**Sampler**

This component automatically generates a list of parametric design vectors, called a “design map”, based on user-defined design variable properties. It outputs this design map as a nested list and Grasshopper, which can be used directly or plugged into other DSE tools, and saves it as a .csv file for documentation and interfacing with outside software. Sampler works on double-click, which will both output a Design Map and write a new file to your directory.

**Inputs:**

**Variables (Var) –** Takes in any number of sliders that are used as design variables for a given project. Sampler automatically reads the bounds of the sliders to set the limits of the design space being explored.

**Number of Samples (N) –** The number of samples to be generated. In the ‘grid’ design mode, rather than automatically truncating the list of samples at the edges or adjusting their spacing, the Design Map contains a list of samples that fully covers the design space in each dimension. This results in a number of samples that is higher than the input N, but is divisible by **Vars** . The user can manually trip this Design Map as needed.

**Type (Type) –** Sets the sampling technique: random, grid, or Latin Hypercube(citation).

**Seed (S)** – Allows the user to refer back to a previously generated Design Map, even if random as involved. If the seed is set to ‘0’, Random or LHC will generate a new Design Map each time it is clicked. Every other integer will reference back to the same Design Map, even if it is randomly generated.

**Filename (F)** – The name of the Design Map file that will be written as a .csv. Does not include extension.

**Directory (Dir) –** The location on your computer where the Design Map will be saved. The directory MUST end with a “\” – otherwise, the file will be written one level higher than the intended directory.

**Outputs:**

**Design Map (DM) –** A nested list of design vectors that span the design space being explored, based on the selected sampling technique.

**Capture**

This component is a general iterator, which allows the user to automatically generate many different design options and record an image, the performance, and/or other properties of each design. Capture works on double-click, which will cause the component to cycle through each design option and recording the results.

**Inputs:**

**Variables (Var) –** Takes in any number of sliders that are used as design variables for a given project. Sampler automatically reads the bounds of the sliders to set the limits of the design space being explored. The sliders should be identical to those used to generate the Design Map.

**Objectives (Obj) –** Reads in a list of the numerical values from performance evaluations generated by the script. Any performance-based measurement (or objective function), generated from either an outside plug-in or a collection of native Grasshopper components, can be recorded. Depending on Mode, Capture can function with either zero, one, or multiple objective functions in this input.

**Design Map (DM) –** A nested list of design vectors to be evaluated, coming either directly from Sampler or from another previously generated list of design options.

**Mode (M) –** Sets the operating mode: screenshots, eval, or both. Screenshot mode records the image of the current Rhino view and saves it as a .png in the directory.

**Properties (P)** – Similar to objectives, reads in a list of numerical secondary design properties to be recorded while Capture is cycling through the different design options. However, these properties are not necessarily objective function evaluations and will be kept separate from the objectives when both are output by Capture.

**.CSV Filename (.csv F)** – The prefix name of the Design Map + Objectives and Properties files that will be written as a .csv to the directory.

**.CSV Directory (.csv Dir) –** The location on your computer where the Design Map + Objectives and Properties files will be saved. The directory MUST end with a “\” – otherwise, the file will be written one level higher than the intended directory.

**Screenshot Filename (SS F)** – The prefix name of the screenshot files that will be saved as a .png in the directory.

**Screenshot Directory (SS Dir) –** The location on your computer where the screenshots will be saved. The directory MUST end with a “\” – otherwise, the file will be written one level higher than the intended directory.

**Outputs:**

**Design Map + Objectives (DM+O) –** A nested list of design vectors and their objective values that span the design space being explored. Standard format for each design is the design vector followed by as many objective values as are recorded.

**Properties (Props) -** A nested list of the secondary design properties recorded while Capture is cycling.

**Reader**

**Inputs:**

**Directory + Filename (Dir + F) -** Takes in the full address on your computer where the desired file is located, including the extension. The format takes in each row and converts it to a nested list, following the standard format for Design Maps. However, this component could read in any file saved in a similar row/column structure, including .txt files.

**Separator (S) –** Sets the character used to separate the data in the .csv. By default it is a space, which is how Design Maps are written, but for other data it could be “,”.

**Outputs:**

**Data (D) -** A nested list of the data contained in the .csv, following the standard format of design variables followed by objective values in the case of DM+O files.

**Writer**

**Inputs:**

**Directory + Filename (Dir + F) -** Takes in the full address on your computer where the file should be written (no extension is necessary).

**Data (D) –** Takes in a nested list of the data that will be converted to a row x column .csv file.

**MOO**

This component implements the NSGA-II multi-objective optimization algorithm (CITATION), which is a non-dominated sorting genetic algorithm. NSGA-II approximates the Pareto front in a given design problem by using crossover and mutation to iteratively breed successive, higher-performing generations of designs. MOO works on double-click, which will implement the algorithm. When the last generation has been reached, a dialog box will inform the user that the process is finished, at which point MOO will return the results as nested lists and write them to a .csv file in the directory. MOO uses the jMetal Framework to implement NSGA-II.

**Inputs:**

**Variables (Vars) –** Takes in any number of sliders that are used as design variables for the optimization. MOO automatically reads the bounds of the sliders to set the limits of the design space being explored during the optimization.

**Objectives (Obj) –** Reads in a list of the numerical values from performance evaluations generated by the script. Any performance-based measurement (or objective function), generated from either an outside plug-in or a collection of native Grasshopper components, can be recorded. Although MOO will still run an optimization with only one objective, it is designed to handle at least two. **Note**: the objectives must be flattened in order for MOO to run properly.

**Population (Pop) –** Sets the population size for the for the multi-objective optimization algorithm. The number of generations tested while the algorithm is running will be the maximum evaluations divided by the population. **Note:** due to the way in which the NSGA-II is implemented, the population size *must* be even, and it should be a factor of the Maximum Evaluations. MOO will still run if the population is not a factor of MaxEval, but it will not run if the population is even.

**Maximum Evaluations (MaxEvals)** – Sets the amount of function evaluations to be conducted during the optimization. Since there is no “threshold” mode for this version of MOO, the MaxEvals effectively sets the stopping condition for the optimization—if the results do not satisfactorily represent the Pareto front, more evaluations, and consequently more generations, should be attempted. MaxEvals should be a multiple of the Population size. If your objective function evaluations take a long time, that duration multiplied by the number of evaluations should give a rough indication of the time it takes to run the whole optimization.

**Filename (F)** – The name of files that will be written to the directory. A successful MOO run will write two files: “LogFile-” + F and “allSolutions” + F. The log file will list details about the optimization for each time the component has been run, but the allSolutions file will reset with the evaluations of only the most recent run. **Note:** outputs may be written with different extensions; as such, “.csv” must be included at the end of the Filename if that is the desired file type.

**Directory (Dir) –** The location on your computer where the Design Map will be saved. The directory MUST end with a “\” – otherwise, the file will be written one level higher than the intended directory.

**Outputs:**

**Pareto (Pareto) –** A nested list of design vectors and objective function values of the final generation produced by MOO. These solutions are the approximation of the Pareto front for a given problem. The quality of the Pareto front depends on the each of the inputs, as well as the problem itself, so some experimentation might be helpful.

**All Solutions (All Solutions) –** A nested list of design vectors and objective function values for all recorded solutions produced while the algorithm is running. The length of this list will match the MaxEvals input.