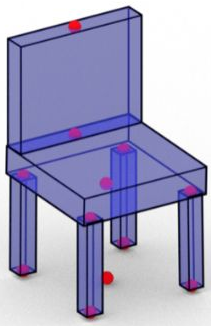
For my capstone, I contributed to the project, ‘ShapeAssembly: Learning to Generate 3D Shape Programs for Hybrid Neural-Procedural Structure Synthesis’ in Prof. Daniel Ritchie’s research group. I helped with the development of the domain specific language for 3D shape representation and the implementation of the language interpreter.

Paper Abstract:

‘Manually authoring 3D shapes is difficult and time consuming. Generative models of 3D shapes offer compelling alternatives. Procedural representations are one such possibility: they offer high-quality and editable results but are difficult to author and often result in limited diversity among the output shapes. On the other extreme are deep generative models: given enough data, they can learn to generate any class of shapes but their out-puts have artifacts and the representation is not editable. In this paper, we take a step towards achieving the best of both worlds by proposinghy-brid neural-procedural shape synthesis. First, we propose ShapeAssembly, a domain-specific low-level “assembly-language” for 3D shape structures. ShapeAssembly programs construct shape structures by declaring cuboid part proxies and attaching them to one another, in a hierarchical fashion.They additionally have continuous free parameters that allow each program to capture a family of related shapes. The ShapeAssembly interpreter is fully differentiable, allowing optimization of program parameters with respect to output geometry. We show how to extract ShapeAssembly programs from existing shape structures in the PartNet dataset. Then, we learn a deep generative model that outputs ShapeAssembly programs. This approach leverages the strengths of each representation: the program captures the subset of shape variability that is easily interpretable and editable, and the deep generative model captures variability and correlations across shape collections that cannot be easily expressed procedurally. We evaluate our approach by assessing the quality of our generated programs in terms of the plausibility, diversity, complexity, and physical validity of the shapes they output. Compared to other recent shape structure generative models, ours is particularly good at outputting physically valid shapes (i.e., connected,stable), even under perturbations of program parameters. We also show how to use our generative model and differentiable program interpreter to infer and fit shape programs to unstructured geometry, such as point clouds.’

ShapeAssembly program example:

```
Assembly Program_0 {
    bbox = Cuboid(0.87, 1.5, 0.86)
    Program_1 = Cuboid(0.82, 0.61, 0.83)
    Program_2 = Cuboid(0.85, 0.7, 0.14)
    cube2 = Cuboid(0.86, 0.18, 0.86)
    attach(Program_1, bot, 0.5, 0.5, bbox, bot, 0.5, 0.5)
    attach(Program_2, top, 0.5, 0.5, bbox, top, 0.5, 0.91)
    attach(cube2, bot, 0.5, 0.5, Program_1, top, 0.5, 0.51)
    attach(cube2, top, 0.5, 0.91, Program_2, bot, 0.5, 0.52)
}
Assembly Program_1 {
    bbox = Cuboid(0.82, 0.61, 0.83)
    cube0 = Cuboid(0.11, 0.6, 0.11)
    cube1 = Cuboid(0.11, 0.6, 0.1)
    cube2 = Cuboid(0.11, 0.61, 0.1)
    cube3 = Cuboid(0.1, 0.61, 0.1)
    attach(cube0, top, 0.5, 0.5, bbox, top, 0.07, 0.94)
    attach(cube0, bot, 0.5, 0.51, bbox, bot, 0.06, 0.07)
    attach(cube1, top, 0.5, 0.5, bbox, top, 0.07, 0.06)
    attach(cube1, bot, 0.5, 0.5, bbox, bot, 0.07, 0.94)
    attach(cube2, top, 0.5, 0.5, bbox, top, 0.93, 0.94)
    attach(cube2, bot, 0.5, 0.5, bbox, bot, 0.93, 0.06)
    attach(cube3, top, 0.5, 0.5, bbox, top, 0.93, 0.06)
    attach(cube3, bot, 0.5, 0.5, bbox, bot, 0.94, 0.94)
}
```



```
Assembly Program_2 {
    bbox = Cuboid(0.85, 0.7, 0.13)
    cube0 = Cuboid(0.83, 0.09, 0.1)
    cube1 = Cuboid(0.13, 0.62, 0.11)
    cube2 = Cuboid(0.16, 0.64, 0.13)
    cube3 = Cuboid(0.15, 0.64, 0.12)
    cube4 = Cuboid(0.13, 0.63, 0.12)
    attach(cube0, top, 0.5, 0.5, bbox, top, 0.51, 0.53)
    attach(cube1, bot, 0.49, 0.5, bbox, bot, 0.08, 0.48)
    attach(cube2, bot, 0.5, 0.5, bbox, bot, 0.36, 0.5)
    attach(cube3, bot, 0.51, 0.5, bbox, bot, 0.65, 0.49)
    attach(cube4, bot, 0.5, 0.49, bbox, bot, 0.93, 0.52)
    attach(cube1, top, 0.49, 0.51, cube0, bot, 0.08, 0.49)
    attach(cube2, top, 0.49, 0.5, cube0, bot, 0.35, 0.5)
    attach(cube3, top, 0.5, 0.52, cube0, bot, 0.61, 0.49)
    attach(cube4, top, 0.49, 0.49, cube0, bot, 0.91, 0.48)
}
```