LaTeX `classdeck` documentation

Jérôme Euzenat

March 25, 2020

Abstract

The `classdeck` package provides commands to draw cards and board for the *Class?* game. It can be used as well for the *Set!* get of which it is a superset.

In order to develop the *Class?* game*Set!*[1], we took inspiration from the TikZ package for the *Set!*[2] game developed by Gwyn Whieldon in 2012[3]. Currently, what remains of Gwyn's style is the spirit and the `vertical stripes` (see §1.4).

We first describe the possible card features (§1): shape, number, colours and patterns. Then we consider some graphical attributes of the drawings (§2: size, orientation, contours) before providing convenient macros for drawing cards (§3). We introduce bundles which are sets of consistent groups of features that are usually implemented in one game material (§4). Finally, some example of the use and associated utilities are presented (§5). In particular, they allow to draw *Class?* classifications.

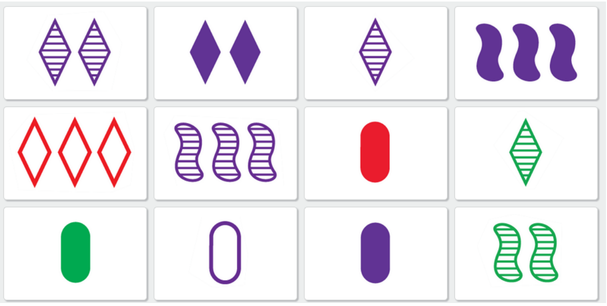
1 *Class?* card features

Class? cards are displayed with respect to some features of objects: shape, number, colors, fillings. For each features, there are three possible values. We show below sets of consistent values.

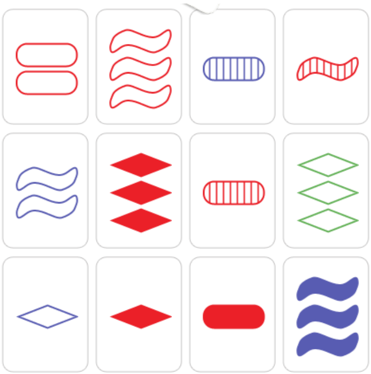
In addition, *Class?* has one joker value representing the absence of knowledge about a feature value. We show below the possible values for each features.

1.1 Shapes

Here are some layouts for available original *Set!* cards:



original



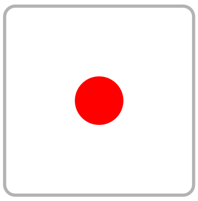
other layout

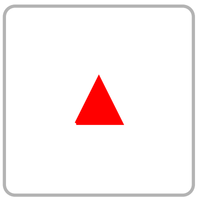
Here are the available consistent sets of shapes:

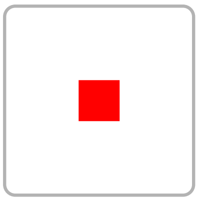
[1] `https://moex.inria.fr/mediation/class/`

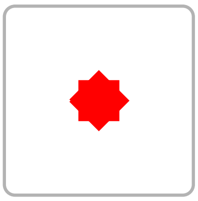
[2] `https://setgame.com`

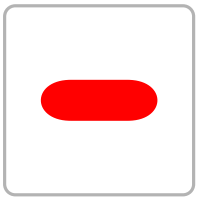
[3] `https://www.ctan.org/pkg/setdeck`

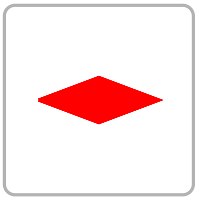

circle

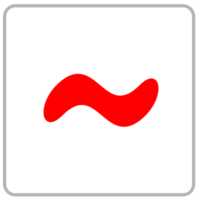

triangle

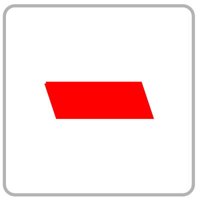

square

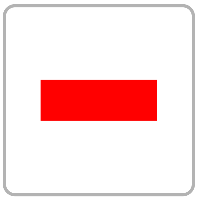

star8

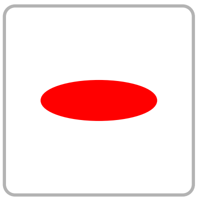

wurst

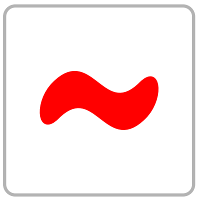

diamond

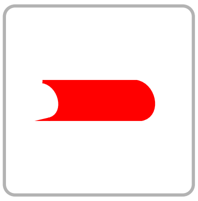

peanut


trapez


rect


oval


squiggle


ticket

In the default bundle, these shapes are called `\firstclassshape`, `\secondclassshape`, `\thirdclassshape`, and `\jokerclassshape`.

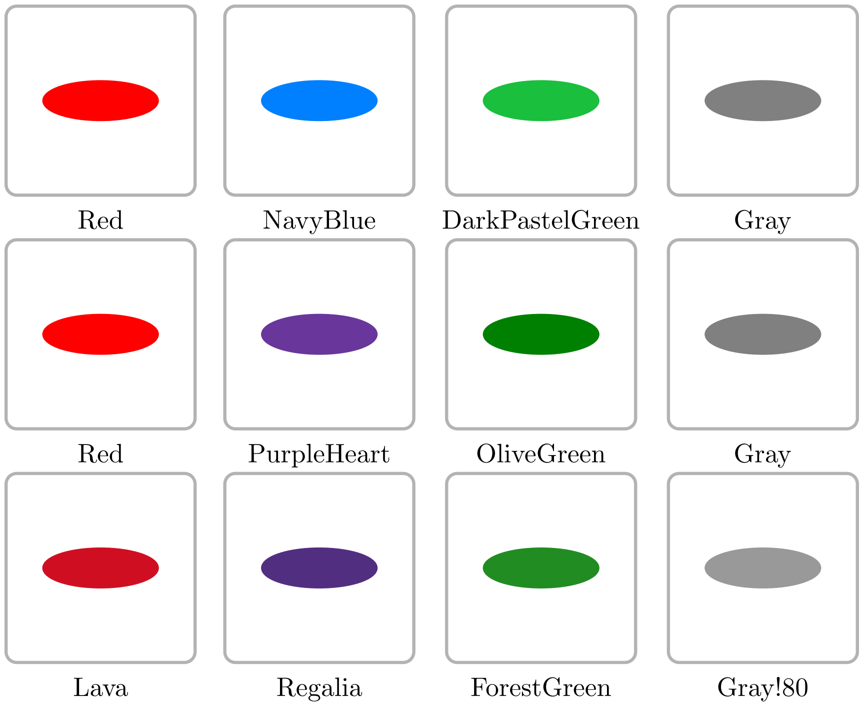
1.2 Numbers

The numbers are classically 1, 2 or 3. The joker value has been arbitrarily set to 0 (it is displayed as 5 elements).

1.3 Colors

Here are available sets of colors. The joker color is in principle set to gray.

In the default bundle, these colors are called `\redclasscolor`, `\blueclasscolor`, `\greenclasscolor` and `\jokerclasscolor`. Here are a sample of their values:

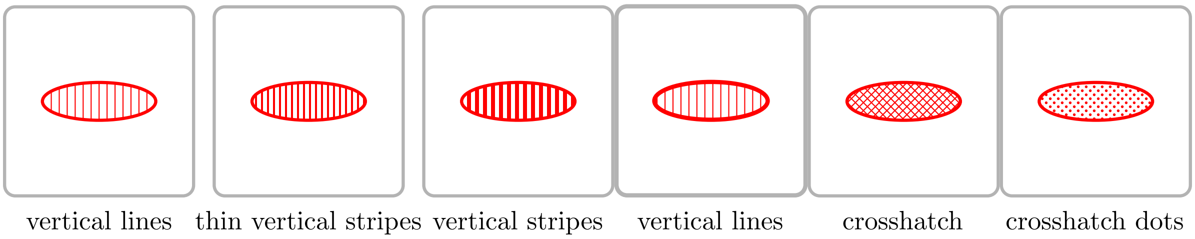


In addition, we preserved the initial convention of `setdeck` to define three variants of these colors: bordercolor, stripecolor (for the `\fillingclasspattern`), fillcolor (for the `\fullclasspattern`) They were defined as:
Red Red, Red!90, Red!40
Blue RoyalPurple, RoyalPurple!90, RoyalPurple!40
Green OliveGreen, OliveGreen!90, PineGreen!40

1.4 Patterns

Filling is characterised by patterns corresponding to either no filling, full filling or a pattern filling. The joker filling is also a pattern. In the default bundle, these patterns are called `\emptyclassfilling`, `\stripedclassfilling`, `\fullclassfilling` and `\jokerclassfilling`.

Here are already some patterns defined in PGF (which can be used as such):



Such patterns are defined as PGF patterns. So actually, the line drawing attributes (`thin`, `thick`, `very thick`, etc.) do not apply to patterns. These are written in pgf and they rely on the size of the initial “point”.

Moreover, it is noticeable that patterns are not easily rotated (or not rotated at all). This is a feature of pgf/tikz that makes it fast to render. Hence so far, when a card is rotated, the stripes keep their initial orientation...

2 Size, form and orientation of cards

Drawing cards goes beyond their features. They may be drawn in a variety of size, orientation, shapes and border.

2.1 Square or rectangular

Card may be square or rectangular.

So far these features are tied to bundles and cannot be easily modified (see §4).

2.2 Orientation

This is a difficult subject because orientation may mean two things: it is possible to orientate rectangular cards with the large side horizontal or vertical. Vertical is a classical way to hold cards, horizontal provides a better filling in case of horizontal shapes; It is also possible to rotate the cards for printing a tree vertically for instance (see §5.4). (but the vertical stripes remain vertical, see §1.4).

We use the first optional card macro parameter to specify orientation (see §3).

This also shows that there is lattitude for eventually deciding the orientation.

This is not difficult, but the orientation must be passed to the whole element, a scope rotation does not work.

2.3 Size

Several sizes of cards may be drawn by scaling and adapting graphic attributes.

It seems that obtaining small cards can simply be obtained by using `thin` instead of `thick` or `very thick`.

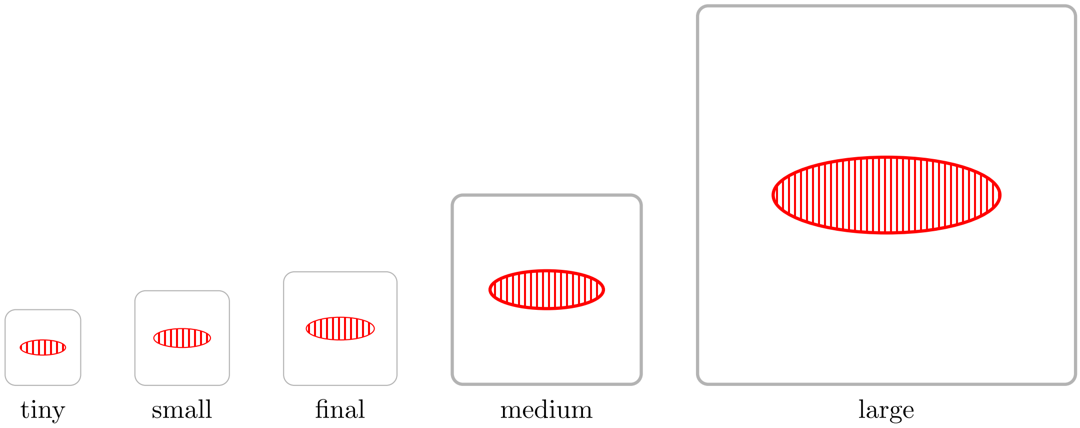
We use the first optional card macro parameter to specify size (see §3). It can be made of three different elements:

- the scale (`scale=.5`)
- the size of strokes (`thin`, `thick`, `very thick`)
- the type of stripes as defined in setdeck (vertical stripes, thin vertical stripes)

We have defined five packages that can be used as this first parameter:

- `tinyclasssize = thin,scale=.2` (+ `thin vertical stripes`)
- `smallclasssize = thin,scale=.25` (+ `thin vertical stripes`)
- `finalclasssize = thin,scale=.30` (+ `thin vertical stripes`)
- `mediumclasssize = very thick,scale=.5`
- `largeclasssize = very thick,scale=1`
- `largerclasssize = ultra thick,scale=1.2` is the size of printed cards (but may be increased)

Here is the result for each of these classes:



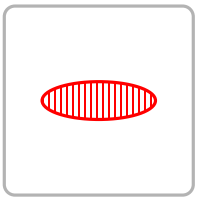
2.4 Borders

It is possible to switch card borders by redefining the variable `\classcontour` or by setting the value of the `outline` package parameter (at the level of the whole document) to:

- `outline` (default): displays card ouline as in most examples in this document;

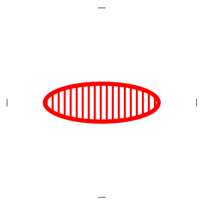
– `nocontour`: does not display any border;
– `guideline`: does display marks for cutting paper around the card.

The result is:



outline

noborder

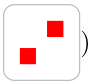


guideline

3 Card macros

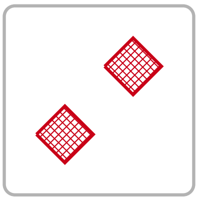
We introduce two commands for displaying cards.
`\classcard[{<ScaleParams>}]{<Number>}{<Filling>}{<Color>}{<Shape>}{<X>}{<Y>}`.

and
`\classpict{<ScaleParams>}{<Number>}{<Shading>}{<Color>}{<Shape>}` or
`\classtext{<Number>}{<Shading>}{<Color>}{<Shape>}`.

The former is usable in tikz and draws the class at the X and Y (should add Z) coordinates; the latter puts the `classcard` within a `tikzpicture`. Hence, it can be inserted anywhere like normal LaTeX commands. This is in particular useful to insert in a tikZ node label. `classtext` is a variation for including cards with texts (for instance here: ).

`\classcard` is a rewrite of `\setcard`, except that, instead of passing numbers, one has to pass values which are actual numbers, pattern, color and shape. Hence, one can use any fancy colours (however, numbers only work from 0–3 and 5).

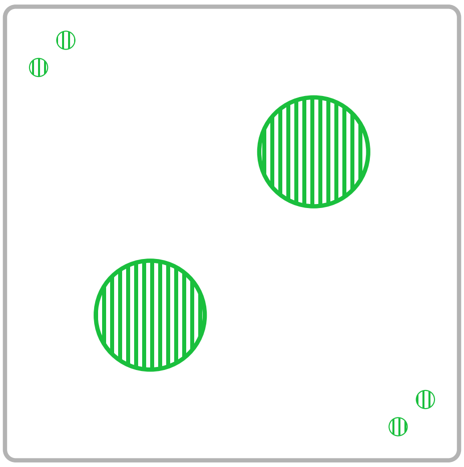
Here is the result of `\classcard{mediumclasssize}{2}{crosshatch}{harvardcrimson}{losange}{0}{0}`:



Specific variables are defined for the default display (see §4). They are:

Number (a number)	Filling (a pattern)	Color (a color)	Shape (a shape)
0	`\jokerclasspattern`	`\jokerclasscolor`	`\jokerclassshape`
1	`\emptyclasspattern`	`\redclasscolor`	`\firstclassshape`
2	`\fillingclasspattern`	`\blueclasscolor`	`\secondclassshape`
3	`\fullclasspattern`	`\greenclasscolor`	`\thirdclassshape`

The first line corresponds to classes.
The actual usable card set is obtained by another macro:



4 Bundles

The easier way to display a consistent set of cards is to use a bundle which specifies the default card set. The bundle is specified by passing the `bundle` parameter when loading the package:

```
\usepackage[bundle=<bundlename>]{classdeck}
```

or later through:

```
\setclassbundle{<bundlename>}
```

You can change this and see the effect on this file or any other.

There are currently three bundles:

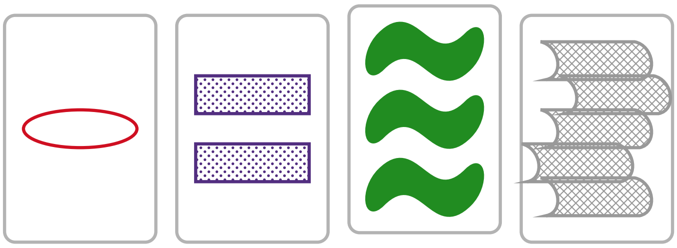
- class: the bundle that we designed for *Class?*;
- setline: the card layout of the *Set!* game that Lise has;
- setjerome: the apparently standard *Set!* game layout.

This document has been loaded with:

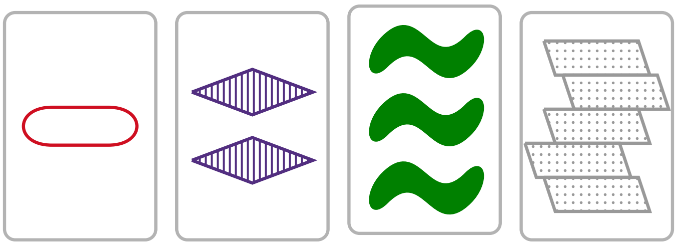
```
\usepackage[bundle=class]{classdeck}
```

and now you can look at the effect of:

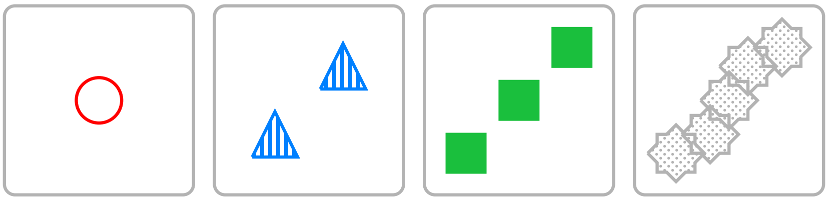
```
\setclassbundle{setline}
```



```
\setclassbundle{setjerome}
```



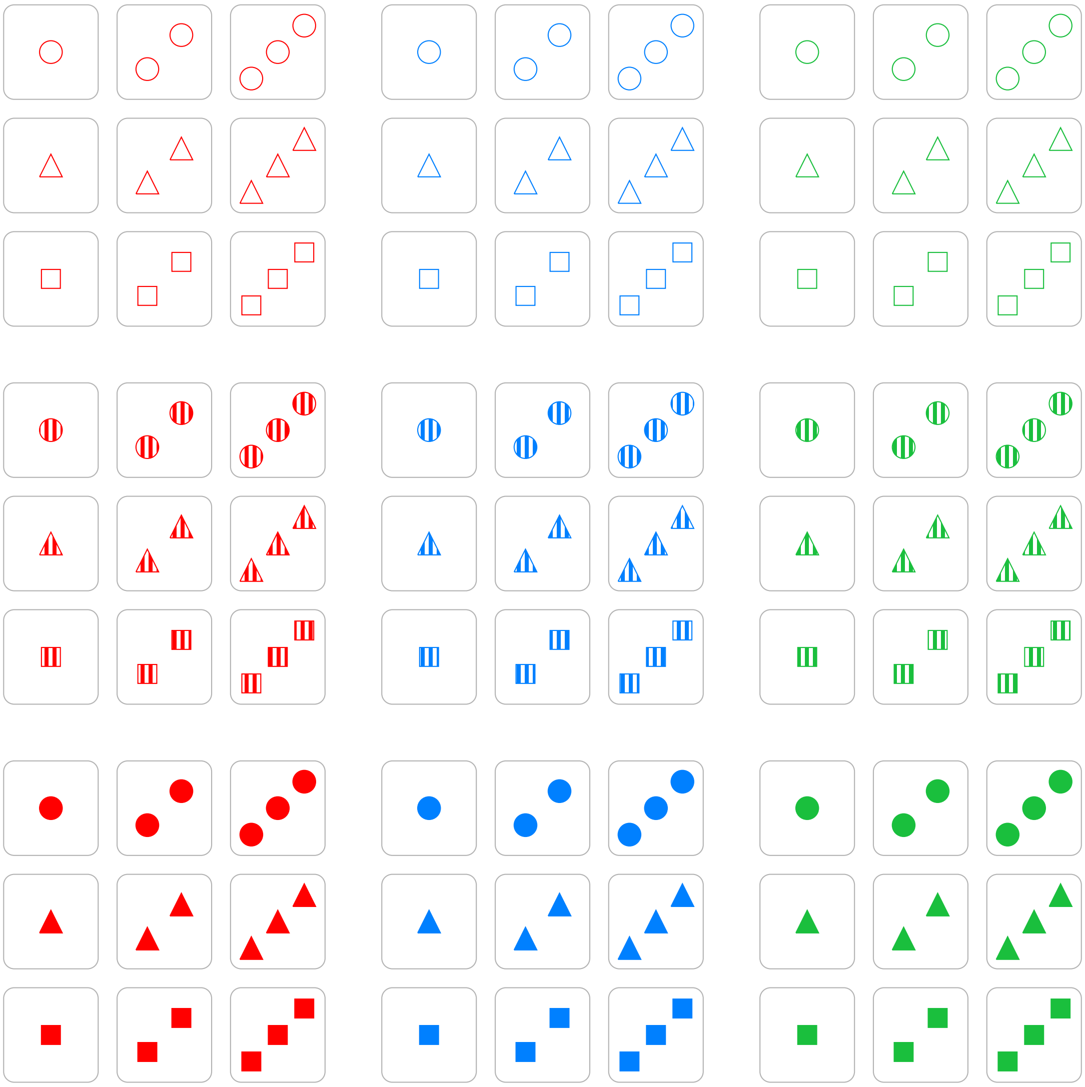
```
\setclassbundle{class}
```



5 Utilities

Here are various ways to use the package. It is particularly useful to display *Class?* classifications (see §5.4). Please look at the source of this document to understand how they have been produced.

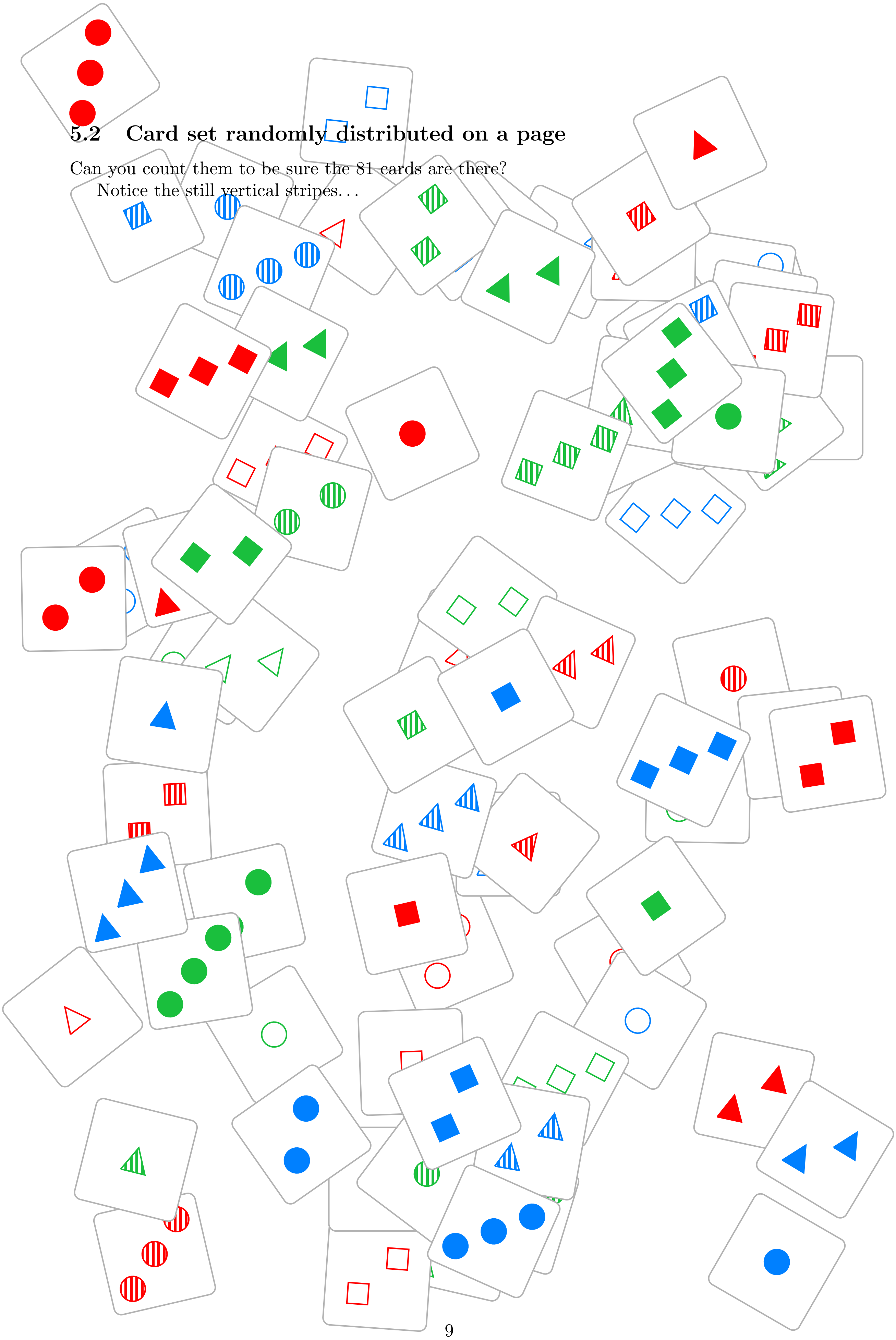
5.1 Card set



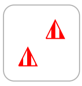
5.2 Card set randomly distributed on a page

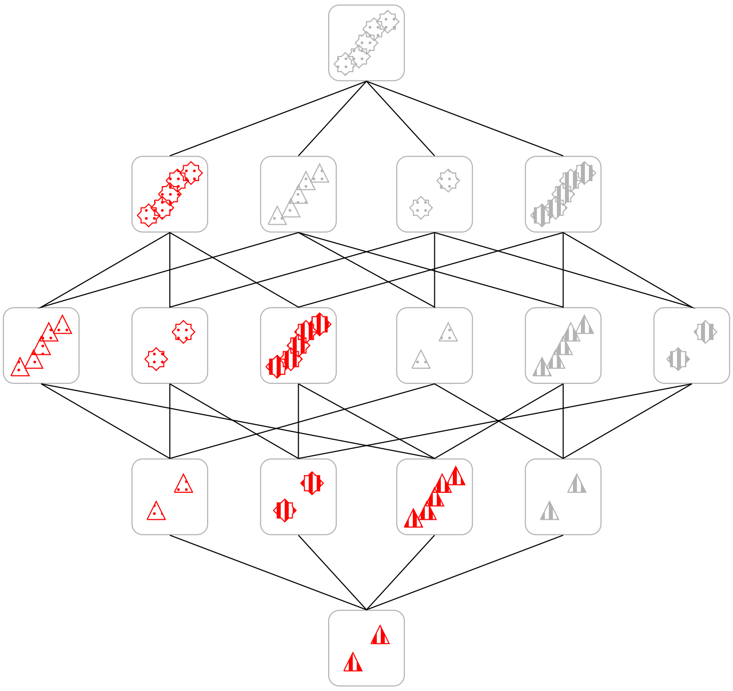
Can you count them to be sure the 81 cards are there?

Notice the still vertical stripes. . .



5.3 Class lattice

This is the lattice of all classes covering the card .

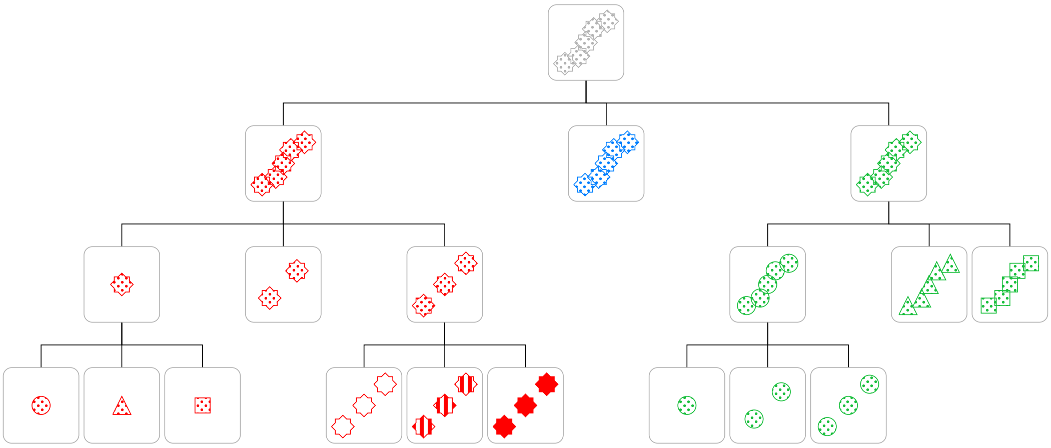


Obtaining the equivalent for all $3^4 - 1 = 80$ other combinations is left as an exercise for the idle reader.

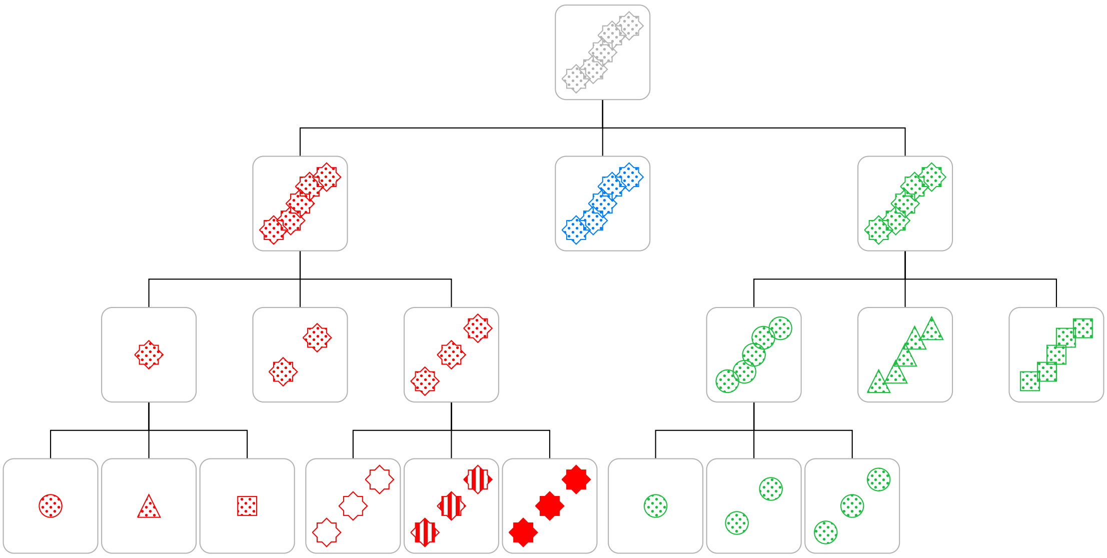
5.4 Trees

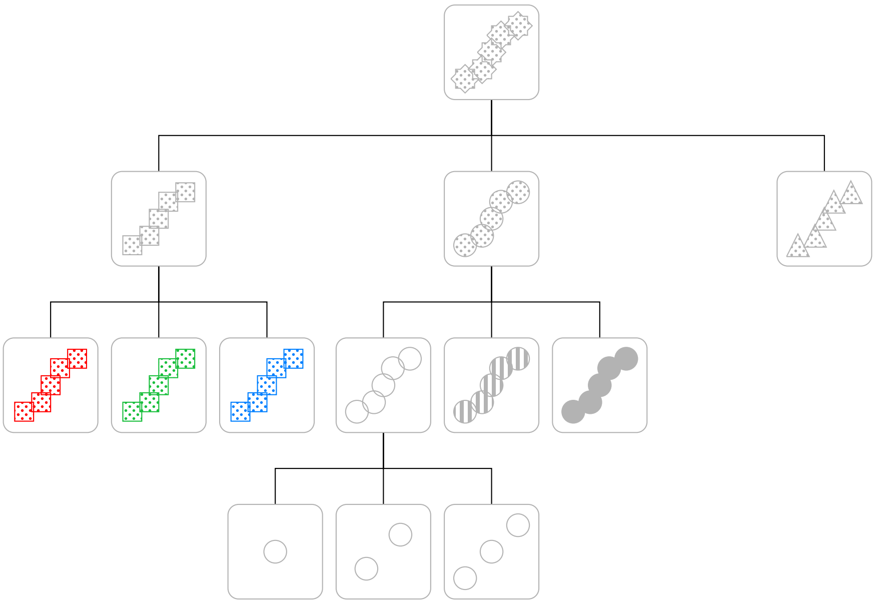
Now we have everything ready, to display board cards, filled or empty.

By using tikZ-qtree and leaving it decide the sibbling distances:



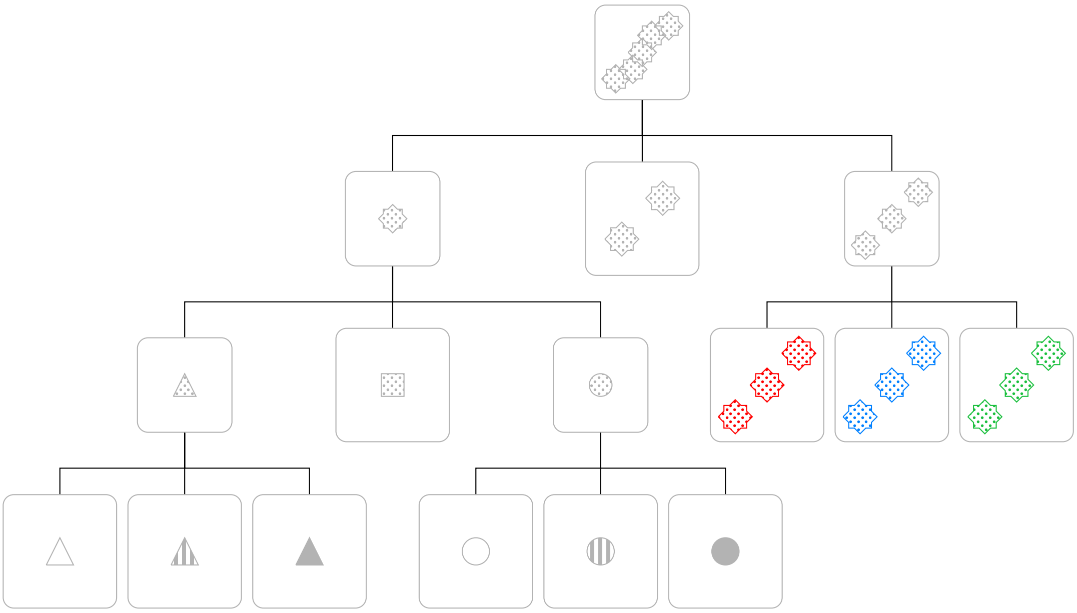
By using tikZ and tuning the sibbling distances:



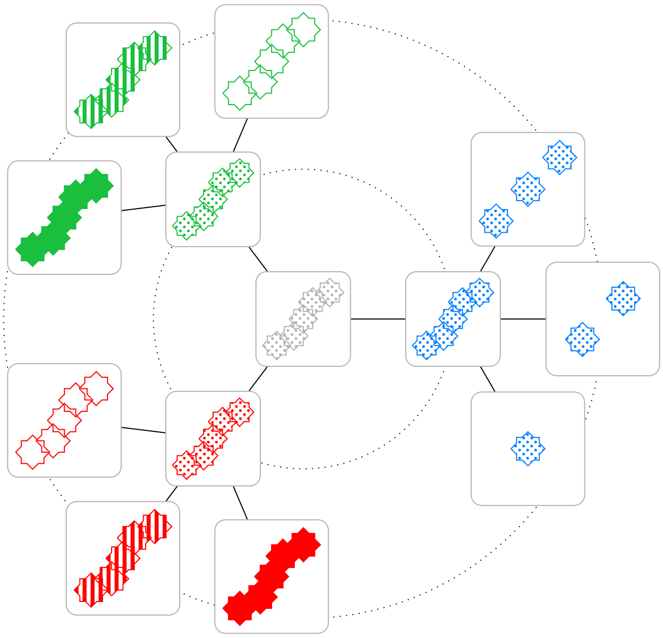


These are those that we use for the game boards:

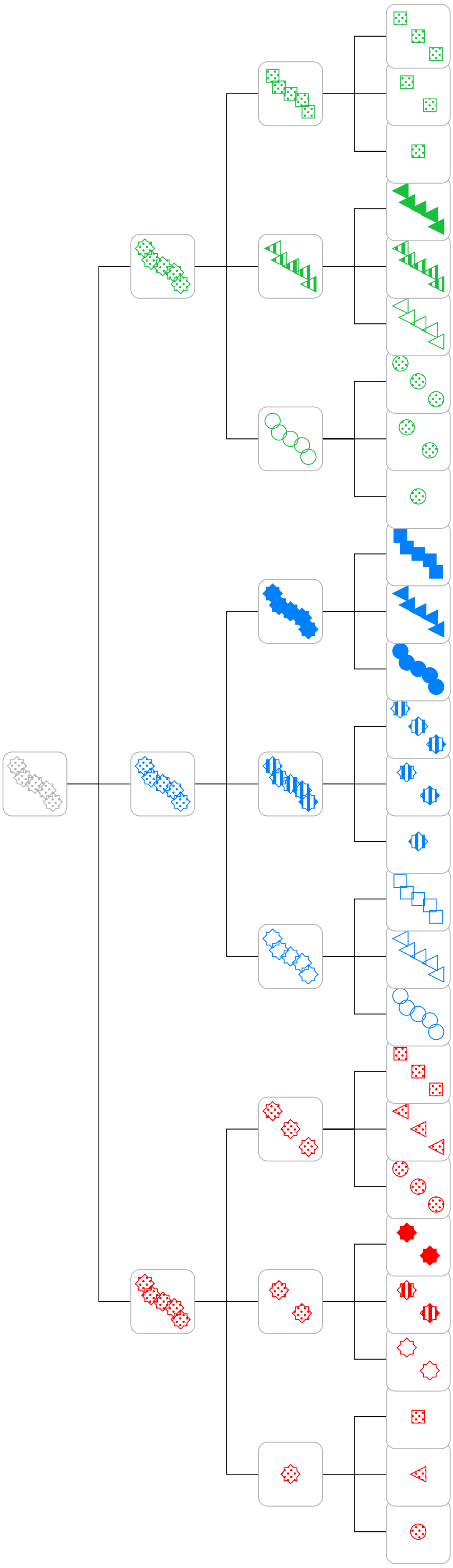
11/3–9

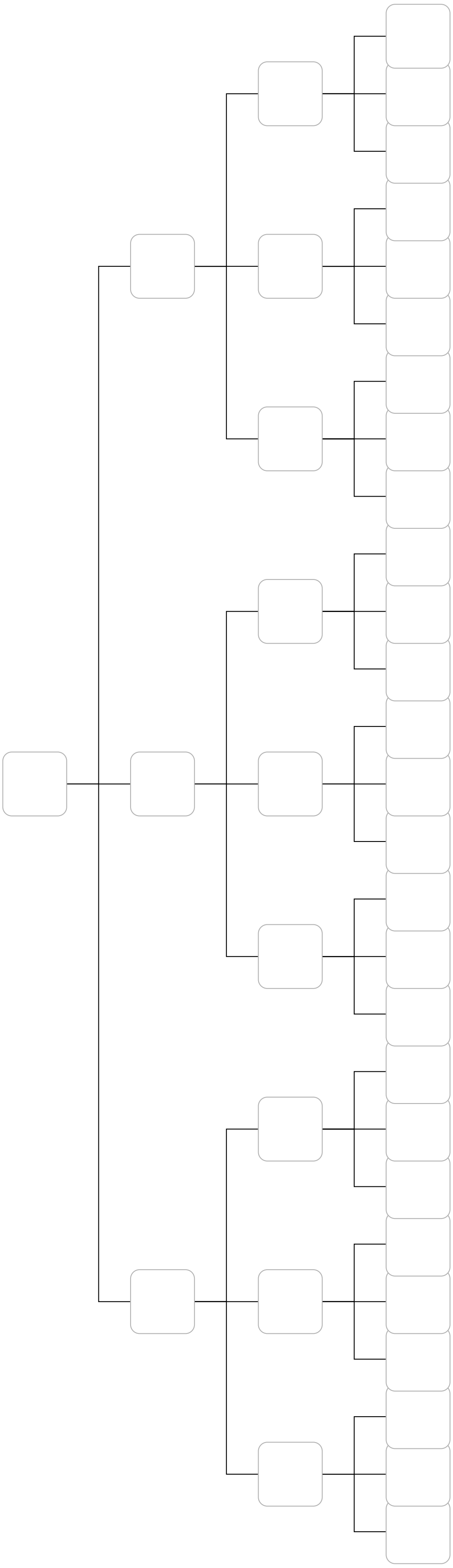


Circular trees are not really more legible:



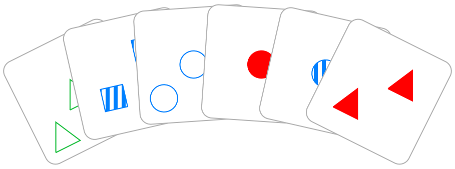
9/2–1





5.5 Miscellaneous

Of course, remain the opportunity to draw a hand of cards as:



6 Known issues

6.1 standalone

When converting TikZ pictures with the `standalone`/`convert` document class, it is necessary to pass it the `preview` option, otherwise the result may be problematically trimmed.