

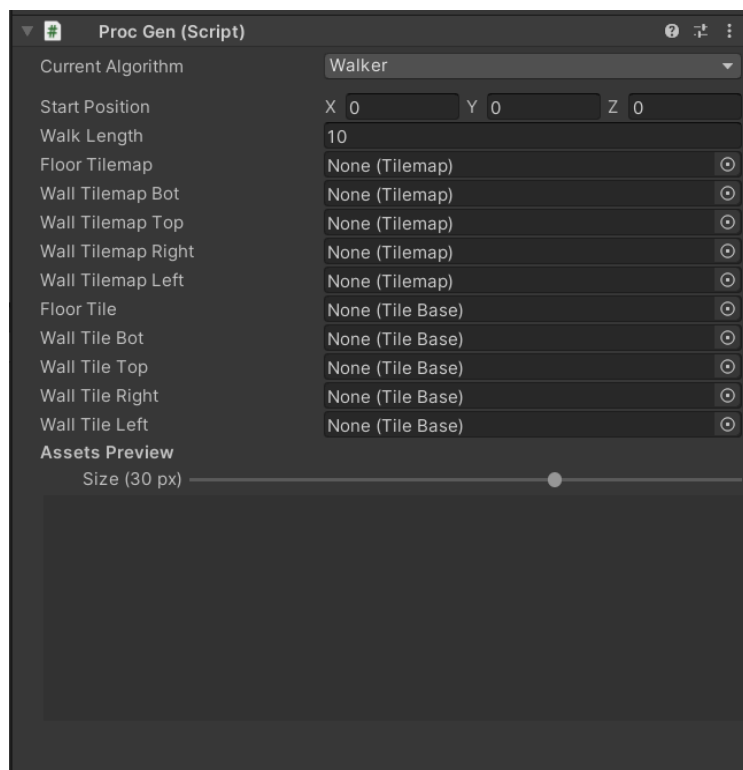
Procedural Generation Toolkit

1.0 Introduction

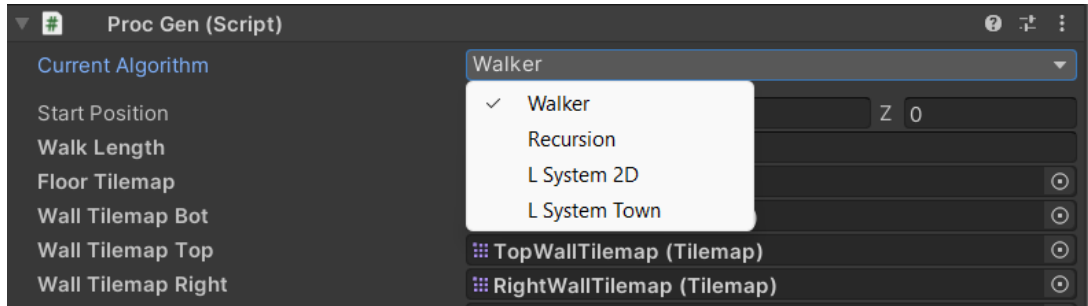
This toolkit provides a collection of a couple well-known procedural algorithms which have been fully implemented.

2.0 Getting Started

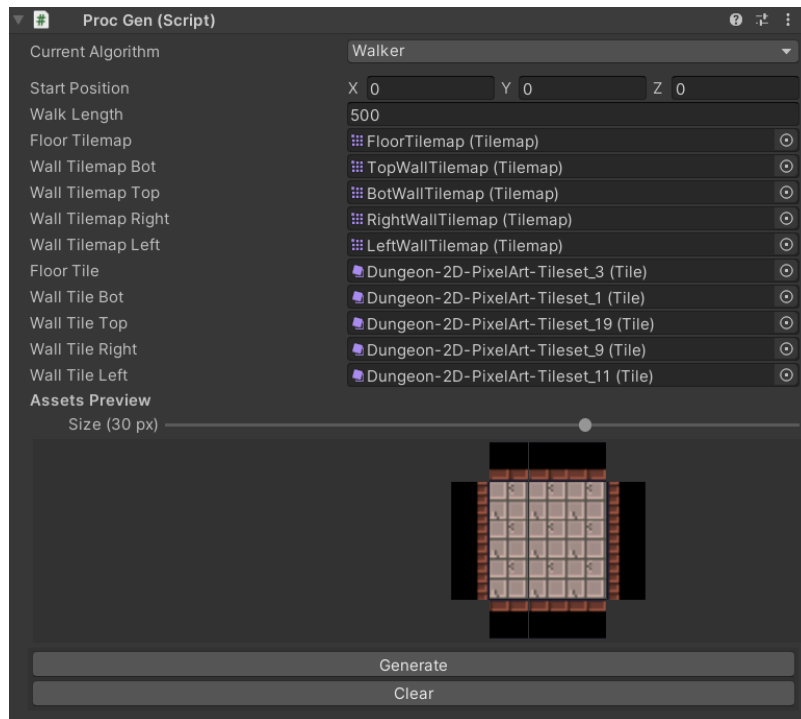
1. Add the ProcGen script in the `Scripts` directory onto a GameObject or use the ProceduralGenerator prefab in the `Prefabs` directory.
2. You will see the following in the inspector:



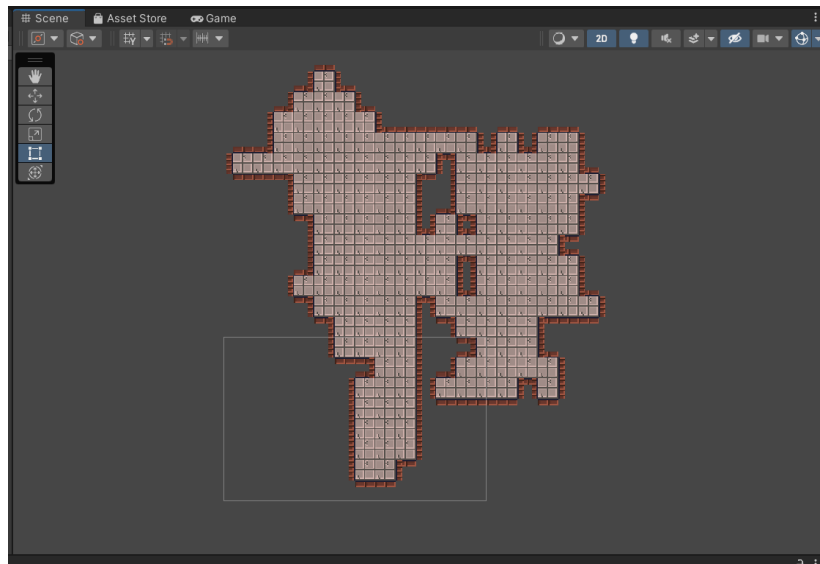
3. Select the procedural algorithm you wish to use with the drop-down menu:



4. Populate the various fields and options for the selected algorithm (Example below using the Walker algorithm):



5. Click the `Generate` button and a map will be generated on the scene using the selected algorithm:



2.1 Algorithms

This section will briefly explain what the provided algorithms do and what the parameters mean.

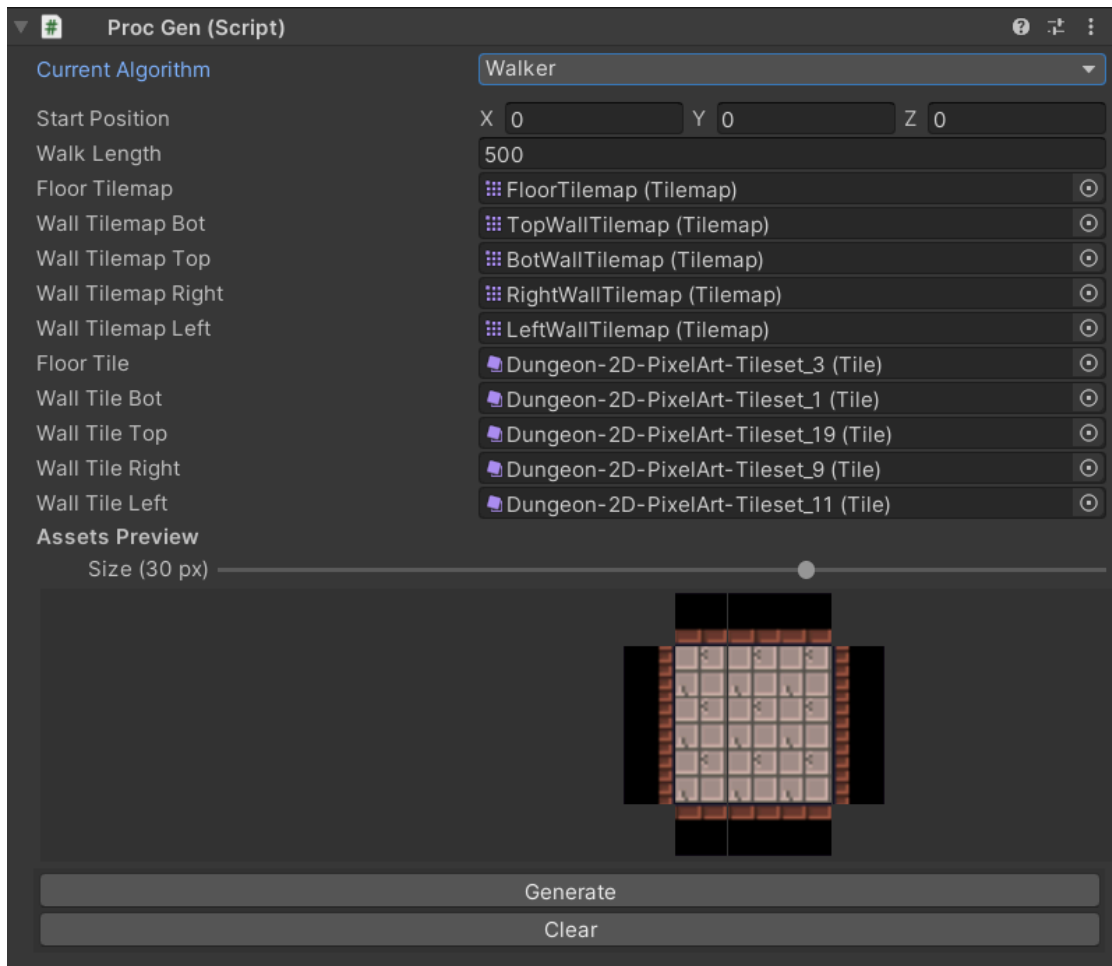
Walker

Algorithm Explained

[Random Walker](#) is a procedural generation algorithm that works by following a “walker” that travels the map randomly and places floor tiles. The walker starts with some initial position and initial direction. Each step, the walker will place a floor tile at its current position and move forward. Then, it will randomly change direction and repeat until it has walked the number of steps provided.

After the walker has finished places all the floor tiles, they will be covered by the appropriate wall tiles to finish the map

Fields



Start Position - The position for the walker to start the algorithm.

Walk Length - The number of steps for the walker to take before finishing.

Floor Tilemap - Tilemap GameObject to populate with floor tiles.

Wall Tilemap Bot/Top/Right/Left - Tilemaps to populate with Wall Tile Bot/Top/Right/Left respectively.

Floor Tile - Tile to use as the floors.

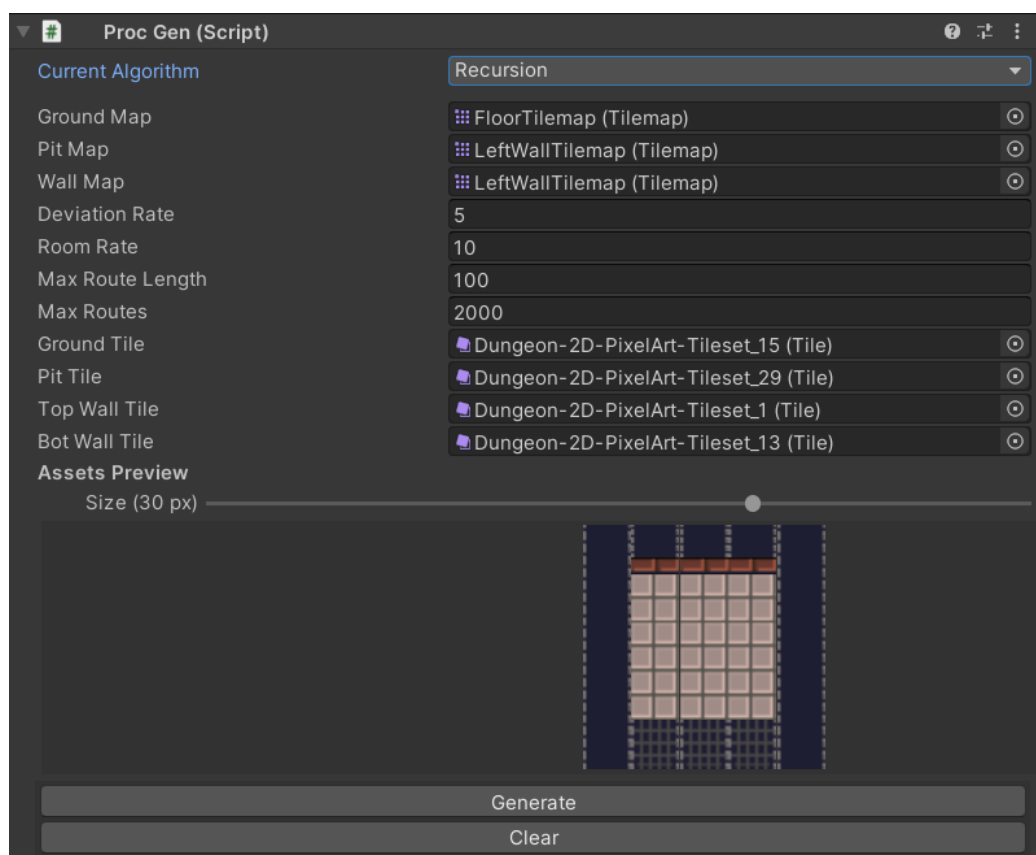
Wall Tile Bot/Top/Right/Left - Tile to use as the Bot/Top/Right/Left walls.

Recursion

Algorithm Explained

The recursion algorithm picks a starting point and moves in a straight line. At each step, we stochastically sample a number and if that number is less than the deviation rate we will make a left turn. If the left turn is not made, we sample another number to determine and repeat the process for right turns. Lastly, we sample a third number to decide whether we should spawn a room. The third number is compared against the room rate.

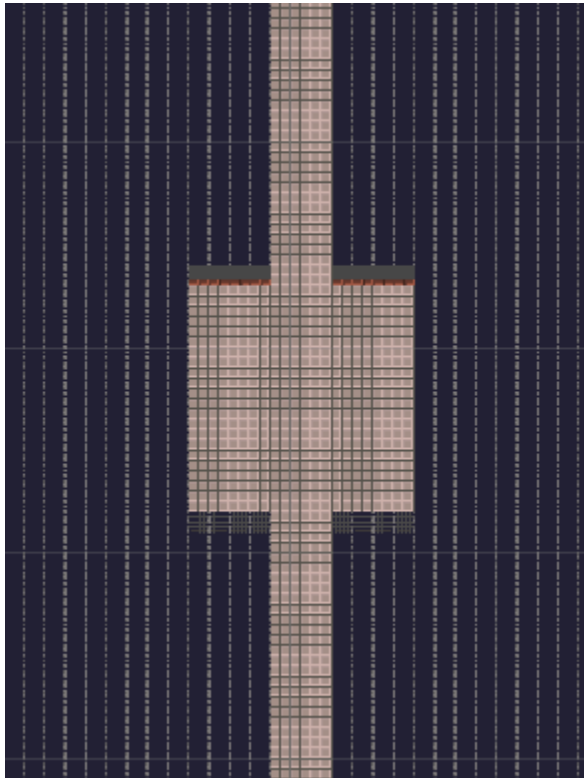
Fields



Ground/Pit/Wall Map - Tilemap GameObject to populate with Ground/Pit/Wall tiles respectively

Deviation Rate - The deviation rate represents the probability we branch off a mainpath into a subpath in our procedural generation algorithm .

Room Rate - The room rate represents that there is a room generated in the middle of a hallway. A “room” are one of these rectangular boxes



Max Route Length - The max route length, represents the maximum distance between the two furthest points on the map

Max Routes - The max routes, represents the maximum number of splits from the main path of our procedurally generated map

Ground/Pit/Top Wall/Bot Wall Tile - Tile to use as the Ground/Pit/Top Wall/Bot Wall respectively

L-System 2D

Algorithm Explained

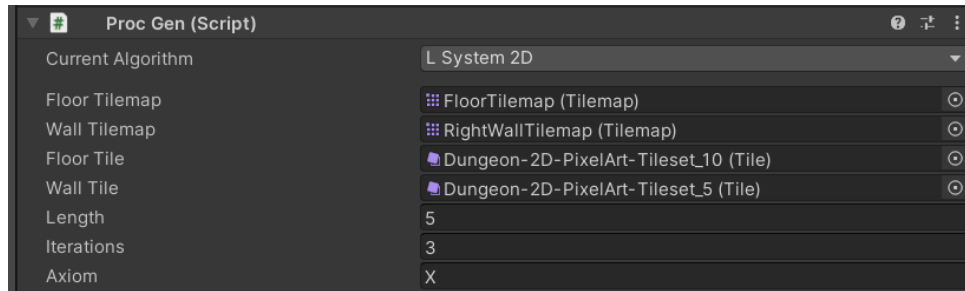
[L-System](#) is a procedural generation algorithm that is based on randomly applying provided rules. The algorithm will start with some initial string, called the axiom, and randomly replace each instance of a recursive character with one of the user-provided rules. These rules represent how the procedural world is to be generated.

The following are brief explanation of what the characters mean:

Character	Description
X	Recurse (Default): This character will be replaced by a rule in the next generation iteration. It can be any character you want but it is 'X' by default.
F	Forward: This tells the procedural generator to move forward in the direction it is currently facing and place floor tiles.
+	Turn Right: This tells the procedural generator to turn right 90 degrees. Note that this only turns the direction it is facing and does not place any tiles.
-	Turn Left: This tells the procedural generator to turn left 90 degrees. Note that this only turns the direction it is facing and does not place any tiles.
[Start Branch: Remember the current position and direction.
]	End Branch: Revert back to the position and direction of the most recent start branch ('[').

This is a very brief explanation. If you want to know more in-depth please play around using the toolkit to get a feel for how it works or feel free to do your own research about how L-System works.

Fields



Floor/Wall Tilemap - Tilemap GameObject to populate with floor tiles and wall tiles respectively

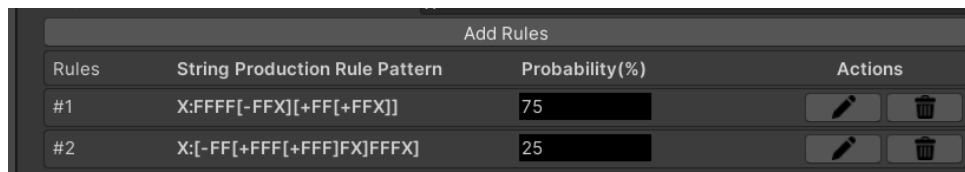
Floor/Wall Tile - Tile to use as the floors and walls respectively

Length - The number of tiles to be generated for each Forward action


Iterations - The number of iterations to apply the rules to create a bigger map

Axiom - The string to start applying the generation from. For the majority of use-cases it will be sufficient to leave it as the default.

Rule Generation



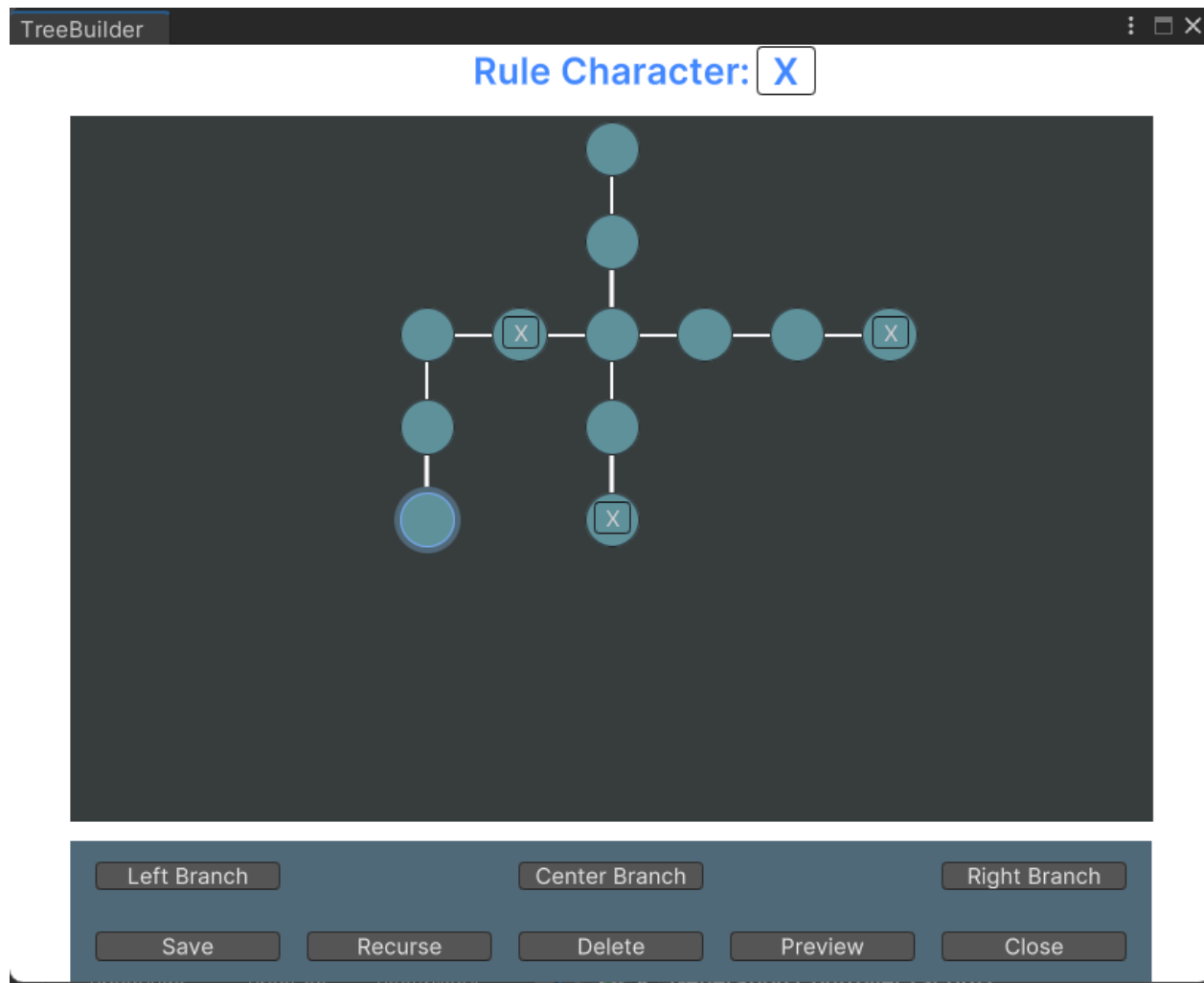
Add Rules - Add a new rule to the list of rules. This will automatically open the generation GUI.

Edit Rule () - Opens the generation GUI for the selected rule. If changes are saved, it will overwrite the selected rule with the new rule.

Delete Rule () - Permanently deletes the selected rule.


Probability (%) - The probability that the selected rule will be applied when faced with a recurse character during generation. The sum of all the probabilities of all the rules for the same rule character (see below for details on what a rule character is) must equal 100%.

The following is the rule generator GUI:



The example above represents a single rule. The string representation of the same rule is “FF[-FXF[+FF]][+FFFX]FFX”.



Currently Selected Node () - All the additional actions (Left, Right, Center, Recurse) will be applied on the currently selected node. You may click on any node to select it.


Left Branch - From the currently selected node's position and direction, turn left ('-' character).

Right Branch - From the currently selected node's position and direction, turn right ('+' character).

Delete - Delete the currently selected node and all of its children.

Preview - Open the preview window to see what this rule looks like with the user provided assets.

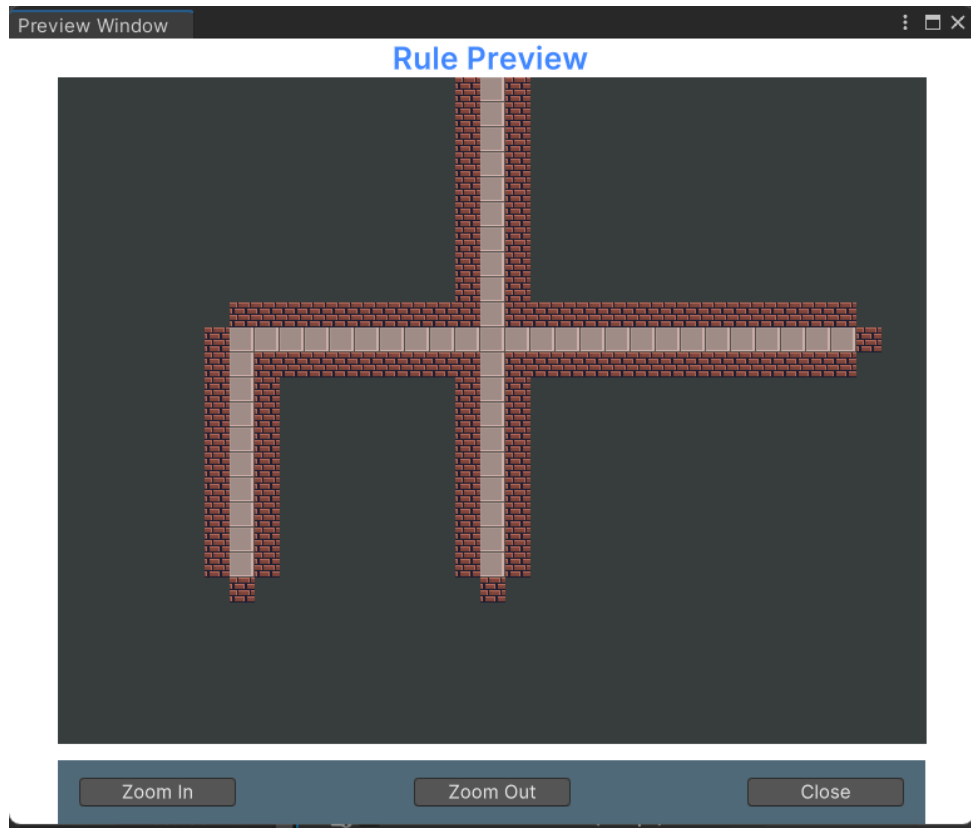
Recurse - Make the currently selected node into a recurse point. A recurse

node is shown by a character inside of it () which represents the character that will be left in that position. This character may be changed to be any other character. Pressing the Recurse button while having a recurse node selected will turn that node back into a non-recurse node.

Rule Character - Select what the recurse character will be for this rule. This is the character that will be replaced when the selected rule is applied.

Preview Window

When the Preview button is pressed in the Rule Generator GUI, the following window will appear, showing a preview of the currently selected rule:



This preview will use the Tile assets provided by the user in the inspector window to give a preview of what the rule shown in the Rule Generator GUI will look like in-game. If the rule is changed, or if the assets are changed, it will be reflected in the preview screen.

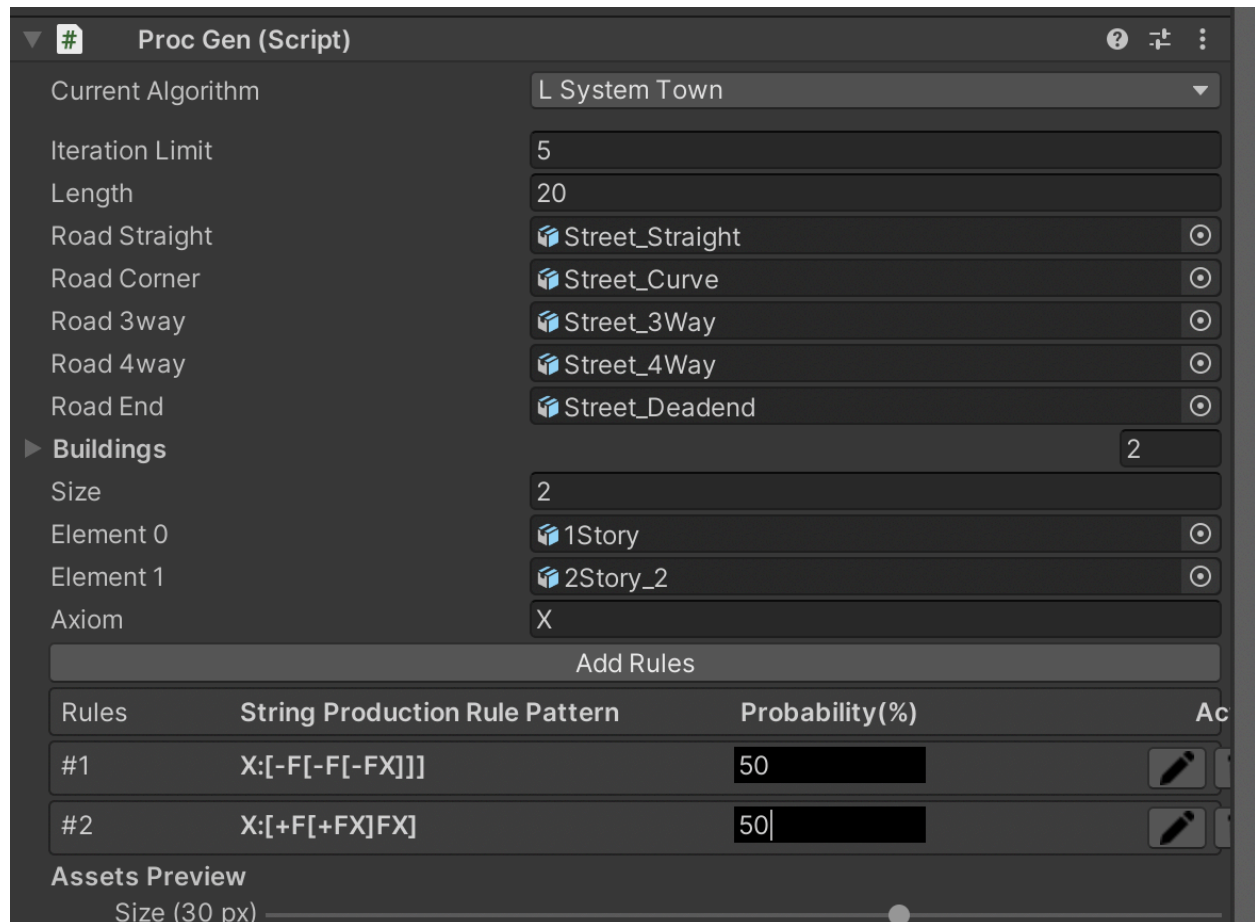
L-System Town

Algorithm Explained

The algorithm to plot the streets is basically the same as the algorithm as L-System 2D, you can refer to the L-System 2D for more details. After the streets are placed initially, there's a fix up to rotate the corner road or replace the straight road with a 3 or 4 way road if there are multiple roads adjacent to each other.

The algorithm to plot the buildings around the streets is the following: there is a list of buildings where the user can input a size and add the building prefabs accordingly. Once the streets are placed, the spaces around the streets are checked to place the randomly chosen buildings (or empty place).

Fields



Iteration Limit - The number of tiles to be generated for each Forward action

Length - The number of iterations to apply the rules to create a bigger map

Axiom - The string to start applying the generation from. For the majority of use-cases it will be sufficient to leave it as the default.

Road Straight - Straight road to place when the character is F.

Road Corner - Corner road to place when the character is + or -, rotated accordingly.





Road 3way - 3way road to replace Road Straight when there are 3 roads adjacent to the current road.

Road 4way - 4way road to replace Road Straight when there are 4 roads adjacent to the current road.

Road End - Road end to place when the character is].

Buildings - a list of buildings to place around the road.

Rule Generation

Add Rules			
Rules	String Production Rule Pattern	Probability(%)	Actions
#1	X:FFFF[-FFX][+FF[+FFX]]	75	 
#2	X: [-FF[+FFF[+FFF]FX]FFFX]	25	 

Rule Generation is the same as L System 2D rule generation, refer to above section for more details.

Note that currently the preview window is not supported for 3D (coming soon)