LDraw Tutorial Part 13

MILS with BlueBrick (II)

By Jetro

In the first part we saw how to create an image of a module and prepare it for use in BlueBrick. In this second part we will see how to add functionality to these modules.

Connection points

One of the strong points of BlueBrick is that it allows for inclusion of connection points in a module. In the second article on the MILS system which is published in this edition, several elements have been introduced that can benefit from these connection points.

BlueBrick already has several types of connections points (for roads on baseplates, train tracks, DUPLO tracks and monorail tracks) and the system can be easily expanded. To this end the following steps must be taken:

1 - Define a connection type

Connection types are defined in the file ConnectionTypeList.xml which is located in the Config folder of your BlueBrick installation. The file can be easily edited using any text editor, like notepad in Windows (to open it right click on the file and select “open with” or open it directly from the program of your choice. If you open it in a browser - usually the default option - you won’t be able to edit it).

Between the tags <ConnectionTypeList> and </Connection TypeList> you will find several blocks of code that look (approximately) like this:

```
<ConnectionType name=”1”>
        <ColorARGB>FFFFFF00</ColorARGB>
        <Size>1</Size>
</ConnectionType>
```

The first field, <ConnectionType name=”1”> contains the name of the connection type. In this case it is “1” which corresponds to train tracks, but you can use a descriptive name for the connection type, as long as you place it between quotes, for example “MILS River”.

The second field, <ColorARGB>, indicates the colour of the dot that will indicate the connection point in ARGB format. This format is similar to RGB which you may already know, but is preceded by two digits indicating the opacity of the colour. To simplify things, you can use an RGB colour and precede it with “FF”. In this way the LEGO colour Dark Blue, represented by 0A3463[1] would become FF0A3463.

The third field, <Size>, indicates the size of this dot.

In order to create a connection type for MILS Rivers, we could add the following code to the list:

```
<ConnectionType name=”MILS River”>
        <ColorARGB>FFFFFF00</ColorARGB>
        <Size>1</Size>
</ConnectionType>
```

Repeat the process for all the connection points you are going to need and save the file in its original location[2]

2 - Adding connection points to a module

To include these connection points in the corresponding modules some lines need to be added to the XML file that goes with the module. This file is located in the same folder as the .gif image of the module, as was explained in the previous article.

After the description of the module the tag <ConnexionList> is added, after which the connection points of the module are described.

The block of code will look like this:

```
<connexion>
        <type>MILS River</type>
        <position>
                <x>0</x>
                <y>-16</y>
        </position>
        <angle>-90</angle>
        <angleToPrev>-90</angleToPrev>
        <angleToNext>0</angleToNext>
        <nextConnexionPreference>1</
nextConnexionPreference>
</connexion>
```

Although it may look complex, it is actually quite simple. Between the tags <type></type> the type of connection is indicated. In this case it’s a MILS River Next the location of the connection point is indicated. This is calculated from the centre of the module, with X increasing from left to right and Y from top to bottom. In this way the centres of the four edges of the module have the following coordinates (starting at the top and going clockwise). X=0, Y=-16; X=16, Y=0; X=0, Y=16 y X=-16, Y=0. To make identifying these points easier I will call them A, B, C and D [Table 1]

The field <angle> indicates the direction of each point. Since in MILS this angle is always perpendicular to the border and the 0º angle is in the direction of the X axis, the values are as follows: A= -90, B=0, C=90, D=180.

	X	Y	Angle
A	0	-16	-90
B	16	0	0
C	0	16	90
D	-16	0	180

The next fields, <angleToPrev> and <angleToNext> depend on the number and location of the connection points. In this example the values are always -90 and 90 respectively, but you need to look at the angle between one point and the next in each case and remember positive values are clockwise.

Finally there is the field <nextConnexionPreference>. This serves to indicated which point is selected by default and in what order the other points are selected. Keeping in mind that this list begins with he number 0, if we want to follow the order A, B, C, D we will have to indicate 1, 2, 3 and 4 respectively, but of course this order can be changed to fit your preferences. [3]

Why make connection points?

Connection points make it easier to place different elements in the layout. After placing the first element you can rotate it using the space bar until placing it in the desired orientation. After that, using the enter key, you can select the next connection point. Now with a simple click on any module in the Parts pane on any module that can connect to this point it will do so automatically. If you now use the space bar, the module will only turn in such a way as to leave a valid connection (it will never allow for the connection of two different types of points).

A second way to (re)create a module

In addition to creating an DLraw version of a module, it is possible to create an image of a module with a different, less time consuming process, although there are some disadvantages

If you take a picture of the top view of a module and crop it (using the process described in the first part to create an image from a virtual module) you can get a representation of an existing module with relatively little work.

However, you should keep in mind the following inconveniences of this method. In the first place, taking a picture from a “bird’s eye view” of a module isn’t as simple as it sounds. The first hurdle is taking the picture perfectly perpendicular to the centre of the module. Otherwise, the picture of the module will not be perfectly square; if for example you take the picture perpendicular to the base of the module, the opposite side will look considerably shorter.

Another factor to keep in mind is the distance to the module. The closer you get to the module, the more deformation there will be at the edges of the picture, so it is a good idea to take the picture from some distance and at a high resolution so later you can crop the image and still get enough resolution for a BlueBrick image.

Finally you should take into account the lighting. If the module has some elevated parts it will be hard to avoid shadows in the picture. At the same time, the colour of the module will almost certainly be different from the ones made using LDraw and, unless you manage to recreate the exact same circumstances, modules photographed at different moments will also have colour divergences.

The big advantage of this method, however, is that keeping these factors in mind you can create a BlueBrick module in a relatively short time.

[1] There is a complete list of RGB values for LEGO® colours at http://beta.ldraw.org/article/547.html
[2] At the dedicated MILS website: www.abellon.net/MILS/index.html there is a section where you can download a modified file that contains the connection points explained in this issue.
[3] At the dedicated MILS website: www.abellon.net/MILS/index.html there is a section where you can download several MILS modules and their corresponding XML files which may serve as a basis for any other module.
#