

Das Paket **robotkarol**

Robot-Karol-Welten und Blockprogramme für Unterrichtsmaterial

Anselm Wagner*

23. August 2026 Version 3.6

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 Web-oberfläche nachempfunden. Einzelne Blöcke lassen sich als Inline-Blöcke in den Fließtext setzen. Das Paket benötigt Lua¹TeX¹ 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; requires LuaLaTeX.

Inhaltsverzeichnis

1	Schnellstart	1
2	Die Umgebung <code>karolwelt</code>	1
2.1	Koordinatensystem	1
2.2	Schlüssel der Umgebung	2
2.3	Objekte	2
2.4	Beispiel in beiden Ansichten	2
3	Einzeilige Welten: <code>\karolzeile</code>	3
3.1	Schlüssel	3
4	Blockprogramme	4
4.1	Himmelsrichtungen	4
5	Inline-Blöcke im Fließtext	5
6	Text-Programme und Struktogramme	6
7	Globale Einstellungen: <code>\karolsetup</code>	6
7.1	Schriften	6
7.2	Weltfarben	6
7.3	Figur der Welten	7
7.4	Blockfarben	7

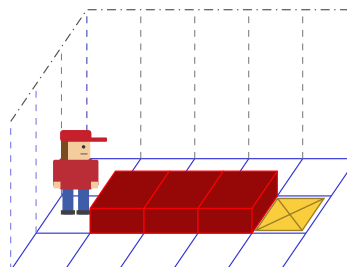
*anselm.wagner@posteo.de

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

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}
```



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.

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)
skalierung	1	Skalierungsfaktor des Bildes
titel		Beschriftung über dem Bild
hinweis		kursiver Hinweis unter dem Bild

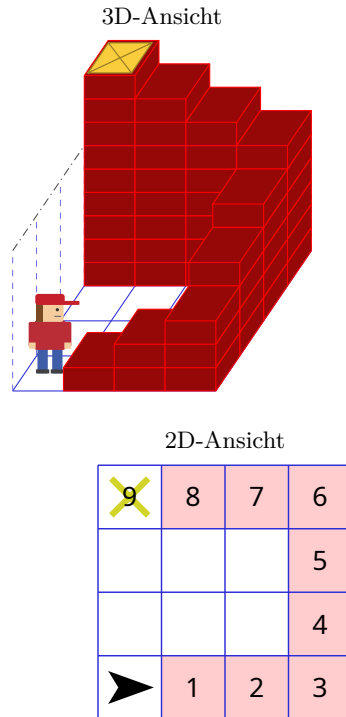
2.3 Objekte

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.

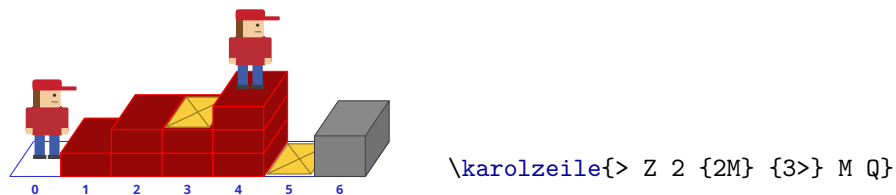
2.4 Beispiel in beiden Ansichten



In der 2D-Ansicht zeigen Zahlen die Stapelhöhe; Ziegelzellen sind rosa, Standard-Marken ein gelb-grünes Kreuz, gefärbte Marken vollflächig in ihrer Farbe, Quader grau, Karol ist der schwarze Richtungskeil.

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.

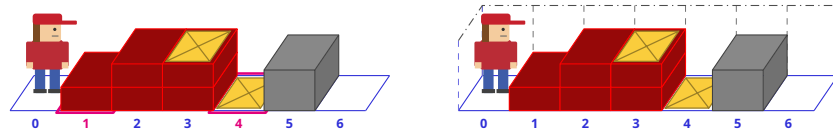


$\> \< \wedge v$ Karol mit Blickrichtung
 $\cdot$ leeres Feld
 $Z, 2 \dots 9$ Ziegel bzw. Stapel
 M, Q Marke, Quader
 $\{\dots\}$ mehrere Objekte auf einem Feld, von unten nach oben: $\{M\>\}$, $\{2M\}$, $\{3\>\}$

3.1 Schlüssel

skalierung	1	Skalierungsfaktor des Bildes
nummern	true	Feldnummern anzeigen
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. **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 zu wuchtig.



Dazu gibt es

<code>\karolpaar[keys][keys2]{vorher}{nachher}</code>	Vorher-nachher-Paar; keys gilt für beide Bilder, keys2 nur für „nachher“
<code>\karolgitter[keys]{n}</code>	leeres Gitter aus n Feldern
<code>\karollegende</code>	Standardlegende der Welsymbole
<code>\karollegendenitem{Welt}{Text}</code>	einzelner Legendeneintrag

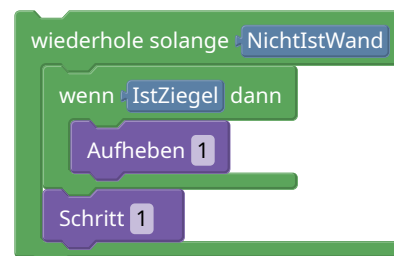
4 Blockprogramme

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}

```



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

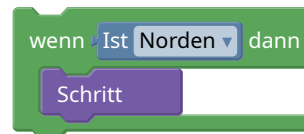
4.1 Himmelsrichtungen

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.

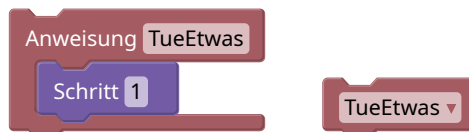


Für sich allein ergeben sie ein vollständiges Bedingungsplättchen. Als Argument eines Kontrollblocks steuern sie nur dessen Beschriftung bei – das Plättchen baut dort wie gewohnt `\bbedingung`:

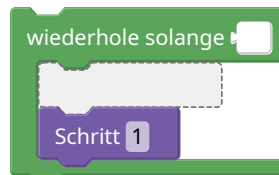
```
\bwenndann{\bist{Norden}}{%
  \bbefehl{Schritt}}
```



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 nahekkommt.

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.

<code>\ibefehl[n]{Name}</code>	Anweisung <code>Schritt 3</code>
<code>\iaufruf[n]{Name}</code>	Aufruf einer eigenen Anweisung
<code>\ianweisung{Name}</code>	Kopf einer eigenen Anweisung
<code>\ikommentar{Text}</code>	Kommentar
<code>\ihauptprogramm</code>	Hauptprogramm-Hut
<code>\iwiederholemal{n}</code>	Kopf der Zählschleife <code>wiederhole 5 mal</code>
<code>\iwiederholesolange{Bed}</code>	Kopf der kopfgesteuerten Schleife
<code>\iwiederholeimmer</code>	Kopf der Endlosschleife
<code>\iwenndann{Bed}</code>	Kopf der Verzweigung
<code>\isonst</code>	sonst-Balken <code>sonst</code>
<code>\ibedingung{Bed}</code>	Bedingung <code>IstMarke</code>
<code>\iist{Norden}</code>	Himmelsrichtung <code>Ist Norden ▾</code>
<code>\inichtist{Norden}</code>	verneint <code>NichtIst Osten ▾</code>
<code>\iluecke[Breite]</code>	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 $\text{\TeX}$ 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{felddbreite=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 T_EX 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:

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

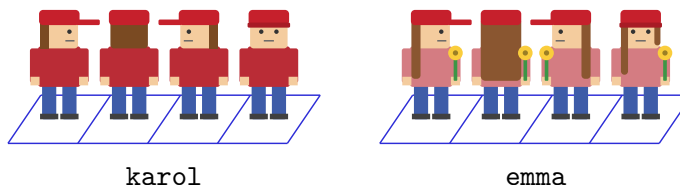
<code>felddbreite</code>	8.4mm	Zell-/Felddbreite aller Welten
<code>ziegelfarbe</code>	reines Rot	Lagenkante; Körper daraus abgeleitet (!59!black)
<code>ziegelkoerperfarbe</code>	150,8,8	nur die Ziegelflächen
<code>ziegelkantenfarbe</code>	255,0,0	nur die Lagenkanten
<code>markenfarbe</code>	250,205,60	Markenplatte
<code>quaderfarbe</code>	128,128,128	Quader / Wand
<code>bodenfarbe, gitterfarbe</code>	45,45,215	Bodengitter
<code>hervorfarbe</code>	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.



Emma ist aus derselben Figur gebaut wie Karol – sie ist ein Nachbau, keine Kopie. 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.

8 Hinweise und Migration

- Das Paket benötigt **LuaLaTeX**.
- 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.
- 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}`.
- 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 L^AT_EX 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.