Simulation Generator: Procedural Generation

Developed by: Jacob Coffland, Mina Shin, and Eli Jukanovich

Under mentorship of Dr. Wesley Deneke

Western Washington University – Computer Science Department

Abstract

Collecting data is an enormous undertaking. To ameliorate this workload for those looking to collect data about human workflows, this project aims to synthesize human workflow data using a procedurally generated environment. In order to achieve the procedurally generated environment, a grid system was created. This was then populated by a rules engine and content generation engine working in tandem. The population, as well as global aspects of the environment, (such as the home model) can be determined by means of an authoring tool, which a range of different parameters from the user.

The current project features a model kitchen in which cabinets and select appliances are placed procedurally generated, with each executing producing a different layout, statistically speaking. the grid-based system was found to be near essential for efficient object population, as well as enabling the app to be scalable and extensible should another team want to build upon the current work.

Background

Procedural content generation (PCG) is widely used in computer graphics and game design to automatically generate environmental content. The key benefit of PCG is that it can create diverse content in a fraction of the time of a manual approach, allowing developers to focus on higher-level aspects of a project.

Motivation

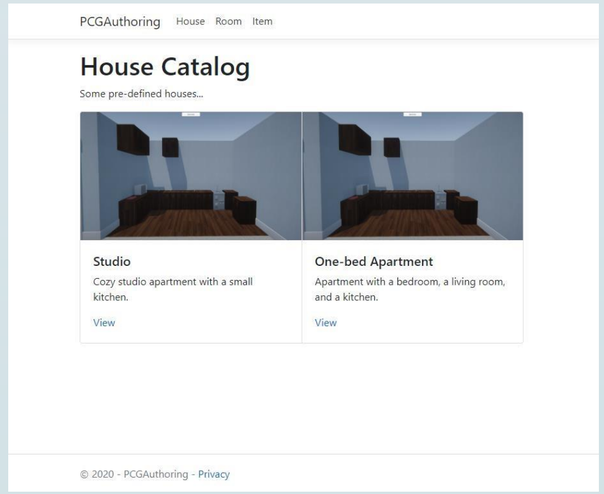
Data collection is one of the most difficult, time consuming, and error prone components in training machine learning algorithms. Studies have shown that training models using synthesized data can improve model performance. The Procedural Content Generator is the first step in automating the generation of massive amounts of well-annotated synthetic data.

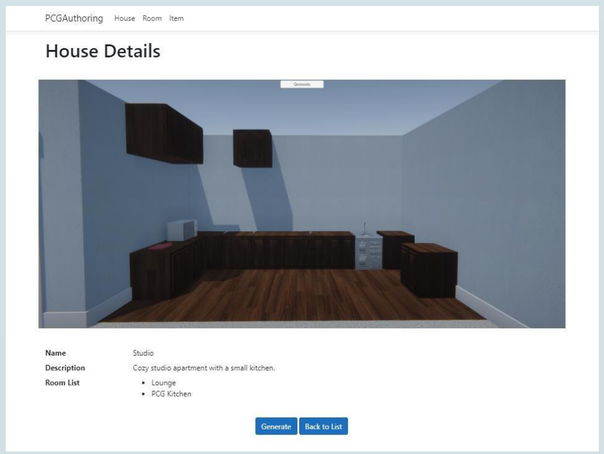
Design Criteria

- Create an interface for users to provide their own specifications for generation
- Procedurally generate a virtual environment that conforms to user-supplied parameters
- Ensure that a generated environment could reasonably be considered a realistic space
- Provide the user with data from the virtual environment

The Authoring Tool

The authoring tool enables users to select a home model and provide parameters that influence content generation. Specifications are sent to the content generator running in the Unity engine, which creates virtual environments until one is found to match the specified requirements. Information regarding that environment is then sent back to the user to view in the authoring tool.





How It Works

First, a grid is created to track individual areas of the virtual environment. Different sections of the grid are then reserved for specific objects. Next, the grid is analyzed to ensure that object placement will conform to both a set of internal rules and user-provided parameters. Finally, each section of the grid is populated.

Content Generation

In the model kitchen, you can see a variety of objects and appliances generated; between five and ten cabinets are placed along the walls before a random number of upper cabinets are placed directly above them. A sink is then generated, replacing a single non-corner cabinet. Between zero and one microwaves are added to the kitchen on a random countertop. Finally, cutting boards and knives are placed.

Summary

We found that using a grid-based system to ensure a finite set of position vectors was crucial for allowing a wide variety of permutations while simultaneously limiting execution costs. The generation of objects in the environment must be done in "layers" so that objects can reference the constantly changing environment; for example, cabinets are generated before appliances such as the kitchen sink, so that the sink may be inserted by replacing an existing cabinet. Similarly, items such as the cutting board are generated near the end so that they can be placed on top of generated countertops.

Future Work

- Evaluate the runtime performance and content diversity
- Generate the entire housing layout, including wall placement
- Integrate with synthetic video generation
- Generate items within cabinets
- Enhance with an ontological model to generalize content classifications

Acknowledgements

A special thanks to our faculty advisor Dr. Wesley Deneke, without whom this project would not have been possible, as well as Gary Plunket who lead a similar project previously.