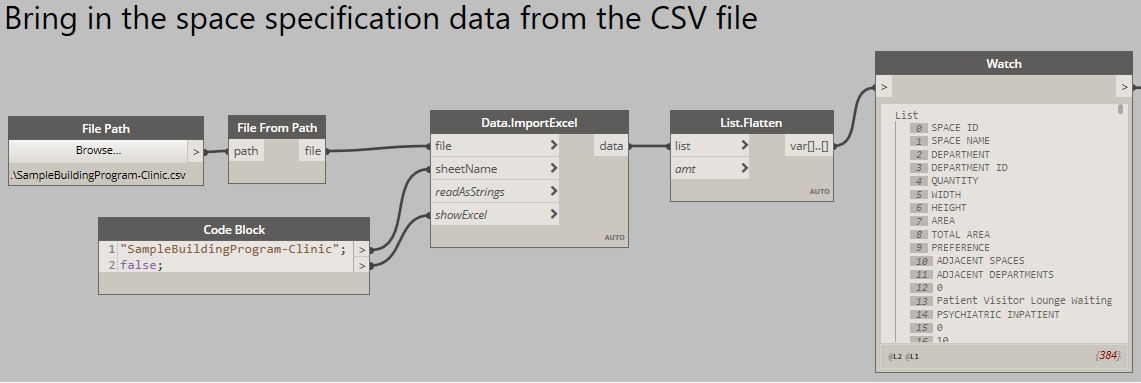
# “Generative bubble diagram” with DynaSpace and Refinery

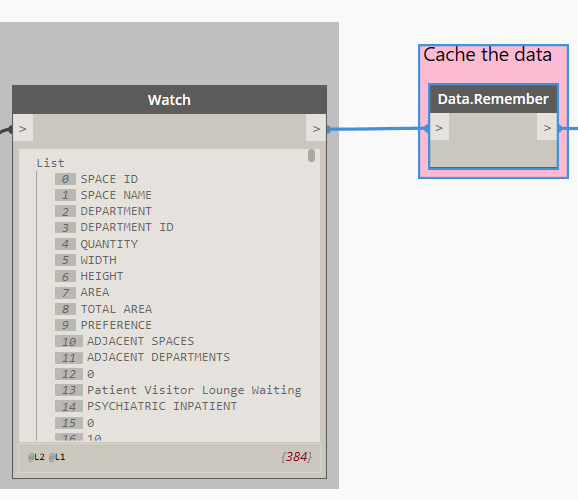
This is the first attempt at using DynaSpace with Refinery. We will use the “DynaSpace GDIR Example.dyn” example script to demonstrate the workflow.

## Basic workflow

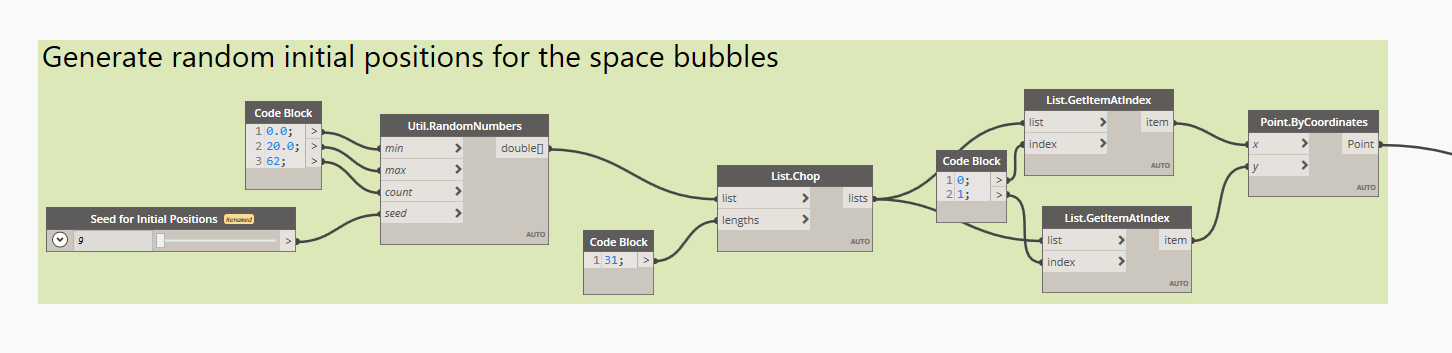
1. Bring in the space specification from the CSV file (The “SampleBuildingProgram-Clinic.csv” file is included in the “extra” folder of DynaShape package)



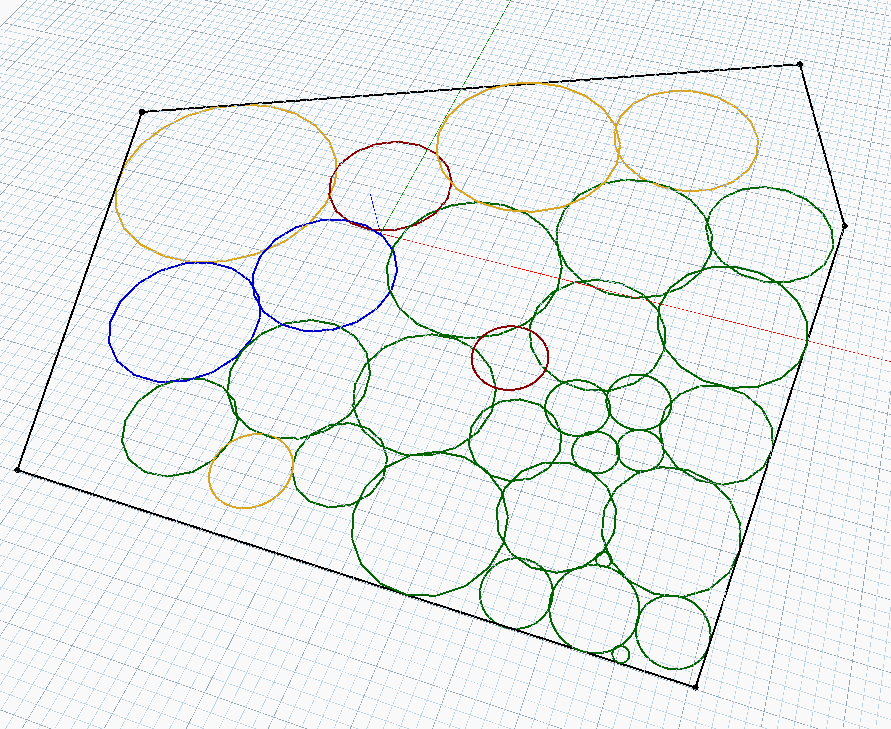
1. Cache the space specification data inside a Data.Remember node (as Refinery will not be able to directly read data from the .csv file)



1. Define the boundary polygon, which is just a list of points that represent the vertices of the polygon (Note: The polygon must be convex. Non-convex polygons will be supported in the future)
2. Generate random initial positions for the centers of the space bubbles. It is generally a good idea to keep these initial random locations within the bounding box of the side boundary



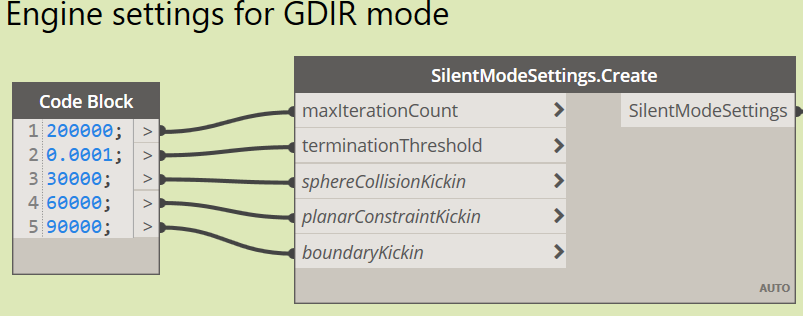
1. Try changing the random seed to see if you get some reasonable-looking outcome



1. Now you can Export the entire graph for GDIR and let GDIR vary the Random Seed for Initial Positions to generate different space arrangement results

## Further tweaking (optional)

Depending on your setup, you might want to consider tweaking the following parameters of the *SilentModeSettings.Create* node:



* The “max iteration” is an upper-limit that prevents the DynaSpace solver from running for too many iterations before it settles for a solution (Running too many iterations will take a long time and might apparently “freeze-up” Dynamo and GDIR). Generally you can set it to a relatively large number and forget about it. But if your setup has many more space bubbles, you might need to increases the “max iteration”, otherwise the generated solution may not be fully completed . As a rule of thumb, if you double the number of space bubbles, you should increases “max iteration” by 4 times.
* The “kickin” parameters. To achieve better result, DynaSpace actually does not enable all constraints right from the beginning (iteration 0), rather it enables the constraints one-by-one in specific order. The Kickin parameters allow the user the fine-tune the time (the iteration) at which a specific constraint will start to become active. We have found that an interval of 10000-30000 iterations between the kick-ins are sufficient.