FME Desktop®

FME and KML
Pathway Training

FME 2013-SP3 Edition

SAFE SOFTWARE

Revisions

Every effort has been made to ensure the accuracy of this document. Safe Software Inc. regrets any errors and omissions that may occur and would appreciate being informed of any errors found. Safe Software Inc. will correct any such errors and omissions in a subsequent version, as feasible. Please contact us at:

Safe Software Inc.
Suite 2017, 7445 – 132nd Street
Surrey, BC
Canada
V3W1J8

www.safe.com

Safe Software Inc. assumes no responsibility for any errors in this document or their consequences, and reserves the right to make improvements and changes to this document without notice.

Trademarks

FME is a registered trademark of Safe Software Inc.

All brand or product names mentioned herein may be trademarks or registered trademarks of their respective holders and should be noted as such.

Documentation Information

Document Name: FME Desktop KML Pathway Training Manual
FME Version: FME 2013-SP3 (Build 13528) 32-bit
Operating System: Windows 7 SP-1, 64-bit
Google Earth Version: Version 7.1.1.1888
Updated: September 2013

Advanced Module   **FME and KML**

Basic KML Styling

KML features can be visualized in many ways, and FME has a set of transformers designed to help do so.
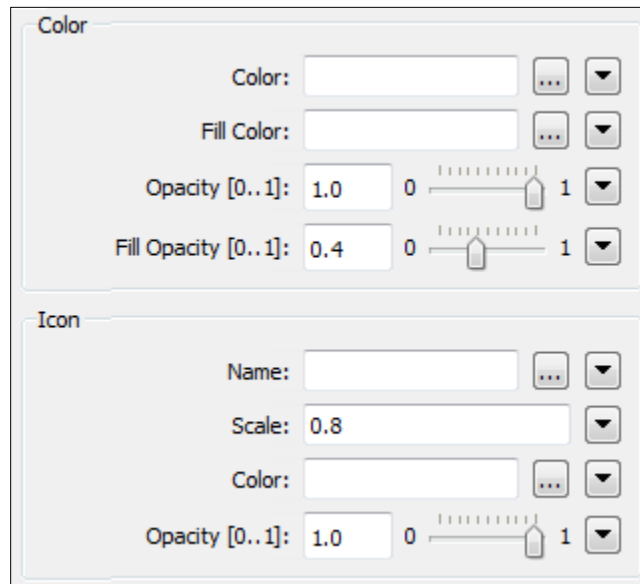
The simplest way to style KML data is with the KMLStyler transformer.

KMLStyler
The main styles that can be set in the KMLStyler are feature color and icon.

Color fields allow you to set the line and fill colors for a feature, and the opacity of that color.

Icon fields allow you to select an icon to represent a point feature, and to set the scale, color, and opacity of that icon.

Example 1: Writing to KML	
Scenario	FME user; City of Interopolis, Planning Department
Data	City Parks, Swimming Pools (Input: MapInfo, Idrisi; Output: KML)
Overall Goal	Write a KML dataset and set feature symbology
Demonstrates	Styling using attributes
Starting Workspace	None
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\KML1-Complete.fmw

The task here is to simply convert a dataset to KML and style the features being written.
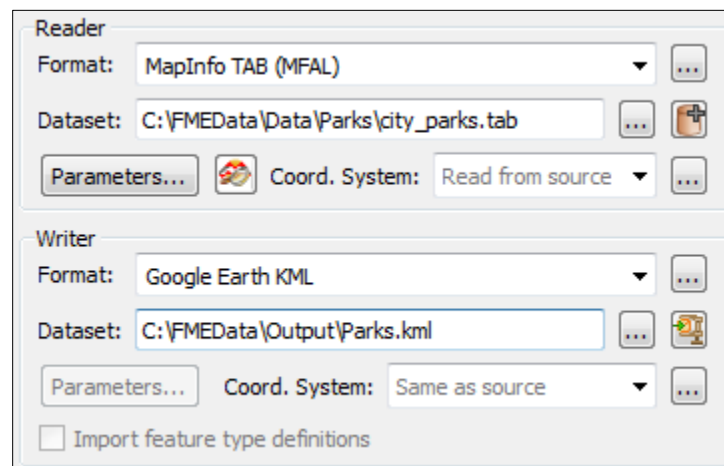
1) Start Workbench
Start FME Workbench.

In the Start Tab select the option to Generate a Workspace. We'll use this functionality to create a translation from MapInfo TAB to KML.
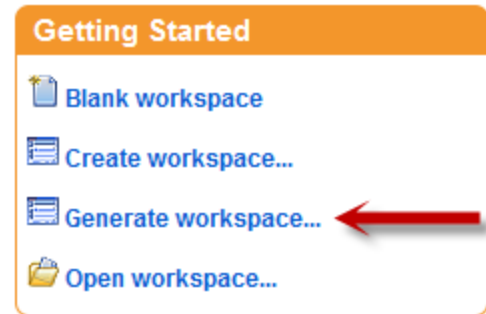
2) Generate Workspace
When prompted, enter values as follows:

Reader Format	MapInfo TAB (MFAL)
Reader Dataset	C:\FMEData\Data\Parks\city_parks.tab

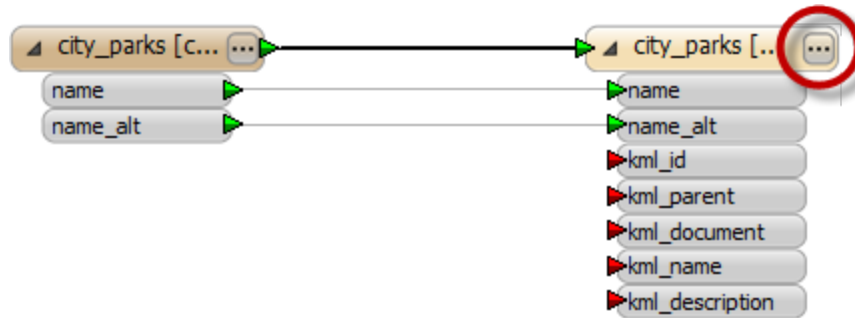|	
Writer Format	Google Earth KML
Writer Dataset	C:\FMEData\Output\Parks.kml

Ensure that the Workflow Option is set to Static Schema.

Click OK to accept this parameters and to generate the workspace.
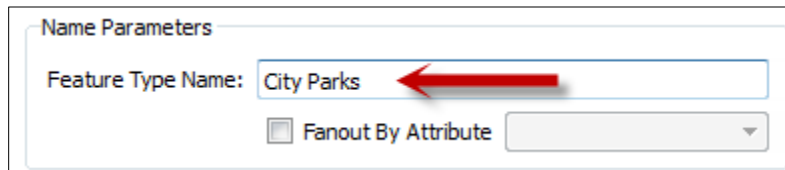
3) Edit Feature Type

With the attributes list expanded, the workspace will look like this:

There is a Feature Type (layer) on the left to represent the source data (Reader), and a Feature Type on the right to represent the output (Writer).

Click on the […] button on the Writer Feature Type to open the properties dialog.

Change the Feature Type name from city_parks to City Parks. This will become the name of the layer when written to KML. Click OK to close the dialog.

4) Add KMLStyler

Now we can style the data with a KMLStyler transformer.

Click on the connection between the Reader and Writer Feature Types. Now type the word "KMLStyler". As you type a list of transformers will appear. Select KMLStyler from the list.
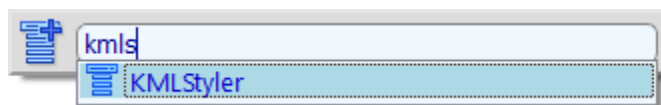
A KMLStyler transformer will be added to the flow of data, like so:

5) Set Parameters
Click the yellow […] button on the KMLStyler. It will open a parameters dialog.

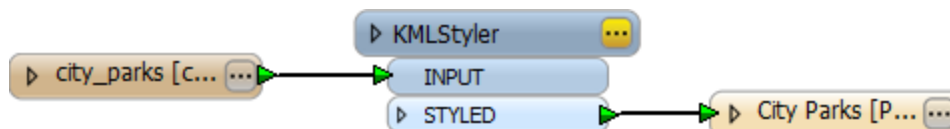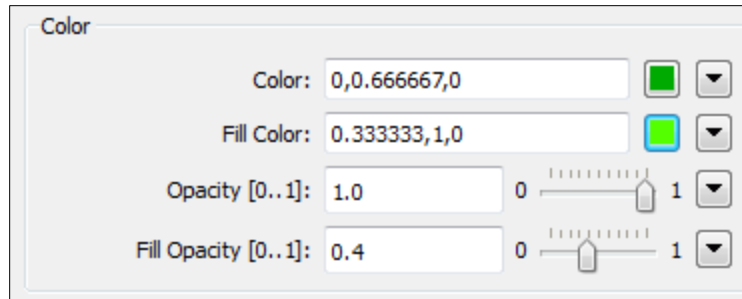Set the outer (line) color to be a dark green. Set the fill color to be a slightly lighter green color.

Leave the line opacity to be 1.0 and the fill opacity to be 0.4. This will produce a solid line but a slightly transparent polygon.

Click OK to close the dialog.

6) Save and Run Translation
Click the save button on the toolbar to save the workspace.
Then click the green "play" button to run the workspace.

The translation will now be carried out and the MapInfo data converted to a styled KML dataset.

```
Translation was SUCCESSFUL with 0 warning(s) (22 feature(s) output)
FME Session Duration: 0.7 seconds. (CPU: 0.4s user, 0.2s system)
```

7) Inspect Data
Right-click on the Writer Feature Type and choose the option to "Open Containing Folder".
This will open an explorer window at the location of the output dataset.

Double-click the dataset to open it in Google Earth, and then examine the translation results.
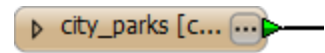
8) Add Second Dataset

Now we'll add some more data to the KML translation.

Back in FME Workbench, select Readers > Add Reader from the menubar. This function will let us add more data. This time we'll add some point features (swimming pool locations).
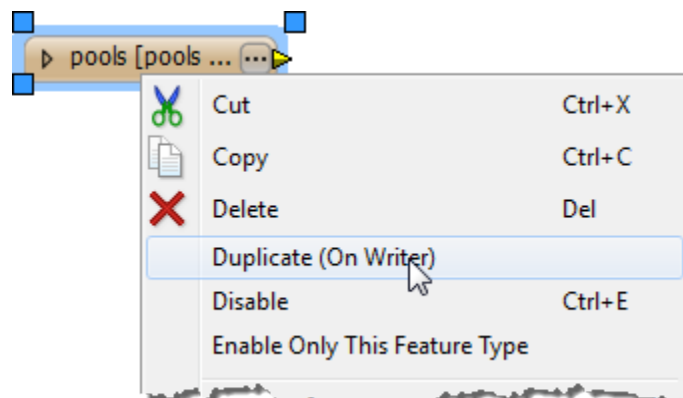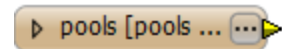
When prompted set the Reader parameters as follows:

Reader Format	IDRISI Vector Format
Reader Dataset	C:\FMEData\Data\Pools\pools.vct

Importantly, set the Coord. System parameter to TX83-CF
This tells FME what the source data coordinate system is.

Click OK to close the dialog and add the new Reader. A new Reader and Feature Type are added to the workspace:

9) Add Output Layer

Now we must add a new layer in the KML to accommodate this data.

Right-click on the Feature Type labelled "pools" and select the option to Duplicate on Writer.

10) Add KMLStyler

As before, select the connection between Reader and Writer Feature Type and type "KMLStyler" to add a KMLStyler transformer.

Open the parameters dialog for this new transformer. Select H2 as the icon to be used.

All other parameters can be left to their default values.
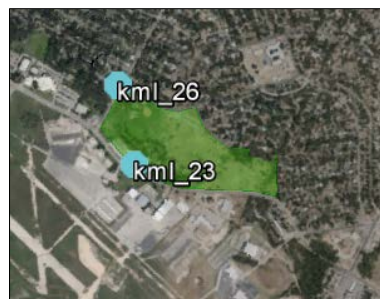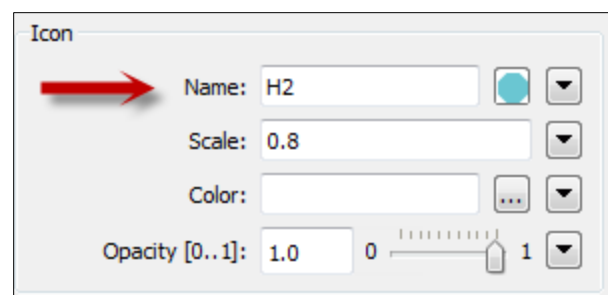
11) Re-Run Translation

Save and re-run the translation. In Google Earth "revert" the dataset. As well as the existing parks, you will now see the swimming pools represented by the selected icon

Advanced KML Styling

Advanced styling techniques add a little more polish to KML datasets than just colors and icons.

Symbology in KML is controlled by a tag called Style. Style is defined in a workspace by the KMLStyler transformer.
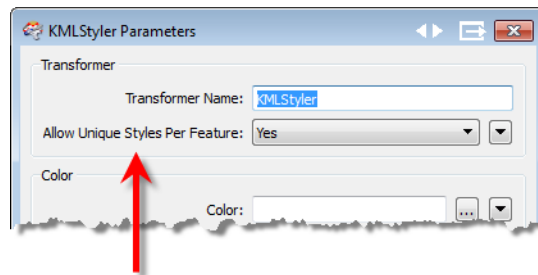
Style Range

In the previous example, style was applied similarly to all features. However, style can also be applied to individual features. The KMLStyler transformer has a parameter to control this. It is called Allow Unique Styles Per Feature.

When all features are to be given the same style, then this parameter should be set to No.

That way, there will be a single Style section in the KML document, referenced by all features.

If the parameter is set to yes, then each feature gets its own style section. This will unnecessarily increase the size of the output file.

When each feature is to get its own style, then set this parameter to yes. The size of the output will be increased, but each feature can have a unique style applied to it. You will need to do this whenever a style element is defined by an attribute.

Display Order

There is no particular parameter or tag in KML by which to order features one above the other.

However, the display order in Google Earth can be controlled by the elevation of the feature. Features with a higher elevation will appear above features with a lower elevation.

Therefore, strict handling of the Z coordinate – plus use of an altitude mode relative to the group – can be used to control the display order of features.

| Example 2: Pollution Monitoring	
Scenario	FME user; City of Interopolis, Planning Department
Data	EPA Facilities, Zipcodes (Input: GML, MIF; Output: KML)
Overall Goal	Show the highest polluting companies in the city
Demonstrates	Styling using attributes
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\KML2-Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\KML2a-Complete.fmw C:\FMEData\Workspaces\PathwayManuals\KML2b-Complete.fmw

The task here is to map the monitored facilities in the city, showing which is creating the most pollution on any particular day. To do so we'll get creative with our KML styling.

1) Start Workbench
Start Workbench and open the beginning workspace.

Notice that it reads from a GML format dataset of EPA (Environment Protection Agency) listed sites, and a MapInfo MIF dataset of zipcode boundaries. The pollution values are set as random numbers. The output is obviously KML.
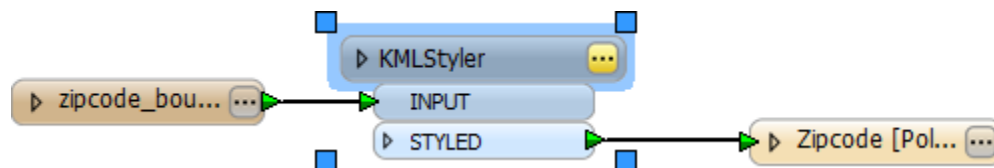
As always, inspect the source data to see what data you are dealing with, and check the *RandomNumberGenerator* parameters.
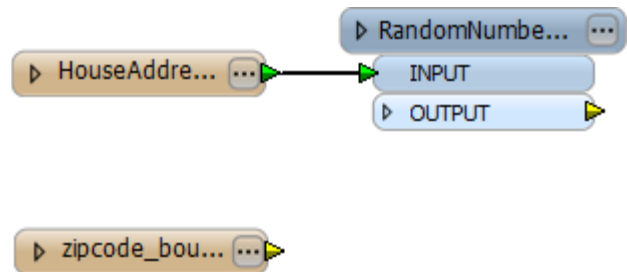
2) Add a KMLStyler
The zipcode data will purely be used as a backdrop to the pollution data. Each zipcode will get the same style applied to it.

Place a *KMLStyler* transformer connected between the zipcode reader and writer feature types.

Open the properties dialog for the *KMLStyler.* Set the "Allow Unique Styles Per Feature" parameter to its most suitable value (considering each feature gets the same style).

Then set a green fill color with a medium opacity value.
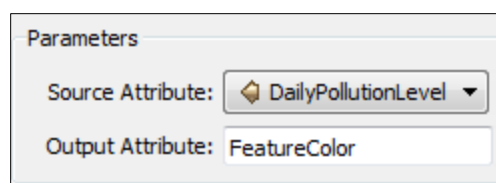
3) Add an AttributeRangeMapper
For the pollution sites, we'll color them according to pollution severity, so each site will need its own style. To start with we'll define color attributes using an *AttributeRangeMapper*

Place an *AttributeRangeMapper* transformer connected to *RandomNumberGenerator:OUTPUT*
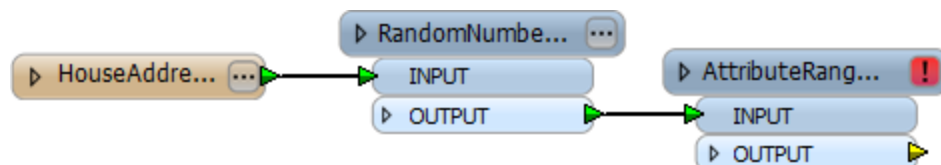
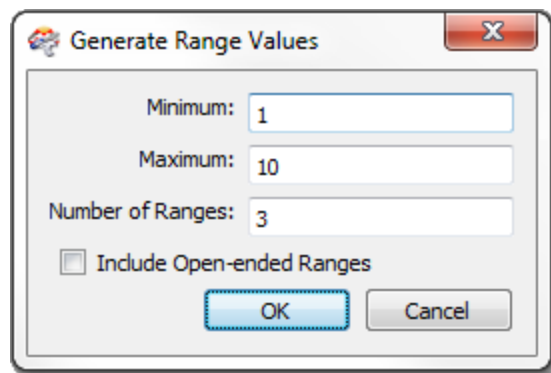Open the parameters dialog by clicking the red [!] button.

Select DailyPollutionLevel as the Source Attribute.
Enter FeatureColor as the Output Attribute name.

Because we know the range of pollution values (they are defined in the RandomNumberGenerator) we can let FME generate ranges for us.

So, click the Generate button.
When prompted enter:

Minimum   1
Maximum   10
Number of Ranges   3

Click OK to return to the main dialog.

In the Output Value field of the main dialog, enter the following:

From 1-4   1,1,0
From 4-7   1,0.5,0
From 7-10   1,0,0

These are RGB values (using the same structure as the KMLStyler) that will result in yellow, orange, and red, for the different levels of pollution.

4) Add a Bufferer
Now add a *Bufferer* transformer. This will turn the point features into ones that possess an area, and can therefore be colored. Obviously, if we started with polygon features we would not need to do this.

Connect it to *AttributeRangeMapper:OUTPUT*

Open the parameters dialog. Set a Buffer Amount of 500.
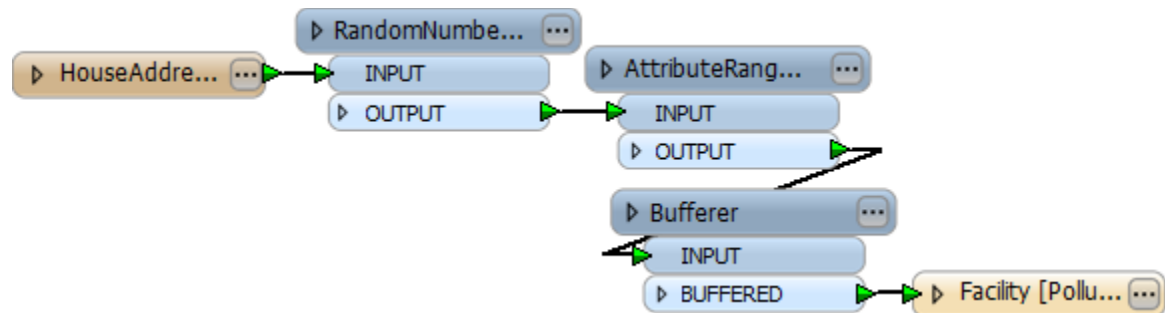
5) Add a KMLStyler
Now add a second *KMLStyler*, this one connected to *Bufferer:BUFFERED*
Open the parameters dialog.

Set the "Allow Unique Styles Per Feature" parameter to Yes. This way each feature can get a different style applied (a different color according to the pollution level).

For Fill Color click the drop-down arrow (to the right) and select Set to Attribute Value > FeatureColor
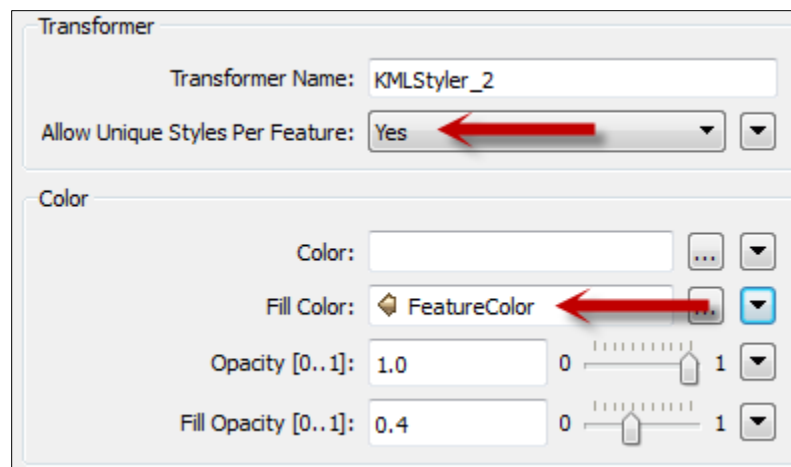
6) Save and Run Workspace
Ensure the two KMLStylers are connected to their respective Writer Feature Types.
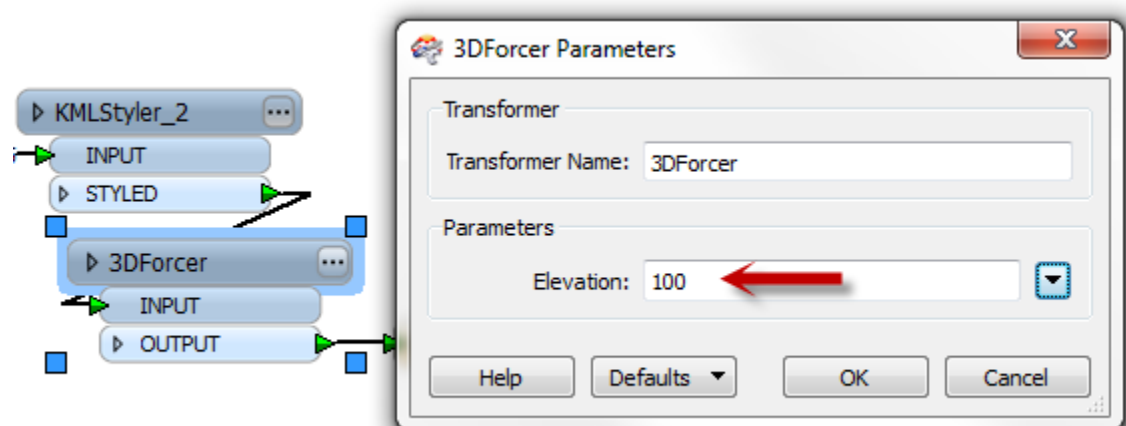Save the workspace and then run the translation.

The output will now look like this. Pollution features are appearing underneath the zipcode areas.

7) Add a 3DForcer
The only way to have the pollution items appear above the zipcode is to give them a higher elevation. To do this, add a 3DForcer transformer after the pollution KMLStyler transformer.
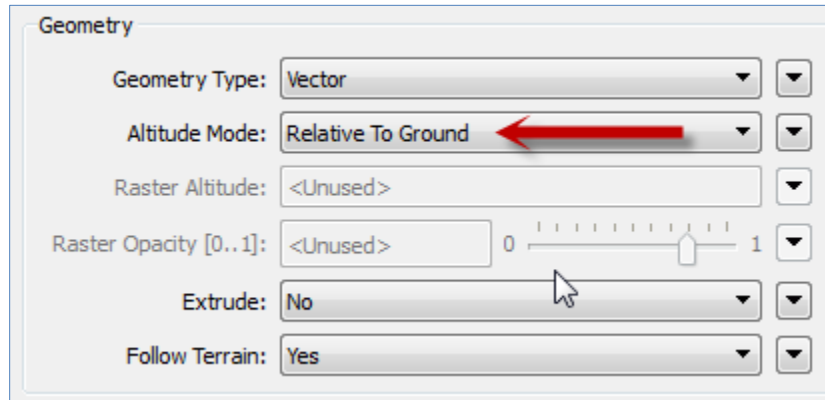
Open the parameters dialog and set the elevation parameter to 100.

8) Add a KMLPropertySetter

Now add a *KMLPropertySetter* transformer after the 3DForcer.

Open the parameters dialog and set the Geometry:Altitude Mode to *Relative to Ground*.

9) Save and Run Workspace

Save then run the workspace. Inspect the output.

The pollution features should now appear above the zipcodes, which are still clamped to ground level. The exact colors may differ, since – remember – the source data is being randomly generated.

Advanced Task

Instead of just coloring the pollution features, why not turn them into three-dimensional columns, the height of which determines the amount of pollution?

The steps required would be:

- Update the *3DForcer* to set elevation relative to pollution. Since pollution is on a scale of 1 to 10, you will probably need to use a multiplying factor. The arithmetic editor built into the *3DForcer* will be ideal for this purpose.

- Change the KMLPropertySetter Extrude parameter to yes.

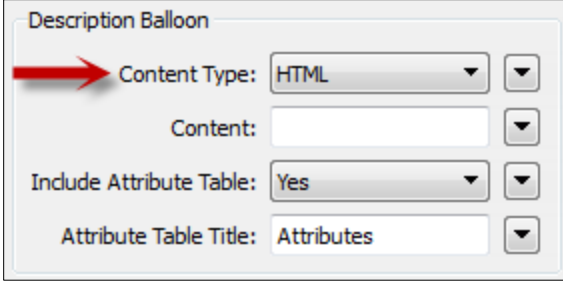If you are unsure how to do this, check with your instructor or open the completed workspace.
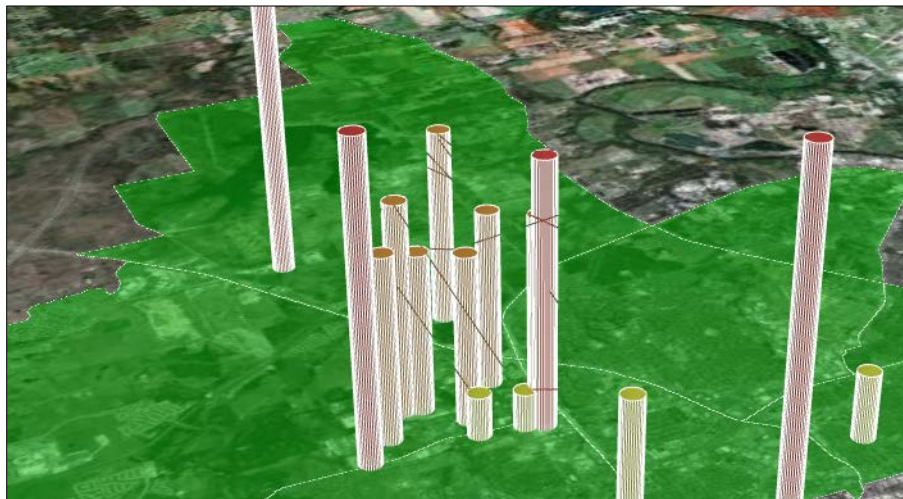
HTML Content

The Description Balloon section in the KMLPropertySetter transformer is where you can make changes to the content that pops up when a feature is clicked on in Google Earth.

Because KML allows HTML content within this field, the possible uses are really quite large.

When using HTML content, it is important to set the Content Type parameter, so that FME can properly encode the text within the Content parameter.

Then, virtually any HTML can be pasted into the Content parameter, or constructed using the String Editor.

For example, in YouTube locate the code to Embed the movie in a website, and paste it into the Content parameter in the *KMLPropertySetter*:
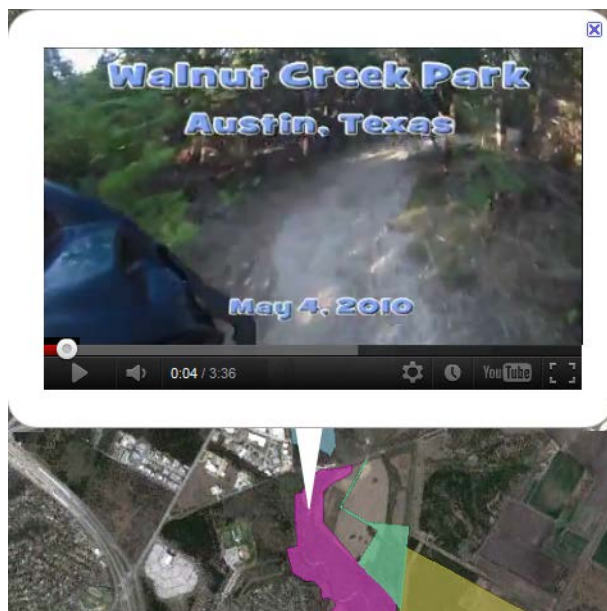

The result is a YouTube movie embedded inside the map, and activated by querying a feature.

Another example could be a link to download the data using FME Server:
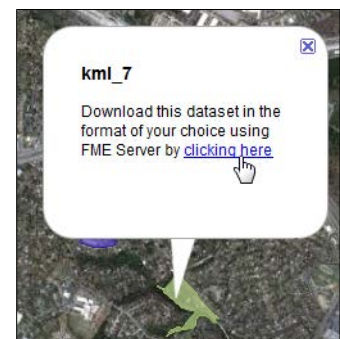
By default, FME includes a copy of the writer attribute schema and values in the output balloon.

To disable this, set the Include Attribute Table parameter to No, in the KMLPropertySetter.

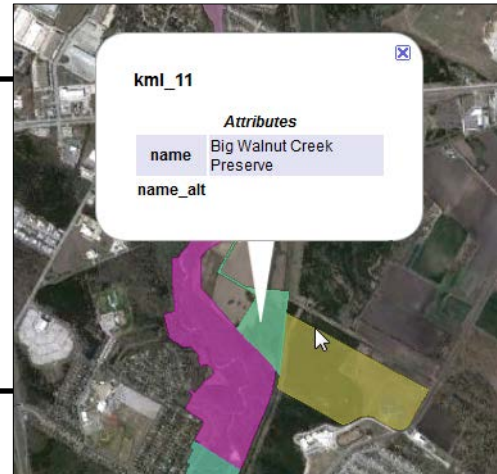| Example 3: Mapping Emergency Services	
Scenario	FME user; City of Interopolis, Planning Department
Data	Emergency Facilities (Input: Idrisi; Output: KML)
Overall Goal	Show emergency facilities in the city
Demonstrates	HTML Balloon Contents
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\KML3Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\KML3aComplete.fmw C:\FMEData\Workspaces\PathwayManuals\KML3bComplete.fmw

Let's look at ways the KML data can be made to show HTML content in the description balloon. This example will add photographs to the balloon contents.

1) Start Workbench
Start Workbench and open the beginning workspace. Notice that it reads from an Idrisi format dataset of Emergency Facilities.
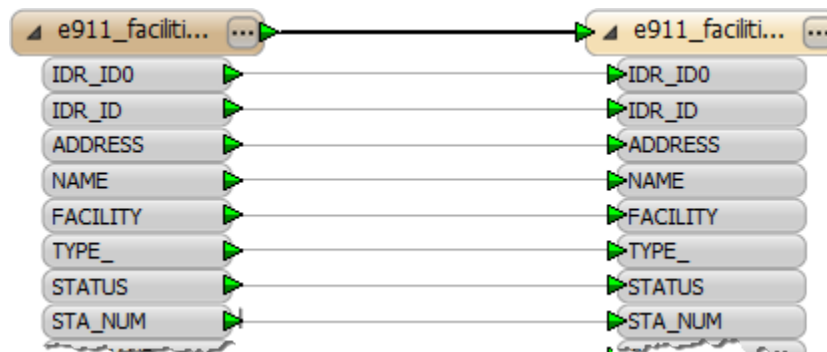

There are various source attributes including one called NAME_FULL.

Check this attribute's values in the Data Inspector (table view) and compare them against the names of images in the folder C:\FMEData\Data\Emergency\Photos

2) Set Coordinate System

First things, first. Idrisi format does not store coordinate system information, and the KML writer will not tolerate data it does not know how to reproject to Latitude/Longitude.

Therefore, in the Navigator window of Workbench, set the Reader coordinate system parameter to TX83-CF.
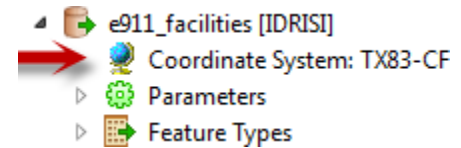
3) Add a KMLPropertySetter

Add a KMLPropertySetter transformer between the Reader and Writer feature types. Open the parameters dialog.

Set the Navigation Tree:Name parameter to the attribute NAME.
Set the Navigation Tree:Summary parameter to the attribute FACILITY.
Ensure the Navigation Tree:Visible parameter is set to Yes.
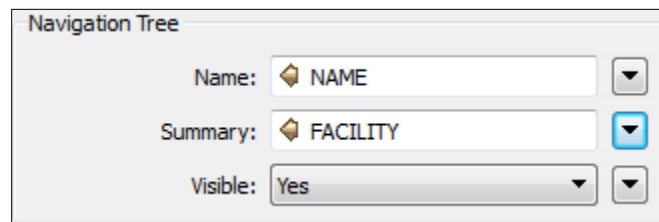
This will set a name and information for the data when it is viewed in Google Earth.

4) Save and Run Workspace

Now save and run the workspace.

In Google Earth the navigation tree will look like this:

Individual features will looks like this:

5) Update KMLPropertySetter
Back in Workbench, re-open the KMLPropertySetter parameters dialog.
Set Description Balloon: Content Type to HTML, and Include Attribute Table to No.
Click the drop-down arrow to the right of the content parameter, and choose Open Editor…

In the String Builder dialog, click the button Switch To Advanced:
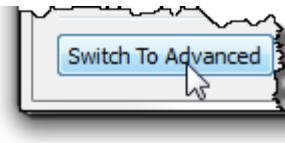
In the editing field enter:

```
<h4>@Value(FACILITY)</h4>
This facility is located at @Value(ADDRESS), Austin, TX
<img src="@Value(NAME_FULL).jpg">
```

Where the content contains @Value(AttributeName) you can insert this by double-clicking the attribute in the menu on the left-hand side of the string builder dialog:
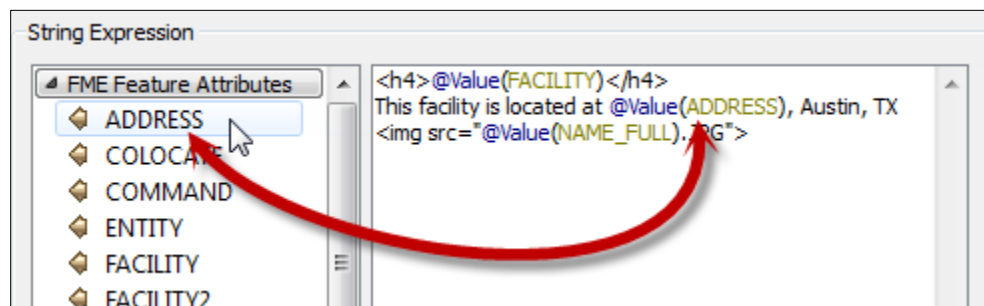
Note that the image name can be given without a path, only because the output is being written to the same folder in which the image files exist. Otherwise this would need to be given the full path.

6) Add KMLStyler
Add a KMLStyler transformer to give a nicer-looking icon to the features.

Run the translation and inspect the output in Google Earth.
Clicking on a feature should show a photo of that facility in the description balloon.
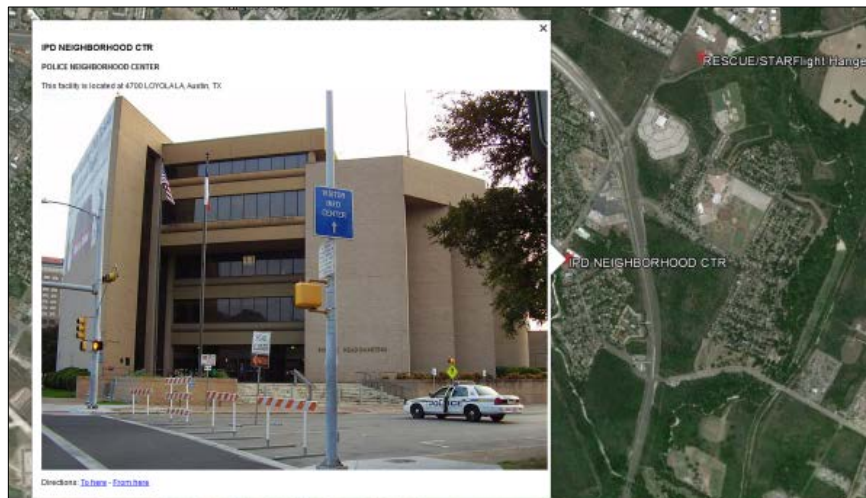
Advanced Tasks

URLs can be used for the icon name in the KMLStyler. Why not search for an image online that could be used as an icon, and paste its URL into the Icon Name parameter?

Also, why not try adding a little more inventive HTML if you are skilled in that field?
Perhaps you could add a link to Google Maps around the facility address part, or use the @XValue and @YValue functions to return a coordinate to search on?

Remember, the coordinates are not in latitude and longitude at that point, and would need reprojecting first!

Manual Styling

With the help of the KMLStyler, it's possible to create various styles manually; i.e. not automatically named, defined and applied using FME. It's even possible to create different styles for the same feature, which occur on different events.

To create a style manually requires a special feature type called Style.
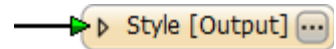
Each feature that enters a Style feature type will define a unique style type in the KML output. The name of the style is defined by a format attribute called kml_id, that you can create with the AttributeCreator transformer.
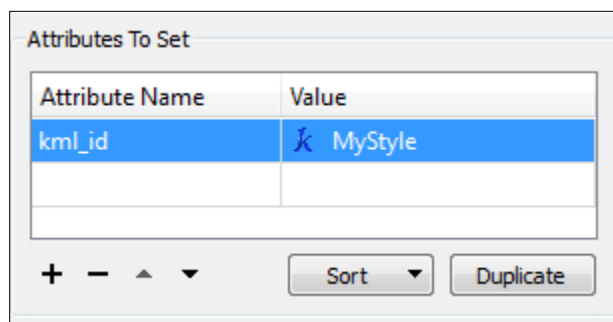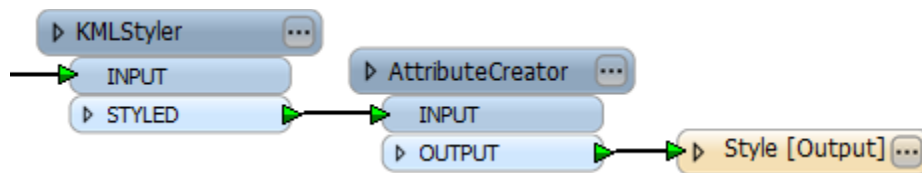
Here a user is creating a style called MyStyle. It will be given the symbology defined in the KMLStyler transformer.

Features reference a style by using one of two format attributes (again with an AttributeCreator):

- kml_style_url_normal
- kml_style_url_highlight

The value of each attribute should be set to the name of the style that is to be used for that feature. "Normal" is the default style used; "highlight" is for when the feature is highlighted – for example by hovering the mouse over the feature.
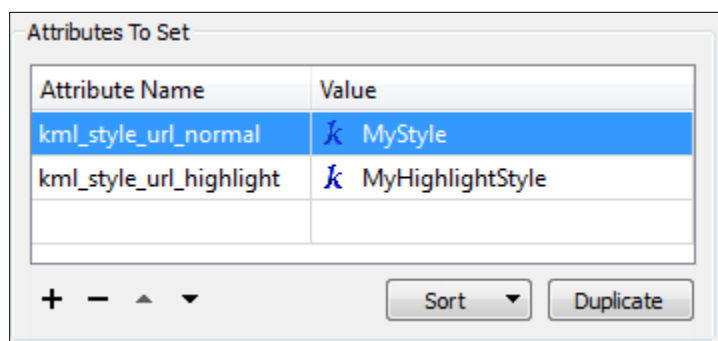
Example 4: Highlight Styling	
Scenario	FME user; City of Interopolis, Planning Department
Data	Emergency Facilities (Input: Idrisi, Jpeg; Output: KML)
Overall Goal	Show emergency facilities in the city
Demonstrates	Manual Styles
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\KML4Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\KML4Complete.fmw

This example follows on from the previous example by applying a highlight style to features.

1) Start Workbench
Start Workbench and open the beginning workspace (or continue with the workspace from the previous example). There is a single *KMLStyler* and so a single style is being applied.

Firstly, create a new Writer feature type. Use either Writers > Add Feature Type (from the menubar) or select the existing feature type and press Ctrl+D to duplicate it.
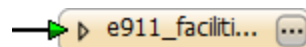
Name the newly created feature type: Style
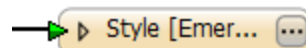
2) Set up Manual Style
Disconnect the *KMLStyler* from the existing writer feature type, and connect it to the Style feature type. Make a second connection from the *KMLPropertySetter:OUTPUT* port to the original writer feature type.

The workspace will now look like this:
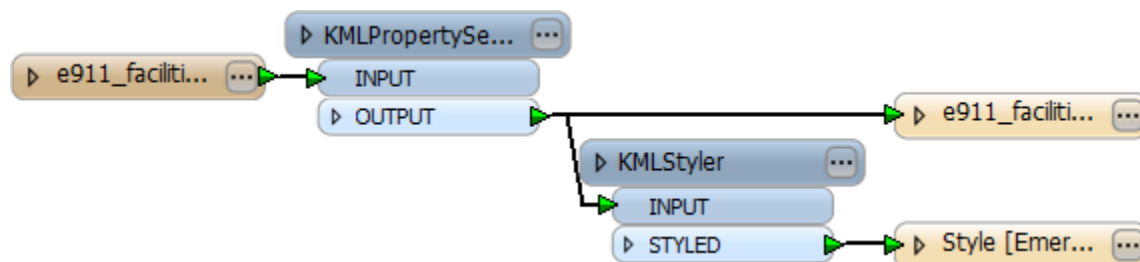
3) Add AttributeCreators

The next task is to set a name for the style, and have the original features make a reference to it. Add two AttributeCreator transformers, one for each writer feature type connection:
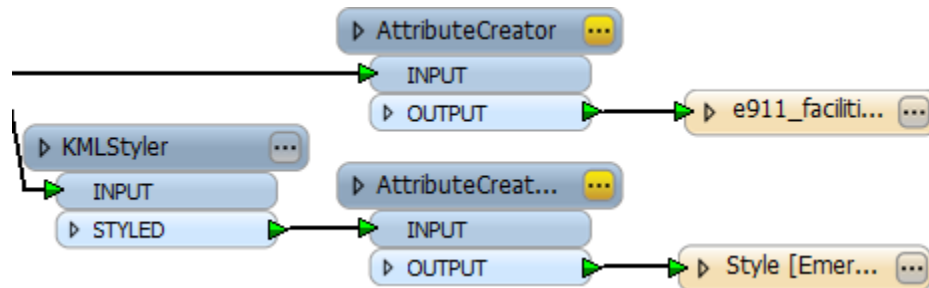
4) Set Style Names

Now each AttributeCreator can be set up to define a style name and reference.
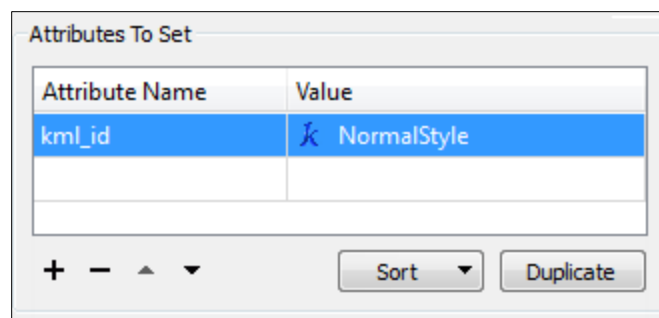
Open the parameters dialog for the *AttributeCreator* connected after the *KMLStyler*. This will define the name of the style.

Create an attribute called kml_id with the value NormalStyle
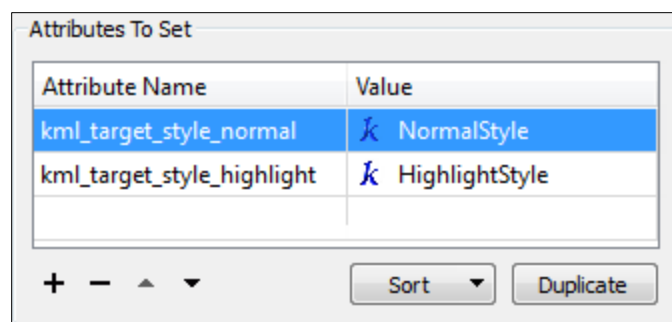
Open the parameters dialog for the other AttributeCreator. This will define references to the style.

Create an attribute called kml_target_style_normal with the value NormalStyle.

Create a second attribute called kml_target_style_highlight with the value HighlightStyle

5) Add KMLStyler

As you'll have noticed, if the workspace is run at this point, the output will have a normal style, but not one for highlights.

To remedy that add a second KMLStyler transformer.

It should be attached as another output from the KMLPropertySetter
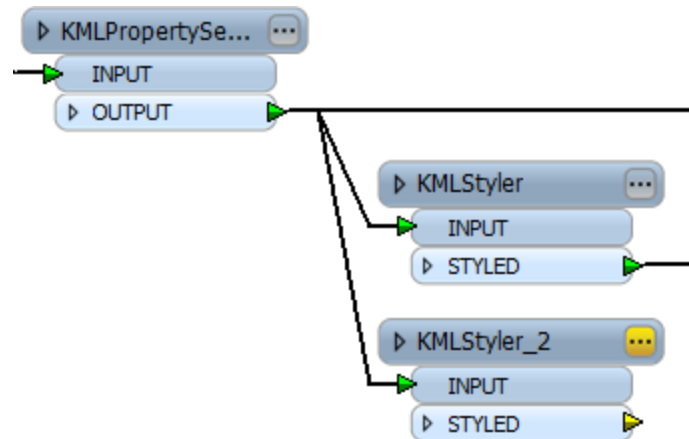
Open the parameters dialog and this time select an icon that is brighter (or larger) than the icon defined for the normal style.

Place another AttributeCreator after this KMLStyler and use it to create kml_id with a value HighlightStyle. Connect this to the Style feature type.

The workspace will now look like this:
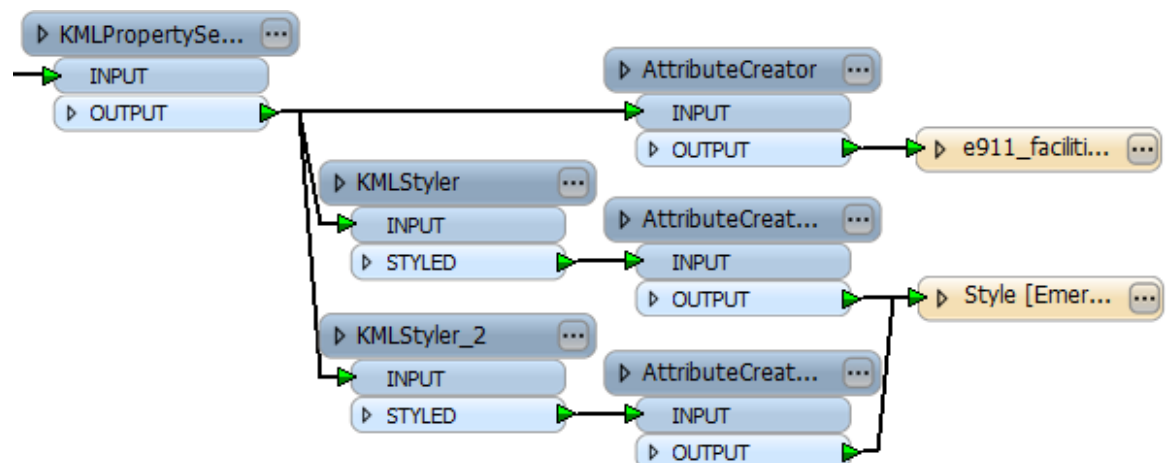
6) Add Sampler

Technically speaking, we only need a single feature to trigger the style creation, and not every single feature. FME will not create duplicate styles, but Best Practice suggests we avoid that scenario anyway.

Add a *Sampler* transformer before the *KMLStyler* transformers. Be sure to connect the SAMPLED port to both of their inputs. Open the parameters dialog for the Sampler and set it up to let only the first feature pass.
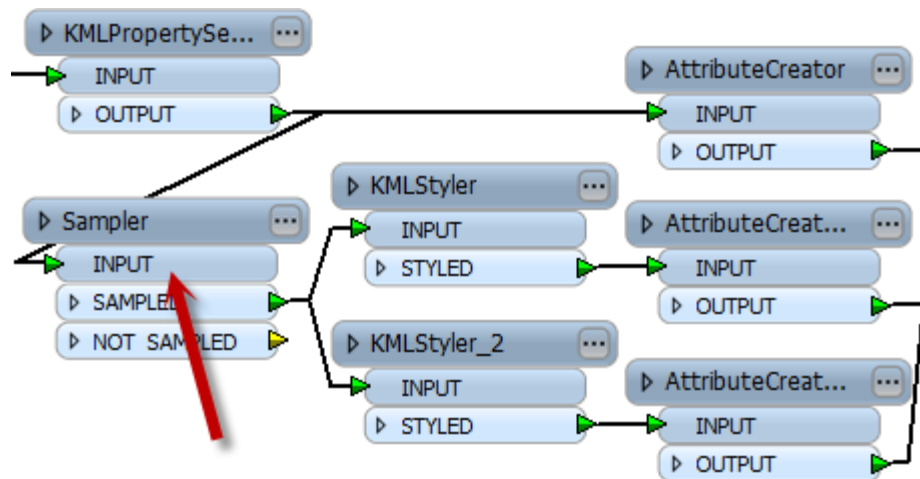
7) Run Workspace

Save and run the workspace. Inspect the output in Google Earth.
Now when you hover over an item, it will be highlighted with a different icon.

Time Functions

To allow temporal mapping all features written to KML format can be assigned a number of time-oriented attributes.

Each feature written to a KML dataset – be it a line point or polygon – may have a related TimePrimitive element. This element can either be a single time stamp (i.e. this feature relates to time X) or a time span (i.e. this feature relates to the period from time X to time Y).

Time Format

KML times adhere to the standard dateTime value defined in the XML specification.
For detailed information the full XML dateTime specification can be found at www.w3.org

For example a date of:

2004-05-22T15:41:00-08:00

…equates to:

3:41pm (Pacific Standard Time) on the 22nd May 2004.

In actual fact the time component of this is not obligatory. Equally valid is:

2004-05-22

Note the 2 digit month & day – 2004-5-22 is not valid.

FME does not validate KML dates to provide a warning or error in the log. You will only find that there is a problem when you open the dataset for display within Google Earth. To reduce problems use the DateFormatter transformer to construct dates in FME.

Time functions are useful for mapping multiple scenarios, including :

- *vehicle tracking*
- *weather systems*
- *sea level changes*
- *census and population*

KMLTimeSetter
KML time stamps can be applied in Workbench using the KMLTimeSetter transformer.
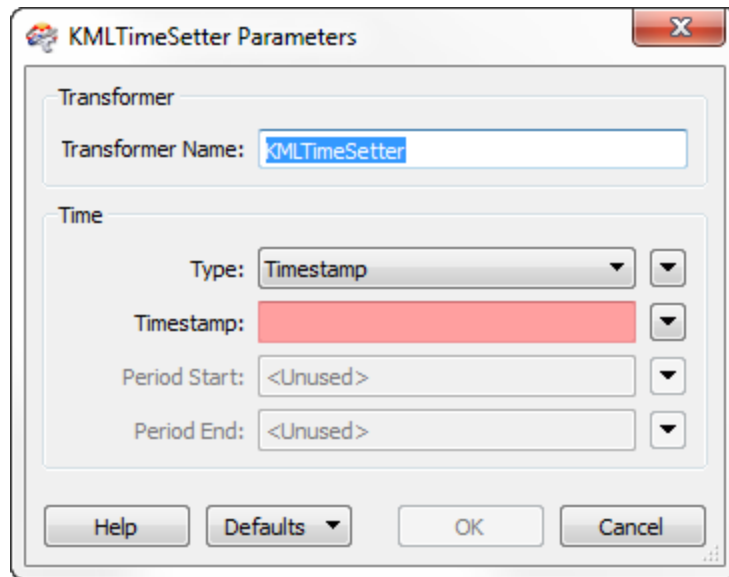
The parameters dialog allows the author to select the type of time primitive (timestamp or timespan) and select the time(s) from various attributes:

In reality the transformer is merely setting the format attributes *kml_timespan_when* (timestamp) or *kml_timespan_begin* and *kml_timespan_end* (timespan).

Viewing Time Data
When a KML dataset contains TimeStamp or TimeSpan tags, Google Earth automatically detects these and adds an extra control for managing the display.
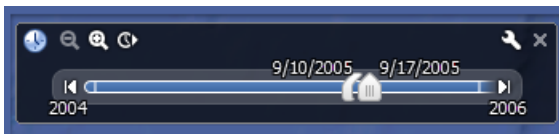
Take particular note of the "play" or "run" button. Clicking this will cause the whole display to be animated.

The time bar will move slowly along the scale from the minimum to the maximum value, and features within the main display will be turned on and off according to whether their time stamp (or time span) falls within the current date range.
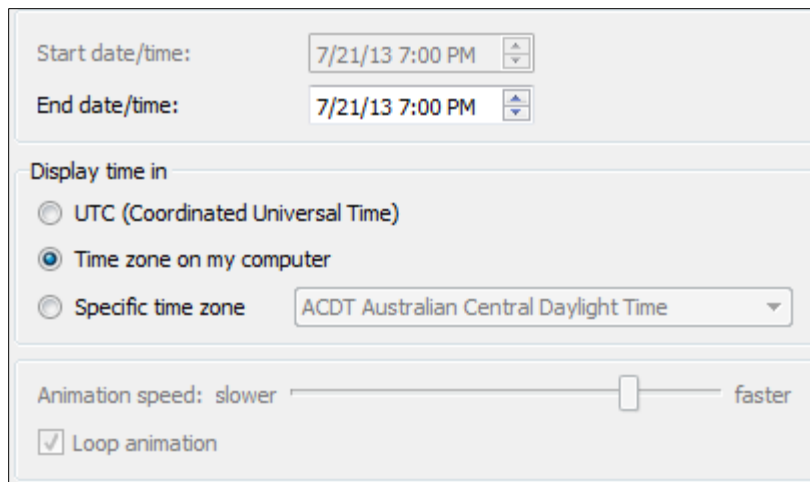
Various options allow the user to adjust the speed of the display, and the date range to be displayed:

Example 5: Time Span Dataset	
Scenario	FME user; City of Interopolis, Planning Department
Data	Cycle Log (Input: GPX; Output: KML)
Overall Goal	Show a cycling route in a city park
Demonstrates	KML Timespans
Starting Workspace	C:\FMEData\Workspaces\PathwayManuals\KML5Begin.fmw
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\KML5Complete.fmw

Let's look at a very simple example of timestamping a KML dataset.

1) Start Workbench
Start Workbench and open the beginning workspace.

Notice that it reads from a GPX dataset representing the GPS log of a cycle ride through a city park.

Inspecting the source data will show other feature types, but the only one in this workspace are the TRACK point features.
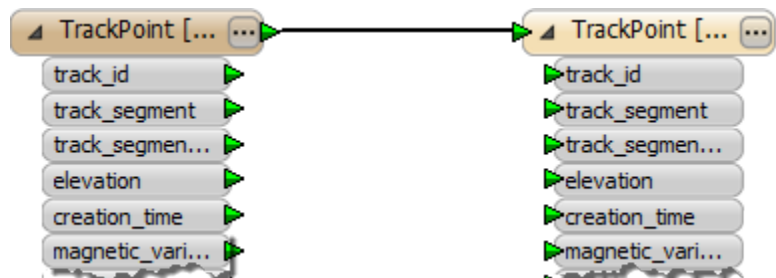
Pay particular attention to the creation_time attribute.
Luckily it is already in the correct format for use in a KML timestamp.

2) Add KMLTimeSetter
Add a KMLTimeSetter transformer between the reader and writer feature types.
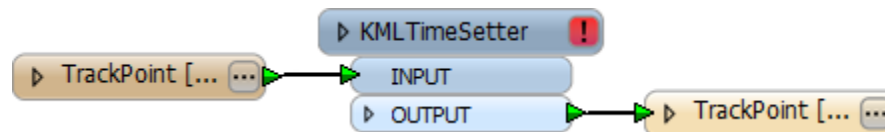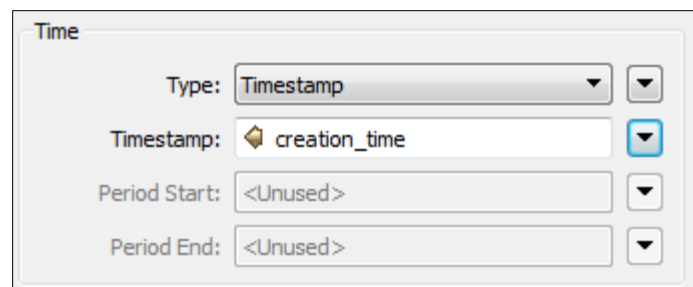
Open the parameters dialog.

Set type to Timestamp.

Set the Timestamp parameter to read from the attribute creation_time

3) Add KMLStyler

Add a KMLStyler transformer to select a suitable icon for the output features.
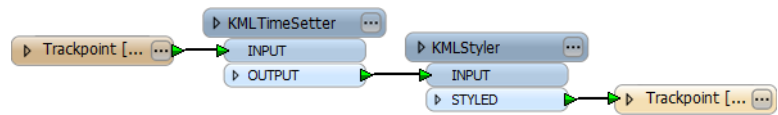
4) Add CoordinateSystemSetter

The source data is – as yet – not referenced with a coordinate system. It should be.

Add a *CoordinateSystemSetter* transformer. This is an alternate method of tagging a feature with a coordinate system. Open the parameters and set the coordinate system to LL83.
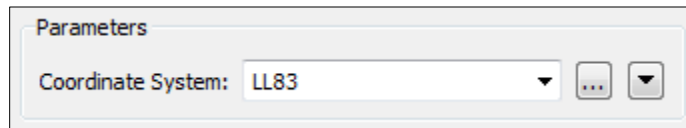
5) Run Workspace

Save and then run the workspace.
Inspect the output in Google Earth.

In the time-control panel, drag the right-hand control into the center of the panel.
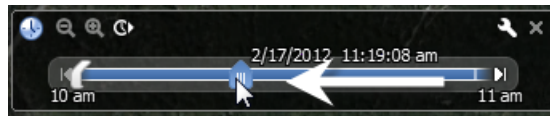
Then drag the left-hand control up close to meet it.
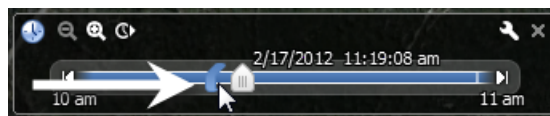
Optionally, click on the wrench icon to open a dialog in which to change the speed of the display:
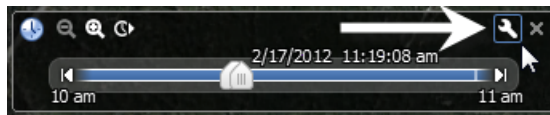
1

Then click the play button to run the animation:
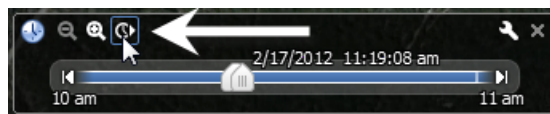
The display will now show an animated sequence, where each feature pops into view for a short time, when its timestamp falls between the left- and right-hand controls.

Raster Handling in KML

KML format supports raster features and FME is equally able to supply data using that structure.

FME can write raster features directly to a KML (or KMZ) dataset. It is even able to write a combination of vector and raster features to the same output file!

The OGC KML writer works with raster features sent to the writer (as opposed to just copying the original source) and carries out proper raster reprojection, rather than just doing a vector reproject on the raster's bounding box.

KML or KMZ?

In most instances you will want to write your raster data to a KMZ output file. You do this by simply specifying KMZ as the output extension to use.

By default, writing data to a file with a KMZ extension will cause a single KMZ file to be created containing both a header document (*doc.kml*) and a sub-folder containing the raster image(s).

When the file extension is set to KML, then the output is somewhat different.

The KML file itself is the header document. It contains pointers to the raster image(s) which are stored in a completely separate folder called *images*

Raster features that are reprojected will often contain a black border around the edge of each file.

This can be fixed by adding an alpha band and setting its values to zero (must be an Alpha-supporting format) for nodata areas.

When both mosaicking and reprojecting raster data, be sure to do the mosaicking first.

Raster Handling Mode
The Raster Handling Mode is an advanced writer parameter found in the navigation window in Workbench**:**

The three available modes are:
* Write
* Copy
* Relative

Advanced
- KML hint attribute: <not set>
- External Style Document: <not set>
- External Schema Document: <not set>
- Generate Raster Ground Overlays: Yes
- Raster Handling Mode: write
- Raster Output Format: tiff
- Preferred Texture Format: Auto
- Copy icons to destination dataset: Yes

Write Mode
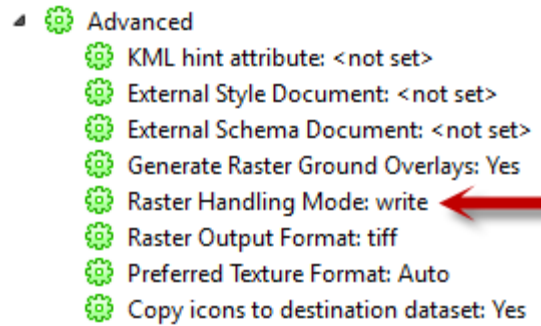Write Mode means that the KML writer uses whatever raster features that are delivered to it. This means that it can handle any raster – regardless of its origin – and write it as it has been processed by any transformers within the workspace.

Copy Mode
Copy Mode means that the KML writer examines an incoming raster feature, and uses its original source dataset, by copying that file into the output KML folder. The resulting GroundOverlay node will reference the file copy.

Obviously this will only work for raster formats that KML supports (PNG, JPEG or TIFF), and could not be used for non-file raster formats such as an Oracle GeoRaster. Also it does not take into account any processing that has taken place within workspace transformers.

Relative Mode
Relative Mode is the same thing as Copy Mode, except that instead of making a copy of the source raster file, it writes a GroundOverlay feature that references the original.

This avoids creating extra copies of the source, but does make data management slightly more difficult when you intend to pass on the KML dataset to another user.

Regionating Raster Data
Raster data can be regionated (split into sections) using a Super-Overlay.
A parameter for controlling super-overlays exists in the Navigator window.

- Omit Document Element: No
- Regionate Vectors (Beta): No
- Generate Super-Overlays: No
- Pyramid PhotoOverlays (Beta): No
- Create Empty Folders: No
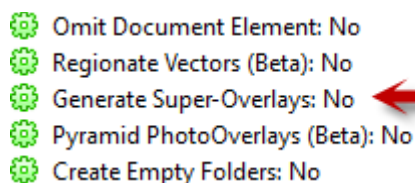
Regionating data – particularly rasters – can make it easier to control and faster to load in Google Earth.

⊿ ⚙ Advanced
    ⚙ KML hint attribute: <not set>
    ⚙ External Style Document: <not set>
    ⚙ External Schema Document: <not set>
    ⚙ Generate Raster Ground Overlays: Yes
    ⚙ Raster Handling Mode: write
    ⚙ Raster Output Format: tiff ⟵
    ⚙ Preferred Texture Format: Auto
    ⚙ Copy icons to destination dataset: Yes

Raster Output Format
The Raster Output Format parameter allows a user to change the type of raster being written to the KML dataset.

The four options are JPEG, TIFF, PNG, or GIF

In general terms, a TIFF output provides better quality, but at the cost of larger file sizes.

Edit Parameter

Raster Output Format: tiff

OK   Cancel

Obviously this parameter has no effect unless the Raster Handling Mode is set to "write".
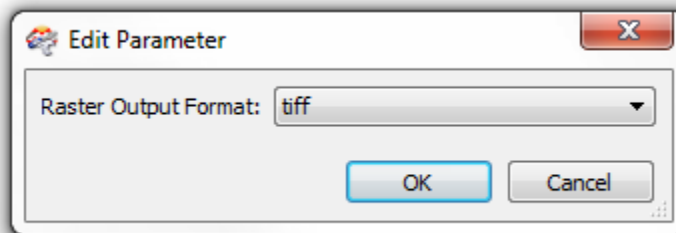
Opacity
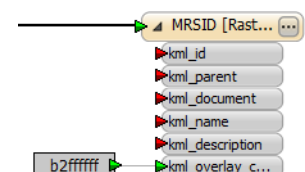Opacity in a raster dataset is set using the format attribute *kml_overlay_color*

However, the value for this attribute is not on a simple 0 to 1 scale, but an ARGB value. ARGB is like a simple RGB value, but the extra A (for Alpha) refers to the opacity value.

Each value of A-R-G-B is on a scale of 0 to 255, but as a way to speed up graphic processing calculations within a KML browser, (and not intentionally to complicate things for a user) the values are each supplied in a two-digit hexadecimal format.

In simpler terms the value for *kml_overlay_color* should be an 8 digit number between 00FFFFFF and FFFFFFFF. The last six digits (FFFFFF) are a mask that is a bitwise AND (if that sort of information interests you), but it is the first two digits – representing the opacity – that are the important part, giving:

00FFFFFF	No transparency	00 is hex for 0
40FFFFFF	25% transparency	40 is hex for 64
7FFFFFFF	50% transparency	7F is hex for 127
C0FFFFFF	75% transparency	C0 is hex for 192
FFFFFFFF	Fully transparent	FF is hex for 255

Although constants are not often part of FME Best Practice, it's unlikely a user would want to set different opacities for different raster features, so in this case a constant is an acceptable method.

⊿ MRSID [Rast... ⋯]
▶ kml_id
▶ kml_parent
▶ kml_document
▶ kml_name
▶ kml_description
b2ffffff ▶   ▶ kml_overlay_c...

Example 6: Raster DEM	
Scenario	FME user; City of Interopolis, Planning Department
Data	Raster DEM (Input: CDED; Output: KMZ-GeoTIFF)
Overall Goal	Display a raster DEM within Google Earth
Demonstrates	KML and Raster
Finished Workspace	C:\FMEData\Workspaces\PathwayManuals\KML6Complete.fmw

This example involves converting a raster DEM to a KML dataset.

1) Start Workbench
Start Workbench and create a workspace to convert a raster DEM dataset to KMZ format.

Reader Format   Canadian Digital Elevation Data (CDED)
Reader Dataset   *C:\FMEData\Data\ElevationModel\RasterDEM-250K.dem*

Writer Format   Google Earth KML
Writer Dataset   *C:\FMEData\Output\DesktopTraining\Elevation.kmz*

2) Add RasterInterpretationCoercer
The source dataset contains raster information held as 32-bit integers (*INT32*) that is incompatible with a KML raster dataset. This can be fixed by changing the raster interpretation.

Add a *RasterInterpretationCoercer* transformer. Open the parameters dialog and set the Destination Interpretation Type to *Gray8*.

3) Check Coordinate System
Check (in the FME Data Inspector) whether the source data has a coordinate system applied to it, and take the appropriate action (if any) to be able to write the data to KML.

4) Add AttributeCreator
Add an AttributeCreator transformer.

Open the parameters dialog and create *kml_overlay_color*
Set the opacity of the output to 70% (B2 is the hex code you need).
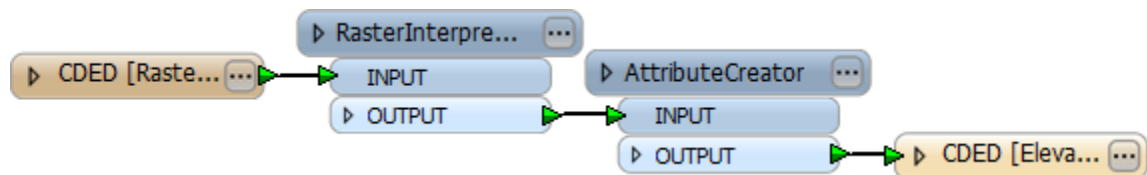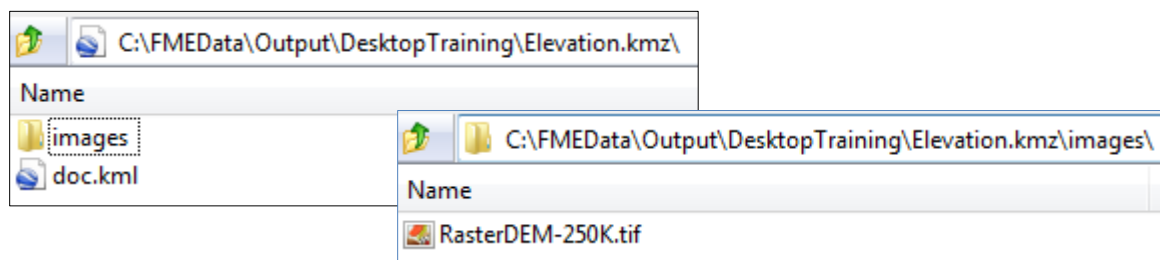
5) Run Workspace
Save the workspace and then run the workspace. Inspect the output in Google Earth.

The output should look something like this. See how the underlying Google Earth background is partly visible under the GroundOverlay.

6) Create Super-Overlay
Explore the contents of the KMZ file using a tool like WinZip.
There should be a single kml file referencing a single image:

However, in Google Earth it is often more efficient to work with a set of tiled raster data.

In the Navigator window, locate the writer parameter Generate Super-Overlays, and set it to Yes.

Now re-run the workspace and re-examine the contents of the KMZ file.
There should be multiple kml files referencing multiple images.

Reload the data in Google Earth. You should see it load as a set of tiles, rather than a single image, and be able to turn tiles on and off in the Places window.

Session Review

This module was designed to assist you in using FME to generate useful and interesting KML format datasets.

What You Should Have Learned from this Module
The following are key points to be learned from this module:

Theory

- **KML** is a format that is to spatial data as HTML is to plain text.

- **KML** is restricted to using the coordinate system LL84 (or EPSG:4326), all features must be three-dimensional, and all must have a unique ID number.

FME Skills

- Use FME to **convert vector data** to KML format.

- Use FME to **convert raster data** to KML format.

- Use the **KMLStyler transformer** to apply basic KML symbology.

- Use the **KMLPropertySetter transformer** to apply names and descriptions

- Create KML **Timestamps and Timespans** to describe data through time