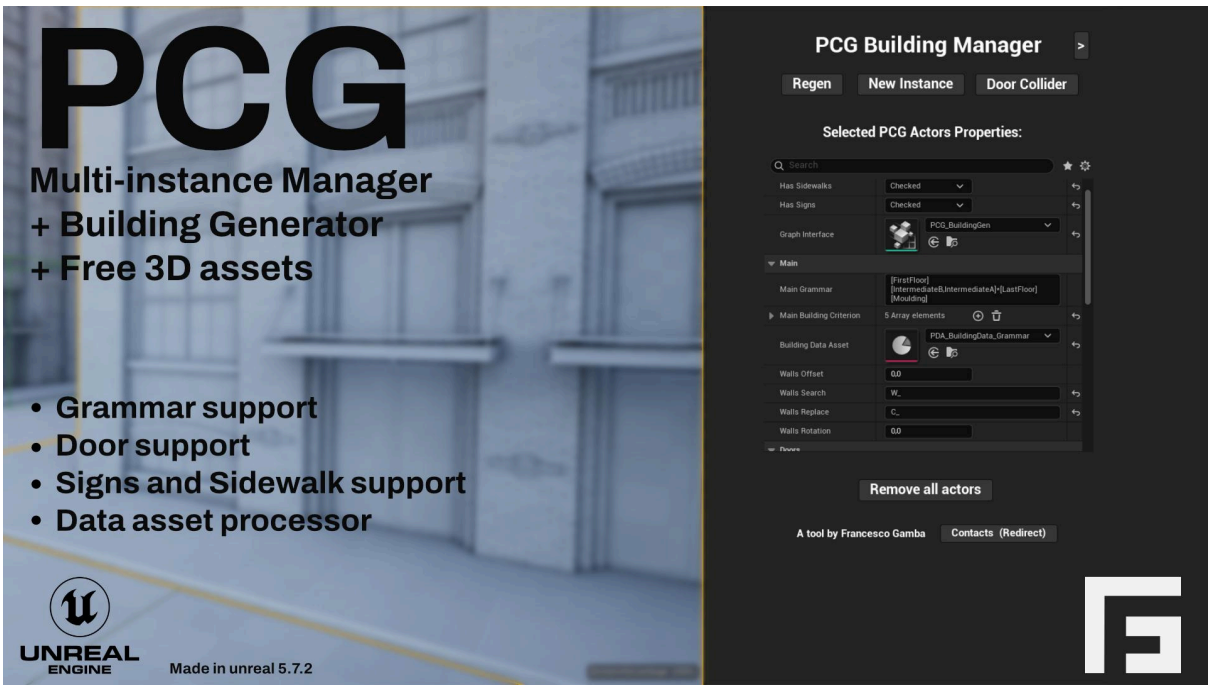
Documentation

Multi Instance PCG Manager + Data Asset Processor + Building Generator



A tool by Francesco Gamba
Developed in Unreal Engine 5.7.2

Contacts:

Francesco060501@gmail.com
ArtStation - Francesco Gamba
LinkedIn - Francesco Gamba

Introduction

Multi instance PCG manager is a tool designed to simplify the level creation & set dressing pipeline leveraging the Procedural Content Generation framework and Unreal Engine Blueprint system.

The tool allows for quick editing of each PCG instance by changing the actor’s containing the instance properties from a simple UI Widget. The tool allows editing multiple instances all at once while also offering some extra features that we will cover later. The tool also includes a Data Asset Processor functionality that can be useful for quickly setting up data assets.

Quick Setup

1. Download the file or clone the repository from https://github.com/Francesco-Gamba/Multi-instance_PCG_Manager

2. Download the Unreal Engine project and migrate the PCG_Manager folder to your project. Migrating the Assets folder is recommended for better understanding of the tool as it contains example meshes but can be avoided.

3. Navigate to /PCG_Manager/WidgetUI and right-click on W_PCG_BuildingManager and press on 'Run Editor Utility Widget'. The widget should appear on your editor screen.

4. The widget uses as defaults the Data Assets contained in /PCG_Manager/DataAssets you edit these Data Assets or you can create your own from scratch using the predefined PDA_ blueprint classes contained in /PCG_Manager/DataAssets/Blueprints.

5. From the main widget page you can spawn a new PCG instance by clicking on the 'New Instance' button. You can trigger the regeneration of the PCG graph by clicking on the 'Regen' button. You can spawn a door trigger collider by clicking the 'Door Collider' Button. To remove ALL instances click on the 'Remove all actors' button (THIS IS NON REVERSIBLE).

6. After generating an instance and clicking on it you can see all the actor (PCG Instance) properties. You can easily change the properties from that panel.

7. When two or more instances are selected the values that match between the instances are kept while the ones that differ will be assigned to null or have custom string appearing, eg.:'Grammars do not match'.

8. When making edits to one or more properties with one or more instances selected all instances will be assigned that property value, effectively overriding the old property value.

Interface parameters

PCG Settings

- **Has Sidewalk:** toggle sidewalks.
- **Has Signs:** toggle sidewalks.
- **Graph Interface:** PCG graph used for generation. You can use different graphs for varying generations.

Main

- **Main Grammar:** The grammar string used for generation.
- **Main Building Criterion:** The grammar module criterion used for generation.
- **Building Data Asset:** The data asset used for generating the building.
- **Walls Offset:** Offset value used to ‘push out’ the walls from actual position. Useful for fixing gaps on corners.
- **Walls Search:** Used to search for walls that can be swapped for corner elements.
- **Walls Replace:** Used to actually replace the walls with corners.
- **Walls Rotation:** Rotation value to fix possible issues.

Doors

- **Door Search:** Used to search for specific modules that can be swapped.
- **Door Replace:** Used to actually replace the searched modules with doors modules.
- **Door Accessory:** Addition module added in the same position as the door.

Sidewalk

- **Sidewalk Grammar:** Main grammar string used for generation.
- **Sidewalk Data Asset:** Data asset used for generation.
- **Sidewalk Search**: Used to search for sidewalks that can be swapped for corner elements.
- **Sidewalk Replace:**Used to actually replace the sidewalks with corners.
- **Sidewalk Offset:** Offset value used to ‘push out’ the modules from actual position. Useful for fixing gaps on corners.
- **Sidewalk Rotation:** Rotation value to fix possible issues.

Signs

- **Signs Data Asset:** Data asset used for generation.
- **Signs Offset:** Offset distance from building corner.

Advanced

- **Building Primitive Default Scale:** Defines the default scale of the building primitive used for generation.
- **Door Trigger Scale:** Define the door collider scale used for intersecting modules that need to be swapped with doors.
- **Ray Cast Distance:** Max distance the ray can travel in the camera direction before colliding with an object to define the point for creating a new building instance. Useful to lower to prevent accidental generation out of bounds.
- **Toggle UI sounds:** Enable/Disable UI feedback sound.

Extract Symbol: Toggle whether or not the Data Asset Processor should extract the grammar symbol from the asset name.

Split Char: Defines which character is used to split the symbol from the actual asset name if any.

eg: “NiceAsset-Wall_A” → split char = ‘-’ → Symbol Extracted = ‘Wall_A’

Data Asset Processor

- PDA Building Grammar: Data Asset of Type PDA_Building_Grammar
- PDA Module Rand: Data Asset of Type PDA_Module_RandTransf
- PDA Module: Data Asset of Type PDA_Module

Data Asset Processor & Data asset

The **Data Asset Processor** allows processing of the sub-modules in the data asset with one click targeting the selected data assets in the **Data Asset Processor** category.

The actions performed are:

- Extracts the grammar symbol (if toggled) using a split char.
  eg: "NiceAsset-Wall_A" → split char = '-' → Symbol Extracted = 'Wall_A'
- Automatically detects the mesh size and assigns its values to the size property.
- Generates a random debug color for the submodule.

This is done to preserve the actual PCG grammar workflow without breaking the default structure.

Data Assets are described in the blueprint classes in the /PCG_Manager/DataAssets/Blueprints folder. These blueprint classes are simple in composition and they are tailored to the following specific uses:

- **PDA_Module_RandTransf:** Used for sub-modules that may have some random transform values like position, rotation and scale. (eg: Signs and scattered trash)
- **PDA_Module:** Used for sub-modules that do not require any additional parameters. (eg: Sidewalks)
- **PDA_Building_Grammar:** The most complete, used to define the actual building grammar, modules and sub-modules.

All these data asset classes rely on structs, some of them are default engine types, others are stored in /PCG_Manager/DataAssets/Structs.

Most of them are very basic but let's dive into the custom type **S_Module_Grammar**.

It is composed by:

- **Symbol:** The symbol used by grammar. eg: Wall_A
- **Mesh:** The actual mesh used.
- **Size:** The size of the mesh used.
- **CPO:** Custom Pivot Offset transform that can be applied to the modules to override the pivot point position after grammar generation.
- **UseCPO:** Boolean value that evaluates whether or not to use the CPO.
- **Scalable:** Grammar requirement, defines if the mesh is scalable or not to fill the grammar segment.
- **DebugColor:** Random color used for debugging shapes when generating. A grammar system requirement.