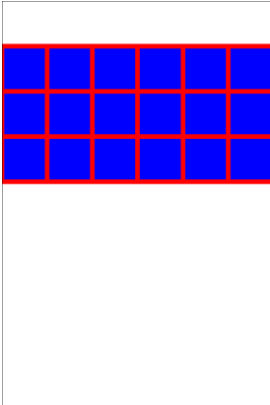


If you need to reduce the distance between images drawn with the HTMLIMAGE command, you can give the left and right margins (9th and 10th parameters) negative values.

When creating a list of frames (with FRAMEBOX or FRAMEHEX) you can indicate a flag to use a particular notation (and therefore give each of them a different name, such as FRAMEA1, FRAMEA2, etc.) or use no flag to give them all the same name (useful when you must use them all and not one in particular).

If you use N notation in FRAMEBOX, you can use a syntax like start_frame..end_frame in FRAMELIST to select a part of them. Example:

```
[a]=framebox(0,0,6,9,1,1,N)
[b]=framelist(a7..a24)
rectangle=1,<b>,#FF0000,#0000FF,0.1
```



The ° symbol can be used as a counter for frames (both when they have the same name and different names). Example:

```
[a]=framebox(0,0,6,9,1,1,N)
font=arial,12,,#FFFFFF,#0000FF
text=1,{°},<a*>
rectangle=1,<a*>,#FF0000,empty,0.1
```

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36
37	38	39	40	41	42
43	44	45	46	47	48
49	50	51	52	53	54

The \$ symbol in FRAMELIST rearranges the frames in a pattern suitable for a score track. Example:

```
[a]=framebox(0,0,6,9,1,1,N)
[b]=framelist($a*)
font=arial,12,,#FFFFFF,#0000FF
text=1,{°},<b*>
rectangle=1,<a*>,#FF0000,empty,0.1
```


1	2	3	4	5	6
12	11	10	9	8	7
13	14	15	16	17	18
24	23	22	21	20	19
25	26	27	28	29	30
36	35	34	33	32	31
37	38	39	40	41	42
48	47	46	45	44	43
49	50	51	52	53	54

In the above example, to reverse the order add a - symbol to the FRAMELIST parameter. Example:

```
[a]=framebox(0,0,6,9,1,1,N)
[b]=framelist(-$a*)
font=arial,12,,#FFFFFF,#0000FF
text=1,{°},<b*>
rectangle=1,<a*>,#FF0000,empty,0.1
```

54	53	52	51	50	49
43	44	45	46	47	48
42	41	40	39	38	37
31	32	33	34	35	36
30	29	28	27	26	25
19	20	21	22	23	24
18	17	16	15	14	13
7	8	9	10	11	12
6	5	4	3	2	1

In FRAMELIST, you can select a frame for each N using the syntax &N after the name(s). Example:

```
[a]=framebox(0,0,6,9,1,1,N)
[b]=framelist($a*&6)
font=arial,12,,#FFFFFF,#0000FF
text=1,{°},<b*>
rectangle=1,<a*>,#FF0000,empty,0.1
```

					1
2					
					3
4					
					5
6					
					7
8					
					9

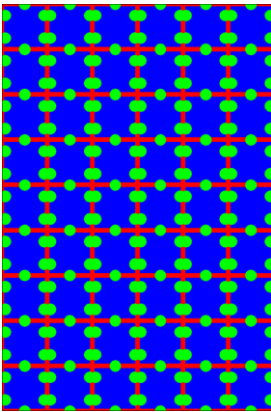
If you need to recall a frame name, you can use the μ symbol. Example:

```
[a]=framebox(0,0,6,9,1,1,N)
font=arial,10,,#FFFFFF,#0000FF
text=1, $\mu$ ,<a*>
rectangle=1,<a*>,#FF0000,empty,0.1
```

A1	A2	A3	A4	A5	A6
A7	A8	A9	A10	A11	A12
A13	A14	A15	A16	A17	A18
A19	A20	A21	A22	A23	A24
A25	A26	A27	A28	A29	A30
A31	A32	A33	A34	A35	A36
A37	A38	A39	A40	A41	A42
A43	A44	A45	A46	A47	A48
A49	A50	A51	A52	A53	A54

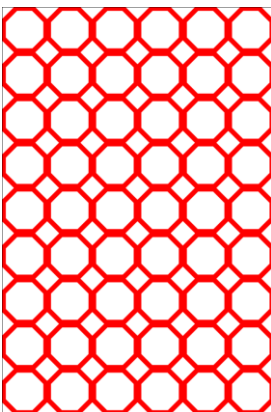
The creation of frames can be nested (i.e., create new frames starting from other frames). Example:

```
[a]=framebox(0,0,6,9,1,1,N)
[b]=frameclock(<a*>,0.5,0.5,6,0,50)
rectangle=1,<a*>,#FF0000,#0000FF,0.1
ellipse=1,<b*>,#00FF00
```



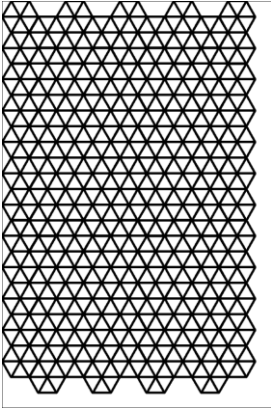
There is no specific command to create frames in a tessellation with squares and octagons, but it can be simulated by drawing octagons in frames created with FRAMEBOX. Example:

```
[a]=framebox(0,0,6,9,1,1,N)
polygon=1,<a*>,8,22.5,#FF0000,empty,0.1
```



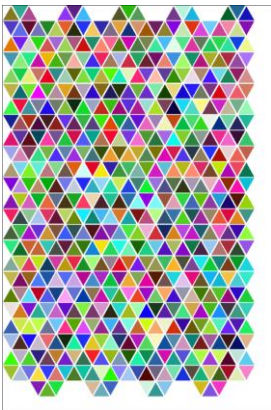
You can't create triangular frames, but you can draw them in a hexagon using a STAR. Example:

```
[t]=framehex(0,0,6,9,0.4,C)
star=1,<t*>,6,90,1,#000000,empty,0.05
polygon=1,<t*>,6,30,#000000,empty,0.05
```



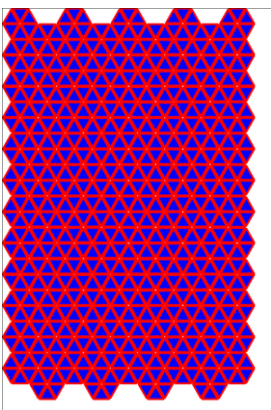
About the last script, instead of a star you can use a FRAMECLOCK to create six frames for each hexagon and draw a triangle in each one. Example:

```
[a]=framehex(0,0,6,9,0.4,C,58,58)
[b]=frameclock(<a*>,0.4,0.4,6)
polygon=1,<b*>,3,60+°*60,#HHHHHH
```



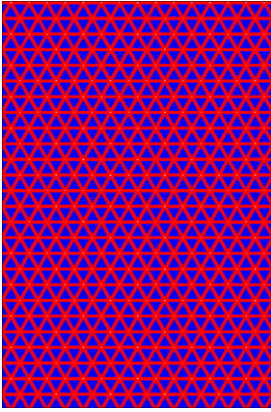
if you use the whole card for a FRAMEHEX only whole hexagons are created anyway, so the edges remain empty. Example:

```
[a]=framehex(0,0,6,9,0.4,C,58,58)
[b]=frameclock(<a*>,0.4,0.4,6)
polygon=1,<b*>,3,60+°*60,#FF0000,#0000FF,0.04
```



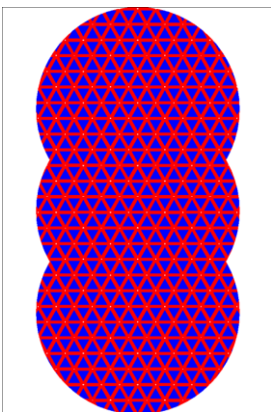
To correct this, you can use a starting area bigger than the card:

```
[a]=framehex(-1,-1,8,11,0.4,C,58,58)
[b]=frameclock(<a*>,0.4,0.4,6)
polygon=1,<b*>,3,60+°*60,#FF0000,#0000FF,0.04
```



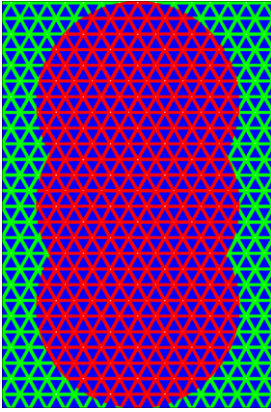
If you want to draw something in a "hollow" shape, you can use a LAYER with a CHROMAKEY. Example, I draw the above image, then set CHROMAKEY as yellow, and draw three yellow circles in a layer:

```
[a]=framehex(-1,-1,8,11,0.4,C,58,58)
[b]=frameclock(<a*>,0.4,0.4,6)
polygon=1,<b*>,3,60+°*60,#FF0000,#0000FF,0.04
chromakey=#FFFF00
layer=100
  circle=1,0,0,100%,50%,#FFFF00
  circle=1,0,25%,100%,50%,#FFFF00
  circle=1,0,50%,100%,50%,#FFFF00
endlayer
```



In the previous example, you can draw something also in the layer but note that you can't use the CHROMAKEY color (anything yellow forms the "hole"). Example:

```
[a]=framehex(-1,-1,8,11,0.4,C,58,58)
[b]=frameclock(<a*>,0.4,0.4,6)
polygon=1,<b*>,3,60+°*60,#FF0000,#0000FF,0.04
chromakey=#FFFF00
layer=100
  polygon=1,<b*>,3,60+°*60,#00FF00,#0000FF,0.04
  circle=1,0,0,100%,50%,#FFFF00
  circle=1,0,25%,100%,50%,#FFFF00
  circle=1,0,50%,100%,50%,#FFFF00
endlayer
```



If you give a layer a name, everything you draw on it isn't immediately drawn on the card but is drawn only when you use the LAYERDRAW command with that name. Example:

```
layer=100,0,0,0,one
  font=arial,64,,#000000
  text=1,"Layer",0,0,100%,100%
endlayer
```

```
layerdraw=1,one,10,-0.1,-0.1
layerdraw=1,one,10,0,-0.1
layerdraw=1,one,10,0.1,-0.1
layerdraw=1,one,10,-0.1,0
layerdraw=1,one,10,0,0
layerdraw=1,one,10,0.1,0
layerdraw=1,one,10,-0.1,0.1
layerdraw=1,one,10,0,0.1
layerdraw=1,one,10,0.1,0.1`
```



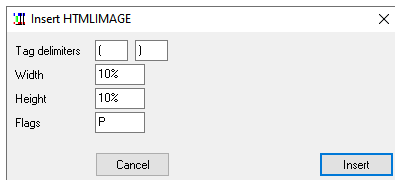
The quickest way to create a copy of a script to be able to modify it while keeping the original is to press CTRL+D (or right-click on the tab at the top and choose "Add new version" entry).

Using HTMLKEY linked to a font with the Z flag it is possible to ensure that two or more words in a text (or a word and an image) are never split over several lines.

Right click on the "copy script" button to copy the script with the format suitable for pasting on BGG, Reddit, BGDF, or Discord.

If you want to quickly edit the content of a file linked with LINK, you can press the Edit button (on the right in the main window), a window will be shown with the contents of the fields of the active card, with the possibility to edit them.

If you have a text with keywords in it for HTMLIMAGE, you can automatically compile the lines selecting the name of the sequence in F8 and clicking the "Insert tags" button; in the dialog window you can specify the delimiters for the keywords, size and flags for the HTMLIMAGE lines.



Unicode characters in a spreadsheet are automatically converted into html codes to be shown with HTMLTEXT, if instead you want to use a TEXT you can disable the conversion with a line LINKUNI=OFF before the LINK (but considering that a TEXT cannot show all the characters).

When a spreadsheet is read, the characters [] { } are converted into codes (to correctly display the brackets in the text). You can change this default behavior with LINKENCODE (specifying the characters to be translated); for example, if you don't want to translate any characters (and therefore use sequences and expressions), you can use:

LINKENCODE=

Formatting in a spreadsheet is automatically converted to html codes, but you can change the conversion with a LINKSTYLES line. For example, if you want to change the bold from to a font (e.g., font1) you can specify:

LINKSTYLES = <font1>, </font1>

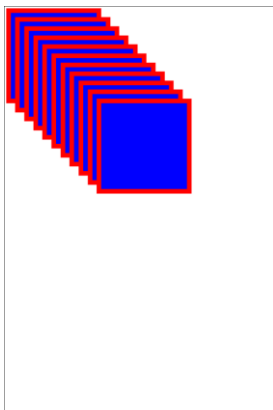
The other parameters are the opening/closing tags for all other formatting.

If you set the HV flags in PAGE to center the cards on the page you don't need to indicate the margins with MARGINS, but if you do those parameters will be the minimum values that the margins will assume when they are calculated to center the cards in the page. In other words, if you have a printer that does not print beyond a certain position on the page, it is still advisable to set the margins with MARGINS, to prevent too narrow margins from being calculated with the HV flags in PAGE.

You can create frames using FRAMETRANS in a loop. Example:

```
<frame>=0.1,0.1,2,2
[frame(num)]%,(num),1,10=frametrans(frame,(num)/5,(num)/5)

rectangle=1,<frame*>,#FF0000,#0000FF,0.1
```



In the visual editor it is impossible to use LINE to draw horizontal or vertical lines, given that the rectangle outlining the drawing has zero height or width respectively, so it is better to use LINERECT, choosing one of the flags that draws only one side of the rectangle.

To create a PDF with cards of different sizes (after having created them with different scripts) you can use MERGEPDF to join the individual PDFs, or MOSAIC to layout individual cards of different sizes.

To see what changes are made to the script during validation you can uncheck the option Config → Interface → "Do not show script after validation".

With very complex HTML the Explorer engine can give errors and result in a blank image, in these cases it can be useful to add the N flag to HTMLTEXT, so that a new object is instantiated each time, instead of always using the same one.

nanDECK can be run through a command line in a batch, for example, this batch (a text file with .bat extension) runs the three scripts (the executable and scripts are assumed to be in the same folder as the batch) and saves the result in PDF files:

```
nandeck test1.txt /createpdf
nandeck test2.txt /createpdf
nandeck test3.txt /createpdf
```

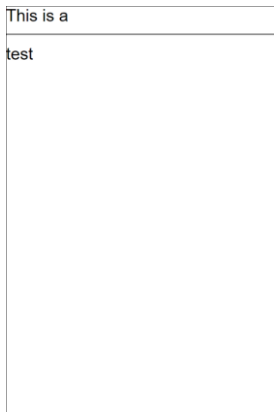
Written in this way, the three scripts are executed in sequence. If you want to execute them at the same time, simply precede each line with the start command:

```
start nandeck test1.txt /createpdf
start nandeck test2.txt /createpdf
start nandeck test3.txt /createpdf
```

Remember that in Win 10/11 you can run a second instance of nanDECK (or any other program) with a shift-click on its taskbar icon (or a scroll-wheel click on it).

In HTMLTEXT normal HTML tags can be used, for example, <hr> to add lines within the text:

```
htmlfont=default,arial,10,,#000000
htmltext=1,"This is a<hr>test",0,0,100%,100%,#FFFFFF,0,BE,100,default
```

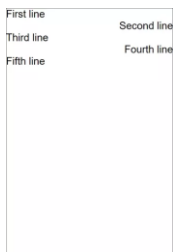


The U flag in HTMLTEXT is used to indicate that the font in the 12th parameter is applied to every single line of the text (each line is separated by default by
). If that parameter is a sequence, each element is applied to each paragraph. Example:

```
[text]="First line<br>Second line<br>Third line<br>Fourth line<br>Fifth line"
```

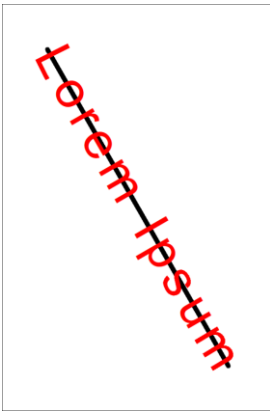
```
htmlfont=default,arial,10,,#000000,left
htmlfont=left,arial,10,,#000000,left
htmlfont=right,arial,10,,#000000,right
```

```
htmltext=1-({text}),[text],0,0,100%,100%,#FFFFFF,0,BEU,100,default,left|right
```



In a BEZIER if you specify the handles in the same position as the start/end point you will draw a line, useful when you want to write text along this "line". Example:

```
bezier=1,1,1,1,1,5,8,5,8,#000000,0.1
font=arial,32,TZ,#FF0000
text=1,Lorem Ipsum,0,0,100%,100%
```




In the previous example, if you don't want to draw the line you can use an EDGE=,NULL line. Example:


```
edge=,NULL
bezier=1,1,1,1,1,5,8,5,8,#000000,0.1
edge=,SOLID
font=arial,32,TZ,#FF0000
text=1,Lorem Ipsum,0,0,100%,100%
```



Usually, an image in HTMLIMAGE higher than the text adds a space between the lines, if you want to remove it you can use a negative top margin in the HTMLIMAGE line. Example:

```
htmlimage=,(clock1),clock-tower.png,1,1,P
htmlimage=,(clock2),clock-tower.png,1,1,P,0,-1
htmlfont=default,arial,12,,#000000
htmltext=1,"Lorem ipsum dolor sit amet, Lorem ipsum (clock1) dolor sit amet,
Lorem ipsum dolor sit amet",0,0,100%,50%,#FFFFFF,0,BE,100,default
htmltext=1,"Lorem ipsum dolor sit amet, Lorem ipsum (clock2) dolor sit amet,
Lorem ipsum dolor sit amet",0,50%,100%,50%,#FFFFFF,0,BE,100,default
```

Lorem ipsum dolor sit amet,
 Lorem ipsum  dolor sit
 amet, Lorem ipsum dolor sit
 amet

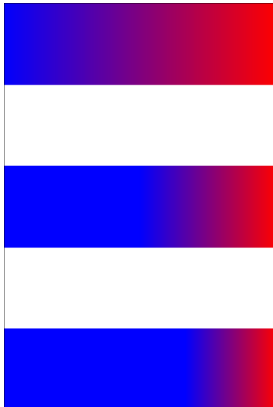
Lorem ipsum dolor sit amet,
 Lorem ipsum  dolor sit
 amet, Lorem ipsum dolor sit
 amet

If you have more than twenty-six icons to represent with the ICONS command, remember that in addition to letters you can also use numbers, so the effective limit for strings of length 1 is thirty-six icons.

The program automatically establishes how many cards are in the deck using the number of rows in the linked spreadsheet, or from the ranges specified in the cards: but for the latter, those resulting from functions are not considered (since the functions are evaluated at build, and the number of cards must be established in the validation).

One of the ways to move the midpoint of a gradient is to add multiple copies of a color into the definition. Example:

```
rectangle=1,0,0,100%,20%,#FF0000#0000FF@0
rectangle=1,0,40%,100%,20%,#FF0000#0000FF#0000FF@0
rectangle=1,0,80%,100%,20%,#FF0000#0000FF#0000FF#0000FF@0
```



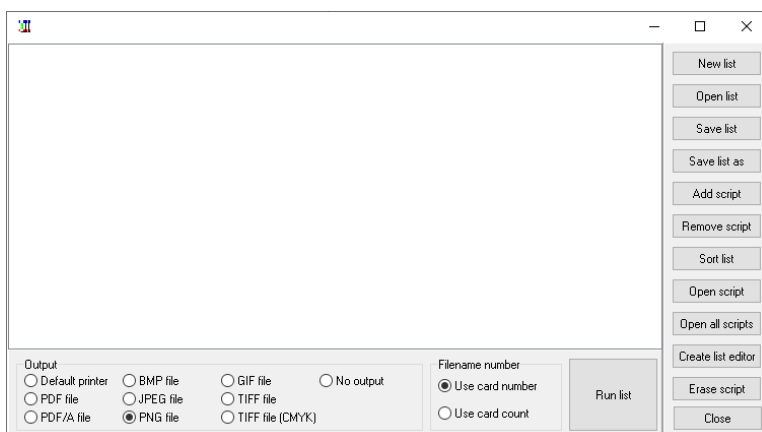
The original purpose of the CONCAT function was to quickly create gradients where it was necessary to indicate the same color multiple times, for example, instead of writing:

```
#FF0000#0000FF#0000FF#0000FF@0
```

you can write:

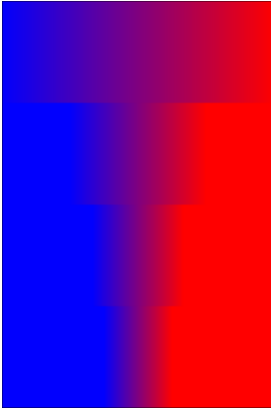
```
concat (#FF0000,1,#0000FF,3,@0,1)
```

If you want to run a series of scripts, instead of using batch files you can use the “Script list” option (from the button of the same name), add all the scripts you need to the list, choose the output, and press the “Run list” button.



Normally a gradient is drawn over the entire extension of the rectangle, but space can be reduced by using a value greater than one as the second parameter of the GRADIENT command. Example:

```
rectangle=1,0,0,100%,25%,#FF0000#0000FF@0
gradients=1,2
rectangle=1,0,25%,100%,25%,#FF0000#0000FF@0
gradients=1,3
rectangle=1,0,50%,100%,25%,#FF0000#0000FF@0
gradients=1,4
rectangle=1,0,75%,100%,25%,#FF0000#0000FF@0
```



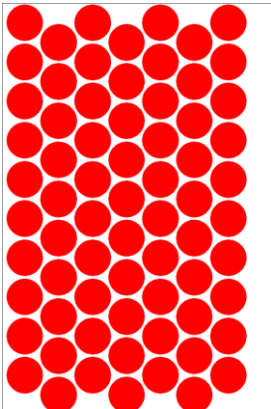
If JOIN parameters are sequences of different lengths, the result will be a sequence with the same number of elements as the longest parameter, while the other sequences will be wrapped.

If the parts within the VISUAL...ENDVISUAL structure become too complex, you can divide them into several separate structures, or move some lines outside, to be able to work only on the others.

FRAMEHEX creates frames arranged in a hexagonal grid, but you can draw anything on them, not just hexagons.

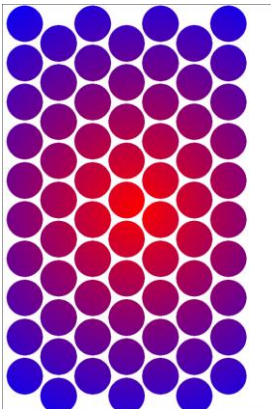
Example:

```
[a]=framehex(0,0,6,9,0.5,,80,80)
ellipse=1,<a>,#FF0000
```



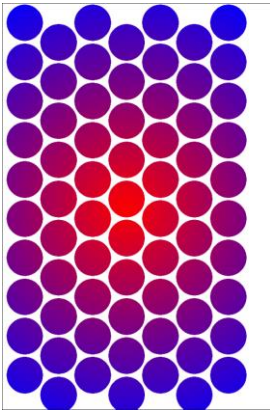
To apply a unique radial gradient in separate shapes, you can use a COLORCHANGE line. Example:

```
[a]=framehex(0,0,6,9,0.5,,80,80)
ellipse=1,<a>,#FF0000
colorchange=1,0,0,100%,100%,#FF0000,#FF0000#0000FF@360
```



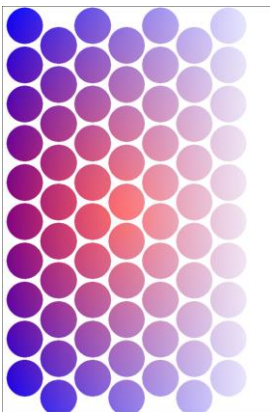
In the previous example the gradient is not centered because it is applied to the entire card (while the frames leave an area on the right empty). To center the gradient, you need to create an area that includes only the frames, with FRAMEMELD, like this:

```
[a]=framehex(0,0,6,9,0.5,,80,80)
[b]=framemeld(a)
ellipse=1,<a>,#FF0000
colorchange=1,<b>,#FF0000,#FF0000#0000FF@360
```



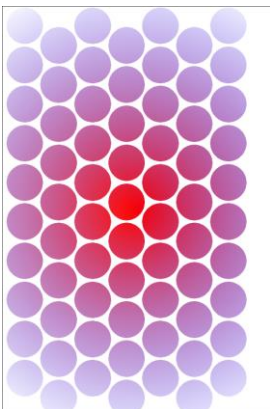
To fade a layer towards transparency you can use the @angle in the first parameter. Example:

```
[a]=framehex(0,0,6,9,0.5,,80,80)
[b]=framemeld(a)
layer=100@0
  ellipse=1,<a>,#FF0000
  colorchange=1,<b>,#FF0000,#FF0000#0000FF@360
endlayer
```



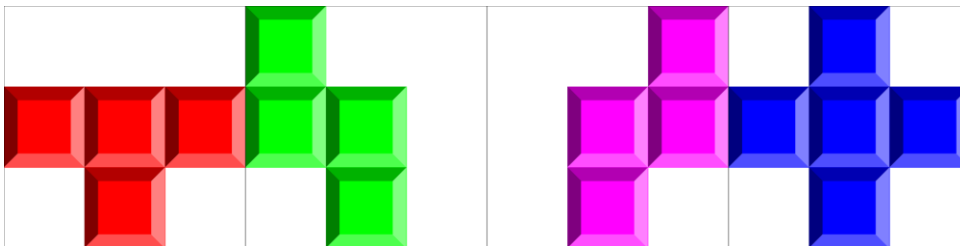
The fading can be radial (with 360 as angle). Example:

```
[a]=framehex(0,0,6,9,0.5,,80,80)
[b]=framemeld(a)
layer=100@360
  ellipse=1,<a>,#FF0000
  colorchange=1,<b>,#FF0000,#FF0000#0000FF@360
endlayer
```



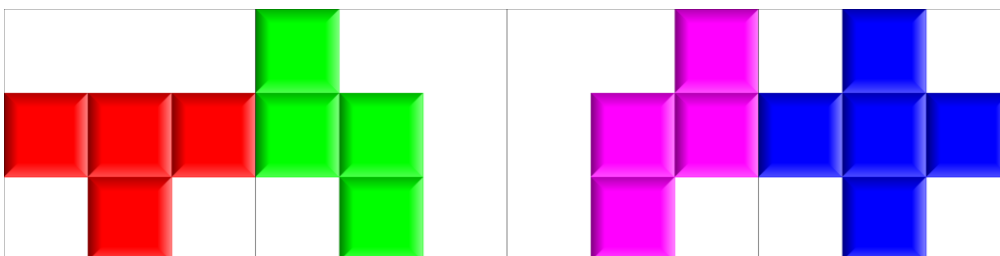
A script to create polyominoes:

```
cardsize=9,9
[a]=framebox(0,0,9,9,3,3,N)
[b1]=framelist(a4,a5,a6,a8)
rectangle=1,<b1>,#FF0000
button=1,<b1>,0.5,I
[b2]=framelist(a1,a4,a5,a8)
rectangle=2,<b2>,#00FF00
button=2,<b2>,0.5,I
[b3]=framelist(a3,a5,a6,a8)
rectangle=3,<b3>,#FF00FF
button=3,<b3>,0.5,I
[b4]=framelist(a2,a4,a5,a6,a8)
rectangle=4,<b4>,#0000FF
button=4,<b4>,0.5,I
```



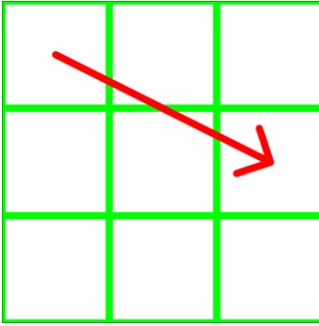
You can add the G flag to BUTTON to draw edges with gradients. Example:

```
cardsize=9,9
[a]=framebox(0,0,9,9,3,3,N)
[b1]=framelist(a4,a5,a6,a8)
rectangle=1,<b1>,#FF0000
button=1,<b1>,0.5,IG
[b2]=framelist(a1,a4,a5,a8)
rectangle=2,<b2>,#00FF00
button=2,<b2>,0.5,IG
[b3]=framelist(a3,a5,a6,a8)
rectangle=3,<b3>,#FF00FF
button=3,<b3>,0.5,IG
[b4]=framelist(a2,a4,a5,a6,a8)
rectangle=4,<b4>,#0000FF
button=4,<b4>,0.5,IG`
```



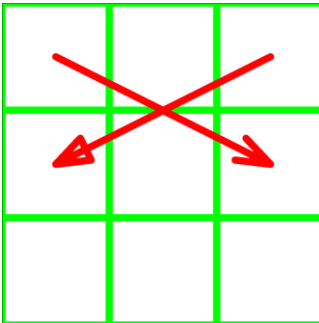
To draw lines or arrows in frames, you can use the PCC parameter (it's the center point of the frame). Example:

```
cardsize=9,9
[a]=framebox(0,0,9,9,3,3,N)
rectangle=1,<a*>,#00FF00,empty,0.2
line=1,<a1,pcc>,<a6,pcc>,#FF0000,0.2,,1
```



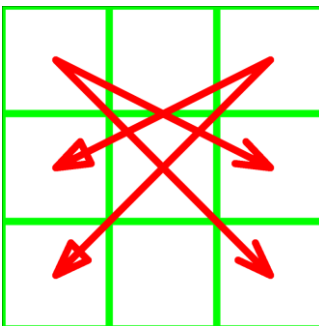
The eleventh parameter of LINE indicates the opening angle of the arrowhead, and if a negative value is used, a third line is drawn. Example:

```
cardsize=9,9
[a]=framebox(0,0,9,9,3,3,N)
rectangle=1,<a*>,#00FF00,empty,0.2
line=1,<a1,pcc>,<a6,pcc>,#FF0000,0.2,,1,,20
line=1,<a3,pcc>,<a4,pcc>,#FF0000,0.2,,1,, -20
```



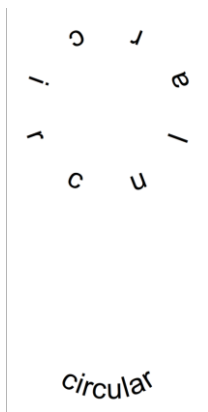
To indicate multiple frames at the same time, you can create new ones with FRAMELIST. Example:

```
cardsize=9,9
[a]=framebox(0,0,9,9,3,3,N)
rectangle=1,<a*>,#00FF00,empty,0.2
[b]=framelist(a6,a9)
[c]=framelist(a4,a7)
line=1,<a1,pcc>,<b,pcc>,#FF0000,0.2,,1,,20
line=1,<a3,pcc>,<c,pcc>,#FF0000,0.2,,1,, -20
```



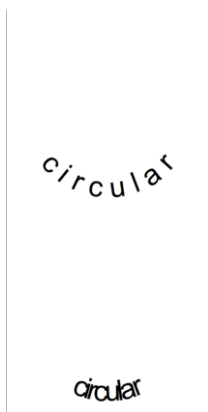
The R flag in FONT places the text on a circle, normally it is extended over its entire diameter, but you can set the distance of the letters in the tenth parameter. Example:

```
cardsize=6,12
font=arial,24,TR,#000000,#FFFFFF
text=1,"circular",0,0,6,6
FONT=arial,24,TR,#000000,#FFFFFF,0,0,0,0,0
text=1,"circular",0,6,6,6
```



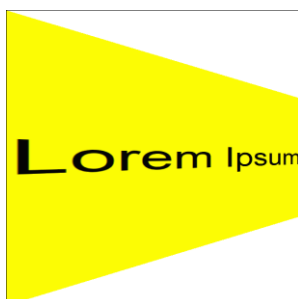
In the above example, the distance can be positive or negative. Example:

```
cardsize=6,12
font=arial,24,TR,#000000,#FFFFFF,0,0,0,0,0.2
text=1,"circular",0,0,6,6
FONT=arial,24,TR,#000000,#FFFFFF,0,0,0,0,-0.1
text=1,"circular",0,6,6,6
```



TRANSFORM can be used not only with IMAGE, but also with LAYER, and can be used to give special effects to text. Example:

```
cardsize=5,5
chromakey=#FFFFFF
transform=1,0,0,100,30,100,70,0,100
layer
  font=arial,24,,#000000,#FFFF00
  text=1,"Lorem Ipsum",0,0,5,5
endlayer
```



In the example above you can use the 14th and 15th parameters in TEXT to stretch the text horizontally and vertically. Example:

```
cardsize=5,5
chromakey=#FFFFFF
transform=1,0,0,100,30,100,70,0,100
layer
```

```
font=arial,24,,#000000,#FFFF00
text=1,"Lorem Ipsum",0,0,5,5,center,center,0,100,,,,100,500
endlayer
```



In the above example, remember to reset the TRANSFORM (without parameters, except range) after using it.

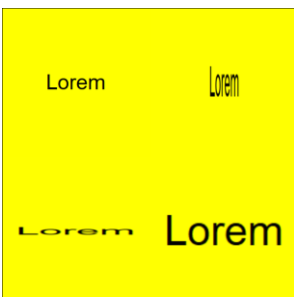
Example:

```
cardsize=5,5
chromakey=#FFFFFF
transform=1,0,0,100,20,100,60,0,100
layer
font=arial,24,,#000000,#FFFF00
text=1,"Lorem Ipsum",0,0,5,2.5,center,center,0,100,,,,100,400
endlayer
transform=1
layer
font=arial,24,,#000000,#FFFF00
text=1,"Lorem Ipsum",0,2.5,5,2.5,center,center,0,100,,,,100,400
endlayer
```



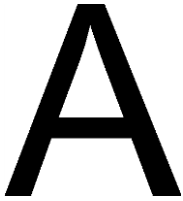
The 14th and 15th parameters in TEXT changes the horizontal and vertical size of the text. Example:

```
cardsize=5,5
font=arial,10,,#000000,#FFFF00
text=1,"Lorem",0,0,2.5,2.5
text=1,"Lorem",2.5,0,2.5,2.5,center,center,0,100,,,,50,200
text=1,"Lorem",0,2.5,2.5,2.5,center,center,0,100,,,,200,50
text=1,"Lorem",2.5,2.5,2.5,2.5,center,center,0,100,,,,200,200
```



If you have a shape drawn on a card, and you want to save it to reduce the border around it as much as possible, you can use the K flag in the SAVE command. Example:

```
font=arial,128,,#000000
text=1,"A",0,0,100%,100%
save=1,"Result.png",0,0,100%,100%,,,,,,K
```



The DOWNLOAD directive can use a sequence of URLs, if there is also a sequence of image names as 2nd parameter. Example:

```
[a]=URL1|URL2|URL3
[b]=test1.png|test2.png|test3.png
download=[a],[b]
```

Since there is a limit of 4 GB of RAM, which corresponds to approximately 600 standard-sized cards, to save memory there are various options:

- disable the "Cache images" option in Config → Interface,
- if the cards don't need 16 million colors you can use 65536 with a DEPTH=,16 line.
- use disk instead of RAM with "Deck file location: on disk" option in Config → Main.

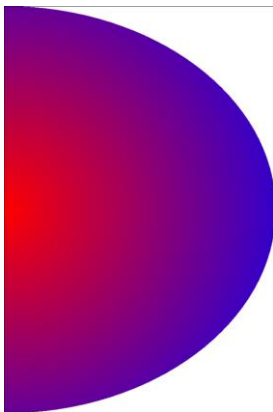
If you need to save cards in multiple ways you can use more than one DISPLAY line or more than one SAVE line (or both) in the same script.

If you want to create just one card, to do a quick test, you can add a RENDER line, (example RENDER=10,10), or right-click on the preview (in the top right corner of the main screen).

To restore nanDECK to the default settings, simply delete the nandek.ini file from the folder where the executable is located.

If you want only part of a directive to appear on a card, remember that in the area parameters you can also provide negative coordinates, or dimensions larger than those of the card. Example:

```
ellipse=1,-100%,0,200%,100%,#FF0000#0000FF@360
```



If you have a LINK line in your script pointing to a Google Sheet, if you press the "Linked data" button it opens that sheet in the default browser.

If you need to save an image and then reload it onto multiple cards (perhaps with different parameters such as the angle) you can do it with a SAVE and various IMAGEs, or with a LAYER and various LAYERDRAWs; the latter is faster because the operation is not done on disk.

In the nandek.ini file you will also find the coordinates of all the windows, if one of these moves off the screen you can fix it by deleting the relevant section, e.g. for the visual editor the section is called [form_vis], when the program is reopened the position of default is set.

The difference between RENDER and PRINT is that with RENDER only the indicated cards are created, but the deck maintains its dimensions (the uncreated cards are blank), whereas with PRINT all the cards are created, but only those indicated are printed (the others are hidden). If you want to create and print only the indicated cards, you must use both.

The plasma cloud effect that is obtained by adding & and a number to a series of colors gives different results with each execution, so if you want to always maintain the same back for all cards with this method, it is better to save a single image created like this and use it for all the backs.

PATTERN is a less flexible version of ICON/ICONS since it is limited to the use of a single image (repeated N times), but in certain cases it can still be useful (for example to display energy bars or similar).

You can use the ' symbol for comments even when ; is set in the Config, and vice versa, that is:

```
'rectangle  
;rectangle
```

Are not executed whether the comment symbol is ' or ;

But this does not apply to label definitions:

```
'[test]=abc
```

If the comment is ' it is not executed

If the comment is ; gives error (the symbol is not recognized as a modifier of the definition).

If you want to speed up the creation of the deck for a screen test you can use DPI=150, the cards will still have sufficient resolution for a screen with 72 or 96 DPI, and then return to the standard resolution for the result to be printed out.

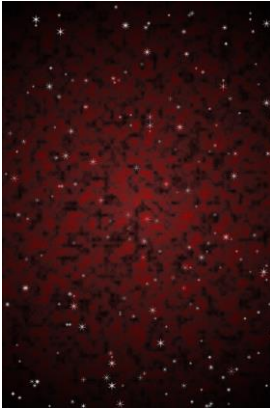
Script to draw a starry sky:

```
oversample=2  
imagefilter=lanczos  
rectangle=1,0,0,100%,100%,#000000  
for=a,1,200  
  star=1,d100%,d100%,d3%,d3%,d5+5,0,d20,#FFFFFF  
next
```



Plasma effect with a radial transparency over a starry sky:

```
oversample=2  
imagefilter=lanczos  
rectangle=1,0,0,100%,100%,#000000  
for=a,1,200  
  star=1,d100%,d100%,d3%,d3%,d5+5,0,d20,#FFFFFF  
next  
layer=80@360  
  rectangle=1,0,0,100%,100%,#FF0000#803030#000000&6  
endlayer`
```



On the print screen ("Print deck" button) at the top left you will find the button to print; if you select the Print to File option you can create print files in particular formats such as HTML, Excel, or Word.

To draw in a grid the texts that are in a spreadsheet like this:

	A	B	C	D	E	F	G
1	Col1	Col2	Col3	Col4	Col5	Col6	Col7
2		1	8 e	l	s	z	stu
3		2	9 f	m	t	abc	vw
4		3	10 g	n	u	def	yz
5		4 a	h	o	v	ghi	z
6		5 b	i	p	w	jkl	zz
7		6 c	j	q	x	mno	zzz
8		7 d	k	r	y	pqr	zzzz
9							
10							
11							

You can use this script:

```
cardsize=24,27
```

```
link=data.xlsx
```

```
[h]=framehex(0,0,24,24.6,2,E)
```

```
polygon=1,<h*>,6,30,#000000,empty,0.1
```

```
font=arial,32,T,#000000
```

```
text=1,{col1?°},<ha*>
```

```
text=1,{col2?°},<hb*>
```

```
text=1,{col3?°},<hc*>
```

```
text=1,{col4?°},<hd*>
```

```
text=1,{col5?°},<he*>
```

```
text=1,{col6?°},<hf*>
```

```
text=1,{col7?°},<hg*>
```

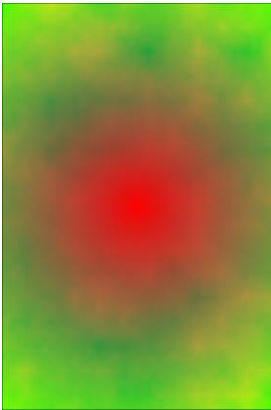
1	e	l	s	z	stu				
2	8	f	m	t	abc	vw			
3	9	g	n	u	def	yz			
4	10	h	o	v	ghi	z			
5	a	i	w	jkl	zz				
6	b	j	p	x	mno	zzz			
7	c	q	y	pqr	zzzz				
	d	r							

If you click the file name in the status bar (bottom row of the window), an Explorer window opens on that folder.

Right clicking the "Add" button adds a comment and all lines with the same directive on the cursor, the same happens on the "Rem" button.

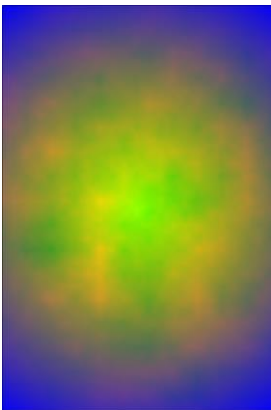
The standard 100@360 transparent gradient is from 100 at the center to 0 at the edge:

```
rectangle=1,0,0,100%,100%,#FFFF00#00FF00&8
layer=100@360
  rectangle=1,0,0,100%,100%,#FF0000#0000FF@360
endlayer
```



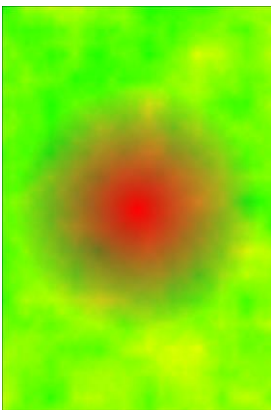
If you add a ! it becomes 0 in the center and 100 on the edge:

```
rectangle=1,0,0,100%,100%,#FFFF00#00FF00&8
layer=100@360!
  rectangle=1,0,0,100%,100%,#FF0000#0000FF@360
endlayer
```



In a transparent gradient you can control the size with the syntax 100@360!N, where N by default is 100, a lower value decreases the diameter, a higher value increases it . Example:

```
rectangle=1,0,0,100%,100%,#FFFF00#00FF00&8
layer=100@360&50
rectangle=1,0,0,100%,100%,#FF0000#0000FF@360
endlayer
```



A right click on the preview (top right) runs the script on the current card; if the "Partial" option is active the script is executed only up to the line where the text cursor is present. With a CTRL+right click the cursor moves to the

next line before execution, in this mode it is possible to see the effects of the lines of code on the card step by step.

A script with easy-to-cut hexagons, with front/back texts:

```
BORDER=RECTANGLE
PAGE=21,29.7, PORTRAIT, HV
DPI=300
[w]=21
[h]=29.7
CARDSIZE=[w],[h]
[size]=3
[row]=3

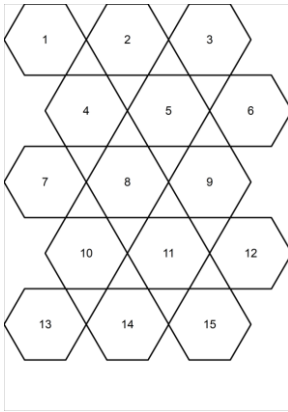
SEQUENCE=seq_front
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
ENDSEQUENCE

SEQUENCE=seq_back
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
thirteen
fourteen
fifteen
ENDSEQUENCE

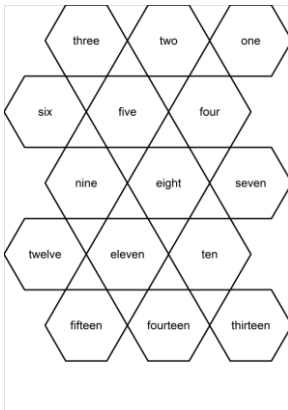
[hex_front]=FRAMEHEX(0,0,[w],[h],[size],X)
POLYGON=1,<hex_front>,6,90,#000000,EMPTY,0.1
FONT=Arial,24,T,#000000
TEXT=1,{seq_front?°},<hex_front>

[hex_back]=FRAMEHEX(0,0,[w],[h],[size],XS)
POLYGON=2,<hex_back>,6,90,#000000,EMPTY,0.1
FONT=Arial,24,T,#000000
TEXT=2,{seq_back?((°-1)£[row])*[row]+[row]-(°-1)#[row]},<hex_back>
```

Result, front:

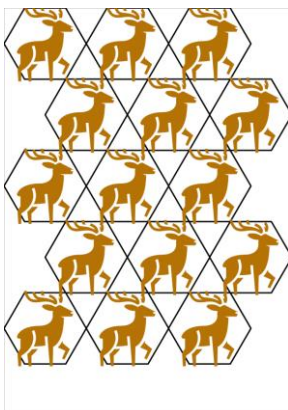


And back:



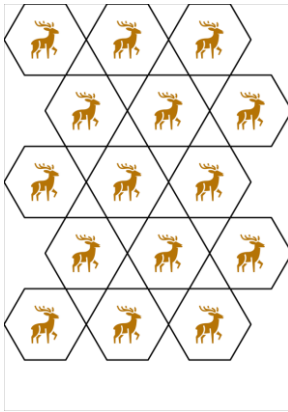
In this example hexagons and images are the same size:

```
[w]=21
[h]=29.7
CARDSIZE=[w],[h]
[size]=3
[hex_front]=FRAMEHEX(0,0,[w],[h],[size],X)
POLYGON=1,<hex_front>,6,90,#000000,EMPTY,0.1
IMAGE=1,deer.png,<hex_front>,0,PN
```



To keep the same hexagons and change the dimensions of the images you can use a second series of frames:

```
[w]=21
[h]=29.7
CARDSIZE=[w],[h]
[size]=3
[hex_front]=FRAMEHEX(0,0,[w],[h],[size],X)
POLYGON=1,<hex_front>,6,90,#000000,EMPTY,0.1
[hex_front_img]=FRAMEHEX(0,0,[w],[h],[size],X,50,50)
IMAGE=1,deer.png,<hex_front_img>,0,PN
```



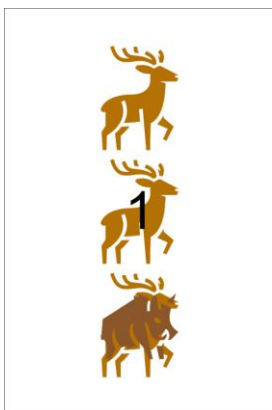
The G flag in HTMLIMAGE is used to overlay text on an image but can also be used to overlay another image. Example:

```
[text]="(deer)<br>(deer1)<br>(deerboar) "

htmlfont=font1,arial,32,,#000000,center
htmlmargins=font1,0,0,0,0,center

htmlimage=,(deer),deer.png,40%,40%,PG
htmlimage=,boar,boar.png,30%,30%,P

htmltext=1,[text],0,0,100%,100%,#FFFFFF,0,BE,100,font1
```



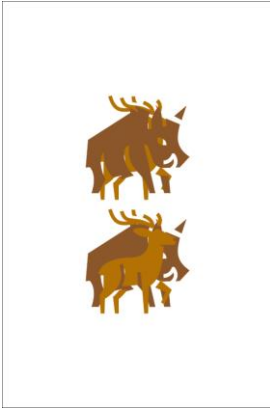
In the previous examples, with the G flag in HTMLIMAGE the text or image is drawn on top of the original image, if you add the U flag they are drawn below. Example:

```
[text]="(deer1boar)<br>(deer2boar) "

htmlfont=font1,arial,32,,#000000,center
htmlmargins=font1,0,0,0,0,center

htmlimage=,(deer1),deer.png,40%,40%,PG
htmlimage=,(deer2),deer.png,40%,40%,PGU
htmlimage=,boar,boar.png,40%,40%,P

htmltext=1,[text],0,0,100%,100%,#FFFFFF,0,BE,100,font1
```



In the Insert menu there is an entry to insert the test text “Lorem Ipsum...”.

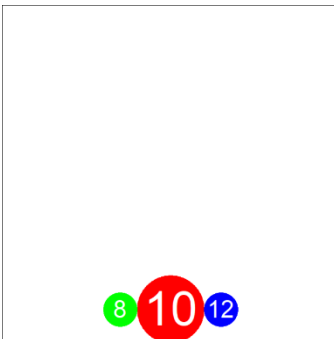
When you need to have content on a card, rotated on multiple sides, calculating the correct positions and angle can be difficult. The best option is to draw everything on a side where it is simple to obtain the values, example:

```
cardsize=6,6
```

```
circle=1,40%,80%,20%,20%,#FF0000
font=arial,24,T,#FFFFFF
text=1,10,40%,80%,20%,20%
```

```
circle=1,30%,85%,10%,10%,#00FF00
font=arial,12,T,#FFFFFF
text=1,8,30%,85%,10%,10%
```

```
circle=1,60%,85%,10%,10%,#0000FF
font=arial,12,T,#FFFFFF
text=1,12,60%,85%,10%,10%
```



And then enclose everything in a rotated layer:

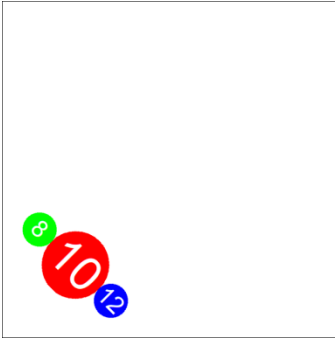
```
cardsize=6,6
```

```
layer=100,0,0,45
  circle=1,40%,80%,20%,20%,#FF0000
  font=arial,24,T,#FFFFFF
  text=1,10,40%,80%,20%,20%
```

```
  circle=1,30%,85%,10%,10%,#00FF00
  font=arial,12,T,#FFFFFF
  text=1,8,30%,85%,10%,10%
```

```
  circle=1,60%,85%,10%,10%,#0000FF
  font=arial,12,T,#FFFFFF
  text=1,12,60%,85%,10%,10%
```

```
endlayer`
```



If the message “Label unknown” is displayed in the log (bottom window), just click on the “Label error” button (bottom right) to highlight the missing label in the script.

In any case, when an error is displayed in the log, just double-click on the log line to position yourself on the line of the script that gave that error.

The column with the shortcut buttons and the one with the list of tags can be positioned to the right or left of the editor, or hidden, by choosing the corresponding option in Config → Main.

By default, the tag column is populated with the “Card” tag and the number corresponding to each card. But you can customize it with one or more TAG lines in the script. For example, this creates ten OBJECT tags with numbers for the first ten cards:

```
tag=1-10,object,$
```

If you press the "Info" button you can see how many cards you have created since you used the program and how many builds you have done (in the current session and since you used the program); these last two numbers can be retrieved in the program with the INFO function (flags L and G).

Easter egg: if you press the image in the Info window, you will see the effect of the MANDALA command in an animation (which will change when you click the mouse, while pressing a keyboard key will close the window).

Code examples

Wargame counters

```
cardsize=2,2
dpi=600
linkmulti=number
link=data.txt

[back_ger]=#C0C0C0
[front_ger]=#000000
[out_ger]=#808080

[back_fre]=#8ADDF4
[front_fre]=#000000
[out_fre]=#808080



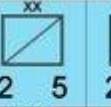
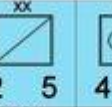
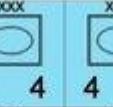

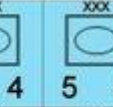




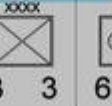
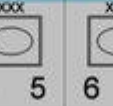



macro=outline,(range1),(text1),(frame),(font1),(size1),(col1),(col2),(col3)
    font=(font1),(size1),(col3),(col2),0.01,0.01
    text=(range1),(text1),(frame)
    font=(font1),(size1),T,(col1),(col2)
    text=(range1),(text1),(frame)
end

[all]=1-{(number)}
<cnt_all>=0,0,2,2
<val_lft>=0.25,1.25,0.5,0.75
<val_cnt>=0.75,1.25,0.5,0.75
<val_rgt>=1.25,1.25,0.5,0.75
<val_id>=0.25,0,1.5,0.25
<img_cnt>=0.45,0.3,1.1,0.9
<img_cnt2>=0.6,0.5,0.8,0.5
rectangle=[all],0,0,2,2,[back_[nation]]

outline=[all],[combat],<val_lft>,Arial,16,[front_[nation]],[back_[nation]],[out_[nation]]
outline=[all],[movement],<val_rgt>,Arial,16,[front_[nation]],[back_[nation]],[out_[nation]]
if=[command]<>0
    outline=[all],[command],<val_cnt>,Arial,16,[front_[nation]],[back_[nation]],[out_[nation]]
endif
outline=[all],[id],<val_id>,Arial,7,[front_[nation]],[back_[nation]],[out_[nation]]
if=[type]=inf
    line=[all],<img_cnt,PTL>,<img_cnt,PBR>,[front_[nation]],0.04
    line=[all],<img_cnt,PBL>,<img_cnt,PTR>,[front_[nation]],0.04
    line=[all],<img_cnt,PTL>,<img_cnt,PBR>,[out_[nation]],0.02
    line=[all],<img_cnt,PBL>,<img_cnt,PTR>,[out_[nation]],0.02
endif
if=[type]=cav
    line=[all],<img_cnt,PBL>,<img_cnt,PTR>,[front_[nation]],0.04
    line=[all],<img_cnt,PBL>,<img_cnt,PTR>,[out_[nation]],0.02
endif
if=[type]=arm
    ellipse=[all],<img_cnt2>,[front_[nation]],EMPTY,0.04
    ellipse=[all],<img_cnt2>,[out_[nation]],EMPTY,0.02
endif
if=[type]=hq
    outline=[all],HQ,<img_cnt>,Arial,16,[front_[nation]],[back_[nation]],[out_[nation]]
endif
rectangle=[all],<img_cnt>,[front_[nation]],"empty",0.05
rectangle=[all],<img_cnt>,[out_[nation]],"empty",0.02
```

Data file (data.txt):

```
nation,type,combat,movement,command,id,number
fre,inf,3,3,0,XXX,2
fre,cav,2,5,0,XX,2
fre,arm,4,4,0,XXX,2
fre,hq,1,4,3,"De Gaulle",1
ger,inf,3,3,0,XXXX,2
ger,arm,6,5,0,XXX,2
ger,arm,5,4,0,XXX,2
ger,hq,1,5,3,Rommel,1
```

 3 3	 3 3	 2 5	 2 5	 4 4	 4 4	 5 4	 5 4	De Gaulle HQ 1 3 4
 3 3	 3 3	 3 3	 3 3	 6 5	 6 5	 5 4	 5 4	Rommel HQ 1 3 5

Dice results

```
cardsize=18,3
border=rectangle
zoom=50
[all]=1-36
```

```
<d4>=0,0,3,3
<d6>=3,0,3,3
<d8>=6,0,3,3
<d10>=9,0,3,3
<d12>=12,0,3,3
<d20>=15,0,3,3
```

```
[col]=#000000
```

```
font=Arial,32,T,[col]
```

```
polygon=[all],<d4>,3,0,[col],EMPTY,0.1
text=[all],"{1d4}",<d4>
```

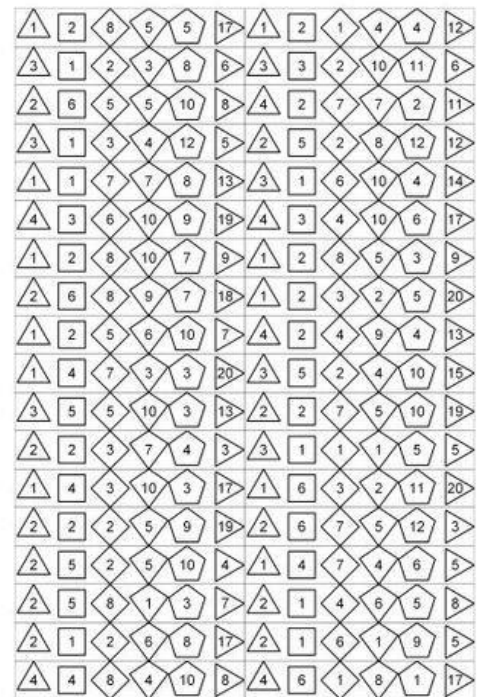
```
polygon=[all],<d6>,4,45,[col],EMPTY,0.1
text=[all],"{1d6}",<d6>
```

```
polygon=[all],<d8>,4,0,[col],EMPTY,0.1
text=[all],"{1d8}",<d8>
```

```
line=[all],9,1,10.5,0,[col],0.1
line=[all],10.5,0,12,1,[col],0.1
line=[all],12,1,10.5,3,[col],0.1
line=[all],10.5,3,9,1,[col],0.1
text=[all],"{1d10}",<d10>
```

```
polygon=[all],<d12>,5,0,[col],EMPTY,0.1
text=[all],"{1d12}",<d12>
```

```
polygon=[all],<d20>,3,90,[col],EMPTY,0.1
text=[all],"{1d20}",<d20>
```



Score track

```
unit=inch
cavassize=15,15

[side_a]=framebox(0,0,14,1,1,1,C)
[side_b]=framebox(14,0,1,14,1,1,C)
[side_c]=framebox(1,14,14,1,1,1,C)
[side_d]=framebox(0,1,1,14,1,1,C)

rectangle=0,<side*>,#000000,empty

font=arial,16,T,#000000
text=0,"{°-1}",<side_a*>
text=0,"{13+°}",<side_b*>,center,center,90
text=0,"{26+16-°}",<side_c*>,center,center,180
text=0,"{40+16-°}",<side_d*>,center,center,270

save=0,"board.png"0,0,15,15
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
15														15
16														16
17														17
18														18
19														19
20														20
21														21
22														22
23														23
24														24
25														25
26														26
27														27
28	11	40	39	38	37	36	35	34	33	32	31	30	29	28

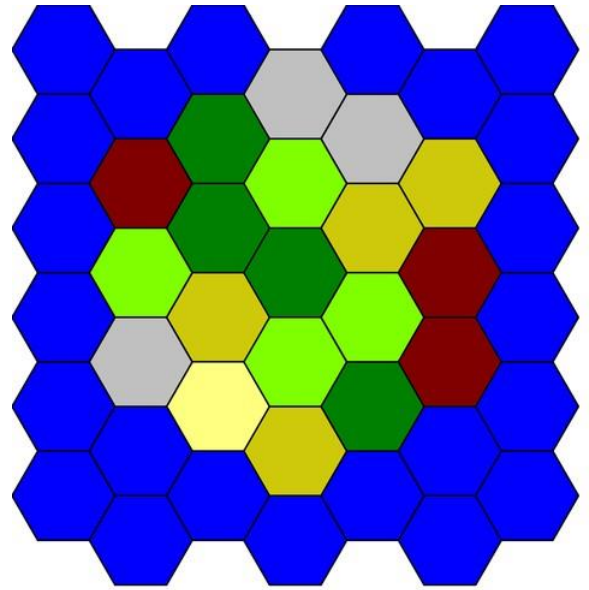
Boggle dice

```
N[a]=01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16
[range]=1-{(a)}
[d01]=LRYTTE
[d02]=VTHRWE
[d03]=EGHWNE
[d04]=SEOTIS
[d05]=ANAEEG
[d06]=IDSYTT
[d07]=OATTOW
[d08]=MTOICU
[d09]=AFPKFS
[d10]=XLDERI
[d11]=HCPOAS
[d12]=ENSIEU
[d13]=YLDEVR
[d14]=ZNRNHL
[d15]=NMIQHU
[d16]=OBBAOJ
CARDSIZE = 4.5, 4.5
FONT = Arial, 96, , #000000
TEXT = [range], [[d[a]]:d6,1], 0, 0, 100%, 100%
RECTANGLE = [range], 0, 0, 100%, 100%, #0000FF, EMPTY, 10%
```

N	S	R	O
V	O	V	L
E	O	S	E
I	T	H	E

Catan map

```
canvassize=35,35
[sea]=framehex(0,0,35,35,3,C)
[map]=framedisk(sea43,sea41)
'sea
polygon=0,<sea*>,6,90,#000000,#0000FF,0.1
'field
polygon=0,<4~!map*>,6,90,#000000,#CEC90B,0.1
'forest
polygon=0,<4~!map*>,6,90,#000000,#008000,0.1
'pasture
polygon=0,<4~!map*>,6,90,#000000,#80FF00,0.1
'mountain
polygon=0,<3~!map*>,6,90,#000000,#C0C0C0,0.1
'hill
polygon=0,<3~!map*>,6,90,#000000,#800000,0.1
'desert
polygon=0,<~!map*>,6,90,#000000,#FFFF80,0.1
save=0,"catan.jpg",0,0,35,35
```

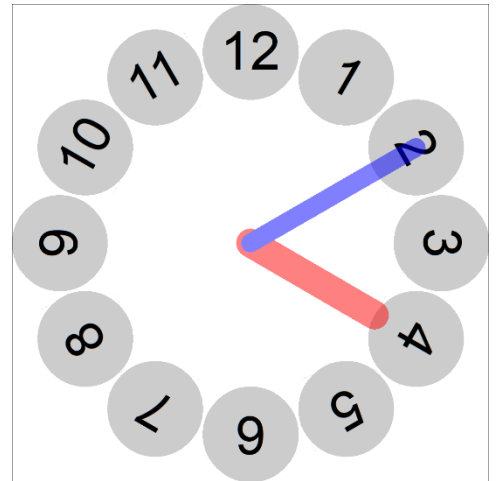


Clock

```
cardsize = 10,10
[clock] = frameclock(1, 1, 8, 8, 2, 2, 12)
ellipse = 1, <clock*>, #CCCCCC
font = Arial, 32, T, #000000
text = 1, "{°}", <clock*>, center, center, °*360/12

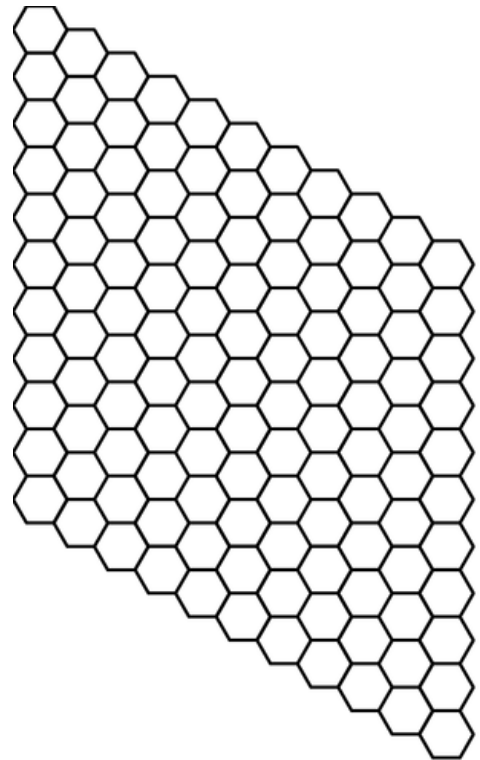
layer = 50
;hours
[hour] = frameclock(2, 2, 6, 6, 1, 1, 12)
line = 1, 5, 5, <hour4, PCC>, #FF0000, 0.6

;minutes
line = 1, 5, 5, <clock2, PCC>, #0000FF, 0.4
endlayer
```



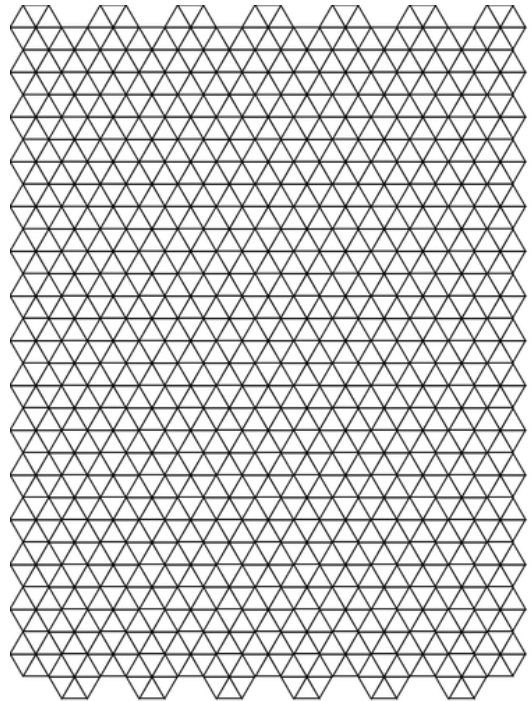
Hex board

```
page=21.59,27.94,portrait,hv
cardsize=16,24
border=None
[base]=framehex(0,0,16,24,0.85,C)
[hex01]=frameline(base0101,base0111)
[hex02]=frameline(base0201,base0211)
[hex03]=frameline(base0302,base0312)
[hex04]=frameline(base0402,base0412)
[hex05]=frameline(base0503,base0513)
[hex06]=frameline(base0603,base0613)
[hex07]=frameline(base0704,base0714)
[hex08]=frameline(base0804,base0814)
[hex09]=frameline(base0905,base0915)
[hex10]=frameline(base1005,base1015)
[hex11]=frameline(base1106,base1116)
polygon=1,<hex*>,6,90,#000000,EMPTY,0.1
```



Triangle map

```
canvassize=21,27  
[h]=framehex(0,0,21,27,1,C)  
star=0,<h*>,6,90,1,#000000,EMPTY,0.05  
hexgrid=0,0,0,21,27,1,,#000000,EMPTY,0.05  
save=0,"triangle.png",0,0,21,27
```



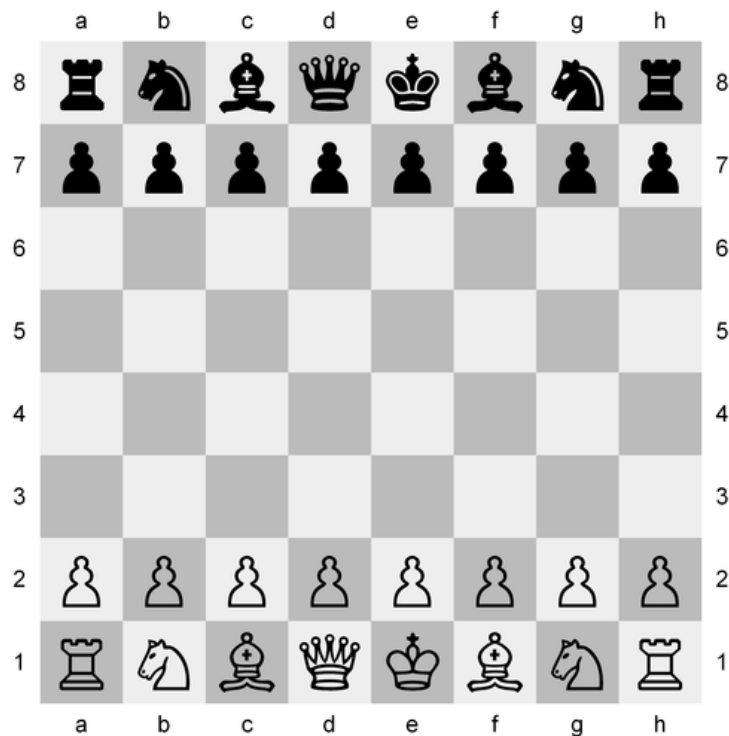
Chess board

With font "Chess Cases". Link: <http://www.enpassant.dk/chess/fonteng.htm#CASES>

```

canvassize=18,18
[ch]=framebox(1,1,16,16,2,2,E)
{[ch_white]=framelist(cha1,chl1,che1,chg1,chw1,chs1,che2,chg2,chw2,chs2,che3,chg3,chw3,chs3,che4,chg4,chw4,chs4,
cha5,chl5,che5,chg5,chw5,chs5,che6,chg6,chw6,chs6,che7,chg7,chw7,chs7,che8,chg8,chw8,chs8)}
{[ch_black]=framelist(chb1,chl1,chf1,chw1,chs1,che2,chg2,chw2,chs2,che3,chg3,chw3,chs3,che4,chg4,chw4,chs4,
chb5,chl5,chf5,chw5,chs5,che6,chg6,chw6,chs6,che7,chg7,chw7,chs7,che8,chg8,chw8,chs8)}
rectangle=0,<ch_white>,#EEEEEE
rectangle=0,<ch_black>,#BBBBBB
[tt]=framebox(1,0,16,1,2,1,N)
[tb]=framebox(1,17,16,1,2,1,N)
[t1]=framelist(tt1,tb1)
[t2]=framelist(tt2,tb2)
[t3]=framelist(tt3,tb3)
[t4]=framelist(tt4,tb4)
[t5]=framelist(tt5,tb5)
[t6]=framelist(tt6,tb6)
[t7]=framelist(tt7,tb7)
[t8]=framelist(tt8,tb8)
[sl]=framebox(0,1,1,16,1,2,N)
[sr]=framebox(17,1,1,16,1,2,N)
[s1]=framelist(sl1,sr1)
[s2]=framelist(sl2,sr2)
[s3]=framelist(sl3,sr3)
[s4]=framelist(sl4,sr4)
[s5]=framelist(sl5,sr5)
[s6]=framelist(sl6,sr6)
[s7]=framelist(sl7,sr7)
[s8]=framelist(sl8,sr8)
font=arial,16,,#000000
text=0,a,<t1>
text=0,b,<t2>
text=0,c,<t3>
text=0,d,<t4>
text=0,e,<t5>
text=0,f,<t6>
text=0,g,<t7>
text=0,h,<t8>
text=0,8,<s1>
text=0,7,<s2>
text=0,6,<s3>
text=0,5,<s4>
text=0,4,<s5>
text=0,3,<s6>
text=0,2,<s7>
text=0,1,<s8>
font="chess cases",48,T,#000000
[wpa]=p
[wkn]=n
[wbi]=b
[wro]=r
[wqu]=q
[wki]=k
[bpa]=o
[bkn]=m
[bbi]=v
[bro]=t
[bqu]=w
[bki]=l
text=0,[wpa],<ch?7>
text=0,[wro],<cha8>
text=0,[wkn],<chw8>
text=0,[wbi],<chc8>
text=0,[wqu],<chd8>
text=0,[wki],<che8>
text=0,[wkn],<chf8>
text=0,[wro],<chh8>
text=0,[bpa],<ch?2>
text=0,[bro],<cha1>
text=0,[bkn],<chb1>
text=0,[bbi],<chc1>
text=0,[bqu],<chd1>
text=0,[bki],<che1>
text=0,[bbi],<chf1>
text=0,[bkn],<chg1>
text=0,[bro],<chh1>
save=0,chessboard2.png,0,0,18,18

```

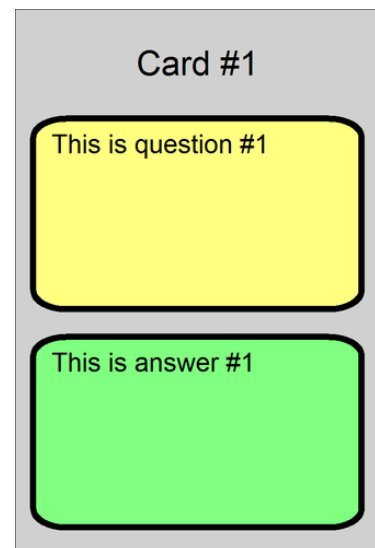


Trivia cards

```
linkmulti=num
link="q&a.txt"
[all]="1-{(id)}"
[card_id]=join("Card #",[id])
[background]=#D0D0D0
[ink]=#000000
[col_q]=#FFFF80
[col_a]=#80FF80
rectangle=[all],0,0,100%,100%,[background]
font=arial,16,,[ink],[background]
text=[all],[card_id],0,0,100%,20%,center,center
roundrect=[all],5%,20%,90%,35%,#000000,[col_q],0.1
font=arial,12,,[ink],[col_q]
text=[all],[question],10%,22%,80%,31%,left,wordwrap
roundrect=[all],5%,60%,90%,35%,#000000,[col_a],0.1
font=arial,12,,[ink],[col_a]
text=[all],[answer],10%,62%,80%,31%,left,wordwrap
```

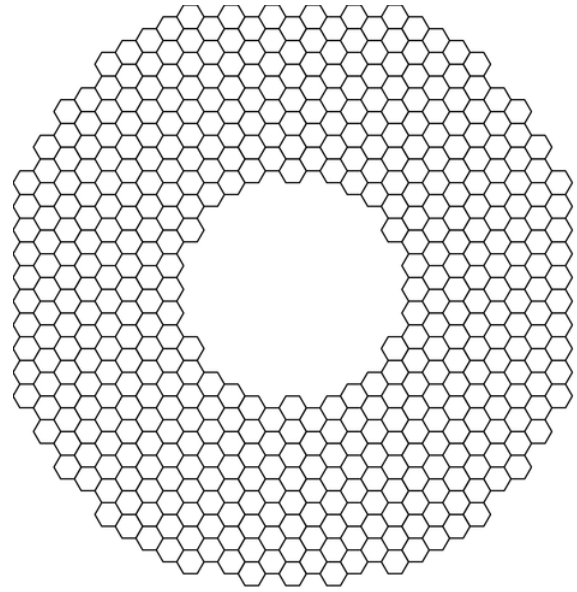
Data file (q&a.txt):

```
id,num,question,answer
1,1,"This is question #1","This is answer #1"
2,1,"This is question #2","This is answer #2"
3,1,"This is question #3","This is answer #3"
4,1,"This is question #4","This is answer #4"
5,1,"This is question #5","This is answer #5"
6,1,"This is question #6","This is answer #6"
7,1,"This is question #7","This is answer #7"
8,1,"This is question #8","This is answer #8"
9,1,"This is question #9","This is answer #9"
10,1,"This is question #10","This is answer #10"
```



Hex racetrack

```
canvassize=42,44  
[hexa]=framehex(0,0,42,44,1,C)  
[hexb]=framedisk(hexa1412,hexa0112)  
[hexc]=framedisk(hexa1412,hexa0912)  
[hexd]=framesub(hexb*,hexc*)  
polygon=0,<hexd*>,6,90,#000000,EMPTY,0.1  
save=0,"track.png",0,0,42,44
```



Tuckbox

```
[img1]=none

[img2]=none
INPUTTEXT="wid","Width (cm)","6"
INPUTTEXT="hei","Height (cm)","9"
INPUTTEXT="dep","Depth (cm)","3"
INPUTCHOICE="extra","Add extra lines","Yes","Yes|No"
INPUTTEXT="img1","Box image (front)","",G
INPUTTEXT="img2","Box image (rear)","",G

[fla]=[dep]/2
[ide]=[fla]/2
[coll]=#000000
[col2]=EMPTY
[thi]=0.1

'Uncomment the following line for A4 paper
PAGE=21,29.7,PORTRAIT,HV

'Uncomment the following line for Letter paper
'PAGE=21.59,27.94,PORTRAIT,HV

BORDER=NONE
CARDSIZE=[fla]+[dep]+[hei]+[dep]+[fla],[dep]+[wid]+[dep]+[wid]+[fla]

IF=[img1]=none
ELSE
    IMAGE=1,[img1],[fla]+[dep],[dep],[hei],[wid],90,P
ENDIF

IF=[img2]=none
ELSE
    IMAGE=1,[img2],[fla]+[dep],[dep]+[wid]+[dep],[hei],[wid],90,P
ENDIF

RECTANGLE=1,[fla]+[dep],0,[hei],[dep],[coll],[col2],[thi]
RECTANGLE=1,[fla],[dep],[dep],[wid],[coll],[col2],[thi]
RECTANGLE=1,[fla]+[dep],[dep],[hei],[wid],[coll],[col2],[thi]
RECTANGLE=1,[fla]+[dep]+[hei],[dep],[dep],[wid],[coll],[col2],[thi]
RECTANGLE=1,[fla]+[dep],[dep]+[wid],[hei],[dep],[coll],[col2],[thi]
RECTANGLE=1,[fla]+[dep],[dep]+[wid]+[dep],[hei],[wid],[coll],[col2],[thi]

LINE=1,[fla]+[dep],0,[fla]+[dep]-[fla],[ide],[coll],[thi]
LINE=1,[fla]+[dep]-[fla],[ide],[fla]+[dep]-[fla],[dep]-[ide],[coll],[thi]
LINE=1,[fla]+[dep]-[fla],[dep]-[ide],[fla]+[dep],[dep],[coll],[thi]

LINE=1,[fla]+[dep]+[hei],0,[fla]+[dep]+[hei]+[fla],[ide],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]+[fla],[ide],[fla]+[dep]+[hei]+[fla],[dep]-[ide],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]+[fla],[dep]-[ide],[fla]+[dep]+[hei],[dep],[coll],[thi]

LINE=1,[fla],[dep],[fla]-[fla],[dep]+[ide],[coll],[thi]
LINE=1,[fla]-[fla],[dep]+[ide],[fla]-[fla],[dep]+[wid]-[ide],[coll],[thi]
LINE=1,[fla]-[fla],[dep]+[wid]-[ide],[fla],[dep]+[wid],[coll],[thi]

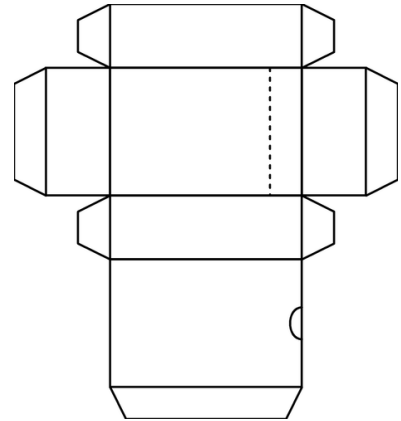
LINE=1,[fla]+[dep]+[hei]+[dep],[dep],[fla]+[dep]+[hei]+[dep]+[fla],[dep]+[ide],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]+[dep]+[fla],[dep]+[ide],[fla]+[dep]+[hei]+[dep]+[fla],[dep]+[wid]-[ide],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]+[dep]+[fla],[dep]+[wid]-[ide],[fla]+[dep]+[hei]+[dep],[dep]+[wid],[coll],[thi]

LINE=1,[fla]+[dep],[dep]+[wid],[fla]+[dep]-[fla],[dep]+[wid]+[ide],[coll],[thi]
LINE=1,[fla]+[dep]-[fla],[dep]+[wid]+[ide],[fla]+[dep]-[fla],[dep]+[wid]+[dep]-[ide],[coll],[thi]
LINE=1,[fla]+[dep]-[fla],[dep]+[wid]+[dep]-[ide],[fla]+[dep],[dep]+[wid]+[dep],[coll],[thi]

LINE=1,[fla]+[dep]+[hei],[dep]+[wid],[fla]+[dep]+[hei]+[fla],[dep]+[wid]+[ide],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]+[fla],[dep]+[wid]+[ide],[fla]+[dep]+[hei]+[fla],[dep]+[wid]+[dep]-[ide],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]+[fla],[dep]+[wid]+[dep]-[ide],[fla]+[dep]+[hei],[dep]+[wid]+[dep],[coll],[thi]

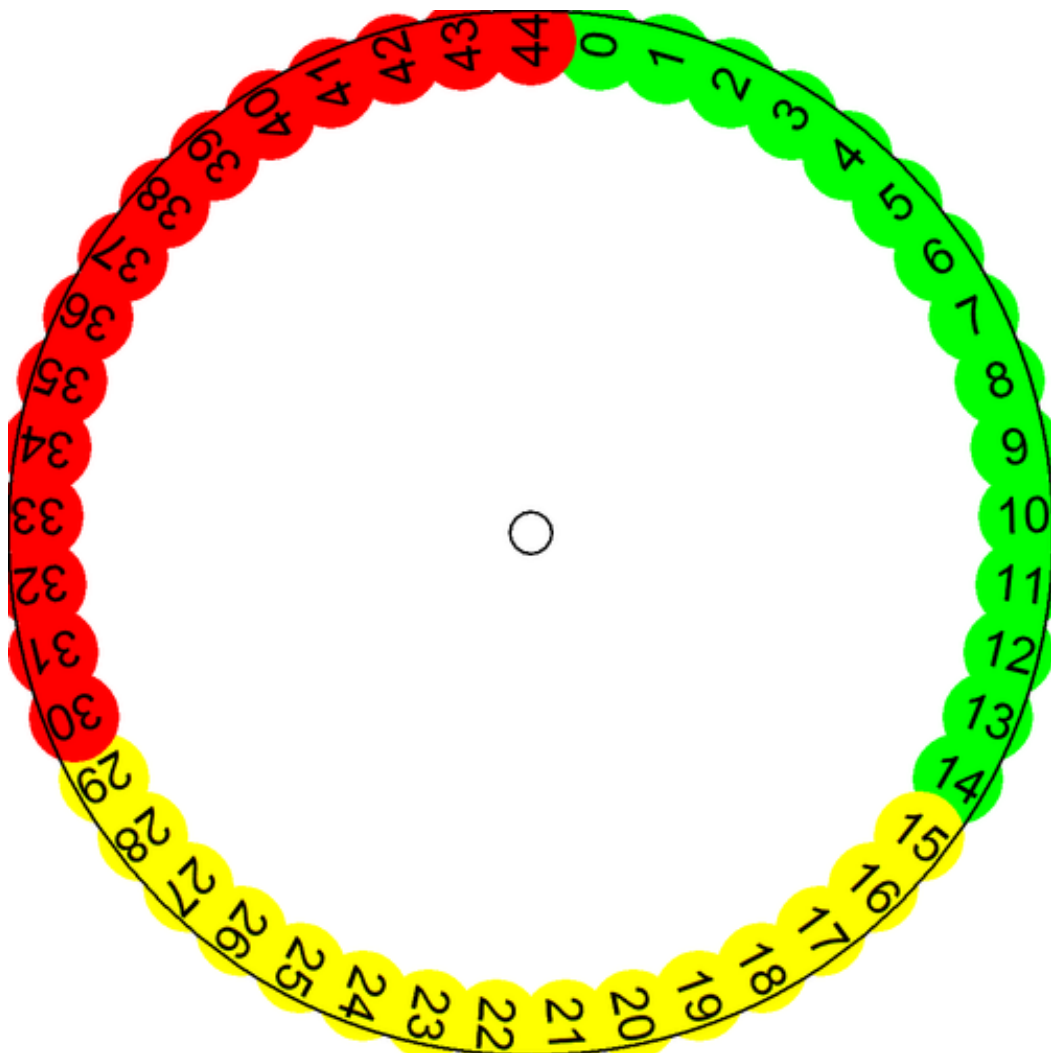
LINE=1,[fla]+[dep],[dep]+[wid]+[dep]+[wid],[fla]+[dep]+[ide],[dep]+[wid]+[dep]+[wid]+[fla],[coll],[thi]
LINE=1,[fla]+[dep]+[ide],[dep]+[wid]+[dep]+[wid]+[fla],[fla]+[dep]+[hei]-[ide],
[dep]+[wid]+[dep]+[wid]+[fla],[coll],[thi]
LINE=1,[fla]+[dep]+[hei]-[ide],[dep]+[wid]+[dep]+[wid]+[fla],[fla]+[dep]+[hei],
[dep]+[wid]+[dep]+[wid],[coll],[thi]

IF=[extra]=Yes
    LINE=1,[fla]+[dep]+[hei]-[fla],[dep],[fla]+[dep]+[hei]-[fla],[dep]+[wid],[coll],[thi],DSS
    {
        BEZIER=1,
        [fla]+[dep]+[hei],[dep]+[wid]+[dep]+[wid]/2-[ide],
        [fla]+[dep]+[hei]-[ide],[dep]+[wid]+[dep]+[wid]/2-[ide],
        [fla]+[dep]+[hei]-[ide],[dep]+[wid]+[dep]+[wid]/2+[ide],
        [fla]+[dep]+[hei],[dep]+[wid]+[dep]+[wid]/2+[ide],
        [coll],[thi]}
    }
ENDIF
```



Number wheel

```
[size]=3.5
[numbers]=45
unit=inch
cardsize=[size],[size]
[num]=frameclock(0.1,0.1,[size]-0.2,[size]-0.2,0.3,0.3,[numbers])
[green]=framelist(num1..num15)
[yellow]=framelist(num16..num30)
[red]=framelist(num31..num45)
font=arial,12,T,#000000
ellipse=1,<green>,#00FF00
ellipse=1,<yellow>,#FFFF00
ellipse=1,<red>,#FF0000
text=1,{°-1},<num*>,center,center,°*360/[numbers]-90
ellipse=1,0,0,100%,100%,#000000,empty,0.01
ellipse=1,48%,48%,4%,4%,#000000,empty,0.01
save=1,"wheel_col.png",0,0,100%,100%
```



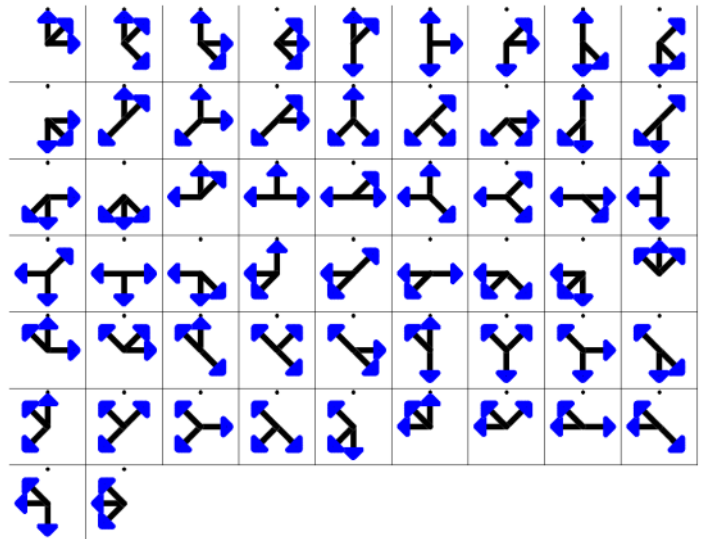
Tripples tiles

```
cardsize=2,2
```

```
'draw and save an image (an arrow)
line=0,1 ,0.2,1 ,1 ,#000000,0.15
line=0,1 ,0.2,1.2,0.4,#0000FF,0.15
line=0,1.2,0.4,0.8,0.4,#0000FF,0.15
line=0,0.8,0.4,1 ,0.2,#0000FF,0.15
save=0,"arrow.bmp",0,0,2,2
```

```
'create all the permutations (256) of two elements (0 and 1) taken eight times
pr[perm]8=0|1
'takes only the tiles with three '1'
[tiles]=filter(+[perm],3)
```

```
[all]=1-{(tiles)}
'draws a dot
ellipse=[all],0.95,0.05,0.1,0.1,#000000
'draws the eight arrows
if=[tiles:8,1]=1
    image=[all],"arrow.bmp",0,0,2,2,0,T
endif
if=[tiles:7,1]=1
    image=[all],"arrow.bmp",-0.5,-0.5,3,3,45,T
endif
if=[tiles:6,1]=1
    image=[all],"arrow.bmp",0,0,2,2,90,T
endif
if=[tiles:5,1]=1
    image=[all],"arrow.bmp",-0.5,-0.5,3,3,135,T
endif
if=[tiles:4,1]=1
    image=[all],"arrow.bmp",0,0,2,2,180,T
endif
if=[tiles:3,1]=1
    image=[all],"arrow.bmp",-0.5,-0.5,3,3,225,T
endif
if=[tiles:2,1]=1
    image=[all],"arrow.bmp",0,0,2,2,270,T
endif
if=[tiles:1,1]=1
    image=[all],"arrow.bmp",-0.5,-0.5,3,3,315,T
endif
```

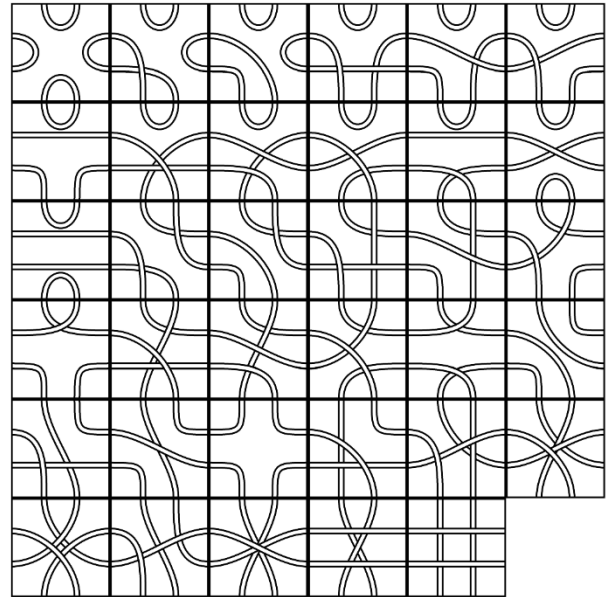


Path tiles

```

oversample=2
macro=tile,(range),(key),(char),(color),(width)
  beziers=(range)
  if=[(key):1,1]=(char)
    beziers=(range),1,0,1,1,(color),(width)
  endif
  if=[(key):2,1]=(char)
    beziers=(range),2,0,2,1,(color),(width)
  endif
  if=[(key):3,1]=(char)
    beziers=(range),3,1,2,1,(color),(width)
  endif
  if=[(key):4,1]=(char)
    beziers=(range),3,2,2,2,(color),(width)
  endif
  if=[(key):5,1]=(char)
    beziers=(range),2,3,2,2,(color),(width)
  endif
  if=[(key):6,1]=(char)
    beziers=(range),1,3,1,2,(color),(width)
  endif
  if=[(key):7,1]=(char)
    beziers=(range),0,2,1,2,(color),(width)
  endif
  if=[(key):8,1]=(char)
    beziers=(range),0,1,1,1,(color),(width)
  endif
end

```



```

cardsize=3,3

```

```

pxxs[list]8=a|a|b|b|c|c|d|d

```

```

[range]=1-{(list)}

```

```

tile=[range],[list],a,#000000,0.2
tile=[range],[list],a,#FFFFFF,0.1
tile=[range],[list],b,#000000,0.2
tile=[range],[list],b,#FFFFFF,0.1
tile=[range],[list],c,#000000,0.2
tile=[range],[list],c,#FFFFFF,0.1
tile=[range],[list],d,#000000,0.2
tile=[range],[list],d,#FFFFFF,0.1

```

```

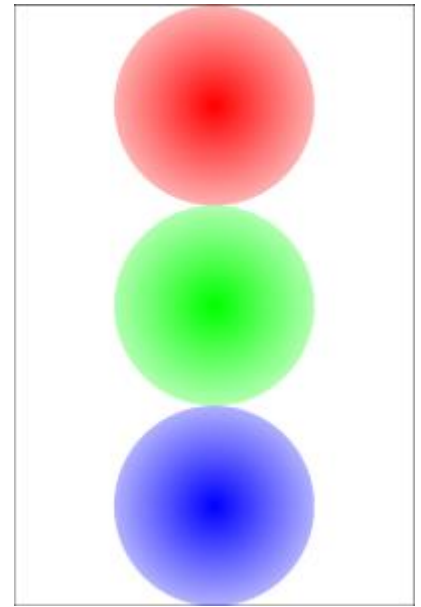
rectangle=[range],0,0,3,3,#000000,EMPTY,0.1

```

Combinations

```
ellipse=0,0,0,6,6,#FF0000#FFFFFF@360
save=0,a.png,0,0,6,6
ellipse=0,0,0,6,6,#00FF00#FFFFFF@360
save=0,b.png,0,0,6,6
ellipse=0,0,0,6,6,#0000FF#FFFFFF@360
save=0,c.png,0,0,6,6
ellipse=0,0,0,6,6,#FF00FF#FFFFFF@360
save=0,d.png,0,0,6,6
ellipse=0,0,0,6,6,#FFFF00#FFFFFF@360
save=0,e.png,0,0,6,6
ellipse=0,0,0,6,6,#00FFFF#FFFFFF@360
save=0,f.png,0,0,6,6
ellipse=0,0,0,6,6,#000000#FFFFFF@360
save=0,g.png,0,0,6,6

c[comb]3=a|b|c|d|e|f|g
[all]=1-{ (comb) }
icon=[all],a,a.png
icon=[all],b,b.png
icon=[all],c,c.png
icon=[all],d,d.png
icon=[all],e,e.png
icon=[all],f,f.png
icon=[all],g,g.png
icons=[all],[comb],1.5,1.5,3,6,3,3,0,P
```



Standard 52-deck of cards

```

sequence=number
A
2
3
4
5
6
7
8
9
10
endsequence

sequence=face
J
Q
K
endsequence

sequence=
suit      |\169\
suit_fnt|Symbol
suit_col|#FF0000

suit      |\168\
suit_fnt|Symbol
suit_col|#FF0000

suit      |\167\
suit_fnt|Symbol
suit_col|#000000

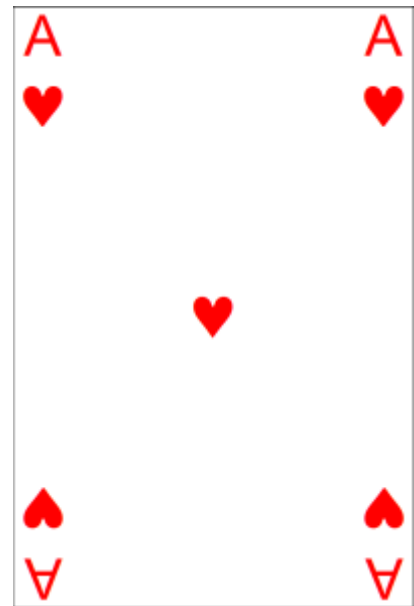
suit      |\170\
suit_fnt|Symbol
suit_col|#000000
endsequence

<corner1>=0,0,15%,20%
<corner1_a>=0,0,15%,10%
<corner1_b>=0,10%,15%,10%
<corner2_a>=85%,0,15%,10%
<corner2_b>=85%,10%,15%,10%
<corner3_a>=0,90%,15%,10%
<corner3_b>=0,80%,15%,10%
<corner4_a>=85%,90%,15%,10%
<corner4_b>=85%,80%,15%,10%
<core>=15%,20%,70%,60%

cards={ (suit) * ((number)+(face))+1}
for=a,1,{ (suit) }
  for=b,1,{ (number) }
    font=Arial,24,T,{suit_col?a}
    text={b+((a)-1)*((number)+(face))},{number?b},<corner1_a>
    text={b+((a)-1)*((number)+(face))},{number?b},<corner2_a>
    text={b+((a)-1)*((number)+(face))},{number?b},<corner3_a>,center,center,180
    text={b+((a)-1)*((number)+(face))},{number?b},<corner4_a>,center,center,180
    font={suit_fnt?a},32,T,{suit_col?a}
    text={b+((a)-1)*((number)+(face))},{suit?a},<corner1_b>
    text={b+((a)-1)*((number)+(face))},{suit?a},<corner2_b>
    text={b+((a)-1)*((number)+(face))},{suit?a},<corner3_b>,center,center,180
    text={b+((a)-1)*((number)+(face))},{suit?a},<corner4_b>,center,center,180
    text={b+((a)-1)*((number)+(face))},{ {suit?a}Xb},<core>,center,charwrap
  next
next
for=a,1,{ (suit) }
  for=b,1,{ (face) }
    font=Arial,24,T,{suit_col?a}
    text={b+((a)-1)*((number)+(face))+(number)},{face?b},<corner1_a>
    text={b+((a)-1)*((number)+(face))+(number)},{face?b},<corner2_a>
    text={b+((a)-1)*((number)+(face))+(number)},{face?b},<corner3_a>,center,center,180
    text={b+((a)-1)*((number)+(face))+(number)},{face?b},<corner4_a>,center,center,180
    font={suit_fnt?a},32,T,{suit_col?a}
    text={b+((a)-1)*((number)+(face))+(number)},{suit?a},<corner1_b>
    text={b+((a)-1)*((number)+(face))+(number)},{suit?a},<corner2_b>
    text={b+((a)-1)*((number)+(face))+(number)},{suit?a},<corner3_b>,center,center,180
    text={b+((a)-1)*((number)+(face))+(number)},{suit?a},<corner4_b>,center,center,180
    font=Arial,128,T,{suit_col?a}
    text={b+((a)-1)*((number)+(face))+(number)},{face?b},<core>
  next
next

rectangle={ (suit) * ((number)+(face))+1},0,0,100%,100%,#FF0000#0000FF#90
font=arial,48,DNT,#FFFFFF
text={ (suit) * ((number)+(face))+1},"nanDECK",0,0,100%,100%

```



Hexagonal tiles

```

cardsize=5,5
border=none

linkmulti=number
link=data.xlsx
polygon=,0,0,100%,100%,6,30,[color_hex]

[conn]=frameclock(0,0,100%,100%,10%,10%,6)

if=(1@[lines])
line=,<conn1,pcc>,50%,50%,[color_line],15%
endif
if=(2@[lines])
line=,<conn2,pcc>,50%,50%,[color_line],15%
endif
if=(3@[lines])
line=,<conn3,pcc>,50%,50%,[color_line],15%
endif
if=(4@[lines])
line=,<conn4,pcc>,50%,50%,[color_line],15%
endif
if=(5@[lines])
line=,<conn5,pcc>,50%,50%,[color_line],15%
endif
if=(6@[lines])
line=,<conn6,pcc>,50%,50%,[color_line],15%
endif

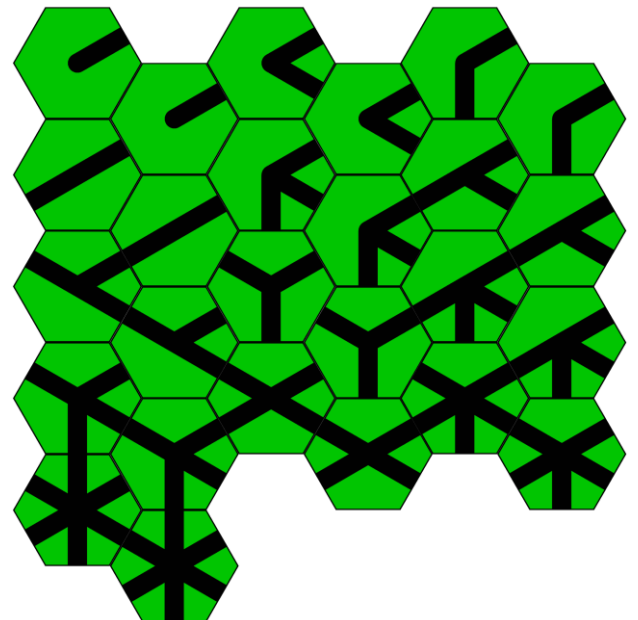
polygon=,0,0,100%,100%,6,30,#000000,empty,2%

chromakey=#FF0000
layer
polygon=,0,0,100%,100%,6,30,#FF0000
endlayer

save=,"tile{$}.png",0,6.735%,100%,86.53%,#FFFFFF

```

	A	B	C	D	E	F
1	Color_hex	Lines	Color_line	Number		
2	#00C500	1	#000000	2		
3	#00C500	12	#000000	2		
4	#00C500	13	#000000	2		
5	#00C500	14	#000000	2		
6	#00C500	123	#000000	2		
7	#00C500	124	#000000	2		
8	#00C500	125	#000000	2		
9	#00C500	135	#000000	2		
10	#00C500	1234	#000000	2		
11	#00C500	1235	#000000	2		
12	#00C500	1245	#000000	2		
13	#00C500	12345	#000000	2		
14	#00C500	123456	#000000	2		
15						
16						



Double-sided printing

page=21,29.7,portrait,hv

link=data.xlsx

```
[range_front]=1-{(front)}  
[range_back]={ (front)+1}-{ (front)*2}
```

```
font=arial,32,,#FFFFFF,#0000FF  
text=[range_front],[front],0,0,100%,100%
```

```
font=arial,32,,#FFFFFF,#FF0000  
text=[range_back],[back],0,0,100%,100%
```

```
duplex=[range_front],[range_back]  
print=duplex
```

	A	B	C
1	front	back	
2	Front 1	Back 1	
3	Front 2	Back 2	
4	Front 3	Back 3	
5	Front 4	Back 4	
6	Front 5	Back 5	
7	Front 6	Back 6	
8			
9			

Save individual cards from a PDF

```
[filename]=deck.pdf
[page_wid]=8.5
[page_hei]=11
[num_wid]=3
[num_hei]=3
[left_marg]=0.5
[top_marg]=0.25
unit=inch
page=[page_wid],[page_hei],portrait,hv
cardsize=[page_wid],[page_hei]
[pages]=pdfpages([filename])
loadpdf=1-[pages],[filename],0,0,[page_wid],[page_hei],{$}
[area_wid]=([page_wid]-[left_marg]*2)
[area_hei]=([page_hei]-[top_marg]*2)
[total]=([num_wid]*[num_hei])
[cut]=framebox([left_marg],[top_marg],[area_wid],[area_hei],[area_wid]/[num_wid]
,[area_hei]/[num_hei],N)
save=1-[pages],"card{($-1)*[total]+°}.png",<cut*>
```