The seatingchart Package

1.0.0beta 2026/07/22

Generation and visualization of seating plans

Matthias Werner*

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

This package allows for easy creation of seating charts, such as those required for examinations. Several automatic placement schemes are predefined, while custom, fine-grained assignments are also possible.

Warning! Starting with version 0.6, the `seatingchart` environment is the central user interface. This is an API break; for backward compatibility, see section 7.

Table of Contents

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

1 Introduction

To conduct exams, we occasionally require seating plans. Over time, we have accumulated several such plans in the form of TikZ-supported LaTeX files. Depending on the number of students in an exam—and how high we assess the risk of attempted cheating—we apply different placement schemes, which requires adapting the files accordingly. Moreover, we are sometimes assigned new rooms for which no seating plans exist yet.

This was the motivation for developing the seatingchart package. It …

- enables quick and easy creation of seating plans;
- separates seat layout from the placement scheme;
- offers a set of standard placement schemes;
- includes a number of predefined rooms with seat layouts;
- allows for *ad hoc* creation of new rooms and placement schemes.

2 Dependencies

The seatingchart package works only with LuaLaTeX and requires a sufficiently modern LaTeX version, at least from July 2022. It loads the following packages:

- etoolbox
- luacode
- tikz

These packages are available in all major TeX distributions and themselves depend on other packages. In particular, tikz loads the xcolor package, whose color definitions are also used by seatingchart.

3 Seat Layout

The package is loaded with `\usepackage{seatingchart}`. Each seating chart is placed in its own environment:

`\begin{seatingchart}[`⟨*options*⟩`]` … `\end{seatingchart}`

The options define the layout and its presentation. A document may contain any number of independently configured seating charts. All executing `\sc...` commands are available only inside the environment with exception of the room declaration commands, cf. Section 6.

3.1 Environment Options

The use of the option `room` distinguishes between two fundamental use cases:

`room` = `{`⟨*Room*⟩`}`

This option key should be used if the desired room

is already predefined.

`layout` = {⟨*institution*⟩} Default: `tu-chemnitz`

The data for the predefined rooms is read from `seatingchart-`⟨*institution*⟩`.sc`. This key can be used if own predefined rooms have been declared, see Section 6.

If the room is **not** already known, it can also be created *ad hoc*, whereas several key–value options are required to describe the layout:

`shape` = `rectangle|arc` Default: `rectangle`

Specifies whether the seating layout is rectangular (`rectangle`) or curved (`arc`).[2]

`rows` = {⟨*number*⟩} (required)

`seats per row` = {⟨*number*⟩} (required)

Defines the number of seat rows and seats per row. The maximum number must always be specified here. For incomplete rows, seats will be removed later, but no additional seats can be added that lie outside the originally defined layout area. **Note:** Aisles between blocks of seats must be counted here as seats as well.

If you specify **both** the `room` key **and** one of the keys `rows` or `seats per row` in the environment options, the result is undefined. A warning will be issued in that case.

All further environment options define the visual representation of the room layout. They can also be set later using `\scConfig`; see Section 3.2.

`blackboard` = `true|false` Default: `false`

Draws a blackboard.

`seat distance` = {⟨*distance*⟩} Default: `2pt`

Distance between seats. This also determines the row spacing. If you want these two values to differ, you must use the following keys:

`seat neighbor distance` = {⟨*Distance*⟩} Default: `2pt`

`row distance` = {⟨*distance*⟩} Default: `2pt`

`rownumbers` = `none|left|right|both` Default: `none`

For rectangular layouts, defines whether to display row numbers on the left, right, or both sides. Counting starts at the front (blackboard side).
Note: For arc-shaped layouts, this function is currently not implemented.

`rownumber distance` = {⟨*distance*⟩} Default: `2pt`

Distance between the row number and the outermost seats, if `rownumbers` is not `none`.

`empty seat background color` = {⟨*color*⟩} Default: `lightgray!20`

`empty seat border color` = {⟨*color*⟩} Default: `lightgray`

2. For example, room A10.316 at TU Chemnitz uses an arc layout.

`assigned seat background color` = {⟨*color*⟩} Default: `lightgray!30`

`assigned seat border color` = {⟨*color*⟩} Default: `black`

Defines background and border colors for empty or assigned seats. The syntax of color names follows the specification of the xcolor package. If the seats should not be differentiated by color, the keys

`seat background color` = ⟨*color*⟩ (initially empty)

`seat border color` = ⟨*color*⟩

can also be used.

`assigned seat label font` = {⟨*font command*⟩} Default: `\small`

`assigned seat label color` = ⟨*color*⟩ Default: `black`

Sets the size and color of the seat label.

3.2 Late Option Selection

`\scConfig`{⟨*key list*⟩}

This command can be used to set all keys mentioned in

Section 3.1, except for `room`, `shape`, `rows`, and `seats per row`, after the `seatingchart` environment has begun.

3.3 Modifying the Seat Layout

Often, not all seats are present in every row. This is already taken into account when specifying a predefined room, but it may still be necessary to remove individual seats — for instance, if a folding seat is broken.

When creating a custom layout, such modifications are typically required. For this purpose, provides the following commands:

`\scRemoveSeatAt`{⟨*row*⟩}{⟨*seat number*⟩}

Removes seat ⟨*seat number*⟩ in row ⟨*row*⟩ from the layout. The values refer to the original layout and are unaffected by previously removed seats. If ⟨*seat number*⟩ is negative, counting starts from the right.

`\scRemoveSeats`{⟨*list*⟩}

Removes all seats listed in the comma-separated ⟨*list*⟩, where each entry is of the form {⟨*row*⟩,⟨*seat number*⟩}. As with `\scRemoveSeatAt`, row and seat numbers refer to the original layout. In the following example, the first three seats on the left and the first seat on the right in the first row are removed:

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

`\scSetAisle[`⟨*start row*⟩`-`⟨*end row*⟩`]{`⟨*seat number*⟩`}`
Inserts an aisle at the position of ⟨*seat number*⟩ through all rows. Use the optional argument if the aisle should not span all rows.

For horizontal aisles (which split a seat block into front and back instead of left and right), please use `\scRemoveSeats`.

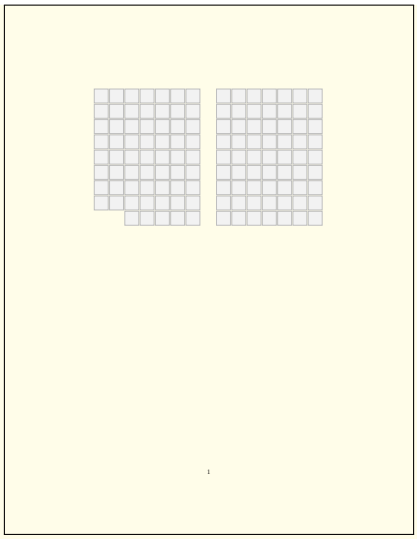
While this is also possible for vertical aisles, automatically generated seating schemes may treat aisles created via `\scSetAisle` differently than those created with `\scRemoveSeats`.

3.4 Output

`\scDrawSeating[seat width=`⟨*width*⟩`, seat height=`⟨*height*⟩`]`
Renders the seating plan early or with explicit dimensions. If it is not called, the chart is rendered automatically at the end of the `seatingchart` environment. seatingchart attempts to calculate the seat dimensions so that the available space for the complete plan is fully utilized. Since this may not always meet all needs, the optional argument allows the width and height of the seats on the plan to be adjusted individually.

The following example shows the code and the result for a simple seating layout with a central aisle and two missing seats in the first row.

```
\documentclass{article}
\usepackage{seatingchart}

\begin{document}
  \begin{seatingchart}[shape=rectangle, rows=9, seats per row=15]
    \scSetAisle{8}
    \scRemoveSeats{{1,1},{1,2}}

  \end{seatingchart}
\end{document}
```



4 Seat Assignment

4.1 Parameters for Constructing Assignment Schemes

A seating plan is only partially useful without indicating which seats are occupied. In seatingchart, seat assignment is handled via assignment schemes. To define such a scheme, the command

`\scSeatingScheme[`⟨*key list*⟩`]{`⟨*name*⟩`}`

`\scSeatingScheme*[`⟨*key list*⟩`]{`⟨*pattern*⟩`}`

The mandatory argument, in the standard version of the command, is a name for a predefined seating scheme (see Section 4.2). In the starred version, a string of "X" and "-" is passed, describing (the beginning of) a row assignment pattern, where X denotes an occupied seat and - an empty one. If the string is shorter than the number of seats in a row, it will be repeated.

For example,

```
tring\scSeatingScheme*{X--}
```

marks every third seat as occupied.

The optional argument accepts a list of key–value pairs that control the remaining settings of the assignment scheme. These are especially important in the starred variant but may also be used in the standard one.

`row sep` = ⟨*number*⟩ Default: 2

Specifies how many rows should lie between two occupied rows.

`start row` = ⟨*row*⟩
First row to be included in the assignment.

`end row` = ⟨*row*⟩
Last row to be included in the assignment. Together, these keys control the rows to which a scheme is applied. This allows different schemes to be used for different rows.

`row restart after` = ⟨*number*⟩ (initially empty)
Resets the seat pattern after ⟨*number*⟩ rows. This allows for alternating row spacing; see the example in Section 5.4.

`aisle counts` = ⟨*number*⟩ Default: `1`
Specifies how many seats wide an aisle should be considered. This allows for wider aisles to be taken into account.

`aisle restarts scheme` = `true`|`false` Default: `false`
Restarts the scheme after an aisle. In that case, the `aisle counts` key has no effect.

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

`ignore removed seats` = `true`|`false` Default: `false`
When determining seat numbers (used e.g. in labels; see Section 4.3), aisles and removed seats are normally counted. This ensures that seat numbers are consistent across rows (in rectangular layouts). If one of these keys is set, aisles or removed seats are no longer counted. This is particularly useful in curved layouts.

`assigned seat label` = {⟨*format string*⟩} Default: `mD`
Specifies how assigned seats should be labeled. By default, this consists of the row number, a thin space, and a letter for the current occupied seat (e.g., "3 c"). A number of other formats can be configured; see Section 4.3 for details.

`\scConfigScheme`{⟨*key list*⟩}
Sets keys just like the optional argument of `\scSeatingScheme`, but does not assign seats.

Key values set by `\scSeatingScheme` or `\scConfigScheme` remain in effect until they are explicitly redefined.

4.2 Predefined Seating Schemes

The seatingchart package defines a set of predefined seating schemes. Some of them also have alternative names.

`1x1`
Every seat is marked.
Alternative name: `all`

`2x2`
One empty seat and one empty row between two occupied ones.
Alternative name: `simple`

`2x2-`
: One empty seat between two occupied ones. After a single empty row, *two* rows with occupied seats follow. This scheme allows the largest number of students in a room during an exam while still maintaining minimal lateral distance and enabling supervision to reach every student (from the front or the back).
Alternative name: `dense`

`2x3`
: Occupied seats have two empty seats between them laterally, and one empty row between occupied rows. This is considered the preferred default scheme for exams in our group.
Alternative name: `sixpack`

`2x3-`
: Rows are occupied as in `2x2-`, but the lateral spacing within an occupied row is two seats.

`2x4`
: Occupied seats have three empty seats between them laterally, and one empty row between occupied rows.

`3x4`
: Occupied seats have three empty seats between them laterally, and two empty rows between occupied rows.

4.3 Seat Labeling

The labeling of assigned seats can be done in various ways, controlled by assigning a format string to the key `assigned seat label`. Four counters are available for this purpose:

- absolute row: the row number in the seat layout
- occupied row: the number of the *assigned* row. Rows without any assigned seats are skipped.
- absolute seat number: the seat number in the seat layout. Whether removed seats are counted depends on the value of `ignore removed seats`.
- occupied seat number: the number of the *assigned* seats. Unassigned seats are skipped in this count.

Each of these counters can be formatted differently. For this, the format string uses format specifiers listed in the following table:

Counter	Description	Rendered as...				
		Arabic numeral	lowercase letter	uppercase letter	lower Roman numeral	upper Roman numeral
absolute row	*based on seat layout*	m	a	A	y	Y
occupied row	*based on assigned rows*	r	b	B	i	I
absolute seat number	*based on seat layout*	n	c	C	x	X
occupied seat number	*based on assigned seats*	s	d	D	j	J

Parts of the label that should not be interpreted as format specifiers must be wrapped in double braces: `{{`⟨*protected string*⟩`}}`.

For example,

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

produces the label "III-B" for the fourth seat (from the left) in the third row.

4.4 Seat List

Especially in the context of examinations, it is useful to generate a seat list, i.e., a tabular mapping between student and assigned seat. In its current version, seatingchart does not create a LaTeX table, but it can support table generation (e.g., using LaTeX or a spreadsheet application).

`\scGenerateSeatingList[`⟨*input file*⟩`]{`⟨*output file*⟩`}`

`\scGenerateSeatingList*[`⟨*input file*⟩`]{`⟨*output file*⟩`}`

Creates a CSV file ⟨*output file*⟩ containing the labels of the assigned seats, one per line.

Note! The label texts written to ⟨*output file*⟩ are extracted from the corresponding LaTeX boxes, and only printable characters are included. When using this function, it's therefore recommended to avoid excessive TeX-level formatting in the `assigned seat label` format string.

In the starred variant, each line is prefixed with the absolute coordinates (row, seat number, relative to the layout), comma-separated, before the label.

If an input file is specified, its content is prepended line-by-line to the generated lines. For example, the input file could contain names and/or student IDs, which will then be matched to the assigned seats in the output. If there are fewer entries in ⟨*input file*⟩ than assigned seats, the corresponding fields in the output will be left empty. If, on the other hand, there are more entries than available seats, seatingchart will report which entries could not be assigned a seat.

4.5 Importing Labels from a File

Names, identifiers, or other labels can be imported directly into the seating plan from a CSV file. It is advisable to first create only the layout and seating scheme and display the seat designators in the plan. Use `assigned seat label=l` for absolute seat numbers, or, for example, `assigned seat label=m,n` for coordinates. Only after checking this reference plan should the input file be prepared and then imported with `\scLoadSeatingList`.

`\scLoadSeatingList[`⟨*key list*⟩`]{`⟨*input file*⟩`}`
The command imports labels from ⟨*input file*⟩ and automatically changes the label format to `l`, so that the imported text is shown on its seat.

`mode` = `sequential|seat|coordinates` Default: `sequential`
Selects how records in the input file are mapped to seats in the layout. The value `sequential` assigns records successively to seats that are already occupied. The value `seat` expects the absolute seat number in the first CSV column. The value `coordinates` expects the absolute row and the seat number within that row in the first two CSV columns.

`header` = `true|false` Default: `false`
Specifies whether the first non-empty line of the input file is a header row and should therefore be skipped during import.

The following example shows a file with absolute coordinates and a header.

```
row,seat,label
1,3,Alice
2,1,"Bob, ID 123"
```

Accordingly, the following options should be used.

```
\scLoadSeatingList[mode=coordinates,header]{participants.csv}
```

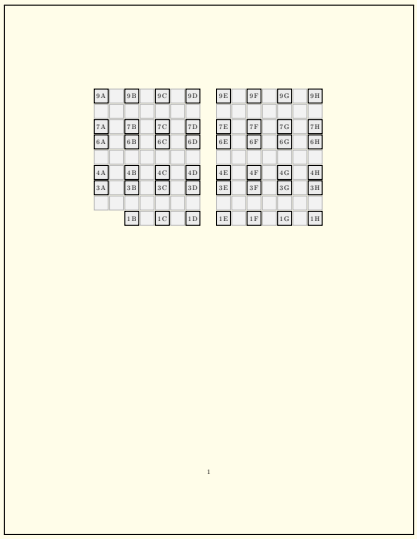
In both addressed modes, seats that were previously empty may be assigned; removed seats and aisles remain unchanged. CSV fields may be enclosed in double quotes. A double quote inside a field is escaped by doubling it.

5 Examples

To demonstrate the behavior of seatingchart, a few usage examples are documented here.

5.1 Dense Occupancy for Example Room in Section 3.4

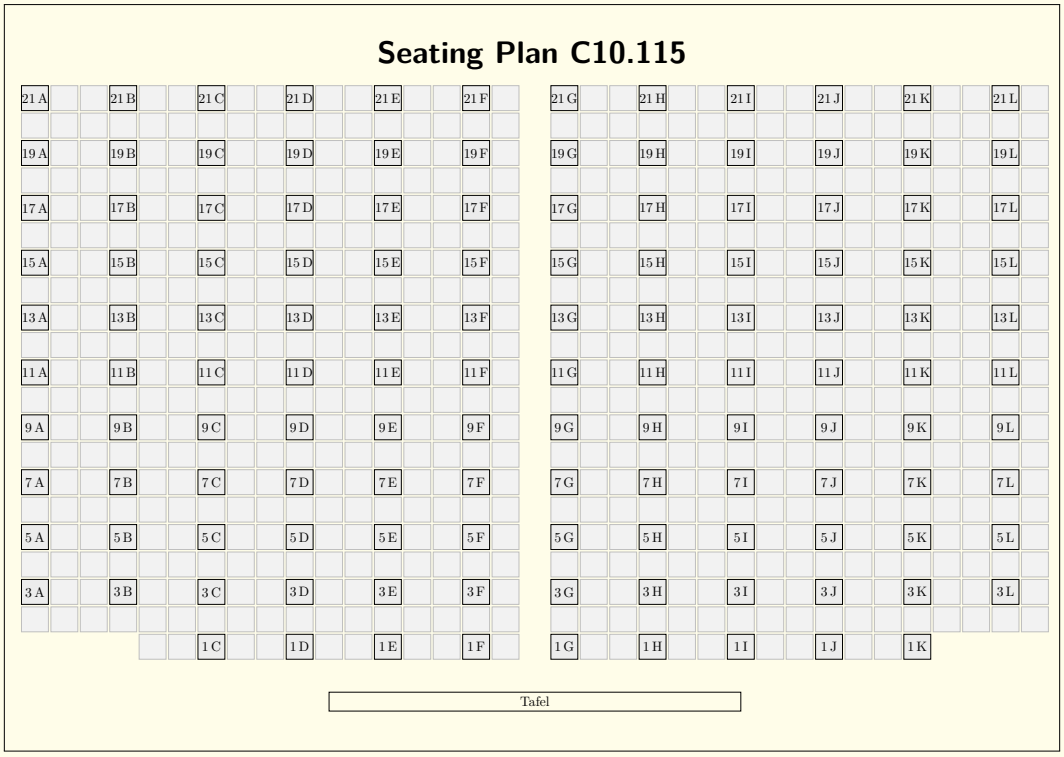
```
\documentclass{article}
\usepackage{seatingchart}

\begin{document}
  \begin{seatingchart}[shape=rectangle, rows=9, seats per row=15]
    \scSetAisle{8}
    \scRemoveSeats{{1,1},{1,2}}
    \scSeatingScheme{2x2-}

  \end{seatingchart}
\end{document}
```



5.2 Predefined Room, Rectangular

```
\documentclass{scrartcl}
  % To make better use of the paper, we use landscape format
  % and reduce the margins
  \usepackage[a4paper,landscape,inner=10pt,outer=10pt,top=1cm,bottom=1cm]{geometry}
  \usepackage{seatingchart}

\begin{document}
```

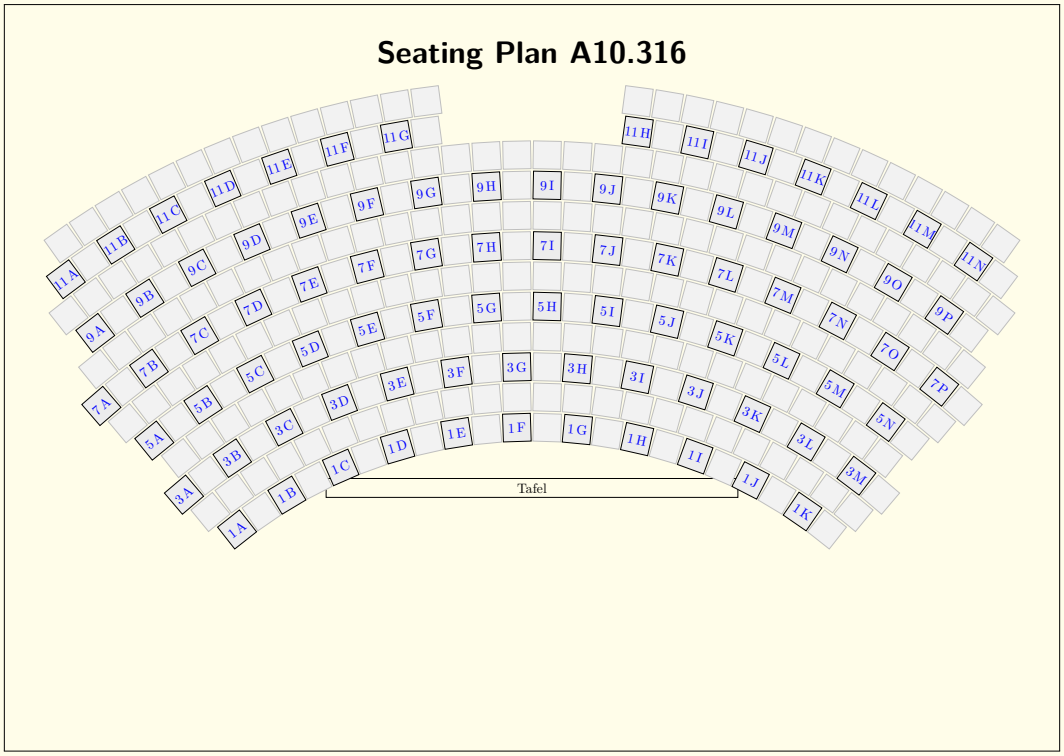
```
  % Default title takes up too much space.
  \centering\textbf{\Huge\sffamily Seating Plan C10.115}\bigskip

  \begin{seatingchart}[room=C10.115,blackboard]
    \scSeatingScheme{sixpack}

  \end{seatingchart}
\end{document}
```



5.3 Predefined Room, Curved

```
\documentclass{scrartcl}
  % To make better use of the paper, we use landscape format
  % and reduce the margins
  \usepackage[a4paper,landscape,inner=10pt,outer=10pt,top=1cm,bottom=1cm]{geometry
  }
  \usepackage{seatingchart}

 \begin{document}
   % Default title takes up too much space.
```

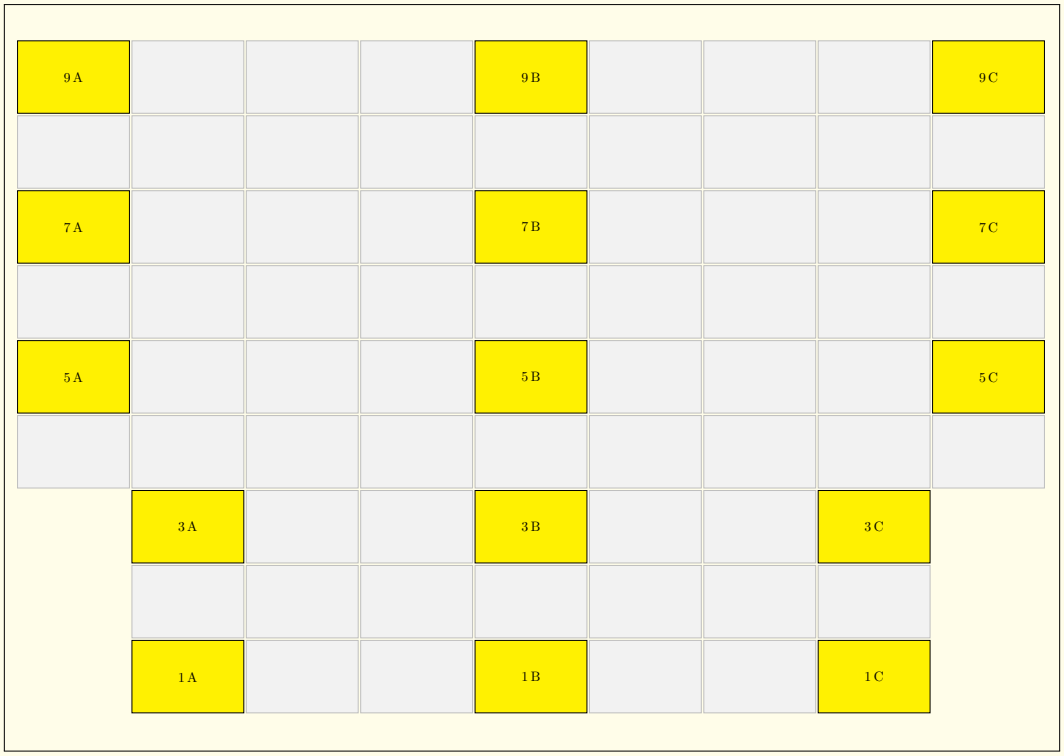
```
  \centering\textbf{\Huge\sffamily Seating Plan A10.316}\bigskip

  \begin{seatingchart}[room=A10.316,blackboard,
    assigned seat label color=blue]
    \scSeatingScheme[ignore removed seats]{2x2}

  \end{seatingchart}
\end{document}
```



5.4 Custom Layout and Seating Scheme

```
\documentclass{scrartcl}
  % To make better use of the paper, we use landscape format
  % and reduce the margins
  \usepackage[a4paper,landscape,inner=10pt,outer=10pt,top=1cm,bottom=1cm]{geometry
  }
  \usepackage{seatingchart}

  \begin{document}
    \begin{seatingchart}[shape=rectangle,rows=9,seats per row=9,
```

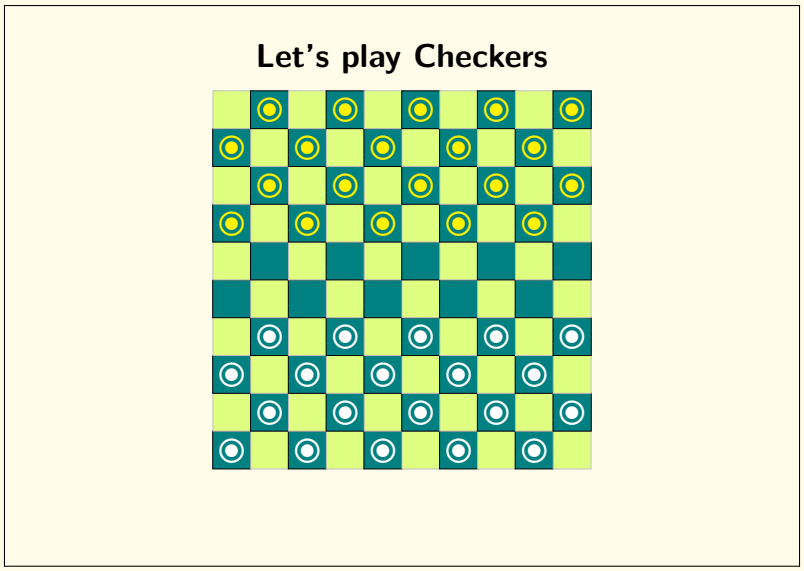
```
      assigned seat background color=yellow]
      \scRemoveSeats{{1,1},{1,-1},{2,1},{2,-1},{3,1},{3,-1}}

    \scSeatingScheme[ignore removed seats,end row=3]{2x3}
    \scSeatingScheme[start row=5, end row=10]{2x4}

    \end{seatingchart}
\end{document}
```



5.5 Checkers

```
\documentclass{scrartcl}
  \usepackage[a5paper,landscape,top=1cm,bottom=1cm]{geometry}
  \usepackage{seatingchart}
  \usepackage{stix}

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

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

    \scDrawSeating[seat width=1cm, seat height=1cm]

    \end{seatingchart}
\end{document}
```



6 Declaration of New Rooms

In its current version, the seatingchart package only contains a relatively small number of rooms at TU Chemnitz with their seating layouts. Users will therefore often have to create their own layouts.

In addition to the option for ad-hoc layout creation, as described in section 3, it is also possible to extend the package itself and create layouts for new rooms, which can then be easily accessed via the `room` key.

For this purpose, custom `seatingchart-*.sc` files can be created and integrated via the `layout` key. Two commands can be used in `seatingchart-*.sc` files:

`\scDeclareRoom{`⟨*room name*⟩`}{`⟨*key list*⟩`}`
A new room layout is created for the room ⟨*room name*⟩. The key list *must* contain the three keys

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

`rows` = ⟨*number of rows*⟩ (required)

`seats per row` = ⟨*seats per row*⟩ (required)
These keys have the same meaning as the keys of the same name from Section 3. This information about the base layout *must* be followed by the key

`init` (required)

The layout can then be adjusted using

`aisle` = ⟨*seat number*⟩
`remove` = `{`⟨*list*⟩`}`
These two keys function analogously to `\scSetAisle` and `\scRemoveSeats`, see Section 3.3.

```
\scDeclareRoom{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}}
}
```

`\scAliasRoom{`⟨*alias*⟩`}{`⟨*room*⟩`}`
Sets ⟨*alias*⟩ as a replacement name for ⟨*room*⟩.

If you would like your institution's layout file to be included in the package on CTAN, please send your file to the maintainer or initiate a pull request on `https://github.com/tuc-osg/seatingchart`. In this case, give the file a meaningful and distinguishable name. For example, *University of Lummerland at Lummerland City* should use `seatingchart-lummerland-uni-lc.sc` rather than `seatingchart-lu.sc`.

7 Compatibility Mode

For existing documents, the package option `legacy` enables the former document-wide behaviour. Layout keys remain package options, executing commands are available throughout the document, and `\scDrawSeating` must be called explicitly:

```
\usepackage[legacy,shape=rectangle,rows=9,seats per row=15]{seatingchart}
```

The compatibility mode is intended for migrating existing sources; new documents should use the `seatingchart` environment. Only in this mode is the former name `\scSeatingList` available as an alias for `\scGenerateSeatingList`.

8 Limitations and Bugs

Even though the seatingchart package is in practical use, at least in our group, it should still be considered experimental. The `beta` in the version number indicates this fact.

There are a number of restrictions, including, among others:

- Several seating arrangements cannot be accommodated. For example, staggered rows or typical conference layouts are not possible.
- The angle for the curved layout is limited to 120°.
- The number of predefined rooms is currently relatively small. There are plans to include additional rooms at TU Chemnitz in future patches. Layout files from other institutions are welcome.

Please use GitHub for bug reports or pull requests: `https://github.com/tuc-osg/seatingchart`.

9 License