Das Paket robotkarol

Robot-Karol-Welten und Blockprogramme für Unterrichtsmaterial

Anselm Wagner*

31. Oktober 2026 Version 4.0

Zusammenfassung

Das Paket robotkarol zeichnet Welten und Blockprogramme im Stil von *Robot Karol Online* (`https://karol.arrrg.de/`) für Arbeitsblätter, Präsentationen und Prüfungen. Welten können als 3D-Schrägbild oder als 2D-Draufsicht ausgegeben werden; Farben und Formen sind der Weboberfläche nachempfunden. Einzelne Blöcke lassen sich als Inline-Blöcke in den Fließtext setzen, Aufgaben der Online-Umgebung mit `\karolonline` verlinken. Aufgaben aus Robot Karol Online lassen sich als JSON-Datei einlesen und Welten dorthin exportieren. Ein Druckmodus stellt alle Ansichten auf tonersparende Graustufen um. Das Paket benötigt LuaLaTeX[1] und expl3. *English:* draws worlds and Blockly-style block programs of the German teaching environment Robot Karol for worksheets; 3D oblique or 2D top view; single blocks can be set inline in running text, aligned to the surrounding baseline; tasks are imported from and exported to Robot Karol Online's JSON format; a toner-saving greyscale print mode; requires LuaLaTeX.

Inhaltsverzeichnis

1 Schnellstart **2**

2 Die Umgebung `karolwelt` **3**
2.1 Koordinatensystem . . . 3
2.2 Schlüssel der Umgebung . . . 3
2.3 Objekte . . . 3
2.4 Beispiel in beiden Ansichten . . . 4
2.5 Start- und Zielzustände . . . 4

3 Einzeilige Welten: `\karolzeile` **6**
3.1 Schlüssel . . . 7

4 Blockprogramme **7**
4.1 Bedingungen mit Anzahl . . . 8
4.2 Himmelsrichtungen . . . 8

5 Inline-Blöcke im Fließtext **9**

6 Text-Programme und Struktogramme **10**

*anselm.wagner@posteo.de

[1] Genauer: eine Engine mit `\directlua`; getestet mit LuaLaTeX.

7 Globale Einstellungen: \karolsetup **10**
7.1 Schriften . . . 10
7.2 Weltfarben . . . 11
7.3 Figur der Welten . . . 11
7.4 Blockfarben . . . 12
7.5 Druckmodus . . . 12

8 Aufgaben verlinken: \karolonline **13**
8.1 Mit oder ohne hyperref . . . 13

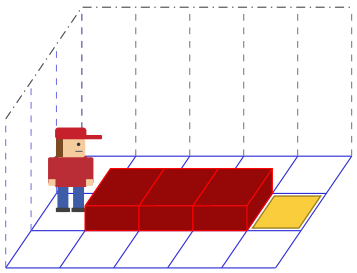
9 Aufgaben aus Robot Karol Online: JSON **13**
9.1 Einlesen und abfragen . . . 14
9.2 Eine Welt aus der Datei . . . 14
9.3 Drei Arten von Aufgaben . . . 15
9.4 Den Quelltext erzeugen . . . 15
9.5 Welten exportieren . . . 16
9.6 Ein Startprogramm mitgeben . . . 17
9.7 Was nicht übertragen wird . . . 17
9.8 Große Welten . . . 17

10 Hinweise und Migration **17**

1 Schnellstart

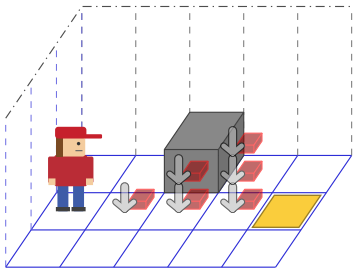
```
\usepackage{robotkarol}   % Kompilierung mit LuaLaTeX
```

```
\begin{karolwelt}[breite=5, laenge=3]
  \karol[x=1, y=2, richtung=osten]
  \ziegel[x=2, y=2, bisx=4]
  \marke[x=5, y=2]
\end{karolwelt}
```



Eine in Robot Karol Online gebaute Aufgabe (*Aufgabe freigeben* → *Als JSON herunterladen*) kann direkt in LaTeX gesetzt werden – hier der erste Auftrag der Datei, Start und Ziel in einem Bild:

```
\karoljsonlesen[name=treppen]{treppen.json}
\karolweltaus[aufgabe=1]{treppen}
```



Umgekehrt wird jede Welt mit dem Schlüssel `export=blatt/1`, `export=blatt/2`, … zu einem Auftrag der Sammlung `blatt`, und

```
\karoljsonschreiben{blatt}{blatt.json}
```

schreibt sie am Dokumentende als JSON-Datei, die Karol Online auf der Startseite mit *Aufgabe aus Datei laden* öffnet. Für die vollständige Beschreibung siehe Abschnitt 9.

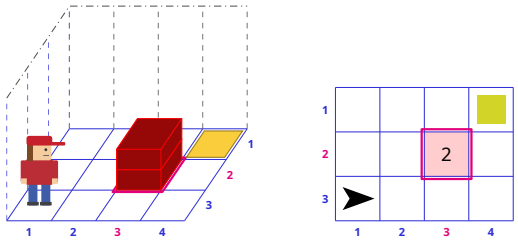
2 Die Umgebung `karolwelt`

```
\begin{karolwelt}[<Schlüssel>]
  <Objekte>
\end{karolwelt}
```

2.1 Koordinatensystem

Die Koordinaten folgen Robot Karol Online: `x` zählt die *Spalten* von links (1 bis `breite`), `y` die *Zeilen von hinten* (1 bis `laenge`). In der 2D-Draufsicht ist die Zeile $y = 1$ folglich die *oberste* Zeile; in der 3D-Ansicht liegt sie am weitesten hinten. `hoehe` bezeichnet wie in Karol Online die *Raumhöhe* (Anzahl Ziegellagen bis zum oberen Rahmen) und wirkt nur auf den 3D-Rahmen.

Mit `nummern` schreibt die Umgebung die Koordinaten an den Rand: die Spaltennummern vor die vorderste Gitterkante, die Zeilennummern in der Draufsicht links daneben. Im Schrägbild stehen die Zeilennummern *rechts* – links lägen sie hinter den Stapeln der ersten Spalte und würden verdeckt werden. Gezählt wird ab 1, es stehen also genau die Zahlen am Rand, die man in `x` und `y` einträgt. Eine hervorgehobene Zelle färbt ihre beiden Nummern mit.



2.2 Schlüssel der Umgebung

`breite`	5	Anzahl Spalten
`laenge`	5	Anzahl Zeilen
`hoehe`	6	Raumhöhe des 3D-Rahmens in Ziegellagen
`ansicht`	`3d`	`3d` (Schrägbild) oder `2d` (Draufsicht)
`zustand`	`beides`	`start`, `ziel` oder `beides` (Abschnitt 2.5)
`auftrag`	`keiner`	`aufbauen` oder `aufraeumen`: die Start- bzw. Zielwelt ist leer (Abschnitt 2.5)
`nummern`	`false`	Koordinaten an den Rand schreiben
`rahmen`	`true`	Raumrahmen zeichnen (nur 3D)
`hervorheben`		Liste hervorzuhebender Zellen, etwa `{2/2, 4/1}`
`skalierung`	1	Skalierungsfaktor des Bildes
`titel`		Beschriftung über dem Bild
`hinweis`		kursiver Hinweis unter dem Bild

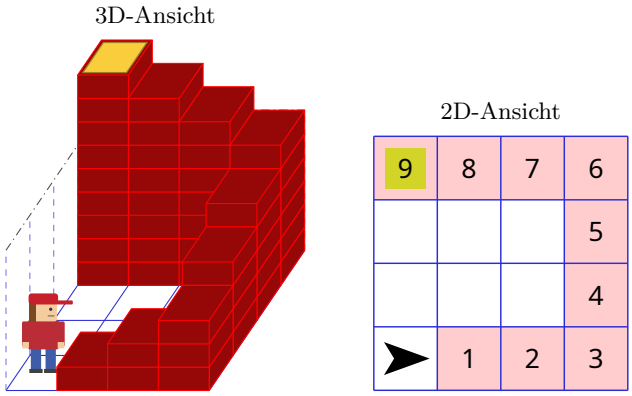
2.3 Objekte

Alle Objekte werden über Schlüssel platziert. `bisx`/`bisy` (auch `bisX`/`bisY`) füllen rechteckige Bereiche.

```
\karol[x=1, y=1, richtung=osten]  % genau einmal je Welt
\ziegel[x=2, y=3, anzahl=4]       % Stapel aus 4 Ziegeln
\ziegel[x=1, y=5, bisx=5]         % Reihe einzelner Ziegel
\marke[x=3, y=3]                  % gelbe Marke
\marke[x=5, y=1, farbe=cyan!60]   % Marke in eigener Farbe
\quader[x=4, y=1, bisy=5]         % Quaderwand; \wand = Synonym
```

`richtung` akzeptiert `osten`, `westen`, `norden` und `sueden` sowie die Synonyme `rechts`, `links`, `hinten` bzw. `oben` und `vorn`, `vorne` bzw. `unten` (Vorgabe: `sueden`). Marken liegen stets *oben auf* dem Ziegelstapel ihrer Zelle; Karol steht auf Stapel und Marke.

2.4 Beispiel in beiden Ansichten



In der 2D-Ansicht zeigen Zahlen die Stapelhöhe; Ziegelzellen sind rosa, Marken eine einfarbige Fläche, die kleiner als das Feld ist — so bleibt der Ziegel ringsum sichtbar und „Marke auf Ziegel“ erkennbar. Quader sind grau, Karol ist der schwarze Richtungskeil.

2.5 Start- und Zielzustände

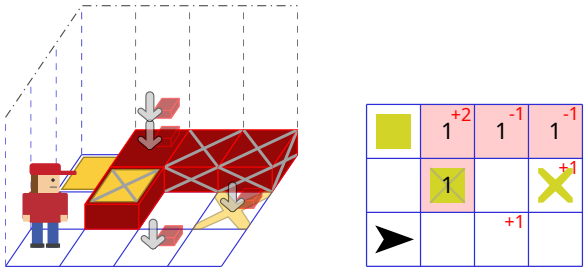
Aufgaben in Robot Karol Online zeigen oft in *einem* Bild, was sich ändern soll: Ziegel und Marken, die bleiben, sehen aus wie immer; was verschwinden soll, ist durchgekreuzt; was entstehen soll, erscheint blass mit einem Pfeil bzw. als X-Fläche. Dasselbe leisten die Schlüssel `start` und `ziel` an den Objekten.

`start`	`weg`	nur im Startzustand: soll verschwinden
`ziel`	`neu`	nur im Zielzustand: soll entstehen
`startanzahl`		Ziegel im Startzustand
`zielanzahl`		Ziegel im Zielzustand

Wer beides oder keines von beiden nennt, bekommt ein Objekt, das in beiden Zuständen vorkommt — unverändert also. Statt `start` und `ziel` darf man die Befehlsnamen von Robot Karol schreiben: `hinlegen` und `aufheben` für Ziegel, `setzen` und `loeschen` für Marken. `startanzahl` und `zielanzahl` gibt es auch mit Bindestrich (`start-anzahl`, `ziel-anzahl`). Die Reihenfolge der Schlüssel spielt keine Rolle: `[anzahl=3, ziel]` und `[ziel, anzahl=3]` bedeuten dasselbe.

```
\begin{karolwelt}[breite=4, laenge=3]
  \karol[x=1, y=3, richtung=osten]
  \ziegel[x=2, y=2]                    % bleibt liegen
  \ziegel[x=4, y=1, start]             % nur im Start: verschwindet
  \ziegel[x=3, y=3, ziel]              % nur im Ziel: entsteht
```

```
  \ziegel[x=3, y=1, aufheben]                    % dasselbe wie start
  \ziegel[x=4, y=2, hinlegen]                    % dasselbe wie ziel
  \ziegel[x=2, y=1, startanzahl=1, zielanzahl=3] % zwei kommen dazu
  \marke[x=1, y=1]                               % bleibt liegen
  \marke[x=2, y=2, loeschen]                     % soll geloescht werden
  \marke[x=4, y=2, setzen]                       % soll gesetzt werden
\end{karolwelt}
```



soll hingelegt werden soll aufgehoben werden soll gesetzt werden
soll gelöscht werden

In der 2D-Draufsicht gibt es keine Kreuze und keine Pfeile: dort steht die Stapelzahl des Startzustands mittig und die Veränderung als rote Zahl in der Ecke.

`\karolzustandslegende` erklärt nur die vier Zustandszeichen; die gewöhnlichen Symbole erklärt weiterhin `\karollegende` (Abschnitt 3). Eine eigene Legende setzt man aus Einträgen der Form

```
\karolzustandslegendenitem{Objekt}{Text}
```

zusammen: das erste Argument ist der Rumpf einer Welt aus einem einzigen Feld (Koordinaten `x=1, y=1`), das zweite die Beschriftung. Zwischen die Einträge gehört `\karollegendentrenner`:

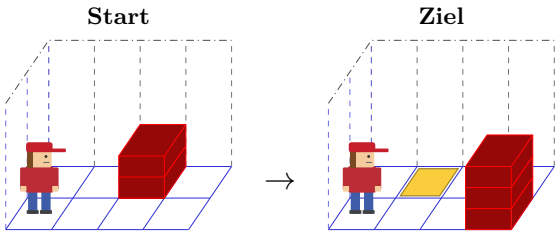
```
\karolzustandslegendenitem{\ziegel[x=1, y=1, anzahl=2, hinlegen]}
  {zwei Ziegel hinlegen}\karollegendentrenner
\karolzustandslegendenitem{\marke[x=1, y=1, setzen]}
  {Marke setzen}
```



Der Schlüssel `zustand` wählt, was gezeigt wird. `beides` ist wie in Karol Online und liefert das Bild oben; `start` und `ziel` zeigen je *eine* Welt in der gewohnten Optik — eine Welt ohne `start`- und `ziel`-Angaben sieht in allen drei Fällen gleich aus.

Für „vorher / nachher“ nebeneinander gibt es die Umgebung `karolweltpaar`. Ihr Rumpf wird zweimal gezeichnet, links im Zustand `links` (Vorgabe `start`), rechts im Zustand `rechts` (Vorgabe `ziel`); dazu `trenner`, `titellinks` und `titelrechts` (Vorgaben: ein Pfeil sowie **Start** und **Ziel**). Alle übrigen Schlüssel reicht sie an beide Welten durch.

```
\begin{karolweltpaar}[breite=4, laenge=2]
  \karol[x=1, y=2, richtung=osten]
  \ziegel[x=3, y=1, anzahl=2, aufheben]
  \ziegel[x=4, y=2, anzahl=3, hinlegen]
  \marke[x=2, y=1, setzen]
\end{karolweltpaar}
```



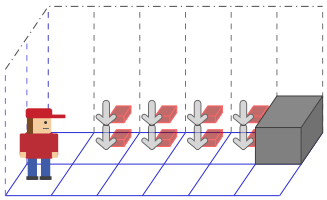
Soll am Ende *alles* weggeräumt sein, wäre an jedem Objekt ein `weg` oder `start` nötig. Umgekehrt bräuchte beim Bau einer Mauer jeder Ziegel ein `neu` oder `ziel`. Dafür gibt es den Schlüssel `auftrag`. Er legt fest, wozu ein Objekt *ohne* eigene Angabe gehört:

`keiner`	zu Start und Ziel (Vorgabe)
`aufraeumen`	nur zum Start – die Zielwelt ist leer (auch `aufräumen`)
`aufbauen`	nur zum Ziel – die Startwelt ist leer

Wer an einem Objekt `start`, `ziel` oder eine Anzahl angibt, behält seine Angabe. Quader und Karol betrifft `auftrag` nicht: Wände stehen in beiden Zuständen.

```
\begin{karolwelt}[breite=6, laenge=3, auftrag=aufraeumen]
  \ziegel[x=2, y=2, bisx=5, anzahl=2]  % alles ohne Angabe = nur Start
  \marke[x=3, y=3, start, ziel]        % Ausnahme: diese Marke bleibt
\end{karolwelt}
```

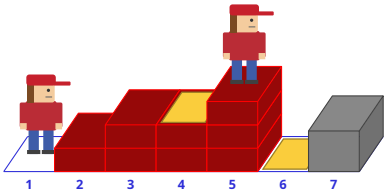
```
\begin{karolwelt}[breite=6, laenge=2, auftrag=aufbauen]
  \karol[x=1, y=2, richtung=osten]
  \ziegel[x=2, y=1, bisx=5, anzahl=2]  % die Mauer entsteht erst
  \quader[x=6, y=1]                    % Wand: steht schon
\end{karolwelt}
```



Nicht ausdrückbar sind eine Zielposition für Karol und ein Farbwechsel einer Marke: `farbe` gilt für beide Zustände, und Robot Karol Online vergleicht die Figur beim Prüfen ohnehin nicht. Quader kennen keinen Zielzustand — Karol kann sie weder setzen noch räumen.

3 Einzeilige Welten: `\karolzeile`

Für schnelle eindimensionale Welten gibt es die kompakte Zeichenketten-Notation (bis Version 2.x hieß das Kommando `\karolwelt`). Gezeichnet wird mit denselben Bausteinen wie in der Umgebung `karolwelt`: gleiche Projektion, gleiches Bodengitter, gleiche Ziegel, Marken, Quader und derselbe Avatar. Eine Zeilen-Welt ist damit optisch nichts anderes als eine `karolwelt` mit `laenge`$=1$ – nur die Eingabe ist kürzer.



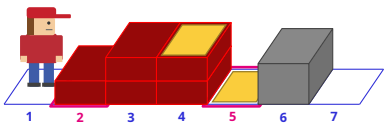

```
\karolzeile{> Z 2 {2M} {3>} M Q}
```

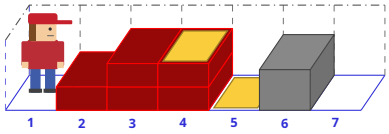
`> < ^ v`	Karol mit Blickrichtung
`.`	leeres Feld
`Z`, `2` … `9`	Ziegel bzw. Stapel
`M`, `Q`	Marke, Quader
`{...}`	mehrere Objekte auf einem Feld, von unten nach oben: `{M>}`, `{2M}`, `{3>}`

3.1 Schlüssel

`skalierung`	1	Skalierungsfaktor des Bildes
`nummern`	`true`	Feldnummern anzeigen (ab 1)
`titel`	—	Zeile über dem Bild
`hinweis`	—	kursive Zeile unter dem Bild
`hervorheben`	—	Komma-Liste von Feldnummern
`rahmen`	`false`	Raumrahmen wie in `karolwelt`
`hoehe`	3	Raumhöhe des Rahmens in Ziegellagen

Die **Feldnummern** stehen vor der vordersten Gitterkante und nicht auf dem Feld: im Schrägbild würde sie sonst jeder Ziegelstapel verdecken. Sie zählen wie `x` in `karolwelt` ab 1; `hervorheben` nimmt dieselben Nummern. `hervorheben` zeichnet einen magentafarbenen Rahmen knapp außerhalb des Feldes und färbt dessen Nummer mit – ein roter Rahmen wäre neben den reinroten Ziegelkanten nicht zu erkennen (Farbe: `hervorfarbe`). Den Raumrahmen zeichnet `\karolzeile` nur auf Wunsch; für eine einzelne Reihe ist er meist nicht nötig.





Dazu gibt es

`\karolpaar[keys][keys2]{vorher}{nachher}`	Vorher-nachher-Paar; `keys` gilt für beide Bilder, `keys2` nur für „nachher”
`\karolgitter[keys]{n}`	leeres Gitter aus n Feldern
`\karollegende`	Standardlegende der Weltsymbole
`\karollegendenitem{Welt}{Text}`	einzelner Legendeneintrag
`\karollegendentrenner`	Abstand zwischen zwei Einträgen; dehnbar, die Legende darf dort umbrechen

4 Blockprogramme

Die Umgebung `karolbloecke` stapelt Blockly-Blöcke; Farben und Formen entsprechen der Karol-Online-Oberfläche.

```
\begin{karolbloecke}
  \bwiederholesolange{NichtIstWand}{%
    \bwenndann{IstZiegel}{%
      \bbefehl[1]{Aufheben}}
    \bbefehl[1]{Schritt}}
\end{karolbloecke}
```



`\bbefehl[n]{Name}`	Anweisung (violett); optionale Zahl als Pille
`\bwiederholemal{n}{...}`	Zählschleife
`\bwiederholesolange{Bed}{...}`	kopfgesteuerte Schleife
`\bwiederholeimmer{...}`	Endlosschleife
`\bwenndann{Bed}{...}`	einseitige Verzweigung
`\bwenndannsonst{Bed}{...}{...}`	zweiseitige Verzweigung
`\bbedingung[n]{IstZiegel}`	Bedingung mit linkem Stecker; auch im Fließtext; optionale Anzahl als Pille; leeres Argument = weißer Leersockel
`\bist{Norden}`	Himmelsrichtung als Auswahlmenü
`\bnichtist{Norden}`	dasselbe verneint
`\bluecke[Breite]`	gestrichelter Geisterblock
`\bkommentar{Text}`	Kommentar (khaki, `//`)
`\baufruf[n]{Name}`	Aufruf einer eigenen Anweisung (rot)
`\banweisung{Name}{...}`	Definition einer eigenen Anweisung
`\bhauptprogramm`	Hauptprogramm-Hut

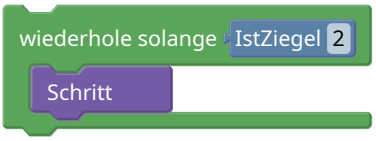
4.1 Bedingungen mit Anzahl

In Robot Karol Online nehmen `IstZiegel` und `NichtIstZiegel` eine Anzahl entgegen – im Blockeditor ein Zahlenfeld, im Programmtext `IstZiegel(2)`. Im Paket steht sie als optionales Argument vor der Bedingung; die Pille trägt kein Auswahldreieck, denn im Original ist es ein Zahlenfeld und kein Menü.



Die Zahl gehört der Bedingung, nicht dem Block. Die Kontrollblöcke reichen sie als erstes, optionales Argument an ihre Bedingung durch – die Pille bleibt darum auch im grünen `\bwenndann` stahlblau.

```
\bwiederholesolange[2]{IstZiegel}{%
  \bbefehl{Schritt}}
```



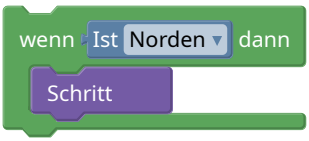
4.2 Himmelsrichtungen

Die acht Blöcke `Ist` / `NichtIst` zeigen die Richtung in Robot Karol Online nicht als Text, sondern als Auswahlmenü: eine helle Pille mit Auswahldreieck. `\bist{...}` und `\bnichtist{...}` bilden das nach und nehmen die Richtung als Argument.

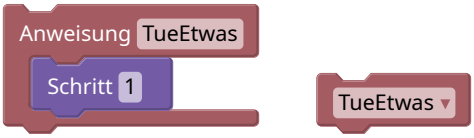


Für sich allein ergeben sie ein vollständiges Bedingungsplättchen. Als Argument eines Kontrollblocks steuern sie nur dessen Beschriftung bei – das Plättchen baut dort wie gewohnt `\bbedingung`:

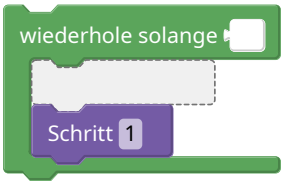
```
\bwenndann{\bist{Norden}}{%
  \bbefehl{Schritt}}
```



Dasselbe Auswahldreieck trägt der Aufrufblock `\baufruf`: dort wählt man in der Oberfläche die aufzurufende Anweisung aus. Die Definition `\banweisung` zeigt den Namen dagegen in einer Pille ohne Dreieck.



Beispiel Lückenaufgabe mit Leersockel und Geisterblock:



5 Inline-Blöcke im Fließtext

Die `\i...`-Kommandos setzen genau *einen* Block in die laufende Zeile. Die Textgrundlinie des Blocks liegt dabei exakt auf der Grundlinie des umgebenden Textes, und die Beschriftung übernimmt dessen Schriftgröße; Kerbe, Nase und Innenabstände sind flacher als im Stapel, damit die Blockhöhe der Zeilenhöhe nahekommt.

```
Mit \iwenndann{IstZiegel} prüft Karol, ob ein Ziegel vor ihm liegt;
\iwiederholemal{5} wiederholt fünfmal, \ibefehl{Schritt} geht weiter.
```

Mit wenn IstZiegel dann prüft Karol, ob ein Ziegel vor ihm liegt; wiederhole 5 mal wiederholt fünfmal, Schritt geht weiter.

Blöcke, die im Stapel mehrzeilig wären – Schleifen, Verzweigungen und eigene Anweisungen –, erscheinen inline **nur mit ihrem Kopf**. `\iwenndann{IstZiegel}` ergibt also den Block wenn IstZiegel dann ohne Rumpf. Für echte Rümpfe bleibt die Umgebung `karolbloecke` zuständig.

`\ibefehl[n]{Name}`	Anweisung Schritt 3
`\iaufruf[n]{Name}`	Aufruf einer eigenen Anweisung
`\ianweisung{Name}`	Kopf einer eigenen Anweisung
`\ikommentar{Text}`	Kommentar
`\ihauptprogramm`	Hauptprogramm-Hut
`\iwiederholemal{n}`	Kopf der Zählschleife wiederhole 5 mal
`\iwiederholesolange[n]{Bed}`	Kopf der kopfgesteuerten Schleife
`\iwiederholeimmer`	Kopf der Endlosschleife
`\iwenndann[n]{Bed}`	Kopf der Verzweigung
`\isonst`	sonst-Balken sonst
`\ibedingung[n]{Bed}`	Bedingung IstMarke, mit Anzahl IstZiegel 2
`\iist{Norden}`	Himmelsrichtung Ist Norden
`\inichtist{Norden}`	verneint NichtIst Osten
`\iluecke[Breite]`	Geisterblock

Die `\b...`-Kommandos erkennen den Fließtext selbst und setzen dort ebenfalls inline; die `\i...`-Fassungen ersparen nur die leeren Rumpf-Argumente und erzwingen den Modus auch im Stapel.

Inline-Blöcke sind breite, unzerbrechliche Kästen. In schmalen Spalten kann TEX sie nicht immer umbrechen – dann hilft `\sloppy` oder eine höhere `\tolerance` für den Absatz.

6 Text-Programme und Struktogramme

Die Umgebung `karolcode` setzt Karol-Quelltext mit Syntaxhervorhebung (listings); `\kb{...}` und `\kw{...}` formatieren Bezeichner und Schlüsselwörter im Fließtext. Für Struktogramme lädt das Paket struktex und ergänzt `\leeranweisung` sowie `\stklucke[Breite]`.

7 Globale Einstellungen: `\karolsetup`

```
\karolsetup{feldbreite=8.4mm, befehlfarbe={RGB}{116,91,165}, ...}
```

7.1 Schriften

Die Oberfläche von Robot Karol Online setzt ihre Beschriftungen in *Noto Sans* und den Programmtext des Editors in *Hack*. Das Paket tut das seit v3.6 ebenso: Beschriftungen der Blöcke, Feldnummern und Stapelzahlen erscheinen in Noto Sans, `karolcode` sowie `\kb` und `\kw` in Hack. Dafür lädt das Paket fontspec. Noto Sans liegt in TEX Live; Hack nicht – ist es im System installiert, wird es benutzt, sonst DejaVu Sans Mono, aus dessen Vorläufer Hack hervorgegangen ist. Welche der beiden es geworden ist, steht in der `.log`-Datei.

Wer die Schriften des eigenen Dokuments übernehmen will – oder fontspec nicht geladen haben möchte –, lädt das Paket mit

```
\usepackage[schrift=dokument]{robotkarol}
```

Dann benutzt das Paket `\sffamily` und `\ttfamily` des Dokuments; die Ausgabe ist Zeichen für Zeichen dieselbe wie bis v3.5. Umschalten geht auch mitten im Dokument, in einer Gruppe nur dort:

`schrift=karol`	Noto Sans und Hack (Vorgabe)
`schrift=dokument`	`\sffamily` und `\ttfamily` des Dokuments
`textschrift={...}`	eigene Schrift für Beschriftungen
`codeschrift={...}`	eigene Schrift für Programmtext

Die beiden letzten Schlüssel nehmen den Umschaltbefehl selbst:

```
\karolsetup{textschrift={\fontspec{Fira Sans}}}
```

Ein `\karolsetup{schrift=karol}` in einem Dokument, das mit `schrift=dokument` geladen wurde, ist wirkungslos (mit Warnung) – die Schriften sind dann gar nicht geladen.

Titel und Hinweis einer Welt bleiben absichtlich in der Schrift des Dokuments: sie sind Text des Dokuments, keine Beschriftung der Zeichnung. Für eigenen Text in der Paketschrift gibt es die Umschalter `\karolschrift` und `\karolcodeschrift`.

7.2 Weltfarben

Sie wirken auf `\karolzeile` *und* die Umgebung `karolwelt`, die dieselben Bausteine zeichnen.

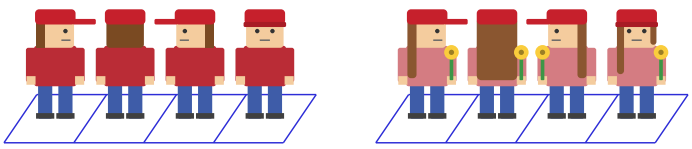
`feldbreite`	8.4 mm	Zell-/Feldbreite aller Welten
`ziegelfarbe`	reines Rot	Lagenkante; Körper daraus abgeleitet (`!59!black`)
`ziegelkoerperfarbe`	150,8,8	nur die Ziegelflächen
`ziegelkantenfarbe`	255,0,0	nur die Lagenkanten
`markenfarbe`	250,205,60	Markenplatte
`quaderfarbe`	128,128,128	Quader / Wand
`bodenfarbe`, `gitterfarbe`	45,45,215	Bodengitter
`hervorfarbe`	226,0,122	`hervorheben`

7.3 Figur der Welten

Das Paket bringt zwei Figuren mit: `karol` (Vorgabe) und `emma` mit langem Haar und einer Blume in der Hand. Umgeschaltet wird mit

```
\karolfigur{emma}          % gleichwertig: \karolsetup{figur=emma}
```

Die Einstellung wirkt auf `\karolzeile` *und* die Umgebung `karolwelt`; in einer Gruppe gesetzt gilt sie nur dort.



karol emma

Emma ist aus derselben Figur gebaut wie Karol. Ihre Farben heißen `karolEmmaShirt`, `karolEmmaHaar`, `karolEmmaBluete` und `karolEmmaStiel` und lassen sich mit `\colorlet` ändern.

7.4 Blockfarben

`befehlfarbe`	116,91,165	Anweisungsblöcke
`schleifenfarbe`, `verzweigungfarbe`	91,165,91	grüne Kontrollblöcke
`bedingungfarbe`	91,128,165	Bedingungen
`anweisungfarbe`	165,91,98	eigene Anweisungen
`kommentarfarbe`	165,150,91	Kommentare
`hutfarbe`	130,123,117	Hauptprogramm-Hut

Die Blockfarben sind aus der Karol-Online-Oberfläche übernommen.

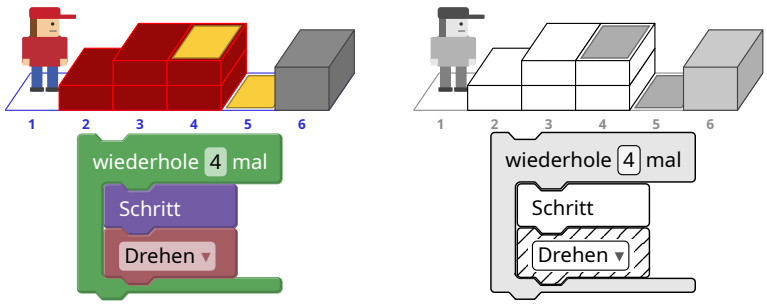
7.5 Druckmodus

Arbeitsblätter werden selten in Farbe vervielfältigt. Der Druckmodus zeichnet deshalb alle Ansichten – Welten, Zeilen-Welten und Blockprogramme – in Graustufen:

```
\usepackage[druck]{robotkarol}    % das ganze Dokument
\karolsetup{druck=true}           % ab hier; in einer Gruppe nur dort
\karolsetup{druck=false}          % zurueck zur Farbdarstellung
```

Der Leitgedanke ist *nicht* das absolute Minimum an Toner zu verwenden, sondern: so wenig wie möglich, so viel wie nötig. Deshalb tragen die Konturen die Information, und die Flächen bleiben so hell wie möglich:

- Ziegel sind *weiß* mit schwarzen Kanten – der Stapel bleibt seitlich abzählbar, kostet aber praktisch keine Farbe.
- Der Quader ist die einzige große graue Fläche der Welt: er teilt seine Form mit dem Ziegel und muss sich von ihm abheben.
- Die Marke ist mittelgrau. Sie ist klein, und sie liegt oft auf einem Ziegel, von dem sie sich absetzen muss.
- Bei den Blöcken bekommt der häufigste – `\bbefehl` – die weiße Fläche; die formgleichen `\baufruf` und `\banweisung` bekommen eine **45-Grad-Schraffur** auf weißem Grund. Ein bloß dunkleres Grau trug diesen Unterschied nicht – beide Blöcke haben dieselbe Form, im Original unterscheidet sie allein die Farbe (violett gegen rot). Beschriftung und Namenspille stehen dabei auf weißem Grund; schraffiert ist nur der Hintergrund. Nebenbei braucht die Schraffur weniger Farbe als eine graue Fläche. Alles übrige unterscheidet sich ohnehin schon durch die Form (C-Maul, Hut, Bedingungsplättchen).
- An die Stelle des 3D-Anschliffs tritt eine schwarze Kontur, und die Eingabepillen bekommen einen Rand: auf einem weißen Block wären sie sonst nicht zu sehen.



`druck=false` (Vorgabe) `druck=true`

Die Farbschlüssel aus den vorigen Abschnitten bleiben dabei unangetastet: sie wirken wieder, sobald der Druckmodus aus ist – im Druckmodus selbst haben sie keine Wirkung, denn eigene Farben würden das Graustufenschema gerade aufheben. Eigene *Markenfarben* (`\marke[x=2, y=1, farbe=cyan]`) werden in ihren Grauwert umgesetzt, bleiben also als verschiedene Grautöne unterscheidbar. Die beiden Farben, die es nur im Druck gibt, heißen `karolDruckKontur` (Blockkonturen und Pillenränder) und `karolDruckAkzent` (Auswahldreieck der Pillen); sie lassen sich mit `\definecolor` ändern.

8 Aufgaben verlinken: `\karolonline`

Eine Aufgabe in Robot Karol Online steckt vollständig in ihrer Adresse: auf `https://karol.arrrg.de/` folgt eine Raute und der Aufgabencode. `\karolonline` macht daraus einen Verweis und setzt dabei nur, was man tatsächlich eintippen muss:

```
Die Aufgabe steht unter \karolonline{2t7t}.
```

Das ergibt: Die Aufgabe steht unter karol.arrrg.de/#2t7t. Der Verweis führt auf `https://karol.arrrg.de/#2t7t`. Der Code wird dabei unverändert übernommen – auf Groß- und Kleinschreibung kommt es bei den Aufgabencodes nicht an.

8.1 Mit oder ohne hyperref

Der Verweis braucht hyperref. Ob robotkarol es lädt, entscheidet eine Paketoption:

`hyperref=true`	hyperref wird geladen (Vorgabe)
`hyperref=false`	hyperref wird nicht geladen

Ohne hyperref setzt `\karolonline` denselben Text, nur ohne Verweis; das Dokument übersetzt in beiden Fällen unverändert.

Geladen wird erst am Ende der Präambel (im Haken `begindocument/before`) – hyperref möchte als letztes Paket geladen werden. Ein Dokument darf es deshalb auch selbst laden, mit eigenen Optionen und an beliebiger Stelle, ohne Optionskonflikt:

```
\usepackage{robotkarol}
\usepackage[colorlinks]{hyperref}
```

Geladen wird dabei ohne Optionen: wie Verweise aussehen – farbig, umrahmt oder unsichtbar –, stellt das Dokument selbst ein, etwa mit `\usepackage[hidelinks]{hyperref}` oder `\hypersetup{...}`.

Und weil `\karolonline` nicht die Paketoption abfragt, sondern ob es `\href` beim Setzen wirklich gibt, verlinkt es auch in einem Dokument, das hyperref trotz `hyperref=false` selbst geladen hat.

Ein Verweis ist ein unteilbarer Block. Steht er in einer schmalen Spalte, braucht der Absatz unter Umständen ein `\sloppy` (wie die Inline-Blöcke auch).

9 Aufgaben aus Robot Karol Online: JSON

Robot Karol Online speichert eine Aufgabe als JSON-Datei: im Editor *Aufgabe freigeben* → *Als JSON herunterladen*. robotkarol liest solche Dateien und kann sie auch schreiben: eine in Karol Online gebaute Aufgabe kann also fast direkt in LaTeX gesetzt werden, und eine in LaTeX gesetzte Welt lässt sich in Karol Online hochladen. Gelesen und geschrieben wird mit LuaLaTeX selbst ohne Shell-Escape. Die Datei `robotkarol.lua` gehört dazu und muss neben `robotkarol.sty` liegen.

Eine Aufgabe enthält eine oder mehrere Teilaufgaben (in Karol Online *Aufträge*), jede mit einer Start- und einer Zielwelt. Eine Teilaufgabe entspricht genau einer Umgebung `karolwelt`: Start und Ziel stehen darin wie in Abschnitt 2.5. Die Koordinaten sind dieselben wie in Karol Online – `x` von links, `y` von hinten, beide ab 1.

9.1 Einlesen und abfragen

```
\karoljsonlesen[name=treppen]{robotkarol-doc-treppen.json}
```

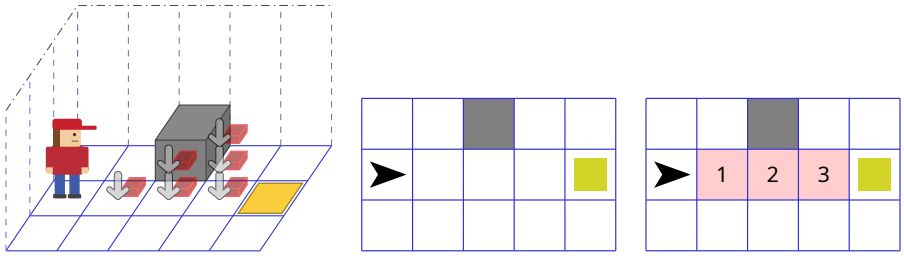
liest die Datei unter dem Namen `treppen` ein; ohne `name` heißt sie wie die Datei ohne Endung. Gesucht wird im Arbeitsverzeichnis, neben der `.tex`-Datei und wie jede andere Eingabedatei. Jede Datei wird je Lauf nur einmal gelesen. Statt des Namens darf überall auch der Dateiname stehen (`treppen.json`); die Datei wird dann bei Bedarf geladen.

`\karoljsontitel{treppen}`	Treppen
`\karoljsonbeschreibung{treppen}`	Zwei Aufträge rund um Treppen.
`\karoljsonanzahl{treppen}`	2
`\karoljsonaufgabentitel{treppen}{1}`	Baue eine Treppe aus drei Stufen.
`\karoljsonzielart{treppen}{1}`	welt
`\karoljsonzielart{treppen}{2}`	frei

Texte kommen unverändert an: `& % # _ $ ~ ^` im Titel werden gesetzt, nicht ausgeführt. Markdown in der Beschreibung wertet das Paket nicht aus; eine Leerzeile beginnt einen neuen Absatz. `\karoljsonanzahl` und `\karoljsonzielart` sind expandierbar, man kann mit ihnen rechnen und vergleichen.

9.2 Eine Welt aus der Datei

```
\karolweltaus{treppen}
\karolweltaus[zustand=start, ansicht=2d]{treppen}
\karolweltaus[zustand=ziel, ansicht=2d]{treppen}
```

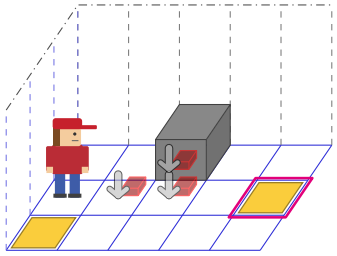


`\karolweltaus[`*Schlüssel*`]{`*Quelle*`}` ist eine Kurzform für eine Umgebung mit dem Schlüssel `quelle`. Welche Teilaufgabe gemeint ist, sagt `aufgabe` (Vorgabe 1). `breite`, `laenge` und `hoehe` kommen aus der Datei; alle anderen Schlüssel der Umgebung wirken wie gewohnt, auch in `karolweltpaar`:

```
\begin{karolweltpaar}[quelle=treppen]\end{karolweltpaar}
```

Mit der Umgebung lässt sich die eingelesene Welt ergänzen. Erst wird die Datei eingetragen, dann der Rumpf; der Rumpf gewinnt je Zelle:

```
\begin{karolwelt}[quelle=treppen, hervorheben={5/2}]
  \ziegel[x=4, y=2, zielanzahl=0]   % Stufe 3 entfällt
  \marke[x=1, y=3]                  % neue Marke, Start und Ziel
\end{karolwelt}
```



Ein Objekt ohne Seitenangabe ersetzt die Zelle in Start und Ziel, eins mit Seitenangabe nur die genannte Seite, und `startanzahl=0` bzw. `zielanzahl=0` löscht den Eintrag aus der Datei. `auftrag` wirkt nur auf die Objekte des Rumpfs.

9.3 Drei Arten von Aufgaben

Karol Online kennt keine eigene Angabe „Aufgabe mit Ziel“ – es vergleicht Start- und Zielwelt. `\karoljsonzielart` liefert das Ergebnis:

`welt`	Start und Ziel sind verschieden. Karol Online prüft, ob das Programm die Zielwelt herstellt. Auch ein Aufräumauftrag (Ziel ohne Ziegel und Marken) gehört dazu.
`frei`	Start und Ziel sind gleich: eine Welt zum Ausprobieren, ohne Prüfung („Ausführung beendet“). Karols Position zählt dabei nicht mit.
`skript`	Die Aufgabe hat ein Prüfskript (*QuestScript*); was es prüft, steht nicht in der Welt.

So kann ein Arbeitsblatt bei einer freien Aufgabe statt eines Zielbildes einen Hinweis setzen:

```
\ExplSyntaxOn
\str_if_eq:eeTF { \karoljsonzielart{treppen}{2} } { frei }
  { Probiere~aus! } { \karolweltaus[aufgabe=2, zustand=ziel]{treppen} }
\ExplSyntaxOff
```

Achtung beim Bauen in Karol Online: Ein neuer Auftrag hat dort eine *leere* Zielwelt, keine Kopie der Startwelt. Wenn man nur die Startwelt baut, erhält man einen Aufräumauftrag. Für eine freie Aufgabe muss im Welteditor *Start in Ziel kopieren* angeklickt werden.

9.4 Den Quelltext erzeugen

Oft soll die Datei nur der Ausgangspunkt sein. `\karolquelltext` setzt den Quelltext, den man sonst abtippen würde, `\karolquelltextdatei` schreibt ihn in eine Datei:

```
\karolquelltext{treppen}
\karolquelltextdatei[aufgabe=2]{treppen}{treppe-2.tex}
```

```
\begin{karolwelt}[breite=5, laenge=3]
  \karol[x=1, y=2, richtung=osten]
  \ziegel[x=2, y=2, ziel]
  \ziegel[x=3, y=2, ziel, anzahl=2]
  \ziegel[x=4, y=2, ziel, anzahl=3]
  \marke[x=5, y=2]
  \quader[x=3, y=1]
\end{karolwelt}
```

Mit `kompakt` werden gleiche Zellen zu Rechtecken zusammengefasst (`bisx`, `bisy`), und eine freie Aufgabe mit nur einer Reihe erscheint als `\karolzeile`. Ist die Ziel- bzw. die Startwelt leer, steht `auftrag=aufraeumen` bzw. `auftrag=aufbauen` im Kopf, und die Objekte brauchen keine Seitenangaben.

9.5 Welten exportieren

Jede Welt mit dem Schlüssel `export` wird zu einer Teilaufgabe. Die Adresse `sammlung/nummer` sagt, in welche Datei und an welche Stelle; `\karoljsonschreiben` schreibt die Sammlung:

```
\begin{karolwelt}[breite=5, laenge=3, export=blatt/1,
                  aufgabentitel={Baue die Treppe fertig.}]
  \karol[x=1, y=2, richtung=osten]
  \ziegel[x=2, y=2]
  \ziegel[x=3, y=2, startanzahl=1, zielanzahl=2]
  \ziegel[x=4, y=2, ziel, anzahl=3]
\end{karolwelt}
\karolzeile[export=blatt/2]{> Z 2 Z .}

\karoljsonschreiben[titel={Treppen}, beschreibung={Arbeitsblatt 3}]
  {blatt}{blatt.json}
```

- Eine Welt ohne `start`/`ziel`-Angaben wird eine freie Aufgabe, `auftrag=aufraeumen` ein geprüfter Aufräumauftrag. Sind Start und Ziel trotz solcher Angaben gleich, gibt es eine Warnung – Karol Online würde nicht prüfen. Eine `\karolzeile` ist immer frei.
- Geschrieben wird am Ende des Dokuments; `\karoljsonschreiben` darf also vor den Welten stehen, die Nummern in beliebiger Reihenfolge. Lücken sind ein Fehler. Eine geschriebene Datei lässt sich erst im nächsten Lauf wieder einlesen.
- Die Datei muss im Ordner des Dokuments oder in einem *vorhandenen* Unterordner liegen: TEX schreibt nicht nach `../` und legt keine Ordner an. Andernfalls meldet das Paket einen Fehler und schreibt die Datei nicht; das PDF entsteht trotzdem.
- Die Datei wird eingerückt geschrieben, ein Wert je Zeile – gut lesbar und für die Versionsverwaltung geeignet. Karol Online liest sie wie seine eigenen, einzeiligen Exporte.
- Fehlt `\karol`, steht Karol auf (1,1) mit Blick nach Süden (mit Warnung); `exportkarol={x=…, y=…, richtung=…}` setzt ihn nur für die Datei, ohne ihn zu zeichnen.
- `erbt=blatt/1` beginnt mit dem Inhalt eines weiter vorn belegten Slots, etwa für eine Variante. `quelle=blatt/1` geht ebenso.
- `karolweltpaar` darf `export` tragen: beide Hälften belegen denselben Slot mit demselben Inhalt.
- Heißt die Sammlung wie eine eingelesene Datei, bleibt alles erhalten, was nicht aus LATEX kommt – Programm, Prüfskript, Aufträge ohne `export`. So lässt sich eine Aufgabe einlesen, in LATEX ändern und zurückschreiben:

```
\karoljsonlesen[name=treppen]{treppen.json}
\begin{karolwelt}[quelle=treppen, aufgabe=2, export=treppen/2]
  \ziegel[x=5, y=1]
\end{karolwelt}
\karoljsonschreiben{treppen}{treppen-neu.json}
```

Den Share-Code, den Karol Online nach dem Hochladen vergibt, setzt `\karolonline` (Abschnitt 8).

9.6 Ein Startprogramm mitgeben

Eine Aufgabe kann ein Programm mitbringen, das beim Öffnen schon im Editor steht – ein Gerüst, eine angefangene Schleife, ein Kommentar mit dem Auftrag. Es steht in einer eigenen Textdatei, geschrieben in der Textsprache von Robot Karol, und `programm` nennt sie:

```
\karoljsonschreiben[titel={Treppen}, programm=treppe-start.karol]
  {blatt}{blatt.json}
```

mit `treppe-start.karol` etwa

```
// Baue die Treppe fertig.
wiederhole 3 mal
  Hinlegen
endewiederhole
```

Die Datei wird bei jedem Lauf gelesen und wie die Welten gesucht. `sprache` wählt den Editor: `blocks` (Vorgabe) öffnet die Aufgabe im Blockeditor, der den Text selbst in Blöcke umsetzt, `karol` im Texteditor. Ob das Programm übersetzbar ist, prüft das Paket nicht – Karol Online meldet einen Fehler erst beim Öffnen. Ohne `programm` wird ein durchgereichtes Programm einer eingelesenen Datei beibehalten. Alles in der Datei ist öffentlich lesbar; die Lösung gehört nicht hinein.

9.7 Was nicht übertragen wird

`\marke[farbe=…]`	Marken haben in Karol Online keine Farbe; sie geht beim Export verloren.
Darstellung	`ansicht`, `zustand`, `skalierung`, `nummern`, `rahmen`, `hervorheben`, `titel`, `hinweis` und die Figur gibt es nur in LaTeX.
Quader im Ziel	robotkarol kennt für Quader keinen Zielzustand und zeigt die der Startwelt (Warnung). Eine Datei mit verschiedenen Quadern in Start und Ziel ist in Karol Online ohnehin unlösbar.
Karol im Ziel	wird nicht gezeichnet, beim Zurückschreiben einer unveränderten Teilaufgabe aber erhalten.
„Nichts ändern“ prüfen	lässt sich nicht ausdrücken: gleiche Start- und Zielwelt heißt in Karol Online *frei*.

9.8 Große Welten

Karol Online erlaubt bis zu 100×100 Felder und Stapel bis 10; robotkarol liest jede solche Datei. Eine dicht gefüllte Welt dieser Größe braucht aber Zeit: gemessen etwa eine halbe Minute in `ansicht=2d` und gut sieben Minuten im Schrägbild. Ab rund tausend belegten Feldern im Schrägbild warnt das Paket mit einer Zeitschätzung.

10 Hinweise und Migration

- Das Paket benötigt **LuaLaTeX**.
- Neu in v4.0 ist die JSON-Schnittstelle zu Robot Karol Online (Abschnitt 9). Dazu gehört die Datei `robotkarol.lua`; sie muss neben `robotkarol.sty` liegen, sonst lässt sich das Paket nicht laden. Bestehende Dokumente setzen unverändert.

- Bis v2.x hieß das Zeichenketten-Kommando `\karolwelt`; ab v3.0 heißt es `\karolzeile`, der Name `karolwelt` bezeichnet die neue Umgebung. Bestehende Dokumente: einfaches Suchen/Ersetzen.
- Die kurzen Objektnamen `\karol`, `\ziegel`, `\marke`, `\quader` und `\wand` sind nur *innerhalb* der Umgebung `karolwelt` belegt. Ein Dokument darf sie also selbst verwenden – außerhalb der Umgebung behalten sie ihre eigene Bedeutung.
- In v3.10 hat sich dreierlei geändert, ohne Rücksicht auf ältere Dokumente. Der Schlüssel `aufraeumen` heißt jetzt `auftrag=aufraeumen`, neu dazu kam `auftrag=aufbauen`. Die Feldnummern von `\karolzeile` zählen ab 1 statt ab 0 – wie die Koordinaten der Umgebung. Damit gilt auch `hervorheben` ab 1: aus `hervorheben={2,4}` wird `hervorheben={3,5}`. Und `karolwelt` beginnt keinen neuen Absatz mehr: zwei Welten mit `\qquad` dazwischen stehen jetzt nebeneinander statt untereinander. Wer sie untereinander will, trennt sie durch eine Leerzeile.
- Vorgegeben ist die Figur `karol`. In v3.5 war für kurze Zeit `emma` die Vorgabe; wer sie behalten will, schreibt in die Präambel ein `\karolfigur{emma}`.
- Neu in v3.7 ist außerdem `\karolonline` für Verweise auf Aufgaben der Online-Umgebung (Abschnitt 8). **Dafür lädt das Paket jetzt hyperref** – wer das nicht will, lädt es mit `hyperref=false` (Abschnitt 8.1).
- Neu in v3.7 ist der Druckmodus (`\usepackage[druck]{robotkarol}`, Abschnitt 7.5). Ohne ihn ist die Ausgabe dieselbe wie in v3.6 – mit einer Ausnahme: die *Marke* sitzt jetzt in beiden Modi mittig auf ihrem Feld. Bis v3.6 fehlte in ihren Bildkoordinaten der Tiefenanteil, sie saß rund 0,3 mm zu weit links und berührte auf einem Ziegel dessen linke Kante.
- Seit v3.6 setzt das Paket seine Beschriftungen in den Schriften von Robot Karol Online und lädt dafür fontspec. Wer die bisherige Ausgabe will, lädt es mit der Paketoption `schrift=dokument` – siehe Abschnitt 7.1.

Lizenz

robotkarol steht unter der LaTeX Project Public License, Version 1.3c oder später (https://www.latex-project.org/lppl.txt), mit dem Wartungsstatus *maintained*. Copyright © 2026 Anselm Wagner.

Farben und Proportionen sind aus der Oberfläche von *Robot Karol Online* gemessen und mit TikZ nachgebaut; es wurden keine Grafiken übernommen.