Das tucseating-Paket

v0.3 2025-07-17

Erstellung von Sitzplänen

Matthias Werner[1]

https://github.com/tuc-osg/tucseating

Das Paket ermöglicht es, Sitzpläne, wie sie z. B. für Prüfungen benötigt werden, einfach zu erstellen. Eine Reihe verschiedener automatischer Platzierungsschemata sind vordefiniert, aber man kann auch feingranular eigene Platzierungen vornehmen. Während das Paket zunächst für den internen Gebrauch an der TU Chemnitz gedacht ist und (einige) vordefinierte Raumpläne enthält, sind einerseits sowohl die Raumdaten leicht erweiter- oder ersetzbar, andererseits können Räume auch ad hoc erstellt werden.

Inhaltsverzeichnis

1. matthias.werner@informatik.tu-chemnitz.de

1 Einführung

Für die Durchführung von Prüfungen benötigen wir mitunter Sitzpläne. So haben sich über die einige mit Pläne in Form von TikZ-unterstützten LaTeX-Dateien angesammelt. Je nach Anzahl der Studierenden in einer Prüfung (und für wie groß wir die Gefahr eines Betrugsversuches bewerten) nutzen wir unterschiedliche Platzierungsschemata, so dass wir die Dateien in anpassen müssen. Außerdem wird uns von Zeit zu Zeit ein neuer Raum zugewiesen, für den wir noch keine Pläne haben.

Dies war die Motivation zur Erschaffung des tucseating-Pakets. Es …

- ermöglicht eine schnelle und einfach Erstellung von Sitzplänen;
- trennt das Sitzlayout und das Platzierungsschema voneinander;
- bietet eine Reihe von Standardschemata für die Platzierung an;
- enthält bereits eine Anzahl vordefinierter Räume mit Layouts der Sitze;
- erlaubt eine *Ad-hoc*-Erstellung neuer Räume und Platzierungsschemata.

2 Abhängigkeiten

Das tucseating-Paket arbeitet nur mit LuaLaTeX und benötigt eine hinreichend moderne LaTeX-Version, mindestens vom Juli 2022. Es lädt folgende Pakete:

- etoolbox
- luacode
- tikz

Diese Pakete sind in allen gängigen TeX-Distributionen vorhanden und haben wiederum andere Pakete als Abhängigkeit. Insbesondere wird durch tikz das Paket xcolor geladen, dessen Farbdefinition auch in tucseating verwendet werden.

3 Sitzlayout

Das Paket wird wie üblich mit `\usepackage[`⟨*optionen*⟩`]{tucseating}` geladen. Dabei kann bereits weitgehend das Layout der Sitze festgelegt werden, also die Darstellung der Sitze im Raum. Die Zuweisung der Sitzbelegung erfogt später, siehe Abschnitt 4.

3.1 Klassenoptionen

Durch die Nutzung bzw. Nichtnutzung der Option `room` werden zwei grundsätzliche Anwendungsfälle unterschieden:

`room` = ⟨*Raum*⟩ Voreinstellung:

Diesen Optionsschlüssel sollte man nutzen, wenn der gewünschte Raum ist bereits in der Datenbank von tucseating vorhanden ist.

`room file` = {⟨*Dateiname*⟩} Voreinstellung: `rooms1`

Die Daten für die vordefinierten Räune werden aus ⟨*Dateiname*⟩`.tsr` gelesen. Dieser Schlüssel kann genutzt werden, wenn man eigene Räume anlegt, siehe Abschnitt 6.

Falls der Raum noch nicht bekannt ist, werden einige Schlüssel-Wert-Paar zur Beschreibung des Raumslayout gebraucht.

`shape` = `rectangle|arc` Voreinstellung: `rectangle`
Hier wird beschrieben, ob das Layout der Sitze rechteckig (`rectangle`) oder bogenförmig (`arc`)[2] ist.

`rows` = ⟨*Anzahl der Sitzreihen*⟩ (erforderlich)

`seats per row` = ⟨*Sitze pro Sitzreihe*⟩ (erforderlich)
Besimmt Anzahl der Sitzreihen und Sitze pro Reihe. Dabei muss immer von der jeweils maximalen Anzahl ausgegangen werden. Für unvollständige Reihen werden später Sitze entfernt, aber es können keine Sitze mehr zugefügt werden, die außerhalb des einmal festgelegten Layoutrahmens liegen. **Achtung:** Auch Gänge zwischen Blöcken von Sitzen müssen hier als Sitze mitgezählt werden.

Wenn Sie bei den Klassenoptionen **sowohl** den Schlüssel `room`, **als auch** eine der Schlüssel `rows` oder `seats per row` angeben, ist das Ergebnis unbestimmt. Entsprechend wird eine Warnung ausgegeben.

Alle weiteren Klassenoptionen legen die Darstellung des Raumlayouts fest. Sie können auch später mit `\tucsConfig` gesetzt werden, vgl. Abschnitt 3.2.

`blackboard` = `true|false` Voreinstellung: `false`
Zeichnet eine Tafel ein.

`seat distance` = ⟨*Abstand*⟩ Voreinstellung: `2pt`
Abstand der Sitze zueinander. Damit wird auch der Reihenabstand bestimmt. Möchte man beides unterschiedlich haben, so muss man die beiden folgenden Schlüssel nutzen:

`seat neighbor distance` = ⟨*Abstand*⟩ Voreinstellung: `2pt`

`row distance` = ⟨*Abstand*⟩ Voreinstellung: `2pt`

`rownumbers` = `none|left|right|both` Voreinstellung: `none`
Legt beim Rechteck-Layout fest, ob links und oder rechts der Reihen die Nummer der Sitzreihe angegeben wird. Dabei wird von vorn (Tafel) gezählt.
Anmerkung: Für das bogenförmige Layout ist diese Funktion derzeit nicht implementiert.

`rownumber distance` = ⟨*Abstand*⟩ Voreinstellung: `2pt`
Abstand der Reihennummern zu den äußeren Sitzen, wenn `rownumbers` nicht `none` ist.

`empty seat background color` = ⟨*Farbe*⟩ Voreinstellung: `lightgray!20`

`empty seat border color` = ⟨*Farbe*⟩ Voreinstellung: `lightgray`

`assigned seat background color` = ⟨*Farbe*⟩ Voreinstellung: `lightgray!30`

2. Z. B. an der TU Chemnitz der Raum A10.316.

`assigned seat border color` = ⟨*Farbe*⟩ Voreinstellung: `black`

Legt die Farben für den Hintergrund und die Umrandung von leeren bzw. durch ein Sitzschema belegten Sitzen fest. Falls die Sitze nicht farblich unterschieden werden sollen, können auch die Schlüssel

`seat background color` = ⟨*Farbe*⟩ (zunächst leer)

`seat border color` = ⟨*Farbe*⟩ (zunächst leer)

verwendet werden.

`assigned seat label font` = {⟨*Fontbefehl*⟩} Voreinstellung: `\small`

`assigned seat label color` = ⟨*Farbe*⟩ Voreinstellung: `black`

Setzt die Größe und Farbe der Sitzbeschriftung.

3.2 Späte Optionswahl

`\tucsConfig`

Mit diesem Kommando können außer `room`, `shape`, `rows` und `seats per row` alle im Abschnitt 3.1 genannten Schlüssel auch außerhalb von `\documentclass`-Optionen gesetzt werden.

3.3 Modifikation des Sitzlayouts

Häufig ist sind nicht in jeder Reihe alle Sitze vorhanden. Bei Angabe eine Raums ist dies bereits berücksichtigt, aber auch hier kann es vorkommen, dass beispielsweise ein Klappsitz defekt ist und aus dem Layout genommen werden muss. Beim Erstellen eines eigenen Layouts ist die Modifikation die Regel. Dafür stellt folgende Kommandos zur Verfügung

`\tucsRemoveSeatAt`{⟨*Reihe*⟩}{⟨*Sitznummer*⟩}

Entfernt den Sitz ⟨*Sitznummer*⟩ in der Reihe ⟨*Reihe*⟩ aus dem Layout. Dabei beziehen sich ⟨*Reihe*⟩ und ⟨*Sitznummer*⟩ auf das vollständige Layout, ändern sich also nicht durch bereits entfernte Sitze. Wenn die ⟨*Sitznummer*⟩ negativ ist, wird von rechts aus gezält

`\tucsRemoveSeats`{⟨*Liste*⟩}

Entfernt alle in der Liste vorkommenden Sitze aus dem Layout. Die Liste enthält dabei kommaseparierte Einträge der Form {⟨*Reihe*⟩,⟨*Sitznummer*⟩}. Für ⟨*Reihe*⟩ und ⟨*Sitznummer*⟩ gilt wie bei `\tucsRemoveSeatAt` das urspüngliche Layout. Im folgenden Beispiel werden die ersten drei Sitze links und der erste Sitz rechts in der ersten Reihe entfernt:

```
1  \tucsRemoveSeats{{1,1},{1,2},{1,3}{1,-1}}
```

`\tucsSetAisle`[⟨*Startreihe*⟩-⟨*Endreihe*⟩]{⟨*Sitznummer*⟩}

An der Stelle von ⟨*Sitznummer*⟩ wird durch alle Reihen ein Gang eingefügt. Nutzen Sie das optionale Argument, wenn der Gang nicht alle Reihen erfassen soll (Stichgang). Für horizontale Gänge (die also einen Sitzblock in einen vorderen und hinteren Teil zerlegen, anstatt

einen rechten und einen linke) nutzen Sie bitte \tucsRemoveSeats. Das ist im Prinzip auch bei vertikalen Gängen möglich, jedoch können bei der automatischen Sitzschemazuweisung mit \tucsSetAisle erzeugte Gänge anders als die mit \tucsRemoveSeats erzeugten Gänge behandelt werden.

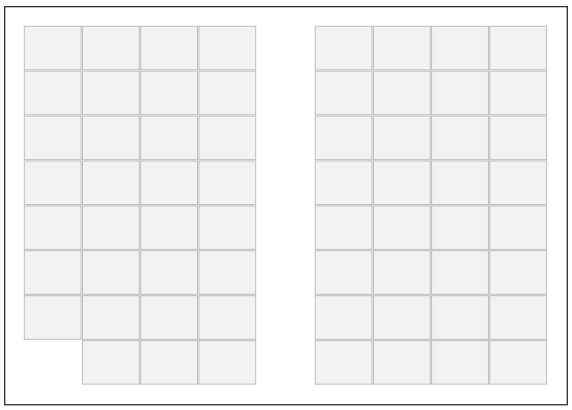
3.4 Ausgabe

\tucsDrawSeating*[seat width=⟨*Breite*⟩, seat height=⟨*Höhe*⟩] (zunächst leer)

Gibt den Sitzplan aus. Dabei versucht tucseating, die Ausmaße der Sitz so zu berechnen, dass der zur Verfügung stehenden Platz des Gesamtplans vollständig ausgenutzt wird. Da damit nicht immer alle Bedürfnisse getroffen werden, können in der über das optionale Argument die Breite und Höhe der Sitze auf dem Sitzplan einzeln eingestellt werden.

Das folgende Beispiel zeigt den Code und das Ergebnis für ein einfaches Sitzplanlayout mit einem Mittelgang und einem fehlenden Sitz in der ersten Reihe.

```
\documentclass{scrartcl}
% Um das Papier gut auszunutzen, nehmen wir es Landscape-Format
% und reduzieren die Ränder
\usepackage[a4paper,landscape,inner=1cm,outer=1cm,top=1cm,bottom=1cm]{
geometry}
\usepackage[shape=rectangle, rows = 8, seats per row=9]{tucseating}
\begin{document}
  \tucsSetAisle{5}
  \tucsRemoveSeatAt{1}{1}
  \tucsDrawSeating
\end{document}
```



4 Sitzplatzbelegung

4.1 Parameter zur Konstruktion von Belegungsschemata

Natürlich ist ein Sitzplan ohne die Markierung von belegten Plätzen nur bedingt nützlich. Die Belegung wird in tucseating mit Sitzplatzschemata realisiert. Zur Erzeugung eines Sitzschemas gibt es den Befehl

\tucsSeatingScheme[⟨*Schlüsselliste*⟩]{⟨*Name*⟩}

\tucsSeatingScheme*[⟨*Schlüsselliste*⟩]{⟨*Schema*⟩}

Das Pflichtargument enthält in der Standardvariante des Befehls eine Bezeichnung für ein vorgegebenes Sitzschema, siehe Abschnitt 4.2. In der Sternvariante wird hier eine Zeichenkette von „X“ und „-“ übergeben, die (den Anfang) eines Sitzschemas für eine einzelne Reihe beschreibt, wobei „X“ für einen belegten und „-“ für einen leeren Platz steht. Ist die Zeichenkette kürzer als die Anzahl der Sitze in der Reihe, wird sie wiederholt angewendet. Reihe mehr Plätze als Beispielsweise wird mit

```
1 \tucsSeatingScheme*{X--}
```

jeder dritte Sitz belegt.

Im optionalen Argument kann wieder eine Liste von Schlüssel-Wert-Paaren übergeben werden, die die restlichen Einstellung für das Sitzschema steuern. Es ist insbesondere in der Sternvariante wichtig, kann aber auch in der Standardvariante genutzt werden. werden:

`row sep` = ⟨*Anzahl*⟩ Voreinstellung: 2

Legt fest, in jeder wievielten Reihe Sitze belegt werden.

`start row` = ⟨*Nummer*⟩ Voreinstellung: 1

`end row` = ⟨*Nummer*⟩ Voreinstellung: ⟨*Anzahl Reihen*⟩

Durch diese beiden Schlüssel wird gesteuert, in welchen Reihen ein Schema angewendet wird. Dadurch können unterschiedliche Schemata für verschiedene Reihen genutzt werden.

`row restart after` = ⟨*Anzahl*⟩ (zunächst leer)

Das Sitzschema wird nach ⟨*Anzahl*⟩ Reihen zurückgesetzt. Damit wird es möglich, alternanierende Reihenabstände zu erzielen, vergleiche Beispiel in Abschnitt 5.3.

`aisle counts` = ⟨*Anzahl*⟩ Voreinstellung: 1

Legt fest, wie viele Sitze ein Gang zählt. Dadurch kann berücksichtigt werden, dass ein Gang häufig breiter als ein Sitzplatz ist.

`aisle restarts scheme` = `true`|`false` Voreinstellung: `false`

Startet das Schema nach einem Gang erneut. Der Schlüssel `aisle counts` wird damit wirkungslos.

`ignore aisle` = `true`|`false` Voreinstellung: `false`

`ignore removed seats` = `true`|`false` Voreinstellung: `false`
Bei der Zählung von Sitzen, die im Platzlabel genutzt werden kann (vgl. Abschnitt 4.3) werden auch entfernte Sitze und Gänge mitgezählt. Dies ist sinnvoll, damit in Reihen (im Rechteck-Grundlayout) sich gleiche Sitznummern hintereinander befinden. Wird einer der Schlüssel gesetzt, werden entfernte Sitze bzw. Gänge nicht mitgezählt. Dies bietet sich insbesondere im Bogen-Grundlayout an.

`assigned seat label` = {⟨*Formatzeichenkette*⟩} Voreinstellung: `m{{\,}}D`
Über diesen Schlüssel kann festgelegt werden, wie die belegten Sitze beschriftet werden. Voreingestellt ist die Nummer der Reihe, gefolgt von einem schmalen Leerzeichen und einem Buchstaben für den laufenden belegten Sitz (z. B. „3 c“). Es sind hier eine Reihe anderer Möglichkeiten einstellbar. Details dazu sind im Abschnitt 4.3 beschrieben.

`\tucsConfigScheme`{⟨*Schlüsselliste*⟩}
Setzt die Schlüssel wie beim optionalen Argument von `\tucsSeatingScheme`, führt aber keine Zuweisung von Sitzen aus.

Schlüsselwerte, die durch `\tucsSeatingScheme` oder `\tucsConfigScheme` gesetzt werden, bleiben erhalten, bis sie explizit neu gesetzt werden.

4.2 Vordefinierte Sitzschemata

Das tucseating-Paket definiert eine Reihe vordefinierter Platzierungsschemata. Manche haben auch einen Alternativnamen.

`1x1`
Jeder Platz ist markiert.
Alternativname: `all`

`2x2`
Zwischen zwei belegten Plätzen ist jeweils ein Platz bzw. eine Reihe Abstand.
Alternativname: `simple`

`2x2-`
Zwischen zwei belegten Plätzen ist jeweils ein freier Platz. Nach einer freien Reihe kommen *zwei* Reihen mit belegten Plätzen. Dies ist das Schema, mit dem in einer Prüfung die größtmögliche Zahl an Studierenden in einem Raum untergebracht werden können, aber trotzdem ein seitlicher Minimalabstand gegeben ist und sich die Aufsicht zu jedem Studierenden kommen kann (entweder von vorn, oder von hinten).
Alternativname: `dense`

`2x3`
Belegte Sitze haben einen seitlichen Abstand von zwei Sitzen und belegte Reihen einen Abstand von einer Leerreihe. Dies wird in unser Gruppe als das anzustrebende Standardschema für Prüfungen betrachtet.
Alternativname: `sixpack`

`2x3-`
Die Reihen werden wie bei `2x2-` belegt, der seitliche Abstand innerhalb einer belegten Reihe beträgt jedoch zwei Sitze.

`2x4`
Belegte Sitze haben einen seitlichen Abstand von drei Sitzen und belegte Reihen einen Abstand von einer Leerreihe.

`3x4`
Belegte Sitze haben einen seitlichen Abstand von drei Sitzen und belegte Reihen einen Abstand von zwei Leerreihen.

4.3 Sitzbeschriftung

Die Beschriftung der zugewiesenen Sitzplätze kann auf unterschiedliche Weise erfolgen, die durch Zuweisung einer Formatzeichenkette an den Schlüssel `assigned seat label` gesteuert wird. Es stehen dafür vier Zähler zur Verfügung:

- absolute Reihe: die Nummer der Reihe im Sitzlayout
- laufende Reihe: die Nummer der *belegten* Reihe. Reihen, in denen keine Sitz zugewiesen werden, werden hier übersprungen.
- absolute Sitznummer: die Nummer des Sitzes im Sitzlayout. Ob hier auch entfernte Sitze berücksichtigt werden, hängt vom Wert des Schlüssels `ignore removed seats` ab.
- laufende Sitznummer: die Nummer der *belegten* Sitze. Unbelegte Sitze werden bei der Zählung übersprungen.

Jeder dieser Zähler kann unterschiedlich formatiert werden. Dafür werden in der Formatzeichenkette verschiedene Formatierungszeichen benutzt, die in der folgenden Tabelle gelistet sind:

Zähler	**Anmerkung**	**Darstellung als…**				
		(arabische) Zahl	Kleinbuchstabe	Großbuchstabe	kleine römische Zahl	große römische Zahl
absolute Reihe	*bezieht sich auf das Sitzlayout*	`m`	`a`	`A`	`y`	`Y`
laufende Reihe	*bezieht sich auf das Belegungsschema*	`r`	`b`	`B`	`i`	`I`
absolute Sitznummer	*bezieht sich auf das Sitzlayout*	`n`	`c`	`C`	`x`	`X`
laufende Sitznummer	*bezieht sich auf das Belegungsschema*	`s`	`d`	`D`	`j`	`J`

Teile der Beschriftung, die nicht als Formatierungszeichen interpretiert werden sollen, werden in doppelte geschweifte Klammern gesetzt ({{⟨*geschützte Zeichenkette*⟩}}). Beispielsweise erzeugt

```
\tucsSeatingScheme[assigned seat label=Y{{-}}D]{2x3}
```

beim vierten Sitz (von links) in der dritten Reihe die Beschriftung „III-B“.

4.4 Sitzplatzliste

Insbesondere wenn es um den Anwendungsfall einer Prüfung geht, ist es nützlich, eine Sitzplatzliste zu erhalten, also eine tabellarische Zuordnung zwischen Prüfling und Sitzplatz. In der aktuellen Version erstellt tucseating keine (LaTeX)-Tabelle, kann aber die Tabellenerstellung (mit LaTeX oder einer Tabellenkalkulation) unterstützen.

`\tucsSeatingList[`⟨*Eingabedatei*⟩`]{`⟨*Ausgabedatei*⟩`}`

`\tucsSeatingList*[`⟨*Eingabedatei*⟩`]{`⟨*Ausgabedatei*⟩`}`

Erstellt eine CVS-Datei ⟨*Ausgabedatei*⟩ mit den Labels der belegten Plätze, jeweils einen pro Zeile.

Achtung! Die in die ⟨*Ausgabedatei*⟩ geschriebenen Labeltexte werden aus den entsprechenden LaTeX-Boxen extrahiert und nur die druckbaren Zeichen ausgegeben. Es empfiehlt sich bei Nutzung dieser Funkion daher, in der Formatzeichenkette von `assigned seat label` auf zuviel TeX-Magie zu verzichten.

In der Sternvariante werden in jeder Zeile die absoluten Koordinaten (Reihe, Platznummer, bezogen auf das Grundlayout) kommasepariert dem Labeln vorangestellt.

Wenn eine Eingabedatei angegeben wurde, wird ihr Inhalt zweilenweise den erzeugten Zeilen vorangestellt. Diese Eingabedatei könnte beispielsweise die Namen oder/und die Immatrikulationsnummern von Studierenden enthalten, die dann in der Ausgabedatei den markierten Plätzen zugeordnet werden. Sind weniger Einträge in ⟨*Eingabedatei*⟩ als Sitze belegt sind, werden die Felder in der Ausgabe leer gelassen. Wenn dagegen die Anzahl der Sitze nicht ausreicht, gibt tucseating eine Information aus, wer nicht platziert werden konnte.

5 Beispiele

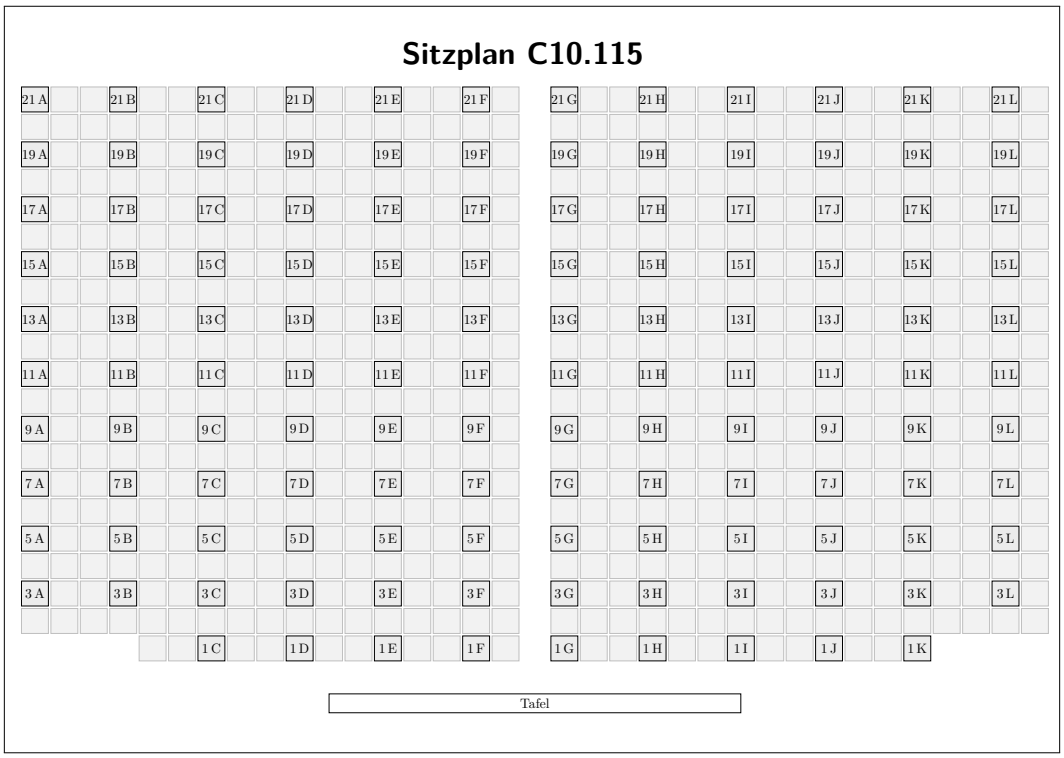
Zur Demonstration des Verhaltens von tucseating sind hier einige Beispiele des Einsatzes dokumentiert.

5.1 Vordefinierter Raum, rechteckig

```
\documentclass{scrartcl}
  % Um das Papier gut auszunutzen, nehmen wir es Landscape-Format
  % und reduzieren die Ränder
  \usepackage[a4paper,landscape,inner=10pt,outer=10pt,top=1cm,bottom=1cm]{
  geometry}
```

```
  \usepackage[
    room=C10.115,
    blackboard
  ]{tucseating}

\begin{document}
  % Standardtitelei nimmt viel Platz weg.
  \centering\textbf{\Huge\sffamily Sitzplan C10.115}\bigskip
  \tucsSeatingScheme{sixpack}
  \tucsDrawSeating
\end{document}
```

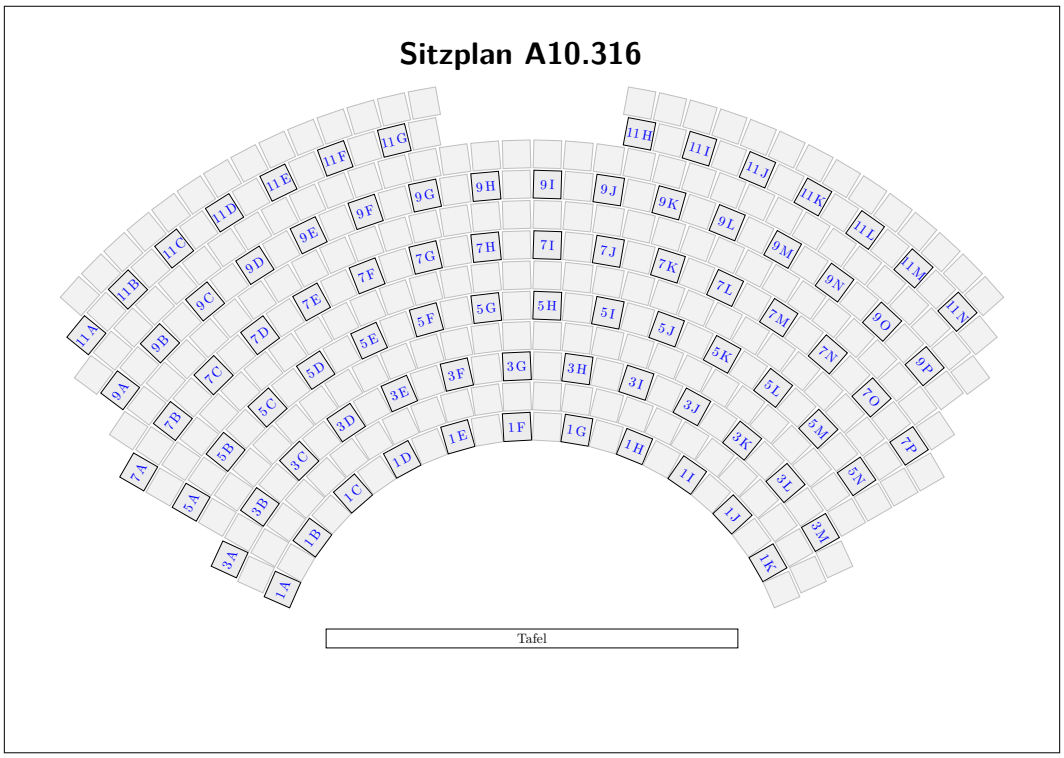


5.2 Vordefinierter Raum, bogenförmig

```
\documentclass{scrartcl}
  % Um das Papier gut auszunutzen, nehmen wir es Landscape-Format
  % und reduzieren die Ränder
  \usepackage[a4paper,landscape,inner=10pt,outer=10pt,top=1cm,bottom=1cm]{
  geometry}
  \usepackage[
```

```
    room=A10.316,
    blackboard,
    assigned seat label color=blue
  ]{tucseating}

\begin{document}
  % Standardtitelei nimmt viel Platz weg.
  \centering\textbf{\Huge\sffamily Sitzplan A10.316}\bigskip
  \tucsSeatingScheme[ignore removed seats]{2x2}
  \tucsDrawSeating
\end{document}
```



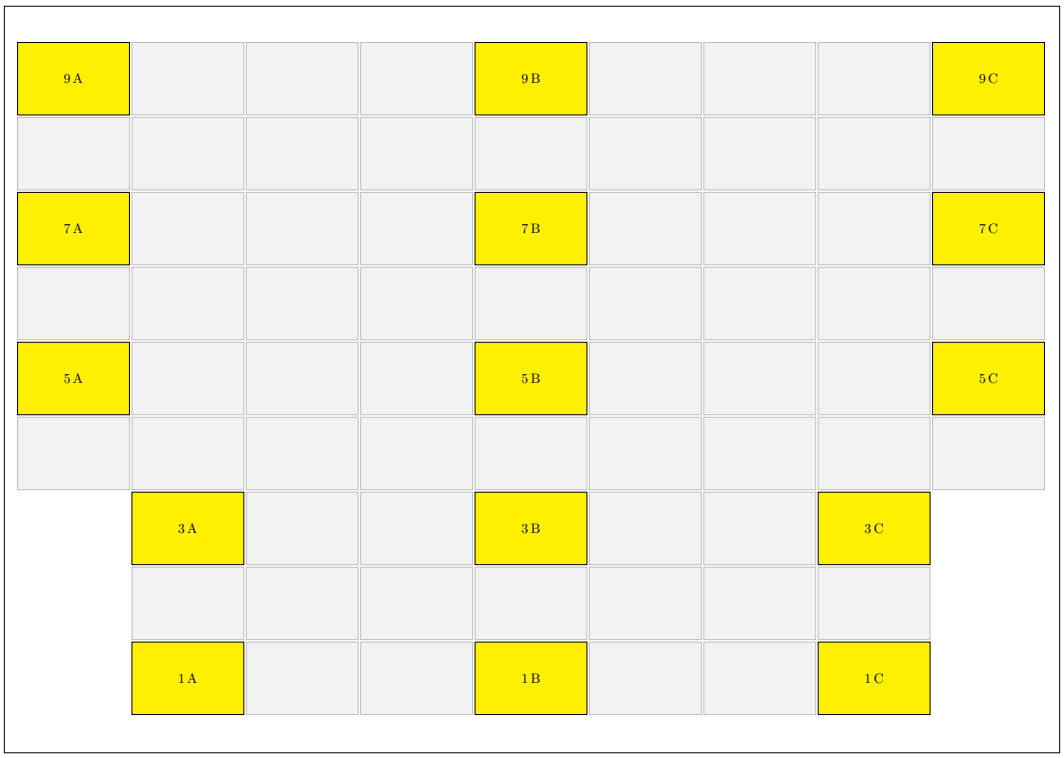
5.3 Eigenes Platzlayout und Sitzschema

```
\documentclass{scrartcl}
  % Um das Papier gut auszunutzen, nehmen wir es Landscape-Format
  % und reduzieren die Ränder
  \usepackage[a4paper,landscape,inner=10pt,outer=10pt,top=1cm,bottom=1cm]{
  geometry}
  \usepackage[
```

```
    shape=rectangle, rows=9, seats per row=9,
    assigned seat background color=yellow
  ]{tucseating}

  \begin{document}
    \tucsRemoveSeats{{1,1},{1,-1},{2,1},{2,-1},{3,1},{3,-1}}

    \tucsSeatingScheme[ignore removed seats,end row=3]{2x3}
    \tucsSeatingScheme[start row=5, end row=10]{2x4}
    \tucsDrawSeating
\end{document}
```



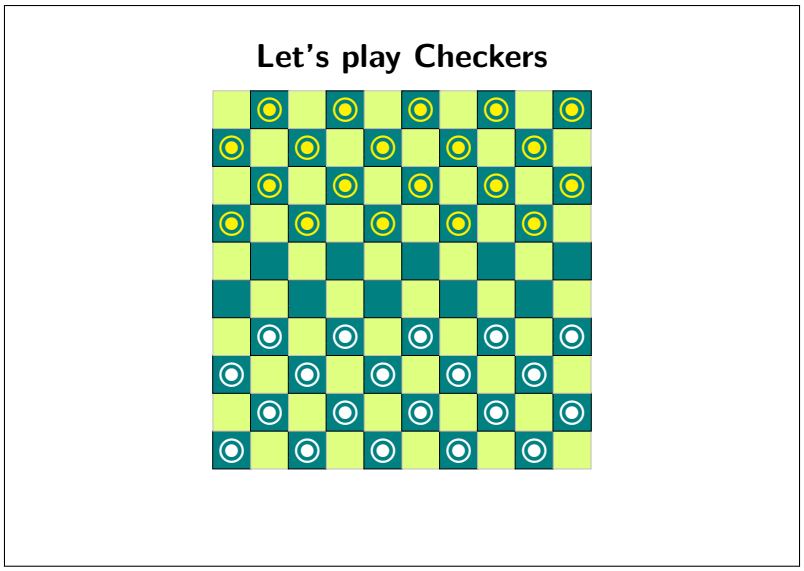
5.4 Dame

```
\documentclass{scrartcl}
  \usepackage[a5paper,landscape,top=1cm,bottom=1cm]{geometry}
  \usepackage[shape=rectangle, rows=10,seats per row=10,
     assigned seat label color=blue, assigned seat background color=teal,
     empty seat background color=lime!50,
     seat distance=0pt]{tucseating}
```

```
\usepackage{stix}
\begin{document}
  \centering\textbf{\Huge\sffamily Let's play Checkers}\bigskip

  \tucsConfigScheme{assigned seat label={}}
  \tucsSeatingScheme*[start row=5, end row=5]{X-}
  \tucsSeatingScheme*[start row=6, end row=6]{-X}
  \tucsConfigScheme{assigned seat label font=\Huge, assigned seat label={{{\
textcolor{white}{$\circledbullet$}}}}}
  \tucsSeatingScheme*[start row=1, end row=3,row sep=2]{X-}
  \tucsSeatingScheme*[start row=2, end row=4]{-X}
  \tucsConfigScheme{assigned seat label={{{\textcolor{yellow}{$\circledbullet
$}}}}}
  \tucsSeatingScheme*[start row=7, end row=9]{X-}
  \tucsSeatingScheme*[start row=8, end row=10]{-X}

  \tucsDrawSeating[seat width=1cm, seat height=1cm]
\end{document}
```



6 Deklaration neuer Raumlayouts

In der augenblicklichen Version hat das tucseating-Paket erst eine gewisse Anzahl von Räumen mit ihrem Sitzlayout erfasst. Nutzer werden daher häufig darauf angewiesen sein, eigene Layouts anzulegen. Neben der Möglichkeit zur Ad-hoc-Kreation von Layouts, wie sie im Abschnitt 3 beschrieben sind, besteht auch die Möglichkeit das Paket selbst zu erweitern und damit Layouts für neue Räume anzulegen, die dann einfach über den `room`-Schlüssel angesprochen werden können. Neue Raumlyouts können über zwei Wege integriert werden:

- Die zu dem tucseating-Paket gehörende Datei `room1.tsr` wird erweitert.
- Es wird eine eigene `.tsr`-Datei angelegt und über die Option `layout file` eingebunden.

Egel welcher Weg gewählt wird, in `.tsr`-Dateien können zwei Befehle genutzt werden:

`\tucsDeclareRoom`{⟨*Raumbezeichnung*⟩}{⟨*Schlüsselliste*⟩}
Ein neues Raumlayout wird für den Raum ⟨*Raumbezeichnung*⟩ angelegt. Die Schlüsselliste *muss* die drei Schlüssel

`shape` = `rectangle|arc` (erforderlich)

`rows` = ⟨*Anzahl der Sitzreihen*⟩ (erforderlich)

`seats per row` = ⟨*Sitze pro Sitzreihe*⟩ (erforderlich)
enthalten, die die analoge Bedeutung zu den gleichnamigen Schlüsseln aus Abschnitt 3 haben. Diese Angaben zum Basislayout *müssen* von dem Schlüssel

`init` (erforderlich)
gefolgt werden. Anschließend kann mit

`aisle` = ⟨*Sitznummer*⟩
`remove` = {⟨*Liste*⟩}
das Layout angepasst werden. Die beiden Schlüssel funktionieren analog zu `\tucsSetAisle` und `\tucsRemoveSeats`, vgl. Abschnitt 3.3.

```
\tucsDeclareRoom{C10.115}{
  shape=rectangle,
  rows=21,
  seats per row=35,
  init,
  aisle=18,
  remove={{1,1},{1,2},{1,3},{1,4},{1,-1},{1,-2},{1,-3},{1,-4}}
}
```

`\tucsAliasRoom`{⟨*Aliasname*⟩}{⟨*Originalname*⟩}
Setzt ⟨*Aliasname*⟩ als einen Ersatznamen für ⟨*Originalname*⟩.

7 Beschränkungen und Fehler

To be done.