Playing Games With LaTeX

Adelheid Bonnetsmüller, bonnetsmueller@icloud.com

DANTE Frühjahrstagung 2024, Weimar

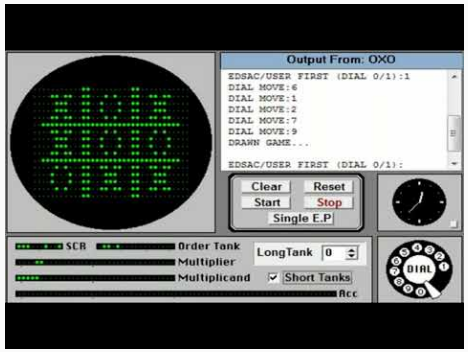
Über das Spielen – Wissenswertes und Historisches

Über das Spielen – Wissenswertes

- erste Spiele beinahe ebenso alt wie die frühesten bildlichen Darstellungen in der Menschheitsgeschichte
- Es gibt einen eigenen Wissenschaftszweig für das Erforschung von Spielen: die Spielewissenschaft
- ein Teilbereich davon beschäftigt sich mit digitalem Spiel: die Ludologie
- Computerspiele:
  - 1947 Cathode-ray tube amusement device (Videospiel)
  - 1951: Nim (eigener Computer Nimrod)
  - 1952: OXO (graphisches Spiel)
  - 1958: Tennis for Two (Analogcomputer und Oszilloskop)
  - 1962: Spacewar! (Two-Player-Game)
  - 1972: Pong (Atari, erstes weltweit beliebtes Spiel)

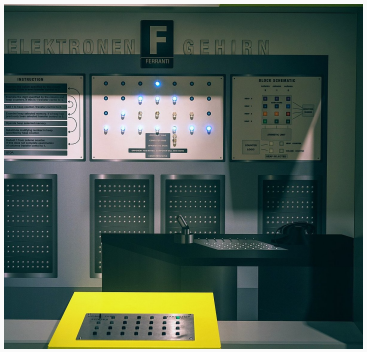
Über das Spielen – Wissenswertes

Cathode Ray Amusement Device

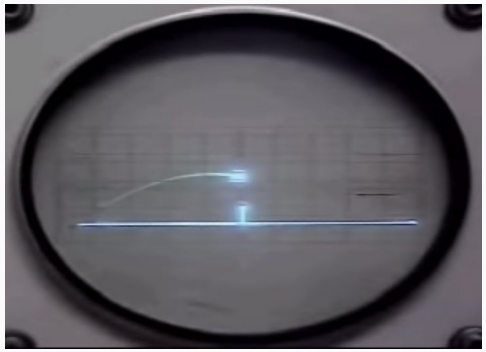


OXO

Spacewars



Nimrod



Tennisfortwo

Pong

Über das Spielen – LaTeX

Über das Spielen – LaTeX

- für Spiele braucht es Anleitungen
- ...oder Spielzüge sollen dargestellt werden
- ...oder Rätsel und Herausforderungen erstellt werden
- für all das gibt es viele LaTeX-Pakete, mit denen das möglich ist.

Über das Spielen – LaTeX

- auf CTAN finden sich aktuell 93 Pakete, mit denen allerlei Rätsel und Spiele gesetzt werden können.
- davon gibt es 19 Pakete, die sich mit Schach,
  6 die sich mit GO,
  11 die sich mit Bridge, Rollenspielen oder Kreuzworträtsel beschäftigen
- und 14 Pakete, die anderes zum Inhalt haben (z.B. `thirteen`)
- das sind ungefähr 50%.
- wir beschäftigen uns mit den anderen 50% (zumindest teilweise)

LaTeX – Spieletypen

- Brettspiele
- Kartenspiele
- Logikspiele
- Computerspiele
- Spieltheorie
- Sonstiges
- Schach
- Go
- Bridge
- Pen & Paper-Rollenspiele
- Kreuzworträtsel

LaTeX – Brettspiele

Für Brettspiele gibt es folgende Pakete

- `psbao` Bao-Diagramme zeichnen (afrikanisches Spiel)
- **Scrabble** (Spielbrett und Spielzüge)
- **TrivialPursuit** (Spielbretter)
- `bg` (Backgammon, Spielbrett und -züge)
- `reverxii` Reversi in PlainTeX
- `othello` und `othelloboard` Spielbrett und -züge
- **wargame** Strategie-Kriegsspiele mit hexagonalem Aufbau setzen
- **hexboard und hexgame** Spielbrett und -züge
- `havannah` Spielbrett und -züge
- `mahjong` Spielsteine
- `context-games` Go und Hex für ConTeXt (obsolet)

Für Kartenspiele gibt es folgende Pakete

- `AcroMemory` Memoryspiel
- `hmtrump` Spielkartensymbole (Doku auf japanisch!)
- **JeuxCartes** Verschiedene Kartenspiele
- `playcards` Karten mit Ziffern selber entwerfen
- `pst-poker` Pokerkarten (PSTricks)
- `setdeck` Kartendecks gestalten

Für Logikspiele gibt es folgende Pakete

- `sudoku` und `sudokubundle` für das Setzen von Sudoku
- `context-sudoku` Sudoku für ConTeXt
- **logicpuzzle** sehr umfangreiches Paket zum Setzen gitterbasierter Logikpuzzle
- **labyrinth und maze** Zum Setzen von Labyrinthen
- **soup** Zum Setzen von Wortsuchrätseln

LaTeX – Computerspiele und Spieletheorie

Für Computerspiele gibt es folgendes Paket

- `puyotikz` PuyoPuyo-Spiel setzen
- **wordle** Wordle-Spiel

Für das Setzen von spieltheoretischen Anwendungen gibt es folgende Pakete:

- `egamaps`
- `istgame`
- `sgame`
- `twoxtowgame`

Weitere Spielepakete

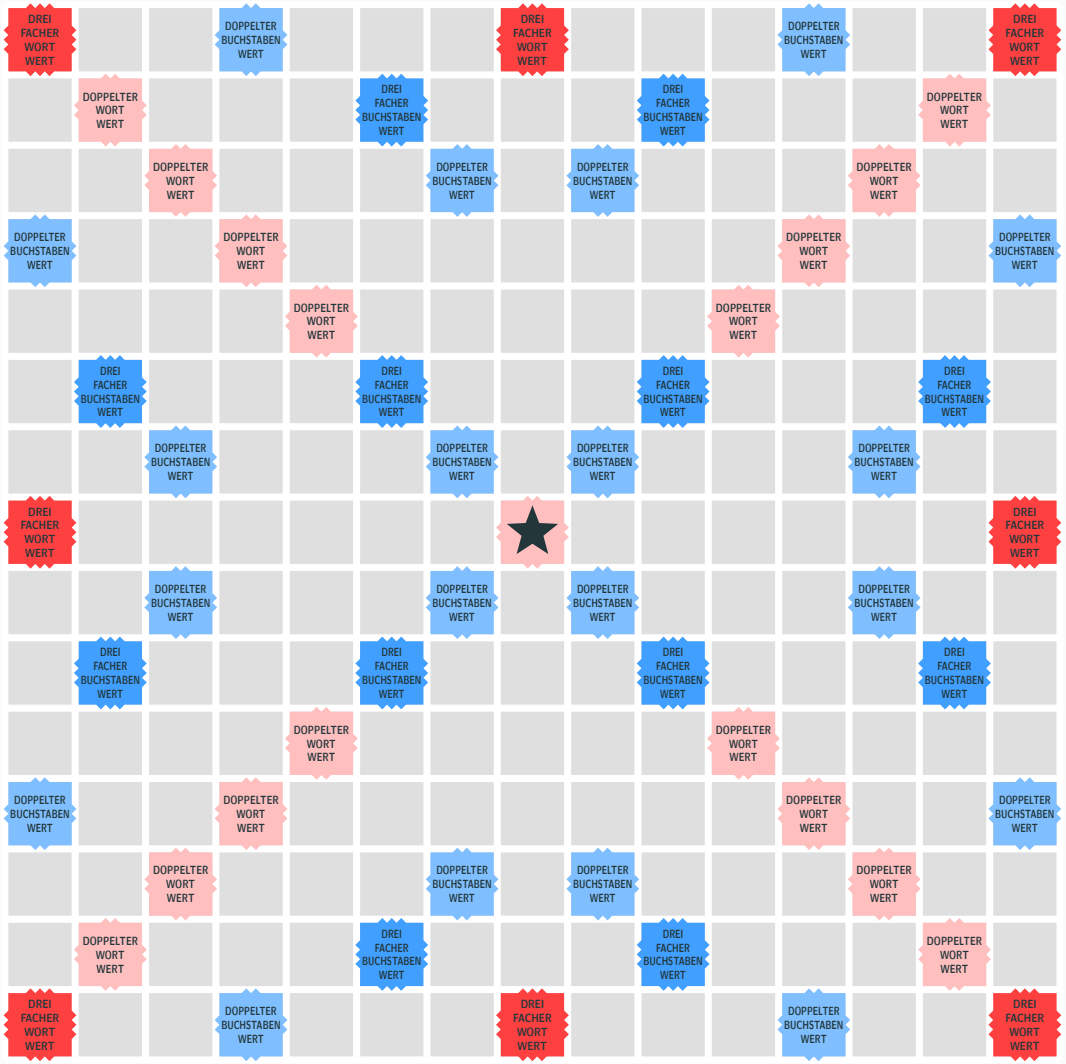
- **TangramTikZ** Tangram-Figuren zeichnen
- `pst-dart` Dartscheiben mit und ohne Pfeile zeichnen (PSTricks)
- **battleship** Schiffe versenken: Spielfeld und -züge
- `hanoi` Türme von Hanoi, PlanTEX
- `I-Ching` und `iching` Zum Setzen des Orakel„spiels“I-Ching
- `Jeopardy` und `jj-game` Für das Setzen von Jeopardy-Spielen
- **nimsticks** Zum Setzen von NIM
- `RoueQuestions` Eine Fragenrad (*Question Wheel*) setzen

Auf den Brettern, die die Welt bedeuten

Scrabble - Allgemein

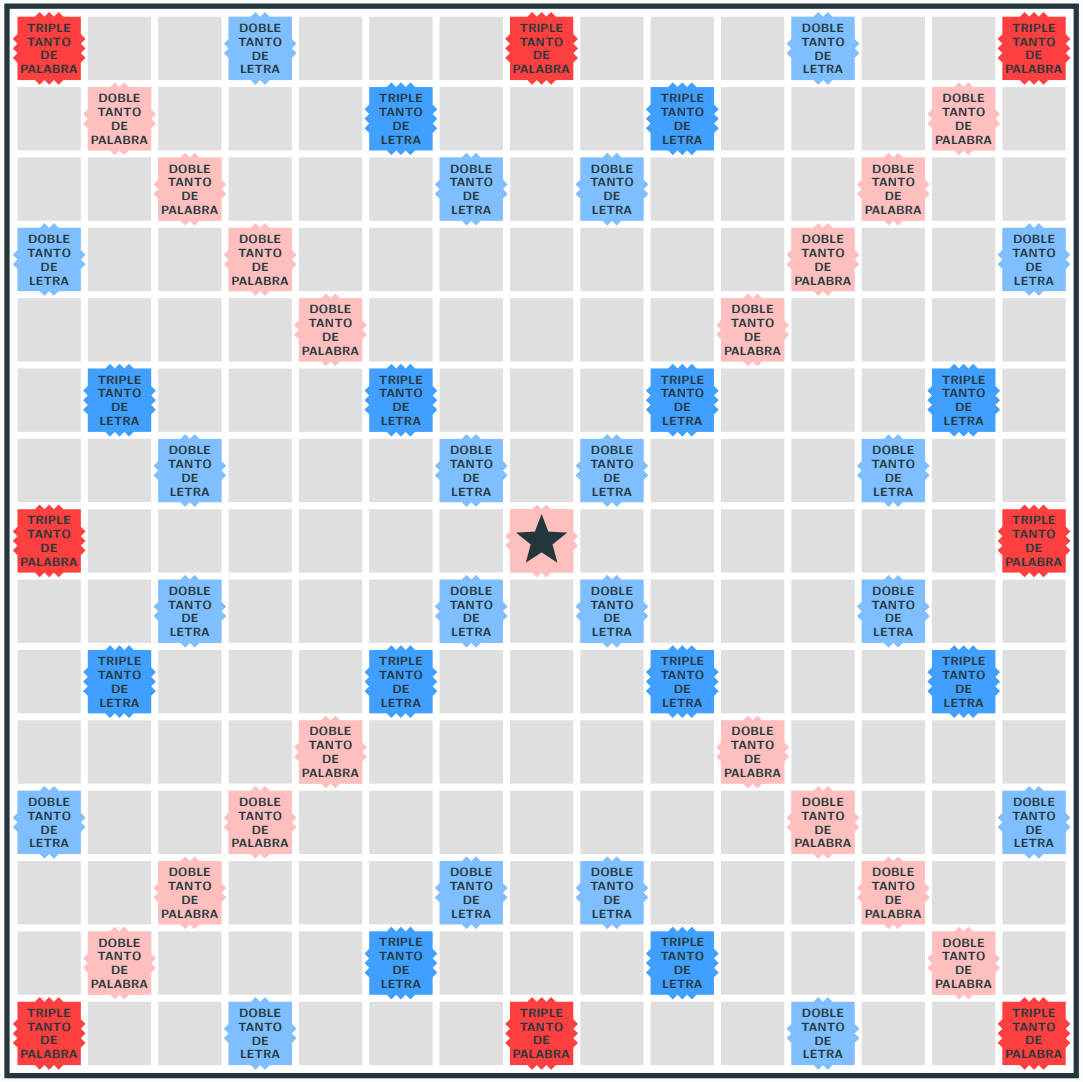
- Autor: Cédric Pierquet, 2023
- Doku auf Englisch und Französisch
- Einbindung über `\usepackage{Scrabble}`
- Abhängigkeiten:
  `tikz, pgf, pgffor, xstring, xparse, simplekv, listofitems`
- Spielbrett in mehreren Sprachen möglich: englisch (EN, default), französisch (FR), deutsch (DE) und spanisch (ES)
- Spielbrett wird über `\ScrabbleBoard<Sprache>[Optionen]` erzeugt.
- Optionen sind
  `Scale=<num (def. 1)>, Border=<true|false>, Help=<false|true>, Labels=<true|false>, ScaleLabels=<num (def. 1)>`

Scrabble - Spielbrett



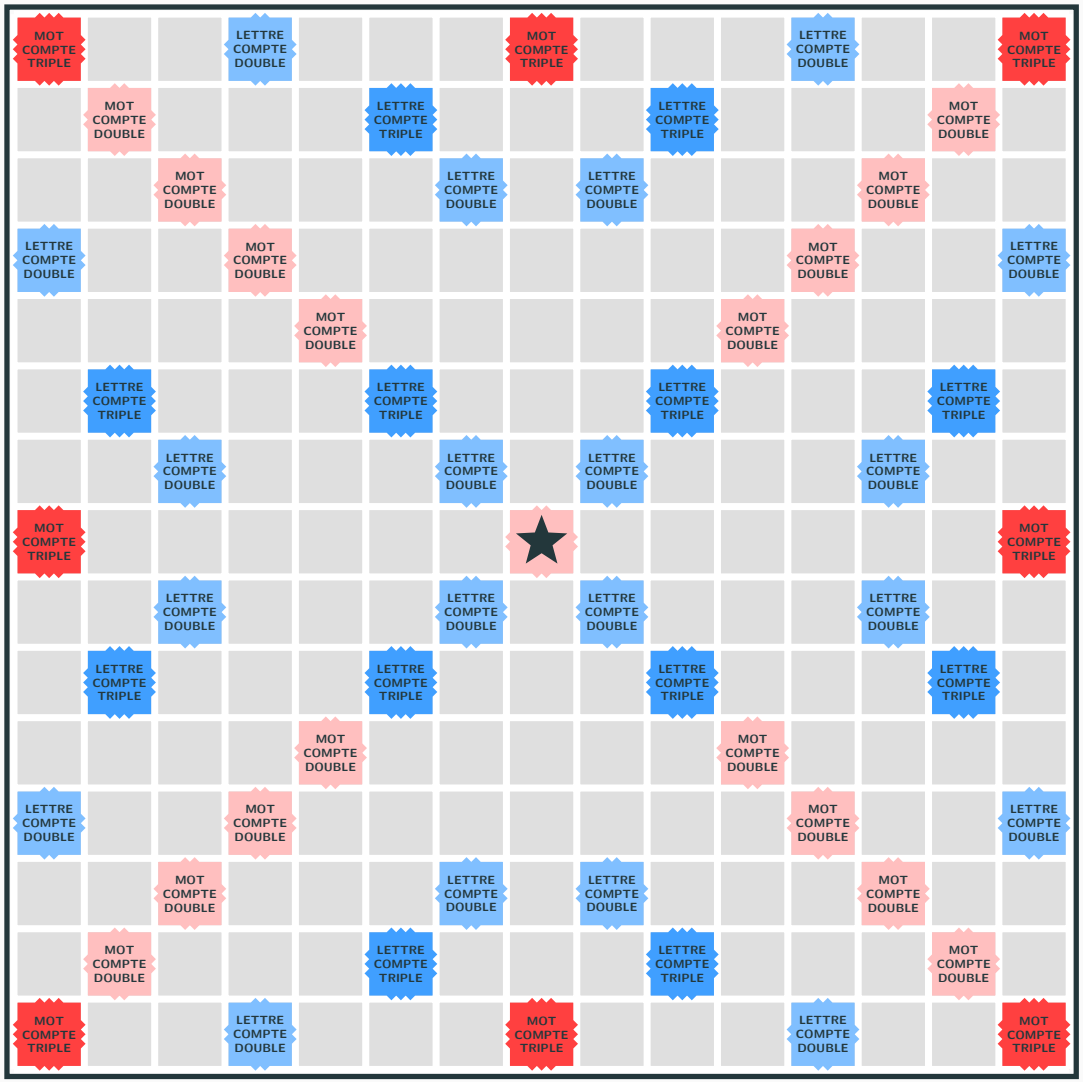

```
\ScrabbleBoard<DE>[Scale=0.45,Border=false]
```

Scrabble - Spielbrett



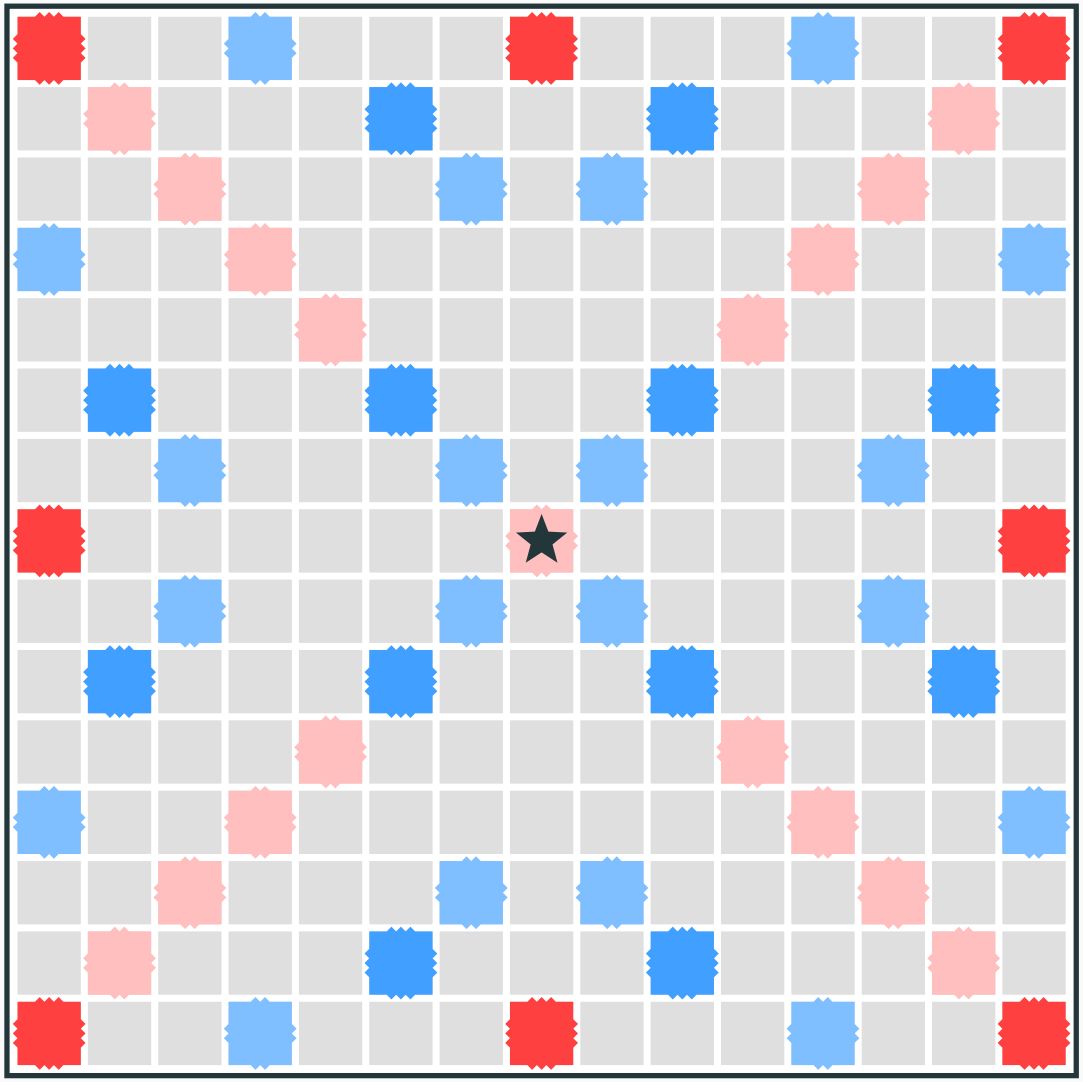

```
\ScrabbleBoard<ES>[Labels=true,Scale=0.45,Border=true]
```

Scrabble - Spielbrett



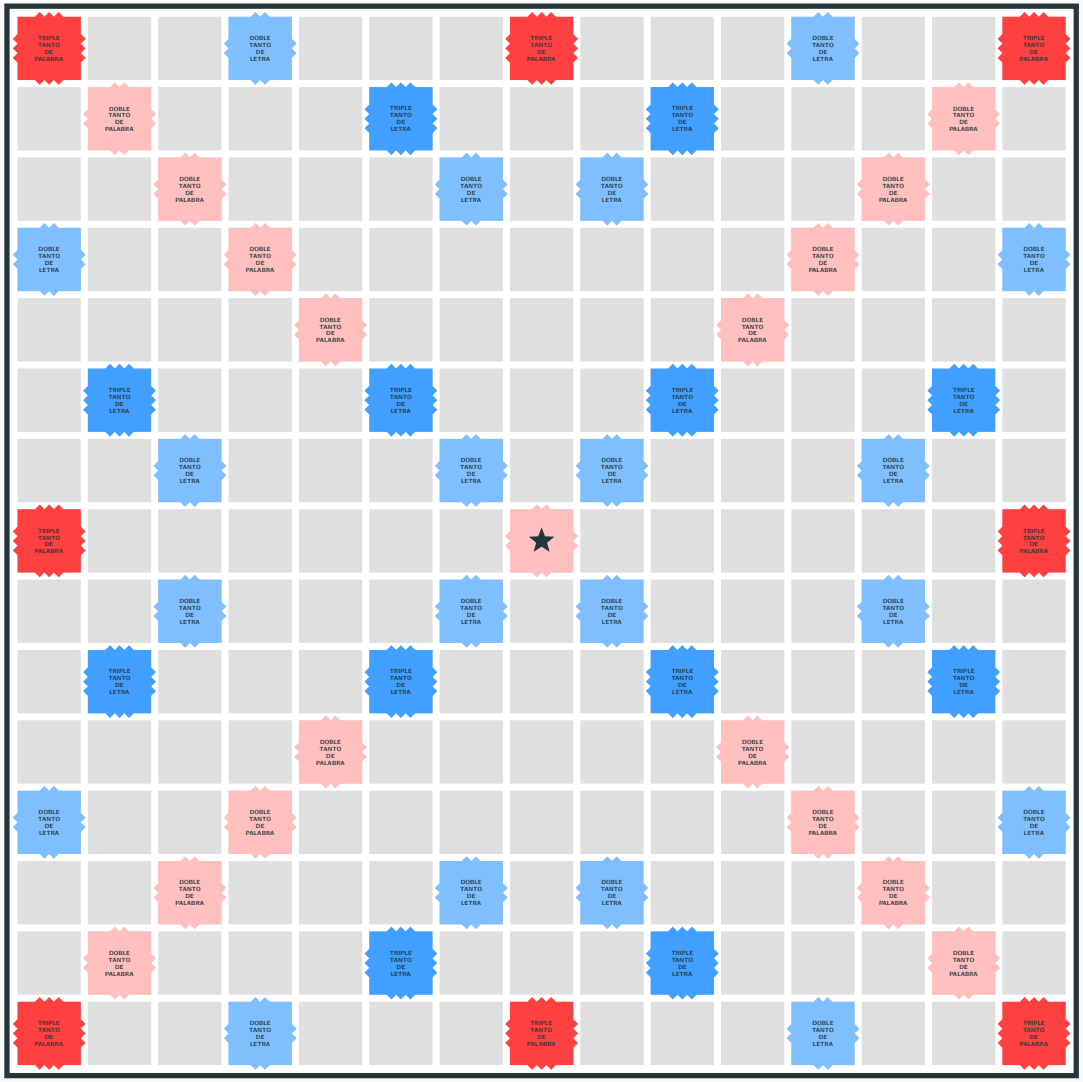

```
\ScrabbleBoard<FR>[Help=true,Scale=0.45]
```

Scrabble - Spielbrett



```
\ScrabbleBoard<ES>[Labels=false,Scale=0.45,Border=true]
```

Scrabble - Spielbrett




```
\ScrabbleBoard<ES>[ScaleLabels=0.5,Scale=0.45]
```

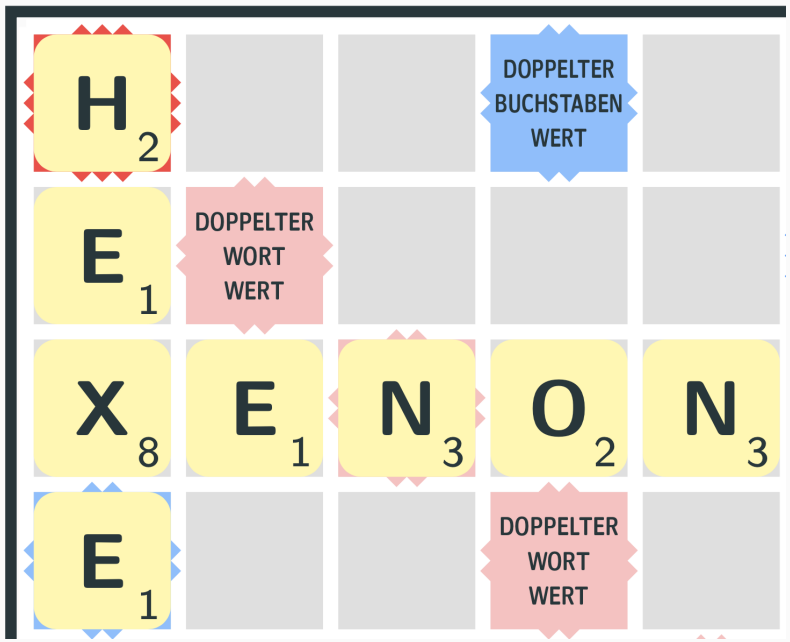
`Scrabble` - paar Wörter dazu

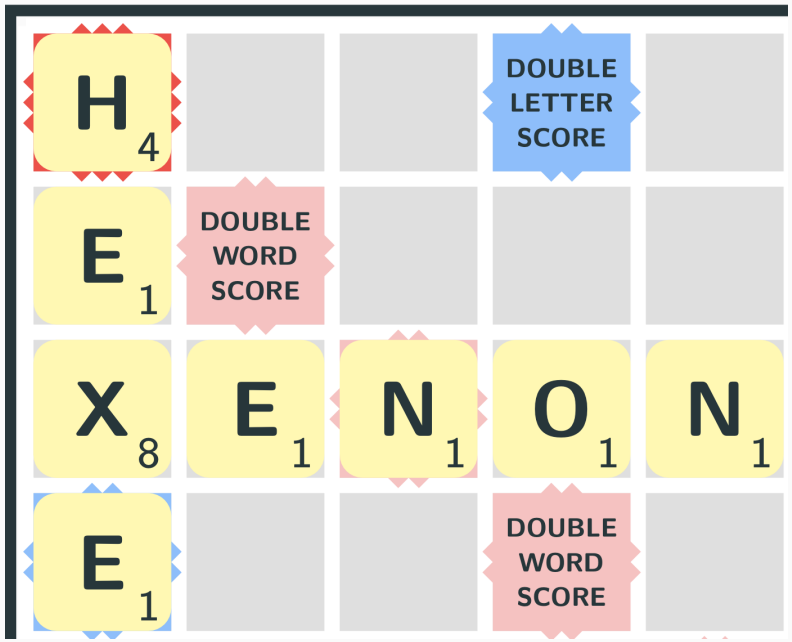
- Ein mit Wörtern gefülltes Brett wird in der Umgebung `begin{EnvScrabble}...\end{EnvScrabble}` gesetzt
- Dieser Umgebung können die gleichen Optionen wie dem Befehl `\Scrabbleboard` mitgegeben werden
- Mit `\ScrabblePutWord[H|V]{.1.}{.2.}` werden die Wörter platziert .1. Wort .2. (x,y)-Position
- zum Plazieren kann die Option `Help=true` hilfreich sein (Zeilen- und Spaltennummern werden angezeigt)

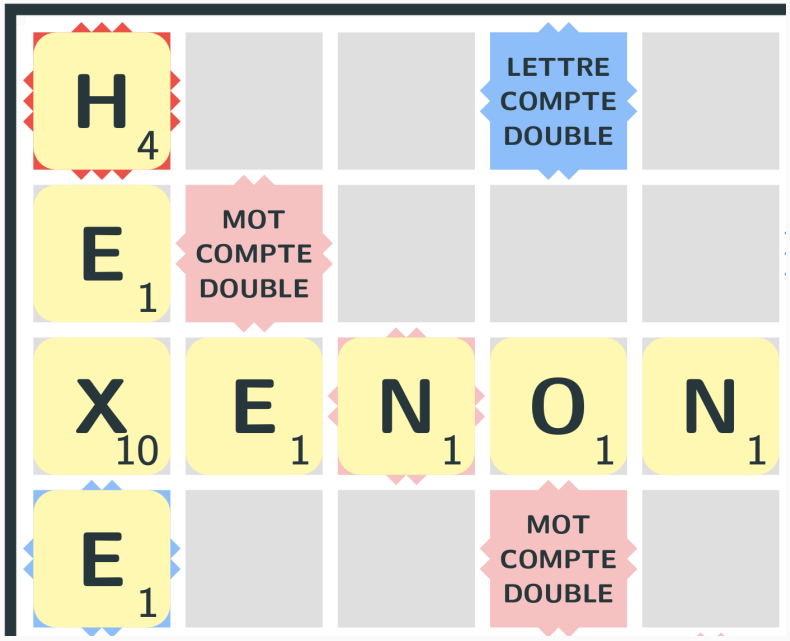
`Scrabble` - paar Wörter dazu

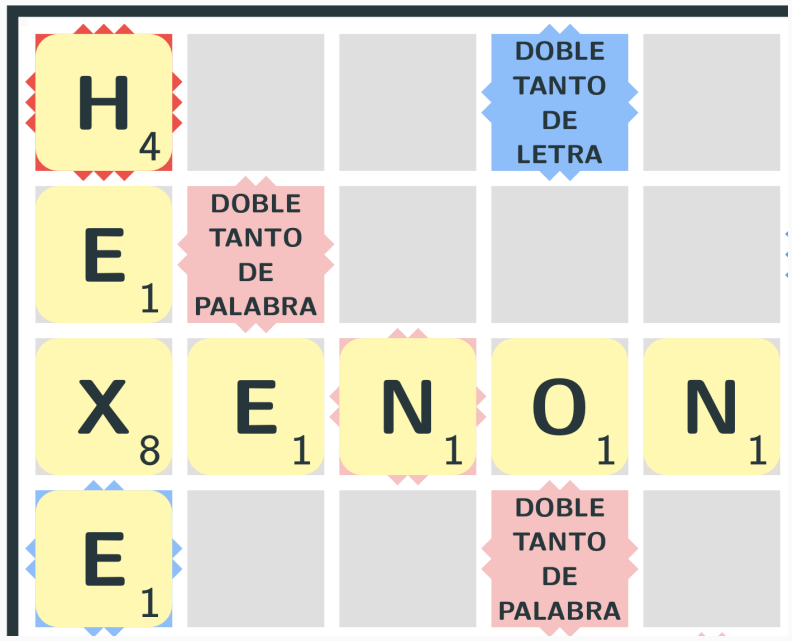
- Beispiel: `\ScrabblePutWord{xenon}{1,13}` setzt das Wort XENON an (x,y)=(1,13)
- die Option H für horizontales Platzieren muss nicht angegeben werden (Standard).
- Vertikales Platzieren geschieht über
  `\ScrabblePutWord[V]{HEXE}{1,15}`
- Die Punkte pro Buchstabe werden automatisch gesetzt (sprachenabhängig)
- Ein Joker wird mit * gesetzt (es erscheint ein leerer Stein).

Scrabble - paar Wörter dazu









Scrabble - paar Wörter dazu; Beispiel

`Trivial Pursuit` - Allgemein

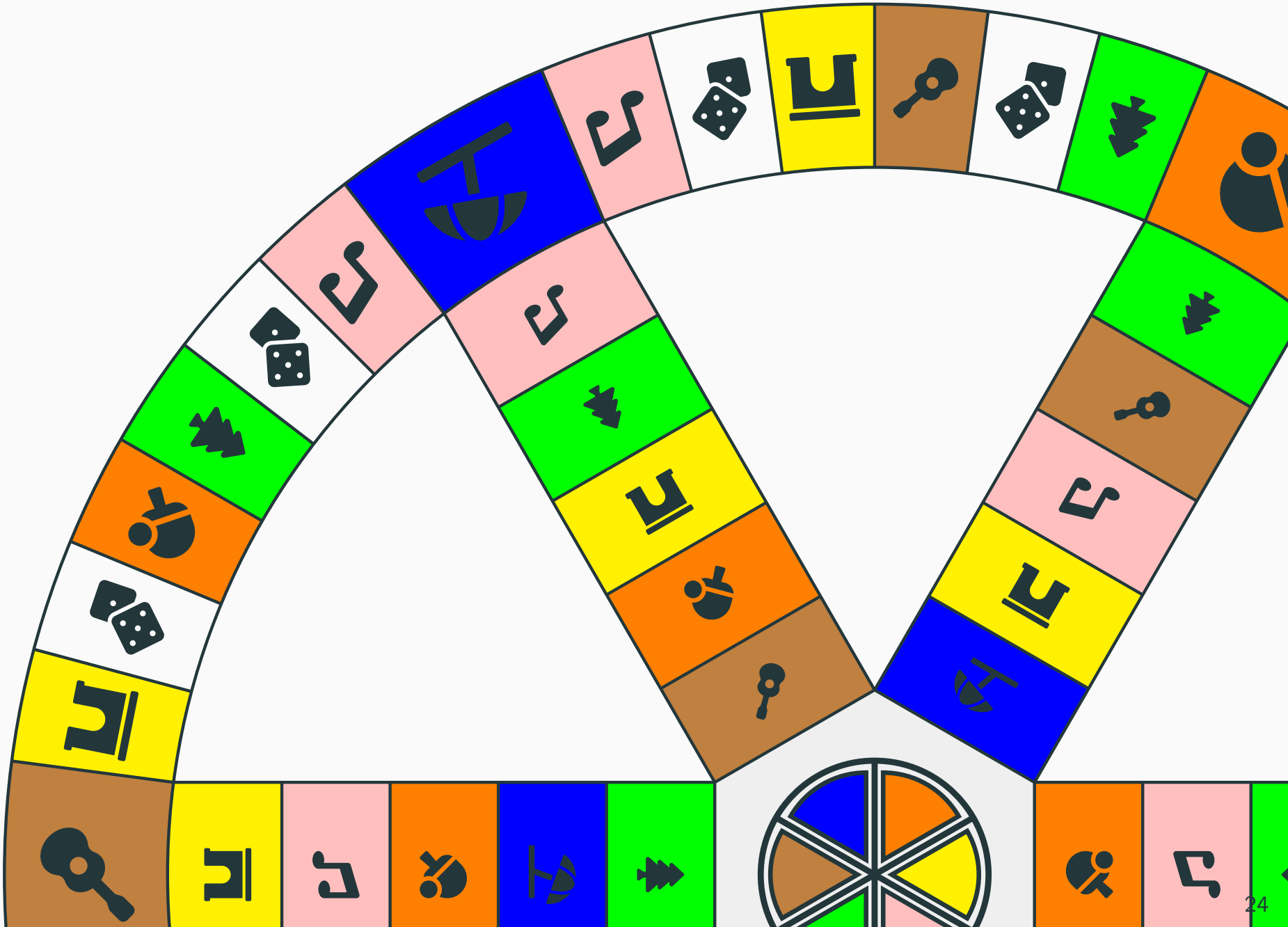
- Autor: Cédric Pierquet, 2023
- ermöglicht das Erzeugen von Spielbrettern für Trivial Pursuit
- es können Farben, Logos (Symbole) und der Radius bzw. die Länge der Zellen angepasst werden.
- aktuell beschränkt auf 6 Kategorien und die Symbole aus dem Paket `fontawesome5`
- Abhängigkeiten:
  `tikz, calc, positioning, calc, fontawesome5, simplekv, xintexpr, listofitems`
- Einbinden über `\usepackage{TrivialPursuit}`

`\BoardTrivialPursuit[.Optionen.]` erzeugt Spielbrett mit den Optionen:

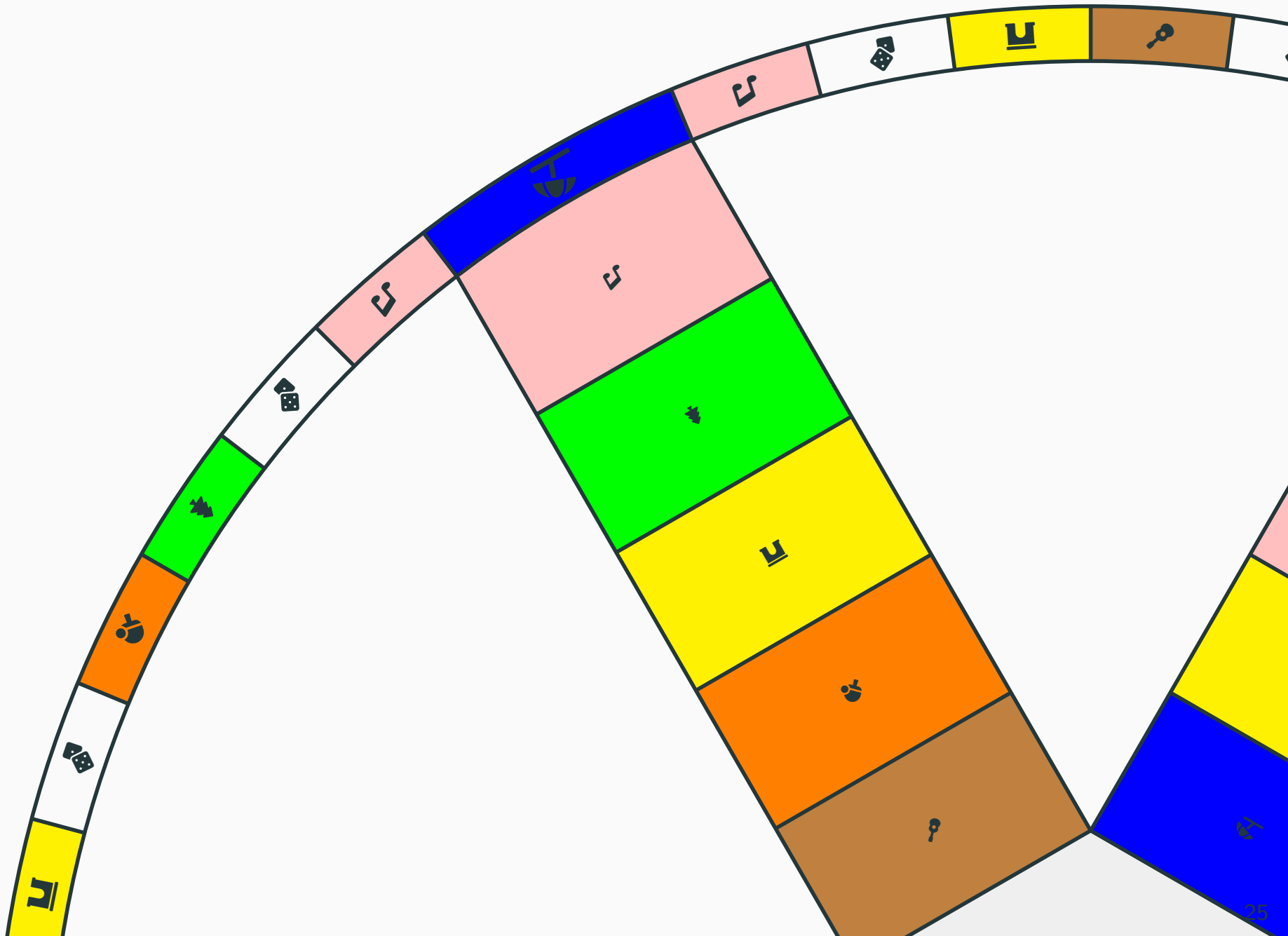
Option	**Default**	**Wert**	**Bedeutung**
`Unit`	1	`<num>`	Größe des Brettes
`Radius`	8	`<num>`	Radius
`BorderHeight`	1.5	`num`	Breite des Randes
`ListColors`			Farben der Kategorien
`ListIcons`			Welche Symbole verwendet v
`Icons`	true	`true|false`	Ausgabe Symbole
`Center`	true	`true|false`	Mitte farbig

Option	Default	Wert	Bedeutung
ColorCenter			Farbe der Mitte
Colors	true	true|false	farbig oder schwarz-weiß
Logo	true	true|false	Ausgabe Logo
Jokers	true	true|false	Ausgabe Jokers
IconJoker	\fadice		Symbol für Joker
Blank	false	true|false	leeres Spielbrett
Thickness	0.8pt	<dim>	Linienstärke
Rotation>	0	<dim>	Drehung

Trivial Pursuit - Ausgabe

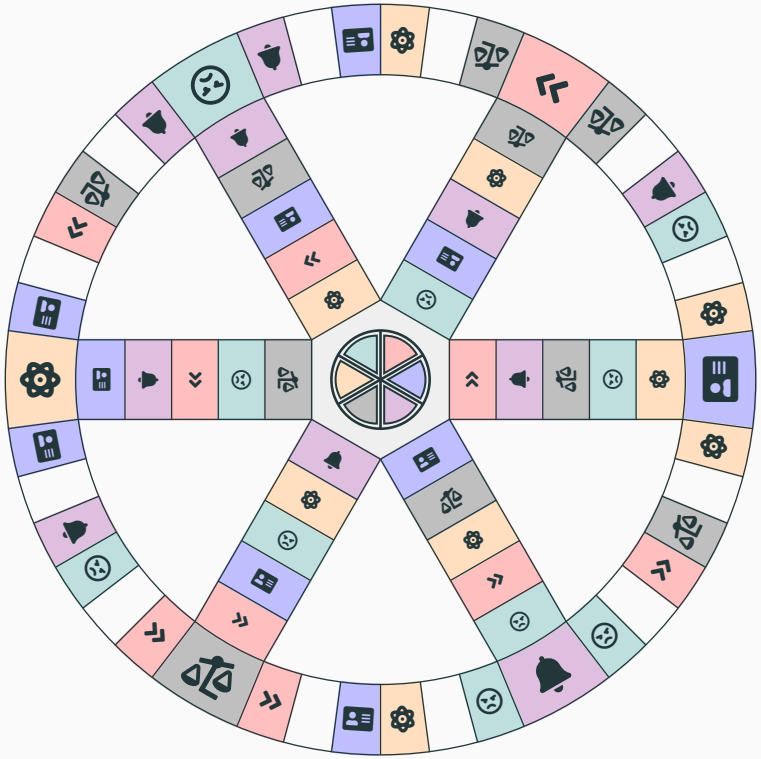


Trivial Pursuit - BorderHeight=0.5 und Radius=10



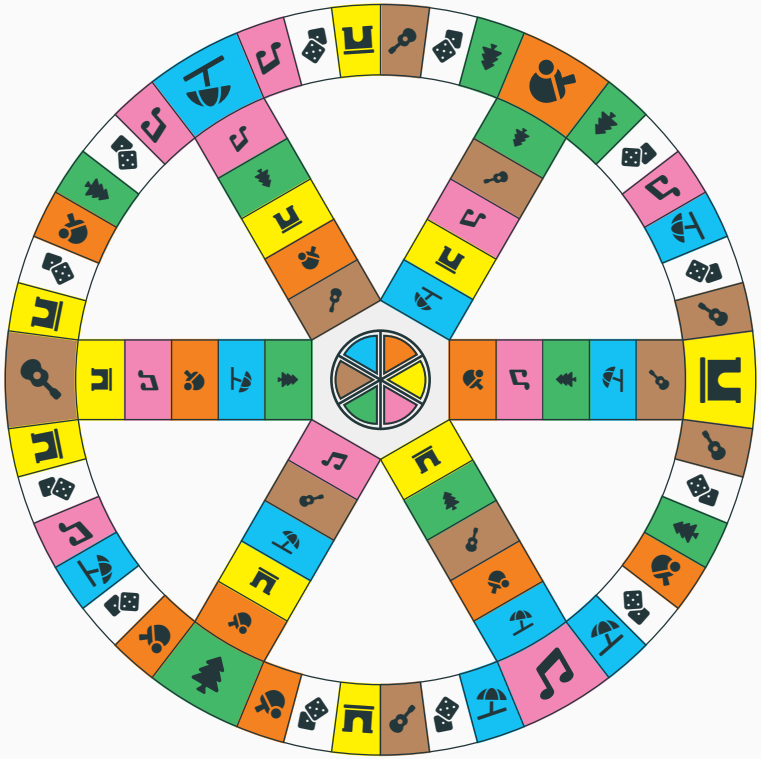
Trivial Pursuit - Farben und Symbole

Mit ListColors und ListIcons können die Kategoriesymbole und -farben sehr frei gesetzt werden.



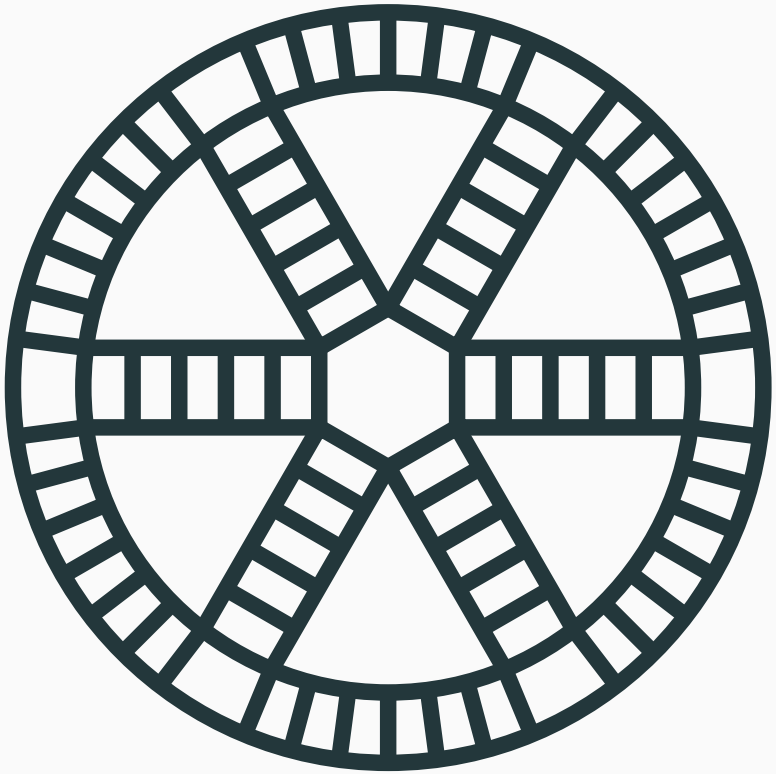
```
\BoardTrivialPursuit[Unit=0.3,
Jokers=false,
ListColors={blue!25,red!25,teal!25,
orange!25,gray!50,violet!25},
ListIcons={ \faAddressCard,
\faAngleDoubleRight,\faAngry[regular],
\faAtom,\faBalanceScaleLeft,\faBell}]
```

Zwei Farbbereiche sind vordefiniert (angelehnt an die Originalfarben): `\TPColorsA` und `\TPColorsB`.



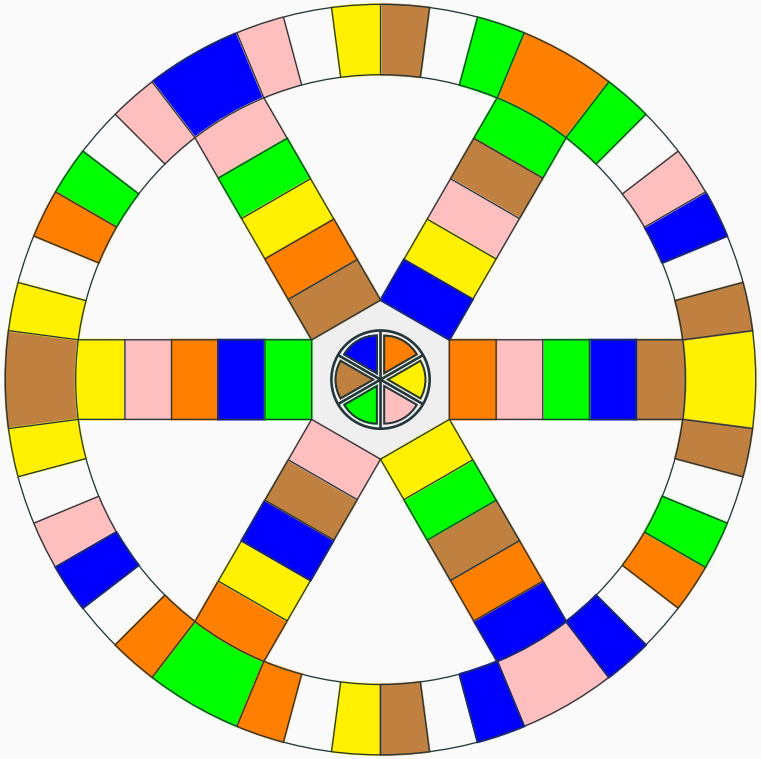
Trivial Pursuit - Linien, Drehungen, Logo

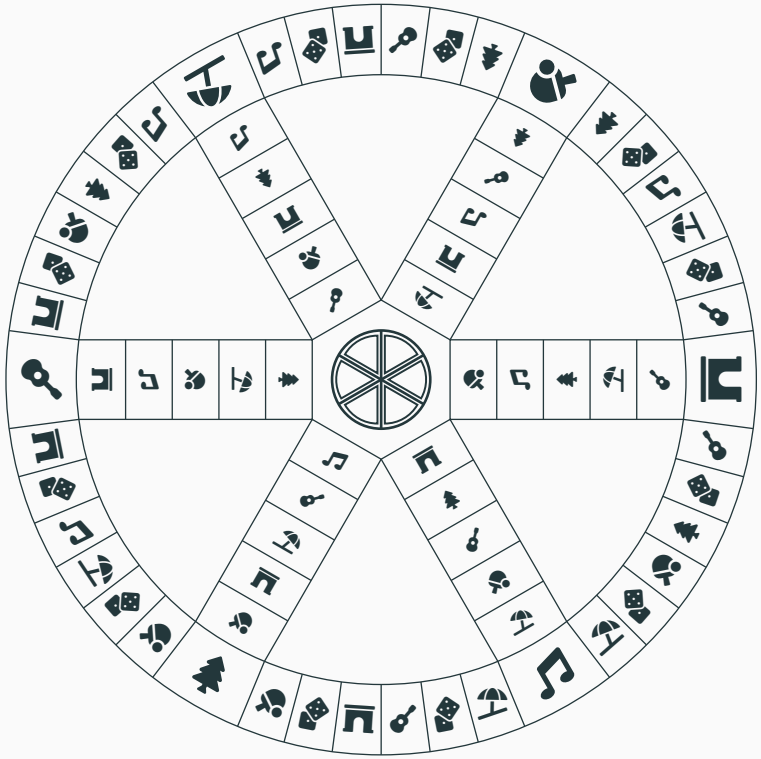
```
\BoardTrivialPursuit[Unit=0.3,Rotation=35,Logo=false]
\BoardTrivialPursuit[Unit=0.3,Thickness=2pt,Blank=true]
```



Trivial Pursuit - schwarz/weiß und Symbole

```
\BoardTrivialPursuit[Unit=0.3,Icons=false]
\BoardTrivialPursuit[Unit=0.3,Colors=false]
```



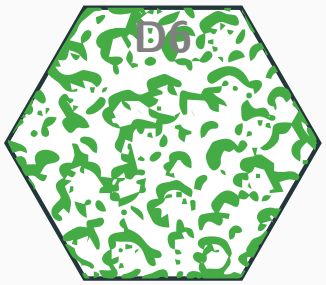


`Wargame` - Allgemein

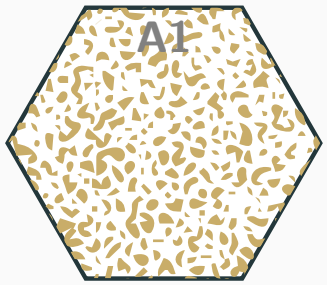
- Autor: Christian Holm Christensen, 2024
- ermöglicht verschiedene teilweise sehr komplexe hexagonale Elemente für Anleitungen, Bretter usw. für Strategie-/Kriegsspiele
- `\usepackage{wargame}`
- ausführliches Tutorial ist auf CTAN hinterlegt
- basiert auf TikZ
- Teile werden über `\hex[.1.](.2.)(.3.)` erzeugt (innerhalb `tikzpicture`-Umgebung
  - .1. terrain, ridges, towns, labels, extra
  - .2. location
  - .3. name
- Es stehen sehr viele Darstellungsmöglichkeiten für Einheiten (Personen oder Gerät) zur Verfügung. Die Symbole entsprechen tatsächlich verwendeter Symbolik.

Wargame - Karte

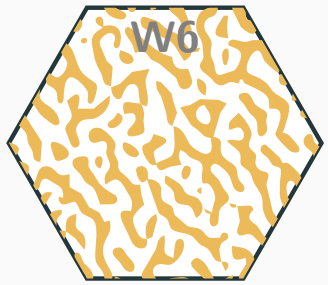
```
\begin{tikzpicture}
\hex[terrain=woods](c=4,r=6);
\end{tikzpicture}
```



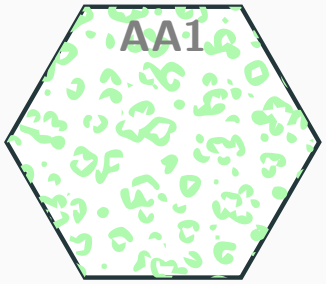
terrain=woods,c=4,r=6



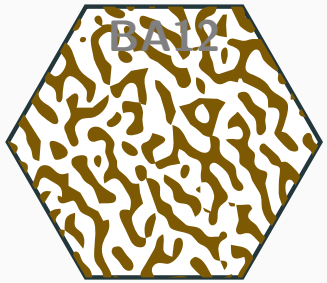
terrain=rough,c=1,r=1



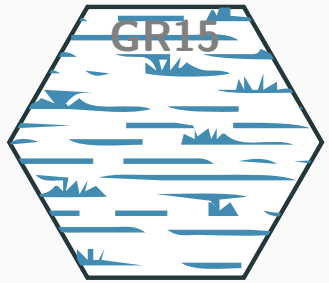
terrain=beach, c=23,r=6



terrain=light woods, c=27,r=1

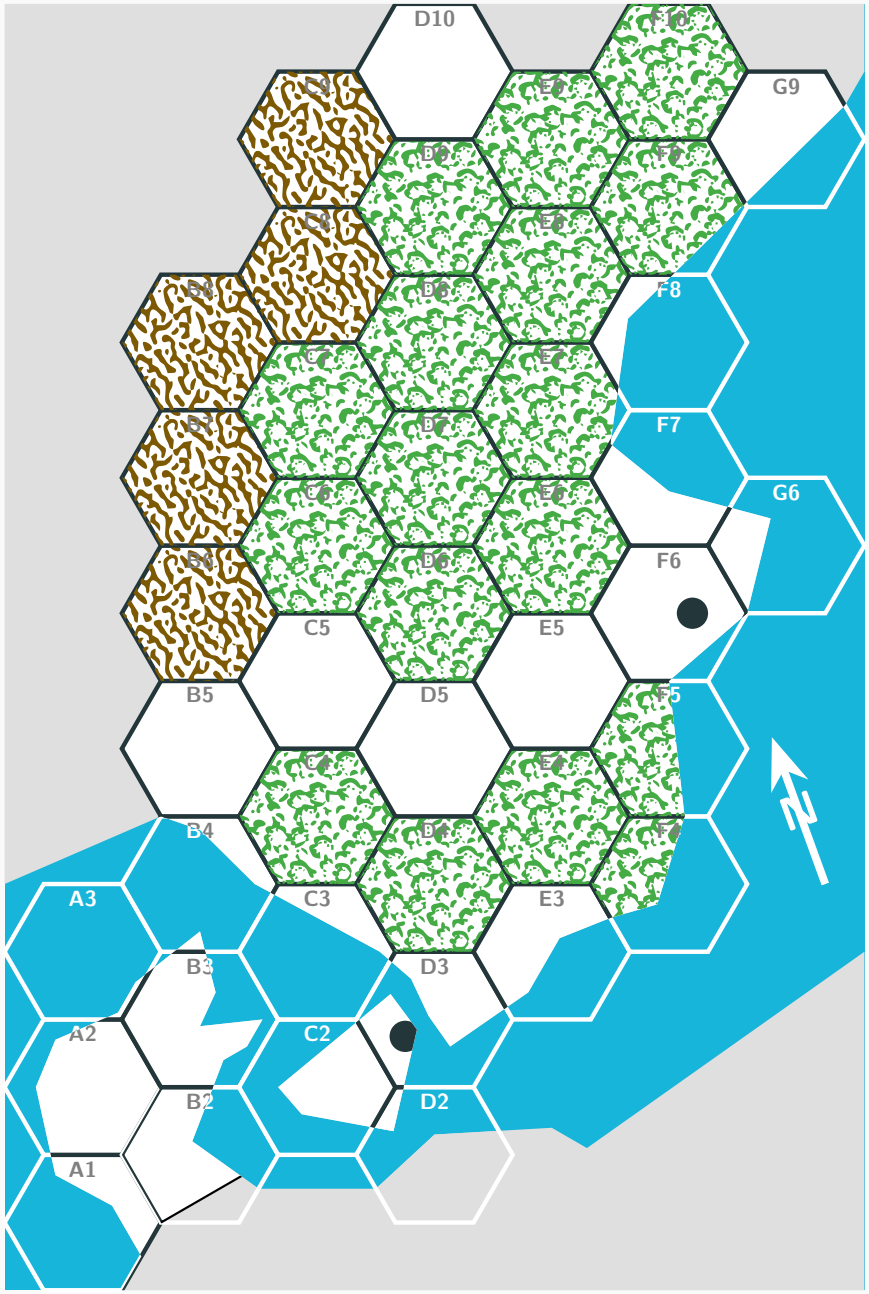


terrain=mountains, c=53,r=12

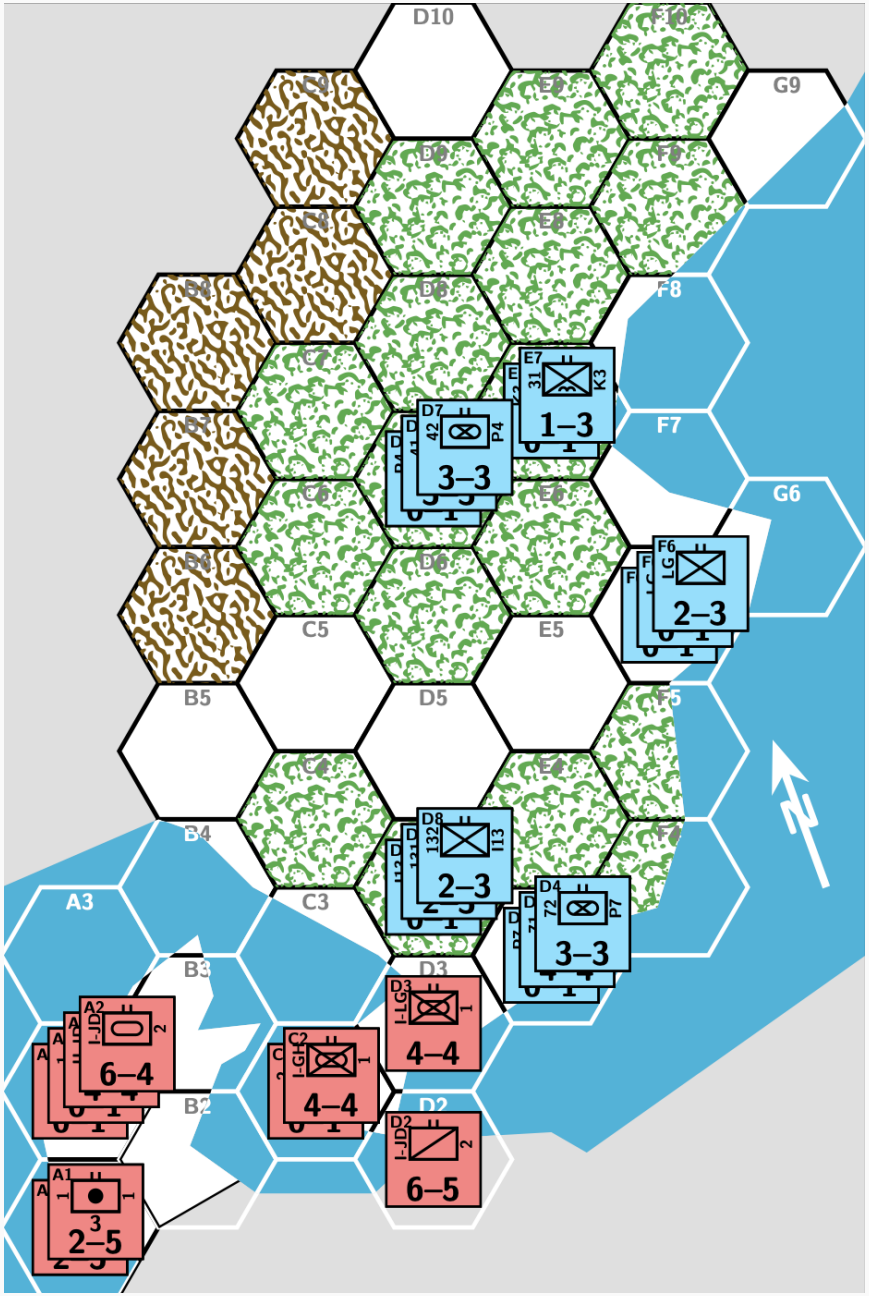


terrain=swamp, c=200,r=15

Wargame - Karte



Wargame - Karte mit Einheiten



`hexgame` und `hexboard` - Allgemein

Zum Spiel:

- Hex ist ein Spiel für zwei Spieler
- Spielziel: man muss einen Weg von der einen blauen zur anderen blauen Seite (bzw. von rot zu rot) setzen
- Rot beginnt. Blau kann danach entweder bei Blau bleiben oder Rot übernehmen.

Zu den Paketen:

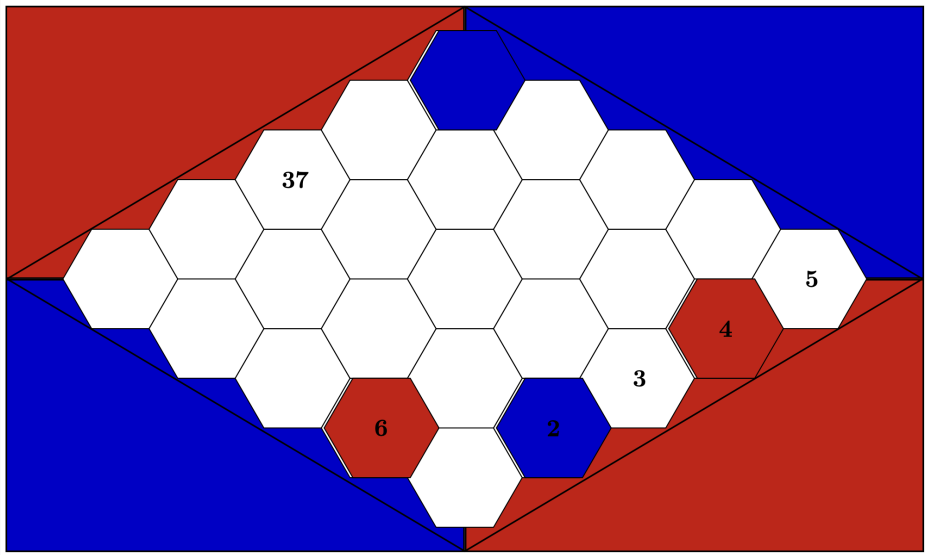
- zwei Pakete, um Hex-Spielbretter und -spielzüge darzustellen
- Autor `hexboard`: Peter Rowlett und Chris Sangwin, 2022
- Autor `hexgame`: Meron Brouwer, 2006
- `hexgame` basiert auf PSTricks, `hexboard` auf TikZ

`hexgame` - Allgemein

- `\usepackage{hexgame}`
- `\begin{hexgame}{.1.}...\end{hexgame}` - Umgebung mit .1. Anzahl der Felder
- `colorhex{.1.}{.2.}` Spielzug darstellen
  .1. Feldnummer .2. `playerone` oder `playertwo`
- `labelhex{.1.}{.2.}` Feld beschriften
  .1. Feldnummer .2. Text

hexgame - Beispiel

```
\begin{hexgame}{5}
\colorhex{2}{playerone}
\colorhex{4}{playertwo}
\colorhex{6}{playertwo}
\colorhex{25}{playerone}
\labelhex{2}{2}
\labelhex{3}{3}
\labelhex{4}{4}
\labelhex{5}{5}
\labelhex{6}{6}
\labelhex{23}{37}
\end{hexgame}
```

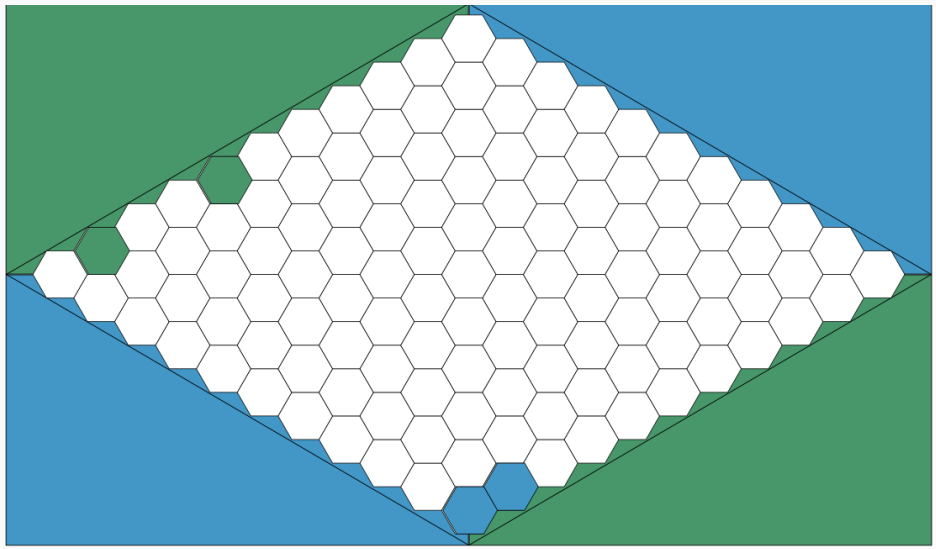


`hexgame` - Optionen

- `\setlength\hexgameboxwidth{}` Breite des Spielfeldes
- `\definecolor{playerone|two}{rgb}{}` Farbe Spieler 1/2

hexgame - Optionen

```
\setlength\hexgameboxwidth
{0.25\textwidth}
\definecolor{playerone}
{rgb}{0.0,0.6,.8}
\definecolor{playertwo}
{rgb}{0.1,0.6,0.4}
\begin{hexgame}{11}
\colorhex{2}{playerone}
\colorhex{112}{playertwo}
\colorhex{1}{playerone}
\colorhex{115}{playertwo}
\end{hexgame}
```



`hexboard` - Allgemein

- bietet mehr Möglichkeiten wie `hexgame`
- mögliche nächste Züge können angezeigt werden
- unregelmäßige Spielbretter oder einzelne Reihen können erzeugt werden
- Spielbretter können auch für drei und mehr Spieler erzeugt werden (oder andere Hex-Games)

`hexboard` - Anwendung

- `\usepackage{hexboard}`

Spielbretter:

- Spielbretter werden innerhalb der `\begin{hexpicture}...\end{hexpicture}` gezeichnet
- `\hexboard{<num>}` zeichnet ein Spielbrett mit <num>x<num> Reihen
- `hexcellshaded[Farbe]{.1.}{.2.}` gefärbte Zelle an Position x=.1. und y=.2. Optional: Farbe (default:gelb)
- Reihenfolge beim Zeichnen beachten! (Vorder-/Hintergrund)

`hexboard` - Anwendung

Spielzüge:

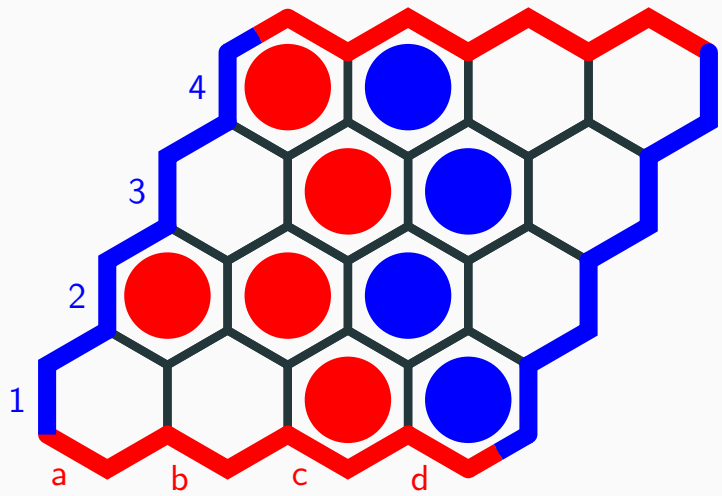
- Spielzüge werden innerhalb der `\begin{hexgame}...\end{hexgame}`-Umgebung gezeichnet
- `\hexmove{.1.}{.2.}` Stein an Position x=.1. und y=.2.
- Züge werden abwechselnd rot und blau gezeichnet (geregelt über Counter: ungerade=rot, gerade=blau)
- Zusätzlich: `\begin{hexgamelabels}...\end{hexgamelabels}`-Umgebung
  - Nummer des Spielzugs wird ausgegeben
  - `\hexskipmove` wird verwendet, wenn blau und rot getauscht werden, die Nummerierung des Spielzugs aber beibehalten werden soll.
  - Befehl funktioniert auch in `\begin{hexgame}`-Umgebung und tauscht dort die Farben

`hexboard` - **Anwendung**

- Größensteuerung über `\hexscale{<num>}` **vor** dem Aufruf der Umgebung
- `setcolorA|B{ }` Anpassungen Farbe Spieler A bzw. B

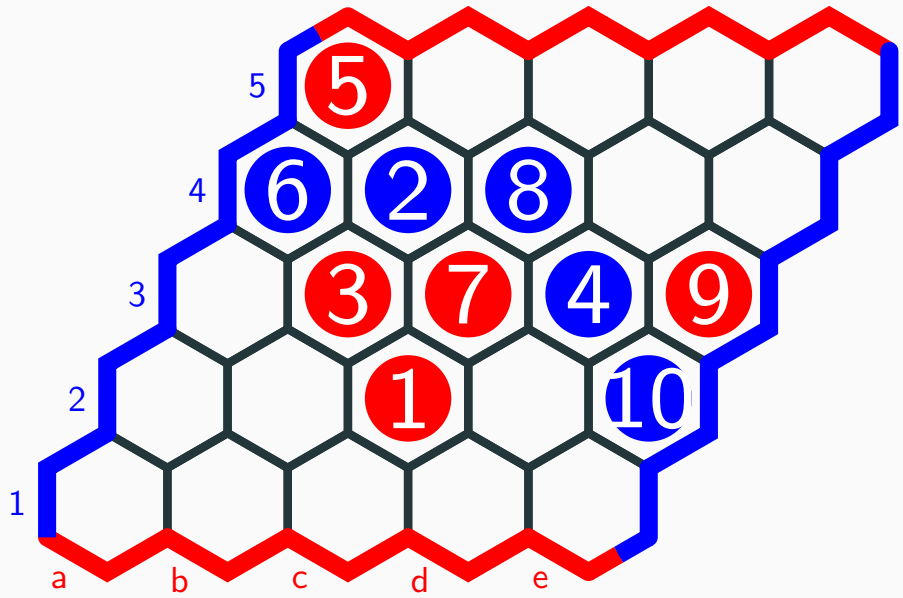
hexboard - Beispiel

```
\begin{hexgame}[4]
\hexmove{a}{2}
\hexmove{c}{3}
\hexmove{b}{2}
\hexmove{c}{2}
\hexmove{c}{1}
\hexmove{d}{1}
\hexmove{b}{3}
\hexmove{b}{4}
\hexmove{a}{4}
\end{hexgame}
```



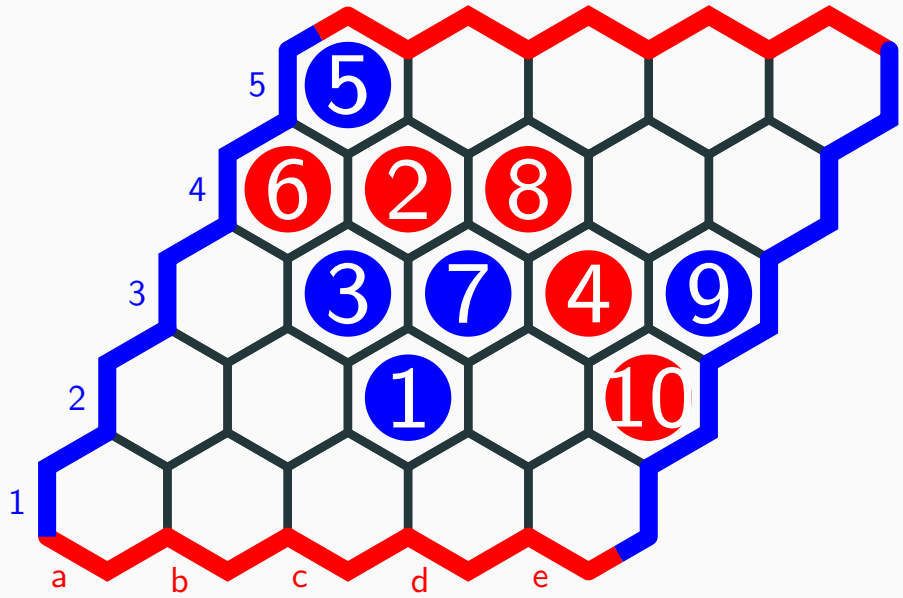
hexboard - Beispiel

```
\begin{hexgamelabels}[5]
\hexmove{c}{2}
\hexmove{b}{4}
\hexmove{b}{3}
\hexmove{d}{3}
\hexmove{a}{5}
\hexmove{a}{4}
\hexmove{c}{3}
\hexmove{c}{4}
\hexmove{e}{3}
\hexmove{e}{2}
\end{hexgamelabels}
```



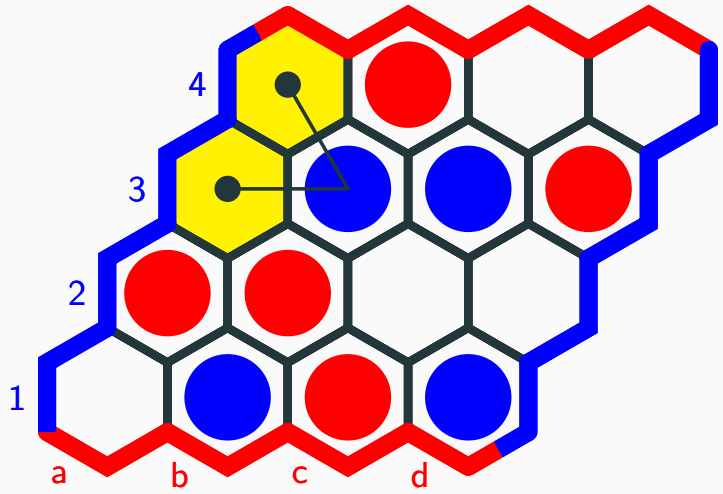
hexboard - Beispiel

```
\begin{hexgamelabels}[5]
\hexskipmove
\hexmove{c}{2}
\hexmove{b}{4}
\hexmove{b}{3}
\hexmove{d}{3}
\hexmove{a}{5}
\hexmove{a}{4}
\hexmove{c}{3}
\hexmove{c}{4}
\hexmove{e}{3}
\hexmove{e}{2}
\end{hexgamelabels}
```



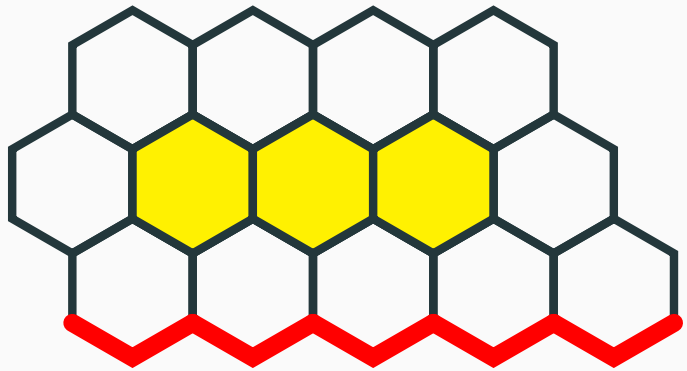
hexboard - Nächste Züge und hervorgehobene Zellen

```
\begin{hexgame}[4]
\hexcellshaded{a}{3}
\hexcellshaded{a}{4}
\hexgrid{4}{4}
\hexmove{b}{2}
\hexmove{b}{1}
\hexmove{c}{1}
\hexmove{d}{1}
\hexmove{a}{2}
\hexmove{b}{3}
\hexmove{b}{4}
\hexmove{c}{3}
\hexmove{d}{3}
\hexdot{a}{4}
\hexdot{a}{3}
\hexconnect{b}{3}{a}{3}
\hexconnect{b}{3}{a}{4}
\end{hexgame}
```



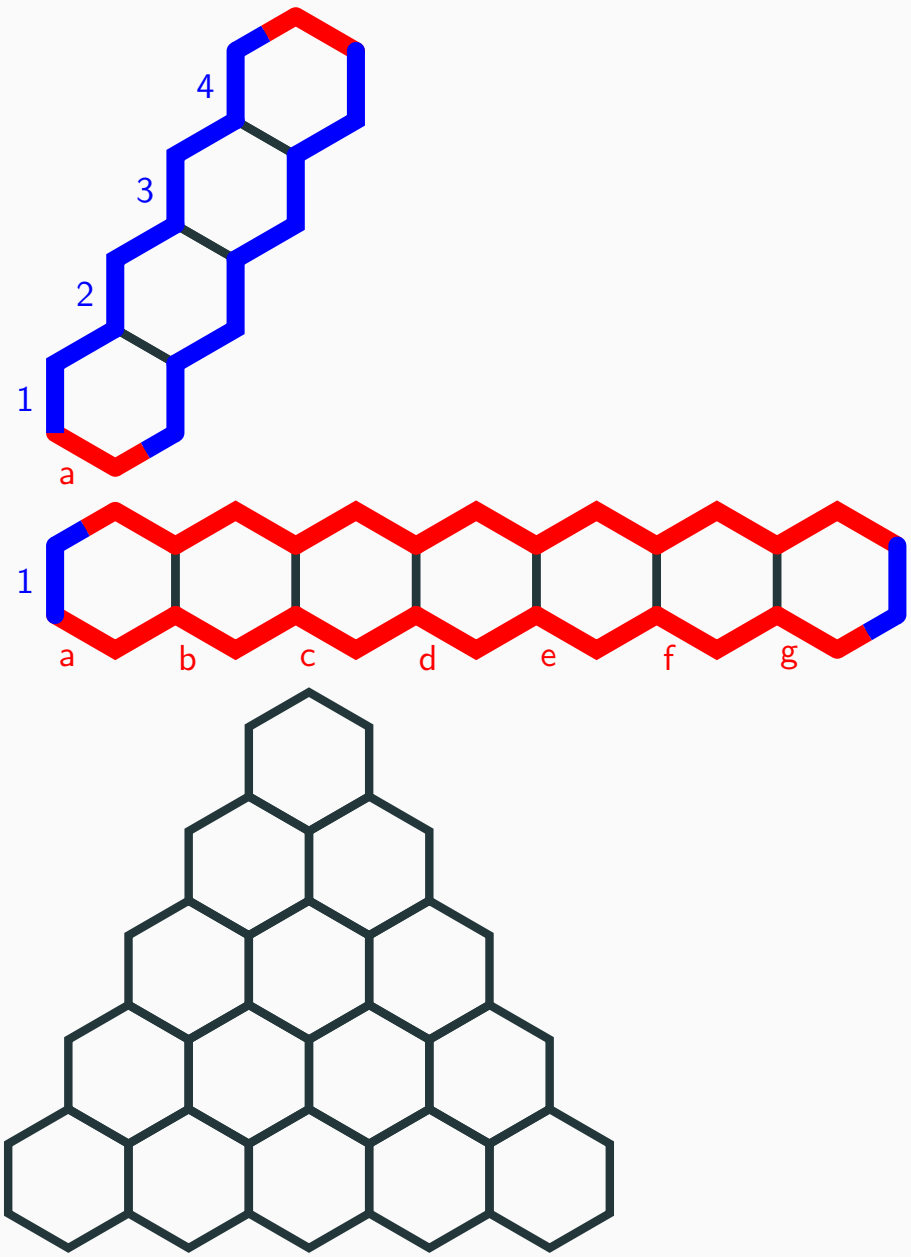
`hexboard` - Einzelne Zellen, Reihen und Sonderfälle

```
\begin{hexpicture}
\hexsubrow{b}{f}{1}
\hexsubrow{a}{e}{2}
\hexsubrow{a}{d}{3}
\hexshadedsubrow{b}{d}{2}
\hexbottomsubborder{b}{f}
\end{hexpicture}
```

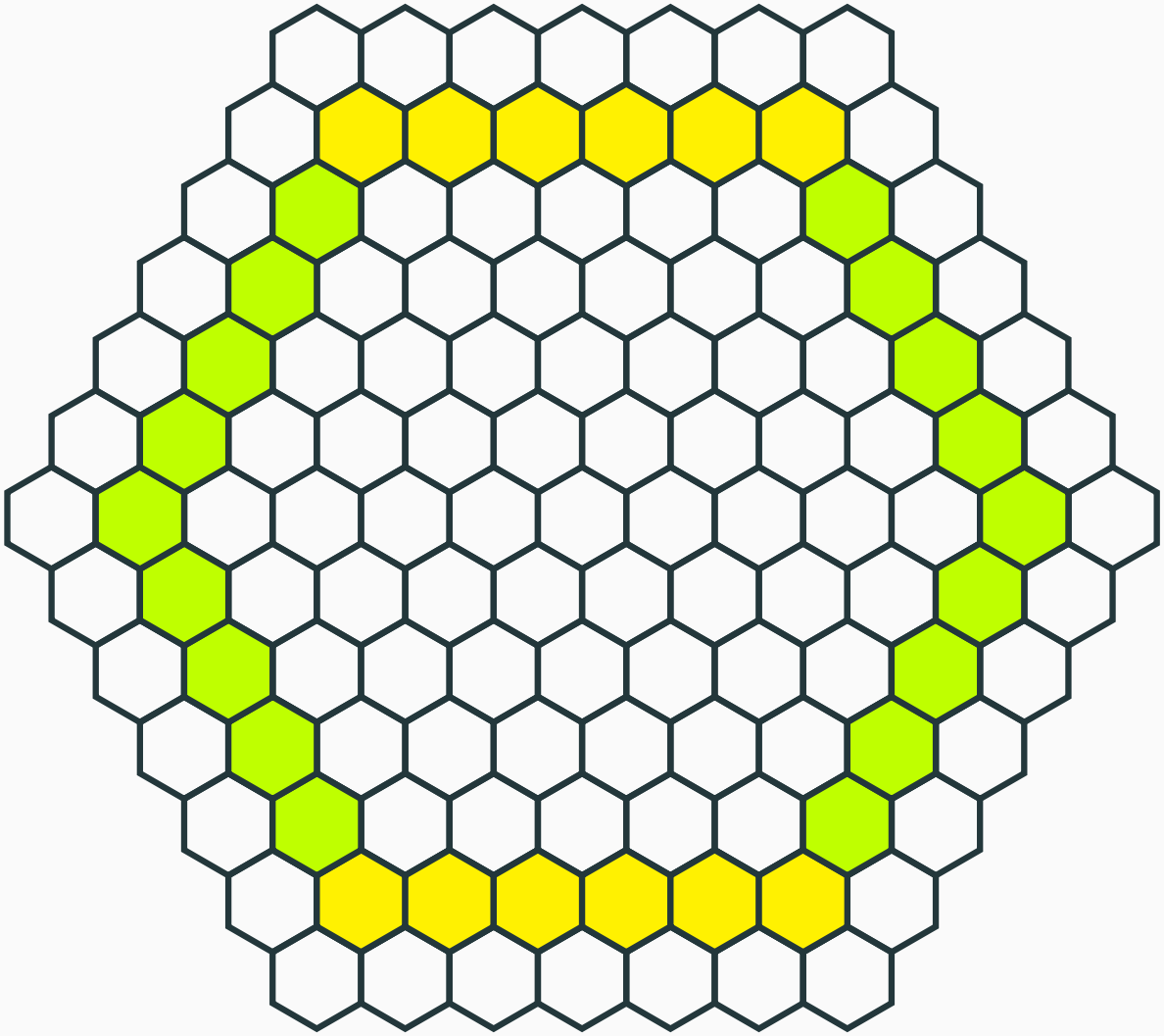


hexboard - Einzelne Zellen, Reihen und Sonderfälle

```
\begin{hexpicture}
\hexgrid{1}{4}
\end{hexpicture}

\begin{hexpicture}
\hexgrid{7}{1}
\end{hexpicture}

\begin{hexpicture}
\hexreducingnoborder{5}{5}
\end{hexpicture}
```



`hexboard` - Einzelne Zellen, Reihen und Sonderfälle



Gute Karten haben

`JeuxCartes` - Allgemein

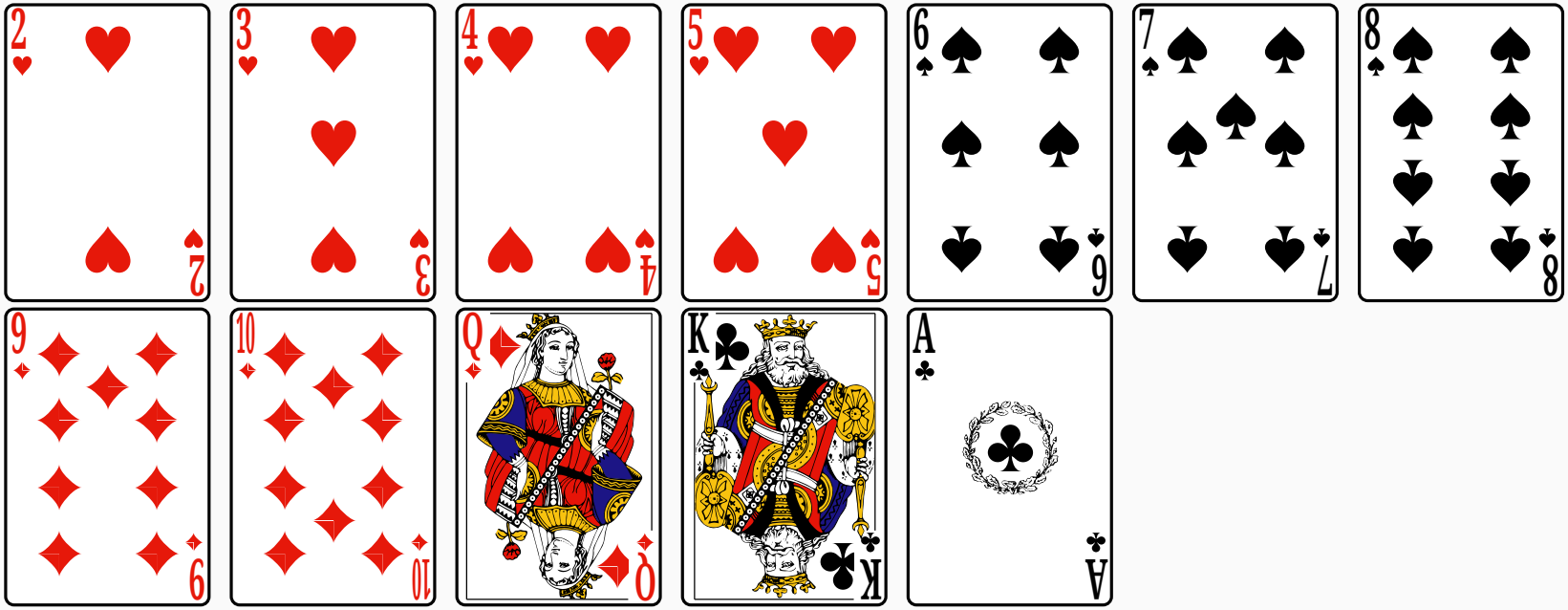
- Autor: Cédric Pierquet, 2023
- Doku nur auf französisch
- Abhängigkeiten:
  `tikz, pifont, pgffor, xinttools, listofitems, xstring, simplekv, randomlist`
- `\usepackage{JeuxCartes}`
- Darstellung einzelner Karten, Kartenstapel oder Mini-Karten (nicht für jedes Spiel ist alles verfügbar)
- folgende Spiele können dargestellt werden:
  - Bataille („Leben und Tod “) – Kinder bzw. Kasinospiel
  - Tarot
  - Uno
  - Poker
  - BlackJack
  - Romme
  - Belote – französisches Kartenspiel

JeuxCartes - Allgemein

- `\AffCarteJeu[Spiel]{Kartenwert}` zeigt einzelne Karte an
- `\AffCarteJeus[Spiel]{Karte1 § Karte 2 § ...}` zeigt mehrere Karte in einer Reihe an
- klassisch (Poker, Belote, Bataille)
  - Kartenwerte `2C,3C,4C,... DC,RC,AC` – C steht für Herz (*Cœur*)
  - analog: P – Pik (*Piquet*); K – Karo (*Carreau*); T – Kreuz (*Trèfle*)
  - `JN, JR` – Joker und `Dos` (Rückseite)
- Tarot
  - Kartenwerte `2C, 3C, 4C,... CC, ..., AC` – C steht für Herz (*Cœur*)
  - analog: P – Pik (*Piquet*); K – Karo (*Carreau*); T – Kreuz (*Trèfle*)
  - `Exc,1AT,2AT, ... 20AT, 21AT` Tarot-Trümpfe (Große Arkana) und `Dos` (Rückseite)

- Uno
  - Kartenwerte `0B,1B,...,8B,9B` – B steht für Blau (*bleu*)
  - analog: R – Rot (*Rouge*); J – Gelb (*Jaune*); V – Grün (*Vert*)
  - `P2B, P2R, ...` +2-Karte
  - `PTB, PTR, ...` Aussetzen (*PasseTour*)
  - `CSB, CSR, ...` Richtungswechsel (*ChangeSens*)
  - `P4` +4-Karte
  - `Coul` Farbwahlkarte
  - `Dos` Rückseite

JeuxCartes - einzelne Karten



```
\AffCarteJeu{2C}
\AffCarteJeu{3C}
\AffCarteJeu{4C}
\AffCarteJeu{5C}
\AffCarteJeu{6P}
\AffCarteJeu{7P}
```

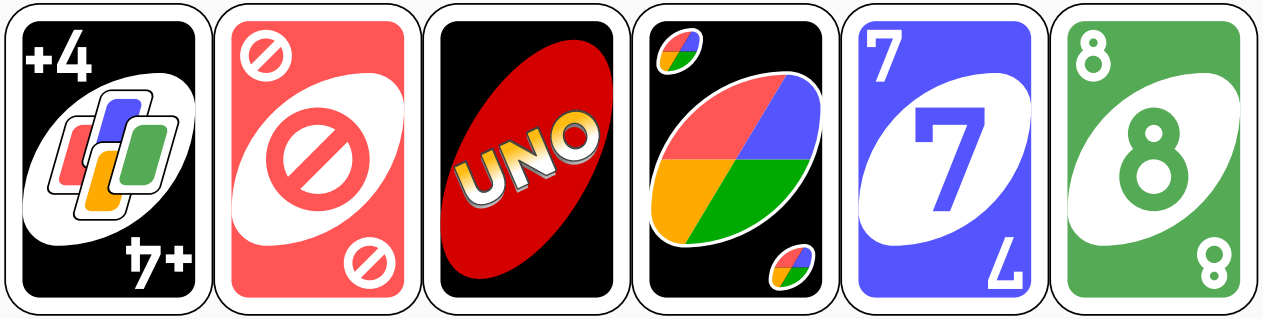
```
\AffCarteJeu{8P}
\AffCarteJeu{9K}
\AffCarteJeu{10K}
\AffCarteJeu{DK}
\AffCarteJeu{RT}
\AffCarteJeu{AT}
```

JeuxCartes - einzelne Karten

```
\AffCarteJeu[TypeJeu=Tarot]{VP}
\AffCarteJeu[TypeJeu=Tarot]{Dos}
\AffCarteJeu[TypeJeu=Tarot]{1AT}
\AffCarteJeu[TypeJeu=Tarot]{10K}
\AffCarteJeu[TypeJeu=Tarot]{Exc}
\AffCarteJeu[TypeJeu=Tarot]{1AT}
\AffCarteJeu[TypeJeu=Tarot]{2AT}
\AffCarteJeu[TypeJeu=Tarot]{3AT}
```

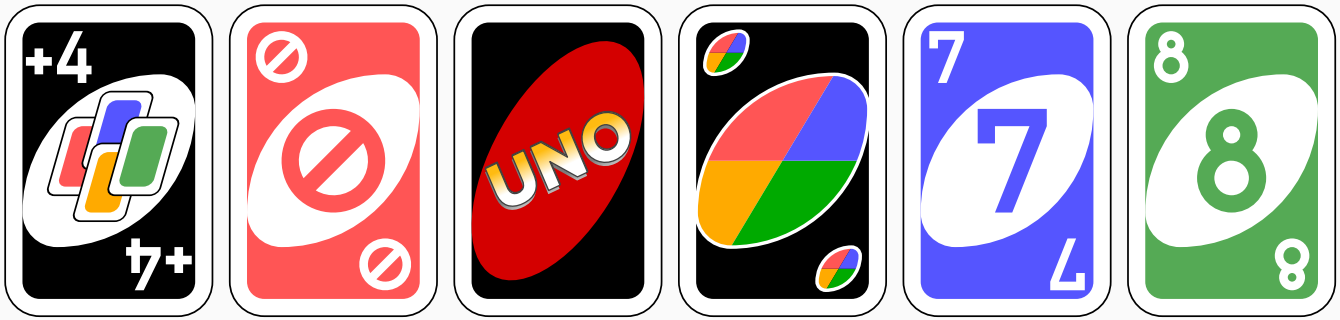
```
\AffCarteJeu[TypeJeu=Tarot]{4AT}
\AffCarteJeu[TypeJeu=Tarot]{5AT}
\AffCarteJeu[TypeJeu=Tarot]{6AT}
\AffCarteJeu[TypeJeu=Tarot]{7AT}
\AffCarteJeu[TypeJeu=Tarot]{8AT}
\AffCarteJeu[TypeJeu=Tarot]{9AT}
\AffCarteJeu[TypeJeu=Tarot]{10AT}
...
```

JeuxCartes - mehrere Karten in einer Reihe




```
\AffCartesJeu[TypeJeu=Uno]{P4 § PTR § Dos § Coul § 7B § 8V}
```

Zum Vergleich:

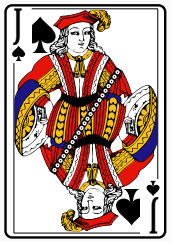
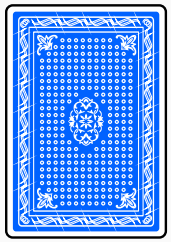
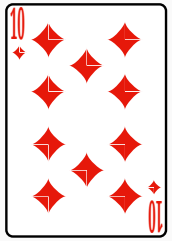



```
\AffCarteJeu[TypeJeu=Uno]{P4}
\AffCarteJeu[TypeJeu=Uno]{PTR}
\AffCarteJeu[TypeJeu=Uno]{Dos}
\AffCarteJeu[TypeJeu=Uno]{Coul}
\AffCarteJeu[TypeJeu=Uno]{7B}
\AffCarteJeu[TypeJeu=Uno]{8V}
```

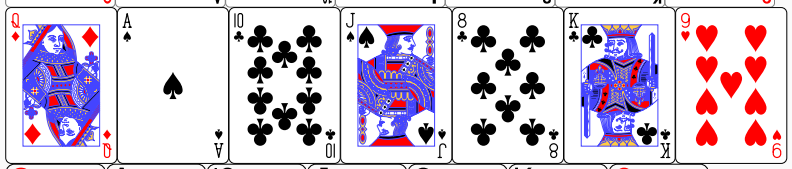
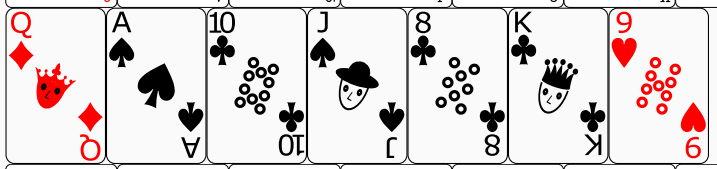
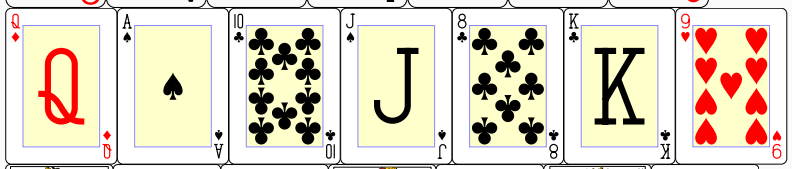
JeuxCartes - Anpassungsmöglichkeiten

Option	Default	Wert	Bedeutung
Hauteur	4.25	<dim>	Größe der Karten
TypeJeu	Poker	Poker|Tarot|Uno	Kartendecktyp
StyleJeu	v1	v1 ... v5, fr, bicycl	Kartendeckvariante bei Poker & BlackJack
Rotation	0	<dim>	Rotation
AlignementV	0.5	<dim>	vertikale Ausrichtung

Außerdem gibt es einige Optionen für die Verwendung innerhalb einer `tikzpicture`-Umgebung

Regulär werden Karten mittig bezogen auf die Grundlinie platziert:

. `AlignmentV=0` bewirkt eine Verschiebung nach oben, so daß die untere Kante auf der Grundlinie liegt  .
`AlignementV=1` verschiebt die Karte nach unten (Oberkante auf der Grundlinie)  und `\Rotation=25` dreht die Karte um 25 Grad  .

JeuxCartes - Kartendecks für Poker und BlackJack

StyleJeu=	Kartendeck
v1	
v2	
v3	
v4	
v5	
fr	
bicycl	

Handkarten werden mit dem Befehl
`\MainCartesJeu[.Optionen.]` ausgegeben
Zusätzliche Optionen:

Option	**Default**	**Wert**	**Bedeutung**
EspH	1	<dim>	horizontaler Abstand zwischen den Karten
EspV	0	<dim>	vertikaler Abstand zwischen den Karten
Eventail	false	true|false	Fächerhand
Inverse	false	true|false	Darstellung von links nach rechts

JeuxCartes - Handkarten

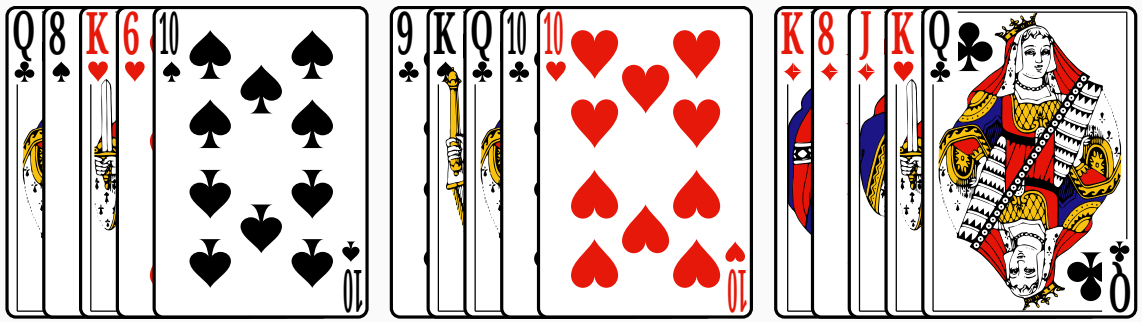
```
\MainCartesJeu{7K § 8P § DT § AC}

\MainCartesJeu[Eventail,EspH=0,EspV=0.1]{7K § 8P § DT § AC}

\MainCartesJeu[Eventail,EspH=0.6]{7K § 8P § DT § AC}

\MainCartesJeu[TypeJeu=Uno,EspH=1.75,Hauteur=2]
{9B § P4 § P2J § 1R § 7R § PTV § 2J § 5J § 9B § PTR § 5V}
```

JeuxCartes - Handkarten

```
\MainCartesJeu[TypeJeu=Tarot,EspH=1,EspV=-0.15,Hauteur=2]
{Exc § 1AT § CC § 8T § 2K § 5AT § 2AT § DP § 7T}

\MainCartesJeu[Inverse,Eventail,Hauteur=2,TypeJeu=Tarot,EspH=0,EspV=0.1]
{Exc § 1AT § 2AT § 3AT § 4AT § 5AT § 6AT § 10AT § 11AT § 15AT § 16AT § 19AT
§ 20AT § 21AT}

\MainCartesJeu[Eventail,Hauteur=2,TypeJeu=Tarot,EspH=0.5,EspV=0.15,
Rotation=20]{DK § 10AT § 16AT § VP § 1AT}
```

JeuxCartes - Zufälle gibt's!

Mit `\MainCartesJeuAleatoire[.Options.]{Kartenanzahl}` werden zufällige Karten ausgegeben. Optionen wie bei `\MainCartesJeu`



```
\MainCartesJeuAleatoire{5}
\MainCartesJeuAleatoire{5}
\MainCartesJeuAleatoire{5}
```

`JeuxCartes` - Klein aber fein

Mit `JeuxCartes` können auch Miniturkarten angegeben werden (nicht für Uno).

Bezeichnung	Wert
2.C, 3.C, 4.C,...C.C,D.C,R.C.,A.C	französische Herzkarten (***C**œur*)
2.C, 3.C, 4.C,...J.C,Q.C,K.C	englische Herzkarten
analog P (Pik / *Pique*), K (Karo / *Carreau*) & T (Kreuz / *Trèfle*)	
JO.N, JO.R	Joker
Exc, 1.AT, 2.AT, ...20.T, 21.AT	Tarot-Trümpfe (Große Arkana)

JeuxCartes - Klein aber fein

Wen man Karten im Text integrieren möchte, eignen sich die Minisymbole gut 7♦ 1✸ V♦ 10♥. Diese sind auf der Grundlinie ausgerichtet und haben die Höhe des Textes. Oder mit englischen Kartennamen K♣ J❦ Q♥ J♠.
Der Befehl `\AffMiniCarteJeu[Option]{Karte}` kennt die Optionen `Largeur` für die Größe (`default: 0.55cm`) und `FondAtout` für die Hintergrundfarbe der Tarottrumpfkarten (`default=yellow`).

JeuxCartes - Klein aber fein

- Auch hier gibt es die Variante, sich zufällig ein Blatt ausgeben zu lassen: `\MainMiniCartesJeuAleatoire[Option]{Anzahl Karten}` und Karten als Reihe anzugeben: `\MainMiniCartesJeu[Option]{Karte 1 § Karte 2 § ...}`
- Als Option steht `TypeJeu=Poker|Belote|Tarot|Bataille|Rami|PokerEN|BeloteEN|BatailleEN|RamiEN` zur Verfügung.

Ist doch alles ganz logisch...

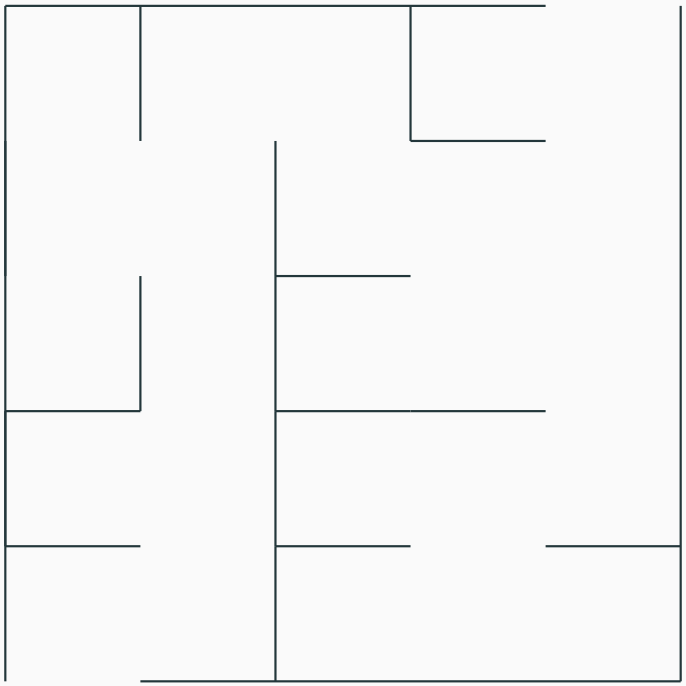
`maze` und `labyrinth` – Allgemein

- zwei Pakete, um Labyrinthe zu erzeugen
- Autor `maze`: Sicheng Du, 2023
- Autor `labyrinth`: Francesco Zigliotto, 2014
- `\usepackage{maze}` bzw.
- `\usepackage{labyrinth}`

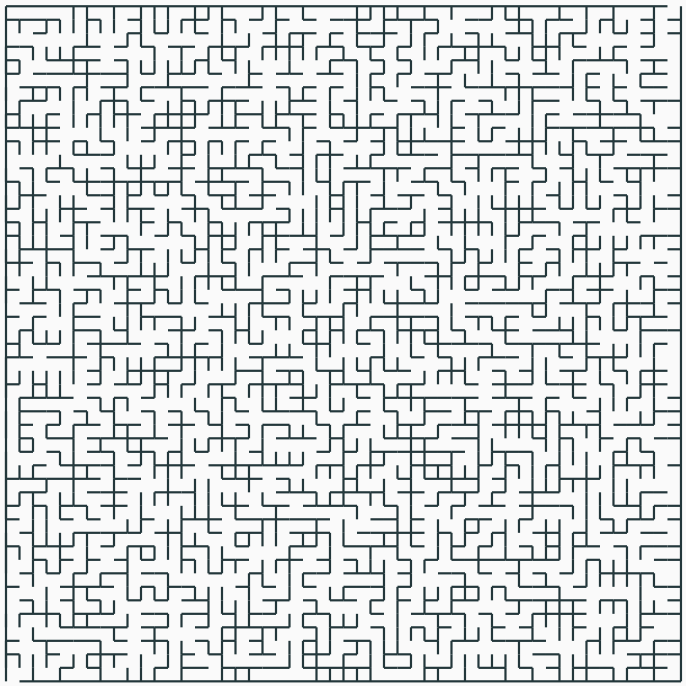
`maze` – Allgemein

- `\maze{size}[seed]`
- `size` Dichte der Wände im Labyrinth.
- Je höher die Anzahl ist, desto komplexer ist das Labyrinth.
- Zahl im Bereich [2,100]
- `seed` optionaler Parameter (Zufallsgenerator).
  - wenn leer gelassen, wird die aktuelle Minute genutzt.

maze – Beispiel 1



```
\maze{5}[1]
```



```
\maze{50}[105]
```

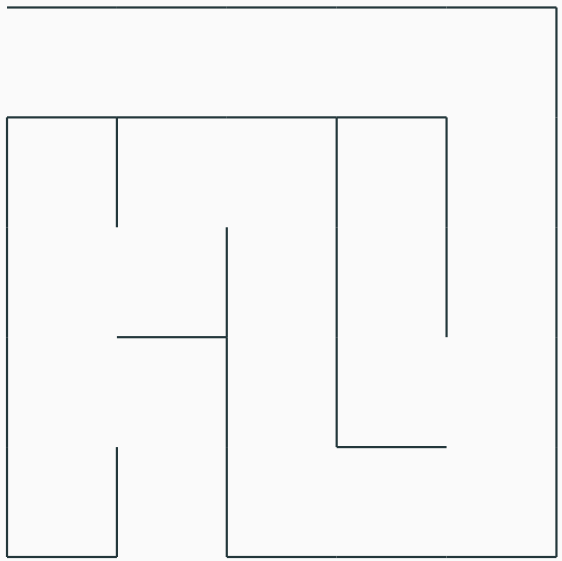
`labyrinth` – Allgemein

- erzeugt Labyrinthe mit Lösungswegen
- Ansatz: Beschreibung mit einer Zeichensequenz
- Labyrinth wird in ein n x m-Gitter zerlegt und horizontale und vertikale Linien getrennt betrachtet
  - horizontale Linien (h): Für jedes Kästchen muß definiert werden, ob untere Linie zum Labyrinth gehört oder nicht (+: gehört dazu, -: gehört nicht dazu).
  - vertikale Linien (v): Es gibt n+1 vertikale Linien. Falls eine Linie zum Labyrinth gehört: +, sonst -
  - Angabe im Code: `\v ++--+ \h ---+-`

`labyrinth` – Allgemein

- `\begin{labyrinth}[.1.]{.2.}{.3.} \end{labyrinth}` mit .1. Optionen, .2. Breite und .3. Höhe in ganzen Zahlen
- Optionen
  - `unit=<dim>` Größe eines Kästchens in pt, default: 11pt
  - `thickness` Linienbreite
  - `centered=true|false` Zentriert Labyrinth

labyrinth – Beispiel

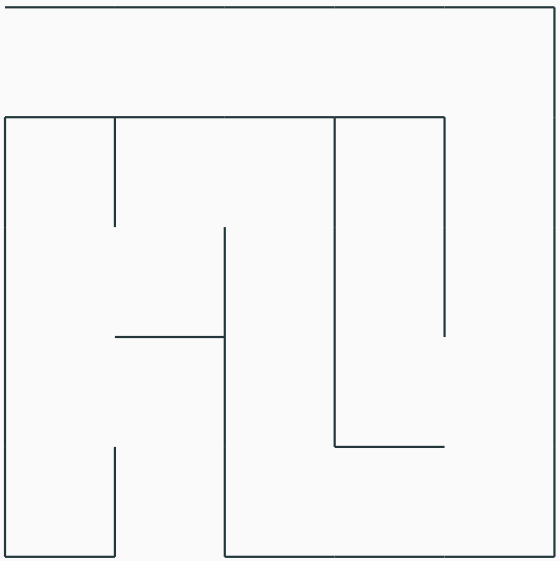


```
\begin{labyrinth}
 [unit=20pt]{6}{5}
\v ------ \h +++++
\v -----+  \h ++++-
\v ++-+++  \h -----
\v +-++++  \h -+---
\v +-++-+  \h ---+
\v +++--+  \h +-+++
\end{labyrinth}
```

`labyrinth` – Vereinfachungen

- alle - am Ende einer Zeile können weggelassen werden
- falls es nur - gibt, kann die ganze Zeile weggelassen werden
- ab drei aufeinanderfolgenden + oder - können diese geschrieben werden als `*<Anzahl>-` oder `*<Anzahl>+`

labyrinth – Beispiel



```
\begin{labyrinth}
 [unit=20pt]{6}{5}
  \h *5+
\v -----+  \h *4+
\v ++-+++
\v +-*4+  \h -+
\v +-++-+  \h ---+
\v +++--+  \h +-+++
\end{labyrinth}
```

`labyrinth` – Lösungswege

- `\labyrinthsolution[Optionen](x,y){Lösungsweg}`
- `(x,y)` Startkoordinaten des Lösungswegs
- `Lösungsweg` wird mit den Zeichen `{u,d,l,r}` angegeben (up, down, left, right). Jeder Schritt ist `\unitlength` lang.
- Angabe des Lösungswege innerhalb der `labyrinth`-Umgebung
- es können mehrere Lösungswege angegeben werden

`labyrinthsolution` – Optionen

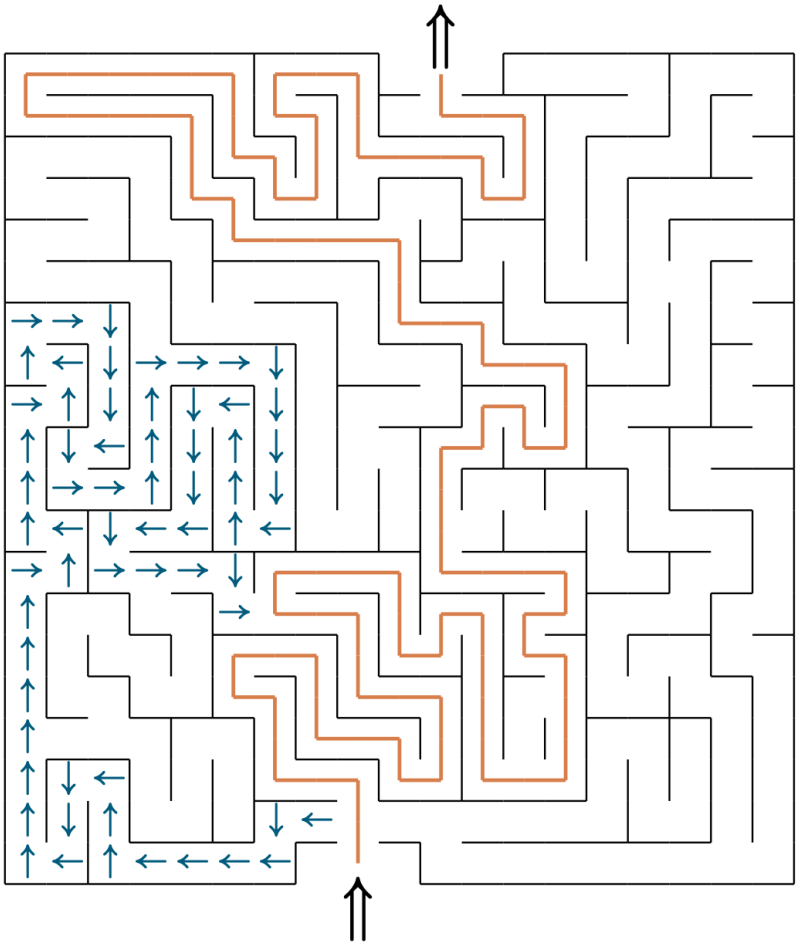
Option	Default	Wert
hidden	false	true / false
thicklines	true	true / false
up	`line(0,1){1}`	Symbol für einen Schritt nach oben (u)
down	`line(0,-1){1}`	Symbol für einen Schritt nach unten (d)
right	`line(1,0){1}`	Symbol für einen Schritt nach rechts (r)
left	`line(-1,0){1}`	Symbol für einen Schritt nach links (l)
font	red	

`autosolution` – automatischer Lösungsweg

```
\autosolution[Optionen]
 (xA,yA)(xB,yB){Richtung erster Pfeil}
```

mit
`(xA,yA)` Startpunkt Lösungsweg
`(xB,yB)` Endpunkt Lösungsweg
`Optionen` wie bei `labyrinthsolution`
sehr lange Rechenzeiten!

labyrinthsolution – Beispiel



```
\begin{labyrinth}[unit=9pt]{19}{20}
...
\autosolution[font=\color{Orange}]
 (8,0)(10,19){u}
\labyrinthsolution[
  font=\color{MidnightBlue}\footnotesize,
  up=\kern2pt$\uparrow$,
  left=$\leftarrow$,
down=\kern2pt$\downarrow$,
right=$\rightarrow$,
...](7,1){%
ldllllluulddluuuuuuuruluuurulurrdd%
dldrruuurrrddddluuuldddlldrrrdr}
\end{labyrinth}
```

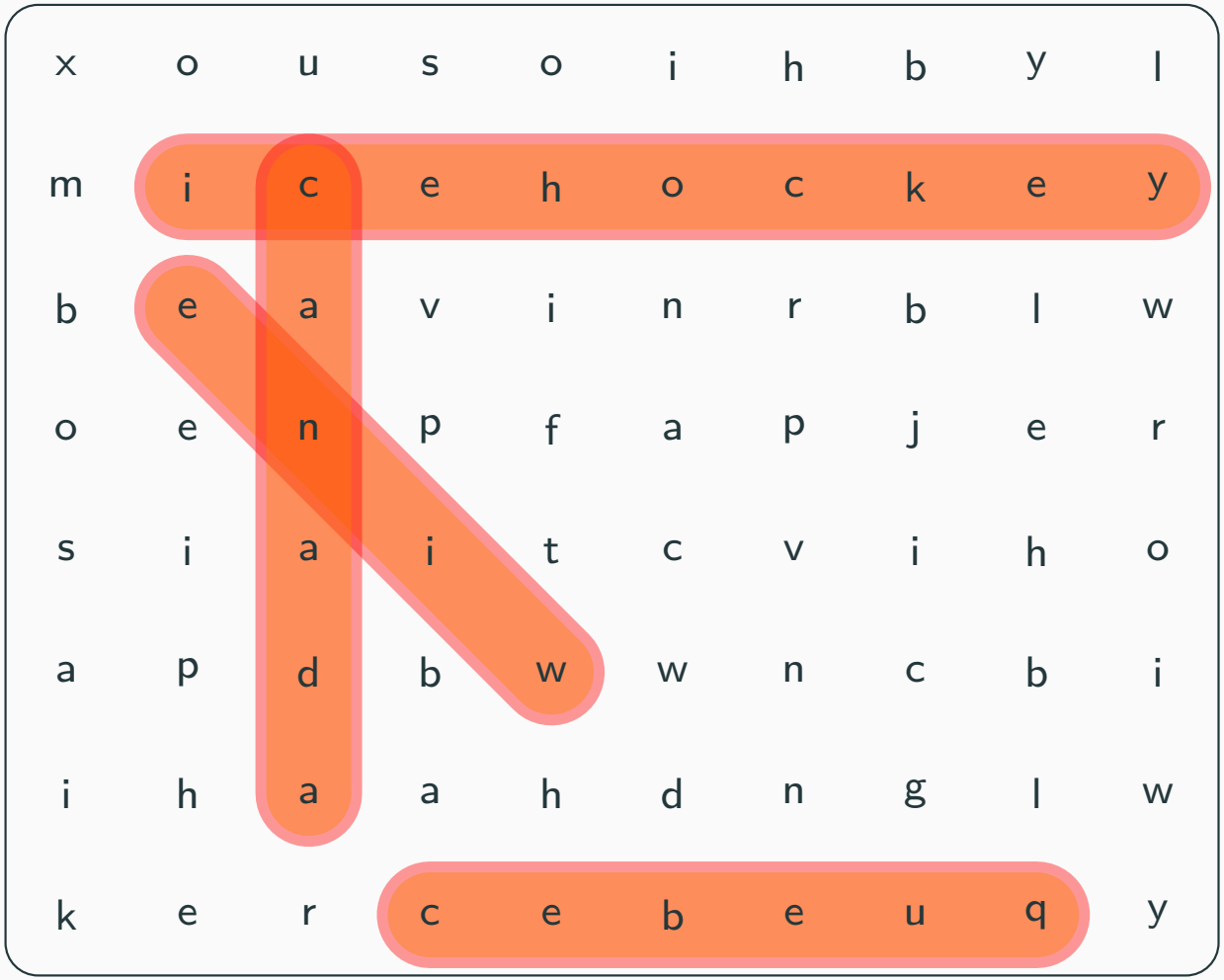
`soup` - Allgemein

- Autor: Thomas Simers, 2019
- In einer „Buchstabensuppe“ müssen bestimmte Wörter gefunden werden
- basiert auf TikZ
- `\usepackage[]{soup}`
- Optionen:
  - `usetikz=true` statt TikZ `tabular` verwenden
  - `highlight=true` Lösungen einblenden (default:false)
  - `highlightcolor=` Farbe, in der Lösung hervorgehoben wird (default: orange)
  - `linecolor=` Linienfarbe der eingerahmten Lösung (default: red)

soup - Beispiel

1. s.th. to drink
2. Town in Eastern Canada
3. Famous Sports in Northern America
4. Country in Northern America

soup - Beispiel



soup - Beispiel

```
\begin{alphabetsoup}*[10][8][\scriptsize]
\hideinsoup*{3}{2}{down}{c,a,n,a,d,a}
    [Country in Northern America]
\hideinsoup*{2}{2}{right}{i,c,e,h,o,c,k,e,y}
     [Famous Sports in Northern America]
\hideinsoup*{9}{8}{left}{q,u,e,b,e,c}
    [Town in Eastern Canada]
\hideinsoup*{5}{6}{upleft}{w,i,n,e}[s.th. to drink]
\end{alphabetsoup}
\begin{enumerate}
\listofclues[\item \theclue]
\end{enumerate}
```

`\begin{alphabetsoup}[width][height][font]`
`\end{alphabetsoup}` Kleinbuchstabensuppe, mit Hinweisen
`\begin{alphabetsoup}*[width][height][font]`
`\end{alphabetsoup}` Kleinbuchstabensuppe, ohne Hinweise
`\begin{Alphabetsoup}[width][height][font]`
`\end{Alphabetsoup}` Großbuchstabensuppe, mit Hinweisen
`\begin{Alphabetsoup}*[width][height][font]`
`\end{Alphabetsoup}` Großbuchstabensuppe, ohne Hinweise

soup - Umgebungen

`\begin{numbersoup}[width][height]{max}[min][font]` `\end{numbersoup}` Zahlensuppe, `max` ist verpflichtend, mit Hinweisen
`\begin{numbersoup}*[width][height]{max}[min][font]` `\end{numbersoup}` Zahlensuppe, ohne Hinweise
`\begin{homemadesoup}[width][height][symbols][font]` `\end{homemadesoup}` hausgemachte Suppe mit Zusammenstellung aus indivueller csv-Liste, mit Hinweisen
`\begin{homemadesoup}*[width][height][symbols][font]` `\end{homemadesoup}` hausgemachte Suppe mit indivueller Zusammenstellung, ohne Hinweise
`width` und `height` ist die Zeilenanzahl in x- und y-Richtung.
`font` bezieht sich auf die Größe, z.B. `footnotesize`

- `\hideinsoup{x}{y}{Richtung}{Sequenz}[Hinweis]` Wörter, die versteckt werden sollen
- `\hideinsoup*{x}{y}{Richtung}{Sequenz}[Hinweis]` Wort wird nicht umrandet
- `\hightlightinsoup{x1}{y1}{x1}{y1}` hebt Wörter im Bereich x1/y1 bis x2/y2 hervor
- `\listofclues[format]` gibt Hinweise aus

`width` und `height` ist die Zeilenanzahl in x- und y-Richtung.
`font` bezieht sich auf die Größe, z.B. `footnotesize`

soup - Beispiele

```
\begin{Alphabetsoup}*[6][6][\footnotesize]
\hideinsoup*{1}{2}{right}{D,A,N,T,E}
\hideinsoup*{6}{1}{downleft}{W,E,I,M,A,R}
\hideinsoup*{6}{6}{up}{G,O,E,T,H,E}
\end{Alphabetsoup}

\begin{numbersoup}*[6][6]{9}[0][\footnotesize]
\hideinsoup{6}{1}{downleft}{0,6,0,4}
\hideinsoup{3}{3}{right}{2,0,2,4}
\end{numbersoup}

\begin{homemadesoup}*[6][6]{
    $\nearrow$, $\searrow$,   $\nwarrow$,   $\swarrow$,
    $\uparrow$, $\downarrow$, $\leftarrow$, $\rightarrow$
    }[\footnotesize]
\hideinsoup{6}{2}{downleft}{$\swarrow$,$\swarrow$,$\swarrow$,$\swarrow$}
\end{homemadesoup}
```

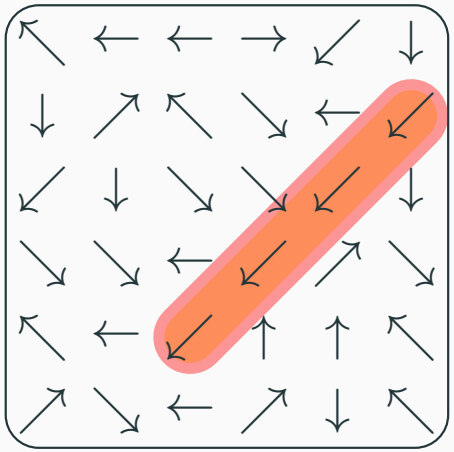
soup - Beispiele

A	D	B	Q	H	W/E
D	A	N	T	E	H
T	S	A	I	O	T
N	T	M	D	S	E
G	A	N	Y	T	O
R	U	B	I	O	G

Alphabetsoup

1	2	0	6	2	0
3	3	2	4	6	9
4	9	2	0	2	4
0	6	4	5	3	6
0	8	2	5	9	6
2	7	3	2	5	6

numbersoup



homemadesoup

`logicpuzzle` - Allgemein und Umgebungen

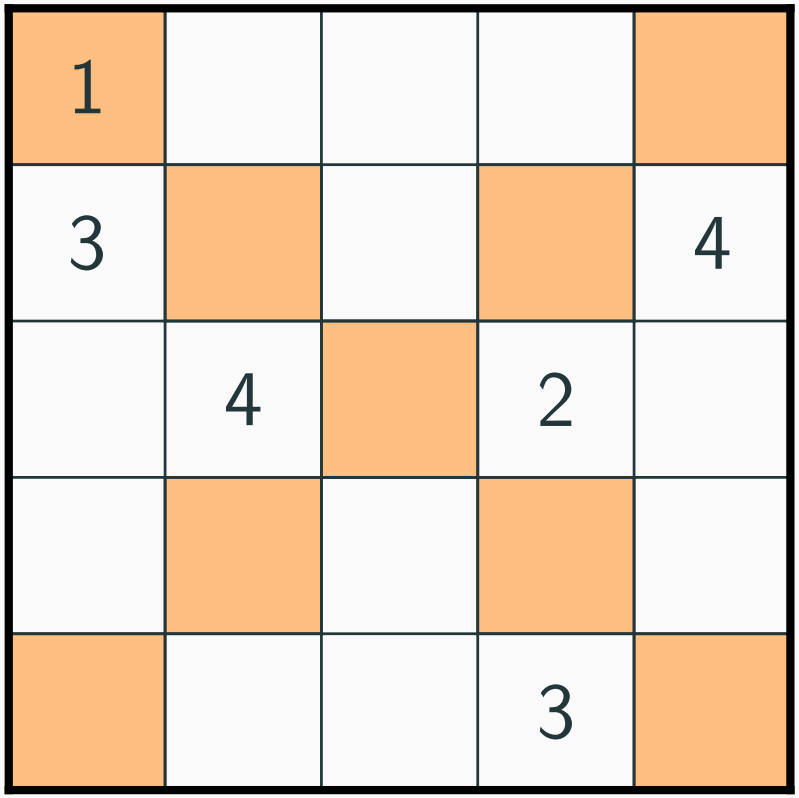
- Autor: Josef Kleber, 2013-2014
- stellt eine sehr große Auswahl an logischen, gitterbasierten Logikpuzzle zur Verfügung
- basiert auf TikZ
- `\begin{logicpuzzle}...\end{logicpuzzle}` als grundlegende Umgebung
- Optionen u.a.
  `rows,colums,width,scale,title,fontsize,color`
  `bgcolor,counterstyle`
- `puzzlebackground` und `puzzleforeground`-Umgebungen für Hintergrund und Vordergrund

`logicpuzzle` - Befehle im Gitter

Es sind Befehle definiert wie beispielsweise

- `setcell(s)`, `setbigcells` Gestaltung einzelner Zellen
- `setrow(s)`, `setcolorrow` Gestaltung von Reihen
- `setcolumn`, `setcolorcolumn` Gestaltung von Spalten
- `fillcell`, `fillrow`, `fillcolumn`, `filldiagonals` Füllungen
- `framearea`, `fillarea`, `colorarea`, Bereiche definieren (über TikZ-Syntax)
- `titleformat` Titel formatieren
- `puzzlecounter`, `setpuzzlecounter`,... diverse Zähler
- `setgridlinestyle`, `setnormallinewith`, `setthicklinewidth` Angaben zu Linien
- für die einzelnen Puzzles werden weitere spezifische Befehle definiert

logicpuzzle - Beispiele

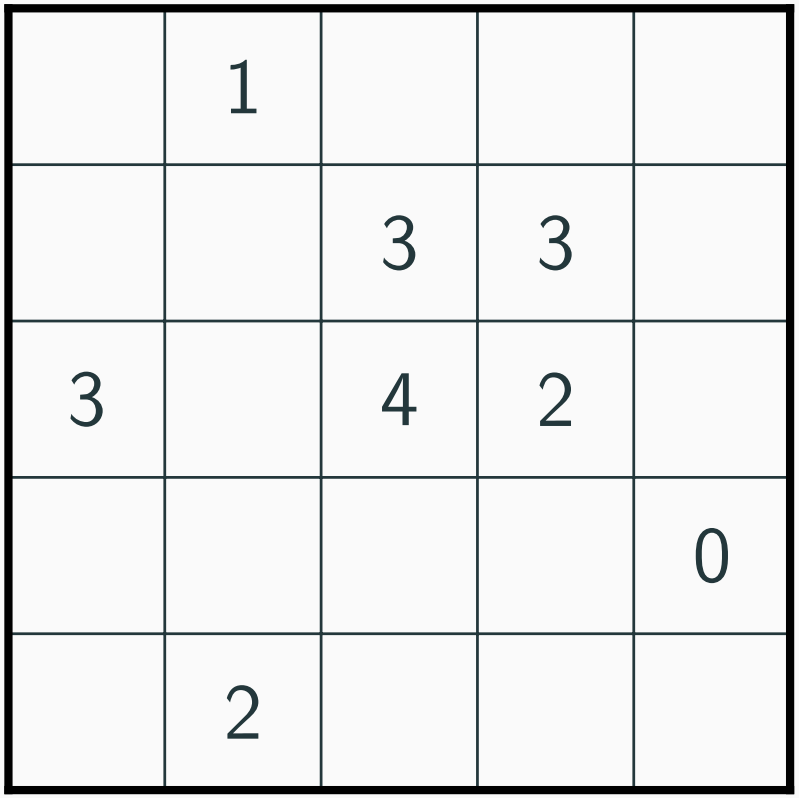



```
\begin{ddsudoku}
    \framepuzzle
    \filldiagonals[orange!50]
    \ddsudokucell{1}{5}{1}
    \ddsudokucell{1}{4}{3}
    \ddsudokucell{2}{3}{4}
    \ddsudokucell{4}{1}{3}
    \ddsudokucell{4}{3}{2}
    \ddsudokucell{5}{4}{4}
\end{ddsudoku}
```

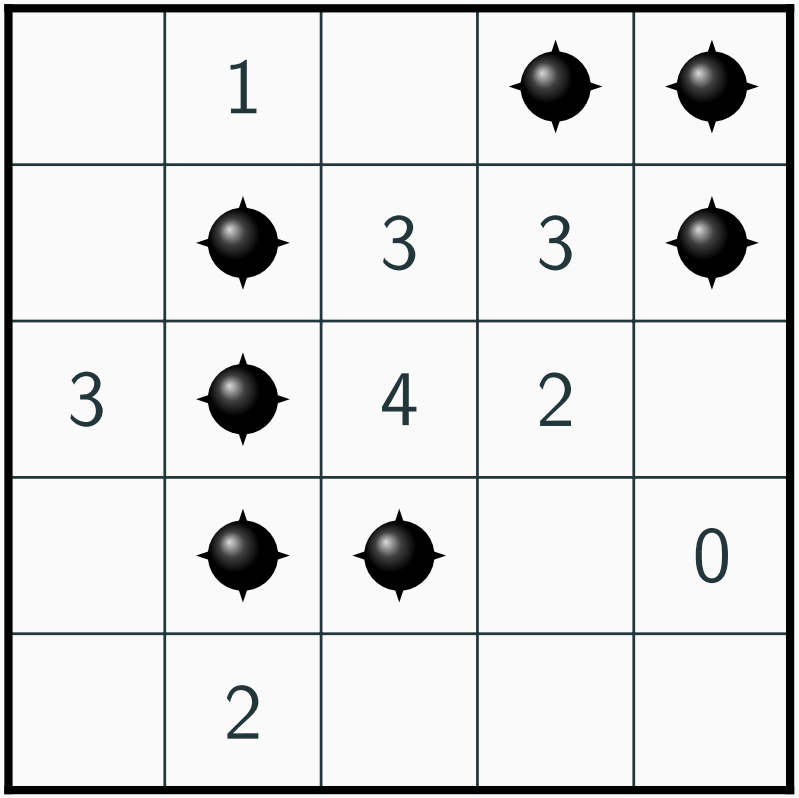
1	3	4	5	2
3	2	5	1	4
5	4	3	2	1
2	5	1	4	3
4	1	2	3	5

```
\begin{ddsudoku}
    \framepuzzle
    \filldiagonals[orange!50]
    \setrow{5}{1,3,4,5,2}
    \setrow{4}{3,2,5,1,4}
    \setrow{3}{5,4,3,2,1}
    \setrow{2}{2,5,1,4,3}
    \setrow{1}{4,1,2,3,5}
\end{ddsudoku}
```

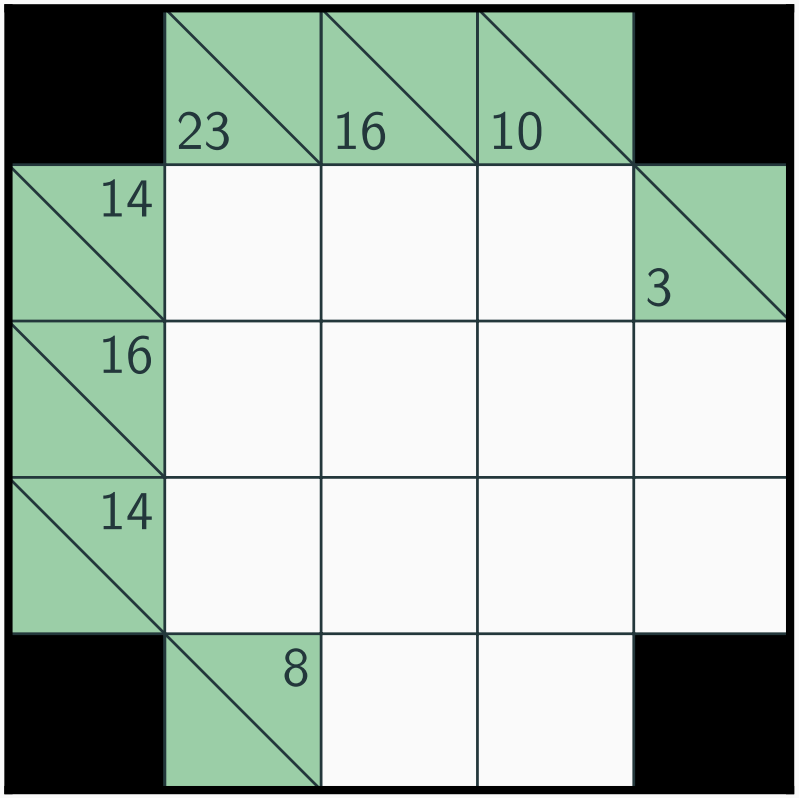
logicpuzzle - Beispiele



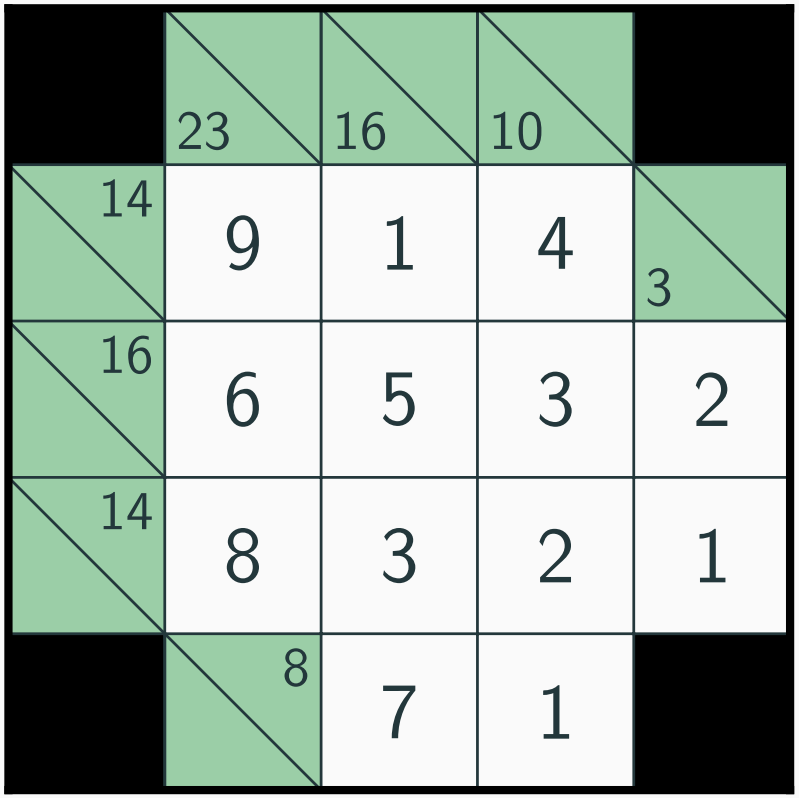

```
\begin{minesweeper}
   \framepuzzle
   \setrow{5}{{},1}
   \setrow{4}{{},{},3,3}
   \setrow{3}{3,{},4,2}
   \setrow{2}{{},{},{},{},0}
   \setrow{1}{{},2}
 \end{minesweeper}
```




```
\begin{minesweeper}
   \framepuzzle
   \setrow{5}{{},1,{},\Mine,
      \Mine}
   \setrow{4}{{},\Mine,3,3,
      \Mine}
   \setrow{3}{3,\Mine,4,2}
   \setrow{2}{{},\Mine,\Mine,
      {},0}
   \setrow{1}{{},2}
 \end{minesweeper}
```




```
\begin{kakuro}
\framepuzzle
\kakurorow{5}{\Black,\KKR{23}{},
    \KKR{16}{},\KKR{10}{},\Black}
\kakurorow{4}{\KKR{}{14},9,1,4,
    \KKR{3}{}}
\kakurorow{3}{\KKR{}{16},6,5,3,2}
\kakurorow{2}{\KKR{}{14},8,3,2,1}
\kakurorow{1}{\Black,\KKR{}{8},7,
    1,\Black}
\end{kakuro}
```




```
\begin{kakuro}[solution]
\framepuzzle
\kakurorow{5}{\Black,\KKR{23}{},
    \KKR{16}{},\KKR{10}{},\Black}
\kakurorow{4}{\KKR{}{14},9,1,4,
    \KKR{3}{}}
\kakurorow{3}{\KKR{}{16},6,5,3,2}
\kakurorow{2}{\KKR{}{14},8,3,2,1}
\kakurorow{1}{\Black,\KKR{}{8},7,
    1,\Black}
\end{kakuro}
```

`logicpuzzle` - Was gibt's noch?

Das war nur ein ganz kleiner Einblick in die Vielfältigen Möglichkeiten von `logicpuzzle`.
Weitere Puzzle, die gesetzt werden können:

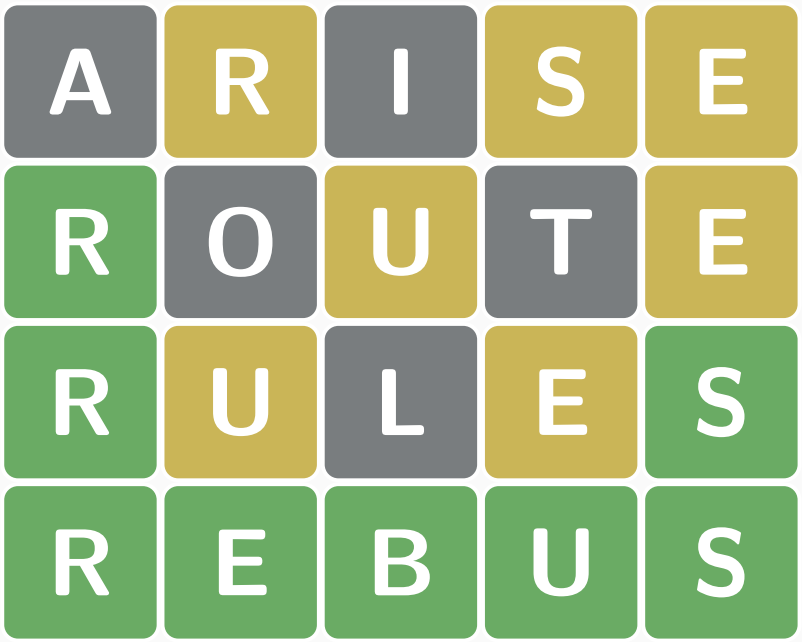
Bokkusu	Bridges	Chaos Sudoku	Four Winds
Hakyuu	Hitori	Kendoku	Killer Sudoku
Laser Beam	Magic Labyrinth	Magnets	Masyu
Battleship	Nonogram	Number Link	Resuko
Schatzsuche	Skyline	Slitherlink	Star Battle
Stars and Arrows	Sudoku	Sun and Moon	Tents and Trees
Tunnel	**2D-Sudoku**	**Minesweeper**	**Kakuro**

It's a digital world

`wordle` - Allgemein

- Autor: Cédric Pierquet, 2023
- Webbasiertes Spiel, bei dem man sechs Versuche hat, um ein Wort aus fünf Buchstaben zu erraten.
- jeder Buchstabe (Rateversuch) wird entsprechend markiert:
  - grün: Buchstabe ist vorhanden und an richtiger Position
  - gelb: Buchstabe ist vorhanden, aber nicht an der geratenen Position
  - grau: Buchstabe kommt überhaupt nicht vor
- das Paket `\usepackage{word}` baut solche Spiele nach mit verschiedenen Anpassungsmöglichkeiten
- Abhängigkeiten: `tikz, simplekv, xstring`

wordle - Beispiel




```
\begin{WordleGrid}{REBUS}
ARISE
ROUTE
RULES
REBUS
\end{WordleGrid}
```

`wordle` - Anpassungen

Option	Default	Wert	Bedeutung
colors			Farbe der Boxen (richtig, halbrichtig, falsch)
rounded	0.1	<dim>	Umrandung rund in mm
unit	1	<dim>	Breite Kästchen in cm
font			Schriftart für Buchstaben
ColorLetters	white		Farbe der Buchstaben
BorderColor	white	Randfarbe	
Letters	true	true|false	Buchstaben ausgeben
Thick	0.25	<dim>	Linienbreite Rand in mm

wordle - Beispiel 2


```
\begin{WordleGrid}
[Unit=0.5,Thickness=0.3,
BorderColor=black,
Colors={lightgray,orange,teal}]
{BURGHOF}
    GLAUBEN
    BARBARE
    BARGELD
    BERGHOF
    BURGHOF
\end{WordleGrid}
```

Kunterbunt und durcheinander ...

`TangramTikZ` – Allgemeines

- Tangram ist ein altes chinesisches Legespiel aus sieben Plättchen
- die Plättchen entstehen durch Zerschneiden eines Quadrats in zwei große Dreiecke, zwei kleine Dreiecke, ein mittelgroßes Dreieck, ein Quadrat und ein Parallelogramm
- aus diesen können zahllose Formen gelegt werden (Tiere, Pflanzen, Schiffe, Häuser ...)
- Paket zur Darstellung dieser Formen
- Autor: Cédric Pierquet, 2023
- Abhängigkeiten:
  `tikz, xstring, xparse, simplekv, listofitems`

Die sieben Plättchen sind folgendermaßen benannt:

`TangBigTri`	großes Dreieck (2x)
`TangMedTri`	mittleres Dreieck (1x)
`TangSmalTri`	kleines Dreieck (2x)
`TangSqua`	Quadrat (1x)
`TangPara`	Parallelogramm (1x)

TangramTikZ – vordefinierte Figuren

Für vordefinierte Figuren steht der Befehl
`\TangramTikZ[.1.]<.2.>{.3.}` zur Verfügung.
Im ersten Argument `[.1.]` können folgende Optionen verwendet werden:

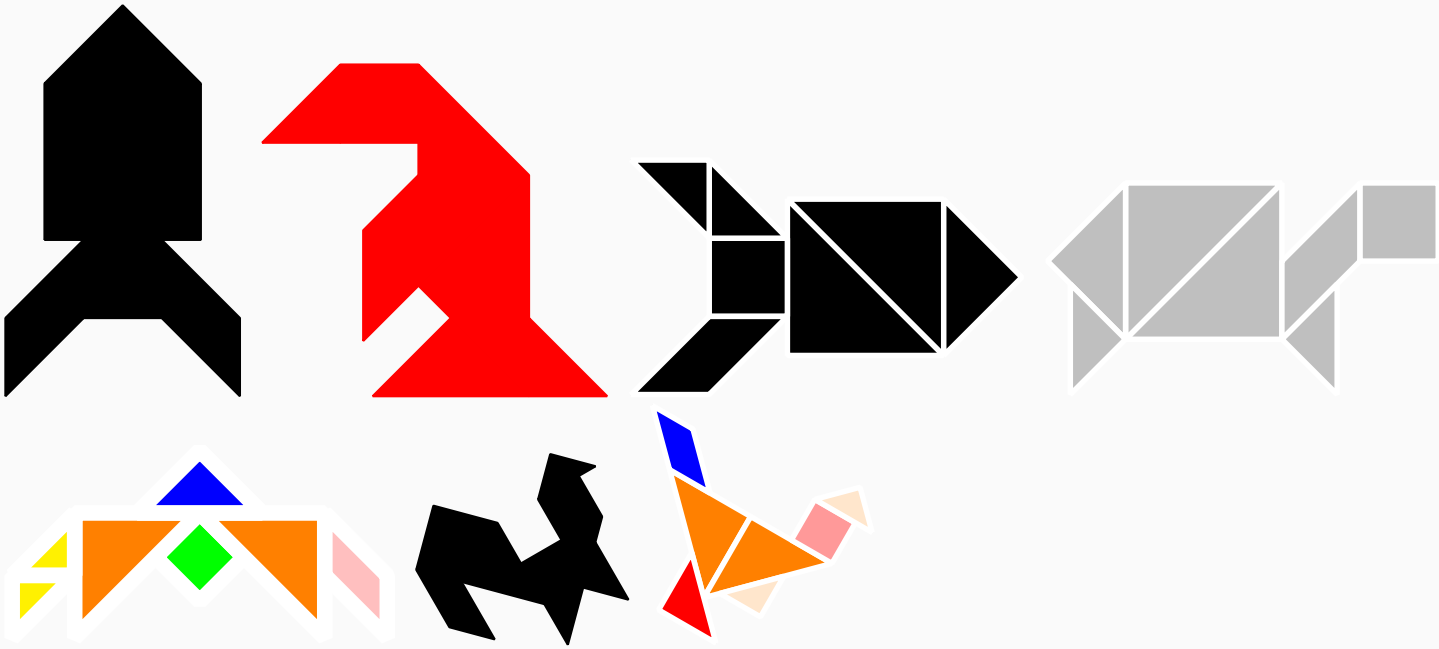
Option	**Default**	**Bezeichnung**
`Puzzle=true|false`	true	einfarbige Plättchen ohne Rand
`Correction=true|false`	false	einfarbige Plättchen mit Rand
`Color=`	black	Farbe für einfarbige Darstellung
`ColorCorrection=true|false`	false	farbige Plättchen mit Rand
`ColorList=`		Liste der Farben für die einzelnen Plättchen
`Sep=`	1pt	Dicke des Randes

`TangramTikZ` – vordefinierte Figuren

Im zweiten Argument `<.2.>` können TikZ-Kommandos mitgegeben werden, z.B. `unit=, rotation=, ...`
Das dritte Argument `{.3.}` enthält der Name der entstehenden Figur. Mögliche Namen sind:

Square	Pinguin	Boat	Home	FirTree	Cat	Swan
Pyramid	Duck	Rocket	Candle	Shirt	Fish	Sailboat
Kangaroo	Dog	Plane	Rabbit	Rooster	Jogger	Dancer
Camel	Flamingo	Heart	Giraffe	Horse	Goat	Lions
Factory	Angel	Tower	Ufo	Chicken	Turtle	Crab und Snail

TangramTikZ – vordefinierte Figuren



```
\TangramTikz<scale=0.5>{Rocket}
\TangramTikz[Color=red]<scale=0.5>{Pinguin}
\TangramTikz[Correction]<scale=0.5,rotate=-90>{Rocket}
\TangramTikz[Correction,Color=lightgray]<scale=0.5>{Turtle}
\TangramTikz[ColorCorrection,ColorList={orange,blue,yellow,green,pink},
     Sep=1mm]<scale=0.4>{Ufo}
\TangramTikz<scale=0.4,rotate=30>{Rooster}
\TangramTikz[ColorCorrection,ColorList={orange,red,orange!20,red!40,blue}]
     <scale=0.4,rotate=-30>{Chicken}
```

TangramTikZ – Figuren manuell erzeugen

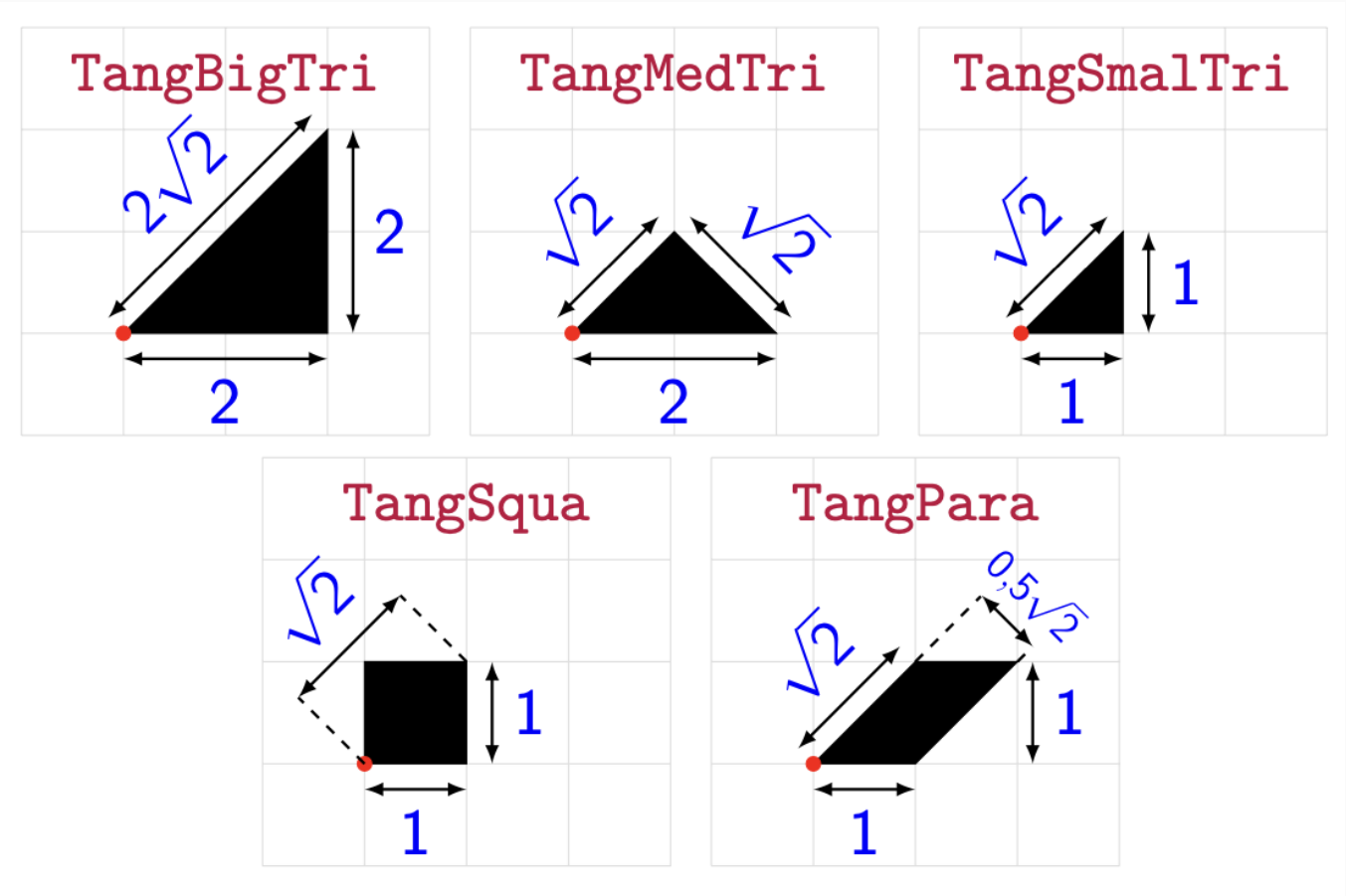
Für das manuelle Zeichnen von Tangram-Figuren steht die Umgebung `\begin{EnvTangramTikZ}...\end{EnvTangramTikZ}` zur Verfügung.
Innerhalb dieser Umgebung können mit `\PieceTangram[.1.]<.2.>(x,y){Name_Plättchen}` die verschiedenen Plättchen platziert werden.
.1. kann sein:

- `TangPuzz` Plättchen ohne Rand
- `TangSol` Plättchen mit weißem Rand

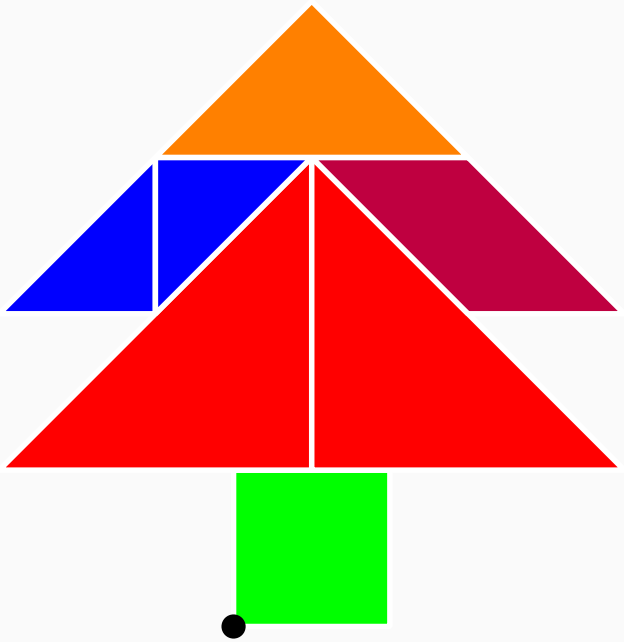
.2. kann sein:

- TikZ-Befehle wie `rotate`, `xscale`, `yscale`

Für das manuelle Erzeugen einer Figur muß man die Größe der einzelnen Plättchen kennen.



TangramTikZ – Figuren manuell erzeugen, Beispiel

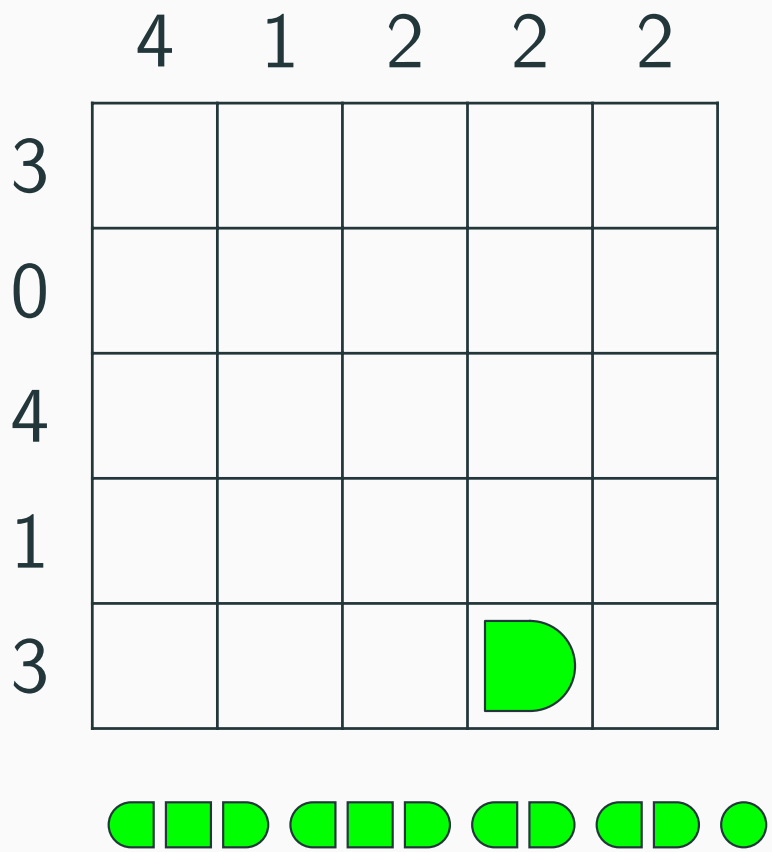


```
\begin{EnvTangramTikz}
\PieceTangram[TangSol={green}]({0},{0})
   {TangSqua}
\PieceTangram[TangSol={red}]({-
1.5},{1})
   {TangBigTri}
\PieceTangram[TangSol={red}]<rotate=-
90>
   ({0.5},{3}){TangBigTri}
\PieceTangram[TangSol={purple}]
   <xscale=-
1,rotate=0>({2.5},{2}){TangPara}
\PieceTangram[TangSol={blue}]
   ({-1.5},{2}){TangSmalTri}
\PieceTangram[TangSol={blue}]
   <xscale=-1,rotate=90>({-
0.5},{2}){TangSmalTri}
\PieceTangram[TangSol={orange}]
   ({-0.5},{3}){TangMedTri}
\end{EnvTangramTikz
```

`Battleship` – Allgemeines

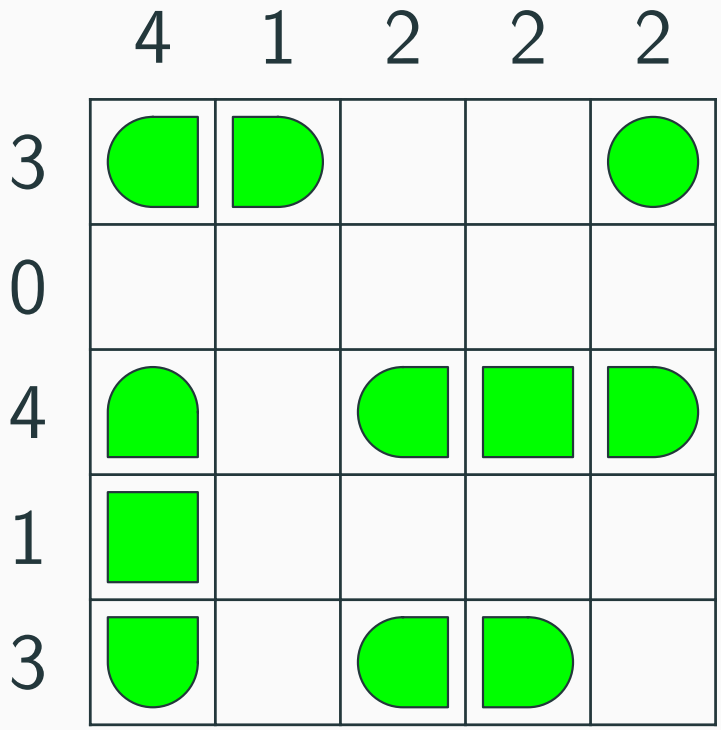
- Autor: Josef Kleber, 2013
- Schiffe versenken
- auch enthalten in `logicpuzzle`
- `\usepackage{battleship}`
- Regel 1: Schiffe berühren sich nicht, auch nicht an Ecken (kann aber gesetzt werden)
- Regel 2: keine diagonalen Schiffe

Battleship – Beispiel




```
\begin{battleship}
\placesegment{4}{1}{\ShipR}
\shipH{4,1,2,2,2}
\shipV{3,1,4,0,3}
\shipbox{3,3,2,2,1}
\end{battleship}
```

Battleship – Beispiel Lösung




```
\begin{battleship}
\placeship{V}{1}{1}{3}
\placeship{H}{1}{5}{2}
\placeship{H}{3}{1}{2}
\placeship{H}{3}{3}{3}
\placeship{H}{5}{5}{1}
\shipH{4,1,2,2,2}
\shipV{3,1,4,0,3}
\end{battleship}
```

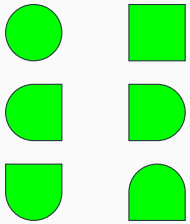
`Battleship` – Optionen

`\begin{battleship}[Optionen]` bietet u. a. folgende Mögichkeiten:

- `rows` Anzahl Reihen (default=5)
- `columns` Anzahl Spalten (default = 5)
- `shipcolor` Farbe der Schiffe (default = green)
- `scale` Größenskalierung (default =1)
- `bgcolor` Hintergrundfarbe (default = empty)
- `sbshipscale` Skalierung der Schiffe in der Box (default = 1)

`Battleship` – Befehle

- `placeship{.1.}{.2.}{.3.}{.4.}`Schiff platzieren,
  .1.: Vertikal oder horizontal .2. Spalte .3. Reihe .4. Länge
- `placesegment{.1.}{.2.}{.3.}` Schiffsegment platzieren:
  .1. Spalte .2. Reihe .3. Schiffsegment

  `\Ship`  `\ShipC`

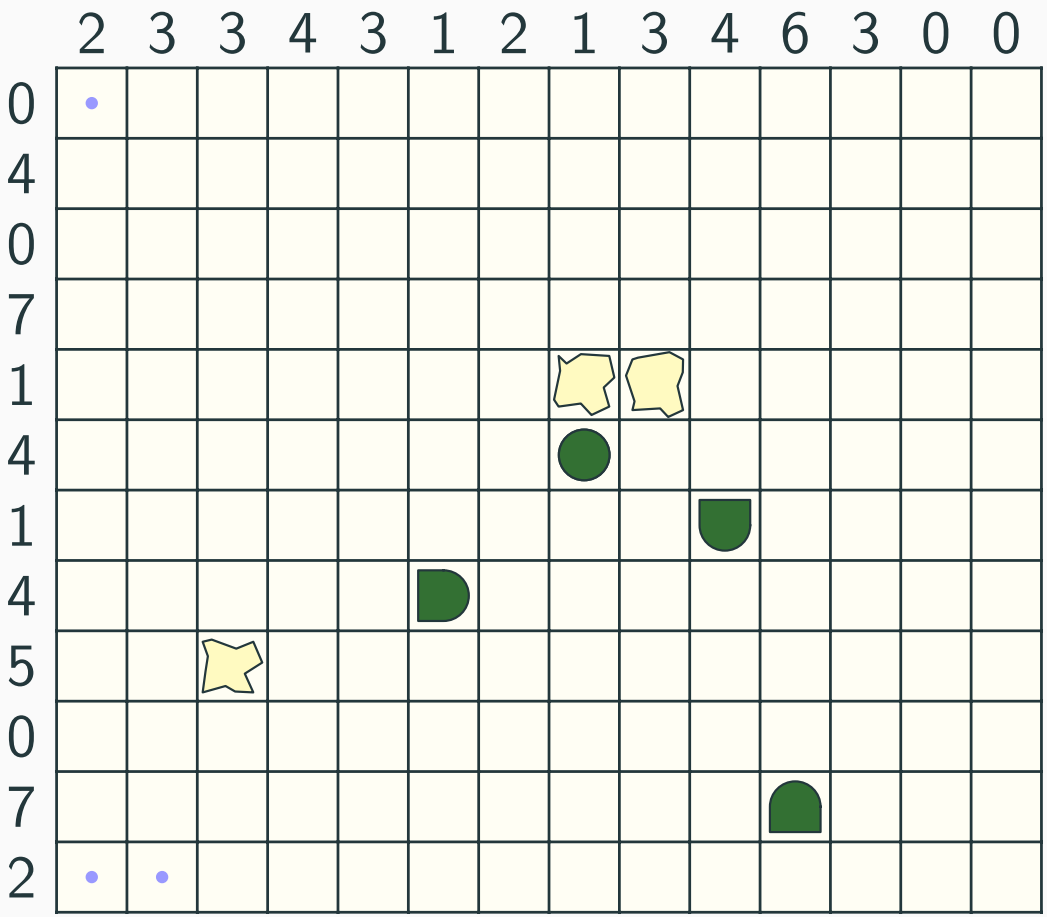
  `\ShipL` `\ShipR`

  `\ShipB` `\ShipT`
- `\placewater{.1.}{.2.}` • Wassermarkierungen platzieren
  .1. Spalte .2. Reihe
- `\placeisland{.1.}{.2.}`  , Inseln platzieren
  .1. Spalte .2. Reihe

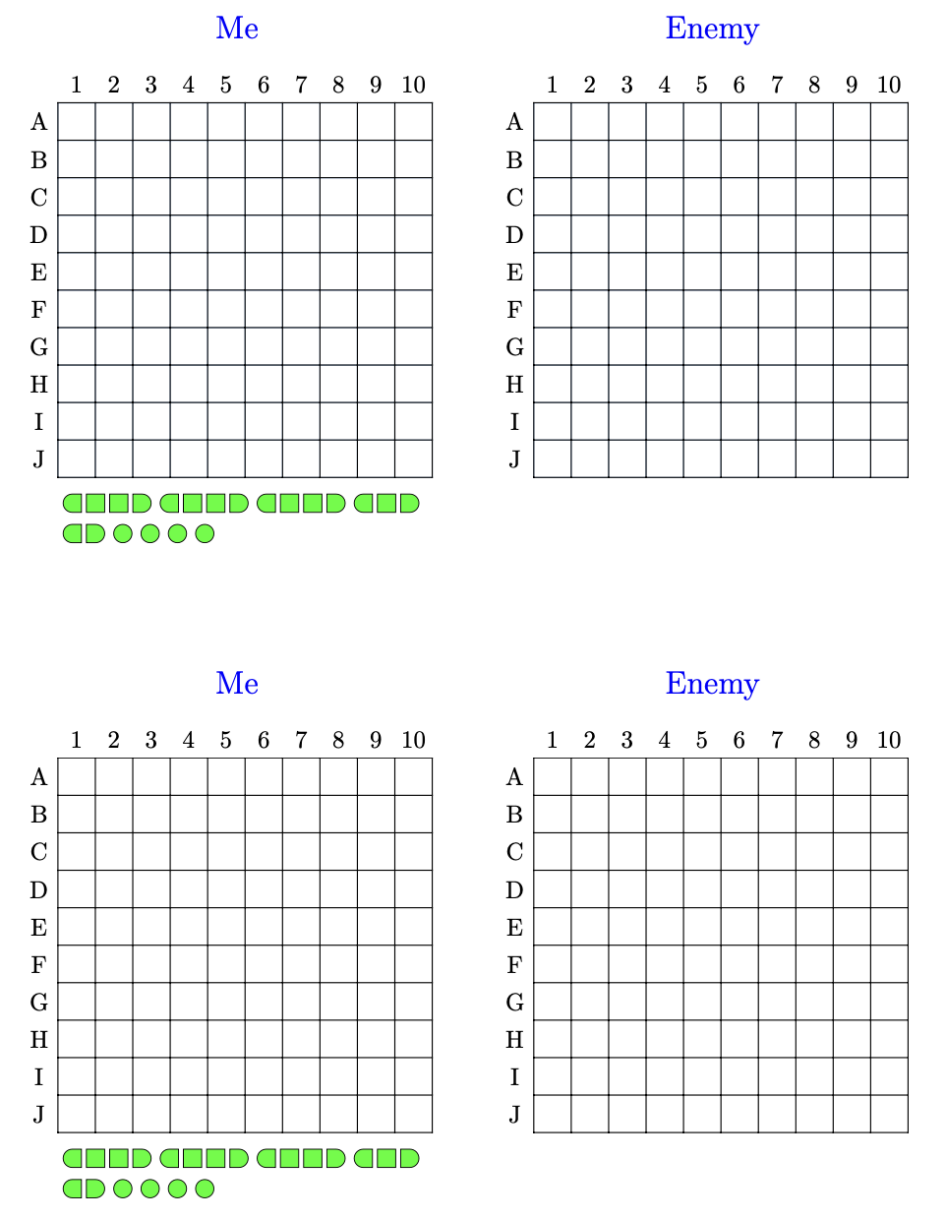
`Battleship` – Befehle

- `shipH` horizonale Nummern für Anzahl Schiffsanteile pro Spalte, v.l.n.r.
- `shipV` vertikale Nummern für Anzahl Schiffsanteile pro Reihe, v.o.n.u
- `definecounterstyle` für Definition eigener Stile
- `\classicgame{4,4,4,3,2,1,1,1,1}` erzeugt A4-Ausgabe für das klassische Spiel mit Stift und Papier. Argumente sind die verwendeten Schiffe.

Battleship – Beispiele 2



Battleship – Beispiele 3




```
\classicgame{4,4,4,3,2,1,1,1,1}
```

`nimsticks` – Allgemeines

Zum Spiel:

- Spiel für 2 Personen, bei dem nacheinander eine Anzahl von z. B. Streichhölzern weggenommen wird.
- Verschiedene Varianten: nur Hölzer aus einer Reihe, nur eine bestimmte Maximalanzahl, ...
- wer das letzte Hölzchen nimmt, gewinnt (Variante Misère: andersrum)

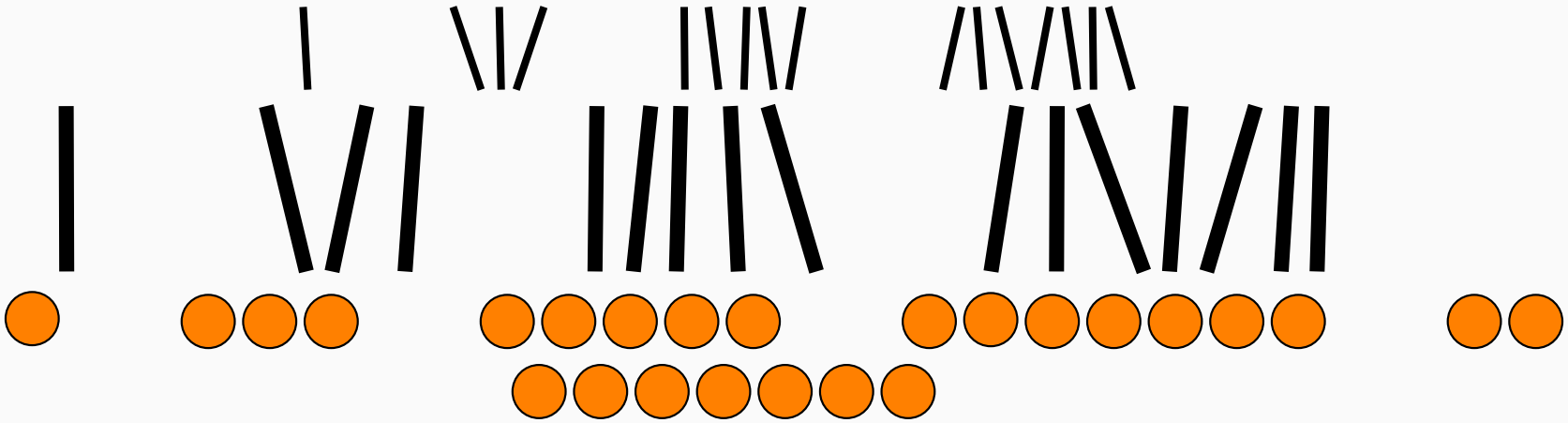
Zum Paket:

- Autor: Peter Rowlett, 2022
- ermöglicht die Darstellung von Nim- bzw. Misère-Spielen (Ausgangssituationen)
- `\usepackage{nimsticks}`
- basiert auf TikZ

nimsticks – Anwendung

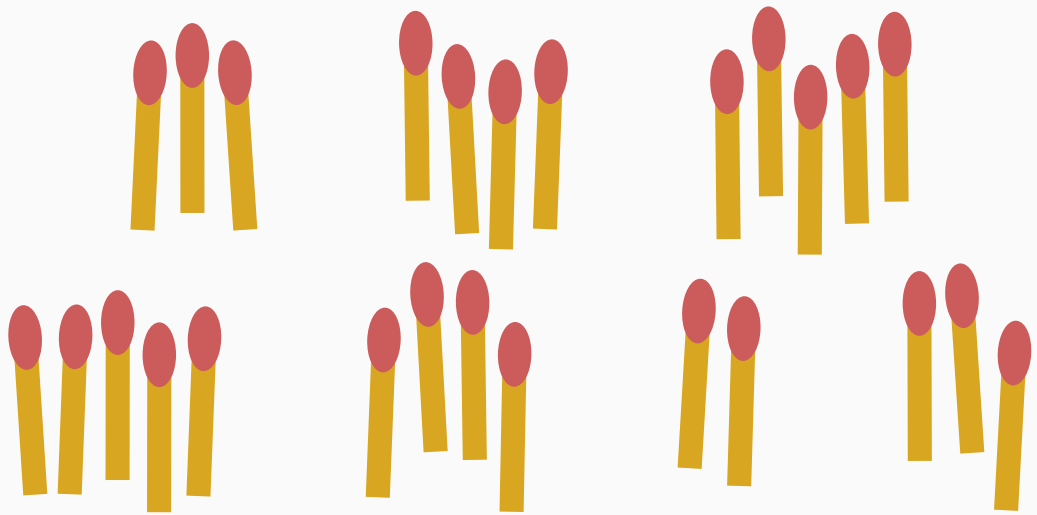
- `\nimgame{5,3,4}` erzeugt eine zufällige Anordnung von 5, 4 und 3 Strichen
- `\setnimstickolour{ }` Farbe der Striche
- Größe hängt von aktueller Textgröße ab (3.3ex)
- `setnimscale{<num>}` weitere Größendefinition
- Mit `\renewcommand{\onenimstick}{...}` können die Symbole angepasst werden
- Mit `\reinitrand[...]` kann die gleiche Zufallsdarstellung nochmal erzeugt werden

nimsticks – Beispiel



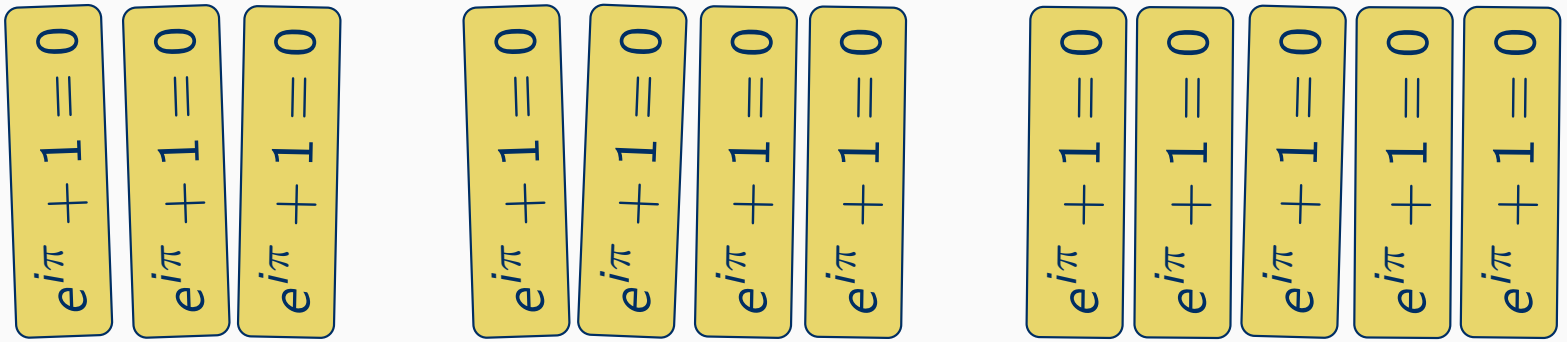
```
\nimgame{1,3,5,7}

\setnimscale{2}\nimgame{1,3,5,7}

\renewcommand{\onenimstick}{
\node[draw,black,fill=orange,circle] at (0,\lift) {};
}
\nimgame{1,3,5,7,9}
```

nimsticks – Beispiel



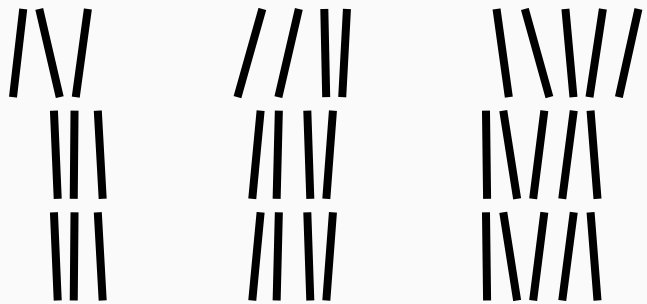
```
\definecolor{goldenrod}{rgb}{0.85, 0.65, 0.13}
\definecolor{indianred}{rgb}{0.8, 0.36, 0.36}
\renewcommand{\onenimstick}{
\draw[goldenrod,fill=goldenrod,rotate=\topx*40]
 (0,\lift) rectangle (0.14,1+\lift);
\draw[indianred,fill=indianred,rotate=\topx*40]
 (0.07,1+\lift) ellipse (0.1 and 0.2);
}

\nimgame{3,4,5}
\nimgame{5,4,2,3}
```

nimsticks – Beispiel




```
\definecolor{coolblack}{rgb}{0.0, 0.18, 0.39}
\definecolor{arylideyellow}{rgb}{0.91, 0.84, 0.42}
\renewcommand{\onenimstick}{
\node[draw,coolblack,fill=arylideyellow,
 rotate=90+\topx*60,anchor=north,rounded
corners=0.5ex] at (\topx,\botx) {\(e^{i\pi} + 1 = 0\)};
}
\nimgame{3,4,5}
```

nimsticks – Beispiel



```
\nimgame{3,4,5}

\reinitrand[first=-100,last=100,seed=1]
\nimgame{3,4,5}
\reinitrand[first=-100,last=100,seed=1]
\nimgame{3,4,5}
```

Anhang: Weitere Spieletypen (der Vollständigkeit halber)

LaTeX – Schach

Schach:

- appelt-chess
- bartel-chess-fonts (Fonts)
- bdfchess
- cchess (chinese chess)
- chess (Fonts)
- chessboard
- chessfss (Fonthandler)
- chessmin
- chess-problem-diagrams
- chinesechess
- chess-problem-diagrams
- enpassant (Fonts)
- MPchess
- schwalbe-chess
- skak
- SkakNew
- TeXmate
- xq (chinese chess)
- xskak (chinese chess)

LaTeX – Pen & Paper Rollenspiele, Bridge und Go

Pen & Paper Rollenspiele:

- `gamebook`
- `Gameboolib`
- `gurps`
- `rpg-module`

Bridge:

- `bridge`
- `bridge-plain`
- `OneDown`

GO:

- `context-sgf`
- `go`
- `igo (Fonts)`
- `metago`
- `psgo`
- `weiqi`

Kreuzworträtsel:

- `crossword`
- `crosswrd`
- `crw`
- `pas-crossword`