GBBG Documentation

Grammar Based Building Generator

Content

Installation

To install the tool the user needs to import the provided unity package into the project. No further actions are required.

Features

- Derivation algorithm: this algorithm executes the derivation process. It selects and applies the rules given by a specific grammar.
- Editable grammar. The rules can be modified form the editor interface.
- Easy creation of rules and shapes
- Visual representation of the derivation. The user can explore all steps of the derivation process using a visual guide.
- Custom interface to edit grammars. Allows the user to create, delete, duplicate and edit rules. Also can create shapes easily for the selected grammar.
- Custom tool to prepare the 3D models for the post-production process.

How it works

A grammar-based generation is a form of procedural generation. The main elements of this system are shapes and rules.

Shapes

Shapes are the fundamental building blocks of the system. They are identified by a unique string called a symbol, which distinguishes each shape and its instances. Shapes represent a defined area of space, known as scope, which can be either two-dimensional or three-dimensional. The scope is characterized by its position, rotation, and scale. Additionally, shapes can be classified as either terminal or non-terminal. Terminal shapes are those that end the derivation process and cannot have any rules applied to them.

Propery	**Data type**	**Description**
Position	Vector3	The global position of the shape.
Rotation	Quaternion	The global rotation of the shape.
Scale	Vector3	The global scale of the shape. It only contains up to 3 decimal places.
Symbol	String	The identifier of the shape.
IsTerminal	Bool	If the shape is terminal or not.
Dimensions	Int	The number of dimensions of the shape. To work properly must be 3 or 2.
PreferedSize	Vector3	The preferred size of the shape. On application of rules with automatic sizing (repeat, grid and margin), the shape will be as close as possible to the preferred size.

Rules

Rules function as operators within the system, with the purpose of modifying and replacing existing shapes. Each rule follows a specific format:

Predecessor -> Operator(Parameters){Successor}

- The **predecessor** specifies the shape to which the rule can be applied, and it must be a non-terminal shape.
- The **operator**, or **rule type**, determines the specific action performed by the rule, such as altering the scope, dividing the predecessor shape in various ways, or other actions.
- The **parameters** vary depending on the rule type and define the characteristics of the shapes generated when the rule is applied.
- The **successors** are the shapes that result from the application of the rule.

There are a total of 6 types of rules in this framework:

Scope Rules

The outcome is a single shape that has undergone translation, rotation, or scaling. When scaling involves changes in dimensions, any dimension that doesn't exist will be represented as zero.

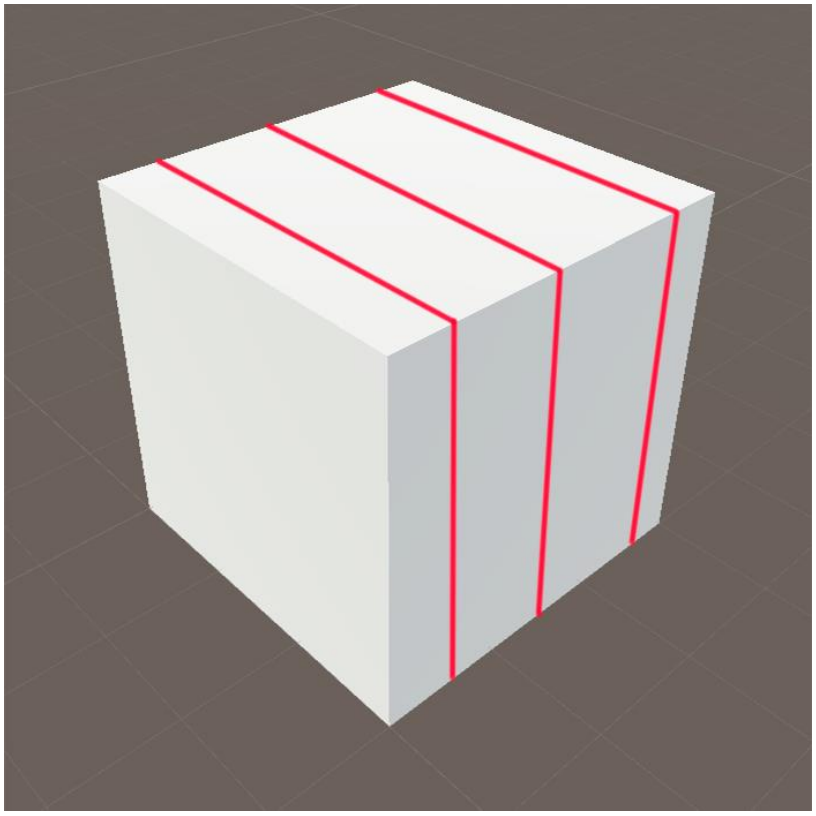
The scope rule type can also be used to replace one shape with another while maintaining its scope, introducing randomness in the selection of a successor to add variety to the derivation process.

In the three parameters of the rule, we can specify whether the given values are added to the existing values or completely overwrite them. For translation and rotation, we can determine whether these operations occur in the global space or the local space of the shape. Lastly, in the scaling operation, we can define whether the parameters are additive or multiplicative.

Split Rules

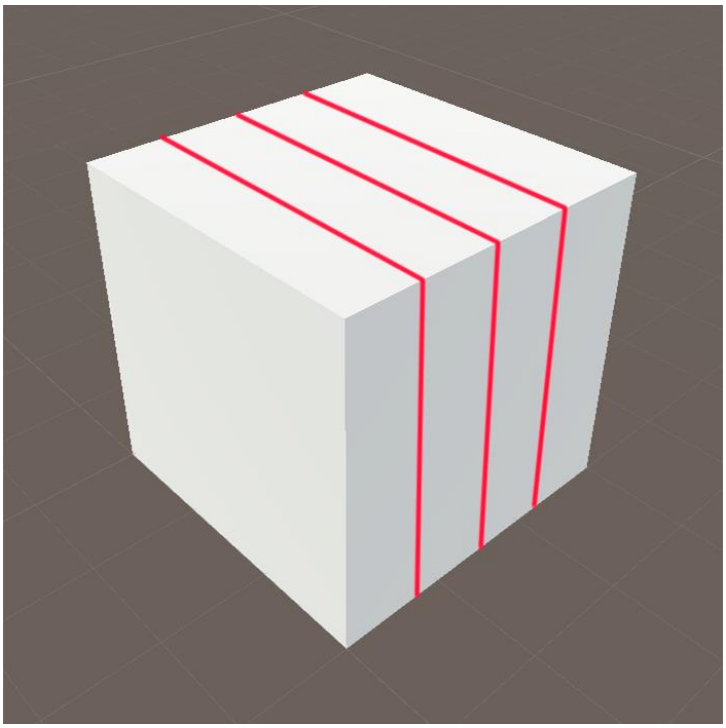
The outcome is a collection of shapes whose combination fills the entire extent of the predecessor shape. The parameters that determine how the rule is applied are as follows:

- The axis that is perpendicular to the planes defining the cuts. These axes correspond to the local axes of the predecessor shape.
- A list of values that specifies the scaling of the resulting shapes along the selected axis. The list should have fewer elements than the list of successors to ensure the cuts align properly.
- A Boolean value that indicates the direction of the cuts. When set to true, it is labelled as "FromRoot," which means the cuts are made from the root towards the rest of the shape. When set to false, it is labelled as "ToRoot," indicating that the cuts are made towards the root.



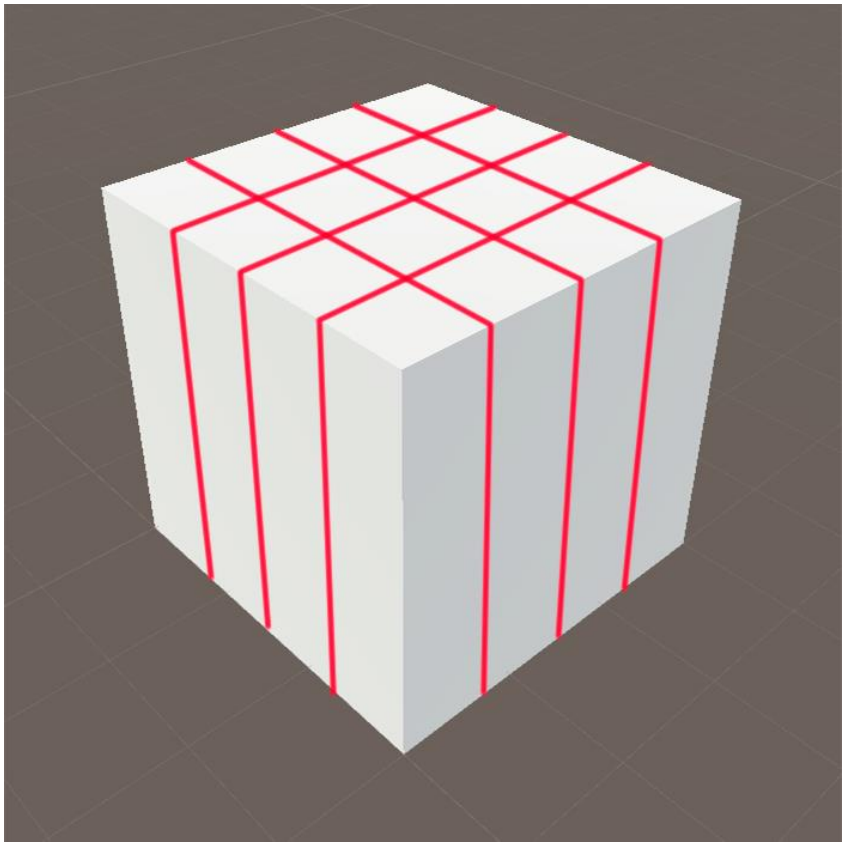
Repeat Rules

The repetition rule functions similarly to division, but if we return to the analogy of mathematical operators, if the division rule is like addition, the repetition rule is like multiplication. In other words, all the successors will be of the same type, and the distance between cuts will adjust according to the preferred scale of the successor shape. Therefore, the only parameter needed will be the axis that defines the division plane.



Grid Rules

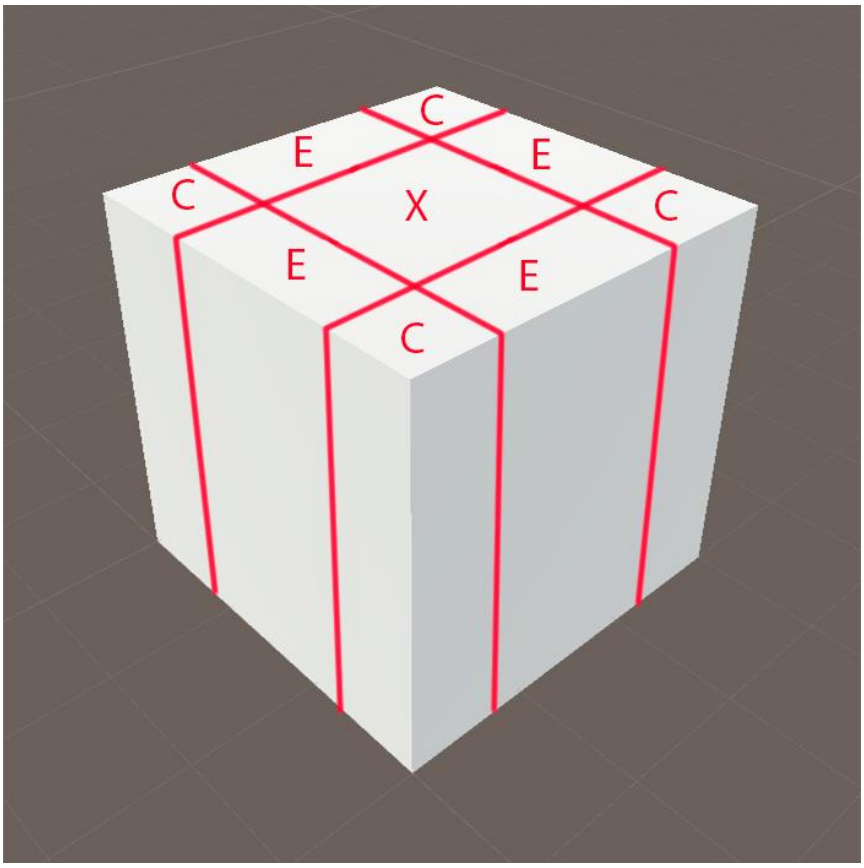
The grid rule operates similarly to the repetition rule, but instead of divisions along a single axis, divisions are made along two axes. Similarly, all successor shapes will have the same scale. The only parameter needed is the axis, which corresponds to the axis that is parallel to both planes of division.



Margin Rules

Margin rules generate up to nine successors arranged in a 3 by 3 pattern. These successors fall into three categories: corners (C), edges (E), and center (X). The parameters that determine how this rule is applied are as follows:

- The axis that is parallel to the division planes.
- The margin value (a floating-point number) that determines the width of the margin.
- The type of margin, whether the margin value is an absolute value or a relative value to the predecessor.
- The orientation of the corners: whether they all face outward, all face in one direction, or all face inward.
- A boolean value that determines whether the central piece should be included among the successors.

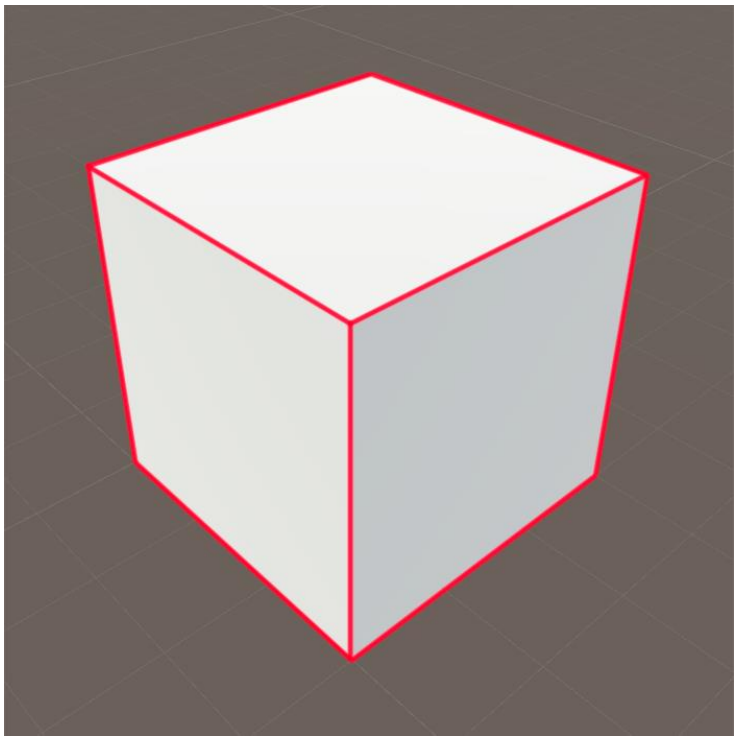


Component Rules

The component rule divides a shape into its constituent parts, dividing a cube into its individual faces. Each of these faces will be represented as a two-dimensional shape. The parameters for this rule are as follows:

- Axis: Specifies the orientation for the next step.
- Division mode: Determines which faces will give rise to a new shape. The options include the top face, bottom face, side faces, or any combination of these.

The resulting shapes are determined based on the order of the successors listed.



Derivation

The derivation is the process of applying rules to the shapes. To apply the rules its done using a recursive function that applies the following algorithm.

It starts with a list of shapes that it is called **axiom** and defines the initial state. The algorithm works as follows:

1. Given an active shape (S) in the list, the algorithm checks if the shape is terminal, if it's non-terminal the algorithm proceeds.
2. A rule is selected from the rules that can be applied to the shape S.
3. The successor or successors (S') to the shape S is computed.
4. The shape S is deactivated and removed from the active list
5. The S' shapes are added to the active shape list.
6. The function is called to derivate this shapes.

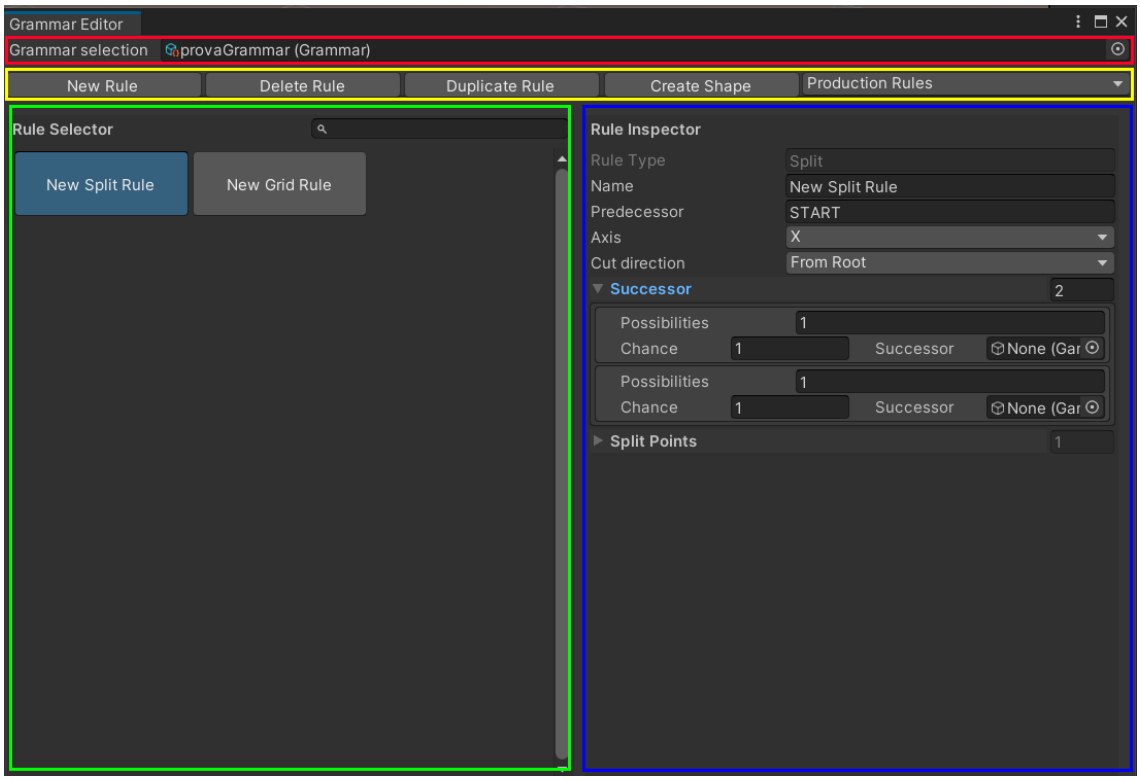
When the derivation finishes all remaining shapes are terminal ones.

Post-Process

After the derivation is done, all terminal shapes are replaced with the final 3D mode. This is done following the post-process rules, which pair a symbol (string) with a 3D model.

Interfaces

Grammar Editor



The grammar editor interface has four main segments.

- (RED) The grammar selector. In this field the user selects a grammar to be edited.
- (YELLOW) The toolbar contains buttons to create, delete and duplicate rules and shapes.
- (GREEN) The rule selector is used to select rules by clicking on it.
- (INSPECTOR) The inspector shows all the parameters that can be edited from the selected rule.

Toolbar

- New Rule: Creates a new rule. A popup window lets the user select the type of rule. This rule will be saved as:

"Assets/GBBG_Assets/<GrammarName>/Rules/<RuleName>.asset

- Delete rule: Deletes a rule permanently. A popup will ask twice to prevent miss-click.
- Duplicate Rule: Creates a new rule with the same type and parameter values as the selected rule.
- Create Shape: Creates a new shape, which can be tri-dimensional, bi-dimensional or empty. The symbol of the shape is defined in the creation box and the shape will be saved as:

"Assets/GBBG_Assets/<GrammarName>/Shapes/<ShapeSymbol><2D|3D|Empty>.asset"

- Production rules / post-process rule selector: Lets the user change the content of the rule selector. It can display production rules or post-process rules.

Rule Selector

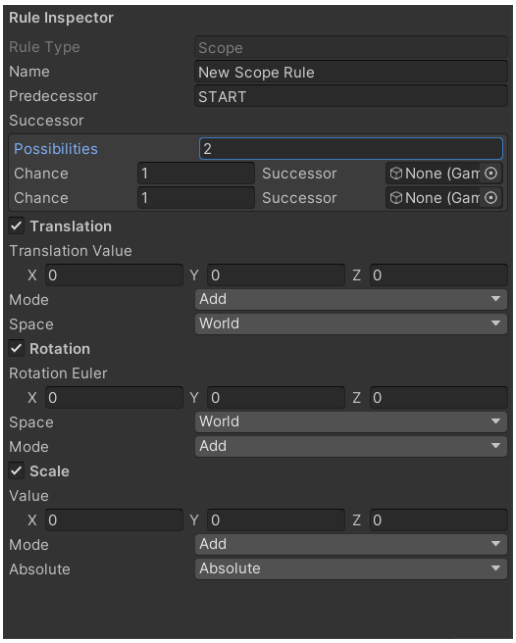
The rules are displayed in a grid and the user selects them by clicking. There is a search bar to filter the rules by name.

Rule Inspector

The rule inspector changes slightly depending of the type of the selected rule, but there's some parameters that are shared across all rules.

- Rule type: indicates the user the type of rule. This parameter cannot be modified.
- Name: Is the name of the saved file and the identifier of the rule on the rule selector.
- Predecessor: The symbol (string) of the shape that this rule can be applied to.
- Successor: the successor can store a list of possible successors, but always must be at least one. For each possible successor the user can specify the chance of each possible successor. Before picking one all chances are normalized, so it doesn't have to add all to one. If there's two possible successors with chances of 1 and 2 respectively, the final chances will be 0.33 and 0.66 over 1.

Scope

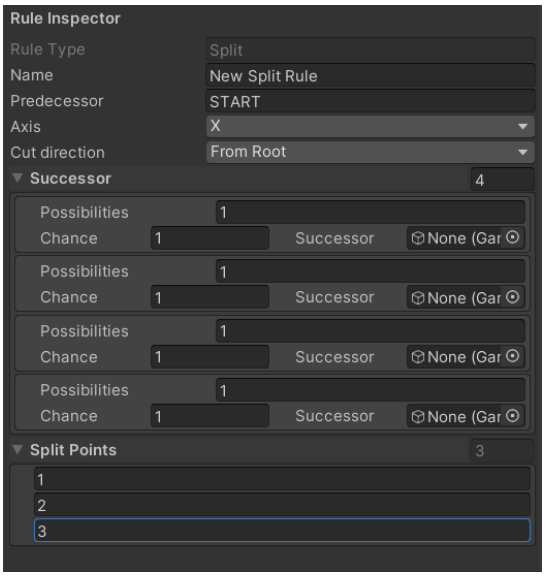


The translation, rotation and scale can be activated using a toggle. If one of them is deactivated it will not affect the successor shape, ignoring the given values of the parameters.

- Translation value: a Vector3 that stores the translation to be applied.
- Translation mode: defines how this translation will be applied.
    - Add: adds the translation value to the current position
    - Set: sets the current position to the translation value.
- Translation space. Defines if the translation is applied relative to the world or relative to the predecessor shape.
- Rotation Euler: a vector3 that stores an Euler rotation.
- Rotation space: defines if the rotation will be applied relative to local or global space.

- Rotation mode: defines if the rotation value will be added or set the rotation to the value.
- Scale value: a Vector3 that stores the scale value.
- Scale mode: defines if the scale will be added or set to.
- Scale absolute/relative:
    - Absolute: the value will be treated as an additive.
    - Relative: the value will be treated as a multiplier.

Split

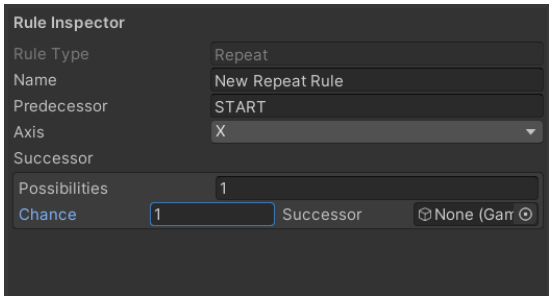


The split rule, instead of having only one successor has a list of them, but the functionality is equal.

The specific parameters of the split rule are:

- Axis: the axis defines the orientation of the cuts. The defined axis will be perpendicular to the division planes. The chosen axis is the local axis of the predecessor shape.
- Cut direction: specifies the order of the cutting. “FromRoot” will make the cuts from the root of the predecessor shape outwards, and “ToRoot” will do the cuts from the far side to the root.
- Split points: define the size of the successor pieces. This list is always one shorter than the successor list because the las shape will always reach to the end of the scope of the predecessor shape.

Repeat

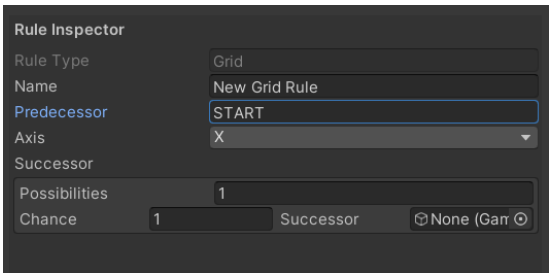


The repeat rule has only one parameter, as the width of the resulting shapes and the number of successors will be calculated based on the preferred size of the shape.

- Axis: the axis defines the orientation of the cuts. The defined axis will be perpendicular to the division planes. The chosen axis is the local axis of the predecessor shape.

Grid

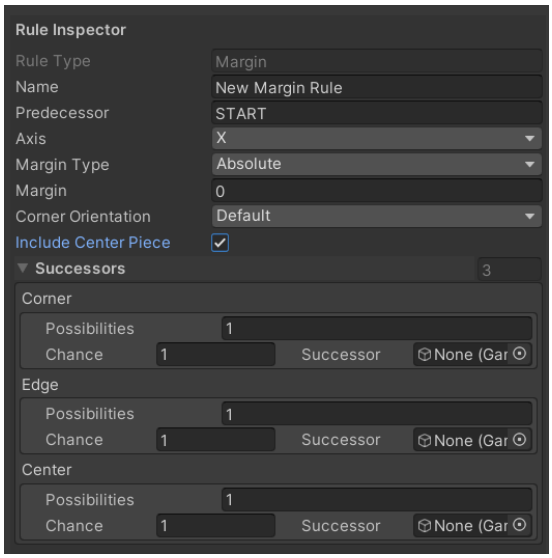
The grid rule has only one parameter, as the width of the resulting shapes and the number of successors will be calculated based on the preferred size of the shape.



- Axis: the axis defines the orientation of the cuts. The defined axis will be parallel to the division planes. The chosen axis is the local axis of the predecessor shape.

Margin

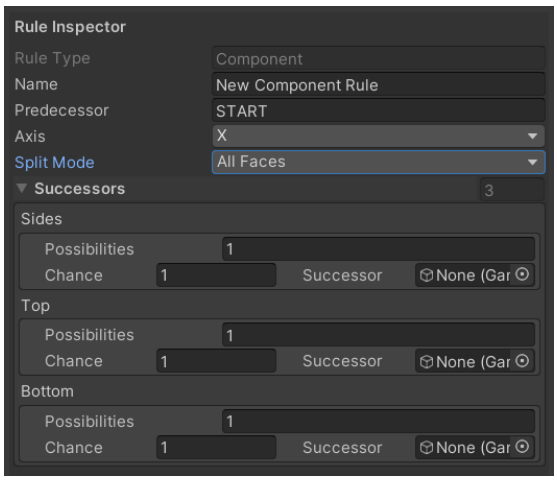
The margin rule has the following parameters:



- Margin type: Absolute will take the margin value as is. Relative will multiply the margin value with the predecessor scale in each axis.
- Margin value: defines the width of the margin, if margin type is set to “relative” margin value must be between 0 and 0.5.
- Corner orientation: defines the orientation of the successor shapes on the corners. By default all face the same direction as the predecessor, but soma plications need them to face out or face in.
- Include centre piece is a toggle that allows the user to deactivate the central piece and get only a ring of successors.
- The successors in this rule are specified by the position. Centre, corner and edge.

Component

The parameters of the component rule are the following:

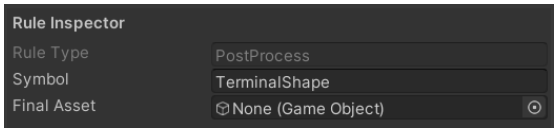


- Axis: defines the orientation to define which faces are top and bottom. The axis will be perpendicular to the top and bottom.
- Split Mode: defines which faces will result after the application of the rule. Depending on the mode selected, the rule will only demand that type of successor.

Post-process

Post-process rules only have two parameters and are distinct of all previous ones.

- Symbol: the symbol of the terminal shape that will be substituted.
- Asset: the prepared 3D model that will be instantiated in its place.

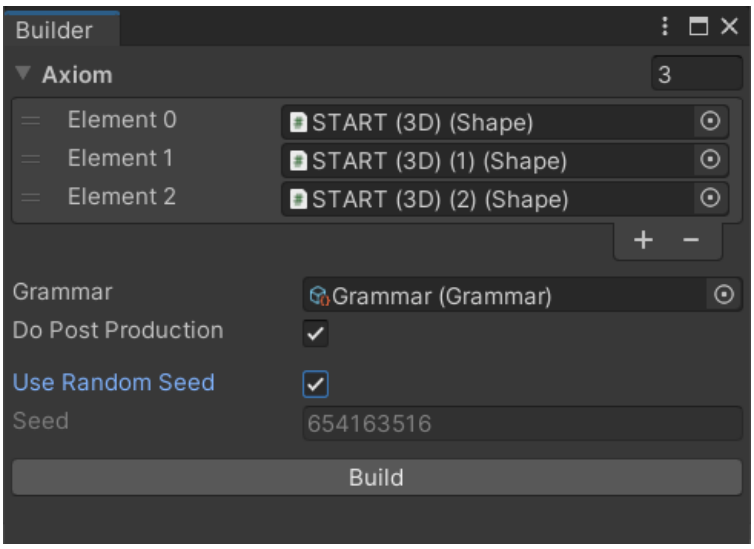


Builder

The builder is the interface that is used to set and execute the derivation process.

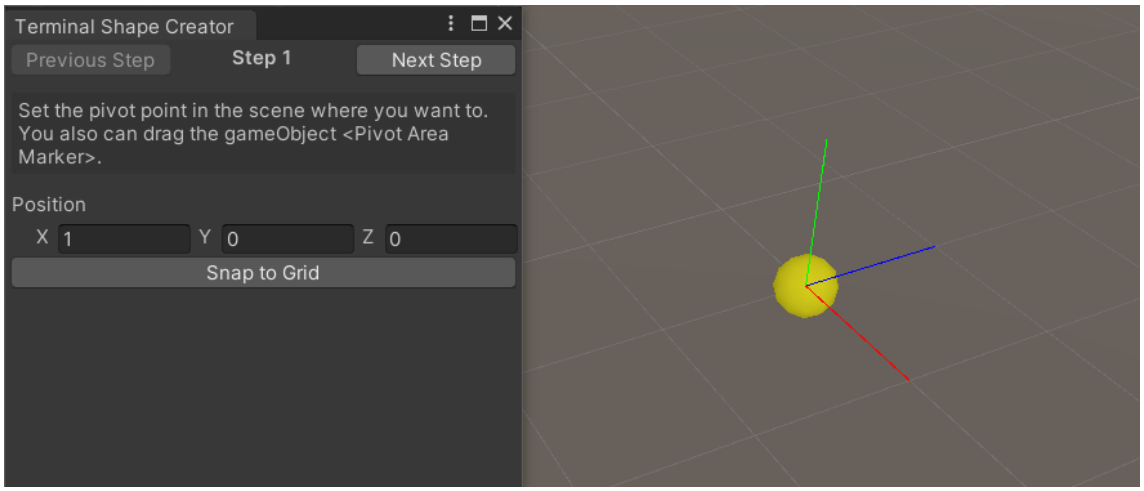
Its parameters are:

- Axiom: List of initial shapes in the derivation process.
- Grammar: the grammar that will be used.
- Do Post Production: Toggle that controls if the post-production is executed.
- Use Random Seed: controls the seed that is used in the derivation process for all random number generations. If true the system will use a random seed each time. If fals the seed will not change.
- Random seed: Displays the current random seed. If “Use Random Seed” is not selected the user can input a specific seed. This seed must be within the bounds of a signed integer.
- Build Button: Executes the derivation process.

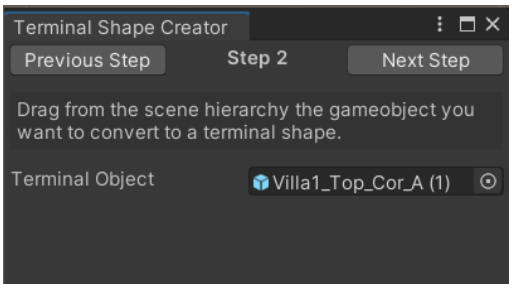


Terminal Shape Creator

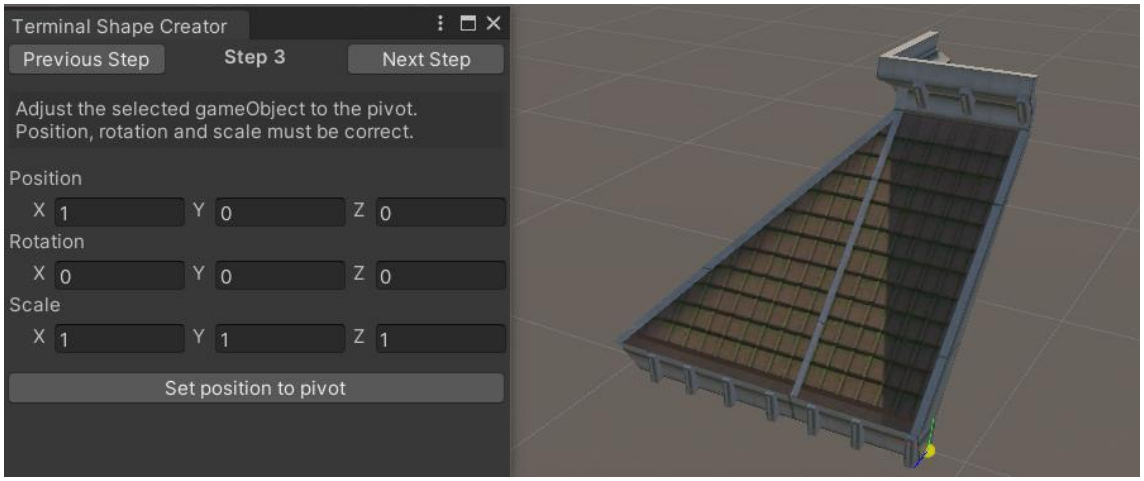
Terminal Shape Creator is a tool that is used to easily prepare 3D models to be used in the post-production process. The tool guides the user across 7 easy steps.



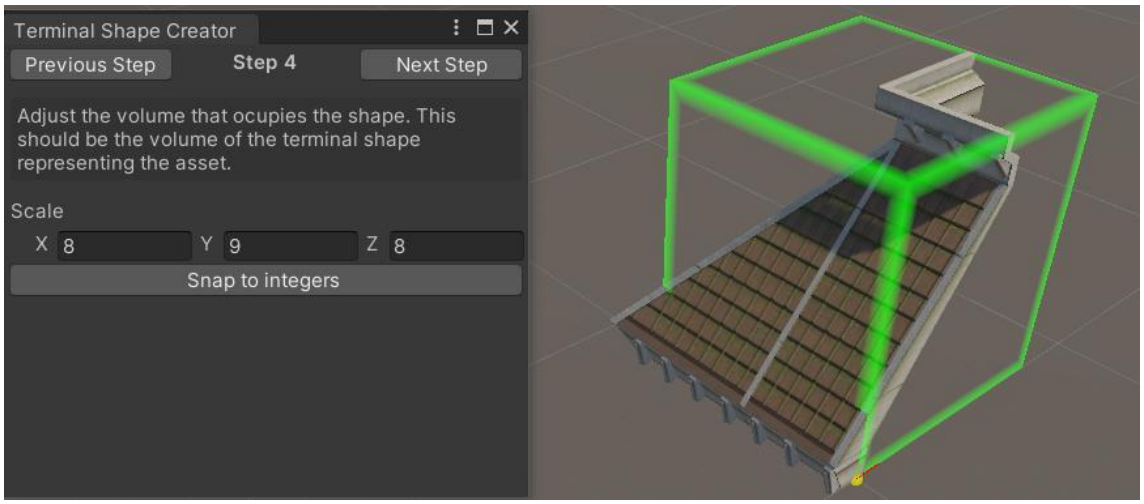
1. Select a position in space using the given GameObject. The position can be selected modifying the vector3 in the interface or moving the GameObject in the scene.



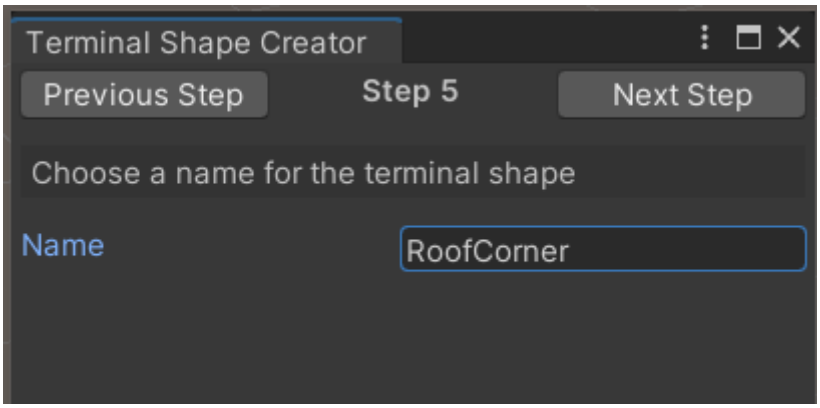
2. Select the 3D model. Import the 3D model to the scene and drag it from the hierarchy to the interface.



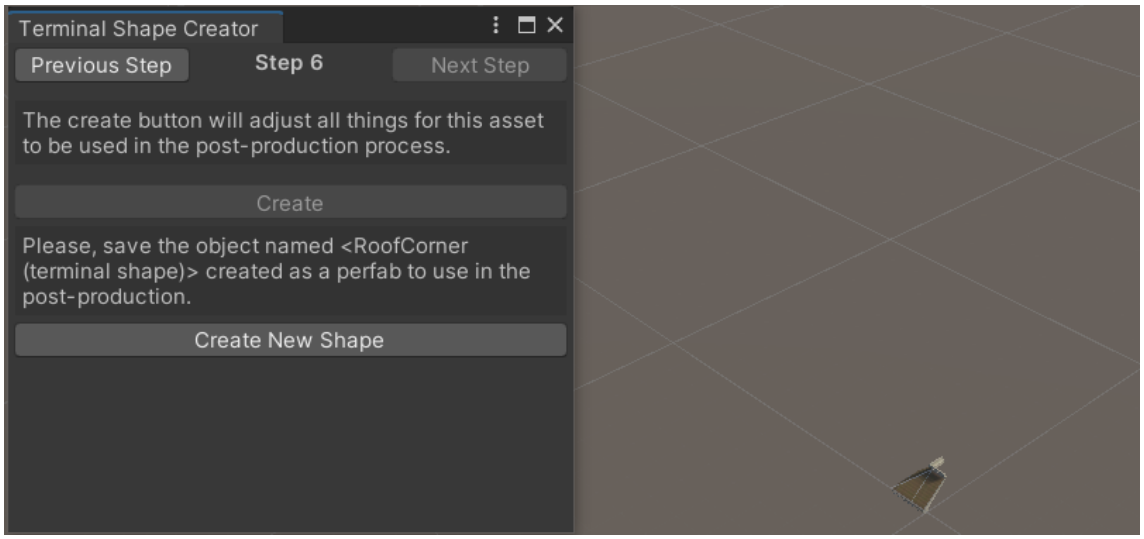
3. Adjust the selected position to match the position and rotation of the root.



4. Define the volume of the 3D model, a visualizer is displayed to help the user see the size of the volume.



5. Name the object

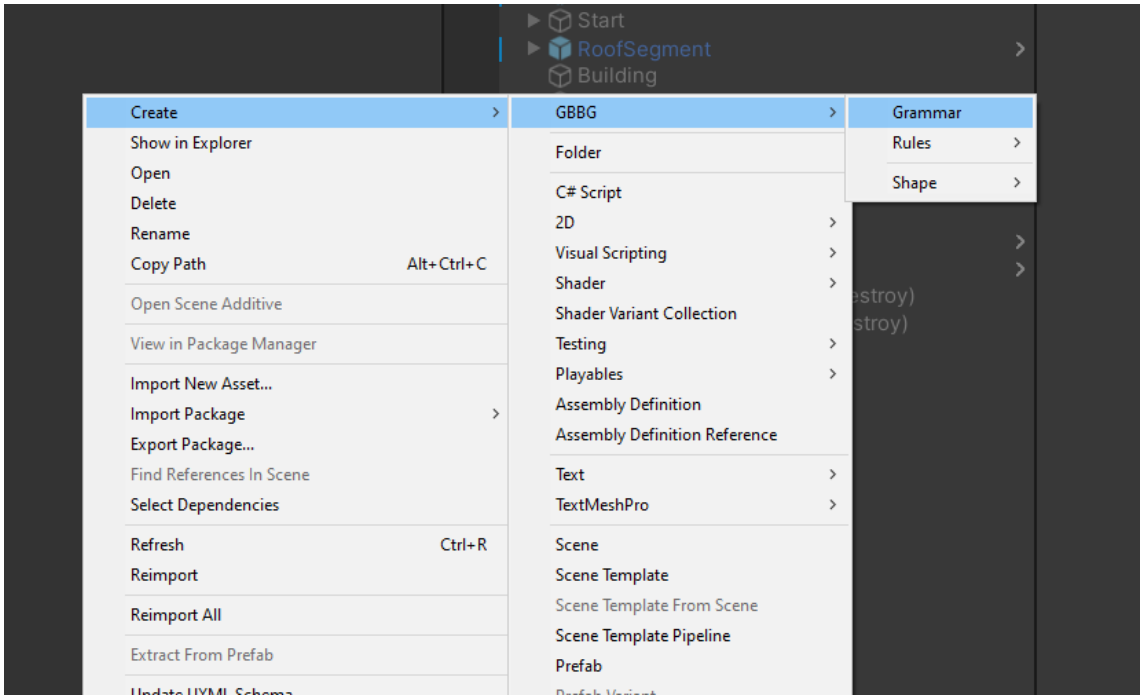


6. Press the “Create” button to apply all the settings.
7. Save the created object as a prefab. This prefab can be used in the post-process rules.

Getting started

Create a grammar

To create a grammar it can be done using Unity's create menu. Rules and shapes can also be created this way but is heavily recommended that they are created using the Grammar Editor.

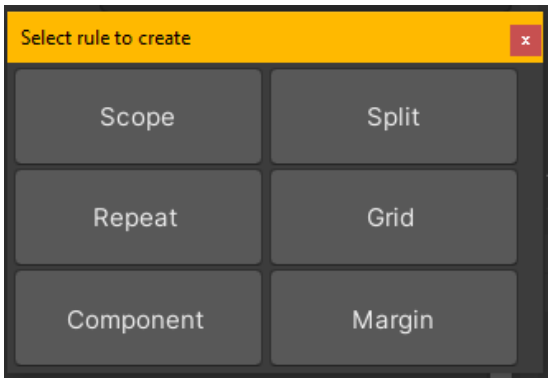


Once the .asset file of the grammar is created the user must open the Grammar Editor. From unity's toolbar: Window > GBBG > Grammar Editor.

Check the Grammar Editor part on this manual for more information about the Grammar Editor.

Create Rules

Form the Grammar Editor, with a grammar selected. Click on the button “New Rule”. On clicking will appear the following pop-up window, select the type of rule to create.

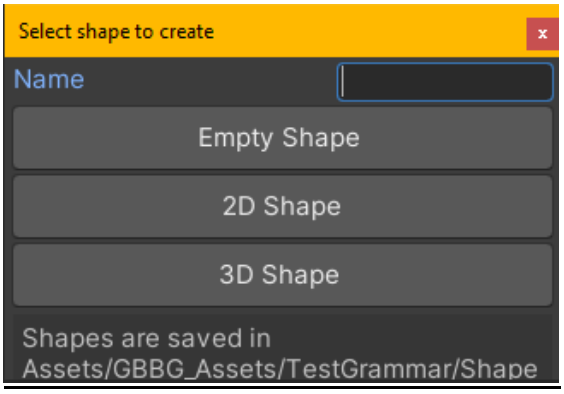


When a rule is created, it will appear in the rule selector section of the Grammar Editor. Make sure to set the grammar editor to production rule mode to view the production rules or to post-process mode to view the post-process rules.

The asset file (.asset) of the rules is saved in the folder: “Assets/GBBG_Assets/<GrammarName>/Rules

Create Shapes

To create shapes from the Grammar Editor the user must use the “Create Shape” button. When clicked it will appear a pop-up window where the user will name the shape (file name and symbol) and then click the button “3D”, “2D” or “Empty” to create the shape of the selected type.



The asset file (prefab) of the shapes is saved in the folder: “Assets/GBBG_Assets/<GrammarName>/Shapes