Micropolis Autoplace Manual

Leo Gummersbach

March 2, 2023

Contents

1 Introduction **2**

2 Standard **2**

3 The Expanded Standard **2**
3.1 Default Modules 2
3.2 Custom Modules 3

4 Use Case **4**

5 User Interface **4**
5.1 Selected Customs 4
5.1.1 Configure a custom module 6
5.2 Generation 6
5.3 Found Customs 7

6 Generation **7**
6.1 Custom connections 7
6.2 Key 9

7 About **9**

List of Figures

1 The software when it gets started 5
2 A custom module is selected 5
3 Configure a custom module 6
4 A module with custom connections 8
5 A module with a specified key 10

1 Introduction

This is the manual for the "Micropolis Autoplace" software. The software can be used to combine different types of modules coming from an expanded Mircopolis standard. The modules are build up from building bricks, most commonly known as Lego.

2 Standard

The Lego users group *TwinLUG* defines the Micropolis standard on their website[1] . They only define one kind of module: a 16x16 plate having a street on exactly two sides, forming a corner. The rest of the module is empty, leaving space for own creations. Therefore, combining multiple modules always forms 32x32 sized city blocks.

3 The Expanded Standard

The Micropolis standard is quite ordinary with only one module defined. The reason for that is simplicity: it is easy to combine a large amount of modules in one city "with very little coordination"[1]. To overcome that ordinariness, the Micropolis standard has been expanded to any possible combination of streets within a module. The module size itself remains 16x16. To keep the simplicity of combining a large amount of modules, this software has been developed as a tool for finding fitting combinations.

3.1 Default Modules

There are six types of modules. Each type has a predefined name, and a variety of subtypes with different amounts of single corners. This sums up to 15 module types. They are called *default modules*. The following table lists all subtypes.

[1] `http://twinlug.com/micropolis-micro-city-standard/`

Name	Type	Image
None0000	None	
None0001	None	
None0011	None	
None0101	None	
None0111	None	
None1111	None	
I00	I	
I01	I	
I10	I	
I11	I	
II	II	
L0	L	
L1	L	
U	U	
O	O	

With the ability to rotate every module, the default modules cover all combination of streets and corners.

3.2 Custom Modules

Custom modules are what this software is about. Every custom module modifies a specific default module, but fills the empty part with a nice looking structure. If you wish to use a custom model

in the software, a picture of it needs to be placed in the `custom_modules` folder. The picture must be a `.png`, optimally showing the custom module in an orthogonal view from the top. Pictures like this can be created with the `stud.io` software[1] . There, one must select view → top in order to get an orthogonal view. For a better looking result, it is possible to render the image.

The streets and corners of the custom modules, as well as additional information about them, can be configured in the software itself. They are saved in a `.json` file in the same directory.

4 Use Case

This software has two use cases:

1. Finding a good looking combination for an existing collection of Micropolis modules.
2. Finding the right default module that is left in your collection in order to form a good looking combination. Therefore, select a larger grid size than the amount of selected custom modules.

5 User Interface

The user interface of the software consists of two parts: the menu on the left side and the shown result on the rest of the screen. When opening the software, this result part is empty.

The menu splits up in three different parts:

1. Selected Customs
2. Generation
3. Found Customs

Picture 1 shows the software when it gets started.

5.1 Selected Customs

This is a list of the custom modules that have been already selected. If a list entry is selected, the preview frame shows it's picture. Clicking the *X* button, the custom module get removed one time from the list of selected customs. Clicking the *Street* button, a separate window opens where you can configure the custom module. Below that button, there is a preview of the configured streets and corners of the module.

By clicking the *Clear* button, every items gets removed from the list of selected customs. Below the list, there is a little summary of how many customs modules are currently selected, as well as how many modules will be generated.

Picture 2 shows the software when a custom module is selected.

[1] `https://www.bricklink.com/v3/studio/download.page`

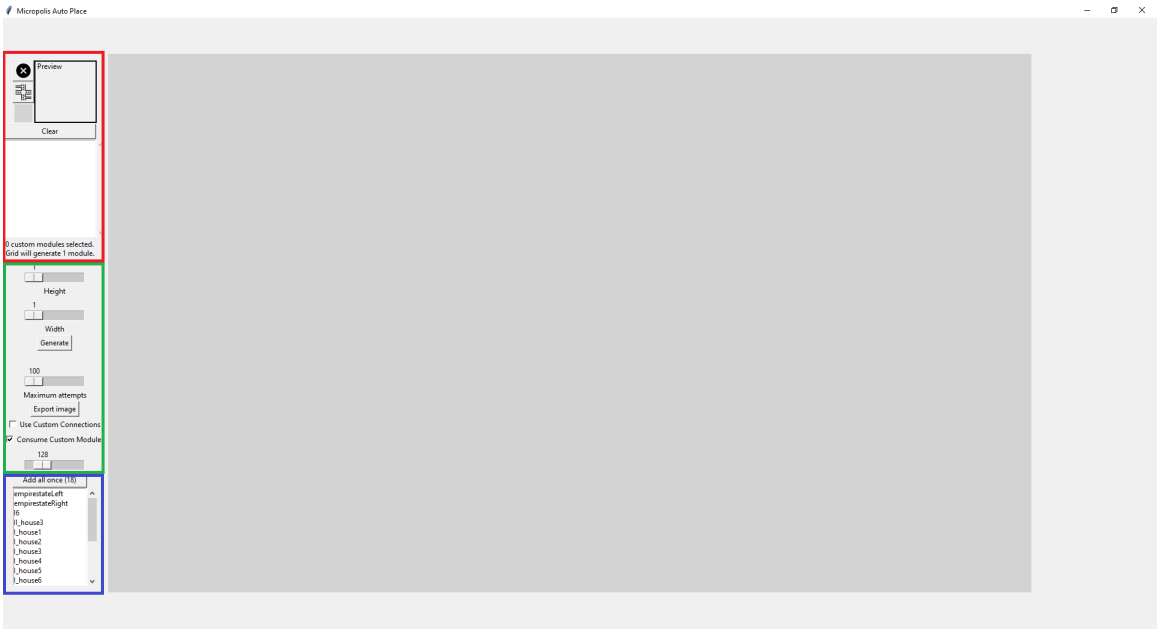

Figure 1: The software when it gets started
The red frame contains the *selected customs* part, the green frame contains the *generation* part and the blue frame contains the *found customs* part.

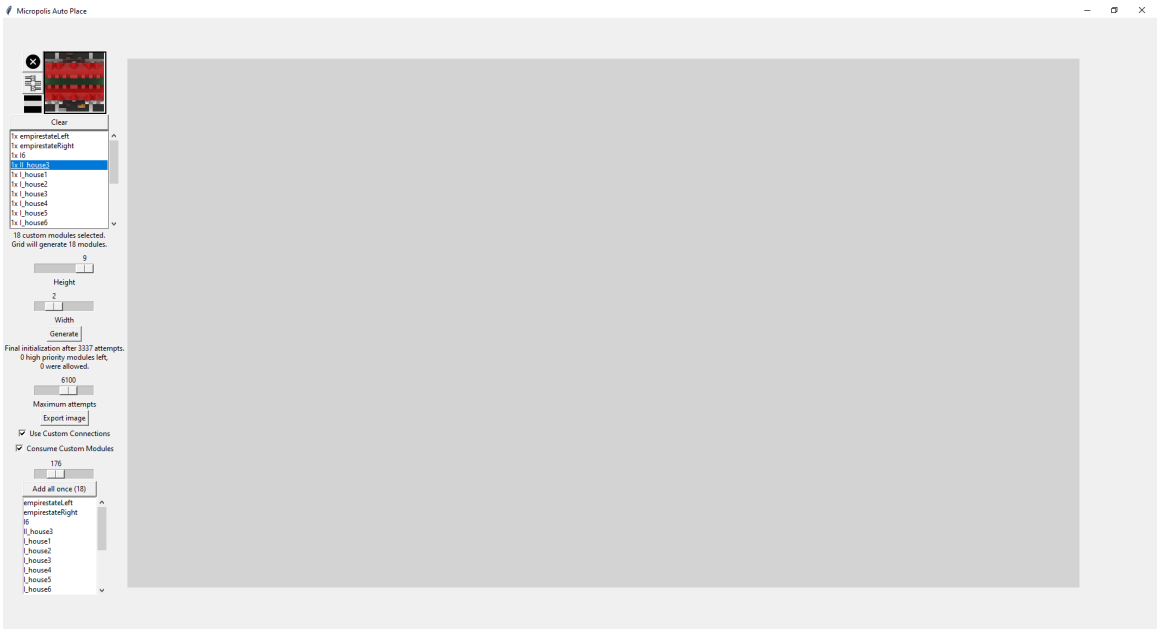

Figure 2: A custom module is selected

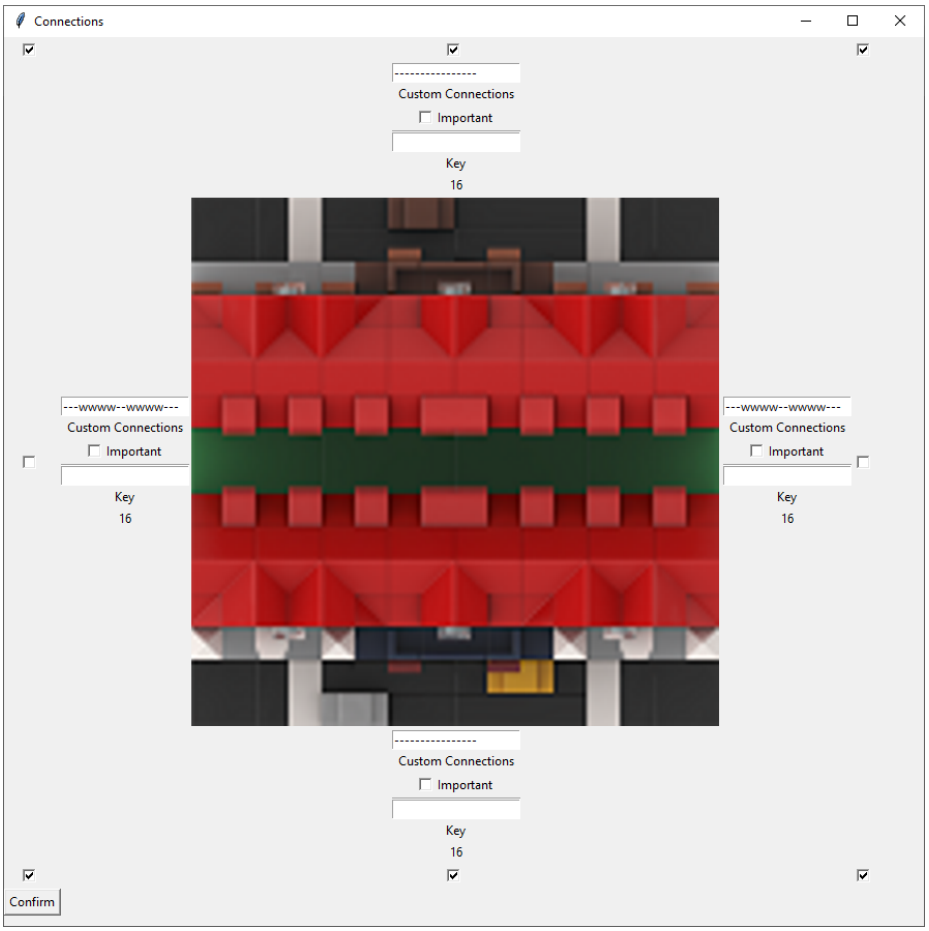

Figure 3: Configure a custom module

5.1.1 Configure a custom module

The separate window for configuring a custom module shows the modules it the center. Around it you can adjust the steets with checkboxes on the edges, and the corners with checkboxes on the corners. For every edge, you may also set up custom connections and a key. When pressing the `Confirm` button on the lower-left side, changes are saved and the window closes when everything is set up correctly. Picture 3 shows the configure window.

5.2 Generation

In the generation section of the menu, one can set up some generation related options. The `Width` and the `Height` scale adjust the size of the generated grid. Size is measured in amount of modules, i.e. 16 stud.

The button `Generate` will generate the grid, more about it in the next section (6). The `Maximum attempts` scale sets up the amounts of maximum attempts for the generation algorithm.

The default value is `100`, and setting a higher value can result in long computation times. The `Export image` button exports the grid image to the `export` folder.

Next up, there are two options. The first one, `Use Custom Connections`, determines if custom connections are used or ignored. The second one, `Consume Custom Modules`, determines if a placed custom module gets removed from the list of selected custom modules or not. Unchecking this option leads to the multiple use of selected custom modules and lets you see what might be possible with infinite bricks.

In the end, there is the scale, where you can adjust the size of the shown generated grid.

5.3 Found Customs

This is a list of the custom modules that have been found by the software in the `custom_modules` folder. If you click on one item, it gets selected and appears in the selected custom module frame on the very top of the menu. You may also select a module multiple times, or select every found module once. The button for this also shows the amount of found modules.

6 Generation

The generation algorithm works straightforward: it starts at the upper-left corner and places a random module. Then it continues completing the row, and the it goes into the next column, filling row for row. For every position, a fitting custom module gets selected randomly. It is fitting, if all neighbor connections are fine, i.e. the existence of streets and corners matches. If no custom module fits, a default modules gets selected.

When the placement is completed, it is determined if all selected custom modules have been placed, how many are left and how many left custom modules were allowed (if the grid size is smaller than the amount of selected custom modules). If the amount of left custom modules is lower or equal to the amount of allowed left custom modules, the algorithm returns and the program draws the result. If not, the algorithm starts over, tries to find a fitting combination. Note that now everytime there exists a fitting combination. The algorithm will restart itself until the maximum amount of attempts is reached. If no optimal combination has been found at this point, the best result gets drawn. Below of the `Generate` button, the drawn result gets explained.

6.1 Custom connections

Every custom module can also have custom connections. They are used to specify the desired neighboring module in a more complex way than just using streets. For example, they can represent train lines, rivers or other, street-like, connections.

Each custom module has for every edge a 16 character long string that represents the custom connection as well as the option if this custom connection is important or not. The default string consists of 16 `-` characters and is not important. So, every character represents a stud on the module's edge. The string goes anti-clockwise (mathematical positive) along the edge.

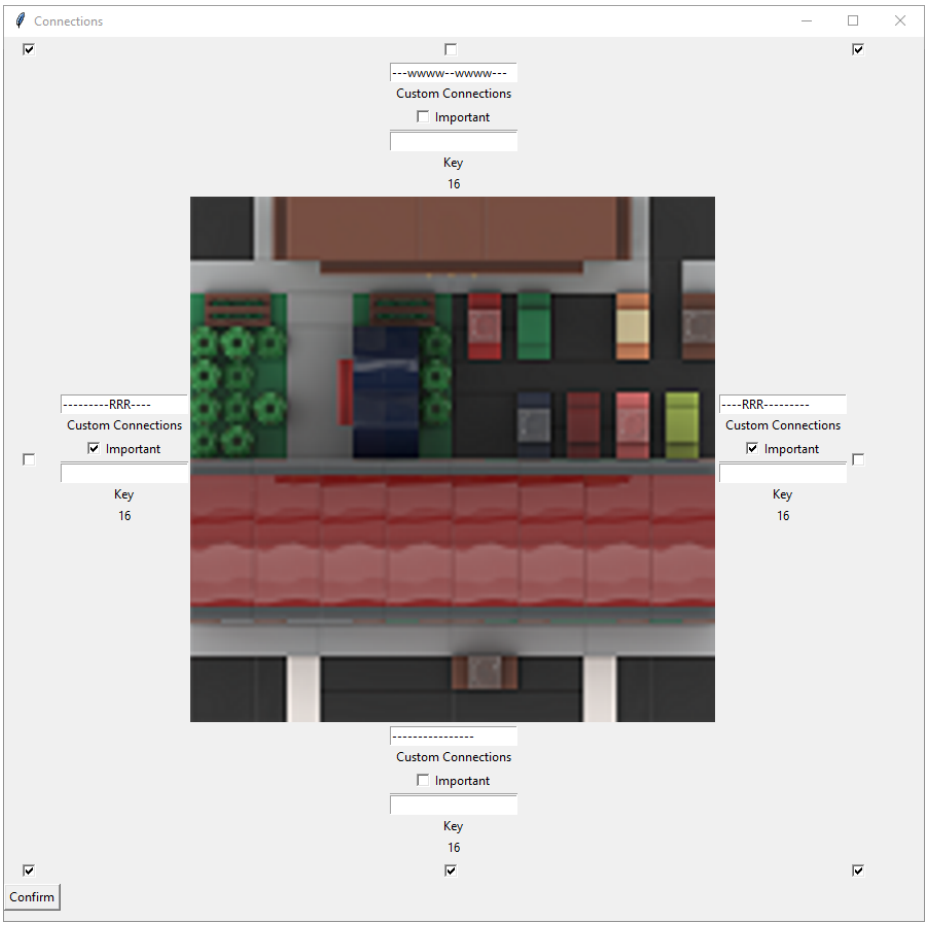

Figure 4: A module with custom connections

If the custom connection on an edge is marked as important, the neighboring module must have the mirrored custom connection string on the adjacent edge. If the custom connection is not important, the neighboring module should have the mirrored custom connection string on the adjacent edge, but it is not required.

Picture 4 shows the custom module configuration window for a module that includes a station. The custom connections on the left and right side are marked as important, as the railway line should continue. The custom connection on the top side desires to places also a building with it's party wall next to this module. But it is not important because it is not a must have.

It is recommended to not set up too much important custom connections and if so, keep an eye on that there are enough modules that allow a connection, also regarding the streets and corners. Important custom connections from a very strong restriction.

Custom connections can use any character for specifying a type of structure that should connect

to another module. Setting up different types of custom connections, one can easily loose control about what represents what. Therefore, it is useful to set up a convention clarifying what a character represents in the model. The following table does so:

ascii value (dec)	character	description
82	R	Railway (elevated, railway tiles are placed 5 plates above the main plate)
83	S	Street
84	T	Train station (major, with roof)
87	W	Water
114	r	Railway on ground level
115	s	Subway (railway line which lays beneath the ground level, usually without roof (like a ditch))
116	t	Train station (minor, without roof)
119	w	Wall (Party wall of a building)

6.2 Key

Every module has a size of 16x16 studs. If you want to use a bigger module, you need to split it up into separate modules. To ensure that they always connect in the generation, you can specify a key on every edge where the other part of the module must connect. You should use every key exactly two times, on the two edges that must connect. Those edges are never placed on the edge of the whole grid.

A special key is the default key, which is the empty string. It is used on every edge that does not specify a key and there are no restrictions for that key.

Picture 5 show one half of a module containing the empire state building. The edge where the other half must connect specifies it's unique key.

7 About

The software uses `Python 3.10`, `Python TKinter 8.6` and `Python Pillow 8.4.0`. It was developed by Leo Gummersbach, all rights reserved.

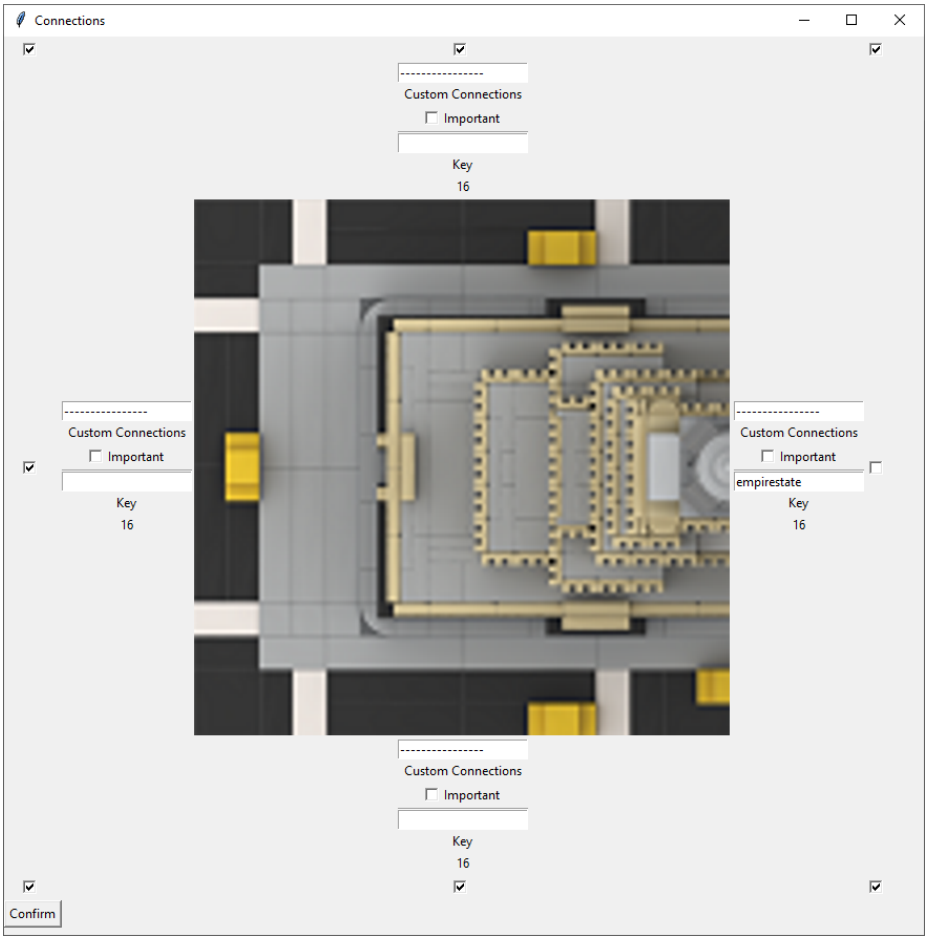

Figure 5: A module with a specified key