

HexBattle

A Toolset for Designing 'Battle Cry' Scenario Maps

Developed by Derek Hohls

Version 1.0 - 30 June 2003

Overview

HexBattle is a toolset for helping to design scenarios for Richard Borg's **Battle Cry**. I have created it as a small token of my appreciation of a great game, and hopefully to give something useful back to the community of gamers who have 'rallied around' to expand the material available, creating dozens of new scenarios and house rules.

Although there is much enthusiasm for **Battle Cry**, it's not that easy to create a smart looking map for a new scenario. The range and quality of new scenario maps developed and presented to date, (see the [Battle Cry Resources](#)), is highly variable. **HexBattle** harnesses the power of XML and SVG to enable anyone to quickly and easily create smart looking maps.

What is HexBattle?

Summary

Simply put, **HexBattle** is a mechanism for game designers to readily create maps (or other output formats) for scenarios. You will still have to undertake the process of historical research to identify interesting battles and carry out the adaptation to the **Battle Cry** format. Once this has been done, **HexBattle** can be used to track the scenario design, and allow changes to easily be made. **HexBattle** can also be used to adapt existing scenario designs by storing them in a common format that allows any number of different layout presentations to be generated.

How it works

HexBattle is based on XML-technologies. XML provides a very powerful mechanism for managing structured data, and has 'spun off' a whole range of related tools and technologies that can all be used in similar ways. Its core benefit is that it allows data and information to be stored independently of the tools that process and present that data (for more reading on XML, see the [XML Resources](#)).

HexBattle utilises XML as follows:

1. Each scenario is stored in an XML file (a simple text file that describes the scenario in a structured way)
2. An XSL file is used to determine **how** that XML file should be converted to its graphic equivalent
3. An XSLT processor (a separate stand alone program) will take the XML file and 'transform' the XML scenario file according to the 'instructions' in the XSL file
4. The result of the transform process is a customized display of the scenario information; this can be

in SVG (graphical view), HTML (text view), or even a simple text file.

Unless you are an XML designer, or want to create new kinds of map layouts (apart from the one most commonly used), you will only need to work with the XML scenario file (described below) - it is not necessary to understand in detail how and why the rest of the process works... it just does!

Why use HexBattle?

If you have not worked with XML before, you may be thinking that this all sounds complex and messy - and that you are better off just using a graphics package. However, after you try the process once or twice, you will see its simplicity and usefulness. Creating a scenario is quick - open up a blank XML template, fill in the names and locations of troops and terrain, along with the basic battle rules and information, and save. Run the transform process (a one line command!). Voila! A neat, attractive map is now available. It's even easier if you want to make changes (to your scenarios, or someone else's) - just open the scenario file, make the correction (e.g. typos, incorrect locations or missing terrain), rerun the transform, and the map is instantly updated. No messing with cut-and-paste or trying to align small, fiddley graphics 'by hand'.

The core XML scenario files are very small (about 5-10k each), and the resulting SVG map files are about 15-30k each. If these are zipped, they shrink even more! This tiny size (compared to the 300k, or more, size of typical 'standard' JPEG or GIF graphics map file) makes them very quick to send, store and display on the web.

All **Battle Cry** scenarios can now be stored in consistent format. This allows them to be readily processed into any number of output layouts, including one as close to the original "look-and-feel" as possible. Hopefully, the ease with which new scenarios can be documented will inspire many others to create and share their work.

How to Use HexBattle

Platform Requirements

NOTE: **HexBattle** has been developed and tested on a Windows machine. In theory, it should be possible to use it on other operating systems, but I am not able to do such testing. [Feedback](#) on your experience in this regard is welcome!

At a minimum you will need a PC with:

- a browser (or other software tool) capable of displaying SVG (e.g. IE5+, Mozilla/Netscape 6+)
- a text or HTML editor (usually standard on most personal computers)

NOTE: no special graphics editor is required. If you are already familiar with XML, or wish to learn more about it, it is suggested that you use obtain an XML-capable editor (see the [XML Resources](#) for some suggestions) for editing the various **HexBattle** files.

Setting Up

Downloads Required

- The **HexBattle** ZIP file (discussed below), can be obtained from John Foley's [Battle Cry site](#)
- The XT XSLT processor (developed by James Clarke) for Windows/UNIX can be obtained from Bill Lindsey's [XT site](#) (if you work on another OS, refer to the [XML Resources](#)). This program is required to convert the XML (text) files into SVG (graphic) maps, or other display formats.
- An SVG browser plug-in can be obtained from [Adobe](#) and is required to display the output map.

NOTE: if you want someone else to convert your XML scenario files for you, you will not need the XSLT processor - but if you cannot see the resulting SVG file, then you will not be able to easily verify that the scenario 'looks' the way you intended.

Installation

In summary, you will need to:

1. Install the SVG viewer (follow the setup instructions for the one you have chosen)
2. Unzip the **HexBattle** ZIP file to a directory of your choice
3. Place the XSLT processor (if it's a stand-alone executable, like XT) in same directory as the **HexBattle** files (or follow the installation and usage instructions for the processor you have chosen)

Explanation of the HexBattle Files

The **HexBattle** toolset (that you have downloaded and unzipped), consists of a number of files:

- A set of image files (.png extension) - these represent the different types of terrain that can be found on a **Battle Cry** map (a full list is shown in the [Terrain Tile Set](#)), as well as the symbols for the **Battle Cry** playing pieces. PNG files are used, rather than GIF format, because this format is in the public domain, and is directly supported by SVG.
- A number of XML files (.xml extension) - these are sample scenarios, as well as a blank template file (`bc_Blank.xml`) which can be used for 'filling in' information for scenarios of your creation
- Some stylesheet (.xsl extension) files - contain the instructions for processing the scenario files (see [XSL "Map Making" Files](#))
- A number of scenario output files - these are files created from the sample scenarios, and are supplied for example purposes. They include files with SVG, HTML and TXT extensions; typically in the format `BattleName.extension`.

Creating a Battle Cry Map with HexBattle

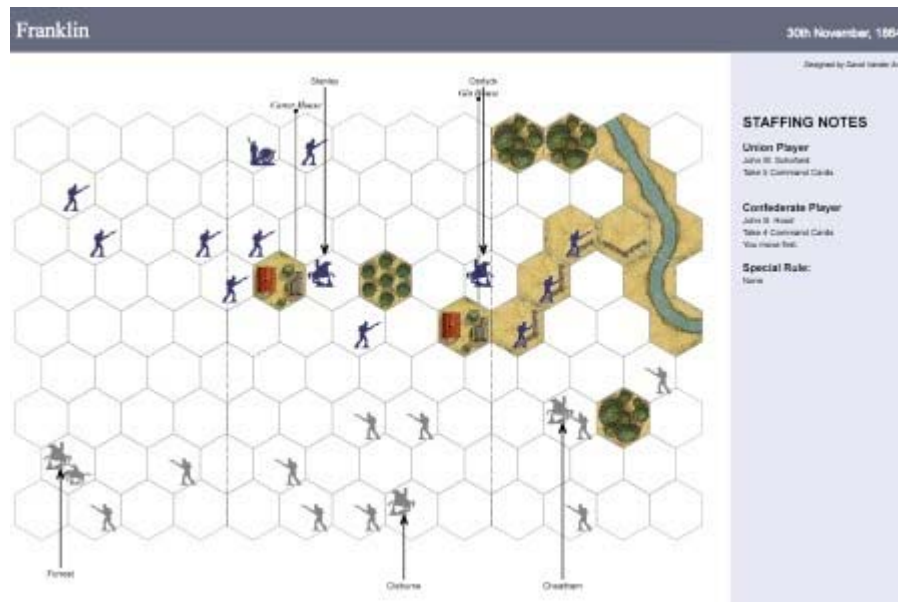
In general, you will work through the following steps:

1. Create an XML scenario file using a simple text editor (such as Notepad, on Windows) or an HTML/XML capable editor. The [XML Scenario File](#) describes all the sections in detail. You can start with the blank template (`bc_Blank.xml`) file, or create one from scratch. This file will typically be named after the battle, e.g. `MyBattle.xml`.
Hint: once you have created the file, you can check that it is well-formed, by opening it up in a web browser capable of recognizing XML. XML files that are **not** well-formed will cause errors in Step 2, and need to be corrected.
2. Process the XML scenario file using the XSLT processor. Typically you need to supply the processor with the name of the input XML file, the name of the XSL file with the processing instructions, and the name of the output file. For example, if you are working with XT, you would enter the following

type of instruction via a command-line:

```
xt MyBattle.xml battle2colorSVG.xsl MyBattle.svg.
```

The resulting output from a scenario (scaled here to 30% size for quick viewing) will look something like:



- View the output file (for example, `MyBattle.svg`, in the example above) by loading it into your browser. If there are obvious errors or omissions, you will need to go back to Step 1 to correct or add the information.
- If you are viewing the SVG using the Adobe plug-in, you can 'right-click' on the image, select 'Copy SVG' from the pop-up menu, and then paste the image (now stored in the buffer) into a graphic program (such as Paint, on Windows). It can then be saved into some other graphic format for printing.

The XML Scenario File

The XML files that store the scenarios are the heart of **HexBattle** - even if the rest of the explanations and discussions in this document seem complex (or long winded), it is certainly possible to understand how to create a scenario file. You can then follow the [step-by-step instructions](#) for turning this file into a graphic map, or even send it someone else to do it for you! If you have not seen an XML file before, first read the [Three Minute Guide to XML](#).

An example scenario file looks this:

```
<?xml version="1.0"?>
<hbml:hexbattle
  id="franklin"
  type="battlecry"
  version="1.0"
  xmlns:hbml="http://users.erols.com/jadf/bc/">
<hbml:meta>
  <hbml:designer>
    <hbml:name>David Vander Ark</hbml:name>
    <hbml:email></hbml:email>
  </hbml:designer>
  <hbml:version>1</hbml:version>
</hbml:meta>
<hbml:scenario>
  <hbml:name abpp="tn036" acc="tn/tn016" oob="">Franklin</hbml:name>
  <hbml:alternate></hbml:alternate>
  <hbml:title></hbml:title>
```

```

<hbml:date year="1864" month="04" day="30">30th November, 1864</hbml:date>
<hbml:period></hbml:period>
<hbml:location>
  <hbml:country>USA</hbml:country>
  <hbml:state code="tn">Tennessee</hbml:state>
</hbml:location>
<hbml:description>Franklin scenario recreates the Confederate assault on
  Union entrenchments at Franklin. </hbml:description>
</hbml:scenario>
<hbml:terrain>
  <hbml:hex row="1" col="10" type="terrain" style="woods"></hbml:hex>
  <hbml:hex row="1" col="11" type="terrain" style="woods"></hbml:hex>
  <hbml:hex row="4" col="7" type="terrain" style="woods"></hbml:hex>
  <hbml:hex row="7" col="12" type="terrain" style="woods"></hbml:hex>
  <hbml:hex row="4" col="5" type="terrain" style="buildings">Carter House</hbml:hex>
  <hbml:hex row="5" col="9" type="terrain" style="buildings">Gin House</hbml:hex>
  <hbml:hex row="4" col="7" type="terrain" style="orchard"></hbml:hex>
  <hbml:hex row="3" col="11" type="terrain" style="fence_one" face="3"></hbml:hex>
  <hbml:hex row="3" col="12" type="terrain" style="fence_two" face="4"></hbml:hex>
  <hbml:hex row="4" col="10" type="terrain" style="fence_two" face="3"></hbml:hex>
  <hbml:hex row="5" col="10" type="terrain" style="fence_two" face="3"></hbml:hex>
  <hbml:hex row="1" col="12" type="terrain" style="river_straight" face="3"></hbml:hex>
  <hbml:hex row="2" col="12" type="terrain" style="river_straight" face="3"></hbml:hex>
  <hbml:hex row="3" col="13" type="terrain" style="river_curve" face="5"></hbml:hex>
  <hbml:hex row="4" col="12" type="terrain" style="river_curve" face="2"></hbml:hex>
  <hbml:hex row="5" col="13" type="terrain" style="river_curve" face="1"></hbml:hex>
</hbml:terrain>
<hbml:forces>
  <hbml:force id="u" type="union" name="Union">
    <hbml:commander>John M. Schofield</hbml:commander>
    <hbml:unit id="UG01" type="general" row="4" col="6">Stanley</hbml:unit>
    <hbml:unit id="UG02" type="general" row="4" col="9">Opdyck</hbml:unit>
    <hbml:unit id="UI01" type="infantry" row="1" col="6"></hbml:unit>
    <hbml:unit id="UI02" type="infantry" row="2" col="1"></hbml:unit>
    <hbml:unit id="UI03" type="infantry" row="3" col="2"></hbml:unit>
    <hbml:unit id="UI04" type="infantry" row="3" col="4"></hbml:unit>
    <hbml:unit id="UI05" type="infantry" row="3" col="5"></hbml:unit>
    <hbml:unit id="UI06" type="infantry" row="3" col="11"></hbml:unit>
    <hbml:unit id="UI07" type="infantry" row="4" col="4"></hbml:unit>
    <hbml:unit id="UI08" type="infantry" row="4" col="10"></hbml:unit>
    <hbml:unit id="UI09" type="infantry" row="5" col="7"></hbml:unit>
    <hbml:unit id="UI10" type="infantry" row="5" col="10"></hbml:unit>
    <hbml:unit id="UA01" type="artillery" row="1" col="5"></hbml:unit>
  <hbml:special>
    <hbml:cards>5</hbml:cards>
    <hbml:instructions></hbml:instructions>
  </hbml:special>
</hbml:force>
  <hbml:force id="c" type="confederate" name="Confederate">
    <hbml:commander>John B. Hood</hbml:commander>
    <hbml:unit id="CG01" type="general" row="7" col="11">Cheatham</hbml:unit>
    <hbml:unit id="CG02" type="general" row="8" col="1">Forrest</hbml:unit>
    <hbml:unit id="CG03" type="general" row="9" col="8">Cleburne</hbml:unit>
    <hbml:unit id="CI01" type="infantry" row="6" col="12"></hbml:unit>
    <hbml:unit id="CI02" type="infantry" row="7" col="7"></hbml:unit>
    <hbml:unit id="CI03" type="infantry" row="7" col="8"></hbml:unit>
    <hbml:unit id="CI04" type="infantry" row="7" col="11"></hbml:unit>
    <hbml:unit id="CI05" type="infantry" row="8" col="3"></hbml:unit>
    <hbml:unit id="CI06" type="infantry" row="8" col="5"></hbml:unit>
    <hbml:unit id="CI07" type="infantry" row="9" col="2"></hbml:unit>
    <hbml:unit id="CI08" type="infantry" row="9" col="6"></hbml:unit>
    <hbml:unit id="CI09" type="infantry" row="9" col="7"></hbml:unit>
    <hbml:unit id="CI10" type="infantry" row="9" col="11"></hbml:unit>
    <hbml:unit id="CC01" type="cavalry" row="8" col="1"></hbml:unit>
  <hbml:special>
    <hbml:cards>4</hbml:cards>
    <hbml:instructions>You move first.</hbml:instructions>
  </hbml:special>
</hbml:force>

```

```

    </hbml:special>
  </hbml:force>
</hbml:forces>
<hbml:notes>
  <hbml:note id="s" type="special">
    <hbml:p>None</hbml:p>
  </hbml:note>
  <hbml:note id="h" type="historical">
    <hbml:p>No summary.</hbml:p>
  </hbml:note>
</hbml:notes>
</hbml:hexbattle>

```

A section-by-section explanation of this file now follows.

Header Information

```

<?xml version="1.0"?>
<hbml:hexbattle
  id="firstbullrun"
  type="battlecry"
  version="1.0"
  xmlns:hbml="http://users.erols.com/jadf/bc/">

```

The above section is the 'standard' header for an XML file - the only thing that will need to change will be the "id"; this is typically the name of the battle (in lowercase), and should be unique across all battles. For multi-day, or more complex, battles, a longer name might be required e.g. "gettysburg-pickett-1". The 'hbml:' is a prefix which appears in all subsequent tags; it is linked to a unique identifier for this type of XML file that distinguishes it from any other XML file.

Meta Information

```

<hbml:meta>
  <hbml:designer>
    <hbml:name>David Vander Ark</hbml:name>
    <hbml:email></hbml:email>
  </hbml:designer>
  <hbml:version>1</hbml:version>
</hbml:meta>

```

The 'meta information' section contains information on the <designer> of the scenario (the person's <name> and <email>) and which <version> of the scenario this particular file represents.

Scenario Information

```

<hbml:scenario>
  <hbml:name abpp="tn036" acc="tn/tn016" oob="">Franklin</hbml:name>
  <hbml:alternate></hbml:alternate>
  <hbml:title></hbml:title>
  <hbml:date startyear="1864" startmonth="04" startday="30">30th November, 1864</hbml:
  <hbml:period></hbml:period>
  <hbml:location>
    <hbml:country>USA</hbml:country>
    <hbml:state code="tn">Tennessee</hbml:state>
  </hbml:location>
  <hbml:description>Franklin scenario recreates the Confederate assault on
    Union entrenchments at Franklin. </hbml:description>
</hbml:scenario>

```

The 'scenario information' section contains details about the actual battle being represented. The <name>

is that conventionally assigned to the battle. The "abpp" identifier refers to the key part of the web address for the ABPP ([American Battlefield Protection Program](#)) web page. Similarly, the "acc" provides a link to the ([American Civil War](#)) web site. The "oob" provides a link to Order-of-Battle pages at the [American Civil War On Line](#) web site.

The <alternate> lists any other name(s) by which the battle is commonly referred to. The <title> is only used when a portion of the battle is being represented by the scenario (as might be the case for larger battle, such as Gettysburg). The <date> is when the battle was fought; this can be a single day, or a date range (e.g. 11th-12th December 1863). The <period> is used when only a given day of the battle is being simulated.

The <location> provides information on the country and state in which the battle took place. The "code" provides a link to the key code for ([American Civil War](#)) state battle map site. The <description> is a brief (one liner) of what the scenario portrays.

Terrain Information

```
<hbml:terrain>
  <hbml:hex row="1" col="10" type="terrain" style="woods"></hbml:hex>
  <hbml:hex row="4" col="5" type="terrain" style="buildings">Carter House</hbml:hex>
  <hbml:hex row="5" col="9" type="terrain" style="buildings">Gin House</hbml:hex>
  <hbml:hex row="4" col="7" type="terrain" style="orchard"></hbml:hex>
  <hbml:hex row="1" col="12" type="terrain" style="river_straight" face="3"></hbml:he
  ...
</hbml:terrain>
```

The 'terrain information' section contains a <hex> entry for each piece of terrain (or other map marking pertinent to the scenario) that appears on the board. Each <hex> entry contains:

- **rows** are numbered from "1" to "9" (starting at the 'top' of the board)
- **columns** are numbered from "1" to "13" (starting at the 'left' of the board)
- **type** is usually "terrain", but can be "special":
 - for "terrain" type hexes, a **style** property is required to identify which specific terrain is displayed and, in some cases, a **face** property as well (to indicate direction) - see the full [Terrain Tile Set](#) list.
 - for "special" types of hexes, a **style** property is required to identify which type is applicable - this can either be "border" (for a border to be drawn around the hex), or "shaded", for a coloured, semi-transparent overlay to fill the hex.

In addition, the name of the terrain feature (hill, buildings, wood, river and so on), or other notation can be supplied e.g. <hex >Carter House</hex> for a building label, or <hex >Union Reinforcements</hex> for special instructions.

Tip: In the case where one type of terrain feature has to overlay another - such as is the case for Field-works placed on, say, a Hill - the overlying terrain <hex> entry should occur **after** (below) the entry for the base terrain. All "special" hexes are also usually placed at the end, as they typically provide additional annotation that must appear 'over' the underlying terrain.

Forces Information

```
<hbml:forces>
  <hbml:force id="u" type="union" name="Union">
  ...
  <hbml:special>
```

```

    <hbml:cards>5</hbml:cards>
    <hbml:instructions></hbml:instructions>
  </hbml:special>
</hbml:force>
<hbml:force id="c" type="confederate" name="Confederate">
  ...
<hbml:special>
  <hbml:cards>4</hbml:cards>
  <hbml:instructions>You move first.</hbml:instructions>
</hbml:special>
</hbml:force>
</hbml:forces>

```

The 'forces information' section contains information on the opposing sides in the battle. There will be one <force> section for each side. Each <force> entry identifies the force:

- **id** is a unique identifier for each force (the defaults of "u" for Union, or "c" for Confederate can be used)
- **type** *must* be either "union" or "confederate" (this key information is used, for example, to identify corresponding image files)
- **name** is used for display purposes

Each section also contains detailed force information (see below) as well as <special> information applicable to it. This includes the number of <cards> available to that side, and <instructions>, such as who it is that starts first.

Force Information

```

<hbml:commander>John Bell Hood</hbml:commander>
<hbml:unit id="CG01" type="general" row="7" col="11">Cheatham</hbml:unit>
<hbml:unit id="CG02" type="general" row="8" col="1">Forrest</hbml:unit>
<hbml:unit id="CI01" type="infantry" row="6" col="12"></hbml:unit>
<hbml:unit id="CI02" type="infantry" row="7" col="7"></hbml:unit>
<hbml:unit id="CC01" type="cavalry" row="8" col="1"></hbml:unit>

```

The 'force information' section contains the detailed breakdown of a side's playing pieces (units). Each <unit> has associated details on:

- **id** is a unique identifier for each force; the suggested notation to be used is a four letter code:
 - U or C - for Union or Confederate,
 - G,I,C or A - for general, infantry, cavalry or artillery
 - A two-digit number (from 01 upwards...)
- **type** must be one of "general", "infantry", "cavalry" or "artillery"
- **rows** are numbered from "1" to "9" (starting at the 'top' of the board)
- **columns** are numbered from "1" to "13" (starting at the 'left' of the board)

The name of the general or unit can also be enclosed in the <unit></unit> tag; this is typically used to provide a label for the piece on the map.

Notes Section

```

<hbml:notes>
  <hbml:note id="s" type="special">
    <hbml:p>None</hbml:p>
  </hbml:note>
  <hbml:note id="h" type="historical">
    <hbml:p>No summary.</hbml:p>

```

```
</hbml:note>  
</hbml:notes>
```

The 'notes' section can contain any number of notes. Typically, there will be a <note> for "special" rules, as indicated by the "special" type. Each note can contain any number of <p>aragraphs - each of these will usually be displayed as one line on the output map; so it's important not to make them too long (about 40 characters or less).

Tip: Bear in mind, when adding special rules, that the XML file may **not** be displayed in graphical format. For this purpose, the <comment> tag has been made available. Use it to provide supplementary information that may be required to understand the special rules, when creating a non-graphical version of the map. For example:

```
<hbml:p>Place infantry reinforcements here<hbml:comment> (Hex 113, 212)</hbml:comment
```

The XSL "Map Making" Files

A number of XSL (stylesheet) files, each containing different instructions for processing any scenario XML file, are supplied with **HexBattle**.

- The `battle2colorSVG.xsl` file will, when processed in conjunction with a scenario XML file, create a map layout in SVG, very similar to the conventional one commonly available on the web.
- The `battle2text.xsl` file will, when processed in conjunction with a scenario XML file, create a text file, that reproduces the key elements of the scenario in a simple text file, using a hex-numbering scheme to identify locations of forces and terrain.
- The `battle2summaryHTML.xsl` file will, when processed in conjunction with a scenario XML file, extract some of the key elements of the scenario into a simple HTML file, summarising forces and terrain.
- The `battle2colorTerrain.xsl` file will, when processed in conjunction with a scenario XML file, create a map layout in SVG, suitable for printing a stand-alone map, showing no forces or any other markings i.e. looking much as the **Battle Cry** board would do after terrain setup.

How to Create Alternative Display Formats

It's possible that the map display created by the XSL file provided with **HexBattle** will not display your scenario exactly the way you would like. In this case, there are four options available to create a more 'customized' layout, presented roughly in order of increasing complexity:

1. Modify the terrain and troop image files - perhaps the ones supplied are not to your liking... if you do this, you will need to keep the new files the same dimensions as the others, otherwise the map will not display properly (unless you also alter the XSL file - see below)
2. Modify the display colors and fonts - you will need to alter the stylesheet section that appears at the top of the `battle2colorSVG.xsl` file. (**Caution:** SVG is far less 'tolerant' of incorrect formatting in CSS than is the case for HTML - poor syntax may lead to the SVG not being displayed at all!)
3. Modify the layout portions of the `battle2colorSVG.xsl` file - to do this, you will obviously require some knowledge of XSL and, quite possibly, SVG; you will then be able to change any aspect of the map appearance. If you believe you have made a substantial improvement to the original, please send through your [feedback](#).
4. Create your own XSL files for other display formats - for example, another type of map layout altogether, or an HTML-only version.

Current Limits and Future Enhancements

Known current limits and 'issues' include:

- The current algorithms used for creating the SVG do not always resolve all the clashes around text overlaps. This can be improved further.
- At present, some new scenarios use additional notation not present in the original scenarios. Not all of these new types of notation are catered for.

Possible future options include:

- New XSL files for other display formats (e.g. a full HTML version; a black-and-white 'sketch' map)
- SVG allows for animation - it is certainly possible to use an XML file to 'track' the movement and activity of troops during a battle, and then use SVG to create a dynamic "battle animation". This would take some work, but would be very cool! Ideas are welcome...

Feedback

As is the case for any other new tool, **HexBattle** can always be improved and refined. I would welcome feedback from anyone who uses it, or has suggestions for improving its use. Please email me - boardgamesbook at yahoo dot com.

I would also be interested in collaborating to extend **HexBattle** to store information for other hex-based wargames that need to track and display information for multiple scenarios.

Acknowledgements

HexBattle would not have been possible - or even necessary! - without the inspiration provided by all the creators of **Battle Cry** scenarios. In addition, the PNG terrain files have not been created from scratch by myself, but represent 'extracted and tidied' images from a number of the existing scenario images, created by various authors.

Resources

XML Resources

- [XML in 10 points](#) is a brief overview of XML from [W3C](#)
- [XML.com](#) has extensive resources for working with XML
- [Free XML Tools](#) has links to numerous tools and software for working with XML
- [HTML-Kit](#) is a great HTML/XML editor, if you are working with Windows (or running WINE under Linux)
- The [XT](#) XSLT processor (developed by James Clarke) can be used to transform XML files.

SVG Resources

- [Adobe](#) provides a free SVG-plugin for your browser, as well as training material and demonstrations

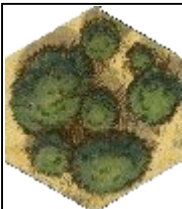


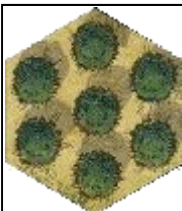

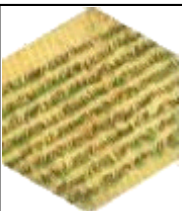
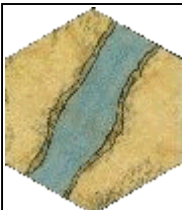

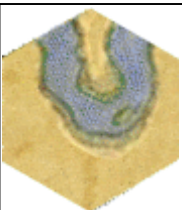
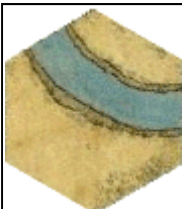
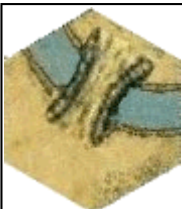
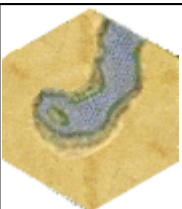



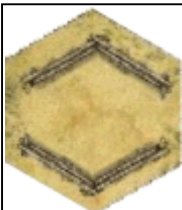


- [SVG overview](#) covers the reasons for, and benefits of, using SVG on the web
- [IBM DeveloperWorks tutorial](#) [requires registration]
- [W3C Official SVG Site](#) has numerous resources and links

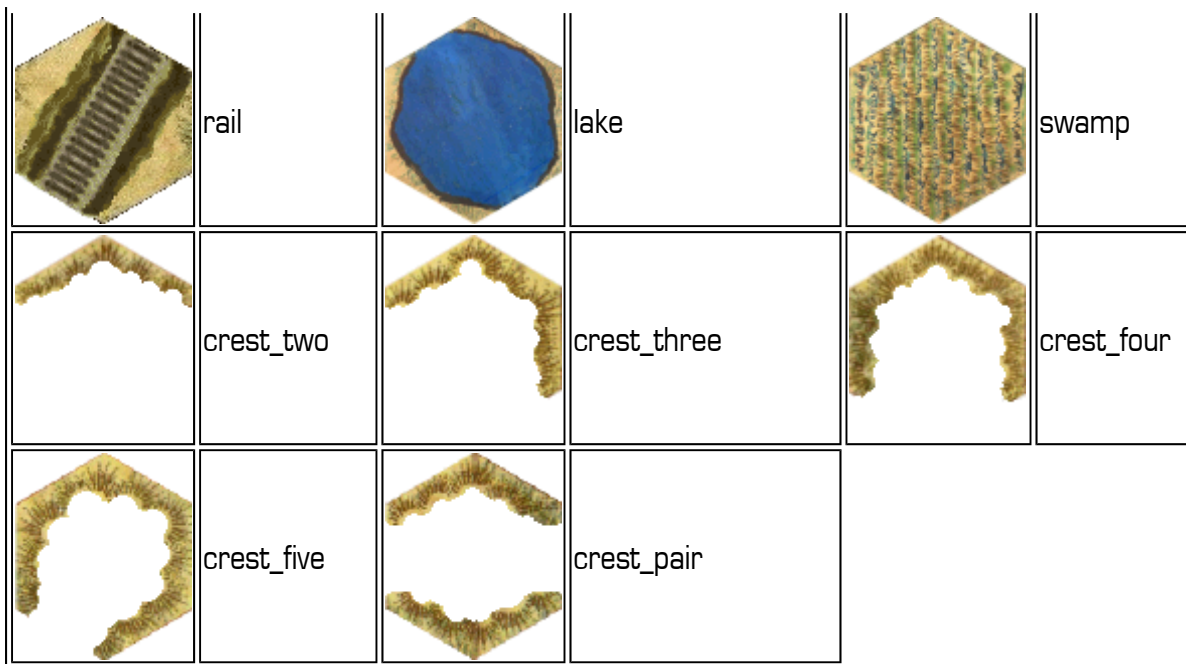
Battle Cry Resources

- [Battle Cry: Command and Colors](#) web site, maintained by John Foley.
- [The Unofficial Battle Cry Web Site](#) is maintained by Rod Lilley.
- [Battle Cry Review](#) by Brian Bankler, includes some scans of sketch maps by Richard Borg
- [The BoardGameGeek](#) has reviews and resources

Terrain Tile Set

This table summarises all the different types of terrain that can be referenced in an XML scenario file.

Tile	Name	Tile	Name	Tile	Name
	woods		hill		buildings
	orchard		rough		field
	river_straight		river_straight_bridge		river_bend
	river_curve		river_curve_bridge		river_end
	fence_one		fence_two		fence_three
	fence_four		field-works		blank



Three Minute Guide to XML

Overview

XML (Extensible Markup Language) is a system for 'marking up' the content of text or data files, in such a way that both humans and computers can easily make sense of it. The purpose of XML is to allow you to focus on describing what the information **means**, rather than how it **appears** (as is usually the case, say, in a word processor, or regular web page).

XML is stored in plain text files. An XML Processor can be used to convert the XML into HTML (the language used by web browsers) for display on a web site. XSLT (Extensible StyleSheet Language Transformations) is often used for this conversion process. The same XML could also be converted into other formats, for example PDF (the format used by Adobe Acrobat), RTF (used by many word processors) or even another kind of XML document! The key point is that information is **stored** in a single format (XML), and can then be readily **presented** in a variety of other formats, without altering the source information.

Notation

The notation used for an XML document is that of 'nested tags'. A **tag** is small bit of information which is NOT displayed, but which can be used by any program working with the document.

A tag appears as a piece of text between two angled brackets, such as: <name>. Tags usually come in pairs, to indicate the start and end of the text that they are marking, for example: <name>Derek</name>. The end tag is shown by the '/'. Some tags are singular and indicated by <tag/>. XML tag names are case-sensitive, and cannot contain spaces.

A nested tag is one which appears inside another (much in the same way as folders can be 'inside' each other on a directory). For example:

```
<name><first>Joe</first><last>Soap</last></name>
```

Another, more readable layout of this same information appears below. It makes no difference from an

XML "point of view" which you use...

```
<name>
  <first>Joe</first>
  <last>Soap</last>
</name>
```

It should be noted that each tag **must** be completely enclosed by another. Any text which does not appear inside tags will cause a problem with the document, except for 'white space' (tabs, spaces or blank lines) which is ignored.

Attributes

Tags can also have additional information associated with them, stored in **attributes**. An attribute is enclosed inside a tag, and consists of a name, followed by an '=', followed by a value, enclosed in "". Different attributes inside the same tag are separated by spaces. For example,

```
<name staff_number="123">
  <first>Joe</first>
  <last>Soap</last>
</name>
```

The 'staff_number' is an attribute of <name> and has a value of '123'. Attributes are useful in providing support information for a tag which, although it might never be displayed, is often used by programs working with the XML document, to aid with identification, selection and formatting.

Special Characters

The consequence of using XML is that certain characters are 'reserved' for use by XML only, including the < and the > and the & characters. These characters are represented in XML documents using the notation &XX; where 'XX' are code letters, as follows:

- gt - greater than (>)
- lt - less than (<)
- amp - ampersand (&)

Layout

All XML documents are "declared" by inserting a special <?xml version="1.0"?> tag at the very top of the document. This allows other programs to recognize and work with the file on the basis that adheres to the XML conventions.

While there can be a number of tags, nested to any depth, inside in an XML document, there must be only one pair of 'outer' tags, which enclose all the others. This is called the **root** tag. So, an example of a small, but complete, XML document would look like:

```
<?xml version="1.0"?>
<staff>
  <name staff_number="123">
    <first>Joe</first>
    <last>Soap</last>
  </name>
  <name staff_number="321">
```

```
<first>Jane</first>  
<last>Smith</last>  
</name>  
</staff>
```

Here, the `<staff>` tag is the 'root' of the document.