The \texttt{wargame} package

Christian Holm Christensen

March 29, 2023

Abstract

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

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
2.7.1 Styling labels ........................................ 13
2.8 Towns .................................................. 13
  2.8.1 Styling towns .................................... 14
2.9 Extra graphics for hexes ................................. 14
2.10 Rivers, borders, and roads ............................. 14
  2.10.1 Styling paths ................................... 16
2.11 Board clipping and frame ............................... 16
2.12 Constructing the physical board ....................... 18
  2.12.1 Split the board over multiple sheets ............. 19
  2.12.2 Foldable board .................................. 20

3 Chits .................................................. 22
  3.1 Styling chits ....................................... 23
  3.2 Defining preset chit types ............................ 24

4 NATO App 6(c) symbols ................................ 25
  4.1 Faction and Command Selection ....................... 26
  4.2 Unit Size (echelon) .................................. 28
  4.3 Unit type identification ............................... 28

5 Implementation ........................................ 28
  5.1 The wargame package ................................ 28
  5.2 The wargame.util TikZ library ....................... 31
    5.2.1 Miscellaneous macros ............................ 31
    5.2.2 Pictures in compound nodes ...................... 33
    5.2.3 Nodes in compound nodes ........................ 35
    5.2.4 Bounding boxes .................................. 37
    5.2.5 Some utilities to get bounding boxes and the like .... 38
    5.2.6 Other Tikz utilities .............................. 39
    5.2.7 Random IDs ...................................... 42
    5.2.8 VASSAL icons ................................... 43
  5.3 The wgexport class .................................. 44
    5.3.1 Making dice ..................................... 55
    5.3.2 Hooks into chits, etc. ........................... 55
  5.4 The wargame.hex TikZ library ......................... 57
    5.4.1 Debugging ....................................... 58
    5.4.2 Suppress terrain pictures ....................... 58
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.3</td>
<td>Hex coordinate system</td>
<td>58</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Hexes</td>
<td>63</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Terrain</td>
<td>70</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Ridges</td>
<td>294</td>
</tr>
<tr>
<td>5.4.7</td>
<td>Towns</td>
<td>297</td>
</tr>
<tr>
<td>5.4.8</td>
<td>Labels</td>
<td>299</td>
</tr>
<tr>
<td>5.4.9</td>
<td>Extra graphics</td>
<td>301</td>
</tr>
<tr>
<td>5.4.10</td>
<td>Some macros</td>
<td>302</td>
</tr>
<tr>
<td>5.4.11</td>
<td>Edges, borders, roads, rivers, and so on</td>
<td>302</td>
</tr>
<tr>
<td>5.4.12</td>
<td>Other paths</td>
<td>305</td>
</tr>
<tr>
<td>5.4.13</td>
<td>Move, attacks, retreats from hex to hex</td>
<td>306</td>
</tr>
<tr>
<td>5.4.14</td>
<td>Board clipping and frame</td>
<td>310</td>
</tr>
<tr>
<td>5.4.15</td>
<td>Board splitting</td>
<td>315</td>
</tr>
<tr>
<td>5.5</td>
<td>The <code>wargame.chit</code> TikZ library</td>
<td>324</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Debugging</td>
<td>324</td>
</tr>
<tr>
<td>5.5.2</td>
<td>The <code>chit</code> key namespace</td>
<td>324</td>
</tr>
<tr>
<td>5.5.3</td>
<td>The <code>chit</code> styles</td>
<td>327</td>
</tr>
<tr>
<td>5.5.4</td>
<td>The <code>\chit</code> shape</td>
<td>327</td>
</tr>
<tr>
<td>5.5.5</td>
<td>The <code>\chit</code> wrapper macro</td>
<td>334</td>
</tr>
<tr>
<td>5.5.6</td>
<td>Predefined <code>chit</code> element pictures</td>
<td>335</td>
</tr>
<tr>
<td>5.5.7</td>
<td>Modifications to chits</td>
<td>337</td>
</tr>
<tr>
<td>5.5.8</td>
<td>Stacking of chits</td>
<td>338</td>
</tr>
<tr>
<td>5.5.9</td>
<td>Making order of battle charts</td>
<td>338</td>
</tr>
<tr>
<td>5.5.10</td>
<td>Table of chits</td>
<td>341</td>
</tr>
<tr>
<td>5.5.11</td>
<td>Battle markers</td>
<td>344</td>
</tr>
<tr>
<td>5.5.12</td>
<td>Dice</td>
<td>345</td>
</tr>
<tr>
<td>5.5.13</td>
<td>Some utilities</td>
<td>349</td>
</tr>
<tr>
<td>5.6</td>
<td>The <code>wargame.natoapp6c</code> TikZ library</td>
<td>351</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Debugging</td>
<td>351</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Colours</td>
<td>351</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Some dimensions</td>
<td>352</td>
</tr>
<tr>
<td>5.6.4</td>
<td>Some utilities</td>
<td>352</td>
</tr>
<tr>
<td>5.6.5</td>
<td>Faction names as macros</td>
<td>353</td>
</tr>
<tr>
<td>5.6.6</td>
<td>Node shapes</td>
<td>354</td>
</tr>
<tr>
<td>5.6.7</td>
<td>‘Friendly’ node shapes</td>
<td>355</td>
</tr>
<tr>
<td>5.6.8</td>
<td>‘Hostile’ node shapes</td>
<td>362</td>
</tr>
</tbody>
</table>
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 \texttt{tikzpicture} using the macro \texttt{\hex}. It takes up to 4 arguments

\begin{verbatim}
\hex[\{key-value-pairs\}][\{location\}][\{name\}]
\end{verbatim}

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

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

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

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

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

\texttt{row}=(\texttt{row}) and \texttt{column}=(\texttt{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). any style key defined for \texttt{TikZ} 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 $\times$ 1.86 cm — or roughly 3/4" $\times$ 3/4" — big. This allows the hexes to fit chits (see Section ??) of size 12 mm $\times$ 12 mm — or roughly 1/2" $\times$ 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 \texttt{TikZ}'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}
2.2 Hex bevels

A bevel (or “shadow-effect”) can be added to hexes using the key `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 `bevel fraction` (default 10%).

2.3 Styling hexes

Typical TikZ options can be passed to the `\hex` macro. For example, if you want to draw the hex borders in red, simply pass `draw=red` in the `[optional]` arguments to `\hex`. Individual parts of the hexes can be styled separately. The default style used by `\hex` is `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 `\hex`s one can define the style `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

\begin{verbatim}
  every hex/.style=
    {dash pattern=on .2cm off .6cm on .2cm off 0cm},
\end{verbatim}

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 \textlangle \textit{column}\textrangle-\textlangle \textit{row}\textrangle pairs, while vertexes and mid-point on edges can be specified separately. The syntax of the coordinates is

\begin{verbatim}
  \textlangle \textit{hex cs:row=⟨hex-row⟩,column=⟨hex-column⟩,vertex=⟨vertex⟩,edge=⟨edge⟩}\rangle
\end{verbatim}

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

\begin{table}[h]
\begin{tabular}{|c|c|c|}
\hline
\textit{vertex} & \textit{Angle} & \textit{edge} & \textit{Angle} \\
\hline
east & E & 0° & north east & NE & 30° \\
north east & NE & 60° & north & N & 90° \\
north west & NW & 120° & north west & NW & 150° \\
west & W & 180° & south west & SW & 210° \\
south west & SW & 240° & south & S & 270° \\
south east & SE & 300° & south east & SE & 330° \\
\hline
\end{tabular}
\caption{Vertex and edge positions}
\end{table}

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

\begin{verbatim}
\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=.3,c=1.3) circle(0.1);
\fill[lightgray](hex cs:r=1.3,c=1.3) circle(0.1);
\end{verbatim}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{Hex coordinate system}
\end{figure}
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 3 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 requires even more flexibility, one can use the Ti\LaTeX\ library \texttt{calc} to add arbitrary offsets, e.g.,

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

\section{2.5 Terrains}

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

The terrain of a hex is selected via the multi-valued key \texttt{terrain}. Sub-keys of this key are

- \texttt{image=\langle graphics-file \rangle} Specifies terrain tile image \langle graphics-file \rangle.
- \texttt{pic=\langle picture-key \rangle} Specifies terrain tile \texttt{TikZ} picture.
- \texttt{code=\langle tikz-code \rangle} Any valid \texttt{TikZ} code
- \texttt{clip=\langle path(s) \rangle} \texttt{TikZ} path specification to clip the terrain within the hex.

The terrain can be clipped by the sub-key \texttt{clip}. This can be useful if the game specifies movement costs in terms of hex-edge crossing, for example \textit{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 \texttt{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>
<th>Symbol</th>
<th>Name</th>
<th>terrain={image=⟨image⟩}</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="wargame.beach" alt="Clear" /></td>
<td>Clear</td>
<td></td>
<td><img src="wargame.beach" alt="Beach" /></td>
<td>Beach</td>
<td>{image=wargame.beach}</td>
</tr>
<tr>
<td><img src="wargame.light_woods" alt="Light woods" /></td>
<td>Light woods</td>
<td>{image=wargame.light_woods}</td>
<td><img src="wargame.woods" alt="Woods" /></td>
<td>Woods</td>
<td>{image=wargame.woods}</td>
</tr>
<tr>
<td><img src="wargame.rough" alt="Rough" /></td>
<td>Rough</td>
<td>{image=wargame.rough}</td>
<td><img src="wargame.swamp" alt="Swamp" /></td>
<td>Swamp</td>
<td>{image=wargame.swamp}</td>
</tr>
<tr>
<td><img src="wargame.mountains" alt="Mountains" /></td>
<td>Mountains</td>
<td>{image=wargame.mountains}</td>
<td><img src="wargame.sea" alt="Sea" /></td>
<td>Sea</td>
<td>{image=wargame.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](hex/terrain/mountain,line width=3pt)</td>
<td>Mountains</td>
<td>{pic=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.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE}</code></td>
<td><img src="image2.png" alt="Symbol" /></td>
<td><code>{hex/large sextant=NE,hex/large sextant=N}</code></td>
</tr>
<tr>
<td><img src="image3.png" alt="Symbol" /></td>
<td><code>{hex/sextant=N}</code></td>
<td><img src="image4.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE,hex/sextant=N,hex/sextant=NW}</code></td>
</tr>
<tr>
<td><img src="image5.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE}</code></td>
<td><img src="image6.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE,hex/sextant=N,hex/sextant=SW,hex/sextant=NW,hex/sextant=SE}</code></td>
</tr>
<tr>
<td><img src="image7.png" alt="Symbol" /></td>
<td><code>{hex/sextant=SW}</code></td>
<td><img src="image8.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE,hex/sextant=N,hex/sextant=SW,hex/sextant=SE,hex/sextant=SW,hex/sextant=S,hex/sextant=SE}</code></td>
</tr>
<tr>
<td><img src="image9.png" alt="Symbol" /></td>
<td><code>{hex/sextant=S}</code></td>
<td><img src="image10.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE,hex/sextant=N,hex/sextant=SW,hex/sextant=SE,hex/sextant=SE,hex/sextant=C}</code></td>
</tr>
<tr>
<td><img src="image11.png" alt="Symbol" /></td>
<td><code>{hex/sextant=SE}</code></td>
<td><img src="image12.png" alt="Symbol" /></td>
<td><code>{hex/sextant=NE,hex/sextant=N,hex/sextant=SW,hex/sextant=SE,hex/sextant=SE,hex/sextant=C}</code></td>
</tr>
</tbody>
</table>

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

Terrains use the key \texttt{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 \texttt{ridges}. The keyword takes a list of hex edges and generates symbology for the ridge on the chosen edges. Note that the edges do 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 \texttt{north east}, \texttt{north}, \texttt{north west}, \texttt{south west}, \texttt{south}, \texttt{south east}.

Table 5 shows some examples.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Symbol} & \textbf{ridges=} & \textbf{Symbol} & \textbf{ridges=} \\
\hline
\multirow{2}{*}{NE} & NE,N & \multirow{2}{*}{S} & NE,N,NW,SW,SE,\text{color=brown!70!black} \\
& & & \text{line width=3pt} \\
\hline
\multirow{2}{*}{N} & NE,N,NW & \multirow{2}{*}{SE} & N,S,NW,SE \\
& & & \\
\hline
\multirow{2}{*}{NW} & NE,N,NW,SW & & \\
& & & \\
\hline
\multirow{2}{*}{SW} & NE,N,NW,SW,\text{line width=3pt} & & \\
& & & \\
\hline
\end{tabular}
\caption{Ridges}
\end{table}
2.6.1 Styling ridges

Every ridge is drawn with the style `tikz/hex/ridges`. Users can customise this style. The default is to draw thin black wave lines (TikZ decoration `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 `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="symbol" alt="No label" /></td>
<td>No label</td>
<td>n/a</td>
<td>none</td>
</tr>
<tr>
<td><img src="symbol" alt="B10" /></td>
<td>User specified</td>
<td>n/a</td>
<td>text=B10</td>
</tr>
<tr>
<td><img src="symbol" alt="A1" /></td>
<td>User specified</td>
<td>n/a</td>
<td>{color=blue,text=A1}</td>
</tr>
<tr>
<td><img src="symbol" alt="0903" /></td>
<td>Two-digit, zero padded numbers</td>
<td>9/3</td>
<td>auto</td>
</tr>
<tr>
<td><img src="symbol" alt="133" /></td>
<td>Column letter, number row</td>
<td>2/3</td>
<td>{auto=alpha column,font=\noexpand\rmfamily} †</td>
</tr>
<tr>
<td><img src="symbol" alt="AG24" /></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

† When specifying macros as key values in the options, for example the value \verb!\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 option `auto=inv y x plus 1` will label the rows inversely, and add one to the column number. This requires that the key `tikz/max hex row` has been set to the largest row number used.

In addition to the sub-keys `none`, `auto`, and `text`, one can also specify the following keys

`place=(coordinates)` specifies the Location of label within the hex. The anchor point of the text will be placed at
this point.\[\langle\langle/\rangle\rangle\text{options} \] at the start of the option (but inside braces \{\ldots\}) 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><code>town=</code></th>
<th>Symbol</th>
<th><code>town=</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>Copenhagen</td>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>Copenhagen</td>
</tr>
<tr>
<td></td>
<td>{name=Copenhagen}</td>
<td></td>
<td>{name=Copenhagen}</td>
</tr>
<tr>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{pic=hex/town/city}</td>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{red,pic=hex/town/city}</td>
</tr>
<tr>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{fill=red}</td>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{name=London}</td>
</tr>
<tr>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{red,name=Paris}</td>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{above=0.8,name=Berlin}</td>
</tr>
<tr>
<td><img src="hex/town/town" alt="Hexagon" /></td>
<td>{place=(.2,.2)}</td>
<td><img src="hex/town/town" 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 \texttt{\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 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 Sovjet 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="image1" alt="hex/fortress" /></td>
<td>hex/fortress</td>
</tr>
<tr>
<td><img src="image2" alt="setup/sovjet" /></td>
<td>setup/sovjet</td>
</tr>
<tr>
<td><img src="image3" alt="setup/german" /></td>
<td>setup/german</td>
</tr>
<tr>
<td><img src="image4" alt="setup/german,hex/fortress" /></td>
<td>{setup/german,hex/fortress}†</td>
</tr>
<tr>
<td><img src="image5" alt="line width=2pt,brown" /></td>
<td>{[{line width=2pt,brown}] fortress 2}‡</td>
</tr>
<tr>
<td><img src="image6" alt="shift={(.2,.2)}setup/sovjet" /></td>
<td>{[shift={(.2,.2)}]setup/sovjet}†</td>
</tr>
<tr>
<td><img src="image7" alt="shift={(.2,.2)},scale=.5,color=gray" /></td>
<td>{[shift={(.2,.2)},scale=.5,color=gray] setup/german}‡</td>
</tr>
<tr>
<td><img src="image8" 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="image9" 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 (\{\ldots\})
‡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 (\{\ldots\}) 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 \road. The road or railroad is specified via a regular \TiKZ path.

\road[\{options\}] \path;
\railroad[\{options\}] \path;

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

### 2.10.1 Styling paths

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

### 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 \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 Ti\kZ \texttt{\texttt{draw[clip]}} command.
A clipping path of that spans from the hex at \textit{lower-left} to \textit{upper-right}. Note, that both of these arguments should only specify the column and row keys. If \textit{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}

\textbf{Figure 7: Snug–fit clipping of board using the macro} \texttt{\boardclip}

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

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

\textbf{where} \textit{lower-left} and \textit{upper-right} are as for \texttt{\boardclip}. The \textit{margin} must be a number, and specifies an optional margin around the hexes. The argument \textit{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 \textit{options} argument to \texttt{\boardclip} to specify a default background color. The frame is drawn with the style \texttt{hex/board frame}

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

\textbf{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.

\textbf{Either} scale the board down so that it fits. Use the \texttt{TikZ} key \texttt{scale=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.

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

\section*{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 × 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.

---

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

- Cut through carton
- Cut groove ($\frac{1}{2}$ through) in carton on back side
- Cut groove ($\frac{1}{2}$ through) in carton on front side

20
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\textsuperscript{1}, can be made with the macro $\texttt{chit}$. The syntax for rendering a chit is

\begin{verbatim}
chit{(key-value-pairs)}{(location)}{(name)}
\end{verbatim}

Figure 9 shows an example of a chit.

\begin{verbatim}
symbol={\{nato app 6(c) spec\}}
left={chit/identifier={1}}
factors={chit/2 factors={4,8}}
right={chit/identifier={2}}
\end{verbatim}

Figure 9: An example of a chit. The lines below and to the right shows two unit lengths. Other global options used are $\texttt{color=white}$ to set the foreground colour, $\texttt{fill=blue}$ for the background, and $\texttt{text=yellow}$ to set the font colour to yellow. The $\texttt{symbol}$ key also contains $\texttt{frame=\{fill=yellow!50!black\}}$ to set the frame fill colour, and $\texttt{ultra thick}$ 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 $\texttt{natoapp}$ macro (see Section 4). The other parts of the chit ($\texttt{factors}$, $\texttt{left}$, $\texttt{right}$, and $\texttt{below}$) are rendered onto the chit via Ti\textsc{k}Z pictures. This allows for a great deal of flexibility in generating chits. For example, above we use the pictures $\texttt{chit/identifier}$ and $\texttt{chit/2 factors}$ 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 $\texttt{full}$. In that case, all other keys ($\texttt{symbol}$, $\texttt{factors}$, $\texttt{left}$, $\texttt{right}$, and $\texttt{below}$) are ignored. Figure 10 shows such an example.

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

Just as $\texttt{hex}$ is really a wrapper around Ti\textsc{k}Z's $\texttt{node}$ macro, so it is with $\texttt{chit}$. This means that an alternative way of making a chit is to do

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

\textsuperscript{1}Since \TeX\ has the concept of counters as in $\texttt{count}$ and \LaTeX\'s $\texttt{newcounter}$, we choose the name ‘chit’ for playing pieces instead.
Since chits are really TikZ 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=\ldots].

### 3.1 Styling chits

Typical TikZ 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.

TikZ 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\(^2\).

---

2The colour pgfstrokecolor is not modified by that.
The styles used by the left, right, setup, factors, and 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}={\textit{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

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

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

2–4

where, in the second example, we have passed additional options to \texttt{chit}. Note that we \textit{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 is serves as a placement guide of the the various parts of the NATO marker as illustrated in Figure 13.

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

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

- **faction** = \{faction\} 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,\ldots\}), and each symbol can be preceeded 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

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

Figure 13: Main keys of \textit{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.
\textit{frame=(keys)} Extra keys for frame.

### 4.1 Faction and Command Selection

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

<table>
<thead>
<tr>
<th>\textit{(command)}</th>
<th>friendly</th>
<th>hostile</th>
<th>neutral</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>air</td>
<td><img src="image" alt="Air Friendly" /></td>
<td><img src="image" alt="Air Hostile" /></td>
<td><img src="image" alt="Air Neutral" /></td>
<td><img src="image" alt="Air Unknown" /></td>
</tr>
<tr>
<td>land</td>
<td><img src="image" alt="Land Friendly" /></td>
<td><img src="image" alt="Land Hostile" /></td>
<td><img src="image" alt="Land Neutral" /></td>
<td><img src="image" alt="Land Unknown" /></td>
</tr>
<tr>
<td>equipment</td>
<td><img src="image" alt="Equipment Friendly" /></td>
<td><img src="image" alt="Equipment Hostile" /></td>
<td><img src="image" alt="Equipment Neutral" /></td>
<td><img src="image" alt="Equipment Unknown" /></td>
</tr>
<tr>
<td>installation</td>
<td><img src="image" alt="Installation Friendly" /></td>
<td><img src="image" alt="Installation Hostile" /></td>
<td><img src="image" alt="Installation Neutral" /></td>
<td><img src="image" alt="Installation Unknown" /></td>
</tr>
<tr>
<td>sea surface</td>
<td><img src="image" alt="Sea Surface Friendly" /></td>
<td><img src="image" alt="Sea Surface Hostile" /></td>
<td><img src="image" alt="Sea Surface Neutral" /></td>
<td><img src="image" alt="Sea Surface Unknown" /></td>
</tr>
<tr>
<td>sub surface</td>
<td><img src="image" alt="Sub Surface Friendly" /></td>
<td><img src="image" alt="Sub Surface Hostile" /></td>
<td><img src="image" alt="Sub Surface Neutral" /></td>
<td><img src="image" alt="Sub Surface Unknown" /></td>
</tr>
<tr>
<td>space</td>
<td><img src="image" alt="Space Friendly" /></td>
<td><img src="image" alt="Space Hostile" /></td>
<td><img src="image" alt="Space Neutral" /></td>
<td><img src="image" alt="Space Unknown" /></td>
</tr>
<tr>
<td>activity</td>
<td><img src="image" alt="Activity Friendly" /></td>
<td><img src="image" alt="Activity Hostile" /></td>
<td><img src="image" alt="Activity Neutral" /></td>
<td><img src="image" alt="Activity Unknown" /></td>
</tr>
</tbody>
</table>

Table 9: Frames for various combinations of \textit{(faction)} and \textit{(command)} combinations. These are drawn with the \texttt{pic} given by \texttt{natoapp6c/\{(faction)\}/\{(command)\}} with the options \texttt{draw=blue,fill=\{(faction)\}}. If no \texttt{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 \textit{frame}. If this is or contains the special value \textit{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 \textit{frame}.

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

Table 10 illustrates this.
Figure 14: Illustration of using the special value `faction` for the `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="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></td>
<td><img src="example.png" alt="Diagram" /></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

```latex
\ProvidesPackage{wargame}
```

Then needed packages

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

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

```latex
\@ifundefined{ifhex@terrain@pic}{%
\newif\ifhex@terrain@pic
\hex@terrain@picfalse}{}
```

Options

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

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

```latex
\usetikzlibrary{wargame.hex,wargame.natoapp6c,wargame.chit}
```
<table>
<thead>
<tr>
<th>Example</th>
<th>echelon</th>
<th>Approx. size</th>
<th>Sub-units</th>
<th>Officer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>team</td>
<td>3–5</td>
<td>none</td>
<td>Corporal or Sergeant</td>
</tr>
<tr>
<td></td>
<td>squad</td>
<td>5–10</td>
<td>1–2 teams</td>
<td>Sergeant</td>
</tr>
<tr>
<td></td>
<td>section</td>
<td>7–13</td>
<td>2–3 teams</td>
<td>Sergeant</td>
</tr>
<tr>
<td></td>
<td>platoon</td>
<td>25–40</td>
<td>Several squads/sections</td>
<td>Second Lieutenant</td>
</tr>
<tr>
<td></td>
<td>company</td>
<td>60–250</td>
<td>Several platoons</td>
<td>Captain</td>
</tr>
<tr>
<td></td>
<td>battalion</td>
<td>3000–10000</td>
<td>2–6 companies</td>
<td>Lieutenant colonel</td>
</tr>
<tr>
<td></td>
<td>regiment</td>
<td>500–2000</td>
<td>3–7 battalions</td>
<td>Colonel</td>
</tr>
<tr>
<td></td>
<td>brigade</td>
<td>2000–5000</td>
<td>Several battalions</td>
<td>Colonel</td>
</tr>
<tr>
<td></td>
<td>division</td>
<td>10000–20000</td>
<td>Several brigades/regiments</td>
<td>Major General</td>
</tr>
<tr>
<td></td>
<td>corps</td>
<td>30000–60000</td>
<td>Several divisions</td>
<td>Lieutenant General</td>
</tr>
<tr>
<td></td>
<td>army</td>
<td>100000</td>
<td>Several corps (5–10 divisions)</td>
<td>General</td>
</tr>
<tr>
<td></td>
<td>army group</td>
<td>120000–500000</td>
<td>Several armies</td>
<td>Field Marshal</td>
</tr>
<tr>
<td></td>
<td>theatre</td>
<td>250000+</td>
<td>Several army groups</td>
<td>Field Marshal</td>
</tr>
<tr>
<td></td>
<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</td>
</tr>
<tr>
<td></td>
<td>Air defence</td>
</tr>
<tr>
<td></td>
<td>Airborne</td>
</tr>
<tr>
<td></td>
<td>Amphibious</td>
</tr>
<tr>
<td></td>
<td>Anti tank/armoured</td>
</tr>
<tr>
<td></td>
<td>Armoured</td>
</tr>
<tr>
<td></td>
<td>Chemical biological radiological nuclear</td>
</tr>
<tr>
<td></td>
<td>Combined arms</td>
</tr>
<tr>
<td></td>
<td>Engineer</td>
</tr>
<tr>
<td></td>
<td>Field artillery</td>
</tr>
<tr>
<td></td>
<td>Infantry</td>
</tr>
<tr>
<td></td>
<td>Mechanised infantry</td>
</tr>
<tr>
<td></td>
<td>Mountaineer</td>
</tr>
<tr>
<td></td>
<td>Naval</td>
</tr>
<tr>
<td></td>
<td>Reconnaissance</td>
</tr>
<tr>
<td></td>
<td>Special Operations Forces</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Echelon &amp; Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxx</td>
<td>Army group</td>
</tr>
<tr>
<td>xxxx</td>
<td>Army</td>
</tr>
<tr>
<td>xxx</td>
<td>Corps</td>
</tr>
<tr>
<td>xx</td>
<td>Division</td>
</tr>
<tr>
<td>x</td>
<td>Brigade</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Regiment</td>
</tr>
<tr>
<td>1 1</td>
<td>Battalion</td>
</tr>
<tr>
<td>1</td>
<td>Company</td>
</tr>
<tr>
<td>••••</td>
<td>Platoon</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\(\La\tex\) library

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

5.2.1 Miscellaneous macros

```
\wargame\logo
```

This will produce the logo for this package.

```
\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\La\TeX\} \wargame},
  \wargame\ logo\ chit/.style={
    chit={symbol={\[
      \text{faction=friendly,}
      \text{command=land,}
      \text{echelon=division,}
      \text{main=infantry}}},
    factors={\text{chit/2 factors={4,3}}},
    left={\text{chit/identifier=III}},
    right={\text{chit/small identifier={10\textsuperscript{th}}}},
    color=white, \fill=red!50!black
  }
}
```

31
\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=-1,r=0) {};
\node[hex={terrain=woods}] (logo woods) at (hex cs:c=-1,r=1) {};
\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=0) {};
\node[wargame logo chit] (logo chit) at (hex cs:) {};
\node[wargame logo text] (logo text) {\wg@logo@text@content};
\end{scope}

\newcommand\wargamelogo[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=-1,r=0) {};
\node[hex={terrain=woods}] (logo woods) at (hex cs:c=-1,r=1) {};
\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=0) {};
\node[wargame logo chit] (logo chit) at (hex cs:) {};
\node[wargame logo text] (logo text) {\wg@logo@text@content};
\end{scope}}

\newcommand\wargamedbglvl{0}
\def\wg@dbg#1#2{\ifnum#1>\wargamedbglvl\relax\else\message{\string#2}\fi}

\newcommand{\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 θ 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.

\newcommand{\wg@sub@nchor}{Get anchor from sub node. We cannot use \pgfpointanchor since that returns the anchor coordinates in the global coordinate system.}
\def\wg@sub@nchor#1#2{% 
\wg@dbg{3}{\text{Get '}}}\text{#2} \text{ in '}}}\text{'#1'}}\text{'}}\% 
\@ifundefined{pgf@sh@ns@#1}{% 
\pgf@x=0cm\pgf@y=0cm}{% 
\pgf@process{% 
\csname pgf@sh@ma@#1\endcsname% MW 
\csname pgf@sh@np@#1\endcsname% 
\pgf@sh@reanchor{\csname pgf@sh@ns@#1\endcsname}{#2}}}% 
\wg@dbg{10}{\text{-> the}}\text{\the\pgf@x,\the\pgf@y}\% 
}

Scratch dimensions 
\newdimen\wg@tmpa 
\newdimen\wg@tmpb 
\newdimen\wg@tmpc 
\newdimen\wg@tmpd 

Macro to easily restore a saved path 
\def\settosave#1{ 
\pgfsyssoftpath@setcurrentpath{#1} 
}

5.2.2 Pictures in compound nodes

\section*{\textit{\texttt{\textasciitilde\texttt{wg@pic}}} Pictures in compound nodes}

The macro \texttt{\textasciitilde\texttt{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{\texttt{\textasciitilde\texttt{wg@pic}}\{⟨options⟩\}\{picture\}\texttt{\textasciitilde\texttt{endwg@pic}}\{⟨prefix⟩\}\{⟨position⟩\}\{⟨options⟩\}}

Note the \texttt{\textasciitilde\texttt{endwg@pic}} at the end of the call to swallow up \texttt{⟨picture⟩}. Typically this macro is used as

\texttt{\edef\args{⟨something⟩}} \texttt{\expandafter\wg@pic\args\texttt{\textasciitilde\texttt{endwg@pic}}\{⟨prefix⟩\}\{⟨position⟩\}\{⟨options⟩\}}

where \texttt{⟨something⟩} typically expands to \texttt{⟨user option⟩}\texttt{⟨picture⟩}

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

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

33
This macro then forwards to \wg@@pic to gobble up \textit{picture}.

1. User options
2. Arguments
3. Prefix
4. Coordinates
5. Fixed options

\def\wg@@@pic#1#2#3#4#5{%  
  \ifx|#2|\wg@dbg{3}{No picture given}%  
  \else%  
  \wg@dbg{3}{\texttt{\textbackslash pic} [\texttt{#5}, \texttt{#1}] at (#4) \{\texttt{#3}\#2\}}%  
  \pic[\texttt{#5},\texttt{#1}] at (#4) \{\texttt{#3}\#2\};%  
  \if\wg@s@ve%  
  \pgf@relevantforpicturesizetrue%  
  \begin{getbbl}%  
  \pic[\texttt{draw=none},\texttt{fill=none},\texttt{transform shape}] at (#4) \{\texttt{#3}\#2\};%  
  \end{getbbl}%  
  \wg@dbg{5}{Clipping to local bounding box}%  
  \clip (L.south west) rectangle (L.north east);%  
  \pgf@relevantforpicturesizefalse \global\wg@s@vefalse%  
  \fi%  
  \fi%  
  \wg@dbg{3}{End of \texttt{WG Pic}}%  
}%

\wg@pic@all

This macro sets all pictures in a list.

1. List
2. Prefix
5.2.3 Nodes in compound nodes

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⟩][⟨body⟩]@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
1. User options
2. Arguments
3. Prefix
4. Coordinates
5. Fixed options

\def\wg@@node[#1]{#2}\end\wg@node{%}
\def\wg@dbg{2}{Options: '#1', body: '#2'}%
\wg@@@node{#1}{#2}%

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

Bounding box dimensions

Enable or disable bounding box tracking

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

$$\frac{150\text{pixel}}{2.54\text{cm}} = \frac{59.055\text{pixel}}{\text{cm}}$$

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

$$1\ pt = \frac{249}{250} \ 12" = \frac{83}{6000} \ 1" = 0.035136$$
The next macro gets an anchors coordinates and stores them (in units of centimetres) in \texttt{tmp@x} and \texttt{tmp@y}.

\begin{verbatim}
\def\wg@get@nchor#1#2{%
  \pgfpointanchor{#1}{#2}%
  \pgfgetlastxy\tmp@x\tmp@y%
  \pgfgetlastxy\tmp@x\tmp@y%
  \pgfpointanchor{#1}{#2}%
  \pgfgetlastxy\tmp@x\tmp@y%
  \pgf@xa=\pgf@x
  \pgf@ya=\pgf@y
  \message{^^JAnchor #1.#2 @ (\the\pgf@xa,\the\pgf@ya)}
}
\end{verbatim}

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

\begin{verbatim}
\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)}
}
\end{verbatim}

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

\begin{verbatim}
\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}
}
\end{verbatim}

5.2.6 Other Tikz utilities

\textbf{tikz/reverseclip}

A reverse clipping path. This is used to cut out stuff outside of path defined.

\begin{verbatim}
\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)}]
\end{verbatim}


\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 \textit{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]{\reset cm}{(-\maxdimen,-\maxdimen)rectangle(\maxdimen,\maxdimen)}},
257 }
258 }
\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@vfalse},\
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.

269 \%
270 \% Save pgf rounded corners macro
271 \def\wg@setcornersarched#1{%
272 \ifx|#1|\else%
273 \edef\pgf@corner@arc{{#1}{#1}}%
274 \pgf@arccornerstrue%
275 \ifdim#1=0pt%
276 \pgf@arccornersfalse%
277 \fi\fi}
278 \newdimen\wg@lw@scaled\wg@lw@scaled=1pt
279 \def\wg@getscale{%
280 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
281 \wg@dbg{4}{Scale is \wg@jaca}
282 \xdef\wg@scale{\wg@jaca}%
283 \def\wg@scaled#1{%
284 \wg@getscale%
285 \wg@dbg{4}{Scaling #1 by \wg@scale}
286 \pgfmathsetmacro{\wg@tmp}{\wg@scale*#1}%
287 \xdef\wg@tmp{\wg@tmp}%
288 \xdef\wg@lw@scale{\wg@tmp}%
289 \wg@dbg{4}{Scaled #1 -> \wg@tmp}%
290 \def\wg@scaled#1{%
291 \wg@getscale%
292 \wg@dbg{4}{Scaling #1 by \wg@scale}
293 \pgfmathsetmacro{\wg@tmp}{\wg@scale*#1}%
294 \xdef\wg@tmp{\wg@tmp}%
295 \xdef\wg@lw@scale{\wg@tmp}%
296 \wg@dbg{4}{Scaled #1 -> \wg@tmp}%
297 \message{^^JScaled rounded corners: \meaning\pgfsetcornersarced}%
298 \tikzset{
299 \pgfgettransformentries{%
300 \wg@jaca}%,
301 \wg@jacc}%,
302 \wg@jacb}%,
303 \wg@jacc}%,
304 \wg@lw@scale=1pt
305 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
306 \def\wg@setcornersarched{\wg@lw@scaled=1pt}
307 \def\wg@getscale{%
308 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
309 \def\wg@jacc{\wg@jacc}%
310 \def\wg@jacb{\wg@jacb}%
311 \def\wg@lw@scale{\wg@lw@scaled=1pt}
312 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
313 \def\wg@jacc{\wg@jacc}%
314 \def\wg@jacb{\wg@jacb}%
315 \def\wg@lw@scale{\wg@lw@scaled=1pt}
316 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
317 \def\wg@jacc{\wg@jacc}%
318 \def\wg@jacb{\wg@jacb}%
319 \def\wg@lw@scale{\wg@lw@scaled=1pt}
320 },
321 scale rounded corners/.style=({
322 \pgf@process(#1)
323 \pgf@x=\pgf@x%
324 \pgf@y=\pgf@y%
325 \pgfmathsetmacro\wg@cornersarched{\the\pgf@x}
326 \newdimen\wg@lw@scaled\wg@lw@scaled=1pt
327 \def\wg@getscale{%
328 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
329 \def\wg@jacc{\wg@jacc}%
330 \def\wg@jacb{\wg@jacb}%
331 \def\wg@lw@scale{\wg@lw@scaled=1pt}
332 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
333 \def\wg@jacc{\wg@jacc}%
334 \def\wg@jacb{\wg@jacb}%
335 \def\wg@lw@scale{\wg@lw@scaled=1pt}
336 \pgfmathsetmacro{\wg@jaca}{sqrt(abs(\wg@jaca*\wg@jacc-\wg@jacb*\wg@jacc))}%
337 \def\wg@jacc{\wg@jacc}%
338 \def\wg@jacb{\wg@jacb}%
339 \def\wg@lw@scale{\wg@lw@scaled=1pt}
340 },
341",}
The normal `pic actions` cannot be used as it causes an infinite loop.

```
\tikzset{
  sub pic actions/.code={%
    \tikz@picmode%
    \edef\opts{\iftikz@mode@draw draw,\else draw=none,\fi
      \iftikz@mode@fill fill\else fill=none\fi}
    \wg@dbg{5}{Sub Mode: \meaning\tikz@picmode \meaning\opts}
    \pgfset{/tikz/.cd}
    \pgfkeysalsofrom{\opts}
  }
}
```

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

```
\def\wg@randomid{%
  \def\wg@uuid{}
  \wg@dbg{3}{Drawing with \texttt{stroke} \`\wg@tempfg\'}
  \iftikz@strokecolor\iftikz@mode@draw\else not\space\fi drawing
  \and fill \`\wg@tempbg\' \iftikz@fillcolor\iftikz@mode@fill\else not\space\fi filling}
}
```

5.2.7 Random IDs

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

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

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;
},
\end{verbatim}

Then, a closed and open trash can

\begin{verbatim}
\tikzset{
  trash can/.pic=%,
  \pic[trash can body];
  \pic[trash can lid];
},
\end{verbatim}

\begin{verbatim}
\tikzset{
  trash can open/.pic=,
  \pic[trash can body];
  \pic[rotate=-30] at (0,.1) \{trash can lid\};
},
\end{verbatim}

Now we can use that to generate some useful icons.
These icons does not use the trash can picture.

5.3 The `wgexport` class

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

Class identification and load `wargame` package

\ProvidesClass{wgexport}
\PassOptionsToClass{multi=tikzpicture, varwidth=false}{standalone}
We need a few utilities before we get to the actual environment. First, we need a tool 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

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 \@tabchar, \@lbchar, and \@rbchar to produce literal characters. \LaTeX already has \@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
\mk@w{
[}
\end{verbatim}

Precede 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 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.

```
468 \def\info{%
469 \@ifstar{\@@info{,}}{\@@info{\@rbchar,}}}%
470 \def\@@info#1#2#3#4{%
471 \chit@dbg{2}{Making image '#2' of type '#3'/'#4' on page \thepage}%
472 \mk@w{ \@lbchar}%
473 \mk@w{ \space "name": "#2",}%
474 \mk@w{ \space "category": "#3",}%
475 \mk@w{ \space "subcategory": "#4",}%
476 \mk@w{ \space "number": \thepage \@rbchar}%
477 \let\oldmk@i\mk@i%
478 \ifx#1,\relax\edef\mk@i{\mk@i\space\space}\fi}
479 \def\end@info{%
480 \let\mk@i\oldmk@i%
481 \mk@w{ \space \@rbchar,}}
```

Make separate images for each counter (single sided).

```
482 \newcommand\chitimages[2][]{%
483 \begingroup%
484 \let\chit@report\do@chit@report%
485 \let\natoapp@report\do@natoapp@report%
486 \chit@dbg{2}{chits to make images of '##2'}%
487 \foreach[count=\ti from 0] \t/x in #2{%
488 \chit@dbg{2}\"Row: '##t' ("##x")%
489 \ifx\t\empty\else\% Ignore empty rows
```
\chit@dbg{5}{~JSubcategory: ‘x’ (default ‘#1’)}
% Take sub-category or default
\ifx\t\x\def\x{#1}\else\ifx\x\empty\def\x{#1}\fi\fi
\foreach \u/\m in \t{%
  \ifx\u\empty\else% Ignore empty cells
  \ifx\u\chit@blank\else%
    \chit@dbg{2}{Next chit ‘\u’ with possible multiplicity ‘\m’}%
  \fi% If not multiplicity defined
  \ifx\m\@empty\def\m{1}\fi% If the same as unit
  \chit@dbg{2}{Next chit ‘\u’ multiplicity ‘\m’}%
  % We only make one copy of the chit, since we can duplicate
  % it in VASSAL
  \info*{\u}{counter}{\x}
  \begin{tikzpicture}
    \chit[\u=\ti]%
  \end{tikzpicture}
  \end@info%
  \foreach \n in {1,...,\m}{% Make a number of copies
    \ifx\u\chit@blank% Ignore blank chit
    \chit@dbg{3}{Ignoring blank chit:’\u’}%
    \else%
    \info{\u}{counter}{#2}
    \begin{tikzpicture}
      \chit[\u=\ti](\c,\r)%
    \end{tikzpicture}
    \fi%
  }%
  \fi%
}\fi%
\chit@dbg{2}{End of inner loop}%
\fi%
\chit@dbg{2}{End of outer loop}%
\endgroup%
}

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

\newcommand\doublechitimages[2][]{%
  \begingroup%
    \let\chit@report\do@chit@report%
    \let\natoapp@report\do@natoapp@report%
    \foreach[count=\ti from 0] \t/\x in #2{%
      \ifx\t\empty\else% Ignore empty rows
        \chit@dbg{5}{~JSubcategory: ‘x’ (default ‘#1’)}
        % Take sub-category or default
        \ifx\t\x\def\x{#1}\else\ifx\x\empty\def\x{#1}\fi\fi
        \foreach \u/\m in \t{%
          \ifx\u\empty\else% Ignore empty cells
            \ifx\u\chit@blank\else%
              \chit@dbg{2}{Next chit ‘\u’ with possible multiplicity ‘\m’}%
              \fi% If not multiplicity defined
          \fi%
        }%
    }%
  \endgroup%
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
The \( \langle \text{name} \rangle \) will be the name of the scope. If this contains the sub-string \text{hex} (upper, lower, or mixed case), then the zone will get a hex grid with numbering attached to it.

If the \( \langle \text{name} \rangle \) contains the sub-string \text{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 \text{T}, so that it won’t collide with the main zone. Similar if \( \langle \text{name} \rangle \) contains \text{oob}, except the separator is set to \text{O}.

If \( \langle \text{name} \rangle \) contains the sub-string \text{pool}, then it is assumed to be a pool of counters, and \text{no} grid is attached.

For \text{zone path} applied to a \text{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 \text{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 \text{boardimage} also records the coordinate options currently in use (keys \text{hex/first row is}, \text{hex/row direction is}, and so on), as well as the current label option (as defined by every \text{hex} or every \text{hex node}).

The information extracted is written to the \text{\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 \text{zoned}, \text{zone scope}, and \text{zone path} are defined in \text{wargame} to be dummies so that one can have them in the definition of the board without impact.
For the key \texttt{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.

\begin{Verbatim}
\pgfdeclaredecoration{record path construction}{initial}{%
\state{initial}[width=0pt,next state=more]{%
\begingroup
\pgf@decorate@inputsegment@first
\ptpoint@to@cm{\the\pgf@x}{\the\pgf@y}
\xdef\tmp@x{[\x,\y]}
\endgroup
}
\state{more}[width=\pgfdecoratedinputsegmentremainingdistance]{%
\begingroup
\pgf@decorate@inputsegment@last
\ptpoint@to@cm{\the\pgf@x}{\the\pgf@y}
\xdef\tmp@y{\tmp@y,[\x,\y]}
\endgroup
}
\state{final}{%
\begingroup
\pgf@decorate@inputsegment@last
\ptpoint@to@cm{\the\pgf@x}{\the\pgf@y}
\xdef\tmp@y{\tmp@y,[\x,\y]}
\endgroup
\mk@w{ \mk@i "zone path \tmp@x\tmp@y": \@lbchar}
\mk@w{ \mk@i  "path": \tmp@y,\@rbchar,
}}%
\end{Verbatim}

\section{boardimage}

The first thing we do is to use the \texttt{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{bd\#i} records the current indentation (which is important in JSON)

\begin{Verbatim}
\newenvironment{boardimage}{3}[3]{%
\def\bd@n{#2}
\newcount\mk@point
\mk@point=0
\let\oomk@i\mk@pos% 
\let\oomk@i\mk@pos%
\info{dummy}{<<dummy>>}{% 
\tikz{}% 
\tikz{}% 
\tikz{\scoped[ every hex/.try,every hex node/.try, ]{% 
\def\hex@col{0}%
}}%
\end{Verbatim}

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

We then monkey-patch \boardframe to also output coordinates to our JSON file. Note that this will probably be embedded in a different object.

Next, we make the style zoned to be applied to the \tikzpicture environment. This records the bounding box of the full picture.
The next style is the `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.

683  \zone scope/.code={%
684    \mk@w{ \mk@i"zone scope ##1": \@lbchar}
685    \let\omk@i\mk@i
686    \edef\mk@i{\mk@i\space}
687    \mk@transform%
688    \%\bd@w{ \@rbchar,}
689    \gdef\wg@export@box{##1}%
690    \pgfkeys{%
691      /tikz/local bounding box=wg export box,
692      /tikz/execute at end scope={
693        \mk@bb{wg export box}
694        \let\mk@i\omk@i
695        \mk@w{ \mk@i\@rbchar,}}
696    }, % pgfkeys
697  }, % zone scope

The next style gets the global coordinates of the current (0,0) point - f.ex. in a node - and outputs that

698  \zone point/.code n args={3}{
699    \pgf@xa=##2 cm
700    \pgf@ya=##3 cm
701    \pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}%
702    \% \pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}
703    \pgf@xa=\pgf@x
704    \pgf@ya=\pgf@y
705    \edef\px{\pgfmathresult}
706    \edef\py{\pgfmathresult}
707    \advance\mk@point1
708    \global\mk@point=\mk@point
709    \mk@w{ \mk@i "point\the\mk@point": \@lbchar "name": "##1", "type": "point", "coords": \[\px,\py]\@rbchar, }
710    %\message{^^JZone point \the\mk@point \space ##1: \pgf@xa,\pgf@ya -> \px,\py}
711  },
712  \zone oob point/.code n args={3}{
713    \pgf@xa=##2 cm
714    \pgf@ya=##3 cm
715    \advance\pgf@xa.1cm
716    \advance\pgf@ya.1cm
717    \pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}%
718    \% \pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}
719    \pgf@xa=\pgf@x
720    \pgf@ya=\pgf@y
721    \edef\px{\pgfmathresult}
722    \edef\py{\pgfmathresult}
723    \advance\mk@point1
724    \global\mk@point=\mk@point
725    \mk@w{ \mk@i "point\the\mk@point": \@lbchar "name": "##1", "parent": "\wg@export@box", "type": "point", "coords": \[\px,\py]\@rbchar, }
726    %\message{^^JZone point \the\mk@point \space ##1: ##2,##3 -> \px,\py}
727  },

52
\def\wg@gennumberm@rkers#1#2#3{
  \message{\~JNumbered markers: Type='#1' Max='#2' Category='#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[Markers]{\markers}
}
\tikzset{
  wg hidden unit/.pic={},
  wg hidden unit/.style={
    chit={
      frame={draw={none},fill=none},
      full=wg hidden unit}}}
\DeclareRobustCommand\battlemarkers[2][BattleMarkers]{
  \wg@gennumberm@rkers{battle marker}{#2}{#1}
  \message{\~JMake a hidden unit and add to Markers category}
  \nopagecolor\pgfkeysalsofrom{Markers}\chitimages[Markers]{\markers}
}

The **zone path** 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 simple 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.
\tikz[\text{scale=.7, transform shape}]{\text{pic\{battle marker=0\}%;}}
\info{\text{clear-battles-icon\{icon\}}{}
\tikz[\text{scale=.4, transform shape}]{\text{\{pic\{eliminate \text{ icon}\};}}
\text{\{pic\{scale=.7, transform shape\} at \{-0.3,0\}\{battle marker=0\};\}}
\}

Make odds markers. Mandatory argument is a list of odds and fill colours. Optional is the group to add the markers to.

\def\wg\text{gencolor\{markers\}#1\#2\#3}{%
\def\markers{}
\def\keys{}
\foreach \o/\f in {#2}{%
  \ifx\o\f\def\f{white}\fi%
  \message{\text{\textbackslash Odds marker \text{"#1 \o'} \text{w/fill \text{"f'}}\text{\}}}%
  \xdef\keys{/tikz/#1 \o/.style={/tikz/#1={\o,\f}},\keys}
  \xdef\markers{\markers,#1 \o}}%
{%}
\nopagecolor\pgfkeysalsofrom{\keys}\text{\textbackslash c\{hitimages\}[\#3]{\text{\{markers\}}}%
\DeclareRobustCommand\oddsmarkers[2][\text{\textbackslash OddsMarkers\}{%}
\text{\textbackslash wg\text{gencolor\{markers\}#2}{\#1}%;}}
\info{\text{\textbackslash odds-battles-icon\{icon\}}{}}
\tikz[\text{\textbackslash scale=.5, transform shape}]{\text{\{pic\{odds marker={?:?,white}\}}}
\info{\text{\textbackslash resolve-battles-icon\{icon\}}{}}
\tikz[\text{\textbackslash scale=.3, transform shape}]{%}
\text{\{pic\{dice\};}}
\text{\{pic\{scale=1.2, transform shape\} at \{-0.2,-0.2\}\{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.

\DeclareRobustCommand\resultmarkers[2][\text{\textbackslash ResultMarkers\}{%}
\text{\textbackslash wg\text{gencolor\{markers\}#2}{\#1}%;}}
Common icons used by many modules

\DeclareRobustCommand\commongicons[2][\text{\textbackslash Commonicons\}{%}
\begin{tikzpicture}[\text{\textbackslash \textbackslash begin\{group\}\{}}%
\nopagecolor%
\tikzset{\text{icon/.style={\textbackslash scale=.4, transform shape}}}%
\%}
\info{\text{\textbackslash pool-icon\{icon\}}{}}
\tikz[\text{\textbackslash icon}]{\text{\{pic\{pool icon\};}}
\%
\info{\text{\textbackslash oob-icon\{icon\}}{}}
\tikz[\text{\textbackslash icon}]{\text{\{pic\{oob icon={#1}\{#2\};\}}%
\%
\info{\text{\textbackslash flip-icon\{icon\}}{}}
\tikz[\text{\textbackslash icon}]{\text{\{pic\{flip icon\};}}%
\%
\info{\text{\textbackslash eliminate-icon\{icon\}}{}}
\tikz[\text{\textbackslash icon}]{\text{\{pic\{eliminate icon\};}}%
5.3.1 Making dice

\begin{verbatim}
dice[[tikz-options]][[node-options]]{{(name)}}{{(name)}}{{(list)}}
\end{verbatim}

1. \textit{(tikz-options)}
2. \textit{(node-options)}
3. \textit{(name)} - an identifier - e.g., the same as \textit{(shape)}.
4. \textit{(shape)} - one of d4, d6, d8, d10, d12, or d20.
5. \textit{(list)} - list of pairs \textit{⟨value⟩}/\textit{⟨printed⟩}, where \textit{⟨value⟩} is the value, and \textit{⟨printed⟩} is the shown value. If \textit{⟨printed⟩} is left out, then \textit{⟨value⟩} is used.

\begin{verbatim}
def\dice\{
\@ifnextchar[[\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,\p){};
\node[shape=#4,transform shape\{\p;}}
\end{verbatim}

5.3.2 Hooks into chits, etc.

TO BE DONE: We could add hook the \texttt{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 contain 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.

```latex
\def\do@chit@report{%
  \chit@dbg{3}{Start of Chit Report}
  \makew{ \makew{ "chit": \@lbchar} }%\message{\meaning{\@chit@rep@line(factors)}}%\message{\meaning{\@chit@rep@line(left)}}%\message{\meaning{\@chit@rep@line(right)}}%\message{\meaning{\@chit@rep@line(upper left)}}%\message{\meaning{\@chit@rep@line(lower left)}}%
}```
5.4 The \texttt{wargame.hex} Ti\texttt{kZ} library

Used Ti\texttt{kZ} libraries

\begin{verbatim}
\RequirePackage{alphalph}
\usetikzlibrary{calc}
\usetikzlibrary{arrows.meta}
\usetikzlibrary{arrows}
\usetikzlibrary{shapes.geometric}
\usetikzlibrary{shapes.symbols}
\usetikzlibrary{shapes.arrows}
\usetikzlibrary{decorations}
\usetikzlibrary{decorations.pathmorphing}
\usetikzlibrary{decorations.pathreplacing}
\usetikzlibrary{decorations.markings}
\usetikzlibrary{wargame.util}
\end{verbatim}
This is a utility macro we will use below.

\def@ifempty#1{\def\temp{#1}\ifx\temp@empty

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}
\def\hexdbglvl\hex@dbg{
\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{ifhex@terrain@pic}{%\newif\ifhex@terrain@pic\hex@terrain@pictrue}{}\def\markpos#1(#2){%\end{verbatim}

5.4.3 Hex coordinate system

\begin{verbatim}
\def\hex@xx\hex@yy{
\xdef\hex@xx{\pgfmathresult}\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
\end{verbatim}

Some offsets along $x$ and $y$ 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.

\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
\end{verbatim}

Some code we need for some options

\begin{verbatim}
\newif\ifhex@label@is@name\hex@label@is@namefalse
\end{verbatim}
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)

Which way does the column and row numbers go
Make labels names of shapes of the hexes so we can use labels to place stuff

```
983  hex/label is name/.is if=hex@label@is@name,
```

If we have uneven number of rows in some columns.

```
984  hex/short bottom columns/.is choice,
985  hex/short bottom columns/odd/.code={%
986   \def\hex@bot@short@col{isodd}
987   \def\hex@got@bot@short{istrue}
988  \hex@dbg(4){Short columns (odd): \meaning\hex@bot@short@col},
989  hex/short bottom columns/even/.code={
990   \def\hex@bot@short@col{iseven}
991   \def\hex@got@bot@short{istrue}
992  \hex@dbg(4){Short column (even): \meaning\hex@bot@short@col},
993  hex/short bottom columns/none/.code={
994   \def\hex@bot@short@col{isfalse}
995   \def\hex@got@bot@short{isfalse}
996  \hex@dbg(4){Short columns (none): \meaning\hex@bot@short@col},
997  hex/short bottom columns=none,
998  hex/short columns/.forward to=hex/short bottom columns,
999  hex/short top columns/.is choice,
1000 hex/short top columns/odd/.code={%
1001   \def\hex@top@short@col{isodd}
1002   \def\hex@got@top@short{istrue}
1003  \hex@dbg(4){Short columns (odd): \meaning\hex@top@short@col},
1004  hex/short top columns/even/.code={
1005   \def\hex@top@short@col{iseven}
1006   \def\hex@got@top@short{istrue}
1007  \hex@dbg(4){Short column (even): \meaning\hex@top@short@col},
1008  hex/short top columns/none/.code={
1009   \def\hex@top@short@col{isfalse}
1010   \def\hex@got@top@short{isfalse}
1011  \hex@dbg(4){Short columns (none): \meaning\hex@top@short@col},
1012  hex/short top columns=none,
1013 }
1014 \message{^^JInitial hex coordinate setup:
1015 Rows: factor=\hex@coords@row@fac, offset=\hex@coords@row@off
1016 Columns: factor=\hex@coords@col@fac, offset=\hex@coords@col@off}
```

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`, will allow us to start the columns with 1.

\begin{tikzpicture}
\t\t\hexcoords/.cd,
\t\t\row/.store in=\hex@row,
\t\t\col/.store in=\hex@col,
\t\t\offset/.store in=\hex@off,
\t\t\column/.store in=\hex@col,
\t\t\r/.store in=\hex@row,
\t\t\offset/.store in=\hex@off,
\t\t\vertex/.is choice,
\t\t\vertex/none/.code={\global\let\hex@vtx\@empty},
\t\t\vertex/east/.code={\def\hex@vtx{0}},
\t\t\vertex/north east/.code={\def\hex@vtx{60}},
\t\t\vertex/north west/.code={\def\hex@vtx{120}},
\t\t\vertex/west/.code={\def\hex@vtx{180}},
\t\t\vertex/south west/.code={\def\hex@vtx{240}},
\t\t\vertex/south east/.code={\def\hex@vtx{300}},
\t\t\vertex/.default=none,
\t\t\v/.forward to=/hex/coords/vertex=#1,
\t\t\edge/.is choice,
\t\t\edge/none/.code={\global\let\hex@edg\@empty},
\t\t\edge/north east/.code={\def\hex@edg{30}},
\t\t\edge/north/.code={\def\hex@edg{90}},
\t\t\edge/north west/.code={\def\hex@edg{150}},
\t\t\edge/south west/.code={\def\hex@edg{210}},
\t\t\edge/south/.code={\def\hex@edg{270}},
\t\t\edge/south east/.code={\def\hex@edg{330}},
\t\t\edge/.default=none,
\t\t\e/.forward to=/hex/coords/edge,
\t\t\}
\end{tikzpicture}

\begin{verbatim}
\hexcoords@reset
\end{verbatim}

This macro resets the hex coordinates to default values. That is row and column 0, no vertex or edge.

\begin{tikzpicture}
\t\\begin{tikzpicture}
\t\t\end{tikzpicture}
\end{tikzpicture}
The following calculates the Cartesian coordinates from Hex coordinates

\[
\text{cs:hex column=(C), row=(R), vertex=(V), edge=(E)}
\]

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

\[
\begin{align*}
x &= 2C \frac{3}{4} r \\y &= r(R - (C \% 2) \sin 60^\circ)
\end{align*}
\]

If \((V)\) or \((E)\) 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.

\[
\def\hex@coords@conv#1{\hex@coords@reset\tikzset{/hex/coords/.cd, #1}}
\]

Then we calculate the \(x\) coordinate and set the dimension \(\text{pgf}\@x\). We do this by

\[
x = c_e \frac{3}{2},
\]

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 `hex/column direction is`) and the offset \(o_c\) (set by `hex/first column is`).

\[
\pgfmathparse{int(\hex@coords@col@fac*(\hex@col+\hex@coords@col@off))}\%\def\hex@eff@col{\pgfmathresult}\%\def\hex@dbg{2}{Effective column: \hex@coords@col@fac * (\hex@col + \hex@coords@col@off) -> \hex@eff@col}\%\pgfmathparse{\hex@eff@col*1.5}\%\def\hex@x{\pgfmathresult}\%
\]

And then for the \(y\) coordinate and set the dimension \(\text{pgf}\@y\).

\[
y = 2 (r_e - c_e \text{ mod } 2) \cos 60^\circ ,
\]

where

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

\begin{verbatim}
\pgfmathparse{int(\hex@coords@row@fac*(\hex@row+\hex@coords@row@off))}\
\xdef\hex@eff@row{\pgfmathresult}\
\hex@dbg{2}{Effective row: \hex@coords@row@fac * (\hex@row + 
\hex@coords@row@off) -> \hex@eff@row}\
\pgfmathparse{(2*\hex@eff@row-mod(round((\hex@col+\hex@coords@col@off)),2))*\hex@yy}\
\pgfmathparse{(2*\hex@eff@row-mod(abs(round(\hex@col+\hex@coords@col@off)),2))*\hex@yy}\
\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}
\ifx\hex@vtx\@empty\else\
\pgfmathparse{\hex@x+\hex@off*cos(\hex@vtx)}\xdef\hex@x{\pgfmathresult}\
\pgfmathparse{\hex@y+\hex@off*sin(\hex@vtx)}\xdef\hex@y{\pgfmathresult}\
\fi\
\end{verbatim}

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

\begin{verbatim}
\ifx\hex@edg\@empty\else\
\pgfmathparse{\hex@x+\hex@off*\hex@yy*cos(\hex@edg)}\xdef\hex@x{\pgfmathresult}\
\pgfmathparse{\hex@y+\hex@off*\hex@yy*sin(\hex@edg)}\xdef\hex@y{\pgfmathresult}\
\fi\
\end{verbatim}

For debugging, we can print out stuff.

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

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 actually fill stuff into the hex, such as terrain and so on.

```
\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 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\wg@tmpe by \hex@bevel@frac 
\wg@tmpf=\wg@tmpb\multiply\wg@tmpf by \hex@bevel@frac 
\wg@tmpg=\wg@tmpc\multiply\wg@tmpg by \hex@bevel@frac 
divide\wg@tmpe100 
divide\wg@tmpf100 
divide\wg@tmpg100 
% Start 
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}% 
% Left 
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}% 
% Left-down 
\pgfpathlineto{\pgfqpoint{-\wg@tmpc}{\wg@tmpd}}% 
% Right down 
\wg@tmpa=-\wg@tmpa% 
\wg@tmpb=-\wg@tmpb% 
\pgfpathlineto{\pgfqpoint{\wg@tmpc}{\wg@tmpd}}% 
% Up, in 
\advance\wg@tmpa\wg@tmpe% 
\advance\wg@tmpb\wg@tmpf% 
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}% 
% Left-down, in 
\advance\wg@tmpc-\wg@tmpg 
\pgfpathlineto{\pgfqpoint{\wg@tmpc}{\wg@tmpd}}% 
% Left, down in 
\endscope}
```
These are the actual user callable anchors. We make anchors for each vertex and mid points on each edge.

\anchor{center}{ \pgfpointorigin}
\anchor{east}{ \east}
\anchor{west}{ \west}
\anchor{north east}{ \northeast}
\anchor{north west}{ \northwest}
\anchor{south west}{ \southwest}
\anchor{north east edge}{ \northeastedge}
\anchor{north west edge}{ \northwestedge}
\anchor{south west edge}{ \southwestedge}
\anchor{north west edge}{ \northwestedge}
\anchor{south west edge}{ \southwestedge}
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.

```latex
The \textit{behind} background path, where we do most of the work.
```
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}{}%
\ifx\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}{}%
\ifx\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 \[⟨options⟩\].

```
\@ifundefined{hex@extra@clip}{\let\hex@extra@clip\empty}{%}
\ifx\hex@extra@clip\empty\else%
\hex@dbg{5}{Extra clipped: ‘\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%
```

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 \hex@row and \hex@column, respectively. We may want to name the generated node from these if the user specified that option (perhaps use \pgfnoderename or similar).

```
\@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\%
```

67
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}{\hex@do@town}
\@ifundefined{hex@c@pic}{\let\hex@c@pic\empty}{\hex@dbg{5}{Extra: ‘\meaning\hex@extra’}}
\pgfpointorigin\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y\wg@pic@all{\hex@extra}{}{\the\wg@tmpa,\the\wg@tmpb}{}
\fi
\end{verbatim}

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 \[⟨options⟩\].

\begin{verbatim}
\@ifundefined{hex@extra}{\let\hex@extra\empty}{\hex@dbg{5}{Extra: ‘\meaning\hex@extra’}}
\pgfpointorigin\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y\wg@pic@all{\hex@extra}{}{\the\wg@tmpa,\the\wg@tmpb}{}
\fi
\end{verbatim}

\begin{verbatim}
/hex/terrain
/hex/town
/hex/label
/hex/ridges
/hex/extra
/hex/extra clipped
\end{verbatim}

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>

\[
\text{\tikzset{}}\\
\text{\tikzset{hex/.search also=/tikz,}}\\
\text{\tikzset{hex/.cd,}}\\
\text{\tikzset{terrain/.store in=\hex@terrain,}}\\
\text{\tikzset{ridges/.store in=\hex@ridges,}}\\
\text{\tikzset{town/.store in=\hex@town,}}\\
\text{\tikzset{extra/.store in=\hex@extra,}}\\
\text{\tikzset{label/.store in=\hex@label,}}\\
\text{\tikzset{extra clipped/.store in=\hex@extra@clip}}\\
\]

```
```

**hex**

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.

\[
\text{\tikzset{}}\\
\text{\tikzset{hex/\texttt{.style=}}(}}\\
\text{\tikzset{hex/\texttt{transform shape,}}(}}\\
\text{\tikzset{hex/\texttt{anchor=center,}}(}}\\
\text{\tikzset{hex/\texttt{draw=pgfstrokecolor,}}(}}\\
\text{\tikzset{hex/\texttt{fill=none,}}(}}\\
\text{\tikzset{hex/\texttt{thick,}}(}}\\
\text{\tikzset{hex/\texttt{solid,}}(}}\\
\text{\tikzset{hex/\texttt{code=}}(}}\\
\text{\tikzset{\texttt{\hex@dbg{\textsc{1}}}{== Hex with options: \'#1\'}}}}\\
\text{\tikzset{\texttt{\pgfkeys{\texttt{/tikz/transform shape,}}}}}}\\
\text{\tikzset{\texttt{\pgfkeys{\texttt{/h}}}}}}\\
\]

The first thing is to set the default graphics options. The key \texttt{every hex} can be set to hex options to be used for all hexes. For example, if one want to label all hexes with an auto-generated label, one can do

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

This, coupled with the \texttt{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.
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@[} %
}\@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)} %
\@ifnextchar[{{\hex@@@}}{\hex@@@}[#1]}{\hex@@@[#1](c=0,r=0)} %
}
```

Third argument is the name to be used.

```latex
\def\hex@@@#1[#2][#3]{{ } 
  \@ifnextchar[..][{\hex@@@}[#1][#2][#3]}{\hex@@@}[#1][#2][#3]() %
\@ifnextchar[{{\hex@@@@}}{\hex@@@@}[#1][#2][#3]}{\hex@@@@}[#1][#2][#3]()} %
}
```

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\relax% 
  \@ifundefined{\hex@l@text}{% 
    \hex@dbg{8}{=== Label text of hex (#2) not defined}%
    \let\hex@l@text\empty% 
  }{}% 
  \if\hex@label@is@name% 
    \hex@dbg{5}{=== Use label text of hex (#2) as name}%
    \if\hex@l@text\@empty% 
      \hex@dbg{8}{=== Argh! Label text is empty! ‘\meaning\hex@l@text’}%
    \else% 
      \pgfnoderename{\hex@l@text}{tmp}% 
    \fi% 
  \else% 
    \hex@dbg{3}{=== Renaming hex to user defined name ‘#3’}%
    \pgfnoderename{#3}{tmp}% 
  \fi% 
  \@ifnextchar;{}{\@gobble}{% 
}
```

5.4.5 Terrain

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.

```latex
\usetikzlibrary{hex/terrain}
```

```latex
\tikzset{
    /hex/terrain/.search also=/tikz,\
    /hex/terrain/.cd,\
    pic/.store in=\hex@t@pic,\
    image/.store in=\hex@t@image,\
    code/.store in=\hex@t@code,\
    clip/.store in=\hex@t@clip,\
    pic/.default=,\
    image/.default=,\
    code/.default=,\
    clip/.default=,\
}
```

```latex
\iffalse
\tikzset{
    /hex/terrain/.cd,\
    beach/.style={pic=hex/terrain/beach},\
    light woods/.style={pic=hex/terrain/light woods},\
    woods/.style={pic=hex/terrain/woods},\
    swamp/.style={pic=hex/terrain/swamp},\
    rough/.style={pic=hex/terrain/rough},\
    mountains/.style={pic=hex/terrain/mountains},\
    village/.style={pic=hex/terrain/village},\
    town/.style={pic=hex/terrain/town},\
    city/.style={pic=hex/terrain/city},\
}
\else
\tikzset{
    /hex/terrain/.cd,\
    beach/.style={image=wargame.beach},\
    light woods/.style={image=wargame.light_woods},\
    woods/.style={image=wargame.woods},\
    swamp/.style={image=wargame.swamp},\
    rough/.style={image=wargame.rough},\
    mountains/.style={image=wargame.mountains},\
    village/.style={image=wargame.village},\
}
\fi
```
Before we go on, we define the macro that actually generates the terrain of a hex.

\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 \hex={terrain={clip=...}}. The first thing into the new scope is to process the keys specified in \hex={terrain=...}. This will set the terrain and the clipping of the terrain.

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.

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\}.
\begin{tikzpicture}

% If we have no image, check if we have pictures.
\ifx\hex@t@image\empty%
  \hex@dbg{8}{No terrain images}%
\else%
  \hex@dbg{5}{Terrain pictures}%
  \foreach \i in \hex@t@pic{%
    \wg@pic@all{\i}{\the\wg@tmpa,\the\wg@tmpb}{}}%
\fi% We have pictures.
\end{tikzpicture}

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 \textit{⟨options⟩}.

\begin{tikzpicture}

% We have images%
\hex@dbg{5}{Terrain images}%
\pgfpointorigin\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%\foreach \i in \hex@t@image{%
  \hex@dbg{10}{Terrain image: ‘\meaning\i’}%
  \expandafter\wg@node{\includegraphics[width=2cm]{\i}}\@endwg@node %
  {\wg@tmpa,\wg@tmpb}{shape=rectangle,\anchor=center,\transform shape,\draw=none}}%
\end{tikzpicture}

Next, we define some example clippings of the terrain images. Specifically, we make clippings to sextants. We do this by first defining a macro.

\def\hex@x@r{.7}
\def\hex@make@sextants#1#2{\tikzset{pics/hex/#2sextant/.is choice,
pics/hex/#2sextant/north east/.style={
  code={
    \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;},
  pics/hex/#2sextant/north/.style={
    code={
      \path[pic actions]( 60:1)--(120:1)--(120:#1)--( 60:#1)--cycle;},
    pics/hex/#2sextant/north west/.style={
      code={
        \path[pic actions](120:1)--(180:1)--(180:#1)--(120:#1)--cycle;},
      pics/hex/#2sextant/south west/.style={
        code={
          \path[pic actions](180:1)--(120:1)--(120:#1)--(180:#1)--cycle;},
        pics/hex/#2sextant/south east/.style={
          code={
            \path[pic actions]( 60:1)--( 0:1)--( 0:#1)--( 60:1)--cycle;},
            pics/hex/#2sextant/east/.style={
              code={
                \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;},
                pics/hex/#2sextant/south/.style={
                  code={
                    \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;},
                      pics/hex/#2sextant/west/.style={
                        code={
                          \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;}}}}}}%}
\\\

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.

\def\hex@x@r{.7}
\def\hex@make@sextants#1#2{\tikzset{pics/hex/#2sextant/.is choice,
pics/hex/#2sextant/north east/.style={
  code={
    \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;},
  pics/hex/#2sextant/north/.style={
    code={
      \path[pic actions]( 60:1)--(120:1)--(120:#1)--( 60:#1)--cycle;},
    pics/hex/#2sextant/north west/.style={
      code={
        \path[pic actions](120:1)--(180:1)--(180:#1)--(120:#1)--cycle;},
      pics/hex/#2sextant/south west/.style={
        code={
          \path[pic actions](180:1)--(120:1)--(120:#1)--(180:#1)--cycle;},
          pics/hex/#2sextant/south east/.style={
            code={
              \path[pic actions]( 60:1)--( 0:1)--( 0:#1)--( 60:1)--cycle;},
                pics/hex/#2sextant/east/.style={
                  code={
                    \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;},
                    pics/hex/#2sextant/south/.style={
                      code={
                        \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;},
                          pics/hex/#2sextant/west/.style={
                            code={
                              \path[pic actions]( 0:1)--( 60:1)--( 60:#1)--( 0:#1)--cycle;}}}}}}}
Next, we define some styles for styling the terrain pictures. Users can change these styles (e.g., by appending to them) to change 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.

```
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.
\ifhex@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
(-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 (-0.2712, 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, 0.8977)
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) .. (0.6461, 0.3037)
The draw style for light woods. The pattern is filled with light green, and outline is not drawn.

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

Next, we have light woods.
\ifhex@terrain@pic
\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, 0.7856) -- (-0.4854, 0.7856) -- (-0.5190, 0.8023) -- controls (-0.5219, 0.7975) and (-0.5245, 0.7958) .. (-0.5272, 0.7916) -- (-0.5881, 0.6872) -- controls (-0.5849, 0.6876) and (-0.5819, 0.6876) .. (-0.5782, 0.6885) -- controls (-0.5524, 0.6946) and (-0.5387, 0.7153) .. (-0.5182, 0.7298) -- controls (-0.4841, 0.7540) and (-0.4420, 0.7539) .. (-0.4346, 0.7864) -- controls (-0.4295, 0.8088) and (-0.4470, 0.8265) .. (-0.4572, 0.8444) -- controls (-0.4631, 0.8549) and (-0.4670, 0.8646) .. (-0.4707, 0.8736) --cycle
( 0.3185, 0.8722)
( 0.3478, 0.8487) and (-0.3526, 0.8080) .. (-0.3290, 0.7808)
( 0.3440, 0.7633) and (-0.2394, 0.7433) .. (-0.2165, 0.7459)
( 0.1895, 0.7488) and (-0.1787, 0.7643) .. (-0.1561, 0.7725)
( 0.1380, 0.7791) and (-0.1179, 0.7766) .. (-0.1025, 0.7906)
( 0.1719, 0.8182) and (-0.0936, 0.8427) .. (-0.1240, 0.8528)
( 0.1323, 0.8192) -- (-0.1912, 0.8359)
( 0.1985, 0.8023) and (-0.1999, 0.7965) .. (-0.2332, 0.7856)
( 0.2512, 0.8363) and (-0.2775, 0.8009) .. (-0.2909, 0.8240)
( 0.2975, 0.8355) and (-0.2884, 0.8535) .. (-0.2756, 0.8719)
--cycle
(-0.1660, 0.8709)
( 0.1609, 0.8538) and (-0.1460, 0.8596) .. (-0.1371, 0.8707)
--cycle
( 0.0768, 0.8689)
( 0.0767, 0.8688) and ( 0.0765, 0.8686) .. ( 0.0764, 0.8685)
( 0.0704, 0.8503) and ( 0.0779, 0.7592) .. ( 0.1533, 0.7700)
( 0.1965, 0.7761) and ( 0.1956, 0.8018) .. ( 0.1871, 0.8359)
( 0.1366, 0.8108)
( 0.1510, 0.8683)
--cycle
( 0.1840, 0.8680)
( 0.1910, 0.8650) and ( 0.1993, 0.8662) .. ( 0.2081, 0.8678)
--cycle
( 0.2214, 0.8677)
( 0.2459, 0.7939)
( 0.2267, 0.7399) .. ( 0.2534, 0.7490)
( 0.2925, 0.8066) .. ( 0.2735, 0.8369)
( 0.2690, 0.8483) and ( 0.2655, 0.8586) .. ( 0.2619, 0.8674)\}
\begin{tikzpicture}

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

% Regular woods.
\ifhex@terrain@pic
  \node[hex/terrain/woods] at (0,0) {hex/terrain/woods} ;
\fi

\end{tikzpicture}

The style for woods. The pattern is filled with a darker green, and outlines are not drawn.
\begin{tikzpicture}
\path
\pgfextra{\pgfsetlinewidth{0.0pt}}
\pgfinterruptpicture
\begin{scope}[scale=0.15,transform shape,\pgfinterrupttikz]
\def\myvertex{1.4142135623730951, 1.224744871391589}
\filldraw \pgfextra{\pgfsetstrokecolor{white}} \myvertex;
\end{scope}
\end{pgfinterrupttikz}
\end{tikzpicture}
.. controls (0.6521,-0.3447) and (0.6439,-0.3431) .. (0.6371,-0.3426)
--cycle

(0.1158,-0.3640)
-- (0.1073,-0.3981)
-- (0.1415,-0.3981)
-- (0.1331,-0.3640)
--cycle

(-0.4660,-0.3701)
.. controls (-0.4757,-0.3670) and (-0.4894,-0.3727) .. (-0.4983,-0.3981)
-- (-0.5580,-0.3811)
.. controls (-0.5508,-0.4276) and (-0.5277,-0.4685) .. (-0.4812,-0.4833)
-- (-0.4898,-0.4066)
.. controls (-0.4444,-0.4042) and (-0.4498,-0.3750) .. (-0.4660,-0.3701)
--cycle

(0.1671,-0.3724)
-- (0.1585,-0.3811)
-- (0.1671,-0.4407)
-- (0.1927,-0.4322)
-- (0.1927,-0.3811)
-- (0.1841,-0.3724)
--cycle

(-0.4061,-0.3746)
.. controls (-0.4337,-0.3682) and (-0.4432,-0.4275) .. (-0.3933,-0.4421)
.. controls (-0.3807,-0.4474) and (-0.3733,-0.4433) .. (-0.3617,-0.4421)
.. controls (-0.3654,-0.4287) and (-0.3677,-0.4184) .. (-0.3740,-0.4070)
.. controls (-0.3858,-0.3861) and (-0.3969,-0.3767) .. (-0.4061,-0.3746)
--cycle

(-0.2091,-0.4066)
-- (-0.2091,-0.4298)
-- (-0.1315,-0.5004)
.. controls (-0.1221,-0.4446) and (-0.1606,-0.4228) .. (-0.2091,-0.4066)
--cycle

(0.6634,-0.4202)
.. controls (0.6063,-0.4164) and (0.5403,-0.4628) .. (0.5936,-0.5175)
.. controls (0.5533,-0.5676) and (0.6039,-0.5913) .. (0.6362,-0.5261)
-- (0.6021,-0.5175)
-- (0.6618,-0.5004)
-- (0.6874,-0.5261)
-- (0.6532,-0.4749)
-- (0.6618,-0.4664)
.. controls (0.6787,-0.4763) and (0.6924,-0.4939) .. (0.7031,-0.4938)
.. controls (0.7492,-0.4516) and (0.7588,-0.4217) .. (0.6869,-0.4249)
.. controls (0.6795,-0.4222) and (0.6716,-0.4207) .. (0.6634,-0.4202)
--cycle

(-0.2595,-0.4322)
-- (-0.2680,-0.4407)
-- (-0.2680,-0.4579)
-- (-0.2595,-0.4664)
-- (-0.2424,-0.4664)
-- (-0.2338,-0.4579)
-- (-0.2338,-0.4407)
-- (-0.2424,-0.4322)
--cycle
\begin{verbatim}
3923  --cycle
3924  ( 0.2950,-0.6455)
3925  .. controls ( 0.3615,-0.6477) and ( 0.3567,-0.6705) .. ( 0.3974,-0.7222)
3926  .. controls ( 0.3592,-0.8249) and ( 0.3353,-0.7947) .. ( 0.2609,-0.7990)
3927  -- ( 0.2438,-0.7649)
3928  .. controls ( 0.2681,-0.7474) and ( 0.2638,-0.7424) .. ( 0.2609,-0.7137)
3929  -- ( 0.2950,-0.6881)
3930  .. controls ( 0.3035,-0.6916) and ( 0.3102,-0.6959) .. ( 0.3198,-0.6974)
3931  .. controls ( 0.3496,-0.7020) and ( 0.3487,-0.6665) .. ( 0.3120,-0.6881)
3932  --cycle
3933  (-0.6122,-0.6460)
3934  .. controls (-0.6202,-0.6484) and (-0.6091,-0.6734) .. (-0.6220,-0.6591)
3935  -- (-0.6084,-0.6829)
3936  .. controls (-0.6039,-0.6880) and (-0.6001,-0.6940) .. (-0.5958,-0.6974)
3937  .. controls (-0.5822,-0.7078) and (-0.5979,-0.7175) .. (-0.5836,-0.7137)
3938  -- (-0.5557,-0.7397)
3939  .. controls (-0.5863,-0.7181) and (-0.4852,-0.7770) .. (-0.5233,-0.7812)
3940  .. controls (-0.4643,-0.7650) and (-0.4542,-0.8172) .. (-0.4400,-0.7940)
3941  .. controls (-0.4310,-0.7794) and (-0.4454,-0.7672) .. (-0.4569,-0.7609)
3942  .. controls (-0.5042,-0.7356) and (-0.5468,-0.7364) .. (-0.5580,-0.6711)
3943  .. controls (-0.5740,-0.6662) and (-0.5960,-0.6410) .. (-0.6122,-0.6460)
3944  --cycle
3945  (-0.2799,-0.6723)
3946  .. controls (-0.2946,-0.6977) and (-0.2726,-0.7009) .. (-0.2645,-0.6954)
3947  .. controls (-0.2563,-0.6899) and (-0.2508,-0.6683) .. (-0.2799,-0.6723)
3948  --cycle
3949  (-0.1741,-0.6796)
3950  -- (-0.1826,-0.6881)
3951  -- (-0.1826,-0.7052)
3952  -- (-0.1571,-0.6796)
3953  --cycle
3954  ( 0.0647,-0.7393)
3955  -- ( 0.0733,-0.7735)
3956  -- ( 0.0902,-0.7735)
3957  -- ( 0.0988,-0.7649)
3958  -- ( 0.0988,-0.7478)
3959  --cycle
3960  ( 0.2267,-0.7564)
3961  -- ( 0.2352,-0.7564)
3962  -- ( 0.2352,-0.7649)
3963  --cycle
3964  (~0.2765,-0.7649)
3965  -- (~0.2850,-0.7990)
3966  -- (~0.2509,-0.7905)
3967  -- (~0.2595,-0.7649)
3968  --cycle
3969  (~0.1620,-0.7999)
3970  .. controls (~0.1665,-0.7994) and (~0.1723,-0.7994) .. (~0.1792,-0.8003)
3971  .. controls (~0.2378,-0.8436) and (~0.2549,-0.8217) .. (~0.2658,-0.8441)
3972  .. controls (~0.2999,-0.8733) and (~0.2098,-0.8685) .. (~0.1746,-0.8466)
3973  .. controls (~0.1557,-0.8347) and (~0.1302,-0.8033) .. (~0.1620,-0.7999)
3974  --cycle
3975  (~0.0021,-0.8033)
\end{verbatim}

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

```
\tikzset{
  hex/terrain/swamp/.style={
    draw=none,
    fill={rgb,100:red,26;green,55;blue,70}
  }
}
```

Swamps. This is probably the shortest of the terrain patterns.

```
\ifhex@terrain@pic
  \tikzset{
    hex/terrain/swamp/.pic={
    ...
  }
\fi
```
\path[hex/terrain/swamp,pic actions,draw=none] (-0.5026, 0.8699) -- (-0.5041, 0.8672) .. controls (-0.3586, 0.8441) and (-0.1148, 0.8722) .. (0.0006, 0.8697) -- (0.2386, 0.8529) -- (0.2386, 0.8699) --cycle (0.4257, 0.8699) -- (0.4257, 0.8529) -- (0.5112, 0.8558) -- (0.5033, 0.8699) --cycle (0.3067, 0.8359) -- (0.2897, 0.7848) -- (0.2726, 0.8188) -- (0.2556, 0.8188) -- (0.2217, 0.7509) -- (0.5719, 0.7509) -- (0.5621, 0.7679) -- (0.5617, 0.7679) -- (0.3746, 0.7848) -- (0.3746, 0.8359) --cycle (-0.3225, 0.7848) -- (-0.3225, 0.7509) -- (0.0856, 0.7509) -- (0.0856, 0.7848) --cycle (-0.5555, 0.7782) -- (-0.5713, 0.7509) -- (-0.5097, 0.7509) --cycle (0.2789, 0.6696) .. controls (0.2234, 0.6713) and (0.1659, 0.6658) .. (0.1195, 0.6658) -- (-0.6117, 0.6658) -- (-0.6117, 0.6318) -- (0.4257, 0.6318) --cycle (0.6297, 0.6318) -- (0.6297, 0.5468) -- (0.5617, 0.5807) .. controls (0.5449, 0.5387) and (0.5194, 0.5474) .. (0.4764, 0.5468) -- (0.2047, 0.5468) .. controls (0.2857, 0.5146) and (0.5508, 0.5135) .. (0.7089, 0.5136) -- (0.6740, 0.5740) -- (0.6638, 0.5637) --cycle (-0.6684, 0.5591) .. controls (-0.6731, 0.5588) and (-0.6784, 0.5577) .. (-0.6832, 0.5571) -- (-0.6990, 0.5298) -- (-0.5777, 0.5298) .. controls (-0.6139, 0.5561) and (-0.6407, 0.5608) .. (-0.6684, 0.5591)
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}
  }
}

Roughs. Again, a bit long.
The image contains a diagram with coordinates and control points. The coordinates are given in a list format, specifying the x and y values for each point in the diagram. The control points are indicated with `.. controls` followed by the x and y values. The diagram appears to be a series of connected lines and curves defined by these points.
\begin{verbatim}
--cycle
(0.2485, 0.8268)
.. controls (0.2378, 0.8296) and (0.2250, 0.8213) .. (0.2103, 0.7930)
.. controls (0.2410, 0.7676) and (0.2451, 0.7555) .. (0.2867, 0.7591)
.. controls (0.2791, 0.7861) and (0.2665, 0.8220) .. (0.2485, 0.8268)
--cycle
(-0.3754, 0.8100)
-- (-0.3754, 0.7930)
-- (-0.3330, 0.7930)
-- (-0.3330, 0.8100)
--cycle
(0.5066, 0.8010)
-- (0.5131, 0.7667)
-- (0.5443, 0.7538)
-- (0.5566, 0.7611)
-- (0.5392, 0.7958)
--cycle
(-0.4008, 0.7930)
-- (-0.4348, 0.7591)
--cycle
(0.1509, 0.7930)
-- (0.1254, 0.7676)
.. controls (0.1432, 0.7361) and (0.1497, 0.7365) .. (0.1849, 0.7336)
.. controls (0.1820, 0.7688) and (0.1824, 0.7753) .. (0.1509, 0.7930)
--cycle
(0.0301, 0.7854)
.. controls (0.0240, 0.7861) and (0.0162, 0.7858) .. (0.0066, 0.7845)
-- (0.0490, 0.7421)
.. controls (0.0527, 0.7709) and (0.0486, 0.7831) .. (0.0301, 0.7854)
--cycle
(-0.2757, 0.7847)
.. controls (-0.2819, 0.7857) and (-0.2896, 0.7857) .. (-0.2990, 0.7845)
-- (-0.2820, 0.7411)
.. controls (-0.3010, 0.7423) and (-0.3576, 0.7485) .. (-0.3704, 0.7411)
.. controls (-0.3832, 0.7314) and (-0.3819, 0.7137) .. (-0.3644, 0.7089)
.. controls (-0.3522, 0.7029) and (-0.3199, 0.7069) .. (-0.3075, 0.7089)
.. controls (-0.2647, 0.7227) and (-0.2326, 0.7776) .. (-0.2757, 0.7847)
--cycle
(0.3631, 0.7676)
-- (0.3122, 0.7479)
.. controls (0.3064, 0.6995) and (0.3021, 0.7030) .. (0.3546, 0.7166)
.. controls (0.3550, 0.6777) and (0.3499, 0.6644) .. (0.3886, 0.6488)
.. controls (0.3854, 0.7398) and (0.3487, 0.6989) .. (0.3631, 0.7676)
--cycle
(0.5753, 0.7676)
.. controls (0.5837, 0.7354) and (0.5927, 0.7219) .. (0.6097, 0.7131)
-- (0.5796, 0.7669)
.. controls (0.5781, 0.7670) and (0.5768, 0.7674) .. (0.5753, 0.7676)
--cycle
(-0.5536, 0.7591)
-- (-0.5706, 0.7082)
-- (-0.5621, 0.6997)
-- (-0.5027, 0.6997)
\end{verbatim}
.. controls (0.5368, 0.5332) and (0.5357, 0.5327) .. (0.5244, 0.5044)
-- (0.5329, 0.4960)
-- (0.5668, 0.5044)
-- (0.5499, 0.4620)
.. controls (0.5250, 0.4718) and (0.5240, 0.4756) .. (0.5074, 0.4535)
-- (0.5584, 0.4280)
.. controls (0.6164, 0.4608) and (0.5800, 0.4918) .. (0.5668, 0.5384)
--cycle
(0.0770, 0.5310)
.. controls (0.0682, 0.5304) and (0.0588, 0.5222) .. (0.0538, 0.5053)
.. controls (0.0343, 0.4401) and (0.0794, 0.3794) .. (0.1169, 0.4450)
-- (0.0830, 0.4535)
.. controls (0.0904, 0.4711) and (0.1010, 0.4920) .. (0.0968, 0.5117)
.. controls (0.0941, 0.5249) and (0.0858, 0.5317) .. (0.0770, 0.5310)
--cycle
(-0.3075, 0.5299)
-- (-0.3414, 0.4790)
-- (-0.3330, 0.4705)
.. controls (-0.2926, 0.4813) and (-0.2724, 0.4931) .. (-0.3075, 0.5299)
--cycle
(-0.6105, 0.5210)
.. controls (-0.6292, 0.5286) and (-0.6359, 0.5102) .. (-0.6385, 0.4790)
-- (-0.5876, 0.5044)
.. controls (-0.5967, 0.5132) and (-0.6043, 0.5185) .. (-0.6105, 0.5210)
--cycle
(-0.6810, 0.5129)
.. controls (-0.6924, 0.5121) and (-0.7036, 0.5121) .. (-0.7147, 0.5086)
.. controls (-0.7151, 0.5085) and (-0.7153, 0.5083) .. (-0.7157, 0.5081)
-- (-0.7430, 0.4612)
.. controls (-0.7297, 0.4478) and (-0.7007, 0.4457) .. (-0.6860, 0.4801)
.. controls (-0.6815, 0.4906) and (-0.6819, 0.5019) .. (-0.6810, 0.5129)
--cycle
(-0.1462, 0.5129)
.. controls (-0.1949, 0.5129) and (-0.2098, 0.5207) .. (-0.2480, 0.4875)
-- (-0.2480, 0.4790)
-- (-0.2311, 0.4620)
-- (-0.1801, 0.4790)
-- (-0.1801, 0.4535)
-- (-0.1462, 0.4536)
--cycle
(0.0066, 0.5044)
-- (-0.0019, 0.4620)
.. controls (-0.0908, 0.4424) and (-0.0252, 0.3738) .. (0.0185, 0.4370)
.. controls (0.0238, 0.4448) and (0.0272, 0.4527) .. (0.0290, 0.4620)
.. controls (0.0322, 0.4784) and (0.0277, 0.4893) .. (0.0236, 0.5044)
--cycle
(-0.5118, 0.4944)
.. controls (-0.5315, 0.4962) and (-0.5506, 0.4944) .. (-0.5676, 0.4798)
.. controls (-0.5973, 0.4546) and (-0.5662, 0.4306) .. (-0.5676, 0.4033)
.. controls (-0.5862, 0.3806) and (-0.5896, 0.3679) .. (-0.5934, 0.3509)
.. controls (-0.6001, 0.3209) and (-0.5656, 0.2986) .. (-0.5452, 0.2838)
-- (-0.5621, 0.2498)
-- (-0.5282, 0.2498)
.. controls (-0.5172,-0.7077) and (-0.5409,-0.6942) .. (-0.5653,-0.6922)
--cycle
(-0.1547,-0.7099)
.. controls (-0.1719,-0.7103) and (-0.1811,-0.7133) .. (-0.1971,-0.7179)
.. controls (-0.1923,-0.7317) and (-0.1881,-0.7454) .. (-0.1792,-0.7574)
.. controls (-0.1162,-0.8422) and (-0.0444,-0.7079) .. (-0.1547,-0.7099)
--cycle
(-0.4348,-0.7179)
.. controls (-0.4394,-0.7549) and (-0.4359,-0.7636) .. (-0.4008,-0.7773)
--cycle
( 0.2358,-0.7179)
-- ( 0.2358,-0.7349)
-- ( 0.3037,-0.7349)
-- ( 0.3037,-0.7179)
--cycle
( 0.4649,-0.7179)
-- ( 0.4734,-0.7858)
.. controls ( 0.4612,-0.7900) and ( 0.4516,-0.7918) .. ( 0.4409,-0.8004)
.. controls ( 0.3964,-0.8360) and ( 0.4585,-0.8927) .. ( 0.4819,-0.8198)
-- ( 0.5159,-0.8283)
.. controls ( 0.5142,-0.8359) and ( 0.5131,-0.8417) .. ( 0.5125,-0.8468)
-- ( 0.5398,-0.8000)
.. controls ( 0.5329,-0.7638) and ( 0.5138,-0.7350) .. ( 0.4649,-0.7179)
--cycle
( 0.2103,-0.7264)
-- ( 0.1509,-0.7349)
-- ( 0.1509,-0.7688)
.. controls ( 0.1894,-0.7657) and ( 0.1982,-0.7645) .. ( 0.2103,-0.7264)
--cycle
(-0.2905,-0.7349)
.. controls (-0.3009,-0.7717) and (-0.3009,-0.7829) .. (-0.2905,-0.8198)
-- (-0.2480,-0.8028)
-- (-0.2311,-0.8367)
.. controls (-0.1820,-0.7845) and (-0.2454,-0.7805) .. (-0.2735,-0.7349)
--cycle
(-0.0019,-0.7349)
.. controls (-0.0202,-0.7845) and (-0.0471,-0.8007) .. (-0.0358,-0.8537)
-- ( 0.0066,-0.8113)
-- ( 0.0236,-0.8113)
.. controls ( 0.0412,-0.8384) and ( 0.0421,-0.8410) .. ( 0.0745,-0.8367)
.. controls ( 0.0599,-0.7914) and ( 0.0500,-0.7437) .. (-0.0019,-0.7349)
--cycle
(-0.5282,-0.7688)
.. controls (-0.5349,-0.8205) and (-0.5012,-0.8219) .. (-0.4603,-0.8113)
-- (-0.4603,-0.7943)
--cycle
( 0.3122,-0.7688)
-- ( 0.3037,-0.7773)
.. controls ( 0.3122,-0.8236) and ( 0.3093,-0.8598) .. ( 0.3608,-0.8698)
-- ( 0.3628,-0.8698)
-- ( 0.3292,-0.7688)
--cycle
(-0.3584,-0.7858)
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.

```latex
\begin{tikzpicture}
\pgfmathsetmacro\myHexSize{0.5}
\foreach \i in {0,...,11}
\foreach \j in {0,...,11}
\fill[\terrainPattern\fillColor] ([x=\myHexSize,y=\myHexSize]\i,\j) -- ([x=\myHexSize,y=\myHexSize]\i+1,\j) -- ([x=\myHexSize,y=\myHexSize]\i+1,\j+1) -- ([x=\myHexSize,y=\myHexSize]\i,\j+1) -- cycle;
\end{tikzpicture}
```

And the mountains pattern. This is the same as the beach pattern, only filled with a darker brown colour.
\begin{verbatim}
--cycle
(-0.5641,-0.6998)
-- controls (-0.5850,-0.6984) and (-0.5808,-0.7367) .. (-0.5766,-0.7507)
-- controls (-0.5748,-0.7566) and (-0.5723,-0.7598) .. (-0.5702,-0.7648)
-- controls (-0.5310,-0.8247) and (-0.5120,-0.7963) .. (-0.5052,-0.8022)
.. controls (-0.5344,-0.7394) and (-0.5352,-0.7120) .. (-0.5535,-0.7030)
.. controls (-0.5576,-0.7010) and (-0.5611,-0.7000) .. (-0.5641,-0.6998)
-- cycle
( 0.1990,-0.7341)
.. controls ( 0.1094,-0.7768) and ( 0.2330,-0.8330) .. ( 0.2586,-0.8828)
-- controls ( 0.3183,-0.8833)
.. controls ( 0.3165,-0.8684) and ( 0.3066,-0.8565) .. ( 0.2991,-0.8428)
.. controls ( 0.2762,-0.8010) and ( 0.2508,-0.7418) .. ( 0.1990,-0.7341)
-- cycle
( 0.3603,-0.7592)
-- ( 0.3859,-0.8188)
.. controls ( 0.1094,-0.7768) and ( 0.2330,-0.8330) .. ( 0.2586,-0.8828)
-- controls ( 0.4178,-0.7853) and ( 0.4108,-0.7527) .. ( 0.3603,-0.7592)
-- cycle
( 0.4369,-0.8443)
.. controls ( 0.4147,-0.8480) and ( 0.3837,-0.8661) .. ( 0.3628,-0.8837)
-- controls ( 0.4240,-0.8842)
-- controls ( 0.4307,-0.8690) and ( 0.4358,-0.8541) .. ( 0.4369,-0.8443)
-- cycle
(-0.3205,-0.8528)
-- controls ( 0.0964,-0.8568) and ( 0.0834,-0.8587) .. ( 0.0710,-0.8600)
-- controls ( 0.0605,-0.8611) and ( 0.0403,-0.8617) .. ( 0.0312,-0.8664)
-- controls ( 0.0240,-0.8701) and ( 0.0203,-0.8751) .. ( 0.0184,-0.8808)
-- cycle
(-0.3266,-0.8779)
-- controls ( 0.0964,-0.8568) and ( 0.0834,-0.8587) .. ( 0.0710,-0.8600)
-- controls ( 0.0605,-0.8611) and ( 0.0403,-0.8617) .. ( 0.0312,-0.8664)
-- controls ( 0.0240,-0.8701) and ( 0.0203,-0.8751) .. ( 0.0184,-0.8808)
-- cycle
(-0.2773,-0.8783)
-- controls ( 0.0964,-0.8568) and ( 0.0834,-0.8587) .. ( 0.0710,-0.8600)
-- controls ( 0.0605,-0.8611) and ( 0.0403,-0.8617) .. ( 0.0312,-0.8664)
-- controls ( 0.0240,-0.8701) and ( 0.0203,-0.8751) .. ( 0.0184,-0.8808)
-- cycle
(-0.2800,-0.8719)
-- controls ( 0.0964,-0.8568) and ( 0.0834,-0.8587) .. ( 0.0710,-0.8600)
-- controls ( 0.0605,-0.8611) and ( 0.0403,-0.8617) .. ( 0.0312,-0.8664)
-- controls ( 0.0240,-0.8701) and ( 0.0203,-0.8751) .. ( 0.0184,-0.8808)
-- cycle
(-0.2850,-0.8655)
-- controls ( 0.0964,-0.8568) and ( 0.0834,-0.8587) .. ( 0.0710,-0.8600)
-- controls ( 0.0605,-0.8611) and ( 0.0403,-0.8617) .. ( 0.0312,-0.8664)
-- controls ( 0.0240,-0.8701) and ( 0.0203,-0.8751) .. ( 0.0184,-0.8808)
-- cycle
( 0.1093,-0.8568)
.. controls ( 0.1050,-0.8698)
-- ( 0.1085,-0.8815)
-- ( 0.1641,-0.8820)
.. controls ( 0.1606,-0.8757) and ( 0.1553,-0.8698) .. ( 0.1463,-0.8649)
.. controls ( 0.1347,-0.8586) and ( 0.1221,-0.8568) .. ( 0.1093,-0.8568)
-- cycle
\end{verbatim}

hex/terrain/town/road
hex/terrain/town/small road
hex/terrain/town/house
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{
  hex/terrain/town/road/.style={
    fill=none,
    draw=gray!50!black,
    scale line widths,
    line width=.3mm
  },
  hex/terrain/town/small road/.style={
    fill=none,
    draw=gray!75!black,
    scale line widths,
    line width=.15mm
  },
  hex/terrain/town/post road/.style={
    fill=none
  },
  hex/terrain/town/house/.style=
    draw=none,
    fill=gray!75!black,
  }
}

hex/terrain/village

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)
      -- ( 0.3884, 0.2029) -- ( 0.3555, 0.1378)
      -- ( 0.3751, 0.0880) -- ( 0.2513,-0.1997)
      -- ( 0.1396,-0.4505) -- ( 0.0641,-0.6512)
      -- ( 0.0070,-0.8700) ;
    \path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
      ( 0.7575, 0.4367) -- ( 0.3945, 0.3375)
\path[hex/terrain/town/house,pic actions] (0.2901, 0.6234) -- (0.3088, 0.6008) -- (0.2674, 0.5667) -- (0.2487, 0.5893) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3456, 0.2854) -- (-0.3335, 0.3120) -- (-0.2847, 0.2898) -- (-0.2968, 0.2632) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.6678,-0.1369) -- (-0.6492,-0.1143) -- (-0.6078,-0.1484) -- (-0.6264,-0.1710) --cycle;
\path[hex/terrain/town/house,pic actions] (0.4610, 0.0967) -- (0.4896, 0.0909) -- (0.4790, 0.0384) -- (0.4503, 0.0442) --cycle;
\path[hex/terrain/town/house,pic actions] (0.4610, 0.0967) -- (0.4896, 0.0909) -- (0.4790, 0.0384) -- (0.4503, 0.0442) --cycle;
\path[hex/terrain/town/house,pic actions] (0.2924,-0.1375) -- (0.3110,-0.0955)
A town.

\ifhex@terrain@pic
\tikzset{
  \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);
    \path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
    ( 0.4358,-0.4688);
    \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
    \path[hex/terrain/town/house,pic actions]
    ( 0.3746, 0.1600)
    -- ( 0.4021, 0.1699)
    -- ( 0.4204, 0.1195)
    -- ( 0.3929, 0.1095)
    --cycle
  }
}\fi
\path[hex/terrain/town/house,pic actions] (-0.0207,-0.0604) -- (-0.0103,-0.0331) -- ( 0.0398,-0.0521) -- ( 0.0294,-0.0794) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.3580,-0.4608) -- ( 0.3837,-0.4748) -- ( 0.3581,-0.5219) -- ( 0.3324,-0.5080) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.5127,-0.3559) -- ( 0.5341,-0.3759) -- ( 0.4975,-0.4151) -- ( 0.4761,-0.3951) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2118,-0.3884) -- ( 0.2245,-0.3620) -- ( 0.2728,-0.3854) -- ( 0.2600,-0.4118) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2118,-0.3884) -- ( 0.2245,-0.3620) -- ( 0.2728,-0.3854) -- ( 0.2600,-0.4118) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.1651,-0.4740) -- ( 0.1775,-0.4475)
\path[hex/terrain/town/house,pic actions]
( 0.1931, -0.0936)
-- ( 0.2058, -0.0673)
-- ( 0.2541, -0.0904)
-- ( 0.2415, -0.1168)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1779, 0.1198)
-- ( 0.1984, 0.0990)
-- ( 0.1603, 0.0613)
-- ( 0.1398, 0.0821)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2533, 0.3192)
-- ( 0.2447, 0.3326)
-- ( 0.2947, 0.3765)
-- ( 0.2541, 0.3425)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3020, 0.2974)
-- ( 0.2826, 0.3192)
\path[hex/terrain/town/house,pic actions]
( 0.3227, 0.3548)
-- ( 0.3421, 0.3329)
-- ( 0.3020, 0.2974)
-- ( 0.2826, 0.3192)
--cycle

\path[hex/terrain/town/house,pic actions]
( -0.2473, 0.2770)
-- (-0.2380, 0.3048)
-- (-0.1871, 0.2879)
-- (-0.1964, 0.2601)
--cycle

\path[hex/terrain/town/house,pic actions]
( -0.1395, 0.3602)
-- (-0.1127, 0.3488)
-- (-0.1335, 0.2995)
-- (-0.1604, 0.3109)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.2901, 0.6234)
-- ( 0.3088, 0.6008)
-- ( 0.2674, 0.5667)
-- ( 0.2487, 0.5893)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.2901, 0.6234)
-- ( 0.3088, 0.6008)
-- ( 0.2674, 0.5667)
-- ( 0.2487, 0.5893)
--cycle

\path[hex/terrain/town/house,pic actions]
( -0.3456, 0.2854)
\path[hex/terrain/town/house,pic actions]
( 0.2924, -0.1375)
\path[hex/terrain/town/house,pic actions]
(-0.3062, 0.5810)
\path[hex/terrain/town/house,pic actions]
( 0.1523, -0.2013)
\path[hex/terrain/town/house,pic actions]
( 0.0857, -0.3676)
\path[hex/terrain/town/house,pic actions]
(0.3110, -0.0955)
\path[hex/terrain/town/house,pic actions]
(-0.2635, 0.5641)
\path[hex/terrain/town/house,pic actions]
(-0.2859, 0.5075)
\path[hex/terrain/town/house,pic actions]
(-0.3285, 0.5243)
\path[hex/terrain/town/house,pic actions]
(0.3667, -0.1202)
\path[hex/terrain/town/house,pic actions]
(-0.2859, 0.5075)
\path[hex/terrain/town/house,pic actions]
(-0.3285, 0.5243)
\path[hex/terrain/town/house,pic actions]
(0.3481, -0.1621)
\path[hex/terrain/town/house,pic actions]
(-0.3285, 0.5243)
\path[hex/terrain/town/house,pic actions]
(0.3110, -0.0955)
\path[hex/terrain/town/house,pic actions]
(-0.3285, 0.5243)
\path[hex/terrain/town/house,pic actions]
(0.2859, 0.5075)
\path[hex/terrain/town/house,pic actions]
(-0.2635, 0.5641)
\path[hex/terrain/town/house,pic actions]
(-0.3062, 0.5810)
\path[hex/terrain/town/house,pic actions]
(0.1029, -0.4409)
\path[hex/terrain/town/house,pic actions]
(0.1603, -0.3520)
\path[hex/terrain/town/house,pic actions]
(0.0889, -0.4846)
\path[hex/terrain/town/house,pic actions]
(0.0449, -0.4224)
\path[hex/terrain/town/house,pic actions]
(0.0310, -0.4661)
\path[hex/terrain/town/house,pic actions]
(0.1718, -0.1598)
\path[hex/terrain/town/house,pic actions]
(0.1523, -0.2013)
\path[hex/terrain/town/house,pic actions]
(0.0889, -0.4846)
\path[hex/terrain/town/house,pic actions]
(0.1029, -0.4409)
\path[hex/terrain/town/house,pic actions]
(0.0449, -0.4224)
\path[hex/terrain/town/house,pic actions]
(0.0310, -0.4661)
\path[hex/terrain/town/house,pic actions]
(0.1718, -0.1598)
\path[hex/terrain/town/house,pic actions]
(0.1523, -0.2013)
\path[hex/terrain/town/house,pic actions]
(0.1718, -0.1598)
\path[hex/terrain/town/house,pic actions]
(0.0857, -0.3676)
\path[hex/terrain/town/house,pic actions]
(0.1718, -0.1598)
\path[hex/terrain/town/house,pic actions]
(0.1523, -0.2013)
\path[hex/terrain/town/house,pic actions]
(0.1718, -0.1598)
\path[hex/terrain/town/house,pic actions]
(0.1523, -0.2013)
\path[hex/terrain/town/house,pic actions] (0.0920,-0.6296) -- (0.1069,-0.5863) -- (0.1645,-0.6060) -- (0.1497,-0.6493) --cycle;
\path[hex/terrain/town/house,pic actions] (0.0920,-0.6296) -- (0.1069,-0.5863) -- (0.1645,-0.6060) -- (0.1497,-0.6493) --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] (0.4115,-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;
\begin{tikzpicture}
\begin{scope}[hex/terrain/town/house,pic actions]
\path (0.2904, 0.2714) -- (0.3361, 0.2681) -- (0.3318, 0.2074) -- (0.2861, 0.2106) -- cycle;
\path (-0.0124, 0.1558) -- (0.0333, 0.1525) -- (0.0290, 0.0918) -- (-0.0167, 0.0950) -- cycle;
\path (0.4665, 0.4396) -- (0.4868, 0.3985) -- (0.4321, 0.3716) -- (0.4119, 0.4127) -- cycle;
\path (-0.2433, -0.1480) -- (-0.2141, -0.1472) -- (-0.2127, -0.2008) -- (-0.2419, -0.2015) -- cycle;
\end{scope}
\end{tikzpicture}
And finally a city
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0040, -0.2086)
-- (-0.1956, -0.1835)
-- (-0.2290, -0.2265)
-- (-0.5068, -0.1372)
;
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
(-0.3396, 0.0437)
-- (-0.2201, 0.4506)
;
\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)
;
\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)
;
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( 0.0723, -0.0367)
-- (-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)
;
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
\path[hex/terrain/town/house,pic actions]
(-0.3878, 0.1398)
-- (-0.3421, 0.1442)
-- (-0.3362, 0.0834)
-- (-0.3819, 0.0790)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.5622, 0.0806)
-- (-0.5234, 0.0560)
-- (-0.5559, 0.0044)
-- (-0.5948, 0.0290)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.6218, 0.1903)
-- (-0.6097, 0.2346)
-- (-0.5508, 0.2185)
-- (-0.5629, 0.1742)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.2884, 0.7423)
-- (-0.2596, 0.7372)
-- (-0.2691, 0.6843)
-- (-0.2980, 0.6896)
--cycle
;
\path[hex/terrain/town/house,pic actions] (0.1936, 0.5180)
-- (0.2045, 0.4909)
-- (0.1547, 0.4709)
-- (0.1437, 0.4981)
--cycle
;
\path[hex/terrain/town/house,pic actions] (0.1936, 0.5180)
-- (0.2045, 0.4909)
-- (0.1547, 0.4709)
-- (0.1437, 0.4981)
--cycle
;
\path[hex/terrain/town/house,pic actions] (-0.1903, 0.4858)
-- (-0.1678, 0.5258)
-- (-0.1146, 0.4960)
-- (-0.1371, 0.4559)
--cycle
;
\path[hex/terrain/town/house,pic actions] (-0.1903, 0.4858)
-- (-0.1678, 0.5258)
-- (-0.1146, 0.4960)
-- (-0.1371, 0.4559)
--cycle
;
\path[hex/terrain/town/house,pic actions] (0.1470, 0.3493)
-- (0.1689, 0.3567)
-- (0.1775, 0.3315)
-- (0.1555, 0.3241)
--cycle
;
\path[hex/terrain/town/house,pic actions] (0.1470, 0.3493)
-- (0.1689, 0.3567)
-- (0.1775, 0.3315)
-- (0.1555, 0.3241)
--cycle
;
\path[hex/terrain/town/house,pic actions] (0.1892, 0.2562)
-- (0.2118, 0.2510)
-- (0.2058, 0.2251)
-- (0.1833, 0.2303)
--cycle
\path[hex/terrain/town/house,pic actions]
\path (-0.4808,-0.6251) -- (-0.4706,-0.6459) -- (-0.4946,-0.6576) -- (-0.5047,-0.6368) --cycle;
\path[hex/terrain/town/house,pic actions]
\path (-0.4514,-0.6075) -- (-0.4393,-0.6272) -- (-0.4620,-0.6412) -- (-0.4740,-0.6215) --cycle;
\path[hex/terrain/town/house,pic actions]
\path (-0.2390,-0.7174) -- (-0.2175,-0.7260) -- (-0.2273,-0.7508) -- (-0.2489,-0.7421) --cycle;
\path[hex/terrain/town/house,pic actions]
\path (-0.2386,-0.5506) -- (-0.2108,-0.5598) -- (-0.2277,-0.6108) -- (-0.2555,-0.6015) --cycle;
\path[hex/terrain/town/house,pic actions]
\path (-0.2386,-0.5506) -- (-0.2108,-0.5598) -- (-0.2277,-0.6108) -- (-0.2555,-0.6015) --cycle;
\path\[hex/terrain/town/house,pic actions\]
(-0.2282,-0.7907)
-- (-0.1823,-0.7907)
-- (-0.1823,-0.8518)
-- (-0.2282,-0.8518)
--cycle
;
\path\[hex/terrain/town/house,pic actions\]
( 0.2275,-0.7989)
-- ( 0.2734,-0.7989)
-- ( 0.2734,-0.8599)
-- ( 0.2275,-0.8599)
--cycle
;
\path\[hex/terrain/town/house,pic actions\]
( 0.2516,-0.7126)
-- ( 0.2808,-0.7126)
-- ( 0.2808,-0.7663)
-- ( 0.2516,-0.7663)
--cycle
;
\path\[hex/terrain/town/house,pic actions\]
( 0.2516,-0.7126)
-- ( 0.2808,-0.7126)
-- ( 0.2808,-0.7663)
-- ( 0.2516,-0.7663)
--cycle
;
\path\[hex/terrain/town/house,pic actions\]
( 0.2516,-0.7126)
-- ( 0.2808,-0.7126)
-- ( 0.2808,-0.7663)
-- ( 0.2516,-0.7663)
--cycle
;
\path\[hex/terrain/town/house,pic actions\]
( 0.1669,-0.7129)
-- ( 0.1954,-0.7199)
-- ( 0.1826,-0.7721)
-- ( 0.1542,-0.7650)

210
\path[hex/terrain/town/house, pic actions]
( 0.0205,-0.5124)
-- ( 0.0287,-0.4843)
-- ( 0.0803,-0.4994)
-- ( 0.0720,-0.5275)
--cycle
;
\path[hex/terrain/town/house, pic actions]
( 0.0205,-0.5124)
-- ( 0.0287,-0.4843)
-- ( 0.0803,-0.4994)
-- ( 0.0720,-0.5275)
--cycle
;
\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]
(-0.0719,-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]
(-0.1330,-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]
(-0.0334,-0.7381)
\path[hex/terrain/town/house, pic actions] (-0.4189, -0.5000)
\path[hex/terrain/town/house, pic actions] (-0.3561, -0.4332)
\path[hex/terrain/town/house, pic actions] (-0.3120, -0.4787)
\path[hex/terrain/town/house, pic actions] (-0.2660, -0.5113)
\path[hex/terrain/town/house,pic actions]
( 0.2587,-0.2944)
-- ( 0.2585,-0.2651)
-- ( 0.3123,-0.2648)
-- ( 0.3125,-0.2941)
--cycle;
\path[hex/terrain/town/house,pic actions]
( 0.1269,-0.2581)
-- ( 0.1359,-0.2303)
-- ( 0.1871,-0.2468)
-- ( 0.1780,-0.2746)
--cycle;
\path[hex/terrain/town/house,pic actions]
( 0.1127,-0.3153)
-- ( 0.1232,-0.2880)
-- ( 0.1733,-0.3074)
-- ( 0.1626,-0.3348)
--cycle;
\path[hex/terrain/town/house,pic actions]
( 0.1928,-0.0951)
-- ( 0.2055,-0.0687)
-- ( 0.2539,-0.0920)
-- ( 0.2412,-0.1184)
--cycle;
\path[hex/terrain/town/house,pic actions]
( 0.1928,-0.0951)
-- ( 0.2055,-0.0687)
\path[hex/terrain/town/house,pic actions]
( 0.1776, 0.1187)
-- ( 0.1982, 0.0978)
-- ( 0.1599, 0.0601)
-- ( 0.1394, 0.0810)
cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2760, 0.3987)
-- ( 0.2946, 0.3760)
-- ( 0.2531, 0.3420)
-- ( 0.2345, 0.3646)
cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2760, 0.3987)
-- ( 0.2946, 0.3760)
-- ( 0.2531, 0.3420)
-- ( 0.2345, 0.3646)
cycle
;
\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]
( 0.3226, 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]
(-0.2277, 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) --cycle
\path[hex/terrain/town/house,pic actions] (-0.2400, 0.3081)
-- (-0.2307, 0.3359)
-- (-0.1797, 0.3189)
-- (-0.1890, 0.2912) --cycle
\path[hex/terrain/town/house,pic actions] (-0.2735, 0.1997)
-- (-0.2631, 0.2270)
-- (-0.2129, 0.2080)
-- (-0.2233, 0.1807) --cycle
\path[hex/terrain/town/house,pic actions] (-0.2735, 0.1997)
-- (-0.2631, 0.2270)
-- (-0.2129, 0.2080)
-- (-0.2233, 0.1807) --cycle
\path[hex/terrain/town/house,pic actions] (-0.3047, 0.1045)
-- (-0.2975, 0.1329)
-- (-0.2455, 0.1200)
-- (-0.2525, 0.0915) --cycle
\path[hex/terrain/town/house,pic actions] (-0.3047, 0.1045)
-- (-0.2975, 0.1329)
-- (-0.2455, 0.1200)
-- (-0.2525, 0.0915) --cycle
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
( 0.4104, 0.6743)
-- ( 0.4255, 0.6491)
-- ( 0.3795, 0.6215)
-- ( 0.3644, 0.6466)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.4437, 0.6203)
-- ( 0.4592, 0.5954)
-- ( 0.4136, 0.5671)
-- ( 0.3981, 0.5918)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.5275, 0.4892)
-- ( 0.5449, 0.4656)
-- ( 0.5018, 0.4337)
-- ( 0.4844, 0.4571)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.5275, 0.4892)
-- ( 0.5449, 0.4656)
-- ( 0.5018, 0.4337)
-- ( 0.4844, 0.4571)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.2900, 0.6234)
-- ( 0.3087, 0.6008)
-- ( 0.2671, 0.5666)
-- ( 0.2485, 0.5892)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.2900, 0.6234)
-- ( 0.3087, 0.6008)
-- ( 0.2671, 0.5666)
-- ( 0.2485, 0.5892)
--cycle

\path[hex/terrain/town/house,pic actions]
( 0.3295, 0.5855)

\path[hex/terrain/town/house,pic actions]
( 0.3295, 0.5855)
-- ( 0.3477, 0.5626)
\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]
( 0.8604, 0.2135)
-- ( 0.8734, 0.1872)
-- ( 0.8254, 0.1634)
-- ( 0.8123, 0.1896)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7675, 0.2368)
-- ( 0.7736, 0.2082)
-- ( 0.7210, 0.1970)
-- ( 0.7150, 0.2257)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7696,-0.1796)
-- ( 0.7978,-0.1875)
-- ( 0.7835,-0.2392)
-- ( 0.7552,-0.2314)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7546,-0.0830)
-- ( 0.7838,-0.0830)
-- ( 0.7838,-0.1367)
-- ( 0.7546,-0.1367)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7546,-0.0830)
-- ( 0.7838,-0.0830)
-- ( 0.7838,-0.1367)
-- ( 0.7546,-0.1367)
--cycle
\path[hex/terrain/town/house,pic actions] ( 0.7114,-0.1735) -- ( 0.7402,-0.1784) -- ( 0.7313,-0.2313) -- ( 0.7024,-0.2265) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.7114,-0.1735) -- ( 0.7402,-0.1784) -- ( 0.7313,-0.2313) -- ( 0.7024,-0.2265) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.6398,-0.0896) -- ( 0.6691,-0.0896) -- ( 0.6691,-0.1433) -- ( 0.6398,-0.1433) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.6398,-0.0896) -- ( 0.6691,-0.0896) -- ( 0.6691,-0.1433) -- ( 0.6398,-0.1433) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.5390,-0.1093) -- ( 0.5377,-0.0801) -- ( 0.5913,-0.0776) -- ( 0.5927,-0.1069) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.5390,-0.1093) -- ( 0.5377,-0.0801) -- ( 0.5913,-0.0776) -- ( 0.5927,-0.1069) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.5171,-0.2250) -- ( 0.5252,-0.1968) -- ( 0.5767,-0.2117) -- ( 0.5687,-0.2399) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.5171,-0.2250) -- ( 0.5252,-0.1968)
\path[hex/terrain/town/house,pic actions]
(-0.1783,-0.5367)
-- (-0.1339,-0.5483)
-- (-0.1492,-0.6074)
-- (-0.1937,-0.5958)
--cycle

\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]
( 0.0570,-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]

\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)
-- (-0.4283,-0.7899)
-- (-0.4151,-0.8496)
-- (-0.4600,-0.8595)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4125,-0.0879)
-- ( 0.4578,-0.0951)
-- ( 0.4483,-0.1553)
-- ( 0.4029,-0.1481)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2078, 0.8568)
-- ( 0.2536, 0.8583)
-- ( 0.2555, 0.7973)
-- ( 0.2097, 0.7958)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2078, 0.8568)
-- ( 0.2536, 0.8583)
-- ( 0.2555, 0.7973)
-- ( 0.2097, 0.7958)
--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]
(-0.0764, 0.3204)
-- (-0.0539, 0.3604)
-- (-0.0007, 0.3306)
-- (-0.0232, 0.2905)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1364, 0.0430)
-- (-0.1139, 0.0831)
-- (-0.0607, 0.0532)
-- (-0.0832, 0.0131)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1364, 0.0430)
-- (-0.1139, 0.0831)
-- (-0.0607, 0.0532)
-- (-0.0832, 0.0131)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1269, 0.1239)
-- (-0.1149, 0.1681)
-- (-0.0560, 0.1521)
-- (-0.0681, 0.1078)
--cycle
\begin{tikzpicture}

% A house in hex grid format
\filldraw[hex/terrain/town/house, pic actions] (0.2218, -0.5853) -- (0.2366, -0.5418) -- (0.2944, -0.5615) -- (0.2796, -0.6051) -- cycle;
\filldraw[hex/terrain/town/house, pic actions] (0.3094, 0.1262) -- (0.3519, 0.1085) -- (0.3284, 0.0522) -- (0.2860, 0.0698) -- cycle;
\filldraw[hex/terrain/town/house, pic actions] (0.2797, 0.1784) -- (0.3041, 0.1395) -- (0.2524, 0.1070) -- (0.2280, 0.1459) -- cycle;
\filldraw[hex/terrain/town/house, pic actions] (0.7950, 0.1548) -- (0.8065, 0.1103) -- (0.7475, 0.0949) -- (0.7359, 0.1394) -- cycle;
\filldraw[hex/terrain/town/house, pic actions] (0.7950, 0.1548) -- (0.8065, 0.1103) -- (0.7475, 0.0949) -- (0.7359, 0.1394) -- cycle;
\filldraw[hex/terrain/town/house, pic actions] (0.7950, 0.1548) -- (0.8065, 0.1103) -- (0.7475, 0.0949) -- (0.7359, 0.1394) -- cycle;
\filldraw[hex/terrain/town/house, pic actions] (0.5739, 0.6926) -- (0.5961, 0.6525) -- (0.5427, 0.6229) -- (0.5205, 0.6632) -- cycle;

\end{tikzpicture}
\path[hex/terrain/town/house,pic actions]
( 0.2434, 0.4429)
-- ( 0.2698, 0.4054)
-- ( 0.2199, 0.3702)
-- ( 0.1935, 0.4077)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4777, 0.5914)
-- ( 0.4980, 0.5501)
-- ( 0.4432, 0.5232)
-- ( 0.4229, 0.5644)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4777, 0.5914)
-- ( 0.4980, 0.5501)
-- ( 0.4432, 0.5232)
-- ( 0.4229, 0.5644)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4936, 0.5331)
-- ( 0.5191, 0.4949)
-- ( 0.4683, 0.4611)
-- ( 0.4428, 0.4993)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4936, 0.5331)
-- ( 0.5191, 0.4949)
-- ( 0.4683, 0.4611)
-- ( 0.4428, 0.4993)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4667, 0.4393)
-- ( 0.4871, 0.3980)
-- ( 0.4323, 0.3711)
-- ( 0.4120, 0.4123)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4667, 0.4393)
-- ( 0.4871, 0.3980)
-- ( 0.4323, 0.3711)
-- ( 0.4120, 0.4123)
--cycle
;
\path[hex/terrain/town/house,pic actions]
\begin{verbatim}
(0.4645, 0.2567)
\end{verbatim}
\path[hex/terrain/town/house,pic actions]
( 0.5509, 0.4874)
-- ( 0.5966, 0.4925)
-- ( 0.6034, 0.4319)
-- ( 0.5577, 0.4267)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1390, 0.6195)
-- ( 0.1654, 0.5820)
-- ( 0.1155, 0.5468)
-- ( 0.0890, 0.5843)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( -0.1780,-0.4082)
-- (-0.1533,-0.3695)
-- (-0.1018,-0.4023)
-- (-0.1265,-0.4410)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.2611,-0.2396)
-- (-0.2175,-0.2543)
-- (-0.2370,-0.3121)
-- (-0.2805,-0.2974)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.2611,-0.2396)
\path[hex/terrain/town/house,pic actions]
( 0.1640,-0.8299)
-- ( 0.1872,-0.8299)
-- ( 0.1872,-0.8565)
-- ( 0.1640,-0.8565)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1330,-0.7413)
-- (-0.1099,-0.7413)
-- (-0.1099,-0.7679)
-- (-0.1330,-0.7679)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3280,-0.8061)
-- (-0.3049,-0.8061)
-- (-0.3049,-0.8327)
-- (-0.3280,-0.8327)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.3280,-0.8061)
-- (-0.3049,-0.8061)
-- (-0.3049,-0.8327)
-- (-0.3280,-0.8327)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.7302,-0.0754)
-- (-0.7099,-0.0866)
-- (-0.7228,-0.1099)
-- (-0.7430,-0.0988)
--cycle
\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]
hex/terrain/mountain

This is an example of a terrain picture.

\begin{tikzpicture}

\path[draw=black, fill=white] (0,0) -- (.3,.9) -- (.45,0) -- cycle;
\path[draw=black, fill=lightgray, pic actions] (-.6,-.9) -- (-.3,.3) -- (0,0) -- (.45,0) -- (.6,-.9) -- cycle;

\end{tikzpicture}

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 \texttt{wave} decoration.

294
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.

\begin{verbatim}
\tikzset{
  hex/ridges pre/.style={
    line cap=round,
    draw=pgfstrokecolor,
    solid,
    /hex/ridges/.cd,
    radius=0.85,\%
    n=4,
    R=.25,
  },
  hex/ridges/.style={
    get scale,
    decoration={
      path has corners=true,
      waves,
      radius=ug\scale\hex@r@R,
      segment length=ug\scale\hex@r@s,
    },
    decorate}
}
\end{verbatim}

To properly set up the ridges, we need to concatenate ridge paths in order. To facilitate that, we define 6 \if s — one for each edge.

\begin{verbatim}
\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 \if s to true. We put these into the family /hex/r so that we can parse them separately.
\end{verbatim}

\begin{verbatim}
\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,
  radius/.store in=\hex@r@r,
  curve radius/.store in=\hex@r@w,
  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,
}\end{verbatim}
\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.

\begin{verbatim}
\edef\hex@r@tmp{[hex/ridges pre, /tikz/every hex ridges/.try, \hex@ridges]}
\expandafter\scope\hex@r@tmp%
\hex@dbg{3}{Ridges: '\meaning\hex@ridges', '\meaning\hex@r@tmp'}
\ifhex@r@ne
\ifhex@r@se
\xdef\hex@r@p{(0:\hex@r@r)--(60:\hex@r@r)}
\else
\xdef\hex@r@p{($(0:\hex@r@r)+(-60:\hex@t/2)$)--(60:\hex@r@r)}
\fi
\hex@dbg{4}{Ridge along north east edge: '\hex@r@p'}
\fi
\ifhex@r@n
\ifhex@r@ne\else
\xdef\hex@r@p{\hex@r@p ($(60:\hex@r@r)+(0:\hex@r@t/2)$)}
\fi
\xdef\hex@r@p{\hex@r@p --(120:\hex@r@r)}
\hex@dbg{4}{Ridge along north edge: '\hex@r@p'}
\fi
\fi
\ifhex@r@nw
\ifhex@r@n\else
\xdef\hex@r@p{\hex@r@p ($(120:\hex@r@r)+(60:\hex@t/2)$)}
\fi
\xdef\hex@r@p{\hex@r@p --(180:\hex@r@r)}
\hex@dbg{4}{Ridge along north west: '\hex@r@p'}
\fi
\fi
\ifhex@r@se
\ifhex@r@ne\else
\xdef\hex@r@p{\hex@r@p ($(60:\hex@r@r)+(0:\hex@r@t/2)$)}
\fi
\xdef\hex@r@p{\hex@r@p --(120:\hex@r@r)}
\hex@dbg{4}{Ridge along north east edge: '\hex@r@p'}
\fi
\fi
\ifhex@r@sw
\ifhex@r@n\else
\xdef\hex@r@p{\hex@r@p ($(120:\hex@r@r)+(60:\hex@t/2)$)}
\fi
\xdef\hex@r@p{\hex@r@p --(180:\hex@r@r)}
\hex@dbg{4}{Ridge along north west: '\hex@r@p'}
\fi
\fi
\end{verbatim}

296
5.4.7 Towns

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

\tikzset{
  hex/town/.style={
    scale line widths, solid, thin, fill=pgfstrokecolor, color=pgfstrokecolor},
  hex/town name/.style={
    transform shape,}
}
\begin{tikzpicture}
\tikzset{
  /hex/town/.search also={/tikz},%
  /hex/town/.cd,
  pic/.store in=\hex@c@pic,
  type/.store in=\hex@c@pic,
  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}
}
\end{tikzpicture}

Next is the namespace for dealing with towns.

\begin{tikzpicture}
\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);}
}
\end{tikzpicture}

And some pictures for making the towns.

The macro to make the towns. This uses same tricks as above.

\begin{tikzpicture}
\tikzset{
  hex/town/.search also={/tikz},%
  hex/town/.cd,
  /tikz/hex/town,%
  \hex@do@town
}
\end{tikzpicture}
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.

\newcounter{hex@l@c}
\newcounter{hex@l@r}

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

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

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.

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

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

\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%
  }
}
\def\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=,%
\def\hex@l@text{%
\ifnum0>\hex@col\AlphAlph{-\hex@col}\else\AlphAlph{\hex@col}\fi%
\hex@row},
\auto/alpha 2 column/.code={%
\hex@l@abs%
\advance\c@hex@l@c27\relax%
\def\hex@l@text{%
\AlphAlph{\value{hex@l@c}}%
\hex@l@n@pad{\hex@row}}},
\auto/inv y x plus 1/.code=,%
\hex@dbg{3}{Inverse row, add one to column with arg ‘#1’}
\let\hex@l@text\@empty%
\ifnum\hex@max@row>0%
\pgfmathtruncatemacro{\hex@l@row}{\hex@max@row-\hex@row}
\pgfmathtruncatemacro{\hex@l@col}{1+\hex@col}
\def\hex@l@text{%
\hex@l@n@pad{\hex@l@col}%
\hex@l@n@pad{\hex@l@row}}
\else\message{Max row number not set}\fi},
\auto/x and y plus 1/.code=,%
\hex@dbg{3}{Inverse row, add one to column with arg ‘#1’}
\pgfmathtruncatemacro{\hex@l@row}{1+\hex@row}
\pgfmathtruncatemacro{\hex@l@col}{1+\hex@col}
\def\hex@l@text{%
\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={\edef\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}
This will pad a number with a 0 if the number is smaller than 10.

```latex
\def\hex@l@n@pad#1{\ifnum#1<10 0\fi#1}
```

\texttt{\hex@do@label}

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 \texttt{\hex@l@text} to be non-empty, then we set the label via a TikZ node.

```latex
\def\hex@do@label{% 
\hex@dbg{1}{Hex label: ‘\meaning\hex@label’}% 
\edef\hex@l@tmp{[/hex/label/.cd,} 
\hex@l@tmp\rotate=0,\place=\location,\every hex label/.try,\hex@l@text\]}% 
\expandafter\scope\hex@l@tmp% 
\hex@dbg{1}{Label: Text: ‘\meaning\hex@l@text’ Location: ‘\meaning\hex@l@location’ Rotation: ‘\meaning\hex@l@rotation’}% 
\@ifundefined{\hex@l@text}{}{% 
\if\hex@l@text\else\node[\rotation] at \location \text{\texttt{\hex@l@text}};\fi}% 
\endscope% 
}
```

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.

\texttt{\hex/fortress}

Draw a fortress. An example of a extra graphics entity.

```latex
\tikzset{% 
\hex/fortress/.pic={ 
\path\draw,\solid,\pic actions\} 
\node[\rotate] at \location \text{\texttt{\hex@l@text}};\fi}% 

301
```
\begin{tikzpicture}[hex/fortress 2/.pic={
\foreach \a in {15,45,...,345}{
--(\a:0.64)
--(\a:0.80)
--(\a+15:0.80)
--(\a+15:0.64)
},
},
\end{tikzpicture}

5.4.10 Some macros

\begin{verbatim}
\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\seahex[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);}}
\end{verbatim}

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

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

\begin{verbatim}
\pgfdeclaredecoration{outline}{init}{
\state{init}[next state=tick,width=0pt]{
\xdef\outlinerev{}},
\state{tick}[width=+\pgfdecorationsegmentlength]{
\xdef\outlinerev{0}\pgfpathmoveto{\pgfpoint\outlinerev cm}}
\end{verbatim}
A decoration to make a fortification line

\pgfdeclaredecoration{fortification}{initial}
{\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}}
  \pgfpathlineto{\pgfpointdecoratedpathlast}
\state{final}
{\pgfpathlineto{\pgfpointorigin}}

Roads, railroads, rivers, borders.

\tikzset{
  hex/road/.style={
    rounded corners=3\pgflinewidth, \ .25cm, 
    color=black, 
    transform shape, 
}
scale line widths, thick, every hex road/.try,

hex/railroad/.style={
%scale line widths, rounded corners=.25cm, color=gray!50!black, transform shape, every hex railroad/.try, postaction={draw,decorate}, decoration={ticks, segment length=9\pgflinewidth, amplitude=3\pgflinewidth,.1cm}
}

hex/river/.style={
color=blue, scale line widths, scale rounded corners, line width=3pt, transform shape, every hex river/.try, decorate, decoration={random steps, segment length=3\pgflinewidth, amplitude=1.5\pgflinewidth, pre=lineto, post=lineto, pre length=.5\pgflinewidth, post length=.5\pgflinewidth}, rounded corners=.75\pgflinewidth},

hex/border/.style={
color=gray, dashed, transform shape, scale line widths, very thick, rounded corners=3\pgflinewidth, every hex border/.try
}

% Fortification line

% hex/fortified line/.style={
draw=brown!50!black, scale line widths, line width=2pt, every hex fortification line/.try, decoration={fortification,raise=-2\pgflinewidth}, decorate},

% every river/.style={},
% every road/.style={}

304
\road
\railroad
\river
\border

\def\road{\% 
% \hex@dbg{3}{Road} 
\@ifnextchar[{oad@}{\road@[]} 
%
}
\def\road@[#1]{\draw[hex/road,every hex road/.try,#1]}
\def\railroad{\%
% \hex@dbg{3}{Rail road}
\@ifnextchar[{ailroad@}{\railroad@[]} 
%
}
\def\railroad@[#1]{\draw[hex/railroad,every hex railroad/.try,#1]}
\def\river{\%
% \hex@dbg{3}{River}
\@ifnextchar[{iver@}{\river@[]} 
%
}
\def\river@[#1]{\draw[hex/river,#1]}
\def\border{\%
% \hex@dbg{3}{Border}
\@ifnextchar[{order@}{\border@[]} 
%
}
\def\border@[#1]{\draw[hex/border,every hex border/.try,#1]}
\def\fortifiedline{\%
% \hex@dbg{3}{Fortified line}
\@ifnextchar[{ortifiedline@}{\fortifiedline@[]} 
%
}
\def\fortifiedline@[#1]{\draw[hex/fortified line,every hex fortified line/.try,#1]}

5.4.12 Other paths

\shiftScalePath
Shifts and scales a path and defines a macro to contain the path

\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.

\def\shiftScalePath#1#2{\%
5.4.13 Move, attacks, retreats from hex to hex

\hex@getscale
Get current scaling factor.

\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}}}%

Key to get the scale

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

Style for moves. Use like

\path[move] (coordinates);
A short move style

\path[short move] (coordinates); 

\path[long move] (coordinates);
\path [move with start] (coordinates); 
\path [move] ...(coordinate) node [hex/move cost] ...;
\path [short line] (start)--(end);
An attack indication style

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

% Argument is fill color
\t\hex/attack/.style={
\t\t\hex/get scale,
\t\t\inherit options/.code={\csname tikz@options\endcsname},
\t\t\inherit options,
\t\t\decorate,
\t\t\decoration={
\t\t\markings,
\t\t\mark=between positions \hex\scale\hex\dy
\t\t\and 1 step 2*\hex\scale\hex\dy with {
\t\t\node [\single arrow,\draw=black,\fill=#1,
\t\t\single arrow head extend=\hex\scale\three pt,
\t\t\inner sep=1mm,
\t\t\minimum width=0.75*\hex\scale\hex\dy,
\t\t\minimum height=\hex\scale\hex\dy,
\t\t\transform shape]{};
\t\t}
\t\t},
\t\t},

Short hands

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

% Default colour is red for attachs
\t\hex/attack/.default=red!70!black,
\t\%\%
\t\hex/retreat/.style={\hex/short line=#1},
\t\%\%
\t\hex/retreat/.default=white,
\t\%
\t\hex/advance/.style={\hex/short line=#1},
\t\%
\t\hex/advance/.default={green!70!black},
\t\%
\t}
5.4.14 Board clipping and frame

\boardframe

Define the bounding box around the board

\boardframe[(\texttt{margin})]\{(\texttt{lower=left})\}[(\texttt{upper-right})]\{\texttt{(margin)}\}

where (\emph{lower-left}) and (\emph{upper-right}) specifies the lower left and upper right hexes (inclusive) of the board.

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.

\def\bo@rdfr@me{
\@ifnextchar[\bo@rdfr@me@}{\bo@rdfr@me@[0]}%
}
\def\bo@rdfr@me@\{\#1\}\#2\#3\#4\#5{
\hex@coords@conv\{\#1\}
% \hex@dbg{0}{\#1 -> 'll='\hex@x', 'll='\hex@y'}
\pgfmathparse{min(\#2,\hex@x)}\xdef\#2{\pgfmathresult}
\pgfmathparse{min(\#3,\hex@y)}\xdef\#3{\pgfmathresult}
\pgfmathparse{max(\#4,\hex@x)}\xdef\#4{\pgfmathresult}
\pgfmathparse{max(\#5,\hex@y)}\xdef\#5{\pgfmathresult}
\hex@dbg{2}{\#1 -> ll='\#2', 'll='\#3', ur='\#4', 'ur='\#5'}
}
\def\bo@rdfr@me@[\#1]\{\#2\#3\#4\#5{
% Define rtmp and a ctmp to by directions
\pgfmathparse{int(\hex@coords@row@fac)}\edef\rtmp{\pgfmathresult}
\pgfmathparse{int(\hex@coords@col@fac)}\edef\ctmp{\pgfmathresult}
% Define vertices for path
\def\ctfv{SW}
\def\ctsv{SE}
\def\cbfv{NE}
\def\cbsv{NW}
\def\rrfv{E}
\def\rrsv{NE}
\def\rlfv{W}
\def\rlsv{NW}
% Swap around some definitions based on the row direction
\ifnum\rtmp<0
\let\max@short\hex@bot@short@col
\let\min@short\hex@top@short@col
\let\swp\ctfv\let\ctfv\cbsv\let\cbsv\swp
\let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp
\def\rrsv{SE}
\def\rlsv{NW}
\fi
% Swap around some definitions based on the row direction
\ifnum\rtmp<0
\let\max@short\hex@bot@short@col
\let\min@short\hex@top@short@col
\let\swp\ctfv\let\ctfv\cbsv\let\cbsv\swp
\let\swp\ctsv\let\ctsv\cbfv\let\cbfv\swp
\def\rrsv{SE}
\def\rlsv{NW}
\fi

310
\texttt{13704 \else}
\texttt{13705 \let\max@short\hex@top@short@col}
\texttt{13706 \let\min@short\hex@bot@short@col}
\texttt{13707 \fi}
\texttt{13708 \% Swap around some definitions based on the column direction}
\texttt{13709 \ifnum\ctmp<0}
\texttt{13710 \let\swp\ctfv\let\ctfv\ctsv\let\ctsv\swp}
\texttt{13711 \let\swp\cbfv\let\cbfv\cbsv\let\cbsv\swp}
\texttt{13712 \let\swp\rrfv\let\rrfv\rlsv\let\rlsv\swp}
\texttt{13713 \let\swp\rrsv\let\rrsv\rlfv\let\rlfv\swp}
\texttt{13714 \fi}
\texttt{13715 \% Define tmp = 0 if no shorts, 1 if top short, 2 if both}
\texttt{13716 \pgfmathparse{ifthenelse(\hex@got@top@short,}
\texttt{13717 \ ifthenelse(\hex@got@bot@short,2,1),0)}\edef\tmp{\pgfmathresult}
\texttt{13718 \% If top-short, set factors}
\texttt{13719 \ifnum\tmp=1}
\texttt{13720 \def\mnf{-1}}
\texttt{13721 \def\mxf{-1}}
\texttt{13722 \def\mnn{}}
\texttt{13723 \def\mxn{}}
\texttt{13724 \% If both short, set factors}
\texttt{13725 \else\ifnum\tmp=2}
\texttt{13726 \def\mnf{\rtmp}}
\texttt{13727 \def\mxf{(-\rtmp)}}
\texttt{13728 \% If inverse rows, set factors}
\texttt{13729 \ifnum\rtmp<0}
\texttt{13730 \def\mnn{}}
\texttt{13731 \def\mxn{not}}
\texttt{13732 \else}
\texttt{13733 \def\mnn{not}}
\texttt{13734 \def\mxn{}}
\texttt{13735 \fi}
\texttt{13736 \% If none is short}
\texttt{13737 \else}
\texttt{13738 \def\mnf{1}}
\texttt{13739 \def\mxf{1}}
\texttt{13740 \def\mnn{not}}
\texttt{13741 \def\mxn{not}}
\texttt{13742 \fi}
\texttt{13743 \% Define row@mn to give least row of column}
\texttt{13744 \def\row@mn##1{}}
\texttt{13745 \pgfmathparse{int(#3+\mnf*}
\texttt{13746 \ hex@coords@row@fac\min@short(#1)*)}
\texttt{13747 \ mnn\min@short(\hex@coords@col@off))}}\edef\lr{\pgfmathresult}
\texttt{13748 \def\ur{\pgfmathresult}}
\texttt{13749 \% Define row@mx to give largest row of column}
\texttt{13750 \def\row@mx##1{}}
\texttt{13751 \pgfmathparse{int(#5+\mxf*}
\texttt{13752 \ hex@coords@row@fac\max@short(#1)*)}
\texttt{13753 \ mxn\max@short(\hex@coords@col@off))}}\edef\ur{\pgfmathresult}
\texttt{13754 \%}
\texttt{13755 \%}
% 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{}
% Loop across least row (can be top if \rtmp<0)
\foreach \c in \{#2,...,#4\}{{
    \row@mn\c
    \row@mx\c
    % \message{\^{}JColumn: \c \rightarrow \lr,\ur (#3,#5)}
}
% Go up (down if \rtmp<0) right side
\row@mn\ur
\row@mx\ur
\foreach \r in \{\lr,...,\ur\}{{
    \xdef\p{\p
        (hex cs:c=#4,r=\r,v=\rrfv)\--
        (hex cs:c=#4,r=\r,v=\rrsv)\--}
        \bo@rdfr@me@u(c=#4,r=\r,v=\rrfv)\@llx\@lly\@urx\@ury
        \bo@rdfr@me@u(c=#4,r=\r,v=\rrsv)\@llx\@lly\@urx\@ury
}
% Go across largest row (can be bottom if \rtmp<0)
\row@mn\#4
\row@mx\#4
\foreach \c in \{#4,...,#2\}{{
    \row@mx\c
    % \message{\^{}JColumn: \c , max:'\ur'}
    \xdef\p{\p
        (hex cs:c=\#4,r=\c,v=\cbfv)\--
        (hex cs:c=\#4,r=\c,v=\cbsv)\--}
        \bo@rdfr@me@u(c=\#4,c=\c,r=\ur,v=\cbfv)\@llx\@lly\@urx\@ury
        \bo@rdfr@me@u(c=\#4,c=\c,r=\ur,v=\cbsv)\@llx\@lly\@urx\@ury
}
% Go up (down if \rtmp<0) left side.
\row@mn\#2
\row@mx\#2
\foreach \c in \{\ur,...,\lr\}{{
    \xdef\p{\p
        (hex cs:c=\#2,r=\c,v=\rlfv)\--
        (hex cs:c=\#2,r=\c,v=\rlsv)\--}
        \bo@rdfr@me@u(c=\#2,c=\c,r=\lr,v=\rlfv)\@llx\@lly\@urx\@ury
        \bo@rdfr@me@u(c=\#2,c=\c,r=\lr,v=\rlsv)\@llx\@lly\@urx\@ury
}
% End path with cycle
\edef\p{\p cycle}
\global\let\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.
\ifx|#1|\else
\foreach[\count\nc in {#2,...,#4}{%}
\row@mn{\c}
\row@mx{\c}
\foreach \r in {\lr,...,\ur}{%}
\hex[#1={\c,\r}](c=\c,r=\r)
}\i
}
\tikzset{%/hex/board/no op/.style args={#1,#2}{}}
% This macro will make the actual hexes using the specified, optional, style. It builds on \bo@rdfr@me above.
\def\boardhexes{%
@ifnextchar[{o@rdhexes}{o@rdhexes[board/no op]}%
}
\def\bo@rdhexes[#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}
\bo@rdfr@me[#1]{\llc}{\llr}{\urc}{\urr}
}

Creates a board frame using \bo@rdfr@me.
\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
\global\let\lly\@lly
\global\let\urx\@urx
\global\let\ury\@ury
% \hex@dbg{0}{Board bounding box ($\llx,\lly$)\times($\urx,\ury$)}%
}}}
\def\bo@rdfr@me[#1](#2)(#3){%
\hex@coords@conv{#1}
}
\def\llc{\hex@col}
\def\llr{\hex@row}
\edef\llc{\hex@col}
\edef\llr{\hex@row}
\def\margin{#1}
\edef\urc{\hex@col}
\edef\urr{\hex@row}
\def\margin{#1}
% This will store the bounding box in tmp node ‘board frame’
\bo@rdfr@me{\llc}{\llr}{\urc}{\urr}
\begin{scope}[board frame bb]
\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}
% This will store the bounding box in tmp node ‘board frame’
\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}
\pgfmathparse{\urx-\llx}\edef\w{\pgfmathresult}
\pgfmathparse{\ury-\lly}\edef\h{\pgfmathresult}
\pgfmathparse{\urx-\llx}\edef\w{\pgfmathresult}
\pgfmathparse{\ury-\lly}\edef\h{\pgfmathresult}
\def\boardpath(#1)(#2){
\hex@coords@reset%
\tikzset{/hex/coords/.cd, #1}
\edef\llc{\hex@col}
\edef\llr{\hex@row}
\edef\urc{\hex@col}
\edef\urr{\hex@row}
% This will store the bounding box in tmp node ‘board frame’
\pgfmathparse{\urx-\llx}\edef\w{\pgfmathresult}
\pgfmathparse{\ury-\lly}\edef\h{\pgfmathresult}
\def\boardpath(#1)(#2){
\hex@coords@reset%
\tikzset{/hex/coords/.cd, #1}
\edef\llc{\hex@col}
\edef\llr{\hex@row}
\edef\urc{\hex@col}
\edef\urr{\hex@row}
% This will store the bounding box in tmp node ‘board frame’
\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{% 
\boardpath(#1)(#2)
\draw \ifx|#3|\else[preaction={#3}]\fi% 
[clip] \hexboardpath;
}
\debuggrid

Show a debug grid. This requires \boardframe.
\def\debuggrid{% 
\foreach \i in {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1} {% 
\pgfmathparse{\i*\boardW+\boardXmin}\% 
\edef\debugx{\pgfmathresult}\% 
\draw [very thin,gray](\debugx,\boardYmin) -- 
(\debugx,\boardYmax) node [below,rotate=90] at 
(\debugx,\boardYmin) {$\i$}; }% 
\foreach \i in {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1} {% 
\pgfmathparse{\i*\boardH+\boardYmin}\% 
\edef\debugx{\pgfmathresult}\% 
\draw [very thin,gray] (\boardXmin,\debugx) -- 
(\boardXmax,\debugx) node [left,rotate=90] at 
(\boardXmin,\debugx) {$\i$}; } }

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

- \texttt{paper=(format)}: Specifies the paper format. One of \texttt{a4}, \texttt{a3}, \texttt{letter}, \texttt{tabloid}. Default is \texttt{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**=(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

\begin{verbatim}
\documentclass[11pt]{standalone}
\usepackage{wargame}
\usepackage{mystyle}
\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 × 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

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

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

\begin{verbatim}
\tikzset{board scale/.style={scale=.9}}
\end{verbatim}

Styles used for drawing things.

\begin{verbatim}
\tikzset% 13933
\end{verbatim}
split/paper outline/.style={
    shape=rectangle,
    draw=red!50!black,
    line width=.5mm},

split/effective outline/.style={
    shape=rectangle,
    draw=green!50!black,
    dashed,
    line width=.5mm},

split/board outline/.style={%
    draw=magenta,
    line width=.5mm,
    dotted},
%
}

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@lry{\split@lry}%
}

Get board dimensions. Argument is node name.

\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:
        \@percentchar\space\@percentchar(\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%
    \advance\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.

\texttt{\def\split@get@init#1#2#3#4{\pgfmathparse{((#1 * #3 - (#1 - 1) * #2) - #4)/2}\xdef\split@off{\pgfmathresult}}}

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

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

Get coordinates of a segment

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

\texttt{\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}}}

318
Write final stuff and close stream

\def\split@footer{%}
\ifsplit@standalone
\split@w{``\string\end{document}}
\fi
\split@w{``@percentchar@percentchar End of `jobname'``J}
\immediate\closeout\split@calcout
}

Initial calculations. This draws the board and then extracts the dimensions of the board. It also defines some styles for drawing the board segments.

\def\split@init#1{%
\node[scale=\split@scale, inner sep=0pt, outer sep=0pt, anchor=north west, transform shape](b){\includegraphics{#1}};
\split@getboard{b}
%}
\split@tmp=\split@blrx cm\advance\split@tmp by -\split@bulx%
\pgfmathparse{abs(\split@tmp)}\edef\split@bw{\pgfmathresult}%
%\split@tmp=\split@buly cm\advance\split@tmp by -\split@blry%
\pgfmathparse{abs(\split@tmp)}\edef\split@bh{\pgfmathresult}%
\pgfmathparse{\paperwidth}\edef\split@pw{\pgfmathresult}%
\pgfmathparse{\paperheight}\edef\split@ph{\pgfmathresult}%
\pgfmathparse{\textwidth}\edef\split@ew{\pgfmathresult}%
\pgfmathparse{\textheight}\edef\split@eh{\pgfmathresult}%
%\hex@dbg{1}{Board:
\split@bulx,\split@blry} \split@bw x\split@bh
\split@blrx,\split@buly} /\split@bw x\split@bh
% JPaper: \split@pw x\split@ph
% JEffective: \split@ew x\split@eh
}
\tikzset{ split/paper size/.style={ shape=rectangle, minimum width=\split@pw, minimum height=\split@ph, split/paper outline, }, split/effective size/.style={ shape=rectangle, minimum width=\split@ew, minimum height=\split@eh, split/effective outline}, split/board size/.style={ shape=rectangle, }}
minimum width=\split\@bw cm,
minimum height=\split\@bh cm,
split/board outline}}
\node[board/.try,split/board size,anchor=north west] {};
}

Calculate effective sheet sizes from sheet dimensions and the defined margin.

\def\split@text@dim#1{%
  \textwidth=\paperwidth%
  \textheight=\paperheight%
  \advance\textwidth by -#1cm%
  \advance\textwidth by -#1cm%
  \advance\textheight by -#1cm%
  \advance\textheight by -#1cm%
  \global\textwidth=\textwidth%
  \global\textheight=\textheight%
}

Options for the \splitboard macro.

\newif\ifsplit@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
    \split@tmp=\textheight
    \global\textheight=\split@tmp
}
The actual macro. The argument is key-value pairs of options.

```latex
\def\splitboard#1{\pgfkeys{/tikz/split/.cd,}
  standalone,\%
  output,\%
  margin,\%
  paper,\%
  image,\%
  overlap,\%
  scale,\%
  ncol,\%
  nrow,\%
  \#1}
\hex@dbg{1}{%
  \Paper: \the\paperwidth'x\the\paperheight'
  \~Effective: \the\textwidth'x\the\textheight'
  \~NCols: \split@ncol'
  \~NRows: \split@nrow'
  \~Overlap: \split@ov cm}
\split@header{\split@out}
\begin{tikzpicture}
\split@init{\split@img}
\split@getinit{\split@nrow}{\split@ncol}{\split@ov}{\split@eh}{\split@ew}{\split@bh}{\split@bw}
\node[split/effective size, above left=\dy and \dx of b.north west, anchor=north west] (c1r1) {};
\node[split/paper size] at (c1r1) {};
\foreach \r [remember=\r as \pr (initially 0)] in {1,...,\split@nrow}{% 
  \ifnum\r>1
    \hex@dbg{3}{Placing first column of row \r}
  \fi
```

322
Macro used by the written file.

1. first coordinate (e.g., \texttt{(hex ak:c=C,r=17)})
2. second coordinate (e.g., \texttt{(hex ak:c=M,r=33)})
3. Crop mark left
4. Crop mark right
5. Crop mark bottom
6. Crop mark top

\begin{tikzpicture}
\begin{scope}
\clip (#1) rectangle (#2);
\node[scale=boardscale, inner sep=0pt, outer sep=0pt, anchor=north west, transform shape]{\includegraphics{boardfile}};
\end{scope}
\end{tikzpicture}
5.5 The \texttt{wargame.chit} Ti\texttt{k}Z library

We define the library for making chits. We load the hex Ti\texttt{k}Z \texttt{wargame.natoapp6c} library and the \texttt{amsmath} and \texttt{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 \texttt{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 \texttt{chit} key namespace

Some stuff to consider wrt. line widths. Setting the line width in the \texttt{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.
The parts of a chit

\begin{verbatim}
14234 \newif\ifchit@clip\chit@cliptrue
14235 \tikzset{
14236 /chit/.search also={/tikz},
14237 /chit/.cd,
14238 full/.store in=\chit@full, full/.initial=,%
14239 symbol/.store in=\chit@symbol, symbol/.initial=,%
14240 left/.store in=\chit@left, left/.initial=,%
14241 unique/.style=\chit/left=#1},% 14242 right/.store in=\chit@right, right/.initial=,%
14243 parent/.style=\chit/right=#1},% 14244 upper left/.store in=\chit@upperleft, upper left/.initial=,%
14245 upper right/.store in=\chit@upperright, upper right/.initial=,%
14246 lower left/.store in=\chit@lowerleft, lower left/.initial=,%
14247 lower right/.store in=\chit@lowerright, lower right/.initial=,%
14248 factors/.store in=\chit@factors, factors/.initial=,%
14249 setup/.store in=\chit@setup, setup/.initial=,%
14250 id/.store in=\chit@id, id/.initial=,%
14251 frame/.store in=\chit@frame, frame/.initial=,%
14252 extra/.store in=\chit@extra, extra/.initial=,%
14253 bev/.store in=\chit@bevel, bev/.initial=,
14254 bevel fraction/.store in=\chit@bevelfrac, bevel fraction/.initial=10,
14255 bev/.is choice,
14256 bevel/none/.style = {/chit/bev=},
14257 bevel/north west/.style = {/chit/bev=1},
14258 bevel/north east/.style = {/chit/bev=2},
14259 bevel/south west/.style = {/chit/bev=3},
14260 bevel/south east/.style = {/chit/bev=4},
14261 bevel/NW/.style = {/chit/bev=1},
14262 bevel/NE/.style = {/chit/bev=2},
14263 bevel/SW/.style = {/chit/bev=3},
14264 bevel/SE/.style = {/chit/bev=4},
14265 bevel/.default = north west,
14266 clip/.is if=\chit@clip%
14267 }
\end{verbatim}

325
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},
    chit/parts/.style={shape=rectangle,transform shape},
    chit/factors/.style={chit/parts,anchor=south},
    chit/left/.style={chit/parts,anchor=south,rotate=90},
    chit/right/.style={chit/parts,anchor=north,rotate=90},
    chit/upper left/.style={chit/parts,anchor=north west},
    chit/upper right/.style={chit/parts,anchor=north east},
    chit/lower left/.style={chit/parts,anchor=south west},
    chit/lower right/.style={chit/parts,anchor=south east},
    chit/setup/.style={chit/parts},
    chit/full/.style={chit/parts},
    chit/frame/.try={draw=pgfstrokecolor},
    chit/bevel highlight/.style={fill=white,opacity=.25},
    chit/bevel shadow/.style={fill=black,opacity=.25},
}
\def\chit@bevel@frac{10}
\newif\ifchit@draw@frame\chit@draw@frametrue
\tikzset{
    chit/frame style/.search also={/tikz},
    chit/frame style/.cd,
    none/.code={\chit@draw@framefalse},
    draw/.code={%
        \chit@dbg{2}{Frame draw option '#1'}
        \edef\tikz@temp{#1}%
        \ifx\tikz@temp\tikz@nonetext%
        \chit@draw@framefalse%
        \else%
        \chit@draw@frametrue%
        \fi
    }
}
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.

\tikzset{"
  chit/.code={%
    \pgfkeys{/tikz/transform shape,/tikz/shape=chit}
    \pgfkeys{/chit/.cd,#1}}%
}\n
We define a counter to set-up unique names for chit nodes.
\newcounter{chit@id}\setcounter{chit@id}{0}

5.5.4 The \texttt{\textbackslash chit} shape

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 (\texttt{\textbackslash @chit@n@to@}). Otherwise, we may also need to scan for an offset given as \((\text{delta-x, delta-y})\).

\def\chit@n@to#1#2{%\n  \chit@dbg{1}{Chit NATO App6(c) first step '#1' '#2'}\n  \@ifnextchar[{%\n    %\message{^^JStart square bracket}%\n    \@chit@n@to{#1}{#2}}{%\n    %\message{^^JNo start square bracket}%\n    \@chit@n@to@{#1}{#2}}%\n}\n
The following macro is called if we had no options.
\def\@chit@n@to@#1#2#3\@end@chit@n@to{%\n  \chit@dbg{4}{Chit NATO App6(c) ended}\n}\n
This is called if we had an option-like argument. Check if we have an offset
\def\chit@n@to@#1#2[#3]{%\n  \chit@dbg{1}{Chit NATO App6(c) second step '#1' '#2' '#3'}\n}
This called if we had option-like argument.

```
\def\@@chit@n@to#1#2#3(#4)\@end@chit@n@to{%
  \chit@dbg{1}{Chit NATO App6(c) w/offset:
    ^^J Options: #3
    ^^J ID: #1
    ^^J Position: #2
    ^^J Offset: #4}
  \node[chit/symbol,natoapp6c={#3,id=#1}] (#1) at ($(#2)+(#4)$) {};
}
```

Get anchor of sub-symbol element in chit. We need to do this, because the symbol is translated and scaled.

```
\def\chit@tr@ns@nchor#1{% 
  \pgf@x=0.4\pgf@x 
  \pgf@y=0.4\pgf@y\advance\pgf@y#1 
}\def\chit@nchor#1#2#3{% 
  \wg@sub@nchor{#1}{#2} 
  \chit@tr@ns@nchor{#3} 
}\def\chit@report{} \tikzset{
  zone turn/.style={},
  zone mult/.style={}
}
```

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

```
\def\chit@bevel@path#1{ 
  \scope[#1] 
  \wg@tmpc=\wg@tmpa\multiply\wg@tmpc by \chit@bevel@frac 
  \wg@tmpd=\wg@tmpb\multiply\wg@tmpd by \chit@bevel@frac 
  \divide\wg@tmpc100 
  \divide\wg@tmpd100 
  \pgfpathmoveto(\pgfqpoint{\wg@tmpa}{\wg@tmpb})% 
  % Move down along edge 
  \wg@tmpb=-\wg@tmpb 
  \pgfpathlineto(\pgfqpoint{\wg@tmpa}{\wg@tmpb})% 
  % Move left along edge 
  \wg@tmpa=-\wg@tmpa 
  \pgfpathlineto(\pgfqpoint{\wg@tmpa}{\wg@tmpb})% 
  % Move in and up 
  \advance\wg@tmpa\wg@tmpc 
  \advance\wg@tmpb\wg@tmpd 
  \pgfpathlineto(\pgfqpoint{\wg@tmpa}{\wg@tmpb})% 
  % Move right, but in 
  \advance\wg@tmpa-\wg@tmpc 
  \advance\wg@tmpb-\wg@tmpd 
  \pgfpathlineto(\pgfqpoint{\wg@tmpa}{\wg@tmpb})% 
\endscope} 
```
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).

Next, we define some saved macros. These are called (and declares internal macros) when the shape is instantised. We define macros for the identifier,

We define the regular anchors of the shape. That is, the centre, corners, and edges.
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 $\texttt{full=⟨args⟩}$.

\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 echelon}{\chit@nchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol below}{\chit@nchor{M\id symbol}{south}{0.1cm}}

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@nchor{M\id symbol}{west}{.2cm}\advance\pgf@x-\margin}
\anchor{right}{\chit@nchor{M\id symbol}{east}{.2cm}\advance\pgf@x+\margin}
\anchor{upper right}{\northeast \advance\pgf@x-\margin \pgf@y-\margin}
\anchor{upper left}{\northeast \advance\pgf@x-\margin \pgf@y+\margin \pgf@x=-\pgf@x \pgf@y=-\pgf@y}
\anchor{lower right}{\northeast \advance\pgf@x-\margin \pgf@y-\margin \pgf@y=-\pgf@y}
\anchor{lower left}{\northeast \advance\pgf@x-\margin \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\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\wg@tmpa=-\wg@tmpa \pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\wg@tmpb=-\wg@tmpb \pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\wg@tmpa=-\wg@tmpa \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 Upper right: \meaning{chit@upper@right}
% ^^J Lower right: \meaning{chit@lower@right}
% ^^J Extra: \meaning{chit@extra}
% ^^J Bevel: \meaning{chit@bevel}
% ^^J Frame: \meaning{chit@frame}
\chit@dbg{1}{Chit report}
\chit@report{}
\chit@dbg{1}{Chit start scope}
\pgfscope
% \ifchit@clip%
\chit@dbg{1}{Chit clip path}
\northeast%
\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\wg@tmpa=-\wg@tmpa \pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\wg@tmpb=-\wg@tmpb \pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\wg@tmpa=-\wg@tmpa \pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfclosepath%
\pgfusepath{clip}%
% \fi%
```

If we do not have the *symbol* key set, then we set the *full* key as a picture.

```
@ifundefined{chit@symbol}{%
% Draw full stuff%
@ifundefined{chit@full}{}{%
\chit@dbg{1}{Chit draw full image: \meaning{chit@full}'}
\center\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfusepath{\chit@full}{}
% With NATO symbol
}%
\edef\symid{id symbol}%
\edef\args{\symid{\the\pgf@x,\the\pgf@y}chit@symbol}%
```

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

```
\chit@dbg{1}{Chit draw symbol image}
\edef\symid{id symbol}%
\symbol%
\edef\args{\symid{\the\pgf@x,\the\pgf@y}chit@symbol}%
```
Having made the NATO App6(c) symbol, which we gave the node name \( \langle \text{id} \rangle \text{symbol} \) where \( \langle \text{id} \rangle \) 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 \texttt{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.
5.5.5 The \chit wrapper macro

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.

\[\def\chit{\chit@dbg{5}{Chit}}\]

Parse for coordinates.

\[\def\chit@[#1]{\chit@dbg{5}{Chit second: '#1'}}\]
Parse for name.

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\LaTeX\ macros.

5.5.6 Predefined chit element pictures

These pictures can be used as the value of chit keys.
\begin{tikzpicture}

% 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,
  },
  chit/1 factor/.style={
  },
  chit/2 factors/.style={
  },
  chit/3 factors/.style={
  },
  chit/4 factors/.style={
  },
  chit/identifier/.pic={
    \chit@dbg{4}{ Chit identifier: \meaning#1}
    \edef\chit@i@tmp{#1}
    \node[chit/identifier,pic actions]{\chit@i@tmp};
  },
  chit/small identifier/.pic={
    \chit@dbg{4}{ Chit small identifier: #1}
    \node[chit/small identifier,pic actions]{#1};
  },
  chit/identifiers/.pic={
    \chit@dbg{4}{ Chit identifiers: #1}
    \node[chit/identifier,pic actions]{\chit@sep{#1}};
  },
  chit/small identifiers/.pic={
    \chit@dbg{4}{ Chit small identifiers: #1}
    \node[chit/small identifier,pic actions]{\chit@sep{#1}};
  },
  chit/identifier macro/.pic={
    \chit@dbg{4}{ Chit identifier macro: \meaning#1}
    \edef\chit@i@tmp{#1}
    \node[chit/identifier,pic actions]{\chit@i@tmp};
  };
}

% 2 factors artillery
\tikzset{
  pics/chit/2 factors artillery/.style args={#1,#2,#3}{
    \chit@dbg{4}{ Chit 2 factors w/artillery: '#1' '#2' '#3'}%
    \node[chit/factor,chit/2 factors]{
      (#1)\overset{\text{\scriptsize #3}}{\text{--}} (#2)};
  },
}

% 3 factors
\tikzset{
  pics/chit/3 factors/.style args={#1,#2,#3}{
    \chit@dbg{4}{ Chit 3 factors: '#1' '#2' '#3'}%
    \node[chit/factor,chit/3 factors]{#1-#2-#3};
  },
}

% 4 factors
\tikzset{
  pics/chit/4 factors/.style args={#1,#2,#3,#4}{
    \chit@dbg{4}{ Chit 4 factors: '#1' '#2' '#3' '#4'}%
    \node[chit/factor,chit/4 factors]{#1-#2-#3-#4};
  },
}
\end{tikzpicture}
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.

\begin{verbatim}
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}}}{\@sh@dechit[.5]}
\def\eliminatechit{\@ifnextchar[{{\@elimin@techit}}}{\@elimin@techit[.25]}
\def\@sh@dechit[#1](#2){\message{Shading chit with opacity '#1'}\pic[transform shape] at (#2) {chit/shade=#1};}
\def\@elimin@techit[#1](#2){\message{Eliminating chit with opacity '#1'}\pic[transform shape] at (#2) {chit/eliminate=#1};}
\end{verbatim}
5.5.8 Stacking of chits

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

Now the code

\def\chit@stack@dir{(.3,.3)}
\def\stackchits(#1){\@ifnextchar({\st@ckchits{#1}}{\st@ckchits{#1}(.3,.3)}%)
\def\st@ckchits#1(#2)#3{\chit@dbg{2}{Stacking chits '#1', '#2', '#3'}\edef\xy{#1}\chit@dbg{4}{Stack start at \xy}\foreach[\count=\i from 0\] {\c/\o in {#3} \{\ifx\c\empty\else\edef\ccc{\c}\chit@dbg{2}{Adding \meaning\ccc\space to stack at \xy}' \o'}\expandafter\ccc\xy\\let\ccc\o\expandafter\ccc\xy\tikzmath{coordinate \cc;\cc = (\xy) + (#2);}{\xdef\xy{\cc}\fi\}\@ifnextchar;{\@gobble}{}}

5.5.9 Making order of battle charts

Macros for making OOBs

Style for turns
\tikzset{
chit/oob turn/.pic={\node[pic actions]{#1};}}
current c, current r, n-columns, cell size, y

14777 \def\chit@oob@cellupdate(#1,#2)#3#4#5{% 14778 \edef\f{\ifwg@oob@inv-1\else1\fi}{} 14779 \chit@dbg{1}{ \space Cell update 'c=|#1|' vs '#4'*('#3'-1)} 14780 \pgfmathparse{int(ifthenelse(abs(#1)>=#4*(#3-1),#5-1,#5))} 14781 \xdef#5{\pgfmathresult} 14782 \pgfmathparse{ifthenelse(abs(#1)>=#4*(#3-1),#2-#4,#2)} 14783 \xdef#2{\pgfmathresult} 14784 \pgfmathparse{ifthenelse(abs(#1)>=#4*(#3-1),0,#1+\f*#4)} 14785 \xdef#1{\pgfmathresult} 14786 \chit@dbg{1}{ \space\space-> '\string#5'=#5 '\string#2'=#2 '\string#1'=#1} 14787 }

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

14788 \def\chit@oob@rowupdate(#1,#2)#3#4{% 14789 \chit@dbg{2}{ Row update c='#1',r='#2',s='#3',e='#4'} 14790 \pgfmathparse{ifthenelse(#1>0,#2-#3,#2)} 14791 \xdef#2{\pgfmathresult} 14792 \xdef#1{0} 14793 \chit@dbg{2}{ \space\space-> update '\string#2'=#2} 14794 }

current c, current r, cell size, extra spacing

14796 \def\chit@oob@turnupdate(#1,#2)#3#4{% 14797 \chit@dbg{2}{ Turn update c='#1',r='#2',s='#3',e='#4'} 14798 \pgfmathparse{#2-ifthenelse(#1>0,#3,0)-#4} 14799 \pgfmathparse{#2-#4-ifthenelse(abs(#1)>0.0001,#3,0)} 14800 \xdef#2{\pgfmathresult} 14801 \xdef#1{0} 14802 \chit@dbg{2}{ \space\space-> update '\string#1'=#1,\string#2'=#2} 14803 }

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.

14804 \newif\ifwg@oob@inv\wg@oob@invt\false 14805 \def\chit@oob@spacer{hspace} 14806 \def\wg@star{oob}{\wg@oob@invtrue\wg@oob} 14807 \def\wg@nostar{oob}{\wg@oob@invt\false\wg@oob} 14808 \def\oob{% 14809 \@ifstar{\wg@star{oob} 14810 }{\wg@nostar{oob}} 14811 } 14812 }

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}
\chit@dbg{2}{OOB: '#1'}
\foreach[\count=\ti from 0] \t/y in #1{
\xdef\o{\r}
\def\c{0}
\ifx\t\y\def\y{0}\fi
\chit@dbg{2}{Turn \ti (r=\r,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%
\if\fi%
\ifx\t\empty\else%
\foreach \u/\m in \t{
%% \chit@dbg{2}{ 'u='\m'}
\ifx\u\empty\else\iffx\m\@empty\def\m{1}\fi\fi\iffx\u\chit@oob@spacer\fi
\chit@oob@spacer%
\chit@dbg{2}{OOB Chit is 'u'}%
\pgfmathparse{\c+#4}%
\xdef\c{\pgfmathresult}%
\ifnum\chitdbglvl>2%
\node[minimum width=#3 cm, minimum height=#3 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
\fi
\fi%
\chit@dbg{1}{ End of chits in turn \ti (c=\c, r=\r, o=\o', y=\y')}
\if\fi%
\chit@dbg{1}{ End of chits in turn \ti (c=\c, r=\r, o=\o', y=\y')}
\if\fi%
\chit@dbg{1}{ End of chits in turn \ti (c=\c, r=\r, o=\o', y=\y')}
\if\fi
\def\c{\empty}
\chit@dbg{2}{ Turn is empty, set c='c'}

340
5.5.10 Table of chits

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.

This ‘if’ controls whether to reset the coordinates to the origin when \chits is called. If true, then reset for a new table.
\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=\ti from 0\] \t/\x in #1{\chit@dbg{2}{Turn \t with option \x}\ifx\t\empty\else\foreach \u/\m in \t{\ifx\u\empty\else\chit@dbg{2}{Next chit \u with possible multiplicity \m}\ifx\u\m\def\m{1}\fi\chit@dbg{2}{Next chit \u multiplicity \m}\foreach \n in {1,...,\m}{\ifx\u\chit@blank\chit@dbg{3}{Ignoring blank chit: \u}\else\chit@cellbg(\c,\r){#3}\chit[@u=\ti](\c,\r)\chit@sng@cellupdate(\c,\r){#2}{#3}\fi}\fi}\fi}\fi}\@ifnextchar;{\@gobble}{}\doublechits\@doublechits\chit@dbl@cellupdate\chit@dbl@flip

1. coordinates
2. coordinates
3. **cell-size**

```latex
\def\chit@dbl@flip(#1,#2)#3{\%
\pgfmathparse{-#1}\%
\xdef\mc{\pgfmathresult}\%
}
```

1. coordinates
2. coordinates
3. Number of columns
4. **cell-size**

```latex
\def\chit@dbl@cellupdate(#1,#2)#3#4{\%
\pgfmathparse{ifthenelse(#1<-#4/2,#2,#4+#2)}\%
\xdef#2{\pgfmathresult}\%
\pgfmathparse{ifthenelse(#1<-#4/2,#4+#1,-(#3-.5)*#4)}\%
\xdef#1{\pgfmathresult}\%
}
```

1. List of list of keys
2. Number of columns
3. size of each cell

The stared version (\texttt{\textbackslash doublechits*}) of this macro continues the previously set chit table.

```latex
\def\doublechits{\%
@ifstar{\chits@resetfalse\@doublechits}{\chits@resettrue\@doublechits}}
```

```latex
\def\@doublechits#1#2#3{\%
\chit@dbg{1}{Setting double-sided chits: #1} \%
\ifchits@reset \%
\pgfmathparse{-(#2-.5)*#3} \%
\xdef\c{\pgfmathresult} \%
\def\r{0} \%
\fi \%
\foreach [count=\ti from 0] \t/\x in #1{ \%
\ifx\t\empty\else \%
\foreach \u/\m in \t{ \%
\ifx\u\empty\else \%
\ifx\m\empty\def\m{1}\else\%
\ifx\u\m\def\m{1}\fi\fi \%
\chit@dbg{2}{'\u'='\m' (\c,\r)} \%
\foreach \n in {1,...,\m}{\%
\ifx\u\chit@blank \%
\chit@dbg{3}{Ignoring blank chit:u} \%
\else \%
\chit@cellbg(\c,\r){#3} \%
\chit[\u=\ti](\c,\r) \%
\fi \%
\fi \%
\fi \%
\fi \%
\fi \%
\fi \%
```

343
5.5.11 Battle markers

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

\tikzset{%
  battle marker/.pic={
    \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}}},
}

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.

\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 (.2,.2) {#1};
    },
  odds marker/.style args={#1,#2}{
}
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:


2. \[ ![Image of a 2-dot die] ![Image of a 2-dot die] ![Image of a 2-dot die] ![Image of a 2-dot die] ![Image of a 2-dot die] ![Image of a 2-dot die] \]

3. \[ ![Image of a 3-dot die] ![Image of a 3-dot die] ![Image of a 3-dot die] ![Image of a 3-dot die] ![Image of a 3-dot die] ![Image of a 3-dot die] \]

4. \[ ![Image of a 4-dot die] ![Image of a 4-dot die] ![Image of a 4-dot die] ![Image of a 4-dot die] ![Image of a 4-dot die] ![Image of a 4-dot die] \]

5. \[ ![Image of a 5-dot die] ![Image of a 5-dot die] ![Image of a 5-dot die] ![Image of a 5-dot die] ![Image of a 5-dot die] ![Image of a 5-dot die] \]


\tikzset{
  dice bg/.style={
    % /utils/exec={

Now some shapes of different dice. This was originally done by David Carlisle. Usage is for example

```latex
\node[shape=d\text{dice},\{node options\}] {\{value\}};
```

where \texttt{(dice)} is one of d4, d6, d8, d10, d12, or d20.
5.5.13 Some utilities

Game turn marker

\tikzset{
  chit/text base/.style={
    shape=rectangle,
  }
}
inner sep=0pt,
align=center,
text width=1.1cm},
chit/number/.style={
  chit/text base,
  font=\sffamily\bfseries\fontsize{12}{14}\selectfont},
chit/game turn/.style={
  chit/text base,
  font=\sffamily\bfseries},
chit/text/.style={
  chit/text base,
  font=\sffamily\bfseries},
chit/small text/.style={
  chit/text base,
  font=\sffamily\bfseries\fontsize{9}{10}\selectfont},
chit/number/.pic={\node[chit/number]{#1};},
chit/game turn/.pic={\node[chit/game turn]{Game\ \Turn};},
chit/text/.pic={\node[chit/text]{#1};},
chit/small text/.pic={\node[chit/small text]{#1};},
game turn chit/.style={
  /chit/full={chit/game turn},
  color=black,
  fill=white},
game turn chit flipped/.style={game turn chit},
dummy chit/.style={fill=white},
}

Marks of chits

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

Stacking mark

\tikzset{
  wg stacking/.style={fill=white,
  /chit/symbol=\{[faction=friendly,command=land]\}},
}

\DeclareRobustCommand\stackmark[1][]\{
  \tikz[baseline=(current bounding box.center),scale=.3,#1]{\begin{scope}\[hex/first row and column are=0,
  hex/row direction is=normal,
  hex/column direction is=normal,
  hex/short columns=none\]
  \stackchits(0,0)(.3,-.3){\noexpand\chit[wg stacking],
  \noexpand\chit[wg stacking],
  \noexpand\chit[wg stacking]}\}}

ZOC mark

\DeclareRobustCommand\zocmark[1][]\{
  \tikz[baseline=($(current bounding box.center)!\$.5!(current bounding box.south)$),scale=.1,#1]{\begin{scope}\[hex/first row and column are=0,
  hex/row direction is=normal,
  hex/column direction is=normal,
  hex/short columns=none\]
  \hex[label=,fill=gray](c=1,r=1)\%
  \hex[label=,fill=white](c=1,r=2)\%
  \end{scope}\}}
Dummy implementations of zones hooks when exporting. Here, these do nothing, but in the \texttt{wgexport} class these are re-implemented.

\begin{tikzpicture}
\end{tikzpicture}

5.6 The \texttt{wargame.natoapp6c} TikZ 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} [4] available at CTAN.

5.6.1 Debugging

\texttt{\natoappdbglvl}
\texttt{\n@to@pp@dbg}

Set the debug level, and make debug message.

\begin{tikzpicture}
\end{tikzpicture}

5.6.2 Colours

\texttt{\c@friendly}
\texttt{\c@hostile}
\texttt{\c@neutral}
\texttt{\c@unknown}

Define standard colours for marker affiliations.

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>friendly</td>
<td></td>
</tr>
<tr>
<td>hostile</td>
<td></td>
</tr>
<tr>
<td>neutral</td>
<td></td>
</tr>
<tr>
<td>unknown</td>
<td></td>
</tr>
</tbody>
</table>
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

5.6.4 Some utilities

\n@to@pp@isclip

This detects if we’re in a node that is being used for clipping

\n@to@pp@saved@fill@color
\n@to@pp@saved@stroke@color

Macros to hold saved colours.
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.

\begin{verbatim}
\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
  \meaning\n@to@pp@saved@fill@color
  \meaning\n@to@pp@saved@stroke@color}
}
\newcommand{\n@to@pp@restore@fill}{%
  \message{^^J=== Restore fill color
  \meaning\n@to@pp@saved@fill@color
  \meaning\n@to@pp@saved@stroke@color}
  \ifx\n@to@pp@saved@fill@color\relax\else%
    \expandafter\pgf@setfillcolor\n@to@pp@saved@fill@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%}
\end{verbatim}

We also make an environment, just to simplify the use

\begin{verbatim}
\newenvironment{\n@to@pp@stroketofill}{%
  \pgfscope
  \n@to@pp@stroke@to@fill%
}{\endpgfscope%}
\end{verbatim}

\subsection*{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.

First, the generic bounding box symbol for all markers.

```
\begin{verbatim}
\pgfdeclareshape{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{south west}{\pgf\x=-\radius\pgf\y=-\radius}
\anchor{north 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{%}
\pgfpathmoveto{\pgfpointpolar{0}{\radius}}%\anchor{center}{\center}
\pgfpathlineto{\pgfpointpolar{45}{\radius}}%\anchor{north east}{\pgf\x=\radius\pgf\y=\radius}
\pgfpathlineto{\pgfpointpolar{90}{\radius}}%\anchor{north west}{\pgf\x=-\radius\pgf\y=\radius}
\pgfpathlineto{\pgfpointpolar{135}{\radius}}%\anchor{south east}{\pgf\x=\radius\pgf\y=-\radius}
\pgfpathlineto{\pgfpointpolar{180}{\radius}}%\anchor{south}{\pgf\x=0cm\pgf\y=-\radius}
\pgfpathlineto{\pgfpointpolar{225}{\radius}}%\anchor{south west}{\pgf\x=-\radius\pgf\y=-\radius}
\pgfpathlineto{\pgfpointpolar{270}{\radius}}%\anchor{center}{\center}
\pgfpathlineto{\pgfpointpolar{315}{\radius}}%\anchor{north west}{\pgf\x=-\radius\pgf\y=\radius}
\pgfpathclose}%\anchor{north east}{\pgf\x=\radius\pgf\y=\radius}
\end{verbatim}
```
5.6.7 ‘Friendly’ node shapes

Macro for friendly air shape

```
def\n@to@friendly@@ir{\v@tmpa=\pgf@x\v@tmpb=\pgf@y%\pgfpathmoveto{\pgfqpoint{-\v@tmpa}{\v@tmpb}}%\cntrl{\v@tmpb=\pgf@y%\pgfpatharc{180}{0}{\v@tmpa and \v@tmpb}}}
```

The friendly air command.

```
\pgfdeclareshape{natoapp6c friendly air}{\inherit\saved\anchor{\north}{\north}\anchor{\east}{\east}\anchor{\west}{\west}\anchor{\south east}{\south east}\anchor{\south west}{\south west}\anchor{\north east}{\north east}\anchor{\northeast}{\northeast}}
```

355
natoapp6c friendly land

Macro for friendly land command

```
\def\n@to@friendly@l@nd{\northeast \wg@tmpa=\pgf@x \wg@tmpb=\pgf@y\n\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}\n\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}\n\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}\n\pgfpathlineto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}\n\pgfclosepath}
```

The friendly land command. The most used command frame.

```latex
\pgfdeclareshape{natoapp6c friendly land}{\n\inheritsavedanchors[from=natoapp6c base]\n\savedanchor\northeast{\pgf@x=1.5\n\northeast\pgf@y=0}\n\anchor{north east}{\northeast}\n\anchor{north west}{\northeast-\pgf@x}\n\anchor{south east}{\northeast-\pgf@y}\n\anchor{south west}{\northeast-\pgf@x-\pgf@y}\n\anchor{north}{\northeast-\pgf@x=0cm}\n\anchor{south}{\northeast-\pgf@y=0cm}\n\anchor{east}{\northeast-\pgf@y=0cm}\n\anchor{west}{\northeast-\pgf@x=0cm}\n\inheritanchor[from=natoapp6c base]{upper}\n\inheritanchor[from=natoapp6c base]{lower}\n\inheritanchor[from=natoapp6c base]{left}\n\inheritanchor[from=natoapp6c base]{right}\n\inheritanchor[from=natoapp6c base]{center}\n```
The friendly activity command. Similar to land command, but with boxes in the corners.
The friendly equipment command. A circle.
The friendly installation command. Similar to the land command, but with a ‘hat’ on top.

\begin{tikzpicture}
\node (natoapp6c friendly installation) at (0,0) [shape=natoapp6c friendly installation] {natoapp6c friendly installation};
\end{tikzpicture}
The friendly sea surface command. Same as equipment command.

15578 \pgfdeclareshape{natoapp6c friendly sea surface}{
15579 \inheritsavedanchors[from=natoapp6c friendly equipment]
15580 \inheritanchor[from=natoapp6c friendly equipment]{inner north east}
15581 \inheritanchor[from=natoapp6c friendly equipment]{inner north west}
15582 \inheritanchor[from=natoapp6c friendly equipment]{inner south west}
15583 \inheritanchor[from=natoapp6c friendly equipment]{inner south east}
15584 \inheritanchor[from=natoapp6c friendly equipment]{north east}
15585 \inheritanchor[from=natoapp6c friendly equipment]{north west}
15586 \inheritanchor[from=natoapp6c friendly equipment]{south east}
15587 \inheritanchor[from=natoapp6c friendly equipment]{south west}
15588 \inheritanchor[from=natoapp6c friendly equipment]{north}
15589 \inheritanchor[from=natoapp6c friendly equipment]{west}
15590 \inheritanchor[from=natoapp6c friendly equipment]{east}
15591 \inheritanchor[from=natoapp6c friendly equipment]{south}
15592 \inheritanchor[from=natoapp6c friendly equipment]{upper}
15593 \inheritanchor[from=natoapp6c friendly equipment]{lower}
15594 \inheritanchor[from=natoapp6c friendly equipment]{left}
15595 \inheritanchor[from=natoapp6c friendly equipment]{right}
15596 \inheritanchor[from=natoapp6c friendly equipment]{center}
15597 \inheritbackgroundpath[from=natoapp6c friendly equipment]
15598 \inheritbehindforegroundpath[from=natoapp6c friendly equipment]
15599 }

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

15600 \pgfdeclareshape{natoapp6c friendly space}{
15601 \inheritsavedanchors[from=natoapp6c friendly air]
15602 \inheritanchor[from=natoapp6c friendly air]{north east}
15603 \inheritanchor[from=natoapp6c friendly air]{north west}
15604 \inheritanchor[from=natoapp6c friendly air]{south east}
15605 \inheritanchor[from=natoapp6c friendly air]{south west}
15606 \inheritanchor[from=natoapp6c friendly air]{north}
15607 \inheritanchor[from=natoapp6c friendly air]{west}
15608 \inheritanchor[from=natoapp6c friendly air]{east}
15609 \inheritanchor[from=natoapp6c friendly air]{south}
15610 \inheritanchor[from=natoapp6c friendly air]{upper}
15611 \inheritanchor[from=natoapp6c friendly air]{lower}
15612 \inheritanchor[from=natoapp6c friendly air]{left}
15613 \inheritanchor[from=natoapp6c friendly air]{right}
15614 \inheritanchor[from=natoapp6c friendly air]{center}
15615 \inheritbackgroundpath[from=natoapp6c friendly air]
15616 \behindforegroundpath{
15617 \begin{tikzpicture}
15618 \node[natoapp6c friendly air] at (0,0) {\textbullet};
15619 \end{tikzpicture}}
15620 \begin{scope}
15621 \path (0,0) -- (0,1) -- (1,1) -- (1,0) -- cycle;
15622 \end{scope}
15623 }}
Macro for friendly sub surface command

\natoapp6c friendly sub surface

The friendly sub surface command.

\pgfdeclareshape{natoapp6c friendly sub surface}{%
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}}% 
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpc}}% 
\pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpd}}% 
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpc}}% 
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}% }

The hostile air command.

\pgfdeclareshape{natoapp6c hostile air}{
\inheritsavedanchors[\from=natoapp6c base]\n\savedanchor{\south east}{\southwest \n@to@pp\r% 
\pgf@x=\n@to@pp\r% 
\pgf@y=-\n@to@pp\r} 
\savedanchor{\cntrl}{\southwest \pgf@x=\n@to@pp\r% 
\pgf@x=\n@to@pp\r% 
\pgf@y=0.414\n@to@pp\r% (sqrt(2)-1)} 
\anchor{\south east}{\southwest} 
\anchor{\south west}{\southwest \pgf@x=-\pgf@x} 
\anchor{\north east}{\southwest \wg@tmpa=\pgf@x \north \pgf@x=\wg@tmpa} 
\anchor{\north west}{\southwest \wg@tmpa=\pgf@x \north \pgf@x=-\wg@tmpa} 
\anchor{\north}{\north} 
\anchor{\east}{\north \pgf@y=\wg@tmpb} 
\anchor{\west}{\north \pgf@y=-\wg@tmpb} 
\anchor{\south}{\southwest \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}}
Macro for hostile land command

```
\def\n@to@hostile@l@nd{\northeast \wg@tmpa=\pgf@x\wg@tmpb=\pgf@y\pgfpathmoveto{\wg@tmpa}{0cm}\pgfpathlineto{0cm}{\wg@tmpb}\pgfpathlineto{0cm}{-\wg@tmpa}\pgfpathlineto{-\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=0cm\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}\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@l@nd}\behindforegroundpath{\n@to@hostile@l@nd}\pgfusepath{stroke}}
```
The hostile activity command. Similar to land command, but with boxes in the corners.
natoapp6c hostile equipment

The hostile equipment command. Same as land command.

natoapp6c hostile installation

The hostile installation command. Similar to land command, but with a ‘hat’ on top.
natoapp6c hostile sea surface

The hostile sea surface command. Same as land command.

\pgfdeclareshape{natoapp6c hostile sea surface}{
\inheritsavedanchors[from=natoapp6c hostile equipment]
\inheritanchor[from=natoapp6c hostile equipment]{inner north east}
\inheritanchor[from=natoapp6c hostile equipment]{inner north west}
\inheritanchor[from=natoapp6c hostile equipment]{inner south west}
\inheritanchor[from=natoapp6c hostile equipment]{inner south east}
\inheritanchor[from=natoapp6c hostile equipment]{north east}
\inheritanchor[from=natoapp6c hostile equipment]{north west}
\inheritanchor[from=natoapp6c hostile equipment]{south east}
\inheritanchor[from=natoapp6c hostile equipment]{south west}
\inheritanchor[from=natoapp6c hostile equipment]{north}
\inheritanchor[from=natoapp6c hostile equipment]{west}
\inheritanchor[from=natoapp6c hostile equipment]{east}
\inheritanchor[from=natoapp6c hostile equipment]{south}
\inheritanchor[from=natoapp6c hostile equipment]{upper}
\inheritanchor[from=natoapp6c hostile equipment]{lower}
\inheritanchor[from=natoapp6c hostile equipment]{left}
\inheritanchor[from=natoapp6c hostile equipment]{right}
\inheritanchor[from=natoapp6c hostile equipment]{center}
\inheritbackgroundpath[from=natoapp6c hostile equipment]
\behindforegroundpath{\pgfusepath{stroke}}
}

natoapp6c hostile space

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

\pgfdeclareshape{natoapp6c hostile space}{
\inheritsavedanchors[from=natoapp6c hostile air]
\inheritanchor[from=natoapp6c hostile air]{north east}
\inheritanchor[from=natoapp6c hostile air]{north west}
\inheritanchor[from=natoapp6c hostile air]{south east}
\inheritanchor[from=natoapp6c hostile air]{south west}
\inheritanchor[from=natoapp6c hostile air]{north}
\inheritanchor[from=natoapp6c hostile air]{west}
\inheritanchor[from=natoapp6c hostile air]{east}
\inheritanchor[from=natoapp6c hostile air]{south}
\inheritanchor[from=natoapp6c hostile air]{upper}
\inheritanchor[from=natoapp6c hostile air]{lower}
\inheritanchor[from=natoapp6c hostile air]{left}
\inheritanchor[from=natoapp6c hostile air]{right}
\inheritanchor[from=natoapp6c hostile air]{center}
\inheritbackgroundpath[from=natoapp6c hostile air]
\behindforegroundpath{\pgfusepath{stroke}}
}
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{\n@to@pp@r}{\wg@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\n@to@pp@r}{\wg@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\n@to@pp@r}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\n@to@pp@r}{\wg@tmpb}}%
\pgfclosepath%
\pgfusepath{fill}
}

The hostile sub surface command

\pgfdeclareshape{natoapp6c hostile sub surface}{%
\inheritsavedanchors[from=natoapp6c base]
\savedanchor{north east}{\northeast}
\anchor{north west}{\northeast\wg@tmpa=\pgf@x\pgf@x=\wg@tmpa}
\anchor{south east}{\northeast\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y}
\anchor{south west}{\northeast\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y}
\anchor{south}{\south}
\anchor{east}{
\northeast\wg@tmpb=\pgf@y\wg@tmpa=\pgf@x%
\south\wg@tmpb=\pgf@y%
\advance\wg@tmpb-\wg@tmpc
\divide\wg@tmpb2%
}

369
5.6.9 ‘Neutral’ node shapes

Macro for neutral shapes

```
\def\n@to@pp@neutr@l@init{\northeast{\pgf@x=\wg@tmpa\pgf@y=\wg@tmpb}}
\def\n@to@pp@neutr@l@left {\pgflineto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@right {\pgflineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@top {\pgflineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@bottom{\pgflineto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@nw {\pgfmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@ne {\pgfmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@se {\pgfmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpb}}}\%
\def\n@to@pp@neutr@l@sw {\pgfmoveto{\pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}}\%
```

The neutral air command

```
\pgfdeclareshape{natoapp6c neutral air}{% 
\inheritsavedanchors[from=natoapp6c base]
\savedanchor{northeast}{\pgf@x=\n@to@pp@r\pgf@y=\n@to@pp@r}
\anchor{north east}{\northeast}
```
natoapp6c neutral land

The neutral land command

```
pgfdeclareshade{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]{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\n\n@to@pp@neutr@l\n\n@to@pp@neutr@l\n\n@to@pp@neutr@l\n\n@to@pp@neutr@l\n\n@to@pp@neutr@l\n\n@to@pp@neutr@l\n
\pgfusepath{stroke}%
}
```

371
The neutral activity command. Similar to land command but with boxes added in the corners.
The neutral equipment command. Same as land command.
The neutral installation command. Similar to land command but with a ‘hat’ on top.

\begin{natoapp6c neutral installation}
\end{natoapp6c neutral installation}
The neutral sea surface command. Same as land command.

```
\pgfdeclareshape{natoapp6c neutral sea surface}{
\inherit\savedanchors[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}{
\inherit\savedanchors[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}
```
The neutral sub surface command

\pgfdeclareshape{natoapp6c neutral sub surface}{%}
\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{\begin{n@to@pp@stroketofill}
\n@to@pp@neutral@init%
\n@to@pp@neutral@se
\n@to@pp@neutral@right%
\n@to@pp@neutral@top%
\n@to@pp@neutral@left%
\pgfusepath{stroke}%
\%
\n@to@pp@neutral@ne
\n@to@pp@neutral@top%
\northeast\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\advance\wg@tmpb-\n@to@pp@space@h
\%
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpc}}%
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{ \wg@tmpa}{\wg@tmpb}}%
\pgfclosepath%
\pgfusepath{fill}%
\end{n@to@pp@stroketofill}
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% 
\ctrlnortheast \wg@tmpb=\pgf@x% 
\pgfpathcurveto{% 
\pgf@point{ \wg@tmpa}{\wg@tmpb}% 
\pgf@point{-\wg@tmpa}{\wg@tmpb}% 
\pgf@point{-\wg@tmpa}{\wg@tmpa}% }
\def\n@to@pp@unknown@left{% 
\innernortheast \wg@tmpa=\pgf@x% 
\ctrlnortheast \wg@tmpb=\pgf@x% 
\pgfpathcurveto{% 
\pgf@point{-\wg@tmpb}{\wg@tmpa}% 
\pgf@point{-\wg@tmpb}{-\wg@tmpa}% 
\pgf@point{-\wg@tmpa}{-\wg@tmpa}% }
\def\n@to@pp@unknown@bottom{% 
\innernortheast \wg@tmpa=\pgf@x% 
\ctrlnortheast \wg@tmpb=\pgf@x% 
\pgfpathcurveto{% 
\pgf@point{-\wg@tmpa}{-\wg@tmpb}% 
\pgf@point{-\wg@tmpa}{-\wg@tmpb}% 
\pgf@point{-\wg@tmpa}{-\wg@tmpa}% }
\def\n@to@pp@unknown@right{% 
\innernortheast \wg@tmpa=\pgf@x% 
\ctrlnortheast \wg@tmpb=\pgf@x% 
\pgfpathcurveto{% 
\pgf@point{-\wg@tmpb}{-\wg@tmpa}% 
\pgf@point{-\wg@tmpb}{\wg@tmpa}% 
\pgf@point{\wg@tmpa}{\wg@tmpa}% }
}
The unknown land command

\begin{verbatim}
\pgfdeclarelength{\natoapp\unknown@width}
\pgfmathsetlengthmacro{\natoapp\unknown@width}{2.2\fig@width}
\pgfdeclarelength{\natoapp\unknown@height}
\pgfmathsetlengthmacro{\natoapp\unknown@height}{2.2\fig@height}
\pgfdeclareshape{natoapp6c unknown land}{
\inheritsavedanchors[from=natoapp6c base]
\savedanchor\innernortheast{\pgf@x=.7\natoapp\unknown@width\pgf@y=.7\natoapp\unknown@height}
\savedanchor\cntrlnortheast{\pgf@x=1.6\natoapp\unknown@width\pgf@y=1.6\natoapp\unknown@height}
\savedanchor\northeast{\pgf@x=1.4\natoapp\unknown@width\pgf@y=1.4\natoapp\unknown@height}
\anchor{inner north east}{\innernortheast}
\anchor{inner north west}{\innernortheast\pgf@x=-\pgf@x}
\anchor{inner south west}{\innernortheast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{inner south east}{\innernortheast\pgf@y=-\pgf@y}
\anchor{northeast}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{north east}{\innernortheast\pgf@x=0cm}
\anchor{north west}{\innernortheast\pgf@x=0cm\pgf@y=-\pgf@y}
\anchor{south west}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{south}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{north}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{west}{\innernortheast\pgf@x=0cm}
\anchor{east}{\innernortheast\pgf@y=0cm}
\anchor{center}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{upper}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{lower}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{left}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\anchor{right}{\innernortheast\pgf@x=0cm\pgf@y=0cm}
\backgroundpath{\natoapp\unknown@init
\innernortheast \wg@tmpa=\pgf@x
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpa}}
\natoapp\unknown@right \natoapp\unknown@top \natoapp\unknown@left \natoapp\unknown@bottom}
\behindforegroundpath{\natoapp\unknown@init
\innernortheast \wg@tmpa=\pgf@x
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{-\wg@tmpa}}
\natoapp\unknown@right \natoapp\unknown@top \natoapp\unknown@left \natoapp\unknown@bottom
\pgfusepath{stroke}}
\end{verbatim}

natoapp6c unknown land

natoapp6c unknown air
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]{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]{north}
  \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}
  \inheritanchor[\from=natoapp6c unknown land]{right}
  \inheritanchor[\from=natoapp6c unknown land]{center}
  \anchor{south}{\innernortheast\pgf@x=0cm\pgf@y=-\pgf@y}
  \anchor{south east}{\northeast\wg@tmpa=\pgf@x\innernortheast\pgf@y=-\pgf@y \pgf@x=\wg@tmpa}
  \anchor{south west}{\northeast\wg@tmpa=\pgf@x\innernortheast\pgf@y=-\pgf@y \pgf@x=-\wg@tmpa}
  \backgroundpath{\n@to@pp@unknown@init \innernortheast \wg@tmpa=\pgf@x \pgfpathmoveto{\wg@tmpa}{-\wg@tmpa}\n@to@pp@unknown@right \n@to@pp@unknown@top \n@to@pp@unknown@left \ifn@to@pp@isclip \pgfpathlineto{0cm}{-\radius}\pgfusepath{stroke}\fi}
  \behindforegroundpath{\n@to@pp@unknown@init \innernortheast \wg@tmpa=\pgf@x \pgfpathmoveto{\wg@tmpa}{-\wg@tmpa}\n@to@pp@unknown@right \n@to@pp@unknown@top \n@to@pp@unknown@left \pgfpathclose}
\}

379
The unknown activity command. Similar to land command, but with boxes in the the ‘corners’.

\pgfdeclarereshape{natoapp6c unknown activity}{
\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]
\behindforegroundpath{
\n@to@pp@unknown@init
\innernortheast \wg@tmpa=\pgf@x%
\begin{n@to@pp@stroketofill}
\pgfpathmoveto{\pgfqpoint{\ug@tmpa}{-\ug@tmpa}}%
\n@to@pp@unknown@right %
\n@to@pp@unknown@top %
\n@to@pp@unknown@left %
\n@to@pp@unknown@bottom%
\pgfusepath{stroke,clip}
%
\n@to@pp@unknown@northeast\n@to@pp@unknown@northwest\n@to@pp@unknown@southeast\n@to@pp@unknown@southwest\n\pgadvance\ug@tmpa0.005cm
\wg@tmpb=\wg@tmpa
\pgfpathlineto{\pgfqpoint{-\wg@tmpc}{\wg@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\wg@tmpc}{\wg@tmpb}}%
\pgfclosepath%
\pgfusepath{fill}
%
\pgfpathmoveto{\pgfqpoint{\ug@tmpc}{\ug@tmpb}}%
\pgfpathlineto{\pgfqpoint{\ug@tmpc}{\ug@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\ug@tmpc}{\ug@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\ug@tmpc}{\ug@tmpb}}%
\pgfclipto{\pgfpoint{-\ug@tmpc}{\ug@tmpb}}%
\pgfusepath{fill}
%
\pgfpathmoveto{\pgfqpoint{\ug@tmpc}{\ug@tmpb}}%
\pgfpathlineto{\pgfqpoint{-\ug@tmpc}{\ug@tmpb}}%
\pgfpathlineto{\pgfqpoint{-\ug@tmpc}{\ug@tmpb}}%
The unknown equipment command. Same as land command.

\pgfdeclareshape{natoapp6c unknown equipment}{
\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 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).

\begin{verbatim}
\pgfdeclarerectshape{natoapp6c unknown installation}{
\pgfshapeinnersep=0pt\pgfshapeoutersep=0pt
\pgfdeclareinnerpath{\n@to@pp@unknown@init
\innernortheast \wg@tmpa=\pgf@x%
\innernortheast \n@to@pp@unknown@right %
\innernortheast \n@to@pp@unknown@top %
\innernortheast \n@to@pp@unknown@left %
\innernortheast \n@to@pp@unknown@bottom%
\pgfusepath{stroke} %
\begin{n@to@pp@stroketofill}
\pgfpoint{-\wg@tmpa}{\wg@tmpb}
\pgfpoint{\wg@tmpa}{\wg@tmpb}
\pgfclosepath
\pgfusepath{clip}%
\end{verbatim}

382
The unknown sea surface command. Same as land command

```
\pgfdeclareshape{natoapp6c unknown sea surface}{%
\inherit{savedanchors}[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.

```
\pgfdeclareshape{natoapp6c unknown space}{%
\inherit{savedanchors}[from=natoapp6c unknown air]
\inheritanchor[from=natoapp6c unknown air]{inner north east}
\inheritanchor[from=natoapp6c unknown air]{inner north west}
```
The unknown sub surface command.

```
pgfdeclareshape{natoapp6c unknown sub surface}{% 
\inheritsaveanchors[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 east}
\inheritanchor[from=natoapp6c unknown land]{inner south west}
\inheritanchor[from=natoapp6c unknown land]{north east}
\inheritanchor[from=natoapp6c unknown land]{north 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}
\behindbackgroundpath{% 
\n@to@pp@unknown@init
\innernortheast \wg@tmpa=\pgf@x%
\begin{n@to@pp@stroketofill}
\pgfpathmoveto{\pgfqpoint{\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
\n@to@pp@space@h
%
\pgfpathmoveto{\pgfqpoint{\radius}{\wg@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\radius}{\wg@tmpa}}%
\pgfpathlineto{\pgfqpoint{-\radius}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\radius}{\wg@tmpb}}%
\pgfclosepath%
\pgfusepath{fill}%
\end{n@to@pp@stroketofill}%
}%
\end{softmax}
```
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@o{0d}(\$\text{#1}+\n@to@pp@e@y,0)\$) \text{circle(0.09)}
\def\n@to@pp@e@o{0b}\text{((#1+$\n@to@pp@e@y,0)$)}
\def\n@to@pp@e@o{0o}\text{((#1+$\n@to@pp@e@y,0)$)}
\def\n@to@pp@e@o{0b}\text{((#1+$\n@to@pp@e@y,0)$)}
\def\n@to@pp@e@o{0o}\text{((#1+$\n@to@pp@e@y,0)$)}
Pictures

\tikzset{
    pics/natoapp6c/s/echelon/.is choice,
    pics/natoapp6c/s/echelon/squad/.style={code={
        \path[draw,fill=pgfstrokecolor,pic actions]
        \foreach \o in {0}\n\@to@pp@e@d{\o};}},
    pics/natoapp6c/s/echelon/section/.style={code={
        \path[draw,fill=pgfstrokecolor,pic actions]
        \foreach \o in {-1,1}\n\@to@pp@e@o{\o};}},
    pics/natoapp6c/s/echelon/platoon/.style={code={
        \path[draw,fill=pgfstrokecolor,pic actions]
        \foreach \o in {-2,0,2}\n\@to@pp@e@o{\o};}},
    pics/natoapp6c/s/echelon/company/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0}\n\@to@pp@e@b{\o};}},
    pics/natoapp6c/s/echelon/battalion/.style={code={
        \path[draw,pic actions]
        \foreach \o in {-1,1}\n\@to@pp@e@b{\o};}},
    pics/natoapp6c/s/echelon/regiment/.style={code={
        \path[draw,pic actions]
        \foreach \o in {-1,1}\n\@to@pp@e@b{\o};}},
    pics/natoapp6c/s/echelon/brigade/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0}\n\@to@pp@e@x{\o};}},
    pics/natoapp6c/s/echelon/division/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0}\n\@to@pp@e@x{\o};}},
    pics/natoapp6c/s/echelon/corps/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0}\n\@to@pp@e@x{\o};}},
    pics/natoapp6c/s/echelon/army/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0}\n\@to@pp@e@x{\o};}},
    pics/natoapp6c/s/echelon/army group/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0}\n\@to@pp@e@x{\o};}},
    pics/natoapp6c/s/echelon/theatre/.style={code={
        \path[draw,pic actions]
        \foreach \o in {0,1,2,3,4} \n\@to@pp@e@x{\o};}},
    pics/natoapp6c/s/echelon/command/.style={code={
        \path[draw,pic actions]
        \foreach \o in {-3,-1,3,5} \n\@to@pp@e@x{\o};}},
    \path[draw,pic actions] (M.north west) rectangle
    (M.north east)+(0,1)\$;},
}
5.6.12 Text on symbols

\begin{verbatim}
/tikz/natoapp6c/normal text
/tikz/natoapp6c/squashed text
/tikz/natoapp6c/small text
/tikz/natoapp6c/small squashed text
\end{verbatim}

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).

\begin{verbatim}
\newcommand\n@to@ppfont[2][b]{{%\fontencoding{T1}\fontfamily{qhv}\fontseries{#1}\fontsize{#2}{0}\selectfont}
\tikzset{%
\natoapp6c/text/.style={shape=rectangle,draw=none,fill=none,transform shape,anchor=center},
\natoapp6c/normal text/.style={font=\n@to@ppfont{12}},
\natoapp6c/squashed text/.style={font=\n@to@ppfont[bc]{12}},
\natoapp6c/small text/.style={font=\n@to@ppfont{10}},
\natoapp6c/squashed small text/.style={font=\n@to@ppfont[bc]{10}},
\n@to@pp@text@normal
\n@to@pp@text@squashed
\n@to@pp@text@small
\n@to@pp@text@smallsquashed}
\end{verbatim}

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

\begin{verbatim}
\n@to@pp@text@normal[2][] {%
\node[\natoapp6c/text,\natoapp6c/normal text,#1]{#2}{}
\n@to@pp@text@squashed[2][] {%
\node[\natoapp6c/text,\natoapp6c/squashed text,#1]{#2}{}
\n@to@pp@text@small[2][] {%
\node[\natoapp6c/text,\natoapp6c/small text,#1]{#2}{}
\n@to@pp@text@smallsquashed[2][] {%
\node[\natoapp6c/text,\natoapp6c/squashed small text,#1]{#2}{}
\end{verbatim}

5.6.13 Text \natoapp6c namespace

\begin{verbatim}
\natoapp6c
\end{verbatim}

Here, we set up the key path /natoapp6c

\begin{verbatim}
\def\natoapp@report{}
\tikzset{
/natoapp6c/.search also=\tikz,\n/natoapp6c/.cd,
\end{verbatim}
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}
\tikzset{
/natoapp6c/.cd,
\t/natoapp6c/id/.store in=\natoapp@id,
\t/fac/.store in=\natoapp@fac,
\t/cmd/.store in=\natoapp@cmd,
\t/ech/.store in=\natoapp@ech,
}
\end{verbatim}

\begin{verbatim}
/natoapp6c/faction
\end{verbatim}

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

\begin{verbatim}
\tikzset{
\natoapp6c/.cd,
\n/faction/.is choice,
\t/faction/none/.code={\let\natoapp@fac\@undefined},
\t/faction/friendly/.style={fac=friendly},
\t/faction/hostile/.style={fac=hostile},
\t/faction/enemy/.style={fac=hostile},
\t/faction/neutral/.style={fac=neutral},
\t/faction/unknown/.style={fac=unknown},
\t/faction/?/.style={fac=unknown},
\t/faction/.initial=friendly,
\}
\end{verbatim}

\begin{verbatim}
/natoapp6c/command
\end{verbatim}

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

\begin{verbatim}
\tikzset{
\natoapp6c/.cd,
\n/command/.is choice,
\t/command/base/.style={cmd=base},
\t/command/activity/.style={cmd=activity},
\t/command/air/.style={cmd=air},
\t/command/missile/.style={cmd=air},
\t/command/equipment/.style={cmd=equipment},
\t/command/installation/.style={cmd=installation},
\}
\end{verbatim}
natoapp6c/echelon

Unit size. The choice is limited to one of the below. The choice is stored in the key natoapp6c/ech.

\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},
}

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

```latex
\newif\ifnatoapp@decoy
\natoapp@decoyfalse
\tikzset{
/natoapp6c/.cd,
main/.store in=\natoapp@main, main/.initial=,%
left/.store in=\natoapp@left, left/.initial=,%
right/.store in=\natoapp@right, right/.initial=,%
upper/.store in=\natoapp@upper, upper/.initial=,%
lower/.store in=\natoapp@lower, lower/.initial=,%
top/.store in=\natoapp@upper,%
bottom/.store in=\natoapp@lower,%
below/.store in=\natoapp@below, below/.initial=,%
frame/.store in=\natoapp@frame, frame/.initial=,%
decoy/.is if=natoapp@decoy,%
}
```

Styles used by the various parts of the symbol.

```latex
\tikzset{
/natoapp6c/parts/.style={
scale line widths,
draw,
shape=rectangle,
transform shape},
natoapp6c/main/.style={natoapp6c/parts},
natoapp6c/modifiers/.style={natoapp6c/parts, scale=.6},
natoapp6c/lower/.style={natoapp6c/parts},
natoapp6c/upper/.style={natoapp6c/parts},
natoapp6c/left/.style={natoapp6c/parts},
natoapp6c/right/.style={natoapp6c/parts},
natoapp6c/echelon/.style={natoapp6c/parts},
natoapp6c/below/.style={natoapp6c/parts}
}
```

5.6.14 The natoapp6c styles

```latex
\tikz/natoapp6c
```

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.

We define a counter to set-up unique names for symbol nodes.

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

5.6.15 The \natoapp6c shape

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).
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}\}
\anchor{left}\{\wg@subanchor{M\id}{left}\}
\anchor{right}\{\wg@subanchor{M\id}{right}\}

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

--- ID: \meaning\natoapp@id
--- Faction: \meaning\natoapp@fac
--- Command: \meaning\natoapp@cmd
--- Echelon: \meaning\natoapp@ech
--- Main: \meaning\natoapp@main
--- Left: \meaning\natoapp@left
--- Right: \meaning\natoapp@right
--- Upper: \meaning\natoapp@upper
--- Lower: \meaning\natoapp@lower
--- Below: \meaning\natoapp@below
--- Shape: \meaning\frameshape
--- Options: \meaning\frameopt\}
If the symbol is empty, then do nothing.

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

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.

\begin{scope}
  \pgfinterruptboundingbox
  \%% Clip to shape in scope
  \%% \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@isclipfalse%
\end{scope}

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.

\begin{scope}
  \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%
\end{scope}

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).

\begin{scope}
  \ifundefined{natoapp@main}{}{
    \n@to@pp@dbg{2}{NATO App6(c) mains: \meaning\natoapp@main}
    \begin{scope}[natoapp6c/main]
      \wg@pic@all{\natoapp@main}{natoapp6c/s/}{M.center}{natoapp6c/main}%
    \end{scope}
  }
  \\\n  \% Modifiers flagged
\end{scope}
```

393
The next thing is to render the various modifiers. We start by flagging this globally.

\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 \texttt{M}).

\% Render lowers
\@ifundefined{natoapp@lower}{}{%
\begin{scope}
\wg@pic@all{natoapp@lower}{natoapp6c/s/}{M.lower}{natoapp6c/modifiers,natoapp6c/lower}%
\end{scope}%
\}%
\% Render uppers
\@ifundefined{natoapp@upper}{}{%
\begin{scope}
\wg@pic@all{natoapp@upper}{natoapp6c/s/}{M.upper}{natoapp6c/modifiers,natoapp6c/upper}%
\end{scope}%
\}%
\% Render lefts
\@ifundefined{natoapp@left}{}{%
\begin{scope}
\wg@pic@all{natoapp@left}{natoapp6c/s/}{M.left}{natoapp6c/modifiers,natoapp6c/left}%
\end{scope}%
\}%
\% Render rights
\@ifundefined{natoapp@right}{}{%
\begin{scope}
\wg@pic@all{natoapp@right}{natoapp6c/s/}{M.right}{natoapp6c/modifiers,natoapp6c/right}%
\end{scope}%
\}%
\% Modifiers end
\n@to@pp@modfalse%
\endpgfinterruptboundingbox
\end{scope}%
\fi%
}

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.

\behindforegroundpath{%
\n@to@pp@dbg{2}{NATO App6(c) foreground path:
\%-- J Echelon: \meaning\natoapp@ech
\%-- J Symbol: \meaning\frameshape
\%-- J Below: \meaning\natoapp@below
\%-- J Frame: \meaning\frameopt}
\%}

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 way refer to it later on.

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

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

If the decoy flag was set, we draw that.
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[(\text{natoapp6c options})](\text{(position)})\text{(identifier)};

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

This macro forwards to \n@toapp.

\n@toapp

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

\n@to@pp

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@to@pp\#1\#2\#3{\%}
5.6.17 Macros for markings

\natoappmark

A macro for making NATO App6(c) markings.

\providecommand\natoappmark[2]\{%,
\tikz[transform shape,
\scale=.25,
baseline=(natoapp6c mark.south east),
natoapp6c mark/.try,
\#1\]{%1
\node[draw,transform
shape,natoapp6c={faction=friendly,command=land,
main=#2}] (natoapp6c mark){}}
% \natoapp[faction=friendly,command=land,main=#2](0,0)(natoapp6c mark)}

\echelonmark

\providecommand\echelonmark[2]\{\tikz[transform shape,scale=.5,#1]\{%
\pic[scale line widths,line width=1pt] {natoapp6c/s/echelon=#2};\}}

Some specific NATO App6(c) markers.

\DeclareRobustCommand\armouredmark[1]\{\natoappmark[#1]{armoured}\}
\DeclareRobustCommand\infantrymark[1]\{\natoappmark[#1]{infantry}\}
\DeclareRobustCommand\artillerymark[1]\{\natoappmark[#1]{{[fill=pgfstrokecolor]artillery}}\}
\DeclareRobustCommand\combinedmark[1]\{\natoappmark[#1]{{combined arms}}\}
\DeclareRobustCommand\pgmark[1]\{\natoappmark[#1]{{armoured,infantry}}\}
\DeclareRobustCommand\reconnaissancemark[1]\{\natoappmark[#1]{{reconnaissance}}\}
\DeclareRobustCommand\corpsmark[1]\{\natoappmark[#1]{{echelon=corps}}\}
\DeclareRobustCommand\divisionmark[1]\{\natoappmark[#1]{{echelon=division}}\}
\DeclareRobustCommand\brigademark[1]\{\natoappmark[#1]{{echelon=brigade}}\}
\DeclareRobustCommand\regimentmark[1]\{\natoappmark[#1]{{echelon=regiment}}\}
\DeclareRobustCommand\sofmark[1]\{\natoappmark[#1]{{infantry,text=SOF}}\}
\DeclareRobustCommand\mountaineermark[1]\{\natoappmark[#1]{{infantry,lower=mountain}}\}
\DeclareRobustCommand\airbornemark[1]\{\natoappmark[#1]{{infantry,lower=airborne}}\}
\DeclareRobustCommand\amphibiousmark[1]\{\natoappmark[#1]{{,lower=amphibious}}\}
\DeclareRobustCommand\airassaultmark[1]\{\natoappmark[#1]{{,upper=air assault}}\}

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 for friendly, hostile, neutral, and unknown factions.

Corps support, base

Special placeholder for symbols To Be Done.

5.6.19 Symbols used when defining weaponry
\begin{tikzpicture}
\end{tikzpicture}

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

\begin{tikzpicture}
\end{tikzpicture}
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.
\tikzset{
\natoapp6c/s/air assault/.pic={%
\path[draw] ([shift={((150:.4)}]}0,-.1)--(0,-.1)--([shift={((30:.4)}]}0,-.1);}%,
}

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

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

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

\tikzset{
\natoapp6c/s/airship/.pic={%
\path (0.45, 0.175) rectangle (-0.45, -0.175);
\begin{tikzpicture}
\tikzset{
natoapp6c/s/amphibious warfare ship/.pic={
\pic{natoapp6c/s/warfare vessel};
\path[draw,fill=pgfstrokecolor]
(0.15, 0.05) --
(0.15, 0.2) --
(-0.15, 0.2) --
(-0.15, 0.05) -- cycle
(0, -0.2) rectangle (0.25, -0.175);},}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
natoapp6c/s/analysis/.pic={
\path[pic actions]
(-0.3,-0.2)--(0.3,-0.2)--(0, -0.4)--cycle (0,-0.2)--(0,0.4);},}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
natoapp6c/s/arrest/.pic={
\path[pic actions] circle(0.2);}
\end{tikzpicture}
natoapp6c/s/armoured personnel carrier

\tikzset{\n  natoapp6c/s/armoured personnel carrier/.pic={\n    \path[draw\{\subpic actions\}](natoapp6c/s/vehicle);\n    \path[pic actions] (.35,.15)--(0,.3)--(-.35,.15);},\n}\n
natoapp6c/s/arctic

\tikzset{\n  natoapp6c/s/arctic/.pic={\n    \draw (-0.325,0.135) arc(180:270:0.075 and 0.15) +(0.5, 0) arc(-90:0:0.075 and 0.15);},\n}\n
natoapp6c/s/automobile

\tikzset{\n  natoapp6c/s/automobile/.pic={\n\begin{scope}\n  \clip (0.2,-0.15) circle(0.05) (-0.2,-0.15) circle(0.05) \[reverseclip\];\n  \path[pic actions] (0.3, -0.15) -- (-0.3, -0.15) -- (-0.3, 0.025) -- (-0.1, 0.025) -- (-0.1, 0.2) -- ( 0.1, 0.2) -- ( 0.1, 0.025) -- ( 0.3, 0.025) -- cycle\n  ( 0.075, 0.025) rectangle (-0.075, 0.175);\n\end{scope}\n\path[pic actions] (0.2, -0.15) circle (0.05)\n\path[pic actions] (-0.2, -0.15) circle (0.05);\n},\n}\n
natoapp6c/s/balloon

\tikzset{\n  natoapp6c/s/balloon/.pic={\n  \path[pic actions] (0, 0.025) circle (0.175);\n\begin{scope}\n  \clip (0, 0.025) circle (0.175) \[reverseclip\];\n  \path[pic actions] (0, 0.025) circle (0.175);\n  \path[pic actions] (-0.08,0) rectangle (0.05,-0.2)--(0.05,0);\n},\n}\n
405
\end{scope},
\}

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/bar/.pic={
\path[fill=pgfstrokecolor] (-.3,-.1) rectangle (.3,.1);},
\}

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/base/.pic={
\path[pic actions] circle(.2);
\path[pic actions]
(-.2,0) -- (.2,0)
( 0,-.2) -- (0,.2)
(225:.2) -- (45:.2)
(135:.2) -- (-45:.2);
}\}
\}

\begin{tikzpicture}
\tikzset{%
\natoapp6c/s/bicycle equipped/.pic={\draw(0,0) circle(.1);},
\}

\begin{tikzpicture}
\tikzset{%
\natoapp6c/s/boat/.pic={
\path[pic actions]
(-0.2, -0.2) --
( 0.2, -0.2) --
( 0.35, 0.05) --
(-0.15, 0.05) --
(-0.075, 0.2) --
(-0.175, 0.2) --
(-0.25, 0.05) --
(-0.35, 0.05) --
cycle;},
\}

\end{tikzpicture}
natoapp6c/s/carrier

\tikzset{
  natoapp6c/s/carrier/.pic={
    \pic{natoapp6c/s/warfare vessel};
    \path[draw,fill=pgfstrokecolor]
    (-0.15, 0.05) --
    (-0.15, 0.2) --
    (-0.3, 0.2) --
    (-0.3, 0.05) -- cycle;},
}

define tikz styles for chemical biological radiological nuclear

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

define tikz styles for civilian military cooperation

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

define tikz styles for civilian police

\tikzset{
  natoapp6c/s/civilian police/.pic={
    \path[draw] (0.225, 0.2)
    to[in=270, out=270, looseness=3] (-0.225, 0.2)
    to [in=270, out=270, looseness=1.5] (0,0.2)
    to [in=270, out=270, looseness=1.5] (0.225, 0.2) -- cycle;},
}

define tikz styles for civilian telecommunications

\tikzset{
  natoapp6c/s/civilian telecommunications/.pic={
    \path[draw] (0.075, -0.2){[line join=bevel] -- (0, 0.1) -- (-0.075, -0.2)}
}

408
\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
\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
\path[pic actions]\
[[Stealth[inset=0pt,scale=0.5]]--[Stealth[inset=0pt,scale=0.5]]\}
\path\[pic actions\]\
[[Stealth[inset=0pt,scale=0.5]]--[Stealth[inset=0pt,scale=0.5]]\}
\path\[pic actions\]\
(-.2, 0) -- (.2, 0);\}
\begin{tikzpicture}[natoapp6c/s/intermodal/.pic={}
\path[fill=pgfstrokecolor] (-.2, .025) -- (-.2, -0.175) -- (0.2, -0.175) -- (0.2, 0.025) -- (0.2, 0.025) -- (-0.2, 0.025) -- (-0.2, -0.175) -- cycle;},
\end{tikzpicture}
\begin{tikzpicture}[natoapp6c/s/corps support/.pic={}
def
@toapp@corps@support{\natoapp@fac}
\expandafter\path[draw] \n@toapp@path;},
\end{tikzpicture}
\begin{tikzpicture}[natoapp6c/s/crime/.pic={}
\path[dashed] (-0.45,0.25)--(0.45,-0.25);},
\end{tikzpicture}
\begin{tikzpicture}[natoapp6c/s/decoy/.pic={}
\path[fill=pgfstrokecolor,draw,yshift=1.5] (0.2, 0) -- (0.4, 0.15) -- (0.4, -0.15) -- cycle
(-0.1, 0) -- (0.1, 0.15) -- (0.1, -0.15) -- cycle
(-0.4, 0) -- (-0.2, 0.15) -- (-0.2, -0.15) -- cycle;},
\end{tikzpicture}
\begin{tikzpicture}[natoapp6c/s/direct communications/.pic={}
\path[draw] (-.35,0) circle(.1) (.35,0) circle(.1);
\pic[fill=pgfstrokecolor]{natoapp6c/s/intermodal};
\end{tikzpicture}
natoapp6c/s/direction finding

\tikzset{
  natoapp6c/s/direction finding/.pic={
    \path[draw] (-.3,.2)--(0,.4)--(.3,.2) (0,.4)--(0,-.4);},
}

natoapp6c/s/diving

\tikzset{
  pics/natoapp6c/s/diving/.is choice,
  pics/natoapp6c/s/diving/none/.style={
    code={
      \path[pic actions] (0,0) circle(.1) (0,0) circle(.25);
      \begin{scope}
        \clip (0,0) circle(.25) [reverseclip];
        \path[pic actions] (-3,-.1) rectangle(.3,.1)
        (0,0) -- (-45:.4) -- (-135:.4) -- cycle;
      \end{scope}
    },
    pics/natoapp6c/s/diving/military/.style={
      code={
        \begin{scope}[even odd rule]
          \clip (0,0) circle(0.1)[reverseclip];
          \pic[fill=pgfstrokecolor]{natoapp6c/s/diving=none};
        \end{scope}
        \path[fill=pgfstrokecolor] (0,0) circle(0.08);
      },
    }
  },
  pics/natoapp6c/s/diving/.default=none,
}

natoapp6c/s/drilling

\tikzset{
  natoapp6c/s/drilling/.pic={
    \path[pic actions] (-0.1,-0.2) -- (0.1,-0.2) -- (0.2, 0.2) -- (-0.2, 0.2) -- cycle;},
}

natoapp6c/s/earthmover

\tikzset{
  natoapp6c/s/earthmover/.pic={
    \pic{natoapp6c/s/tank};
    \path[pic actions] (.3,
17619 .3)--(.175, .35)--(-.175, .35)--(-.3, .3)
17620 (0, .2)--(0, .35);
17621 },
17622 }

natoapp6c/s/electric power

\tikzset{
  natoapp6c/s/electric power/.pic=
  \path[\text{pic actions}]
  (-0.05, 0) .. controls(-0.06, 0.14) ..
  ( 0, 0.09) .. controls(0.03, 0.06) ..
  ( 0, 0.06) .. controls(-0.03, 0.06) ..
  ( 0, 0.09) .. controls(0.06, 0.14) ..
  (0.05, 0)
  ($(-55:0.125) + (0, 0.075)$) arc(-55:235:0.125) arc(415:360:0.05) --
  +(0, -0.08) arc(180:360:0.05035) --
  +(0, 0.08) arc(180:125:0.05) -- cycle;
},
17635 }

natoapp6c/s/electronic ranging

\tikzset{
  natoapp6c/s/electronic ranging/.pic=
  \path[draw] (135:.225) arc (135:315:.225)--cycle (0,0)--(225:-.225);},
17639 }

natoapp6c/s/engineer

\tikzset{
  natoapp6c/s/engineer/.pic={
  \path[draw]
  (.4,-.2)--(.4,.2)--(-.4,.2)--(-.4,-.2) (0,.2)--(0,-.2);},
17650 }

413
\tikzset{
natoapp6c/s/enhanced location reporting system/.pic={
  \path[draw] (0, -0.3) -- (0, 0.3) (-0.2, -.3) -- (0, 0.-.1) -- (0.2, -.3);},
}

\tikzset{
natoapp6c/s/environmental protection/.pic={
  \path[draw] (0, 0.2)
  -- (0.1, 0.05)
  -- (0.05, 0.05)
  -- (0.15, -0.05)
  -- (0.1, -0.05)
  -- (0.2, -0.15)
  -- (0.15, -0.15)
  -- (0.05, -0.15)
  -- (0.05, -0.2)
  -- (-0.05, -0.2)
  -- (-0.05, -0.15)
  -- (-0.2, -0.15)
  -- (-0.1, -0.05)
  -- (-0.15, -0.05)
  -- (-0.05, 0.05)
  -- (-0.05, 0.05)
  -- (-0.1, 0.05)
  -- cycle;},
}

\tikzset{
natoapp6c/s/explosion/.pic={
  \node [shape=rectangle, starburst, draw, minimum width=0.9cm, minimum height=0.9cm, starburst point height=0.25cm, starburst points=12] {};},
}

\tikzset{
natoapp6c/s/finance/.pic={
}}
\tikzset{
  natoapp6c/s/fishing_vessel/.pic={
    \path[\text{\textbackslash pic actions}] (-0.15, -0.2) -- (0.15, -0.2) -- (0.25, 0.025) -- (-0.05, 0.025) -- (-0.05, 0.125) -- (-0.2, 0.125) -- (-0.2, 0.025) -- (-0.25, 0.025) -- cycle
    (0.025, 0.025) -- +(45:0.2);},
}

\tikzset{
  natoapp6c/s/fire_protection/.pic={
    \path[\text{\textbackslash fill=p\textbackslash g\textbackslash stroke\textbackslash color}] (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;
  },
}

\tikzset{
  natoapp6c/s/fixed_and_rotary_wing/.pic={
    \path[\text{\textbackslash xscale=.45,yscale=.75,\textbackslash pic actions}] pic {natoapp6c/s/fixed_wing};
    \path[\text{\textbackslash yscale=.45,\textbackslash xscale=.7,rotate=90,\textbackslash pic actions}] pic {
      natoapp6c/s/rotary_wing};
  },
}

\tikzset{
  natoapp6c/s/fixed_wing

\tikzset{
  natoapp6c/s/fuel/.pic={
    path[draw] (0,0) -- (135:.3) -- (45:.3) -- cycle (0,0) -- (0,-.3);},
}
\tikzset{
  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,
}
\tikzset{
  natoapp6c/s/graffiti/.pic={
    path[pic actions]
    (0.05, 0.2) arc (90:270:0.05)
    arc (450:270:0.05)
    arc (90:270:0.05)
    arc (450:270:0.05)
    (-0.05, 0.2) arc (90:270:0.05)
    arc (450:270:0.05)
    arc (90:270:0.05)
    arc (450:270:0.05);},
}
\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};},
}
\begin{tikzpicture}
  \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,
  }
\end{tikzpicture}

\begin{tikzpicture}
  \tikzset{
    \natoapp6c/s/headquarters/.pic={
      \path[pic actions] (M.north west) -- ++(0,-.3) --
      ([shift=(-90:.3)]M.north east) -- (M.north east) -- cycle;},
  }
\end{tikzpicture}

\begin{tikzpicture}
  \tikzset{
    \natoapp6c/s/house/.pic={
      \path[pic actions]
      (-.125,-.175) rectangle (.125,.075)
      (.167,.075) -- (0,.225) -- (.167,.075) -- cycle;},
  }
\end{tikzpicture}
\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);
},
}
\natoapp6c/s/jam
\tikzset{
\natoapp6c/s/jam/.pic={
\path[draw]
(0.75, 0) to[out=90, in=90, looseness=2.25] (0.65, 0)
(0.65, 0) to[out=-90,in=-90,looseness=2.25] (0.55, 0)
(0.55, 0) to[out=90, in=90, looseness=2.25] (0.45, 0)
(0.45, 0) to[out=-90,in=-90,looseness=2.25] (0.35, 0)
(0.35, 0) to[out=90, in=90, looseness=2.25] (0.25, 0)
(0.25, 0) to[out=-90,in=-90,looseness=2.25] (0.15, 0)
(0.15, 0) to[out=90, in=90, looseness=2.25] (0.05, 0)
(0.05, 0) to[out=-90,in=-90,looseness=2.25] (-0.05, 0)
(-0.05, 0) to[out=90, in=90, looseness=2.25] (-0.15, 0)
(-0.15, 0) to[out=-90,in=-90,looseness=2.25] (-0.25, 0)
(-0.25, 0) to[out=90, in=90, looseness=2.25] (-0.35, 0)
(-0.35, 0) to[out=-90,in=-90,looseness=2.25] (-0.45, 0)
(-0.45, 0) to[out=90, in=90, looseness=2.25] (-0.55, 0)
(-0.55, 0) to[out=-90,in=-90,looseness=2.25] (-0.65, 0)
(-0.65, 0) to[out=90, in=90, looseness=2.25] (-0.75, 0)
(-0.75, 0);
},
}
\natoapp6c/s/jamming
\tikzset{
\natoapp6c/s/jamming/.pic={
\path(0,.4) pic {natoapp6c/s/jam} (0,.26) pic {natoapp6c/s/jam};
},
}\end{tikzpicture}
\begin{tikzpicture}
\def\landingMine{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\personnel{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\tank{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\mine{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\landMine{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\labour{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\killing{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\def\jetski{\begin{scope}[scale=0.5,rotate=45,transform shape]
\draw[fill=gray!50,rounded corners=2pt] (-0.5,0) -- (0.5,0) -- (0.5,-1) -- (-0.5,-1) -- cycle;
\end{scope}}
\end{tikzpicture}
natoapp6c/s/land missile
\tikzset{
natoapp6c/s/land missile/.pic={\pic{natoapp6c/s/missile launcher};},
}

natoapp6c/s/laser
\tikzset{
natoapp6c/s/laser/.pic={\path[draw,line join=round,line cap=round,pic actions]
( 0.1, -0.25) --
(-0.1, -0.225) --
( 0.1, -0.2) --
(-0.1, -0.175) --
( 0.1, -0.15) --
( 0, -0.1375) --
( 0, -0.0125) --
(-0.1, 0) --
( 0.1, 0.025) --
(-0.1, 0.05) --
( 0.1, 0.075) --
( 0, 0.0875) --
( 0, 0.25) --
( 0.1, 0.2) --
( 0, 0.25) --
(-0.1, 0.2);},
}

natoapp6c/s/launcher
\tikzset{
natoapp6c/s/launcher/.pic={\path[draw] (-.3,-.2) -- (.3,.2) -- (.3,-.2);},
}

natoapp6c/s/laundry
\tikzset{
natoapp6c/s/laundry/.pic={\path[draw] (0,-.3) -- (0,.1)
(0,.1) -- ++(150:.25)
(0,.1) -- ++(180:.2)
(0,.1) -- ++(210:.25);},
}

422
\texttt{natoapp6c/s/machine gun}
\begin{verbatim}
\tikzset{
  natoapp6c/s/machine gun/.pic={
    \pic[draw]{natoapp6c/s/rifle};
    \pic[draw]{natoapp6c/s/weapon=machine gun};},
}
\end{verbatim}

\texttt{natoapp6c/s/main gun}
\begin{verbatim}
\tikzset{
  natoapp6c/s/main gun/.pic={
    \path[pic actions] (M.north west) -- ++(.25,0) --
    ([shift=(0:.25)]M.south west) -- (M.south west) -- cycle;},
}
\end{verbatim}

\texttt{natoapp6c/s/maintenance}
\begin{verbatim}
\tikzset{
  natoapp6c/s/maintenance/.pic=
    \path[pic actions]
    (-0.075,-0.2)
    --(0.075,-0.2)
    --(.075,-.075)
    --(.2,-.075)
    --(.2,.075)
    --(.075,.075)
    --(-0.075,.2)
    --(-0.075,-.075)
    --(-.2,-.075)
    --(-.2,.075)
    --(-.075,.075)
    --(.38,.17) -- cycle;
  },
}
\end{verbatim}

\texttt{natoapp6c/s/medic}
\begin{verbatim}
\tikzset{
  natoapp6c/s/medic/.pic=
    \path[pic actions]
    (-0.075,-0.2)
    --(0.075,-0.2)
    --(.075,-.075)
    --(.2,-.075)
    --(.2,.075)
    --(.075,.075)
    --(-0.075,.2)
    --(-0.075,-.075)
    --(-.2,-.075)
    --(-.2,.075)
    --(-.075,.075)
    --(.38,.17) -- cycle;
  },
}
\end{verbatim}

423
\begin{tikzpicture}

\tikzset{
  natoapp6c/s/medical/.pic={
    \path[draw] (-1,0) -- (1,0) (0,-1) -- (0,1);
  },
}

\tikzset{
  natoapp6c/s/medical treatment/.pic={
    \path[draw] (0,0) pic {natoapp6c/s/medical}
    (\[xscale=.5,shift={(0,-.2)}\]M.west) -- (\[xscale=.5,shift={(0,.2)}\]M.west)
    (\[xscale=.5,shift={(0,-.2)}\]M.east) -- (\[xscale=.5,shift={(0,.2)}\]M.east);
  },
}

\tikzset{
  natoapp6c/s/mine/.pic={
    \path[fill=pgfstrokecolor,draw] (0,0) ellipse(.2 and .15)
    (0,0) -- ++(60:.3)
    (0,0) -- ++(90:.3)
    (0,0) -- ++(120:.3)
    (0,0) -- ++(240:.3)
    (0,0) -- ++(270:.3)
    (0,0) -- ++(300:.3);
  },
}

\tikzset{
  natoapp6c/s/mine clearing equipment/.pic={
    \path[pic actions]
    (0,0.2) -- (0,0) -- (0.35, -0.2) -- (-0.35, -0.2) -- (0,0);
  },
}
\end{tikzpicture}
natoapp6c/s/mobile advisor and support

\tikzset{%
natoapp6c/s/mobile advisor and support/.pic={
  \path[draw] (-.35,0) circle(.1) (.35,0) circle(.1);
  \path[draw,fill=pgfstrokecolor,pic actions]
    ( 0.15, 0.025) --
    (-0.25, 0.025) --
    (-0.25, -0.025) --
    ( 0.15, -0.025) --
    ( 0.15, -0.075) --
    ( 0.25, 0) --
    ( 0.15, 0.075) -- cycle;},
}

natoapp6c/s/moored

\tikzset{%
natoapp6c/s/moored/.pic={
  \path[draw] (0,.2) -- (0,-.05) (-.3,-.05) -- (.3,-.05);},
}

natoapp6c/s/mortar

\tikzset{%
natoapp6c/s/mortar/.pic={
  \path[draw] (0,-.15) circle(.05)
    (0,.2) -- ([shift=(225:.1)]0,.2) -- ([shift=(45:.1)]0,.2);},
}

natoapp6c/s/motorized

\tikzset{%
natoapp6c/s/motorized/.pic={
  \path[draw] (M.north) -- (M.south);},
}

picas/natoapp6c/s/motorised/.style={natoapp6c/s/motorized},
\begin{verbatim}
\texttt{natoapp6c/s/orbiter shuttle}
\begin{verbatim}
\tikzset{%
  natoapp6c/s/orbiter shuttle/.pic={
    \path[pic actions]
    (\$0, 0.3\)!0.35!(0.125, -0.15)\$) --
    (0.125, -0.15) -- (-0.125, -0.15) --
    (\$-0.125, -0.15\)!0.65!(0, 0.3)\$
    to[in=105, out=75] cycle
    (0, -0.20) -- (0, -0.15); },
}
\end{verbatim}
\end{verbatim}

\texttt{natoapp6c/s/ordnance}
\begin{verbatim}
\tikzset{%
  natoapp6c/s/ordnance/.pic={
    \path[\texttt{draw}] (0,0) ellipse(.2 and .15);
    \begin{scope}
      \clip (0,0) ellipse(.2 and .15) [\texttt{reverseclip}];
      \path[\texttt{draw}] (0,0) -- ++(50:.3)
      (0,0) -- ++(70:.3)
      (0,0) -- ++(110:.3)
      (0,0) -- ++(130:.3)
    \end{scope},
  }
}
\end{verbatim}

\texttt{natoapp6c/s/organisation}
\begin{verbatim}
\tikzset{%
  \texttt{pics/natoapp6c/s/organisation/.style={natoapp6c/s/group}},
}
\end{verbatim}

\texttt{natoapp6c/s/over snow}
\begin{verbatim}
\tikzset{%
  natoapp6c/s/over snow/.pic=
  \ifs@to@pp\below%
    \draw [[\texttt{shift=}((-15.,15))]]M.south west) to[in=180, out=270]
    ++(.15,-.15) -- (M.south east);
  \else%
    \draw [[\texttt{shift=}((.3,.1))]]M.west) to[in=180, out=-90]
    (\texttt{shift=}(.5,-.1))M.west) --
    (\texttt{shift=}(-.3,-.1))M.east);
  \fi},
}
\end{verbatim}
\end{verbatim}

429
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/poisoning/.pic={
\path[\tikzactions] (0, 0.055) circle (0.145)
(0.3, 0) -- (-0.3, -0.2)
(-0.3, 0) -- (0.3, -0.2);},
}
\end{tikzpicture}

\begin{tikzpicture}
\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);},
}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/printed media/.pic={
\path[\tikzactions] (0.2, 0) -- (-0.2, 0)
(0, 0.1) circle (0.085)
(0, -0.1) circle (0.085);},
}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/psychological/.pic={
\path[\tikzactions] (-.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);},
}
\end{tikzpicture}

431
\tikzset{natoapp6c/s/quarry/.pic={
    \path[draw] (-.2,-.2) -- (.18,.18) (.2,-.2) -- (-.18,.18)
    (25:.255) arc(25:65:.255)
    (115:.255) arc(115:155:.255);
    \%
    \%
    }},
\tikzset{natoapp6c/s/quartermaster/.pic={
    \path[draw] (-.4,.1) -- (.1,.1) (.25,.1) circle(.15)
    (-.3,.1) -- (-.3,-.15) (-.15,.1) -- (-.15,-.15)
    (-.3,-.08) -- (-.15,-.08);},
\tikzset{natoapp6c/s/radar/.pic={
    \path[draw] (-.2,.2) arc (150:300:.25) (-.24,.01) -- (0,.2) --
    (0,0) -- (.2,.2);},
\tikzset{natoapp6c/s/radio/.pic={
    \path[draw] (-.2,.2) -- (-.13,.25) -- (-.07,.2) -- (0,.25) --
    (.07,.2) -- (.13,.25) -- (.2,.2) (0,.25) -- (0,-.05) (0,-.15) circle(.1);},
\tikzset{natoapp6c/s/radio relay/.pic={
    \path[draw] (-.2,.25) -- (.2,.25) (0,.25) -- (0,-.05) (0,-.15) circle(.1);},
\end{tikzpicture}
\begin{itemize}
\item
\item
\item
\item
\end{itemize}
\end{document}
natoapp6c/s/rocket launcher/.style={
code={
\pic[\draw]{natoapp6c/s/rocket launcher=base};
\pic[\draw]{natoapp6c/s/weapon=anti tank};
},
pics/natoapp6c/s/rocket launcher/anti tank/.style={
code={
\pic[\draw]{natoapp6c/s/rocket launcher=base};
\pic[\draw]{natoapp6c/s/weapon=anti tank};
},
pics/natoapp6c/s/rocket launcher/single/.style={
code={
\pic[\draw]{natoapp6c/s/rocket launcher=base};
\pic[\draw]{natoapp6c/s/weapon=bottom};},
pics/natoapp6c/s/rocket launcher/multiple/.style={
code={
\pic[\draw]{natoapp6c/s/rocket launcher=single};
\pic[\draw]{natoapp6c/s/weapon=multi fire};},
pics/natoapp6c/s/rocket launcher/single head/.style={
code=\%,
\pic[\yshift=4,\draw]{natoapp6c/s/weapon=rifle};},
pics/natoapp6c/s/rocket launcher/multiple head/.style={
code=\%,
\pic[\yshift=-4,\draw]{natoapp6c/s/weapon=rifle};
\pic[\yshift=-6,\draw]{natoapp6c/s/weapon=rifle};},
pics/natoapp6c/s/rocket launcher/.default=single,
}

\tikzset{\%
\natoapp6c/s/rotary wing/.pic={
\path[pic actions]
(0.44, 0.15) -- (0.44, -0.15) -- (-0.44, 0.15) -- (-0.44, -0.15) --
\cycle;},
}
\tikzset{\%
\natoapp6c/s/runway/.pic=\%
\path[\draw] (-.3,-.15) -- (.3,-.15) (-.2,.2) -- (.2,.2);},
}
\tikzset{\%
\natoapp6c/s/sailing boat/.pic=\%
\path[\draw]
(-0.15, -0.2) --
}
\tikzset{\n\pic{natoapp6c/s/satellite=none};\n}\path[pic actions]\n(-0.075, 0.13) circle (0.07)\n++(-30:0.22) -- ++(0, -0.025) -- (-0.005, 0.13) -- cycle;\n}\n\path[pic actions]\n(0, 0.075) -- (0, 0.125)\n(0, 0.125) arc (270:340:0.25 and 0.1)\n(0, 0.125) arc (270:200:0.25 and 0.1);\n}
```latex
\begin{tikzpicture}
\end{tikzpicture}
```
\tikzset{%
\pics/natoapp6c/s/sea mine/.is choice,
\pics/natoapp6c/s/sea mine/top half/.style={
  code={
    \path[pic actions] (0,0) arc(0:35:.2 and .175) --
    (42:.34 and .3) -- (48:.34 and .3) --
    (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) --
    (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}
    \end{scope}
  }},
\pics/natoapp6c/s/sea mine/neutralized/.style=natoapp6c/s/sea mine/neutralised,
\pics/natoapp6c/s/sea mine/.default=full,
}
\tikzset{%
\natoapp6c/s/seabed installation/.pic={%
  \path[pic actions]
  (-0.25, -0.2) --
  (-0.25, 0.2) --
  (0.05, 0.025) --
  (0.05, -0.025) --
  (-0.125, -0.025) --
  (-0.125, 0.2) --
  (-0.25, 0.2) -- cycle;},
\natoapp6c/s/seabed installation/.pic={%
  \path[pic actions]
  (-0.25, -0.2) --
  (-0.25, 0.2) --
  (0.05, 0.025) --
  (0.05, -0.025) --
  (-0.125, -0.025) --
  (-0.125, 0.2) --
  (-0.25, 0.2) -- cycle;},
\natoapp6c/s/seabed installation/.pic={%
  \path[pic actions]
  (-0.25, -0.2) --
  (-0.25, 0.2) --
  (0.05, 0.025) --
  (0.05, -0.025) --
  (-0.125, -0.025) --
  (-0.125, 0.2) --
  (-0.25, 0.2) -- cycle;},
\natoapp6c/s/seabed installation/.pic={%
  \path[pic actions]
  (-0.25, -0.2) --
  (-0.25, 0.2) --
  (0.05, 0.025) --
  (0.05, -0.025) --
  (-0.125, -0.025) --
  (-0.125, 0.2) --
  (-0.25, 0.2) -- cycle;},
\natoapp6c/s/seabed installation/.pic={%
  \path[pic actions]
  (-0.25, -0.2) --
  (-0.25, 0.2) --
  (0.05, 0.025) --
  (0.05, -0.025) --
  (-0.125, -0.025) --
  (-0.125, 0.2) --
  (-0.25, 0.2) -- cycle;},

438
natoapp6c/s/search
\tikzset{
natoapp6c/s/search/.pic={
  \path[draw] (-.3,-.2)--(0,-.4)--(.3,-.2) (0,.4)--(0,-.4);},
}

natoapp6c/s/searching
\tikzset{
natoapp6c/s/searching/.pic={
  \path[pic actions]
  (-0.4, 0)
  arc (180:0:0.1)
  arc (180:360:0.1)
  arc (180:0:0.1)
  arc (180:270:0.1) -- +(0.1, 0)
  (0.3, -0.05) -- (0.4, -0.1) -- (0.3, -0.15);},
}

natoapp6c/s/semi trailer truck
\tikzset{
natoapp6c/s/semi trailer truck/.pic={
  \pic[scale=.75,xshift=-2,draw]{natoapp6c/s/utility vehicle};
  \path[pic actions] (0.21, -0.025) -- (0.35, -0.025)
  (0.35, 0.05) -- (0.35, -0.1);},
}

natoapp6c/s/sensor
\tikzset{
natoapp6c/s/sensor/.pic={
  \path[fill=pgfstrokecolor] (-.3,0) arc (270:360:.3) arc (180:270:.3) arc
  (90:180:.3) arc (0:90:.3);},
}

natoapp6c/s/ship
\tikzset{
natoapp6c/s/ship/.pic={
  \path[pic actions]
  (-0.2, -0.2) --
  ( 0.2, -0.2) --
  ( 0.35, 0.05) --
}
(0.15, 0.05) --
(0.15, 0.2) --
(-0.15, 0.2) --
(-0.15, 0.05) --
(-0.35, 0.05) --
cycle;
}

\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={
  \ifn@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
}
\tikzset{
\if\fi
}
\tikzset{
\if\fi
}
\tikzset{
\if\fi
}
\tikzset{
\if\fi
}
\tikzset{
\if\fi
}
\tikzset{\natoapp6c/s/submarine/.pic={\path[fill=pgfstrokecolor,pic actions]\(0.4, 0\) -- \(0.25, 0.15\) -- \(-0.25, 0.15\) -- \(-0.4, 0\) -- \(-0.25, -0.15\) -- \(0.25, -0.15\) -- cycle;},}

\tikzset{\natoapp6c/s/submersible/.pic={\path[pic actions]\(0, -0.05\) + (106.6:0.35 and 0.15) arc (106.6:433.4:0.35 and 0.15) |- (0, 0.2) -| cycle;},}

\tikzset{\natoapp6c/s/supply/.pic={\path[pic actions]\(M\).east-\(0,.25\) -- \(M\).west-\(0,.25\);},}

\tikzset{\natoapp6c/s/surface combatant/.pic={\pic {natoapp6c/s/warfare vessel};\path[draw,fill=pgfstrokecolor]\(0.12,0.05\) -- \(0.12,0.14\) -- \(0.06,0.14\) -- \(0.06,0.2\) -- \(0.24,0.2\) -- \(0.24,0.272\) -- \(0.06,0.272\) -- \(0.06,0.35\) -- \(-0.06,0.35\) -- \(-0.06,0.272\) --}}
\begin{tikzpicture}
\begin{scope}
\clip (-1,0) rectangle (1,1);
\end{scope}
\end{tikzpicture}
5.6.21 Some extra MIL-STD symbols

Extra NATO App6(c) symbol (from MIL-STD)

\begin{center}
\begin{tikzpicture}
\node (pic) {
\begin{footnotesize}
\begin{verbatim}
\tikzset{%
natoapp6c/s/prison/.pic={
  \path[pic actions] (-.3,-.3)rectangle(.3,.3)
  (-.23,-.30)--(-.23, .3)
  (.23,-.30)--( .23, .3)
  (-.08,-.30)--(-.08,-.2)
  (-.08,-.15) circle (.05)
  (-.08,-.1) --(-.08, .3)
  (.08,-.30)--( .08,-.2)
  (.08,-.15) circle (.05)
  (.08,-.1) --(.08, .3)
  (0,.15) circle(.07 and .1);
}
\end{verbatim}
\end{footnotesize}
};
\end{footnotesize}
\end{tikzpicture}
\end{center}

A list of all defined symbols
\begin{verbatim}
A list of all defined symbols
\end{verbatim}
type=vmedium,
type=vheavy,
type, above corps support, air assault with organic lift, air decoy, air assault, air defence, air strip, air traffic, airship, airborne, ammunition, amphibious, amphibious warfare ship, analysis, arrest, artillery, anti tank anti armour, antenna, armoured, armoured fighting vehicle, armoured personnel carrier, arctic, automobile, balloon, bar, base, bicycle equipped, boat, booby trap, bottomed, bridge=none, bridge=fixed, bridge=folding, bridge=hollow, 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=none,
diving=military,
diving,
drilling,
drill,
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,
flotation,
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,
jet ski,
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,
railway,
sailing boat,
satellite=none,
satellite=astronomical,
satellite=bio,
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}
453
\end{document}
\end{verbatim}
The category of an image is important later on when we generate the VASSAL module. Recognised categories are

- **counter** for counter images. Such an image will trigger the creation of a VASSAL game piece.
- **board** for board images. Images of this kind will result in VASSAL board (or Map) elements.
- **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.
- **chart** for charts. These images will be made VASSAL charts — i.e., pop-up windows which contains some graphics for the players reference.
- **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 **sub-category** is mainly used for counters. Above, we gave the sub-categories **Allied, Axis, and 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 **board, charts, oob, and front** has no or little effect.

One we have processed the file above to generate our PDF (Say **export.pdf**), then we can process it (and the CSV file) with a Python script to make our draft VASSAL module

```
export.py export.pdf export.csv -o Game.vmod -t Game -v 0.1 \
-d "My game" -r rules.pdf
```

This will generate the draft module **Game.vmod**. Note that we add the rules (**-r rules.pdf**) to the module so that the module is complete.
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 where the entry is used.
\n\n\n\nNATO App6(c) frame:
\n\nNATO App6(c) symbols:
\n\nNATO App6(c) keys: