The \texttt{wargame} package

Christian Holm Christensen

February 3, 2024

Abstract

This package provides tools to typesetting manuals, board, and counters for wargames using \LaTeX. Licensed under Creative Commons Attribution-ShareAlike International License, version 4.

Contents

1 Introduction 4

2 Hex Boards 4

2.1 Placing hexes 4

2.2 Hex bevels 6

2.3 Styling hexes 6

2.4 Hex coordinate system 7

2.5 Terrains 8

2.5.1 Styling terrains 11

2.6 Ridges 11

2.6.1 Styling ridges 12

2.7 Labels 12
5.4.3 Hex coordinate system .................................................. 60
5.4.4 Hexes ........................................................................... 66
5.4.5 Terrain ........................................................................... 73
5.4.6 Ridges ........................................................................... 297
5.4.7 Towns ........................................................................... 300
5.4.8 Labels ........................................................................... 302
5.4.9 Extra graphics ................................................................. 304
5.4.10 Some macros ................................................................. 305
5.4.11 Edges, borders, roads, rivers, and so on ......................... 305
5.4.12 Other paths .................................................................... 308
5.4.13 Move, attacks, retreats from hex to hex ......................... 308
5.4.14 Board clipping and frame ............................................. 312
5.4.15 Board splitting ............................................................. 318

5.5 The \tikzlibrary{wargame.chit} library .................................. 327
5.5.1 Debugging ...................................................................... 327
5.5.2 The \chitkeynamespace .................................................. 327
5.5.3 The \chitstyles ............................................................... 329
5.5.4 The \chitshape ............................................................... 330
5.5.5 The \chitwrappermacro .................................................. 337
5.5.6 Predefined \chitelementpictures ................................... 338
5.5.7 Modifications to chits .................................................... 340
5.5.8 Stacking of chits ............................................................ 340
5.5.9 Making order of battle charts ....................................... 341
5.5.10 Table of chits .............................................................. 347
5.5.11 Battle markers ............................................................. 351
5.5.12 Dice ............................................................................ 352
5.5.13 Some utilities ............................................................. 356

5.6 The \tikzlibrary{wargame.natoapp6c} library ......................... 359
5.6.1 Debugging ...................................................................... 359
5.6.2 Colours .......................................................................... 359
5.6.3 Some dimensions .......................................................... 360
5.6.4 Some utilities ............................................................... 360
5.6.5 Faction names as macros .............................................. 361
5.6.6 Node shapes ................................................................. 361
5.6.7 'Friendly' node shapes .................................................. 363
5.6.8 'Hostile' node shapes .................................................... 370
1 Introduction

This package provides tools for typesetting classic, hex-based wargames. The package allows an author to design a board, or map, comprised of hex, using a relatively simple interface. Units are typeset using a similar interface. Unit types are identified using the NATO Joint Military Symbology [2] standard.

This document is meant as a reference manual (although far from complete). A separate tutorial is available, and may be the best starting point.

2 Hex Boards

The package provides a number of facilities to set-up a board comprised of hexagon fields (“hexes”).

2.1 Placing hexes

A hex can be added to the current \tikzpicture using the macro \hex. It takes up to 4 arguments

\hex\{\{key-value-pairs\}\}(\{location\})(\{name\})

The \{key-value-pairs\} specify the hex. Valid options are

\texttt{terrain=\{terrain-keys\}} specifies the terrain of the hex. More on in this in Section 2.5.
\texttt{ridges=\{ridges-keys\}} specifies where ridges are drawn in the hex. Section 2.6.
\texttt{label=⟨label-keys⟩} specifies the how to output the hex label, if any. This is expanded upon in Section 2.7.

\texttt{town=⟨town-keys⟩} specifies that a town (or similar) is present in the hex. The various keys are described in Section 2.8.

\texttt{bevel=⟨bevel-keys⟩} specifies that a bevel should be added to the hex. The various keys are described in Section 2.2.

\texttt{extra=(extra-keyx) and extra clipped=(extra-keyz)} allows the user to put custom graphics in the hexes. See also Section 2.9 for more.

\texttt{row=⟨row⟩ and column=⟨column⟩} Keys to set hex coordinates. Mainly used when using \texttt{\node} rather than \texttt{\hex}. These coordinates should be specified in the \texttt{hex cs} coordinate system (Section 2.4).

\texttt{any style key defined for Ti\textit{k}Z pictures.}

The \texttt{(location)} argument specifies the coordinates, in the hex coordinate system where to put the hex. More about the coordinate system is given in Section 2.4. Note, the numbers by default starts from the lower–left corner, but can be changed via options.

The elements are rendered in the following order

1. The terrain, clipped to the hex shape.
2. The hex, including circumference and fill
3. The ridges, if any
4. The label, if any
5. Extra graphics clipped to the hex
6. Bevel if selected
7. Town, if any
8. Extra graphics which may extend beyond the confines of the hex.

Figure 1 illustrates some of the components of a hex. The hexes are 2 unit lengths wide. Typically, the unit length is one centimetre, which means the hexes are roughly 2 cm × 1.86 cm — or roughly 3/4” × 3/4” — big. This allows the hexes to fit chits (see Section ??) of size 12 mm × 12 mm — or roughly 1/2” × 1/2” — nicely. If one wants larger chits or hexes one should take care to scale both by a similar amount.

Note that the macro \texttt{\hex} is really a short hand for Ti\textit{k}Z’s \texttt{\node} macro, but with preset options. An alternative to using the \texttt{\hex} macro is to do

\begin{verbatim}
\node[hex={(key-value-pairs)}] ([name]) at ([location]);
\end{verbatim}

This can be useful when placing explanatory graphics or the like. The main difference between using \texttt{\hex} and the raw \texttt{\node[hex=...]} is that the former can automatically generate labels and set shape coordinates in the picture. If you want that for your board, it is recommended to use \texttt{\hex}. For example, if one does

\begin{verbatim}
\begin{tikzpicture}
    every hex=label={auto=alpha column},
    hex/labels is name=true
\end{tikzpicture}
\end{verbatim}
then one can refer to the location of the hex by its label i.e., \((A1)\). Since the hex is really a Ti\(\LaTeX\) node, we can also use anchors defined for hex node shape, such as \((A1.\text{west})\), \((A1.\text{north edge})\), and so on. This is not possible if one uses the \texttt{node} macro.

### 2.2 Hex bevels

A bevel (or “shadow-effect”) can be added to hexes using the key \texttt{bevel}, with a value that specifies where the light comes from (e.g., north west or NW). The percentage of the half width of a chit of the bevel can be specified by the key \texttt{bevel fraction} (default 10%).

### 2.3 Styling hexes

Typical Ti\(\LaTeX\) options can be passed to the \texttt{hex} macro. For example, if you want to draw the hex borders in red, simply pass \texttt{draw=red} in the \texttt{[\langle optional\rangle]} arguments to \texttt{hex}. Individual parts of the hexes can be styled separately. The default style used by \texttt{hex} is \texttt{tikz/hex/hex}. Users can redefine this style to suit their needs. If one does not want to change the default style, or pass the same argument to all \texttt{hexes} one can define the style \texttt{tikz/every hex}. For example, if one wants to auto label all hexes, one can do

\begin{tikzpicture}
\begin{scope}[every hex/.style={label=auto}]
% Hexes
\end{scope}
\end{tikzpicture}
For example, to render only the corners of the hexes, as popular among some designers, one can do

```latex
every hex/.style={
    dash pattern=on .2cm off .6cm on .2cm off 0cm
},
```

Note that the dash pattern should be 1cm long and the last element should be off 0cm so the dash pattern is started afresh on each hex edge.

### 2.4 Hex coordinate system

The package defines a coordinate system based on hexes. The centre of a hex is specified as \( (\text{column})-(\text{row}) \) pairs, while vertexes and mid-point on edges can be specified separately. The syntax of the coordinates is

\[
(\text{hex cs}:\text{row}=(\text{hex-row}),\text{column}=(\text{hex-column}),\text{vertex}=(\text{vertex}),\text{edge}=(\text{edge}))
\]

where \( (\text{vertex}) \) and \( (\text{edge}) \) are optional. The hex row and column defaults both to 0 and can be decimal numbers. The \text{row}, \text{column}, \text{vertex}, and \text{edge} keywords may be shortened to \text{r}, \text{c}, \text{v}, and \text{e}, respectively. Possible vertexes and edges are listed in Table 1.

<table>
<thead>
<tr>
<th>vertex</th>
<th>Angle</th>
<th>edge=</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>east</td>
<td>0°</td>
<td>north east</td>
<td>NE 30°</td>
</tr>
<tr>
<td>north east</td>
<td>NE 60°</td>
<td>north</td>
<td>N 90°</td>
</tr>
<tr>
<td>north west</td>
<td>NW 120°</td>
<td>north west</td>
<td>NW 150°</td>
</tr>
<tr>
<td>west</td>
<td>W 180°</td>
<td>south west</td>
<td>SW 210°</td>
</tr>
<tr>
<td>south west</td>
<td>SW 240°</td>
<td>south</td>
<td>S 270°</td>
</tr>
<tr>
<td>south east</td>
<td>SE 300°</td>
<td>south east</td>
<td>SE 330°</td>
</tr>
</tbody>
</table>

Table 1: Vertex and edge positions

In Figure 2 is an example of a picture drawn in this coordinate system.

Hexes and lines drawn with

```latex
\hex(0,0)\hex(0,1)\hex(1,0)\hex(1,1)
\draw[blue!50!black] (hex cs:r=0,c=0) -- (hex cs:r=1,c=1);
\draw[red!50!black] (hex cs:r=0,c=0,vertex=E) -- (hex cs:r=1,c=1,edge=NE);
\fill[lightgray](hex cs:r=.3,c=.3) circle(0.1);
\fill[lightgray](hex cs:r=1.3,c=.3) circle(0.1);
\fill[lightgray](hex cs:r=1.3,c=1.3) circle(0.1);
```

Figure 2: Hex coordinate system
**Important:** When the horizontal distance to the centre of a hex becomes less than $-\cos 60^\circ$ or larger than $b - \cos 60^\circ$ we effectively have a new hex column, and the coordinates are shifted upward or downward for smaller or larger numbers. Figure ?? illustrates this. This can make it a little hard to specify coordinates relative to a hex centre. Alternatively one may use vertex or edge specifications together with a relative offset in those directions. If one require even more flexibility, one can use the TikZ library `calc` to add arbitrary offsets, e.g.,

\begin{verbatim}
\coordinate at ($(hex cs:c=1,r=10)+(.2,.2)$);
\end{verbatim}

2.5 Terrains

Terrains are rendered using tile images or TikZ pictures. The available terrains are shown in Tables 2 and 3. Users can provide their own tile images and select those via `terrain={image=⟨graphics-file⟩}` or defined TikZ pictures and select those via `terrain={pic=⟨picture-key⟩}`. In all cases, the terrain graphics is clipped to the hex.

The terrain of a hex is selected via the multi-valued key `terrain`. Sub-keys of this key are

- `image=⟨graphics-file⟩` Specifies terrain tile image ⟨graphics-file⟩.
- `pic=⟨picture-key⟩` Specifies terrain tile TikZ picture.
- `code=⟨tikz-code⟩` Any valid TikZ code
- `clip=⟨path(s)⟩` TikZ path specification to clip the terrain within the hex.

The terrain can be clipped by the sub-key `clip`. This can be useful if the game specifies movement costs in terms of hex-edge crossing, for example *First Blood* [1]. In that case, a hex may be, for example, a jungle hex, but some edges are clear. Thus movements across such an edge would count as moving into clear territory while moving over other edges will count as moving into a jungle. This is, of course, not how most games count movement costs, but this package nonetheless facilitates such rules. Table 4 shows a few examples of predefined clippings of terrain.

Users can define TikZ pictures that specify clipping paths as needed. For example, one could add clipping to the terrain to ensure that other graphics in the hex stands out.
### Table 2: Terrains specified via tile images

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>terrain={image=⟨image⟩}</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image=wargame.beach" alt="Clear" /></td>
<td>Clear</td>
<td><img src="image=wargame.beach" alt="Clear" /></td>
</tr>
<tr>
<td><img src="image=wargame.light_woods" alt="Light woods" /></td>
<td>Light woods {image=wargame.light_woods}</td>
<td><img src="image=wargame.light_woods" alt="Light woods" /></td>
</tr>
<tr>
<td><img src="image=wargame.rough" alt="Rough" /></td>
<td>Rough</td>
<td><img src="image=wargame.rough" alt="Rough" /></td>
</tr>
<tr>
<td><img src="image=wargame.mountains" alt="Mountains" /></td>
<td>Mountains {image=wargame.mountains}</td>
<td><img src="image=wargame.mountains" alt="Mountains" /></td>
</tr>
<tr>
<td><img src="image=wargame.beach" alt="Beach" /></td>
<td>Beach</td>
<td><img src="image=wargame.beach" alt="Beach" /></td>
</tr>
<tr>
<td><img src="image=wargame.woods" alt="Woods" /></td>
<td>Woods</td>
<td><img src="image=wargame.woods" alt="Woods" /></td>
</tr>
<tr>
<td><img src="image=wargame.swamp" alt="Swamp" /></td>
<td>Swamp</td>
<td><img src="image=wargame.swamp" alt="Swamp" /></td>
</tr>
<tr>
<td><img src="image=wargame.sea" alt="Sea" /></td>
<td>Sea</td>
<td><img src="image=wargame.sea" alt="Sea" /></td>
</tr>
</tbody>
</table>

### Table 3: Terrains specified via TikZ pictures

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>terrain={pic=⟨image⟩}</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Mountains](image=hex/terrain/mountain,line width=3pt)</td>
<td>Mountains {pic=hex/terrain/mountain,line width=3pt}</td>
<td>![Mountains](image=hex/terrain/mountain,line width=3pt)</td>
</tr>
</tbody>
</table>

Table 2: Terrains specified via tile images

Table 3: Terrains specified via TikZ pictures
<table>
<thead>
<tr>
<th>Symbol</th>
<th>terrain={clip=,...}</th>
<th>Symbol</th>
<th>terrain={clip=,...}</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Symbol" /></td>
<td>{hex/sextant=NE}</td>
<td><img src="image2" alt="Symbol" /></td>
<td>{hex/large sextant=NE, hex/large sextant=N}</td>
</tr>
<tr>
<td><img src="image3" alt="Symbol" /></td>
<td>{hex/sextant=N}</td>
<td><img src="image4" alt="Symbol" /></td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=NW}</td>
</tr>
<tr>
<td><img src="image5" alt="Symbol" /></td>
<td>{hex/sextant=SW}</td>
<td><img src="image6" alt="Symbol" /></td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=SW, hex/sextant=S}</td>
</tr>
<tr>
<td><img src="image7" alt="Symbol" /></td>
<td>{hex/sextant=S}</td>
<td><img src="image8" alt="Symbol" /></td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=SW, hex/sextant=S, hex/sextant=SE}</td>
</tr>
<tr>
<td><img src="image9" alt="Symbol" /></td>
<td>{hex/sextant=SE}</td>
<td><img src="image10" alt="Symbol" /></td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=SW, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
<tr>
<td><img src="image11" alt="Symbol" /></td>
<td>{hex/sextant=C}</td>
<td><img src="image12" alt="Symbol" /></td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=SW, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
</tbody>
</table>

Table 4: Terrain clipped via clip sub-key
2.5.1 Styling terrains

Terrains use the key `tikz/hex/terrain` to render the terrains. This is mainly useful for terrains made from TikZ pictures.

2.6 Ridges

Ridges, or hill or mountain slopes, can be added to a hex via the keyword `ridges`. The keyword takes a list of hex edges and generates symbology for the ridge on the chosen edges. Note that the edges does not have to be continuous, as illustrated in the bottom right of Table 5, nor in any particular order. The edges are specified as compass direction `north east, north, north west, south west, south, south east.`

Table 5 shows some examples.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>ridges=</th>
<th>Symbol</th>
<th>ridges=</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>NE,N</td>
<td>N</td>
<td>NE,N,NW</td>
</tr>
<tr>
<td>N</td>
<td>NE,N,NW</td>
<td>NW</td>
<td>NE,N,NW,SW</td>
</tr>
<tr>
<td>NW</td>
<td>NE,N,NW,SW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW</td>
<td>NE,N,NW,SW,S,line width=3pt</td>
<td>S</td>
<td>NE,N,NW,SW,S,SE,color=brown!70!black</td>
</tr>
<tr>
<td>S</td>
<td>NE,N,NW,SW,S,SE,color=brown!70!black</td>
<td>SE</td>
<td>N,S,NW,SE</td>
</tr>
</tbody>
</table>

Table 5: Ridges
2.6.1 Styling ridges

Every ridge is drawn with the style \texttt{tikz/hex/ridges}. Users can customise this style. The default is to draw thin black wave lines (TikZ decoration \texttt{waves}). The default style also takes care to auto scale line widths.

2.7 Labels

Labels can be placed on the hexes via the keyword \texttt{label}. The label can either be auto-generated or given explicitly. Table 6 shows the various choices.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Column/Row</th>
<th>label=</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="No label" /></td>
<td>No label</td>
<td>n/a</td>
<td>none</td>
</tr>
<tr>
<td><img src="image" alt="User specified" /></td>
<td>User specified</td>
<td>n/a</td>
<td>text=B10</td>
</tr>
<tr>
<td><img src="image" alt="User specified" /></td>
<td>User specified</td>
<td>n/a</td>
<td>{color=blue,text=A1}</td>
</tr>
<tr>
<td><img src="image" alt="Two-digit, zero padded numbers" /></td>
<td>Two-digit, zero padded numbers</td>
<td>9/3</td>
<td>auto</td>
</tr>
<tr>
<td><img src="image" alt="Column letter, number row" /></td>
<td>Column letter, number row</td>
<td>2/3</td>
<td>{auto=alpha column,font=\noexpand\rmfamily} \textsuperscript{†}</td>
</tr>
<tr>
<td><img src="image" alt="Two letter column, two digit row" /></td>
<td>Two letter column, two digit row</td>
<td>6/24</td>
<td>{auto=alpha 2 column,anchor=north east}</td>
</tr>
</tbody>
</table>

Table 6: Labels

\textsuperscript{†}When specifying macros as key values in the options, for example the value \texttt{\noexpand\rmfamily} for the key font above, we have to put a \texttt{\noexpand} in front if the macro. This is to prevent early expansion of the macro, which would cause errors. A minor nuisance.

The option \texttt{auto=inv y x plus 1} will label the rows inversely, and add one to the column number. This requires that the key \texttt{tikz/max hex row} has been set to the largest row number used.

In addition to the sub-keys \texttt{none}, \texttt{auto}, and \texttt{text}, one can also specify the following keys

\texttt{place=(coordinates)} specifies the Location of label within the hex. The anchor point of the text will be placed at
this point.

[(f)options ] at the start of the option (but inside braces {...}) can be used to give additional style options.

2.7.1 Styling labels

All labels use the style tikz/hex/label. By default, this places the label at the top of the hex, and renders the text as gray script sized text. Users can customise this style. If a user does not want to change the default style, or want to pass the same option to all labels, then one can set the key tikz/every label to those options.

2.8 Towns

Towns in hexes are made via the key town. This key takes several sub-keys, as illustrated in Table 7

<table>
<thead>
<tr>
<th>Symbol</th>
<th>town=</th>
<th>Symbol</th>
<th>town=</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Hexagon" /></td>
<td>{pic=hex/town/city}</td>
<td><img src="image2.png" alt="Hexagon" /></td>
<td>{name=Copenhagen}</td>
</tr>
<tr>
<td><img src="image3.png" alt="Hexagon" /></td>
<td>{fill=red}</td>
<td><img src="image4.png" alt="Hexagon" /></td>
<td>{name=London}</td>
</tr>
<tr>
<td><img src="image5.png" alt="Hexagon" /></td>
<td>{red,name=Paris}</td>
<td><img src="image6.png" alt="Hexagon" /></td>
<td>{above=0.8,name=Berlin}</td>
</tr>
<tr>
<td><img src="image7.png" alt="Hexagon" /></td>
<td>{place={(.2,.2)}}</td>
<td><img src="image8.png" alt="Hexagon" /></td>
<td>{font=\noexpand\itshape,name=Amsterdam}</td>
</tr>
</tbody>
</table>

Table 7: Towns

†When specifying macros as key values in the options, for example the value \rmfamily for the key font above, we have to put a \noexpand in front if the macro. This is to prevent early expansion of the macro, which would cause errors. A minor nuisance.

The sub-keys available for the town key are

- **pic=(town-pic)** The name of a TikZ picture. Currently defined are hex/town/town and hex/town/city. Users can provide alternate definitions or new types by defining TikZ pictures.

- **place=(coordinates)** Location of label within the hex. The anchor point of the text will be placed at this point.
name=⟨name⟩ Name of town

### 2.8.1 Styling towns

Towns uses two styles: `tikz/hex/town` for the town graphics, and `tikz/hex/town name` for the name of the town. In addition, a user may set the key `tikz/every hex town` to contain options to be passed to all towns.

### 2.9 Extra graphics for hexes

Additional graphics for hexes can be added by the two keys `extra` and `extra clipped`. The difference between the two are that graphics specified by `extra clipped` are clipped (restricted) to the hex, while graphics given by `extra` may extend beyond the hex. Both keys accept a comma separated list of arguments, where each element has the syntax:

\[
[(options)](⟨placement⟩)(⟨picture⟩)
\]

Both `⟨options⟩` and `⟨placement⟩` are optional, and specifies keys to draw `⟨picture⟩` with and the relative location in the hex, respectively. The required argument `⟨picture⟩` must name a TikZ picture, for example `hex/fortress`. This can be useful for marking hexes on the board. For example to mark a set-up hex for one faction of the game.

One could for example define the following pictures to define set-up points for a Soviet and German faction:

```latex
setup/sovjet/.pic={
\path[fill=red,draw=yellow,pic actions]
  ( 90:.4)--(126:.15)--
  (162:.4)--(198:.15)--
  (234:.4)--(270:.15)--
  (306:.4)--(342:.15)--
  ( 18:.4)--( 54:.15)--cycle;},
setup/german/.pic={
\path[fill,pic actions]
  (-.4, -.1) rectangle(.4,.1)
  (-.1, -.4) rectangle(.1,.4);
\path[draw,pic actions]
  (-.4,-.2) -- (-.2,-.2) -- (-.2,-.4)
  (-.4,.2) -- (-.2,.2) -- (-.2,.4)
  (.4,.2) -- (.2,.2) -- (.2,.4)
  (.4,-.2) -- (.2,-.2) -- (.2,-.4);}
foo/large/.pic={
\path[fill=gray,pic actions] (-1,-.5) rectangle(1,.5);}.
}\}
```

We can place extra graphics in hexes as shown in Table 8.

To finish off this part on hexes and what we can do with those, we generate a map in Figure 4.

### 2.10 Rivers, borders, and roads

Rivers and borders follow the hex sides and are added to the current TikZpicture using \river and \border macros respectively. They are specified as regular TikZ paths. It is useful to utilise the hex coordinate system for this.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>extra=</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="hex/fortress" /></td>
<td>hex/fortress</td>
</tr>
<tr>
<td><img src="image" alt="setup/sovjet" /></td>
<td>setup/sovjet</td>
</tr>
<tr>
<td><img src="image" alt="setup/german" /></td>
<td>setup/german</td>
</tr>
<tr>
<td><img src="image" alt="setup/german,hex/fortress" /></td>
<td>{setup/german,hex/fortress}†</td>
</tr>
<tr>
<td><img src="image" alt="foo/large" /></td>
<td>foo/large</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>extra clipped=</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="foo/large" /></td>
<td>foo/large</td>
</tr>
</tbody>
</table>

Table 8: Hex extra graphics. Note that in the last line we use the graphics foo/large with extra clipped (compare to line just above) to restrict the graphics to the hex.

†When specifying more than one item, the list must be enclosed in braces ({...})

‡When an item in the list of extra contains a comma (,), for example in a list of graphics options, then we need to enclose the inner list and the whole list in braces ({...}) to protect against unwanted expansion.
Rivers are essentially borders, but are randomized to give a more aesthetically pleasing output.

Roads and railroads typically go from hex–center to hex–center, and are added using the macro \texttt{road}. The road or railroad is specified via a regular TikZ path.

\begin{align*}
\texttt{river} & \texttt{[\langle options\rangle]} \ (\texttt{path}) ; \\
\texttt{border} & \texttt{[\langle options\rangle]} \ (\texttt{path}) ;
\end{align*}

Towns and cities conveniently serve as places to split up a road at.

\textbf{2.10.1 Styling paths}

Rivers, roads, railroads, and borders are styled by \texttt{hex/river}, \texttt{hex/road}, \texttt{hex/railroad}, and \texttt{border}, respectively, and the keys \texttt{every hex river}, \texttt{every hex road}, \texttt{every hex railroad}, and \texttt{every hex border} will also be applied. The latter can be defined by the user.

\textbf{2.11 Board clipping and frame}

In the river, border, and road example above, the roads extend beyond the hexes, which does not look very nice. One way to deal with this, is to draw a clipping box around the hexes.

This technique works fine for examples in a manual, it has a somewhat displeasing effect for a full board. The package therefore defines the macro \texttt{boardclip} which clips the graphics according to the defined hexes.
Figure 5: Adding rivers, boarders, and roads

Figure 6: Clipping for a manual using a \texttt{TikZ} \texttt{draw[clip]} command.
\boardclip((lower-left))((upper-right)){(options)}

A clipping path of that spans from the hex at (lower-left) to (upper-right). Note, that both of these arguments should only specify the column and row keys. If (options) is non-empty, then the clipping path is drawn with those options.

\begin{scope}
\boardclip(c=0,r=0)(c=3,r=3)\{}
\testmap
\end{scope}

Figure 7: Snug-fit clipping of board using the macro \boardclip

This is particularly useful together with the \boardframe macro. This macro will put a frame around the board, optionally with a margin.

\boardframe[(margin)]((lower-left))upper-right

where (lower-left) and (upper-right) are as for \boardclip. The (margin) must be a number, and specifies an optional margin around the hexes. Thus argument (options) specifies how the frame is drawn. The idea is to first draw the frame, then the clipping shape, and then the hexes. One should take care to use the (options) argument to \boardclip to specify a default background color. The frame is drawn with the style hex/board frame.

The \boardframe macro prints the position of the rectangle to the log output, if one needs to do some more stuff around the board.

2.12 Constructing the physical board

If the board is not too large, so that it may fit on a paper format that can easily be printed (say A4, A3, Letter, or Tabloid), one can simply print the board and glue it onto a sturdy surface (say 1½ mm poster carton). However, if the board is large, meaning it does not fit on a piece of printable paper, then one has two options.

Either scale the board down so that it fits. Use the TikZ key scale=\textit{factor} as an argument to the \texttt{tikzpicture} environment in which you create the board. In this case, you should make sure you also scale the chits by the
same \textit{factor}, again via the \texttt{scale} key.

\textbf{Or} you can split the board over several pages. The package provides a number of tools to help with this.

### 2.12.1 Split the board over multiple sheets

First, make sure you produce a standalone PDF of the board only.

\begin{verbatim}
\documentclass{standalone}
\usepackage{wargame}
\begin{tikzpicture}[scale=SCALE]
% Define the board here.
\end{tikzpicture}
\end{verbatim}

and that you have created this PDF — say \texttt{board.pdf}.

Next, prepare another document in which we will do the calculations. For example

\begin{verbatim}
\documentclass[11pt]{standalone}
\usepackage{wargame}
\begin{document}
\splitboard{paper=letter,margin=.7,ncol=2,nrow=2,overlap=1}
\end{document}
\end{verbatim}
to calculate the split of board.pdf over $2 \times 2$ letter paper sheets, with a non-printable margin of 7mm, and an overlap between the segments of 1cm.

The possible keys for the \splitboard macro are

- **paper=(format)**: Specifies the paper format. One of a4, a3, letter, tabloid. Default is a4.
- **landscape**: Sets the paper format to be in landscape mode (default is portrait).
- **margin=(size in centimetres)**: Size of margins on each sheet in centimetres without unit. That is put 0.6 for 6mm, not 6mm. Default is 0.6. This should be slightly larger (by roughly 5%) than the least margin required by the printer used. Must be given before paper to have any effect.
- **ncol=(number of columns)**: Sets the number of columns of sheets.
- **nrow=(number of rows)**: Set the number of rows of sheets.
- **overlap=(size in centimetres)**: Sets the size of the overlap between sheets in centimetres without unit. That is put 2 for 2cm, not 2cm. Default is 2.
- **image=(image file name)**: File name of the board image (a PDF). Default is board
- **output=(output file name)**: File name (without .tex ending) to write calculated split to.
- **standalone**: Boolean flag. If true, then output file will be a standalone document (i.e., has a \documentclass).
- **scale=(scale)**: Set scale of board.

The macro will produce a file named (output file name).tex which can be included in another document to generate the split board PDF. Crop marks will be added to the board segments to make it easier to align the parts.

### 2.12.2 Foldable board

To make a fold-able board use for example the below template to create grooves and cuts.

![Foldable board template](image)
This will fold the board down to a fourth of the size of the full map. For example, if the board is A1 (84 cm × 59.4 cm) it will fold down to A4 (21 cm × 29.7 cm) for easier storage.
3 Chits

Chits, or playing counters, can be made with the macro \chit. The syntax for rendering a chit is

\chit{⟨key-value-pairs⟩}{⟨location⟩}{⟨name⟩}

Figure 9 shows an example of a chit.

\begin{figure}[h]
\centering
\begin{tikzpicture}
\node[chit={symbol=⟨nato app 6(c) spec⟩, left={chit/identifier={1}}, factors={chit/2 factors={4,8}}}, right={chit/identifier={2}}] at (0,0) {
\begin{tabular}{c}
1 \\
0.6 \\
0 \\
-0.6 \\
-1
\end{tabular}
\begin{tabular}{c}
-1 \\
0 \\
0.6 \\
1
\end{tabular}
};
\end{tikzpicture}
\end{figure}

Figure 9: An example of a chit. The lines below and to the right shows two unit lengths. Other global options used are color=white to set the foreground colour, fill=blue for the background, and text=yellow to set the font colour to yellow. The symbol key also contains frame={fill=yellow!50!black} to set the frame fill colour, and ultra thick to set the line width of the NATO App6(C) symbol. Note that the line width is automatically scaled.

The example in Figure 9 shows an infantry mountaineer army unit with attack factor 4, and movement factor 8. The NATO App6(c) symbol is given in terms of keywords for the \natoapp macro (see Section 4). The other parts of the chit (factors, left, right, and below) are rendered onto the chit via TikZ pictures. This allows for a great deal of flexibility in generating chits. For example, above we use the pictures chit/identifier and chit/2 factors to render the left- and right-hand identifiers, and the factors, respectively.

Full frame chits — that is chits which are not typically designating units or faction specific chits, e.g., a turn marker — can be made by using the key full. In that case, all other keys (symbol, factors, left, right, and below) are ignored. Figure 10 shows such an example.

The size of the chits are 1.2 × 1.2 unit lengths squared. This is tuned so that the chits will fit within the hexes produced by the \hex command (see Section ). In Figure 11 we illustrate this. Typically the unit is one centimetre, which means the chits are 12 mm × 12 mm — or roughly 1/2" × 1/2", which is a fairly good size for most games.

Just as \hex is really a wrapper around TikZ’s \node macro, so it is with \chit. This means that an alternative way of making a chit is to do

\begin{figure}[h]
\centering
\begin{tikzpicture}
\node[chit={⟨key-value-pairs⟩}]{⟨name⟩} at ⟨location⟩;
\end{tikzpicture}
\end{figure}

\footnote{Since \TeX has the concept of counters as in \texttt{\count} and its \texttt{\newcounter}, we choose the name ‘chit’ for playing pieces instead.}
chit made with
\tikzset{
    wg/big text/.pic={
        \node[font=\sffamily\fontsize{18}{0}\selectfont\{#1\};]
    }
}
\tikz{
    \chit[full={wg/big text={Turn}},
        black,fill=yellow!20!white](0,0)
}\end{tikzpicture}

Figure 10: An example of a full-frame chit.

Figure 11: Example of chits fit within hexes.

Since chits are really Ti\k Z nodes we can use anchors on the chit. Unlike for \hex where there are additional features available when using the dedicated macro, there really isn’t much difference between \chit and \node[chit=...].

3.1 Styling chits

Typical Ti\k Z options can be passed to the \chit macro. For example, if you want to draw the chit with a red foreground, simply pass draw=red in the [⟨optional⟩] arguments to \chits. Individual parts of the hexes can be styled separately.

Important: To set the colours of the various elements, one should use

- **color=(foreground and text)** Selects the foreground colour of lines, text, and so on, including for the NATO App6(C) symbol.
- **fill=(background)** Selects the background colour of the full chit. By default this is transparent.
- **text=(text foreground)** Selects the colour used for text in the chit. This overrides color for text.
- **draw=(foreground)** This sets the colour for foreground elements, excluding text.

\k Z allows one to pass a ⟨colour⟩ as arguments for drawing and understands that as giving the foreground and text colours. However, that key is deprecated for this library, as it does not properly propagate through\footnote{The colour \texttt{pgfstrokecolor} is not modified by that.}.
The styles used by the \texttt{left}, \texttt{right}, \texttt{setup}, \texttt{factors}, and \texttt{symbol} elements are \texttt{tikz/chit/left}, \texttt{tikz/chit/right}, \texttt{tikz/setup}, \texttt{tikz/factors}, and \texttt{tikz/symbol} respectively. A user can redefine these to change the appearance of the chits. For example, one could make the symbol larger by setting a different \texttt{scale}, move the factors to the side by changing \texttt{shift}, and so on.

Pictures used by these elements are also styled by similar keys. For example, the picture \texttt{chit/identifier} is styled by \texttt{tikz/chit/identifier}.

A bevel (or “shadow-effect”) can be added to chits using the key \texttt{bevel}, with a value that specifies where the light comes from (e.g., north west or NW). The percentage of the half width of a chit of the bevel can be specified by the key \texttt{bevel fraction} (default 10\%). This can be used for both symbol or full chits.

In addition, one can define the key \texttt{tikz/every chit} to be the default options for all chits.

By default, the outer “frame” of a chit is drawn with the same graphics options as the chit it self (i.e., same fill and stroke colour). To change that, one can pass \texttt{frame=\{options\}} as part of the chit options.

### 3.2 Defining preset chit types

One can conveniently pre-define some chit styles. For example, given the style definition

\begin{verbatim}
\tikzset{  
  my chit/.style={/chit/symbol={[
    faction=friendly,  
    command=land,  
    main=armoured]},  
  /chit/left={chit/identifier={Mine}},  
  /chit/factors={chit/2 factors={2,4}}}
\end{verbatim}

We can use that to make different chits with some commonalities defined by that style. For example

\begin{center}
\begin{tabular}{cc}
\includegraphics[width=0.2\textwidth]{mine.png} & \includegraphics[width=0.2\textwidth]{mine2.png}
\end{tabular}
\end{center}

where, in the second example, we have passed additional options to \texttt{chit}. Note that we \texttt{must} give the full path to the \texttt{chit} keys when defining a style like this.
4 NATO App 6(c) symbols

The NATO markers are designed to fit within the template shown in Figure 12. The template serves as a placement guide of the various parts of the NATO marker as illustrated in Figure 13.

Figure 12: Template for NATO symbols

\natoapp[(key-value-pairs)]((location))((name))

where all arguments are optional. Keys are defined to fill in the various parts of the markers. These keys are

- **faction**=(fraction) Selects the faction used for the symbol. See also Section 4.1.
- **command**=(command) Selects the command used for the symbol. See also Section 4.1.
- **main**=(mains) Specifies the main symbol(s). This can be a comma separated list of specifiers (delimited by braces \{first,second,...\}), and each symbol can be preceded by an optional argument to shift, scale, rotate, etc., the individual symbols.
- **left**=(lefts), **right**=(rights), **top**=(tops), **bottom**=(bottoms), **below**=(belows) Specifies the left-, right-hand, top, bottom, and lower symbol(s). The format of the arguments (lefts), (rights), (tops), (bottoms), and (belows) has the same format as (mains).

The figure is typeset by

\natoapp[faction=none, command=base, echelon=army, main={text=M}, top={text=T}, bottom={text=B}, left={text=L}, right={text=R}, below={text=V}]

Figure 13: Main keys of \natoapp. The bottom and right hand bars indicate one unit of length.

Other keys are available to further customise the appearance of the symbols

- **echelon**=(size) The size of the unit described. Possible values are team, squad, section, platoon, company, battalion, regiment, brigade, division, corps, army, army group, theatre, and command.

25
4.1 Faction and Command Selection

Table 9 shows the various bases used for the various *faction/command* combinations. Also shown in the table is the base template for main identifiers.

<table>
<thead>
<tr>
<th>⟨command⟩</th>
<th>friendly</th>
<th>hostile</th>
<th>neutral</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>air</td>
<td><img src="image1" alt="Frame Image" /></td>
<td><img src="image2" alt="Frame Image" /></td>
<td><img src="image3" alt="Frame Image" /></td>
<td><img src="image4" alt="Frame Image" /></td>
</tr>
<tr>
<td>land</td>
<td><img src="image5" alt="Frame Image" /></td>
<td><img src="image6" alt="Frame Image" /></td>
<td><img src="image7" alt="Frame Image" /></td>
<td><img src="image8" alt="Frame Image" /></td>
</tr>
<tr>
<td>equipment</td>
<td><img src="image9" alt="Frame Image" /></td>
<td><img src="image10" alt="Frame Image" /></td>
<td><img src="image11" alt="Frame Image" /></td>
<td><img src="image12" alt="Frame Image" /></td>
</tr>
<tr>
<td>installation</td>
<td><img src="image13" alt="Frame Image" /></td>
<td><img src="image14" alt="Frame Image" /></td>
<td><img src="image15" alt="Frame Image" /></td>
<td><img src="image16" alt="Frame Image" /></td>
</tr>
<tr>
<td>sea surface</td>
<td><img src="image17" alt="Frame Image" /></td>
<td><img src="image18" alt="Frame Image" /></td>
<td><img src="image19" alt="Frame Image" /></td>
<td><img src="image20" alt="Frame Image" /></td>
</tr>
<tr>
<td>sub surface</td>
<td><img src="image21" alt="Frame Image" /></td>
<td><img src="image22" alt="Frame Image" /></td>
<td><img src="image23" alt="Frame Image" /></td>
<td><img src="image24" alt="Frame Image" /></td>
</tr>
<tr>
<td>space</td>
<td><img src="image25" alt="Frame Image" /></td>
<td><img src="image26" alt="Frame Image" /></td>
<td><img src="image27" alt="Frame Image" /></td>
<td><img src="image28" alt="Frame Image" /></td>
</tr>
<tr>
<td>activity</td>
<td><img src="image29" alt="Frame Image" /></td>
<td><img src="image30" alt="Frame Image" /></td>
<td><img src="image31" alt="Frame Image" /></td>
<td><img src="image32" alt="Frame Image" /></td>
</tr>
</tbody>
</table>

Table 9: Frames for various combinations of ⟨faction⟩ and ⟨command⟩ combinations. These are drawn with the `pic` given by `natoapp6c/⟨faction⟩/⟨command⟩` with the options `draw=blue,fill=⟨faction⟩`. If no `fill` is specified, then the background will be transparent. Note, the template for main identifiers is also shown on top of each frame.

The fill color of the frame is set by the key `frame`. If this is or contains the special value `faction`, then the frame fill colour will be the standard for the faction as illustrated in figure 14.

Elements of the frame can be controlled by the key `frame`.

`frame=(keys)` Additional keys to pass to the frame drawing. The special option `faction` will make the frame be filled with the standard faction color.

Table 10 illustrates this.
Figure 14: Illustration of using the special value \texttt{faction} for the \texttt{frame} key

<table>
<thead>
<tr>
<th>Example</th>
<th>frame={color,...}</th>
<th>frame={fill,...}</th>
<th>frame={draw,...}</th>
<th>frame={line width,...}</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Example" /></td>
<td>red</td>
<td>yellow</td>
<td>blue</td>
<td>pink magenta</td>
</tr>
<tr>
<td></td>
<td>thick</td>
<td>thin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>red, green, blue</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Illustration of frame colour choices
4.2 Unit Size (echelon)

The size of a unit a marker represents is given by the echelon keyword. Table 11 shows the various markers and approximate unit sizes.

4.3 Unit type identification

See Table 12.

References


5 Implementation

5.1 The wargame package

First, package identification

\ProvidesPackage{wargame}

Then needed packages

\RequirePackage[svgnames]{xcolor}
\RequirePackage{tikz}

A switch to include terrain pictures (which take a lot of memory for some reason).

\Ifundefined{ifhex@terrain@pic}{\newif\hex@terrain@pic\hex@terrain@picfalse}{%\hex@terrain@pictrue}

Options

\DeclareOption{noterrainpic}{%\hex@terrain@picfalse}
\DeclareOption{terrainpic}{%\hex@terrain@pictrue}
\ProcessOptions\relax

Finally, the used Ti\textit{k}Z libraries

\usetikzlibrary{wargame.hex,wargame.natoapp6c,wargame.chit}
<table>
<thead>
<tr>
<th>Echelon</th>
<th>Approx. Size</th>
<th>Sub-units</th>
<th>Officer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team</td>
<td>3–5</td>
<td>none</td>
<td>Corporal or Sergeant</td>
</tr>
<tr>
<td>Squad</td>
<td>5–10</td>
<td>1–2 teams</td>
<td>Sergeant</td>
</tr>
<tr>
<td>Section</td>
<td>7–13</td>
<td>2–3 teams</td>
<td>Sergeant</td>
</tr>
<tr>
<td>Platoon</td>
<td>25–40</td>
<td>Several squads/sections</td>
<td>Second Lieutenant</td>
</tr>
<tr>
<td>Company</td>
<td>60–250</td>
<td>Several platoons</td>
<td>Captain</td>
</tr>
<tr>
<td>Battalion</td>
<td>300–1,000</td>
<td>2–6 companies</td>
<td>Lieutenant colonel</td>
</tr>
<tr>
<td>Regiment</td>
<td>500–2,000</td>
<td>3–7 battalions</td>
<td>Colonel</td>
</tr>
<tr>
<td>Brigade</td>
<td>2,000–5,000</td>
<td>Several battalions</td>
<td>Colonel</td>
</tr>
<tr>
<td>Division</td>
<td>10,000–20,000</td>
<td>Several brigades/regiments</td>
<td>Major General</td>
</tr>
<tr>
<td>Corps</td>
<td>30,000–60,000</td>
<td>Several divisions</td>
<td>Lieutenant General</td>
</tr>
<tr>
<td>Army</td>
<td>100,000</td>
<td>Several corps (5–10 divisions)</td>
<td>General</td>
</tr>
<tr>
<td>Army Group</td>
<td>120,000–500,000</td>
<td>Several armies</td>
<td>Field Marshal</td>
</tr>
<tr>
<td>Theatre</td>
<td>250,000+</td>
<td>Several army groups</td>
<td>Field Marshal</td>
</tr>
<tr>
<td>Command</td>
<td>Not a unit size, but designator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11: Illustration of echelon values. Approximate sizes and command officer titles are typical modern day United States of America army values and identifiers. Historically the unit sizes have changed, as has officer titles. Furthermore, both the unit sizes, names, and command officer titles may vary from country to country, even across command.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type &amp; Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚲</td>
<td>Air assault AA</td>
</tr>
<tr>
<td>⚲</td>
<td>Air defence ADA</td>
</tr>
<tr>
<td>⚲</td>
<td>Airborne AB</td>
</tr>
<tr>
<td>⚲</td>
<td>Amphibious AM</td>
</tr>
<tr>
<td>⚲</td>
<td>Anti tank/armoured AT</td>
</tr>
<tr>
<td>⚲</td>
<td>Armoured AR</td>
</tr>
<tr>
<td>⚲</td>
<td>Chemical biological radiological nuclear CAR</td>
</tr>
<tr>
<td>⚲</td>
<td>Combined arms FA</td>
</tr>
<tr>
<td>⚲</td>
<td>Engineer ENG</td>
</tr>
<tr>
<td>⚲</td>
<td>Field artillery FA</td>
</tr>
<tr>
<td>⚲</td>
<td>Infantry IN</td>
</tr>
<tr>
<td>⚲</td>
<td>Mechanised infantry M</td>
</tr>
<tr>
<td>⚲</td>
<td>Mountaineer MTN</td>
</tr>
<tr>
<td>⚲</td>
<td>Naval N</td>
</tr>
<tr>
<td>⚲</td>
<td>Reconnaissance REC</td>
</tr>
<tr>
<td>⚲</td>
<td>Special Operations Forces SOF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Echelon &amp; Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚲</td>
<td>Army group AG</td>
</tr>
<tr>
<td>⚲</td>
<td>Army A</td>
</tr>
<tr>
<td>⚲</td>
<td>Corps -</td>
</tr>
<tr>
<td>⚲</td>
<td>Division D³</td>
</tr>
<tr>
<td>⚲</td>
<td>Brigade BD</td>
</tr>
<tr>
<td>⚲</td>
<td>Regiment REGT</td>
</tr>
<tr>
<td>⚲</td>
<td>Battalion BN</td>
</tr>
<tr>
<td>⚲</td>
<td>Company COY</td>
</tr>
<tr>
<td>⚲</td>
<td>Platoon PLT</td>
</tr>
<tr>
<td>⚲</td>
<td>Section</td>
</tr>
<tr>
<td>⚲</td>
<td>Squad</td>
</tr>
</tbody>
</table>

Table 12: Some abbreviations of unit type identifications
5.2 The wargame.util Ti\textit{kZ} library

This library contains some utilities for use in the other libraries.

5.2.1 Miscellaneous macros

\texttt{\wargamelogo}

This will produce the logo for this package.

\begin{verbatim}
\tikzset{
  wargame logo text/.style={
    font=\sffamily\bfseries\fontsize{12}{14}\selectfont,
    scale=2.8,
    inner sep=0,
    text width=1.8cm,
    transform shape,
    align=center},
  wargame logo text content/.store in=\wg@logo@text@content,
  wargame logo text content={{\huge\LaTeX} wargame},
  wargame logo chit/.style={
    chit={[symbol={[
      faction=friendly,
      command=land,
      echelon=division,
      main=infantry]},
    factors={chit/2 factors={4,3}},
    left={chit/identifier=III},
    right={chit/small identifier={10\textsuperscript{th}}},
    color=white,
    fill=red!50!black
    ]},
  wargame logo/.style={
    transform shape,
    every hex/.style={fill=gray!5!white,draw=gray!75!black},
    hex/first row is=0,
    hex/first column is=0,
} \end{verbatim}
hex/short top columns=none,
hex/short bottom columns=none,
hex/row direction is=normal,
hex/column direction is=normal
}
\newcommand\wargamelogo[1]{%
\begin{scope}[wargame logo,#1]
    \node[hex={fill=gray!30!white}] (logo center) at (hex cs:c=0,r=0) {};
    \node[hex={terrain=light woods}] (logo light woods) at (hex cs:c=0,r=1) {};
    \node[hex={terrain=city}] (logo city) at (hex cs:c=0,r=-1){};
    \node[hex={terrain=woods}] (logo woods) at (hex cs:c=-1,r=0){};
    \node[hex={terrain=mountains}] (logo mountains) at (hex cs:c=-1,r=1){};
    \node[hex={terrain=beach}] (logo beach) at (hex cs:c=1,r=1) {};
    \node[hex={terrain=swamp}] (logo swamp) at (hex cs:c=1,r=0) {};
    \node[wargame logo chit] (logo chit) at (hex cs:) {};
    \node[wargame logo text] (logo text) {\wg@logo@text@content};
\end{scope}}
\newcount\wargamedbglvl\wargamedbglvl=0
\def\wg@dbg#1#2{\ifnum#1>\wargamedbglvl\relax\else\message{^^J#2}\fi}
\def\wg@addto@macro#1#2{%\toks\expandafter\expandafter\expandafter{\expandafter#1#2}\xdef#1{\the\toks@}}
\def\wg@sub@nchor{\pgfpointanchor{\wargame}{sub node}{position}}

Debugging support. The counter \wargamedbglvl sets the debug level. The package code then uses \wg@dbg to print out debugging messages. This macro takes two arguments — the first is the least debug level at which the message is printed, and the second is the message itself.

\begin{itemize}
\item \wargamedbglvl\wargamedbglvl=0
\item \ifnum\wargamedbglvl>\wargamedbglvl\relax\else\message{\wargamedbglvl}\fi
\end{itemize}

The macro \wg@addto@macro{(macro)}{(other)} adds the definition of the macro (other) to the macro (macro). This uses the \toks trick of storing the tokens of the definition of a (macro) and (other) into @ and expanding that token into the definition of (macro). Effectively, this means that the top-level definition of (macro) and (other) are expanded (i.e., macros used in the definition of either macro is not expanded) and then that becomes the new definition of (macro).

We will use this macro to do shallow definitions of macros to contain keys and such.

Get anchor from sub node. We cannot use \pgfpointanchor since that returns the anchor coordinates in the global coordinate system.
5.2.2 Pictures in compound nodes

\texttt{\textbackslash wg@pic}

The macro \texttt{\textbackslash wg@pic} will render a \texttt{pic}. This is used by the \texttt{natoapp6cs}, \texttt{chit}, and \texttt{hex} node shapes extensively. The arguments are

1. Prefix
2. Position
3. Fixed options
4. User options
5. Picture.

That is, the macro expects calls like

\texttt{\textbackslash wg@pic\{\langle options\rangle\}\langle picture\rangle\textbackslash @end\textbackslash wg@pic\{\langle prefix\rangle\}\{\langle position\rangle\}\{\langle options\rangle\}}

Note the \texttt{\textbackslash @end\textbackslash wg@pic} at the end of the call to swallow up \texttt{\langle picture\rangle}. Typically this macro is used as

\texttt{\textbackslash def\textbackslash args\{\langle something\rangle\}\textbackslash expandafter\textbackslash wg@pic\textbackslash args\textbackslash @end\textbackslash wg@pic\{\langle prefix\rangle\}\{\langle position\rangle\}\{\langle options\rangle\}}

where \texttt{\langle something\rangle} typically expands to \texttt{\{\langle user option\rangle\}}\texttt{\langle picture\rangle}

First, the top-level macro \texttt{\textbackslash wg@pic} that looks for user options.

\texttt{\textbackslash def\textbackslash wg@pic\{\%}
\texttt{\textbackslash if\textbackslash next\textbackslash char\{\langle \textbackslash wg@pic\rangle\}\{\\textbackslash wg@pic\textbackslash[]\}\%}
\texttt{\textbackslash }
This macro then forwards to \wg@@pic to gobble up \textit{picture}.

1. User options

2. Arguments

\begin{verbatim}
def\wg@@pic[#1]#2@endwg@pic{\%  
  \wg@dbg{2}{Options: '#1', picture: '#2'}\%  
  \wg@@@pic{#1}{#2}\%}
\end{verbatim}

1. User options

2. Arguments

3. Prefix

4. Coordinates

5. Fixed options

\begin{verbatim}
def\wg@@@pic#1#2#3#4#5{\%  
  \ifx|#2|\wg@dbg{3}{No picture given}\%  
  \else\%  
    ^^J User options:  #1  
    ^^J Picture:      #2  
    ^^J Prefix:       #3  
    ^^J Coordinates:  #4  
    ^^J Fixed options: #5}\%  
  \pic[#5,#1] at (#4) {#3#2};\%  
  \if\wg@s@ve\%  
    \pgf@relevantforpicturesizetrue\%  
    \begin{getbbl}\%  
      \pic[draw=none,fill=none,transform shape] at (#4) {#3#2};\%  
    \end{getbbl}\%  
    \wg@dbg{5}{Clipping to local bounding box}\%  
    \clip (L.south west) rectangle (L.north east);\%  
  \fi\%  
  \fi\%}
\end{verbatim}

\texttt{\wg@pic@all}

This macro sets all pictures in a list.

1. List

2. Prefix
3. Position

4. Styles

\def \wg@pic@all#1#2#3#4{%
  \wg@dbg{2}{WG picture loop
  ^^J List: \meaning#1
  ^^J Prefix: ‘#2’
  ^^J Position: ‘#3’
  ^^J Styles: ‘#4’}
\foreach \p in #1{% 
\wg@dbg{2}{WG picture element: \meaning\p}%
\expandafter \wg@pic\p \@endwg@pic {#2}{#3}{#4}%
}%
}

5.2.3 Nodes in compound nodes

\wg@node

The macro \wg@node will render a node. This can be used by the natoapp6cs, chit, and hex node shapes. The arguments are

1. Prefix
2. Position
3. Fixed options
4. User options
5. Body.

That is, the macro expects calls like

\wg@node\[⟨options⟩]\langlebody\rangle\@endwg@node\{⟨prefix⟩\}{⟨position⟩\}{⟨options⟩}

Note the \@endwg@node at the end of the call to swallow up ⟨body⟩. Typically this macro is used as

\edef\args{⟨something⟩} \expandafter\wg@node\args\@endwg@node\{⟨prefix⟩\}{⟨position⟩\}{⟨options⟩}

where ⟨something⟩ typically expands to ⟨⟨user option⟩⟩⟨body⟩

First, the top-level macro \wg@node that looks for user options.

\def \wg@node{% 
  @ifnextchar[\{\wg@node\{\wg@node[]}% }
}

This macro then forwards to \wg@node to gobble up ⟨body⟩.

1. User options
2. Arguments
This macro sets all pictures in a list.

1. List
2. Prefix
3. Position
4. Styles
5.2.4 Bounding boxes

Bounding box dimensions

Enable or disable bounding box tracking

Reset the bounding box tracking dimensions

Save PGF’s bounding box algorithm

Our bounding box algorithm

Environment that tracks the local bounding box

getbbl (env.) Environment that tracks the local bounding box
5.2.5 Some utilities to get bounding boxes and the like

All coordinates, and such are recorded in centimetres. It is worth remembering that the Tikz coordinate system has the y-axis point upward, while typical image software has the y-axis point down. pdftocairo typically assumes a 150 PPI (pixels-per-inch) resolution.

That means that scaling factor becomes

\[
\frac{150 \text{ pixel}}{2.54 \text{ cm}} = 59.055 \frac{\text{pixel}}{\text{cm}}
\]

Since we want to write all dimensions in centimetres, we need to be able to convert pt dimensions to centimetres. We make two macros to do that for us.

The exact definition of 1pt is

\[
1 \text{ pt} = \frac{249}{250} 12'' \frac{1}{864} = \frac{83}{6000} 1'' = 0.035136
\]
The next macro gets an anchor’s coordinates and stores them (in units of centimetres) in \tmp@x and \tmp@y.

\def\wg@get@nchor#1#2{\pgfpointanchor{#1}{#2}\pgfgetlastxy\tmp@x\tmp@y\pgfpointtransformed{\pgfpoint{\tmp@x}{\tmp@y}}\pgf@xa=\pgf@x\pgf@ya=\pgf@y%% \message{^^JAnchor #1.#2 @ (\the\pgf@xa,\the\pgf@ya)}\wg@pt@to@cm{\the\pgf@xa}\edef\tmp@x{\pgfmathresult}\wg@pt@to@cm{\the\pgf@ya}\edef\tmp@y{\pgfmathresult}}

This does the same as above, but transforms to the global coordinate system.

\def\wg@get@global@nchor#1#2{\pgfpointanchor{#1}{#2}\pgfgetlastxy\tmp@x\tmp@y\pgfpointtransformed{\pgfpoint{\tmp@x}{\tmp@y}}\pgf@xa=\pgf@x\pgf@ya=\pgf@y%% \message{^^JAnchor #1.#2 @ (\the\pgf@xa,\the\pgf@ya)}\wg@pt@to@cm{\the\pgf@xa}\edef\tmp@x{\pgfmathresult}\wg@pt@to@cm{\the\pgf@ya}\edef\tmp@y{\pgfmathresult}}

This records the bounding box given by a named node. The result is stored in the macros \llx, \lly, \urx, and \ury.

\def\wg@get@bb#1{\wg@get@nchor{#1}{south west}\edef\llx{\tmp@x}\edef\lly{\tmp@y}\wg@get@nchor{#1}{north east}\edef\urx{\tmp@x}\edef\ury{\tmp@y}}
\def\wglogbb#1{\wg@get@bb{#1}%% \message{^^J#1' BB: (\llx,\lly) x (\urx,\ury)^^J}}

### 5.2.6 Other Tikz utilities

<table>
<thead>
<tr>
<th><strong>tikz/reverseclip</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A reverse clipping path. This is used to cut out stuff outside of path defined.</td>
</tr>
</tbody>
</table>

\tikzstyle{reverseclip}=[insert path={(current bounding box.north east) -- (current bounding box.south east) -- (current bounding box.south west) -- (current bounding box.north west) -- (current bounding box.north east)}]
\textbf{tikz/clip even odd rule}

A reverse clipping path

\begin{verbatim}
250 \tikzset{
251   clip even odd rule/.code={\pgfseteorule}, % Credit to Andrew Stacey
252 }
\end{verbatim}

\textbf{tikz/invclip}

Inverse clipping. This should be an option after the path to do the inverse clipping by. This works by adding a \textit{large} (page) path to the current path, and then use that as clipping.

\begin{verbatim}
253 \tikzset{
254   invclip/.style={
255     clip,insert path={
256       [clip even odd rule]{
257         [reset cm](-\maxdimen,-\maxdimen)rectangle(\maxdimen,\maxdimen)
258       }
259     },
260   }
\end{verbatim}

\textbf{save clip}

An option for use with sub-elements of NATO App 6(c) or chit nodes. This will save the current path as a clipping path for the next paths to be drawn in the sub-element

\begin{verbatim}
261 \newif\ifwg@s@ve\wg@s@vefalse
262 \tikzset{
263   save clip/.is choice,  
264   save clip/true/.code={\global\wg@s@vetrue},
265   save clip/false/.code={\global\wg@s@vefalse},
266   save clip/.default={true},
267   save clip/.initial={false},
268 }
\end{verbatim}

\textbf{scale line widths}

Scales any line width specified in the node options.

Use like

\begin{verbatim}
\tikzset{
  some/.style={
    scale line widths,
    line width=1pt}
}
\end{verbatim}
Note that the order is important.

\begin{verbatim}
\% Save pgf rounded corners macro
\let\wg@pgfsetcornersarced\pgfsetcornersarced
\def\wg@setcornersarched#1{%
\def\arg{#1}%
\let\isarched\relax% Cannot set \ifpgf@arccorners directly inside
\ifx\arg\@empty\else%
\edef\pgf@corner@arc{{#1}{#1}}%
\let\isarched\pgf@arccornerstrue%
\ifdim#1=0pt%
\let\isarched\pgf@arccornersfalse%
\fi%
\fi%
\isarched}
\newdimen\wg@lw@scaled\wg@lw@scaled=1pt
\def\wg@getscale{%
\pgfgettransformentries{\wg@jaca}{\wg@jacb}{\wg@jacc}{\wg@jacd}{\wg@tmp}{\wg@tmp}%
\pgfmathsetmacro{\wg@jac}{sqrt(abs(\wg@jaca*\wg@jacd-\wg@jacb*\wg@jacc))}%
\wg@dbg{4}{Scale is \wg@jac}%
\xdef{\wg@scale}{\wg@jac}%
\def\wg@scaled#1{%
\wg@getscale%
\wg@dbg{4}{Scaling #1 by \wg@scale}%
\pgfmathsetmacro{\wg@tmp}{\wg@scale*#1}%
\xdef{\wg@tmp}{\wg@tmp pt}%
\xdef{\wg@lw@scale}{\wg@tmp}%
\wg@dbg{4}{Scaled #1 -> \wg@tmp}}%

\def\wg@scaled#1{%
\wg@getscale%
\pgfmathsetmacro{\wg@jac}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
\wg@dbg{4}{Scaling #1 by \wg@scale}%
\pgfmathsetmacro{\wg@scal}{\wg@scale*#1}%
\xdef{\wg@tmp}{\wg@scal}%
\xdef{\wg@lw@scale}{\wg@tmp}%
\wg@dbg{4}{Scaled #1 -> \wg@tmp}%
\xdef{\the\wg@lw@scaled}{\wg@lw@scale}%
\xdef{\the\wg@tmp}{\wg@tmp pt}%
\pgfmathsetlength{\pgflinewidth}{\wg@tmp pt}%
\wg@dbg{4}{Did set line width \wg@tmp pt}%
}
\tikzset{%
\% Get current scale and store in \wg@scale
get scale/.code={\wg@getscale},%

scale line widths/.style={%
/utils/exec=\def{\tikz@semiaddlinewidth\pgfsetlinewidth{\wg@lw@scale}}%
\xdef{\the\tikz@semiaddlinewidth}{\the\tikz@semiaddlinewidth}%
}%
\def\wg@process{%
\xdef{\the\wg@process}{\the\wg@process}%
\xdef{\the\pgf@process}{\the\pgf@process}%
\xdef{\the\pgf@parse}{\the\pgf@parse}%
\}
\}
\}
\%
\end{verbatim}

sub pic actions

This is key that propagates actions to sub pictures of pictures. The normal pic actions cannot be used as it causes an infinite loop.

wg/debug show

Show debugging information
5.2.7 Random IDs

This macro sets the macro $\wg@uuid$ to some random hex number.

\begin{verbatim}
\def\wg@randomid{\def\wg@uuid{}
\foreach \i in {1,...,8}{\pgfmathparse{Hex(random(0,15))}\xdef\wg@uuid{\wg@uuid\pgfmathresult}}}
\end{verbatim}

5.2.8 VASSAL icons

Some icons that may be useful in VASSAL. We put them here so they may be used in manuals and the like too.

First, the line style

\begin{verbatim}
\tikzset{
\trash can line/.style={scale line widths, scale rounded corners, line width=.5mm, ->},
}
\end{verbatim}

Then, the body and lid of a trash can.

\begin{verbatim}
\tikzset{
\trash can body/.pic=\{
\path[fill=black, scale line widths, scale rounded corners, rounded corners=.05cm]
(-.3,.2) --++(.6,0) --++(-.1,-.7) --++(-.4,0) --cycle;
\path[fill=white]
(-.025,-.4) arc(180:360:.025) --++( 0,.5) arc(0:180:.025) --cycle;
\path[fill=white]
(-.125,-.4) arc(180:360:.025) --++(-.07,.5) arc(0:180:.025) --cycle;
\path[fill=white]
(.075,-.4) arc(180:360:.025) --++(.07,.5) arc(0:180:.025) --cycle;
},
\trash can lid/.pic=\{
\path[fill=black, scale line widths, scale rounded corners, rounded corners=.05cm]
(-.35,.23) --++(.7,0) --++(-.07,.07) --++(-.56,0) --cycle;
\path[fill=black]
(-.15,.3) --++(.05,0) --++(0,.05) --++(.2,0) --++(0,-.05)
--++(.05,0) --++(0,.05) arc(0:90:.05) --++(-.2,0) arc(90:180:.05)
--cycle;
},
\end{verbatim}

Then, a closed and open trash can

\begin{verbatim}
\tikzset{
\trash can/.pic=\{
\pic[\trash can body];
\pic[\trash can lid];
},
\trash can open/.pic=\{
\end{verbatim}
Now we can use that to generate some useful icons.

\begin{tikzpicture}[scale=0.5]
\pic{trash can body};
\pic[rotate=-30] at (0,.1) {trash can lid};
\end{tikzpicture}

These icons does not use the trash can picture.
5.3 The \texttt{wgexport} class

This document class is used for exporting game component to be used in a VASSAL module libraries.

Class identification and load \texttt{wargame} package

\begin{verbatim}
\ProvidesClass{wgexport}
\PassOptionsToClass{multi=tikzpicture, varwidth=false}{standalone}
\DeclareOption{noterrainpic}{\PassOptionsToPackage{\CurrentOption}{wargame}}
\DeclareOption{terrainpic}{\PassOptionsToPackage{\CurrentOption}{wargame}}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{standalone}}
\ProcessOptions\relax
\LoadClass{standalone}
\RequirePackage{wargame}
\end{verbatim}

We need a few utilities before we get to the actual environment. First, we need a tools to write out literal left and right curly braces. We do a bit of catcode hackery to accomplish that.

\begin{verbatim}
\begingroup
\catcode'\^^I=12
\def\@tabchar{\^^I}
\catcode'<=1 \catcode'>=2
\catcode'{=12 \catcode'}=12
\gdef\@lbchar<{>
\gdef\@rbchar<}>
\endgroup
\end{verbatim}

Above, we temporarily set the tab, and left and right curly brace characters to be regular letters (12), and the catcodes of less than and greater than to be those of left and right curly braces respectively. We then define the macros \texttt{\@tabchar}, \texttt{\@lbchar}, and \texttt{\@rbchar} to produce literal characters. \LaTeX already has \texttt{\@percentchar}.

Everything we do should go inside this environment. The single optional argument is the file name stem of the output JSON file.

\begin{verbatim}
\newenvironment{imagelist}[1]{\jobname\%}{
\newwrite\mk@out\% \def\mk@i{}\% \def\mk@w{\immediate\write\mk@out}\% \immediate\openout\mk@out=#1.json \immediate\openout\mk@out=#1.json \mk@w{[} \mk@w{\mk@i \@lbchar "name":"End of list", "category": "\texttt{\@percentchar}\texttt{\@percentchar}"\@rbchar} \mk@w{]} \immediate\closeout\mk@out}
\end{verbatim}

Preceed all images (tikzpicture) with this command

First argument is the name of the image. This can be anything. Note that for counters, if the name ends in \texttt{flipped} then it is considered the backside of a counter.

Second argument is the type of image. Recognised types are
- board for boards
- oob for OOBs
- chart for charts
- counter for counters
- front for front page

Other types can be used, and the images will be exported, but the Python script pays no particular attention to those then. Use for example to prepare images for help or the like.

The third argument is the sub type. This is most relevant for the counters. Sub types can be anything, but since the counters will receive different prototypes based on the sub type, it makes sense to divide into sub types a la

- factions
- common markers

The faction sub types should just be the name of the faction. E.g., Allies, Axis, Soviet, NATO, Warsaw Pact. Spaces should not matter.

For common markers, there are a few names that are recognised specifically by the Python script. These are

- common
- all
- marker
- markers

Counters that has these sub-types will no be considered to belong to any faction.

Note that the Python script uses the faction names to guess the players of the game, and uses them in several places.

Make separate images for each counter (single sided).
First optional argument is the group to put the chits into. Second optional argument is options to give to each Tikz picture environment. Third, mandatory, argument is the list of chit identifiers to render.

487 \def \wg@add@drop@margin{\%}
488 \@ifundefined{wg@drop@margin}{\%}
489 \dimen0 = \wg@drop@margin
490 \ifdim \dimen0 > 0pt \% path \$(\text{current bounding box.north east})+(45:\wg@drop@margin)$
491 \path ($(\text{current bounding box.south west})+(225:\wg@drop@margin)$);
492 \fi}
493 \def \chitimages{\%}
494 \ifnextchar[\chitimages\[\]}{\chitimages\[\]}\%}
495 \def \@chitimages[#1]{\%}
496 \def \@@chitimages[#1][#2]#3{\%}
497 \begingroup\%}
498 \let \chit@report \do@chit@report\%
499 \let \natoapp@report \do@natoapp@report\%
500 \chit@dbg{2}{chits to make images of \#3}\%
501 \foreach[count=\ti from 0]\%}
502 \chit\[\u=\ti\]\%}
503 \wg@add@drop@margin\%
504 \endgroup\%

Make separate images for each counter (double sided). The back-side counters must be defined by append `flipped` the front face name.

First optional argument is the group to put the chits into. Second optional argument is options to give to each Tikz picture environment. Third, mandatory, argument is the list of chit identifiers to render.
Special for boards, we have the environment `boardimage`. Like `\info` we must specify the name and sub-category of the board, but the category is assumed to be `board` (though the optional argument can specify a different category).

Within this environment some specific styles are defined that allows the user to specify VASSAL zones on the board. For this to work properly, the parent `tikzpicture` must have the style `zoned`. This style will record the bounding box of the picture which we will need to calculate VASSAL coordinates later on.

Other styles are `zone scope`, to be applied to `scopes` in the picture, and `zone path` to be applied to `paths` (or `\draw`, `\fill`, or the like) in the picture. These will record coordinates of these elements in side the picture. The Python script will then define VASSAL zones based on these coordinates.

For `zone scope` applied to a `scope`, what is recorded are

- The current coordinate transformation matrix
- The current translation
- The bounding box, within the current transformation and translation.

To define a zone in the board, simply enclose it in a

```
\begin{scope}[zone scope=name]
  ...
\end{scope}
```

The `(name)` will be the name of the scope. If this contains the sub-string `hex` (upper, lower, or mixed case), then the zone will get a hex grid with numbering attached to it.

If the `(name)` contains the sub-string `turn` (any case), then it is assumed to be a turn track and a rectangular grid will be attached. The column and row separator will be set to T, so that it won’t collide with the main zone. Similar if `(name)` contains `oob`, except the separator is set to O.
If ⟨name⟩ contains the sub-string pool, then it is assumed to be a pool of counters, and no grid is attached.

For zone path applied to a path, what is recorded is the path coordinates (as straight line segments) in the global coordinate system.

Both styles take one argument — the name of the zone. If that name contains the sub-string hex anywhere in the name, then the zone is assumed to contain a hex grid. Otherwise, a rectangular grid (of fixed size) will be applied to it.

The environment boardimage also records the coordinate options currently in use (keys hex/first row is, hex/row direction is, and so on), as well as the current label option (as defined by every hex or every hex node).

The information extracted is written to the \jobname.json file as a sub-object (with name given by the first optional argument) of the image object. In that way, we can later on easily get the information from our catalogue of images.

Note, the styles zoned, zone scope, and zone path are defined in wargame to be dummies so that one can have them in the definition of the board without impact.

\def\mk@transform{% 
  \pgfgettransformentries{\mxx}{\myy}{\myx}{\myy}{\ptdx}{\ptdy} 
  \wg@pt@to@cm{\ptdx} \edef\dx{\pgfmathresult} 
  \wg@pt@to@cm{\ptdy} \edef\dy{\pgfmathresult} 
  \mk@w{\mk@i "xx": \mxx,} 
  \mk@w{\mk@i "xy": \myy,} 
  \mk@w{\mk@i "yx": \myx,} 
  \mk@w{\mk@i "yy": \myy,} 
  \mk@w{\mk@i "dx": \dx,} 
  \mk@w{\mk@i "dy": \dy,} 
}

\def\mk@bb#1{% 
  \wg@get@bb{#1} 
  \mk@w{\mk@i "lower left": [\llx,\lly],} 
  \mk@w{\mk@i "upper right": [\urx,\ury],} 
  \begingroup 
  \wg@get@global@nchor{#1}{south west} 
  \mk@w{\mk@i "global lower left": [\tmp@x,\tmp@y],} 
  \wg@get@global@nchor{#1}{north east} 
  \mk@w{\mk@i "global upper right": [\tmp@x,\tmp@y]} 
  \endgroup 
}

\def\mk@pos#1(#2){% 
  \wg@dbg{10}{\"Marking #2 with #1 - start\} 
  \coordinate[transform shape] (tmp) at (#2) {}; 
  \wg@get@nchor{tmp}{center} 
  \wg@dbg{3}{\"Marking #2 with #1 - \tmp@x,\tmp@y\} 
  \tikzset{zone point={#1}{\tmp@x}{\tmp@y}} 
}

For the key zone path to work, we need to be able to record the path as it moves along. To that end, we make a custom decoration that will do that for us, and, once the path is finished, write the path to our JSON file.

\pgfdeclaredecoration{record path construction}{initial}{% 
  \state{initial}[width=Opt,next state=more]{% 
    \begingroup 
    \pgfdecorateinputsegment{first} 
    \ptpoint@to@cm{\the\pgf@x}{\the\pgf@y} 
  }

50
Now we can make our environment.

The first thing we do is to use the \texttt{\textbackslash info} macro to mark the image. Then we open our JSON file. We make a short-hand macro for writing to that file. The macro \texttt{\textbackslash bd@i} records the current indentation (which is important in JSON).

\begin{Verbatim}
\newenvironment{boardimage}[3][board]{\% \def\bd@n{#2} \newcount\mk@point \mk@point=0 \let\oomk@i\mk@i \let\markpos\mk@pos \\textbackslash info{dummy}{<<dummy>>}{}% \mk@w{ \mk@i "zones": \@lbchar} \edef\mk@i{\mk@i\space "path": [\texttt{\textbackslash wg@path}] \@rbchar,} \%\textbackslash info*{#2}{#1}{#3} \mk@w{ \mk@i "units": "cm",} \% \textbackslash info*{#2}{#1}{#3}\% \mk@w{ \mk@i "zones": \@lbchar}\% \edef\mk@i{\mk@i\space \texttt{\textbackslash wg@path}\texttt{.} \@rbchar,} \% The next thing we do is to make an object. The first things we put in are the units used (“cm”), and the grid options.
\end{Verbatim}

Then, to extract the label option, we make a dummy node with the styles every hex and every hex node, so we can extract that option.

\begin{Verbatim}
\% every hex/.try,every hex node/.try, \}% \def\hex@col{0} \def\hex@row{0} \node[hex,inner sep=0,outer sep=0]{% \%message("\textbackslash Hex label: '\\meaning\hex@label'\textbackslash )\% \global\let\mk@label\hex@label})% \end{Verbatim}
We then monkey-patch \\texttt{\textbackslash boardframe} to also output coordinates to our JSON file. Note that this will probably be embedded in a different object.

\begin{verbatim}
\let\oldbo@rdframe\bo@rdframe\%
\def\bo@rdframe[#1](##2)(##3){\%
\oldbo@rdframe[#1](##2)(##3)\%
\mk@w{ \mk@i "board frame": \@lbchar}\
\mk@w{ \mk@i "lower left": [\llx,\lly],}\
\mk@w{ \mk@i "upper right": [\urx,\ury],}\
\mk@w{ \mk@i "margin": \margin,}\
\mk@w{ \mk@i "width": \w,}\
\mk@w{ \mk@i "height": \h \@rbchar,}}%
\end{verbatim}

Next, we make the style \texttt{zoned} to be applied to the \texttt{tikzpicture} environment. This records the bounding box of the full picture.

\begin{verbatim}
\tikzset{
  zoned/.code={% Apply to whole picture
    \pgfkeys{%
      % This needs to be done in the picture!
    /tikz/execute at end picture={%}
    \mk@w{ \mk@i "zoned": \@lbchar}\
    \mk@transform%
  }
\}
\end{verbatim}

The next style is the \texttt{zone scope}. At the start of the scope we record the current transformation matrix. Then we install a handler to extract the bounding box at the end of the scope. Note that we increase indentation here.

\begin{verbatim}
zone scope/.code=({%
\mk@w{ \mk@i "zone scope ##1": \@lbchar}\
\let\omk@i\mk@i\
\edef\mk@i{\mk@i\space}\
\mk@transform%
\mk@bb{current bounding box}\
\mk@w{ \mk@i \@rbchar,}
}
\end{verbatim}
The next style gets the global coordinates of the current (0,0) point - f.ex. in a node - and outputs that

```
zone point/.code n args={3}{
\pgf@xa=#2 cm
\pgf@ya=#3 cm
\pgfpointintransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}%
\pgfpointintransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}%
\pgf@xa=\pgf@x
\pgf@ya=\pgf@y
\pgf@pt@to@cm{\the\pgf@xa}\edef\px{\pgfmathresult}
\pgf@pt@to@cm{\the\pgf@ya}\edef\py{\pgfmathresult}
\advance\mk@point1
\global\mk@point=\mk@point
\mk@w{ \mk@i "point\the\mk@point": \@lbchar "name": "##1", "type": "point", "coords": [\px,\py]
% parent": "\wg@export@box", "type": "point", "coords": [\px,\py]
% \@rbchar, }
}
```

The `zone point` style is a bit more simple, but only because the bulk of the work is done in a decoration. We need to be able to pass a name to that decoration, so we make a key for that. The user need not think about that though.
That finishes the first part of the environment. At the end of the environment, we simply write the name of the picture, and close our JSON output.

Make battle markers. Mandatory argument is how many markers, optional is the group to add the markers to.

\def\wg@gennumberm@rkers#1#2#3#4{
  \message{\text{Numbered markers: Type='\texttt{#1}' Max='\texttt{#2}' Category='\texttt{#3}'}
  \def\markers{}\def\keys{}
  \foreach \i in {1,...,#2}{\xdef\keys{/tikz/#1 \i/.style={/tikz/#1=\i},\keys}\xdef\markers{\markers,#1 \i}}
  \nopagecolor\pgfkeysalsofrom{\keys}\chitimages{\#3}{\#4}{\markers}
}
\tikzset{
  wg hidden unit/.pic={},
  wg hidden unit/.style={
    chit={
      no chit drop, 
      frame={draw=none,fill=none}, 
      full=wg hidden unit}}}
\def\battlemarkers{\@ifnextchar[{{@battlemarkers}{\battlemarkers[\]}}}%
\def{@battlemarkers[#1]{\@ifnextchar[{{@@battlemarkers[#1]}{\battlemarkers[#1][BattleMarkers]}}%}
\def@@battlemarkers[#1][#2]#3{\wg@gennumberm@rkers{battle marker}{#3}{#2}{#1}\
  \message{\text{Make a hidden unit and add to Markers category}}
  \nopagecolor\pgfkeysalsofrom{\keys}\chitimages{Markers}{{wg hidden unit}}
Make odds markers. Mandatory argument is a list of odds and fill colours. Optional is the group to add the markers to.

\def\wg@gencolorm@rkers#1#2#3#4{\def\markers{}\def\keys{}
\foreach \o/\f/\n [count=\i] in {#2}{\ifx\n\f\def\n{\o}\fi\ifx\o\f\def\f{white}\fi\message{^^JColour no \i marker '#1 \n' w/fill '\f' text '\o'}}\protected@xdef\keys{/tikz/#1 \n/.style={/tikz/#1={\o,\f}},\keys}\xdef\markers{\markers,#1 \n}\nopagecolor\pgfkeysalsofrom{\keys}\chitimages[#3][#4]{\markers}\}
\def\oddsmarkers{\@ifnextchar[{{\@oddsmarkers}}{\oddsmarkers[]}}\def\@oddsmarkers[#1]{{\@ifnextchar[{{\@@oddsmarkers[#1]}}{\oddsmarkers[#1][OddsMarkers]}}}\def\@@oddsmarkers[#1][#2]#3{\wg@gencolorm@rkers{odds marker}{#3}{#2}{#1}\info{odds-battles-icon}{icon}{}\tikz[\scale=.5,\transform shape,\auto icon more/.try]{\pic{odds marker={?:?,white}}}\info{resolve-battles-icon}{icon}{}\tikz[\scale=.3,\transform shape,\auto icon more/.try]{\pic{dice}}\tikz[\scale=1.2,\transform shape] at (-.2,-.2) {\pic{battle marker=0};}}

Make results markers. Mandatory argument is a list of results and fill colours. Optional is the group to add the markers to.
First optional argument are extra styles Second is category Third is marker list

\def\resultmarkers{%  
\@ifnextchar[{{\resultmarkers}{\resultmarkers[]}}%  
}\def\@resultmarkers[#1]{%  
\@ifnextchar[{{\@@resultmarkers[#1]}\resultmarkers[#1][ResultMarkers]}%  
}\def\@@resultmarkers[#1][#2]#3{%  
\w@g@gencolorm@rkers{result marker}{#3}{#2}{#1}}%  
\DeclareRobustCommand\commonicons[3][{}]{%  
\begingroup%  
\nopagecolor%  \tikzset{auto icon/.style={scale=.4,transform shape,#1}}%  \info{pool-icon}{icon}{}%  \tikz[auto icon,auto icon more/.try]{\pic{pool icon};}%  \info{oob-icon}{icon}{}%  \tikz[auto icon,auto icon more/.try]{\pic{oob icon={#2}{#3}};}%  \info{flip-icon}{icon}{}%  \tikz[auto icon,auto icon more/.try]{\pic{flip icon};}%  \info{eliminate-icon}{icon}{}%  \tikz[auto icon,auto icon more/.try]{\pic{eliminate icon};}%  \info{restore-icon}{icon}{}%  \tikz[auto icon,auto icon more/.try]{\pic{restore icon};}%  \info{dice-icon}{icon}{}%  \tikz[auto icon,scale=.9,auto icon more/.try]{\pic{dice};}%  \info{unit-icon}{icon}{}%  \tikz[auto icon,scale=.7,auto icon more/.try]{%  \chit[fill=#2,  
symbol={[  
scale line widths,  
line width=1pt,  
faction=friend,  
command=land,  
main=infantry,  
scale=1.3](0,.15)]}%  \info{layer-icon}{icon}{}%  \begin{tikzpicture}[scale=.25]  \foreach \i in {-1,0,1}{  \scoped[shift={(0,\i*.15)}]{  \draw[black,fill=white] (-.5,0)  --(0,.3) --(.5,0) --(0,-.3) --cycle;  \}}}}%
5.3.1 Making dice

\begin{tikzpicture}
\draw[scale line widths, line width=2pt, fill=white] (-.5,0) to[out=70,in=110] (.5,0) to[out=-110,in=-70] cycle;
\begin{scope}[even odd rule]
\clip circle(.2);
\fill circle(.2) (125:.18) circle(.1);
\end{scope}
\end{tikzpicture}

\def\dice{\@ifnextchar[\wg@dice}{\wg@dice[\]}\[
\def\wg@dice[#1]{\@ifnextchar[\wg@@dice{#1}}\[
\def\wg@@dice#1[#2]#3#4#5{\foreach \v/\p in {#5}{\node[shape=#4, transform shape, draw=none, fill=black, opacity=.5] at (.05, -.03){};\node[shape=#4, #2, transform shape, chit drop] {\p};\wg@add@drop@margin{}}}}

5.3.2 Hooks into chits, etc.

TO BE DONE: We could add hook the hex shape that would allow us to write out the settings for each of these. This would allow us to make data files that contain the information available in the \LaTeX code.
If one then assumed that for example the upper left corner holds the start-up hex, then one could use that information. The code below exports the chit information to the JSON file. Together with the battle, odds, and result markers stuff above, this allows the exporter to almost automatically set up battle odds and result calculations. The fields exported are

- Left and right identifiers
- Upper left, upper right, lower left, and lower right identifiers. (some care must be taken if these contains graphics and not just text.)
- Factors
- NATO symbol
  - Faction, command, echelon
  - Mains
  - Left, right, top, and bottom attributes and modifiers
  - Below attribute

The exporter can set up prototypes for NATO types, echelons, etc. The exporter can also set factors as marks on the units.
Report out NATO App6 symbol settings

\def\do{natoapp\report{%
\mk\w{ \mk i "natoapp6c": \@lback}
\@chit\rep\line{id}{id}
\@chit\rep\line{faction}{natoapp@fac}
\@chit\rep\line{command}{natoapp@cmd}
\@chit\rep\line{echelon}{natoapp@ech}
\@chit\rep\line{main}{natoapp@main}
\@chit\rep\line{left}{natoapp@left}
\@chit\rep\line{right}{natoapp@right}
\@chit\rep\line{upper}{natoapp@upper}
\@chit\rep\line{lower}{natoapp@lower}
\@chit\rep\line{below}{natoapp@below}
\mk\w{ \mk i \@rbchar }
}
\tikzset{
chit drop margin/.store in=\wg@drop@margin,
chit drop shadows/.code={
\pgfkeysalso{%
/tikz/every chit node/.prefix style={chit drop={#1}},
/tikz/chit has drop=true}
},
chit drop shadows/.default=,
marker drop shadows/.default=,
\pgfkeysalso{%
/tikz/every battle marker/.prefix style={chit drop={#1}},
/tikz/every odds marker/.prefix style={chit drop={#1}},
/tikz/every result marker/.prefix style={chit drop={#1}},
/tikz/auto icon more/.prefix style={no chit drop}}
marker drop shadows/.default={
\chit has drop=false,
shadow xshift=0.04cm,
shadow yshift=-0.04cm,
shadow blur radius=0.04cm}
}}
5.4 The wargame.hex Ti\textsc{\textregistered}Z library

Used Ti\textsc{\textregistered}Z libraries

\begin{enumerate}
\item \RequirePackage{alphalph}
\item \usetikzlibrary{calc}
\item \usetikzlibrary{arrows.meta}
\item \usetikzlibrary{arrows}
\item \usetikzlibrary{shapes.geometric}
\item \usetikzlibrary{shapes.symbols}
\item \usetikzlibrary{shapes.arrows}
\item \usetikzlibrary{decorations}
\item \usetikzlibrary{decorations.pathmorphing}
\item \usetikzlibrary{decorations.pathreplacing}
\item \usetikzlibrary{decorations.markings}
\item \usetikzlibrary{wargame.util}
\end{enumerate}

\texttt{@ifempty}

This is a utility macro we will use below.

\begin{verbatim}
\def\@ifempty#1\def\temp{#1}\ifx\temp\@empty
\end{verbatim}

5.4.1 Debugging

The counter \texttt{\hexdbglvl} sets the debug level, and the macro \texttt{\hex@dbg} prints out (conditionally) debug messages.

\begin{verbatim}
\newcount\hexdbglvl\hexdbglvl=\wargamedbglvl
\def\hex@dbg#1#2{%
\ifnum#1>\hexdbglvl\relax\else\message{^^J#2}\fi
\end{verbatim}

5.4.2 Suppress terrain pictures

\begin{verbatim}
\@ifundefined{if\hex@terrain@pic}{% \newif\if\hex@terrain@pic \hex@terrain@pictrue
\end{verbatim}

5.4.3 Hex coordinate system

\begin{verbatim}
\hex@xx
\hex@yy
\end{verbatim}

Some offsets along \textit{x} and \textit{y} due to offset of every second hex column.
\[ \delta_x = \cos 60^\circ \]
\[ \delta_y = \sin 60^\circ \]

These numbers are calculated once here and then used several times in the following code.

```latex
\begin{verbatim}
\pgfmathparse{cos(60)} \xdef\hex@xx{\pgfmathresult}
\pgfmathparse{sin(60)} \xdef\hex@yy{\pgfmathresult}
\pgfmathparse{\hex@yy*cos(30)} \xdef\hex@e@xx{\pgfmathresult}
\pgfmathparse{\hex@yy*sin(30)} \xdef\hex@e@yy{\pgfmathresult}
\newdimen\hex@radius \hex@radius=1cm
\newdimen\hex@dx \expandafter\hex@dx=\hex@xx cm
\newdimen\hex@dy \expandafter\hex@dy=\hex@yy cm
\newdimen\hex@e@dx \expandafter\hex@e@dx=\hex@e@xx cm
\newdimen\hex@e@dy \expandafter\hex@e@dy=\hex@e@yy cm

Some code we need for some options
\newif\ifhex@label@is@name \hex@label@is@namefalse
\def\hex@short@col{isfalse}
\def\hex@got@short{isfalse}
\pgfmathdeclarefunction{isfalse}{1}{% 
  \begingroup
  \def\pgfmathresult{0}\
  \pgfmath@smuggleone\pgfmathresult
  \endgroup}
\pgfmathdeclarefunction{istrue}{1}{% 
  \begingroup
  \def\pgfmathresult{1}\
  \pgfmath@smuggleone\pgfmathresult
  \endgroup}

What follows is a way to configure the hex coordinate system. For example, if the rows goes down, then we can flag that, but still add hexes straightforwardly. Similar for columns. We can also specify that the first row or column has number 1 (instead of 0). Since this is dealt with a the coordinate level, it means most of the rest of the code is agnostic to these choices.

Which is the first coordinate (0 or 1)
\tikzset{
  hex/first row is/.is choice, 
  hex/first row is/0/.code=\def\hex@coords@row@off{0}, 
  hex/first row is/1/.code=\def\hex@coords@row@off{-1}, 
  hex/first row is=0, 
  hex/first column is/.is choice, 
  hex/first column is/0/.code=\def\hex@coords@col@off{0}, 
  hex/first column is/1/.code=\def\hex@coords@col@off{-1}, 
  hex/first column is=0, 
  hex/first row and column are/.is choice, 
  hex/first row and column are/0/.style={
  hex/first row is=0, 
\end{verbatim}
```
Which way does the column and row numbers go

- hex/row direction is/.is choice,
- hex/row direction is/normal/.code={$\def\hex@coords@row@fac{1}$},
- hex/row direction is/reversed/.code={$\def\hex@coords@row@fac{-1}$},
- hex/row direction is/up/.style={hex/row direction is=normal},
- hex/row direction is/down/.style={hex/row direction is=reversed},
- hex/row direction is/positive/.style={hex/row direction is=normal},
- hex/row direction is/negative/.style={hex/row direction is=reversed},
- hex/row direction is=normal,

- hex/column direction is/.is choice,
- hex/column direction is/normal/.code={$\def\hex@coords@col@fac{1}$},
- hex/column direction is/reversed/.code={$\def\hex@coords@col@fac{-1}$},
- hex/column direction is/right/.style={hex/column direction is=normal},
- hex/column direction is/left/.style={hex/column direction is=reversed},
- hex/column direction is/positive/.style={hex/column direction is=normal},
- hex/column direction is/negative/.style={hex/column direction is=reversed},
- hex/column direction is=normal,

Make labels names of shapes of the hexes so we can use labels to place stuff

- hex/label is name/.is if=hex@label@is@name,

If we have uneven number of rows in some columns.

- hex/short bottom columns/.is choice,
- hex/short bottom columns/odd/.code={$\\def\hex@bot@short@col{isodd}\\def\hex@got@bot@short{istrue}\\hex@dbg{4}{Short columns (odd): \meaning\hex@bot@short@col}$},
- hex/short bottom columns/even/.code={$\\def\hex@bot@short@col{iseven}\\def\hex@got@bot@short{istrue}\\hex@dbg{4}{Short column (even): \meaning\hex@bot@short@col}$},
- hex/short bottom columns/none/.code={$\\def\hex@bot@short@col{isfalse}\\def\hex@got@bot@short{isfalse}\\hex@dbg{4}{Short columns (none): \meaning\hex@bot@short@col}$},
- hex/short bottom columns=none,

- hex/short top columns/.is choice,
- hex/short top columns/odd/.code={$\\def\hex@top@short@col{isodd}\\\def\hex@got@top@short{istrue}\\hex@dbg{4}{Short columns (odd): \meaning\hex@top@short@col}$},
- hex/short top columns/even/.code={$\\def\hex@top@short@col{iseven}\\\def\hex@got@top@short{istrue}\\hex@dbg{4}{Short column (even): \meaning\hex@top@short@col}$},
- hex/short top columns/none/.code={$\\def\hex@top@short@col{isfalse}\\\def\hex@got@top@short{isfalse}\\hex@dbg{4}{Short columns (none): \meaning\hex@top@short@col}$},
- hex/short top columns=none,
We define the keys for hexagon coordinates. These are the row, column, possible vertex or edge. Vertexes and edges are defined as multiple-choice. offset specifies the offset from the centre in the direction of a vertex or edge. By default, the offset is one, meaning all the way to the vertex or edge.

The key inverse row specifies that the rows are given from the top down, but coordinates should be calculated as if the row was negative. This (should) allow us to design boards where rows increase downward, while still keeping the interface and remaining code somewhat reasonable and agnostic.

Similarly, the key column 1, will allow us to start the columns with 1.
This macro resets the hex coordinates to default values. That is row and column 0, no vertex or edge.

The following calculates the Cartesian coordinates from Hex coordinates

Given the hexagon column $C$ and row $R$ with hexagon radius $r$, the centre of the hexagon is at

$$
x = 2C \frac{3}{4} r \\
y = r(R - (C\%2) \sin 60^\circ)
$$

If $\langle V \rangle$ or $\langle E \rangle$ are given, then these are added to the centre point.

Note, $C$ and $R$ may be fractional numbers, which will specify a point inside a hex.

We set-up the translation to Cartesian coordinates. First thing is to reset keys in /hex/coords, and then parse out the keys given.
where
\[ c_e = f_c(c + o_c) \]
is the effective column (stored in \hex@eff@col) calculated from is the direction factor \( f_c \) (set by \texttt{hex/column direction is}) and the offset \( o_c \) (set by \texttt{hex/first column is}).

\begin{verbatim}
1186 \pgfmathparse{int(\hex@coords@col@fac*(\hex@col+\hex@coords@col@off))}\%
1187 \xdef\hex@eff@col{\pgfmathresult}\%
1188 \hex@dbg{2}{Effective column: \hex@coords@col@fac * (\hex@col +
1189 \hex@coords@col@off) \rightarrow \hex@eff@col}\%
1190 \pgfmathparse{\hex@eff@col*1.5}\%
1191 \xdef\hex@x{\pgfmathresult}\%
\end{verbatim}

And then for the \( y \) coordinate and set the dimension \pgf@y.
\[ y = 2 \left( r_e - c_e \mod 2 \right) \cos 60^\circ \]

where
\[ r_e = 2 f_r (r + o_r) - (c + o_c) \mod 2 \]
is the effective row (stored as \hex@eff@row) calculated from the the direction factor \( f_r \) (set by \texttt{hex/row direction is}) and the offset \( o_r \) (set by \texttt{hex/first row is}).

\begin{verbatim}
1192 \pgfmathparse{int(\hex@coords@row@fac*(\hex@row+\hex@coords@row@off))}\%
1193 \xdef\hex@eff@row{\pgfmathresult}\%
1194 \hex@dbg{2}{Effective row: \hex@coords@row@fac * (\hex@row +
1195 \hex@coords@row@off) \rightarrow \hex@eff@row}\%
1196 \pgfmathparse{(2*\hex@eff@row-mod(\round(\hex@col+\hex@coords@col@off),2))*\hex@yy}\%
1197 \pgfmathparse{(2*\hex@eff@row-mod(abs(\round(\hex@col+\hex@coords@col@off),2))*\hex@yy)}\%
1198 \xdef\hex@y{\pgfmathresult}\%
\end{verbatim}

If we have a vertex specification add that location to the current coordinates. If not, set the point.

\begin{verbatim}
1199 \ifx\hex@vtx\@empty\else\%
1200 \pgfmathparse{\hex@x+\hex@off*cos(\hex@vtx)}\xdef\hex@x{\pgfmathresult}\%
1201 \pgfmathparse{\hex@y+\hex@off*sin(\hex@vtx)}\xdef\hex@y{\pgfmathresult}\%
1202 \fi\%
1203 \% \ifx\hex@vtx\@empty\pgfpointxy{\hex@x}{\hex@y}\else\%
1204 \% \pgfpointadd{\pgfpointxy{\hex@x}{\hex@y}}{\pgfpointpolarxy{\hex@vtx}{1}}\fi\%
1205 \% \pgfpointscale{\hex@off}{\pgfpointpolarxy{\hex@vtx}{1}}\fi\%
\end{verbatim}

If we have an edge specification add that location to the current coordinates.

\begin{verbatim}
1206 \ifx\hex@edg\@empty\else\%
1207 \pgfmathparse{\hex@x+\hex@off*\hex@yy*cos(\hex@edg)}\%
1208 \xdef\hex@x{\pgfmathresult}\%
1209 \pgfmathparse{\hex@y+\hex@off*\hex@yy*sin(\hex@edg)}\%
1210 \xdef\hex@y{\pgfmathresult}\%
1211 \fi\%
1212 \% \ifx\hex@edg\@empty\else\%
1213 \% \pgfpointadd{\pgfpointxy{\hex@x}{\hex@y}}{\pgfpointpolarxy{\hex@edg}{\hex@yy}}\%
1214 \% \pgfpointscale{\hex@off}{\pgfpointpolarxy{\hex@edg}{\hex@yy}}\fi\%
\end{verbatim}

65
For debugging, we can print out stuff.

\begin{verbatim}
\pgfpointxy{\hex@x}{\hex@y}
\hex@dbg{2}{Hex coordinates: #1
\^J c='\hex@col'
\^J r='\hex@row'
\^J v='\hex@vtx'
\^J e='\hex@edg'
\^J o='\hex@off'
\^J x='\hex@x'
\^J y='\hex@y'}%
\global\let\hex@x\hex@x%
\global\let\hex@y\hex@y%
\global\let\hex@row\hex@row%
\global\let\hex@col\hex@col%
}
\tikzdeclarecoordinatesystem{hex}{
\hex@coords@conv{#1}}
\end{verbatim}

\subsection*{5.4.4 Hexes}

In this part, we make macros etc. for the hexes.

A hex shape. We make a node of this shape if we are to give a name to the hex added. We add a bunch of anchors to it so we may easily refer to it. This is also where we actual fill stuff into the hex, such as terrain and so on.

\begin{verbatim}
\tikzset{
/\hex/.cd,
bev/.store in=\hex@bevel, bev/.initial=,
bevel fraction/.store in=\hex@bevel@frac, bevel fraction/.initial=10,
bevel/.is choice,
bevel/none/.style   = {/\hex/bev=},
bevel/north west/.style = {/\hex/bev=1},
bevel/north east/.style = {/\hex/bev=2},
bevel/south west/.style = {/\hex/bev=3},
bevel/south east/.style = {/\hex/bev=4},
bevel/NW/.style     = {/\hex/bev=1},
bevel/NE/.style     = {/\hex/bev=2},
bevel/SW/.style     = {/\hex/bev=3},
bevel/SE/.style     = {/\hex/bev=4},
bevel/.default      = {north west},
}
\def\hex@bevel@frac{10}
\tikzset{
\hex@bevel@frac{10}
\hex@beve.highlight/.style={fill=white,opacity=.25},
\hex@bevel shadow/.style={fill=black,opacity=.25},
}
\newdimen\wg@tmpe
\newdimen\wg@tmpf
\newdimen\wg@tmpg
\def\hex@bevel@path#1{%
\scope[#1]
\wg@tmpe=\wg@tmpa\multiply\hex@bevel@frac
\end{verbatim}

\end{verbatim}
\begin{verbatim}
1258 \wgtmpf=\wgtmb\multiply\wgtmpf by \hexbevelfrac
1259 \wgtmpg=\wgtmpc\multiply\wgtmpg by \hexbevelfrac
1260 \divide\wgtmpe100
1261 \divide\wgtmpf100
1262 \divide\wgtmpg100
1263 % Start
1264 \pgfpathmoveto{\pgfqpoint{\wgtma}{\wgtmb}}%
1265 % Left
1266 \pgfpathlineto{\pgfqpoint{-\wgtma}{\wgtmb}}%
1267 % Left-down
1268 \pgfpathlineto{\pgfqpoint{\wgtmc}{\wgtmd}}%
1269 % Right down
1270 \wgtmpe=-\wgtmpe%
1271 \wgtmpf=-\wgtmpf%
1272 \pgfpathlineto{\pgfqpoint{\wgtma}{\wgtmb}}%
1273 % Up, in
1274 \advance\wgtma\wgtmpe%
1275 \advance\wgtmb\wgtmpf%
1276 \pgfpathlineto{\pgfqpoint{\wgtma}{\wgtmb}}%
1277 % Left-down, in
1278 \advance\wgtmpe-\wgtmpe%
1279 \pgfpathlineto{\pgfqpoint{\wgtma}{\wgtmb}}%
1280 % Left, down in
1281 \advance\wgtmpf-\wgtmpf
1282 \pgfpathlineto{\pgfqpoint{\wgtma}{\wgtmb}}%
1283 % Start, down in
1284 \advance\wgtma-\wgtmpe
1285 \pgfpathlineto{\pgfqpoint{\wgtma}{\wgtmb}}%
1286 % %
1287 \pgfclosepath%
1288 \pgfusepath{fill}
1289 \endscope}
1290 \hexdbg{5}{Base vertex: \hexxx,\hexyy}
1291 \hexdbg{5}{Base edges: \hexxxe,\hexyye}
1292 \pgfdeclareshape{hex/hex}{%
1293    \savedim\radius{\pgf@x=\hexradius}
1294    \savedsnowedge{\pgf@y=\hexradius}
1295    \savedanode{\pgf@y=\hexradius}
1296 \savedanchor{\east}{\pgfpoint{\hexradius}{0cm}}
1297 \savedanchor{\west}{\pgfpoint{-\hexradius}{0cm}}
1298 \savedanchor{\northeast}{\pgfpoint{\hexdx}{\hexdy}}
1299 \savedanchor{\northwest}{\pgfpoint{-\hexdx}{\hexdy}}
1300 \savedanchor{\southwest}{\pgfpoint{-\hexdx}{-\hexdy}}
1301 \savedanchor{\southeast}{\pgfpoint{\hexdx}{-\hexdy}}
1302 \savedanchor{\northedge}{\pgfpoint{0cm}{\hexdy}}
1303 \savedanchor{\southeast}{\pgfpoint{0cm}{-\hexdy}}
1304 \savedanchor{\northeastedge}{\pgfpoint{\hexdx}{\hexdy}}
1305 \savedanchor{\northwestedge}{\pgfpoint{-\hexdx}{\hexdy}}
1306 \savedanchor{\southwestedge}{\pgfpoint{-\hexdx}{-\hexdy}}
1307 \savedanchor{\southeastedge}{\pgfpoint{\hexdx}{-\hexdy}}
1308 \savedmacro\init{%
1309 \def\hexpath{%
1310 \pgfpathmoveto{\east}%
1311 \pgfclosepath%

\end{verbatim}
These are the actual user callable anchors. We make anchors for each vertex and mid points on each edge.

Next we make some short hand aliases for each of these anchors.

The next part is commented out because its not obvious we’ll use these.
The background path. This path may be drawn when the node is drawn. However, we will do most of the work in the \behindbackgroundpath which gets drawn after this path.

\behindbackgroundpath{\init{\hexpath}}

The behind background path, where we do most of the work.

\behindforegroundpath{\
\hexdbg{2}{Hex behind foreground path:
--JTerrain: 'meaning\hex@terrain'
--JRidges: 'meaning\hex@ridges'
--JTown: 'meaning\hex@town'
--JExtra clipped: 'meaning\hex@extra@clip'
--JLabel: 'meaning\hex@label'
--JExtra: 'meaning\hex@extra'
--JLast node name:'meaning\tikzlastnode'
--JHex row: 'meaning\hex@row'
--JHex col: 'meaning\hex@col'
}%
\init%

We start a scope and clip to the hex path first.

\scope
\hexpath
\pgfusepath{clip}%

Anything inside this scope is clipped to the hex path. The next step is to see if we have a specified terrain for the hex.

\ifundefined{hex@terrain}{\let\hex@terrain\empty}{}%
\if\hex@terrain\empty\else\hex@do@terrain\fi%

This concludes the processing of the terrain of the hex. Next, we must see if the user specified ridges.

\ifundefined{hex@ridges}{\let\hex@ridges\empty}{}%
\if\hex@ridges\empty\else\hex@do@ridges\fi%
This concludes the processing of the ridges of the hex. Next, we should process any extra (clipped) stuff specified. The user may pass options to each picture by preceding it with \texttt{\{options\}}.

\begin{verbatim}
\@ifundefined{hex@extra@clip}{\let\hex@extra@clip\empty}{}
\ifx\hex@extra@clip\empty\else%
\hex@dbg{5}{Extra clipped: \texttt{\meaning\hex@extra}}
\pgfpointorigin\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\wg@pic@all{\hex@extra@clip}{}{\the\wg@tmpa,\the\wg@tmpb}{}%
\fi%
\end{verbatim}

This concludes the extra stuff put in the hex. Next, we should place the label is specified. Note, we may know the hex row and column at this point, stored in \texttt{\hex@row} and \texttt{\hex@column}, respectively. We may want to name the generated node from these if the user specified that option (perhaps use \texttt{\pgfnoderename} or similar).

\begin{verbatim}
\@ifundefined{hex@label}{\let\hex@label\empty}{}
\ifx\hex@label\empty\else\hex@do@label\fi%
\@ifundefined{hex@bevel}{\let\hex@bevel\empty}{}
\ifx\hex@bevel\empty\else
\northeast \wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\west \wg@tmpc=\pgf@x\wg@tmpd=\pgf@y%
\ifcase\hex@bevel\relax
\or \wg@tmpa=-\wg@tmpa \wg@tmpc=-\wg@tmpc
\or \wg@tmpb=-\wg@tmpb \wg@tmpd=-\wg@tmpd
\or \wg@tmpa=-\wg@tmpa \wg@tmpc=-\wg@tmpc
\or \wg@tmpb=-\wg@tmpb \wg@tmpd=-\wg@tmpd
\fi
\hex@bevel@path{chit/bevel highlight}
\northeast \wg@tmpa=-\pgf@x\wg@tmpb=-\pgf@y%
\west \wg@tmpc=-\pgf@x\wg@tmpd=-\pgf@y%
\ifcase\hex@bevel\relax
\or \or \wg@tmpa=-\wg@tmpa \wg@tmpc=-\wg@tmpc
\or \wg@tmpb=-\wg@tmpb \wg@tmpd=-\wg@tmpd
\or \wg@tmpa=-\wg@tmpa \wg@tmpc=-\wg@tmpc
\or \wg@tmpb=-\wg@tmpb \wg@tmpd=-\wg@tmpd
\fi
\hex@bevel@path{chit/bevel shadow}
\fi
\endscope%
\end{verbatim}

This concludes the label processing, and stuff that should be clipped to the hex shape. If the user specified a town, we can now make that.

\begin{verbatim}
\@ifundefined{hex@town}{\let\hex@town\empty}{}
\@ifundefined{hex@c@pic}{\let\hex@c@pic\empty}{}
\ifx\hex@town\empty\else\hex@do@town\fi%
\@ifundefined{hex@extra@clip}{\let\hex@extra@clip\empty}{}
\ifx\hex@extra@clip\empty\else%
\hex@dbg{5}{Extra clipped: \texttt{\meaning\hex@extra}}
\pgfpointorigin\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\wg@pic@all{\hex@extra@clip}{}{\the\wg@tmpa,\the\wg@tmpb}{}%
\fi%
\end{verbatim}

70
We can now add extra (non-clipped) stuff. We assume that extra stuff is pictures. The user may pass options to each picture by preceding it with \texttt{\{\langle options\rangle\}}.

Next, we set up the name space for hex keys. This is the top level name space for hexes. Sub keys \texttt{terrain}, \texttt{ridges}, \texttt{town}, \texttt{extra}, \texttt{label}, and \texttt{extra clipped}, store their arguments in macros and we expand these later on. This allows us to scope some of the keys given to those specific parts.

Define keys for hexagon options. These are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>terrain</td>
<td>Terrain</td>
</tr>
<tr>
<td>label</td>
<td>Label on hex</td>
</tr>
<tr>
<td>town</td>
<td>Town in hex. Optionally with a name</td>
</tr>
<tr>
<td>ridges</td>
<td>Ridge markings on hex</td>
</tr>
<tr>
<td>extra</td>
<td>More</td>
</tr>
<tr>
<td>extra clipped</td>
<td>More clipped to hex</td>
</tr>
</tbody>
</table>

The next key is the real work horse of the show. Specifying the \texttt{hex} key to a node effectively creates a hex for us. Now, there are some things we cannot do outright in the node shape code. For example, we cannot set the name of the node created from the shape code. Therefore, the use of \texttt{\hex} is often the right choice.
The first thing is to set the default graphics options. The key `every hex` can be set to hex options to be used for all hexes. For example, if one wants to label all hexes with an auto-generated label, one can do

```latex
\tikzset{every hex/.style={label={auto=numbered}}}\)
```

This, coupled with the `hex/label is name` option allows us to set up the board with really minimal effort. We can then use the board coordinates when placing units, and other things.

Now we have set up these tools we can go on and define the user facing macro.

\begin{verbatim}
\hex
\hex@
\hex@@
\end{verbatim}

This will add a hex to the output graphics. Note, the macro need not be followed by a semi-colon (;).

First argument is optional options.

```latex
\def\hex{%
\@ifnextchar[\hex@{\hex@[]}%}
```

Second optional argument is the coordinates. These should be given in the hex coordinate system.

```latex
\def\hex@[#1]{%
\@ifnextchar({\hex@@{#1}}{\hex@@{#1}(c=0,r=0)}%}
```

Third argument is the name to be used.

```latex
\def\hex@@#1(#2){%
\@ifnextchar({\hex@@@{#1}{#2}}{\hex@@@{#1}{#2}()}%}
```

Now for the real work-horse. First thing is to reset keys and parse them out from the arguments.

```latex
% Third argument is name
\def\hex@@@#1#2(#3){%
\node[hex={#1}] (tmp) at (hex cs:#2) {};%
\hex@dbg{8}{=== Label text: '\meaning\hex@l@text'}%
\ifx|#3|\relax%
\@ifundefined{hex@l@text}{%...
\hex@dbg{72}{...}...
```

72
With the above main routine for making hexes, we turn to decorating a hex with a terrain.

We make the namespace `/hex/terrain` to hold the specific terrain keys. Keys used by terrain identifiers are

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>Terrain tile image</td>
</tr>
<tr>
<td>pic</td>
<td>Terrain TikZ picture</td>
</tr>
<tr>
<td>code</td>
<td>Arbitrary TikZ code</td>
</tr>
<tr>
<td>clip</td>
<td>TikZ path to clip terrain</td>
</tr>
</tbody>
</table>

Now, we have the keys we’ll need for selecting the terrain. These live in the namespace `/hex/terrain`, and we can select between pictures or images (external graphics files) for making the terrain. We define some short hands to easily select the common terrains.
Before we go on, we define the macro that actually generates the terrain of a hex.

\begin{figure}[h]
\centering
\begin{tikzpicture}
\end{tikzpicture}
\caption{Terrain Diagram}
\end{figure}

\hex@do@terrain

If we do have a terrain specified, we start a new scope, this time to clip the terrain by the clipping path specified by \texttt{hex=\{terrain=\{clip=\ldots\}\}}. The first thing into the new scope is to process the keys specified in \texttt{hex=\{terrain=\ldots\}}. This will set the terrain and the clipping of the terrain.

\@ifundefined{hex@t@clip}{\let\hex@t@clip\empty}{}
\ifx\hex@t@clip\empty\else\fi

We check to see if we have any clipping pictures. If so, we process these in turn and append the soft path to a macro. Once this is done, we use the soft path as a clipping path for the rest of the (terrain) scope.
We're now ready to make the terrain. First, we check to see if the relevant storage macros are undefined and if so, let them to empty so that we can deal more easily with the various cases.

Possible make rotation. We define a scope and rotate within that.

If we have specified code for the terrain, then execute that.

First we check if we have not got terrain images, but terrain pictures. If we have that, we process these in turn. Note, the user can give options to each terrain picture by preceding the picture name with \[⟨options⟩\].

If the user specified images rather than pictures, then we process these in turn. Again, the user can specify options to each terrain image by preceding the image file name with \[⟨options⟩\].
Next, we define some example clippings of the terrain images. Specifically, we make clippings to sextants. We do this by first defining a macro.

\hex@make@sextants

When executed this macro will generate some paths that will clip to sextants. The first argument is the inner radius of the sextant and the second argument is the (possible empty) prefix to put in front of the \texttt{sextant} name.
Next, we define some styles for styling the terrain pictures. Users can change these styles (e.g., by appending to them) to change say the colour of the terrain graphics.

### hex/terrain/beach

The style for beach hexes. The pattern is filled with a yellowish colour, and drawing of the outline is disabled.

```latex
\tikzset{
\hex/terrain/beach/.style={%
fill={rgb,100:red,93;green,73;blue,35},%
draw=none%}
}%
```

### hex/terrain/beach

Now for the actual patterns. We go in the same order as above — i.e, we start with the beach pattern. This is rather long.

```latex
\if\hex@terrain@pic
\tikzset{
\hex/terrain/beach/.pic={
\path[\hex/terrain/beach,pic actions,draw=none]
(-0.4931, 0.8848) -- (-0.4998, 0.8734) .. controls (-0.4908, 0.8731) and (-0.4813, 0.8762) .. (-0.4762, 0.8847)
--cycle
}
\hex@make@sextants{.7}{}
\hex@make@sextants{.3}{large }
\hex@make@sextants{0}{full }
\}
```
1928 -- ( 0.1631, 0.2450)
1929 -- ( 0.0965, 0.2281)
1930 -- ( 0.1689, 0.1131)
1931 -- ( 0.2065, 0.0861)
1932 .. controls ( 0.2453, 0.0564) and ( 0.2384, 0.0410) .. ( 0.2923, 0.0323)
1933 -- ( 0.2988,-0.0188)
1934 .. controls ( 0.2994,-0.0695) and ( 0.2657,-0.0796) .. ( 0.2249,-0.0579)
1935 .. controls ( 0.1337,-0.0093) and ( 0.1545, 0.0219) .. ( 0.1102, 0.0744)
1936 .. controls ( 0.0914, 0.0967) and ( 0.0807, 0.1010) .. ( 0.0539, 0.1089)
1937 .. controls ( 0.0562, 0.0613) and ( 0.0756,-0.0434) .. ( 0.0403,-0.0825)
1938 .. controls (-0.0336,-0.1168) .. (-0.0567,-0.1294)
1939 .. controls (-0.0777,-0.0729) .. (-0.0703,-0.0546)
1940 .. controls (-0.0562, 0.0040) .. ( 0.0152, 0.0389)
1941 -- (-0.0397, 0.0480)
1942 -- (-0.0737, 0.0578)
1943 .. controls (-0.0806, 0.0391) and (-0.0849, 0.0192) .. (-0.1018, 0.0068)
1944 .. controls (-0.1154,-0.0032) and (-0.1352,-0.0018) .. (-0.1438,-0.0212)
1945 .. controls (-0.1562,-0.0491) and (-0.1117,-0.1243) .. (-0.0874,-0.1373)
1946 .. controls (-0.0745,-0.1434) and (-0.0687,-0.1394) .. (-0.0567,-0.1373)
1947 .. controls (-0.0358,-0.2033) and (-0.0062,-0.1612) .. ( 0.0370,-0.1500)
1948 -- ( 0.1050,-0.1379)
1949 .. controls ( 0.0882,-0.0871) and ( 0.0808,-0.0999) .. ( 0.0965,-0.0443)
1950 .. controls ( 0.1454,-0.0619) and ( 0.1336,-0.0743) .. ( 0.1664,-0.0940)
1951 .. controls ( 0.1897,-0.1081) and ( 0.2226,-0.1052) .. ( 0.2361,-0.1388)
1952 .. controls ( 0.2495,-0.1724) and ( 0.2245,-0.1963) .. ( 0.2412,-0.2584)
1953 .. controls ( 0.2526,-0.2569) and ( 0.2622,-0.2548) .. ( 0.2735,-0.2584)
1954 .. controls ( 0.2987,-0.2708) and ( 0.3225,-0.3241) .. ( 0.3212,-0.3506)
1955 .. controls ( 0.3203,-0.3711) and ( 0.3053,-0.3950) .. ( 0.3008,-0.4443)
1956 -- ( 0.2497,-0.4187)
1957 .. controls ( 0.2599,-0.4479) and ( 0.2621,-0.4475) .. ( 0.2905,-0.4528)
1958 .. controls ( 0.2877,-0.4715) and ( 0.2799,-0.4999) .. ( 0.2905,-0.5182)
1959 .. controls ( 0.2991,-0.5392) and ( 0.3228,-0.5357) .. ( 0.3346,-0.5182)
1960 .. controls ( 0.3506,-0.4943) and ( 0.3355,-0.4515) .. ( 0.3532,-0.4203)
1961 .. controls ( 0.3716,-0.3881) and ( 0.4096,-0.3844) .. ( 0.4084,-0.3499)
1962 .. controls ( 0.4074,-0.3241) and ( 0.3866,-0.3087) .. ( 0.3728,-0.2897)
1963 -- ( 0.3426,-0.2337)
1964 -- ( 0.2989,-0.1879)
1965 .. controls ( 0.2810,-0.1587) and ( 0.2976,-0.1327) .. ( 0.3187,-0.1323)
1966 .. controls ( 0.3342,-0.1319) and ( 0.3489,-0.1451) .. ( 0.3603,-0.1541)
1967 .. controls ( 0.3817,-0.1712) and ( 0.4026,-0.1894) .. ( 0.4144,-0.2146)
1968 .. controls ( 0.4299,-0.2477) and ( 0.4289,-0.2977) .. ( 0.4712,-0.3110)
1969 .. controls ( 0.4957,-0.3188) and ( 0.5167,-0.3024) .. ( 0.5044,-0.2753)
1970 .. controls ( 0.4967,-0.2585) and ( 0.4769,-0.2471) .. ( 0.4676,-0.2227)
1971 .. controls ( 0.4582,-0.1981) and ( 0.4681,-0.1743) .. ( 0.4488,-0.1492)
1972 .. controls ( 0.4286,-0.1227) and ( 0.3809,-0.1095) .. ( 0.3621,-0.0696)
1973 .. controls ( 0.3402,-0.0230) and ( 0.3896, 0.0270) .. ( 0.3092, 0.0408)
1974 -- ( 0.3532, 0.1933)
1975 -- ( 0.3944, 0.2536)
1976 -- ( 0.3433, 0.2765)
1977 --cycle
1978 ( 0.2497, 0.2450)
1979 -- ( 0.2782, 0.2025)
1980 .. controls ( 0.2843, 0.1911) and ( 0.2884, 0.1815) .. ( 0.2900, 0.1685)
The draw style for light woods. The pattern is filled with light green, and outline is not drawn.

```latex
\tikzset{
  hex/terrain/light woods/.style={
    draw=none,
    fill={rgb,100:red,69;green,98;blue,69}
  }
}
```

Next, we have light woods.

```latex
\ifhexterrainpic
\tikzset{
  hex/terrain/light woods/.pic={
    \path[hex/terrain/light woods,pic actions,draw=none]
    (-0.4795, 0.8736)
    -- (-0.5104, 0.8207)
    .. controls (-0.5041, 0.8191) and (-0.4967, 0.8182) .. (-0.4854, 0.8192)
    -- (-0.4770, 0.8108)
    -- (-0.4854, 0.7856)
    -- (-0.5190, 0.8023)
    .. controls (-0.5219, 0.7975) and (-0.5245, 0.7958) .. (-0.5272, 0.7916)
    --cycle
  };
}
```
.. controls (-0.7963,-0.2270) and (-0.7967,-0.1536) .. (-0.8471,-0.1391)
.. controls (-0.8514,-0.1378) and (-0.8560,-0.1374) .. (-0.8606,-0.1378)
--cycle
( 0.9187,-0.1555)
.. controls ( 0.9083,-0.1585) and ( 0.8971,-0.1627) .. ( 0.8847,-0.1675)
.. controls ( 0.8669,-0.1743) and ( 0.8469,-0.1785) .. ( 0.8399,-0.1989)
.. controls ( 0.8307,-0.2247) and ( 0.8481,-0.2329) .. ( 0.8679,-0.2399)
-- ( 0.8847,-0.2063)
.. controls ( 0.8863,-0.2068) and ( 0.8871,-0.2069) .. ( 0.8886,-0.2074)
--cycle
( 0.8679,-0.2399)
.. controls ( 0.8663,-0.2432) and ( 0.8651,-0.2468) .. ( 0.8637,-0.2502)
-- ( 0.8693,-0.2405)
.. controls ( 0.8687,-0.2402) and ( 0.8685,-0.2401) .. ( 0.8679,-0.2399)
--cycle
( 0.4392,-0.1558)
-- ( 0.4332,-0.1807)
.. controls ( 0.4281,-0.2431) and ( 0.5089,-0.2120) .. ( 0.4618,-0.1688)
.. controls ( 0.4519,-0.1597) and ( 0.4457,-0.1593) .. ( 0.4392,-0.1558)
--cycle
(-0.6846,-0.1952)
.. controls (-0.6966,-0.1951) and (-0.7082,-0.2013) .. (-0.7157,-0.2171)
.. controls (-0.7203,-0.2268) and (-0.7199,-0.2378) .. (-0.7207,-0.2483)
-- (-0.6787,-0.2399)
-- (-0.6703,-0.2735)
.. controls (-0.6096,-0.2523) and (-0.6488,-0.1955) .. (-0.6846,-0.1952)
--cycle
(-0.1120,-0.2035)
.. controls (-0.1188,-0.2048) and (-0.1255,-0.2071) .. (-0.1323,-0.2089)
.. controls (-0.1785,-0.2217) and (-0.2021,-0.2285) .. (-0.1912,-0.2819)
-- (-0.0988,-0.2483)
-- (-0.0651,-0.2740)
.. controls (-0.0333,-0.3228) and (-0.0165,-0.2917) .. (-0.0209,-0.2740)
.. controls (-0.0243,-0.2616) and (-0.0384,-0.2481) .. (-0.0474,-0.2386)
.. controls (-0.0590,-0.2264) and (-0.0744,-0.2085) .. (-0.0911,-0.2040)
.. controls (-0.0982,-0.2021) and (-0.1052,-0.2023) .. (-0.1120,-0.2035)
--cycle
( 0.1647,-0.2053)
.. controls ( 0.1471,-0.2058) and ( 0.1297,-0.2092) .. ( 0.1164,-0.2148)
.. controls ( 0.0833,-0.2632) and ( 0.1207,-0.3872) .. ( 0.1933,-0.3346)
.. controls ( 0.2031,-0.3275) and ( 0.2109,-0.3165) .. ( 0.2178,-0.3068)
.. controls ( 0.2722,-0.2297) and ( 0.2177,-0.2039) .. ( 0.1647,-0.2053)
--cycle
( 0.3262,-0.2328)
.. controls ( 0.3073,-0.2617) and ( 0.3314,-0.2707) .. ( 0.3420,-0.2638)
.. controls ( 0.3522,-0.2572) and ( 0.3585,-0.2285) .. ( 0.3262,-0.2328)
--cycle
( 0.1534,-0.2399)
-- ( 0.2039,-0.2483)
-- ( 0.1534,-0.2987)
--cycle
( 0.5217,-0.2636)
.. controls ( 0.5071,-0.2632) and ( 0.4918,-0.2708) .. ( 0.4798,-0.2909)
The style for woods. The pattern is filled with a darker green, and outlines are not drawn.

```
thex/terrain/woods

\tikzset{
  hex/terrain/woods/.style={
    draw=none,
    fill={rgb,100:red,27;green,67;blue,27}
  }
}
```

**hex/terrain/woods**

Regular woods.
The pattern for swamps. The pattern is filled with a light blue.

\begin{tikzpicture}
\fill[rgb,100:red,26;green,55;blue,70] (-0.5,0.87) -- (-0.496,0.86) .. controls (-0.35,0.84) and (-0.12,0.87) .. (0.0,0.87) -- (0.24,0.86) -- (0.24,0.87) --cycle (0.42,0.87) -- (0.42,0.86) -- (0.5,0.87) -- (0.5,0.86) --cycle (0.31,0.84) -- (0.28,0.8) -- (0.27,0.82) -- (0.25,0.82)
\end{tikzpicture}

Swamps. This is probably the shortest of the terrain patterns.
(0.1536, 0.0789)
--- (0.1536, 0.0196)
--- (0.0856, 0.0534)
--- (0.0686, -0.0145)
--- (0.7658, -0.0145)
.. controls (0.6332, 0.0380) and (0.4479, -0.0524) .. (0.3406, 0.0534)
--- (0.3236, 0.0534)
--- (0.2897, 0.0976)
--- (0.2556, 0.0196)
--- (0.2386, 0.0876)
--- (0.1705, 0.0365)
---cycle
--- (0.3917, 0.0876)
.. controls (0.3917, 0.1216) and (0.5022, 0.1216) .. (0.3917, 0.1216)
---cycle
--- (-0.9351, 0.1208)
--- (-0.9518, 0.0921)
--- (-0.9518, 0.0876)
--- (-0.6117, 0.1045)
---cycle
--- (-0.9144, 0.0213)
.. controls (-0.9468, 0.0204) and (-0.9775, 0.0109) .. (-0.9996, -0.0116)
--- (-0.9982, -0.0141)
--- (-0.8158, 0.0026)
.. controls (-0.8449, 0.0142) and (-0.8804, 0.0222) .. (-0.9144, 0.0213)
---cycle
--- (-0.6287, 0.0196)
.. controls (-0.5470, -0.0404) and (-0.2796, -0.0145) .. (-0.1695, -0.0145)
--- (-0.1695, 0.0196)
---cycle
--- (-0.9488, -0.0996)
--- (-0.9292, -0.1335)
--- (-0.4756, -0.1335)
--- (-0.4756, -0.0996)
---cycle
--- (-0.2886, -0.0996)
--- (-0.2886, -0.1335)
--- (0.2726, -0.1335)
.. controls (0.2164, -0.0920) and (0.1871, -0.0997) .. (0.1195, -0.0996)
---cycle
--- (0.5478, -0.1025)
.. controls (0.5070, -0.1018) and (0.4651, -0.1086) .. (0.4257, -0.1165)
--- (0.6638, -0.1335)
.. controls (0.6286, -0.1113) and (0.5887, -0.1031) .. (0.5478, -0.1025)
---cycle
--- (0.8928, -0.1132)
.. controls (0.8481, -0.1114) and (0.8007, -0.1165) .. (0.7658, -0.1165)
--- (0.9264, -0.1394)
--- (0.9384, -0.1186)
.. controls (0.9238, -0.1157) and (0.9087, -0.1137) .. (0.8928, -0.1132)
\begin{tikzpicture}[\hex@terrain@rough,\pic actions,draw=none]
\path
(-0.2701, 0.8873)
.. controls (-0.2982, 0.8927) and (-0.3250, 0.8675) .. (-0.3296, 0.8537)
.. controls (-0.3363, 0.8337) and (-0.3058, 0.8263) .. (-0.2820, 0.8610)
.. controls (-0.2717, 0.8450) and (-0.2591, 0.8228) .. (-0.2441, 0.8112)
.. controls (-0.2057, 0.7817) and (-0.1394, 0.7709) .. (-0.1208, 0.8270)
-- (-0.2226, 0.8355)
.. controls (-0.2359, 0.8698) and (-0.2532, 0.8840) .. (-0.2701, 0.8873)
--cycle

(-0.1081, 0.8792)
.. controls (-0.1371, 0.8680) and (-0.1265, 0.8900) .. (-0.1377, 0.8610)
.. controls (-0.1121, 0.8691) and (-0.1163, 0.8536) .. (-0.1081, 0.8792)
--cycle
( 0.1762, 0.8752)
-- ( 0.1761, 0.8710)
.. controls ( 0.1746, 0.8556) and ( 0.1707, 0.8704) .. ( 0.1822, 0.8575)
.. controls ( 0.1958, 0.8423) and ( 0.2514, 0.8065) .. ( 0.2435, 0.8694)
--cycle
( 0.3216, 0.8740)
.. controls ( 0.3061, 0.8744) and ( 0.2932, 0.8668) .. ( 0.2896, 0.8414)
.. controls ( 0.2869, 0.8222) and ( 0.3049, 0.8110) .. ( 0.3122, 0.7930)
\end{tikzpicture}

The style for rough hexes. The pattern is filled with a light brown, and outlines are not drawn.

```
\tikzset{
  hex/terrain/rough/.style={
    draw=none,
    fill={rgb,100:red,79;green,68;blue,41}
  }
}
```

hex/terrain/rough

Roughs. Again, a bit long.
The style for mountains. The pattern is filled with a darker brown, and outlines are not drawn. Note that the mountain pattern is the same as the beach pattern, just with a different colour.

\begin{lstlisting}
\tikzset{
    hex/terrain/mountains/.style={
        draw=none,
        fill={rgb,100:red,49;green,35;blue,1}
    }
}
\end{lstlisting}

And the mountains pattern. This is the same as the beach pattern, only filled with a darker brown colour.
\ifhex@terrain@pic
\tikzset{
  hex/terrain/mountains/.pic={
    \path [hex/terrain/mountains,pic actions,draw=none]
      (-0.4931, 0.8848) -- (-0.4998, 0.8734)
        .. controls (-0.4908, 0.8731) and (-0.4813, 0.8762) .. (-0.4762, 0.8847)
      --cycle
    (-0.4032, 0.8841)
      .. controls (-0.4004, 0.8804) and (-0.3988, 0.8794) .. (-0.3956, 0.8745)
      .. controls (-0.3760, 0.8443) and (-0.3811, 0.8330) .. (-0.3456, 0.8112)
      .. controls (-0.3250, 0.7986) and (-0.3271, 0.7770) .. (-0.2531, 0.8032)
      .. controls (-0.2294, 0.8375) and (-0.2984, 0.8503) .. (-0.3193, 0.8690)
      .. controls (-0.3243, 0.8735) and (-0.3281, 0.8785) .. (-0.3321, 0.8835)
      --cycle
    (-0.2462, 0.8828)
      .. controls (-0.2425, 0.8681) and (-0.2383, 0.8546) .. (-0.2293, 0.8461)
      .. controls (-0.2102, 0.8280) and (-0.1892, 0.8390) .. (-0.1859, 0.8669)
      .. controls (-0.1854, 0.8711) and (-0.1871, 0.8772) .. (-0.1875, 0.8822)
      --cycle
    (-0.0997, 0.8815)
      .. controls (-0.0971, 0.8706) and (-0.0941, 0.8597) .. (-0.0907, 0.8493)
      -- (-0.0570, 0.8578)
      .. controls (-0.0570, 0.8629) and (-0.0560, 0.8730) .. (-0.0553, 0.8812)
      --cycle
    ( 0.0213, 0.8805)
      .. controls ( 0.0222, 0.8725) and ( 0.0235, 0.8650) .. ( 0.0262, 0.8587)
      .. controls ( 0.0391, 0.8281) and ( 0.0706, 0.8199) .. ( 0.0917, 0.7894)
      .. controls ( 0.1112, 0.7609) and ( 0.1058, 0.7286) .. ( 0.1050, 0.6961)
      -- ( 0.1731, 0.7216)
      -- ( 0.1203, 0.8649)
      -- ( 0.1097, 0.8797)
      --cycle
    ( 0.2978, 0.8781)
      .. controls ( 0.2985, 0.8773) and ( 0.3002, 0.8756) .. ( 0.3008, 0.8749)
      .. controls ( 0.2854, 0.8687) and ( 0.2549, 0.8572) .. ( 0.2421, 0.8487)
      .. controls ( 0.2026, 0.8224) and ( 0.1905, 0.7567) .. ( 0.2046, 0.7132)
      .. controls ( 0.2146, 0.6819) and ( 0.2330, 0.6680) .. ( 0.2394, 0.6280)
      .. controls ( 0.2413, 0.6160) and ( 0.2468, 0.5527) .. ( 0.2446, 0.5437)
      .. controls ( 0.2396, 0.5232) and ( 0.2211, 0.5122) .. ( 0.2231, 0.4913)
      .. controls ( 0.2261, 0.4603) and ( 0.2686, 0.4388) .. ( 0.2891, 0.4194)
      .. controls ( 0.3020, 0.4071) and ( 0.3136, 0.3895) .. ( 0.3281, 0.3799)
      .. controls ( 0.3688, 0.3533) and ( 0.3905, 0.3863) .. ( 0.4199, 0.3902)
      .. controls ( 0.4350, 0.3921) and ( 0.4560, 0.3849) .. ( 0.4710, 0.3812)
      -- ( 0.4795, 0.4067)
      -- ( 0.4965, 0.4067)
      .. controls ( 0.5008, 0.3961) and ( 0.5009, 0.3893) .. ( 0.5112, 0.3811)
      .. controls ( 0.5112, 0.3811) and ( 0.6172, 0.3385) .. ( 0.6481, 0.3037)
-- (-1.0000, 0.0020)
-- (-0.9548, -0.0788)
.. controls (-0.9170, -0.0310) and (-0.9342, -0.0158) .. (-1.0000, 0.0068)
--cycle
(-0.2643, 0.0054)
.. controls (-0.2853, -0.0295) and (-0.2523, -0.0713) .. (-0.2182, -0.0843)
.. controls (-0.2024, -0.0902) and (-0.1781, -0.0944) .. (-0.1687, -0.0757)
.. controls (-0.1530, -0.0441) and (-0.2378, 0.0095) .. (-0.2643, 0.0054)
--cycle
(0.6299, -0.0102)
.. controls (0.6155, -0.0145) and (0.6071, -0.0342) .. (0.6128, -0.0510)
.. controls (0.6198, -0.0721) and (0.6440, -0.0790) .. (0.6606, -0.0986)
.. controls (0.6738, -0.1143) and (0.6761, -0.1328) .. (0.6948, -0.1437)
.. controls (0.7092, -0.1520) and (0.7311, -0.1484) .. (0.7401, -0.1664)
.. controls (0.7461, -0.3524) and (0.7289, -0.3324) .. (0.7620, -0.4187)
.. controls (0.7631, -0.4216) and (0.7642, -0.4246) .. (0.7652, -0.4275)
-- (0.8003, -0.3672)
.. controls (0.7976, -0.3636) and (0.7942, -0.3606) .. (0.7918, -0.3568)
.. controls (0.7778, -0.3349) and (0.7645, -0.2537) .. (0.7970, -0.2417)
.. controls (0.8206, -0.2330) and (0.8347, -0.2671) .. (0.8432, -0.2822)
-- (0.8469, -0.2872)
-- (0.8787, -0.2326)
-- (0.8594, -0.1993)
.. controls (0.8496, -0.1847) and (0.7996, -0.1314) .. (0.7847, -0.1281)
.. controls (0.7712, -0.1229) and (0.7642, -0.1268) .. (0.7518, -0.1281)
.. controls (0.7451, -0.1148) and (0.7397, -0.1014) .. (0.7293, -0.0886)
-- (0.6461, -0.0117)
.. controls (0.6402, -0.0090) and (0.6347, -0.0087) .. (0.6299, -0.0102)
--cycle
(-0.5178, -0.0844)
.. controls (-0.5461, -0.0820) and (-0.5852, -0.0947) .. (-0.5902, -0.1144)
.. controls (-0.6007, -0.1557) and (-0.5621, -0.1731) .. (-0.5414, -0.1997)
.. controls (-0.5274, -0.2177) and (-0.5229, -0.2355) .. (-0.5044, -0.2525)
.. controls (-0.4888, -0.2669) and (-0.4706, -0.2705) .. (-0.4543, -0.2923)
.. controls (-0.4420, -0.3087) and (-0.4220, -0.3707) .. (-0.4141, -0.3932)
.. controls (-0.3620, -0.3875) and (-0.3060, -0.4031) .. (-0.3060, -0.3592)
.. controls (-0.3060, -0.3272) and (-0.3358, -0.3272) .. (-0.3716, -0.3008)
.. controls (-0.4367, -0.2529) and (-0.4253, -0.2451) .. (-0.4621, -0.1914)
.. controls (-0.4700, -0.1800) and (-0.4814, -0.1685) .. (-0.4867, -0.1556)
.. controls (-0.4970, -0.1308) and (-0.4804, -0.1088) .. (-0.4966, -0.0923)
.. controls (-0.5011, -0.0877) and (-0.5087, -0.0853) .. (-0.5178, -0.0844)
--cycle
(-0.4165, -0.0846)
.. controls (-0.4357, -0.0807) and (-0.4622, -0.1075) .. (-0.4395, -0.1440)
.. controls (-0.4316, -0.1566) and (-0.4254, -0.1571) .. (-0.4141, -0.1634)
.. controls (-0.4094, -0.1522) and (-0.4042, -0.1415) .. (-0.4019, -0.1294)
.. controls (-0.3960, -0.1004) and (-0.4049, -0.0870) .. (-0.4165, -0.0846)
--cycle
(-0.9358, -0.1125)
-- (-0.8813, -0.2098)
.. controls (-0.8768, -0.1903) and (-0.8656, -0.1673) .. (-0.8723, -0.1485)
.. controls (-0.8801, -0.1269) and (-0.9022, -0.1274) .. (-0.9358, -0.1125)
For villages, towns, and cities, we need three styles: one for houses, and separate styles for regular and small roads. Note that we draw using the stroke colour for roads and houses.

\ifhex@terrain@pic
\tikzset{
\input{hex/terrain/town/road}
\input{hex/terrain/town/small road}
\input{hex/terrain/town/house}
}\fi
Now for village, town, and city patterns.

\tikzset{
  hex/terrain/village/.pic={
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      (0.0073, 0.8700) -- (0.3952, 0.3373)
      -- (0.3952, 0.3373) -- (0.3884, 0.2029)
      -- (0.3884, 0.2029) -- (0.3555, 0.1378)
      -- (0.3555, 0.1378) -- (0.3751, 0.0880)
      -- (0.3751, 0.0880) -- (0.2513,-0.1997)
      -- (0.2513,-0.1997) -- (0.1396,-0.4505)
      -- (0.1396,-0.4505) -- (0.0641,-0.6512)
      -- (0.0641,-0.6512) -- (0.0070,-0.8700)
      -- (0.0070,-0.8700) -- (0.0070,-0.8700)
    ;
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      (0.7575, 0.4367)
      -- (0.3945, 0.3375)
      -- (0.3945, 0.3375) -- (0.3952, 0.3373)
    ;
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      (-0.1900,-0.0806)
      -- (-0.1155, 0.1588)
    ;
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      (-0.1308, 0.1580)
      -- (-0.7603, 0.4394)
    ;
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      (-0.6615,-0.2309)
      -- (-0.6777,-0.3255)
    ;
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      (-0.7607,-0.4327)
    ;
  }
}
\path[hex/terrain/town/house,pic actions]
( 0.5094, 0.3292)
-- ( 0.5505, 0.3494)
-- ( 0.5773, 0.2947)
-- ( 0.5362, 0.2746)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1323, 0.2640)
-- (-0.0890, 0.2489)
-- (-0.1092, 0.1914)
-- (-0.1524, 0.2065)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4115,-0.5373)
-- ( 0.4390,-0.5006)
-- ( 0.4877,-0.5372)
-- ( 0.4601,-0.5739)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3095, 0.1272)
-- ( 0.3519, 0.1095)
-- ( 0.3284, 0.0533)
-- ( 0.2861, 0.0710)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3095, 0.1272)
-- ( 0.3519, 0.1095)
-- ( 0.3284, 0.0533)
-- ( 0.2861, 0.0710)
--cycle
;
\path[hex/terrain/town/house,pic actions]
A town.

\begin{center}
\begin{tikzpicture}
\begin{scope}[hex/terrain/town/.pic={
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.1432,-0.4518) -- (-0.0320,-0.2906) -- ( 0.0745,-0.0351) -- ( 0.1130,-0.0387);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0729,-0.0352) -- (-0.1716, 0.0254);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
(-0.2493, 0.5648) -- (-0.2192, 0.4501);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0677,-0.6538) -- ( 0.1754,-0.7052) -- ( 0.4358,-0.4688);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0439,-0.3617) -- (-0.0921,-0.5012) -- (-0.2865,-0.3243) -- (-0.4420,-0.4608) -- (-0.5795,-0.4446) -- (-0.6421,-0.3520);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0622,-0.6515) -- (-0.0316,-0.6176) -- (-0.0221,-0.5364);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0048,-0.2069)
},
\end{scope}
\end{tikzpicture}
\end{center}
\path[hex/terrain/town/house,pic actions] (-0.4493,-0.3075) -- (-0.4043,-0.2868) -- (-0.3710,-0.3593) -- (-0.4160,-0.3799) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.4493,-0.3075) -- (-0.4043,-0.2868) -- (-0.3710,-0.3593) -- (-0.4160,-0.3799) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5264,-0.1066) -- (-0.5514,-0.0681) -- (-0.5002,-0.0349) -- (-0.4753,-0.0733) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5264,-0.1066) -- (-0.5514,-0.0681) -- (-0.5002,-0.0349) -- (-0.4753,-0.0733) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.1978, 0.1663) -- (-0.1534, 0.1549) -- (-0.1685, 0.0960) -- (-0.2130, 0.1074) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.1978, 0.1663) -- (-0.1534, 0.1549) -- (-0.1685, 0.0960) -- (-0.2130, 0.1074) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2259, 0.4898) -- ( 0.2453, 0.4680) -- ( 0.2052, 0.4324) -- ( 0.1858, 0.4542) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2259, 0.4898) -- ( 0.2453, 0.4680)
\path[hex/terrain/town/house,pic actions]
( 0.2834,-0.4196)
\path[hex/terrain/town/house,pic actions]
( 0.1447,-0.5170)
\path[hex/terrain/town/house,pic actions]
( 0.0154,-0.5671)
\path[hex/terrain/town/house,pic actions]
(-0.2958,-0.3614)
\path[hex/terrain/town/house,pic actions]
(-0.2958,-0.3614)
\path[hex/terrain/town/house,pic actions]
(-0.0041,-0.2944)
-- ( 0.0064,-0.2671)
-- ( 0.0564,-0.2862)
-- ( 0.0460,-0.3135)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0041,-0.2944)
-- ( 0.0064,-0.2671)
-- ( 0.0564,-0.2862)
-- ( 0.0460,-0.3135)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1877,-0.2296)
-- (-0.1764,-0.2026)
-- (-0.1270,-0.2233)
-- (-0.1383,-0.2503)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1877,-0.2296)
-- (-0.1764,-0.2026)
-- (-0.1270,-0.2233)
-- (-0.1383,-0.2503)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1170,-0.3014)
-- (-0.1067,-0.2740)
-- (-0.0566,-0.2928)
-- (-0.0668,-0.3202)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1170,-0.3014)
-- (-0.1067,-0.2740)
-- (-0.0566,-0.2928)
-- (-0.0668,-0.3202)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0719,-0.3499)
-- (-0.0428,-0.3468)
-- (-0.0371,-0.4001)
-- (-0.0661,-0.4032)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0719,-0.3499)
-- (-0.0428,-0.3468)
-- (-0.0371,-0.4001)
\path[hex/terrain/town/house,pic actions] ( 0.0204,-0.2046) -- ( 0.0398,-0.1631) -- ( 0.0950,-0.1890) -- ( 0.0755,-0.2305) --cycle ;
\path[hex/terrain/town/house,pic actions] ( 0.5094, 0.3292) -- ( 0.5505, 0.3494) -- ( 0.5773, 0.2947) -- ( 0.5362, 0.2746) --cycle ;
\path[hex/terrain/town/house,pic actions] ( 0.5094, 0.3292) -- ( 0.5505, 0.3494) -- ( 0.5773, 0.2947) -- ( 0.5362, 0.2746) --cycle ;
\path[hex/terrain/town/house,pic actions] (-0.0647, 0.4710) -- (-0.0215, 0.4559) -- (-0.0416, 0.3984) -- (-0.0848, 0.4135) --cycle ;
\path[hex/terrain/town/house,pic actions] (-0.0647, 0.4710) -- (-0.0215, 0.4559) -- (-0.0416, 0.3984) -- (-0.0848, 0.4135) --cycle ;
\path[hex/terrain/town/house,pic actions] (-0.1476,-0.3704) -- (-0.1403,-0.3251) -- (-0.0802,-0.3347) -- (-0.0873,-0.3799) --cycle ;
\path[hex/terrain/town/house,pic actions] (-0.1476,-0.3704) -- (-0.1403,-0.3251) -- (-0.0802,-0.3347) -- (-0.0873,-0.3799) --cycle ;
\path[hex/terrain/town/house,pic actions] (-0.0755, 0.3210) --cycle ;
\path[hex/terrain/town/house,pic actions]
( 0.4115,-0.5373)
-- ( 0.4390,-0.5006)
-- ( 0.4877,-0.5372)
-- ( 0.4601,-0.5739)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.3095, 0.1272)
-- ( 0.3519, 0.1095)
-- ( 0.3284, 0.0533)
-- ( 0.2861, 0.0710)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.3095, 0.1272)
-- ( 0.3519, 0.1095)
-- ( 0.3284, 0.0533)
-- ( 0.2861, 0.0710)
--cycle

\path[hex/terrain/town/house,pic actions]
(-0.3558, 0.0291)
-- (-0.3124, 0.0144)
-- (-0.3318,-0.0433)
-- (-0.3753,-0.0287)
--cycle

\path[hex/terrain/town/house,pic actions]
(-0.3558, 0.0291)
-- (-0.3124, 0.0144)
-- (-0.3318,-0.0433)
-- (-0.3753,-0.0287)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.2904, 0.2714)
-- ( 0.3361, 0.2681)
-- ( 0.3318, 0.2074)
-- ( 0.2861, 0.2106)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.2904, 0.2714)
-- ( 0.3361, 0.2681)
\path[hex/terrain/town/house,pic actions]
(0.1275,-0.0001)
-- (0.1734,-0.0022)
-- (0.1707,-0.0630)
-- (0.1248,-0.0610)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.0645,-0.5272)
-- (-0.0415,-0.5245)
-- (-0.0383,-0.5509)
-- (-0.0612,-0.5536)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.3209, 0.2176)
-- (-0.2989, 0.2105)
-- (-0.3069, 0.1853)
-- (-0.3289, 0.1923)
--cycle
\path[hex/terrain/town/house,pic actions]
(0.3746, 0.1600)
-- (0.4021, 0.1699)
-- (0.4204, 0.1195)
-- (0.3929, 0.1095)
--cycle
\path[hex/terrain/town/house,pic actions]
(0.3746, 0.1600)
-- (0.4021, 0.1699)
-- (0.4204, 0.1195)
-- (0.3929, 0.1095)
--cycle
}
And finally a city
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(0.0066, 0.8705)
-- (0.3952, 0.3367)
-- (0.3885, 0.2021)
-- (0.3555, 0.1368)
-- (0.3751, 0.0869)
-- (0.2511, -0.2014)
-- (0.1393, -0.4528)
-- (0.0636, -0.6538)
-- (0.0063, -0.8731)
-- (0.0063, -0.8731)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(0.1775, 0.6355)
-- (0.4288, 0.7459)
-- (0.5543, 0.5148)
-- (0.5543, 0.5148)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(0.7584, 0.4363)
-- (0.3946, 0.3369)
-- (0.3946, 0.3369)
-- (0.3946, 0.3369)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(0.1428, -0.4540)
-- (-0.0329, -0.2925)
-- (0.0739, -0.0364)
-- (0.1645, -0.0483)
-- (-0.0329, -0.2925)
-- (0.0739, -0.0364)
-- (0.1645, -0.0483)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(0.0723, -0.0367)
-- (-0.5150, 0.0791)
-- (-0.5150, 0.0791)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(-0.1911, -0.0821)
-- (-0.0760, 0.3934)
-- (-0.0774, 0.3920)
-- (-0.0760, 0.3934)
-- (-0.0774, 0.3920)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(0.0439, 0.3452)
-- (-0.3449, 0.4978)
-- (-0.4614, 0.2954)
-- (-0.4614, 0.2954)
-- (-0.3449, 0.4978)
-- (-0.4614, 0.2954)
-- (-0.4614, 0.2954)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(-0.4152, 0.2683)
-- (-0.7626, 0.4390)
-- (-0.7626, 0.4390)
-- (-0.7626, 0.4390)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(-0.5846, 0.3486)
-- (-0.8106, -0.1286)
-- (-0.7727, -0.2079)
-- (-0.7727, -0.2079)
-- (-0.8106, -0.1286)
-- (-0.7727, -0.2079)
-- (-0.7727, -0.2079)
;
\path[hex/terrain/town/road, pic actions, hex/terrain/town/post road]
(-0.7053, -0.1745)
\path[hex/terrain/town/house,pic actions] ( 0.1222,-0.7854) -- ( 0.1514,-0.7854) -- ( 0.1514,-0.8390) -- ( 0.1222,-0.8390) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.3031,-0.7156) -- ( 0.3325,-0.7156) -- ( 0.3325,-0.7693) -- ( 0.3031,-0.7693) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.3574,-0.7174) -- ( 0.3867,-0.7174) -- ( 0.3867,-0.7712) -- ( 0.3574,-0.7712) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.3574,-0.7174) -- ( 0.3867,-0.7174) -- ( 0.3867,-0.7712) -- ( 0.3574,-0.7712) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.3742,-0.8016) -- ( 0.4036,-0.8016) -- ( 0.4036,-0.8554) -- ( 0.3742,-0.8554) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.3742,-0.8016) -- ( 0.4036,-0.8016) -- ( 0.4036,-0.8554) -- ( 0.3742,-0.8554) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.4107,-0.8072) -- ( 0.4400,-0.8072)
\path[hex/terrain/town/house,pic actions] (-0.0719,-0.6560) -- (-0.0678,-0.6849) -- (-0.1209,-0.6925) -- (-0.1250,-0.6635) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.1330,-0.6411) -- (-0.1375,-0.6700) -- (-0.1906,-0.6618) -- (-0.1861,-0.6329) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.0334,-0.7381) -- (-0.0042,-0.7381) -- (-0.0042,-0.7917) -- (-0.0334,-0.7917) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.0998,-0.7315) -- (-0.0706,-0.7315) -- (-0.0706,-0.7852) -- (-0.0998,-0.7852) --cycle;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.1269,-0.8074)
-- (-0.0977,-0.8054)
-- (-0.0940,-0.8590)
-- (-0.1232,-0.8610)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.2787,-0.7975)
-- (-0.2495,-0.7956)
-- (-0.2459,-0.8492)
-- (-0.2751,-0.8511)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.3966,-0.5592)
-- (-0.3802,-0.5834)
-- (-0.4246,-0.6135)
-- (-0.4411,-0.5892)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.3966,-0.5592)
-- (-0.3802,-0.5834)
-- (-0.4246,-0.6135)
-- (-0.4411,-0.5892)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.4189,-0.5000)
-- (-0.4033,-0.5248)
-- (-0.4488,-0.5534)
-- (-0.4644,-0.5286)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.4189,-0.5000)
-- (-0.4033,-0.5248)
-- (-0.4488,-0.5534)
-- (-0.4644,-0.5286)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\filldraw (-0.3561,-0.4332)
-- (-0.3310,-0.4483)
-- (-0.3586,-0.4944)
\path[hex/terrain/town/house,pic actions]
(-0.3837,-0.4793) --cycle
\path[hex/terrain/town/house,pic actions]
(-0.3561,-0.4332) -- (-0.3310,-0.4483) -- (-0.3586,-0.4944) -- (-0.3837,-0.4793) --cycle
\path[hex/terrain/town/house,pic actions]
(-0.3120,-0.4787) -- (-0.2896,-0.4976) -- (-0.3241,-0.5386) -- (-0.3466,-0.5198) --cycle
\path[hex/terrain/town/house,pic actions]
(-0.2660,-0.5113) -- (-0.2456,-0.5323) -- (-0.2840,-0.5697) -- (-0.3045,-0.5487) --cycle
\path[hex/terrain/town/house,pic actions]
(-0.3939,-0.2212) -- (-0.3666,-0.2319) -- (-0.3863,-0.2819) -- (-0.4135,-0.2712) --cycle
\path[hex/terrain/town/house,pic actions]
(-0.3939,-0.2212) -- (-0.3666,-0.2319) -- (-0.3863,-0.2819) -- (-0.4135,-0.2712) --cycle
\path[hex/terrain/town/house,pic actions]
\begin{tikzpicture}
\foreach \x in {0,0.5,1,1.5,2}
\foreach \y in {0,0.5,1,1.5,2}
\draw[fill=gray!10] (\x+0.5,\y+0.5) circle (0.3cm);
\end{tikzpicture}
\path[hex/terrain/town/house,pic actions]
  ( 0.3226, 0.3543)
  -- ( 0.3420, 0.3323)
  -- ( 0.3018, 0.2967)
  -- ( 0.2824, 0.3185)
  --cycle
  ;
\path[hex/terrain/town/house,pic actions]
  (-0.2277, 0.3599)
  -- (-0.2171, 0.3872)
  -- (-0.1671, 0.3676)
  -- (-0.1777, 0.3404)
  --cycle
  ;
\path[hex/terrain/town/house,pic actions]
  (-0.1722, 0.5368)
  -- (-0.1616, 0.5642)
  -- (-0.1116, 0.5446)
  -- (-0.1223, 0.5174)
  --cycle
  ;
\path[hex/terrain/town/house,pic actions]
  (-0.1722, 0.5368)
  -- (-0.1616, 0.5642)
  -- (-0.1116, 0.5446)
  -- (-0.1223, 0.5174)
  --cycle
  ;
\path[hex/terrain/town/house,pic actions]
  (-0.2400, 0.3081)
  -- (-0.2307, 0.3359)
  -- (-0.1797, 0.3189)
  -- (-0.1890, 0.2912)
\path[hex/terrain/town/house,pic actions] (-0.2343, 0.0622)
-- (-0.2239, 0.0896)
-- (-0.1737, 0.0706)
-- (-0.1840, 0.0432)
cycle
\path[hex/terrain/town/house,pic actions] (-0.2343, 0.0622)
-- (-0.2239, 0.0896)
-- (-0.1737, 0.0706)
-- (-0.1840, 0.0432)
cycle
\path[hex/terrain/town/house,pic actions] (-0.1289, 0.0933)
-- (-0.1186, 0.1207)
-- (-0.0683, 0.1017)
-- (-0.0787, 0.0743)
cycle
\path[hex/terrain/town/house,pic actions] (-0.1289, 0.0933)
-- (-0.1186, 0.1207)
-- (-0.0683, 0.1017)
-- (-0.0787, 0.0743)
cycle
\path[hex/terrain/town/house,pic actions] ( 0.2223, 0.7399)
-- ( 0.2483, 0.7532)
-- ( 0.2727, 0.7054)
-- ( 0.2467, 0.6920)
cycle
\path[hex/terrain/town/house,pic actions] ( 0.2223, 0.7399)
-- ( 0.2483, 0.7532)
-- ( 0.2727, 0.7054)
-- ( 0.2467, 0.6920)
cycle
\path[hex/terrain/town/house,pic actions] ( 0.5440, 0.7476)
-- ( 0.5576, 0.7217)
-- ( 0.5102, 0.6965)
-- ( 0.4965, 0.7224)
cycle
\path[hex/terrain/town/house,pic actions] ( 0.5440, 0.7476)
-- ( 0.5576, 0.7217)
\path[hex/terrain/town/house,pic actions]
(-0.3053, 0.3741)
-- (-0.2932, 0.4008)
-- (-0.2443, 0.3786)
-- (-0.2564, 0.3519)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.6751, 0.0465)
-- (-0.6666, 0.0746)
-- (-0.6152, 0.0590)
-- (-0.6237, 0.0310)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7316,-0.0595)
-- (-0.7215,-0.0320)
-- (-0.6711,-0.0505)
-- (-0.6812,-0.0780)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7748,-0.1355)
-- (-0.7629,-0.1088)
-- (-0.7138,-0.1305)
-- (-0.7257,-0.1573)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7748,-0.1355)
-- (-0.7629,-0.1088)
-- (-0.7138,-0.1305)
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4687, 0.0016)
-- (-0.4334, 0.0016)
-- (-0.4334, -0.0022)
-- (-0.4687, -0.0022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.5170, 0.0545)
-- (-0.4876, 0.0545)
-- (-0.4876, 0.0008)
-- (-0.5170, 0.0008)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4695, 0.1540)
-- (-0.4413, 0.1461)
-- (-0.4557, 0.0943)
-- (-0.4839, 0.1022)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4511, 0.2320)
-- (-0.4227, 0.2241)
-- (-0.4372, 0.1724)
-- (-0.4654, 0.1803)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4511, 0.2320)
-- (-0.4227, 0.2241)
-- (-0.4372, 0.1724)
-- (-0.4654, 0.1803)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4191, 0.1396)
-- (-0.3904, 0.1335)
-- (-0.4017, 0.0809)
-- (-0.4304, 0.0871)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4191, 0.1396)
-- (-0.3904, 0.1335)
-- (-0.4017, 0.0809)
-- (-0.4304, 0.0871)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0048,-0.2963)

247
\path[hex/terrain/town/house,pic actions]
(-0.0048,-0.2963)
-- ( 0.0056,-0.2689)
-- ( 0.0558,-0.2881)
-- ( 0.0453,-0.3155)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0804,-0.1001)
-- ( 0.0908,-0.0728)
-- ( 0.1410,-0.0919)
-- ( 0.1305,-0.1192)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0979,-0.2299)
-- (-0.0449,-0.2387)
-- (-0.0498,-0.2676)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1281,-0.2251)
-- (-0.1394,-0.2521)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1776,-0.2043)
-- (-0.1776,-0.2043)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1776,-0.2043)
-- (-0.1776,-0.2043)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1776,-0.2043)
-- (-0.1776,-0.2043)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1776,-0.2043)
-- (-0.1776,-0.2043)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1776,-0.2043)
-- (-0.1776,-0.2043)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1776,-0.2043)
-- (-0.1776,-0.2043)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4612, 0.0957)
-- ( 0.4899, 0.0898)
-- ( 0.4793, 0.0371)
-- ( 0.4506, 0.0430)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.4612, 0.0957)
-- ( 0.4899, 0.0898)
-- ( 0.4793, 0.0371)
-- ( 0.4506, 0.0430)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.5422,-0.0041)
-- ( 0.5437,-0.0333)
-- ( 0.4900,-0.0361)
-- ( 0.4885,-0.0068)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.5422,-0.0041)
-- ( 0.5437,-0.0333)
-- ( 0.4900,-0.0361)
-- ( 0.4885,-0.0068)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.6654,-0.0050)
-- ( 0.6664,-0.0343)
-- ( 0.6128,-0.0361)
-- ( 0.6117,-0.0068)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.6654,-0.0050)
-- ( 0.6664,-0.0343)
-- ( 0.6128,-0.0361)
-- ( 0.6117,-0.0068)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.7573, 0.2817)
-- ( 0.7646, 0.2533)
-- ( 0.7124, 0.2402)
-- ( 0.7053, 0.2686)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.7573, 0.2817)
\path [hex/terrain/town/house, pic actions]
\path [hex/terrain/town/house, pic actions]
\path [hex/terrain/town/house, pic actions]
\path [hex/terrain/town/house, pic actions]
\path [hex/terrain/town/house, pic actions]
\path [hex/terrain/town/house, pic actions]
\path [hex/terrain/town/house, pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\begin{center}
\begin{tikzpicture}

% Define the house shape
\node (house) at (0,0) {
\fill[blue!20] (0.0,0.0) -- (0.3,0.0) -- (0.3,0.3) -- (0.0,0.3) -- cycle;
\};

% Define the house on the left side
\node (house1) at (-0.5,0.5) {
\fill[blue!20] (0.0,0.0) -- (0.3,0.0) -- (0.3,0.3) -- (0.0,0.3) -- cycle;
\};

% Define the house on the right side
\node (house2) at (0.5,0.5) {
\fill[blue!20] (0.0,0.0) -- (0.3,0.0) -- (0.3,0.3) -- (0.0,0.3) -- cycle;
\};

% Define the house on the bottom
\node (house3) at (-0.5,-0.5) {
\fill[blue!20] (0.0,0.0) -- (0.3,0.0) -- (0.3,0.3) -- (0.0,0.3) -- cycle;
\};

% Define the house on the top
\node (house4) at (0.5,-0.5) {
\fill[blue!20] (0.0,0.0) -- (0.3,0.0) -- (0.3,0.3) -- (0.0,0.3) -- cycle;
\};

\end{tikzpicture}
\end{center}
\path[hex/terrain/town/house,pic actions]
( 0.3106, -0.7770)
-- ( 0.3564, -0.7770)
-- ( 0.3564, -0.8380)
-- ( 0.3106, -0.8380)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0626, -0.7954)
-- (-0.0196, -0.8113)
-- (-0.0406, -0.8686)
-- (-0.0837, -0.8527)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0626, -0.7954)
-- (-0.0196, -0.8113)
-- (-0.0406, -0.8686)
-- (-0.0837, -0.8527)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0570, -0.7843)
-- ( 0.1025, -0.7910)
-- ( 0.0936, -0.8514)
-- ( 0.0481, -0.8446)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0570, -0.7843)
-- ( 0.1025, -0.7910)
-- ( 0.0936, -0.8514)
-- ( 0.0481, -0.8446)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0906, -0.6908)
-- ( 0.1345, -0.7046)
-- ( 0.1161, -0.7629)
-- ( 0.0723, -0.7490)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0906, -0.6908)
-- ( 0.1345, -0.7046)
-- ( 0.1161, -0.7629)
-- ( 0.0723, -0.7490)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4731, -0.7998)
\path[hex/terrain/town/house,pic actions]
  ( 0.2923,-0.1390)  -- ( 0.3109,-0.0970)
  -- ( 0.3667,-0.1218)  -- ( 0.3481,-0.1638)
  --cycle
\path[hex/terrain/town/house,pic actions]
  ( 0.6866,-0.0789)  -- ( 0.7324,-0.0789)
  -- ( 0.7324,-0.1400)  -- ( 0.6866,-0.1400)
  --cycle
\path[hex/terrain/town/house,pic actions]
  ( 0.8206,-0.0922)  -- ( 0.8649,-0.1044)
  -- ( 0.8487,-0.1632)  -- ( 0.8045,-0.1511)
  --cycle
\path[hex/terrain/town/house,pic actions]
  ( -0.3075, 0.5809)  -- (-0.2648, 0.5640)
  -- (-0.2872, 0.5072)  -- (-0.3299, 0.5241)
  --cycle
\path[hex/terrain/town/house,pic actions]
(-0.7746, 0.3900)
-- (-0.7312, 0.3750)
-- (-0.7511, 0.3173)
-- (-0.7945, 0.3322)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7746, 0.3900)
-- (-0.7312, 0.3750)
-- (-0.7511, 0.3173)
-- (-0.7945, 0.3322)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.8224, 0.3024)
-- (-0.7807, 0.2831)
-- (-0.8064, 0.2277)
-- (-0.8481, 0.2470)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.8224, 0.3024)
-- (-0.7807, 0.2831)
-- (-0.8064, 0.2277)
-- (-0.8481, 0.2470)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7172, 0.2999)
-- (-0.6959, 0.3406)
-- (-0.6418, 0.3122)
-- (-0.6632, 0.2715)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7172, 0.2999)
-- (-0.6959, 0.3406)
-- (-0.6418, 0.3122)
-- (-0.6632, 0.2715)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7505, 0.2368)
-- (-0.7273, 0.2764)
-- (-0.6746, 0.2456)
-- (-0.6979, 0.2060)
--cycle
;
\path[hex/terrain/town/house,pic actions] (0.5097, 0.3286) -- (0.5510, 0.3488) -- (0.5778, 0.2940) -- (0.5366, 0.2738) --cycle;
\path[hex/terrain/town/house,pic actions] (0.4014, 0.8173) -- (0.4429, 0.8369) -- (0.4689, 0.7817) -- (0.4274, 0.7621) --cycle;
\path[hex/terrain/town/house,pic actions] (0.2627, 0.7599) -- (0.3055, 0.7765) -- (0.3276, 0.7196) -- (0.2848, 0.7030) --cycle;
\path[hex/terrain/town/house,pic actions] (0.1763, 0.7193) -- (0.2174, 0.7400) -- (0.2448, 0.6855) -- (0.2038, 0.6648) --cycle;
\path[hex/terrain/town/house,pic actions]
( 0.1763, 0.7193)
-- ( 0.2174, 0.7400)
-- ( 0.2448, 0.6856)
-- ( 0.2038, 0.6648)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.0655, 0.4707)
-- (-0.0222, 0.4555)
-- (-0.0424, 0.3979)
-- (-0.0858, 0.4131)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.0655, 0.4707)
-- (-0.0222, 0.4555)
-- (-0.0424, 0.3979)
-- (-0.0858, 0.4131)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.0019, 0.5606)
-- ( 0.0452, 0.5454)
-- ( 0.0251, 0.4878)
-- (-0.0183, 0.5030)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.0019, 0.5606)
-- ( 0.0452, 0.5454)
-- ( 0.0251, 0.4878)
-- (-0.0183, 0.5030)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.0634, 0.0555)
-- ( 0.1067, 0.0403)
-- ( 0.0865,-0.0174)
-- ( 0.0432,-0.0022)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.0634, 0.0555)
-- ( 0.1067, 0.0403)
-- ( 0.0865,-0.0174)
-- ( 0.0432,-0.0022)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.0445, 0.0687)
-- (-0.0010, 0.0535)
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
( 0.4450, 0.0292)
-- ( 0.4606,-0.0141)
-- ( 0.4032,-0.0348)
-- ( 0.3876, 0.0085)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.9043, 0.1125)
-- ( 0.9184, 0.0687)
-- ( 0.8603, 0.0500)
-- ( 0.8462, 0.0937)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.9043, 0.1125)
-- ( 0.9184, 0.0687)
-- ( 0.8603, 0.0500)
-- ( 0.8462, 0.0937)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.7148,-0.1425)
-- ( 0.5891,-0.1425)
-- ( 0.5806,-0.1876)
-- ( 0.5207,-0.1764)
-- ( 0.5291,-0.1313)
--cycle
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
(-0.4969,-0.5222)
-- (-0.4767,-0.5634)
-- (-0.5315,-0.5902)
-- (-0.5518,-0.5490)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3963,-0.6922)
-- (-0.3778,-0.7343)
-- (-0.4338,-0.7588)
-- (-0.4522,-0.7168)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.6145,-0.5157)
-- (-0.5944,-0.5570)
-- (-0.6493,-0.5836)
-- (-0.6694,-0.5423)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.6145,-0.5157)
-- (-0.5944,-0.5570)
-- (-0.6493,-0.5836)
-- (-0.6694,-0.5423)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.8791, -0.2053)
-- (-0.8402, -0.1810)
-- (-0.8079, -0.2327)
-- (-0.8468, -0.2571)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.8791, -0.2053)
-- (-0.8402, -0.1810)
-- (-0.8079, -0.2327)
-- (-0.8468, -0.2571)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.9144, 0.1039)
-- (-0.8919, 0.1439)
-- (-0.8388, 0.1140)
-- (-0.8613, 0.0740)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.9144, 0.1039)
-- (-0.8919, 0.1439)
-- (-0.8388, 0.1140)
-- (-0.8613, 0.0740)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4095, 0.2277)
-- (-0.3639, 0.2229)
-- (-0.3703, 0.1622)
-- (-0.4159, 0.1670)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4095, 0.2277)
-- (-0.3639, 0.2229)
-- (-0.3703, 0.1622)
-- (-0.4159, 0.1670)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3590, 0.4200)
-- (-0.3365, 0.4600)
-- (-0.2833, 0.4302)
-- (-0.3058, 0.3901)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3590, 0.4200)
-- (-0.3365, 0.4600)

277
\path[hex/terrain/town/house,pic actions]
(-0.6443,-0.1022)
-- (-0.6321,-0.0579)
-- (-0.5733,-0.0740)
-- (-0.5854,-0.1183)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.6032,0.2357)
-- (-0.5912,0.2800)
-- (-0.5323,0.2639)
-- (-0.5443,0.2196)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7230,0.0020)
-- (-0.7026,0.0432)
-- (-0.6479,0.0162)
-- (-0.6682,-0.0250)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7230,0.0020)
-- (-0.7026,0.0432)
-- (-0.6479,0.0162)
-- (-0.6682,-0.0250)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.5055,0.2596)
-- (-0.4629,0.2423)
-- (-0.4859,0.1858)
-- (-0.5285,0.2031)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.5055,0.2596)
-- (-0.4629,0.2423)
-- (-0.4859,0.1858)
-- (-0.5285,0.2031)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(0.1337,0.3296)
-- (0.1283,0.2840)
-- (0.0677,0.2911)
\path[hex/terrain/town/house,pic actions]
(-0.1683,-0.0622)
-- (-0.1535,-0.0188)
-- (-0.0958,-0.0385)
-- (-0.1106,-0.0820)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1842,-0.1310)
-- (-0.1694,-0.0876)
-- (-0.1116,-0.1073)
-- (-0.1264,-0.1508)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1167,-0.5813)
-- ( 0.1315,-0.5379)
-- ( 0.1892,-0.5576)
-- ( 0.1744,-0.6011)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1167,-0.5813)
-- ( 0.1315,-0.5379)
-- ( 0.1892,-0.5576)
-- ( 0.1744,-0.6011)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0916,-0.6322)
-- ( 0.1064,-0.5888)
-- ( 0.1642,-0.6085)
-- ( 0.1493,-0.6520)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0916,-0.6322)
-- ( 0.1064,-0.5888)
-- ( 0.1642,-0.6085)
-- ( 0.1493,-0.6520)
\path[hex/terrain/town/house,pic actions]
( 0.6499, 0.5535)
-- ( 0.6714, 0.5129)
-- ( 0.6174, 0.4844)
-- ( 0.5959, 0.5250)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.4994, 0.7998)
-- (-0.4558, 0.8143)
-- (-0.4364, 0.7565)
-- (-0.4800, 0.7419)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.4994, 0.7998)
-- (-0.4558, 0.8143)
-- (-0.4364, 0.7565)
-- (-0.4800, 0.7419)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.3350, 0.7630)
-- (-0.2917, 0.7475)
-- (-0.3125, 0.6901)
-- (-0.3558, 0.7057)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.3350, 0.7630)
-- (-0.2917, 0.7475)
-- (-0.3125, 0.6901)
-- (-0.3558, 0.7057)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.4472, 0.6282)
-- (-0.4040, 0.6124)
-- (-0.4252, 0.5551)
-- (-0.4683, 0.5710)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.4472, 0.6282)
-- (-0.4040, 0.6124)
-- (-0.4252, 0.5551)
-- (-0.4683, 0.5710)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.5200, 0.6603)
-- (-0.4792, 0.6394)
\path[hex/terrain/town/house,pic actions]
(-0.4531, -0.1794)
-- (-0.4094, -0.1941)
-- (-0.4289, -0.2518)
-- (-0.4725, -0.2372)
cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.4531, -0.1794)
-- (-0.4094, -0.1941)
-- (-0.4289, -0.2518)
-- (-0.4725, -0.2372)
cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3573, 0.0277)
-- (-0.3138, 0.0131)
-- (-0.3333, -0.0447)
-- (-0.3768, -0.0300)
cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3573, 0.0277)
-- (-0.3138, 0.0131)
-- (-0.3333, -0.0447)
-- (-0.3768, -0.0300)
cycle
;
\path[hex/terrain/town/house,pic actions]
(0.3354, -0.4695)
-- (0.3141, -0.5101)
-- (0.2601, -0.4816)
-- (0.2815, -0.4410)
cycle
;
\path[hex/terrain/town/house,pic actions]
(0.3354, -0.4695)
-- (0.3141, -0.5101)
-- (0.2601, -0.4816)
-- (0.2815, -0.4410)
cycle
;
\path[hex/terrain/town/house,pic actions]
(0.6206, -0.4111)
-- (0.6599, -0.4350)
-- (0.6281, -0.4872)
-- (0.5889, -0.4632)
cycle
;
\path[hex/terrain/town/house,pic actions]
(0.6206, -0.4111)
-- (0.6599, -0.4350)
-- (0.6281, -0.4872)
-- ( 0.5889, -0.4632)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.6061, -0.5834)
-- ( 0.6495, -0.5984)
-- ( 0.6296, -0.6561)
-- ( 0.5861, -0.6411)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2902, 0.2707)
-- ( 0.3361, 0.2673)
-- ( 0.3317, 0.2065)
-- ( 0.2859, 0.2098)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2215, 0.2766)
-- ( 0.2673, 0.2733)
-- ( 0.2630, 0.2124)
-- ( 0.2172, 0.2157)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0159, 0.4498)
-- ( 0.0299, 0.4466)
-- ( 0.0256, 0.3857)
-- ( -0.0202, 0.3889)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
( 0.0718, 0.3637)
-- ( 0.0933, 0.3723)
-- ( 0.1032, 0.3476)
-- ( 0.0817, 0.3390)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.2555, 0.2647)
-- (-0.2413, 0.2902)
-- (-0.1944, 0.2641)
-- (-0.2086, 0.2385)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.2832, 0.1509)
-- (-0.2826, 0.1802)
-- (-0.2289, 0.1792)
-- (-0.2295, 0.1498)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.5694, 0.6977)
-- (-0.5248, 0.6870)
-- (-0.5390, 0.6277)
-- (-0.5837, 0.6384)
--cycle
This is an example of a terrain picture.
5.4.6 Ridges

A hex can be decorated with up to 6 ridges — one for each edge of the hexagon. The first thing is to set up the graphics style to use for the ridges. We use the `wave` decoration.

If rounded corners are set for ridges, (e.g., via `every hex ridges`), then it should be 0pt or 4pt (roughly 2mm) or larger. Otherwise, one will get a “dimension too large” error.
To properly set up the ridges, we need to concatenate ridge paths in order. To facilitate that, we define 6 \ifs — one for each edge.

\newif\ifhex@r@ne
\newif\ifhex@r@n
\newif\ifhex@r@nw
\newif\ifhex@r@sw
\newif\ifhex@r@s
\newif\ifhex@r@se

Next is the keys for each edge. These will set the above \ifs to true. We put these into the family /hex/r so that we can parse them separately.

\tikzset{%
/\hex/ridges/.search also={/tikz},
/\hex/ridges/.cd,
north east/.is if=hex@r@ne,
north/.is if=hex@r@n,
north west/.is if=hex@r@nw,
south west/.is if=hex@r@sw,
south/.is if=hex@r@s,
south east/.is if=hex@r@se,
r/.store in=\hex@r@r,
n/.store in=\hex@r@n,
R/.store in=\hex@r@R,
NE/.is if=hex@r@ne,
N/.is if=hex@r@n,
NW/.is if=hex@r@nw,
SW/.is if=hex@r@sw,
S/.is if=hex@r@s,
SE/.is if=hex@r@se,
r/.store in=\hex@r@r,
n/.store in=\hex@r@n,
R/.store in=\hex@r@R,
%}

\hex@do@ridges

This is the macro that actually generates the ridge. We use the same PGF filtered key parsing trick as above. Note that the routine below is handcrafted since it is relatively simple.
5.4.7 Towns

Similar to above, we define a namespace and family for towns. First thing is the graphics style for towns.

\begin{verbatim}
\tikzset{%
  hex/town/.style={
    solid, scale line widths, thin,
    fill=pgfstrokecolor, color=pgfstrokecolor,\hex{town name}/.style={
    transform shape, shape=rectangle, above right=.1, color=pgfstrokecolor, font=\sffamily\fontsize{11}{13}\selectfont}
  }\end{verbatim}

Next is the namespace for dealing with towns.

\begin{verbatim}
\tikzset{%
  /hex/town/.search also={/tikz,%,
  /hex/town/.cd, pic/.store in=\hex@c\pic, type/.store in=\hex@c\pic,}\end{verbatim}
place/.store in=\hex@c@pos,
location/.store in=\hex@c@pos,
name/.store in=\hex@c@name,
village/.style={pic=hex/town/village},
town/.style={pic=hex/town/town},
city/.style={pic=hex/town/city}
}

And some pictures for making the towns.
\tikzset{%
  hex/town/village/.pic={\path[\fill,solid,pic actions] circle(.1);},
  hex/town/town/.pic={\path[\fill,solid,pic actions] circle(.2);},
  hex/town/city/.pic={%
    \path[\fill,solid,pic actions] circle(.25);
    \path[\draw,solid,pic actions] circle(.35);%
  }%
}
\hex@do@town

The macro to make the towns. This uses same tricks as above.
\def\hex@c@nameparse{%
  \@ifnextchar[^{\hex@c@nameparse}{\hex@c@nameparse[^]}}%
}
\def\hex@c@nameparse[^#1]#2\endhex@c@nameparse{%
  \def\hex@c@node{node[shape=rectangle,hex/town name,#1]{#2}}%
}
\def\hex@do@town{%
  \edef\hex@c@tmp{[
    /hex/town/.cd,%
    town,%
    /tikz/hex/town,%
    /tikz/every hex town/.try,
    \hex@town]
  }
  \expandafter\scope\hex@c@tmp%
    \ifx\hex@c@pic\empty\else\fi%
    \@ifundefined{hex@c@pos}{\let\hex@c@pos\empty}{%
      \@ifundefined{hex@c@name}{\let\hex@c@name\empty}{%
        \hex@c@nameparse\hex@c@name\endhex@c@nameparse%
        \ifx\hex@c@pos\empty\def\hex@c@pos{(0,0)}\fi%
        \hex@c@pos{2}{Town:
          ^^J text=\hex@c@name
          ^^J pic=\hex@c@pic
          ^^J place=\hex@c@pos
          ^^J node=\hex@c@node
        }%
        \filldraw \hex@c@pos pic{\hex@c@pic} \hex@c@node;
      \fi%
    \fi%
    \endscope%
}
5.4.8 Labels

Like terrains, we will set up some macros for dealing with labels.

To process coordinates and turn them into labels, we set up two counters.

\begin{verbatim}
\newcounter{hex@l@c}
\newcounter{hex@l@r}
\end{verbatim}

In case we want to invert the row axis, we set-up a key to set the maximum row number.

\begin{verbatim}
\def\hex@max@row{-1}
\tikzset{
  max hex row/.store in=\hex@max@row,
}
\end{verbatim}

Again, we will make a separate namespace/family for the handling labels. We also define a counter which we will use to typeset alphabetic column numbers.

First a graphics style.

\begin{verbatim}
\tikzset{%
  hex/label/.style={%
    draw=none,%,
    shape=rectangle,%,
    anchor=north,%,
    color=gray,%,
    font=\sffamily\bfseries\scriptsize,%
    inner sep=0},%
}
\end{verbatim}

Next, the choices of how to make a label. These are put in the \texttt{/hex/label} family to make it easy to parse out only these keys. This uses some macros defined below. Note, this uses the macros \texttt{\hex@col} and \texttt{\hex@row} defined by the hex coordinate system.

\begin{verbatim}
\tikzset{%
  /hex/label/.search also={/tikz},
  /hex/label/.cd,
  none/.code={\global\let\hex@l@text\@empty},
  auto/.is choice,
  auto/none/.code={\global\let\hex@l@text\@empty},
  auto/numbers/.code={\hex@l@abs
    \xdef\hex@l@text{\hex@l@n@pad{\the\c@hex@l@c}\hex@l@n@pad{\the\c@hex@l@r}}},
  auto/alpha column/.code={\xdef\hex@l@text{
    \ifnum0>\hex@col\AlphAlph{-\hex@col}\else\AlphAlph{\hex@col}\fi\hex@rov}},
  auto/alpha 2 column/.code={\xdef\hex@l@text{
    \advance\c@hex@l@c27\relax
    \AlphAlph{\value{hex@l@c}}}%
}
\end{verbatim}
\def\hex@l@abs{
\setcounter{hex@l@c}{\hex@col}
\setcounter{hex@l@r}{\hex@row}
\expandafter\ifnum\value{hex@l@c}<0\multiply\c@hex@l@c by-1\fi%
\expandafter\ifnum\value{hex@l@r}<0\multiply\c@hex@l@r by-1\fi%
% \hex@dbg{0}{\hex@col->\the\c@hex@l@c\space\hex@row->\the\c@hex@l@r}
\hex@l@n@pad{\hex@l@col},
\hex@l@n@pad{\hex@l@row]},
auto/.default=numbers,
text/.store in=\hex@l@text,
text/.code={\gdef\hex@l@text{#1}},
place/.store in=\hex@l@pos,
location/.forward to=/hex/label/place,
rotate/.store in=\hex@l@rot
\}

This takes the absolute value of row and column numbers.
\def\hex@l@abs{
\setcounter{hex@l@c}{\hex@col}
\setcounter{hex@l@r}{\hex@row}
\expandafter\ifnum\value{hex@l@c}<0\multiply\c@hex@l@c by-1\fi%
\expandafter\ifnum\value{hex@l@r}<0\multiply\c@hex@l@r by-1\fi%
% \hex@dbg{0}{\hex@col->\the\c@hex@l@c\space\hex@row->\the\c@hex@l@r}
\hex@l@n@pad{\hex@l@col},
\hex@l@n@pad{\hex@l@row]},
auto/.default=numbers,
text/.store in=\hex@l@text,
text/.code={\gdef\hex@l@text{#1}},
place/.store in=\hex@l@pos,
location/.forward to=/hex/label/place,
rotate/.store in=\hex@l@rot
\}

This will pad a number with a 0 if the number is smaller than 10.
\long\def\hex@l@n@pad#1{%
\ifnum#1<10 0\fi%
#1}

This macro puts in the label. First, we reset label keys, then we read in the keys from the argument. If this results in the macro \hex@l@text to be non-empty, then we set the label via a Ti\LaTeX node.
5.4.9 Extra graphics

To make the interface a bit more flexible we allow for adding arbitrary stuff to the hexes.

Some examples of pictures to add in the extra stuff.

```
\tikzset{%
\hex/fortress/.pic={%
\path[draw,solid,pic actions]
(0:.9) --
(0:.7) --
(60:.7) -- (60:.9) -- (60:.7) --
(120:.7) -- (120:.9) -- (120:.7) --
(180:.7) -- (180:.9) -- (180:.7) --
(240:.7) -- (240:.9) -- (240:.7) --
(300:.7) -- (300:.9) -- (300:.7) --
(0:.7) -- cycle;}}
```

```
\tikzset{%
\hex/fortress 2/.pic={%
\draw[pic actions,transform shape] (0:0.64)
foreach \a in {15,45,...,345}{
```

```
5.4.10 Some macros

\DeclareRobustCommand\fortmark[1][scale=.25]{\tikz[#1,transform shape]{\pic{hex/fortress 2}}}
\providecommand\terrainmark[2][scale=.2]{\tikz[#1]{\hex[label=,terrain=#2]}}
\providecommand\clearhex[1][scale=.2]{\tikz[#1]{\hex[label=]}}
\providecommand\woodshex[1][scale=.2]{\terrainmark[#1]{woods}}
\providecommand\mountainhex[1][scale=.2]{\terrainmark[#1]{mountains}}
\providecommand\cityhex[1][scale=.2]{\terrainmark[#1]{city}}
\providecommand\beachhex[1][scale=.2]{\terrainmark[#1]{beach}}
\providecommand\seaehex[1][scale=.2]{\tikz[#1]{\hex[label=,fill=sea]}}
\providecommand\riverhex[1][scale=.2]{\tikz[#1]{%}
\hex[label=](c=0,r=0)%
\river(hex cs:e=SW)--(hex cs:e=NE);}}
\providecommand\roadhex[1][scale=.2]{\tikz[#1]{%}
\hex[label=](c=0,r=0)%
\road(hex cs:e=SW)--(hex cs:e=NE);}}

5.4.11 Edges, borders, roads, rivers, and so on

Styles of drawing edges, borders, rivers, and railroads.

\pgfdeclaredecoration{outline}{init}{%
\state{init}[next state=tick,width=0pt]{%
\xdef\outlinerev{}%}
\state{tick}[width=+\pgfdecorationsegmentlength]%{}
\pgfpathlineto{%\the\pgf@x,\the\pgf@y}%
\pgftransformaddtransform{%\the\pgf@x,\the\pgf@y}%
\xdef\outlinerev{\the\pgf@x/\the\pgf@y,\outlinerev}
A decoration to make a fortification line

\pgfdeclaredecoration{fortification}{initial}
\begin{tikzpicture}
\begin{scope}
\state{initial}[width=4\pgflinewidth]
\pgfpathlineto{\pgfpoint{2\pgflinewidth}{0}}
\pgfpathlineto{\pgfpoint{2\pgflinewidth}{2\pgflinewidth}}
\pgfpathlineto{\pgfpoint{4\pgflinewidth}{2\pgflinewidth}}
\pgfpathlineto{\pgfpoint{4\pgflinewidth}{0}}
\end{scope}
\state{final}
\pgfpathlineto{\pgfpointdecoratedpathlast}
\end{tikzpicture}
\end{document}
\def\road{%
\hex@dbg{3}{Road}
}%
\def\railroad{%
\hex@dbg{3}{Railroad}
}%
\def\river{%
\hex@dbg{3}{River}
}%
\def\border{%
\hex@dbg{3}{Border}
}%
5.4.12 Other paths

\shiftScalePath{⟨macro⟩}{⟨relative-coordinates⟩}

where ⟨relative-coordinates⟩ is a comma separated list of relative coordinates (to the lower-left and upper-right corners) ⟨x⟩/⟨y⟩

Note, this requires that \boardXmin, \boardYmin and \boardXmax, \boardYmax is defined. This can be done using the \boardframe macro.

5.4.13 Move, attacks, retreats from hex to hex

\hex@getscale
Get current scaling factor.

\begin{verbatim}
def\hex@getscale#1{%
\begingroup
\pgfgettransformentries{% scaleA}{scaleB}{scaleC}{scaleD}{whatevs}{whatevs}%
\pgfmathsetmacro{#1}{sqrt(abs(scaleA* scaleD - scaleB* scaleC))}%
\expandafter
\endgroup
\expandafter\def\expandafter#1\expandafter{#1}%
}
\end{verbatim}

Key to get the scale

\begin{verbatim}
tikzset{%
  \hex/get scale/.code={
    \hex@getscale{\hex@scale},
  },
}
\end{verbatim}

Style for moves. Use like

\begin{verbatim}
\path[move] (coordinates);\end{verbatim}

\begin{verbatim}
tikzset{%
  % Argument is colour
  hex/move/.style={
    hex/get scale, decorate, decoration={
      markings, mark=between positions 0 and 1 step 0.75*\hex@scale*\hex@dy with {
        \node [single arrow, single arrow head extend=.1*\hex@scale*\hex@dy, fill=\hex@colour, inner sep=0.05*\hex@scale*\hex@dy, minimum width=0.02*\hex@scale*\hex@dy, minimum height=\hex@scale*\hex@dy/2, transform shape]{};
      }},
  },
}
\end{verbatim}

A short move style

\begin{verbatim}
\path[short move] (coordinates);\end{verbatim}

\begin{verbatim}
tikzset{%
  % Argument is colour
  hex/short move/.style={
    hex/get scale, decorate,
  },
}
\end{verbatim}
\path[move with start] \node [inner sep=0, circle, minimum size=\hex@scale*5mm, fill=#1,];
A move cost style

\path\[move\] ...\(coordinate\)node\[hex/move\ \text{cost}\] ...;

A short line style for retreats, advances, and so on

\path\[short\ \text{line}\] (start)--(end);
An attack indication style

\path[attack] (start)--(end);

\begin{Verbatim}
\% Argument is fill color
\hex/attack/.style={
\hex/get scale,
\inherit options/.code={\csname tikz@options\endcsname},
\inherit options,
deorate,
decoration= {
\node [regular polygon,
\fill=#1,
\draw=#1,
\regular polygon sides=3,
\inner sep=0,
\minimum size=0.75*\hexscale*\hexdy,
\rotate=-90,
\transform shape]{};
}
},
}
\end{Verbatim}

Short hands

\path[attack] (start)--(end);

\begin{Verbatim}
\% Default colour is red for attaches
\hex/attack/.default=red!70!black,
\%
\hex/retreat/.style={\hex/short line=#1},
\hex/retreat/.default=white,
\%
\hex/advance/.style={\hex/short line=#1},
\hex/advance/.default={green!70!black},
\}
\end{Verbatim}

5.4.14 Board clipping and frame

\boardframe

Define the bounding box around the board

\begin{Verbatim}
\boardframe[⟨margin⟩][⟨lower=left⟩][⟨upper-right⟩][⟨margin⟩]
\end{Verbatim}

where ⟨lower-left⟩ and ⟨upper-right⟩ specifies the lower left and upper right hexes (inclusive) of the board.

\begin{Verbatim}
\def\boardframe{
\@ifnextchar[\bo@rdframe{\bo@rdframe[0]}\%
}
\end{Verbatim}
Below is our new implementation of `boardframe`. This is split into parts.

First, a macro that will define the path around rectangular placed hexes. This takes 4 mandatory arguments: lower left column and row, and upper right column and row, in that order. It also accepts an optional argument. If this is not empty, then it is assumed to be a style to apply, and hexes will be drawn using that style. The style will be passed the hex coordinates and can react accordingly:

\begin{verbatim}
\def\boardframe{
  \@ifnextchar[\{\boardframe\}{\boardframe[\}]
\}
\def\boardframe@[\#1\#2\#3\#4\#5{\hexcoords@conv\#1]\}
\pgfmathparse{min(\#2,\hex@x)}\xdef\#2{\pgfmathresult}\n\pgfmathparse{min(\#3,\hex@y)}\xdef\#3{\pgfmathresult}\n\pgfmathparse{max(\#4,\hex@x)}\xdef\#4{\pgfmathresult}\n\pgfmathparse{max(\#5,\hex@y)}\xdef\#5{\pgfmathresult}\n\hexdbg{2}{\#1 -> ll='#2,'#3, ur='#4,'#5}}
\def\boardframe@[\#1\#2\#3\#4\#5{\hexcoords@conv\#1]\}
\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}\n\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}\n\def\ctfv{SW}\n\def\ctsv{SE}\n\def\cbfv{NE}\n\def\cbsv{NW}\n\def\rrfv{E}\n\def\rrsv{NE}\n\def\rlfv{W}\n\def\rlsv{SW}\n\ifnum\rtmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbsv\let\cbsv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlfv\let\rlfv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\ifnum\ctmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbfv\let\cbfv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlsv\let\rlsv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}\n\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}\n\edef\ctfv{SW}\n\edef\ctsv{SE}\n\edef\cbfv{NE}\n\edef\cbsv{NW}\n\edef\rrfv{E}\n\edef\rrsv{NE}\n\edef\rlfv{W}\n\edef\rlsv{SW}\n\ifnum\rtmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbfv\let\cbfv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlsv\let\rlsv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}\n\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}\n\edef\ctfv{SW}\n\edef\ctsv{SE}\n\edef\cbfv{NE}\n\edef\cbsv{NW}\n\edef\rrfv{E}\n\edef\rrsv{NE}\n\edef\rlfv{W}\n\edef\rlsv{SW}\n\ifnum\rtmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbfv\let\cbfv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlsv\let\rlsv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}\n\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}\n\edef\ctfv{SW}\n\edef\ctsv{SE}\n\edef\cbfv{NE}\n\edef\cbsv{NW}\n\edef\rrfv{E}\n\edef\rrsv{NE}\n\edef\rlfv{W}\n\edef\rlsv{SW}\n\ifnum\rtmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbfv\let\cbfv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlsv\let\rlsv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}\n\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}\n\edef\ctfv{SW}\n\edef\ctsv{SE}\n\edef\cbfv{NE}\n\edef\cbsv{NW}\n\edef\rrfv{E}\n\edef\rrsv{NE}\n\edef\rlfv{W}\n\edef\rlsv{SW}\n\ifnum\rtmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbfv\let\cbfv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlsv\let\rlsv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}\n\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}\n\edef\ctfv{SW}\n\edef\ctsv{SE}\n\edef\cbfv{NE}\n\edef\cbsv{NW}\n\edef\rrfv{E}\n\edef\rrsv{NE}\n\edef\rlfv{W}\n\edef\rlsv{SW}\n\ifnum\rtmp<0\n  \let\max@short\hex@bot@short@col\n  \let\min@short\hex@top@short@col\n  \let\swp\ctfv\let\ctfv\cbfv\let\cbfv\swp\n  \let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp\n  \let\swp\rrsv\let\rrsv\rlsv\let\rlsv\swp\n\else\n  \let\max@short\hex@top@short@col\n  \let\min@short\hex@bot@short@col\n\fi\n\hexdbg{2}{\#1 -> ll='#2,'#3, ur='#4,'#5}}
\end{verbatim}
\pgfmathparse{ifthenelse(hex@got@top@short,1,0)}\edef\tmp{\pgfmathresult}
% If top-short, set factors
\ifnum\tmp=1
  \def\mnf{-1}
  \def\mxf{-1}
  \def\mnn{}
  \def\mxn{}
% If both short, set factors
\else\ifnum\tmp=2
  \def\mnf{\rtmp}
  \def\mxf{(-\rtmp)}
% If inverse rows, set factors
\ifnum\rtmp<0
  \def\mnn{}
  \def\mxn{not}
\else
  \def\mnn{not}
  \def\mxn{}
\fi
% If none is short
\else
  \def\mnf{1}
  \def\mxf{1}
  \def\mnn{not}
  \def\mxn{not}
\fi\fi
\def\row@mn##1{% 
  \pgfmathparse{int(#3+\mnf*\hex@coords@row@fac*\min@short(##1)*
\mnn(\min@short(\hex@coords@col@off)))}
  \edef\lr{\pgfmathresult}}
\def\row@mx##1{% 
  \pgfmathparse{int(#5+\mxf*\hex@coords@row@fac*\max@short(##1)*
\mxn(\max@short(\hex@coords@col@off)))}
  \edef\ur{\pgfmathresult}}
% Below defines a path around the perimeter of the hexes.
\def\@llx{10000}
\def\@lly{10000}
\def\@urx{-10000}
\def\@ury{-10000}
% Start with an empty path
\def\p{}
\foreach \c in {#2,...,#4}{% 
  \row@mn{\c}
  \row@mx{\c}

314
\foreach \c in {#2,...,#4}{
  \row@mn{\c}
  \xdef\p{\p (hex cs:c=\c,r=\lr,v=\ctfv) -- (hex cs:c=\c,r=\lr,v=\ctsv) --}
  \bordfrmeu(c=\c,r=\lr,v=\ctfv) \@llx \@lly \@urx \@ury
  \bordfrmeu(c=\c,r=\lr,v=\ctsv) \@llx \@lly \@urx \@ury
}
\row@mn{#4}
\row@mx{#4}
\foreach \r in {\lr,...,\ur}{
  \xdef\p{\p (hex cs:c=#4,r=\r,v=\rrfv) -- (hex cs:c=#4,r=\r,v=\rrsv) --}
  \bordfrmeu(c=#4,r=\r,v=\rrfv) \@llx \@lly \@urx \@ury
  \bordfrmeu(c=#4,r=\r,v=\rrsv) \@llx \@lly \@urx \@ury
}
\row@mn{#2}
\row@mx{#2}
\foreach \r in {\ur,...,\lr}{
  \xdef\p{\p (hex cs:c=#2,r=\r,v=\rlfv) -- (hex cs:c=#2,r=\r,v=\rlsv) --}
  \bordfrmeu(c=#2,r=\r,v=\rlfv) \@llx \@lly \@urx \@ury
  \bordfrmeu(c=#2,r=\r,v=\rlsv) \@llx \@lly \@urx \@ury
}
\edef\p{\p cycle}
\hex@board@path\p
\hex@dbg{3}{Hex board path: \meaning\hex@board@path'}
\% If an optional argument was given, then use that to actually make
\hexes.
\% End path with cycle
\edef\p{\p cycle}
\% Define global path
\global\let\hex@board@path\p
\% Go across largest row (can be bottom if \rtmp<0)
\foreach \c in {#4,...,#2}{
  \row@mx{\c}
}
\% Go up (down if \rtmp<0) left side.
\foreach \c in {#2,...,#4}{
  \row@mn{\c}
}
\% Go up (down if \rtmp<0) right side
\foreach \c in {#2,...,#4}{
  \row@mn{\c}
  \xdef\p{\p (hex cs:c=\c,r=\lr,v=\ctfv) -- (hex cs:c=\c,r=\lr,v=\ctsv) --}
  \bordfrmeu(c=\c,r=\lr,v=\ctfv) \@llx \@lly \@urx \@ury
  \bordfrmeu(c=\c,r=\lr,v=\ctsv) \@llx \@lly \@urx \@ury
}
\% \message{^^JColumn: \c -> \lr,'\ur' (#3,#5)}
}
This is a no operations style used as default for the macro \texttt{boardhexes} below.

\begin{verbatim}
\tikzset{%
    /hex/board/no op/.style args={#1,#2}{}%}
\end{verbatim}

This macro will make the actual hexes using the specified, optional, style. It builds on \texttt{boardframe} above.

\begin{verbatim}
\def\boardhexes{%
    \@ifnextchar[{oardhexes}{\boardhexes[board/no op]}%
}\def\boardhexes[#1](#2)(#3){%
    \hex@coords@conv{#2}
    \edef\llc{\hex@col}
    \edef\llr{\hex@row}
    \hex@coords@conv{#3}
    \edef\urc{\hex@col}
    \edef\urr{\hex@row}
    \def\margin{#1}
    \bo@rdfr@me[#1]{\llc}{\llr}{\urc}{\urr}
}
\end{verbatim}

Creates a board frame using \texttt{boardframe}.

\begin{verbatim}
\tikzset{board frame bb/.code={
    \pgfkeys{
        /tikz/local bounding box=tmp board frame,
        /tikz/transform shape,
        /tikz/execute at end scope={%
        \hex@dbg{1}{Getting board frame BB}
        \wg@get@bb{tmp board frame}
        \global\let\llx@llx\@llx
        \global\let\lly@lly\@lly
        \global\let\urx@urx\@urx
        \global\let\ury@ury\@ury
        \hex@dbg{0}{Board bounding box (\llx,\lly)x(\urx,\ury)}%
    }}}}%}
\end{verbatim}

\begin{verbatim}
\def\bo@rdframe[#1](#2)(#3){%
    \hex@coords@conv{#2}
    \edef\llc{\hex@col}
    \edef\llr{\hex@row}
    \hex@coords@conv{#3}
    \edef\urc{\hex@col}
    \edef\urr{\hex@row}
    \def\margin{#1}
    \bo@rdfr@me[#1]{\llc}{\llr}{\urc}{\urr}
    \begin{scope}[/board frame bb]
\end{scope}%%
}\end{verbatim}

316
\begin{scope}
\expandafter\path\hex@board@path;
\end{scope}
\hex@dbg{1}{Board frame LL: -> \llx,\lly}
\pgfmathparse{\llx+ifthenelse(\llx<0,-1,1)*\margin}\edef\llx{\pgfmathresult}
\pgfmathparse{\lly+ifthenelse(\lly<0,-1,1)*\margin}\edef\lly{\pgfmathresult}
\hex@dbg{1}{Board frame UR: -> \urx,\ury}
\pgfmathparse{\urx+ifthenelse(\urx<0,-1,1)*\margin}\edef\urx{\pgfmathresult}
\pgfmathparse{\ury+ifthenelse(\ury<0,-1,1)*\margin}\edef\ury{\pgfmathresult}
\pgfmathparse{\urx-\llx}\edef\w{\pgfmathresult}
\pgfmathparse{\ury-\lly}\edef\h{\pgfmathresult}
%% Print to the log
\hex@dbg{0}{Board Frame: (\llx,\lly)x(\urx,\ury) (\llc,\llr)x(\urc,\urr)}
\draw[hex/board frame/.try](\llx,\lly) rectangle(\urx,\ury);
%% Store macros
\xdef\boardXmin{\llx}
\xdef\boardYmin{\lly}
\xdef\boardXmax{\urx}
\xdef\boardYmax{\ury}
\end{scope}
\end{tikzpicture}

\def\boardpath(#1)(#2){
\hex@coords@reset%
\tikzset{/hex/coords/.cd, #1}
\edef\llc{\hex@col}
\edef\llr{\hex@row}
\tikzset{/hex/coords/.cd, #2}
\edef\urc{\hex@col}
\edef\urr{\hex@row}
\begin{scope}[local bounding box=board frame]
\expandafter\path\hex@board@path
\end{scope}
\global\let\hexboardpath\hex@board@path
}
\def\boardclip(#1)(#2)#3{\def\boardpath(#1)(#2){
\hex@coords@reset%
\tikzset{/hex/coords/.cd, #1}
\edef\llc{\hex@col}
\edef\llr{\hex@row}
\boardframe{\llc}{\llr}{\urc}{\urr}
\begin{scope}[local bounding box=board frame]
\expandafter\path\hex@board@path
\end{scope}
\global\let\hexboardpath\hex@board@path
\}
\draw \ifx|#3|\else[preaction={#3}]\fi%
Show a debug grid. This requires \boardframe.

\debuggrid

Some dummy styles. These will be defined by the export class to facilitate getting information from the board.

\tikzset{
  zoned/.style={},
  zone scope/.style={},
  zone path/.style={}
}

5.4.15 Board splitting

\splitboard

Calculates how to split a board into sheets of paper.

\splitboard{[options]}

where options are

- \textbf{paper=⟨format⟩}: Specifies the paper format. One of \texttt{a4}, \texttt{a3}, \texttt{letter}, \texttt{tabloid}. Default is \texttt{a4}.

- \texttt{landscape}: Sets the paper format to be in landscape mode (default is portrait).

- \texttt{margin=⟨size in centimetres⟩}: Size of margins on each sheet in centimetres \textit{without} unit. That is put \texttt{0.6} for 6mm, \textit{not} \texttt{6mm}. Default is 0.6. This should be \textit{slightly} larger (by roughly 5\%) than the least margin required by the printer used. \textit{Must} be given \texttt{before} \texttt{paper} to have any effect.

- \texttt{ncol=⟨number of columns⟩}: Sets the number of columns of sheets.

- \texttt{nrow=⟨number of rows⟩}: Set the number of rows of sheets.
• **overlap** = *(size in centimetres)*: Sets the size of the overlap between sheets in centimetres *without* unit. That is put 2 for 2cm, *not* 2cm. Default is 2.

• **image** = *(file name)*: File name of the board image (a PDF). Default is `board`.

• **output** = *(file name)*: File name (without `.tex` ending) to write calculated split to.

• **standalone**: Boolean flag. If true, then output file will be a standalone document (i.e., has a `\documentclass`).

• **scale** = *(scale)*: Set scale of board.

The macro will produce a file named `\jobname_out.tex` which can be included in another document to generate the split board PDF.

To use, make, for example, the file `calcsplit.tex` with the content

```latex
\documentclass[11pt]{standalone}
\usepackage{wargame}
\usepackage{mystyle}
\begin{document}
\splitboard{paper=letter,margin=.7,ncol=2,nrow=2,overlap=1}
\end{document}
```

To calculate the split of `board.pdf` over 2 × 2 letter paper sheets, with a non-printable margin of 7mm, and an overlap between the segments of 1cm.

The final split document can then be

```latex
\documentclass[11pt]{article}
\usepackage[letterpaper,margin=7mm]{geometry}
\begin{document}
\input{calcsplit_out}
\end{document}
```

If you need to scale down the board, define the style `board scale`. E.g.,

```latex
\tikzset{board scale/.style={scale=.9}}
```

Styles used for drawing things.

```latex
\tikzset{%
\setlength{\parindent}{0pt}% Margin must be <1cm
split/paper outline/.style={
  shape=rectangle,
  draw=red!50!black,
  line width=.5mm},
split/effective outline/.style={
  shape=rectangle,
  draw=green!50!black,
  dashed,}
```
A scratch dimension used

\newdimen\split@tmp

Get upper right and lower left corners of node. Argument is node name.

\def\split@getem#1{%
  \draw (#1.north east);%
  \pgfgetlastxy{\split@ulx}{\split@uly}%
  \xdef\split@ulx{\split@ulx}%
  \xdef\split@ulx{\split@ulx}%
  \draw (#1.south west);%
  \pgfgetlastxy{\split@lrx}{\split@lry}%
  \xdef\split@lrx{\split@lrx}%
  \xdef\split@lrx{\split@lrx}%
}\def\split@getboard#1{%
  \split@getem{#1}%
  \xdef\split@bulx{\split@ulx}%
  \xdef\split@buly{\split@uly}%
  \xdef\split@blrx{\split@lrx}%
  \xdef\split@blry{\split@lry}%
  \split@w{\@percentchar\space Board:
    (\split@bulx,\split@buly)(\split@blrx,\split@blry)}}

Adjust placement of markers and cut lines.

1. Dimension to adjust
2. Overlap dimension (with units)

\def\split@adj#1#2{%
  \split@tmp=#2%
  \divide\split@tmp by 2%
  \add\split@tmp by #1%
  \edef\t{the\split@tmp}}

Get initial offset in a direction.

1. Number of segments in direction
2. Overlap in centimetres (without unit)
3. Effective size, in centimetres (without unit), of sheets in direction
4. Full size, in centimetres (without unit), of board in direction.

\def\split@get@init#1#2#3#4{%\pgfmathparse{((#1 * #3 - (#1 - 1) * #2) - #4)/2}\xdef\split@off{\pgfmathresult}\hex@dbg{2}{((#1 * #3 - (#1 - 1) * #2) - #4)/2 \rightarrow \"\split@off\"}}

Get initial offset of first segment.

1. Number of rows
2. Number of columns
3. Overlap in centimetres (without unit)
4. Effective height, in centimetres (without unit), of sheets
5. Effective width, in centimetres (without unit), of sheets
6. Full height, in centimetres (without unit), of board
7. Full width, in centimetres (without unit), of board

\def\split@getinit#1#2#3#4#5#6#7{%\split@get@init{#1}{#3}{#4}{#6}\xdef\dy{\split@off \text{cm}}\split@get@init{#2}{#3}{#5}{#7}\xdef\dx{\split@off \text{cm}}}

Get coordinates of a segment

1. Column number
2. Row number
3. Overlap, in centimetres (without unit)

\def\split@getcoords#1#2#3{%\hex@dbg{2}{Getting coords 'c#1r#2'}\split@getem{c#1r#2}\edef\sulx{\split@ulx}\edef\suly{\split@uly}\edef\slrx{\split@lrx}\edef\slry{\split@lry}\edef\mlx{\split@blrx}\edef\mrx{\split@bulx}\edef\mty{\split@buly}\edef\mby{\split@blry}\pgfmathparse{int(#1-1)}\edef\pc{\pgfmathresult}\pgfmathparse{int(#2-1)}\edef\pr{\pgfmathresult}\pgfmathparse{int(#1+1)}\edef\nc{\pgfmathresult}\pgfmathparse{int(#2+1)}\edef\nr{\pgfmathresult}\pgfutil@ifundefined{pgf@sh@ns@c\pc r#2}{}{% Left\hex@dbg{3}{\space Getting left 'c\pc r#2'}\split@getem{c\pc r#2}\split@adj{\split@ulx}{-#3}\edef\mlx{\t}}%\pgfutil@ifundefined{pgf@sh@ns@c\nc r#2}{}{% Right\hex@dbg{3}{\space Getting right 'c\nc r#2'}%\split@getem{c\nc r#2}\split@adj{\split@ulx}{#3}\edef\mlx{\t}}}%\edef\mby{\split@blry}\edef\mty{\split@buly}\edef\mlx{\split@blrx}\edef\mrx{\split@bulx}\edef\mty{\split@buly}\edef\mby{\split@blry}\edef\mlx{\split@blrx}\edef\mrx{\split@bulx}\edef\mty{\split@buly}\edef\mby{\split@blry}}
\def\split@footer{% 
\ifsplit@standalone 
\split@w{\string\end{document}}
\fi
\split@w{\string\@percentchar\@percentchar End of \string\jobname'^^J}
\immediate\closeout\split@calcout
\end{document}
Initial calculations. This draws the board and then extracts the dimensions of the board. It also defines some styles for drawing the board segments.

\begin{verbatim}
def\split@init#1{% 
ode[\split@scale,inner sep=0pt,outer sep=0pt,anchor=north west,transform shape](b){\includegraphics{#1}}; \split@getboard(b) \node[split/paper size,anchor=north west]{\includegraphics{#1}}; \node[split/effective size,anchor=north west]{\includegraphics{#1}}; \node[split/board size,anchor=north west]{\includegraphics{#1}}; }
\end{verbatim}

Calculate effective sheet sizes from sheet dimensions and the defined margin.
Options for the \splitboard macro.

newif\lifsplit@standalone\split@standalonetrue
\tikzset{
split/.search also={/tikz},
split/.cd,
margin/.store in=\split@margin,
paper/.is choice,
paper/a4/.code={\hex@dbg{3}{A4 paper for split}\
\global\paperwidth=21cm\
\global\paperheight=29.7cm\
\split@text@dim{\split@margin}},
paper/a3/.code={\hex@dbg{3}{A3 paper for split}\
\global\paperheight=42cm\
\global\paperwidth=29.7cm\
\split@text@dim{\split@margin}},
paper/letter/.code={\hex@dbg{3}{Letter paper for split}\
\paperheight=27.9cm,\
\paperwidth=21.6cm,\
\split@text@dim{\split@margin}},
paper/tabloid/.code={\hex@dbg{3}{Tabloid paper for split}\
\paperheight=43.2cm,\
\paperwidth=27.9cm,\
\split@text@dim{\split@margin}},
landscape/.code={\hex@dbg{3}{Landscape option for split}\
\split@tmp=\paperheight\
\global\paperheight=\split@tmp\
\global\paperwidth=\split@tmp\
\split@tmp=\textheight\
\global\textheight=\split@tmp\
\global\textwidth=\split@tmp},
standalone/.is if=split@standalone,
scale/.store in=\split@scale,
output/.store in=\split@out,
ncol/.store in=\split@ncol,
nrow/.store in=\split@nrow,
overlap/.store in=\split@ov, % Centimeter, no unit
image/.store in=\split@img,
The actual macro. The argument is key-value pairs of options.
\begin{tikzpicture}
\node[split/effective size, above left=\dy and \dx of b.north west,anchor=north west] (c1r1) {};
\node[split/paper size] at (c1r1) {};
\end{tikzpicture}
\foreach \r [remember=\r as \pr (initially 0)] in {1,...,\split@nrow}{%
Macro used by the written file.

1. first coordinate (e.g., (hex ak:c=C, r=17))
2. second coordinate (e.g., (hex ak:c=M, r=33))
3. Crop mark left
4. Crop mark right
5. Crop mark bottom
6. Crop mark top
5.5  The wargame.chit Ti\kZ library

We define the library for making chits. We load the hex Ti\kZ wargame.natoapp6c library and the amsmath and amstext packages as we need those.

\begin{verbatim}
\RequirePackage{amsmath}
\RequirePackage{amstext}
\usetikzlibrary{wargame.util,wargame.natoapp6c,math}
\end{verbatim}

5.5.1 Debugging

\begin{verbatim}
\chitdbglvl
\chit@dbg
\end{verbatim}

Some macros for debugging. Similar to what we have in wargame.hex (see Section 5.4.

\begin{verbatim}
\newcount\chitdbglvl\chitdbglvl=\wargamedbglvl
\def\chit@dbg#1#2{\ifnum#1>\chitdbglvl\relax\else\message{^^J#2}\fi}
\end{verbatim}

5.5.2 The chit key namespace

Some stuff to consider wrt. line widths. Setting the line width in the chit scope overrides frame settings. The frame stroke can be larger but not smaller. Setting the stroke width in the symbol scope sets it for the symbol only. Thus, to get a thin border, we need to

- Set a small line width in the top chit scope.
- Possible set a larger line width in the frame sub-scope.
- Set a larger line width in the symbol sub-scope.

I do not know why this is.

\begin{verbatim}
/chit/full
/chit/symbol
/chit/left
/chit/right
/chit/upper left
/chit/upper right
/chit/lower left
/chit/lower right
/chit/factors
/chit/extra
/chit/setup
/chit/bevel
/chit/id
\end{verbatim}

The parts of a chit
\newif\ifchit@clip\chit@cliptrue
\tikzset{%
\chit/.search also={/tikz},
\chit/.cd,
full/.store in=\chit@full, full/.initial=,%
symbol/.store in=\chit@symbol, symbol/.initial=,%
left/.store in=\chit@left, left/.initial=,%
unique/.style={/chit/left={#1}},%
right/.store in=\chit@right, right/.initial=,%
parent/.style={/chit/right={#1}},%
upper left/.store in=\chit@upper@left, upper left/.initial=,%
upper right/.store in=\chit@upper@right, upper right/.initial=,%
lower left/.store in=\chit@lower@left, lower left/.initial=,%
lower right/.store in=\chit@lower@right, lower right/.initial=,%
factors/.store in=\chit@factors, factors/.initial=,%
setup/.store in=\chit@setup, setup/.initial=,%
id/.store in=\chit@id, id/.initial=,%
frame/.store in=\chit@frame, frame/.initial=,%
extr@/.store in=\chit@extr@, extr@/.initial=,%
bev/.store in=\chit@bevel, bev/.initial=,%
bevel fraction/.store in=\chit@bevel@frac, bevel fraction/.initial=10,
bevel/.is choice,
bevel/none/.style = {/chit/bev=},
bevel/north west/.style = {/chit/bev=1},
bevel/north east/.style = {/chit/bev=2},
bevel/south west/.style = {/chit/bev=3},
bevel/south east/.style = {/chit/bev=4},
bevel/NW/.style = {/chit/bev=1},
bevel/NE/.style = {/chit/bev=2},
bevel/SW/.style = {/chit/bev=3},
bevel/SE/.style = {/chit/bev=4},
bevel/.default = north west,
clip/.is if=\chit@clip%
}

/tikz/chit/full
/tikz/chit/symbol
/tikz/chit/left
/tikz/chit/right
/tikz/chit/upper left
/tikz/chit/upper right
/tikz/chit/lower left
/tikz/chit/lower right
/tikz/chit/factors
/tikz/chit/setup
/tikz/chit/id

Styles of each element in a chit. Users may override these at their own peril. That is, it is OK to override them, but the user should be careful.

\tikzset{
\chit/symbol/.style={scale=.4,transform shape},
5.5.3 The chit styles

This key sets up a node to make a chit. The key takes a single argument which in turn must contain key–value pairs in the /chit (or /tikz) namespace(s). We set the shape parameter of the node, and calls the passed keys in the /chit namespace to set-up elements of the chit.

We define a counter to set-up unique names for chit nodes.
5.5.4 The \chitshape

These macros puts the NATO App6(c) symbol into a chit. The first macro takes the identifier and position of the symbol, and then scans for options. If no options are given, then we go directly to the rendering (\@chitnato). Otherwise, we may also need to scan for an offset given as \langle\delta-x,\delta-y\rangle.

14439 \def\chitnato#1#2{%  
14440 \chitnatodebug{1}{Chit NATO App6(c) first step '#1' '#2'}%  
14441 \ifnextchar{[}{
14442 \chitnato{#1}{#2}}{%
14443 \chitnatodebug{1}{Chit NATO App6(c) w/o offset:}
14444 \chitnato{#1}{#2}{
14445 \chitnatodebug{4}{Chit NATO App6(c) ended}}%
14446 }

This called if we had no options.

14448 \def\@chitnato#1#2#3\@endchitnato{%  
14449 \chitnatodebug{1}{Chit NATO App6(c) w/o offset:}
14450 \chitnatodebug{1}{Chit NATO App6(c) second step '#1' '#2' '#3'}%
14451 \ifnextchar{[}{
14452 \@chitnato{#1}{#2}{#3}{0,0}}%
14453 }

This called if we had an option-like argument. Check if we have an offset.

14456 \def\@@chitnato#1#2#3(#4)\@endchitnato{%  
14457 \chitnatodebug{1}{Chit NATO App6(c) w/offset:}
14458 \chitnatodebug{1}{Chit NATO App6(c) w/offset:}
14459 }

This called if we had option-like argument.
Get anchor of sub-symbol element in chit. We need to do this, because the symbol is translated and scaled.

14467 \def\chit@anchor#1{%
14468 \pgf@x=0.4\pgf@x%
14469 \pgf@y=0.4\pgf@y\advance\pgf@y#1}

14470 \def\chit@anchor#1#2#3{%
14471 \wd@subanchor[#1]{#2}
14472 \chit@anchor[#3]{}
14473 \def\chit@report{}
14474 \tikzset{
14475 zone turn/.style={},
14476 zone mult/.style={}
14477 }

Now follows the actual chit shape. This is rather long, so we will break it up a bit

14478 \def\chit@bevel@path#1{
14479 \scope[#1]
14480 \wd@tmpc=\wd@tmpa\multiply\wd@tmpc by \chit@bevel@frac
14481 \wd@tmpd=\wd@tmpb\multiply\wd@tmpd by \chit@bevel@frac
14482 \divide\wd@tmpc100
14483 \divide\wd@tmpd100
14484 \pgfpathmoveto(\pgfpoint{\wd@tmpa}{\wd@tmpb})%
14485 \pgfpathlineto(\pgfpoint{\wd@tmpa}{\wd@tmpb})%
14486 \pgfpathlineto(\pgfpoint{\wd@tmpa}{\wd@tmpb})%
14487 \pgfpathlineto(\pgfpoint{\wd@tmpa}{\wd@tmpb})%
14488 \pgfpathlineto(\pgfpoint{\wd@tmpa}{\wd@tmpb})%
14489 \pgfclosepath%
14490 \pgfusepath{fill}
14491 \endscope
14492 }

The first thing is we declare some saved anchors. These are computed (and defined as internal macros) when the shape is instantised. The anchors give the centre and north east corner of the node, the place to put the NATO App6(c) symbol and factors. We also set a dimension for the margins (corner and factors elements).

14498 \pgfdeclareshape{chit}[]{
14499 savedanchor{center}{\pgf@x=0cm\pgf@y=0cm}
14500 savedanchor{northeast}{\pgf@x=0.6cm\pgf@y=0.6cm}
Next, we define some saved macros. These are called (and declares internal macros) when the shape is instantised. We define macros for the identifier,

\begin{verbatim}
\savedmacro{id}{
  \chit@dbg{4}{Chit ID: \meaning\chit@id}%
  \@ifundefined{chit@id}{}{\let\chit@id\pgfutil@empty}%
  \chit@dbg{3}{Chit ID stored: \meaning\chit@id}%
}
\end{verbatim}

We define the regular anchors of the shape. That is, the centre, corners, and edges.

\begin{verbatim}
\anchor{center}{\center}
\anchor{north east}{\northeast}
\anchor{north west}{\northeast\pgf@x=-\pgf@x}
\anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{south east}{\northeast\pgf@y=-\pgf@y}
\anchor{north}{\northeast\pgf@x=0cm}
\anchor{south}{\northeast\pgf@x=0cm\pgf@y=-\pgf@y}
\anchor{east}{\northeast\pgf@y=0cm}
\anchor{west}{\northeast\pgf@x=-\pgf@x\pgf@y=0cm}
\anchor{symbol north east}{\chit@nchor{M\id symbol}{north east}{0.2cm}}
\anchor{symbol north west}{\chit@nchor{M\id symbol}{north west}{0.2cm}}
\anchor{symbol south east}{\chit@nchor{M\id symbol}{south east}{0.2cm}}
\anchor{symbol south west}{\chit@nchor{M\id symbol}{south west}{0.2cm}}
\anchor{symbol north}{\chit@nchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol west}{\chit@nchor{M\id symbol}{west}{0.2cm}}
\anchor{symbol south}{\chit@nchor{M\id symbol}{south}{0.2cm}}
\anchor{symbol east}{\chit@nchor{M\id symbol}{east}{0.2cm}}
\anchor{symbol upper}{\chit@nchor{M\id symbol}{upper}{0.2cm}}
\anchor{symbol lower}{\chit@nchor{M\id symbol}{lower}{0.2cm}}
\anchor{symbol left}{\chit@nchor{M\id symbol}{left}{0.2cm}}
\anchor{symbol right}{\chit@nchor{M\id symbol}{right}{0.2cm}}
\anchor{symbol echelon}{\chit@nchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol below}{\chit@nchor{M\id symbol}{south}{0.1cm}}
\end{verbatim}

Next, we want to be able to reference the symbol anchors too. So we define these anchors from the embedded node anchors. Note, these anchors will not exist if the chit is made with \texttt{full=⟨args⟩}.

\begin{verbatim}
\anchor{symbol north east}{\chit@nchor{M\id symbol}{north east}{0.2cm}}
\anchor{symbol north west}{\chit@nchor{M\id symbol}{north west}{0.2cm}}
\anchor{symbol south east}{\chit@nchor{M\id symbol}{south east}{0.2cm}}
\anchor{symbol south west}{\chit@nchor{M\id symbol}{south west}{0.2cm}}
\anchor{symbol north}{\chit@nchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol west}{\chit@nchor{M\id symbol}{west}{0.2cm}}
\anchor{symbol south}{\chit@nchor{M\id symbol}{south}{0.2cm}}
\anchor{symbol east}{\chit@nchor{M\id symbol}{east}{0.2cm}}
\anchor{symbol upper}{\chit@nchor{M\id symbol}{upper}{0.2cm}}
\anchor{symbol lower}{\chit@nchor{M\id symbol}{lower}{0.2cm}}
\anchor{symbol left}{\chit@nchor{M\id symbol}{left}{0.2cm}}
\anchor{symbol right}{\chit@nchor{M\id symbol}{right}{0.2cm}}
\anchor{symbol echelon}{\chit@nchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol below}{\chit@nchor{M\id symbol}{south}{0.1cm}}
\end{verbatim}

332
Some anchors to sub-elements. Some of them only exists if we have NATO App6(c) symbol in the chit.

\anchor{symbol} \{\symbol\}
\anchor{factors} \{\factors\}
\anchor{left} {\chit@anchor{\id symbol}{west}{.2cm}\advance\pgf@x-\margin}
\anchor{right} {\chit@anchor{\id symbol}{east}{.2cm}\advance\pgf@x+\margin}
\anchor{upper right} {\northeast \advance\pgf@x-\margin \advance\pgf@y-\margin\%
\anchor{upper left} {\northeast \advance\pgf@x-\margin \advance\pgf@y-\margin \pgf@x=-\pgf@x\%
\anchor{lower right} {\northeast \advance\pgf@x-\margin \advance\pgf@y-\margin \pgf@y=-\pgf@y\%
\anchor{lower left} {\northeast \advance\pgf@x-\margin \advance\pgf@y-\margin \pgf@x=-\pgf@x \pgf@y=-\pgf@y\%

Now for the actual path. For the background path, we simply specify the frame. This is so that this will get drawn (and possibly filled) using the appropriate options.

\backgroundpath{%
%% This is the outline of the chit only. The rest of the chit is
%% made on the foreground "path".
\chit@dbg{1}{Chit drawing background path}
\northeast\%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfclosepath
}

Finally, we make the foreground rendered path. This is where we do the most stuff. We do it in the behind foreground path so that we can ensure things are drawn the way we want it.

The first thing is to set-up the clipping to the chit frame.

\behindforegroundpath{%
\chit@dbg{1}{Chit drawing foreground path}
% \chit@dbg{4}{%
% Chit foreground: \meaning\id
% ^^J ID (set): \meaning\chit@id
% ^^J Symbol: \meaning\chit@symbol
% ^^J Full: \meaning\chit@full
% ^^J Factors: \meaning\chit@factors
% ^^J Left: \meaning\chit@left
% ^^J Right: \meaning\chit@right
% ^^J Upper left: \meaning\chit@upper@left
% ^^J Lower left: \meaning\chit@lower@left
% ^^J Upper right: \meaning\chit@upper@right
% ^^J Lower right: \meaning\chit@lower@right
333
If we do not have the symbol key set, then we set the full key as a picture.

Otherwise, we put in a node with shape natoapp6c and pass the symbol key–value pairs as options.

Having made the NATO App6(c) symbol, which we gave the node name ⟨id⟩symbol where ⟨id⟩ is the ID of this chit, we can make the rest of the chit elements. These are the left and right elements, which are set west and east of the symbol, respectively; the factors; and the four corner elements.

If the respective elements have not been specified, we do not make them.

First the left and right elements. Note that these uses the anchors of the embedded natoapp6c node for placement.
Next, we want to put in the corner elements. But before we do that, we use our saved anchors and dimensions to calculate the coordinates. Note that the corner elements are anchored to the corners (plus margin) of the chit frame.

With the coordinates extracted, we set the four corner elements. Note, for the anchoring to work, we should specify pictures that have anchors (e.g., nodes). If not, we must take care to give offsets or the like.
Finally, we put in the unit factors. They are put at the bottom of the chit frame (plus margin) and are typically anchored to the south anchor of the element. Note, we can put in several factors if need be.

\begin{scope}\[
\begin{pgfinterruptscope}
\pgfinterruptpicture
\chitageneric{\meaning\chit@factors}{0,0,\the\wg@tmpb}{chit/factors}%
\end{scope}\]
\end{pgfinterruptscope}
\begin{scope}\[
\begin{pgfinterruptscope}
\pgfinterruptpicture
\chitageneric{\meaning\chit@extra}{0,0,\the\wg@tmpb}{chit/factors}%
\end{scope}\]
\end{pgfinterruptscope}
% End of full or symbol
\end{pgfinterruptscope}
\% Make bevel?
\chitageneric{\meaning\chit@bevel}{1,0,0}{chit/bevel highlight}
\chitageneric{\meaning\chit@bevel}{1,0,0}{chit/bevel shadow}
% Draw frame?
\chitageneric{\meaning\chitframeopt}{1,0,0}{chit/frame style/.cd, chit/frame/.try, \chitframeopt}
\edef\tmp@opt{[chit/frame style/.cd,chit/frame/.try,\chitframeopt]}
\expandafter\scope\tmp@opt
\northeast\[
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\fi
\end{scope}
\% End of full or symbol
\end{pgfinterruptscope}
\% Put in factors
\ifarundefined{chit@factors}{}{\%
\chitageneric{\meaning\chit@factors}{0,0,\the\wg@tmpb}{chit/factors}%\%
\fi
\% Put in extra
\ifarundefined{chit@extra}{}{\%
\chitageneric{\meaning\chit@extra}{0,0,\the\wg@tmpb}{chit/factors}%\%
\fi
\% Put in full or symbol
\end{pgfinterruptscope}
\% Make bevel?
\ifarundefined{chit@bevel}{}{\%
\chitageneric{\meaning\chit@bevel}{1,0,0}{chit/bevel empty}\%
\fi
\% Make bevel?
\northeast\[
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\fi
\end{scope}
\% Draw frame?
\chitageneric{\meaning\chitframeopt}{1,0,0}{chit/frame style/.cd, chit/frame/.try, \chitframeopt}
\edef\tmp@opt{[chit/frame style/.cd,chit/frame/.try,\chitframeopt]}
\expandafter\scope\tmp@opt
\northeast\[
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\[
\fi
\end{scope}
\% End of full or symbol
\end{pgfinterruptscope
5.5.5 The \chit wrapper macro

\chit
\chit@0
\chit@@

The macro to make the chits. This is a wrapper around a node with shape chit. The syntax of this macro is

```
\chit[⟨chit options⟩](⟨position⟩)(⟨identifier⟩);
```

Note that the trailing semi-colon is optional. Here (chit options) are any key-value pairs in the /chit (and /tikz) namespace.

The first macro parses for options.

\begin{verbatim}
\def\chit{%
  \chit@dbg{5}{Chit}
  \@ifnextchar[\chit@}{\chit@[
}
\end{verbatim}

```
Parse for coordinates.

\begin{verbatim}
\def\chit@[#1]{%
  \chit@dbg{5}{Chit second: '#1'}
  \@ifnextchar({\chit@@{#1}}{\chit@@{#1}(0,0)}%)
\end{verbatim}

```

Parse for name.

\begin{verbatim}
\def\chit@@#1(#2){%
  \@ifnextchar({\chit@@@{#1}{#2}}{\chit@@@{#1}{#2}()}%)
\end{verbatim}

```

The work horse. This simply makes a node with the shape chit. Note, we allow for a trailing semi-colon (;) to have a similar feel to other Ti\k Z macros.

The macro will execute the style /tikz/every chit if defined. Note that this will be executed before the usual every chip node style.

\begin{verbatim}
\def\chit@@@#1#2(#3){%
  \chit@dbg{5}{Chit final: "J Options: #1
  \@ifnextchar{\chit@0#2(#3){% #2
  \@ifnextchar{\chit@0#2(#3){% #2
  \chit@0#2(#3){% #2
\end{verbatim}

337
5.5.6 Predefined chit element pictures

\DeclareRobustCommand\chit@sep[2]{\%
\foreach[count=\is] \s in {#2}{\%
\ifnum\is>1\relax#1\fi\%
\s}}
/tikz/pics/chit/1 factor
/tikz/pics/chit/2 factors
/tikz/pics/chit/2 factors artillery
/tikz/pics/chit/3 factors
/tikz/pics/chit/4 factors
/tikz/pics/chit/identifier
/tikz/pics/chit/small identifier
/tikz/pics/chit/identifier macro

These pictures can be used as the value of chit keys.
\node[chit/identifier,pic actions]{$\text{#1}$};

\node[chit/identifier,pic actions]{\chit@sep{$\text{#1}$}};

\chit@dbg{4}{ Chit identifiers: '###'}
\node[chit/identifier,pic actions]{$\text{#1}$};

\chit@dbg{4}{ Chit small identifier: '###'}
\node[chit/small identifier,pic actions]{$\text{#1}$};

\chit@dbg{4}{ Chit small identifiers: '###'}
\node[chit/small identifier,pic actions]{$\text{#1}$};

\chit@dbg{4}{ Chit identifier macro: \meaning\text{#1}}
\edef\chit@i@tmp{$\text{#1}$}
\edef\chit@i@tmp{$\text{#1}$}
\node[chit/identifier,pic actions]{$\text{#1}$};

Styles used by the above pictures. Users can change these as they see fit.

\tikzset{
chit/factor/.style={
shape=rectangle,
font=\sffamily\bfseries\fontsize{12}{14}\selectfont,
anchor=base,
inner sep=0,
% text=pgfstrokecolor,
draw=none,
fill=none,
transform shape,
},
chit/1 factor/.style={},
chit/2 factors/.style={},
chit/3 factors/.style={},
chit/4 factors/.style={text/.append style=\fontsize{10}{12}\selectfont},
chit/identifier/.style={
shape=rectangle,
font=\sffamily\bfseries\fontsize{8}{9}\selectfont,
inner sep=0,
% text=pgfstrokecolor,
draw=none,
fill=none,
transform shape,
}
5.5.7 Modifications to chits

These defines overlays one can add on top of chits, for example to shade a chit, put a semi-transparent red cover to indicate elimination, and similar.

```
\tikzset{
  pics/chit/shade/.style={
    code={%
      \path[fill=white,opacity=#1,pic actions] (-.6,-.6) rectangle(.6,.6);}},
  pics/chit/eliminate/.style={
    code={%
      \path[fill=red,opacity=#1,pic actions] (-.6,-.6) rectangle(.6,.6);}},
  pics/chit/shade/.default=0.5,
  pics/chit/eliminate/.default=0.25,
  dummy chit/.style={draw=none,fill=none,chit={}},
}
\def\shadechit{%
  \@ifnextchar[{{\sh@dechit[]}\]}
}\def\eliminatechit{%
  \@ifnextchar[{{\elimin@techit[]}\]}
}\def\sh@dechit[#1](#2){%
  % \message{^^JShading chit with opacity '#1'}%
  \pic[transform shape] at (#2) {chit/shade=#1};%
  \@ifnextchar;{\@gobble}{}%
}\def\elimin@techit[#1](#2){%
  \pic[transform shape] at (#2) {chit/eliminate=#1};%
  \@ifnextchar;{\@gobble}{}%
}
```

5.5.8 Stacking of chits

Stacking of chits. The key chit/stack direction sets the default direction to make the stack in.

```
\tikzset{
  chit/stack direction/.store in=\chit@stack@dir,
  chit/stack direction/.initial={(.3,.3)},
}
```

Now the code
5.5.9 Making order of battle charts

Macros for making OOBs

Style for turns

```
\def\chit@oob@cellupdate(#1,#2)#3#4#5{\edef\f{\ifwg@oob@inv-1\else1\fi}\chit@dbg{1}{Cell update 'c=|#1|' vs '#3'*('#3-1')}\pgfmathparse{int(ifthenelse(abs(#1)>=#4*(#3-1),#5-1,#5))}\xdef#5{\pgfmathresult}\pgfmathparse{ifthenelse(abs(#1)>=#4*(#3-1),#2-#4,#2)}\xdef#2{\pgfmathresult}\pgfmathparse{ifthenelse(abs(#1)>=#4*(#3-1),0,#1+\f*#4)}\xdef#1{\pgfmathresult}\chit@dbg{1}{-> '#5'=#5 '#2'=#2 '#1'=#1}}
```

current c, current r, cell size, extra vertical spacing

```
\def\chit@oob@cellupdate(#1,#2)#3#4#5{\edef\f{\ifwg@oob@inv-1\else1\fi}\chit@dbg{1}{Cell update 'c=|#1|' vs '#3'*('#3-1')}\pgfmathparse{int(ifthenelse(abs(#1)>=#4*(#3-1),#5-1,#5))}\xdef#5{\pgfmathresult}\pgfmathparse{ifthenelse(abs(#1)>=#4*(#3-1),#2-#4,#2)}\xdef#2{\pgfmathresult}\pgfmathparse{ifthenelse(abs(#1)>=#4*(#3-1),0,#1+\f*#4)}\xdef#1{\pgfmathresult}\chit@dbg{1}{-> '#5'=#5 '#2'=#2 '#1'=#1}}
```

current c, current r, cell size, extra vertical spacing
\def\chit@oob@rowupdate(#1,#2)#3#4{% 
  \chit@dbg{2}{ Row update c='#1',r='#2',s='#3',e='#4'}
  \pgfmathparse{ifthenelse(#1>0,#2-#3,#2)}%
  \pgfmathparse{#2-#3-#4}%
  \xdef#2{\pgfmathresult}%
  \xdef#1{0}%
  \chit@dbg{2}{ \space\space-> update '\string#2'=#2} 
}

current c, current r, cell size, extra spacing

\def\chit@oob@turnupdate(#1,#2)#3#4{% 
  \chit@dbg{2}{ Turn update c='#1',r='#2',s='#3',e='#4'}
  \pgfmathparse{#2-ifthenelse(#1>0,#3,0)-#4}%
  \pgfmathparse{#2-#4-ifthenelse(abs(#1)>0.0001,#3,0)}
  \xdef#2{\pgfmathresult}%
  \xdef#1{0}%
  \chit@dbg{2}{ \space\space-> update '\string#1'=#1,'\string#2'=#2} 
}

chit list, n-colls, cell size, extra vertical spacing
This expects a list of lists of chits, one list per turn; the maximum number of columns; the size of cells, extra spacing between turns.

Note, the list of lists leaf elements should be styles for the chits.
This depends on the Tikz pic chit/oob turn which takes the number as argument.

\newif\ifwg@oob@inv\wg@oob@invfalse
\def\chit@oob@spacer{hspace}
\def\chit@oob@vspacer{vspace}
\def\wg@star@oob{\wg@oob@invtrue\wg@oob}
\def\wg@nostar@oob{\wg@oob@invfalse\wg@oob}
\def\oob{%
  \@ifstar{\wg@star@oob}{\wg@nostar@oob}%
}

The inner macro of $\oob$. The arguments are

1. The list of lists of chits styles
2. The maximum number of columns
3. The width of each cell
4. Additional row spacing between turns

\def\wg@oob#1#2#3#4{ 
  \def\r{0} 
  \pgfmathparse{#3*(#2-1)}%
  \edef\a{\pgfmathresult} 
  \edef\aa{\pgfmathresult} 
  \chit@dbg{2}{OOB: '\string#1'} 
}
\foreach[count=\ti from 0] {t/y in #1{\t/\y in #1{\n}
\\xdef\o{\r}
\\xdef\c{0}
\\ifx\t\y\def\y{0}\fi
\\chit@dbg{2}{Turn \ti\space(t, \t, y=\y):'}
\\ifwg@oob@inv%
\\pic[transform shape] at ( .5*#3, \r) {chit/oob turn=\ti};% was dx=0.5
\\else
\\pic[transform shape] at (-.5*#3, \r) {chit/oob turn=\ti};% was dx=-0.5
\\fi%
\\ifx\t\empty\else%
\\foreach \u/\m in \t{
\\\% \chit@dbg{2}{\u='\m'}
\\\xdef\m{0}
\\\ifx\u\empty\else
\\\ifx\m\@empty\def\m{1}\fi
\\\foreach \n in {1,...,\m}{%
\\\chit@dbg{2}{OOB Chit is \u='chit@oob@spacer'}%
\\\ifx\u\chit@oob@spacer%
\\\chit@dbg{3}{Chit \u is spacer 'chit@oob@spacer'}%
\\\pgfmathparse{\c+1}\xdef\c{\pgfmathresult}%
\\\else%
\\\ifx\u\chit@oob@vspacer%
\\\chit@dbg{3}{Chit \u is vspacer 'chit@oob@vspacer'}%
\\\pgfmathparse{\c+1}\xdef\c{\pgfmathresult}%
\\\else%
\\\ifnum\chitdbglvl>2%
\\\node[minimum width=\c cm, minimum height=\c cm, draw, transform shape] at (\c, \r) {};
\\\fi
\\\ifx\u\chit@blank\else%
\\\chit[\u=\ti, zone oob point={\u}{\c}{\r}](\c, \r);%
\\\fi%
\\\chit@oob@cellupdate(\c, \r){#2}{#3}{\y}
\\\fi%
\\fi}
\\chit@dbg{2}{End of chits in turn}
\\chit@oob@turnupdate increments the row
\\ifx\t\empty
\\\xdef\c{#3}
\\\ifnum\y<0%
\\\chit@dbg{2}{Turn is empty, set c='\c'}
\\\fi
\\\%\ifnum\y<0 No explicit number of rows given

343
% \def\c(#3)
% \chit@dbg{2}{ No explicit number of rows given, set c='\c'}
%\fi
% In case the user gave an explicit number of rows, add the rows
% that are missing. \y is initially set to the number of
% requested rows, and then decremented every time we go down one
% row. So if the number of rows we did so far is N, and the
% requested number of rows is M, then the loop below adds M-N
% rows.
%\ifnum\y>0%
% \chit@dbg{2}{ Looping rows from 2 to \y, break when row > \y}%
% \foreach \rr in {2,...,\y}{
% \ifnum\rr>\y% A little funny, but \y can be negative!
% \chit@dbg{2}{ \space Breaking loop \rr\space > \y}%
% \else%
% \chit@oob@rowupdate(\c,\r){#3}{0}% Extra spacing?
% \fi
% \fi
% This will zero \c. However, if on entry |\c|>0, then we also
% increment the row
% \chit@oob@turnupdate(\c,\r){#3}{#4}
%\fi
%\chit@dbg{3}{End of OOB (c='\c',r='\r',y='\y')}% Horizontal flow OOB
\def\wg@star@hoob{\wg@oob@invtrue\wg@hoob}
\def\wg@nostar@hoob{\wg@oob@invfalse\wg@hoob}
\def\hoob{\ifstar{\wg@star@hoob}{\wg@nostar@hoob}{\wg@hoob}}
\def\wg@hoob#1#2#3#4{
\def\r{0}
\def\c{0}
\pgfmathparse{#3*(#2-1)}\edef\a{\pgfmathresult}
\chit@dbg{2}{OOB: '{#1}'}
\foreach[\ti/\y in #1] {count=\ti from 0} \t/\y in #1{

The inner macro of \hoob. The arguments are

1. The list of lists of chits styles
2. The maximum number of columns
3. The width of each cell
4. Additional row spacing between turns
\def\o{\r}
% \def\c{0}
\ifx\t\y\def\y{0}\fi
\chit@dbg{2}{Turn \ti\space(\r,\t,y=\y):'}
\ifx\t\empty\else
% Count how many are left for this turn
\chit@dbg{2}{At start of turn \t\space string\c=c\}
\def\l{\c}%
\let\ig\empty
\foreach \u/\m in \t{
\ifx\ig\empty
\ifx\u\empty\else
\ifx\u\m\def\m{1}\fi
\chit@dbg{2}{\chit@oob@spacer}%
\pgfmathparse{\l+\m*#4}\xdef\l{\pgfmathresult}
\chit@dbg{2}{Got \m\space hspace (#4) -> \l}
\else
\def\ig{1}
\chit@dbg{2}{Got vspace -> \l (\ig)}
\else
\pgfmathparse{\l+\m*#3}
\xdef\l{\pgfmathresult}
\chit@dbg{2}{Got \m\space units -> \l}
\fi
\fi
\fi
% Check if there's enough room
\chit@dbg{2}{To fill the rest of turn needs '\l' compared to
'\a' (#3*(#2-1))}
\pgfmathparse{ifthenelse(abs(\l)>=#3*(#2-1),0,1)}%
\xdef\l{\pgfmathresult}
\chit@dbg{2}{Break or not '\l'}
\ifnum\l=0\chit@oob@turnupdate(\c,\r){#3}{#4}\fi
\fi
\ifwg@oob@inv%
\pic[transform shape] at (\c+.5*#3,\r) {chit/oob turn=\ti};% was dx=0.5
\else
\pic[transform shape] at (\c-.5*#3,\r) {chit/oob turn=\ti};% was dx=-0.5
\fi
%\chit@oob@cellupdate(\c,\r){#2}{#3}{\y}
\ifx\t\empty\else%
\def\l{0}
\foreach \u/\m in \t{
\ifx\u\empty\else
% \chit@dbg{2}{ 'u'='m'}
\fi
\chit@dbg{2}{008 Chit is 'u' 'chit@oob@spacer'}}
\fi
\chit@dbg{2}{008 Chit is 'u' is spacer 'chit@oob@spacer'}

345
\pgfmathparse{\c+#4}\%
\\def\c{\pgfmathresult}\%
\else\%
  \ifx\u\chit@oob@vspacer\%
    \chit@dbg{3}{Chit \textquoteleft\u\textquoteleft is vspacer \textquoteleft\chit@oob@vspacer\textquoteleft\} \%
    \pgfmathparse{ifthenelse(abs(\c)<0.0001,0,#3)} \%
    \def\ll{\pgfmathresult} \%
    \chit@dbg{2}{\string\ll='\ll'} \%
    \chit@oob@rowupdate(\c,\r){\ll}{#4} \%
    \def\lv{1} \%
  \else\%
    \ifnum\chitdbglvl>2\%
      \node[minimum width=#3cm,minimum height=#3cm, draw,transform shape] at (\c,\r) {}; \%
    \fi\%
    \ifx\u\chit@blank\else\%
      \chit[\u=\ti,zone oob point={\u}{\c}{\r}]({\c},({\c}\r}); \%
    \fi\%
    \chit@oob@cellupdate(\c,\r){#2}{#3}{\y} \%
  \fi\%
\fi\%
\fi\%
\chit@dbg{2}{ End of chits in turn \ti\space(c='\c',r='\r',o='\o',y='\y')\} \%
\% \dashline{Not relevant, I think\}%
\% IF no units where given, then we force \c to be non-zero so that\%
\% \chit@oob@turnupdate increments the row\%
\% \ifx\t\@empty \%
\% \def\c{#3} \%
\% \chit@oob@turnupdate increments the row \%
\% \fi\%
\% \ifnum\y<0\%
\% No explicit number of rows given \%
\% \def\c(#3) \%
\% \chit@oob@turnupdate increments the row \%
\% \fi\%
\% ---\%
\%\ifnum\y>0\%
\% No explicit number of rows given, set c='\c'\%
\% \if\%
\% \fi\%
\% ---\%
\%\ifnum\y>0\%
\% Looping rows from 2 to \y, break when row > \y\%
\%\foreach \rr in {2,...,\y}\{ \%
  \%\ifnum\rr<\y\%
  \% A little funny, but \y can be negative! \%
  \% \chit@oob@turnupdate(\c,\r){#3}{0} % Extra spacing?

346
\% This will zero \c. However, if on entry |\c|>0, then we also
\% increment the row
\% \chit@oob@turnupdate(c,r){#3}{#4}
\%
\% --- Not relevant I think
\% Horizontal spacer
\%
\pgfmathparse{ifthenelse(abs(\c)\geq\a,1,0)}\xdef\l{\pgfmathresult}
\pgfmathparse{\c+1.5*#4}\
\xdef\c{\pgfmathresult}
\ifnum\lv=1%
\chit@oob@rowupdate(c,r){0}{#4}
\else
\chit@oob@cellupdate(c,r){#2}{#3}{\y}
\ifnum\y<0
\chit@oob@turnupdate(c,r){#3}{#4}
\else
\fi
\fi
\xdef\y{0}
\chit@dbg{2}{End of turn \ti\space(c='\c',r='\r',o='\o',y='\y')}
\chit@dbg{3}{End of OOB (c='\c',r='\r',y='\y')}
\@ifnextchar;{\@gobble}{}\}

5.5.10 Table of chits
\tikzset{
  chit/cell background/.style={fill=black},
  %chit/cell background flipped/.style={fill=black},
  blank chit/.style={[chit/frame={draw=none,fill=none}],
  chit/grid lines/.style={dashed},
}
These macros are used when we set tables of chits. This allows us to define blank spaces in the table by giving the
element blank chit.
def\chit@blank{blank chit}
def\chit@cellbg(#1,#2)#3\%
draw[chit/cell background](#1-#3/2,#2-#3/2) rectangle++(#3,#3);
def\chit@celldblbg(#1,#2)#3\%
draw[chit/cell background,chit/cell background flipped/.try]%
(#1-#3/2,#2-#3/2) rectangle++(#3,#3);
\ifchits@reset
This ‘if’ controls whether to reset the coordinates to the origin when \chits is called. If true, then reset for a new
\chits
\@chits
\chit@sng@cellupdate

\def\chit@sng@cellupdate(#1,#2)#3#4{%  
  \chit@dbg{2}{Current '#1' vs '#4'*('#3'+1)}
  \pgfmathparse{ifthenelse(#1>=#4*(#3-1),#2-#4,#2)}%
  \xdef#2{\pgfmathresult}%
  \pgfmathparse{ifthenelse(#1>=#4*(#3-1),0,#1+#4)}%
  \xdef#1{\pgfmathresult}%
}%

The stared version (\chits*) of this macro continues the previously set chit table.

\def\chits{%  
  \@ifstar{\chits@resetfalse\@chits}{\chits@resettrue\@chits}}
\def\@chits#1#2#3{%  
  \ifchits@reset  
    \def\r{0}%  
    \def\c{0}%  
  \fi  
  \chit@dbg{1}{Chits to make: #1}%
  \foreach[\count=	i from 0\] \t/\x in #1{%  
    \chit@dbg{2}{Turn '\t' with option '\x'}
    \ifx\t\empty\else%  
      \foreach \u/\m in \t{%  
        \chit@dbg{2}{Next chit '\u' with possible multiplicity '\m'}
        \ifx\u\empty\def\m{1}\fi%
        \ifx\u\m\def\m{1}\fi%
        \chit@dbg{2}{Next chit '\u' multiplicity '\m'}
        \foreach \n in {1,...,\m}{%  
          \chit@cellbg(\c,\r){#3}  
          \chit[@=\ti](\c,\r)%  
          \chit@sng@cellupdate(\c,\r){#2}{#3}%  
        }%  
      }%  
    }%  
  }%  
  \@ifnextchar;{\@gobble}{}%}

\chitgrid
1. columns
2. rows
3. cell-size

\def\chitgrid#1#2#3{\
  \pgfmathparse{#3/2}\edef\rmin{\pgfmathresult}\%
  \pgfmathparse{#2*#3-#3/2}\edef\rmax{\pgfmathresult}\%
  %\draw[red](-#3/2,\rmin)rectangle(#3*#1-#3/2,-\rmax);\
  \foreach \cc in {0,...,#1}{\n    \draw[chit/grid lines] (\cc*#3-#3/2,3*#3/4)--(\cc*#3-#3/2,-\rmax-#3/4);\n    %\chit@dbg{0}{Drawing horizontal lines from '\rmin, '-\rmin', ..., '-\rmax'}\n  }\n  \foreach \rr in {\rmin,-\rmin,...,-\rmax}{\n    %\chit@dbg{0}{Horizontal line at '\rr'}\n    \draw[chit/grid lines] (-3*#3/4,\rr)--(#1*#3-#3/4,\rr);\n  }\n}\n
\doublechits\ndef\chit@dbl@cellupdate\ndef\chit@dbl@flip

1. coordinates
2. coordinates
3. cell-size

\def\chit@dbl@flip(#1,#2)#3{\n  \pgfmathparse{-#1}\xdef\mc{\pgfmathresult}\%
}\n\def\chit@dbl@cellupdate(#1,#2)#3#4{\n  \pgfmathparse{ifthenelse(#1<-#4/2,#2,#4+#2)}\xdef#2{\pgfmathresult}\%
  \pgfmathparse{ifthenelse(#1<-#4/2,#4+#1,-(#3-.5)*#4)}\xdef#1{\pgfmathresult}\%
}\n
1. List of list of keys
2. Number of columns
3. size of each cell

349
The stared version (`\doublechits*`) of this macro continues the previously set chit table.

\begin{verbatim}
15254 \def\doublechits{%
15255 \@ifstar{\chits@resetfalse\@doublechits}{\chits@resettrue\@doublechits}}
15256 \def\@doublechits#1#2#3{%
15257 \chit@dbg{1}{Setting double-sided chits: #1}
15258 \ifchits@reset
15259 \pgfmathparse{-(#2-.5)*#3}
15260 \edef\c{\pgfmathresult}
15261 \edef\r{0}
15262 \fi
15263 \foreach [count=\ti from 0] \t/\x in #1{
15264 \ifx\t\empty\else%
15265 \foreach \u/\m in \t{%
15266 \ifx\u\empty\else%
15267 \ifx\m\empty\def\m{1}\else%
15268 \ifx\u\m\def\m{1}\else%
15269 \chit@dbg{2}{'\u'='\m' (\c,\r)}
15270 \foreach \n in {1,...,\m}{%
15271 \ifx\u\chit@blank
15272 \chit@dbg{3}{Ignoring blank chit:\u}
15273 \else
15274 \chit@cellbg(\c,\r){#3}
15275 \chit[\u=\ti](\c,\r)
15276 \chit@dbl@flip(\c,\r){#3}
15277 \chit@celldblbg(\mc,\r){#3}
15278 \chit[\u space flipped=\ti,zone turn=\t,zone mult=\n](\mc,\r)
15279 \chit@dbl@cellupdate(\c,\r){#3}
15280 \fi
15281 \fi
15282 }%
15283 \fi
15284 }
15285 \fi
15286 }
15287 \foreach \cc in {0,...,\#2}{
15288 \chit@dbg{0}{Drawing horizontal lines from \('-\rmin, '-\rmin', ..., '\r'}
15289 \chit\rr in {-\rmin, \rmin,...,\r}{
15290 \chit@dbl@bg(0){Horizontal line at '\rr'}
15291 \chit\rr in {-\rmin, \rmin,...,\r}{
15292 \chit@dbl@bg(0){Horizontal line at '\rr'}
15293 \pgfmathparse{(#3/2)/\edef\rmin{\pgfmathresult}}%
15294 \foreach \rr in {-#2*#3/4,...,-#3/4}{
15295 \chit@dbg{0}{Drawing horizontal lines from \'-\rmin, '-\rmin', ...,'\rr'}
15296 \draw[dashed] (-\rmin, \rmin,...,\rr){
15297 \chit@dbl@bg(0){Horizontal line at '\rr'}
15298 }%
15299 \draw[dashed] (-#2*#3/4,\rr)--(#2*#3/4,\rr);}
15300 \edef\r{0}
\end{verbatim}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{\doublechitgrid} & 1. columns \\
\hline
\end{tabular}
\caption{A table with 1 column and 1 row.}
\end{table}
\begin{verbatim}
\def\doublechitgrid#1#2#3{\%
\pgfmathparse{#3/2}\edef\rmin{\pgfmathresult}\%
\pgfmathparse{#2*#3-#3/2}\edef\rmax{\pgfmathresult}\%
\foreach \cc in {0,...,#1}{
  \draw[chit/grid lines] (\cc*#3,-3*#3/4)--(\cc*#3,\rmax+#3/4);
  \draw[chit/grid lines] (-\cc*#3,-3*#3/4)--(-\cc*#3,\rmax+#3/4);\}
%\chit@dbg{0}{Drawing horizontal lines from \rmin, \rmin', ..., \rmax'}
\foreach \rr in {-\rmin,\rmin,...,\rmax}{
  \draw[chit/grid lines] (-#1*#3-#3/4,\rr)--(#1*#3+#3/4,\rr);\}
\}
%
\end{verbatim}

5.5.11 Battle markers

Takes 1 arguments - the identifier.
Define every battle marker to change the style.

\begin{verbatim}
\tikzset{\%
  battle marker/.pic={
    \node[shape=circle,\node[shape=circle,font=\sffamily\bfseries,inner sep=0pt,minimum size=5mm,draw=black,fill=yellow!85!black,every battle marker/.try] at (-.3,.3) {\ifnum#1>0\relax #1\fi};\%
},
  battle marker/.style={
    chit={full={battle marker=#1},frame={draw=none}},
  }
}\%
\end{verbatim}

Takes two arguments - the odds and the fill colour. The latter is useful to differentiate the severity of an attack.
Define every odds marker to change the style.

\begin{verbatim}
\tikzset{\%
  pics/odds marker/.style args={#1,#2}{
    code={
      \node[shape=circle,font=\sffamily\bfseries\large,inner sep=0pt,minimum size=8mm,draw=black,fill=#2,every odds marker/.try] at (.16,-.16) {#1};
    }
  },
  odds marker/.style args=(#1,#2){
}
\end{verbatim}
Takes two arguments - the result and the fill colour. The latter is useful to differentiate the severity of an attack. Define every result marker to change the style.

```
\tikzset{
  pics/result marker/.style args={#1,#2}{
    code={
      \message{^^JResults marker #1 (#2)}
      \node[shape=circle,
        font=\sffamily\bfseries\large,
        inner sep=0pt,
        minimum size=8mm,
        draw=black,
        fill=#2,
        every result marker/.try] at (0,0) {#1};},
    result marker/.style args={#1,#2}{
      chit={full={result marker={#1,#2}},frame={draw=none}}
    }
  }
}
```

### 5.5.12 Dice

First, a regular 6-sided dice with configurable number of dots. Use like

```
\pic[⟨pic options⟩]{dice=⟨eyes⟩}
```

For example:

1. ![Image of a dice with one dot](image)
2. ![Image of a dice with two dots](image)
3. ![Image of a dice with three dots](image)
4. ![Image of a dice with four dots](image)
5. ![Image of a dice with five dots](image)
6. ![Image of a dice with six dots](image)

```
\tikzset{
  dice bg/.style={
    % /utils/exec={
```
Now some shapes of different dice. This was originally done by David Carlisle. Usage is for example

\node[shape=(dice),⟨node options⟩] {⟨value⟩};

where ⟨dice⟩ is one of d4, d6, d8, d10, d12, or d20.
Tetrahedron

Cubic

Octahedron
Decahedron

\pgfdeclare形状{d10}{
\anchor{center}{\pgfpointorigin} % within the node, (0,0) is the center
\anchor{text}{
% this is used to center the text in the node
\pgfpoint{-0.5\wd\pgfnodeparttextbox}{-0.5\ht\pgfnodeparttextbox}}
\背景路径{ % draw border
\pgfpathmoveto{\pgfpoint{0cm}{0.5cm}}
\pgfpathlineto{\pgfpoint{0.294cm}{-0.154cm}}
\pgfpathlineto{\pgfpoint{0cm}{-0.3cm}}
\pgfpathlineto{\pgfpoint{-0.294cm}{-0.154cm}}
\pgfpathlineto{\pgfpoint{0cm}{0.5cm}}
\pgfpathlineto{\pgfpoint{0.475cm}{0.1cm}}
\pgfpathlineto{\pgfpoint{0.475cm}{-0.1cm}}
\pgfpathlineto{\pgfpoint{0cm}{-0.5cm}}
\pgfpathlineto{\pgfpoint{-0.475cm}{-0.1cm}}
\pgfpathlineto{\pgfpoint{-0.475cm}{0.1cm}}
\pgfpathlineto{\pgfpoint{0cm}{-0.5cm}}
\pgfusepath{draw} %draw interiaor}
}

Dodecahedron

\pgfdeclare形状{d12}{
\anchor{center}{\pgfpointorigin} % within the node, (0,0) is the center
\anchor{text}{
% this is used to center the text in the node
\pgfpoint{-0.5\wd\pgfnodeparttextbox}{-0.5\ht\pgfnodeparttextbox}}
\背景路径{ % draw border
\pgfpathmoveto{\pgfpoint{0cm}{0.5cm}}
\pgfpathlineto{\pgfpoint{0.294cm}{0.405cm}}
\pgfpathlineto{\pgfpoint{0.475cm}{0.173cm}}
\pgfpathlineto{\pgfpoint{0.475cm}{-0.173cm}}
\pgfpathlineto{\pgfpoint{0.294cm}{-0.405cm}}
\pgfpathlineto{\pgfpoint{0cm}{-0.5cm}}
\pgfpathlineto{\pgfpoint{-0.294cm}{-0.405cm}}
\pgfpathlineto{\pgfpoint{-0.475cm}{-0.173cm}}
\pgfpathlineto{\pgfpoint{-0.475cm}{0.173cm}}
\pgfpathlineto{\pgfpoint{-0.294cm}{0.405cm}}
\pgfpathlineto{\pgfpoint{0cm}{0.5cm}}
\pgfpathlineto{\pgfpoint{0.332cm}{-0.282cm}}
\pgfpathlineto{\pgfpoint{0.332cm}{0.108cm}}
\pgfusepath{draw} %draw interiaor
}

355
5.5.13 Some utilities

This style allows us to add a fading drop-shadow to chits.

\usetikzlibrary{shadows.blur}
\newif\ifwg@chit@drop\wg@chit@dropfalse
\tikzset{
chit has drop/.is if=wg@chit@drop,
chit has drop/.default=true,
chit has drop/.initial=false,
/tikz/render blur shadow/.add code={%
  \chit@dbg{2}{Number of blur steps: \pgfbs@steps}%
  \ifnum\pgfbs@steps=0\else
    \chit@dbg{2}{Making shadow blur}%
  \fi%
},
\tikzset{%
  chit drop/.code={%
    % \message{^^J Args '#1'}%
    \pgfkeysalso{%
      chit has drop=true,
      /tikz/blur shadow={shadow blur steps=5, 
        shadow opacity=25, 
        shadow xshift=0.05cm, 
        shadow yshift=-0.05cm, 
        shadow blur radius=0.05cm,
        #1}}%
    \ifnum\pgfbs@steps=0%
      \gdef\wg@drop@margin{0pt}%
    \else%
      \ifwg@chit@drop%
        \pgfmathparse{
          \pgfbs@radius+
          veclen( 
            \pgfkeysvalueof{/tikz/shadow xshift}, 
            \pgfkeysvalueof{/tikz/shadow yshift})}
        \xdef\wg@drop@margin{\pgfmathresult pt}%
      \else%
        \gdef\wg@drop@margin{0pt}%
      \fi
    \fi%
    % \message{^^J Drop margin is '\wg@drop@margin' '
    \pgfbs@radius' '% \pgfkeysvalueof{/tikz/shadow xshift}',
    \pgfkeysvalueof{/tikz/shadow yshift})'}%
  },
  chit drop/.default=,
  no chit drop/.code={%
    \pgfkeysalso{
      /tikz/blur shadow={shadow blur steps=0})}
    \gdef\wg@drop@margin{0pt}%
  }%
}

Game turn marker
\tikzset{
  chit/text base/.style={
    shape=rectangle,
    inner sep=0pt,
    align=center,
    text width=1.1cm},
}
15583 \chit/number/.style={
15584  \chit/text base,
15585  font=\sffamily\bfseries\fontsize{12}{14}\selectfont},
15586 \chit/game turn/.style={
15587  \chit/text base,
15588  font=\sffamily\bfseries},
15589 \chit/text/.style={
15590  \chit/text base,
15591  font=\sffamily\bfseries},
15592 \chit/small text/.style={
15593  \chit/text base,
15594  font=\sffamily\bfseries\fontsize{9}{10}\selectfont},
15595 \chit/number/.pic={\node[\chit/number]{#1};},
15596 \chit/game turn/.pic={\node[\chit/game turn]{Game\\Turn};},
15597 \chit/text/.pic={\node[\chit/text]{#1};},
15598 \chit/small text/.pic={\node[\chit/small text]{#1};},
15599 game turn chit/.style={
15600  /\chit/full={\chit/game turn},
15601  color=black,
15602  fill=white},
15603 game turn chit flipped/.style={game turn chit},
15604 dummy chit/.style={fill=white},
15605 }

Marks of chits
15606 \providecommand\chitmark[2][]{\tikz[scale=.25,#1]{\chit[#2]}}

Stacking mark
15607 {\tikzset{
15608  wg stacking/.style={fill=white,
15609     /\chit/symbol={\{faction=friendly,command=land\}}},
15610 }
15611 \DeclareRobustCommand\stackmark[1][]{%
15612 \tikz[baseline=(current bounding box.center),scale=.3,#1]{
15613   \stackchits(0,0)(.3,-.3){%\noexpand\chit[wg stacking],
15614     \noexpand\chit[wg stacking],
15615     \noexpand\chit[wg stacking]}}

ZOC mark
15617 \DeclareRobustCommand\zocmark[1][]{%
15618 \tikz[baseline=($(current bounding box.center)!0.5!(current bounding box.south)),scale=.1,#1]{%
15619   \begin{scope}[hex/first row and column are=0,
15620     hex/row direction is=normal,
15621     hex/column direction is=normal,
15622     hex/short columns=none}
15623     \hex[label=.fill=gray](c=1,r=1)%
15624     \hex[label=.fill=white](c=1,r=2)%
15625     \hex[label=.fill=white](c=1,r=0)%
15626     \hex[label=.fill=white](c=0,r=0)%
15627     \hex[label=.fill=white](c=0,r=1)%
358
5.6 The war\texttt{game.natoapp6c} Ti\texttt{kZ} library

In this section we define the code for the TikZ library. The library defines a number of \texttt{pic} keys we can use to draw various parts of a marker. The markers conform to NATO App 6(c) specification. The implementation here is heavily inspired by the package \texttt{milsymb} \cite{milsymb} available at CTAN.

5.6.1 Debugging

\begin{verbatim}
\setcounter{natoappdbglvl}{\wargamedbglvl}
\def\natoappdbg#1#2{\ifnum#1>\natoappdbglvl\relax\else\message{\textnormal{\dquote{#2}}}\fi}
\end{verbatim}

5.6.2 Colours

\begin{verbatim}
\definecolor{friendly}{RGB}{128, 224, 255}
\definecolor{hostile}{RGB}{255, 128, 128}
\definecolor{neutral}{RGB}{170, 255, 170}
\end{verbatim}

Define standard colours for marker affiliations.
5.6.3 Some dimensions

We define a number of dimensions which we will use in the following. They provide a rough parameterisation of the node shapes, but shouldn’t really be changed. We have them here so that the code uses as few hard coded numbers as possible.

The dimensions are:

- Installation ‘hat’ x coordinate
- Installation ‘hat’ height
- Activity width of boxes
- Height of space bar
- Radius of the symbol

```latex
\newdimen\toppinstx
\toppinstx=0.2cm
\newdimen\toppinsth
\toppinsth=0.15cm
\newdimen\toppactw
\toppactw=0.15cm
\newdimen\toppspaceh
\toppspaceh=0.1cm
\newdimen\toppr
\toppr=0.5cm
```

5.6.4 Some utilities

```latex
\iftoppisclip
This detects if we’re in a node that is being used for clipping
```

```latex
\def\toppisclip{clip}
\def\toppiscliptoken{FF}\fi%
\message{^^Jclip is \meaning\pgf@up@clip}\
\ifx\pgf@up@clip\toppiscliptoken
\newif\iftoppisclip
\toppisclipfalse
```

```latex
\toppsavedfillcolor
\toppsavedstrokecolor
Macros to hold saved colours.
```

```latex
\let\toppsavedfillcolor\relax
\let\toppsavedstrokecolor\relax
```

360
Macro to get stroke and fill colours and set the fill colour to the stroke colour, and to restore to the old setting. This is used by the frame shapes below to make sure that filled elements of the frame uses the same colour as the for strokes.

\newcommand\n@to@pp@stroke@to@fill{\%
\expandafter\let\expandafter\n@to@pp@saved@stroke@color\csname\string\color@pgfstrokecolor\endcsname\%
\expandafter\let\expandafter\n@to@pp@saved@fill@color\csname\string\color@pgffillcolor\endcsname\%
\expandafter\pgf@setfillcolor\n@to@pp@saved@stroke@color\%
\message{^^J=== Set fill to stroke color
\message{^^J Old fill: \meaning\n@to@pp@saved@fill@color
\message{^^J Old stroke: \meaning\n@to@pp@saved@stroke@color}}\%
\}
\newcommand\n@to@pp@restore@fill{\%
\message{^^J=== Restore fill color
\message{^^J Old fill: \meaning\n@to@pp@saved@fill@color
\message{^^J Old stroke: \meaning\n@to@pp@saved@stroke@color}}\%
\ifx\n@to@pp@saved@fill@color\relax\else\%
\expandafter\pgf@setfillcolor\n@to@pp@saved@stroke@color\%
\fi\%
\global\let\n@to@pp@saved@fill@color\relax
\global\let\n@to@pp@saved@stroke@color\relax
\}
\newenvironment{n@to@pp@stroketofill}{\pgfscope\n@to@pp@stroke@to@fill}{\n@to@pp@restore@fill\endpgfscope\}

We also make an environment, just to simplify the use
\newenvironment{n@to@pp@stroketofill}{\pgfscope\n@to@pp@stroke@to@fill}{\n@to@pp@restore@fill\endpgfscope\}

5.6.5 Faction names as macros
\def\n@to@pp@friendly{friendly}
\def\n@to@pp@hostile{hostile}
\def\n@to@pp@neutral{neutral}
\def\n@to@pp@unknown{unknown}

5.6.6 Node shapes
Here we define bases for all commands and affiliations. These are defined as node shapes. This means we will render the NATO App6(c) symbols as nodes with embedded nodes of the relevant shape.

361
First, the generic bounding box symbol for all markers.

![natoapp6c base](image)

Place-holder symbol. This shape will form the basis of many of the other frame shapes. We define the relevant sizes and anchors.

```latex
\pgfdeclarerectangle{natoapp6c base}{%
\saveddimen\radius{\pgf@x=\nato@pp@r}
\saveddimen\liney{\pgf@x=.2cm}
\saveddimen\linex{\pgf@x=0.41cm}
\savedanchor\center{\pgf@x=0cm\pgf@y=0cm}
\savedanchor\upper{\pgf@x=0cm\pgf@y=0.35cm}
\anchor{north east}{\pgf@x=\radius\pgf@y=\radius}
\anchor{north west}{\pgf@x=-\radius\pgf@y=\radius}
\anchor{south west}{\pgf@x=-\radius\pgf@y=-\radius}
\anchor{south east}{\pgf@x=\radius\pgf@y=-\radius}
\anchor{south}{\pgf@x=0cm\pgf@y=-\radius}
\anchor{north}{\pgf@x=0cm\pgf@y=\radius}
\anchor{west}{\pgf@x=-\radius\pgf@y=0cm}
\anchor{east}{\pgf@x=\radius\pgf@y=0cm}
\anchor{center}{\center}
\anchor{upper}{\upper}
\anchor{lower}{\upper\pgf@y=-\pgf@y}
\anchor{left}{\upper\pgf@x=-\pgf@y\pgf@y=0cm}
\anchor{right}{\upper\pgf@x=\pgf@y\pgf@y=0cm}
\savedmacro\init{%
\def\octagon{%
\pgfpathmoveto{\pgfqpointpolar{0}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{45}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{90}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{135}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{180}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{225}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{270}\radius}{\radius}}%
\pgfpathlineto{\pgfqpointpolar{315}\radius}{\radius}}%
\pgfpathclose}
\def\topline{%
\pgfpathmoveto{\pgfqpoint{\linex}{\liney}}{%}
\pgfpathlineto{\pgfqpoint{-\linex}{\liney}}{}}%
\def\bottomline{%
\pgfpathmoveto{\pgfqpoint{\linex}{-\liney}}{%}
\pgfpathlineto{\pgfqpoint{-\linex}{-\liney}}{}}%
\backgroundpath{%
\init%
\octagon}
\behindforegroundpath{%
\init%
\octagon%
\pgfusepath{stroke}{}}%
362
```
5.6.7 ‘Friendly’ node shapes

The friendly air command.

```
\pgfdeclareshape{natoapp6c friendly air}{
\inheritsavedanchors[from=natoapp6c base]
\savedanchor\southeast{\pgf@x=1.1\n@to@pp@r\pgf@y=-\n@to@pp@r}
\savedanchor\cntrl{\pgf@x=0cm\pgf@y=2.6\n@to@pp@r}
\savedanchor\north{\pgf@x=0cm\pgf@y=1.6\n@to@pp@r}
\anchor{south east}{\southeast}
\anchor{south west}{\southeast\pgf@x=-\pgf@x}
\anchor{north east}{\southeast\pgf@x=\pgf@x}\north\pgf@x=\pgf@x}
\anchor{north west}{\southeast\pgf@x=\pgf@x}\north\pgf@x=-\pgf@x}
\anchor{north}{\north}
\anchor{east}{\pgf@y=-\pgf@y}
\anchor{west}{\pgf@y=\pgf@y}
\anchor{south}{\southeast}\pgf@x=0cm}
```

Macro for friendly air shape

```
\def\n@to@friendly@ir{% 
\southeast \wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}%
\cntrl \vg@tmpb=\pgf@y%
\pgfpatharc{180}{0}{\wg@tmpa and \wg@tmpb}}
```
The friendly land command. The most used command frame.
The friendly activity command. Similar to land command, but with boxes in the corners.
natoapp6c friendly equipment

The friendly equipment command. A circle.

\begin{n@toappstroketo}
The friendly installation command. Similar to the land command, but with a ‘hat’ on top.

```latex
\pgfdeclare形状{natoapp6c friendly installation}{%
\inherited锚点{from=natoapp6c friendly land}{center}
\inherited锚点{from=natoapp6c friendly land}{inner north east}
\inherited锚点{from=natoapp6c friendly land}{inner north west}
\inherited锚点{from=natoapp6c friendly land}{inner south west}
\inherited锚点{from=natoapp6c friendly land}{inner south east}
\inherited锚点{from=natoapp6c friendly land}{north east}
\inherited锚点{from=natoapp6c friendly land}{north west}
\inherited锚点{from=natoapp6c friendly land}{south east}
\inherited锚点{from=natoapp6c friendly land}{south west}
\inherited锚点{from=natoapp6c friendly land}{north}
\inherited锚点{from=natoapp6c friendly land}{west}
\inherited锚点{from=natoapp6c friendly land}{east}
\inherited锚点{from=natoapp6c friendly land}{south}
\inherited锚点{from=natoapp6c friendly land}{upper}
\inherited锚点{from=natoapp6c friendly land}{lower}
\inherited锚点{from=natoapp6c friendly land}{left}
\inherited锚点{from=natoapp6c friendly land}{right}
\inherited锚点{from=natoapp6c friendly land}{center}
\inherited背景路径{from=natoapp6c friendly land}
\behindforeground路径{
\begin{notopstroke}
\natofriendly\pgfusepath{stroke}
\n
\northeast \wg@tmpa=\pgf@y\%
\wg@tmpb=\wg@tmpa\advance\wg@tmpb\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advance\nato\advan
```
The friendly space command. Similar to air command, but with a bar on top.
Macro for friendly sub surface command

\def\natoapp6c friendly@sub{\pgfdeclareshape{natoapp6c friendly sub surface}{\inherit{\natoapp6c base}{\northeast}{\pgf@x=1.1\northeast\pgf@x}{\south}{\pgf@y=1.6\south\pgf@y}{\as{\natoapp6c friendly@sub}{\pgf@x}{\pgf@y}}}}

The friendly sub surface command.
5.6.8 ‘Hostile’ node shapes

The hostile air command

Macro for hostile air shape

\def\n@to@hostile@@ir{\southwest\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y
\cntrl\wg@tmpc=\pgf@y\north\wg@tmpd=\pgf@y
\pgfpathmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}
\pgfusepath{stroke}}
The hostile air command.

\pgfdeclareshape{natoapp6c hostile air}{
\inheritsavedanchors[from=natoapp6c base]
\savedanchor{southeast}{\pgf@x=
\n@to@pp@r\pgf@y=-\n@to@pp@r}
\savedanchor{cntrl}{\pgf@x=\n@to@pp@r\pgf@y=0.414\n@to@pp@r% (sqrt(2)-1)}
\savedanchor{north}{\pgf@x=0cm\pgf@y=1.414\n@to@pp@r}
\anchor{south east}{\southeast}
\anchor{south west}{\southeast\pgf@x=-\pgf@x}
\anchor{north east}{\southeast\n@to@pp@r\pgf@x=\n@to@pp@r\pgf@y=\n@to@pp@r
\advance\pgf@x-\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y}
\anchor{north west}{\southeast\n@to@pp@r\pgf@x=\n@to@pp@r\pgf@y=\n@to@pp@r
\advance\pgf@x-\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y}
\anchor{north}{\north}
\anchor{east}{\north\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y
\divide\pgf@x\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y
\advance\pgf@x\pgf@x=\pgf@x\pgf@y=\pgf@y
\pgf@x-\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y}
\anchor{west}{\north\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y
\divide\pgf@x\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y
\advance\pgf@x\pgf@x=\pgf@x\pgf@y=\pgf@y
\pgf@x-\pgf@y\pgf@x=\pgf@x\pgf@y=\pgf@y}
\anchor{south}{\southeast\pgf@x=0cm}
\inheritanchor[from=natoapp6c base]{upper}
\inheritanchor[from=natoapp6c base]{lower}
\inheritanchor[from=natoapp6c base]{left}
\inheritanchor[from=natoapp6c base]{right}
\inheritanchor[from=natoapp6c base]{center}
\backgroundpath{\n@to@hostile@@ir}
\behindforegroundpath{\n@to@hostile@@ir
\pgfusepath{stroke}}
}
Macro for hostile land command

\def\n@to@hostile@land{%
ortheast \wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{0cm}}%
\pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{0cm}}%
\pgfpathlineto{\pgfqpoint{0cm}{-\wg@tmpb}}%
\pgfclosepath%
}

The hostile land command.

\pgfdeclareshape{natoapp6c hostile land}{%
\inheritsavedanchors[from=natoapp6c base]
\savedanchor\northeast{%
\pgf@x=1.414\n@to@pp@r%
\pgf@y=1.414\n@to@pp@r}
\anchor{north east}{\northeast}
\anchor{north west}{\northeast\pgf@x=-\pgf@x}
\anchor{south east}{\northeast\pgf@y=-\pgf@y}
\anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{north}{\northeast\pgf@x=0cm}
\anchor{south}{\northeast\pgf@x=0cm\pgf@y=-\pgf@y}
\anchor{east}{\northeast\pgf@y=0cm}
\anchor{west}{\northeast\pgf@x=-\pgf@x\pgf@y=0cm}
\anchor{upper}{\northeastanchor[from=natoapp6c base]{upper}}
\anchor{lower}{\northeastanchor[from=natoapp6c base]{lower}}
\anchor{left}{\northeastanchor[from=natoapp6c base]{left}}
\anchor{right}{\northeastanchor[from=natoapp6c base]{right}}
\anchor{center}{\northeastanchor[from=natoapp6c base]{center}}
\backgroundpath{%
\n@to@hostile@land%
}
\behindforegroundpath{%
\n@to@hostile@land%
\pgfusepath{stroke}%
}
}

The hostile activity command. Similar to land command, but with boxes in the corners.

\pgfdeclareshape{natoapp6c hostile activity}{%
The hostile equipment command. Same as land command.

\pgfdeclareshape{natoapp6c hostile equipment}{%
  \inheritsavedanchors[from=natoapp6c hostile land]
  \inheritanchor[from=natoapp6c hostile land]{inner north east}
  \inheritanchor[from=natoapp6c hostile land]{inner north west}
  \inheritanchor[from=natoapp6c hostile land]{inner south west}
  \inheritanchor[from=natoapp6c hostile land]{inner south east}
  \inheritanchor[from=natoapp6c hostile land]{north east}
  \inheritanchor[from=natoapp6c hostile land]{north west}
  \inheritanchor[from=natoapp6c hostile land]{south east}
  \inheritanchor[from=natoapp6c hostile land]{south west}
  \inheritanchor[from=natoapp6c hostile land]{north}
  \inheritanchor[from=natoapp6c hostile land]{west}
  \inheritanchor[from=natoapp6c hostile land]{east}
  \inheritanchor[from=natoapp6c hostile land]{south}
  \inheritanchor[from=natoapp6c hostile land]{upper}
  \inheritanchor[from=natoapp6c hostile land]{lower}
  \inheritanchor[from=natoapp6c hostile land]{left}
  \inheritanchor[from=natoapp6c hostile land]{right}
  \inheritanchor[from=natoapp6c hostile land]{center}
  \inheritbackgroundpath[from=natoapp6c hostile land]
  \inheritbehindforegroundpath[from=natoapp6c hostile land]
}:

The hostile installation command. Similar to land command, but with a ‘hat’ on top.

\pgfdeclareshape{natoapp6c hostile installation}{%
  \inheritsavedanchors[from=natoapp6c hostile land]
  \inheritanchor[from=natoapp6c hostile land]{center}
  \inheritanchor[from=natoapp6c hostile land]{inner north east}
  \inheritanchor[from=natoapp6c hostile land]{inner north west}
  \inheritanchor[from=natoapp6c hostile land]{inner south west}
  \inheritanchor[from=natoapp6c hostile land]{inner south east}
  \inheritanchor[from=natoapp6c hostile land]{inner east}
  \inheritanchor[from=natoapp6c hostile land]{inner west}
  \inheritanchor[from=natoapp6c hostile land]{inner south}
  \inheritanchor[from=natoapp6c hostile land]{inner north}
  \inheritanchor[from=natoapp6c hostile land]{upper}
  \inheritanchor[from=natoapp6c hostile land]{lower}
  \inheritanchor[from=natoapp6c hostile land]{left}
  \inheritanchor[from=natoapp6c hostile land]{right}
  \inheritanchor[from=natoapp6c hostile land]{center}
  \inheritbackgroundpath[from=natoapp6c hostile land]
  \inheritbehindforegroundpath[from=natoapp6c hostile land]
}:
The hostile sea surface command. Same as land command

\pgfdeclareshape{natoapp6c hostile sea surface}{
  \inheritsavedanchors[from=natoapp6c hostile land]
  \inheritanchorborder[from=natoapp6c hostile land]
  \inheritanchor[from=natoapp6c hostile land]{north east}
  \inheritanchor[from=natoapp6c hostile land]{north west}
  \inheritanchor[from=natoapp6c hostile land]{south east}
  \inheritanchor[from=natoapp6c hostile land]{south west}
  \inheritanchor[from=natoapp6c hostile land]{north}
  \inheritanchor[from=natoapp6c hostile land]{west}
  \inheritanchor[from=natoapp6c hostile land]{east}
  \inheritanchor[from=natoapp6c hostile land]{south}
  \inheritanchor[from=natoapp6c hostile land]{upper}
  \inheritanchor[from=natoapp6c hostile land]{lower}
  \inheritanchor[from=natoapp6c hostile land]{left}
  \inheritanchor[from=natoapp6c hostile land]{right}
  \inheritanchor[from=natoapp6c hostile land]{center}
  \inheritbackgroundpath[from=natoapp6c hostile land]
  \behindforegroundpath{
    \begin{n@to@pp@stroketofill}
    \n@to@hostile@l@nd%
    \pgfusepath{stroke}
    \northeast \wg@tmpa=\pgf@x\wg@tmpb\pgf@y%
    \wg@tmpc=\wg@tmpb
    \advance\wg@tmpc\n@to@pp@inst@h%
    \advance\wg@tmpc-0.05cm%
    \pgfpathmoveto{\pgfqpoint{\wg@tmpa}{0cm}}
    \pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpc}}
    \pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpc}}
    \pgfpathlineto{\pgfqpoint{-\wg@tmpa}{0cm}}
    \pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpb}}
    \pgfclosepath
    \pgfusepath{clip}
    \% \n@to@pp@inst@x \wg@tmpd=\wg@tmpb%
    \advance\wg@tmpd\n@to@pp@inst@h%
    \pgfpathmoveto{\pgfqpoint{\n@to@pp@inst@x}{\wg@tmpc}}
    \pgfpathlineto{\pgfqpoint{-\n@to@pp@inst@x}{\wg@tmpc}}
    \pgfpathlineto{\pgfqpoint{-\n@to@pp@inst@x}{\wg@tmpd}}
    \pgfpathlineto{\pgfqpoint{\n@to@pp@inst@x}{\wg@tmpd}}
    \pgfclosepath
    \pgfusepath{fill}
  \end{n@to@pp@stroketofill}
}
natoapp6c hostile space

The hostile space command. Similar to air command, but with bar on top.

\begin{n@to@pp@stroketofill}
\n@to@Hostile@@ir
\pgfusepath{stroke,clip}
\north\wg@tmpa=\pgf@y\wg@tmpb=\pgf@y
\advance\wg@tmpb-\n@to@pp@space@h
\end{n@to@pp@stroketofill}
Macro for hostile sub surface command

\def\n@to@hostile@sub{%}
\northeast \wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\cntrl \wg@tmpc=\pgf@y%
\south \wg@tmpd=\pgf@y%
\pgfpathmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpc}}%
\pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpd}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpc}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfclosepath%
\pgfusepath{fill}%
\end{n@to@pp@stroketofill}
}

The hostile sub surface command

\pgfdeclareshape{natoapp6c hostile sub surface}{%
\inherit\savedanchors{\from=natoapp6c base}
\anchor{north east}{\northeast}
\anchor{north west}{\northeast\pgf@x=-\pgf@x}
\anchor{south east}{\northeast\wg@tmpa=\pgf@x\south\pgf@x=\wg@tmpa}
\anchor{south west}{\northeast\wg@tmpa=\pgf@x\south\pgf@x=-\wg@tmpa}
\anchor{south}{\south}
\anchor{east}{
\northeast\wg@tmpb=\pgf@y\wg@tmpa=\pgf@x%
\south\wg@tmpc\pgf@y%
\advance\wg@tmpb2%
\advance\wg@tmpc%
\pgf@x=\wg@tmpa%
\pgf@y=\wg@tmpb}
\anchor{west}{
\northeast\wg@tmpb=\pgf@y\wg@tmpa=\pgf@x%
\south\wg@tmpc\pgf@y%
\advance\wg@tmpb\wg@tmpc%
\pgf@x=\wg@tmpa%
\pgf@y=\wg@tmpb}
5.6.9 'Neutral' node shapes

Macro for neutral shapes

\def\n\to\pp\neutr\r{\pgfdeclareshape{\n\to\pp\neutr\r}{\n\inherit\anchor{northeast}{\pgf@x=0cm}\n\inherit\anchor{north}{\pgf@x=0cm}\n\inherit\anchor{north west}{\pgf@x=-\pgf@x}\n\inherit\anchor{south east}{\pgf@x=\pgf@x}\n\inherit\anchor{south}{\pgf@x=0cm}\n\inherit\anchor{east}{\pgf@y=0cm}\n\inherit\anchor{west}{\pgf@y=-\pgf@y}\n\inherit\anchor{south west}{\pgf@x=-\pgf@x\pgf@y=-\pgf@y}\n\anchor{north east}{\northeast}\n\anchor{north west}{\northeast\pgf@x=-\pgf@x}\n\anchor{south east}{\northeast\pgf@x=\pgf@x}\n\anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}\n\anchor{top}{\northeast}\n\anchor{bottom}{\northeast\pgf@y=-\pgf@y}\n\anchor{center}{\northeast}\n\behindforegroundpath{\n@to@hostile@sub}\n\behindforegroundpath{\n@to@hostile@sub}\n\pgfusepath{stroke}\n\backgroundpath{\n@to@hostile@sub}}\n\n\def\n@to\pp@neutr@l@init{\n\northeast\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y\n\def\n@to\pp@neutr@l@left {\pgflineto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}}\n\def\n@to\pp@neutr@l@right {\pgflineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@top {\pgflineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@bottom{\pgflineto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}}\n\def\n@to\pp@neutr@l@nw {\pgfmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@ne {\pgfmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@se {\pgfmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}}\n\def\n@to\pp@neutr@l@sw {\pgfmoveto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}}\n}\n
\def\n@to\pp@neutr@l@init{\n\northeast\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y\n\def\n@to\pp@neutr@l@left {\pgflineto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}}\n\def\n@to\pp@neutr@l@right {\pgflineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@top {\pgflineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@bottom{\pgflineto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}}\n\def\n@to\pp@neutr@l@nw {\pgfmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@ne {\pgfmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}\n\def\n@to\pp@neutr@l@se {\pgfmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}}\n\def\n@to\pp@neutr@l@sw {\pgfmoveto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}}\n}\n
The neutral air command

\pgfdeclareshape{natoapp6c neutral air}{\n\inherit\anchor{northeast}{\pgf@x=0cm}\n\inherit\anchor{north}{\pgf@x=0cm}\n\inherit\anchor{north west}{\pgf@x=-\pgf@x}\n\inherit\anchor{south east}{\pgf@x=\pgf@x}\n\inherit\anchor{south}{\pgf@x=0cm}\n\inherit\anchor{east}{\pgf@y=0cm}\n\inherit\anchor{west}{\pgf@y=-\pgf@y}\n\inherit\anchor{south west}{\pgf@x=-\pgf@x\pgf@y=-\pgf@y}\n\anchor{north east}{\northeast}\n\anchor{north west}{\northeast\pgf@x=-\pgf@x}\n\anchor{south east}{\northeast\pgf@x=\pgf@x}\n\anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}\n\anchor{top}{\northeast}\n\anchor{bottom}{\northeast\pgf@y=-\pgf@y}\n\anchor{center}{\northeast}\n\behindforegroundpath{\n@to@hostile@sub}\n\behindforegroundpath{\n@to@hostile@sub}\n\pgfusepath{stroke}\n\backgroundpath{\n@to@hostile@sub}
The neutral land command

\pgfdeclareshape{natoapp6c neutral land}{% 
\inheritsavedanchors[from=natoapp6c neutral air] 
\inheritanchor[from=natoapp6c neutral air]{north east} 
\inheritanchor[from=natoapp6c neutral air]{north west} 
\inheritanchor[from=natoapp6c neutral air]{south east} 
\inheritanchor[from=natoapp6c neutral air]{south west} 
\inheritanchor[from=natoapp6c neutral air]{north} 
\inheritanchor[from=natoapp6c neutral air]{west} 
\inheritanchor[from=natoapp6c neutral air]{east} 
\inheritanchor[from=natoapp6c neutral air]{south} 
\inheritanchor[from=natoapp6c neutral air]{upper} 
\inheritanchor[from=natoapp6c neutral air]{lower} 
\inheritanchor[from=natoapp6c neutral air]{left} 
\inheritanchor[from=natoapp6c neutral air]{right} 
\inheritanchor[from=natoapp6c neutral air]{center} 
\backgroundpath{% 
\n@to@pp@neutr@l@init% 
\n@to@pp@neutr@l@ne 
\n@to@pp@neutr@l@top% 
\n@to@pp@neutr@l@left% 
\n@to@pp@neutr@l@bottom% 
\pgfclosepath 
% 
\behindforegroundpath{% 
\n@to@pp@neutr@l@init% 
\n@to@pp@neutr@l@ne 
\n@to@pp@neutr@l@top% 
\n@to@pp@neutr@l@left% 
\n@to@pp@neutr@l@bottom% 
\pgfclosepath 
% 
\n@to@pp@neutr@l@init% 
\n@to@pp@neutr@l@ne 
% \n@to@pp@neutr@l@top% 
\n@to@pp@neutr@l@left% 
\n@to@pp@neutr@l@bottom% 
\pgfclosepath 
} 
}
The neutral activity command. Similar to land command but with boxes added in the corners.
The neutral equipment command. Same as land command

\pgfdeclareimage{\natoapp6c neutral equipment}{\natoapp6c neutral equipment}\n
The neutral installation command. Similar to land command but with a ‘hat’ on top.

The neutral installation command.

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

The neutral installation command.

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```

```latex
\pgfdeclarelayer{natoapp6c neutral installation}
\pgfsetlayers{natoapp6c neutral installation}
```

```latex
\pgfdeclarelayer{natoapp6c neutral land}
\pgfsetlayers{natoapp6c neutral land}
```
The neutral sea surface command. Same as land command.

\pgfdeclareshape{natoapp6c neutral sea surface}{%
\inheritsavedanchors[from=natoapp6c neutral equipment]
\inheritanchor[from=natoapp6c neutral equipment]{inner north east}
\inheritanchor[from=natoapp6c neutral equipment]{inner north west}
\inheritanchor[from=natoapp6c neutral equipment]{inner south west}
\inheritanchor[from=natoapp6c neutral equipment]{inner south east}
\inheritanchor[from=natoapp6c neutral equipment]{north east}
\inheritanchor[from=natoapp6c neutral equipment]{north west}
\inheritanchor[from=natoapp6c neutral equipment]{south east}
\inheritanchor[from=natoapp6c neutral equipment]{south west}
\inheritanchor[from=natoapp6c neutral equipment]{north}
\inheritanchor[from=natoapp6c neutral equipment]{west}
\inheritanchor[from=natoapp6c neutral equipment]{east}
\inheritanchor[from=natoapp6c neutral equipment]{south}
\inheritanchor[from=natoapp6c neutral equipment]{upper}
\inheritanchor[from=natoapp6c neutral equipment]{lower}
\inheritanchor[from=natoapp6c neutral equipment]{left}
\inheritanchor[from=natoapp6c neutral equipment]{right}
\inheritanchor[from=natoapp6c neutral equipment]{center}
\inheritbackgroundpath[from=natoapp6c neutral equipment]
\inheritbehindforegroundpath[from=natoapp6c neutral equipment]
}

The neutral space command. Similar to air command but with a bar.

\pgfdeclareshape{natoapp6c neutral space}{%
\ inheritsavedanchors[from=natoapp6c neutral air]
\inheritanchor[from=natoapp6c neutral air]{north east}
\inheritanchor[from=natoapp6c neutral air]{north west}
\inheritanchor[from=natoapp6c neutral air]{south east}
\inheritanchor[from=natoapp6c neutral air]{south west}
\inheritanchor[from=natoapp6c neutral air]{north}
\inheritanchor[from=natoapp6c neutral air]{west}
\inheritanchor[from=natoapp6c neutral air]{east}
\inheritanchor[from=natoapp6c neutral air]{south}
\inheritanchor[from=natoapp6c neutral air]{upper}
\inheritanchor[from=natoapp6c neutral air]{lower}
\inheritanchor[from=natoapp6c neutral air]{left}
\inheritanchor[from=natoapp6c neutral air]{right}
\inheritanchor[from=natoapp6c neutral air]{center}
}
The neutral sub surface command

\pfgdeclareshape{natoapp6c\ neutral\ sub\ surface}{%
\inheritanchor[from=natoapp6c\ neutral\ air]{north east}
\inheritanchor[from=natoapp6c\ neutral\ air]{north west}
\inheritanchor[from=natoapp6c\ neutral\ air]{south east}
\inheritanchor[from=natoapp6c\ neutral\ air]{south west}
\inheritanchor[from=natoapp6c\ neutral\ air]{upper}
\inheritanchor[from=natoapp6c\ neutral\ air]{center}
\backgroundpath{% 
\pgfpathlineto{\pgfqpoint{-\pgf@xa}{\pgf@ya}}}
\pgfusepath{fill}%
}
5.6.10 ‘Unknown’ node shapes

Macro to define unknown path elements

\def\n@to@pp@unknown@init{% 
\def\n@to@pp@unknown@top{% 
\innernortheast \wg@tmpa=\pgf@x%
\cntrlnortheast \wg@tmpb=\pgf@x%
\pgfpathcurveto{% 
\pgfqpoint{\wg@tmpa}{\wg@tmpb}\pgfqpoint{-\wg@tmpa}{\wg@tmpb}\pgfqpoint{-\wg@tmpa}{\wg@tmpa}}
\def\n@to@pp@unknown@left{% 
\innernortheast \wg@tmpa=\pgf@x%
\cntrlnortheast \wg@tmpb=\pgf@x%
\pgfpathcurveto{% 
\pgfqpoint{-\wg@tmpb}{\wg@tmpa}\pgfqpoint{-\wg@tmpb}{-\wg@tmpa}\pgfqpoint{-\wg@tmpa}{-\wg@tmpa}}
\def\n@to@pp@unknown@bottom{% 
\innernortheast \wg@tmpa=\pgf@x%
\cntrlnortheast \wg@tmpb=\pgf@x%
\pgfpathcurveto{% 
\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}\pgfqpoint{\wg@tmpa}{-\wg@tmpb}\pgfqpoint{\wg@tmpa}{-\wg@tmpa}}
\def\n@to@pp@unknown@right{% 
\innernortheast \wg@tmpa=\pgf@x%
\cntrlnortheast \wg@tmpb=\pgf@x%
\pgfpathcurveto{% 
\pgfqpoint{-\wg@tmpa}{\wg@tmpb}\pgfqpoint{-\wg@tmpa}{\wg@tmpb}\pgfqpoint{\wg@tmpa}{\wg@tmpb}}
\pgfusepath{stroke}
}
The unknown air command. To consider: Should clipping path extend below the actual symbol to include that part of the base symbol?

\pgfdeclareshape{natoapp6c unknown air}{
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor[from=natoapp6c unknown land]{inner north east}
\inheritanchor[from=natoapp6c unknown land]{inner north west}
\backgroundpath{% 
\n@to@pp@unknown@init
\innernortheast \wg@tmpa=\pgf@x%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpa}}%
\n@to@pp@unknown@right %
\n@to@pp@unknown@top %
\n@to@pp@unknown@left %
\n@to@pp@unknown@bottom%
}%
\behindforegroundpath{% 
\n@to@pp@unknown@init
\innernortheast \wg@tmpa=\pgf@x%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpa}}%
\n@to@pp@unknown@right %
\n@to@pp@unknown@top %
\n@to@pp@unknown@left %
\n@to@pp@unknown@bottom%
} \pgfusepath{stroke}
The unknown activity command. Similar to land command, but with boxes in the the ‘corners’.

\pgfdeclareshape{natoapp6c unknown activity}{%
The unknown equipment command. Same as land command.

\begin{n@to@pp@stroketofill}
\pgfpathlineto{\pgfqpoint{\wg@tmpb}{\wg@tmpc}}
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpc}}
\pgfclosepath
\pgfusepath{fill}
\end{n@to@pp@stroketofill}

\begin{n@to@pp@stroketofill}
\pgfpathmoveto{\pgfqpoint{-\wg@tmpb}{\wg@tmpc}}
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpc}}
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpc}}
\pgfpathlineto{\pgfqpoint{-\wg@tmpb}{-\wg@tmpc}}
\pgfclosepath
\pgfusepath{fill}
\end{n@to@pp@stroketofill}

The unknown installation command. Similar to land command, but with a ‘hat’ on top. Note, NATO App6(d) makes
the ‘hat’ lower part disconnected from the main symbol. I find that ugly, so we do it like NATO App6(c).
The unknown sea surface command. Same as land command

```latex
\pgfdeclareshape{natoapp6c unknown sea surface}{%
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor[from=natoapp6c unknown land]{inner north east}
\inheritanchor[from=natoapp6c unknown land]{inner north west}
\inheritanchor[from=natoapp6c unknown land]{inner south west}
\inheritanchor[from=natoapp6c unknown land]{inner south east}
\inheritanchor[from=natoapp6c unknown land]{north east}
\inheritanchor[from=natoapp6c unknown land]{north west}
\inheritanchor[from=natoapp6c unknown land]{south east}
\inheritanchor[from=natoapp6c unknown land]{south west}
\inheritanchor[from=natoapp6c unknown land]{north}
\inheritanchor[from=natoapp6c unknown land]{west}
\inheritanchor[from=natoapp6c unknown land]{east}
\inheritanchor[from=natoapp6c unknown land]{south}
\inheritanchor[from=natoapp6c unknown land]{upper}
\inheritanchor[from=natoapp6c unknown land]{lower}
\inheritanchor[from=natoapp6c unknown land]{left}
\inheritanchor[from=natoapp6c unknown land]{right}
\inheritanchor[from=natoapp6c unknown land]{center}
\inheritbackgroundpath[from=natoapp6c unknown land]
\inheritbehindforegroundpath[from=natoapp6c unknown land]
}%
```

The unknown space command. Similar to air command, but with a top bar.

```latex
\pgfdeclareshape{natoapp6c unknown space}{%
\inheritsavedanchors[from=natoapp6c unknown air]
\inheritanchor[from=natoapp6c unknown air]{inner north east}
\inheritanchor[from=natoapp6c unknown air]{inner north west}
\inheritanchor[from=natoapp6c unknown air]{inner south west}
\inheritanchor[from=natoapp6c unknown air]{inner south east}
\inheritanchor[from=natoapp6c unknown air]{north east}
\inheritanchor[from=natoapp6c unknown air]{north west}
\inheritanchor[from=natoapp6c unknown air]{south east}
\inheritanchor[from=natoapp6c unknown air]{south west}
\inheritanchor[from=natoapp6c unknown air]{north}
\inheritanchor[from=natoapp6c unknown air]{west}
\inheritanchor[from=natoapp6c unknown air]{east}
\inheritanchor[from=natoapp6c unknown air]{south}
\inheritanchor[from=natoapp6c unknown air]{upper}
\inheritanchor[from=natoapp6c unknown air]{lower}
\inheritanchor[from=natoapp6c unknown air]{left}
\inheritanchor[from=natoapp6c unknown air]{right}
\inheritanchor[from=natoapp6c unknown air]{center}
\inheritbackgroundpath[from=natoapp6c unknown land]
\inheritbehindforegroundpath[from=natoapp6c unknown land]
}%
```
The unknown sub surface command.

\pgfdeclareshape{natoapp6c unknown sub surface}{
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor[from=natoapp6c unknown land]{inner north east}
\inheritanchor[from=natoapp6c unknown land]{inner north west}
\inheritanchor[from=natoapp6c unknown land]{inner south west}
\inheritanchor[from=natoapp6c unknown land]{inner south east}
\inheritanchor[from=natoapp6c unknown land]{south east}
\inheritanchor[from=natoapp6c unknown land]{south west}
\inheritanchor[from=natoapp6c unknown land]{south}
\inheritanchor[from=natoapp6c unknown land]{west}
\inheritanchor[from=natoapp6c unknown land]{east}
\inheritanchor[from=natoapp6c unknown land]{upper}
\inheritanchor[from=natoapp6c unknown land]{lower}
\inheritanchor[from=natoapp6c unknown land]{left}
\behindforegroundpath{\begin{n@to@pp@unknown@init}
\innersun@pp@northeast \wg@tmpa=\pgf@x%
\begin{n@to@pp@stroketofill}
\pgfpathmoveto{\pgfpoint{\wg@tmpa}{-\wg@tmpa}}
\n@to@pp@unknown@right %
\n@to@pp@unknown@top %
\n@to@pp@unknown@left %
\pgfusepath{stroke,clip} %
%\northeast\wg@tmpa=\pgf@y\wg@tmpb=\pgf@y
\advance\wg@tmpb-\n@to@pp@space@h %\notop\\
\pgfpathmoveto{\pgfpoint{\radius}{\wg@tmpa}}
\pgfpathlineto{\pgfpoint{-\radius}{\wg@tmpa}}
\pgfpathlineto{\pgfpoint{-\radius}{\wg@tmpb}}
\pgfpathlineto{\pgfpoint{\radius}{\wg@tmpb}}
\pgfclosepath%
\pgfusepath{fill}
\end{n@to@pp@stroketofill} %\notop
\end{n@to@pp@unknown@init}}
}
5.6.11 Echelons

Dimensions

\def\n@to@pp@e@y{.12}
\def\n@to@pp@e@yy{.24}

Paths as macros

\def\n@to@pp@e@d#1{($(#1*\n@to@pp@e@y,0)$) circle(0.09)}
\def\n@to@pp@e@b#1{($(#1*\n@to@pp@e@y,-\n@to@pp@e@y)$) -- ($(#1*\n@to@pp@e@y,\n@to@pp@e@y)$)}
\def\n@to@pp@e@x#1{($(-\n@to@pp@e@y,-\n@to@pp@e@y)+(#1*\n@to@pp@e@y,0)$)++(\n@to@pp@e@yy,\n@to@pp@e@yy)}
\def\n@to@pp@e@y#1{($(-\n@to@pp@e@y,\n@to@pp@e@y)+(#1*\n@to@pp@e@y,0)$)+++(\n@to@pp@e@yy,\n@to@pp@e@yy)}

Pictures
\tikzset{
5.6.12 Text on symbols

/tikz/natoapp6c/normal text  
tikz/natoapp6c/squashed text  
tikz/natoapp6c/small text  
tikz/natoapp6c/small squashed text
NATO App6 does not specify any particular font for text symbols (main, modifiers, or amplifiers) but here we choose to use TeX Gyro Heros (a Gothic font, i.e., Helvetica-like).

17135 \newcommand\n@to@ppfont[2][b]{%  
17136 \fontencoding{T1}\fontfamily{qhv}\fontseries{#1}\fontsize{#2}{0}\selectfont}  
17137 \tikzset{%  
17138 natoapp6c/text/.style={%  
17139 shape=rectangle,%  
17140 draw=none,%  
17141 fill=none,%  
17142 transform shape,%  
17143 anchor=center},  
17144 natoapp6c/normal text/.style={font=\n@to@ppfont{12}},  
17145 natoapp6c/squashed text/.style={font=\n@to@ppfont[bc]{12}},  
17146 natoapp6c/small text/.style={font=\n@to@ppfont{10}},  
17147 natoapp6c/squashed small text/.style={font=\n@to@ppfont[bc]{10}},  
17148 }

\n@to@pp@text@normal
\n@to@pp@text@squashed
\n@to@pp@text@small
\n@to@pp@text@smallsquashed

These macros are short-hands for making a node at (0,0) in the local scope.

17149 \newcommand\n@to@pp@text@normal[2][] {%  
17150 \node[natoapp6c/text,natoapp6c/normal text,#1][#2]}  
17151 \newcommand\n@to@pp@text@squashed[2][] {%  
17152 \node[natoapp6c/text,natoapp6c/squashed text,#1][#2]}  
17153 \newcommand\n@to@pp@text@small[2][] {%  
17154 \node[natoapp6c/text,natoapp6c/small text,#1][#2]}  
17155 \newcommand\n@to@pp@text@smallsquashed[2][] {%  
17156 \node[natoapp6c/text,natoapp6c/squashed small text,#1][#2]}

5.6.13 Text natoapp6c namespace

/natoapp6c

Here, we set up the key path /natoapp6c

17157 \def\natoapp\report{}  
17158 \tikzset{  
17159 /natoapp6c/.search also={/tikz},  
17160 /natoapp6c/.cd,  
17161 }

Choices of faction, command, and echelon
The keys id, specfac, cmd, and ech are internal keys used to store the choice of faction, command, and echelon, respectively, in.

\begin{verbatim}
	noapp6c/id
	noapp6c/fac
	noapp6c/cmd
	noapp6c/ech

\end{verbatim}

Choice of \textit{faction}. This is limited to predefined values. The choice is stored in the key \texttt{natoapp6c/fac}.

\begin{verbatim}
	noapp6c/faction

\end{verbatim}

Choice of \textit{command}. This is limited to predefined values. The choice is stored in the key \texttt{natoapp6c/cmd}.

\begin{verbatim}
	noapp6c/command

\end{verbatim}
The various parts of the symbols. The keys upper and lower are aliases for top and bottom, respectively. The choices are stored in macros

\newif\ifnatoapp@decoy
\natoapp@decoyfalse
\tikzset{
\natoapp6c/.cd,
\natoapp@decoytrue
\natoapp6c/command/.style={cmd=none},
\natoapp6c/echelon

\tikzset{
\natoapp6c/.cd,
\echelon/.is choice,
\echelon/none/.style={ech=},
\echelon/team/.style={ech=},
\echelon/squad/.style={ech=squad},
\echelon/section/.style={ech=section},
\echelon/platoon/.style={ech=platoon},
\echelon/company/.style={ech=company},
\echelon/battalion/.style={ech=battalion},
\echelon/regiment/.style={ech=regiment},
\echelon/brigade/.style={ech=brigade},
\echelon/division/.style={ech=division},
\echelon/corps/.style={ech=corps},
\echelon/army/.style={ech=army},
\echelon/army group/.style={ech=army group},
\echelon/theatre/.style={ech=theatre},
\echelon/command/.style={ech=command},
\echelon/dummy/.style={ech=dummy},
}\natoapp6c/main
\natoapp6c/left
\natoapp6c/right
\natoapp6c/top
\natoapp6c/bottom
\natoapp6c/below
\natoapp6c/frame

The various parts of the symbols. The keys upper and lower are aliases for top and bottom, respectively. The choices are stored in macros

\newif\ifnatoapp@decoy
\natoapp@decoyfalse
\tikzset{
\natoapp6c/.cd,
\natoapp@decoytrue
\natoapp6c/command/.style={cmd=none},
\natoapp6c/echelon

\tikzset{
\natoapp6c/.cd,
\echelon/.is choice,
\echelon/none/.style={ech=},
\echelon/team/.style={ech=},
\echelon/squad/.style={ech=squad},
\echelon/section/.style={ech=section},
\echelon/platoon/.style={ech=platoon},
\echelon/company/.style={ech=company},
\echelon/battalion/.style={ech=battalion},
\echelon/regiment/.style={ech=regiment},
\echelon/brigade/.style={ech=brigade},
\echelon/division/.style={ech=division},
\echelon/corps/.style={ech=corps},
\echelon/army/.style={ech=army},
\echelon/army group/.style={ech=army group},
\echelon/theatre/.style={ech=theatre},
\echelon/command/.style={ech=command},
\echelon/dummy/.style={ech=dummy},
}\natoapp6c/main
\natoapp6c/left
\natoapp6c/right
\natoapp6c/top
\natoapp6c/bottom
\natoapp6c/below
\natoapp6c/frame

The various parts of the symbols. The keys upper and lower are aliases for top and bottom, respectively. The choices are stored in macros

\newif\ifnatoapp@decoy
\natoapp@decoyfalse
\tikzset{
\natoapp6c/.cd,
\natoapp@decoytrue
\natoapp6c/command/.style={cmd=none},
\natoapp6c/echelon

\tikzset{
\natoapp6c/.cd,
\echelon/.is choice,
\echelon/none/.style={ech=},
\echelon/team/.style={ech=},
\echelon/squad/.style={ech=squad},
\echelon/section/.style={ech=section},
\echelon/platoon/.style={ech=platoon},
\echelon/company/.style={ech=company},
\echelon/battalion/.style={ech=battalion},
\echelon/regiment/.style={ech=regiment},
\echelon/brigade/.style={ech=brigade},
\echelon/division/.style={ech=division},
\echelon/corps/.style={ech=corps},
\echelon/army/.style={ech=army},
\echelon/army group/.style={ech=army group},
\echelon/theatre/.style={ech=theatre},
\echelon/command/.style={ech=command},
\echelon/dummy/.style={ech=dummy},
}\natoapp6c/main
\natoapp6c/left
\natoapp6c/right
\natoapp6c/top
\natoapp6c/bottom
\natoapp6c/below
\natoapp6c/frame

The various parts of the symbols. The keys upper and lower are aliases for top and bottom, respectively. The choices are stored in macros
5.6.14 The natoapp6c styles

This key sets up a node to make a NATO App6(c) symbol. The key takes a single argument which in turn must contain key–value pairs in the /natoapp6c (or /tikz) namespace(s). We set the shape parameter of the node, and calls the passed keys in the /natoapp6c namespace to set-up elements of the chit.

\begin{tikzpicture}
\def\natoapp6c{\pgfkeys{/tikz/transform shape, /tikz/shape=natoapp6c}}
\end{tikzpicture}
We define a counter to set-up unique names for symbol nodes.

\newcounter{natoappid}\setcounter{natoappid}{0}

5.6.15 The \texttt{natoapp6c} shape

We define an \texttt{if} to allow us to detect if something is rendered below the frame

\newif\ifn@to@pp@below\n@to@pp@belowfalse\%
\newif\ifn@to@pp@mod\n@to@pp@modfalse\%

Next, we define the mother shape of NATO App6(c) nodes. This is a composite node with sub-nodes for the various parts (including the frame) of the symbol.

It is quite complex so we will go through the implementation in bits.

First, we make some saved anchors (the centre) and macros (identifier, frame type, and frame options).

\pgfdeclarerectangle{natoapp6c}{%}
\savedanchor{center}{\pgf@x=0cm\pgf@y=0cm}
\savedmacro{id}{%\n@to@pp@dbg{3}{NATO App6(c) id (set): \meaning
natoapp@id}\@ifundefined{natoapp@id}{\let
natoapp@id\pgfutil@empty}{}%\ifx
natoapp@id\pgfutil@empty\relax%\wg@r@ndom@id%\edef\id{natoapp6c\wg@uuid}%%\else%
\edef\id{natoapp@id}%%\fi%\n@to@pp@dbg{3}{NATO App6(c) id: \meaning\id}}%
\savedmacro{frameshape}{%\let\frameshape\pgfutil@empty%\@ifundefined{natoapp@cmd}{\def\frameshape{base}}{%\edef\frameshape{natoapp@cmd}\@ifundefined{natoapp@fac}{\def\frameshape{base}}{%\edef\frameshape{natoapp@fac\space\frameshape}}}%%\n@to@pp@dbg{3}{NATO App6(c) Frame shape: \meaning\frameshape}}%
\savedmacro[frameopt]{%\let\frameopt\pgfutil@empty%\@ifundefined{natoapp@frame}{\def\frameopt{}}{%\edef\frameopt{natoapp@frame}}}%
\n@to@pp@dbg{3}{NATO App6(c) Frame options: \meaning\frameopt}%
}

Then we define a number of regular anchors

\anchor{center} {\center}
The remaining anchors depend on the shape being used. We reference the anchors of the embedded node of the frame.

```
\anchor\{north east\}{\wg\subanchor{M\id}{north east}}
\anchor\{north west\}{\wg\subanchor{M\id}{north west}}
\anchor\{south east\}{\wg\subanchor{M\id}{south east}}
\anchor\{south west\}{\wg\subanchor{M\id}{south west}}
\anchor\{north\}{\wg\subanchor{M\id}{north}}
\anchor\{west\}{\wg\subanchor{M\id}{west}}
\anchor\{south\}{\wg\subanchor{M\id}{south}}
\anchor\{east\}{\wg\subanchor{M\id}{east}}
\anchor\{upper\}{\wg\subanchor{M\id}{upper}}
\anchor\{lower\}{\wg\subanchor{M\id}{lower}}
```

The next two anchors are a little funny.

```
\anchor\{echelon\}{
\n@to@pp@dbg{3}{NATO App6(c) get echelon anchor}\%
\wg\subanchor{M\id}{north}\%
\wg@tmpa=\n@to@pp@e@y cm\%
\advance\pgf@y\wg@tmpa\%
}\%
\anchor\{below\}{
\n@to@pp@dbg{3}{NATO App6(c) get below anchor}\%
\wg\subanchor{M\id}{south}\%
\wg@tmpa=\n@to@pp@e@y cm\%
\advance\pgf@y-\wg@tmpa}
```

All right, so time to make the actual frame. Note that we do this in a ‘behind’ path so we can actually draw stuff. First, we flag that we’re not in a modifier, nor in the ‘below’ part.

```
\behindbackgroundpath{\%
\n@to@pp@dbg{3}{NATO App6(c) background path: \meaning\id
^^J ID: \meaning\natoapp@id
^^J Faction: \meaning\natoapp@fac
^^J Command: \meaning\natoapp@cmd
^^J Echelon: \meaning\natoapp@ech
^^J Main: \meaning\natoapp@main
^^J Left: \meaning\natoapp@left
^^J Right: \meaning\natoapp@right
^^J Upper: \meaning\natoapp@upper
^^J Lower: \meaning\natoapp@lower
^^J Below: \meaning\natoapp@below
^^J Shape: \meaning\frameshape
^^J Options: \meaning\frameopt}
\natoapp@report
\n@to@pp@modfalse
\n@to@pp@belowfalse
```

If the symbol is empty, then do nothing.

```
\ifx\frameshape\pgfutil@empty\%
\n@to@pp@dbg{2}{NATO App6(c) has no frame!}\%
\else
```

400
We start a scope because we want to do some clipping here. Then, we use the frame to clip the remaining part. Note that we do this via a node which we give the identifier \( M \). Various elements of the symbol can then refer to this shape to define paths, etc.

```latex
\begin{scope}
\pgfinterruptboundingbox
\message{^^JClipping to NATO App6(c) shape}
\n@to@pp@iscliptrue%
\n@to@pp@dbg{2}{NATO App6(c) frame node \( M \) (clip)}
\pgfnode{natoapp6c \frameshape}{center}{}{M}{\pgfusepath{clip}}
\n@to@pp@iscliptfalse%
```

Next, we should see if we need to fill the frame. We do that by expanding the passed frame key-values in a scope, and then get the fill colour.

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

If the fill colour is not \( \relax \), then we fill the frame. Note that this is done in the background, so when we draw in the foreground we will render on top of the fill.

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

Now we need to render some of the elements of the symbol. We start with the main elements. We can specify many main elements (to make composite symbols).

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

The next thing is to render the various modifiers. We start by flagging this globally.

```latex
\n@to@pp@modtrue
\n@to@pp@dbg{2}{NATO App6(c) modifiers}
```

Below we render the lower, upper, left, and right elements. This is all done in the same way. Note that the elements positions are dictated by anchors of the frame shape (via shape identifier \( M \)).

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

\textit{Note that this is done in the background, so when we draw in the foreground we will render on top of the fill.}

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

Now we need to render some of the elements of the symbol. We start with the main elements. We can specify many main elements (to make composite symbols).

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

The next thing is to render the various modifiers. We start by flagging this globally.

```latex
\n@to@pp@modtrue
\n@to@pp@dbg{2}{NATO App6(c) modifiers}
```

Below we render the lower, upper, left, and right elements. This is all done in the same way. Note that the elements positions are dictated by anchors of the frame shape (via shape identifier \( M \)).

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```

```latex
\edef\tmp@opt{[\frameopt]}
\expandafter\scope\tmp@opt
% Get fill color {possibly from frame key}
\expandafter\let\expandafter\tmp@fill\csname\string\color@pgffillcolor\endcsname%
\ifx\tmp@fill\relax\else
\n@to@pp@dbg{2}{NATO App6(c) frame fill}
\pgfnode{natoapp6c \frameshape}{center}{}{}{\pgfusepath{fill}}%
\fi%
\endscope%
```
That concludes rendering most of the symbol. We have not put in the echelon, below element, or drawn the frame yet. That we will do on the foreground path.

In the foreground ‘behind’ path we render the echelon, below element, and draw the frame.

We check if we have a frame. If not, stop.

We want to draw the rest of the symbol as a part of the frame, so we expand the frame options in a scope.
First thing in this scope is to draw the actual frame. Again, this is done via a node with the right shape. Note that we label this node as $M(id)$ so we may refer to it later on.

17396    \n@to@pp@dbg{2}{NATO App6(c) inner node 'M\id' ===}
17397    \pgfnode{natoapp6c \frameshape}{center}{\{M\id\} \pgfusepath{stroke}}

If the user gave an echelon, then put that in. Note that echelons are limited to predefined values.

17398    % Put in the echelon
17399    \@ifundefined{natoapp@ech}{% \iffalse\natoapp@ech\pgfutil@empty\else%
17400    \def\args{echelon=\natoapp@ech}
17401    \expandafter\wg@pic\args\@endwg@pic%
17402    \n@to@pp@e@y cm\{\natoapp6c/echelon\}
17403    \fi%
17404    }

If the user want something under the frame, put that in.

17405    % Put in stuff below main
17406    \@ifundefined{natoapp@below}{% \iffalse\natoapp@below\pgfutil@empty\else%
17407    \n@to@pp@belie\n@true
17408    \begin{scope}
17409    \wg@pic@all{\natoapp@below}{natoapp6c/s/}{\$(M.south)+(0,-\n@to@pp@e@yy)$}{\natoapp6c/below}
17410    \end{scope}
17411    \n@to@pp@belie\n@false}

If the decoy flag was set, we draw that.

17412    ifnatoapp@decoy
17413    \scope[dash pattern=on 3\pgflinewidth off 2\pgflinewidth]%
17414    \n@to@pp@dbg{1}{Drawing decoy modifier}%
17415    \wg@sub@nchor{M\id}{north east}
17416    \wg@tmpa=\pgf@x%
17417    \wg@tmpb=\pgf@y%
17418    \pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
17419    \pgfpathmoveneto{\pgfqpoint{\wg@temp}{\wg@tmpb}}%
17420    \pgfpathlineto{\n@to@pp@e@yy cm\n@to@pp@e@yy cm}%
17421    \advance\wg@temp\n@to@pp@e@yy cm%
17422    \advance\wg@tmpc\wg@tmpb%
17423    \pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpc}}%
17424    \pgfpathlineto{\pgfqpoint{-\wg@temp}{\wg@tmpb}}%
17425    \pgfusepath{stroke}%
17426    \endscope%
17427    \fi%
17428    \endscope
17429    \fi%
17430    \\fi%
17431    \fi%
17432    }
17433    }

That finished the shape for NATO App6(c) symbols. We could stop here, but for convenience we define a wrapper macro.
5.6.16 The \natoapp wrapper macro

\natoapp

This is a wrapper macro for inserting a node with a NATO App6(c) symbol in it. The syntax of the macro is

\natoapp[\langle natoapp6c options \rangle]\langle position \rangle[\langle identifier \rangle];

Note that the trailing semi-colon (;) is optional.

This macro forwards to \n@toapp.

This macro takes care to parse the location argument — if any. It forwards to \n@toapp.

\n@toapp

This is the main work-horse of the wrapper. It makes a node with the shape natoapp6c passing the relevant parameters. The syntax of the macro is

\n@toapp

5.6.17 Macros for markings

\natoappmark

A macro for making NATO App6(c) markings.

\natoappmark
Some specific NATO App6(c) markers.

5.6.18 Utility macros used in the symbols

Here, we define the main symbols used when making markers. Since some of these symbols share code, we will create some regular \TeX macros to hold the path definitions. This is by far the simplest way of storing just the path specifications.
Corps support, base

Special placeholder for symbols To Be Done.

Symbols used when defining weaponry
natoapp6c/s/type

(Weight) class of weapons: light, medium, heavy

\tikzset{
  pics/natoapp6c/s/type/.is choice,
  pics/natoapp6c/s/type/light/.style={
    code={
      \path [fill=pgfstrokecolor,pic actions] (-0.2, -0.12) rectangle (.2,-.08);},
    pics/natoapp6c/s/type/medium/.style={
      code={
        \path [fill=pgfstrokecolor,pic actions]
        (-0.2, -0.12) rectangle (.2,-.08)
        (-0.2, -0.22) rectangle (.2,-.18);},
    pics/natoapp6c/s/type/heavy/.style={
      code={
        \path [fill=pgfstrokecolor,pic actions]
        (-0.2, -0.12) rectangle (.2,-.08)
        (-0.2, -0.22) rectangle (.2,-.18)
        (-0.2, -0.32) rectangle (.2,-.28);},
    pics/natoapp6c/s/type/vlight/.style={
      code={
        \path [fill=pgfstrokecolor,pic actions]
        (-.025,-.025) rectangle (.025,.25);},
    pics/natoapp6c/s/type/vmedium/.style={
      code={
        \path [fill=pgfstrokecolor,pic actions]
        (-.075,-.025) rectangle (-.025,.25)
        (.025, -.025) rectangle (.075,.25);},
    pics/natoapp6c/s/type/vheavy/.style={
      code={
        \path [fill=pgfstrokecolor,pic actions]
        (-.125,-.025) rectangle (-.075,.25)
        (-.025,-.025) rectangle (.025,.25)
        (.075,-.025) rectangle (.125,.25);},
  }\tikzset{...}
5.6.20 The symbols

Next, we define all the symbols. Note that we define them all as if they are in the main section of the symbol, since top, bottom, and below symbols are automatically scaled.

408
natoapp6c/s/air defence

\tikzset{
  natoapp6c/s/air defence/.pic={%
  \ifx\natoapp@fac\natoapp@friendly%
  \def\natoapp@opt{[out=90,in=90,looseness=.675]%}
  \else\ifx\natoapp@fac\natoapp@neutral%
  \def\natoapp@opt{[out=90,in=90,looseness=1]%}
  \else%
  \def\natoapp@opt{[out=45,in=135,looseness=1.5]%}
  \fi\fi%
  \edef\natoapp@path{(M.south west) to \natoapp@opt (M.south east)}
  \path[draw] \natoapp@path;},
}

natoapp6c/s/air strip

\tikzset{
  natoapp6c/s/air strip/.pic={%
  \path[fill=pgfstrokecolor] (-.4,-.1) rectangle(.4,0);
  \path[rotate=45,fill=pgfstrokecolor] (-.4,0) rectangle (.4,.1);
  }%
}

natoapp6c/s/air traffic

\tikzset{
  natoapp6c/s/air traffic/.pic={%
  \path[fill=pgfstrokecolor]
  (0.33,0.21)--(0.33, -0.21)--(-0.33, 0.21)--(-0.33,-0.21)--cycle;},
}

natoapp6c/s/airship

\tikzset{
  natoapp6c/s/airship/.pic={%
  % \path (0.45, 0.175) rectangle (-0.45, -0.175);
  \path[pic actions] (0, 0) ellipse (0.45 and 0.15);
  \begin{scope}
  \clip (0, 0) ellipse (0.45 and 0.15) [reverseclip];
  \path[pic actions]
  (0.2,0)--(0.3,0.175)--(0.4,0.175)--(0.375,0)
  --(0.4,-0.175)--(0.3, -0.175)--cycle;
  \end{scope}
}

409
\end{scope},
}

\tikzset{%
\n@to@pp@neutral\natoapp@fac%
\draw (0,-0.05) arc(0:180:0.15);
\draw (0,-0.05) arc(180:0:0.15);
\else%
\draw (0,-0.05) arc(0:180:0.2);
\draw (0,-0.05) arc(180:0:0.2);
\fi},
}

\tikzset{%
\natoapp6c/s/ammunition/.pic={
\path[draw]
(0.175,-0.175)--(-0.175,-0.175)
(0.125,-0.175)--(0.125, 0) to[out=90,in=90,looseness=2.75]
(-0.125, 0)--(-0.125, -0.175);},
}

\tikzset{%
\natoapp6c/s/amphibious/.pic={
\def\n@to@pp@tmp{0}
\ifn@to@pp@below\def\n@to@pp@tmp{-0.1}\fi
\ifn@to@pp@mod
\path[draw,shift={(0,\n@to@pp@tmp)}](1.21,0)
to[out=-90,in=-90, looseness=2.25] (1.05, 0)
to[out= 90,in= 90, looseness=2.25] (0.89, 0)
to[out=-90,in=-90, looseness=2.25] (0.73, 0)
to[out= 90,in= 90, looseness=2.25] (0.57, 0)
to[out= 90,in= 90, looseness=2.25] (0.41, 0)
to[out= 90,in= 90, looseness=2.25] (0.25, 0)
to[out= 90,in= 90, looseness=2.25] (0.08, 0)
to[out= 90,in= 90, looseness=2.25] (-0.08, 0)
to[out= 90,in= 90, looseness=2.25] (-0.25, 0)
to[out= 90,in= 90, looseness=2.25] (-0.41, 0)
to[out= 90,in= 90, looseness=2.25] (-0.57, 0)
to[out= 90,in= 90, looseness=2.25] (-0.73, 0)
to[out= 90,in= 90, looseness=2.25] (-0.89, 0)
to[out= 90,in= 90, looseness=2.25] (-1.05, 0)
to[out= 90,in= 90, looseness=2.25] (-1.21, 0)
}
\else
\path[shift={(0,\n@to@pp@tmp)}](0.73, 0)
to[out= 90,in= 90, looseness=2.25] (0.57, 0)
to[out=-90,in=-90, looseness=2.25] (0.41, 0)
to[out= 90,in= 90, looseness=2.25] (0.25, 0)
to[out=-90,in=-90, looseness=2.25] (0.08, 0)
to[out= 90,in= 90, looseness=2.25] (-0.08, 0)
to[out=-90,in=-90, looseness=2.25] (-0.25, 0)
to[out= 90,in= 90, looseness=2.25] (-0.41, 0)
to[out=-90,in=-90, looseness=2.25] (-0.57, 0)
to[out= 90,in= 90, looseness=2.25] (-0.73, 0)
\fi
\}
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(0.15, 0.05) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(-0.15, 0.2) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(0.15, 0.2) --
(-0.15, 0.2) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(-0.15, 0.2) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(-0.15, 0.2) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(-0.15, 0.2) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

\tikzset{%
\path[draw,fill=pgfstrokecolor]
(-0.15, 0.2) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);,
\}

411
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/artillery/.pic={
\path[pic actions] circle(0.2);},
}
\natoapp6c/s/anti tank anti armour
\tikzset{
\natoapp6c/s/anti tank anti armour/.pic={%}
\path[natoapp@fac=n@to@pp@unknown\%]
\path[pic actions] (225:.5)--(M.north)--(315:.5);
\else%
\path[pic actions] (M.south west)--(M.north)--(M.south east);%
\fi},
}
\natoapp6c/s/antenna
\tikzset{
\natoapp6c/s/antenna/.pic={
\path[draw]
(0, -0.3) -- (0, 0.3) (-0.125, 0.3) -- (0, 0.2) -- (0.125, 0.3);
},
}
\natoapp6c/s/armoured
\tikzset{
\natoapp6c/s/armoured/.pic={
\path[draw]
(-0.275,0.2) arc(90:270:0.2)--(0.275, -0.2) arc(270:450:0.2)--cycle;,}
}
\natoapp6c/s/armoured fighting vehicle
\tikzset{
\natoapp6c/s/armoured fighting vehicle/.pic={
\path[fill=pgfstrokecolor] (-.4,-.2) rectangle (-.3,.2) (.3,-.2) rectangle (.4,.2);
\path[pic actions] (-.3,0) -- (0,.2) -- (.3,0) -- (0,-.2) -- cycle;,}
}
\natoapp6c/s/armoured personnel carrier
\tikzset{
\natoapp6c/s/armoured personnel carrier
\end{tikzpicture}

412
\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/bar};
\end{tikzpicture}

\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/base};
\end{tikzpicture}

\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/bicycle equipped};
\end{tikzpicture}

\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/boat};
\end{tikzpicture}

\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/booby trap};
\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
    natoapp6c/s/booby trap/.pic={
        \path[dashed, fill=pgfstrokecolor] (0, 0.2) ellipse(0.2 and 0.065);
    },
\tikzset{
    natoapp6c/s/bottomed/.pic={
        \path[draw,fill=pgfstrokecolor] (-0.33,.1) rectangle(0.33,.2);
    },
\tikzset{
    natoapp6c/s/bridge/.is choice,
\tikzset{
    natoapp6c/s/bridge/fixed/.style={
        \pic{natoapp6c/s/bridge=none};\pic{natoapp6c/s/type=vlight};
    },
\tikzset{
    natoapp6c/s/bridge/folding/.style={
        \pic{natoapp6c/s/bridge=none};
        \path[draw] (.1,-.2) -- (-.1,-.2) -- (-.1,.2) -- (.1,.2);
    },
\tikzset{
    natoapp6c/s/bridge/hollow/.style={
        \pic{natoapp6c/s/bridge=none};
        \path[draw] (.1,-.2) -- (-.1,-.2) -- (-.1,.2) -- (.1,.2) -- cycle;
    },
\tikzset{
    natoapp6c/s/bridge/.default=none,
\end{scope},
\end{scope},
\begin{scope}
\clip (0, -0.2) ellipse(0.2 and 0.065) [reverseclip];
\path[dashed, fill=pgfstrokecolor] (-0.2, -0.2) -- (0, 0.2) -- (0.2, -0.2);
\end{scope},
\end{scope}}
\end{tikzpicture}
\begin{itemize}
\item \texttt{\textbackslash tikzset{}}
\item \texttt{natoapp6c/s/carrier/.pic={}}
\item \texttt{\textbackslash pic{natoapp6c/s/warfare vessel};}
\item \texttt{\textbackslash path[draw,fill=pgfstrokecolor]}
\item \texttt{(-0.15, 0.05) --}
\item \texttt{(-0.15, 0.2) --}
\item \texttt{(-0.3, 0.2) --}
\item \texttt{(-0.3, 0.05) -- cycle;},
\end{itemize}

\begin{itemize}
\item \texttt{\textbackslash tikzset{}}
\item \texttt{natoapp6c/s/chemical biological radiological nuclear/.pic={}}
\item \texttt{\path[draw,fill=pgfstrokecolor] (-0.29,0.1) circle(0.096) (0.29,0.1) circle(0.096);}
\item \texttt{\path[pic actions] (0.15,-0.2) arc(0:90:0.45 and 0.375)}
\item \texttt{(-0.15,-0.2) arc(180:90:0.45 and 0.375);},
\end{itemize}

\begin{itemize}
\item \texttt{\textbackslash tikzset{}}
\item \texttt{natoapp6c/s/civilian military cooperation/.pic=}\texttt{\textbackslash pic{}}
\item \texttt{\path[draw] (.375,.2)--(-.375,.2)--(-.375,-.025)}
\item \texttt{to[in=270, out=270, looseness=0.75] (.375,-.025)--cycle;},
\end{itemize}

\begin{itemize}
\item \texttt{\textbackslash tikzset{}}
\item \texttt{natoapp6c/s/civilian police/.pic=}\texttt{\textbackslash pic{}}
\item \texttt{\path[draw] (0.225, 0.2)}
\item \texttt{to[in=270, out=270, looseness=3] (-0.225, 0.2)}
\item \texttt{to [in=270, out=270, looseness=1.5] (0.0,2)}
\item \texttt{to [in=270, out=270, looseness=1.5] (0.225, 0.2) -- cycle;},
\end{itemize}

\begin{itemize}
\item \texttt{\textbackslash tikzset{}}
\item \texttt{natoapp6c/s/civilian telecommunications/.pic={}}
\item \texttt{\path[draw] (0.075, -0.2){[line join=bevel] -- (0, 0.1) -- (-0.075, -0.2)}}
\item \texttt{(0.085, -0.05) -- (-0.085, -0.05)}
\end{itemize}
\begin{tikzpicture}

\tikzset{
  natoapp6c/s/coast guard vessel/.pic={
    \pic[draw] {natoapp6c/s/ship};
    \path[\pic actions] (0.15, 0.05) -- (0, -0.2) (0.2, 0.05)--(0.05, -0.2);},
}

\begin{scope}[xshift=-4.5, yshift=-5]

\tikzset{
  natoapp6c/s/combatant/.pic={
    \begin{scope}
    \path[\pic actions]
    (0.3213,0.0534) .. controls (0.3186,0.0295) and (0.3072,0.0136) ..
    (0.2925,0.0063) .. controls (0.2777,-0.0010) and (0.2605,0.0001) ..
    (0.2461,0.0068) .. controls (0.2317,0.0136) and (0.2198,0.0265) ..
    (0.2163,0.0433) .. controls (0.2147,0.0513) and (0.2150,0.0601) ..
    (0.2179,0.0694) .. controls (0.1304,0.1129) and (0.0223,0.1961) ..
    (0.0013,0.3209) .. controls (0.0601,0.1809) and (0.1770,0.0912) ..
    (0.3213,0.0534) -- cycle
    \end{scope}
    \begin{scope}
    \path[\pic actions]
    (0.2304,0.0633) .. controls (0.2287,0.0570) and (0.2287,0.0513) ..
    (0.2298,0.0461) .. controls (0.2323,0.0340) and (0.2409,0.0245) ..
    (0.2520,0.0193) .. controls (0.2630,0.0141) and (0.2760,0.0135) ..
    (0.2864,0.0186) .. controls (0.2932,0.0220) and (0.2992,0.0277) ..
    (0.3033,0.0370) .. controls (0.2845,0.0413) and (0.2597,0.0498) ..
    (0.2304,0.0633) -- cycle
    \end{scope}
    (0.1785,0.1137) .. controls (0.2446,0.1612) and (0.3061,0.2300) ..
    (0.3214,0.3209) .. controls (0.2864,0.2377) and (0.2310,0.1723) ..
    (0.1614,0.1249) .. controls (0.1443,0.1138) and (0.1011,0.0871) ..
    (0.0014,0.0535) .. controls (0.0441,0.0295) and (0.0154,0.0136) ..
    (0.0302,0.0063) .. controls (0.0449,-0.0010) and (0.0621,0.0001) ..
    (0.0765,0.0069) .. controls (0.0909,0.0137) and (0.1028,0.0265) ..
    (0.1063,0.0433) .. controls (0.1079,0.0513) and (0.1076,0.0602) ..
    (0.1047,0.0694) .. controls (0.1230,0.0785) and (0.1422,0.0893) ..
    (0.3213,0.0534) .. controls (0.3186,0.0295) and (0.3072,0.0136) ..
    (0.2925,0.0063) .. controls (0.2777,-0.0010) and (0.2605,0.0001) ..
    (0.2461,0.0068) .. controls (0.2317,0.0136) and (0.2198,0.0265) ..
    (0.2163,0.0433) .. controls (0.2147,0.0513) and (0.2150,0.0601) ..
    (0.2179,0.0694) .. controls (0.1304,0.1129) and (0.0223,0.1961) ..
    (0.0013,0.3209) .. controls (0.0601,0.1809) and (0.1770,0.0912) ..
    (0.3213,0.0534) -- cycle
    \end{scope}
\end{tikzpicture}
\begin{tikzpicture}
\begin{scope}
\path[draw] (0.275, 0.2) -- (-0.275, -0.2) (0.275, -0.2) -- (-0.275, 0.2);
\end{scope}
\end{tikzpicture}

natoapp6c/s/combined arms

\tikzset{
natoapp6c/s/combined arms/.pic={
\path[draw] pic {natoapp6c/s/armoured};
\path[draw] (0.275, 0.2) -- (-0.275, -0.2) (0.275, -0.2) -- (-0.275, 0.2);}.,
}

natoapp6c/s/computer system

\tikzset{
natoapp6c/s/computer system/.pic={
\path[draw,fill=pgfstrokecolor,pic actions]
(-.3, .28) rectangle ( .3, .3)
(-.3, -.18) rectangle ( .3, -.2)
(-.3, -.18) rectangle (-.3, .28)
( .3, -.18) rectangle ( .3, .28)
(-.3, -.3) rectangle ( .3, -.28)
(-.05,-.28) rectangle ( .05,-.18);},
}

natoapp6c/s/control

\tikzset{
natoapp6c/s/control/.pic={
\path[pic actions]
[[Stealth[inset=0pt,scale=0.5]]--[Stealth[inset=0pt,scale=0.5]]
(0, .2) -- (0, -.2);
\path[pic actions]
[[Stealth[inset=0pt,scale=0.5]]--[Stealth[inset=0pt,scale=0.5]]
(-.2, 0) -- (2, 0);},
}
\begin{verbatim}
17995 .3)--(.175,.35)--(-.175,.35)--(-.3,.3)
17996 (0,.2)--(0,.35);
17997 },
17998 }
\end{verbatim}
\begin{tikzpicture}[pic actions={
\path[draw] (-.3,-.25) rectangle(.3,0)
(-.3,0) -- ++(60:.28) -- ([shift=(120:.28)]0,.3) -- (.3,0);},
\path[fill=pgfstrokecolor] (0,0) circle(.2)
(0,0) -- (60:.3) -- (120:.3) -- cycle
(0,0) -- (-30:.3) -- (30:.3) -- cycle
(0,0) -- (150:.3) -- (210:.3) -- cycle
(0,0) -- (240:.3) -- (300:.3) -- cycle;
}\end{tikzpicture}
natoapp6c/s/fixed wing/.pic={
\path[pic actions]
(-0.36,0.125) arc (77:275:0.075 and 0.125) -- (0,0) -- cycle
( 0.36,0.125) arc (-275:-77:-0.075 and 0.125) -- (0,0)
--cycle;},
}

natoapp6c/s/flame thrower

tikzset{
  natoapp6c/s/flame thrower/.pic={
\path[pic actions]
(-0.1, -0.4) -- (-0.1, 0.3) to[out=90,in=90,looseness=2]
(0.1, 0.3) -- (0.1, 0.275);},
}

natoapp6c/s/floating

tikzset{
  natoapp6c/s/floating/.pic={
\path[draw]
(-0.5, 0.100) --
(-0.417, 0.242) --
(-0.333, 0.100) --
(-0.250, 0.242) --
(-0.167, 0.100) --
(-0.083, 0.242) --
( 0.0, 0.100) --
( 0.083, 0.242) --
( 0.167, 0.100) --
( 0.250, 0.242) --
( 0.333, 0.100) --
( 0.417, 0.242) --
( 0.5, 0.100);},

  pics/natoapp6c/s/surfaced/.style=natoapp6c/s/floating,
}

natoapp6c/s/food

tikzset{
  natoapp6c/s/food/.pic={
\path[pic actions]
( 0.075, 0.2) to[out=210, in=150, looseness=1]
( 0.075, -0.2) to[out=180, in=180, looseness=1.5]
( 0.075, 0.2) -- cycle;},
}
natoapp6c/s/fuel

\tikzset{
  natoapp6c/s/fuel/.pic={
    path[draw] (0,0) -- (135:.3) -- (45:.3) -- cycle (0,0) -- (0,-.3);},
}

natoapp6c/s/grenade launcher

\tikzset{
  pics/natoapp6c/s/grenade launcher/.is choice, 
  pics/natoapp6c/s/grenade launcher/none/.style={
    code={
      pic[draw]{natoapp6c/s/rifle};
      pic[draw]{natoapp6c/s/weapon=grenade launcher};
    },
  },
  pics/natoapp6c/s/grenade launcher/non lethal/.style={
    code={
      pic[draw]{natoapp6c/s/non lethal weapon};
      pic[draw]{natoapp6c/s/weapon=grenade launcher};
    },
  },
  pics/natoapp6c/s/grenade launcher/.default=none,
}

natoapp6c/s/graffiti

\tikzset{
  natoapp6c/s/graffiti/.pic={
    path[pic actions]
    arc (90:270:0.05)
    arc (450:270:0.05)
    arc (90:270:0.05)
    arc (450:270:0.05)
    arc (-0.05, 0.2)
    arc (90:270:0.05)
    arc (450:270:0.05)
    arc (90:270:0.05)
    arc (450:270:0.05);,
  }
}

natoapp6c/s/group

\tikzset{
  natoapp6c/s/group/.pic={
    path(-.23,.05) pic [draw,scale=.8] {natoapp6c/s/individual};
    path(0,.05) pic [draw,scale=.8] {natoapp6c/s/individual};
    path(.23,.05) pic [draw,scale=.8] {natoapp6c/s/individual};,
  }
}
\tikzset{
\pics/natoapp6c/s/gun/.is choice,
\pics/natoapp6c/s/gun/base/.style={
  code={
    \pic[draw]{natoapp6c/s/weapon=base};
    \pic[draw]{natoapp6c/s/weapon=top};
    \pic[draw]{natoapp6c/s/weapon=multi fire};},
\pics/natoapp6c/s/gun/air defence/.style={
  code={
    \pic[draw]{natoapp6c/s/gun/base};
    \pic[draw]{natoapp6c/s/weapon=air defence};},
\pics/natoapp6c/s/gun/anti tank/.style={
  code={
    \pic[draw]{natoapp6c/s/gun/base};
    \pic[draw]{natoapp6c/s/weapon/anti tank};},
\pics/natoapp6c/s/gun/direct/.style={
  code={
    \pic[draw]{natoapp6c/s/gun/base};
    \pic[draw]{natoapp6c/s/weapon=bottom};},
\pics/natoapp6c/s/gun/recoilless/.style={
  code={
    \pic[draw]{natoapp6c/s/rifle};
    \pic[yshift=-4,draw]{natoapp6c/s/weapon=multi fire};},
\pics/natoapp6c/s/gun/.default=direct,

\natoapp6c/s/headquarters
\tikzset{
\natoapp6c/s/headquarters/.pic={
  \path[pic actions] (\M.north west) -- ++(0,\M.north east) -- cycle;},

\natoapp6c/s/house
\tikzset{
\natoapp6c/s/house/.pic={
  \path[pic actions]
  (-.125,-.175) rectangle (.125,.075)
  (-.167,.075) -- (0,.225) -- (.167,.075) -- cycle;},

\begin{tikzpicture}
\tikzset{
  natoapp6c/s/jagged wave/.pic={
    \draw (0.3, -0.05) -- (0.2, 0.05) -- (0.1, -0.05) -- (0, 0.05) -- (-0.1, -0.05) -- (-0.2, 0.05) -- (-0.3, -0.05);
  },
}

\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
  natoapp6c/s/jam/.pic={
    \path[draw] (0.75, 0) to[out=90, in=90, looseness=2.25] (0.65, 0) to[out=-90,in=-90,looseness=2.25] (0.55, 0) to[out=90, in=90, looseness=2.25] (0.45, 0) to[out=-90,in=-90,looseness=2.25] (0.35, 0) to[out=90, in=90, looseness=2.25] (0.25, 0) to[out=-90,in=-90,looseness=2.25] (0.15, 0) to[out=90, in=90, looseness=2.25] (0.05, 0) to[out=-90,in=-90,looseness=2.25] (-0.05, 0) to[out=90, in=90, looseness=2.25] (-0.15, 0) to[out=-90,in=-90,looseness=2.25] (-0.25, 0) to[out=90, in=90, looseness=2.25] (-0.35, 0) to[out=-90,in=-90,looseness=2.25] (-0.45, 0) to[out=90, in=90, looseness=2.25] (-0.55, 0) to[out=-90,in=-90,looseness=2.25] (-0.65, 0) to[out=90, in=90, looseness=2.25] (-0.75, 0);
  },
}

\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
  natoapp6c/s/jamming/.pic={
    \path(0,.4) pic {natoapp6c/s/jam} (0,.26) pic {natoapp6c/s/jam};
  },
}

\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/jetski/.pic={
\path[pic actions]
( 0.3, -0.2) --
(-0.3, -0.2) --
(-0.35,-0.1) --
(-0.1, 0.2) --
( 0, 0.2) --
( 0, 0.1) --
(-0.05, 0.1) --
(-0.1, -0.05) --
( 0.3, -0.05) --
( 0.3, -0.2) -- cycle;
},
}
\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/killing/.pic={
\path[draw] (-.45,.25)--(.45,-.25);}
}
\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/labour/.pic={
\path[draw] (-.15,.2) -- (.15,.2) (0,.2) -- (0,0)
(-.15,0) -- ++(300:.3) -- ++(60:.3) -- cycle;},
}
\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/land mine/.is choice,
pics/natoapp6c/s/land mine/personnel/.style={
code={\pic[fill=pgfstrokecolor]{natoapp6c/s/land mine=none};
\path[pic actions] (135:0.35) -- (0, 0) -- (45:0.35);}},
pics/natoapp6c/s/land mine/tank/.style={
code={\pic[fill=pgfstrokecolor]{natoapp6c/s/land mine=none};}},
pics/natoapp6c/s/land mine/none/.style={
code={\path[pic actions] (0,0) circle(0.25);}},
pics/natoapp6c/s/land mine/.default=none,
}
\end{tikzpicture}
\begin{tikzpicture}

\pic [natoapp6c/s/medical] at (0,0) {};

\pic [natoapp6c/s/medical treatment] at (0,0) {};

\pic [natoapp6c/s/mine] at (0,0) {};

\pic [natoapp6c/s/mine clearing equipment] at (0,0) {};

\end{tikzpicture}
natoapp6c/s/mortuary affairs

\tikzset{
  natoapp6c/s/mortuary affairs/.pic={
    \path[draw] (-.1,-.2) rectangle (.1,.2)
    (0,-.17) -- (0,.17) (-.07,.1) -- (.07,.1);},
}

natoapp6c/s/mountain

\tikzset{
  natoapp6c/s/mountain/.pic={
    \path[draw,fill=pgfstrokecolor] (0,.2) -- ++(-60:.7) -- ++(180:.7) -- cycle;
  },
}

natoapp6c/s/naval

\tikzset{
  natoapp6c/s/naval/.pic={
    \begin{scope}[pic actions]
    \path[draw]
    (0,.13) circle (.08) (-.2,.04) -- (.2,.04) (0,.04)
    -- (0,-.25) (210:.25) arc (210:340:.25);
    \path[draw,shift=(210:.25),rotate=30] \arrow;
    \path[draw,shift=(340:.25),rotate=-30] \arrow;
    \end{scope},
  }
}

natoapp6c/s/navigation

\tikzset{
  natoapp6c/s/navigation/.pic={
    \path[draw]
    (.17,-.2) -- (0,.2) -- (-.17,-.2)
    ($(-180:.17)+(0,.05)$) arc[radius=.17,start angle=-180,end angle=0];},
}

natoapp6c/s/navy task

\tikzset{
  natoapp6c/s/navy task/.pic={
  },
}
(\texttt{natoapp6c/s/non combatant})

18515 \tikzset{
18516 natoapp6c/s/non combatant/.pic={
18517 \path\[draw,fill=pgfstrokecolor\]
18518 (-0.25, -0.2) -- (-0.25, 0.1) -- (-0.15, 0.2)
18519 ( 0.25, -0.2) -- ( 0.25, 0.1) -- ( 0.15, 0.2);},
18520 }

\begin{enumerate}
\item \texttt{natoapp6c/s/non lethal weapon}
\item \texttt{natoapp6c/s/nuclear}
\item \texttt{natoapp6c/s/observer}
\end{enumerate}
\begin{lstlisting}[language=TeX]
\begin{tikzpicture}
\node at (0,0) {\textbf{Orbiter Shuttle}};
\end{tikzpicture}
\end{lstlisting}
\tikzset{
  natoapp6c/s/poisoning/.pic={
    \path[pic actions] (0, 0.055) circle (0.145)
    (0.3, 0) -- (-0.3, -0.2)
    (-0.3, 0) -- (0.3, -0.2);},
}

\tikzset{
  natoapp6c/s/postal/.pic={
    \path[draw] (-.25,.25) -- (.08,.25)
    to [out=-90,in=120,looseness=1] (.25,-.25)
    to [out=150,in=-90,looseness=1] (-.25,.25);
  },
}

\tikzset{
  natoapp6c/s/printed media/.pic={
    \path[pic actions] (0.2, 0) -- (-0.2, 0)
    (0, 0.1) circle (0.085)
    (0, -0.1) circle (0.085);},
}

\tikzset{
  natoapp6c/s/psychological/.pic={
    \path[pic actions] (-.25,.15) -- (-.1,.15) -- (.1,.25)
    -- ++(0,-.5) -- (-.1,-.15) -- (-.25,-.15) -- cycle
    (.1,.15) -- (.25,.15)
    (.1,.05) -- (.25,.05)
    (.1,-.05) -- (.25,-.05)
    (.1,-.15) -- (.25,-.15);},
}

439
18644 \tikzset{%
18645  natoapp6c/s/quarry/.pic={%
18646    \path[draw] (-.2,-.2) -- (.18,.18) (.2,-.2) -- (-.18,.18)
18647    (25:.255) arc(25:65:.255)
18648    (115:.255) arc(115:155:.255);
18649    %((shift={((115:.08)})-.1,.1) arc (115:155:.08)
18650    %((shift={(70:.08)})-.1,.1) arc (70:110:.08);
18651  },
18652 }

18653 \tikzset{%
18654  natoapp6c/s/quartermaster/.pic={%
18655    \path[draw] (-.4,.1) -- (.1,.1) (.25,.1) circle(.15)
18656    (-.3,.1) -- (-.3,-.15) (-.15,.1) -- (-.15,-.15)
18657    (-.3,-.08) -- (-.15,-.08);%
18658 }

18659 \tikzset{%
18660  natoapp6c/s/radar/.pic=%
18661    \path[draw] (-.2,.2) arc (150:300:.25) (-.24,.01) -- (0,.2) --
18662    (0,0) -- (.2,.2);%
18663 }

18664 \tikzset{%
18665  natoapp6c/s/radio/.pic=%
18666    \path[draw] (-.2,.2) -- (.2,.2) arc (-.24,.01) -- (0,.2) --
18667    (.07,.2) -- (.13,.25) arc (.2,.2) (0,.25) -- (0,.05) (0,.15) circle(.1);%
18668 }

18669 \tikzset{%
18670  natoapp6c/s/radio relay/.pic=%
18671    \path[draw] (-.2,.2) arc (.2,.25) (0,.25) -- (0,.05) (0,.15) circle(.1);%
18672 }
natoapp6c/s/recovery unmanned systems

\tikzset{%
    natoapp6c/s/recovery unmanned systems/.pic={% 
    \path[draw] (-.5,.15) to [out=-80,in=180] (0,-.15) to 
    [out=0,in=260] (.5,.15);},
}\n
natoapp6c/s/rifle

\tikzset{%
    natoapp6c/s/rifle/.pic={% 
    \pic[draw]{natoapp6c/s/weapon=full};
    \pic[draw]{natoapp6c/s/weapon=rifle};},
}\n
natoapp6c/s/rising

\tikzset{%
    natoapp6c/s/rising/.pic={% 
    \path[draw,fill=pgfstrokecolor] (0, 0.2) -- (0, -0.167) 
    (0.1, -0.2) -- (-0.1, -0.2) -- (0, 0.0);},
}\n
natoapp6c/s/riverine

\tikzset{%
    natoapp6c/s/riverine/.pic={% 
    \ifn@to@pp@below% 
    \path[pic actions] (M.south west) 
    to [out=-90,in=-90,looseness=.5] (M.south east) -- cycle;
    \else% 
    \path[pic actions] (-.5,.15) to [out=-80,in=180] (0,-.15) to 
    [out=0,in=260] (.5,.15) -- cycle;
    \fi},
}\n
natoapp6c/s/rocket launcher

\tikzset{%
    pics/natoapp6c/s/rocket launcher/.is choice,
    pics/natoapp6c/s/rocket launcher/base/.style={
    code={
    \pic[draw]{natoapp6c/s/weapon=base};
    \pic[draw]{natoapp6c/s/weapon=rifle};
}}
\tikzset{\natoapp6c/s/rocket launcher/.style={\n\path\[pic actions\]^{}\n(0.44, 0.15) -- (0.44, -0.15) -- (-0.44, 0.15) -- (-0.44, -0.15) -- cycle;},\n\path\[pic\]\n\n\tikzset{\natoapp6c/s/runway/.pic=\%\n\path\[draw\] (-.3,-.15) -- (.3,-.15) (-.2,-.2) -- (.2,.2);},\n\path\[draw\]\n\n\tikzset{\natoapp6c/s/sailing boat/.pic=\%\n\path\[draw\]\n(-0.15, -0.2) -- \n\n\tikzset{\natoapp6c/s/rotary wing/.pic=\%\n\path\[pic actions\]^{}\n(0.44, 0.15) -- (0.44, -0.15) -- (-0.44, 0.15) -- (-0.44, -0.15) -- cycle;},\n\path\[pic\]
pics/natoapp6c/s/satellite/navigation/.style={
  code={
  \begingroup\tikz@picmode
  \pic[yshift=-3.75,scale=.9]{natoapp6c/s/satellite=none};
  \endgroup
  \pic[scale=.5,yshift=3.5]{natoapp6c/s/navigation};
},
}
}
pics/natoapp6c/s/satellite/earth observing/.style={
  code={
  \begingroup\tikz@picmode
  \pic[yshift=3.75, scale=0.9]{natoapp6c/s/satellite=none};
  \endgroup
  \path[pic actions]
  (0, 0.065) -- +(315:0.125)
  (0, 0.065) -- +(225:0.125)
  (0, -0.12) circle (0.08);
},
}
pics/natoapp6c/s/satellite/tether/.style={
  code={
  \begingroup\tikz@picmode
  \pic[yshift=-3.75, scale=0.9]{natoapp6c/s/satellite=none};
  \endgroup
  \path[pic actions]
  (0, -0.066) -- +(30:0.3)
  (0, -0.066) +(30:0.375) circle(0.075);
},
}
pics/natoapp6c/s/satellite/small/.style={
  code={
  \begingroup\tikz@picmode
  \pic[scale=0.6]{natoapp6c/s/satellite=none};
  \endgroup
  \path[pic actions]
  (0.05, 0.2) -- ( 0, 0.1) -- (-0.05, 0.2)
  (0.05, -0.2) -- ( 0, -0.1) -- (-0.05, -0.2)
  (-0.4, 0.05) -- (-0.3, 0) -- (-0.4, -0.05)
  ( 0.4, 0.05) -- ( 0.3, 0) -- ( 0.4, -0.05);
},
}
pics/natoapp6c/s/satellite/reconnaissance/.style={
  code={
  \pic[yshift=-1,fill=pgfstrokecolor]{natoapp6c/s/satellite=none};
  \path[pic actions]
  (-0.075, -0.05) -- +(250:0.1)
  (-0.025, -0.05) -- +(260:0.1)
  ( 0.025, -0.05) -- +(260:0.1)
  ( 0.075, -0.05) -- +(290:0.1);
},
}
pics/natoapp6c/s/satellite/.default=none,
\usetikzlibrary{calc}
\tikzset{
pics/natoapp6c/s/sea mine/.is choice,
pics/natoapp6c/s/sea mine/top half/.style={
code={
\path[draw,join=bevel,pic actions]
(0,0) arc(0:35:0.2 and 0.175) --
(42:.34 and .3) -- (48:.34 and .3) --
% ($(45:.1)+(40:.2)$) -- ($(45:.1)+(50:.2)$) --
(55:.2 and .175) arc(50:75:.2 and .175) --
(80:.26 and .23) -- (100:.26 and .23) --
(105:.2 and .175) arc(100:125:.2 and .175) --
(132:.34 and .3) -- (138:.34 and .3) --
%($(135:.1)+(130:.2)$) -- ($(135:.1)+(140:.2)$) --
(145:.2 and .175) arc(145:180:.2 and .175);
}},
pics/natoapp6c/s/sea mine/bottom half/.style={
code={
\path[pic actions] (0,0) arc(0:-180:.2);}},
pics/natoapp6c/s/sea mine/full/.style={
code={
\pic[fill=pgfstrokecolor]{natoapp6c/s/sea mine=top half};
\pic[fill=pgfstrokecolor]{natoapp6c/s/sea mine=bottom half};}}
pics/natoapp6c/s/sea mine/neutralised/.style={
code={
\begin{scope}[even odd rule]
\clip [rotate=42] (-.4,-.015) rectangle (.4,.015) [reverseclip];
\clip [rotate=-42] (-.4,-.015) rectangle (.4,.015) [reverseclip];
\pic {natoapp6c/s/sea mine=full};
\end{scope}
}},
pics/natoapp6c/s/sea mine/neutralized/.style=natoapp6c/s/sea mine/neutralised,
pics/natoapp6c/s/sea mine/.default=full,
}\n}
\usetikzlibrary{calc}
\tikzset{
\natoapp6c/s/seabed installation/.pic={
\path[pic actions]
(-.25, -.2) --
(0.25, -.2) --
(0.25, -.075) --
(0.05, -.075) --
(0.05, 0.025) --
(-.125, 0.025) --
(-.125, 0.2) --
(-.25, 0.2) -- cycle;},
}\n}
\tikzset{%
  natoapp6c/s/signal/.pic={%
\path[draw] (M.north west) -- (0,-.1) -- (0,.1) -- (M.south east);},
}

\tikzset{%
  natoapp6c/s/signals intelligence/.pic={%
\path[draw] (-.2,.2) -- (-.13,.25) -- (-.07,.2) -- (0,.25) --
(.07,.2) -- (.13,.25) -- (.2,.2) (0,.25) -- (0,-.2);},
}

\tikzset{%
  natoapp6c/s/ski/.pic={%
\path[draw] (-.15,-.15) -- (.1,.2) (.15,-.15) -- (-.1,.2)
(-.1,-.2) -- (-.2,-.1)
(.1,-.2) -- (.2,-.1);
},
}

\tikzset{%
  natoapp6c/s/sled/.pic={%
\ifs@to@pp@below%
  \draw ([shift={(-.15,.15)}]M.south west) to[in=180, out=-90]
++.15,.-15) -- (M.south east) to[in=-90, out=0]
([shift={(.15,.15)}]M.south east);
\else%
  \draw ([shift={(.3,.1)}]M.west) to[in=180, out=-90]
([shift={(.5,-.1)}]M.west) --
([shift={(-.5,-.1)}]M.east) to[in=-90, out=0]
([shift={(-.3,.1)}]M.east);
\fi
}

448
\tikzset{
  pics/natoapp6c/s/small squashed text/.style={
    code={\@to@pp@text@squashed{#1};}},
}

\tikzset{
  pics/natoapp6c/s/small text/.style={
    code={\@to@pp@text@small{#1};}},
}

\tikzset{
  natoapp6c/s/sniper/.pic={
    \path [draw] (-.2,.2)--(-.05,.2) (.05,.2)--(.2,.2) (0,.15)--(0,-.2);
  },
}

\tikzset{
  natoapp6c/s/space station/.pic={
    \path [join=bevel, pic actions]
    (-80:.15 and .06) -- (0.025, 0.175) arc(0:180:0.025) -- (-100:.15 and .06)
    (80:.25 and 0.1)+(-0.0125)$ \arcl{80:.25 and 0.1} --
    (-260:.15 and .06) arc (-260:80:.15 and .06) -- cycle
    (-82:.25 and .1) -- (0.025, -0.175) arc(360:180:0.025) -- (-98:.25 and .1);
  },
}

\tikzset{
  pics/natoapp6c/s/squashed text/.style={
    code={\@to@pp@text@squashed{#1};}},
}
natoapp6c/s/topographic

\tikzset{
  natoapp6c/s/topographic/.pic={
    \path[draw] (0,.05) -- (0,.2)
    (0,.05) -- (-.1,-.2)
    (0,.05) -- (.1,-.2)
    (-30:.15) arc[radius=.15,start angle=-30,end angle=-150];},
}

natoapp6c/s/torpedo

\tikzset{
  natoapp6c/s/torpedo/.pic={
    \path[draw,fill=pgfstrokecolor,pic actions]
    (-0.35, 0) --
    (-0.3, 0.075) --
    ( 0.25, 0.075) --
    ( 0.35, -0.075) --
    ( 0.35, 0.075) --
    ( 0.25, -0.075) --
    (-0.3, -0.075) -- cycle;},
}

natoapp6c/s/towed

\tikzset{
  natoapp6c/s/towed/.pic={
    \ifn@to@pp@below%
    \path[pic actions] (M.south east) -- (M.south west)
    ([shift={(.08,0)}]M.south east) circle(.08)
    ([shift={(-.08,0)}]M.south west) circle(.08);
    \else%
    \path[draw] (-.32,0) -- (.32,0) (-.4,0) circle(.08) (.4,0) circle(.08);
    \fi},
}

natoapp6c/s/tracked

\tikzset{
  natoapp6c/s/tracked/.pic={
    \ifn@to@pp@below%
    \path[pic actions]
    ([shift={(.08,-.16)}]M.south west)
    arc [radius=.08,start angle=-90,end angle=-270]
    -- ([shift={(-.08,0)}]M.south east)
    \else%
    \path[draw] (0,.16) -- (0,.8) (0,.8) -- (0,1.5) -- (0,.16);
    \fi},
}
\begin{tikzpicture}
\tikzset{\natoapp6c/s/train locomotive/.pic={
\path[pic actions]
(-.35,-.3)--(-.35,.3)--(.35,.3)--(.35,0)--(0,0)--(0,.3)--cycle;},}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{\natoapp6c/s/transportation/.pic={
\path[pic actions] (0,0) circle(.2)
(180:.2) -- (0:.2)
(225:.2) -- (45:.2)
(270:.2) -- (90:.2)
(315:.2) -- (135:.2);},}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{\natoapp6c/s/unexploded ordnance/.pic={
\begin{scope}[transparency group=knockout]
\path[pic actions,draw,fill=pgfstrokecolor] (0,0) circle(.2);
\end{scope},}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{\natoapp6c/s/unmanned/.pic={
\path[pic actions]
(0,-0.1)
--(0.45,0.05)
}}
\end{tikzpicture}
natoapp6c/s/utility vehicle
\tikzset{
natoapp6c/s/utility vehicle/.pic={
\pic[draw]{natoapp6c/s/vehicle};
\path[pic actions]
(0.35, 0.3) to[in=-90, out=-90, looseness=1] (-0.35, 0.3); },
}

natoapp6c/s/vehicle
\tikzset{
natoapp6c/s/vehicle/.pic={
\path[draw,fill=pgfstrokecolor,pic actions](0.38, -0.2) rectangle (0.42, 0.15);},
}

natoapp6c/s/video imagery
\tikzset{
natoapp6c/s/video imagery/.pic={
\path[draw,fill=pgfstrokecolor,pic actions](0.38, -0.2) rectangle (0.42, 0.15);},
}

natoapp6c/s/warfare vessel
\tikzset{
natoapp6c/s/warfare vessel/.pic={
\path[draw,fill=pgfstrokecolor] (0, -0.2) -- (0.3, 0.05) -- (-0.3, 0.05) -- cycle;},
}
\tikzset{
\n\n\path[pic actions]
\n\n(-0.3, 0.05) -- (0, 0.05) to[in=90, out=0] (0.3, -0.2)
\n(0, 0.05) -- (0, 0.2)
\n(0.075, 0.2) -- (-0.075, 0.2);},
\}
\n\tikzset{
\n\n\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
\path[pic actions]
\n\n\begin{scope}[\shift={(.4,-.16)}]M.south west\end{scope}
\path[\shift={(-.08,0)}]M.south east
\path[\shift={(.08,-.08)}]M.south west\circle(.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\path[\shift={(.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.32,0)}]circle(0.08) (.32,0) circle(0.08);
\path[\shift={(-.4,0)}]circle(0.08);
\path[\shift={(0,-.08)}]circle(0.08);
\path[\shift={(0,.08)}]circle(0.08);
\fi},
\}
5.6.21 Some extra MIL-STD symbols

Extra NATO App6(c) symbol (from MIL-STD)

```latex
\begin{tikzpicture}

\def\natoapp6c/s/prison{
\path[pic actions] (-.3,-.3)rectangle(.3,.3)
\path[pic actions] (-.23,-.30)--(-.23, .3)
\path[pic actions] (.23,-.30)--( .23, .3)
\path[pic actions] (-.08,-.30)--(-.08,-.2)
\path[pic actions] (-.08,.15) circle (.05)
\path[pic actions] (-.08,-.1) --(-.08, .3)
\path[pic actions] (.08,-.30)--( .08,-.2)
\path[pic actions] (.08,.15) circle (.05)
\path[pic actions] (.08,-.1) --(.08, .3)
\path[pic actions] (0,.15) circle(.07 and .1);
\}
\end{tikzpicture}
```

A list of all defined symbols

```latex
\begin{enumerate}
\item weapon=base,
\item weapon=top,
\item weapon=bottom,
\item weapon=rifle,
\item weapon=machine gun,
\item weapon=grenade launcher,
\item weapon=missile launcher,
\item weapon=non lethal,
\item weapon=multi fire,
\item weapon=air defence,
\item weapon=anti tank,
\item weapon=full,
\item weapon,
\item type=light,
\item type=medium,
\item type=heavy,
\item type=vlight,
\end{enumerate}
```
19264  type=vmedium,
19265  type=vheavy,
19266  type,
19267  above corps support,
19268  air assault with organic lift,
19269  air decoy,
19270  air assault,
19271  air defence,
19272  air strip,
19273  air traffic,
19274  airship,
19275  airborne,
19276  ammunition,
19277  amphibious,
19278  amphibious warfare ship,
19279  analysis,
19280  arrest,
19281  artillery,
19282  anti tank anti armour,
19283  antenna,
19284  armoured,
19285  armoured fighting vehicle,
19286  armoured personnel carrier,
19287  arctic,
19288  automobile,
19289  balloon,
19290  bar,
19291  base,
19292  bicycle equipped,
19293  boat,
19294  booby trap,
19295  bottomed,
19296  bridge=none,
19297  bridge=fixed,
19298  bridge=folding,
19299  bridge=hollow,
19300  bridge,
19301  capsule,
19302  carrier,
19303  chemical biological radiological nuclear,
19304  civilian military cooperation,
19305  civilian police,
19306  civilian telecommunications,
19307  coast guard vessel,
19308  combat support,
19309  combatant,
19310  combined arms,
19311  computer system,
19312  control,
19313  convoy,
19314  corps support,
19315  crime,
19316  decoy,
direct communications,
direction finding,
diving=none,
diving=military,
diving,
drilling,
earthmover,
electric power,
electronic ranging,
electronic warfare wide,
engineer,
enhanced location reporting system,
environmental protection,
explosion,
finance,
fishing vessel,
fire protection,
fixed and rotary wing,
fixed wing,
flame thrower,
floating,
surfaced,
food,
fuel,
grenade launcher=none,
grenade launcher=non lethal,
grenade launcher,
graffiti,
group,
gun=base,
gun=air defence,
gun=anti tank,
gun=direct,
gun=recoilless,
gun,
headquarters,
house,
howitzer,
in position,
individual,
infantry,
intermodal,
jagged wave,
jam,
jamming,
jetski,
killing,
labour,
land mine=personnel,
land mine=tank,
land mine=none,
land mine,
laser, launcher, laundry, machine gun, main gun, maintenance, medic, medical, medical treatment, mine, mine clearing equipment, mine warfare vessel, missile, missile launcher=base, missile launcher=none, missile launcher=air defence, missile launcher=anti tank, missile launcher=surface to surface, missile launcher, mobile advisor and support, moored, mortar, mortuary affairs, mountain, naval, navigation, navy task, non combatant, non lethal weapon, nuclear, observer, orbiter shuttle, ordnance, organisation, over snow, pack animal, patrol, patrolling, physician, pipeline, poisoning, postal, printed media, psychological, quarry, quartermaster, radar, radio, radio relay, radio relay line of sight, radio teletype, railroad,
reconnaissance,
recovery unmanned systems,
rifle,
rising,
riverine,
rocket launcher=base,
rocket launcher=anti tank,
rocket launcher=single,
rocket launcher=multiple,
rocket launcher=single head,
rocket launcher=multiple head,
rocket launcher,
rotary wing,
runday,
sailing boat,
skate
satellite=none,
satellite=astronomical,
satellite=biologic,
satellite=communications,
satellite=navigation,
satellite=earth observing,
satellite=tether,
satellite=small,
satellite=reconnaissance,
satellite,
sea mine=top half,
sea mine=bottom half,
sea mine=full,
sea mine=neutralised,
sea mine=neutralized,
sea mine,
seabed installation,
search,
searching,
semi trailer truck,
sensor,
ship,
signal,
signals intelligence,
ski,
sled,
small squashed text=TXT,
small text=TXT,
sniper,
space station,
squashed text=TXT,
submarine,
submersible,
supply,
surface combatant,
survey,
tactical satellite,
tank,
A Generate draft VASSAL module

We can use the code you wrote for your game pieces (counters, maps, tables), to generate a draft VASSAL module. To that end, use the document class \texttt{wgexport}, and some simple macros to export your graphics to a single PDF. A provided Python script then processes this to generate the draft VASSAL module.

The generated VASSAL module is not the final thing, but it is a good start.

A.1 Example

Suppose we have defined counters and markers like

\begin{verbatim}
allied 1 id axis 1 ad out of supply
allied 2 ad axis 2 ad game turn
allied 3 abid axis 3 ic
\end{verbatim}

via Tikz styles. Also assume that we have macros

\begin{verbatim}
\board \oob \charts \front
\end{verbatim}

which produces \texttt{tikzpictures} to the board, OOBs, charts, and cover, respectively. All this is defined in our package \texttt{mygame}. Of course that we have our rules in the file \texttt{game.pdf}.

We prepare a simple \LaTeX\ source file

\begin{verbatim}
\documentclass{wgexport}
\usepackage{mygame}
\begin{document}
\end{document}
\end{verbatim}
\begin{imagelist} \% Records image meta info
\chitimages{(}
  {allied 1 id,allied 2 ad,allied 3 abid}/Allied,\%
  {axis 1 ad,axis 2 ad,axis 3 ic}/Axis,\%
  {out of supply, game turn}/Markers)}
\info{Board}{board} \board
\info{OOB}{oob} \oob
\info{Charts}{chart} \chart
\info{Cover}{front} \front
\end{imagelist}
\end{document}

When we run \LaTeX{} on this, we will get a PDF where each page is a separate image and the page is cropped to image. In addition we will get a CSV (comma-separated-values) file \texttt{export.csv} which contains some meta information about each page. In particular, it identifies the name of each page, the category, and sub category of the image.

For chits, the name of the image is the style name (e.g., \texttt{game turn}). For other images, it is the first argument to \texttt{\info} above.

The category is for chits is always \texttt{counter}. For other images, it is the second argument to the \texttt{\info} macro (e.g., \texttt{board}).

The category of an image is important later on when we generate the VASSAL module. Recognised categories are

- \texttt{counter} for counter images. Such an image will trigger the creation of a VASSAL game piece.
- \texttt{board} for board images. Images of this kind will result in VASSAL board (or Map) elements.
- \texttt{oob} for Order of Battle tables. This will also result in a VASSAL map being created, but one that is displayed as a pop-up and with a rectangular grid. This is useful for placing units in an Order of Battle chart.
- \texttt{chart} for charts. These images will be made VASSAL charts — i.e., pop-up windows which contains some graphics for the players reference.
- \texttt{front} for the cover image. This will become the module splash image. Only one such image (the first) will be used.

Other categories may be used, and the corresponding image will be added to the VASSAL module. However, they will no be processed in any specific way.

The \texttt{sub-category} is mainly used for counters. Above, we gave the sub-categories \texttt{Allied}, \texttt{Axis}, and \texttt{Markers}. The sub-categories will help to identify the factions of the game, and counter prototypes will be made for each category. The sub-categories of \texttt{board}, \texttt{charts}, \texttt{oob}, and \texttt{front} has no or little effect.

One we have processed the file above to generate our PDF (Say \texttt{export.pdf}), then we can process it (and the CSV file) with a Python script to make our draft VASSAL module

\begin{verbatim}
export.py export.pdf export.csv -o Game.vmod -t Game -v 0.1 \ 
-d "My game" -r rules.pdf
\end{verbatim}

This will generate the draft module \texttt{Game.vmod}. Note that we add the rules (\texttt{-r rules.pdf}) to the module so that the module is complete.

462
Once the module has been generated, one can open it in the VASSAL editor and further customise it. For example, the grids used in the boards needs to be adjusted, and one may want to make initial set-ups or add all counters to the OOB.

Of course, running the Python script will overwrite all changes, so perhaps it is a good idea to work on a copy of the output file.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers in roman refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

```latex
\tikz/chit/upper right (key) 14398 \@resultmarkers .................. 868, 870
\tikz/natoapp6c (key) 14398 \@battlemarkers ............... 806, 808
\tikz/natoapp6c/below (key) 17234 \@chit\n@to .................. 14438, 14444, 14456
\tikz/natoapp6c/echelon (key) 17234 \@chit\n@to\@d .................. 14438, 14446, 14448
\tikz/natoapp6c/center (key) 14818 \@chit\@r\@e\@p\@Line .................. 948, 964, 966, 986, 970, 972,
\tikz/natoapp6c/above (key) 14818 \@chits .................. 15194, 15203, 15204
\tikz/natoapp6c/right (key) 14818 \@doublechitimages .......... 549, 551
\tikz/natoapp6c/left (key) 14818 \@doublechits .......... 15243, 15255, 15256
\tikz/natoapp6c/center (key) 14818 \@end@chit\n@to ................. 14448, 14460, 14629
\tikz/natoapp6c/center (key) 14818 \@end\@g\@d\@e ................. 127, 153, 1594
\tikz/natoapp6c/center (key) 14818 \@end\@g\@d\@g\@p\@ic ........ 86, 121, 1554, 17402
\tikz/natoapp6c/center (key) 14818 \@gobble .................. 1496, 14768,
\tikz/natoapp6c/center (key) 14818 14870, 14873, 14905, 15031,
\tikz/natoapp6c/center (key) 14818 15179, 15232, 15298, 17450
\tikz/natoapp6c/center (key) 14818 \@hex\n@t\@o\r\@false ............... 1498
\tikz/natoapp6c/center (key) 14818 \@defempty .................. 1036, 1037
\tikz/natoapp6c/center (key) 14818 \@if\@st\@ar ................. 474, 14943, 15035, 15203, 15255
\tikz/natoapp6c/center (key) 14818 \@if\@un\@d\@f\@i\@ned\@c\@o\@l\@or ............... 347
\tikz/natoapp6c/center (key) 14818 \@lb\@ch\@ar ............... 458, 468, 477,
\tikz/natoapp6c/center (key) 14818 656, 677, 684, 685, 688, 698,
\tikz/natoapp6c/center (key) 14818 709, 717, 742, 759, 767, 958, 989
\tikz/natoapp6c/center (key) 14818 \@ll\@x ............... 13890, 13907,
\tikz/natoapp6c/center (key) 14818 13908, 13917, 13918, 13927,
\tikz/natoapp6c/center (key) 14818 13928, 13937, 13938, 13977
\tikz/natoapp6c/center (key) 14818 \@ll\@y ............... 13891, 13907,
\tikz/natoapp6c/center (key) 14818 13908, 13917, 13918, 13927,
\tikz/natoapp6c/center (key) 14818 13928, 13937, 13938, 13978
\tikz/natoapp6c/center (key) 14818 \@od\@d\@markers ............... 849, 851
\tikz/natoapp6c/center (key) 14818 \@percent\char\ ............... 14097, 14143, 14145, 14153,
\tikz/natoapp6c/center (key) 14818 14163, 14165, 14166, 14172
\tikz/natoapp6c/center (key) 14818 \@rb\char\ ............... 459, 469, 474, 486, 657, 687,
\tikz/natoapp6c/center (key) 14818 692, 693, 703, 712, 721, 728,
\tikz/natoapp6c/center (key) 14818 743, 761, 768, 781, 982, 984, 1001
```
NATO App6(c) symbols:
- antenna
- amphibious warfare ship
- airship
- analysis

NATO App6(c) keys:
- anti tank anti armour
- arctic
- armoured fighting vehicle
- armoured personnel carrier
- arrest
- artillery
- automobile
- balloon
- bar
- base
- bicycle equipped
- boat
- bottomed
- bridge
- capsule
- carrier
- chemical biological
- radiological nuclear
- civilian military cooperation
- civilian police
- civilian telecommunications
- coast guard vessel
- combat support
- combatant
- combined arms
- computer system
- control
- convoy
- corps support
- crime
- decoy
- direct communications
- direction finding
- diving
- drilling
- earthmover
- electric power
- electronic ranging
- electronic warfare
- engineer
- enhanced location reporting system
- fixed and rotary wing
- fixed wing
- flame thrower
- floating
- food
- fuel
- graffiti
- grenade launcher
- group
- gun
- headquarters
- house
- howitzer
- in position
- individual
- infantry
- intermodal
- jam
- jamming
- jet ski
- killing
- labour
- land mine
- land missile
- laser
- launcher
- laundry
- machine gun
- main gun
- maintenance
- medic
- medical
- medical cooperation
- mine
- mine clearing equipment
- mine warfare vessel
- missile
- mobile advisor and support
- moored
- mortar
- mortuary affairs
- motorized
- mountain
- naval
- navigation
- navy task
- non combatant
- non lethal weapon
- nuclear
- observer
- orbiter shuttle
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATOAP6C/S/AIRBORNE</td>
<td>symbol</td>
<td>17622</td>
</tr>
<tr>
<td>NATOAP6C/S/AIRSHIP</td>
<td>symbol</td>
<td>17611</td>
</tr>
<tr>
<td>NATOAP6C/S/AMMUNITION</td>
<td>symbol</td>
<td>17632</td>
</tr>
<tr>
<td>NATOAP6C/S/AMPHIBIOUS</td>
<td>symbol</td>
<td>17638</td>
</tr>
<tr>
<td>NATOAP6C/S/AMPHIBIOUS WARFARE SHIP</td>
<td>symbol</td>
<td>17674</td>
</tr>
<tr>
<td>NATOAP6C/S/ANALYSIS</td>
<td>symbol</td>
<td>17684</td>
</tr>
<tr>
<td>NATOAP6C/S/ANTENNA</td>
<td>symbol</td>
<td>17706</td>
</tr>
<tr>
<td>NATOAP6C/S/ANTI TANK</td>
<td>symbol</td>
<td>17698</td>
</tr>
<tr>
<td>NATOAP6C/S/ARCTIC</td>
<td>symbol</td>
<td>17724</td>
</tr>
<tr>
<td>NATOAP6C/S/ARMoured</td>
<td>symbol</td>
<td>17710</td>
</tr>
<tr>
<td>NATOAP6C/S/ARMoured Fighting Vehicle</td>
<td>symbol</td>
<td>17714</td>
</tr>
<tr>
<td>NATOAP6C/S/ARMoured Personnel Carrier</td>
<td>symbol</td>
<td>17710</td>
</tr>
<tr>
<td>NATOAP6C/S/ARREST</td>
<td>symbol</td>
<td>17680</td>
</tr>
<tr>
<td>NATOAP6C/S/ARTILLERY</td>
<td>symbol</td>
<td>17694</td>
</tr>
<tr>
<td>NATOAP6C/S/AUTOMOBILE</td>
<td>symbol</td>
<td>17720</td>
</tr>
<tr>
<td>NATOAP6C/S/BALLOON</td>
<td>symbol</td>
<td>17740</td>
</tr>
<tr>
<td>NATOAP6C/S/BAR</td>
<td>symbol</td>
<td>17757</td>
</tr>
<tr>
<td>NATOAP6C/S/BASE</td>
<td>symbol</td>
<td>17761</td>
</tr>
<tr>
<td>NATOAP6C/S/BOAT</td>
<td>symbol</td>
<td>17771</td>
</tr>
<tr>
<td>NATOAP6C/S/BOoby Trap</td>
<td>symbol</td>
<td>17744</td>
</tr>
<tr>
<td>NATOAP6C/S/BOTTOMED</td>
<td>symbol</td>
<td>17787</td>
</tr>
<tr>
<td>NATOAP6C/S/BRIDGE</td>
<td>symbol</td>
<td>17795</td>
</tr>
<tr>
<td>NATOAP6C/S/CAPSULE</td>
<td>symbol</td>
<td>17815</td>
</tr>
<tr>
<td>NATOAP6C/S/ARMOUR</td>
<td>symbol</td>
<td>17824</td>
</tr>
<tr>
<td>NATOAP6C/S/CHRONICAL</td>
<td>symbol</td>
<td>17833</td>
</tr>
<tr>
<td>NATOAP6C/S/FISHING VESSEL</td>
<td>symbol</td>
<td>18065</td>
</tr>
<tr>
<td>NATOAP6C/S/FIXED AND ROTARY WING</td>
<td>symbol</td>
<td>18088</td>
</tr>
<tr>
<td>NATOAP6C/S/FIXED WING</td>
<td>symbol</td>
<td>18083</td>
</tr>
<tr>
<td>NATOAP6C/S/FLOATING</td>
<td>symbol</td>
<td>18102</td>
</tr>
<tr>
<td>NATOAP6C/S/FOOD</td>
<td>symbol</td>
<td>18126</td>
</tr>
<tr>
<td>NATOAP6C/S/FOODS</td>
<td>symbol</td>
<td>18133</td>
</tr>
<tr>
<td>NATOAP6C/S/FOOD STACK</td>
<td>symbol</td>
<td>18149</td>
</tr>
<tr>
<td>NATOAP6C/S/FOOD STORE</td>
<td>symbol</td>
<td>18160</td>
</tr>
<tr>
<td>NATOAP6C/S/GASOLINE</td>
<td>symbol</td>
<td>18163</td>
</tr>
<tr>
<td>NATOAP6C/S/GAS STATION</td>
<td>symbol</td>
<td>18169</td>
</tr>
<tr>
<td>NATOAP6C/S/GAS STATION</td>
<td>symbol</td>
<td>18193</td>
</tr>
<tr>
<td>NATOAP6C/S/HEALTHCARE</td>
<td>symbol</td>
<td>18199</td>
</tr>
<tr>
<td>NATOAP6C/S/HIGHER</td>
<td>symbol</td>
<td>18205</td>
</tr>
<tr>
<td>NATOAP6C/S/HYDRAULIC</td>
<td>symbol</td>
<td>18213</td>
</tr>
<tr>
<td>NATOAP6C/S/INDIVIDUAL</td>
<td>symbol</td>
<td>18218</td>
</tr>
<tr>
<td>NATOAP6C/S/INFANTRY</td>
<td>symbol</td>
<td>18223</td>
</tr>
<tr>
<td>NATOAP6C/S/INTERMIDIAN</td>
<td>symbol</td>
<td>18227</td>
</tr>
<tr>
<td>NATOAP6C/S/JAMMING</td>
<td>symbol</td>
<td>18241</td>
</tr>
<tr>
<td>NATOAP6C/S/JETSIE</td>
<td>symbol</td>
<td>18271</td>
</tr>
<tr>
<td>NATOAP6C/S/JETTIE</td>
<td>symbol</td>
<td>18276</td>
</tr>
<tr>
<td>NATOAP6C/S/KILLING</td>
<td>symbol</td>
<td>18291</td>
</tr>
<tr>
<td>NATOAP6C/S/LABOUR</td>
<td>symbol</td>
<td>18294</td>
</tr>
<tr>
<td>NATOAP6C/S/LAND MINE</td>
<td>symbol</td>
<td>18299</td>
</tr>
<tr>
<td>NATOAP6C/S/LAND MISSILE</td>
<td>symbol</td>
<td>18310</td>
</tr>
<tr>
<td>NATOAP6C/S/MAINTENANCE</td>
<td>symbol</td>
<td>18313</td>
</tr>
<tr>
<td>NATOAP6C/S/LEADER</td>
<td>symbol</td>
<td>18333</td>
</tr>
<tr>
<td>NATOAP6C/S/LEADER</td>
<td>symbol</td>
<td>18347</td>
</tr>
<tr>
<td>NATOAP6C/S/MAIN GUN</td>
<td>symbol</td>
<td>18344</td>
</tr>
<tr>
<td>NATOAP6C/S/MAIN GUN</td>
<td>symbol</td>
<td>18349</td>
</tr>
<tr>
<td>NATOAP6C/S/MAINTENANCE</td>
<td>symbol</td>
<td>18354</td>
</tr>
<tr>
<td>NATOAP6C/S/MEDIC</td>
<td>symbol</td>
<td>18365</td>
</tr>
<tr>
<td>NATOAP6C/S/MEDICAL</td>
<td>symbol</td>
<td>18385</td>
</tr>
<tr>
<td>NATOAP6C/S/MEDICAL</td>
<td>symbol</td>
<td>18388</td>
</tr>
<tr>
<td>NATOAP6C/S/MEDICAL TREATMENT</td>
<td>symbol</td>
<td>18391</td>
</tr>
<tr>
<td>NATOAP6C/S/MINE CLEARING EQUIPMENT</td>
<td>symbol</td>
<td>18405</td>
</tr>
<tr>
<td>NATOAP6C/S/MINE WARFARE VESSEL</td>
<td>symbol</td>
<td>18410</td>
</tr>
<tr>
<td>NATOAP6C/S/MISSILE</td>
<td>symbol</td>
<td>18416</td>
</tr>
<tr>
<td>NATOAP6C/S/MISSILE LAUNCHER</td>
<td>symbol</td>
<td>18431</td>
</tr>
<tr>
<td>NATOAP6C/S/MOBILE ADVISOR AND SUPPORT</td>
<td>symbol</td>
<td>18459</td>
</tr>
<tr>
<td>NATOAP6C/S/MOORED</td>
<td>symbol</td>
<td>18471</td>
</tr>
<tr>
<td>NATOAP6C/S/MOORING</td>
<td>symbol</td>
<td>18475</td>
</tr>
<tr>
<td>NATOAP6C/S/MORTUARY AFFAIRS</td>
<td>symbol</td>
<td>18484</td>
</tr>
<tr>
<td>NATOAP6C/S/MOTORIZED</td>
<td>symbol</td>
<td>18498</td>
</tr>
<tr>
<td>NATOAP6C/S/MOUNTAIN</td>
<td>symbol</td>
<td>18489</td>
</tr>
<tr>
<td>NATOAP6C/S/NAVAL</td>
<td>symbol</td>
<td>18494</td>
</tr>
<tr>
<td>NATOAP6C/S/NAVIGATION</td>
<td>symbol</td>
<td>18515</td>
</tr>
<tr>
<td>NATOAP6C/S/NAVY TASK</td>
<td>symbol</td>
<td>18511</td>
</tr>
</tbody>
</table>