Matrix: closeness → Add Agents preferred size and shape to data → Initalize agent seed location randomly → Extract agents locations → Find the shortest connection in the grid between the agent voxels and the prefered neighbour voxels → Normalize all values per voxel → Closeness value

Matrix: entrance accessibility → Initalize agent seed location randomly → Extract agents locations → Find the shortest connection in the grid between the agent voxels and the prefered entrance voxelz → Normalize all values per voxel → Entrance accessibility value

Matrix: noise sensitivity, silence level → Convert the silence level activity values from the matrix to Decibel values → Initalize agent seed location randomly → Extract agents locations → Lattice from each agent location and noise level → Extract the coordinates of the centroids of all voxels → Calculate euclidian distance → Occupation distance lattice from the voxels → Compute the noise lattice from the distance lattice → Calculate the summed noise level → Noise lattice → Change the values from the noise lattice back to values with a weight between 0 and 1 (1 being the most quiet and 0 being the loudest) → Silence lattice → Dictionary with the noise sensitivity data from the matrix and the silence lattice → Find the shortest connection in the grid between the agent voxel and the prefered neighbour voxels based on their silence property → Normalize all values per voxel → Internal noise value

Matrix: floor preference → Initalize floor lattice → Cut out a column → Specify a range array based on the number of voxels in the column → Compute distances based on the range column → Floor level lattice → Define ground floor and first floor in lattice → Derive data ground floor and first floor from the matrix and put it into a dictionary → Initalize agent seed location randomly → Extract agents locations → Find the shortest connection in the grid between the agent voxels and the prefered floor preference → Normalize all values per voxel → Floor preference value

Matrix: sun access → Put data from sun access into dictionary → Initalize agent seed location randomly → Extract agents locations → Find the shortest connection in the grid between the agent voxels and the prefered sun access → Normalize all values per voxel → Sun access value

Matrix: sky view → Put data from sky view into dictionary → Initalize agent seed location randomly → Extract agents locations → Find the shortest connection in the grid between the agent voxels and the prefered sky view → Normalize all values per voxel → Sky view value

Matrix: noise sensitivity → Put data from external noise access into dictionary → Initalize agent seed location randomly → Extract agents locations → Find the shortest connection in the grid between the agent voxels and the prefered external noise → Normalize all values per voxel → External noise value

Multiply the weights of all functions with their corresponding voxel value → Suitability list of agent_id each per voxel → Choose the best neighbouring voxel by each function type → If occupied ignore voxel → Let the agents grow till they have reached their maximum area → Spatial configuration