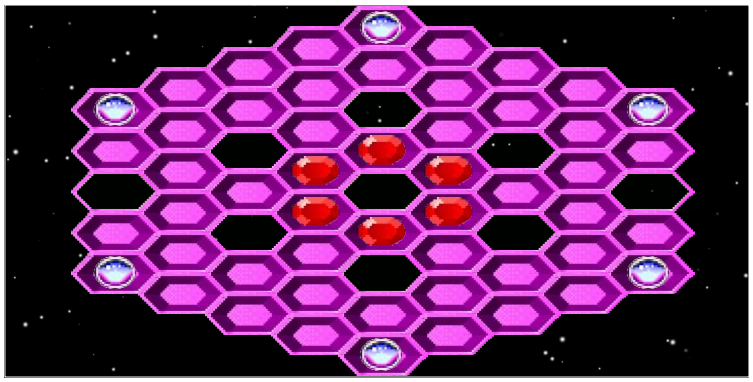
Documentation for: `eagle-maps.sty` v1.0

Florian Sihler*

October 25, 2019



Abstract

This package was designed to create maps consisting of different tiles from a top-down perspective. It supports the creation of tiles and players, externalization and some other features regarding typesetting the maps. All Code highlighting will be performed with the `LILLYxLISTINGS`-Library from the Lilly-Framework.[1]

1 Prerequisites

1.1 Required Packages

This Package depends on `tikz`[2] and `environ`.[3] Furthermore the `tikz`-libraries `calc` and `external` will be loaded.

1.2 How to use this library

Just put: `\input{eagle-maps}` into your preamble and place the `eagle-maps.sty` in the same folder. Otherwise, you can place it into your `texmf`-tree to make it globally available.

1.3 How to read the documentation

Any command will be described like this:

▷ `\commandName[optional Arg]{Mandatory Arg 1}{Mandatory Arg 2}`
This text will describe the Command. It can link to other commands/environment as well: `env@environmentName` → p. 1!

*florian.sihler@uni-ulm.de
[1] https://github.com/EagleoutIce/LILLY
[2] https://www.ctan.org/pkg/pgf
[3] https://www.ctan.org/pkg/environ

▷ env@environmentName{Mandatory Arg 1}{Mandatory Arg 2}

This text will describe the environment (the **env@**-prefix is just for clarification and explicitly *no* part of the name). It can link to other commands/environment as well: \commandName → p. 1!

2 Functions and Features

2.1 Setting Sizes

▷ \emSetFieldSize{width}{height}

This command will set the output-size of the field. They will be stored in the underlying registers: **em@len@field@w** and **em@len@field@h**. This command is similar to \emSetTileSize → p. 2 and \emSetPlayerSize → p. 2.

▷ \emSetTileSize{width}{height}

This command will set the size of a single tile. They will be stored in the underlying registers: **em@len@tile@w** and **em@len@tile@h**. This command is similar to \emSetFieldSize → p. 2 and \emSetPlayerSize → p. 2.

▷ \emSetPlayerSize{width}{height}

This command will set the size of a player-token. They will be stored in the underlying registers: **em@len@player@w** and **em@len@player@h**. This command is similar to \emSetFieldSize → p. 2 and \emSetTileSize → p. 2.

▷ \emSetTileOffset{offset x}{offset y}

Will set the offset between two tiles in the same row (**offset x**) and between two lines (**offset y**). For convenient use **offset x** should be at least the width of a single tile.

2.2 Create Players and Tiles

▷ \emCreateTile *[Drawer-Args]* {Tile-Shortcut}{Tile-Drawer}

This will create a new tile with the name **Tile-Shortcut**. Without any modifications you can select between those **Tile-Drawer**s: **void** (empty), **color** (rectangle with color **Drawer-Args**) and **image** (rectangle with image, located at **Drawer-args**).[4] If you want to create a black-squared tile named **B** (to use it with env@eagle-map* → p. 3 or env@eagle-map → p. 3 it's recommended (but not necessary) to use only one letters), you can do the following: \emCreateTile[black]{T}{color}. See also: \emSetTileSize → p. 2 and \emSetTileOffset → p. 2.

▷ \emCreatePlayer *[Default-Drawer-Arg]* {Shortcut}{Player-Name}{Color}{Drawer}

This command will create \<player> which will expand to the name of the Player, \<player>id expanding to the (unique) Player-ID and \<player>set which is a little bit special. It takes an optional Argument which gets passed to the drawer and a mandatory one - the target-position. \<player>set *has* to be used inside of a env@eagle-map → p. 3, env@eagle-map* → p. 3 or env@tikzpicture environment. The Drawer can be either **color** (**Default-Drawer-Arg** will correspond to the color) and **image** (**Default-Drawer-Arg** will correspond to the image-path).[5] A small example:

[4] *You can create your own drawer by defining a macro named \em@draw@tile@<name> consuming one argument: **{Drawer-Args}***

[5] *You can create your own drawer by defining a macro named \em@draw@player@<name> consuming five arguments: **{coordinate}{player-color}{optional-argument}{player-shortcut}***

```
\emCreatePlayer{playerOne}{Florian}{green}{color}
\playerOne % → Florian
\playerOneid % → 5
```

See also: \emSetPlayerSize → p. 2. For more Information about practical usage, consult: \emSetPlayer-Lines → p. 4 and \emSetPlayerColumns → p. 5.

2.3 Typeset a Map

▷ \emSetField{Map-Data}

Will set the data for the current map. Has to be a comma-separated list of slash-separated pairs whereas the key corresponds to the offset as a factor of tiles and the value is a list of Tiles. A small example: \emSetField↩ {{1/T},{0/TPT},{1/T}} (created with env@eagle-map* → p. 3 using the tile created in the description of \emCreateTile → p. 2, P being the purple-variant):

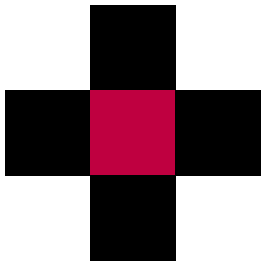


Figure 1: Example of a Map

▷ env@eagle-map*[Map-Data]

If you do not give a Map-Data the last Map set by \emSetField → p. 3 will be used. The current Drawing-Order is work-in-progress, but you can change the background of the drawn-map by \emSetBackground → p. 3.

▷ env@eagle-map[Map-Data]{name}

Works similar to env@eagle-map* → p. 3 but tries to externalize the image! *Please note: if you want to use externalization you have to set \tikzexternalize into the preamble after importing this package.* name corresponds to the file-name of the externalized graphic. If you want to create a png-file instead of a pdf (only works if the shell-tool convert is available) see: \emUsePng → p. 4 and \emSetPngDensity → p. 4.

▷ \emSetBackground[*]{Path}

Will set the Path to the Background-Image. The unstarred-Version will change the Background-Drawer to the Image-Drawer!.

▷ \em@draw@map

Low-Level-Variant (called by env@eagle-map* → p. 3 and env@eagle-map → p. 3) to draw the current map. Can be used inside of a \tikzpicture.

2.4 Externalization

See also: env@eagle-map → p. 3. To activate Externalization you *must* type \tikzexternalize into your preamble. Consult the pgf-Documentation for details. *Externalization needs* ***-shell-escape*** *(or* ***write18****) to be enabled!*

▷ \emUsePng

From now on (locked by the current group) the externalization-process will try to convert the graphic to a png. The density can be controlled by \emSetPngDensity → p. 4.

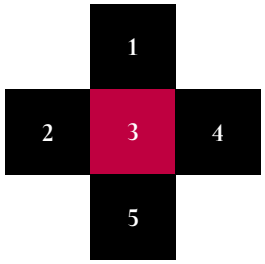
▷ \emSetPngDensity{density}

Will change the density for all externalized pngs for the current group. See also: \emUsePng → p. 4.

2.5 Coordinates and other Routines

Whilst drawing the map \em@draw@map will create a named coordinate for every Tile it places. The Coordinates will be prefixed with emline and suffixed with an incrementing number. Lets make an example, using the map created in the description of \emSetField → p. 3:

```
\emSetField{1/T, 0/TPT, 1/T}
\begin{eagle-map*}
    \foreach \i in {1,...,5}{
        \node[white] at (emline\i) {\bfseries \i};
    }
\end{eagle-map*}
```

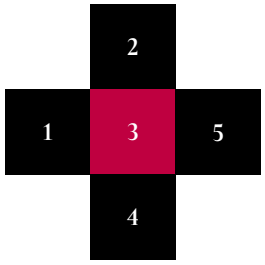


▷ \emCalculateCols *[Level-Distance]* {ColumnData}

As it exceeded the time i had for this package there is no implicit converter to translate line- to column-based coordinates (i've written way over 100 lines trying to create one), so there is this routine which expect the comma-separated list to be built up by slash-separated triplets: startOfColumnX(factor)/startOfColumnY↩(factor)/lengthOfColumn. If you don't have quadratic-tiles and the height of the tile (\emSetTile-Size → p. 2) differs from the actual y-offset (\emSetTileOffset → p. 2), you can specify the offset you like via Level-Distance.

Lets redo the example with column-based coordinates (they will be prefixed with emcol):

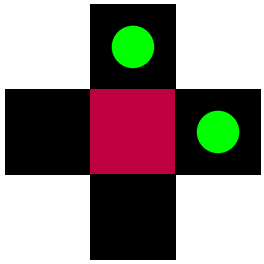
```
\emSetField{1/T, 0/TPT, 1/T}
\begin{eagle-map*}
    \emCalculateCols{
        0/1/1,% first col starts at (0,-1)
        1/0/3,% second col starts at (1,0)
        2/1/1 % third/last col starts at (2,-1)
    }
    \foreach \i in {1,...,5}{
        \node[white] at (emcol\i) {\bfseries \i};
    }
\end{eagle-map*}
```



▷ `\emSetPlayerLines{PlayerList}`

This Command accepts a comma-separated List of slash-separated tuples whereas the key refers to the **emline**-coordinate id and the value to the player to be set there. We use `\playerOne` created in the explanation for `\emCreatePlayer` → p. 2

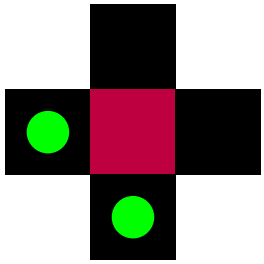
```
\emSetField{1/T, 0/TPT, 1/T}
\begin{eagle-map*}
   \emSetPlayerLines{1/playerOne, 4/playerOne}
\end{eagle-map*}
```



▷ `\emSetPlayerColumns{PlayerList}`

Works similar to `\emSetPlayerLines` → p. 4 but the key corresponds to the **emcol**-coordinate. Again, with `\playerOne` from `\emCreatePlayer` → p. 2:

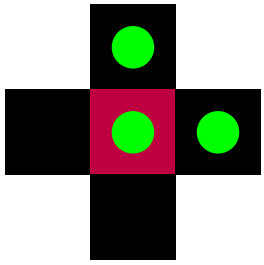
```
\emSetField{1/T, 0/TPT, 1/T}
\begin{eagle-map*}
   \emCalculateCols{0/1/1, 1/0/3, 2/1/1}
   \emSetPlayerColumns{1/playerOne, 4/playerOne}
\end{eagle-map*}
```



▷ `\emSetOnePlayerLines{player}{LineID-List}`

Works similar to `\emSetPlayerLines` → p. 4 but this time you don't have to specify the player to use. Again, with `\playerOne` from `\emCreatePlayer` → p. 2:

```
\emSetField{1/T, 0/TPT, 1/T}
\begin{eagle-map*}
   \emSetOnePlayerLines{playerOne}{1,3,4}
\end{eagle-map*}
```

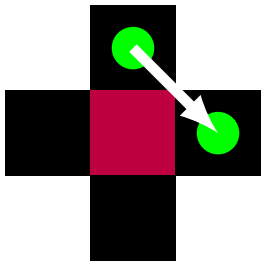


▷ `\emSetOnePlayerColumns{player}{ColID-List}`

As `\emSetPlayerColumns` → p. 5 is equivalent to `\emSetPlayerLines` → p. 4 this is the equivalent to `\emSetOnePlayerLines` → p. 5.

By the way, you can refer to the placed player-tokens as well, they will be numbered with each call to `\<player>set` (which `\emSetPlayerLines` → p. 4 and `\emSetPlayerColumns` → p. 5 will both execute) to be **<player-short><number>**. Let's reuse the example from `\emSetPlayerLines` → p. 4:

```
\emSetField{1/T, 0/TPT, 1/T}
\begin{eagle-map*}
    \emSetPlayerLines{1/playerOne, 4/playerOne}
    \draw[-latex,white,line width = 4pt]
          (playerOne1) -- (playerOne2);
\end{eagle-map*}
```



3 Included Extensions

3.1 Chess

To Load this extension, you have to pass **chess** as an argument while loading this package: `\usepackage[chess]{eagle-maps}` . Loading this Package will create the Players **black** and **white**. Furthermore it will cause the **chessfss**-Package to be loaded.

▷ `env@em-chess*`

Will create a Chess-Field using `env@eagle-map*` → p. 3. Inside of this chess-field you can access individual tiles by their (uppercase) chess-coordinate (like **A7**, **G5**, …).

▷ `env@em-chess{FileName}`

Will create a Chess-Field using `env@eagle-map` → p. 3 passing the **FileName**. See `env@em-chess*` → p. 6.

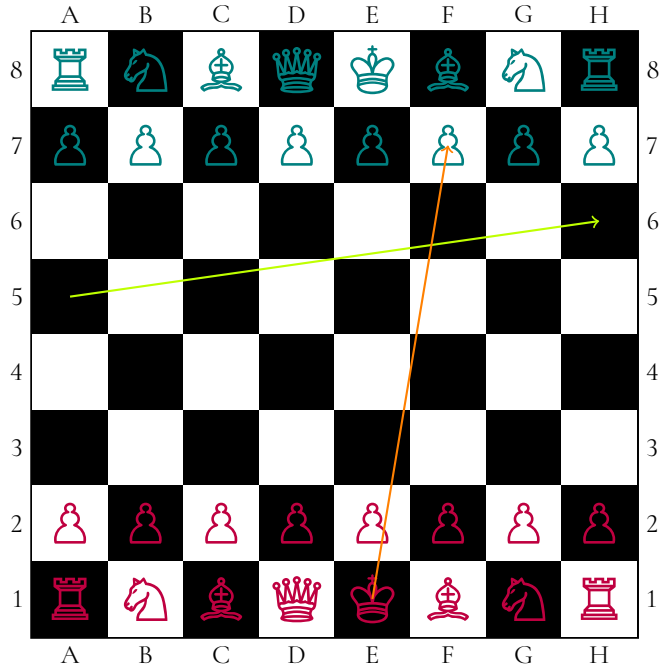
▷ `\emChessSetBoard{BoardList}`

Takes a 2-Dimensional List of tokens. **white** will set a white pawn, **black** a black pawn (please note, that the default colors are red and blue, as i love them :P). All other figures can be accessed via **<color>:<figure-name>**. So: **white:king**, **black:queen** and so on. If the Figure has a twin (like the rooks), they're named **rookA** and **rookB** respectively. Furthermore they will create coordinates named **<color><figure-name>**. The pawns will be lettered from **A-H**. Let's see an example:

```
\begin{em-chess*}
    \emChessSetBoard{%
        {black:rookB, black:knightB, black:bishopB, black:queen, black:king, ↩
            black:bishopA, black:knightA, black:rookA},
        {black,black,black,black,black,black,black,black},
        {},
        {},
        {},
        {},
        {white,white,white,white,white,white,white,white},
        {white:rookA, white:knightA, white:bishopA, white:queen, white:king,↩
            white:bishopB, white:knightB, white:rookB}%
    }

    \draw[lime,thick,->] (A5) -- (H6);
    \draw[orange,thick,->] (whiteking) -- (blackpawnF);
\end{em-chess*}
```

Results in:



3.2 Hexxagon

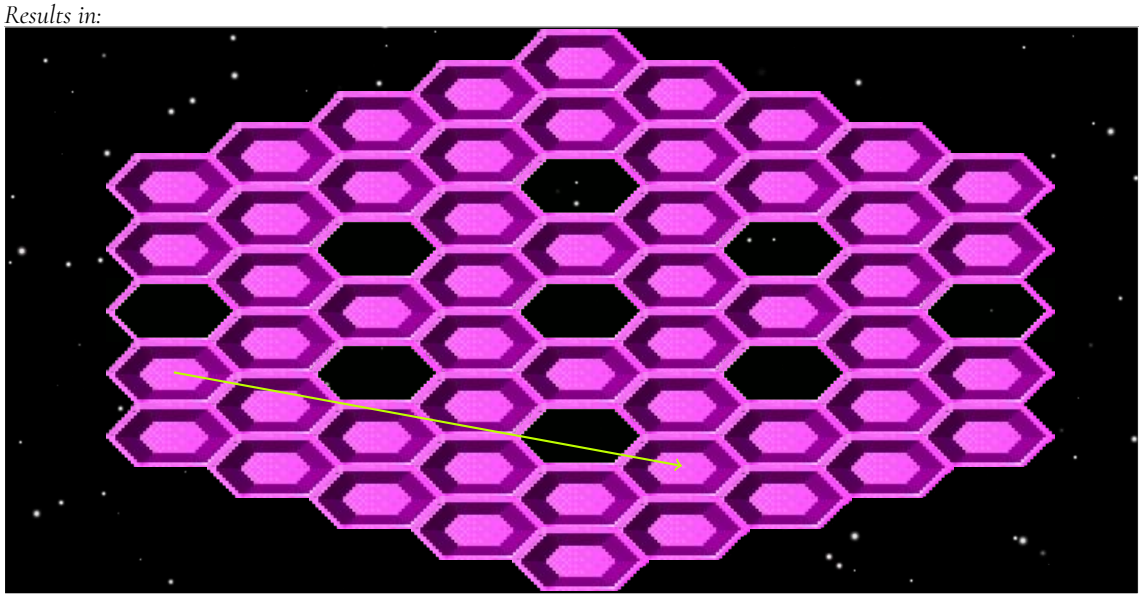
This extension is work in progress. You can enable it by adding **hexxagon** to the list of package-options: `\usepackage[hexxagon]{eagle-maps}`. Furthermore this Package needs the folder **eagle-maps-images** to be located in the same directory! Loading this Package will create the Players **playerA** and **playerB** using `\emCreatePlayer` → p. 2. Furthermore `\emCalculateCols` → p. 4 will be execute with the correct data.

▷ env@em-hexxagon* *[FieldData]*

Will create a Hexxagon-Field using env@eagle-map* → p. 3:

```
\begin{em-hexxagon*}
    \draw[lime,thick,->] (emcol4) -- (emcol42);
\end{em-hexxagon*}
```

Results in:



Please Note! If you provide a custom Field `\emCalculateCols` → p. 4 *won't be executed. You can use the default Map-Data used for the included-field by executing* `\emHexxagonDefaultCols` → p. 9 *inside of the* env@em-hexxagon* → p. 8.

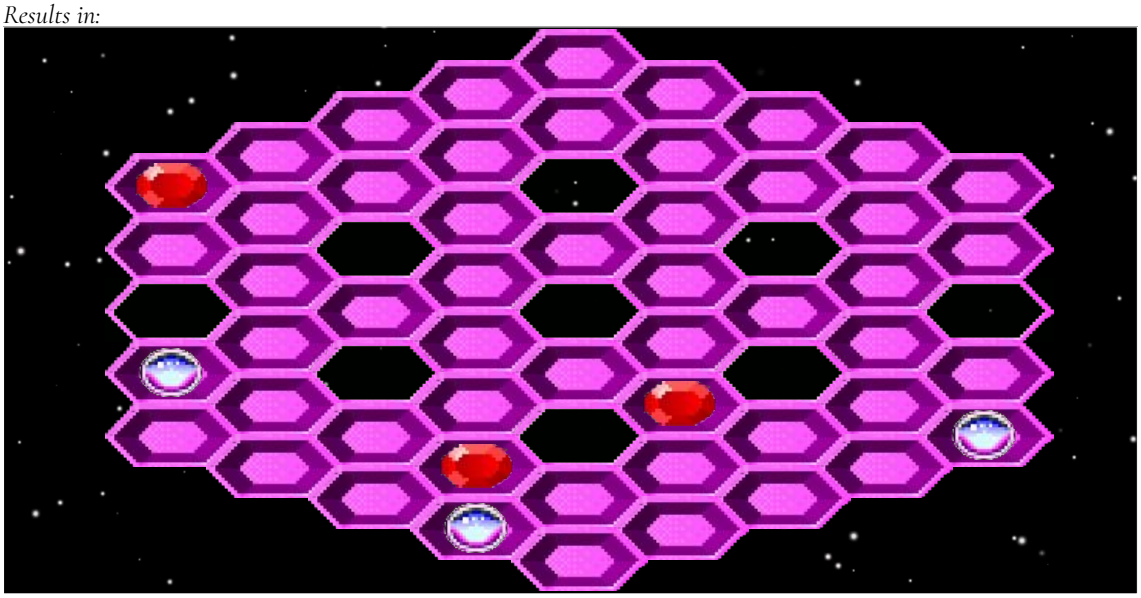
▷ env@em-hexxagon *[FieldData]* {FileName}

Will create a Hexxagon-Field using env@eagle-map → p. 3 passing the **FileName**. See env@em-hexxagon* → p. 8 (the rule regarding **FieldData** applies!).

A little example:

```
\begin{em-hexxagon*}
    \emSetPlayerColumns{%
        1/playerA, 4/playerB, 25/playerA,
       26/playerB, 41/playerA, 61/playerB}
\end{em-hexxagon*}
```

Results in:



▷ `\emHexxagonDefaultCols`

Will call `\emCalculateCols` → p. 4 with the default map-Data for Hexxagon-Fields.

This is another example for the usage of the Hexagon field:

```
\begin{center}
    \resizebox{0.5\linewidth}{!}{
        \begin{em-hexxagon*}
            \foreach \i in {1,...,61}{
                \node[rectangle,fill=white, fill opacity=0.75] at (emcol\i) {\i↩
                    };
            }
        \end{em-hexxagon*}
    }\resizebox{0.5\linewidth}{!}{
        \begin{em-hexxagon*}
            \emSetOnePlayerColumns{playerA}{22,23,30,32,39,40}
            \emSetOnePlayerColumns{playerB}{1,5,27,35,57,61}
        \end{em-hexxagon*}
    }
\end{center}
```

Results in: