# INTRODUCTION

As a long time minature gamer, I found that the preparation of readable and consistent maps was one of the more daunting tasks. Doing them by hand led to inaccuracies and they soon were destroyed as people marked them up. What was needed was a computerized method of drawing and then producing maps. I first tried MS-Powerpoint, and found that, while useable, it was a bit cumbersome, not to mention expensive. MappingBoard (henceforth abbreviated MB) looked like a solution to my problems but it came with NO documentation, and some of the things it does are not obvious, so I initially gave up in frustration. However, MB looked to be so powerful that I finally took the plunge to learn how to use it. I thought that as I did so, I would write some documentation to make it easier for people who are following behind. I hope that this effort will help others. This documentation is for version 9.2 (a beta version as of DEC2012) but most of the comments apply to 9.1.1 as well. I am passing it for review to the author of the MB software, Matthew Tyler, who hopefully will help me avoid glaring inaccuracies. It does not pretend to be exhaustive, but is more in the order of a users guide. As it is a “work in progress”, I welcome comments and suggestions and will incorporate them to make it better.

Jim Baker

Fire and Fury Games

Dec 2012

[INTRODUCTION 1](#_Toc346263479)

[MAPPING BOARD SCREEN 9](#_Toc346263480)

[Menu 9](#_Toc346263481)

[Template Area 9](#_Toc346263482)

[Layer Area 9](#_Toc346263483)

[Map Edit Area 9](#_Toc346263484)

[Property Area 10](#_Toc346263485)

[GENERAL CONCEPTS 11](#_Toc346263486)

[Drag and Drop 11](#_Toc346263487)

[Objects 11](#_Toc346263488)

[Selecting Objects 11](#_Toc346263489)

[Creating New Objects 11](#_Toc346263490)

[Drawing Objects 11](#_Toc346263491)

[Linear Objects 11](#_Toc346263492)

[Area Objects 12](#_Toc346263493)

[Symbols 12](#_Toc346263494)

[Pens 12](#_Toc346263495)

[Colors 13](#_Toc346263496)

[Predefined colors 13](#_Toc346263497)

[RGB 13](#_Toc346263498)

[Position 14](#_Toc346263499)

[Rotation 14](#_Toc346263500)

[DrawOrder 14](#_Toc346263501)

[Textures/Images 14](#_Toc346263502)

[Transparency 16](#_Toc346263503)

[Manipulating the shape of objects 17](#_Toc346263504)

[Extending the box 17](#_Toc346263505)

[Path points. 17](#_Toc346263506)

[Moving path points 17](#_Toc346263507)

[Deleting path points 18](#_Toc346263508)

[Adding path points 18](#_Toc346263509)

[Curved 18](#_Toc346263510)

[Closed 18](#_Toc346263511)

[Tension 18](#_Toc346263512)

[A Caveat 18](#_Toc346263513)

[Layers 19](#_Toc346263514)

[Creating a layer 19](#_Toc346263515)

[Deleting a layer 19](#_Toc346263516)

[Selecting a layer 19](#_Toc346263517)

[Moving objects between layers 19](#_Toc346263518)

[MENU AREA 20](#_Toc346263519)

[Dropdown Menus 20](#_Toc346263520)

[File 20](#_Toc346263521)

[Open, Save, Save As 20](#_Toc346263522)

[Print 20](#_Toc346263523)

[Export to Bitmap 20](#_Toc346263524)

[Edit 20](#_Toc346263525)

[Undo/Redo 20](#_Toc346263526)

[Copy 20](#_Toc346263527)

[Delete 20](#_Toc346263528)

[Bring to Front/Take to back 20](#_Toc346263529)

[Group/Ungroup 20](#_Toc346263530)

[Flip Horizontally/Vertically/Both 21](#_Toc346263531)

[Feature 21](#_Toc346263532)

[Move to Layer 21](#_Toc346263533)

[Join Features 21](#_Toc346263534)

[Use Default Style 21](#_Toc346263535)

[Tools 21](#_Toc346263536)

[Options 21](#_Toc346263537)

[Help 21](#_Toc346263538)

[About 21](#_Toc346263539)

[Menu Toolbar 21](#_Toc346263540)

[New Map/Open/Save/Export to Image 21](#_Toc346263541)

[Open Object/Save Object 21](#_Toc346263542)

[Undo/Redo/To Front/To Back/Group/Ungroup/Delete/Copy/Flip Horizontal/Flip Vertical 22](#_Toc346263543)

[Canvas 22](#_Toc346263544)

[MAP EDIT AREA 23](#_Toc346263545)

[Map canvas 23](#_Toc346263546)

[Map Canvas Properties 24](#_Toc346263547)

[Draw Freehand 24](#_Toc346263548)

[Zoom 24](#_Toc346263549)

[Grid 25](#_Toc346263550)

[GridColor 25](#_Toc346263551)

[GridLineWidth 25](#_Toc346263552)

[Grid Transparency 25](#_Toc346263553)

[Map 25](#_Toc346263554)

[BackgroundImageStyle 25](#_Toc346263555)

[GridHeight/GridWidth 25](#_Toc346263556)

[GridType 25](#_Toc346263557)

[Highlight Color 25](#_Toc346263558)

[MapBackGroundColor 25](#_Toc346263559)

[MapBackGroundImage 25](#_Toc346263560)

[MapHeight/Mapwidth 26](#_Toc346263561)

[Margins 26](#_Toc346263562)

[TEMPLATES 27](#_Toc346263563)

[Drawing Template Menu 27](#_Toc346263564)

[Common Drawing Properties 27](#_Toc346263565)

[Rect 27](#_Toc346263566)

[Line 27](#_Toc346263567)

[Circle 28](#_Toc346263568)

[RRect 28](#_Toc346263569)

[Text 28](#_Toc346263570)

[Image 31](#_Toc346263571)

[Arc 31](#_Toc346263572)

[Poly 31](#_Toc346263573)

[Pen 31](#_Toc346263574)

[Viewport 31](#_Toc346263575)

[Mapping Template Menu 33](#_Toc346263576)

[Basic types of Mapping objects (also applies to Textured Mapping) 33](#_Toc346263577)

[Linear 33](#_Toc346263578)

[Area 33](#_Toc346263579)

[Symbols 33](#_Toc346263580)

[Linear Mapping Objects 34](#_Toc346263581)

[Road 34](#_Toc346263582)

[River 34](#_Toc346263583)

[Bridge 34](#_Toc346263584)

[Rail 34](#_Toc346263585)

[Wadi 34](#_Toc346263586)

[Escarpment 34](#_Toc346263587)

[Hedge 34](#_Toc346263588)

[Area Mapping Objects 35](#_Toc346263589)

[Hill 35](#_Toc346263590)

[Rise 35](#_Toc346263591)

[Wood 35](#_Toc346263592)

[Town 35](#_Toc346263593)

[Lake 35](#_Toc346263594)

[Marsh 35](#_Toc346263595)

[Rough Ground 35](#_Toc346263596)

[Symbols 35](#_Toc346263597)

[Town Block 35](#_Toc346263598)

[Town Block 2 35](#_Toc346263599)

[Tabletop Template Menu 36](#_Toc346263600)

[Linear 36](#_Toc346263601)

[Road 36](#_Toc346263602)

[River 36](#_Toc346263603)

[Bridge 36](#_Toc346263604)

[Rail 36](#_Toc346263605)

[Area 36](#_Toc346263606)

[Hill 36](#_Toc346263607)

[Rise 36](#_Toc346263608)

[Woods 36](#_Toc346263609)

[3-d Hill 37](#_Toc346263610)

[3-d Rise 37](#_Toc346263611)

[Textured Mapping Template Menu 38](#_Toc346263612)

[Area 38](#_Toc346263613)

[3-d Hill 38](#_Toc346263614)

[3-d Rise 38](#_Toc346263615)

[Symbols Template Menu 39](#_Toc346263616)

[Military Symbols 39](#_Toc346263617)

[General Properties 39](#_Toc346263618)

[Dash Style 39](#_Toc346263619)

[Fill Color/Filled 39](#_Toc346263620)

[Pen Color/Pen Width 39](#_Toc346263621)

[Rotation 39](#_Toc346263622)

[Transparency 40](#_Toc346263623)

[Symbol 40](#_Toc346263624)

[Type 40](#_Toc346263625)

[Subtype 40](#_Toc346263626)

[Unit Size 40](#_Toc346263627)

[Font Size 40](#_Toc346263628)

[Unit Name 40](#_Toc346263629)

[Parent Name 40](#_Toc346263630)

[Strength 40](#_Toc346263631)

[Unit Count 40](#_Toc346263632)

[IsHeadQuarters 40](#_Toc346263633)

[Fill Level/SecondFillColor 40](#_Toc346263634)

[Crosshairs 40](#_Toc346263635)

[Trig Point 41](#_Toc346263636)

[TownBlock 41](#_Toc346263637)

[Small Town Block 41](#_Toc346263638)

[Church 41](#_Toc346263639)

[Square House 41](#_Toc346263640)

[Small House 41](#_Toc346263641)

[Bunker 41](#_Toc346263642)

[Anti-Tank Ditch/Single Concertina/Double Concertina/Triple Concertina/Dragons Teeth 41](#_Toc346263643)

[Custom Template Menu 42](#_Toc346263644)

[Tarmac Road 42](#_Toc346263645)

[Feature Template Menu 43](#_Toc346263646)

[Custom Objects and Features 44](#_Toc346263647)

[Modifying existing objects. 44](#_Toc346263648)

[Saving a “feature” 46](#_Toc346263649)

[Saving a Template object 47](#_Toc346263650)

[Pen Properties 48](#_Toc346263651)

[The Center Line 50](#_Toc346263652)

[Pen Drawing Order 50](#_Toc346263653)

[Pen Properties 50](#_Toc346263654)

[Width 50](#_Toc346263655)

[PenColor 50](#_Toc346263656)

[PenImage 50](#_Toc346263657)

[OffsetX 51](#_Toc346263658)

[OffsetY 51](#_Toc346263659)

[StartCap/EndCap 52](#_Toc346263660)

[Fade Edges 52](#_Toc346263661)

[Dash 52](#_Toc346263662)

[Dash 52](#_Toc346263663)

[Dot 52](#_Toc346263664)

[DashDot 53](#_Toc346263665)

[DashDotDot 53](#_Toc346263666)

[Custom 53](#_Toc346263667)

[DashCap 55](#_Toc346263668)

[A combination example, showing offsets and dashes: 55](#_Toc346263669)

[Compound Pen 56](#_Toc346263670)

[Combining Pen Effects 58](#_Toc346263671)

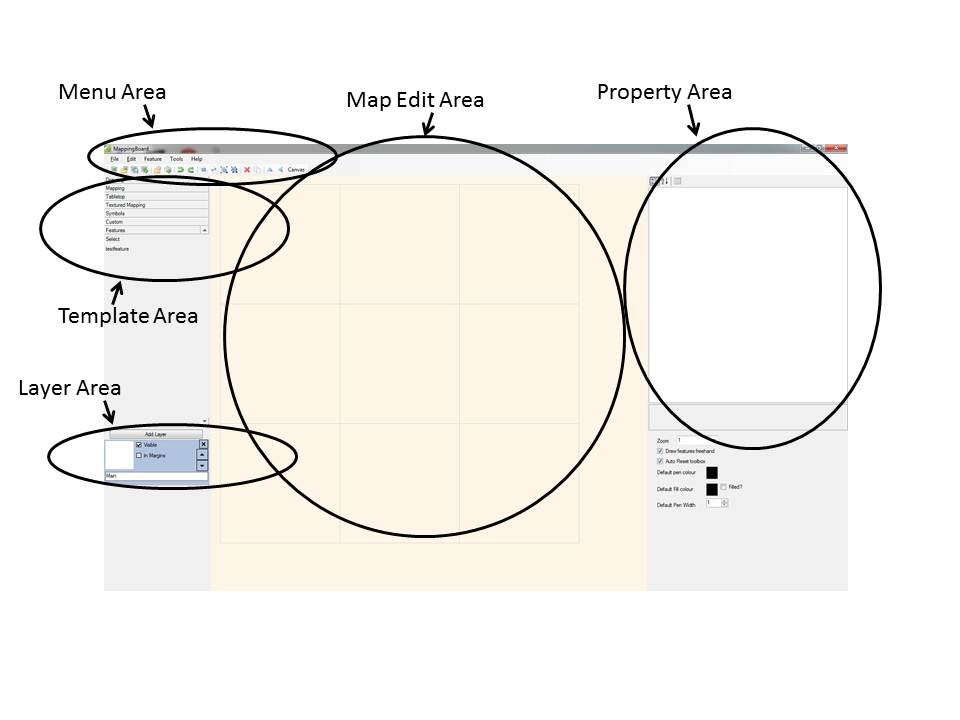
[Building a Basic Worm Fence 59](#_Toc346263672)

[Faking an offset - a dash pattern trick 61](#_Toc346263673)

[The Final Worm Fence 62](#_Toc346263674)

[Sharing the Wealth 62](#_Toc346263675)

# MAPPING BOARD SCREEN



The MB screen is divided into 5 basic areas:

## Menu

The menu is at the top left. It contains both menu dropdowns (File, Edit, Feature, Tools, Help) and a toolbar with icons that invoke commonly used tasks. Many of these functions will be familiar to most computer users, while a few are specific to MB.

## Template Area

This area is to the left of the screen. There are several dropdown lists that allow you to select objects from predefined templates to be put on your map.

## Layer Area

Below the template area is the layer area which allows you to organize your map into different layers.

## Map Edit Area

The center of the screen is where you build your map. It shows as 6x6 unit area of the map. If your map is larger than this, scrollbars will appear to allow you to access the full map. An important thing to realize is that (in version 9.1.1 at least) the Edit Area is ***NOT*** WSIWYG (what you see is what you get). To see what will be produced you need to do File/Print, but as this invokes a Print Preview by default, you don’t need to worry about wasting paper.

## Property Area

The Property area is on the right of the screen. This allows you to adjust the appearance of the currently selected object.

# GENERAL CONCEPTS

Before we get started with all of the details, there are some general concepts which need to be explained.

## Drag and Drop

Rather than make the assumption that you know what this means, a quick explanation is in order. When you position your cursor in the map area you do a drag-and-drop operation by pressing on the left mouse button and holding it down while you move it to another place. When you release the mouse button the operation is complete. What will happen will depend on what you are doing. If you are drawing a linear object, the drag-and-drop is used to define the end points or path. If you are creating a symbol, you will define the location of the object. If you have selected an object for moving, it will be dragged from its initial position and dropped in the new position (hence the name of the operation).

## Objects

You build your map by putting *Objects* on it. Each object has a set of *properties* which are set to different *values* to define how the object displays. While each object has a unique set of properties, there are certain that are common to all of them and perform the same way.

### Selecting Objects

Left clicking on an object will select it for editing and possibly moving (if the cursor is the pointing finger). Its box will be highlighted in the map area and it will turn a shade of light red. If there are several objects in the same area, clicking will select the topmost. You can hold down the shift key and continue clicking to cycle through the objects in the same area.

To select several objects in the same area at once, create the drag/drop a box around them. When you release the mouse key the selected objects will all turn light red. You can then group them or move them as a group.

### Creating New Objects

The first step is to select an object type from the template area. There are many different types of objects, but they can be roughly defined (by me - not necessarily a technical definition) as “drawing”, “linear”, “area”, or “symbol”. The general process is to select an object type and then drag it to the place on the map edit area where you want it to appear. A new object of that type will be added at that location.

#### Drawing Objects

These are things such as rectangles, circles, etc. that do not necessarily have any special appearance properties associated with them.

#### Linear Objects

These are special mapping objects that have a distinct appearance predefined, such as color and pattern. You draw a line on the map where you want the object to be and it will follow this line (assuming freehand drawing is enabled). Note that some of these objects, such as escarpments, have a distinct side associated with them. Drawing them from left->right will put the pattern on a different side than if you draw them right-> left (you can used the flip horizontal/vertical actions to change this simply). Most linear objects are built from multiple “pens” (see below).

#### Area Objects

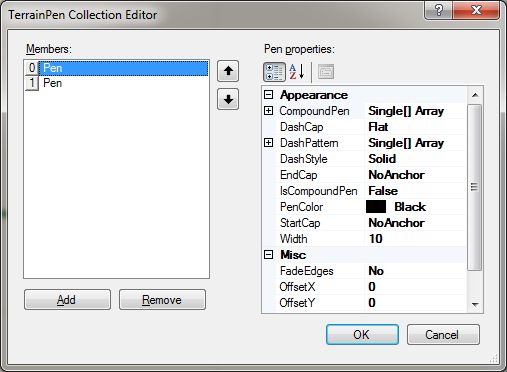
These represent an area of a specific type of terrain. When you draw them on the map, it will close your curve to create a distinct area, often with a default pattern or color filling the area.

#### Symbols

Symbols create a symbol on the map at the position you specify. Some of these have predefined sizes, which cannot be changed, even if you change the size properties or draw a bigger box. Others (see the concertina objects on the symbol template) have a symbol that repeats within the object. For these you can modify the “repeats” property to the number of repititions to make the objec t longer or shorter.

### Pens

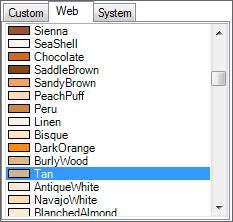
Many of the linear object types have “pens” defined in the properties area, when you define an object, you can think of one or more pens drawing along the lines you define on the map. When we get to the specific objects, we will find that to change the appearance you will need to modify the “pen” properties instead of the more obvious general properties. When an object is built by pens, you will need to modify the pen collection:



Each pen has its own pattern, and they combine to produce what you see on the map. At the end of this documentation we will go into gruesome detail about pen properties.

### Colors

Many of the properties in MB define colors. Colors can either be chosen from a list of predefined colors or as a specific set of RGB (red green blue) numbers.



#### Predefined colors

There are 3 sets of predefined colors “custom”, “web”, and “system”. To change a color to a predefined color, click on the dropdown arrow to the right of the property, select one of the tabs and then click on the desired color.

#### RGB

Rather than using a predefined color, type in 3 numbers 0-255 separated by commas. The first is the red intensity, the second green, the third blue. For example, bright red is 255,0,0, white is 255,255,255, and black is 0,0,0. If this corresponds to one of the predefined colors, the system will substitute the name. All of the examples above will translate to a precise name. However, a color which does not correspond to a predefined color (for example 255,14,200- a bright purple) will leave the 3 numbers in the property and still work.

### Position

Position is defined as (x,y) coordinates relative to the map, with 0,0 being the upper left corner of the map. These are the PosX and PosY properties. Every object has a point on it from which the position is measured. Symbols usually determine their position from the top/center, linear and area features measure them from the upper left corner, and there may be other positions used. In any case, you can move objects by selecting them and dragging them to their new positions using drag-and-drop.

When an object is selected for moving, the cursor will turn into a pointing finger:



Alternatively, you can modify the PosX and PosY numbers in the properties area for precise control.

### Rotation

Some objects have a “rotation” property. This is measured counterclockwise relative to the horizontal in degrees, so a rotation of 30 would be tilted down 30 degrees to the right. Note that this is most often relevant to objects that normally line up along horizontal and vertical lines, but can be specified for other objects as well. This is very useful if you want to specify several houses or other symbols along a road so that their edges line up with the road. You will see that the selected “box” for a rotated object is still horizontal and vertical, but that the picture is rotated within that box.

### DrawOrder

This property determines the “z-order” (what objects appear on top) when you produce the final map. It also facilitates drawing of certain features where you want the borders to merge, such as crossroads and river forks. This can be confusing, because the map edit area does not show what will print (at least in MB version 9.1.1). Using the Edit->Bring To Front or Edit->Take to Back will affect the Edit Area but will not affect the final product, which is determined by DrawOrder.

The best way to see the effect of DrawOrder is to experiment as follows:

1) From the Mapping Tab of the Template area, select and draw a road (default DrawOrder 2), a river (DrawOrder 1) and a railroad (DrawOrder 3) so that they cross each other. Then use File->Print to produce a Print Preview. You will see that the preview shows the Rail on top of the Road and the Road on top of the river. Now select the river and use Edit->Bring to Front so that it looks like it is on top of the other two. Produce another print preview and you will see that the river still prints below the other features.

2) Draw two roads that cross each other and produce a Print Preview. You will see that the crossroads has been merged in the final map.

### Textures/Images

While it easier to fill an area with a solid color, it is often more attractive to have a textured background. Textures are just image files (.jpg, .png) that are used. Only a very few images come with the basic MB installation, but there are a lot of them free for download on the web. The MB website lists a couple of good sources:

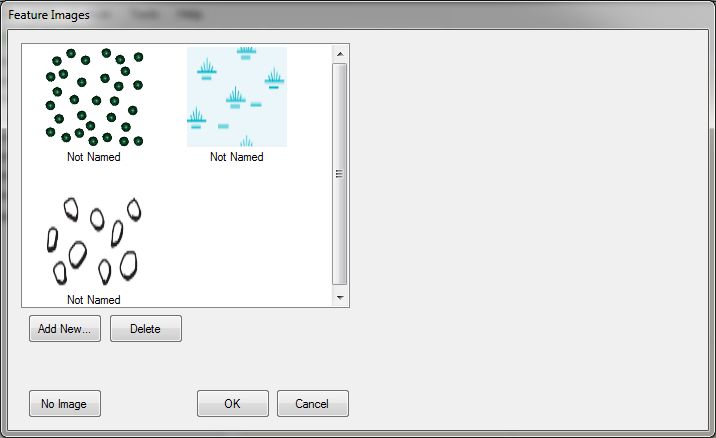
<http://www.davegh.com/index.php> David Gurrea

<http://www.absolutecross.com/downloads/textures> Absolute Cross

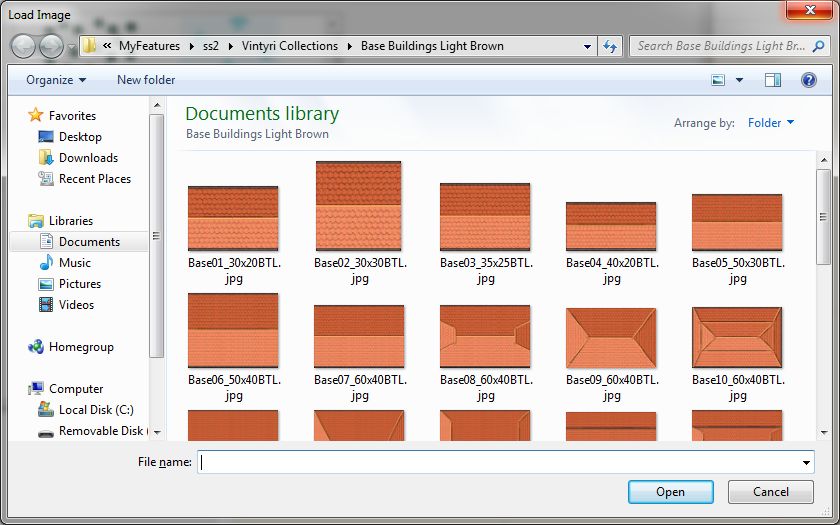
<http://www.vintyri.org/vintyri/vccindex.htm> The Vintyri Cartographic Collection - this one is especially useful, with both textured backgrounds and top-down buildings.

Anyway, download the image(s) you need. By default, MB 9.2 looks for them in the directory specified for “My Features” (see Tools->Options Preferences), but the lookup procedure allows browsing, so you could put them somewhere else or in a subdirectory to organize them.

When you want to specify an image, either as a background or actual image, a dialogue box will come up asking you to select an image.



It contains images that are already available, but also allows you to “Add New”, where you can browse to find an image file. Here is a lookup for some of the excellent Vintyri buildings:



How this image is used depends on the type of object, but you usually have to change another property to tell the map that it is supposed to use an image instead of normal color fill. For example, the Hill object has “FillImage” and “FillMode” properties.

### Transparency

The Transparency property indicates whether the background will show through an object. Low numbers mean that the object is opaque, while high numbers give preference to the background.

## Manipulating the shape of objects

This applies to almost all of the mapping objects, both linear and area, and some of the drawing objects.

### Extending the box

When you select an object, the rectangle that contains the object appears on the map. Using your mouse, you can change the size and shape of most objects by dragging the borders of the rectangle.



### Path points.

If you create almost any complex area or linear figure, the border of the area or the path of a linear figure is defined by a series of points. When the object is selected, these points appear as small boxes with a pink center. If you don’t get the shape of you object correct the first time, you can change its shape or path by manipulating these points. Here is a selected Road object.

The path points look like this:

### Moving path points

You can change the position of a single point by dragging it within the figure. When you select an object and move your mouse over it, your cursor first appears as a hand with an index finger extended.



If you hold down the left mouse button and move your mouse, the entire object will move.

However, if you position the mouse over a single point, the cursor will change to crossed double-headed arrows:



if you hold down the left mouse button and move the mouse while this symbol is active, the path point will move within the object and change the path/shape of the object.

### Deleting path points

To delete a path point, position your cursor over the point to be deleted and right click. 4 options will come up:

Delete/Delete Point/Add to template/Update Template

Select “Delete Point”. It will remove that point. In 9.1.1, even though the point is deleted, it looks like nothing changes and you need to unselect and reselect the object. However, this seems to have been fixed in 9.2.

Frankly, this menu is a bit dangerous, as Delete/Add to Template/Update Template all apply to the entire object. It would have been better to have the ability to delete/add a point in a separate menu. However, I think this is a function of the underlying tool which was used to create MB.

### Adding path points

While it would have been logical to delete and add points from a single menu, this is not how it is done. Instead, to add a point, you must select the “small green box” symbols that lie between path points:



When you position your cursor over the green box, it will change to a + symbol. If you left click with this symbol active, a new path point will be created at that spot. Unlike deleting a point, a new point becomes immediately active. This is a bit sensitive, as clicking without the + symbol will have no effect.

### Curved

This property describes how the path between the path points is drawn. “False” will give you straight lines. “True” will give you a smooth curve.

### Closed

This will define whether the last point is connected to the first. By default, linear objects have this set to “false”, while area objects have it set to “true”, but you can change this if you want.

### Tension

This defines the “curviness” of the line between points as a number between 0 and 1. A tension of 0 is a straight line, while 1 is very curvy (try this out for some interesting effects).

### A Caveat

Manipulating an object shape by dragging points can occasionally be frustrating, especially in curvy objects, as the program will adjust the curve as you move the point, and sometimes this has “interesting” effects. Also, Delete Point doesn’t work very well in 9.1.1 (but seems to in 9.2). Don’t give up - “practise makes perfect”.

## Layers

By default, every feature you put on the map is contained in a single layer, named “Main”. In a complicated map, this can make it hard to select/adjust small features without accidentally grabbing something else and messing it up. To help with this, MB allows you to organize your map in “layers”. You can put your basic terrain (hills/rivers/etc.) in one, and then smaller features (houses, hedges, etc.) in another. These are all merged when you print the map.

### Creating a layer

The “Add Layer” button on the lower left allows you to add a new layer and change its name. The “visible” and “in margins” properties are used in printing.

“Visible” might be extremely useful if you were creating multiple versions of a map for different players in a game. You could put starting positions for each side in different layers while you edit and then change the visible property when you print.

“In Margins” is also quite useful. If this is false, the export process will cut the layer off at the edges of the map. You can be untidy at the edges (and therefore quicker) so roads, hills, etc can freely go over the edge of the map and be cut off in the export. You can also put labels, deployment areas, etc. in a layer that has “In Margins” set to true and these will appear when exporting.

### Deleting a layer

In MB 9.1.1 you are unable to delete a layer, but in 9.2, there is an X box that will allow you to delete it.

WARNING - deleting a layer is not reversible by ctrl-Z and will delete the objects in it. Also, you need to reselect an active layer after you delete one.

### Selecting a layer

Click on the layer you want in the lower left. When you have a layer selected, you will only be able to select objects in that layer (exception - the underlying map canvas seems to be available in all). This is a bit confusing, because the layer name is at the bottom of the selected area. The selected layer is highlighted in light blue.

### Moving objects between layers

Select the object in its source layer and then choose the Feature “Move to Layer” option from the menu bar. It will then be in the new layer.

# MENU AREA

## Dropdown Menus

### File

#### **Open, Save, Save As**

These have their conventional meanigs. If you open a new map before saving the one you are working on, it will ask you to save.

#### Print

By default this brings up print preview, which is very useful as some of the features of Mapping Board will not be seen until you print (for example, the lines in road junctions are smoothed out).

#### Export to Bitmap

This will create an image file from your current map, in the format and location you specify.

### Edit

#### Undo/Redo

These have the conventional meaning (and key shortcuts) that they do for most Microsoft editing programs. It affects the last action you performed.

#### Copy

This creates a copy of the currently selected object near the original object. In 9.1.1, copying works correctly for most of the objects, but some, primarily those in the symbols menu, did not have all of the properties filled in. However, the object is created and all you need to do is select it and move it or expand it and the missing properties will be correctly changed.

#### Delete

Deletes the currently selected object

#### Bring to Front/Take to back

These affect the z-order (which object appears on top) of the selected object *in the Map Edit Area only*. While occasionally useful to allow you to grab an object while editing, this can be quite confusing, because the z-order in the finished product is determined by the Draworder property of each object, with the lowest numbers appearing in the back, *no matter what object appears on top in the Map Edit Area*. The final picture of what appears can be seen by using the print preview in the File dropdown.

#### Group/Ungroup

These allow you to group together multiple objects so that they can be moved as one, or splitting a previously created group. Select multiple objects by creating a box around them. Select a point and then drag/drop a box area around the desired objects. When you release the mouse, objects in the area will be selected and can then be grouped. Grouped objects can be treated as one object in the Edit Area.

Note - I could find no easy way of selecting/unselecting individual objects while leaving others selected. The method used in MS products (ctrl-click on individual objects) doesn’t work. This makes this feature a bit less useful than it might be.

#### Flip Horizontally/Vertically/Both

Works on the currently selected object to change the orientation.

### Feature

#### Move to Layer

This is used to move the selected object to a different layer.

#### Join Features

This merges multiple objects. I tried this and had some rather unpredictable results, as all the properties seem to be merged. Also, as ctrl-Z (undo) does not reverse the effects of this, be careful when using it.

#### Use Default Style

If you make changes to the properties of an object, this reverts back to the default for the object type.

### Tools

#### Options

Specifies the default libraries for your stored maps, features, and exported bitmaps.

### Help

#### About

Shows the current version and copyright info.

## Menu Toolbar

Most of these icons give quick access to similar functions in the dropdown menus.

### New Map/Open/Save/Export to Image

These perform the same functions as the corresponding entries in the File Dropdown menu

### Open Object/Save Object

One of the key ideas of MB is “reuse”, where you specify the properties of an object and then save it so you don’t need to recreate your work. MB allows you to do this in two ways, one of which is invoked by these two icons. Save Object allows you to save the currently selected object as a “feature”. By default, features are saved as files with a “.feat” extension in the directory specified in the Tools->Options menu entry for “features”. If you want to use a feature, use “Open Object” to select a file and then click on the Map Edit area. This will create an exact copy of your saved object at the position that you used when you created it. Also, available features appear in the “features” dropdown of the template area. If you select one from there, you click on the map at the position where you want it to appear. The other method of reuse is to create custom objects and put them in your object template toolbars, which essentially makes them a permanent part of your MB application. MB currently has no easy way to easily delete objects you add to your template, so unless you plan to use a saved object frequently, saving an object as a feature gives you more control, as feature files can be deleted using normal file handling processes. Also, as .feat files are just normal files, they can be distributed to other MB users.

### Undo/Redo/To Front/To Back/Group/Ungroup/Delete/Copy/Flip Horizontal/Flip Vertical

These perform the same functions as the corresponding entries in the Edit Dropdown.

### Canvas

Selects the Canvas (background) in the Map Edit Area. Clicking on the map somewhere where there are no objects will also do this.

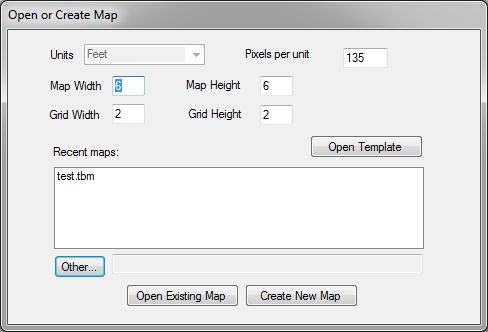
# MAP EDIT AREA

This is where you build your map. Objects are selected and placed on the map and can be moved to where you want them. To see what your map will look like, do a print preview.

## Map canvas

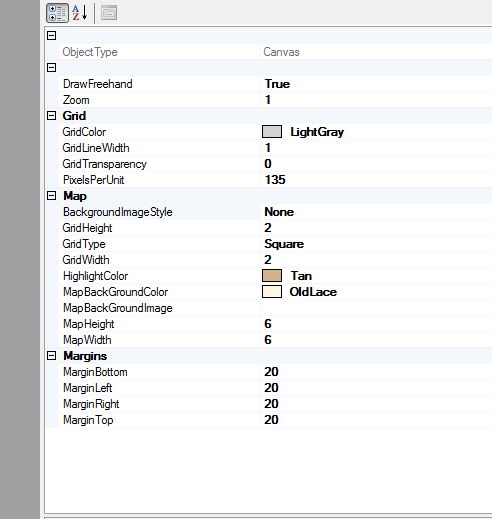
The properties of the map itself are defined in a special object called the *Canvas* which underlies all layers and ultimately contains all objects. Like other objects, the map canvas has certain properties that can be modified to give special effects and appearances.

When you enter MB for the first time, it will ask you if you want to create a new map or work with an old map.

 You specify the intial map size (default 6x6) and grid size (default 2) when you create it. These properties will be the intial values of the blank canvas and can be changed later. The map edit area defaults to a 6x6 square. If the map is larger than this, scrollbars will allow access to the rest of it.

To change underlying map, click anywhere on the map that does not contain a feature. This will access the map canvas. Alternatively, use the “canvas” menu toolbar button.

### Map Canvas Properties



#### Draw Freehand

This is not used in the canvas itself but is used when you draw objects on it. If set to true, the line drawn will follow the mouse as you move it and allows you to create curves. If set to false, you can only draw straight lines.

#### Zoom

This determines how the map is viewed on the screen, but does not affect the map itself. It allows you to do get in close to adjust features that are close together. Default 1.

#### Grid

This allows you control over how the grid looks.

##### GridColor

Adjust the color of the grid line.

##### GridLineWidth

How thick the grid lines are. Default 1 - thin line.

##### Grid Transparency

How prominent the grid lines are. A transparency of 0 (default) makes them prominent, while 100 makes them invisible.

#### Map

##### BackgroundImageStyle

This allows you to decide to use a texture for the entire map. The default of None means that no bitmap will be used. Stretch and Tile say to use an image and are associated with the MapBackGroundImage property. Normal will just display one iteration of the image, which may not be enough to fill the entire map. Stretch will expand the image to fill the entire map if it is not big enough, while Tile will repeat the image until it fills the map. The most useful backgrounds are produced with smooth textures in Tile mode.

##### GridHeight/GridWidth

These two properties define the density of the grid on the map. They are associated with the Mapheight/MapWidth properties. For example, a Mapheight of 6 with a GridHeight of 2 will give you 3 grid lines. These were specified initially when you created the map and can be modified here.

##### GridType

You can choose a Square grid, or two different types of hex grids (columns/rows).

##### Highlight Color

Used in creating 3-D objects.

##### MapBackGroundColor

If you are not using a texture image, this property allows you to select an underlying color for the whole map.

##### MapBackGroundImage

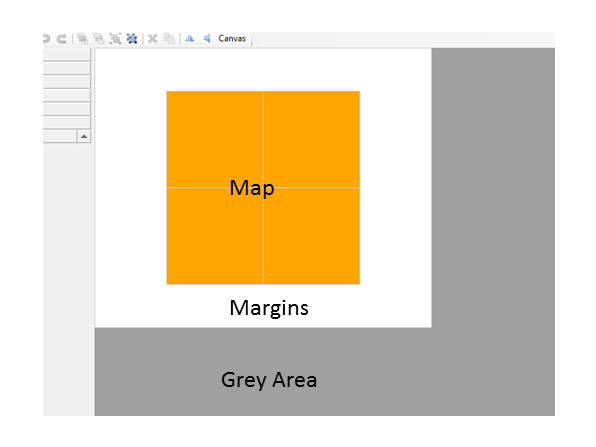
Select an image for the map background, to be used if the BackgroundImageStyle is specified. The properties under this seem to be derived from the image itself and cannot be changed. Note that if you have a large image, it will fill the map area and you will see every detail, which may be distracting. Either choose a smooth image or use a photoeditor to shrink the image so that “tile” can be used to smooth it out.

##### MapHeight/Mapwidth

The dimensions of the map - specified when you create it. These are related to the Gridheight/Gridwidth properties to define the density of the grid.

##### Margins

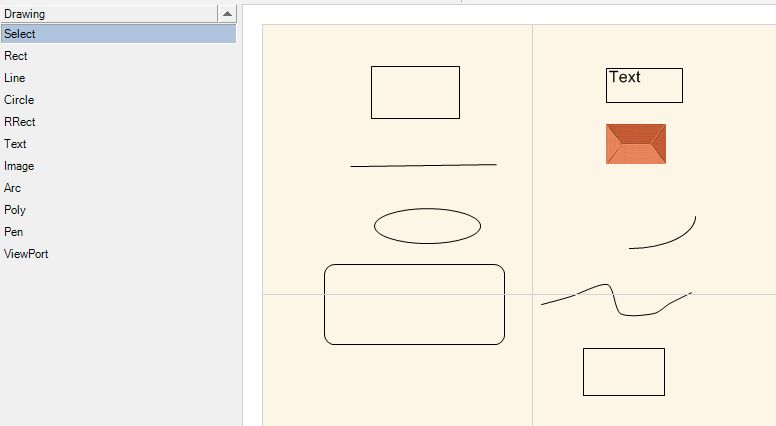
MB has a central map area surrounded by margins, and both will be used when you print out your map. However, margins are special in that objects and text in them will only appear if the “In Margins” property of their layer (see Layers above) is checked. Margins appear as a white area around the map itself (try changing MapBackGroundColor to see the margins more clearly, as in the picture below). By default, there is a margin area of 20 pixels on each side of the map, but you can modify those widths in the Canvas properties. Margins allow you to be a bit sloppy in your main map, as things that spill over into the margins will be cut off. Also, you can specify a layer for things that you want to appear in the margin, such as entry areas and labels, and set “In Margins” to true for that layer. Depending on the size of your map, there may be a dark grey area outside the margins. Even though it looks as though Objects can spill into this dark grey area, this area will not be used.



# TEMPLATES

The template menus contain object types that you can copy to your map. Each object type starts with a basic set of properties, which can be modified to create the effects you need.

## Drawing Template Menu



These are the basic drawing tools, and have very few adornments. You can draw boxes, lines, ovals, and similar features. You generally have control over position, rotation, size, and colors, but not much else. Two of the more useful of these are Text and Image.

### Common Drawing Properties

FillColor and Filled control the interior of the object.

PenColor, PenWidth, ShowBorder control the border of the object, or the line itself if the object is not a closed curve.

Transparency and Rotation have their general meanings.

### Rect

Draws a rectangle.

### Line

Draws a straight line. You have control over the beginning and end of the line through the StartCap and EndCap properties, but these are hard to see unless the penwidth (default 1) is increased. Dash style gives you the ability to draw a dashed line with various patterns.

### Circle

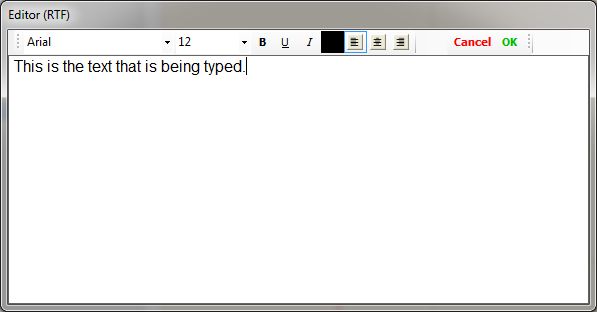
This should more appropriately be called Oval, as it will create a rounded symbol in your drag/drop box and you rarely can get this exact by drawing. If you need a circle, make the Height and Width properties the same.

### RRect

Creates a rectangle with rounded edges.

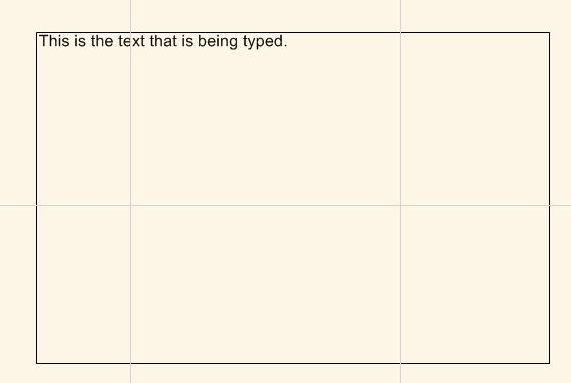
### Text

Allows you to enter text and creates a rectangle object with the text inside. You have control of the text font, size, and value when you create it, as it brings up an RTF editor:

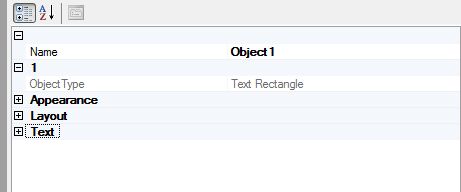


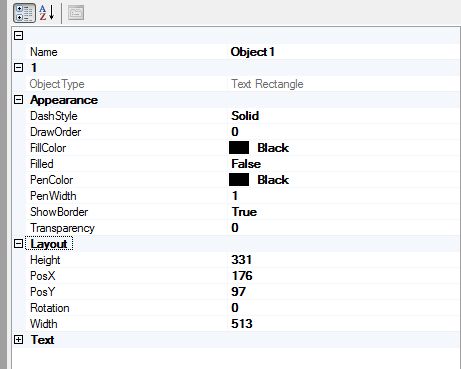
Note that the bold, underline, and italic will apply to the entire text you enter.

Once created, the text will appear in a rectangle on the screen, but you have control over this rectangle in the properties area as well as being able to modify the text itself.

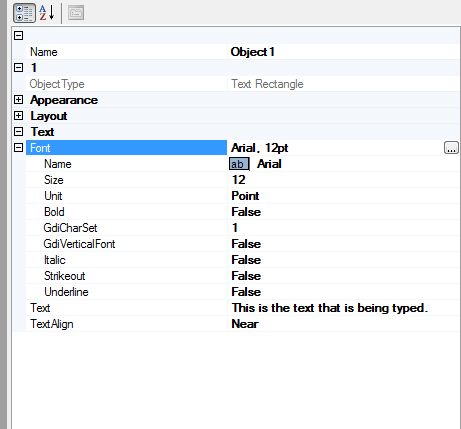


The properties of a text box are grouped into 3 sections “Appearance”, “Layout” and “Text”. To see these properties, click on the + box if they are not expanded.





Appearance and Layout primarily affect the rectangle that enclosed the text. The one property in these sections that affects the text itself is “PenColor”, which defines the color of both the rectangle border and the text. If you don’t want theto see the surrounding rectangle set ShowBorder to false.



The Text section gives you control of what you entered initially in the RTF editor. To change the text that appears, modify the Text property in the text section. You can also change the font, but note that you need to have a font installed to use it or you will get an error (that is far beyond the scope of this document). Most of the common fonts (Arial, Times New Roman, etc.) will work fine.

### Image

Import an external image into the map. This is an extremely cool object, as there are lots of nice images (especially see the Vintyri Cartographic Collection mentioned above) of houses, trees, etc. that you can use. When you specify the image, it will be placed on the map with its default size derived from the image file, which may take up more map real estate than you want. However, you can change this by modifying the Height and Width properties - just be sure to do so proportionately or you will distort the image. For example, if the basic image is 500x400, you should make it 50x40 rather than 50x50.

### Arc

This creates an open curved line based on an oval. Essentially, you see part of the border of an oval, defined by the StartAngle and SweepAngle (specified in degrees, clockwise from 0 degrees to the right). As with the line object, you have control over the start and end of the line. You can fill the arc, which draws a chord from between the start and end point and fills in that area. You can specify negative values for the sweep, which can change which end of the line is considered the “end”.

### Poly

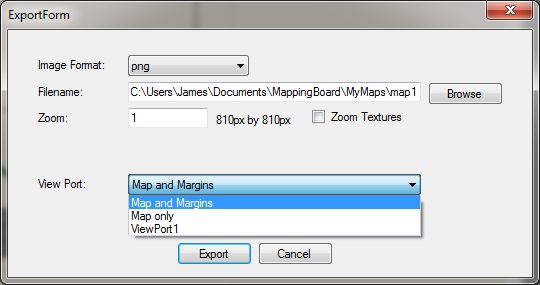
This eventually allows you to create a polygon, which is a figure with multiple points connected by straight lines. However, its use is not intuitive, as initially you will only be able to create a line with two endpoints. You need to add path points (see the section on modifying the shape of the object). The first “small green box” for adding points is found at the lower right of the object. An easier way to produce polygon with straight sides is use the pen below, set the tension to 0 and close the curve. If you just create a line with several points, you can make it a closed curve (connecting the last point to the first) by setting the “closed” property to true.

### Pen

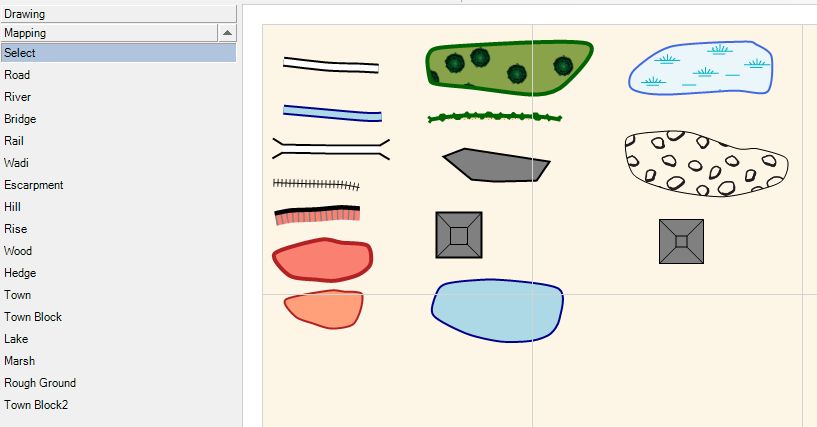
This allows you to draw a freeform line. Setting the “closed” property to true will create an area (however, it is much less useful than some of the area Mapping objects). As you draw, the program will create points along the line through which it draws. Once you create a line, use you can manipulate the shape of the object as described earlier.

### Viewport

This is a new feature in 9.2 that allows you to produce only part of a map when you export it to an image. On the map itself it doesn’t seem to do anything, and just creates a rectangle, but when you produce an image from a map with a viewport defined (File->Export to Bitmap or the “Export to Image” toolbar icon), it will bring up the following screen, allowing you to either print the entire map or just what is in a single viewport. Note that if you export a map with a smaller viewport in it, the viewport rectangle outline will not appear in the exported image.



## Mapping Template Menu



The mapping objects are predefined patterns that you can use to build your map.

### Basic types of Mapping objects (also applies to Textured Mapping)

#### Linear

At their highest level, the behavior of Linear mapping objects is similar to the PEN drawing object, which is in turn just an unclosed polygon. You draw the path of the object on the map, and it will create multiple path points connected by a curve, which are controled by the techniques outlined in “manipulating the shape of objects”. The interesting thing about linear objects is that they actually consist of a “collection” several “pens” that follow the line you draw to give the object its unique pattern. If you want to change the appearance of a linear object, you will need to work with the pens rather than the object properties that appear in the property window, which control the path of the object and the overall “polygon” that is used to define the line. For example, if you want to apply a textured image to a linear object, you need to find the correct pen and assign it an image. In the discussion of the linear objects below, we will concentrate on the pens.

#### Area

Most of these are just simple curved closed polygons with predefined fill colors or patterns. You can modify these properties in the property window, especially the fill and border properties. The shape of these areas can be modified controled by the techniques outlined in “manipulating the shape of objects

#### Symbols

These are fixed symbols, with the unchangeable color, size, and pattern. You can manipulate position, rotation, and (sometimes) repeats.

### Linear Mapping Objects

#### Road

The road consists of two pens. Pen[0] is the border of the road, and Pen[1] is the interior of the road, which is drawn on top of Pen[1]. Note that Pen[0] has a default width of 10, while Pen[1] has a default width of 6. These differing widths mean that the borders of the road can be seen with 2 pixels on each side of the interior. As long as the width of Pen[1] is at less than Pen[0], you will see the border on each side of the road (although you probably want to make it at least two less. To change the color of the borders of the road, modify the PenColor of Pen[0]. To change the color of the road itself, modify the PenColor of Pen[1]. You could also apply a texture to the road by modifying the PenImage property of Pen[1]. Note that images in the pen seem to be “tiled”, so that if they are smaller than the pen, they are repeated. This makes smooth textures (such as some of the Vinytri mud textures) quite attractive.

Roads have an interesting feature during the printing process. Crossroads of roads with identical pens are smoothed over so you don’t see lines through the intersection.

#### River

Rivers are structured almost identically to roads, with two pens. Pen[0] is the border and Pen[1] is the interior. Just as with roads, you can change the colors and apply image textures to the pen. Also, river junctions are smoothed just like roads.

#### Bridge

The bridge object is just a road with a different StartCap and EndCap on the border pen (Pen[0]). You can apply textures to the bridge interio (Pen[1]), and the bridge symbol endcaps will extend beyond the colored area. Note that Bridges are normally created with a higher DrawOrder (3) than roads (1), so that they will print on top of the road. However, if you change their draworder of the bridge to 1, it will merge with the road.

#### Rail

Rail lines have two pens. Pen[0] is the center line of the rail and Pen[1] provides the cross hatching, by having a custom dash pattern that gives the cross hatching.

#### Wadi

The wadi is another two pen object. The center of the wadi is Pen[1], while the grey hatching is Pen[0] with a custom dash style.

#### Escarpment

Escarpments are a 3 pen object. Pen[0] is the salmon color, Pen[1] the grey cross hatching, and Pen[2] the black border. The “directional” border is obtained using the CompoundPen properties of Pen[2], so that only one stripe of it is visible (see the Pen Properties section below). The effect of this is that the border side will be determined by how you draw it. You could make a small valley by drawing one escarpment left->right and another right->left.

#### Hedge

This is a complicated object with multiple pens, all of which combine to give you an uneven green line.

### Area Mapping Objects

#### Hill

A simple closed curved area. You can modify the interior and border.

#### Rise

Rise is another simple closed curved area with a different default interior color from Hill.

#### Wood

While the basic shape of this is another simple closed curved area, the interior shows how the FillMode of “ImageAndColorFill” is used. The basic image is transparent with Tree shapes, and the default fill color is a light green. The result is tree shapes on a light green background.

#### Town

A simple curved area with a grey background

#### Lake

A simple curved area with a light blue background.

#### Marsh

This simple curved area shows how ImageFill is used. The default Marsh image has a non-transparent background color, so setting this to ImageAndColorFill will have no effect.

#### Rough Ground

By Default, this simple curved area as a transparent image of rocks with no background color. If you change this to ImageAndColorFill and specify a background color, you will see the rocks within a colored background.

### Symbols

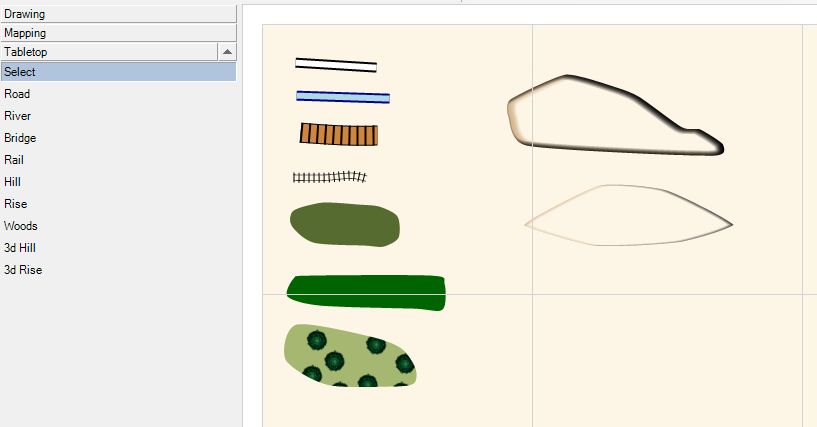
#### Town Block

This is an image with a fixed size and shape. You can specify repeats to have multiple copies of the image side by side. Rotation can be used to align the block along roads and terrain features.

#### Town Block 2

Similar to Town Block but with a slightly different image.

## Tabletop Template Menu



This menu gives you more map objects. Most of them are similar to those on the Mapping menu except that their borders or patterns are slightly different.

### Linear

#### Road

This looks the same as the Mapping Road.

#### River

This looks the same as the Mapping River.

#### Bridge

Rather than the standard bridge symbol, this bridge looks like wooden planks side-by-side

#### Rail

A double tracked railroad instead of the single crosshatched line that is found on the Mapping menu.

### Area

#### Hill

An area with a green background and no border.

#### Rise

An area with a dark green background and no border.

#### Woods

This looks identical to the woods object from the Mapping menu, except it does not have a border.

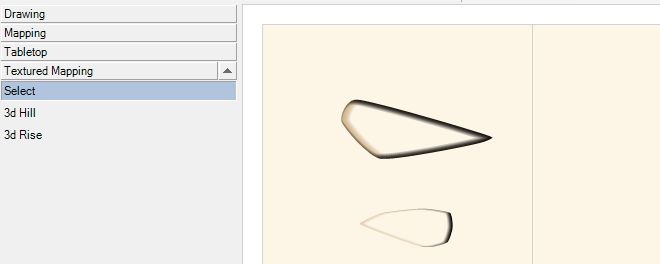
#### 3-d Hill

This is an interesting object because the border is a pen collection, designed to give a 3-d effect.

#### 3-d Rise

Similar to 3-d Hill, this object gives a 3-d effect.

## Textured Mapping Template Menu



This menu contains two items which look the same as their counterparts in the Tabletop menu.

### Area

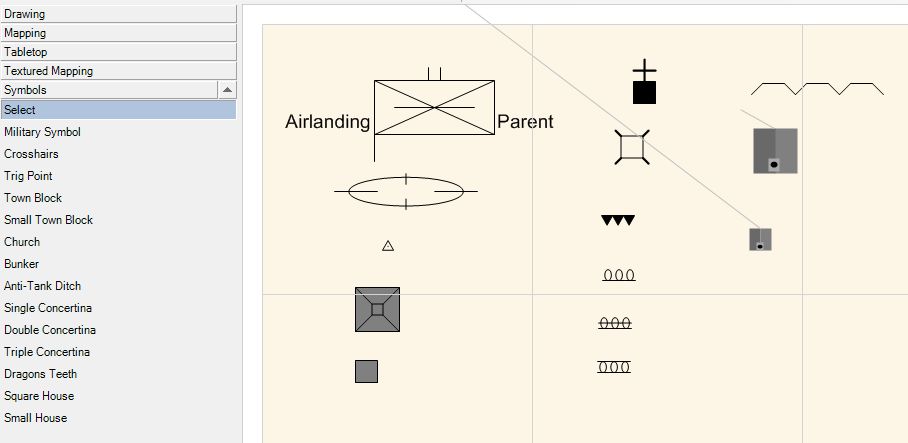
#### 3-d Hill

This is an interesting object because the border is a pen collection, designed to give a 3-d effect.

#### 3-d Rise

Similar to 3-d Hill, this object gives a 3-d effect.

## Symbols Template Menu



The objects in this template generally create an object with a fixed look. For many of these, you cannot change the basic size and shape of the symbol, but have flexibility as to its position and repetition.

### Military Symbols

These are one of the nicer features of MB. There are lots of options here that will allow you to show almost any military organization on your map. Each military symbol consists of a symbol box with modifications defined by the Symbol properties. The location of the symbol is measured from the top middle of the symbol box.

#### General Properties

##### Dash Style

Allows you change the pattern on the outside of the symbol. The default of solid should work fine.

##### Fill Color/Filled

These allow you to select the background color of the symbol. The default is no fill. You can use this to specify different nationalities.

##### Pen Color/Pen Width

These allow you to specify the color and weight of the lines in the symbol and the font color of any accompanying text.

##### Rotation

Allows you to align the symbol and accompanying attributes in different directions, allowing you to specify front lines - For most of these this works fine, but the mortar symbol offsets the circle at the base of the mortar (fixed in 9.2).

##### Transparency

If you want the background to show through the image, choose a transparency >0.

#### Symbol

The symbol properties allow you control over the symbol appearance. The best way to learn the different options is to play with them.

##### Type

The basic pattern of the symbol box - infantry is a cross in a box, armor an oval, etc. The example above was infantry.

##### Subtype

These modify the basic symbol. For example, specifying a type of infantry as motorized will put two wheel symbols underneath the infantry symbol. The example above used a subtype of “airlanding”.

##### Unit Size

This puts a standard size symbol (from squad to army group) on top of the symbol box. The example above was a battalion.

##### Font Size

Determines the size of the text used in the unit and parent name.

##### Unit Name

This text is put to the lower left of the symbol.

##### Parent Name

This text appears to the lower right of the symbol box.

##### Strength

Puts a (+), (-) at the top right of the symbol box, or a battlegroup symbol over the unit strength.

##### Unit Count

Allows you to put multiple boxes behind the symbol in a tiled stack (try it to see what it looks like). Only the top one is visible.

##### IsHeadQuarters

If true, a vertical line is put to the bottom left of the symbol box.

##### Fill Level/SecondFillColor

These allow you to fill up the symbol with a second color starting from the bottom

### Crosshairs

This symbol is an oval with lines around the outside specifying a specific location. It has standard properties. Note that the Repeats attribute doesn’t seem to work properly, but would be of limited utility anyway.

### Trig Point

This symbol creates a small triangle with a dot in the middle of it.

### TownBlock

This symbol creates a box with a “raised roof” symbol (try it to see what this looks like). The size and shape of this symbol seems to be fixed. I played with this and found that the only two properties that seemed to change anything were Repeats and the Rotation. Even colors were fixed. Fortunately Repeats and Rotation are the most useful, allowing you to create a block of several house symbols and align them along a road.

### Small Town Block

This is similar to town block, but it just creates a small grey box. As with TownBlock, the size and color seem fixed, but you can create blocks and align them using Repeats and Rotation.

### Church

Another symbol with a fixed shape and size, this creates a black box with a cross on top.

### Square House

This creates a single grey house symbol with a shaded roof. Unlike the town block symbols, you can adjust the size and color of this symbol, but the results are not as attractive as the default.

### Small House

This is similar to Square house, but the symbol is smaller.

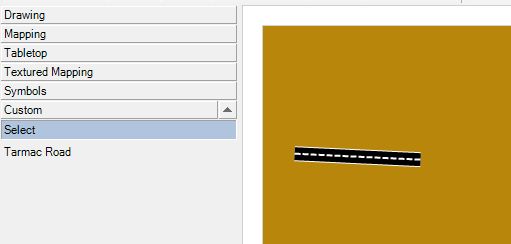
### Bunker

This creates a fixed size symbol similar to the town block, but with a white background and open sides.

### Anti-Tank Ditch/Single Concertina/Double Concertina/Triple Concertina/Dragons Teeth

All of these function similarly, with the basic symbol being 3 repetitions of a fortification symbol. You can specify repeats and rotation to create a longer line of the fortification.

## Custom Template Menu



This is where you store objects that you want to add to your templates permanently. Note that you can also store objects as “features” that will make them just as reusable and make them easier to delete than custom objects. Features give you the exact object that you saved, while adding custom object templates allows them to be used as the other templates (in fact they are incorporated into your MB application). The basic implementation of MB comes with one Custom feature, a variation on a road object.

### Tarmac Road

This is a multipen linear object that gives the effect of a black road with a white dashed line running down the middle of it. It also has a white border, but this is often hard to see unless you have changed the background color of your canvas (as above). Pen[0] is the white border. Pen[1] is the black center. Pen[2] is the white dashed line down the middle.

## Feature Template Menu

Initially empty, in MB 9.2, this will be populated by named features that you create.

# Custom Objects and Features

This section will show how to create reusable objects and save them as both features and custom template objects. It will also highlight the differences between these two types of reusable objects. As a sometime American Civil War gamer, two terrain features that constantly show up on my battlefields are wooden fences and stone walls. None of the predefined objects look right, so I decided to try to build these. Both are linear objects, so it would seem that the best way to do this would be by modifying existing objects and modify their pens.

## Modifying existing objects.

First up is the stone wall. I decided to make this a simple linear object with a textured center, using one of the Vintyri textures, which I had downloaded from the website referred to above. As a road is a fairly simple object, I decided to use it for a base, so I created a road object onto my map. Roads consist of two pens, the pen[0] is the border, the pen[1] is the interior. I brought up the pen collection and changed the PenImage property of pen[1] to one of the textures. The one I liked best was actually called “River Bed 2”, but there were several others that would work as well. Also, I decided to thin down the border, so I increased the width of the interior from 6 to 8. As the border pen is 10 wide, this left a one pixel overlap on each side of the interior. The result was changing the basic road:



to this:

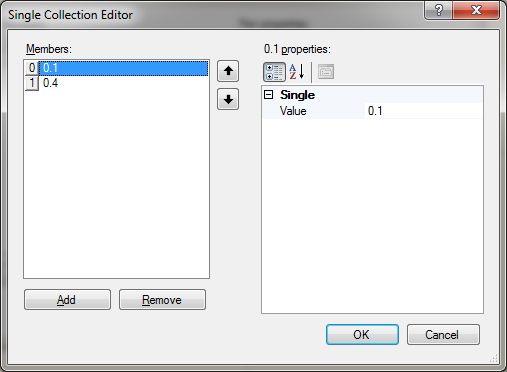


Now I am going to work on a fence. I will show how to create a prototype worm rail fence later, but for this section, I will use a simpler pattern. I am going to modify the basic rail line:



changing the color and increasing the distance between the cross hatch lines. There are two pens in the rail line. Pen[0] is the center line and Pen[1] is the cross hatches. I chose “sienna” for the color of both. I also increased the width of Pen[0] to 2 because the center line was a bit thin.

Now I looked at pen[1] and found that it had a DashPattern array which looked like this:



As I explain below in “Pen Properties”, the [0] entry of this array controls the width of the cross hatch, while the [1] controls the distance between them. I changed these to .3 and 2. The resulting figure looked like:



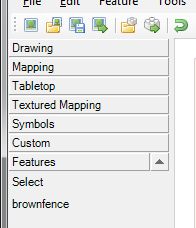
While not exactly what I want, it is close enough for wargame maps.

## Saving a “feature”

I decided that I would save the fence object as a feature. I selected it and used the toolbar “Save object” button to save it and give it a name of “brownfence”. Two things happened:

1) a file called brownfence.feat was created in the MyFeatures directory.

2) an entry appeared at the bottom of the template area called “brownfence”:



Now I can select this object and click on the map to a brownfence. This will create an object exactly the same size and shape as the fence object I saved. To change its size and length, I need to use the techniques outlined in “manipulating the shape of objects”, changing the path points and the enclosing box.

To delete the brownfence, I need to delete the corresponding .feat file and then restart MB. It will disappear from the select list.

## Saving a Template object

I am going to save the stone wall as a template object. First I select it and then right click it. One of the options is “Add to Template”, which allows you to choose the name and id of the object, and the selection menu where you want your new object to reside. I am going to put it on the Custom Menu as StoneWall as both its name and ID. It is now the second object on the custom menu.

The difference between a template and feature can be seen when I use this object. Unlike features, where you get exactly what you stored and need to manipulate it after you create it, the new Stonewall object acts like other linear objects, where you can draw the desired path on the screen. Also, unlike features, where you can delete the corresponding .feat file, template objects can now only be deleted by updating the .xml file where object definitions are stored, which may be daunting if you are not familiar with XML. The relevant file is defaultterrain.xml found in the "\Local Settings\Application Data\Tylermade Software\MappingBoard\" folder. In XP that's in the "Documents and Settings" folder in Windows 7 "Users" (note that this location is used for 9.2. in 9.1.1, this .xml file is found in the 0.9.1.1 subdirectory at this location).

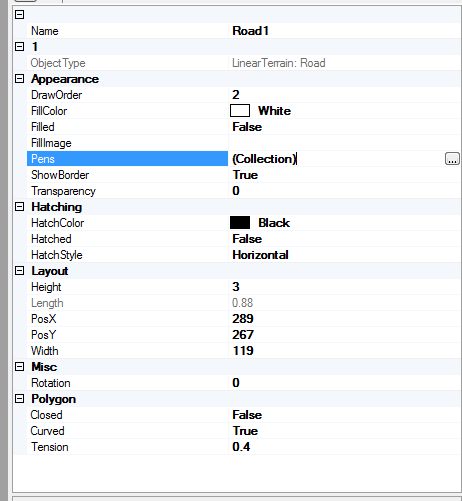
# Pen Properties

Linear objects and compound borders consist of a collection of pens that follow the center line. The properties of all the pens combine to give you the final appearance of the object. Building complex linear objects is often a process of trial and error, but the results are worth the work. Earlier examples have shown the use of simple pen properties to build custom objects.

The best way to learn properties is to play with them. We are going to start by using a short section of the Mapping Road object, which is one of the simplest of the linear objects.

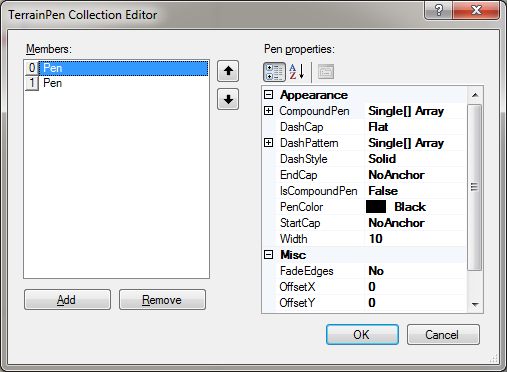
 when selected 

The property area for this line looks like:

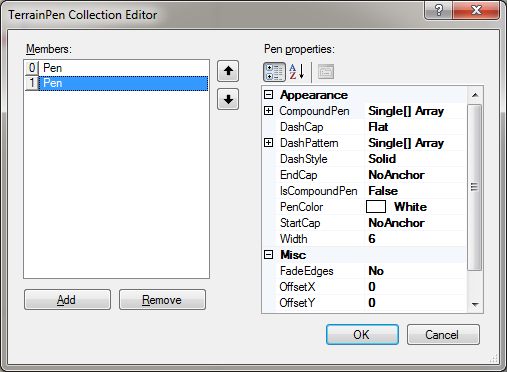


An important thing to realize about this object is that almost none of the properties in the basic property area will affect the appearance of the road unless you close the polygon. Almost all of the appearance of the road other than position and size is determined by the pens. To access the pens, select the pens property and click on the box with 3 dots at the right of the Pens line

This will bring up the TerrainPen Collection Editor, which defaults to the first pen (by default, pens are numbered starting at 0, so the first pen is actually Pen[0)]:



Roads consist of two pens, the second pen (Pen[1]) has these properties:



As we stated earlier, the road gets its effect by having the border (Pen[0]) slightly wider than the interior (Pen[1]). As we are going to move these around, and want to see them on the map, set the color of the interior (Pen[1]) to some color other than white. I chose “Firebrick” (note that the thin white line that you see in some of the images below is a map grid line, rather than part of the object itself):



## The Center Line

Pens are drawn along the center of the line you specify when you draw the object. By default, pens are centered on this line, and start where the line starts. In the discussion below, we are assuming that the line is being drawn from left to right, more or less horizontally. If you draw an object right to left, the effects may be reversed.

## Pen Drawing Order

Pen[0] is drawn first, then Pen[1] on top of Pen[0], etc. In the case of the road, the interior is drawn on top of the border.

## Pen Properties

### Width

This is the width of the pen, in pixels.

### PenColor

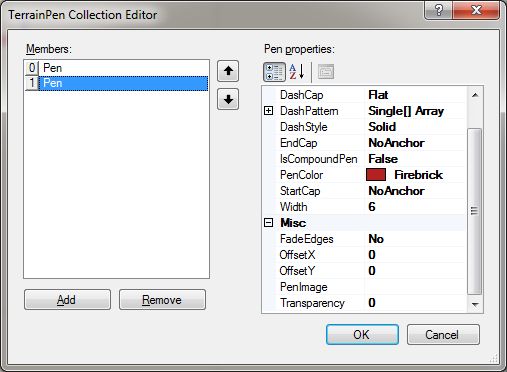
The color of the pen, selected normally.

### PenImage

The image file used to fill the pen (overrides PenColor, and is used in “Tile” mode). See an example of using this in the section above where I created a stone wall.

### OffsetX

By default, pens start at the beginning of the center line. However, you can start it to the left and right of the starting line. A + number in this field will start the pen that number of pixels to the right of the beginning of the start line, a - number will start it to the left of the beginning of the start line (assuming the pen is being drawn left to right). Later we will see how to use this to build a complex object. The offsets are not visible on the basic pen property screen and you need to scroll down to see them:



Here I set the OffsetX of my red center to 10. Look at what happens:



The red center starts after the beginning of the black border .

### OffsetY

By default, pens are centered on the center line. A + number in this field will offset the pen below the center line by the specified number of pixels, a - number will offset the pen above the center line (again assuming the pen is being drawn left to right). Here I set OffsetY to 20.



### StartCap/EndCap

By default, the start and end of the pen are flat (no anchor). However, you can choose from a variety of predefined patterns to change the start and end of the line. Play with these to see the options to see what is available. Here is a triangle startcap and a round endcap on the red center:



### Fade Edges

This shades the edges of the line. In this example, I chose a Wide fade for the border (pen[0]):



Note that it actually made the road wider than the basic width, and the black faded to grey.

You could use this to get some interesting effects. Here I added a new background pen (pen[0]) with a wide faded edge and a brown color to the Tabletop Rail object to produce a railroad on a raised embankment:



### Dash

The simplest objects have their dash pattern set to solid, which is just a solid straight line. There are several other predefined patterns. Going back to our basic road with a red center, we have:

#### Dash



#### Dot



#### DashDot

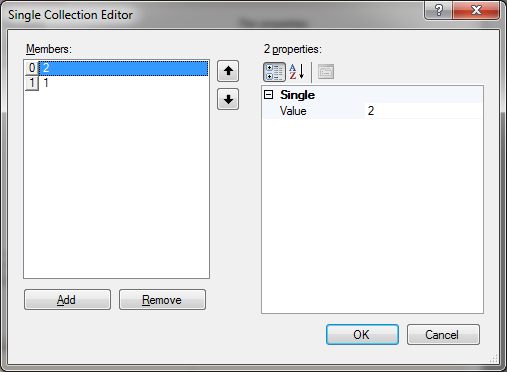


#### DashDotDot



#### Custom

Up to this point, the size and pattern of the red areas are predefined, but the Custom tells the software that you will define the size and pattern of the line in the “DashPattern” array, which you can access by selecting “DashPattern” and clicking on the box with 3 dots at the right. This will bring up the “Single Collection Editor” screen, that will allow you to modify add and remove array members:



This array consists of a series of members, each of which has a numeric value. The first (array member 0 - remember that these arrays start with the “0th” entry) will define the length of the first area where the line will appear (solid). The second entry (array member 1) will define the length of the first area where the line will not appear (void). A value of 1 is the width of the pen in pixels. You can continue this series as long as you want. When the software gets to the end of the series, it will restart at the beginning and repeat until it reaches the end of the line. Note that if you specify your dash pattern as custom, you MUST set up one of these arrays or the software will hang (you can get around the hang by pressing continue, but if you don’t do it right, you will lose your work). Also, it will hang if it encounters 0 or negative members, although positive fractions are ok (see example 6 below).

Example 1

DashPattern[0]=1



We told it to have a solid area equal to the width of the pen, but then did not define a void length - so it repeated the solid, resulting in the basic line.

Example 2

DashPattern[0]=1, DashPattern[1]=1



Here we alternate solid and voids of length 1 - note that this looks the same as the predefined “dot” pattern.

Example 3

DashPattern[0]=2, DashPattern[1]=1



This one has double length solids followed by single length voids.

Example 4

DashPattern[0]=1, DashPattern[1]=2



Here the voids are longer than the solids.

Example 5.

You can specify more than 2 entries, the pattern you specify will repeat.

DashPattern[0]=1, DashPattern[1]=2,DashPattern[2]=2,DashPattern[3]=1



Example 6

You can also specify fractional numbers, but they must be greater than 0 or it will hang.

DashPattern[0]=.4, DashPattern[1]=2



The railroad patterns of the Mapping templates use this to get the effect of the thin railroad ties.

In summary, be aware of the solid-void-solid-void pattern and you can figure out how to set up custom dash patterns.

### DashCap

The final dash property is DashCap, which is similar to the Start/End Cap properties, although there are only two options - triangle and round. These are put on each end of the solid areas, either predefined or custom.

Here is the custom pattern in example 6 above with triangle dash caps:



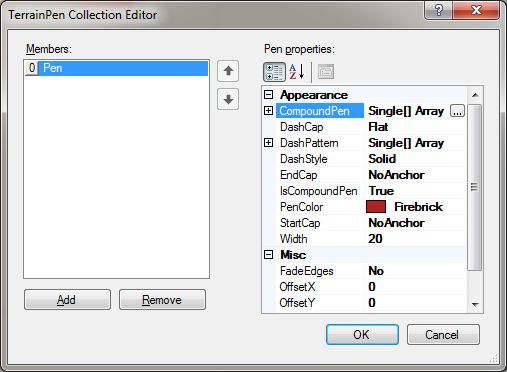
### A combination example, showing offsets and dashes:

I have two pens here, both have a dash pattern of “dot” and a width of 10, but the second pen has both an OffsetY and OffsetX of 10 as well - the result is a checkerboard pattern.

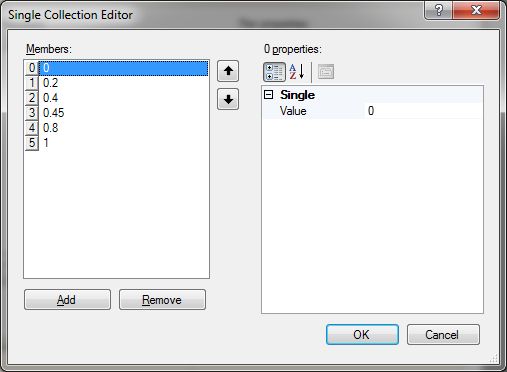


### Compound Pen

The last property of the pen is CompoundPen. The Dash properties determined solid-void patterns down the length of the pen. CompoundPen defines solid-void patterns across the width of the pen, creating parallel stripes. However, the meanings of the Compound Pen array are different and a bit trickier to use than Dash patterns. Compound Pens are only used if the IsCompoundPen property is set to true. If this is the case, you MUST specify a compound Pen array just as was required for when you set the Dash to Custom. You invoke this screen by clicking on the box with 3 dots that appears when you select the CompoundPen:



This brings up the single collection editor that should be familiar from dash pattern entries:



CompoundPen arrays are specified as a series of (start, end) pairs. We will notate this in the discussion below as start->end. The first number is the fraction from the top of the pen (measured in terms of the pen width) that the stripe starts, the second is the fraction of the pen width that the stripe ends (remember that we are drawing our examples from left to right - if we draw from right to left, the distances are measured from the bottom). Members 0 and 1 of the array constitute the first pair, 2 and 3 the second etc. The program will get upset at you (hang - be careful and press continue) if:

1) You have any number greater than 1 or less than 0

2) The start number of any pair is greater than the end number.

3) stripes overlap-for example .3->.6 overlaps .4->.7.

4) You have a stripe that starts below another-for example .4->.7 starts below .2->.3

5) You specify an odd number of members, so that one of the start/end pairs is incomplete.

If you avoid the above pitfalls, you will define one or more stripes that go down the length of the pen.

Example 1: Based on our road with a black edge and firebrick center. Both have a solid dash pattern. However, I increased the width of these so that the inner line is 30 pixels wide so we can see the result. Our starting point is:



Now we set the IsCompoundPen property of the interior pen[1] to true and set up our first 2 entries as .3,.5 yielding:



Firebrick stripe starts .3 from the top of the red pen area and continues to .5 from the top.

Example 2:

You can have more than one stripe. Here the first line is .3->.5 and the second .7>-.9



That’s all there is to it.

### Combining Pen Effects

Here is an example where we have a compound pen combined with a dash pattern:



## Building a Basic Worm Fence

A worm fence is a zigzag pattern that is not replicated by any of the custom objects. As an exercise I decided to try to build this using pen features. First, I looked at the pattern I wanted to build. I built a table to show the pattern. x’s are solid, empty cells are void. The (x ) are actually in the second repetition of the triangle pattern:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | X |  |  |  |  |  |  |  | (x) |
|  |  |  | x |  | x |  |  |  |  |  | (x) |  |
|  |  | x |  |  |  | x |  |  |  | (x) |  |  |
|  | x |  |  |  |  |  | x |  | (x) |  |  |  |
| x |  |  |  |  |  |  |  | (x) |  |  |  |  |

This has 5 rows, so I need 5 pens. To make calculations easy, I will set the width of each pen to 10, with the basic width unit being 2 pixels (.2 of the pen width). I define each pen as a compound pen with a stripe that will handle one row. Each has a width of .2, so the first pen goes from 0->.2, the second from .2->.4, the third from .4->.6, the 4th from .6->.8 and the last from .8->1.0. The 5 stripes will thus cover the entire height of the 10 pixel object.

I then look at the dash pattern. I will define them all as custom dash patterns. As the first entry in a dash pattern is solid, I move all the patterns to the left, so the first solids line up in my table. Each box in the pattern has a width of .2. Note that they all line up and have a width of 8 boxes (1.6):

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x |  |  |  |  |  |  |  | (x) |  |  |  |  |
| x |  | x |  |  |  |  |  | (x) |  |  |  |  |
| x |  |  |  | x |  |  |  | (x) |  |  |  |  |
| x |  |  |  |  |  | x |  | (x) |  |  |  |  |
| x |  |  |  |  |  |  |  | (x) |  |  |  |  |

Dash Patterns - each solid is .2 width (2 pixels)

Pen 0 =solid .2, void 1.4

Pen 1 =solid .2, void .2, solid .2, void 1.0

Pen 2 = solid .2,void .6 ( this actually repeats .2,.6,.2,.6, but since the second repetition is the same, we only need specify it once)

Pen 3 = solid .2, void 1.0, solid .2, void .2

Pen 4. = solid .2, void 1.4

Finally I set the offsetX for each pen. The offsets will make the dash pattern start in different places to give me my repreating triangles.

Pen 0 - 8 pixels (4 x .2 x width(10))

Pen 1 - 6 pixels

Pen 2 - 4 pixels

Pen 3 - 2 pixels

Pen 4 - 0

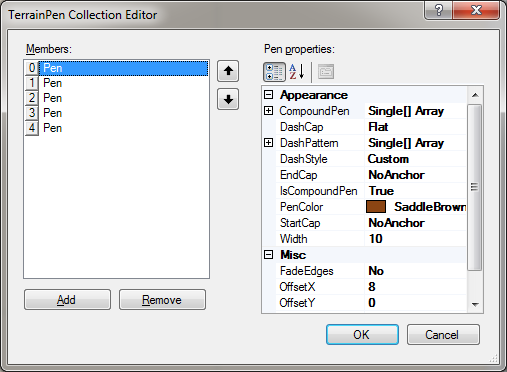
The result is:



which is close to what I want. Note that after I created this, I saved it as a custom template. I later completely replaced it (see below), but this basic zig-zag is a good demo for this documentation.

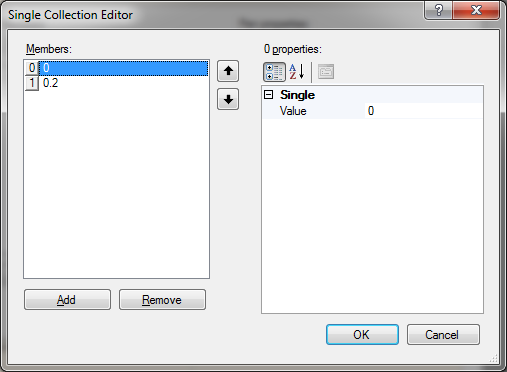
Here are the properties, CompoundPen array, and DashPattern Array of pen[0] so you can see how the above discussion is implemented:

Properties



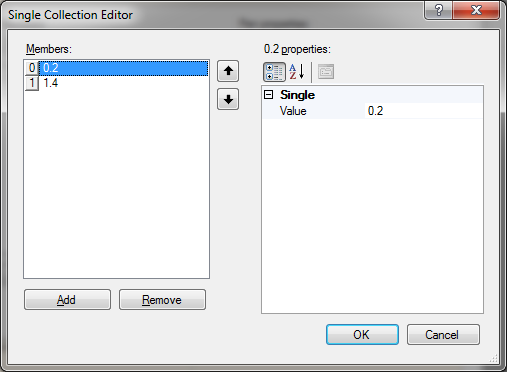
Note the PenColor and OffsetX . Also, the DashStyle=Custom and IsCompoundPen= True tell the software to use the arrays.

CompoundPen Array



The topmost stripe starts at the top (0) and ends at .2 from the top.

Dash Array



As I figured in my plan, the top most dash pattern has a solid of .2 followed by a void of 1.4 (measured in terms of the pen width of 10). When combined with the OffsetX of 8, this will give me the top of my triangle.

The other 4 pens are defined similarly, and they combine to give the repeating zig-zag.

Here are some thoughts on building linear Pen objects:

Because you want the pattern to repeat, the total width of each dash pattern (add up the solid and void widths) must be the same or a multiple of some basic width. In the worm fence example, the dash pattern widths are either 1.6 or 0.8, so this is satisfied, as all are multiples of 0.8. Also, because dash patterns are based on the pen width, all the pens in such a complex pattern should have the same width, or the dash patterns will get “out of sync” because of the different lengths.

While CompoundPen specifications cannot overlap within the same pen, there is nothing to prevent patterns in different pens from overlapping. You can use this to smooth out some rough edges.

Remember that you have other effects you can use, such as Fading and End Caps, that just might give you the effect you need. If your pattern doesn’t look quite right, try playing with these to see if you can get the effects you need.

## Faking an offset - a dash pattern trick

Building the worm fence using the technique above requires that you use offsets to make the zigzags occur in the proper place. This is because dash patterns always start with a solid. Patterns using offsets occasionally get distorted as you go to the vertical, and also the process of moving all the patterns to the left is a bit mind-bending. However, you can “fake” an offset in the dash pattern by starting the dash pattern with a solid that is legal (positive number) but too thin for the graphics to render. For example, (with a pen width of 10 to make the math easy), let’s say we want an offset of 8, which corresponds .8 of the pen width. Instead of using OffsetX, we could start our dash pattern with the sequence (.01, .79). The solid of .01 is legal, but too thin to show up. Adding the .01 to the void of .79 gives us a first dash sequence .8 in width that looks like a void and is equivalent to an offset of 8.

## The Final Worm Fence

I uploaded a more sophisticated version of the worm fence to the MappingBoard Yahoo Group : <http://games.groups.yahoo.com/group/MappingBoard/>

The version in the Yahoo group uses 10 pens (each does a stripe covering 1/10 of the width of the object) with the fake offset DashPattern technique to build the following:



If you have stayed with me to this point, you should be able to import this feature file and examine the properties to see how I created this pattern from the DashPattern and compound Pens to create this effect.

# Sharing the Wealth

To make the MB software more useful, everyone who creates a really good looking complex object is encouraged to save it as a .feat file and upload it to the Yahoo group for use by others. Also, if you find mistakes in this document or have better insights into how a particular feature can be used, please let me know (my email is in the Yahoo group), and I will be happy to modify it.