The **wargame** package

Christian Holm Christensen

September 26, 2022

**Abstract**

This package provides tools to typesetting manuals, board, and counters for wargames using \LaTeX. Licensed under Creative Commons Attribution-ShareAlike International License, version 4.0.

**Contents**

1. **Introduction** 3

2. **Hex Boards** 4
   2.1 Placing hexes 4
   2.2 Styling hexes 5
   2.3 Hex coordinate system 5
   2.4 Terrains 7
      2.4.1 Styling terrains 7
   2.5 Ridges 10
      2.5.1 Styling ridges 10
   2.6 Labels 11
      2.6.1 Styling labels 12
   2.7 Towns 12
      2.7.1 Styling towns 13
   2.8 Extra graphics for hexes 13
   2.9 Rivers, borders, and roads 13
   2.10 Board clipping and frame 15

3. **Chits** 19
   3.1 Styling chits 19
   3.2 Defining preset chit types 21

4. **NATO App 6(c) symbols** 22
4.1 Faction and Command Selection ........................................... 23
4.2 Unit Size (echelon) .......................................................... 25
4.3 Unit type identification ...................................................... 25

5 Implementation ........................................................................ 25
5.1 The \texttt{wargame} package .................................................. 25
5.2 The \texttt{wargame.util} \texttt{TikZ} library .................................. 28
  5.2.1 Miscellaneous macros .................................................. 28
  5.2.2 Pictures in compound nodes ........................................... 29
  5.2.3 Nodes in compound nodes .............................................. 31
  5.2.4 Bounding boxes .......................................................... 32
  5.2.5 Other \texttt{Tikz} utilities ................................................. 34
  5.2.6 Random IDs .............................................................. 36
5.3 The \texttt{wgexport} class ..................................................... 37
5.4 The \texttt{wargame.hex} \texttt{TikZ} library .................................. 48
  5.4.1 Debugging ................................................................. 48
  5.4.2 Suppress terrain pictures ............................................... 48
  5.4.3 Hex coordinate system .................................................. 48
  5.4.4 Hexes ........................................................................ 54
  5.4.5 Terrain ...................................................................... 59
  5.4.6 Ridges ....................................................................... 282
  5.4.7 Towns ..................................................................... 284
  5.4.8 Labels ....................................................................... 285
  5.4.9 Extra graphics ........................................................... 288
  5.4.10 Some macros ............................................................ 289
  5.4.11 Edges, borders, roads, rivers, and so on ......................... 289
  5.4.12 Other paths .............................................................. 291
  5.4.13 Move, attacks, retreats from hex to hex ......................... 292
  5.4.14 Board clipping and frame ............................................ 295
5.5 The \texttt{wargame.chit} \texttt{TikZ} library ................................. 300
  5.5.1 Debugging ................................................................. 300
  5.5.2 The \texttt{chit} key namespace .......................................... 300
  5.5.3 The \texttt{chit} styles ....................................................... 301
  5.5.4 The \texttt{chit} shape ....................................................... 302
  5.5.5 The \texttt{chit} wrapper macro .......................................... 307
  5.5.6 Predefined \texttt{chit} element pictures ............................... 308
  5.5.7 Other pictures ........................................................... 310
A Generate draft VASSAL module
A.1 Example

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.
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\langle key-value-pairs\rangle\langle\location\rangle\langle\name\rangle
\end{verbatim}

The \langle key-value-pairs\rangle specify the hex. Valid options are

- \texttt{terrain}={terrain-keys} specifies the terrain of the hex. More on in this in Section 2.4.
- \texttt{ridges}={ridges-keys} specifies where ridges are drawn in the hex. Section 2.5.
- \texttt{label}={label-keys} specifies the how to output the hex label, if any. This is expanded upon in Section 2.6.
- \texttt{town}={town-keys} specifies that a town (or similar) is present in the hex. The various keys are described in Section 2.7.
- \texttt{extra}={extra-keyx} and \texttt{extra clipped}={extra-keyx} allows the user to put custom graphics in the hexes. See also Section 2.8 for more.

any style key defined for \texttt{TikZ} pictures.

The \langle\location\rangle argument specifies the coordinates, in the hex coordinate system where to put the hex. More about the coordinate system is given in Section 2.3. Note, the numbers starts from the lower–left corner.

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. Town, if any
7. 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.
2.2 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 \hexes 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}

2.3 Hex coordinate system

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

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

where \(⟨vertex⟩\) and \(⟨edge⟩\) 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 shorted to \textit{r}, \textit{c}, \textit{v}, and \textit{e}, respectively. Possible vertexes and edges are listed in Table 1.

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

\textbf{Important}: When the horizontal distance to the centre of a hex becomes less than \(- \cos 60°\) or larger than \(b - \cos 60°\) we effectively have a new hex column, and the coordinates are shifted upward or downward for smaller or larger numbers. Figure ?? illustrates this. This can make it a little hard to specify coordinates relative to a hex centre.
<table>
<thead>
<tr>
<th>vertex=</th>
<th>Angle</th>
<th>edge=</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>east</td>
<td>E</td>
<td>north east</td>
<td>NE</td>
</tr>
<tr>
<td>north east</td>
<td>NE</td>
<td>north</td>
<td>N</td>
</tr>
<tr>
<td>north west</td>
<td>NW</td>
<td>north west</td>
<td>NW</td>
</tr>
<tr>
<td>west</td>
<td>W</td>
<td>south west</td>
<td>SW</td>
</tr>
<tr>
<td>south west</td>
<td>SW</td>
<td>south</td>
<td>S</td>
</tr>
<tr>
<td>south east</td>
<td>SE</td>
<td>south east</td>
<td>SE</td>
</tr>
</tbody>
</table>

Table 1: Vertex and edge positions

Hexes and lines drawn with

```latex
\hex(0,0)\hex(0,1)\hex(1,0)\hex(1,1)
\draw[blue!50!black] (hex cs:r=0,c=0) --
  (hex cs:r=1,c=1);
\draw[red!50!black] (hex cs:r=0,c=0,vertex=E) --
  (hex cs:r=1,c=1,edge=NW);
\fill[lightgray](hex cs:.3,.3) circle(0.1);
\fill[lightgray](hex cs:1.3,.3) circle(0.1);
\fill[lightgray](hex cs:0.3,1.3) circle(0.1);
\fill[lightgray](hex cs:1.3,1.3) circle(0.1);
```

Figure 2: Hex coordinate system
Alternatively one may use vertex or edge specifications together with a relative offset in those directions.

2.4 Terrains

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

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

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

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

Users can define TikZ pictures that specify clipping paths as needed. For example, one could add clipping to the terrain to ensure that other graphics in the hex stands out.

2.4.1 Styling terrains

Terrains use the key `tikz/hex/terrain` to render the terrains. This is mainly useful for terrains made from TikZ pictures.
<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="image=wargame.beach" alt="Clear" /></td>
<td>Clear</td>
<td></td>
<td><img src="image=wargame.beach" alt="Beach" /></td>
<td>Beach</td>
<td>{image=wargame.beach}</td>
</tr>
<tr>
<td><img src="image=wargame.light_woods" alt="Light woods" /></td>
<td>Light woods</td>
<td>{image=wargame.light_woods}</td>
<td><img src="image=wargame.woods" alt="Woods" /></td>
<td>Woods</td>
<td>{image=wargame.woods}</td>
</tr>
<tr>
<td><img src="image=wargame.rough" alt="Rough" /></td>
<td>Rough</td>
<td>{image=wargame.rough}</td>
<td><img src="image=wargame.swamp" alt="Swamp" /></td>
<td>Swamp</td>
<td>{image=wargame.swamp}</td>
</tr>
<tr>
<td><img src="image=wargame.mountains" alt="Mountains" /></td>
<td>Mountains</td>
<td>{image=wargame.mountains}</td>
<td><img src="image=wargame.sea" alt="Sea" /></td>
<td>Sea</td>
<td>{image=wargame.sea}</td>
</tr>
</tbody>
</table>

Table 2: Terrains specified via tile images

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

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>![Hexagon symbol with sextant=NE]</td>
<td>{hex/sextant=NE}</td>
<td>![Hexagon symbol with sextant=NE, large sextant]</td>
<td>{hex/large sextant=NE, hex/large sextant=N}</td>
</tr>
<tr>
<td>![Hexagon symbol with sextant=N]</td>
<td>{hex/sextant=N}</td>
<td>![Hexagon symbol with sextant=N, NW, SW]</td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=NW, hex/sextant=SW}</td>
</tr>
<tr>
<td>![Hexagon symbol with sextant=NW]</td>
<td>{hex/sextant=NW}</td>
<td>![Hexagon symbol with sextant=NE, NW, SW, S, SE]</td>
<td>{hex/sextant=NE, hex/sextant=N, hex/sextant=NW, hex/sextant=SW, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
<tr>
<td>![Hexagon symbol with sextant=SW]</td>
<td>{hex/sextant=SW}</td>
<td>![Hexagon symbol with sextant=NE, S, SE, C]</td>
<td>{hex/sextant=NE, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
<tr>
<td>![Hexagon symbol with sextant=S]</td>
<td>{hex/sextant=S}</td>
<td>![Hexagon symbol with sextant=NE, S, SE, C]</td>
<td>{hex/sextant=NE, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
<tr>
<td>![Hexagon symbol with sextant=SE]</td>
<td>{hex/sextant=SE}</td>
<td>![Hexagon symbol with sextant=NE, S, SE, C]</td>
<td>{hex/sextant=NE, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
<tr>
<td>![Hexagon symbol with sextant=C]</td>
<td>{hex/sextant=C}</td>
<td>![Hexagon symbol with sextant=NE, S, SE, C]</td>
<td>{hex/sextant=NE, hex/sextant=S, hex/sextant=SE, hex/sextant=C}</td>
</tr>
</tbody>
</table>

Table 4: Terrain clipped via clip sub-key
2.5 Ridges

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

Table 5 shows some examples.

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

Table 5: Ridges

2.5.1 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.
2.6 Labels

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

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

Table 6: Labels

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

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

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

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

\texttt{[(/)]options} at the start of the option (but inside braces \{\ldots\}) can be used to give additional style options.
2.6.1 Styling labels

All labels use the style \texttt{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 \texttt{tikz/every label} to those options.

2.7 Towns

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>\texttt{town=}</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Hexagon with a black dot" /></td>
<td>{\texttt{pic=hex/town/town}}</td>
</tr>
<tr>
<td><img src="image2" alt="Hexagon with a black dot and a white dot" /></td>
<td>{\texttt{pic=hex/town/city}}</td>
</tr>
<tr>
<td><img src="image3" alt="Hexagon with a red dot" /></td>
<td>{\texttt{fill=red}}</td>
</tr>
<tr>
<td>![Hexagon with a red dot and the name Paris}(image4)</td>
<td>{\texttt{red,name=Paris}}</td>
</tr>
<tr>
<td><img src="image5" alt="Hexagon with a black dot and \texttt{place={(0.2,0.2)}}" /></td>
<td>{\texttt{place={(0.2,0.2)}}}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>\texttt{town=}</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Hexagon with a black dot and a white dot" /></td>
<td>Copenhagen {\texttt{name=Copenhagen}}</td>
</tr>
<tr>
<td><img src="image2" alt="Hexagon with a black dot and a white dot" /></td>
<td>London {\texttt{name=London}}</td>
</tr>
<tr>
<td><img src="image3" alt="Hexagon with a red dot and the name Paris" /></td>
<td>Paris {\texttt{red,name=Paris}}</td>
</tr>
<tr>
<td><img src="image5" alt="Hexagon with a black dot and \texttt{place={(0.2,0.2)}}" /></td>
<td>Berlin {\texttt{above=0.8,name=Berlin}}</td>
</tr>
<tr>
<td><img src="image1" alt="Hexagon with a black dot and a white dot" /></td>
<td>Amsterdam {\texttt{font=\texttt{\textbar expand\textitshape,name=Amsterdam}}}</td>
</tr>
</tbody>
</table>

Table 7: Towns

*When specifying macros as key values in the options, for example the value \texttt{\textbar rmfamily} for the key \texttt{font} above, we have to put a \texttt{\textbar 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 \texttt{town} key are

- \texttt{pic=(town-pic)} The name of a TikZ picture. Currently defined are \texttt{hex/town/town} and \texttt{hex/town/city}. Users can provide alternate definitions or new types by defining TikZ pictures.
- \texttt{place=(coordinates)} Location of label within the hex. The anchor point of the text will be placed at this point.
- \texttt{name=(name)} Name of town
2.7.1 Styling towns

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

2.8 Extra graphics for hexes

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

\[(\langle options\rangle)(\langle placement\rangle)(\langle picture\rangle)\]

Both \texttt{\langle options\rangle} and \texttt{\langle placement\rangle} are optional, and specifies keys to draw \texttt{\langle picture\rangle} with and the relative location in the hex, respectively. The required argument \texttt{\langle picture\rangle} must name a TikZ picture, for example \texttt{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.

2.9 Rivers, borders, and roads

Rivers and borders follow the hex sides and are added to the current TikZ picture using \texttt{\river} and \texttt{\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><code>{setup/german,hex/fortress}</code>†</td>
<td></td>
</tr>
<tr>
<td><code>[[{line width=2pt,brown}] fortress 2]</code>‡</td>
<td></td>
</tr>
<tr>
<td><code>{[shift={(0.2,0.2)}]setup/sovjet}</code>†</td>
<td></td>
</tr>
<tr>
<td><code>[[{shift={(0.2,0.2)},scale=.5,color=gray}] setup/german]</code>‡</td>
<td></td>
</tr>
<tr>
<td><img src="image4" 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="image5" alt="Foo Large Clipped" /></td>
<td>foo/large</td>
</tr>
</tbody>
</table>

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

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

Roads and railroads typically go from hex–center to hex–center, and are added using the macro \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 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 therefor defines the macro \boardclip which clips the graphics according to the defined hexes.

\boardclip((lower-left))(⟨upper-right⟩){⟨options⟩}

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

This is particularly useful together with the \boardframe macro. This macro will put a frame around the board, optionally with a margin.
Figure 5: Adding rivers, boarders, and roads

Figure 6: Clipping for a manual using a TikZ \texttt{draw[clip]} command.
\begin{scope}
  \boardclip(c=0,r=0)(c=3,r=3)\
  \testmap
\end{scope}

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

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

where \texttt{⟨lower-left⟩} and \texttt{⟨upper-right⟩} are as for \texttt{\boardclip}. The \texttt{⟨margin⟩} must be a number, and specifies an optional margin around the hexes, The argument \texttt{⟨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 \texttt{⟨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.
Figure 8: Combining a frame and clipping
3 Chits

Chits, or playing counters\(^1\), 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.

![Example of a chit](image)

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\LaTeX\,pictures. This allows for a great deal of flexibility in generating chits. For example, above we use the \texttt{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 × 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 mm × 12 mm — or roughly 1/2" × 1/2", which is a fairly good size for most games.

3.1 Styling chits

Typical Ti\LaTeX\ options can be passed to the \texttt{\chit} macro. For example, if you want to draw the chit with a red foreground, simply pass \texttt{draw=red} in the \texttt{[optional]} arguments to \texttt{\chits}. Individual parts of the hexes can be

\(^1\)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.
Figure 10: An example of a full-frame chit.

Figure 11: Example of chits fit within hexes.

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\(^2\).

The styles used by the *left*, *right*, *setup*, *factors*, and *symbol* elements are tikz/chit/left, tikz/chit/right, tikz/setup, tikz/factors, and 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 *scale*, move the factors to the side by changing *shift*, and so on.

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

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

\(^2\)The colour pgfstrokecolor is not modified by that.
3.2 Defining preset chit types

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

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

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

\begin{verbatim}
\begin{chit}
  2–4
\end{chit}
\begin{chit}[identifier=Mine,2 factors={2,4}]
  2–4
\end{chit}
\end{verbatim}

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 fig:natoapp:usage.

![Figure 12: Template for NATO symbols](image)

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

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

- **faction** Selects the faction used for the symbol. See also Section 4.1.
- **command** Selects the command used for the symbol. See also Section 4.1.
- **main** Specifies the main symbol(s). This can be a comma separated list of specifiers (delimited by braces \{first,second,...\}), and each symbol can be proceeded by an optional argument to shift, scale, rotate, etc., the individual symbols.
- **left**, **right**, **top**, **bottom**, **below** 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

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

![Figure 13: Main keys of \text{natoapp}. The bottom and right hand bars indicate one unit of length.](image)

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

- **echelon** 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.
frame=(keys) Extra keys for frame.

4.1 Faction and Command Selection

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

<table>
<thead>
<tr>
<th>(command)</th>
<th>friendly</th>
<th>hostile</th>
<th>neutral</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>air</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>land</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>equipment</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>installation</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>sea surface</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>sub surface</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>space</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
<tr>
<td>activity</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

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

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

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

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

Table 10 illustrates this.
Figure 14: Illustration of using the special value `faction` for the `frame` key

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

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@picture}
\ProcessOptions\relax
```

Finally, the used TikZ 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><img src="image1.png" alt="Image" /></td>
<td>Team</td>
<td>3–5</td>
<td>None</td>
<td>Corporal or Sergeant</td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td>Squad</td>
<td>5–10</td>
<td>1–2 teams</td>
<td>Sergeant</td>
</tr>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td>Section</td>
<td>7–13</td>
<td>2–3 teams</td>
<td>Sergeant</td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td>Platoon</td>
<td>25–40</td>
<td>Several squads/sections</td>
<td>Second Lieutenant</td>
</tr>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td>Company</td>
<td>60–250</td>
<td>Several platoons</td>
<td>Captain</td>
</tr>
<tr>
<td><img src="image6.png" alt="Image" /></td>
<td>Battalion</td>
<td>300–1,000</td>
<td>2–6 companies</td>
<td>Lieutenant colonel</td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td>Regiment</td>
<td>500–2,000</td>
<td>3–7 battalions</td>
<td>Colonel</td>
</tr>
<tr>
<td><img src="image8.png" alt="Image" /></td>
<td>Brigade</td>
<td>2,000–5,000</td>
<td>Several battalions</td>
<td>Colonel</td>
</tr>
<tr>
<td><img src="image9.png" alt="Image" /></td>
<td>Division</td>
<td>10,000–20,000</td>
<td>Several brigades/regiments</td>
<td>Major General</td>
</tr>
<tr>
<td><img src="image10.png" alt="Image" /></td>
<td>Corps</td>
<td>30,000–60,000</td>
<td>Several divisions</td>
<td>Lieutenant General</td>
</tr>
<tr>
<td><img src="image11.png" alt="Image" /></td>
<td>Army</td>
<td>100,000</td>
<td>Several corps (5–10 divisions)</td>
<td>General</td>
</tr>
<tr>
<td><img src="image12.png" alt="Image" /></td>
<td>Army Group</td>
<td>120,000–500,000</td>
<td>Several armies</td>
<td>Field Marshal</td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td>Theatre</td>
<td>250,000+</td>
<td>Several army groups</td>
<td>Field Marshal</td>
</tr>
<tr>
<td><img src="image14.png" alt="Image" /></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 AA</td>
</tr>
<tr>
<td></td>
<td>Air defence ADA</td>
</tr>
<tr>
<td></td>
<td>Airborne AB</td>
</tr>
<tr>
<td></td>
<td>Amphibious AM</td>
</tr>
<tr>
<td></td>
<td>Anti tank/armoured AT</td>
</tr>
<tr>
<td></td>
<td>Armoured AR</td>
</tr>
<tr>
<td></td>
<td>Chemical biological radiological nuclear CB</td>
</tr>
<tr>
<td></td>
<td>Combined arms CAR</td>
</tr>
<tr>
<td></td>
<td>Engineer ENG</td>
</tr>
<tr>
<td></td>
<td>Field artillery FA</td>
</tr>
<tr>
<td></td>
<td>Infantry IN</td>
</tr>
<tr>
<td></td>
<td>Mechanised infantry M</td>
</tr>
<tr>
<td></td>
<td>Mountaineer MTN</td>
</tr>
<tr>
<td></td>
<td>Naval N</td>
</tr>
<tr>
<td></td>
<td>Reconnaissance REC</td>
</tr>
<tr>
<td></td>
<td>Special Operations Forces SOF</td>
</tr>
</tbody>
</table>

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

Table 12: Some abbreviations of unit type identifications
5.2 The wargame.util TikZ library

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

5.2.1 Miscellaneous macros

\texttt{\textbackslash wg@dbg}

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

\begin{verbatim}
13 \newcount\wargamedbglvl \wargamedbglvl=0
14 \def\wg@dbg#1#2{\ifnum#1>\wargamedbglvl\relax\else\message{^^J#2}\fi}
\end{verbatim}

\texttt{\textbackslash wg@addto@macro}

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

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

\begin{verbatim}
16 \long\def\wg@addto@macro#1#2{\begingroup\toks@\expandafter\expandafter\expandafter{\expandafter#1#2}\xdef#1{\the\toks@}\endgroup}
\end{verbatim}

\texttt{\textbackslash wg@sub@nchor}

Get anchor from sub node. We cannot use \texttt{\textbackslash pgf@pointanchor} since that returns the anchor coordinates in the global coordinate system.

\begin{verbatim}
21 \def\wg@sub@nchor#1#2{\wg@dbg{3}{^^JGet '#2' in '#1'}\ifundefined{pgf@sh@ns@#1}{\pgf@x=0cm\pgf@y=0cm}\else\csname pgf@sh@ma@#1\endcsname\csname pgf@sh@np@#1\endcsname\pgf@sh@reanchor{\csname pgf@sh@ns@#1\endcsname}{#2}\xdef\wg@tmpa{\the\pgf@x,\the\pgf@y}\fi}
\end{verbatim}

Scratch dimensions

\begin{verbatim}
31 \newdimen\wg@tmpa
\end{verbatim}
Macro to easily restore a saved path

\def\settosave#1{\pgfsyssoftpath@setcurrentpath{#1}}

5.2.2 Pictures in compound nodes

The macro \texttt{\wl@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

\wl@pic\{⟨options⟩\}\{picture\}\ast\wl@pic{⟨prefix⟩}{⟨position⟩}{⟨options⟩}

Note the \texttt{\ast\wl@pic} at the end of the call to swallow up \texttt{(picture)}. Typically this macro is used as

\edef\args\{⟨something⟩\} \expandafter\wl@pic\args\ast\wl@pic{⟨prefix⟩}{⟨position⟩}{⟨options⟩}

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

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

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

This macro then forwards to \texttt{\wl@@pic} to gobble up \texttt{(picture)}.

1. User options
2. Arguments

\def\wl@@pic[#1][#2]\ast\wl@pic{%\@ifnextchar[^\wl@dbg{2}{Options: '#1', picture: '#2'}]%\wl@debug{2}{Options: '#1', picture: '#2'}%\wl@@pic[#1][#2]%}

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}{^^JWG Pic:
  ^^^J User options: #1
  ^^^J Picture: #2
  ^^^J Prefix: #3
  ^^^J Coordinates: #4
  ^^^J Fixed options: #5}\
  \pic[#5,#1] at (#4) {#3#2};\
  \if\wg@s@ve\
  \pgf@relevantforpicturesizetrue\
  \begin{getbbl}\
  \pic[draw=none,fill=none,transform shape] at (#4) {#3#2};\
  \end{getbbl}\
  \wg@dbg{5}{Clipping to local bounding box}\n  \clip (L.south west) rectangle (L.north east);\
  \pgf@relevantforpicturesizefalse \global\wg@s@vefalse\
  \fi\
  \fi\
  \wg@dbg{3}{End of WG Pic}\
}\
\wg@pic@all

This macro sets all pictures in a list.

1. List
2. Prefix
3. Position
4. Styles

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

The macro \texttt{\texttt{\textbackslash wg@node}} will render a node. This can be used by the \texttt{natoapp6cs}, \texttt{chit}, and \texttt{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

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

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

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

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

\begin{verbatim}
\def\wg@node{%\@ifnextchar[{$\wg@@node$}{\wg@@node\[\]}}% 
\def\wg@@node[#1]#2\@endwg@node{% 
\wg@dbg{2}{Options: '⟨user option⟩', body: '⟨body⟩'}% 
\wg@@@node{#1}{#2}\} 
\end{verbatim}

This macro then forwards to \texttt{\texttt{\textbackslash wg@node}} to gobble up \texttt{⟨body⟩}.
This macro sets all pictures in a list.

1. List
2. Prefix
3. Position
4. Styles

5.2.4 Bounding boxes

Bounding box dimensions

Enable or disable bounding box tracking
Reset the bounding box tracking dimensions

\def\wg@resetbb{%
\global\wg@bb@minx=16000pt\relax%
\global\wg@bb@miny=16000pt\relax%
\global\wg@bb@maxx=-16000pt\relax%
\global\wg@bb@maxy=-16000pt\relax%
}\}

Save PGF's bounding box algorithm

\let\old@pgf@protocolsize\pgf@protocolsizes
\wg@protocolsizes

Our bounding box algorithm

\def\wg@protocolsizes#1#2{%
\old@pgf@protocolsize{#1}{#2}
\ifwg@notrelevantforpathsize\else%
\ifdim#1<\wg@bb@minx\global\wg@bb@minx#1\fi%
\ifdim#1>\wg@bb@maxx\global\wg@bb@maxx#1\fi%
\ifdim#2<\wg@bb@miny\global\wg@bb@miny#2\fi%
\ifdim#2>\wg@bb@maxy\global\wg@bb@maxy#2\fi%
\fi
}\}

getbbl (env.) Environment that tracks the local bounding box

\newenvironment{getbbl}{%\wg@resetbb%
\wg@notrelevantforpathsizefalse%
\global\let\pgf@protocolsizes\wg@protocolsizes}{%\gdef\pgf@sh@ns@L{rectangle}
\gdef\pgf@sh@np@L{\def\southwest{\pgfqpoint{\the\wg@bb@minx}{\the\wg@bb@miny}}\def\northeast{\pgfqpoint{\the\wg@bb@maxx}{\the\wg@bb@maxy}}%}
\gdef\pgf@sh@nt@L{\pgfpictureid}
\global\let\pgf@protocolsizes\old@pgf@protocolsize
\}

getbb (env.) Environment to track global bounding box

\newenvironment{getbb}{%
5.2.5 Other Tikz utilities

**tikz/reverseclip**

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

```
\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)}]
```

**tikz/clip even odd rule**

A reverse clipping path

```
\tikzset{
    clip even odd rule/.code={\pgfseteorule}, % Credit to Andrew Stacey
}
```

**tikz/invclip**

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

```
\tikzset{
    invclip/.style={
        clip,insert path=
        [clip even odd rule]{
            [reset cm](-\maxdimen,-\maxdimen)rectangle(\maxdimen,\maxdimen)
        }
    }
```
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.

```latex
\newif\ifwg@s@ve
\tikzset{
  save clip/.is choice,
  save clip/true/.code={\global\wg@s@vetrue},
  save clip/false/.code={\global\wg@s@vefalse},
  save clip/.default={true},
  save clip/.initial={false},
}
```

scale line widths

Scales any line width specified in the node options.

Use like

```latex
\tikzset{
  some/.style={
    scale line widths,
    line width=1pt
  }
}
```

Note that the order is important.
206 \relative line width/.style={%
207 \pgfmathsetmacro{\wg@lv}{#1\times}\pgflinewidth{\wg@lw pt}}
208 \tikz@addoption{\pgfsetlinewidth{\wg@lw pt}}
209 \pgfmathsetlength\pgflinewidth{\wg@lw pt}}
210 }

\subpic actions

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

213 \tikzset{
214 \subpic actions/.code={%
215 \tikz@picmode%
216 \edef\opts{%
217 \iftikz@mode@draw draw,\else draw=none,\fi
218 \iftikz@mode@fill fill\else fill=none\fi}
219 \wg@dbg{5}{^^JSub Mode: \meaning\tikz@picmode \meaning\opts}
220 \pgfkeysalsofrom\opts
221 \pgfkeysalsofrom\opts
222 }

\wg/debug show

Show debugging information

223 \tikzset{
224 \wg/debug show/.code={%
225 \extractcolorspec{pgfstrokecolor}\wg@tmp@fg
226 \def\wg@tmp@bg{none}
227 \@ifundefinedcolor{pgffillcolor}{}{
228 \extractcolorspec{pgffillcolor}\wg@tmp@bg
229 \begingroup
230 \tikz@mode
231 \wargamedbglvl=#1
232 \wg@dbg{3}{Drawing with w/stroke ‘\wg@tmp@fg’
233 \iftikz@mode@draw\else not\space\fi drawing
234 and fill ‘\wg@tmp@bg’ \iftikz@mode@fill\else
235 not\space\fi filling}
236 \endgroup
237 }
238 }

5.2.6 Random IDs

239 \def\wg@r@ndom@id{
240 \def\wg@uuid{}
241 \foreach \i in {1,...,8}{%
242 \pgfmathparse{Hex(random(0,15))}
5.3 The \texttt{wgexport} class

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

Class identification and load \texttt{wargame} package

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

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

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

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

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

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

Preceed all images (\texttt{tikzpicture}) with this command

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

\begin{verbatim}
\makewithout{\mk@i \@lbchar name":"End of list", "category": "<<eol>>",
\"subcategory": "" \@rbchar }
\makewithout{\mk@i \@lbchar name":"End of list", "category": "<<eol>>",
\"subcategory": "" \@rbchar }
\immediate\closeout\mk@out
\]}
\end{verbatim}
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.

```latex
\def\info{\def\info#1#2#3#4{\chit@dbg{2}{Making image '#2' of type '#3'/'#4' on page \thepage}\mk@w{ \@lbchar}\mk@w{ \space "name": '#2',}\mk@w{ \space "category": '#3',}\mk@w{ \space "subcategory": '#4', }\mk@w{ \space "number": \thepage #1}\let\oldmk@i\mk@i\if\def\info#1#2#3#4{\mk@w{ \space \oldmk@i\mk@i}\let\mk@i\oldmk@i\fi}}\mk@w{ \space \oldmk@i\mk@i}}\ifx#1,\relax\edef\mk@i{\mk@i{\space space}{space}}\fi}}
\def\end\info{\def\mk@i{\mk@i{\space \@rbchar}}}\def\end\info{\def\mk@i{\mk@i{\space \@rbchar}}}
```
Make separate images for each counter (single sided).

\newcommand\chitimages[2][]{%
   \begingroup%
   \let\chit@report\do@chit@report%
   \let\natoapp@report\do@natoapp@report%
   \chit@dbg{2}{chits to make images of '#2'}%
   \foreach[count=\ti from 0] \t/x in #2{%
      \chit@dbg{5}{\vspace{1em}Subcategory: \t (default \#1)}%
      \ifx\t\empty\else Ignore empty rows\fi%
      \foreach\ti in \t{x}{%
         \chit@dbg{2}{Next chit \t with possible multiplicity \m}%
         \ifx\m\empty\def\m{1}\fi% If not multiplicity defined
         \ifx\m\@empty\chit@dbg{3}{Ignoring blank chit: \t}%
         \else%
         \chit@dbg{2}{Next chit \t multiplicity \m}%
         \chit@dbg{2}{End of inner loop}%
      \fi%
      }%
   \ifx\t\empty\else Ignore empty cells\fi%
   \foreach\t/x in \t{x}{%
      \chit@dbg{5}{\vspace{1em}Subcategory: \t (default \#1)}%
      \ifx\t\empty\def\t{#1}\fi% Take sub-category or default
      \foreach \t in \t{x}{%
         \chit@dbg{2}{Next chit \u with possible multiplicity \m}%
         \ifx\m\empty\def\m{1}\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}%
         \chit@dbg{2}{End of inner loop}%
      }%
   \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{%
      \chit@dbg{5}{\vspace{1em}Subcategory: \t (default \#1)}%
      \ifx\t\empty\else Ignore empty rows\fi%
      \chit@dbg{2}{End of inner loop}%
   \chit@dbg{2}{End of outer loop}%
   \endgroup%
\foreach \u/\m in \t{% 
\ifx\u/\empty\else% Ignore empty cells
\chit@dbg{2}{Next chit \"u\' with possible multiplicity \"m\'}%
\ifx\m/\empty\def\m{1}\fi% If not multiplicity defined
\ifx\u/\m\def\m{1}\fi% If the same as unit
\chit@dbg{2}{Next chit \"u\' multiplicity \"m\'}%
%% Flipped chit
\edef\s{\u/\space flipped}%
%% 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%
\info*{\s}{counter}{\x}%
\begin{tikzpicture}%
\chit[\s/\=\ti]%
\end{tikzpicture}%
\end@info%
%% \foreach \n in {1,...,\m}{% Make a number of copies
%% \ifx\u/\chit@blank%
%% \chit@dbg{3}{Ignoring blank chit:\u}%
%% \else%
%% \info{\u}{counter}{\#2}
%% \begin{tikzpicture}
%% \chit[\u/\=\ti](\c/\r)%
%% \end{tikzpicture}
%% \fi%
%% }
\fi%
\endgroup%
}

Special for boards, we have the environment \texttt{boardimage}. Like \texttt{info} we must specify the name and sub-category of the board, but the category is assumed to be \texttt{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 \texttt{tikzpicture} must have the style \texttt{zoned}. This style will record the bounding box of the picture which we will need to calculate VASSAL coordinates later on.

Other styles are \texttt{zone scope}, to be applied to \texttt{scopes} in the picture, and \texttt{zone path} to be applied to \texttt{paths} (or \texttt{draw}, \texttt{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 \texttt{zone scope} applied to a \texttt{scope}, what is recorded are

- The current coordinate transformation matrix
- The current translation
- The bounding box, within the current transformation and translation.
To define a zone in the board, simply enclose it in a

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

The \text{name} 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 \text{name} 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 T, so that it won’t collide with the main zone. Similar if \text{name} contains \text{oob}, except the separator is set to O.

If \text{name} contains the sub-string \text{pool}, then it is assumed to be a pool of counters, and 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 \text{every hex} or \text{every hex node}). All coordinates, and such are recorded in centimetres. It is worth remembering that the Tikz coordinate system has the \text{y} axis point upward, while typical image software has the \text{y} axis point down. \text{pdftocairo} typically assumes a 150 PPI (pixels-per-inch) resolution.

That means that scaling factor becomes

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

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.

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

The exact definition of 1pt is

\[
1\text{pt} = \frac{249}{250}\text{in} = \frac{1}{864} = \frac{83}{6000}\text{cm} = 0.03514598035145980351
\]
\pgfpointanchor{#1}{#2}\
\pgfgetlastxy{\tmp@x}{\tmp@y}\
\pt@to@cm{\tmp@x}\edef{\tmp@x}{\pgfmathresult}\
\pt@to@cm{\tmp@y}\edef{\tmp@y}{\pgfmathresult}\
\def{\mk@get@global@anchor(#1)}{\pgfpointanchor{#1}{#2}}\
\pgfpointanchor{#1}{#2}\
\pgfgetlastxy{\tmp@x}{\tmp@y}\
\pgfpointtransformed{\pgfpoint{\tmp@x}{\tmp@y}}\pgf@xa=\pgf@x\pgf@ya=\pgf@y\
\pt@to@cm{\pgf@xa}\edef{\pdf@xa}{\pgfmathresult}\
\pt@to@cm{\pgf@ya}\edef{\pdf@ya}{\pgfmathresult}\
\def{\mk@get@anchor(#1)}{\pgfpointanchor{#1}{south west}}\
\pgfpointanchor{#1}{south west}\pgfgetlastxy{\tmp@llx}{\tmp@lly}\
\pgfpointanchor{#1}{north east}\pgfpointanchor{#1}{north east}\pgfgetlastxy{\tmp@urx}{\tmp@ury}\
\pt@to@cm{\tmp@llx}\edef{\llx}{\pgfmathresult}\
\pt@to@cm{\tmp@lly}\edef{lly}{\pgfmathresult}\
\pt@to@cm{\tmp@urx}\edef{\urx}{\pgfmathresult}\
\pt@to@cm{\tmp@ury}\edef{\ury}{\pgfmathresult}\
\mk@get@anchor{#1}{south west}\
\edef{\llx}{\tmp@x}\edef{lly}{\tmp@y}\
\mk@get@anchor{#1}{north east}\
\edef{\urx}{\tmp@x}\edef{\ury}{\tmp@y}\
\def{\mk@transform}{\pgfgettransformentries{\mxx}{\mxy}{\myx}{\myy}{\ptdx}{\ptdy}\
\pt@to@cm{\ptdx}\edef{\pdf@dx}{\pgfmathresult}\
\pt@to@cm{\ptdy}\edef{\pdf@dy}{\pgfmathresult}\
\mk@w{\mk@i "xx": \mxx,}\
\mk@w{\mk@i "xy": \mxy,}\
\mk@w{\mk@i "yx": \myx,}\
\mk@w{\mk@i "yy": \myy,}\
\mk@w{\mk@i "dx": \pdf@dx,}\
\mk@w{\mk@i "dy": \pdf@dy,}\
\def{\mk@bb(#1)}{\get@bb(#1)\mk@w{\mk@i "lower left": \[\llx,\lly\],}\
\mk@w{\mk@i "upper right": \[\urx,\ury\],}\
\begingroup\pgftransforminvert\pgfpointanchor{#1}{south west}\pgfgetlastxy{\tmp@llx}{\tmp@lly}\
\pgfpointtransformed{\pgfpoint{\tmp@llx}{\tmp@lly}}\pdf@xa=\pgf@x\pdf@ya=\pgf@y}
For the key \textit{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.
Now we can make our environment
The first thing we do is to use the \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 \bd@i records the current indentation (which is important in JSON)

\newenvironment{boardimage}[3][board]{%
  \def\bd@n{#2}
  \newcount\mk@point
  \mk@point=0
  \let\oomk@i\mk@i%
  \let\markpos\mk@pos%
Then, to extract the label option, we make a dummy node with the styles every hex and every hex node, so we can extract that option.

\info{dummy}{<<dummy>>}{}%\tikz{}\	ikz{\scoped[every hex/.try,every hex node/.try]{%
\node[inner sep=0,outer sep=0]{%
\global\let\mk@label\hex@label}}}%
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.

\info*{#2}{#1}{#3}%
\mk@w{ \mk@i "zones": \@lbchar}%
\edef\mk@i{\mk@i\space}
%% Everything is made into centimeters
\mk@w{ \mk@i "units": "cm",}%
\@ifundefined{mk@label}{}{\mk@w{ \mk@i "labels": "\mk@label",}}%
%% Write out coordinate options as "coords" object
\mk@w{ \mk@i"coords": \@lbchar}%
\mk@w{ \mk@i "row": \@lbchar}%
\mk@w{ \mk@i "column": \@lbchar}%
\mk@w{ \mk@i "offset": \hex@coords@row@off,}%
\mk@w{ \mk@i "factor": \hex@coords@row@fac \@rbchar,}%
\mk@w{ \mk@i "top short": \hex@top@short@col,}%
\mk@w{ \mk@i "bottom short": \hex@bot@short@col \@rbchar}%
We then monkey-patch \boardframe to also output coordinates to our JSON file. Note that this will probably be embedded in a different object.

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

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

\begin{verbatim}
519 \tikzset{
520  zoned/.code={% Apply to whole picture
521    \pgfkeys{%
522      % This needs to be done in the picture!
523    /tikz/execute at end picture={% 
524      \mk@w{ \mk@i "zoned": \@lbchar}
525      \mk@transform%
526      \mk@bb{current bounding box}
527      \mk@w{ \mk@i \@rbchar,}
528    }
529  },
530}
\end{verbatim}

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

\begin{verbatim}
531  zone scope/.code={%
532    \mk@w{ \mk@i"zone scope ##1": \@lbchar}
533    \let\omk@i\mk@i
534    \edef\mk@i{\mk@i\space}
535    \mk@transform%
536    \bd@w{ \@rbchar,}
537    \gdef\wg@export@box{##1}%
538    \pgfkeys{%
539    /tikz/local bounding box=wg export box,
540    /tikz/execute at end scope={
541      \mk@bb{wg export box}
542      \let\mk@i\omk@i
543      \mk@w{ \mk@i\@rbchar,}},
544  } % pgfkeys
545  }, % zone scope
\end{verbatim}

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

\begin{verbatim}
546  zone point/.code n args={3}{
547    \pgf@xa=#2 cm
548    \pgf@ya=#3 cm
549    \pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}
550    \pgfkeys{%
551    /pgfpointtransformed{\pgfpoint{Opt}{Opt}}
552    \pgf@xa=\pgf@x
553    \pgf@ya=\pgf@y
554    \pt@to@cm{\the\pgf@xa}\edef\px{\pgfmathresult}
555    \pt@to@cm{\the\pgf@ya}\edef\py{\pgfmathresult}
556  } % pgfkeys
557  }, % zone point
\end{verbatim}

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

```latex
\pgf@ya=##3 cm
\advance\pgf@ya.1cm
\pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}
\pgfpointtransformed{\pgfpoint{\pgf@xa}{\pgf@ya}}
\pgf@xa=\pgf@ya
\edef\px{\pgfmathresult}
\edef\py{\pgfmathresult}
\mk@w{ \mk@i "point\the\mk@point": \@lbchar "name": "##1", "type": "point", "coords": [\px,\py] \@rbchar, }
```

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.

```latex
\mk@w{ \mk@i "name": "\bd@n" }
\let\mk@i\oomk@i
\mk@w{ \mk@i\@rbchar}
\end@info
```

TO BE DONE: We could add hooks to both the `hex` and `chit` shapes 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 PDF\LaTeX{} code. For example, we could write a counters

- Left and right identifiers
- Upper left, upper right, lower left, and lower right identifiers. (some care must be taken if these contains graphics and not just text.)
- Factors
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. Not sure how to use it though.
5.4 The wargame.hex Ti\textit{K}Z library

Used Ti\textit{K}Z 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}

\@ifempty
This is a utility macro we will use below.

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

5.4.1 Debugging

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

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

5.4.2 Suppress terrain pictures

\begin{verbatim}
\@ifundefined{ifhex@terrain@pic}{\newif\ifhex@terrain@pic\hex@terrain@pictrue}{\hex@terrain@picfalse}
\def\markpos#1(#2){}
\end{verbatim}

5.4.3 Hex coordinate system

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

Some offsets along $x$ and $y$ due to offset of every second hex column.
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
\def\hex@short@col{isfalse}
\def\hex@got@short{isfalse}
\pgfmathdeclarefunction{isfalse}{1}{\begingroup\def\pgfmathresult{0}\pgfmath@smuggleone\pgfmathresult\endgroup}
\pgfmathdeclarefunction{istrue}{1}{\begingroup\def\pgfmathresult{1}\pgfmath@smuggleone\pgfmathresult\endgroup}
\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

If we have uneven number of rows in some columns.

If we have uneven number of rows in some columns.
We define the keys for hexagon coordinates. These are the row, column, possible vertex or edge. Vertexes and edges are defined as multiple-choice. offset specifies the offset from the centre in the direction of a vertex or edge. By default, the offset is one, meaning all the way to the vertex or edge.

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

Similarly, the key column 1, will allow us to start the columns with 1.
\hex@coords@reset

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

\def\hex@coords@reset{%
  \tikzset{%
    /hex/coords/.cd,
    column=0,
    row=0,
    edge=none,
    vertex=none,
    offset=1}"

The following calculates the Cartesian coordinates from Hex coordinates

\verb|(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

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

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 \texttt{/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 \verb|\pgf@x|. We do this by

\[
x = C \cdot \frac{3}{2},
\]
where

\[ c_e = f_e(c + o_e) \]

is the effective column (stored in `\texttt{hex@eff@col}`) calculated from \( c_e = f_e(c + o_e) \), \( f_e \) (set by `\texttt{hex/column direction is}`) and the offset \( o_e \) (set by `\texttt{hex/first column is}`).

\[ \text{pgfmathparse{int(}(\texttt{hex@coords@col@fac*(\texttt{hex@col}+\texttt{hex@coords@col@off})))}\text{}} \]
\[ \text{xdef\texttt{}}\text{}`\texttt{hex@eff@col}{\texttt{pgfmathresult}}\text{}` \]
\[ \text{hex@dbg}{2}{Effective column: } \texttt{hex@coords@col@fac} \ast (\texttt{hex@col} - \texttt{hex@coords@col@off}) \rightarrow \texttt{hex@eff@col} \]
\[ \text{\texttt{pgfmathparse{hex@eff@col+1.5}}%} \]
\[ \text{xdef\texttt{}}\text{}`\texttt{hex@x{\texttt{pgfmathresult}}%} \]
\[ \text{\texttt{expandafter\pgf@x=hex@x\ cm\%}} \]

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

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

where

\[ r_e = 2f_r(r + o_r) - (c + o_e) \mod 2, \]

is the effective row (stored as \( \texttt{hex@eff@row} \)) calculated from the direction factor \( f_r \) (set by `\texttt{hex/row direction is}`) and the offset \( o_r \) (set by `\texttt{hex/first row is}`).

\[ \text{\texttt{pgfmathparse{int(}(\texttt{hex@coords@row@fac*(\texttt{hex@row}+\texttt{hex@coords@row@off})))}\text{}} \]
\[ \text{xdef\texttt{}}\text{}`\texttt{hex@eff@row}{\texttt{pgfmathresult}}\text{}` \]
\[ \text{\texttt{hex@dbg}{2}{Effective row: } \texttt{hex@coords@row@fac} \ast (\texttt{hex@row} + \texttt{hex@coords@row@off}) \rightarrow \texttt{hex@eff@row} \]
\[ \text{\texttt{pgfmathparse{2*hex@eff@row-mod(round((\texttt{hex@col}+\texttt{hex@coords@col@off})),2))\ast\texttt{hex@yy}}%} \]
\[ \text{xdef\texttt{}}\text{}`\texttt{hex@y{\texttt{pgfmathresult}}%} \]
\[ \text{\texttt{expandafter\pgf@x=hex@y\ cm\%}} \]

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

\[ \text{\texttt{ifx\texttt{}}\texttt{hex@vtx\empty}pgfpointxy\{(\texttt{hex@x})\{\texttt{hex@y}\}\texttt{else} \%} \]
\[ \text{pgfpointadd{pgfpointxy\{(\texttt{hex@x})\{\texttt{hex@y}\}\}}\{\texttt{f1}} \]
\[ \text{pgfpointscale\{(\texttt{hex@off})\pgfpointpolarxy\{(\texttt{hex@vtx})\{1\}\}}\{\texttt{f1}} \]

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

\[ \text{\texttt{ifx\texttt{}}\texttt{hex@edg\empty}else\%} \]
\[ \text{pgfpointadd{pgfpointxy\{(\texttt{hex@x})\{\texttt{hex@y}\}\}}\{\texttt{f1}} \]
\[ \text{pgfpointscale\{(\texttt{hex@off})\pgfpointpolarxy\{(\texttt{hex@edg})\{\texttt{hex@y}\}\}}\{\texttt{f1}} \]

For debugging, we can print out stuff.

\[ \text{\texttt{hex@dbg}{2}{Hex coordinates: #1 }\]
\[ \text{J c=\texttt{hex@col}} \]
\[ \text{J r=\texttt{hex@row}} \]
\[ \text{J v=\texttt{hex@vtx}} \]
\[ \text{J e=\texttt{hex@edg}} \]

53
5.4.4 Hexes

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

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

These are the actual user callable anchors. We make anchors for each vertex and mid points on each edge.

%
Next we make some shorthand aliases for each of these anchors.

\anchor{E}{east}
\anchor{W}{west}
\anchor{NE}{northeast}
\anchor{NW}{northwest}
\anchor{SW}{southwest}
\anchor{SE}{southeast}
\anchor{N edge}{northedge}
\anchor{S edge}{southedge}
\anchor{NE edge}{northeastedge}
\anchor{NW edge}{northwestedge}
\anchor{SW edge}{southwestedge}
\anchor{SE edge}{southeastedge}

The next part is commented out because it’s not obvious we’ll use these.

\savedanchor{chitnorth}{\pgfqpoint{0cm}{0.6cm}}
\savedanchor{chitsouth}{\pgfqpoint{0cm}{-0.6cm}}
\savedanchor{chiteast}{\pgfqpoint{0.6cm}{0cm}}
\savedanchor{chitwest}{\pgfqpoint{-0.6cm}{0cm}}
\savedanchor{chitnortheast}{\pgfqpoint{0.6cm}{0.6cm}}
\savedanchor{chitnorthwest}{\pgfqpoint{-0.6cm}{0.6cm}}
\savedanchor{chitsouthwest}{\pgfqpoint{-0.6cm}{-0.6cm}}
\savedanchor{chitsoutheast}{\pgfqpoint{0.6cm}{-0.6cm}}

\anchor{chit north}{chitnorth}
\anchor{chit south}{chitsouth}
\anchor{chit east}{chiteast}
\anchor{chit west}{chitwest}
\anchor{chit north east}{chitnortheast}
\anchor{chit north west}{chitnorthwest}
\anchor{chit south west}{chitsouthwest}
\anchor{chit south east}{chitsoutheast}

\anchor{chit N}{chitnorth}
\anchor{chit S}{chitsouth}
\anchor{chit E}{chiteast}
\anchor{chit W}{chitwest}
\anchor{chit NE}{chitnortheast}
\anchor{chit NW}{chitnorthwest}
\anchor{chit SW}{chitsouthwest}
\anchor{chit SE}{chitsoutheast}

The background path. This path may be drawn when the node is drawn. However, we will do most of the work in
the \behindbackgroundpath which gets drawn after this path.

\behindbackgroundpath{\init\hexpath}

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

\behindforegroundpath{%
\hexdbg{2}{Hex behind foreground path:
"\meaning\hex@terrain"
"\meaning\hex@ridges"
"\meaning\hex@town"
"\meaning\hex@extra@clip"
"\meaning\hex@label"
"\meaning\hex@extra"
"\meaning\tikzlastnode"
"\meaning\hex@row"
"\meaning\hex@col"}%
\init%

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

\scope%
\hexpath%
\pgfusepath{clip}%

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

\@ifundefined{hex@terrain}{\let\hex@terrain\empty}{}%
\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 \([\langle options\rangle]\).

\@ifundefined{hex@extra@clip}{\let\hex@extra@clip\empty}{}%
\ifx\hex@extra@clip\empty\else\hex@do@extra@clip\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%
\endscope%

56
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.

\@ifundefined{hex@town}{\let\hex@town\empty}{}
\@ifundefined{hex@c@pic}{\let\hex@c@pic\empty}{}
\ifx\hex@town\empty\else\hex@do@town\fi%

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

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

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

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>

\tikzset{%
\hex/.search also={/tikz},%
\hex/.cd,%
terrain/.store in=\hex@terrain,%
ridges/.store in=\hex@ridges,%
town/.store in=\hex@town,%
extra/.store in=\hex@extra,%
label/.store in=\hex@label,%
extra clipped/.store in=\hex@extra@clip%
}
The next key is the real work horse of the show. Specifying the `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 `\hex` is often the right choice.

```latex
\tikzset{
  \hex/hex/.style={
    transform shape,
    anchor=center,
    draw=pgfstrokecolor,
    fill=none,
    thick,
    solid},
\hex/.code={
  \hex@dbg{1}{=== Hex with options: '#1'}%\}
  \pgfkeys{/tikz/transform shape,/tikz/shape=hex/hex}
  \pgfkeys{/hex/.cd,/tikz/hex/hex,/tikz/every hex/.try,#1}}
```

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

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

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

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

```
\hex
\hex@
\hex@@
```

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.

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

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

```
\def\hex@[#1]{%
  \@ifnextchar[{{\hex@@{#1}}{\hex@@{#1}()}%\)
\}
```

Third argument is the name to be used.

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

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

```
\tikzset{
\hex/terrain/.search also={/tikz},%
\hex/terrain/.cd,
\hex@t@pic/.store in=\hex@t@pic,%
\hex@t@image/.store in=\hex@t@image,%
\hex@t@code/.store in=\hex@t@code,%
```
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 \texttt{hex=\{terrain=...\}}. The first thing into the new scope is to process the keys specified in \texttt{hex=\{terrain=...\}}. This will set the terrain and the clipping of the terrain.

\def\hex@@do@terrain{\hex@@dbg{5}{Terrain: \meaning\hex@terrain}\% \edef\hex@t@tmp{[/hex/terrain/.cd,\hex@terrain]}\% \expandafter\scope\hex@t@tmp% Scope for terrain clipping. \
\hex@@dbg{5}{Terrain: \meaning\hex@t@pic} \hex@@pic\% \hex@@dbg{5}{Terrain: \meaning\hex@t@image} \hex@@image\% \hex@@dbg{5}{Terrain: \meaning\hex@t@code} \hex@@code\% \hex@@dbg{5}{Terrain: \meaning\hex@t@clip} \hex@@clip\% Scope for terrain clipping.}

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.

\@ifundefined{hex@t@clip}{\let\hex@t@clip\empty}{% 
\ifx\hex@t@clip\empty\else
\def\hex@t@c{}
\foreach \c in \hex@t@clip{%
  \hex@dbg{5}{Clipping to \texttt{\char`\`c}}
  \expandafter\wg@pic\c\@endwg@pic{}{\wg@tmpa,\wg@tmpb}{%
    save path=\hex@t@tmp
  }%
  \wg@addto@macro\hex@t@c\hex@t@tmp % Append to clipping
%}
\pgfsyssoftpath@setcurrentpath{\hex@t@c}% Set path
\clip;% Clip to the path
% End of clipping terrain
\fi%
\fi % End of clipping terrain

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.

% Now switch between how to draw the terrain. If some of the macros are undefined, define them to be empty
\@ifundefined{hex@t@pic}{\let\hex@t@pic\empty}{%
\@ifundefined{hex@t@image}{\let\hex@t@image\empty}{%

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 \{\langle\textit{options}\rangle\}.

% If we have no image, check if we have pictures.
\ifx\hex@t@image\empty%
  \hex@dbg{8}{No terrain images}%
  \ifx\hex@t@pic\empty%
% We have pictures
  \hex@dbg{5}{Terrain pictures}%
  \pgfpointorigin\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
  \wg@pic@all{\hex@t@pic}{}{\the\wg@tmpa,\the\wg@tmpb}{}%
% We have pictures.
%}
\fi% We have images
\else % 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 %
%}
\fi%
\endscope% End of terrain scope
\endscope% End of terrain picture by preceding the picture name with \{\langle\textit{options}\rangle\}. 

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 \{\langle\textit{options}\rangle\}. 

\else % 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 %
%}
\fi%
\endscope% End of terrain scope
\endscope% End of terrain
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@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)--(240:1)--(240:#1)--(180:#1)--cycle;},
  pics/hex/#2sextant/south/.style={
    code={
      \path[pic actions](240:1)--(300:1)--(300:#1)--(240:#1)--cycle;},
  pics/hex/#2sextant/south east/.style={
    code={
      \path[pic actions](300:1)--(360:1)--(360:#1)--(300:#1)--cycle;},
  pics/hex/#2sextant/center/.style={
    code={
      \path[pic actions]
        (0:1) --
        (60:#1) --
        (120:#1) --
        (180:#1) --
        (240:#1) --
        (300:#1) --cycle;},
  pics/hex/#2sextant/NE/.style=hex/#2sextant/north east,
  pics/hex/#2sextant/NE/.style=hex/#2sextant/north east,
  pics/hex/#2sextant/NW/.style=hex/#2sextant/north west,
  pics/hex/#2sextant/NW/.style=hex/#2sextant/north west,
  pics/hex/#2sextant/SW/.style=hex/#2sextant/south west,
  pics/hex/#2sextant/SW/.style=hex/#2sextant/south west,
  pics/hex/#2sextant/SE/.style=hex/#2sextant/south east,
  pics/hex/#2sextant/SE/.style=hex/#2sextant/south east,
  pics/hex/#2sextant/C/.style=hex/#2sextant/center,
  pics/hex/#2sextant/C/.style=hex/#2sextant/center,
}
}
\hex@make@sextants\{.7\}{}
\hex@make@sextants\{.3\}{large }
\hex@make@sextants\{0\}{full }
```
Next, we define some styles for styling the terrain pictures. Users can change these styles (e.g., by appending to them) to change say the colour of the terrain graphics.

**hex/terrain/beach**

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

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

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.

**hex/terrain/beach**

```latex
\begin{verbatim}
\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, 0.8847)
      --cycle
      (-0.4032, 0.8841)
      .. controls (-0.4004, 0.8804) and (-0.3988, 0.8794) .. (-0.3956, 0.8847)
      --cycle
      (-0.2462, 0.8828)
      .. controls (-0.2425, 0.8681) and (-0.2383, 0.8546) .. (-0.2293, 0.8461)
      --cycle
      (-0.2462, 0.8828)
      .. controls (-0.2425, 0.8681) and (-0.2383, 0.8546) .. (-0.2293, 0.8461)
      --cycle
      (-0.0997, 0.8815)
      .. controls (-0.0971, 0.8706) and (-0.0941, 0.8597) .. (-0.0907, 0.8493)
      --cycle
      (-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)
      --cycle
      ( 0.0213, 0.8805)
      .. controls ( 0.0222, 0.8725) and ( 0.0235, 0.8650) .. ( 0.0262, 0.8587)
      -- ( 0.1731, 0.7216)
      .. controls ( 0.1203, 0.8649)
      --cycle
      ( 0.1097, 0.8797)
    }
  }
\end{verbatim}
```
\texttt{hex/terrain/light woods}

The draw style for light woods. The pattern is filled with light green, and outline is not drawn.

\begin{verbatim}
\tikzset{
  . controls (-0.2612,-0.6950) and (-0.2660,-0.6690) .. (-0.2914,-0.6672)
  --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)
  -- (-0.5492,-0.8022)
  . controls (-0.5310,-0.8247) and (-0.5120,-0.8367) .. (-0.5062,-0.8717)
  . controls (-0.5044,-0.8642) and (-0.5062,-0.8717) .. (-0.5063,-0.8763)
  -- (-0.4585,-0.8767)
  . controls (-0.4596,-0.7984) and (-0.5013,-0.7963) .. (-0.5234,-0.7583)
  . 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)
  -- ( 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 ( 0.4178,-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)
  -- ( 0.4240,-0.8842)
  . controls ( 0.4307,-0.8690) and ( 0.4358,-0.8541) .. ( 0.4369,-0.8443)
  --cycle
  (-0.3205,-0.8528)
  -- (-0.3266,-0.8779)
  -- (-0.2773,-0.8783)
  . controls (-0.2800,-0.8719) and (-0.2850,-0.8655) .. (-0.2963,-0.8600)
  --cycle
  ( 0.1093,-0.8668)
  . 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)
  -- ( 0.1002,-0.8815)
  -- ( 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,-0.8668) .. ( 0.1093,-0.8568)
  --cycle
  \fi
\end{verbatim}
Next, we have light woods.

\tikzset{
  \path[\hex/terrain/light woods, pic actions, draw=none]
  (-0.4795, 0.8736) -- (-0.5104, 0.8207)
  .. controls (-0.5041, 0.8191) and (-0.4967, 0.8182) .. (-0.4854, 0.8192)
  -- (-0.4770, 0.8108)
  -- (-0.4854, 0.7856)
  -- (-0.5190, 0.8023)
  .. controls (-0.5219, 0.7975) and (-0.5245, 0.7958) .. (-0.5272, 0.7916)
  -- (-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, 0.7884)
  .. 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)
  .. controls (-0.3478, 0.8487) and (-0.3526, 0.8080) .. (-0.3290, 0.7808)
  .. controls (-0.3140, 0.7633) and (-0.2394, 0.7433) .. (-0.2165, 0.7459)
  .. controls (-0.1895, 0.7488) and (-0.1787, 0.7643) .. (-0.1561, 0.7725)
  .. controls (-0.1380, 0.7791) and (-0.1179, 0.7786) .. (-0.1025, 0.7906)
  .. controls (-0.0719, 0.8182) and (-0.0936, 0.8427) .. (-0.1240, 0.8528)
  -- (-0.1323, 0.8192)
  -- (-0.1912, 0.8359)
  .. controls (-0.1995, 0.8023) and (-0.1999, 0.7965) .. (-0.2332, 0.7856)
  .. controls (-0.2512, 0.8363) and (-0.2775, 0.8009) .. (-0.2909, 0.8240)
  .. controls (-0.2975, 0.8355) and (-0.2884, 0.8535) .. (-0.2756, 0.8719)
  --cycle
  (-0.1660, 0.8709)
  .. controls (-0.1609, 0.8538) and (-0.1460, 0.8596) .. (-0.1371, 0.8707)
  --cycle
  (0.0768, 0.8689)
  .. controls (0.0767, 0.8688) and (0.0765, 0.8686) .. (0.0764, 0.8685)
  .. controls (0.0704, 0.8503) and (0.0779, 0.7592) .. (0.1533, 0.7700)
  .. controls (0.1955, 0.7761) and (0.1956, 0.8018) .. (0.1871, 0.8359)
  -- (0.1366, 0.8108)
  -- (0.1510, 0.8863)
  --cycle
  (0.1840, 0.8680)
  .. controls (0.1910, 0.8650) and (0.1993, 0.8662) .. (0.2081, 0.8678)
  --cycle
2390 --cycle
2391 (-0.5947,-0.3156)
2392 -- (-0.6030,-0.3407)
2393 -- (-0.5778,-0.3240)
2394 --cycle
2395 (-0.0014,-0.3194)
2396 .. controls (-0.0278,-0.3131) and (-0.0544,-0.3497) .. (-0.0058,-0.3694)
2397 .. controls ( 0.0049,-0.3738) and ( 0.0162,-0.3735) .. ( 0.0273,-0.3744)
2398 .. controls ( 0.0265,-0.3636) and ( 0.0269,-0.3522) .. ( 0.0223,-0.3420)
2399 .. controls ( 0.0163,-0.3283) and ( 0.0074,-0.3214) .. (-0.0014,-0.3194)
2400 --cycle
2401 (-0.6508,-0.3284)
2402 .. controls (-0.6656,-0.3290) and (-0.6816,-0.3373) .. (-0.6955,-0.3576)
2403 .. controls (-0.7159,-0.3441) and (-0.7363,-0.3292) .. (-0.7624,-0.3378)
2404 .. controls (-0.7705,-0.3405) and (-0.7771,-0.3450) .. (-0.7829,-0.3501)
2405 -- (-0.7599,-0.3912)
2406 -- (-0.7543,-0.3744)
2407 -- (-0.6619,-0.4164)
2408 -- (-0.6619,-0.3660)
2409 -- (-0.6367,-0.4164)
2410 .. controls (-0.5723,-0.3945) and (-0.6064,-0.3266) .. (-0.6508,-0.3284)
2411 --cycle
2412 (-0.5287,-0.3512)
2413 .. controls (-0.5489,-0.3498) and (-0.5690,-0.3591) .. (-0.5764,-0.3751)
2414 .. controls (-0.5966,-0.4192) and (-0.5398,-0.3912) .. (-0.5274,-0.3828)
2415 -- (-0.5190,-0.4500)
2416 -- (-0.5358,-0.4333)
2417 -- (-0.5358,-0.4248)
2418 -- (-0.5611,-0.4248)
2419 .. controls (-0.5664,-0.4641) and (-0.5605,-0.4735) .. (-0.5442,-0.5088)
2420 -- (-0.5274,-0.5088)
2421 .. controls (-0.4972,-0.4701) and (-0.4647,-0.4466) .. (-0.4806,-0.3915)
2422 .. controls (-0.4882,-0.3649) and (-0.5085,-0.3527) .. (-0.5287,-0.3512)
2423 --cycle
2424 ( 0.5485,-0.3594)
2425 .. controls ( 0.5504,-0.3686) and ( 0.5513,-0.3797) .. ( 0.5560,-0.3899)
2426 .. controls ( 0.5831,-0.4490) and ( 0.6446,-0.3809) .. ( 0.5811,-0.3594)
2427 .. controls ( 0.5706,-0.3579) and ( 0.5594,-0.3583) .. ( 0.5485,-0.3594)
2428 --cycle
2429 ( 0.3311,-0.3646)
2430 .. controls ( 0.3201,-0.3659) and ( 0.3103,-0.3791) .. ( 0.3084,-0.3917)
2431 .. controls ( 0.3035,-0.4255) and ( 0.3481,-0.5315) .. ( 0.4140,-0.4669)
2432 .. controls ( 0.4439,-0.4932) and ( 0.4962,-0.5278) .. ( 0.5388,-0.5107)
2433 .. controls ( 0.5684,-0.4999) and ( 0.5806,-0.4516) .. ( 0.5232,-0.4333)
2434 -- ( 0.5232,-0.4669)
2435 .. controls ( 0.4932,-0.4550) and ( 0.4949,-0.4563) .. ( 0.4644,-0.4669)
2436 -- ( 0.4560,-0.4248)
2437 -- ( 0.4224,-0.4417)
2438 -- ( 0.3888,-0.4080)
2439 -- ( 0.3617,-0.4500)
2440 -- ( 0.3617,-0.4080)
2441 .. controls ( 0.3546,-0.3735) and ( 0.3422,-0.3632) .. ( 0.3311,-0.3646)
2442 --cycle
The style for woods. The pattern is filled with a darker green, and outlines are not drawn.

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

Regular woods.

```latex
\ifhex@terrain@pic
\tikzset{
```

91
.. controls (0.0102,-0.2651) and (-0.0913,-0.2616) .. (-0.0376,-0.3640)
.. controls (-0.1141,-0.3685) and (-0.1262,-0.4016) .. (-0.0926,-0.4664)
.. controls (-0.0856,-0.4795) and (-0.0758,-0.5040) .. (-0.0668,-0.5138)
.. controls (-0.0449,-0.5377) and (0.0001,-0.5440) .. (0.0307,-0.5431)
-- (0.0221,-0.4579)
-- (-0.0206,-0.4833)
.. controls (-0.0180,-0.4388) and (-0.0055,-0.4140) .. (-0.0633,-0.4237)
.. controls (-0.0449,-0.4795) and (-0.0758,-0.5040) .. (-0.0668,-0.5138)
.. controls (-0.0449,-0.3341) and (0.0150,-0.3375) .. (-0.0335,-0.3640)
.. controls (-0.0031,-0.3175) and (0.0507,-0.3021) .. (0.0895,-0.3132)
.. controls (0.0967,-0.3153) and (0.1020,-0.3182) .. (0.1069,-0.3214)
-- (0.0733,-0.3214)
-- (0.0733,-0.3555)
-- (0.1073,-0.3555)
-- (0.1073,-0.3217)
.. controls (0.1209,-0.3306) and (0.1314,-0.3431) .. (0.1671,-0.3555)
.. controls (0.1861,-0.3011) and (0.1658,-0.3044) .. (0.1276,-0.2793)
.. controls (0.1121,-0.2692) and (0.1003,-0.2524) .. (0.0809,-0.2495)
--cycle
(0.5253,-0.2617)
.. controls (0.5159,-0.2894) and (0.5137,-0.2935) .. (0.5423,-0.3042)
--cycle
(-0.5836,-0.2872)
-- (-0.5921,-0.3299)
.. controls (-0.5632,-0.3196) and (-0.5624,-0.3175) .. (-0.5665,-0.2872)
--cycle
(0.4825,-0.3113)
.. controls (0.4709,-0.3098) and (0.4621,-0.3187) .. (0.4571,-0.3470)
.. controls (0.4065,-0.2945) and (0.3565,-0.3080) .. (0.3141,-0.3613)
.. controls (0.3029,-0.3754) and (0.2880,-0.3874) .. (0.2903,-0.4075)
.. controls (0.2921,-0.4247) and (0.3027,-0.4361) .. (0.3120,-0.4493)
.. controls (0.2945,-0.4516) and (0.2350,-0.4574) .. (0.2234,-0.4665)
.. controls (0.2011,-0.4843) and (0.2099,-0.5378) .. (0.2182,-0.5602)
-- (0.2352,-0.5602)
.. controls (0.2421,-0.5417) and (0.2492,-0.5116) .. (0.2706,-0.5071)
.. controls (0.3048,-0.4961) and (0.3439,-0.5674) .. (0.3618,-0.5071)
-- (0.3618,-0.4833)
-- (0.3974,-0.4919)
-- (0.4059,-0.4579)
-- (0.3376,-0.4493)
-- (0.3717,-0.3896)
.. controls (0.4306,-0.3991) and (0.4623,-0.4570) .. (0.4656,-0.3640)
.. controls (0.4971,-0.3707) and (0.5062,-0.3751) .. (0.5253,-0.3470)
-- (0.5451,-0.3694)
.. controls (0.6171,-0.4271) and (0.5795,-0.2610) .. (0.5253,-0.3384)
.. controls (0.5089,-0.3248) and (0.4943,-0.3128) .. (0.4825,-0.3113)
--cycle
(0.6371,-0.3426)
.. controls (0.6165,-0.3414) and (0.6075,-0.3499) .. (0.6021,-0.3811)
.. controls (0.6375,-0.3781) and (0.6440,-0.3786) .. (0.6618,-0.3470)
.. 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)
-- (-0.4896, -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, -0.3811)
-- ( 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, -0.4241)
.. controls (-0.3654, -0.4287) and (-0.3677, -0.4184) .. (-0.3740, -0.4070)
.. controls (-0.3858, -0.4446) and (-0.3796, -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 ( 0.6767, -0.4763) and ( 0.6924, -0.4939) .. ( 0.7031, -0.4938)
.. controls ( 0.7482, -0.4516) and ( 0.7588, -0.4217) .. ( 0.6869, -0.4249)
.. controls ( 0.6795, -0.4222) and ( 0.6716, -0.4207) .. ( 0.6534, -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
(-0.3947, -0.4820)
The pattern for swamps. The pattern is filled with a light blue.

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

\section*{hex/terrain/swamp}

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

\begin{tikzpicture}
\tikzset{
  hex/terrain/swamp/.pic={
    \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
  }
}
\end{tikzpicture}
hex/terrain/rough

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

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

hex/terrain/rough

Roughs. Again, a bit long.

\ifhex@terrain@pic
\tikzset{
  hex/terrain/rough/.pic={
    \path[hex/terrain/rough,pic actions,draw=none]
    (-0.2701, 0.8873) .. controls (-0.3250, 0.8675) and (-0.3296, 0.8537) .. (-0.2982, 0.8927) .. controls (-0.3058, 0.8263) and (-0.2717, 0.8450) .. (-0.2591, 0.8228) .. controls (-0.2441, 0.8112) and (-0.2057, 0.7817) .. (-0.2226, 0.8356) .. controls (-0.2359, 0.8698) and (-0.2532, 0.8840) .. (-0.2701, 0.8873)
    --cycle
    (-0.1081, 0.8792) .. controls (-0.1377, 0.8610) and (-0.1265, 0.8900) .. (-0.1371, 0.8680) .. controls (-0.1163, 0.8536) and (-0.1121, 0.8691) .. (-0.1081, 0.8792)
    --cycle
    ( 0.1762, 0.8752) .. controls ( 0.1761, 0.8710) and ( 0.1707, 0.8704) .. ( 0.1746, 0.8556) .. controls ( 0.1958, 0.8423) and ( 0.2514, 0.8065) .. ( 0.3216, 0.8740)
    --cycle
    ( 0.3216, 0.8740)
  }
}\fi
.. controls (0.3061, 0.8744) and (0.2932, 0.8668) .. (0.2896, 0.8414)
-- (0.3292, 0.7930)
-- (0.3702, 0.8584)
.. controls (0.3556, 0.8652) and (0.3372, 0.8736) .. (0.3216, 0.8740)
--cycle
(0.4696, 0.8697)
.. controls (0.4362, 0.8687) and (0.4116, 0.8113) .. (0.4594, 0.7865)
-- (0.4565, 0.8238)
-- (0.5034, 0.8485)
.. controls (0.4927, 0.8641) and (0.4807, 0.8700) .. (0.4696, 0.8697)
--cycle
(-0.0783, 0.8695)
-- (-0.0698, 0.8185)
-- (-0.0528, 0.8185)
.. controls (-0.0488, 0.8507) and (-0.0499, 0.8533) .. (-0.0783, 0.8695)
--cycle
(0.0321, 0.8695)
.. controls (-0.0074, 0.8534) and (-0.0195, 0.8453) .. (-0.0104, 0.8015)
.. controls (0.0252, 0.8183) and (0.0356, 0.8295) .. (0.0321, 0.8695)
--cycle
(-0.4155, 0.8596)
.. controls (-0.4417, 0.8307) and (-0.4165, 0.8213) .. (-0.4032, 0.8284)
.. controls (-0.3903, 0.8353) and (-0.3789, 0.8639) .. (-0.4155, 0.8596)
--cycle
(-0.4857, 0.8525)
.. controls (-0.4972, 0.8321) and (-0.5172, 0.8207) .. (-0.5389, 0.8116)
-- (-0.5595, 0.7763)
.. controls (-0.5377, 0.7748) and (-0.5144, 0.7944) .. (-0.4942, 0.8100)
-- (-0.4857, 0.7845)
.. controls (-0.4503, 0.8051) and (-0.4552, 0.8169) .. (-0.4688, 0.8525)
--cycle
(0.1002, 0.8511)
.. controls (0.0869, 0.8528) and (0.0769, 0.8478) .. (0.0696, 0.8260)
-- (0.1509, 0.8185)
-- (0.1509, 0.8355)
.. controls (0.1302, 0.8408) and (0.1135, 0.8493) .. (0.1002, 0.8511)
--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)
--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)
-- (-0.3770, -0.8622)
-- controls (-0.3770, -0.8627) and (-0.3768, -0.8631) .. (-0.3768, -0.8636)
-- (-0.3401, -0.8539)
-- (-0.3245, -0.7858)
--cycle
(0.2612, -0.7858)
-- controls (0.2125, -0.7858) and (0.1976, -0.7780) .. (0.1594, -0.8113)
-- controls (0.2029, -0.8570) and (0.2552, -0.8596) .. (0.2612, -0.7858)
--cycle
(-0.1292, -0.7943)
-- (-0.1886, -0.8537)
-- controls (-0.1463, -0.8604) and (-0.1341, -0.8487) .. (-0.1038, -0.8198)
--cycle
(-0.4348, -0.8367)
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.

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

And the mountains pattern. This is the same as the beach pattern, only filled with a darker brown colour.
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.

```latex
\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
  },
}\fi
```
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)
-- ( 0.0070,-0.8700);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( 0.7575, 0.4367)
-- ( 0.3945, 0.3375)
-- ( 0.3945, 0.3375);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
(-0.1900,-0.0806)
-- (-0.1155, 0.1588);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
(-0.1308, 0.1580)
-- (-0.7603, 0.4394);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
(-0.6615,-0.2309)
-- (-0.6777,-0.3255)
-- (-0.7607,-0.4327);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
(-0.6676,-0.2405)
-- (-0.4599,-0.1067)
-- (-0.1877,-0.0679);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
(-0.6676,-0.2405)
-- (-0.4599,-0.1067)
-- (-0.1877,-0.0679);
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( 0.2082,-0.3003)
-- ( 0.4578,-0.4855)
-- ( 0.5914,-0.3675)
-- ( 0.7607,-0.4420);
A town.
\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)
-- ( 0.4358,-0.4688);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0439,-0.3617)
-- (-0.0921,-0.5012)
-- (-0.2865,-0.3243)
-- (-0.4420,-0.4608)
-- (-0.5795,-0.4446)
-- (-0.6421,-0.3520);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0622,-0.6515)
-- (-0.0316,-0.6176)
-- (-0.0221,-0.5364);
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.0073, 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)
\path[hex/terrain/town/house,pic actions]
( 0.2761, 0.3992)
-- ( 0.2947, 0.3765)
-- ( 0.2533, 0.3425)
-- ( 0.2347, 0.3651)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.2761, 0.3992)
-- ( 0.2947, 0.3765)
-- ( 0.2533, 0.3425)
-- ( 0.2347, 0.3651)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.2761, 0.3992)
-- ( 0.2947, 0.3765)
-- ( 0.2533, 0.3425)
-- ( 0.2347, 0.3651)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.2761, 0.3992)
-- ( 0.2947, 0.3765)
-- ( 0.2533, 0.3425)
-- ( 0.2347, 0.3651)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.2761, 0.3992)
-- ( 0.2947, 0.3765)
-- ( 0.2533, 0.3425)
-- ( 0.2347, 0.3651)
--cycle
\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]
( 0.3227, 0.3548)
-- ( 0.3421, 0.3329)
-- ( 0.3020, 0.2974)
-- ( 0.2826, 0.3192)
--cycle
\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]
(-0.2473, 0.2770)
-- (-0.2380, 0.3048)
-- (-0.1871, 0.2879)
-- (-0.1964, 0.2601)
--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]
(-0.1395, 0.3602)
\path[hex/terrain/town/house,pic actions]
( 0.2901, 0.6234)
\path[hex/terrain/town/house,pic actions]
(-0.3456, 0.2854)
\path[hex/terrain/town/house,pic actions]
(-0.3040, 0.3746)
\path[hex/terrain/town/house,pic actions]
(-0.7420,-0.2456)
\path[hex/terrain/town/house,pic actions]
(-0.7302,-0.2189)
\path[hex/terrain/town/house,pic actions]
(-0.6812,-0.2407)
\path[hex/terrain/town/house,pic actions]
(-0.6930,-0.2674)
\path[hex/terrain/town/house,pic actions]
(-0.7420,-0.2456)
\path[hex/terrain/town/house,pic actions]
(-0.0647, 0.4710)
-- (-0.0215, 0.4559)
-- (-0.0416, 0.3984)
-- (-0.0848, 0.4135)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0647, 0.4710)
-- (-0.0215, 0.4559)
-- (-0.0416, 0.3984)
-- (-0.0848, 0.4135)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1476,-0.3704)
-- (-0.1403,-0.3251)
-- (-0.0802,-0.3347)
-- (-0.0873,-0.3799)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1476,-0.3704)
-- (-0.1403,-0.3251)
-- (-0.0802,-0.3347)
-- (-0.0873,-0.3799)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0755, 0.3210)
-- (-0.0531, 0.3610)
-- ( 0.0001, 0.3312)
-- (-0.0224, 0.2912)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0755, 0.3210)
-- (-0.0531, 0.3610)
-- ( 0.0001, 0.3312)
-- (-0.0224, 0.2912)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.1354, 0.0442)
-- (-0.1129, 0.0842)
-- (-0.0599, 0.0544)
-- (-0.0823, 0.0144)
\path[hex/terrain/town/house,pic actions]
(-0.1354, 0.0442)
-- (-0.1129, 0.0842)
-- (-0.0599, 0.0544)
-- (-0.0823, 0.0144)
--cycle

\path[hex/terrain/town/house,pic actions]
(-0.1672,-0.0608)
-- (-0.1524,-0.0175)
-- (-0.0948,-0.0371)
-- (-0.1096,-0.0805)
--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]
( 0.3095, 0.1272)
\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]
(-0.0124, 0.1558)
-- (0.0333, 0.1525)
-- (0.0333, 0.1525)
-- (0.0290, 0.0918)
-- (0.0167, 0.0950)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(0.2904, 0.2714)
-- (0.3361, 0.2681)
-- (0.3318, 0.2074)
-- (0.2861, 0.2106)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(0.2904, 0.2714)
-- (0.3361, 0.2681)
-- (0.3318, 0.2074)
-- (0.2861, 0.2106)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0124, 0.1558)
-- (0.0333, 0.1525)
-- (0.0333, 0.1525)
-- (0.0290, 0.0918)
-- (0.0167, 0.0950)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(0.2904, 0.2714)
-- (0.3361, 0.2681)
-- (0.3318, 0.2074)
-- (0.2861, 0.2106)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.0124, 0.1558)
-- (0.0333, 0.1525)
-- (0.0333, 0.1525)
-- (0.0290, 0.0918)
-- (0.0167, 0.0950)
--cycle

181
And finally a city

\ifhex@terrain@pic
\tikzset{
hex/terrain/city/.pic={
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.6475, 0.4068)
-- ( 0.7314,-0.0575)
-- ( 0.7314,-0.0575)
;}
\path[hex/terrain/town/small road,pic actions,hex/terrain/town/post road]
( 0.3200,-0.0497)
-- ( 0.7360,-0.0572)
}
\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)
-- (-0.3449, 0.4978)
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( 0.0439, 0.3452)
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( -0.4152, 0.2683)
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( -0.8106,-0.1286)
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( -0.6697,-0.2422)
\path[hex/terrain/town/road,pic actions,hex/terrain/town/post road]
( 0.2080,-0.3022)
\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/house,pic actions]
( 0.1146, 0.0405)
-- ( 0.1598, 0.0323)
-- ( 0.1489,-0.0277)
-- ( 0.1036,-0.0196)
--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]
( 0.8016, 0.0292)
-- ( 0.8235, 0.0367)
-- ( 0.8321, 0.0116)
-- ( 0.8103, 0.0040)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.8016, 0.0292)
-- ( 0.8235, 0.0367)
-- ( 0.8321, 0.0116)
-- ( 0.8103, 0.0040)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7392, 0.1737)
-- ( 0.7609, 0.1816)
-- ( 0.7702, 0.1568)
-- ( 0.7485, 0.1487)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7392, 0.1737)
-- ( 0.7609, 0.1816)
-- ( 0.7702, 0.1568)
-- ( 0.7485, 0.1487)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3736, 0.7805)
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path{hex/terrain/town/house,pic actions}
(0.2516,-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)
--cycle
\
\path{hex/terrain/town/house,pic actions}
(0.1222,-0.7854)
-- (0.1514,-0.7854)
-- (0.1514,-0.8390)
-- (0.1222,-0.8390)
--cycle
\
\path{hex/terrain/town/house,pic actions}
(0.3031,-0.7156)
-- (0.3325,-0.7156)
-- (0.3325,-0.7693)
-- (0.3031,-0.7693)
--cycle
\
\path{hex/terrain/town/house,pic actions}
(0.3031,-0.7156)
-- (0.3325,-0.7156)
-- (0.3325,-0.7693)
-- (0.3031,-0.7693)
--cycle
\
\path{hex/terrain/town/house,pic actions}
\path[hex/terrain/town/house,pic actions]
( 0.3574,-0.7174)
-- ( 0.3867,-0.7174)
-- ( 0.3867,-0.7712)
-- ( 0.3574,-0.7712)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3742,-0.8016)
-- ( 0.4036,-0.8016)
-- ( 0.4036,-0.8554)
-- ( 0.3742,-0.8554)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4107,-0.8072)
-- ( 0.4400,-0.8072)
-- ( 0.4400,-0.8610)
-- ( 0.4107,-0.8610)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4612,-0.7886)
-- ( 0.4905,-0.7886)
-- ( 0.4905,-0.8423)
-- ( 0.4612,-0.8423)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4612,-0.7886)
-- ( 0.4905,-0.7886)
-- ( 0.4905,-0.8423)
-- ( 0.4612,-0.8423)
\path[hex/terrain/town/house,pic actions]
( 0.1786,-0.4343)
-- ( 0.1889,-0.4069)
-- ( 0.2391,-0.4259)
-- ( 0.2289,-0.4532)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1647,-0.4763)
-- ( 0.1772,-0.4497)
-- ( 0.2258,-0.4725)
-- ( 0.2134,-0.4990)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1647,-0.4763)
-- ( 0.1772,-0.4497)
-- ( 0.2258,-0.4725)
-- ( 0.2134,-0.4990)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2335,-0.5197)
-- ( 0.2460,-0.4932)
-- ( 0.2946,-0.5160)
-- ( 0.2821,-0.5425)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2335,-0.5197)
-- ( 0.2460,-0.4932)
-- ( 0.2946,-0.5160)
-- ( 0.2821,-0.5425)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2832,-0.4218)
-- ( 0.2956,-0.3952)
-- ( 0.3443,-0.4180)
-- ( 0.3318,-0.4445)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2832,-0.4218)
-- ( 0.2956,-0.3952)
-- ( 0.3443,-0.4180)
-- ( 0.3318,-0.4445)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2064,-0.6136)
-- ( 0.2189,-0.5871)
-- ( 0.2675,-0.6099)
-- ( 0.2551,-0.6364)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.2064,-0.6136)
-- ( 0.2189,-0.5871)
-- ( 0.2675,-0.6099)
-- ( 0.2551,-0.6364)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1443,-0.5195)
-- ( 0.1550,-0.4922)
-- ( 0.2050,-0.5120)
-- ( 0.1942,-0.5392)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1443,-0.5195)
-- ( 0.1550,-0.4922)
-- ( 0.2050,-0.5120)
-- ( 0.1942,-0.5392)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3128,-0.5333)
-- ( 0.3394,-0.5455)
-- ( 0.3173,-0.5943)
-- ( 0.2906,-0.5823)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3128,-0.5333)
-- ( 0.3394,-0.5455)
-- ( 0.3173,-0.5943)
-- ( 0.2906,-0.5823)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1781,-0.6526)
-- ( 0.1883,-0.6250)
-- ( 0.2386,-0.6435)
-- ( 0.2286,-0.6710)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1781,-0.6526)
-- ( 0.1883,-0.6250)
-- ( 0.2386,-0.6435)
\path[hex/terrain/town/house,pic actions] (-0.2660,-0.5113) -- (-0.2456,-0.5323) -- (-0.2840,-0.5697) -- (-0.3045,-0.5487) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.2660,-0.5113) -- (-0.2456,-0.5323) -- (-0.2840,-0.5697) -- (-0.3045,-0.5487) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3939,-0.2212) -- (-0.3666,-0.2319) -- (-0.3863,-0.2819) -- (-0.4135,-0.2712) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3939,-0.2212) -- (-0.3666,-0.2319) -- (-0.3863,-0.2819) -- (-0.4135,-0.2712) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3038,-0.2403) -- (-0.2765,-0.2509) -- (-0.2961,-0.3010) -- (-0.3234,-0.2902) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3038,-0.2403) -- (-0.2765,-0.2509) -- (-0.2961,-0.3010) -- (-0.3234,-0.2902) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3532,-0.2251) -- (-0.3255,-0.2346) -- (-0.3428,-0.2854) -- (-0.3705,-0.2760) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.3532,-0.2251) -- (-0.3255,-0.2346)
\path[hex/terrain/town/house,pic actions] (-0.4272,-0.6520) -- (-0.4128,-0.6774) -- (-0.4595,-0.7039) -- (-0.4740,-0.6785) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5374,-0.6782) -- (-0.5236,-0.7040) -- (-0.5710,-0.7292) -- (-0.5848,-0.7034) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5214,-0.7131) -- (-0.5038,-0.7365) -- (-0.5468,-0.7687) -- (-0.5644,-0.7453) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5214,-0.7131) -- (-0.5038,-0.7365) -- (-0.5468,-0.7687) -- (-0.5644,-0.7453) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2847,-0.1917) -- ( 0.2954,-0.1644) -- ( 0.3454,-0.1841) -- ( 0.3347,-0.2114) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2847,-0.1917) -- ( 0.2954,-0.1644) -- ( 0.3454,-0.1841) -- ( 0.3347,-0.2114) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2692,-0.2397) -- ( 0.2775,-0.2116) -- ( 0.3291,-0.2268) -- ( 0.3208,-0.2548) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.2692,-0.2397) -- ( 0.2775,-0.2116) -- ( 0.3291,-0.2268) -- ( 0.3208,-0.2548) --cycle;
\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] ( 0.2587,-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] ( 0.1269,-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] ( 0.1127,-0.3153)
\path[hex/terrain/town/house,pic actions] (-0.6244, 0.4780) \path[hex/terrain/town/house,pic actions] (-0.6236, 0.1401) \path[hex/terrain/town/house,pic actions] (-0.3872, 0.3590) \path[hex/terrain/town/house,pic actions] (-0.6525, 0.1021)

\path[hex/terrain/town/house,pic actions] (-0.5977, 0.4661) \path[hex/terrain/town/house,pic actions] (-0.6195, 0.4170) \path[hex/terrain/town/house,pic actions] (-0.5661, 0.1611) \path[hex/terrain/town/house,pic actions] (-0.5705, 0.1321)

\path[hex/terrain/town/house,pic actions] (-0.6463, 0.4289) \path[hex/terrain/town/house,pic actions] (-0.6192, 0.1691) \path[hex/terrain/town/house,pic actions] (-0.3829, 0.3880) \path[hex/terrain/town/house,pic actions] (-0.3341, 0.3510)

\path[hex/terrain/town/house,pic actions] (-0.6463, 0.4289) \path[hex/terrain/town/house,pic actions] (-0.5705, 0.1321) \path[hex/terrain/town/house,pic actions] (-0.3341, 0.3510) \path[hex/terrain/town/house,pic actions] (-0.6036, 0.0799)

\path[hex/terrain/town/house,pic actions] (-0.6195, 0.4170) \path[hex/terrain/town/house,pic actions] (-0.5705, 0.1321) \path[hex/terrain/town/house,pic actions] (-0.3341, 0.3510) \path[hex/terrain/town/house,pic actions] (-0.6036, 0.0799)
\path[hex/terrain/town/house,pic actions]
(-0.7316,-0.0595)
-- (-0.7215,-0.0320)
-- (-0.6711,-0.0505)
-- (-0.6812,-0.0780)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.7748,-0.1355)
-- (-0.7629,-0.1088)
-- (-0.7138,-0.1305)
-- (-0.7257,-0.1573)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.6698,-0.1385)
-- (-0.6512,-0.1159)
-- (-0.6098,-0.1501)
-- (-0.6284,-0.1727)
--cycle
\path[hex/terrain/town/house,pic actions]
(-0.3325,-0.1175)
-- (-0.3067,-0.1313)
-- (-0.3320,-0.1787)
-- (-0.3579,-0.1649)
--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]
\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]
( 0.5422,-0.0041)
-- ( 0.5437,-0.0333)
-- ( 0.4900,-0.0361)
-- ( 0.4885,-0.0068)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.6654,-0.0050)
-- ( 0.6664,-0.0343)
-- ( 0.6128,-0.0361)
-- ( 0.6117,-0.0068)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7573, 0.2817)
-- ( 0.7646, 0.2533)
-- ( 0.7124, 0.2402)
-- ( 0.7053, 0.2686)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7573, 0.2817)
-- ( 0.7646, 0.2533)
-- ( 0.7124, 0.2402)
-- ( 0.7053, 0.2686)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7393, 0.3502)
-- ( 0.7456, 0.3216)
-- ( 0.6931, 0.3101)
-- ( 0.6868, 0.3387)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7393, 0.3502)
-- ( 0.7456, 0.3216)
-- ( 0.6931, 0.3101)
-- ( 0.6868, 0.3387)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.8114, 0.0963)
-- ( 0.8188, 0.0679)
-- ( 0.7668, 0.0543)
\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]
(0.6398, -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)
\path[hex/terrain/town/house,pic actions]
(-0.7770,-0.2886)
-- (-0.7319,-0.2679)
-- (-0.6985,-0.3406)
-- (-0.7437,-0.3613)
--cycle
;
\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]
(-0.0626,-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] (0.0906,-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] (-0.4731,-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] (0.4125,-0.0879) -- (0.4578,-0.0951) -- (0.4483,-0.1553) -- (0.4029,-0.1481) --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]
(-0.3208,-0.3936)
\path[hex/terrain/town/house,pic actions]
(-0.3069,-0.3498)
\path[hex/terrain/town/house,pic actions]
(-0.2487,-0.3684)
\path[hex/terrain/town/house,pic actions]
(-0.2627,-0.4122)
\path[hex/terrain/town/house,pic actions]
(-0.3208,-0.3936)
\path[hex/terrain/town/house,pic actions]
( 0.1634,-0.1430)
\path[hex/terrain/town/house,pic actions]
( 0.1790,-0.0997)
\path[hex/terrain/town/house,pic actions]
( 0.2365,-0.1205)
\path[hex/terrain/town/house,pic actions]
( 0.2209,-0.1637)
\path[hex/terrain/town/house,pic actions]
( 0.1634,-0.1430)
\path[hex/terrain/town/house,pic actions]
( 0.1790,-0.0997)
\path[hex/terrain/town/house,pic actions]
( 0.2365,-0.1205)
\path[hex/terrain/town/house,pic actions]
( 0.2209,-0.1637)
\path[hex/terrain/town/house,pic actions]
( 0.1520,-0.2030)
\path[hex/terrain/town/house,pic actions]
( 0.1715,-0.1614)
\path[hex/terrain/town/house,pic actions]
( 0.2268,-0.1873)
\path[hex/terrain/town/house,pic actions]
( 0.2072,-0.2289)
\path[hex/terrain/town/house,pic actions]
( 0.1520,-0.2030)
\path[hex/terrain/town/house,pic actions]
( 0.1715,-0.1614)
\path[hex/terrain/town/house,pic actions]
( 0.2268,-0.1873)
\path[hex/terrain/town/house,pic actions]
( 0.2072,-0.2289)
\path[hex/terrain/town/house,pic actions]
( 0.0852,-0.3696)
\path[hex/terrain/town/house,pic actions]
( 0.1047,-0.3280)
\path[hex/terrain/town/house,pic actions]
( 0.1600,-0.3540)
\path[hex/terrain/town/house,pic actions]
( 0.1404,-0.3956)
\path[hex/terrain/town/house,pic actions]
\path[hex/terrain/town/house,pic actions]
\path
(-0.0445, 0.0687)
-- (-0.0010, 0.0535)
-- (-0.0213, -0.0041)
-- (-0.0646, 0.0110)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.9043, 0.1125)
-- ( 0.9184, 0.0687)
-- ( 0.8603, 0.0500)
-- ( 0.8462, 0.0937)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.7148,-0.2814)
-- ( 0.7591,-0.2935)
-- ( 0.7430,-0.3524)
-- ( 0.6987,-0.3402)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.5891,-0.1425)
-- ( 0.5806,-0.1876)
-- ( 0.5207,-0.1764)
-- ( 0.5291,-0.1313)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.5865, 0.0684)
-- ( 0.5782, 0.0233)
-- ( 0.5181, 0.0345)
-- ( 0.5266, 0.0796)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.5865, 0.0684)
-- ( 0.5782, 0.0233)
-- ( 0.5181, 0.0345)
-- ( 0.5266, 0.0796)
--cycle
\path[hex/terrain/town/house,pic actions]
( 0.4044,-0.3422)
-- ( 0.4189,-0.2987)
\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;
\path[hex/terrain/town/house,pic actions] (-0.6443,-0.1022) -- (-0.6321,-0.0579) -- (-0.5733,-0.0740) -- (-0.5854,-0.1183) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.6032, 0.2357) -- (-0.5912, 0.2800) -- (-0.5323, 0.2639) -- (-0.5443, 0.2196) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.6032, 0.2357) -- (-0.5912, 0.2800) -- (-0.5323, 0.2639) -- (-0.5443, 0.2196) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.7230, 0.0020)
\path[hex/terrain/town/house,pic actions]
(-0.7230, 0.0020)
-- (-0.7026, 0.0432)
-- (-0.6479, 0.0162)
-- (-0.6682,-0.0250)
--cycle
;
\path[hex/terrain/town/house,pic actions]
(-0.5055, 0.2596)
-- (-0.4629, 0.2423)
-- (-0.4859, 0.1858)
-- (-0.5285, 0.2031)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1337, 0.3296)
-- ( 0.1283, 0.2840)
-- ( 0.0677, 0.2911)
-- ( 0.0731, 0.3367)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1337, 0.3296)
-- ( 0.1283, 0.2840)
-- ( 0.0677, 0.2911)
-- ( 0.0731, 0.3367)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1476, 0.4414)
-- ( 0.1506, 0.3955)
-- ( 0.0897, 0.3916)
-- ( 0.0867, 0.4375)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.1476, 0.4414)
-- ( 0.1506, 0.3955)
-- ( 0.0897, 0.3916)
-- ( 0.0867, 0.4375)
--cycle
;\path[hex/terrain/town/house,pic actions]
( 0.1167,-0.5813)
-- ( 0.1315,-0.5379)
-- ( 0.1892,-0.5576)
-- ( 0.1744,-0.6011)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0916,-0.6322)
-- ( 0.1064,-0.5888)
-- ( 0.1642,-0.6085)
-- ( 0.1493,-0.6520)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.0916,-0.6322)
-- ( 0.1064,-0.5888)
-- ( 0.1642,-0.6085)
-- ( 0.1493,-0.6520)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3791,-0.5978)
-- ( 0.3941,-0.5544)
-- ( 0.4518,-0.5741)
-- ( 0.4369,-0.6176)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.3791,-0.5978)
-- ( 0.3941,-0.5544)
-- ( 0.4518,-0.5741)
-- ( 0.4369,-0.6176)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.4116,-0.5397)
-- ( 0.4392,-0.5029)
-- ( 0.4880,-0.5396)
-- ( 0.4604,-0.5764)
--cycle
;
\path[hex/terrain/town/house,pic actions]
( 0.7950, 0.1548)
\path[hex/terrain/town/house,pic actions]
( 0.5739, 0.6926)
\path[hex/terrain/town/house,pic actions]
( 0.6499, 0.5535)
\path[hex/terrain/town/house,pic actions]
(-0.4994, 0.7998)
\path[hex/terrain/town/house,pic actions]
(-0.4994, 0.7998)
\path[hex/terrain/town/house,pic actions] (-0.3350, 0.7630) -- (-0.2917, 0.7475) -- (-0.3125, 0.6901) -- (-0.3558, 0.7057) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.4472, 0.6282) -- (-0.4040, 0.6124) -- (-0.4252, 0.5551) -- (-0.4683, 0.5710) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5200, 0.6603) -- (-0.4792, 0.6394) -- (-0.5070, 0.5851) -- (-0.5480, 0.6061) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5301, 0.5359) -- (-0.4915, 0.5109) -- (-0.5247, 0.4597) -- (-0.5633, 0.4846) --cycle;
\path[hex/terrain/town/house,pic actions] (-0.5301, 0.5359) -- (-0.4915, 0.5109) -- (-0.5247, 0.4597) -- (-0.5633, 0.4846)
\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] (-0.0147, 0.1985) -- ( 0.0078, 0.1934) -- ( 0.0020, 0.1675) -- (-0.0206, 0.1726) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.7613, 0.0272) -- ( 0.7841, 0.0313) -- ( 0.7889, 0.0050) -- ( 0.7661, 0.0009) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.7613, 0.0272) -- ( 0.7841, 0.0313) -- ( 0.7889, 0.0050) -- ( 0.7661, 0.0009) --cycle;
\path[hex/terrain/town/house,pic actions] ( 0.0160, 0.0427)
hex/terrain/mountain

This is an example of a terrain picture.

\tikzset{
  \hex/terrain/mountain/.pic={
    \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;
  }
}

hex/terrain/tree

\tikzset{
  \hex/terrain/tree/.pic={
    \path[draw,very thick,pic actions]
    (-.15,.0)
    arc (269:135:.1)
    arc (215: 90:.1)
    arc (180: 45:.1)
    arc (135: 0:.1)
    arc ( 90:-45:.1)
  }
}
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.

\begin{verbatim}
\tikzset{
    hex/ridges/.style={
        line cap=round,
        draw=pgfstrokecolor,
        rounded corners=.25cm,
        scale line widths,
        decoration={
            path has corners=true,
            pre=waves,
            post=waves,
            pre length=-.1cm,
            post length=0cm,
            waves,
            radius=.2cm,
            segment length=.2cm,
            decorate}
    }
\end{verbatim}

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

\begin{verbatim}
\newif\ifhex@r@ne
\newif\ifhex@rn
\newif\ifhex@nw
\newif\ifhex@sw
\newif\ifhex@s
\newif\ifhex@se
\end{verbatim}

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

\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,
\end{verbatim}
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.

\def\hexdo@ridges{%
  \edef\hex@r@tmp{[/hex/ridges/.cd,\%
    radius=0.8,\%
    /tikz/every hex ridges/.try,\%
    \hex@ridges}\]
}
\expandafter\scope\hex@r@tmp\
  \hex@dbg{3}{Ridges: '\meaning\hex@ridges', '\meaning\hex@r@tmp'
    "north east' =\ifhex@r@ne yes\else no\fi
    "north' =\ifhex@r@n yes\else no\fi
    "north west' =\ifhex@r@nw yes\else no\fi
    "south west' =\ifhex@r@sw yes\else no\fi
    "south' =\ifhex@r@s yes\else no\fi
    "south east' =\ifhex@r@se yes\else no\fi
    "radius' =\hex@r@r}
  \def\hex@rp{}
  \ifhex@r@ne
    \def\hex@rp{(0:\hex@r@r)--(60:\hex@r@r)}
  \fi
  \ifhex@r@n
    \hex@dbg{4}{Ridge along north edge: '\hex@rp'}
    \ifhex@r@ne\else
      \xdef\hex@rp{(60:\hex@r@r)}
    \fi
    \xdef\hex@rp{\hex@rp --(120:\hex@r@r)}
  \fi
  \ifhex@r@nw
    \hex@dbg{4}{Ridge along northwest edge: '\hex@rp'}
    \ifhex@r@nw\else
      \xdef\hex@rp{(60:\hex@r@r)}
    \fi
    \xdef\hex@rp{\hex@rp --(120:\hex@r@r)}
  \fi
  \ifhex@r@sw
    \hex@dbg{4}{Ridge along southwest edge: '\hex@rp'}
    \ifhex@r@sw\else
      \xdef\hex@rp{(120:\hex@r@r)}
    \fi
    \xdef\hex@rp{\hex@rp --(180:\hex@r@r)}
  \fi
  \ifhex@r@se
    \hex@dbg{4}{Ridge along southeast edge: '\hex@rp'}
    \ifhex@r@se\else
      \xdef\hex@rp{(120:\hex@r@r)}
    \fi
    \xdef\hex@rp{\hex@rp --(180:\hex@r@r)}
  \fi
  \ifhex@r@n\else
    \def\hex@rp{\hex@rp
}
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,
    thin,
    fill=pgfstrokecolor,
    color=pgfstrokecolor},
  hex/town name/.style={
    transform shape,
    shape=rectangle,
    above right=.1,
    color=pgfstrokecolor,
    font=\sffamily\normalsize}
}
```

Next is the namespace for dealing with towns.

```
\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}
}
```
And some pictures for making the towns.

```latex
\tikzset{
  hex/town/village/.pic={\path[fill,pic actions] circle(.1);},
  hex/town/town/.pic={\path[fill,pic actions] circle(.2);},
  hex/town/city/.pic={
    \path[fill,pic actions] circle(.25);
    \path[draw,pic actions] circle(.35);}
}
\hex@do@town

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

```latex
\def\hex@c@nameparse{%
  \@ifnextchar[{{\hex@c@namep@rse}{\hex@c@namep@rse[]}}%]
  \def\hex@c@namep@rse[#1]#2\endhex@c@nameparse{\def\hex@c@node{node[shape=rectangle,hex/town name,#1]{#2}}}
\def\hex@do@town{\edef\hex@c@tmp\[/\path[hex/town/.cd,,town,\path[tikz/hex/town,\path[every hex town/.try,\hex@town}\]
\expandafter\scope\hex@c@tmp\ifx\hex@c@pic\empty\else\@ifundefined{hex@c@pos}{\let\hex@c@pos\empty}{\@ifundefined{hex@c@name}{\let\hex@c@name\empty}{\expandafter\hex@c@nameparse\hex@c@name\endhex@c@nameparse\ifx\hex@c@pos\empty\def\hex@c@pos{(0,0)}\fi\hex@dbg{2}{Town:\text=\hex@c@name\pic=\hex@c@pic\place=\hex@c@pos\node=\hex@c@node}}\filldraw \hex@c@pos pic{\hex@c@pic} \hex@c@node;\fi\endscope}
```

5.4.8 Labels

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

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

```latex
\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.

12821 \def\hex@max@row{-1}
12822 \tikzset{
12823 \max hex row/.store in=\hex@max@row,
12824 }

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.

12825 \tikzset{%
12826 hex/label/.style=({%
12827 draw=none,%
12828 shape=rectangle,%
12829 anchor=north,%
12830 color=gray,%
12831 font=\sffamily\bfseries\scriptsize,%
12832 inner sep=0},%
12833 )

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.

12834 \tikzset{%
12835 /hex/label/.search also={/tikz},
12836 /hex/label/.cd,
12837 none/.code={\global\let\hex@l@text\@empty},
12838 auto/.is choice,
12839 auto/none/.code={\global\let\hex@l@text\@empty},
12840 auto/numbers/.code=({%
12841 \hex@l@abs%
12842 \xdef\hex@l@text{\hex@l@n@pad{\the\c@hex@l@c}}%
12843 \hex@l@n@pad{\the\c@hex@l@r}}),
12844 auto/alpha column/.code=({%
12845 \hex@l@abs%
12846 \let\hex@l@text\@empty%
12847 \ifnum0>\hex@col\AlphAlph{-\hex@col}\else\AlphAlph{\hex@col}\fi%
12848 \hex@l@text},
12849 auto/alpha 2 column/.code=({%
12850 \hex@l@abs%
12851 \ifnum0>\hex@col\AlphAlph{\hex@l@c}\else\AlphAlph{-\hex@col}\fi%
12852 \hex@l@text},
12853 \xdef\hex@l@text{\hex@l@n@pad{\hex@row}}),
12854 auto/inv y x plus 1/.code=({%
12855 \hex@dbg{3}\{Inverse row, add one to column with arg \texttt{#1}\}
12856 \let\hex@l@text\@empty%
12857 \ifnum0>\hex@max@row\hex@max@row\texttt{-}\hex@row
12858 \pgfmathtruncatemacro{\hex@l@row}{\hex@row}
12859 \pgfmathtruncatemacro{\hex@l@col}{1+\hex@col}
12860 \xdef\hex@l@text{\hex@l@n@pad{\hex@l@col}}%
12861 }}

286
\hex@l@n@pad\{\hex@l@row}\}
\else\message{Max row number not set}\fi,
auto/x and y plus 1/.code={%
\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={\gdef\hex@l@text{#1}},
place/.store in=\hex@l@pos,
location/.forward to=/hex/label/place,
rotate/.store in=\hex@l@rot\}
\hex@l@abs
This takes the absolute value of row and column numbers.
\def\hex@l@abs{
\setcounter{hex@l@c}{\hex@col}
\setcounter{hex@l@r}{\hex@row}
\expandafter\ifnum\value{hex@l@c}<0\multiply\c@hex@l@c by-1\fi%
\expandafter\ifnum\value{hex@l@r}<0\multiply\c@hex@l@r by-1\fi%
% \hex@dbg{0}{\hex@col->\the\c@hex@l@c\space\hex@row->\the\c@hex@l@r}
\hex@l@n@pad
This will pad a number with a 0 if the number is smaller than 10.
\long\def\hex@l@n@pad#1{% 
\ifnum#1<10 0\fi%
#1}
\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 \hex@l@text to be non-empty, then we set the label via a \LaTeX\ node.
\def\hex@do@label{%
\hex@dbg{1}{Hex label: \meaning\hex@label}%
\edef\hex@l@tmp{%
\hex@label/.cd,%
\rotate=0,%
\place={(#0:.8)},% 
/tikz/hex/label/.try,%
/tikz/every hex label/.try,%
\hex@l@text}%
\hex@l@n@pad\hex@l@row\}
\hex@l@n@pad\hex@l@col\}
\hex@l@n@pad\hex@l@row\}
\hex@l@n@pad\hex@l@col\}
\hex@l@n@pad\hex@l@row\}
\hex@l@n@pad\hex@l@col\}
\hex@l@n@pad\hex@l@row\}
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.

**hex/fortress**

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

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

**hex/fortress 2**

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

```
\tikzset{
\hex/fortress 2/.pic={
\draw[pic actions,transform shape] (0:0.64)
\foreach \a in {15,45,...,345}{
--(\a:0.64)
--(\a:0.80)
--(\a+15:0.80)
--(\a+15:0.64)
--cycle;}}
```
5.4.10 Some macros

\DeclareRobustCommand\fortmark[1][scale=.25]{\tikz[#1,transform shape]{%
\pic{hex/fortress 2}}}
\providecommand\terrainmark[2][scale=.2]{\tikz[#1]{\hex[label=,terrain=#2]}}
\providecommand\clearhex[1][scale=.2]{\tikz[#1]{\hex[label=}}}
\providecommand\woodshex[1][scale=.2]{\terrainmark[#1]{woods}}
\providecommand\mountainhex[1][scale=.2]{\terrainmark[#1]{mountains}}
\providecommand\cityhex[1][scale=.2]{\terrainmark[#1]{city}}
\providecommand\beachhex[1][scale=.2]{\terrainmark[#1]{beach}}
\providecommand\seahex[1][scale=.2]{\tikz[#1]{\hex[label=,fill=sea]}}
\providecommand\riverhex[1][scale=.2]{\tikz[#1]{\hex[label=,fill=black]}}

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

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

\pgfdeclaredecoration{outline}{init}{
\state{init}[next state=tick,width=0pt]{\xdef\outlinerev{}}
\state{tick}[width=+\pgfdecorationsegmentlength]{
\pgfpathlineto{\pgfpointadd{\pgfpointorigin}{\pgfpointpolar{\pgfdecorationsegmentangle}{\pgfdecorationsegmentamplitude}}}}
\state{final}{%}
\pgf@xa=\pgf@x\pgf@ya=\pgf@y
\message{``\the\pgf@x,\the\pgf@y}\
\pgf@x=\pgf@xa\pgf@y=\pgf@ya\
\message{``\the\pgf@x,\the\pgf@y}\
\foreach \x/\y in \outlinerev{\ifx\x\empty\else\ifx\y\empty\else}
\begin{tikzpicture}
\pgfmathsetmacro{x}{\xa}
\pgfmathsetmacro{y}{\ya}
\pgfmathsetmacro{\nlt}{\lineto{\xa}{\ya}}
\fi
\fi
\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
hex/road/.style={
  rounded corners=3\pgflinewidth,% .25cm,
  color=black,
  transform shape,
  scale line widths,
  thick},
hex/railroad/.style={
  %scale line widths,
  rounded corners=.25cm,
  color=gray!50!black,
  transform shape,
  postaction={draw,decorate},
  decoration={ticks,
    segment length=9\pgflinewidth,
    amplitude=3\pgflinewidth,%.1cm
  },
hex/river/.style={
  color=blue,
  scale line widths,
  line width=3pt,
  transform shape,
  decorate,
  decoration={random steps,
    segment length=.3cm,
    amplitude=.15cm,
    pre=lineto,
    post=lineto,
    pre length=.05cm,
    post length=.05cm},
  rounded corners=.08cm},
hex/border/.style={
  color=gray,
  rounded corners=3pt,
  dashed,
  transform shape,
  scale line widths,
  very thick
},
\end{tikzpicture}
\begin{tikzpicture}
\pgfsetlayers{main,background}
\pgfsetlayer{main}
\begin{scope}
\tikzset{
hex/road/.style={
  rounded corners=3\pgflinewidth,% .25cm,
  color=black,
  transform shape,
  scale line widths,
  thick},
hex/railroad/.style={
  %scale line widths,
  rounded corners=.25cm,
  color=gray!50!black,
  transform shape,
  postaction={draw,decorate},
  decoration={ticks,
    segment length=9\pgflinewidth,
    amplitude=3\pgflinewidth,%.1cm
  },
hex/river/.style={
  color=blue,
  scale line widths,
  line width=3pt,
  transform shape,
  decorate,
  decoration={random steps,
    segment length=.3cm,
    amplitude=.15cm,
    pre=lineto,
    post=lineto,
    pre length=.05cm,
    post length=.05cm},
  rounded corners=.08cm},
hex/border/.style={
  color=gray,
  rounded corners=3pt,
  dashed,
  transform shape,
  scale line widths,
  very thick
},
\end{scope}
\end{tikzpicture}
\road
\railroad
\river
\border

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.
5.4.13 Move, attacks, retreats from hex to hex

\texttt{\textbackslash hex\textbackslash getscale}

Get current scaling factor.

\begin{verbatim}
1061 \def\hex@getscale#1{\% 1062 \begingroup 1063 \pgfgettransformentries{\% 1064 \scaleA}{} 1065 \scaleB}{} 1066 \scaleC}{} 1067 \scaleD}{} 1068 \whatevs}{} 1069 \whatevs}{} 1070 \pgfmathsetmacro{#1}{sqrt(abs(\scaleA*\scaleD-\scaleB*\scaleC))} 1071 \expandafter 1072 \endgroup 1073 \expandafter\def\expandafter#1\expandafter{#1} 1074 }
\end{verbatim}

Key to get the scale

\begin{verbatim}
1075 \tikzset{\% 1076 hex/get scale/.code={ 1077 \hex@getscale(\hex@scale), 1078 } 1079 }
\end{verbatim}

Style for moves. Use like

\begin{verbatim}
\texttt{\textbackslash path\{move\} (coordinates);}
\end{verbatim}

\begin{verbatim}
1080 \tikzset{\% 1081 % Argument is colour 1082 hex/move/.style={ 1083 hex/get scale, 1084 decorate, 1085 decoration={ 1086 mark=between positions 0 and 1 step 0.75*\hex@scale*\hex@dy with { 1087 \node [single arrow, 1088 single arrow head extend=3pt, 1089 fill=#1, 1090 inner sep=\hex@scale*.5mm, 1091 minimum width=\hex@scale*2mm, 1092 minimum height=\hex@scale*\hex@dy/2, 1093 transform shape]{}; 1094 } 1095 }, 1096 }
\end{verbatim}

A short move style

\begin{verbatim}
\texttt{\textbackslash path\{short move\} (coordinates);}
\end{verbatim}
% Argument is colour
hex/short move/.style={
  hex/get scale,
  decorate,
  decoration={
    markings,
    mark=at position 0 and 1 step 0.5*\hex@scale*\hex@dy with {
      \node [single arrow,
      fill=#1,
      single arrow head extend=3pt,
      inner sep=\hex@scale*.5mm,
      minimum width=\hex@scale*2mm,
      minimum height=\hex@scale*\hex@dy/3,
      transform shape]{};
    }
  },
},
A short move style
\path[long move] \langle coordinates \rangle;

% Argument is colour
hex/long move/.style={
  hex/get scale,
  transform shape,
  decorate,
  decoration={
    markings,
    mark=at position 0 and -.7*\hex@scale*\hex@dy step 2*\hex@scale*\hex@dy with {
      \node [single arrow,
      fill=#1,
      single arrow head extend=3pt,
      anchor=west,
      inner sep=\hex@scale*.25mm,
      outer sep=.3*\hex@scale*\hex@dy,
      minimum width=\hex@scale*2mm,
      minimum height=1.4*\hex@scale*\hex@dy,
      transform shape]{};
    }
  },
},
A short move style
\path[move with start] \langle coordinates \rangle;

% Argument is colour
hex/move with start/.style={
  hex/get scale,
  decorate,
  decoration={
    markings,
    mark=at position 0 with {
      };
  },
}
\node [inner sep=0, circle, minimum size=\hex@scale*5mm, fill=#1, transform shape] {};,
mark=between positions 0 and 1 step 0.75*\hex@scale*\hex@dy with {
\node [single arrow, single arrow head extend=\hex@scale*3pt, fill=#1, inner sep=\hex@scale*1mm, minimum width=\hex@scale*3mm, minimum height=\hex@scale*\hex@dy/2, transform shape]();
}
% Default fill colour is black
hex/move/.default=black,
hex/move with start/.default=black,
hex/short move/.default=black,
hex/long move/.default=black,
% Arguments are draw and fill color
A move cost style
\path[move] ...(coordinate)node[hex/move cost] ...;

hex/move cost/.style 2 args={
minimum size=1mm, inner sep=0.1mm, circle, fill=#2, transform shape, text=#1, font=\sffamily\bfseries\Large},
hex/move cost/.default={black}{none},
% Argument is fill colour
A short line style for retreates, advances, and so on
\path[short line] (start)--(end);

hex/short line/.style={%
hex/get scale, inherit options/.code={\csname tikz@options\endcsname}, inherit options, decorate, decoration={
mark=between positions \hex@scale*\hex@dy and 1 step 2*\hex@scale*\hex@dy with {
\node [single arrow,draw=black,fill=#1, single arrow head extend=\hex@scale*3pt, inner sep=1mm, minimum width=0.75*\hex@scale*\hex@dy, mark=between positions 0 and 1 step 0.75*\hex@scale*\hex@dy with {
\node [single arrow,draw=black,fill=#1, single arrow head extend=\hex@scale*3pt, inner sep=1mm, minimum width=0.75*\hex@scale*\hex@dy, minimum height=\hex@scale*\hex@dy/2, transform shape]();
}
% Default fill colour is black
hex/move/.default=black,
hex/move with start/.default=black,
hex/short move/.default=black,
hex/long move/.default=black,
% Arguments are draw and fill color
A move cost style
\path[move] ...(coordinate)node[hex/move cost] ...;

hex/move cost/.style 2 args={
minimum size=1mm, inner sep=0.1mm, circle, fill=#2, transform shape, text=#1, font=\sffamily\bfseries\Large},
hex/move cost/.default={black}{none},
% Argument is fill colour
A short line style for retreates, advances, and so on
\path[short line] (start)--(end);

hex/short line/.style={%
hex/get scale, inherit options/.code={\csname tikz@options\endcsname}, inherit options, decorate, decoration={
mark=between positions \hex@scale*\hex@dy and 1 step 2*\hex@scale*\hex@dy with {
\node [single arrow,draw=black,fill=#1, single arrow head extend=\hex@scale*3pt, inner sep=1mm, minimum width=0.75*\hex@scale*\hex@dy, minimum height=\hex@scale*\hex@dy/2, transform shape]();
}
% Default fill colour is black
hex/move/.default=black,
hex/move with start/.default=black,
hex/short move/.default=black,
hex/long move/.default=black,
% Arguments are draw and fill color
A move cost style
\path[move] ...(coordinate)node[hex/move cost] ...;

hex/move cost/.style 2 args={
minimum size=1mm, inner sep=0.1mm, circle, fill=#2, transform shape, text=#1, font=\sffamily\bfseries\Large},
hex/move cost/.default={black}{none},
% Argument is fill colour
A short line style for retreates, advances, and so on
\path[short line] (start)--(end);
An attack indication style

```
\path[\textcolor{red}{\textbf{attack}}] (start)--(end);
```

Short hands

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

5.4.14 Board clipping and frame

```
\boardframe
```

Define the bounding box around the board

```
\boardframe[\text{(margin)}][\text{(lower=left)}][\text{(upper-right)}]{(margin)}
```
where \((\text{lower-left})\) and \((\text{upper-right})\) specifies the lower left and upper right hexes (inclusive) of the board.

\begin{verbatim}
\tikzset{
  hex/board frame/.style={draw}
}
\def\boardframe{%
\@ifnextchar[{\bo@rdframe}{\bo@rdframe[0]}%}
\def\bo@rdframe[#1](#2)(#3){%
  \hex@coords@conv{#2}
  \edef\llx{\hex@x}
  \edef\lly{\hex@y}
  \edef\llc{\hex@col}
  \edef\llr{\hex@row}
  \edef\llr{\hex@row}
  \edef\ellc{\hex@eff@col}
  \edef\ellr{\hex@eff@row}
  \%
  \def\margin{#1}
  \%
  \hex@dbg{2}{%
  Board Hex range: (\llc,\llr)x(\urc,\urr)
  ^\JEffective range: (\ellc,\ellr)x(\eurc,\eurr)
  ^\JB: (\llx,\lly)x(\urx,\ury)}%
  \ifnum\hexdbglvl>1
    %
    \draw[red,very thick](hex cs:c=\llc,r=\llr) rectangle(hex cs:c=\urc,r=\urr);
    \draw[red,ultra thick,dashed](\llx,\lly) rectangle(\urx,\ury);
    \draw[->,very thick,blue] (0,0) -- (0,1) (0,0) -- (1,0);
  \fi
  \%
  \Calculate how many half hex hides to add to the "bottom"
  \%
  \def\oddeven{isodd}
  \ifnum\hex@coords@row@fac<0\def\oddeven{iseven}\fi%
  \pgfmathparse{
    ifthenelse(\hex@got@bot@short(\ellc),
      ifthenelse(\hex@bot@short@col(\llc)*not(\oddeven(\ellc)),2,
        ifthenelse(\hex@bot@short@col(\llc),0,1)),
      ifthenelse(\oddeven(\ellc),1,2))
  }
  \edef\olly{\pgfmathresult}%
  \hex@dbg{2}{Delta lly: \olly half heights}
  \%
  \Calculate how many half hex heights to add to the "top"
  \%
  \def\oddeven{iseven}
  \ifnum\hex@coords@row@fac<0\def\oddeven{isodd}\fi%
  \pgfmathparse{
    ifthenelse(\hex@got@top@short(\urc),
      ifthenelse(\hex@top@short@col(\urc)*\oddeven(\eurc),0,
        ifthenelse(\hex@top@short@col(\urc),0,1)),
      ifthenelse(\oddeven(\urc),1,2))
  }
  \edef\olly{\pgfmathresult}%
  \hex@dbg{2}{Delta lly: \olly half heights}
  \%
  \Calculate how many half hex heights to add to the "top"
  \%
  \def\oddeven{iseven}
  \ifnum\hex@coords@row@fac<0\def\oddeven{isodd}\fi%
  \%
  \pgfmathparse{
    ifthenelse(\hex@got@top@short(\urc),
      ifthenelse(\hex@top@short@col(\urc)*\oddeven(\eurc),0,
        ifthenelse(\hex@top@short@col(\urc),0,1)),
      ifthenelse(\oddeven(\urc),1,2))
  }
  \edef\olly{\pgfmathresult}%
  \hex@dbg{2}{Delta lly: \olly half heights}
\end{verbatim}
\edef\oury{\pgfmathresult}\
\hex@dbg{2}{\text{Delta ury: } \oury \text{ half heights}}}

\begin{itemize}
\item Calculate new LLY and URY
\item Calculate width and height
\item Clip the board to not show incomplete hexes
\end{itemize}
Show a debug grid. This requires \boardframe.

Some dummy styles. These will be defined by the export class to facilitate getting information from the board.
5.5 The \texttt{wargame.chit} Ti\kern.5pt kZ library

We define the library for making chits. We load the hex Ti\kern.5pt kZ \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

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

The parts of a chit

\begin{verbatim}
\tikzset{\
  /chit/.search also={/tikz},
  /chit/.cd,
  full/.store in=\chit@full, full/.initial=,\%
  symbol/.store in=\chit@symbol, symbol/.initial=,\%
  left/.store in=\chit@left, left/.initial=,\%
  unique/.style={/chit/left={#1}},\%
  right/.store in=\chit@right, right/.initial=,\%
}\end{verbatim}
Parent/.style={/chit/right={#1}},%
upper left/.store in=\chit@upper@left, upper left/.initial=,%
upper right/.store in=\chit@upper@right, upper right/.initial=,%
lower left/.store in=\chit@lower@left, lower left/.initial=,%
lower right/.store in=\chit@lower@right, lower right/.initial=,%
factors/.store in=\chit@factors, factors/.initial=,%
setup/.store in=\chit@setup, setup/.initial=,%
id/.store in=\chit@id, id/.initial=,%
frame/.store in=\chit@frame, frame/.initial=,%

\tikzset{
\chit/.code={
\pgfkeys{/tikz/transform shape,/tikz/shape=chit}.
}\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},
}\tikzset{
\\

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}\\

301
We define a counter to set-up unique names for chit nodes.
\newcounter{chit@id}\setcounter{chit@id}{0}

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

\def\chit@n@to#1#2{\% Without a following start square bracket '\[' by-pass to final
\chit@dbg{4}{Chit NATO App6(c) first step '#1' '#2'}\@ifnextchar[\@chit@n@to{#1}{#2}}{\@chit@n@to@{#1}{#2}}\%]}

The following macro is called if we had no options.
\def\@chit@n@to@#1#2#3\@end@chit@n@to{\chit@dbg{4}{Chit NATO App6(c) w/o offset: \^J Options: #3 \^J ID: #1 \^J Position: #2}\node[chit/symbol,natoapp6c={#3,id=#1}] (#1) at (#2) {};

This is called if we had an option-like argument. Check if we have an offset
\def\@chit@n@to#1#2\[#3\]{\chit@dbg{4}{Chit NATO App6(c) second step '#1' '#2' '#3'}\@ifnextchar({\@@chit@n@to{#1}{#2}{#3}}{\@@chit@n@to{#1}{#2}{#3}(0,0)}%)

This called if we had option-like argument.
\def\@@chit@n@to#1#2#3(#4)\@end@chit@n@to{\chit@dbg{4}{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@trans@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.

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).

\pgfdeclareshape{chit}{\savedanchor\center{\pgf@x=0cm\pgf@y=0cm}\savedanchor\northeast{\pgf@x=0.6cm\pgf@y=0.6cm}\savedanchor\symbol{\pgf@x=0cm\pgf@y=0.2cm}\savedanchor\factors{\pgf@x=0cm\pgf@y=-0.5cm}\saveddimen\margin{\pgf@x=0.04cm}

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

\savedmacro\id{\chit@dbg{4}{Chit ID: \meaning\chit@id}\@ifundefined{chit@id}{\let\chit@id\pgfutil@empty}{}\ifx\chit@id\pgfutil@empty\wg@r@ndom@id\edef\id{chit\wg@uuid}\else\edef\id{\chit@id}\fi\chit@dbg{4}{Chit ID stored: \meaning\chit@id}}\savedmacro\chitframeopt{\let\chitframeopt\pgfutil@empty\@ifundefined{chit@frame}{}{\edef\chitframeopt{\chit@frame}}\n@to@pp@dbg{3}{Chit Frame options: \meaning\chitframeopt}}

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⟩}.

```latex
\anchor{symbol north east}{\chitanchor{M\id symbol}{north east}{0.2cm}}
\anchor{symbol north west}{\chitanchor{M\id symbol}{north west}{0.2cm}}
\anchor{symbol south east}{\chitanchor{M\id symbol}{south east}{0.2cm}}
\anchor{symbol south west}{\chitanchor{M\id symbol}{south west}{0.2cm}}
\anchor{symbol north}{\chitanchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol west}{\chitanchor{M\id symbol}{west}{0.2cm}}
\anchor{symbol south}{\chitanchor{M\id symbol}{south}{0.2cm}}
\anchor{symbol east}{\chitanchor{M\id symbol}{east}{0.2cm}}
\anchor{symbol upper}{\chitanchor{M\id symbol}{upper}{0.2cm}}
\anchor{symbol lower}{\chitanchor{M\id symbol}{lower}{0.2cm}}
\anchor{symbol echelon}{\chitanchor{M\id symbol}{north}{0.2cm}}
\anchor{symbol below}{\chitanchor{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.

```latex
\anchor{symbol} {⟨symbol⟩}
\anchor{factors} {⟨factors⟩}
\anchor{left} {\chitanchor{M\id symbol}{west}{.2cm}\advance\pgf@x-\margin}
\anchor{right} {\chitanchor{M\id symbol}{east}{.2cm}\advance\pgf@x+\margin}
\anchor{upper right} {\northeast\advance\pgf@x-\margin\advance\pgf@y-\margin}
\anchor{upper left} {\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{lower right} {\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{lower left} {\northeast\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.

```latex
\backgroundpath{
\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y
}%
```

%% This is the outline of the chit only. The rest of the chit is
%% made on the foreground "path".
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.

```latex
\behindforegroundpath{%
\chit@dbg{4}{%
Chit foreground: \meaning\id
^^J ID (set): \meaning\chit@id
^^J Symbol: \meaning\chit@symbol
^^J Full: \meaning\chit@full
^^J Factors: \meaning\chit@factors
^^J Left: \meaning\chit@left
^^J Right: \meaning\chit@right
^^J Upper left: \meaning\chit@upper@left
^^J Lower left: \meaning\chit@lower@left
^^J Upper right: \meaning\chit@upper@right
^^J Lower right: \meaning\chit@lower@right}
\chit@report{}
\pgfscope

\northeast%
\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfclosepath
\pgfusepath{clip}
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}{}{%
\center\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfclosepath
\chit\full
\clip}
\edef\symid{\id symbol}%
\symbol%
\edef\args{{\symid}{\the\pgf@x,\the\pgf@y}\chit@symbol}%
\chit@dbg{6}{Arguments to chit NATO symbol: \meaning\args}%
\expandafter\chit@n@to\args\@end@chit@n@to%
\}%
\percentage
\else
\edef\symid{\id symbol}%
\symbol%
\edef\args{\symid}{\the\pgf@x,\the\pgf@y}\chit\symbol%
\chit@dbg{6}{Arguments to chit NATO symbol: \meaning\args}%
\expandafter\chit@n@to\args\@end@chit@n@to%
\}%
\fi
\%
\northeast%
\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
\pgfclosepath
\pgfclosepath
\pgfusepath{clip}
```

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

Otherwise, we put in a node with shape natoapp6c and pass the symbol key–value pairs as options.
Having made the NATO App6(c) symbol, which we gave the node name \( \langle \text{id} \rangle \text{symbol} \) 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.
5.5.5 The \chit wrapper macro

\chit
\chit@
\chit@@

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

\chit[(chit options)]((position))((identifier));

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

The first macro parses for options.
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 TiKZ macros.

\def\chit@@@#1#2(#3){% 
\chit@dbg{4}{ Chit 1 factor: #1}% 
\node[chit/factor,chit/1 factor,pic actions]{#1};}% 
\tikzset{% 
chit/1 factor/.pic={ 
\node[chitdbg]{Chit 1 factor: #1};}% 
\node[chitdbg]{Chit 1 factor: #1} (tmp) at (#2){};% 
\pgfnoderename{#3}{tmp}%;% 
\fi%; 
\chit@dbg;\@gobble}%; 

5.5.6 Predefined chit element pictures

These pictures can be used as the value of chit keys.
13713 pics/chit/2 factors/.style args={#1,#2}{%
13714 code={%
13715 \chit@dbg{4}{ Chit 2 factors: #1 and #2}%
13716 \node[chit/factor,chit/2 factors,pic actions]{#1--#2};}},
13717 pics/chit/2 factors artillery/.style args={#1,#2,#3}{%
13718 code={%
13719 \chit@dbg{4}{ Chit 2 factors w/artillery: '#1' '#2' '#3'}%
13720 \node[chit/factor,chit/2 factors,pic actions]{%
13721 #1\overset{\text{\scriptsize #3}}{\text{--}}#2};}},
13722 pics/chit/3 factors/.style args={#1,#2,#3}{%
13723 code={%
13724 \chit@dbg{4}{ Chit 3 factors: '#1' '#2' '#3'}%
13725 \node[chit/factor,chit/3 factors,pic actions]{#1-#2-#3};}},
13726 pics/chit/4 factors/.style args={#1,#2,#3,#4}{%
13727 code={%
13728 \chit@dbg{4}{ Chit 4 factors: '#1' '#2' '#3' '#4'}%
13729 \node[chit/factor,chit/4 factors,pic actions]{#1-#2-#3-#4};}},
13730 chit/identifier/.pic={%
13731 \chit@dbg{4}{ Chit identifier: '#1'}%
13732 \node[chit/identifier,pic actions]{#1};
13733 },
13734 chit/small identifier/.pic={%
13735 \chit@dbg{4}{ Chit small identifier: '#1'}%
13736 \node[chit/small identifier,pic actions]{#1};
13737 },
13738 chit/identifier macro/.pic={%
13739 \chit@dbg{4}{ Chit identifier macro: \meaning#1}
13740 \edef\chit@i@tmp{#1}
13741 \node[chit/identifier,pic actions]{\chit@i@tmp};
13742 }

/tikz/chit/factor
/tikz/chit/1 factor
/tikz/chit/2 factors
/tikz/chit/3 factors
/tikz/chit/4 factors
/tikz/chit/identifier
/tikz/chit/small identifier

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

13743 \tikzset{%
13744 chit/factor/.style={
13745 shape=rectangle,
13746 font=\sffamily\bfseries\large,
13747 anchor=base,
13748 inner sep=0,
13749 text=pgfstrokecolor,
13750 draw=none,
13751 fill=none,
13752 },
13753 chit/1 factor/.style={},

309
5.5.7 Other pictures

Pictures for frame, factors, left, right, and below.

\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,
}
\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.

\begin{verbatim}
\def\chit@stack@dir{(.3,.3)}
\end{verbatim}

Now the code

\begin{verbatim}
\def\chit@stack@dir{(.3,.3)}
\end{verbatim}
5.5.9 Making order of battle charts

Macros for making OOBs

Style for turns

```latex
\tikzset{
\chit/oob turn/.pic={\node[pic actions]{#1};}}
```

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

```latex
\def\chit/oob@rowupdate(#1,#2)#3#4{\%
  \chit@dbg{2}{Row update c='#1',r='#2',s='#3',e='#4'}\%
}
```

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

chit list, n-colls, cell size, extra vertical spacing
This expects a list of lists of chits, one list per turn; the maximum number of columns; the size of cells, extra spacing between turns.
Note, the list of lists leaf elements should be styles for the chits.
This depends on the Tikz pic chit/oob turn which takes the number as argument.
\newif\ifwg@oob@inv\wg@oob@inftalse
\def\wg@star@oob{\wg@oob@inftalse\wg@oob}
\def\wg@nostar@oob{\wg@oob@inftalse\wg@oob}
\def\oob{%
@ifstar{\wg@star@oob}{\wg@nostar@oob}%
}%
\def\wg@oob#1#2#3#4{
\def\r{0}
\chit@dbg{1}{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{1}{Turn \ti\space(#r,#t,y=#y):’}
ifwg@oob@inftalse%
\pic at (.5,\r) {chit/oob turn=\ti};%
else%
\pic at (-.5,\r) {chit/oob turn=\ti};%
}\fi%
\ifx\t\empty\else%
foreach \u/\m in \t{
% \chit@dbg{2}{ ‘\u’=‘\m’}
\ifx\u\empty\else
\ifx\m\empty{\def\m{1}}\fi
\ifx\u\empty{\def\m{1}}\fi
\foreach \n in {1,...,\m}{%
\ifx\u\chit@blank\else

312
\chit\{u=\ti,zone oob point={\u}{\c}{\r}}\(\c,\r\);
\fi
\chit@oob@cellupdate(\c,\r){\#2}{\#3}{\y}
}
\fi

5.5.10 Table of chits
\tikzset{
  chit/cell background/.style={fill=black},
  blank chit/.style={/chit/frame={draw=none,fill=none}},
}

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.

The stared version (\chits*) of this macro continues the previously set chit table.
\doublechits
@doublechits
\chit@dbl@cellupdate
\chit@dbl@flip

1. coordinates
2. coordinates
3. cell-size

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

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

\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 (\doublechits*) of this macro continues the previously set chit table.

\def\doublechits{% 
  \@ifstar{\chits@resetfalse@doublechits}{\chits@resettrue@doublechits}\}
\def\@doublechits#1#2#3#4{% 
  \chit@dbg{1}{Setting double-sided chits: #1}% 
  \ifchits@reset% 
    \pgfmathparse{-(#2-.5)*#3} 
    \xdef\c{\pgfmathresult}% 
  \else% 
    \pgfmathparse{-(#2-.5)*#3} 
    \xdef\c{\pgfmathresult}% 
  \fi% 
\}
5.5.11 Some utilities

Game turn marker

\tikzset{
  \number chit/.pic={
    \node [shape=rectangle, font=\sffamily\bfseries Large]{% 
      \begin{tabular}{c} #1 \end{tabular}};},
  \game turn/.pic={
    \node [shape=rectangle, font=\sffamily\bfseries]{% 
      \begin{tabular}{c} Game/\textbackslash Turn \end{tabular}};},
  \game turn/.style={
    /chit/full={\game turn},
    \color=black,
    fill=white},
  \game turn flipped/.style={\game turn},
  \dummy chit/.style={fill=white},
}

Marks of chits

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

\tikzset{
  wg stacking/.style={fill=white,
  /chit/symbol={\{faction=friendly,command=land\}}},
}\DeclareRobustCommand\stackmark[1][1]{%
  \tikz[baseline=(current bounding box.center),scale=.3,#1]{%\
    \stackchits(0,0)(.3,-.3){%\
      \chit[wg stacking],%\
      \chit[wg stacking],%\
      \chit[wg stacking]}}}}

ZOC mark

\DeclareRobustCommand\zocmark[1][1]{%
  \tikz[baseline=(current bounding box.center),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)\%\
    \hex[label=,fill=white](c=1,r=0)\%\
    \hex[label=,fill=white](c=0,r=0)\%\
    \hex[label=,fill=white](c=0,r=1)\%\
    \hex[label=,fill=white](c=2,r=1)\%\
    \hex[label=,fill=white](c=2,r=0)\%
  \end{scope}}}}

\tikzset{
  zone point/.code n args={3}{},
  zone oob point/.code n args={3}{}}

5.6 The wargame.natoapp6c Ti\kZ library

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

5.6.1 Debugging

\usetikzlibrary{wargame.util}\usetikzlibrary{calc}\usetikzlibrary{arrows.meta}\usetikzlibrary{shapes.symbols}\usetikzlibrary{positioning,intersections}

Set the debug level, and make debug message.

\natoappdbglvl@to@pp@dbg

317
5.6.2 Colours

\c@friendly
\c@hostile
\c@neutral
\c@unknown

Define standard colours for marker affiliations.

<table>
<thead>
<tr>
<th>Name</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>friendly</td>
<td>light blue</td>
</tr>
<tr>
<td>hostile</td>
<td>red</td>
</tr>
<tr>
<td>neutral</td>
<td>green</td>
</tr>
<tr>
<td>unknown</td>
<td>yellow</td>
</tr>
</tbody>
</table>

\definecolor{friendly}{RGB}{128, 224, 255}
\definecolor{hostile}{RGB}{255, 128, 128}
\definecolor{neutral}{RGB}{170, 255, 170}
\definecolor{unknown}{RGB}{255, 255, 128}
\tikzset{faction/.code={\ifnum\natoapp@fac<1\relax\else\message{^^J\natoapp@fac}\fi}}

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

\newdimen\n@to@pp@inst@x\n@to@pp@inst@x=0.2cm
\newdimen\n@to@pp@inst@h\n@to@pp@inst@h=0.15cm
\newdimen\n@to@pp@act@w\n@to@pp@act@w=0.15cm
\newdimen\n@to@pp@space@h\n@to@pp@space@h=0.1cm
\newdimen\n@to@pp@r\n@to@pp@r=0.5cm
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.

\n@to@pp@stroke@to@fill
\n@to@pp@restore@fill

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.
We also make an environment, just to simplify the use

\begin{newenvironment}{\nupto\strokeofill}{\pgfscope\nupto\strokeofill\pgfendscope}{\nupto\restoreoffill%}

5.6.5 Faction names as macros

\def\upto\friendly{friendly} \def\upto\hostile{hostile} \def\upto\neutral{neutral} \def\upto\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.

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

\begin{tikzpicture}
\node at (0,0) [circle, draw] {natoapp6c base};
\end{tikzpicture}

\pgfdeclareshape{natoapp6c base}{
\saveddimen\radius{\pgf@x=\upto\r}
\saveddimen\liney{.2cm}
\saveddimen\linex{.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}{-\radius\pgf@y=-\radius}
\anchor{north west}{-\radius\pgf@y=\radius}
\anchor{south east}{\radius\pgf@y=-\radius}
\anchor{south}{0cm-\radius}
\anchor{north}{0cm\radius}
\anchor{west}{-\radius0cm}
\anchor{east}{\radius0cm}
\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{\octagon}
\pgfpathmoveto{\pgfqpointpolar{0}\radius} %
\pgfpathlineto{\pgfqpointpolar{45}\radius}
\endpgfonlayer{Q}
\end{tikzpicture}
5.6.7 ‘Friendly’ node shapes

Macro for friendly air shape
\begin{verbatim}
\def\n@to@friendly@@ir{\
\southeast \wg@tmpa=\pgf@x\wg@tmpb=\pgf@y% 
\pgfpathmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}% 
\cntrl \wg@tmpb=\pgf@y% 
\pgfpatharc{180}{0}{\wg@tmpa and \wg@tmpb}}% 
\pgfusepath{stroke}\topline% 
\pgfusepath{stroke}\bottomline% 
\end{verbatim}

The friendly air command.
\begin{verbatim}
\pgfdeclareshape{natoapp6c friendly air}{% 
\inheritsavedanchors[from=natoapp6c base]% 
\savedanchor\southeast{\pgf@x=1.1\n@to@pp\r} 
\savedanchor\cntrl{\pgf@x=0cm\pgf@y=2.6\n@to@pp\r} 
\savedanchor\north{\pgf@x=0cm\pgf@y=1.6\n@to@pp\r} 
\anchor{south east}{\southeast} 
\end{verbatim}
Macro for friendly land command

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

The friendly land command. The most used command frame.

\pgfdeclare SHAPE{natoapp6c friendly land}{\inherit saved anchors [from=natoapp6c base]}

322
The friendly activity command. Similar to land command, but with boxes in the corners.
The friendly equipment command. A circle.

\begin{natoapp6c friendly equipment}{}% 
\pgfdeclareshape{natoapp6c friendly equipment}{% 
  inheritsavedanchors[from=natoapp6c base] 
  \savedanchor\northeast{% 
    \pgf@x=\n@to@pp\r% 
    \pgf@y=\n@to@pp\r} 
  anchor{north east}{\northeast} 
  anchor{north west}{\northeast\pgf@x=-\pgf@x} 
  anchor{south east}{\northeast\pgf@y=-\pgf@y} 
}
The friendly installation command. Similar to the land command, but with a ‘hat’ on top.
The friendly sea surface command. Same as equipment command.

\pgfdeclaremessage{natoapp6c friendly sea surface}{%}
\inherit\savedmessage{from=natoapp6c friendly equipment}
\inherit\anchor{inner north east}{from=natoapp6c friendly equipment}
\inherit\anchor{inner north west}{from=natoapp6c friendly equipment}
\inherit\anchor{inner south west}{from=natoapp6c friendly equipment}
\inherit\anchor{inner south east}{from=natoapp6c friendly equipment}
\inherit\anchor{north east}{from=natoapp6c friendly equipment}
\inherit\anchor{north west}{from=natoapp6c friendly equipment}
\inherit\anchor{south east}{from=natoapp6c friendly equipment}
\inherit\anchor{south west}{from=natoapp6c friendly equipment}
\inherit\anchor{north}{from=natoapp6c friendly equipment}
\inherit\anchor{west}{from=natoapp6c friendly equipment}
\inherit\anchor{east}{from=natoapp6c friendly equipment}
\inherit\anchor{south}{from=natoapp6c friendly equipment}
\inherit\anchor{upper}{from=natoapp6c friendly equipment}
\inherit\anchor{lower}{from=natoapp6c friendly equipment}
\inherit\behindforegroundpath{from=natoapp6c friendly equipment}
\inherit\backgroundpath{from=natoapp6c friendly equipment}
\inherit\behindforegroundpath{from=natoapp6c friendly equipment}
\inherit\backgroundpath{from=natoapp6c friendly equipment}

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

\pgfdeclaremessage{natoapp6c friendly space}{%}
\inherit\savedmessage{from=natoapp6c friendly space}
\inherit\anchor{north east}{from=natoapp6c friendly space}
Macro for friendly sub surface command

\def\n@to@friendly@sub{\northeast \wg@tmpa=\pgf@x \wg@tmpb=\pgf@y
\pgfpathmoveto{\pgfpoint{\n@to@pp@r}{\wg@tmpa}}{\wg@tmpa}}
\pgfpathlineto{\pgfpoint{-\n@to@pp@r}{\wg@tmpa}}{\wg@tmpa} 
\pgfpathlineto{\pgfpoint{-\n@to@pp@r}{\wg@tmpb}}{\wg@tmpb} 
\pgfpathlineto{\pgfpoint{\n@to@pp@r}{\wg@tmpb}}{\wg@tmpb} 
\pgfclosepath
\pgfusepath{fill}
}

The friendly sub surface command.

\pgfdeclareshape{natoapp6c friendly sub surface}{% 
\inheritss{from=natoapp6c base}
\savedanchor{\northeast}{\pgf@x=1.1\n@to@pp@r}
\savedanchor{\cntrl}{\pgf@x=0cm \pgf@y=-2.6\n@to@pp@r}
}
5.6.8 ‘Hostile’ node shapes

The hostile air command

Macro for hostile air shape

```latex
\def\mtohostile@ir{\pgfpathmoveto{\pgfqpoint{-\pgf@x}{\pgf@y}}\pgfpathlineto{\pgfqpoint{-\pgf@x}{\pgf@y}}}
```

```latex
\begin{tikzpicture}
  \node[hostile air] at (0,0) {natoapp6c hostile air};
\end{tikzpicture}
```
The hostile air command.

\pgfdeclareshape{natoapp6c hostile air}{
\inheritsavedanchors[from=natoapp6c base]
\savedanchor\southeast{\pgf@x=\n@to@pp@r\pgf@y=-\n@to@pp@r}
\savedanchor\cntrl{\pgf@x=\n@to@pp@r\pgf@y=0.414\n@to@pp@r% (sqrt(2)-1)}
\savedanchor\north{\pgf@x=0cm\pgf@y=1.414\n@to@pp@r}
\anchor{south east}{\southeast}
\anchor{south west}{\southeast\pgf@x=-\pgf@x}
\anchor{north east}{\southeast\wg@tmpa=\pgf@x\north\pgf@x=\wg@tmpa}
\anchor{north west}{\southeast\wg@tmpa=\pgf@x\north\pgf@x=-\wg@tmpa}
\anchor{north}{\north}
\anchor{east}{\north\wg@tmpb\pgf@y\southeast\wg@tmpc=\pgf@y\wg@tmpa=\pgf@x\advance\wg@tmpb-\wg@tmpc\divide\wg@tmpb2\advance\wg@tmpb\wg@tmpc\pgf@x=\wg@tmpa\pgf@y=\wg@tmpb}
\anchor{west}{\north\wg@tmpb\pgf@y\southeast\wg@tmpc=\pgf@y\wg@tmpa=\pgf@x\advance\wg@tmpb-\wg@tmpc\divide\wg@tmpb2\advance\wg@tmpb\wg@tmpc\pgf@x=-\wg@tmpa\pgf@y=\wg@tmpb}
\anchor{south}{\southeast\pgf@x=0cm}
\inheritanchor[from=natoapp6c base]{upper}
\inheritanchor[from=natoapp6c base]{lower}
\inheritanchor[from=natoapp6c base]{left}
\inheritanchor[from=natoapp6c base]{right}
\inheritanchor[from=natoapp6c base]{center}
\backgroundpath{\n@to@hostile@@ir}
\behindforegroundpath{\n@to@hostile@@ir}
pgfsetfillpattern{stroke}
\pgfusepath{stroke}
}
Macro for hostile land command

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

The hostile land command.

\pgfdeclareshape{natoapp6c hostile land}{
\inherit\saved\northeast {%}
\anchor{north east}{\northeast}
\anchor{north west}{\northeast\pgf@x=-\pgf@x}
\anchor{south east}{\northeast\pgf@y=-\pgf@y}
\anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{north}{\northeast\pgf@x=0cm}
\anchor{south}{\northeast\pgf@x=0cm\pgf@y=-\pgf@y}
\anchor{east}{\northeast\pgf@y=0cm}
\anchor{west}{\northeast\pgf@x=-\pgf@x\pgf@y=0cm}
\inheritanchor[from=natoapp6c hostile land]{upper}
\inheritanchor[from=natoapp6c hostile land]{lower}
\inheritanchor[from=natoapp6c hostile land]{left}
\inheritanchor[from=natoapp6c hostile land]{right}
\inheritanchor[from=natoapp6c hostile land]{center}
\backgroundpath{\n@to@hostile@land%}
\behindforegroundpath{\n@to@hostile@land%
\pgfusepath{stroke}%
}
}

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

\pgfdeclareshape{natoapp6c hostile activity}{
\inherit\saved\northeast {%}
\anchor{north east}{\northeast}
\anchor{north west}{\northeast\pgf@x=-\pgf@x}
\anchor{south east}{\northeast\pgf@y=-\pgf@y}
\anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
\anchor{north}{\northeast\pgf@x=0cm}
\anchor{south}{\northeast\pgf@x=0cm\pgf@y=-\pgf@y}
\anchor{east}{\northeast\pgf@y=0cm}
\anchor{west}{\northeast\pgf@x=-\pgf@x\pgf@y=0cm}
\inheritanchor[from=natoapp6c hostile activity]{upper}
\inheritanchor[from=natoapp6c hostile activity]{lower}
\inheritanchor[from=natoapp6c hostile activity]{left}
\inheritanchor[from=natoapp6c hostile activity]{right}
\inheritanchor[from=natoapp6c hostile activity]{center}
\backgroundpath{\n@to@hostile@land%}
\behindforegroundpath{\n@to@hostile@land%
\pgfusepath{stroke}%
}
}
natoapp6c hostile equipment

The hostile equipment command. Same as land command.

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

```latex
\pgfdeclareshape{natoapp6c hostile sea surface}{%
  \inheritsavedanchors[from=natoapp6c hostile equipment]
  \inheritanchor[from=natoapp6c hostile equipment]{inner north east}
  \begin{pgfuseshape}
    \n@to@pp@inst@x=0.05cm
    \n@to@pp@inst@y=0.05cm
    \pgf@x=2.0cm
    \pgf@y=2.0cm
    \n@to@pp@inst@h=\n@to@pp@inst@x
    \n@to@pp@inst@w=\n@to@pp@inst@x
    \n@to@pp@inst@y=\n@to@pp@inst@h
    \n@to@pp@inst@r=\n@to@pp@inst@w
    \pgf@xa=\pgf@x
    \pgf@ya=\pgf@y
    \pgf@xa=-\pgf@xa
    \pgf@ya=-\pgf@ya
    \advance\pgf@xa-\n@to@pp@inst@x
    \advance\pgf@ya-\n@to@pp@inst@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@ya+\pgf@y}}
    \pgfpathlineto{\pgfqpoint{-\pgf@xa}{\pgf@ya+\pgf@y}}
    \pgfpathlineto{\pgfqpoint{-\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
    \pgfclosepath
    \pgfusepath{stroke}
  \end{pgfuseshape}
```
The hostile space command. Similar to air command, but with bar on top.
Macro for hostile sub surface command

```
\def\n@to@hostile@sub{\northeast \wg@tmpa=\pgf@x \wg@tmpb=\pgf@y
\cntrl \wg@tmpc=\pgf@y
\south \wg@tmpd=\pgf@y
\pgfpathmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpc}}
\pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpd}}
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpc}}
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}
}\natoapp6c hostile sub surface
```

The hostile sub surface command

```
\pgfdeclareshape{natoapp6c hostile sub surface}{
\inheritsavedanchors[from=natoapp6c base]
\savedanchor\northeast{\pgf@x=\n@to@pp@r\pgf@y=\n@to@pp@r}
\savedanchor\cntrl{\pgf@x=\n@to@pp@r\pgf@y=-0.414\n@to@pp@r}
\savedanchor\south{\pgf@x=0cm\pgf@y=-1.414\n@to@pp@r}
\anchor{north east}{\northeast}
\anchor{north west}{\northeast\wg@tmpa=\pgf@x\south\wg@tmpa=\pgf@x}
\anchor{south east}{\northeast\wg@tmpa=\pgf@x\south\wg@tmpa=-\pgf@x}
\anchor{south west}{\northeast\wg@tmpa=\pgf@x\south\wg@tmpa=-\pgf@x}
\anchor{south}{\south}
\anchor{east}{\northeast\wg@tmpb=\pgf@y\wg@tmpa=\pgf@x}
\south\wg@tmpc\pgf@y%}
\advance\wg@tmpb\pgf@y\pgf@x%}
\divide\wg@tmpb\pgf@y%}
\pgf@y=\wg@tmpb%}
\pgf@y=\wg@tmpb%}
\anchor{west}{\northeast\wg@tmpb=\pgf@y\wg@tmpa=\pgf@x%}
\south\wg@tmpc\pgf@y%}
\divide\wg@tmpb\pgf@y%}
\advance\wg@tmpb\pgf@y%}
```

335
5.6.9 ‘Neutral’ node shapes

Macro for neutral shapes

\def\n@to@pp@neutr@l@init{%
    \northeast\wg@tmpa=\pgf@x\wg@tmpb=\pgf@y%
    \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@northwest{\pgfmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}%
    \def\n@to@pp@neutr@l@northeast{\pgfmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}%
    \def\n@to@pp@neutr@l@north{\pgfmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}%
    \def\n@to@pp@neutr@l@southwest{\pgfmoveto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}}%
    \def\n@to@pp@neutr@l@south{\pgfmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}}%
\}

\pgfdeclareshape{natoapp6c neutral air}{
    \inheritsavedanchors[from=natoapp6c base]
    \savedanchor{northeast}{\northeast}
    \anchor{north east}{\northeast}
    \anchor{north west}{\northeast\pgf@x=-\pgf@x}
    \anchor{south east}{\northeast\pgf@y=-\pgf@y}
    \anchor{south west}{\northeast\pgf@x=-\pgf@x\pgf@y=-\pgf@y}
    \anchor{north}{\northeast\pgf@x=0cm}
    \anchor{east}{\northeast\pgf@y=0cm}
    \anchor{west}{\northeast\pgf@x=-\pgf@x\pgf@y=0cm}
    \anchor{south}{\northeast\pgf@x=0cm\pgf@y=-\pgf@y}
}

The neutral node shapes
The neutral land command

\pgfdeclarecoordinatesystem{natoapp6c neutral land}{%
\inheritedanchor[from=natoapp6c neutral air]{north east}
\inheritedanchor[from=natoapp6c neutral air]{north west}
\inheritedanchor[from=natoapp6c neutral air]{south east}
\inheritedanchor[from=natoapp6c neutral air]{south west}
\inheritedanchor[from=natoapp6c neutral air]{north}
\inheritedanchor[from=natoapp6c neutral air]{west}
\inheritedanchor[from=natoapp6c neutral air]{east}
\inheritedanchor[from=natoapp6c neutral air]{south}
\inheritedanchor[from=natoapp6c neutral air]{upper}
\inheritedanchor[from=natoapp6c neutral air]{lower}
\inheritedanchor[from=natoapp6c neutral air]{left}
\inheritedanchor[from=natoapp6c neutral air]{center}
\backgroundpath{%
\n@to@pp@neutr@l@init%
\n@to@pp@neutr@l@ne%
\n@to@pp@neutr@l@top%
\n@to@pp@neutr@l@left%
\pgfclosepath%
}
\behindforegroundpath{%
\n@to@pp@neutr@l@init%
\n@to@pp@neutr@l@ne%
\n@to@pp@neutr@l@top%
\n@to@pp@neutr@l@left%
\pgfusepath{stroke}%
}
}
natoapp6c neutral activity

The neutral activity command. Similar to land command but with boxes added in the corners.
The neutral equipment command. Same as land command

\pgfdeclareshape{natoapp6c neutral equipment}{
\inheritsavedanchors[from=natoapp6c neutral land]
\inheritanchor[from=natoapp6c neutral land]{center}
\inheritanchor[from=natoapp6c neutral land]{inner north east}
\inheritanchor[from=natoapp6c neutral land]{inner north west}
\inheritanchor[from=natoapp6c neutral land]{inner south west}
\inheritanchor[from=natoapp6c neutral land]{inner south east}
\inheritanchor[from=natoapp6c neutral land]{north east}
\inheritanchor[from=natoapp6c neutral land]{north west}
\inheritanchor[from=natoapp6c neutral land]{south east}
\inheritanchor[from=natoapp6c neutral land]{south west}
\inheritanchor[from=natoapp6c neutral land]{north}
\inheritanchor[from=natoapp6c neutral land]{west}
\inheritanchor[from=natoapp6c neutral land]{east}
\inheritanchor[from=natoapp6c neutral land]{south}
\inheritanchor[from=natoapp6c neutral land]{upper}
\inheritanchor[from=natoapp6c neutral land]{lower}
\inheritanchor[from=natoapp6c neutral land]{left}
\inheritanchor[from=natoapp6c neutral land]{right}
\inheritanchor[from=natoapp6c neutral land]{center}
\inheritbackgroundpath[from=natoapp6c neutral land]
The neutral installation command. Similar to land command but with a ‘hat’ on top.
natoapp6c neutral sea surface

The neutral sea surface command. Same as land command.

\begin{Verbatim}
\pgfdeclareshape{natoapp6c neutral sea surface}{
  \inheritsavedanchors[from=natoapp6c neutral equipment]
  \inheritanchor[from=natoapp6c neutral equipment]{inner north east}
  \inheritanchor[from=natoapp6c neutral equipment]{inner north west}
  \inheritanchor[from=natoapp6c neutral equipment]{inner south west}
  \inheritanchor[from=natoapp6c neutral equipment]{inner south east}
  \inheritanchor[from=natoapp6c neutral equipment]{north east}
  \inheritanchor[from=natoapp6c neutral equipment]{north west}
  \inheritanchor[from=natoapp6c neutral equipment]{south east}
  \inheritanchor[from=natoapp6c neutral equipment]{south west}
  \inheritanchor[from=natoapp6c neutral equipment]{north}
  \inheritanchor[from=natoapp6c neutral equipment]{west}
  \inheritanchor[from=natoapp6c neutral equipment]{east}
  \inheritanchor[from=natoapp6c neutral equipment]{south}
  \inheritanchor[from=natoapp6c neutral equipment]{upper}
  \inheritanchor[from=natoapp6c neutral equipment]{lower}
  \inheritanchor[from=natoapp6c neutral equipment]{left}
  \inheritanchor[from=natoapp6c neutral equipment]{right}
  \inheritanchor[from=natoapp6c neutral equipment]{center}
  \inheritbackgroundpath[from=natoapp6c neutral equipment]
  \behindforegroundpath{\pgfusepath{use as bounding box}}
}\end{Verbatim}

natoapp6c neutral space

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

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

\begin{tikzpicture}
\node[natoapp6c neutral sub surface]{};
\end{tikzpicture}
5.6.10 ‘Unknown’ node shapes

Macro to define unknown path elements

```latex
\def\n@to@pp@unknown@init{%
  \def\n@to@pp@unknown@top{\innernortheast \wg@tmpa=\pgf@x%\cntrlnortheast \wg@tmpb=\pgf@x%
    \pgfpathcurveto{% \pgfqpoint{\wg@tmpa}{\wg@tmpb}}{% \pgfqpoint{-\wg@tmpa}{\wg@tmpb}}{% \pgfqpoint{-\wg@tmpa}{\wg@tmpa}}}
  \def\n@to@pp@unknown@left{\innernortheast \wg@tmpa=\pgf@x%\cntrlnortheast \wg@tmpb=\pgf@x%
    \pgfpathcurveto{% \pgfqpoint{-\wg@tmpb}{\wg@tmpa}}{% \pgfqpoint{-\wg@tmpb}{-\wg@tmpa}}{% \pgfqpoint{-\wg@tmpa}{-\wg@tmpa}}}
  \def\n@to@pp@unknown@bottom{\innernortheast \wg@tmpa=\pgf@x%\cntrlnortheast \wg@tmpb=\pgf@x%
    \pgfpathcurveto{% \pgfqpoint{-\wg@tmpa}{-\wg@tmpb}}{% \pgfqpoint{\wg@tmpa}{-\wg@tmpb}}{% \pgfqpoint{\wg@tmpa}{-\wg@tmpa}}}
  \def\n@to@pp@unknown@right{\innernortheast \wg@tmpa=\pgf@x%\cntrlnortheast \wg@tmpb=\pgf@x%
    \pgfpathcurveto{% \pgfqpoint{-\wg@tmpb}{-\wg@tmpa}}{% \pgfqpoint{\wg@tmpb}{\wg@tmpa}}{% \pgfqpoint{\wg@tmpa}{\wg@tmpa}}}%
}
\pgfdeclareshape{natoapp6c unknown land}{
  \inheritsavedanchors[from=natoapp6c base]
  \savedanchor\innernortheast{\pgf@x=.7\n@to@pp@r\pgf@y=.7\n@to@pp@r}
}
```

The unknown land command

```latex
\pgfdeclareshape{natoapp6c unknown land}{
  \inheritsavedanchors[from=natoapp6c base]
  \savedanchor\innernortheast{\pgf@x=.7\n@to@pp@r\pgf@y=.7\n@to@pp@r}
}
```
The unknown air command. To consider: Should clipping path extend below the actual symbol to include that part of the base symbol?
The unknown activity command. Similar to land command, but with boxes in the 'corners'.

```latex
\pgfdeclareshape{natoapp6c unknown activity} {
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor[from=natoapp6c unknown land]{inner north east}
\backgroundpath{\pgfusepath{stroke}}
\behindforegroundpath{\pgfusepath{stroke}}
}
```

natoapp6c unknown activity

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

\pgfdeclareshape{natoapp6c unknown activity} {
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor[from=natoapp6c unknown land]{inner north east}
}
The unknown equipment command. Same as land command.

```latex
\pgfdeclarereshape{natoapp6c unknown equipment}{%
\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]
}\end{tikzpicture}
```

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).
\pgfdeclareshape{natoapp6c unknown installation}{
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor{center}{center}
\inheritanchor{inner north east}{inner north east}
\inheritanchor{inner north west}{inner north west}
\inheritanchor{inner south east}{inner south east}
\inheritanchor{inner south west}{inner south west}
\inheritanchor{north east}{north east}
\inheritanchor{north west}{north west}
\inheritanchor{south east}{south east}
\inheritanchor{south west}{south west}
\inheritanchor{north}{north}
\inheritanchor{west}{west}
\inheritanchor{east}{east}
\inheritanchor{south}{south}
\inheritanchor{upper}{upper}
\inheritanchor{lower}{lower}
\inheritanchor{left}{left}
\inheritanchor{right}{right}
\inheritbackgroundpath[from=natoapp6c unknown land]
\behindforegroundpath{\n@to@pp@unknown@init
\innernortheast \wg@tmpa=\pgf@x%
\n@to@pp@unknown@right
\n@to@pp@unknown@top
\n@to@pp@unknown@left
\n@to@pp@unknown@bottom
\pgfusepath{stroke}
\begin{n@to@pp@stroketofill}
\pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpa}}\n@to@pp@unknown@top
\northeast\wg@tmpb=\pgf@y\wg@tmpc=\pgf@y%
\advance\wg@tmpb\n@to@pp@inst@h%
\advance\wg@tmpc-0.05cm%
\advance\wg@tmpc-\n@to@pp@inst@h%
\pgfpathlineto{\pgfqpoint{-\wg@tmpa}{\wg@tmpb}}
\pgfpathlineto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}
\pgfclosepath
\pgfusepath{fill}
\pgfusepath{clip}
\pgfusepath{fill}
The unknown sea surface command. Same as land command

\begin{n@to@pp@stroketofill}
\end{n@to@pp@stroketofill}

\begin{n@to@pp@stroketofill}
\end{n@to@pp@stroketofill}

The unknown sea surface command. Same as land command

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

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

\begin{n@to@pp@stroketofill}
\end{n@to@pp@stroketofill}

\begin{n@to@pp@stroketofill}
\end{n@to@pp@stroketofill}

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

\pgfdeclareshape{natoapp6c unknown sub surface}{
\inheritsavedanchors[from=natoapp6c unknown land]
\inheritanchor[from=natoapp6c unknown land]{inner north east}
\inheritanchor[from=natoapp6c unknown land]{inner north west}
\inheritanchor[from=natoapp6c unknown land]{inner south west}
\inheritanchor[from=natoapp6c unknown land]{inner south east}
\inheritanchor[from=natoapp6c unknown land]{south east}
\inheritanchor[from=natoapp6c unknown land]{south west}
\inheritanchor[from=natoapp6c unknown land]{south}
\inheritanchor[from=natoapp6c unknown land]{west}
\inheritanchor[from=natoapp6c unknown land]{east}
\inheritanchor[from=natoapp6c unknown land]{upper}
\inheritanchor[from=natoapp6c unknown land]{lower}
\inheritanchor[from=natoapp6c unknown land]{left}
\inheritanchor[from=natoapp6c unknown land]{right}
5.6.11 Echelons

Dimensions

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

Paths as macros

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

Pictures

\tikzset{
  pics/natoapp6c/s/echelon/.is choice,
5.6.12 Text on symbols

/tikz/natoapp6c/normal text
/tikz/natoapp6c/squashed text
/tikz/natoapp6c/small text
/tikz/natoapp6c/small squashed text

352
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}},
}
\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
\n@to@pp@text@squashed
\n@to@pp@text@small
\n@to@pp@text@smallsquashed
\end{verbatim}

\textbf{5.6.13 Text natoapp6c namespace}

Here, we set up the key path \texttt{/natoapp6c}

\begin{verbatim}
\def\natoapp6c\report{}
\tikzset{
  /natoapp6c/.search also={/tikz},
  /natoapp6c/.cd,
}
\end{verbatim}

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

\begin{verbatim}
\tikzset{
  /natoapp6c/.cd,
  id/.store in=natoapp@id,
  fac/.store in=natoapp@fac,
  cmd/.store in=natoapp@cmd,
  ech/.store in=natoapp@ech,
}
\end{verbatim}

\section*{natoapp6c/faction}

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

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

\section*{natoapp6c/command}

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

\begin{verbatim}
\tikzset{
  /natoapp6c/.cd,
  command/.is choice,
  command/base/.style={cmd=base},
  command/activity/.style={cmd=activity},
  command/air/.style={cmd=air},
  command/missile/.style={cmd=air},
  command/equipment/.style={cmd=equipment},
  command/installation/.style={cmd=installation},
  command/land/.style={cmd=land},
  command/sea surface/.style={cmd=sea surface},
  command/space/.style={cmd=space},
  command/sub surface/.style={cmd=sub surface},
  command/sub surface/.style={cmd=sub surface},
}\end{verbatim}
natoapp6c/echelon

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


\end{tikzpicture}

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

\begin{tikzpicture}[natoapp6c/.cd,]
    \newif\ifnatoapp@decoy
    \ifnatoapp@decoyfalse\tikzset{natoapp6c/.cd,}
    \tikzset{natoapp@main, natoapp@left, natoapp@right, natoapp@upper, natoapp@lower, natoapp@below,}
\end{tikzpicture}
5.6.14 The natoapp6c styles

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

\tikzset{%
  natoapp6c/.code={%
    \pgfkeys{/tikz/transform shape=/tikz/shape=natoapp6c}
    \pgfkeys{/pgfkeys=/natoapp6c/.cd,#1}}}

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

\newcounter{natoappid}\setcounter{natoappid}{0}
5.6.15 The \natoapp6c shape

We define an \if to allow us to detect if something is rendered below the frame.

15649 \newif\ifn@to@pp@below\n@to@pp@belowfalse%
15650 \newif\ifn@to@pp@mod\n@to@pp@modfalse%

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

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

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

Then we define a number of regular anchors.

The remaining anchors depend on the shape being used. We reference the anchors of the embedded node of the frame.
The next two anchors are a little funny.

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.

If the symbol is empty, then do nothing.

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 \textit{M}. Various elements of the symbol can then refer to this shape to define paths, etc.

\begin{scope}
\pgfinterruptboundingbox
\t\% Clip to shape in scope
\t\% \message{\string"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 \texttt{frame} key-values in a scope, and \textit{then} get the fill colour.

\begin{scope}
\edef\tmp@opt{\[rameopt\]} \expandafter\scope\expandafter\tmp@opt
\begingroup
% Get fill color {possibly from frame key}
\let\tmp@fill\csname\string\color@pgffillcolor\endcsname
\ifx\tmp@fill\relax
\else%
\pgfnode{natoapp6c \frameshape}{center}{\tmp@fill}{\pgfusepath{fill}}%
\fi%
\endscope
\endgroup
\fi%
\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}[natoapp6c/main]
\begin{scope}
\wg@pic@all{natoapp@main}{natoapp6c/s/}{M.center}{natoapp6c/main}%
\end{scope}
\end{scope}

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

\begin{scope}
\n@to@pp@modtrue
\n@to@pp@dbg{2}{NATO App6(c) modifiers}
\end{scope}

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 \textit{M}).

\begin{scope}
\begin{scope}
% Render lowers
\n@to@pp@modtrue
\n@to@pp@dbg{2}{lower elements}
\end{scope}
\end{scope}
That concludes rendering most of the symbol. We have not put in the echelon, below element, or drawn the frame yet. That we will do on the foreground path.

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

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

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

15793  \n@to@pp@dbg{2}{NATO App6(c) inner node 'M\id' ===}
15794  \pgfnode{natoapp6c \frameshape}{center}{}{M\id}\pgfusepath{stroke}

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

15795  \% Put in the echelon
15796  \@ifundefined{natoapp@ech}{}{%
15797  \ifx\natoapp@ech\pgfutil@empty\else%
15798  \def\args{echelon=\natoapp@ech}
15799  \expandafter\wg@pic\args\@endwg@pic%
15800  \natoapp6c/s/\{$(M.north)+(0,1.2*\n@to@pp@e@y)\}$\{natoapp6c/echelon\}
15801  \n@to@pp@false
15802  }

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

15803  \% Put in stuff below main
15804  \@ifundefined{natoapp@below}{}{%
15805  \n@to@pp@belowtrue
15806  \begin{scope}
15807  \wg@pic@all\{natoapp@below\}{natoapp6c/s/}\{$(M.south)+(0,-\n@to@pp@e@y)\}$\{natoapp6c/below\%
15808  \n@to@pp@belowfalse
15809  \end{scope}%
15810  \n@to@pp@belowfalse

If the decoy flag was set, we draw that.

15812  \textbf{\@ifnatoapp@decoy}%
15813  \scope[dash pattern=on 3\pgflinewidth off 2\pgflinewidth]%
15814  \n@to@pp@dbg{1}{Drawing decoy modifier}%
15815  \wg@sub@nchor{\mid\id}{north east}  
15816  \wg@tmpa=\pgf@x%
15817  \wg@tmpb=\pgf@y%
15818  \pgfpathmoveto{\pgfqpoint{\wg@tmpa}{\wg@tmpb}}%
15819  \wg@tmpc=\n@to@pp@e@y cm%
15820  \advance\wg@tmpc\n@to@pp@e@y cm%
15821  \pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpc}}%
15822  \pgfpathlineto{\pgfqpoint{0cm}{\wg@tmpc}}%
15823  \pgfusepath{stroke}%
15824  \endscope%
15825  \n@to@pp@false
15826  \n@to@pp@false
15827  \endscope%
15828  \n@to@pp@false
15829  }
15830  }

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[(natoapp6c options)](⟨position⟩)⟨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

5.6.17 Macros for markings

\natoappmark

A macro for making NATO App6(c) markings.

A macro for making NATO App6(c) markings.
Some specific NATO App6(c) markers.

\DeclareRobustCommand\armouredmark[1]{\natoappmark[#1]{armoured}}
\DeclareRobustCommand\infantrymark[1]{\natoappmark[#1]{infantry}}
\DeclareRobustCommand\artillerymark[1]{{[fill=pgfstrokecolor]artillery}}
\DeclareRobustCommand\combinedmark[1]{\natoappmark[#1]{combined arms}}
\DeclareRobustCommand\pgmark[1]{{armoured,infantry}}
\DeclareRobustCommand\reconnaissancemark[1]{\natoappmark[#1]{reconnaissance}}
\DeclareRobustCommand\corpsmark[1]{{,echelon=corps}}
\DeclareRobustCommand\divisionmark[1]{{,echelon=division}}
\DeclareRobustCommand\brigademark[1]{{,echelon=brigade}}
\DeclareRobustCommand\regimentmark[1]{{,echelon=regiment}}
\DeclareRobustCommand\sofmark[1]{{infantry,text=SOF}}
\DeclareRobustCommand\mountaineermark[1]{{infantry,lower=mountain}}
\DeclareRobustCommand\airbornemark[1]{{infantry,lower=airborne}}
\DeclareRobustCommand\amphibiousmark[1]{{lower=amphibious}}
\DeclareRobustCommand\airassaultmark[1]{{infantry,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
\begin{verbatim}
\def\n@toapp\corps\support#1{
  \ifx\n@to\pp\friendly#1\n@toapp\corps\sup\friendly%
  \else\ifx\n@to\pp\hostile#1\n@toapp\corps\sup\hostile%
  \else\ifx\n@to\pp\neutral#1\n@toapp\corps\sup\neutral%
  \else\ifx\n@to\pp\unknown#1\n@toapp\corps\sup\unknown%
  \fi\fi\fi\fi}
\end{verbatim}

**natoapp6c/s/TBD**

Special placeholder for symbols To Be Done.

\begin{verbatim}
\tikzset{
  natoapp6c/s/TBD/.pic={\n@toapp\text\normal{\color{magenta}TBD};}}
\end{verbatim}

5.6.19 Symbols used when defining weaponry

\begin{verbatim}
\tikzset{
  pics/natoapp6c/s/weapon/.is choice,
  pics/natoapp6c/s/weapon/base/.style={
    code={\path \[pic actions\] (0,-0.2)--(0,.2);}},
  pics/natoapp6c/s/weapon/top/.style={
    code={\path \[pic actions\] (0,.2)--(0,.35);}},
  pics/natoapp6c/s/weapon/bottom/.style={
    code={\path \[pic actions\] (0,-.35)--(0,-.2);}},
  pics/natoapp6c/s/weapon/rifle/.style={
    code={\path \[pic actions\] (0.2, 0.1)--(0, 0.35)--(-0.2,0.1);}},
  pics/natoapp6c/s/weapon/machine gun/.style={
    code={\path \[pic actions\] (0.2, -0.35)--(-0.2, -0.35);}},
  pics/natoapp6c/s/weapon/grenade launcher/.style={
    code={\path \[pic actions\] (0,0) circle (0.1);}},
  pics/natoapp6c/s/weapon/missile launcher/.style={
    code=%
    \path [pic actions] (0.2, 0.15)
    to[out=90,in=90,looseness=1.75] (-0.2, 0.15);}},
  pics/natoapp6c/s/weapon/non lethal/.style={
    code={\path [pic actions] (-2,0.35) -- (2,0.35);}},
  pics/natoapp6c/s/weapon/multi fire/.style={
    code={\path[pic actions] (-2,-.2)--(2,.2) (-2,-.2)--(-2,0.2);}},
  pics/natoapp6c/s/weapon/air defence/.style={
    code=%
    \path[pic actions] (0.2, -0.4)
    to[out=90,in=90,looseness=1.7] (-0.2, -0.4) -- cycle;}},
  pics/natoapp6c/s/weapon/anti tank/.style={
    code={\path[pic actions] (0.2, -0.4)--(0,-0.2)--(-0.2,-0.4);}},
  \end{verbatim}

364
(Weight) class of weapons: light, medium, heavy

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.
\begin{tikzpicture}[natoapp6c/s/above corps support/.pic={% 
\edef\n@toapp@path{\n@toapp@corps@support{\natoapp@fac}}\n\expandafter\path[draw] \n@toapp@path;\n\expandafter\path[draw,xscale=-1] \n@toapp@path;},
\end{tikzpicture}

\begin{tikzpicture}[natoapp6c/s/air assault with organic lift/.pic={% 
\ifx\n@toapp@hostile\natoapp@fac% 
\def\n@toapp@path{(-.75,-.2)--(-.15,-.2)--(0,-.5)--(.15,-.2)--(.75,-.2)}\n\else\n\def\n@toapp@path{(-.75,-.2)--(-.1,-.2)--(0,-.325)--(.1,-.2)--(.75,-.2)}\n\fi\n\path[draw] \n@toapp@path;},
\end{tikzpicture}

\begin{tikzpicture}[natoapp6c/s/air decoy/.pic={% 
\pic[pic actions] {natoapp6c/s/decoy};\n\path[fill=pgfstrokecolor,pic actions] (0.4, -0.2) rectangle (-0.4, -0.15);},
\end{tikzpicture}

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

\begin{tikzpicture}[natoapp6c/s/air defence/.pic={% 
\ifx\natoapp@fac\n@toapp@friendly% 
\def\n@toapp@opt{[out=90,in=90,looseness=.675]}\n\else\ifx\natoapp@fac\n@toapp@neutral% 
\def\n@toapp@opt{[out=90,in=90,looseness=1]}\n\else\def\n@toapp@opt{[out=45,in=135,looseness=1.5]}\n\fi\fi% 
\def\n@toapp@path{(M.south west) to \n@toapp@opt (M.south east)}\n\path[draw] \n@toapp@path;},
\end{tikzpicture}
\[\text{\texttt{natoapp6c/s/air strip}}\]
\begin{verbatim}
\tikzset%
   natoapp6c/s/air strip/.pic={
   \path[fill=pgfstrokecolor] (-.4,-.1) rectangle (.4,0);
   \path[rotate=45,fill=pgfstrokecolor] (-.4,0) rectangle (.4,.1);
}
\end{verbatim}

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

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

\[\text{\texttt{natoapp6c/s/airborne}}\]
\begin{verbatim}
\tikzset%
   natoapp6c/s/airborne/.pic={
   \begin{scope}
   \clip (0, 0) ellipse (0.45 and 0.15) [reverseclip];
   \path[fill=pgfstrokecolor]
   (0.2,0) -- (0.3,0.175) -- (0.4,0.175) -- (0.375,0) -- (0.4,-0.175) -- (0.3, -0.175) -- cycle;
   \end{scope},
}
\end{verbatim}

367
\tikzset{
  natoapp6c/s/ammunition/.pic={
    \path[draw] (0.175,-0.175)--(-0.175,-0.175)
    (0.125,-0.175)--(0.125, 0) to[out=90,in=90,looseness=2.75]
    (-0.125, 0)--(-0.125, -0.175);
  },
}

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

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

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

\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);},}

\tikzset{natoapp6c/s/arrest/.pic={\path[pic actions] circle(0.2);\pic[scale=.8]{natoapp6c/s/individual};},}

\tikzset{natoapp6c/s/artillery/.pic={\path[pic actions] circle(0.2);},}

\tikzset{natoapp6c/s/anti tank anti armour/.pic={\ifx\natoapp@fac\@to@pp@unknown\path[draw,pic actions] (225:.5)--(M.north)--(315:.5);\else\fi},}
\begin{tikzpicture}[pic actions,draw]
\node[anchor=south east,inner sep=0pt] at (0,0) {
\begin{tikzpicture}[\tikzset{\natoapp6c/s/antenna/.pic={\path[draw,pic actions] (0, -0.3) -- (0, 0.3) (-0.125, 0.3) -- (0, 0.2) -- (0.125, 0.3);},\natoapp6c/s/armoured/.pic={\path[draw] (-0.275,0.2) arc(90:270:0.2)--(0.275, -0.2) arc(270:450:0.2)--cycle;},\natoapp6c/s/armoured fighting vehicle/.pic={\path[fill=pgfstrokecolor] (-.4,-.2) rectangle (-.3,.2) (.3,-.2) rectangle (.4,.2);\path[pic actions] (-.3,0) -- (0,.2) -- (.3,0) -- (0,-.2) -- cycle;},\natoapp6c/s/armoured personnel carrier/.pic={\pic{\natoapp6c/s/vehicle};\path[pic actions] (.35,.15)--(0,.3)--(-.35,.15)--(0,.3)--(-.35,.15);},\natoapp6c/s/arctic/.pic={\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},\fi\};\end{tikzpicture}}
\end{tikzpicture}
\tikzset{
  natoapp6c/s/bicycle equipped/.pic={
    \draw(0,0) circle(.1);
  },
}

\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;},
}

\tikzset{
  natoapp6c/s/booby trap/.pic={
    \path[draw](0, -0.2) ellipse(0.2 and 0.065);
    \begin{scope}
      \clip (0, -0.2) ellipse(0.2 and 0.065) [reverseclip];
      \path[draw] (-0.2, -0.2) -- ( 0.2) -- (0.2, -0.2);
    \end{scope},
}

\tikzset{
  natoapp6c/s/bottomed/.pic={
    \path[draw,fill=pgfstrokecolor] (-0.33,.1) rectangle(0.33,.2);},
}
\tikzset{
  pics/natoapp6c/s/bridge/.is choice,
  pics/natoapp6c/s/bridge/none/.style={
    code={
      \path[pic actions]
      (0.35,-0.15)--(0.25,-0.05)--(-0.25,-0.05)--(-0.35,-0.15)
      (0.35, 0.15)--(0.25, 0.05)--(-0.25, 0.05)--(-0.35, 0.15);},
  pics/natoapp6c/s/bridge/fixed/.style={
    code={
      \pic{natoapp6c/s/bridge};\pic{natoapp6c/s/type=vlight};},
  pics/natoapp6c/s/bridge/folding/.style={
    code={
      \pic{natoapp6c/s/bridge=none};
      \path[draw] (.1,-.2) -- (-.1,-.2) -- (-.1,.2) -- (.1,.2);},
  pics/natoapp6c/s/bridge/hollow/.style={
    code={
      \pic{natoapp6c/s/bridge=none};
      \path[draw] (.1,-.2) -- (-.1,-.2) -- (-.1,.2) -- (.1,.2) -- cycle;},
  pics/natoapp6c/s/bridge/.default=none,
}
}\tikzset{
  natoapp6c/s/capsule/.pic={
    \path[pic actions]
    ($ (0.25, -0.2)!0.1!(0, 0.5)$) --
    ($ (0.25, -0.2)!0.5!(0, 0.5)$) to[in=75, out=105, looseness=0.75]
    ($ (0, 0.5)!0.5!(-0.25, -0.2)$) --
    ($ (0, 0.5)!0.9!(-0.25, -0.2)$) to[in=285, out=255, looseness=0.55]
    cycle;},
}\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;},
}\tikzset{
  natoapp6c/s/chemical biological radiological nuclear
\tikzset{\natoapp6c/s/corps support/.pic={\edef\n@toapp@path{\n@toapp@corps@support{\natoapp@fac}}}\natoapp6c/s/crime\tikzset{\natoapp6c/s/crime/.pic={\expandafter\path[\draw] \n@toapp@path;}}}\natoapp6c/s/decoy\tikzset{\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;}}}\natoapp6c/s/direct communications\tikzset{\natoapp6c/s/direct communications/.pic={\path[\draw] (-0.35,0) circle(.1) (.35,0) circle(.1);\pic[\fill=pgfstrokecolor]{\natoapp6c/s/intermodal};}}}\natoapp6c/s/direction finding\tikzset{\natoapp6c/s/direction finding/.pic={\path[\draw] (-0.3,.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{}}}
\tikzset{\n  natoapp6c/s/electronic ranging/.pic={\n    \path[draw] (135:.225) arc (135:315:.225)--cycle (0,0)--(225:-.225);},
}\n
\tikzset{\n  natoapp6c/s/electronic warfare wide/.pic={\n    % OBS
    \node[natoapp6c/text,natoapp6c/normal text] at(-.25,0){E};
    \node[natoapp6c/text,natoapp6c/normal text] at(.25,0){W};
  },
}\n
\tikzset{\n  natoapp6c/s/engineer/.pic={\n    \path[draw] (.4,-.2)--(.4,.2)--(-.4,.2)--(-.4,-.2) (0,.2)--(0,-.2);},
}\n
\tikzset{\n  natoapp6c/s/enhanced location reporting system/.pic={\n    \path[draw] (0, -0.3) -- (0, 0.3) (-0.2, -.3) -- (0, 0.-.1) -- (0.2, -.3);},
}\n
\tikzset{\n  natoapp6c/s/environmental protection/.pic={\n    \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)
  },
}
\begin{verbatim}
16428 (0.15, -0.15)
16429 (0.05, -0.15)
16430 (0.05, -0.2)
16431 (-0.05, -0.2)
16432 (-0.05, -0.15)
16433 (-0.2, -0.15)
16434 (-0.1, -0.05)
16435 (-0.15, -0.05)
16436 (-0.05, 0.05)
16437 (-0.1, 0.05)
16438 cycle;
16439 }
\end{verbatim}

\textbf{natoapp6c/s/explosion}

\begin{verbatim}
16440 \tikzset{\%
16441 natoapp6c/s/explosion/.pic={%\n16442 \node [shape=rectangle,\n16443 starburst,\n16444 draw,\n16445 minimum width=0.9cm,\n16446 minimum height=0.9cm,\n16447 starburst point height=0.25cm,\n16448 starburst points=12] {};},\n16449 }
\end{verbatim}

\textbf{natoapp6c/s/finance}

\begin{verbatim}
16450 \tikzset{\%
16451 natoapp6c/s/finance/.pic={%\n16452 \path[draw] (-.3,-.25) rectangle(.3,0)\n16453 (.3,0) -- ++(60:.28) -- ([shift=(120:.28)].3,0) -- (.3,0);},\n16454 }
\end{verbatim}

\textbf{natoapp6c/s/fishing_vessel}

\begin{verbatim}
16455 \tikzset{\%
16456 natoapp6c/s/fishing_vessel/.pic={%\n16457 \path[pic actions]\n16458 (-0.15, -0.2) --\n16459 ( 0.15, -0.2) --\n16460 ( 0.25, 0.025) --\n16461 (-0.05, 0.025) --\n16462 (-0.05, 0.125) --\n16463 (-0.2, 0.125) --\n16464 (-0.2, 0.025) --\n16465 (-0.25, 0.025) -- cycle\n16466 (0.025, 0.025) -- (0.025, 0.2)
16467 }
\end{verbatim}
\tikzset{
  \settoheight{\baselineskip}{t}
  \natoapp6c/s/fire protection/.pic={
    \path[fill=pgfstrokecolor] (0,0) circle(.2)
    (0,0) -- (60:.3) -- (120:.3) -- cycle
    (0,0) -- (-30:.3) -- (30:.3) -- cycle
    (0,0) -- (150:.3) -- (210:.3) -- cycle
    (0,0) -- (240:.3) -- (300:.3) -- cycle;
  },
}

\tikzset{
  \settoheight{\baselineskip}{t}
  \natoapp6c/s/fixed and rotary wing/.pic={
    \path[xscale=.45,yscale=.75,pic actions] pic {natoapp6c/s/fixed wing};
    \path[yscale=.45,xscale=.7,rotate=90, pic actions] pic {
      \natoapp6c/s/rotary wing;
    },
  },
}

\tikzset{
  \settoheight{\baselineskip}{t}
  \natoapp6c/s/fixed wing/.pic={
    \path[pic actions]
    (-0.36,0.125) arc (77:275:0.075 and 0.125) -- (0,0) -- cycle
    ( 0.36,0.125) arc (-275:-77:-0.075 and 0.125) -- (0,0)
    --cycle;},
}

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

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

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

\tikzset{%
natoapp6c/s/howitzer/.pic={
\pic[draw]{natoapp6c/s/weapon=base};
\pic[draw]{natoapp6c/s/weapon=top};
\pic[draw]{natoapp6c/s/weapon=multi fire};
\pic[yshift=-8,draw]{natoapp6c/s/weapon=grenade launcher};
},
}

\tikzset{%
natoapp6c/s/in position/.pic={
\path[draw,fill=pgfstrokecolor]
(-.3,-.01) rectangle (-.2,.01) (.2,-.01) rectangle (.3,.01);},
\tikzset{
    natoapp6c/s/individual/.pic=
    \path[\text{pic actions}]
    (0,.08) -- (0,-.3) (-.15,0) -- (.15,0) (0,.18) circle(.1);,
}

\tikzset{
    natoapp6c/s/infantry/.pic=
    \path[draw] (-.75,.5) -- (.75,-.5) (-.75,-.5) -- (.75,.5);,
}

\tikzset{
    natoapp6c/s/intermodal/.pic=
    \path[\text{pic actions}]
    ( 0.15, 0.025) --
    (-0.15, 0.025) --
    (-0.15, 0.075) --
    (-0.25, 0) --
    (-0.15, -0.075) --
    (-0.15, -0.025) --
    ( 0.15, -0.025) --
    ( 0.15, -0.075) --
    ( 0.25, 0) --
    ( 0.15, 0.075) -- cycle;,
}

\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/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);},
}

natoapp6c/s/machine gun

\tikzset{
  natoapp6c/s/machine gun/.pic={
    \pic[draw]{natoapp6c/s/rifle};
    \pic[draw]{natoapp6c/s/weapon=machine gun};,
}

natoapp6c/s/main gun

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

388
\tikzset{\nataapp6c/s/maintenance/.pic={\
\path[fill=pgfstrokecolor] \\
(-.38,.25) to[out=0,in=90,looseness=1.5] (-.2,.05) -- (.2,.05) \\
to [out=90,in=180,looseness=1.5] (.38,.25) -- ++(0,-.08) \\
to [out=180,in=90,looseness=1.5] (.28,0) \\
to [out=-90,in=180,looseness=1.5] (.38,-.17) -- ++(0,-.08) \\
to [out=180,in=0,looseness=1.5] (-.38,-.25) -- ++(0,.08) \\
to [out=0,in=180,looseness=1.5] (-.28,0) \\
to [out=90,in=0,looseness=1.5] (-.38,.17) -- cycle; \\
}, \\
}

\tikzset{\nataapp6c/s/medic/.pic={\
\path[pic actions] \\
(-0.075,-0.2) --(0.075,-.2) \\
--(0.075,-.075) \\
--(.2,-.075) \\
--(.2,.075) \\
--(.075,.075) \\
--(.075,.2) \\
--(-.075,.2) \\
--(-.075,.075) \\
--(-.2,.075) \\
--(-.2,-.075) \\
--(-.075,-.075) \\
--cycle;}, \\
}

\tikzset{\nataapp6c/s/medical/.pic={\\path[draw] (-1,0) -- (1,0) (0,-1) -- (0,1);}, \\
}

\tikzset{\nataapp6c/s/medical treatment{
\begin{tikzpicture}
\tikzset{
  natoapp6c/s/moored/.pic={
    \path[draw] (0,.2) -- (0,-.05) (-.3,-.05) -- (.3,-.05);},
}
\end{tikzpicture}

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

\begin{tikzpicture}
\tikzset{
  natoapp6c/s/motorized/.pic={
    \path[draw] (M.north) -- (M.south);},
  pics/natoapp6c/s/motorised/.style={natoapp6c/s/motorized},
}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
  natoapp6c/s/mortuary affairs/.pic={
    \path[draw] (-.1,-.2) rectangle (.1,.2) (0,-.17) -- (0,.17) (-.07,.1) -- (.07,.1);},
}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
  natoapp6c/s/mountain/.pic={
    \path[draw,fill=pgfstrokecolor] (0,.2) ++(-60:.7) ++(180:.7) -- cycle;},
}
\end{tikzpicture}
\tikzset{
  natoapp6c/s/naval/.pic={
    \def\arrow{(0,0) -- (-.02,0) -- ++(60:.04) -- ++(-60:.04) -- cycle}
    \begin{scope}[pic actions]
      \path[draw]
      (0,.13) circle (.08) (-.2,.04) -- (.2,.04) (0,.04)
      -- (0,-.25) (210:.25) arc (210:340:.25);
      \path[draw,shift=(210:.25),rotate=30] \arrow;
      \path[draw,shift=(340:.25),rotate=-30] \arrow;
    \end{scope},
  }
}

\tikzset{
  natoapp6c/s/navigation/.pic={
    \path[draw]
    (.17,-.2) -- (0,.2) -- (-.17,-.2)
    ($(-180:.17)+(0,.05)$) arc [radius=.17,start angle=-180,end angle=0],
  }
}

\tikzset{
  natoapp6c/s/navy task/.pic={
    \path[pic actions]
    (-0.25, -0.2) -- (-0.25, 0.1) -- (-0.15, 0.2)
    ( 0.25, -0.2) -- ( 0.25, 0.1) -- ( 0.15, 0.2),
  }
}

\tikzset{
  natoapp6c/s/non combatant/.pic={
    \path[draw,fill=pgfstrokecolor]
    (-0.25, -0.2) --
    (-0.25, 0.05) --
    (-0.15, 0.05) --
    (-0.15, 0.2) --
    (0.15, 0.2) --
    (0.15, 0.05) --
    (0.25, 0.05) --
    (0.25, -0.2) -- cycle;
  }
}
natoapp6c/s/non lethal weapon

16919 \tikzset{
16920 natoapp6c/s/non lethal weapon/.pic={
16921 \pic[draw]{natoapp6c/s/weapon};
16922 \pic[draw]{natoapp6c/s/weapon=non lethal};
16923 }

natoapp6c/s/nuclear

16924 \tikzset{
16925 natoapp6c/s/nuclear/.pic=
16926 \path[fill=pgfstrokecolor,pic actions] (0,0) circle(.05)
16927 (0: .3) arc(0 : 60:.3) -- ( 60:.1) arc( 60: 0:.1) -- cycle
16928 (180:.3) arc(180: 120:.3) -- ( 120:.1) arc( 120: 180:.1) -- cycle
16929 (-60:.3) arc(-60:-120:.3) -- (-120:.1) arc(-120:-60:.1) -- cycle;
16930 },
16931 }

natoapp6c/s/observer

16932 \tikzset{
16933 natoapp6c/s/observer/.pic=
16934 \path[pic actions] (0.25,-.2)--(-.25,-.2)--(0,.2)--cycle;,
16935 }

natoapp6c/s/orbiter shuttle

16936 \tikzset{
16937 natoapp6c/s/orbiter shuttle/.pic=
16938 \path[pic actions]
16939 ($(0, 0.3)!0.35!(0.125, -0.15)$) --
16940 (0.125, -0.15) -- (-0.125, -0.15) --
16941 ($(0.125, -0.15)!0.65!(0, 0.3)$) to[in=105, out=75] cycle
16942 (0, -0.20) -- (0, -0.15); },
16943 }

natoapp6c/s/ordnance

16945 \tikzset{
16946 natoapp6c/s/ordnance/.pic=
16947 \path[draw] (0,0) ellipse(.2 and .15);
16948 \begin{scope}

394
\begin{tikzpicture}
\tikzset{
\natoapp6c/s/patrolling/.pic={
\path[draw]
(0.25, 0.05) -- (-0.05, 0.05) -- (0.05, -0.05) -- (-0.4, -0.05)
(-0.3, 0) -- (-0.4, -0.05) -- (-0.3, -0.1)
node [natoapp6c/text,natoapp6c/small text, scale=.5,anchor=west,inner sep=0] at (0.25, 0.05) {P};
},
}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/physician/.pic={
\pic{natoapp6c/s/medical};
\path[draw] (0.1, 0.05) -- (-0.1, 0.05);},
}
\end{tikzpicture}

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/pipeline/.pic={
\path[draw] (-.15,-.15) rectangle (.15,.15)
(-.3,.1) -- (-.15,.1) (-.3,-.1) -- (-.15,-.1)
(.3,.1) -- (.15,.1) (.3,-.1) -- (.15,-.1)
(-.05,.15) rectangle (.05,.25) (-.1,.25) rectangle (.1,.30);},
}
\end{tikzpicture}

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

\begin{tikzpicture}
\tikzset{
\natoapp6c/s/postal/.pic={
},
}
\end{tikzpicture}
\}

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

\tikzset{\natoapp6c/s/psychological/.pic={\n\path[pic actions] (-.25,.15) -- (-.1,.15) -- (.1,.25) -- ++(0,-.5) -- (-.1,-.15) -- (-.25,-.15) -- cycle (.1,.15) -- (.25,.15) (.1,.05) -- (.25,.05) (.1,-.05) -- (.25,-.05) (.1,-.15) -- (.25,-.15);},\n\}

\tikzset{\natoapp6c/s/quarry/.pic={\n\path[draw] (-.2,-.2) -- (.18,.18) (.2,-.2) -- (-.18,.18) (25:.255) arc(25:65:.255) (115:.255) arc(115:155:.255); %\([\text{shift=}((115:.08)}]-.1,.1) \text{arc} \((115:155:.08) %\([\text{shift=}((70:.08)}]-.1,.1) \text{arc} \((70:110:.08) \},\n\}

\tikzset{\natoapp6c/s/quartermaster/.pic={\n\path[draw] (-.4,.1) -- (.1,.1) (.25,.1) circle(.15) (-.3,.1) -- (-.3,-.15) (-.15,.1) -- (-.15,-.15) (-.3,-.08) -- (-.15,-.08);},\n\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

\}

397
\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/radar};
\end{tikzpicture}
\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/radio};
\end{tikzpicture}
\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/radio relay};
\end{tikzpicture}
\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/radio relay line of sight};
\end{tikzpicture}
\begin{tikzpicture}
\node at (0,0) {natoapp6c/s/radio teletype};
\end{tikzpicture}

398
\tikzset{\n  natoapp6c/s/rising/.pic={\n    \path[draw,fill=pgfstrokecolor] (0, 0.2) -- (0, -0.167) \n    (0.1, -0.2) -- (-0.1, -0.2) -- (0, 0.0);},\n}\n
\tikzset{\n  natoapp6c/s/riverine/.pic={\n    \ifn@to@pp@below\n    \path[\textcolor{white}{pic actions}] (M.south west) \n    to [out=-90,in=-90,looseness=.5] (M.south east) -- cycle;\n    \else\n    \path[\textcolor{white}{pic actions}] (-.5,.15) to \n    [out=-80,in=180] (0,-.15) to \n    [out=0,in=260] (.5,.15) -- cycle;\n    \fi},\n}\n
\tikzset{\n  pics/natoapp6c/s/rocket launcher/.is choice,\n  pics/natoapp6c/s/rocket launcher/base/.style={\n    code={\n      \pic[draw]{natoapp6c/s/weapon=base};\n      \pic[draw]{natoapp6c/s/weapon=rifle};\n      \pic[yshift=-4,draw]{natoapp6c/s/weapon=rifle};\n    }},\n  pics/natoapp6c/s/rocket launcher/anti tank/.style={\n    code={\n      \pic[draw]{natoapp6c/s/rocket launcher=base};\n      \pic[draw]{natoapp6c/s/weapon=anti tank};\n    }},\n  pics/natoapp6c/s/rocket launcher/single/.style={\n    code={\n      \pic[draw]{natoapp6c/s/rocket launcher=base};\n      \pic[draw]{natoapp6c/s/weapon=bottom};\n    }},\n  pics/natoapp6c/s/rocket launcher/multiple/.style={\n    code={\n      \pic[draw]{natoapp6c/s/rocket launcher=single};\n      \pic[yshift=-6,draw]{natoapp6c/s/weapon=multi fire};\n    }},\n  pics/natoapp6c/s/rocket launcher/single head/.style={\n    code={\n      \pic[yshift=4,draw]{natoapp6c/s/weapon=rifle};\n    }},\n  pics/natoapp6c/s/rocket launcher/multiple head/.style={\n    code={\n      \pic[yshift=-4,draw]{natoapp6c/s/weapon=rifle};\n    }},
\begin{tikzpicture}
\draw[fill=blue!20] (0,0) circle (1);
\end{tikzpicture}
\begin{tikzpicture}
\tikzset{
    pics/natoapp6c/s/satellite/small/.style={
        code={
            \begingroup	ikz@picmode
            \pic[scale=0.6]{natoapp6c/s/satellite=none};
            \endgroup
        }},
    pics/natoapp6c/s/satellite/reconnaissance/.style={
        code={
            \pic[yshift=-1,fill=pgfstrokecolor]{natoapp6c/s/satellite=none};
            \path[draw,join=bevel,pic actions]
            (-0.075, -0.05) -- +(250:0.1)
            (-0.025, -0.05) -- +(260:0.1)
            ( 0.025, -0.05) -- +(280:0.1)
            ( 0.075, -0.05) -- +(290:0.1);
        }},
    pics/natoapp6c/s/satellite/.default=none,
}
\end{tikzpicture}
natoapp6c/s/seabed installation

\tikzset{
  natoapp6c/s/seabed installation/.pic=\
  \path[pic actions]
  (-0.25, -0.2) --
  ( 0.25, -0.2) --
  ( 0.25, -0.075) --
  ( 0.05, -0.075) --
  ( 0.05,  0.025) --
  (-0.125,  0.025) --
  (-0.125,  0.2) --
  (-0.25,  0.2) -- cycle;},
}

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;},
}

natoapp6c/s/signal

\tikzset{%
  natoapp6c/s/signal/.pic={%
    \path[draw] (M.north west) -- (0,-.1) -- (0,.1) -- (M.south east);},
}

natoapp6c/s/signals intelligence

\tikzset{%
  natoapp6c/s/signals intelligence/.pic={%
    \path[draw] (-.2,.2) -- (-.13,.25) -- (-.07,.2) -- (0,.25) --
}

405
\begin{tikzpicture}

\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{
  pics/natoapp6c/s/small squashed text/.style={
    code={\n@to@pp@text@smallsquashed{#1};}},
}\}

\tikzset{
  pics/natoapp6c/s/small text/.style={
    code={\n@to@pp@text@small{#1};}},
}\}

\tikzset{
  natoapp6c/s/sniper
}\}

\end{tikzpicture}
\tikzset{
natoapp6c/s/sniper/.pic={
\path[draw] (-.2,.2)--(-.05,.2) (.05,.2)--(.2,.2) (0,.15)--(0,-.2);},
}

\tikzset{
natoapp6c/s/space station/.pic=
\path[\join=bevel,pic actions]
(-80:.15 and .06) -- (0.025, 0.175) arc(0:180:0.025) -- (-100:.15 and .06)
($80:.25 and 0.1)+(0,-.0125)$) arc(80:-260:.25 and .1) --
(-260:.15 and .06) arc (-260:80:.15 and .06) -- cycle
(-82:.25 and .1) -- (0.025, -0.175) arc(360:180:0.025) -- (-98:.25 and .1);
},
}

\tikzset{
pics/natoapp6c/s/squashed text/.style={code={\n@to@pp@text@squashed{#1};}},
}

\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]
($0,-0.05) + (106.6:0.35 and 0.15)$)
arcc (106.6:433.4:0.35 and 0.15) |- (0, 0.2) -| cycle;
},
}
\tikzset{\n\n\n}$natoapp6c/s/supply$\n\n\tikzset{\n\n\n}$natoapp6c/s/surface\ combatant$\n\n\tikzset{\n\n\n}$natoapp6c/s/survey$\n\n\tikzset{\n\n\n}$natoapp6c/s/tactical\ satellite$\n\n\tikzset{\n\n\n}
\tikzset{
\natoapp6c/s/tank/.pic={
\pic[draw]{natoapp6c/s/vehicle};
\path[pic actions] (0.35, 0.2) -- (-0.35, 0.2);},
}

\tikzset{
\natoapp6c/s/text/.style={code={
\n@to@pp@dbg{3}{Text: '#1'}%
\n@to@pp@text@normal{#1};}},
}

\tikzset{
\natoapp6c/s/topographic/.pic=
\path[draw] (0,.05) -- (0,.2)
(0,.05) -- (-.1,-.2)
(0,.05) -- (.1,.2)
(-30:.15) arc[radius=.15,start angle=-30,end angle=-150];},
}

\tikzset{
\natoapp6c/s/torpedo/.pic=
\path[draw,fill=pgfstrokecolor,pic actions]
(-0.35, 0) --
(-0.3, 0.075) --
(0.25, 0.075) --
(0.35, -0.075) --
(0.35, 0.075) --
(0.25, -0.075) --
(-0.3, -0.075) -- cycle;},
}
\tikzset{
\n\n\n\aftergroup\relax
}
\tikzset{natoapp6c/s/unexploded ordnance/.pic={
\begin{scope}[transparency group=knockout]
\path[pic actions] (0,0) circle(.2);
\pic[opacity=0]{natoapp6c/s/small squashed text=UXO};
\end{scope},
}}

\tikzset{natoapp6c/s/unmanned/.pic={
\path[pic actions]
(0,-0.1) --(0.45,0.05) --(0.45,0.1) --(0,0.025) --(-0.45,0.1) --(-0.45,0.05) --cycle;,}
}}

\tikzset{natoapp6c/s/utility vehicle/.pic={
\path[pic actions]
(-0.35, 0.2) -- (-0.35, -0.2) -- ( 0.35, -0.2) -- ( 0.35, 0.2)
}}
\tikzset{%
natoapp6c/s/video imagery/.pic={
  \path[pic actions]
  (-0.4, 0.2) -- (-0.4, -0.2) -- (0.05, -0.2) -- (0.2, 0.2) -- cycle
  (0.075, -0.15) -- (0.4, -0.15)
  (0.16, 0.1) -- (0.4, 0.1);
  \path[draw,fill=pgfstrokecolor,pic actions](0.38,-.2) rectangle (0.42,.15);},
}

\tikzset{%
natoapp6c/s/warfare vessel/.pic={
  \path[draw,fill=pgfstrokecolor] (0, -0.2) -- (0.3, 0.05) -- (-0.3, 0.05) -- cycle;
},
}

\tikzset{%
natoapp6c/s/water/.pic={
  \path[pic actions]
  (-0.3, 0.05) -- (0, 0.05) to[in=90, out=0] (0.3, -0.2)
  (0, 0.05) -- (0, 0.2)
  (0.075, 0.2) -- (-0.075, 0.2);},
}

\tikzset{%
pics/natoapp6c/s/wheeled/.is choice,
pics/natoapp6c/s/wheeled/and tracked/.style={
  code={
    \ifn@to@pp@below%
      \path[pic actions]
      ([shift={(.4,-.16)}]M.south west) arc [radius=.08,start angle=-90,end angle=-270]
      -- ([shift={(-.08,0)}]M.south east)
      arc [radius=.08,start angle=90,end angle=-90]
    \else
      \path[pic actions]
      (M.west) arc [radius=.08,start angle=-90,end angle=-270]
      -- (M.east)
      arc [radius=.08,start angle=90,end angle=-90]
    \fi
  },
}

5.6.21 Some extra MIL-STD symbols

Extra NATO App6(c) symbol (from MIL-STD)
A list of all defined symbols

\def\tttopsall{
weapon=base,
weapon=top,
weapon=bottom,
weapon=rifle,
weapon=machine gun,
weapon=grenade launcher,
weapon=missile launcher,
weapon=non lethal,
weapon=multi fire,
weapon=air defence,
weapon=anti tank,
weapon=full,
weapon,
type=light,
type=medium,
type=heavy,
type=vlight,
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,
amoured 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,
corps support,
crime,
decoy,
direct communications,
direction finding,
diving=none,
diving=military,
diving,
drilling,
drill,
edcrementor,
edelectric power,
electronic ranging,
electronic warfare wide,
engineer,
enhanced location reporting system,
environmental protection,
explosion,
finance,
fishing vessel,
fire protection,
fixed and rotary wing,
fixed wing,
flame thrower,
floating,
surfaced,
food,
fuel,
grenade launcher=none,
grenade launcher=non lethal,
grenade launcher,
graffiti,
group,
gun=base,
gun=air defence,
gun=anti tank,
gun=direct,
gun=recoilless,
gun,
headquarters,
house,
howitzer,
in position,
individual,
infantry,
intermodal,
jagged wave,
jam,
jamming,
jetski,
killing,
labour,
land mine=personnel,
land mine=tank,
land mine=none,
land mine,
land missile,
laser,
launcher,
launder,
machine gun,
main gun,
maintenance,
medic,
medical,
medical treatment,
mine,
machine clearing equipment,
mine warfare vessel,
mine,
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,
motorized,
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, runway, sailing boat, satellite=none, satellite=astronomical, satellite=biological, satellite=communications, satellite=navigation, satellite=earth observing, satellite=tether, satellite=small, satellite=reconnaissance, satellite, sea mine=top half,
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

allied 1 id  axis 1 ad  out of supply
allied 2 ad  axis 2 ad  game turn
allied 3 abid axis 3 ic

via Tikz styles. Also assume that we have macros

\board  \oob  \charts  \front

which produces tikzpictures to the board, OOBs, charts, and cover, respectively. All this is defined in our package mygame. Of course that we have our rules in the file game.pdf.

We prepare a simple LaTeX source file

\documentclass{wgexport}
\usepackage{mygame}
\begin{document}
\begin{imagelist} \chitimages{{allied 1 id,allied 2 ad,allied 3 abid}/Allied,\
{axis 1 ad,axis 2 ad,axis 3 ic}/Axis,\
{out of supply, game turn}/Markers})
\info{Board}{board}{} \board
\info{OOB}{oob}{} \oob
\info{Charts}{chart}{} \chart
\info{Cover}{front}{} \front
\end{imagelist}
\end{document}

When we run LaTeX on this, we will get a PDF where each page is a separate image and the page is cropped to image. In addition we will get a CSV (comma-separated-values) file export.csv which contains some meta information about each page. In particular, it identifies the name of each page, the category, and sub category of the image.

For chits, the name of the image is the style name (e.g., game turn). For other images, it is the first argument to \info above.

The category is for chits is always counter. For other images, it is the second argument to the \info macro (e.g., board).

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

```bash
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 underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>/hex/label (key)</th>
<th>/hex/ridges (key)</th>
<th>/hex/terrain (key)</th>
<th>/hex/town (key)</th>
<th>/tikz/chit/identifier (key)</th>
<th>/tikz/chit/1 factor (key)</th>
<th>/tikz/chit/2 factors (key)</th>
<th>/tikz/chit/3 factors (key)</th>
<th>/tikz/chit/4 factors (key)</th>
<th>/tikz/chit/factor (key)</th>
<th>/tikz/chit/setup (key)</th>
<th>/tikz/chit/right (key)</th>
<th>/tikz/chit/left (key)</th>
<th>/tikz/chit/upper left (key)</th>
<th>/tikz/chit/upper right (key)</th>
<th>/tikz/chit/terrain (key)</th>
<th>/tikz/chit/town (key)</th>
<th>/tikz/chit/ridges (key)</th>
<th>/tikz/chit/full (key)</th>
<th>/tikz/natoapp6c (key)</th>
<th>/tikz/natoapp6c/below (key)</th>
<th>/tikz/natoapp6c/echelon (key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>oob</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>full</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>951</td>
<td>15558</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
<td>13446</td>
</tr>
<tr>
<td>id</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>left</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>right</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>setup</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>symbol</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>upper left</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>upper right</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>extra</td>
<td>951</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
<tr>
<td>extra clipped</td>
<td>951</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
<td>13428</td>
</tr>
</tbody>
</table>

420