

Taxiway Validator

Operation

Description

By Martin Connor ©2020

Version 0.9b

Foreword

If you have ever worked on Airport Scenery for a very large airport, you will know that a lot of effort goes into placing Taxiways Signs. I currently count 1081 at Chicago O'Hare, but there are probably more.

You do your best to make sure they are all accurate, pointing the right way etc. but mistakes happen. Then you hear that your airport has been undergoing major construction and taxiways have been bulldozed, new ones added and worst of all, the taxiway designations have been changed. U is now G2 etc.

It seemed that every time I tried to fly out of my version of KORD, I would find a sign that was incorrect somewhere. Remembering where and going back to correct it was challenging, so I set about working out a way to check them using software.

How?

I started off in Airport Design Editor as I needed a source of data and ADE can generate an XML file with all the data I needed when compiling.

Having read all the sections I needed, it was then a case of finding a way of determining where a taxisign should be relative to a taxiwayPoint.

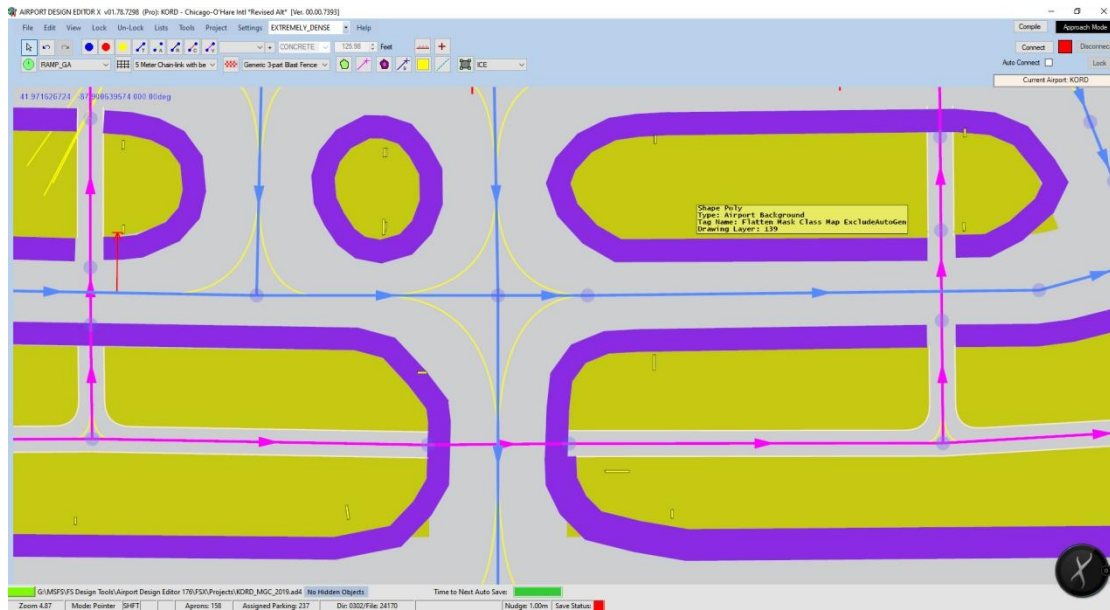
For the uninitiated, I will try and explain some of the structure of FSX.

The underlying structure is based on TaxiwayPoints. These are simple reference points which have the Coordinates of that point, a unique index number and an orientation field.

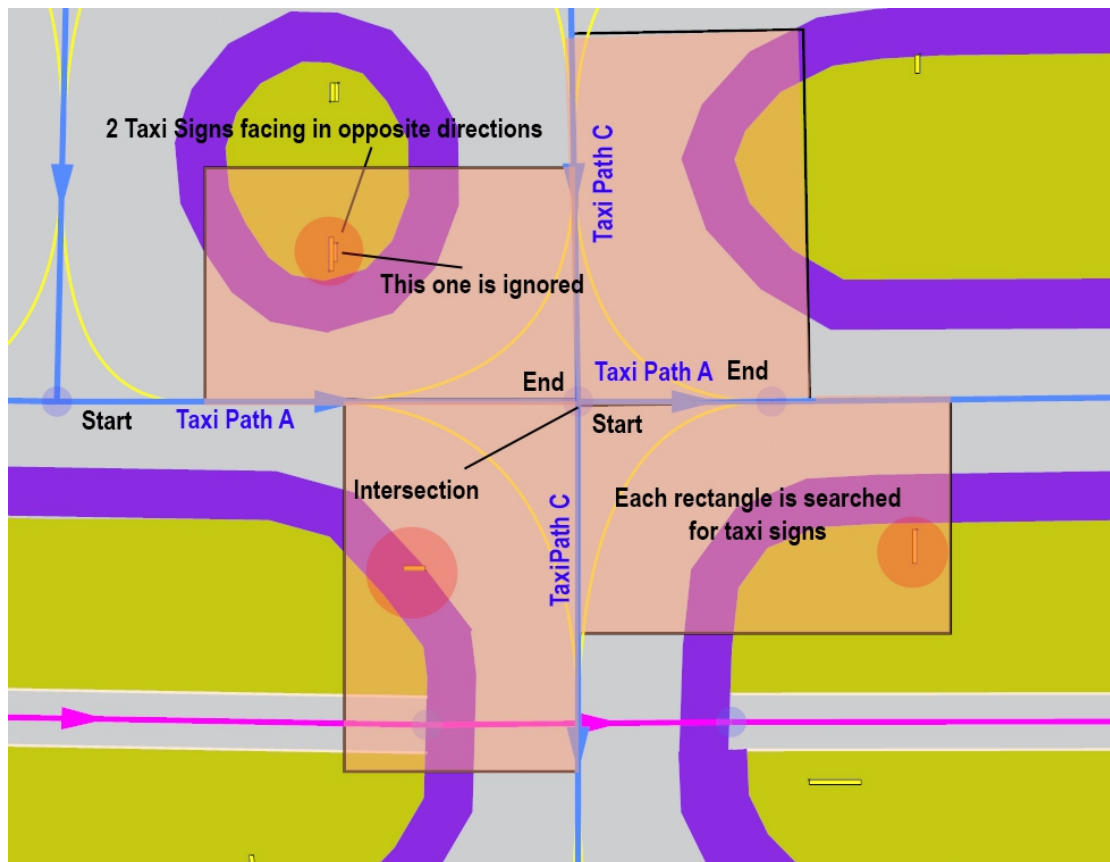
Links between these points are called TaxiwayPaths, and these describe such factors as whether this is a TAXI way VEHICLE, PATH-as in on an apron, or CLOSED. Runways are also TaxiwayPaths, though a separate RUNWAYS section exists to describe them in more detail.

So, TaxiwayPaths join TaxiwayPoints together, so each TWPath has a start index and end index field. They also have a width field, which I may use in a later version to cover the fact that at least in the USA, signs are only required on the left side approaching an intersection unless the taxiway is over 150 ft wide.

Another Section I use is the TaxiName. Rather than store the Taxiway Designators in the TaxiwayPaths, they chose to record them separately. This only entails an index and a name. I.e. 25 and A. I think this may have been an after thought, since the TaxiwayPath field name is **name** when it is actual the name index.



Examining a complex airport, I discovered that the taxisigns are usually around 100 ft from the intersection as drawn on ADE. Although there is a rule for their distance from the edge of the taxiway, I decided that in order to capture them reliably, I would need to draw a virtual box around an area to the right of the taxiway as seen from the start index of the TaxiwayPoint that is an intersection. After many iterations, I decided the easiest solution was to adopt a check as follows:

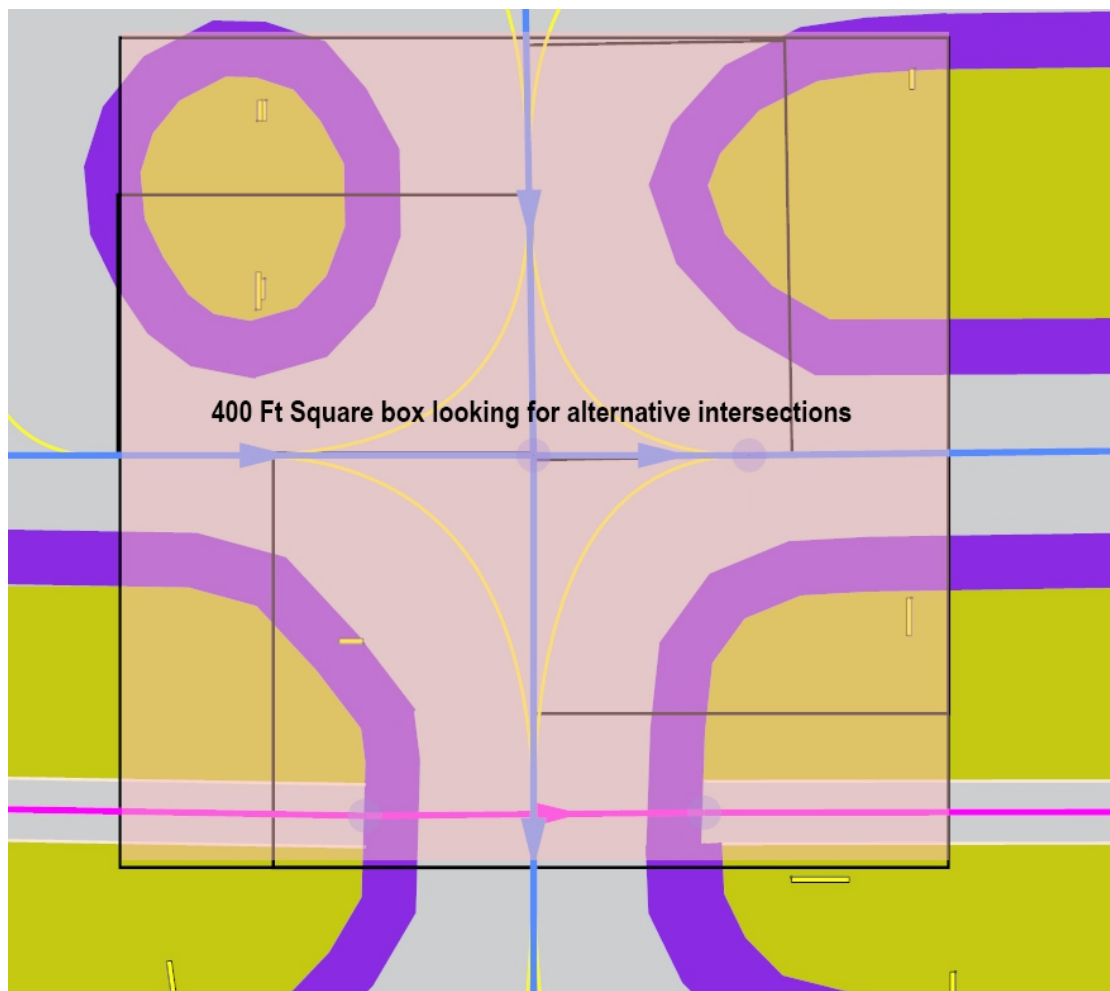


Each TaxiwayPath has a start index and an end index. The path will be drawn from the start using the bearing field. Therefore, following the taxiway A into and out of the intersection, it can be seen that the colored rectangle indicating the search area for each paths taxisign is located in a different position when the end is located at the intersection.

The intersections are found simply by counting the number of paths entering each TaxiwayPoint. If it is more than two, it is listed as an intersection. One can manually click on the list to trigger a lookup of the paths that pass through it, and clicking on each path will conduct a search in the appropriate virtual rectangle to see if any taxisigns are present. If a taxisign contains a designator that is not in the current path list, it is a potential error.

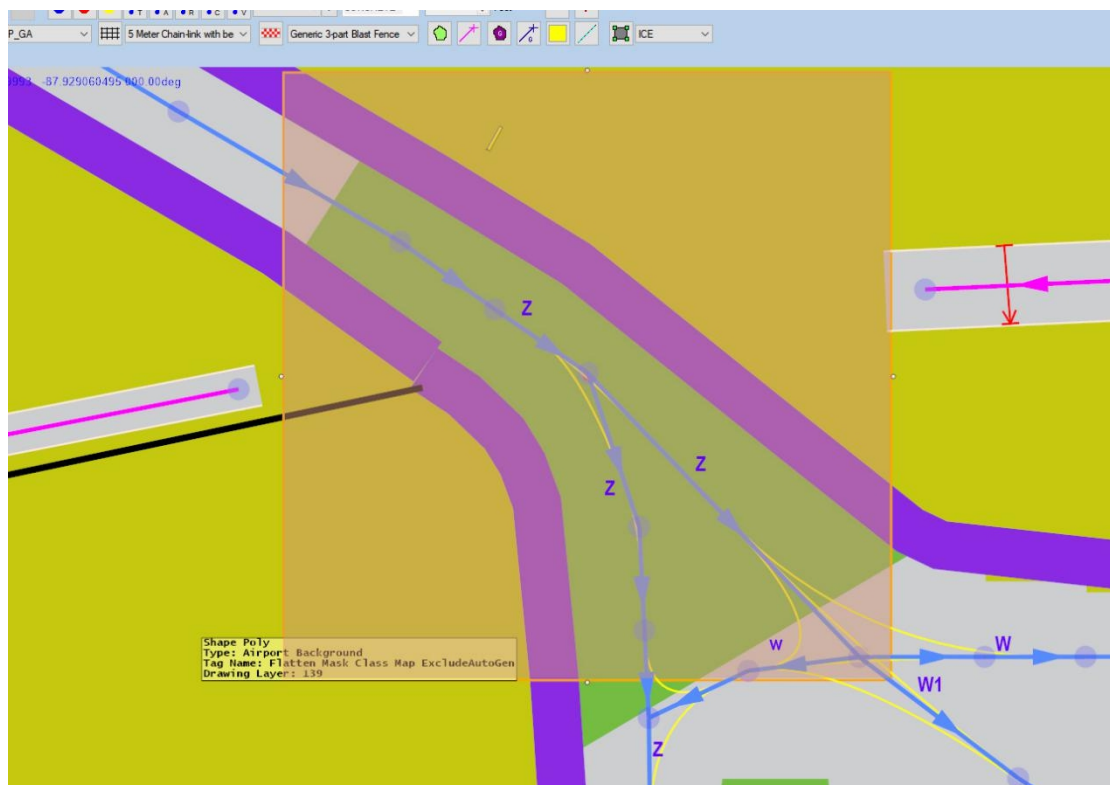
In Auto Mode, the software steps through each taxiway Name, looking for matching paths. Duplicate paths are removed as these are triggered more than once depending on how many paths pass through an intersection. From the diagram, you can see how the rectangles are generated from the intersect point along each path, always to the right of the taxiway and out across the width to a default distance of 125 feet. The user can change this value as well as how far it extended from the intersect point.

The next diagram shows how another box is drawn and searched for intersections.



When taxipaths are fairly close together, the signs will sometimes legitimately indicate taxiways that do not pass through the intersection. These are optionally dealt with by ticking the Look Deeper check box next to the Square Size Box. When this is ticked, a box of that size (default 400ft) is virtually drawn around the intersection, and any taxiway points that are found inside the box are checked for path names. If a path matching the taxi sign designator that that was thought to be incorrect is discovered, that taxisign will no longer be flagged as bad.

The last wrinkle I discovered was that when an intersection has a valid path name that is the same for each path, it normally means it is a split which will eventually end up on a new path, but maybe several links ahead. The default of 400ft for the square search was too small for that case, so you will observe the square sometimes flashing to 500ft during processing as it extends the range whilst looking at such an intersection.



Whilst this diagram has only one taxi sign, it is an example of how the system works to detect potential taxiways. Taxiway Z splits, and a sign will indicate that W goes to the left and possibly W1 as well. Having discovered that the intersection being examined only has Z passing through it, the Square Search for other taxiways will expand to 500ft (diagram is actual 450) and it will pick up two more intersections. One is between Z,W and W1 and the other just W and Z. These new Taxiway Names will be compared with the taxi sign. Provided the taxisign contains only Z,W or W1, it will no longer be flagged as bad.