GENA

By Procedural Worlds

GeNa is a sophisticated spawning system that enables you to rapidly and intuitively populate your scenes with assets.

Version 1.5.0.0 – December 2016

Contents

Installation

Installing GeNa will create the following folder structure:

GeNa:

Documentation: GeNa documentation

Masks: Some sample GeNa masks

Scripts: GeNa source code

Introduction

Based on the Wagiman Aboriginal word “Ge-Na” which means “to put” or “to plant”, GeNa is a powerful and intuitive system that enables you to quickly populate your scenes with terrain details (grass), terrain trees, and prefabs.

The process consists of adding a spawner into your scene, adding resources to it, visualising where those resources will be spawned, and then spawning those resources into your scene.

The spawner is the object that does the work. It contains the spawner specific configuration and intercepts and acts on your commands in the editor. It can also be used at runtime to spawn new objects into your scene. You add it into your scene by selecting GameObject->GeNa->Add Spawner.

Visualising the area in which your spawn can operate is done with SHIFT + Left click. When you do this the area that is considered valid to spawn in is coloured with green balls. You can change the way in which the target location is interpreted by editing the spawners Spawn Criteria.

Spawning (the process of placing your resources into the scene) is initiated by pressing either the CTRL + Left mouse button to kick off a local spawn, or SHIFT + CTRL + Left mouse button to kick off a global spawn.

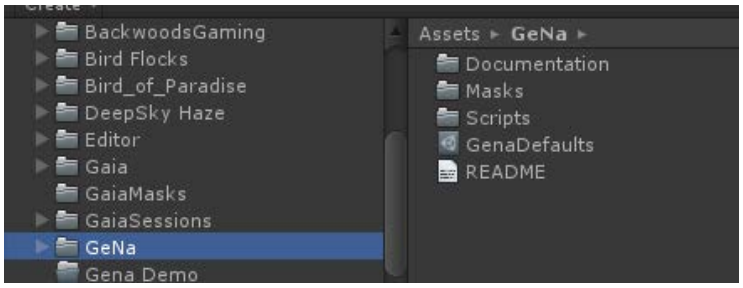
When a spawn is initiated, the spawner uses the Spawn Type algorithm in its Placement Criteria to choose a spawn location, and then evaluates that location against the Spawn Criteria. If the location falls within that criteria, the spawner then randomly chooses one of its active spawn prototypes, and runs a random check against that resources Success Rate. If all this checks out then the resource is spawned, and the spawner repeats the process until all the Instances requested have been spawned within a reasonable margin for error.

The spawn iteration will terminate when the spawned Instances count has been exceeded, or the spawner cannot find valid locations to spawn.

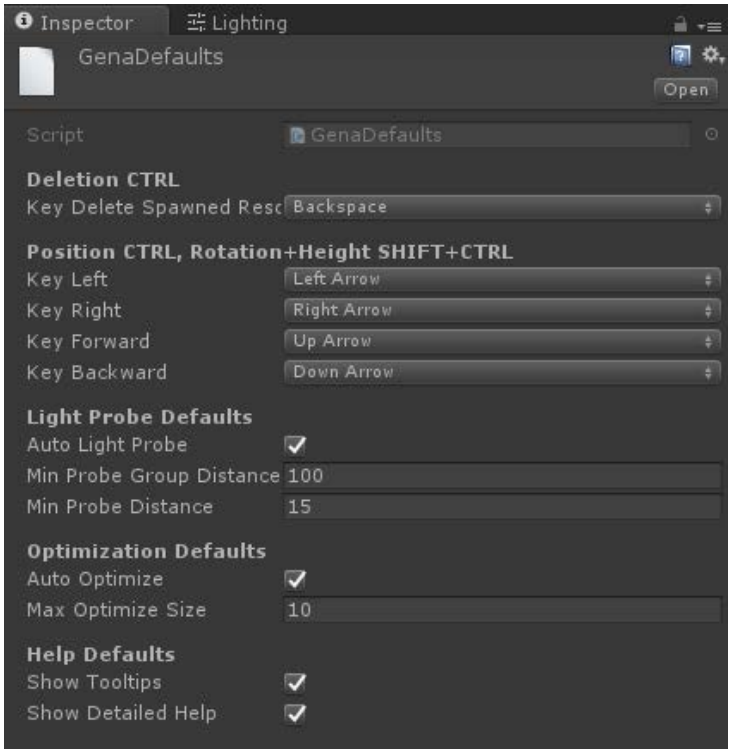
NOTE: The instances count is enforced only per iteration, so multiple spawn iterations will have a cumulative effect on the instance count as more resources are spawned.

A local spawn is a single iteration of the spawner, spawning instances up to and including its spawn range, and a global spawn is a single iteration, applied multiple times so that the entire scene is spawned on.

Defaults File

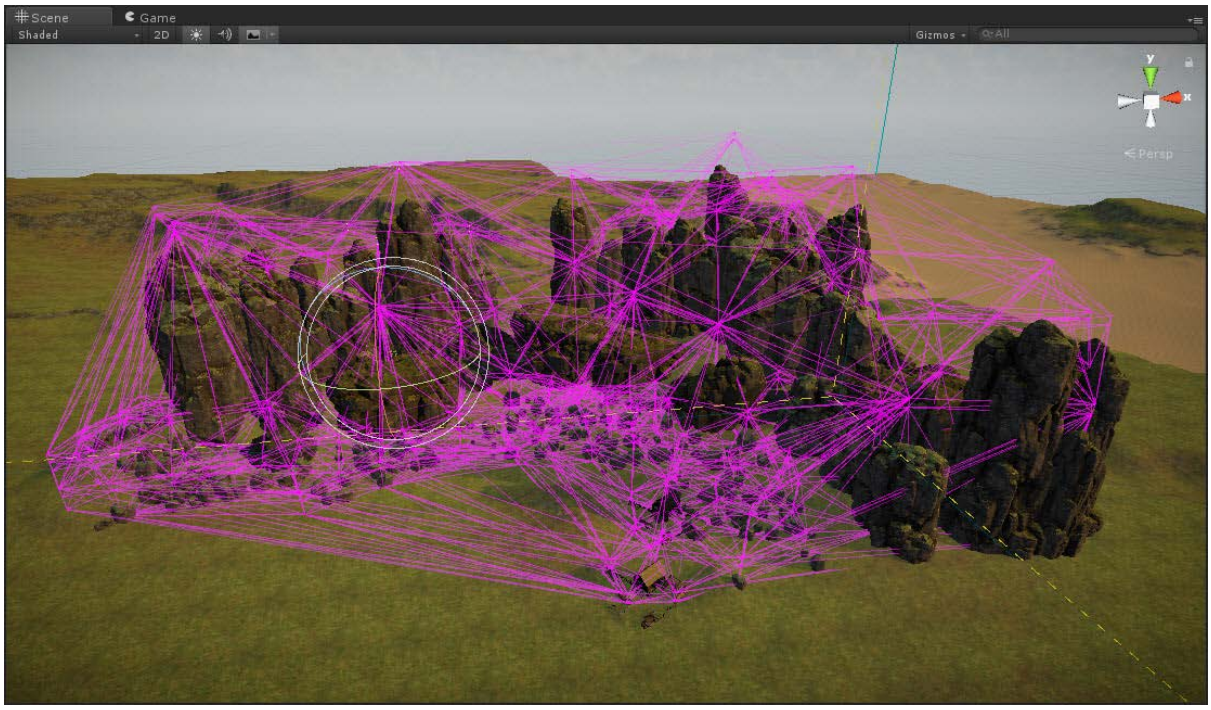


Located in the GeNa directory, the defaults file allows you to change the defaults that are added to every new spawner. There can be only one of these files per project, and it you mess this file up then just delete it and GeNa will recreate it with factory defaults.



The settings here are reflected in the way the spawner behaves.

Light Probe & Optimisation System



To get the best lighting out of the Unity Global Illumination (GI) system you should always allow your light to bake. The problem with this is that even a relatively simple scene can take hours or days.

The biggest culprit are small objects that have been set to Lightmap Static. This causes Unity to undergo a time and resource intensive baking process that is not only slow in the editor, but can also cause performance issues at runtime.

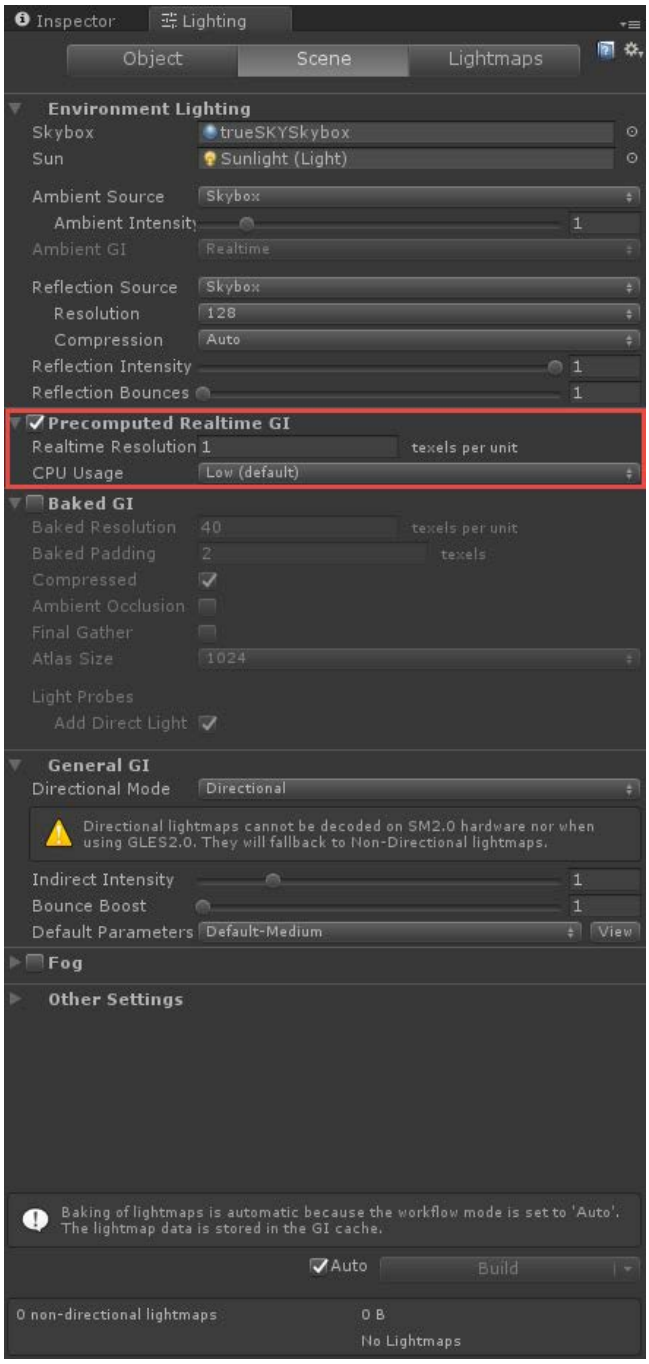
The solution to this is to use light probes. Light probes are light weight (groan), fast, and completely remove the need of lightmap baking. However, in addition to setting light probes up in your scene, the mesh renderers in the assets also need to be configured to use blend probes. When a prefab is spawned using the GeNa optimisation system it is also modified to use blend probes.

On top of this, many assets are not configured correctly to take advantage of Unity batching, and these can compound performance issues. When prefabs are spawned as optimised prefabs by GeNa they are also set up for batching.

The GeNa light-probe and optimization system addresses these issues by changing the way objects are spawned – it makes sure that the respective static flags are set optimally, that all mesh renderers have been configured to take advantage of light probes, and it automatically places light probes.

Outdoor Lighting & GI Settings

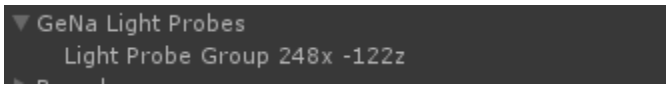
The following image shows optimal light baking settings for typical outdoor scenes. If you have a lot of houses or structures then you could perhaps increase the Realtime Resolution to 2, however the Unity recommendation is between 0.5 or 1. Note that Baked GI has been disabled as it adds overhead to the scene. The Unity recommendation is to use just one. I tend to vary lighting by time of day so I prefer to use real time GI.



Light Probe System

To enable the Light Probe system in your spawner go into Advanced Settings and ensure that Add Light Probes is selected. GeNa will use these settings to add light probes when you spawn Prefabs into your scene.

When GeNa creates light probes it will also create a GeNa Light Probes object in your scene and then parents each Light Probe Group under it. The naming standard appends the x & z location onto the light probe group so it's easy to locate and enable or disable individual probe groups.



The probes are added in stacks of 3 to best capture the light. One is ½ meter above the spawn point, one ½ meter above top of the prefab, and another 5 meters above that.

The probe system enforces minimal distances between probe groups and probes so that you do not get too great a density of them, as this can be detrimental to performance.

The **Min PG Dist** setting is the minimum distance between Light Probe Groups that are added to your scene.

The **Min Probe Dist** setting is the minimum distance between the Light Probes that are added to your scene. Adding too many light probes can be detrimental to performance, so do not make this setting too small.

Here is a great tutorial from Unity on Precomputed Realtime GI:
https://unity3d.com/learn/tutorials/topics/graphics/introduction-precomputed-realtime-gi?playlist=17102

Spawn Optimisation System

To enable the Spawn Optimisation system in your spawner go into Advanced Settings and ensure that Spawn Optimizer is selected. GeNa will use these configuration settings to automatically optimise prefabs as they are spawned into your scene.

Optimisation consists of ensuring that all meshes are set to use blend probes (you must also turn on the Light Probe System to take advantage of this unless you prefer to manually place your light probes), and that all non-moving objects are marked as static to enable batching.

The **Smaller Than (m)** setting is the size below which objects will be automatically optimised when they are spawned, however you can also influence and override this in your resource settings.

If **Moving Object** is set, it will not be configured for batching.

If **Outdoor Object** is set, the mesh renderers on that object will be configured to pick light up from blend probes and the skybox, otherwise just from blend probes if this object gets optimised.

If **Force Optimise** is set, then this object will always be optimised when the optimisation system is switched on in the spawner.

If **Can Optimise** is set, then the object will be considered for optimisation. If you unselect this then the other flags that you set up on your resource will be used instead and no optimisation will be performed on this resource.

NOTE: By default, the connection to your original prefab is automatically maintained, so of you change your prefab, be aware that this may undo some of the settings that GeNa set when it spawned it into the scene.

Here is the Unity documentation on batching:
https://docs.unity3d.com/Manual/DrawCallBatching.html.

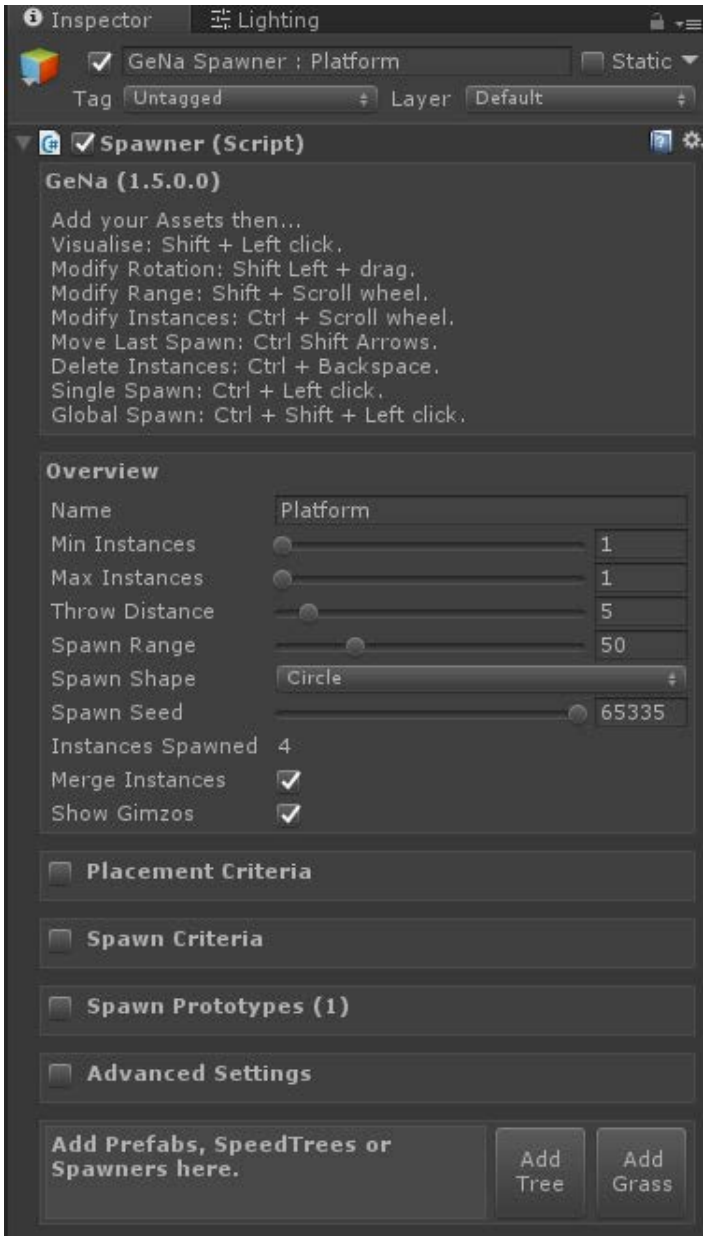
Interface

The spawner interface enables you to configure your spawner for use.

It is broken down into panels that can be opened and closed as needed.

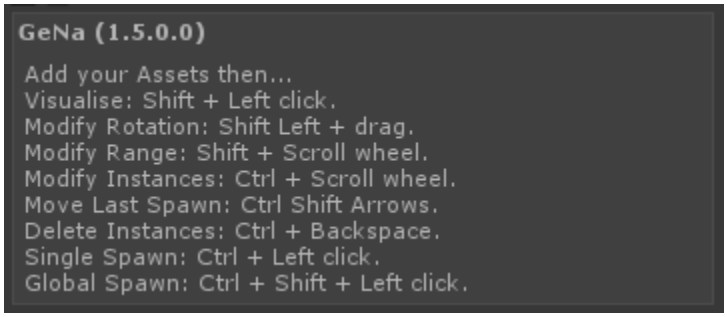
GeNa will start your session with a reasonable set of defaults based on the resources you add. The idea is to refine the operation of the spawner by incrementally changing its settings. If you get stuck with a spawner then just delete it and start again.

You can also save your spawners as prefabs if you do not want to lose the settings between scenes.



Gena Interface

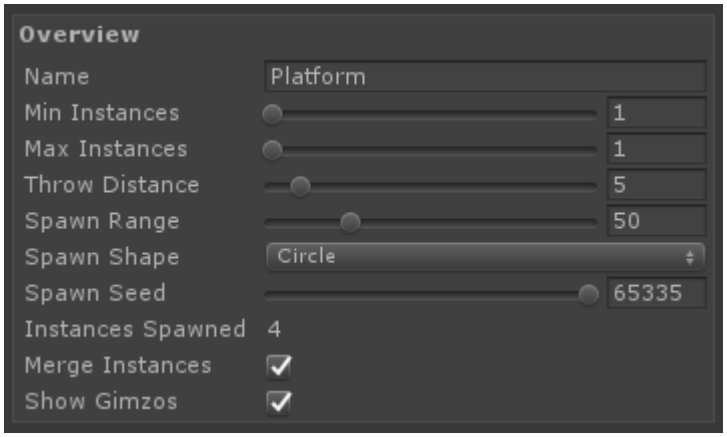
Shortcuts Panel



The shortcuts section shows the version of GeNa you are using, and then the mouse and keyboard controls for the spawner. It is there for convenience and can be hidden by unchecking 'Show Detailed Help' in the Advanced Settings part of your spawner. For more information on the keyboard and mouse controls please see the Mouse And Keyboard controls section of the manual.

Overview Panel

The overview section shows the most commonly used spawner controls.



Name: The name of the spawner.

Min Instances: The minimum number of prototype instances that will be spawned per spawner iteration.

Max Instances: The maximum number of prototype instances that will be spawned per spawner iteration.

NOTE: Gena will choose a random value between min and max. This allows successive iterations to spawn different numbers of instances which is a great way of introducing variability into your spawns.

Throw Distance: The distance that will be used by the various spawn algorithms to place newly spawned instances. How it is used will vary from algorithm to algorithm.

Spawn Range: The maximum range of the spawner in meters. Prototypes will only be spawned in this range.

Spawn Shape: The shape of the spawn – circle or square.

Instances Spawned: The total Prototype instances that have been spawned.

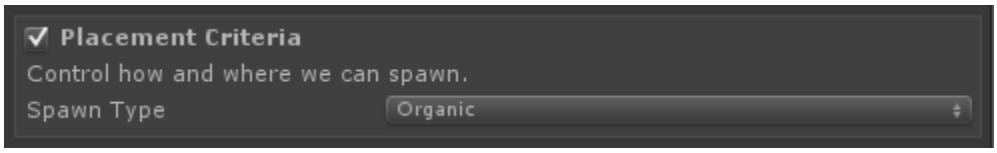
Merge Instances: When spawning prefabs, merge or parent the newly spawned prefabs with the last lot, otherwise create a new top level set of prefabs. BY unchecking this you can see the results of individual prefab spawns.

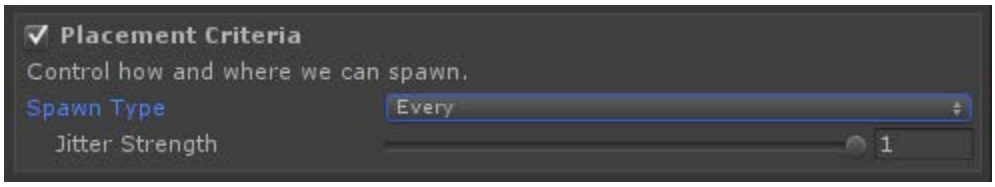
Show Gizmos: Show or hide spawner Gizmos. Hiding can be useful at times.

Placement Criteria Panel

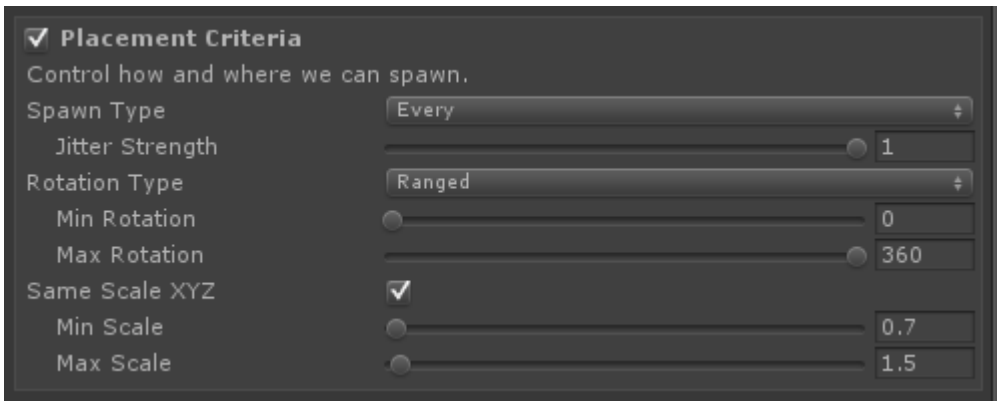
This section controls the algorithms used to select where and how Prototypes are placed into your scene. It will adapt based on how the spawner is configured.

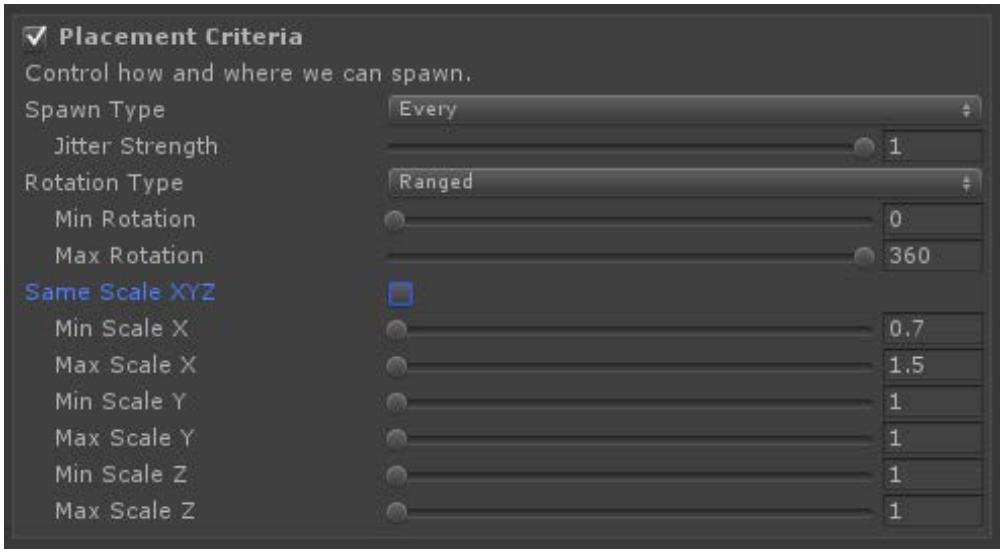
Grass Only:





Trees / Prefabs (or Grass + Image Mask):





Spawn Type: The algorithm used to control where new prototype instances are spawned. All distances are all related to the Throw Distance which is set in the Overview section.

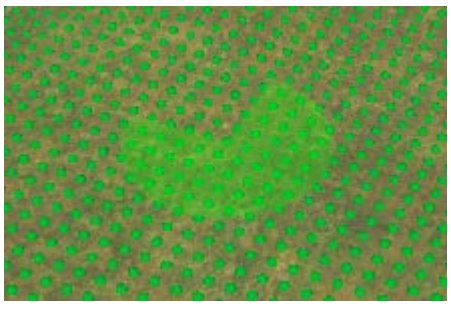
- *Centered*: All spawn attempts will be centered from where you clicked up to the Throw Distance.
- *Every*: Spawns will be attempted at every location in the Spawn Range. Locations will start at the edge of the spawn range and then be incremented by the Throw Distance, plus or minus the Throw Distance * Jitter Strength. Set Jitter Strength to zero for straight lines, and to one for random lines.
- *Organic*: Spawns will be attempted up to the Throw Distance from other successful spawns.
- *Last Spawn*: Spawns will be attempted from up to the Throw Distance from last instance spawned.

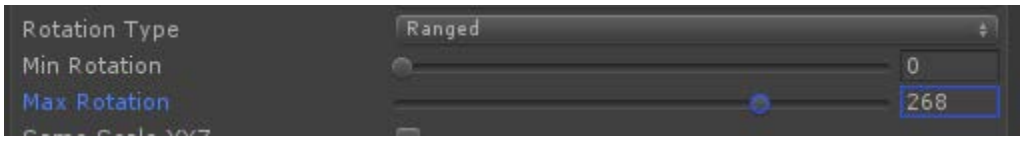
Jitter Strength: Only shown when Every Location has been chosen as the Spawn Type. Controls how much jitter is injected into the selection of the next location. A value of zero will generate straight lines at intervals based on the Throw Distance (defined in the Overview setting), and a value of one will randomise the next location over the throw distance.

Rotation Type: Controls how the Prototype is rotated in the Y axis when it is spawned – i.e. which direction it points in. Individual resources within a prototype will be further offset by adding their own offset to the overall rotation that is applied so that they maintain their rotation relative to the parent prototype.

- *Ranged*: Rotation will randomised between the Min and Max rotation settings.

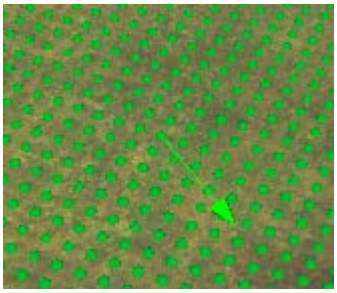
  This is visualised as an arc when there is a range:

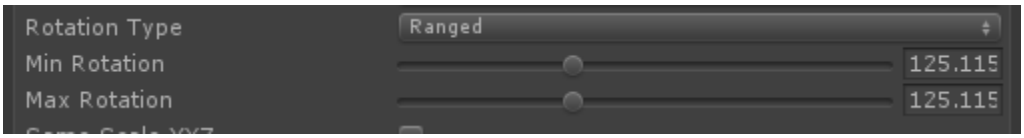




  And an arrow when there is no range – i.e. both min and max rotation are the

same:





NOTE: If you select “Draggable Rotation” in Advanced Settings you can press the Shift Key and then drag the mouse with the left mouse button clicked to easily get exactly the rotation you are looking for.

- *Last Spawn Center*: Spawned objects will be rotated from the centre of the last spawned object to the current spawn location.
- *Last Spawn Closest*: Spawned objects will be rotated from the closest point of the last spawned object to the current spawn location. This will get better results with things like fences. Note: This can only work when the spawned object has a collider.

Same Scale XYZ: When selected will use the same scale in the x, y, and z axis, otherwise will use the specific scale overrides.

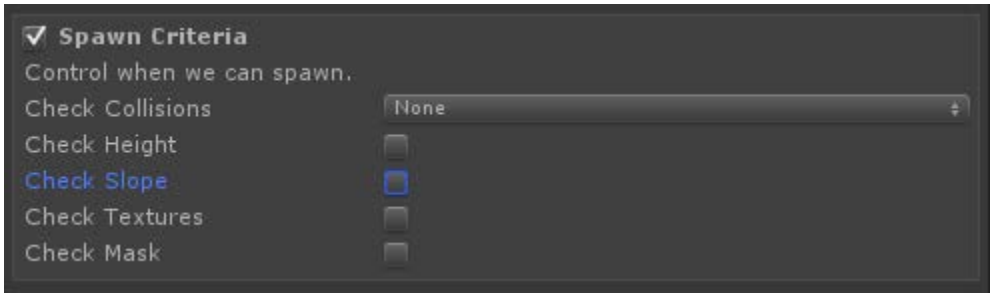
Min & Max Scale: The object spawned will be randomly scaled in this range.

Min & Max Scale X, Y. Z: The object spawned will be randomly scaled in these ranges.

Use Gravity: When selected, new prefabs will be spawned using the Gravity system. When you have finished with Gravity, make sure you turn it off. See the gravity work flow for more information on how to use this.

Spawn Criteria Panel

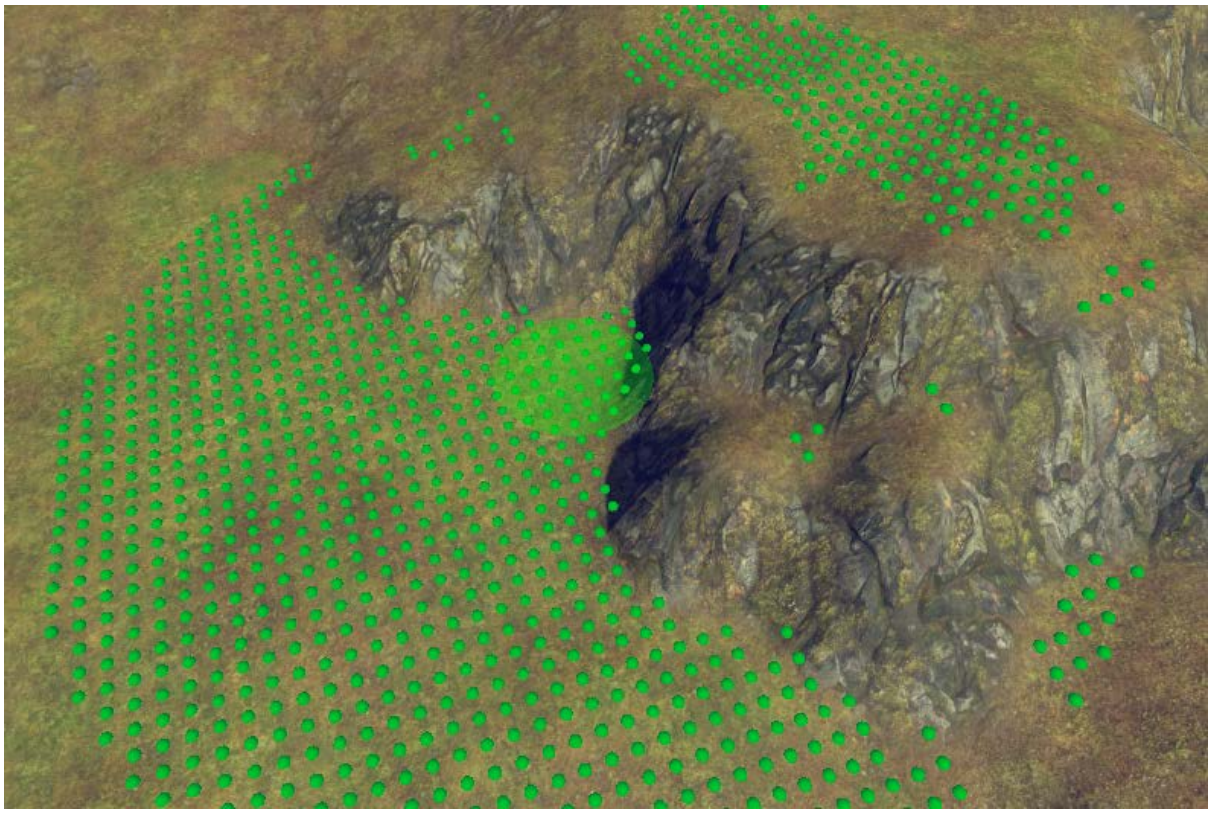
This section controls the when we can spawn.



You can enable as many or as few criteria as you like, and these criteria will only be tested when it has been enabled.

The spawner uses the characteristics of where you clicked on a terrain or mesh to find similar locations on that terrain or mesh, within the ranges you have set in your criterial.

For example, if you click at a height of say 50m, and have a height range of 20m, then it will select for all heights 10m either side of where you clicked, for an overall range of 20m.



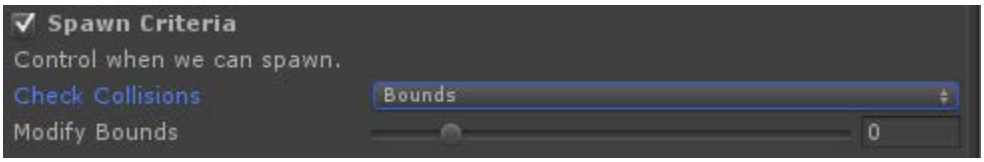
The spawner helps you to understand the significance of your spawn criteria via the visualiser. Hit Shift plus Left Mouse Button to choose a location as your source.

Check Collisions: Controls whether the spawner will spawn its Prototype in a clear space or not. The spawner uses Raycast's to detect collisions and a custom tree management system to determine whether a location is clear to spawn on or not. If you want your object not to have something else spawned within it then then you need to make sure that it has a collider attached (with the exception of terrain trees which use another mechanism to avoid this).

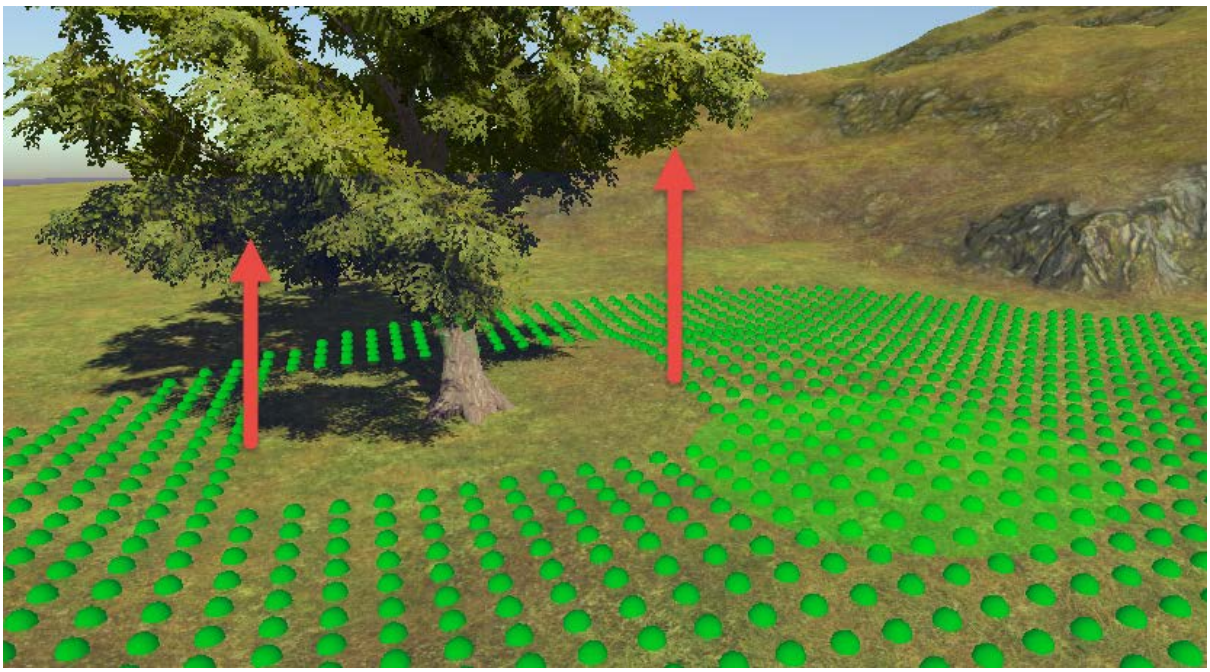
- *None*: Don't check for collisions.
- *Point*: Only check at the point the spawn was initiated. Good for grouping things closely together.

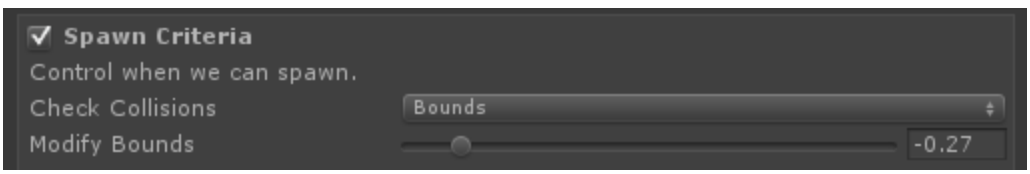
  NOTE: You would typically use this setting with grass and spawn grass as the last thing in your scene. This will stop the grass from being spawned through rocks or buildings.

- *Bounds*: Check the entire volume of the object to be spawned. Only spawns if spawn criteria are met across the entire physical volume of the Prototype.



When you select this Bounds the Modify Bounds option is shown. This enables you to control how these are calculated and is based on the size of the prototype multiplied by this value. You can group things more closely together and potentially spawn overlapping Prototypes by setting it to a negative value, or make sure they are spaced further apart by increasing it from zero. For example, a value of one will only spawn a Prototype if there is at least its own size again of free space in that location – you would use this for things like buildings, where as a value of -0.5 would allow you to clump trees more closely together.



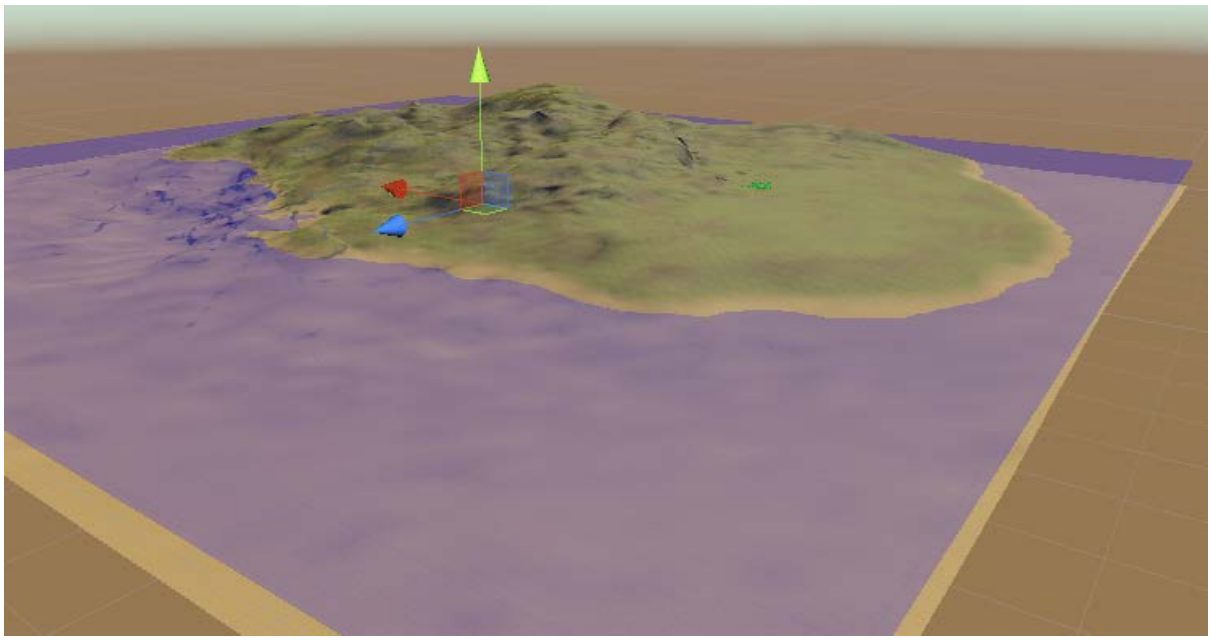


In the example above we can see how bounds testing is avoiding the space around the tree. Notice that the negative bounds setting is being reflected by the visualiser and it showing us that we can spawn within the boundaries of the existing tree.

NOTE: Bounds testing is very CPU expensive as it requires the entire volume occupied by the prototype to be tested to determine whether a space is free to be spawned into. Visualisation does a far less robust test of bounds compared to placing them in the scene to maintain good editor performance. You can modify these resolutions in Advanced Settings section to change this.

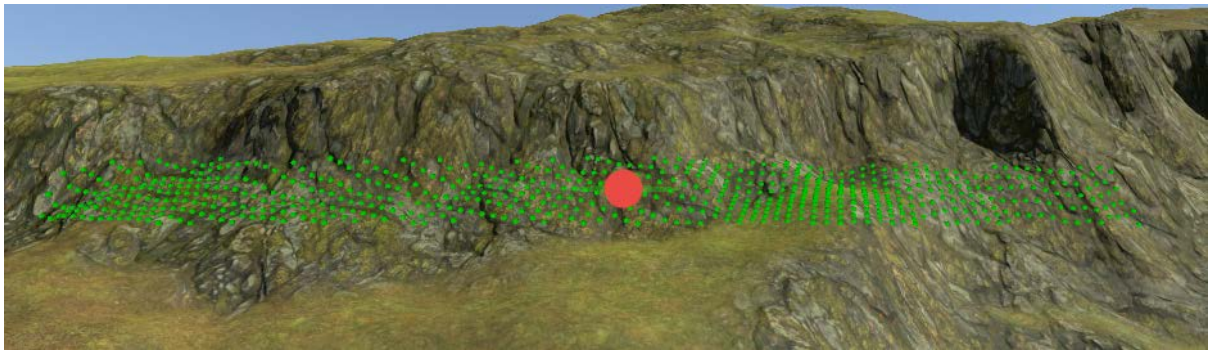
Check Height: Determines whether height is a factor in choosing where the spawner will spawn its Prototype.

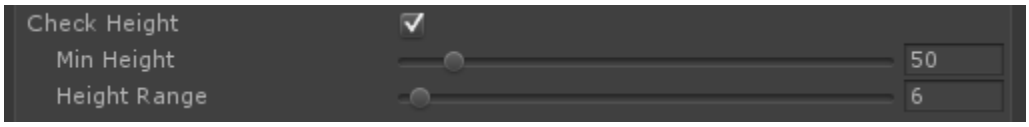
- *Min Height*: The minimum height at which something will be spawned. This can be thought of as your sea level, and nothing will be spawned below this level. It is visualised as a pale blue plane.



TIP: If you find that nothing is spawning, check your minimum height, and lower it if necessary!

- *Height Range*: Spawns only within the height range bisected by the height of the point where the visualisation or spawn was initiated i.e. where you shift or ctrl left clicked on the terrain or mesh.

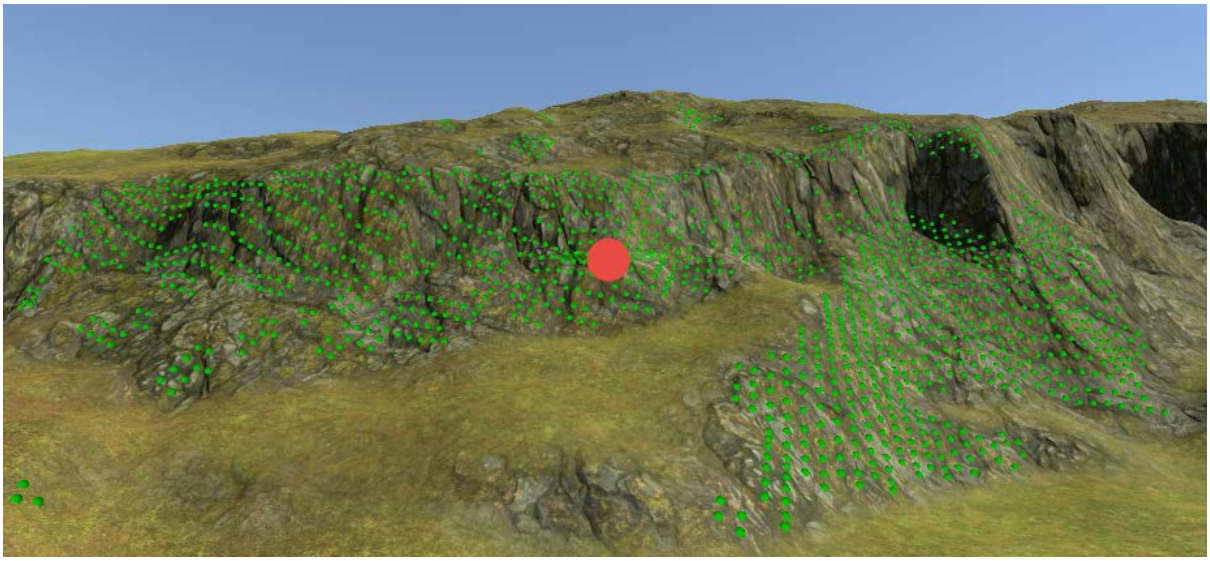




In this example, we have chosen a height range of 6 meters, a minimum height of 50 meters, and you can see approximately where on the terrain the shift click occurred. The visualiser shows the range of possible locations that it will consider for spawning.

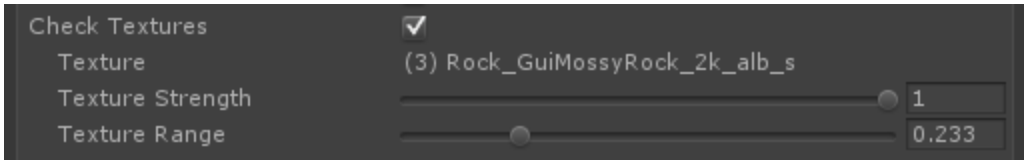
Check Slope: Determines whether slope will be used as a range in which the spawner will spawn its Prototype.

- *Slope Range*: Spawns only within in the slope range bisected by the slope of the point where the visualisation or spawn was initiated i.e. where you shift or ctrl left clicked on the terrain or mesh.

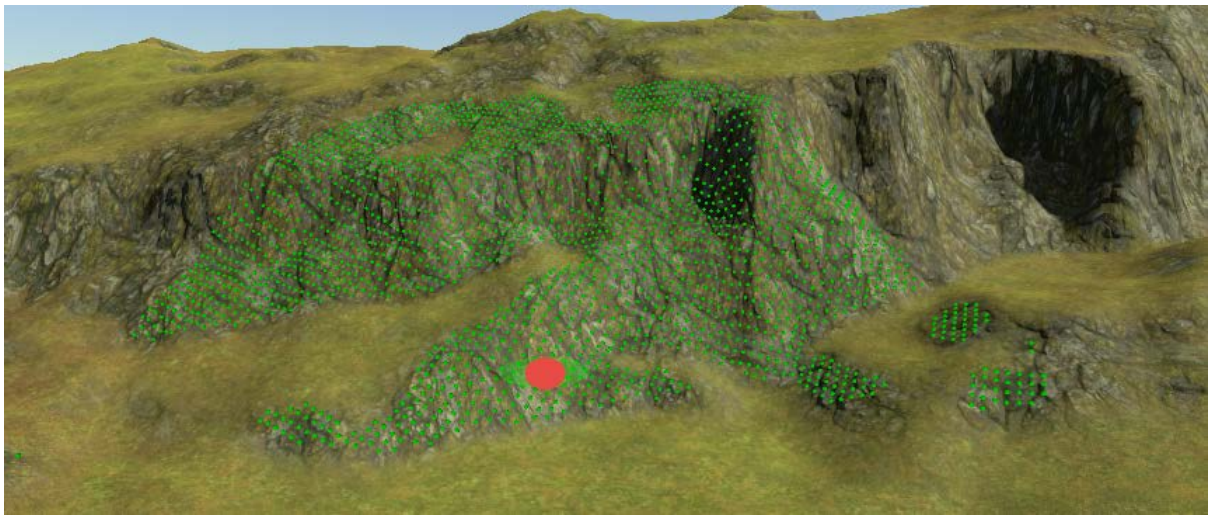


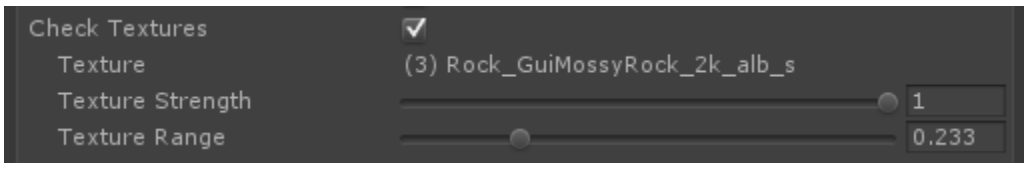
In this example, we have chosen a slope range of 26 degrees, and you can see approximately where on the terrain the shift click occurred. The visualiser shows the range of possible locations that it will consider for spawning that fall within the selected range.

Check Textures: Determines whether the terrain texture will be used as a range in which the spawner will spawn its Prototype.



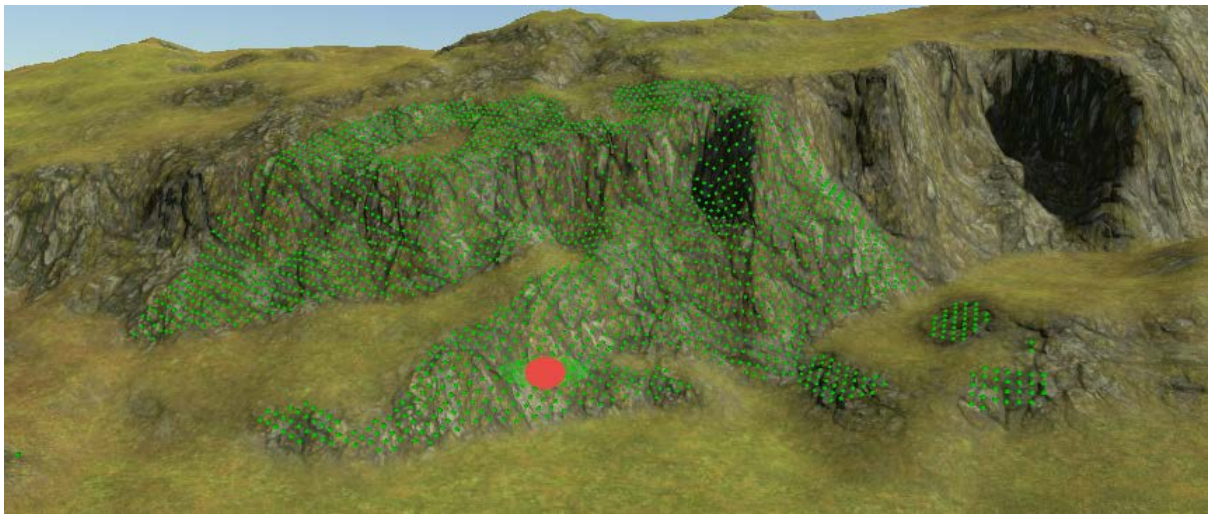
- *Texture*: The name of the texture that was clicked on.
- *Texture Strength*: The strength of the texture that was clicked on. Can be modified to select for other strengths.
- *Texture Range*: Spawns only within in the texture strength range bisected by the dominant texture of the point where the visualisation or spawn was initiated i.e. where you shift or ctrl left clicked on the terrain. This does not work for meshes.

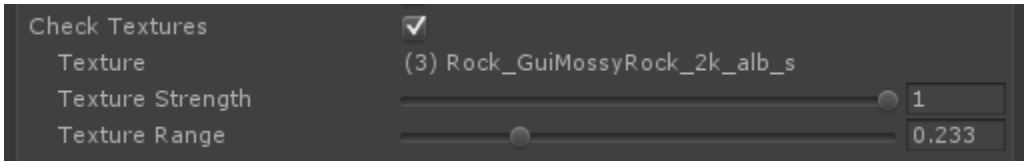




NOTE: You can change the texture strength and range to get some quite subtle effects.

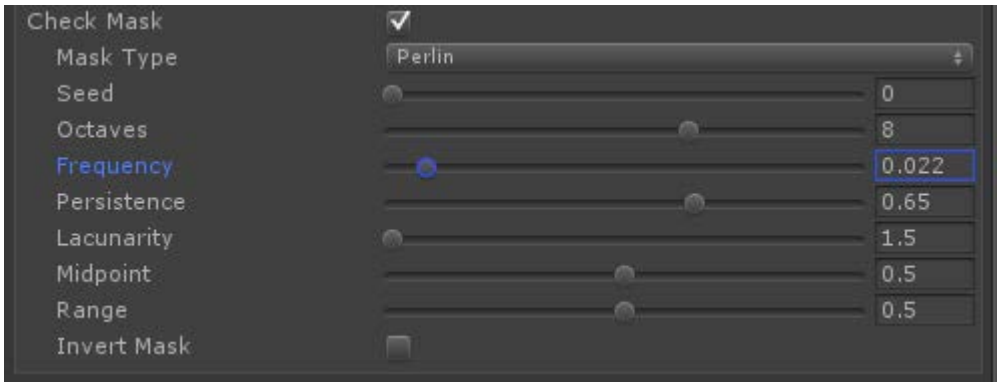
- *Texture Range*: Spawns only within in the texture range bisected by the dominant texture of the point where the visualisation or spawn was initiated i.e. where you shift or ctrl left clicked on the terrain. This does not work for meshes.





NOTE: You can change the texture strength and range that you test for to get some quite subtle effects.

Check Mask: Uses a mask to determine where the spawner will spawn its Prototype.



Noise based masks overlay noise on the spawner at a world coordinate level to determine where Prototypes can be spawned.

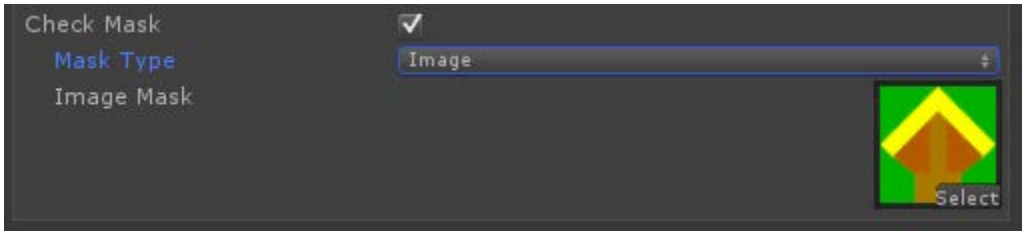


Image based masks use images to determine the location, scale and density of the Prototypes to be spawned.

Mask Type: The type of mask to use. The mask will be reflected in the visualiser.

- *Perlin, Billow, Ridged*: Noise masks applied at world scale.
- *Image*: Image mask applied at spawner scale.

Noise Masks:

- *Seed*: The unique seed.
- *Octaves*: The amount of detail in the noise - more octaves mean more detail and longer calculation time.
- *Frequency*: The frequency of the first octave. Smaller values create larger noise patterns.
- *Persistence*: The roughness of the noise. Controls how quickly amplitudes diminish for successive octaves.
- *Lacunarity*: The frequency multiplier between successive octaves.

- *Mid Point*: The part of the noise curve that is treated as the midpoint of the noise function.

- *Range*: The range around the midpoint used to select the noise for spawning.

Image Masks:

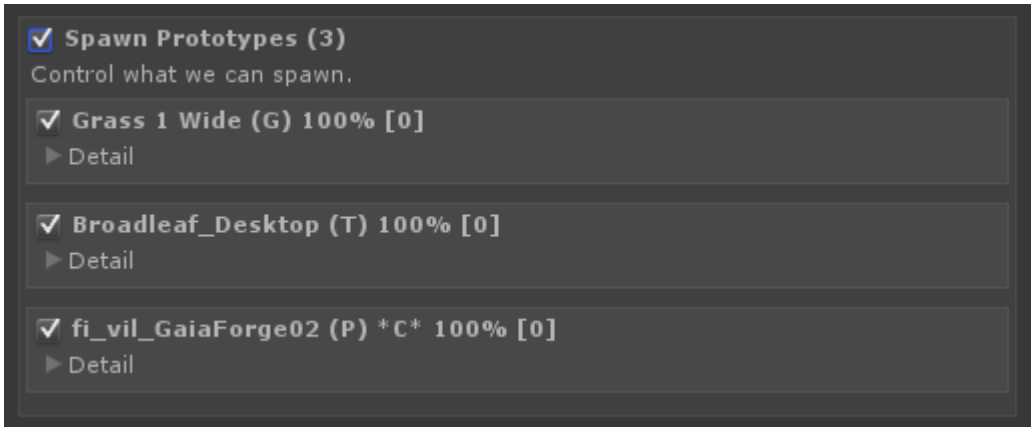
- *Image Mask*: The colour image to be used as a mask. The mask will be scaled to the range of the spawner. The primary colours of the image can have Prototypes assigned to them, and the alpha channel of the image if it exists can be used to control spawn size and density.

  You will select Prototypes against this image in the Spawn Prototypes section (see the relevant image masking section in there).

Spawn Prototypes Panel

This section defines the Protoypes and the settings that control how the spawn into a scene.

A Prototype consists of one or more physical resources that can be spawned into a scene – such as Terrain Grass (details), Terrain Trees, and Prefabs.



Each Prototype will have one entry in the Spawn Prototypes panel.

They are identified as follows:

[Check]	– If checked then the Prototype is active and can be spawned.
Name	– Prototype name, prefabs will be parented under this.
(G/T/P)	– Terrain (G)rass, Terrain (T)ree, (P)refab – type of resource.
C	– Conforms to terrain normal (only for prefabs)
%	– Percent chance of being spawned if selected
[999]	– The number of instances spawned.

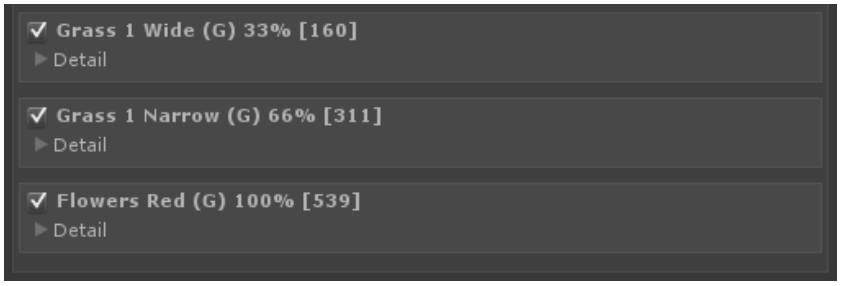
To see the detail of how the prototype has been configured open the detail tab by clicking on the triangle. The detail will vary depending on the type of prototype and whether it is being applied via an image mask.

When a spawn iteration occurs, Gena will choose a location on the terrain, evaluate it for fitness to spawn, and then randomly choose a prototype to spawn there.

When that prototype is chosen to be spawned, a success check is made. If the prototype fails this check, then it will fail to spawn.

To change the relative ratios of how each resource spawns compared to another, change their success rates.

The success factor is shown as a percentage in the Prototype title.

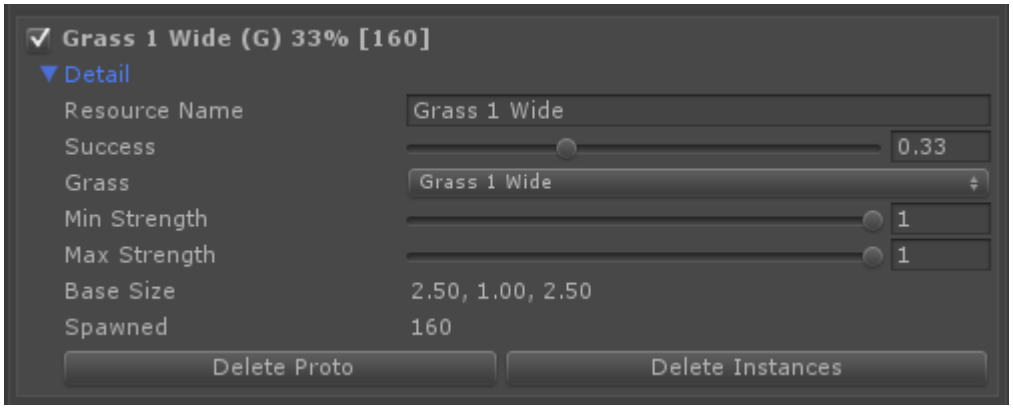


The relative success ratio influences the proportions of each object being spawned, as shown in the instance counts.

To enable or disable different resources to be considered for spawning just click the checkbox next to the resource. Inactive resources will be excluded from the spawn.

Grass Prototypes

Grass Prototypes describe terrain details (I only use them for grass), and how they are applied by the spawner.



Resource Name: Informational name.

Success: Percentage change of a successful spawn. Zero means no change, one means no problems.

Grass: A drop down that interrogates your terrain and allows you to select a grass. You must have previously added the grass to the terrain to select it here. See : https://docs.unity3d.com/Manual/terrain-Grass.html.

Min Strength: The minimum strength of the terrain grass to be spawned. This equates to target strength on the grass control in your terrain settings.

Max Strength: The maximum strength of the terrain grass to be spawned. This equates to target strength on the grass control in your terrain settings.

Base Size: The basic size of a grass instance. An informational value that is pulled from the way it has been applied to the terrain.

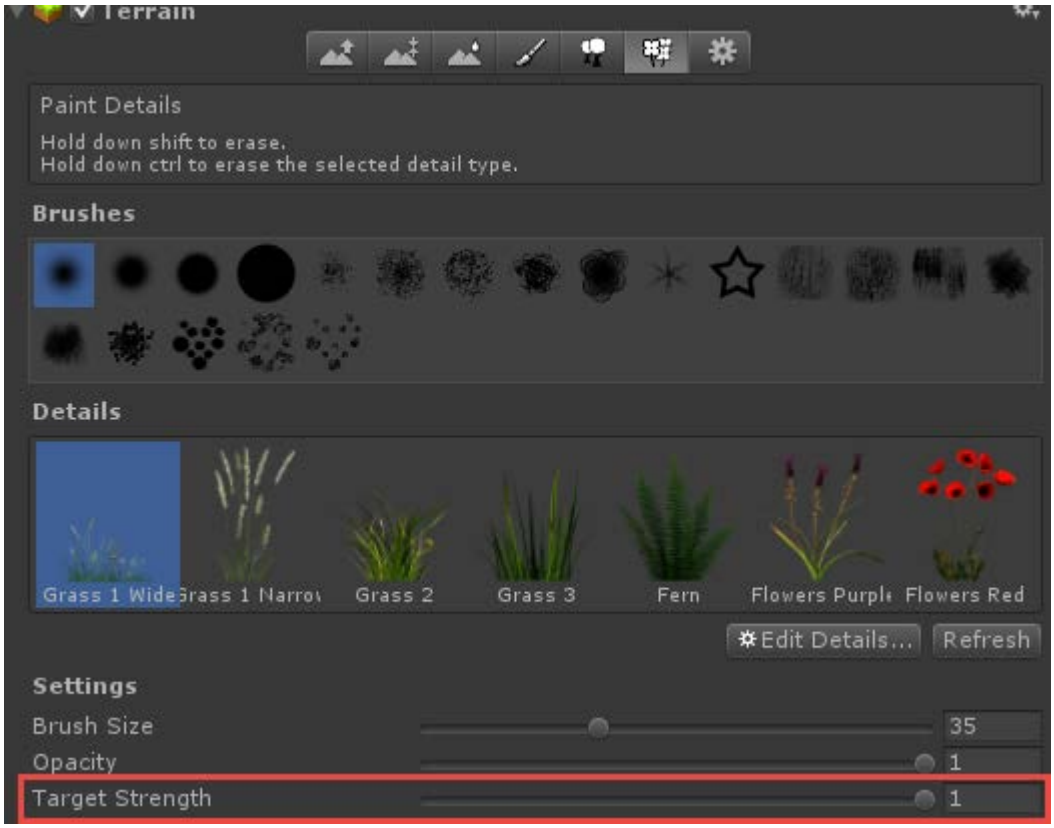
Spawned: The number of instances that have been spawned.

Delete Proto: Delete this prototype from the spawner.

Delete Instances: Delete all instances of this grass from ALL of your terrains.

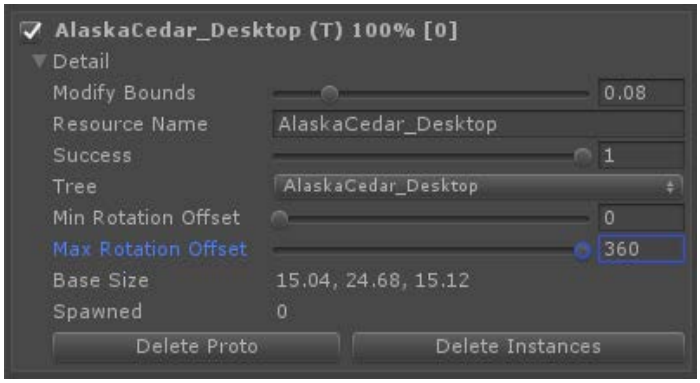
NOTE: This will delete ALL instances of this resource from your scene regardless of whether they were spawned by this spawner or not, so treat with care!

TIP: To determine the best minimum and maximum settings for your grass, go into your scene and paint some onto your terrain at different strengths. Make sure you set the opacity to 1 to get the full impact of the strength settings. Keep the values that look the best for your environment as every grass and environment is different.



Tree Prototypes

Tree Prototypes describe terrain trees, and how they are applied by the spawner. Trees can also be spawned as prefabs / game objects. See next section for more detail.



Modify Bounds: Modifies relative tree volume when bounds based spawning is used.

Resource Name: Informational name.

Success: Percentage change of a successful spawn. Zero means no change, one means no problems.

Tree: A drop down that interrogates your terrain and allows you to select a tree. You must have previously added the tree to the terrain to select it here. See : https://docs.unity3d.com/Manual/terrain-Trees.html.

Min & Max Rotation Offset: When spawning, a random offset between the minimum and maximum value will be added to the rotation selected for the tree in the Placement Criteria. In some scenarios you can want the tree to have a different offset than the one being applied to the spawn, particularly when using draggable rotation with complex POI spawns.

Base Size: The basic size of a tree instance. An informational value determined by instantiating an instance of the tree and calculating its renderer and collider bounds.

Spawned: The number of instances that have been spawned.

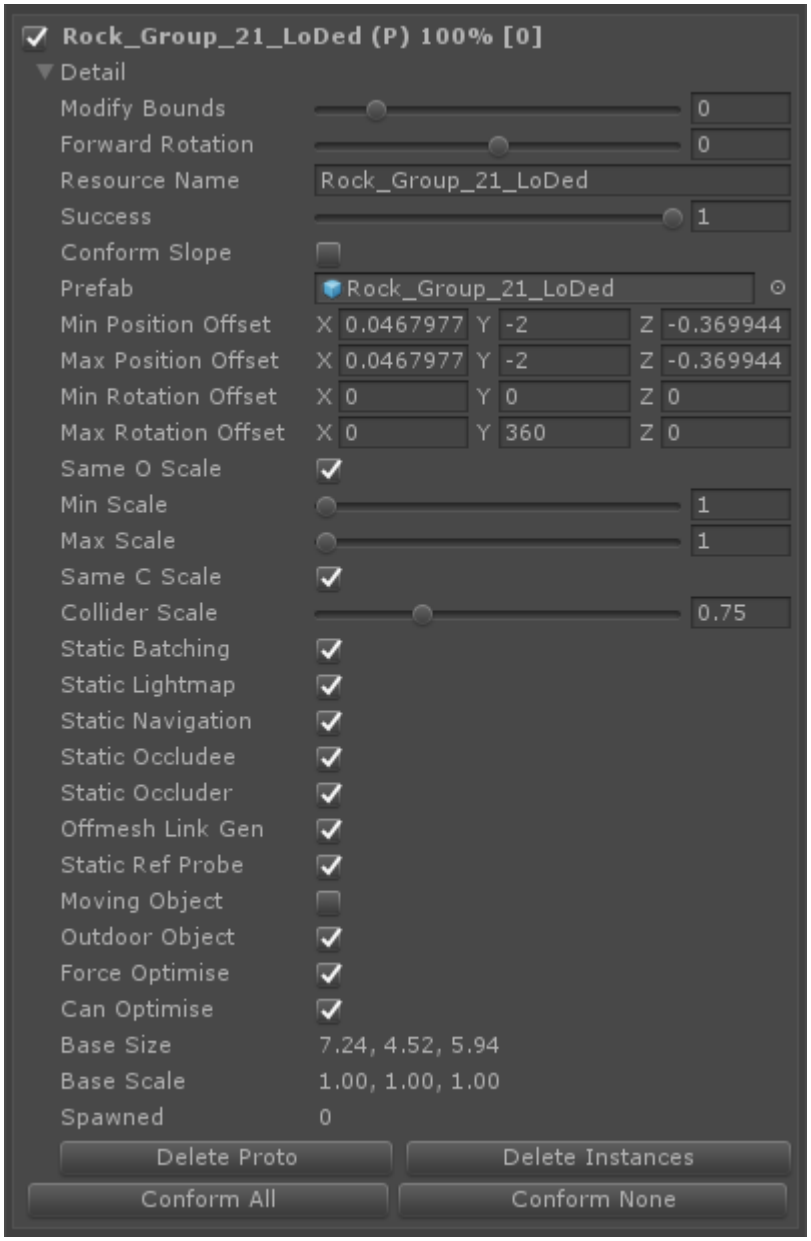
Delete Proto: Delete this prototype from the spawner.

Delete Instances: Delete all instances of this tree from ALL of your terrains.

NOTE: This will delete ALL instances of this resource from your scene regardless of whether they were spawned by this spawner or not, so treat with care!

Prefab Prototypes

Prefab Prototypes describe prefabs or collections of prefabs (structures, villages etc), and how they are applied by the spawner.



Modify Bounds: Modifies relative prefab volume when bounds based spawning is used.

Forward Rotation: An additional rotation that will be added to the rotation in the Placement Criteria so that the prototype will always face in the direction of the arrow when draggable rotation has been selected. Allows you to get very precise

control over how the object will spawn.

Resource Name: Informational name. Used as the name of this prefab as it is spawned into the scene.

Success: Percentage change of a successful spawn. Zero means no change, one means no problems.

Conform Slope: Conform the prefab to the normal of the slope on which it will be spawned.

Prefab: The prefab that will be spawned. This can be any prefab. NOTE: The prefab must have a collider so that GeNa can detect it and stop other objects from spawning in the same space.

Min / Max Position Offset: The offset from the spawn position that will be applied to this prefab when it is spawned. Done as a random range so that an object can be positioned differently from spawn to spawn. Make the min and max the same if you want to always have it spawn in the same spot. Used extensively in Poi based spawning to ensure that individual resources spawn correctly relative to the point that was clicked on the terrain.

Min / Max Rotation Offset: The offset from the spawn rotation specified in the Placement Criteria that will be applied to this prefab when it is spawned. Done as a random range so that an object can rotate differently from spawn to spawn. Make the min and max the same if you want to always have it spawn with the same rotation.

Same O Scale: Apply a constant scale override to the scale of the object spawned, or apply a ranged scale. Spawned scale will be the original object scale multiplied by this scale, multiplied by the scale assigned in Placement Criteria.

Min / Max Scale: Apply a random range for the scale of the object to be spawned.

Same C Scale: Apply a constant scale override to the scale of the collider on the object spawned when using gravity, or apply a per dimension scale.

Collider Scale: The collider scale to apply when spawning objects for gravity.

Static Batching / Lightmap / Navigation / Occludee / Occluder / Offmesh Link / Static Ref Probe: Instance static values to apply when spawning. Can only be

applied in editor mode, not during runtime spawning. See here for more information: https://docs.unity3d.com/Manual/StaticObjects.html.

Moving Object: Used to signify that this prefab can move. This influences the way the automatic optimisation system works. Moving objects will not be batched when spawning using optimisation system.

Outdoor Object: Used to signify whether an object is an outdoor or indoor object when spawning. This influences whether or not the skybox is included in blend probe settings when the automatic optimisation system is used.

Force Optimise: Will force automatic optimisation of this object when the automatic optimisation system has been enabled on this spawner. Otherwise, optimisation will occur only when "can optimise" has been set and the object is smaller than the optimisation size threshold on the spawner in advanced settings.

Can Optimise: Enables this object to be considered for optimisation. Automatic optimisation of this object will occur when the automatic optimisation system has been enabled on this spawner and the object is smaller than the optimisation size threshold on the spawner in advanced settings.

Base Size: The basic size of a prefab instance. An informational value determined by instantiating an instance of the prefab and calculating its renderer and collider bounds.

Base Scale: The original scale values for the prefab instance. An informational value determined by instantiating an instance of the prefab.

Spawned: The number of instances that have been spawned.

Delete Proto: Delete this prototype from the spawner.

Delete Instances: Delete all instances of this resource from your scene.

NOTE: This will delete ALL instances of this resource from your scene regardless of whether it was spawned by this spawner or not, so treat with care!

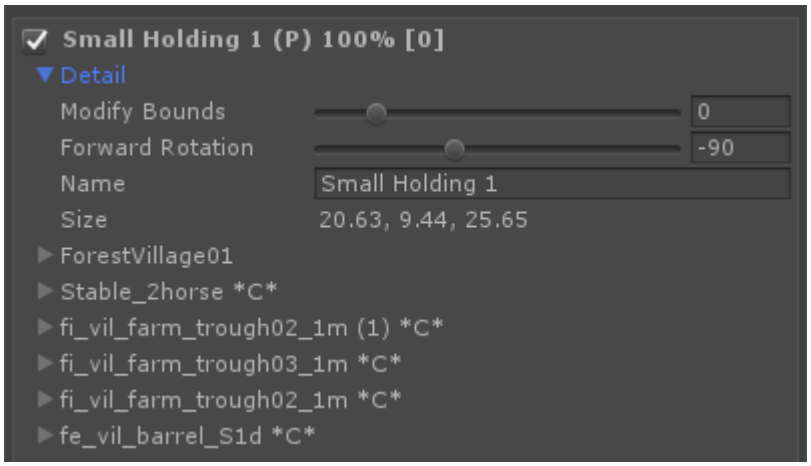
Conform All: Set Conform to true on all resources.

Conform None: Set Conform to false on all resources.

Prefab Prototype Collections (POI's)

When prefabs are added as collections rather than singletons they will be spawned as collections and will maintain their layout relative to each other when they are placed into the environment.

This is a cool way to get things like farms and graveyards. You can design them once, and then place them as many times as you like.

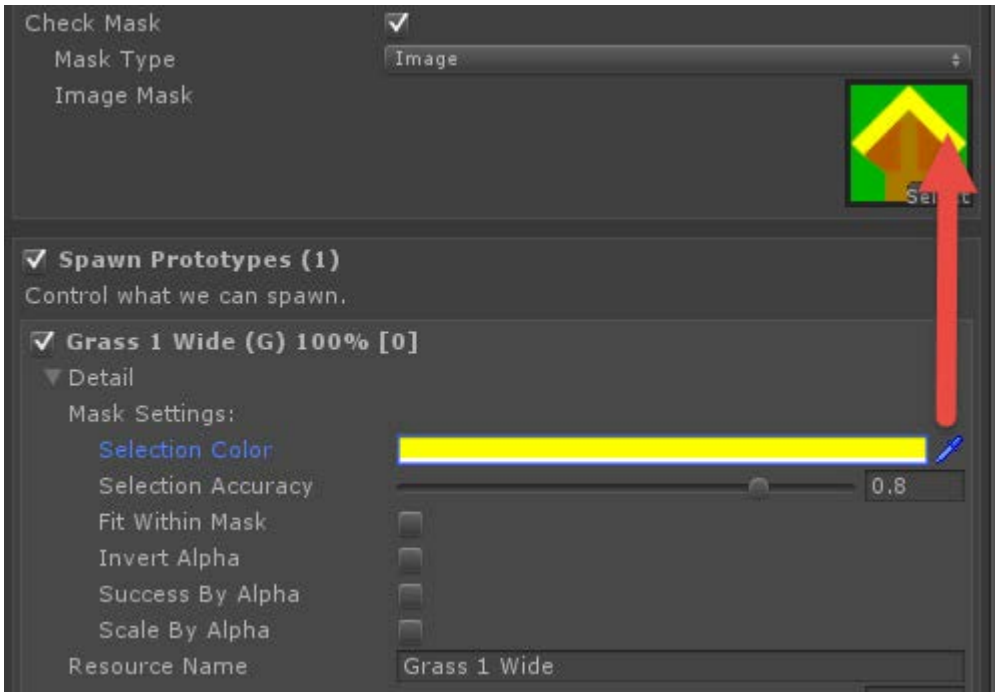


The Forward Rotation setting allows you to pre-rotate the entire POI so that the forward arrow will always generate the right "forward" rotation when draggable rotation has been set in Advanced Settings. This allows for very precise positioning of your POI.

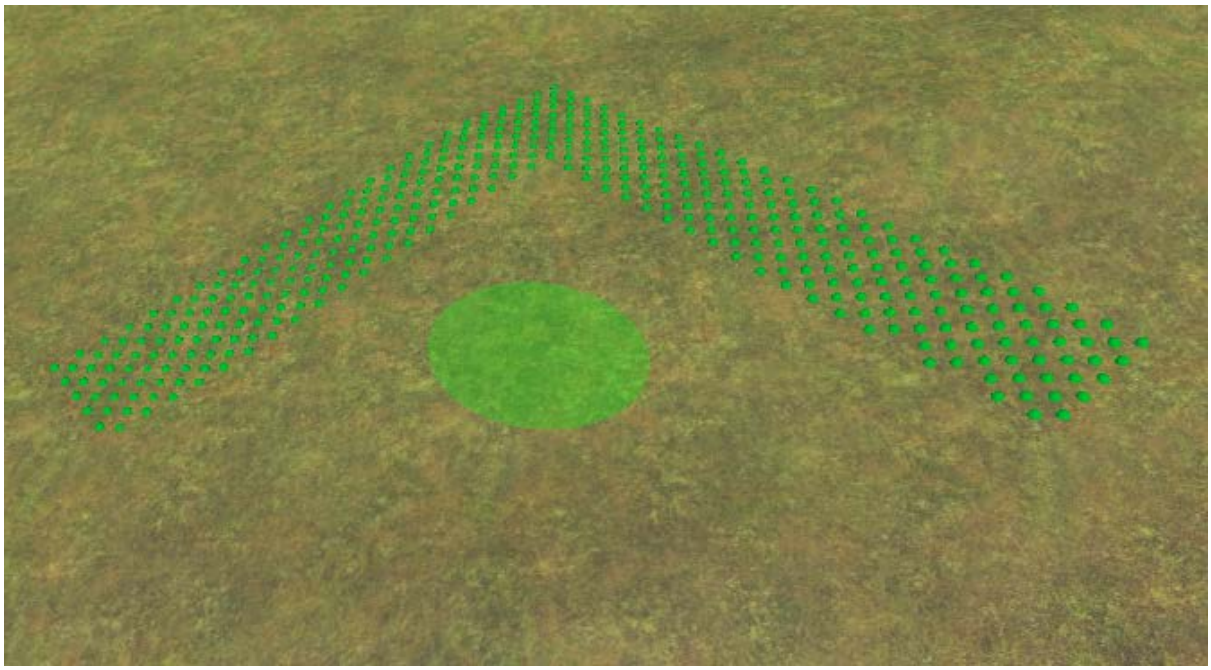
In the image above you can see that this prototype has many resources. You can change the settings of each resource individually – refer to the previous Prefab Prototypes sections for more detail.

Image Masking Settings

When you select choose to use an Image Mask, a new section called Mask Settings will show on your Prototype and allow you to key your Prototype into the colour on the mask that you want to have it spawned on, and then set the rest of the spawn related settings.



A nice way to do this is to select the colour dropper and then click on the preview image on the colour you want to use.



As you key each new Prototype into the mask, it will appear in the visualiser (you may need to shift left click again to see it). You can activate and inactivate individual resources and see the impact of this in your visualiser.

Selection Colour: The mask colour this Prototype will spawn on. Multiple Prototypes can be keyed to the same colour.

Selection Accuracy: The accuracy of the colour selection. In general you should never change this.

Fit Within Mask: When doing bounds based spawning this will ensure that the entire volume of the object being spawned will be contained within the mask. Good for buildings for example, but not so good for trees. It will depend on your use case.

Invert Alpha: Invert the alpha channel. Can be useful when making Prototypes spawn in mutually exclusive places off one mask.

Success By Alpha: The prototype success factor will be influenced by the alpha channel. Works in the same way as the Prototype success control.

Scale By Alpha: The prototype scale will be influenced by the alpha channel. Works in the same way as the Prototype scale control.

Min / Max Scale: The range that the alpha value will be scaled do. An alpha value of zero will be the minimum value, and an alpha value of one will be the maximum value.

Advanced Settings Panel

This section defined the advanced settings for the spawner.

These settings would typically only apply to less common spawner settings.

Force Spawn: Will always force a Prototype instance to spawn at the location you click on. You want to disable this for mask based spawning.

Show Tooltips: Will show or hide tooltips in your spawner.

Show Detailed Help: Will show or hide the help section at the top of the spawner.

Use Large Ranges: Will enable you to set much larger values for large spawns.

Scale Nearest Int: Will force your objects to spawn to a scale that is based on the nearest whole number. Good for some performance scenarios.

Draggable Rotation: Enables draggable rotation mode. In this mode pressing Shift + Left Clicking and dragging with the left mouse button will allow you to get very fine control of the rotation with immediate visual feedback. If your prefab is not facing the direction you expect then modify its Forward Rotation until it is correct.

Add Collider To POI: A sphere collider will be added to POI based spawns to ensure that nothing else will spawn on top of them. This collider will be disabled at runtime.

Add Light Probes: This will cause light probe stacks to be added to the scene when a prefab is successfully spawned. A light prob stack is added ½ a meter above the spawn point, then ½ a meter above the top of the prefab, and then another 5 meters above that to ensure good light coverage.

GeNa will ensure that light probe groups are placed at least "Min PG Dist" meters away from each other, and that individual light probe stacks to be placed at least "Min Probe Dist" meters away from each other.

Spawn Optimizer: This will cause the spawner to optimize the way prefabs are placed during spawning. The optimizer will optimize all resources that fit its criteria. To be considered for optimization the resource must have "Can Opmtimize" set, and be smaller than the "Smaller Than (m)" size (after scaling). Alternatively, if "Force Optimize" is set then it will always be optimised. See the section on Optimisation for more information.

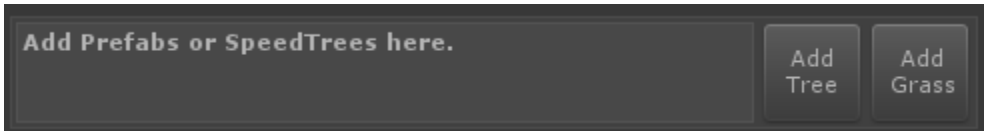
Actual Bounds Res: The resolution that bounds will be calculated at when spawning instances. The smaller this value is the more accurate the bounds checking will be. This will also make the spawn slower as it needs to do more work.

Visualiser Bounds Res: The resolution that bounds will be calculated at when visualising. The smaller this value is the more accurate the bounds checking will be. This will also make the visualiser slower as it needs to do more work. The visualiser will automatically reduce the resolution to keep this process to under 200 milliseconds.

Visualiser Resolution: The dimension of the points that will be checked for every time the visualiser evaluates the terrain. A value of 50 means that 50 x 50 = 2500 locations will be checked every time you update the visualiser – the bigger the value the more time this takes. If the time to calculate this exceeds 200 milliseconds Gena will automatically reduce this. This ensures that the Unity editor remains responsive regardless of the power of the computer it is running on.

Add Resources Panel

This section allows new resources to be added to the spawner.



Add Grass: Click this to add a Terrain Grass Prototype. It will be default always be configured to the first grass in the terrain. Go in and edit the prototype to select the right grass.

Add Tree: Click this to add a Terrain Tree Prototype. It will be default always be configured to the first tree in the terrain. Go in and edit the prototype to select the right tree.

Add Prefabs or SpeedTrees here: Drag and drop Prefabs, or SpeedTree .spm files here to add individual Prefab based Prototypes.

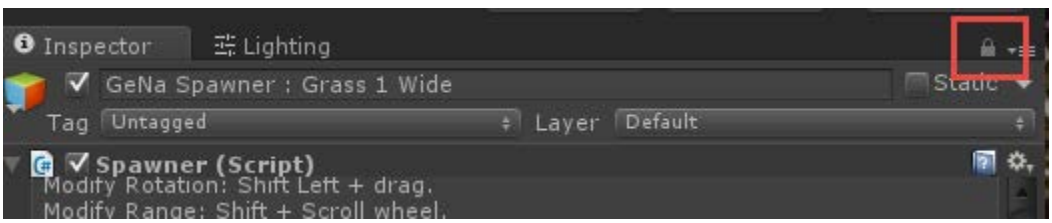
To add structures a work flow needs to be followed.

1. Create a plane and set its Y value to -0.05.

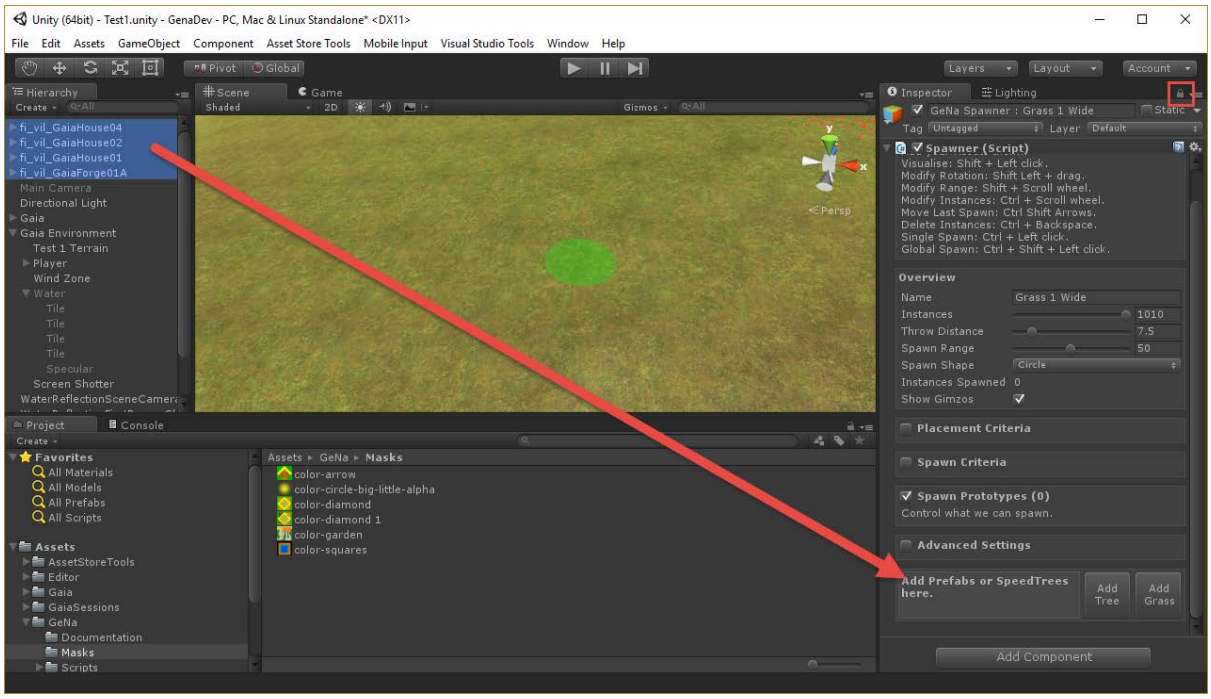
   NOTE: The reason we do this is that Gena spawns all objects relative to the point where the collision hits the terrain or mesh. The Y Offset of the prefab is then added to this, so a value of -0.05 will ensure that it is embedded by 5 centimeters.

2. Lay your prefabs out on that plane in the way that you would like them to be spawned into your scene. As per the previous comment, Gena will take the height the object is placed at and reflect it in its Y offset. In addition to sinking objects into the terrain slightly, you could also make objects float by placing them above the plane and GeNa will reflect this in what it spawns.

3. Select your spawner and lock it in the inspector

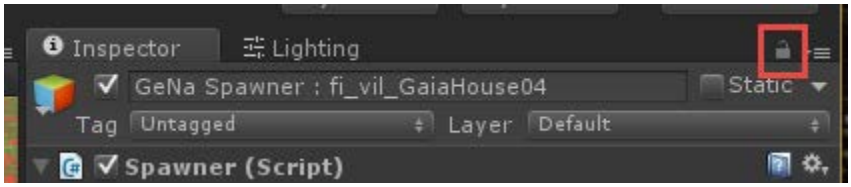


4. Select the objects as a group, and physically drag them all onto the prefab pad.

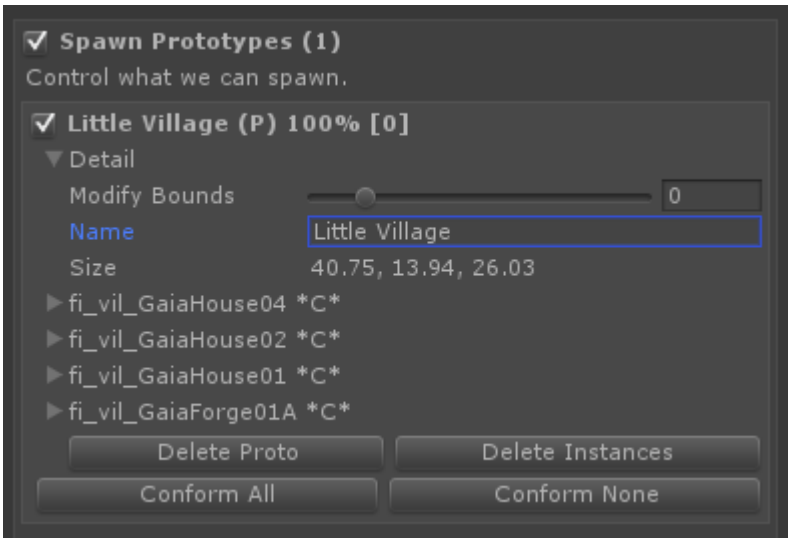


NOTE: Do not child prefabs under other objects. This must be done without any form of hierarchy.

5. Unlock the inspector.



6. Change the name to something that makes sense for when this prototype is spawned into your environment.



7. Go in and fine tune the individual prefab settings. The most common thing you will do will be to set the prefab to conform to the normal of the environment or now. For example buildings would typically be build straight up, whereas paving stones or implements would conform to the slope of the environment they are being spawned in.

Workflows

Terrain Based Workflow

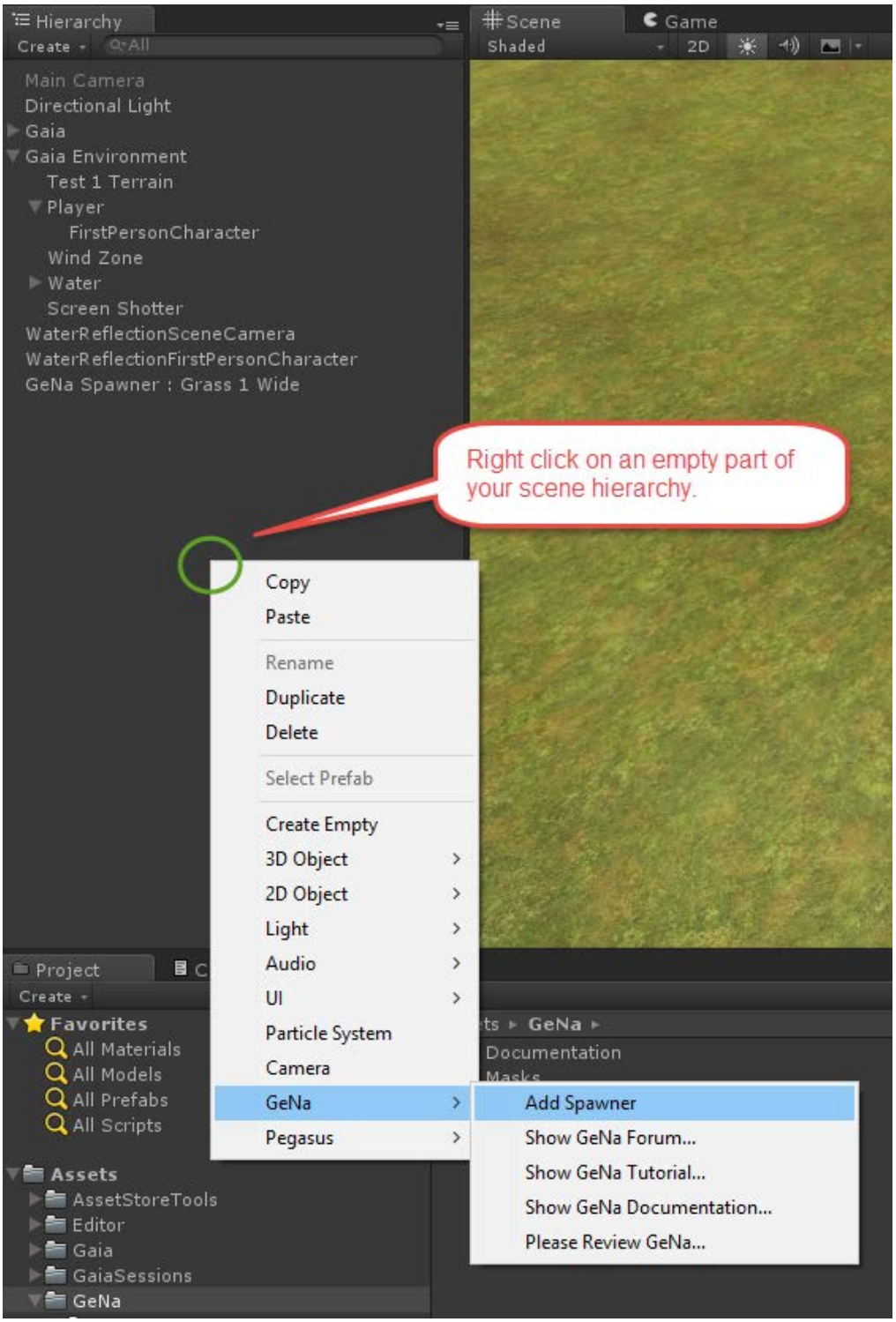
GeNa can place Terrain Details (Grass), Terrain Trees and any sort of Prefab into a Unity terrain, or just Prefabs onto a mesh, and has a collection of algorithms that create different patterns ranging from individual placement, to simulated organic growth to mask and object based layouts.

Because there are so many things that GeNa can do, I will mention them at a high level here, and recommend that you check out the video tutorials on the Procedural Worlds web site, and YouTube channel.

Here is an introduction to the general workflow – in this example we will place some grass into the terrain.

1. Create a Unity terrain with your favourite terrain tool (Gaia is mine ☺), and make sure that you have added some terrain details and trees.
   See: https://docs.unity3d.com/Manual/terrain-Grass.html
   https://docs.unity3d.com/Manual/terrain-Trees.html

2. Add a spawner to your scene by selecting the GameObject->GeNa->Add Spawner menu, or right clicking on an empty section of your Hierarchy window and selecting GeNa->Add Spawner.



NOTE: It does not actually matter where the spawner is located as it is merely a worker. GeNa does its work based on where you click in your scene.

3. To add Terrain Detail (Grass), click on the Add Grass button near the bottom right of the spawner. This will pick the first grass object in your terrain and add it as a Prototype into the spawner.

   NOTE: Every resource or collection of resources that can be spawned as one instance is called a Prototype. A prototype can consist of terrain grass, terrain trees, a prefab, or a collection of prefabs. When you have collections of prefabs within a single prototype this is considered a single entity, and spawned as a single instance. This an awesome way of spawning things like farms or graveyards in which you want to maintain the overall way the objects have been laid out as a group.

4. Check the Spawn Prototypes panel – the number in brackets is the number of prototypes that this spawner is managing.

5. Open the Spawn Prototype section by clicking on the Spawn Prototype check box. You will see the Grass that was added, and a detail drawer.

   Pay attention to the title of the prototype. It shows the Prototype Name, Type (G) Terrain Grass, (T) Terrain Trees or (P) Prefabs, the success rate of the spawn represented as a percentage and the number of instances that have been spawned.

6. Click on the detail drawer triangle to see the grass details.

7. Let's explore the detail parameters for this grass prototype.

   a. The name of the resource will be reflected in the name. You can change this if you want – on grass and trees it won't have much of an effect, however on prefabs this will influence the name of the prefab spawned into the scene.
   b. The next option is the chance of a successful spawn. This is a value in the range of zero to one, that acts to additionally reduce the chance that this resource will be spawned. If you find that you are getting too much of an individual resource being spawned you can reduce this to thin it out.
   c. The next option is the Grass itself. This is a drop down that picks up the grasses that have been applied into the terrain via the terrain settings. You can easily change the Grass that will be spawned by choosing another grass from this dropdown.
   d. Min and Max strength relates to the strength of this grass as it is applied to the scene. The is the same as the Target Strength for that grass, as it would be applied to the scene if you were to use the terrain grass paintbrush. You can apply the grass to the scene manually with the terrain grass paint brush to get a sense of how the strength settings would be applied when this grass is spawned into the scene. The actual strength that will be applied will be some value between the minimum and maximum strength.

e. Base Size – this in an informational display that is pulled from the grass in the terrain itself. It is used to influence the throw distance of the grass when spawning it into the scene. You can think of the throw distance as the distance that a seed would be thrown by this grass when it reproduces.
f. Spawned – this is the number of grass instances that have been spawned into your scene.

8. Choose the grass that you would like to spawn into the scene by choosing it in the Grass dropdown.

9. When you are happy with the grass that you have selected then close this up by unchecking the Spawn Prototypes panel.

10. Now let's explore the Overview settings section at the top of the spawner.

    a. Name – This is the name of the spawner. Changing this will change the name of the spawner in the scene hierarchy.

    b. Instances – This is the number of prototypes that the spawner will attempt to spawn on every spawn iteration. You will generally want more than one blade of grass per spawn, so change this to suit your purposes.

       You can control the number of instances by editing them directly, or by pressing Ctrl and then using the scroll wheel on your mouse to increase or decrease them.

       NOTE: GeNa will try to make reasonable guesses about the starting point for this, but you can override this as you want. If you want to enable very large numbers of instances to be spawned per iteration then go into the Advanced Settings panel and enable 'Use Large Ranges'.

    c. Throw Distance – This is the distance range that each additional grass location will be chosen based on the Spawn algorithm used. Grass tends to work well with the Organic spawn algorithm or the Last Spawn algorithm.

    d. Spawn Range – This is the maximum distance from the location that you clicked that this spawn iteration will be able to spawn instances. Try increasing and decreasing it by pressing the Shift Key and scrolling the mouse wheel.

       When global spawning is chosen, the same limitation will be applied, at each location that the global spawn initiated.

       Note: If you want a larger potential spawn distance from where you

clicked, then enabled ‘Use Large Ranges’ in the Advanced settings tab.

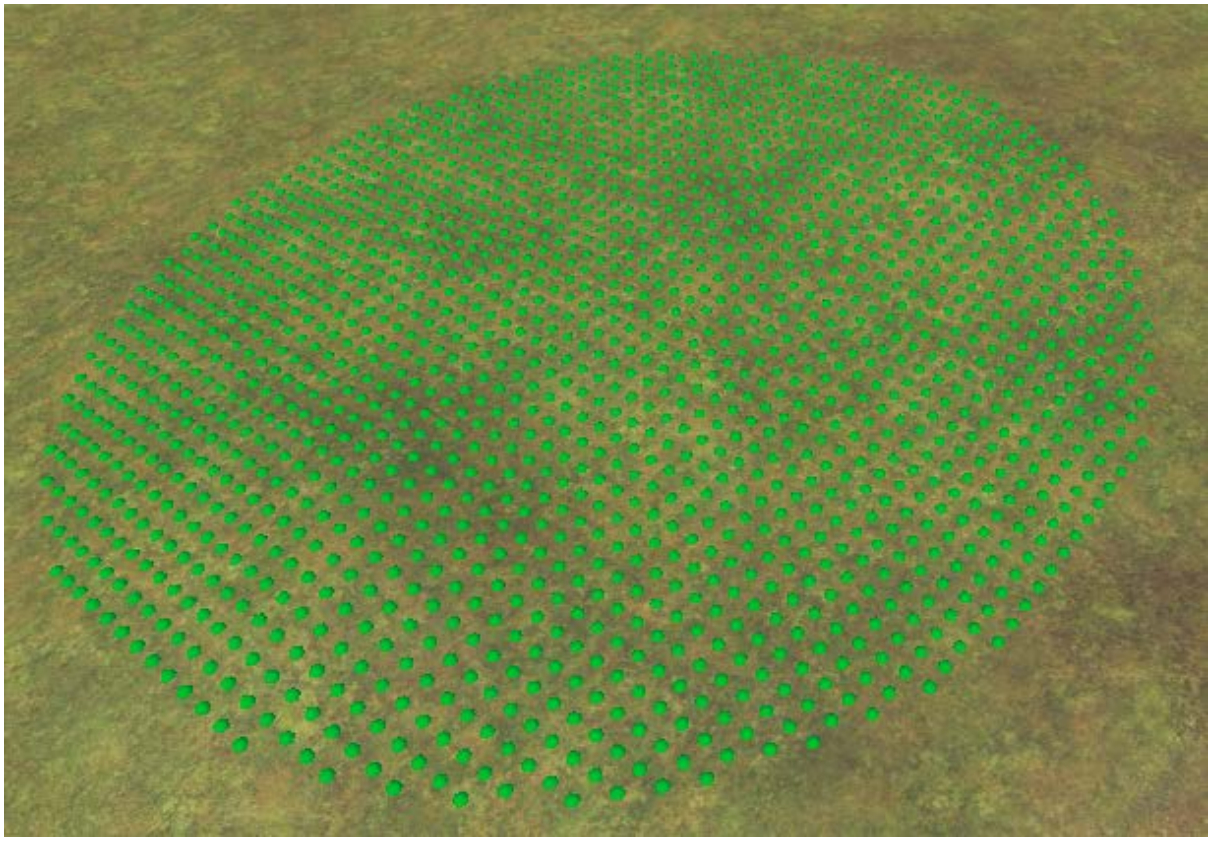
e. Spawn Shape – this is the general shape that will be used to limit the spawn.

f. Instances Spawned – this is the total number of Prototype instances that have been spawned by this spawner.

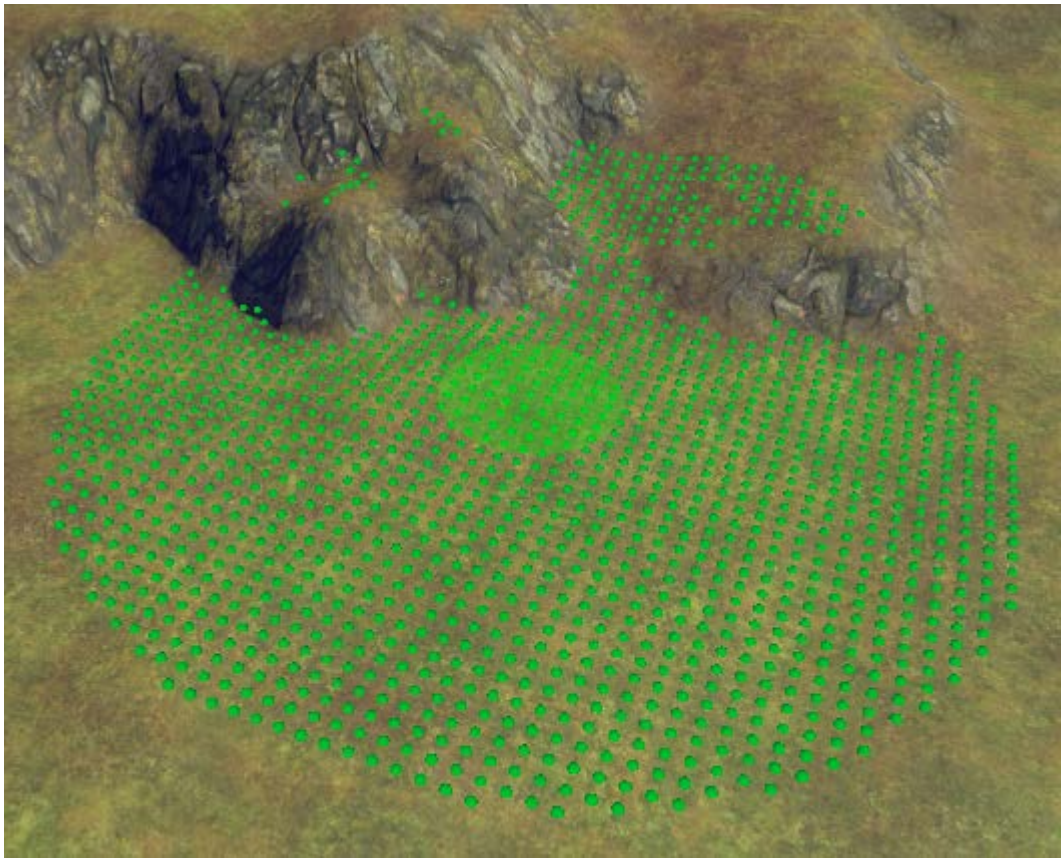
g. Show Gizmos – click this to show or hid your gizmos.

11. To see where the grass will spawn in the scene, making sure that your Spawner object is selected in your scene hierarchy, now press the Shift Key and while holding it down, press the Left Mouse button.

    This will display a visualisation of where your grass can potentially spawn. Not that this is just a high-level visualisation to give you a sense – specific locations will depend on your environment.

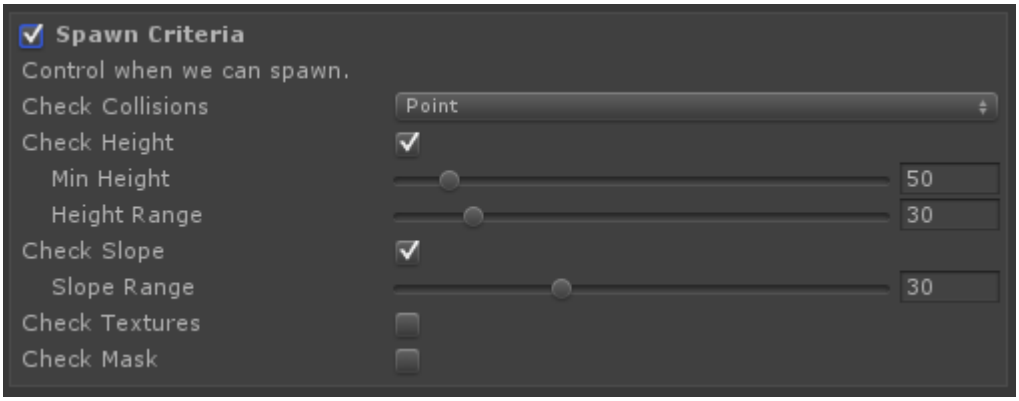


    The green dots represent areas that can be spawned on.



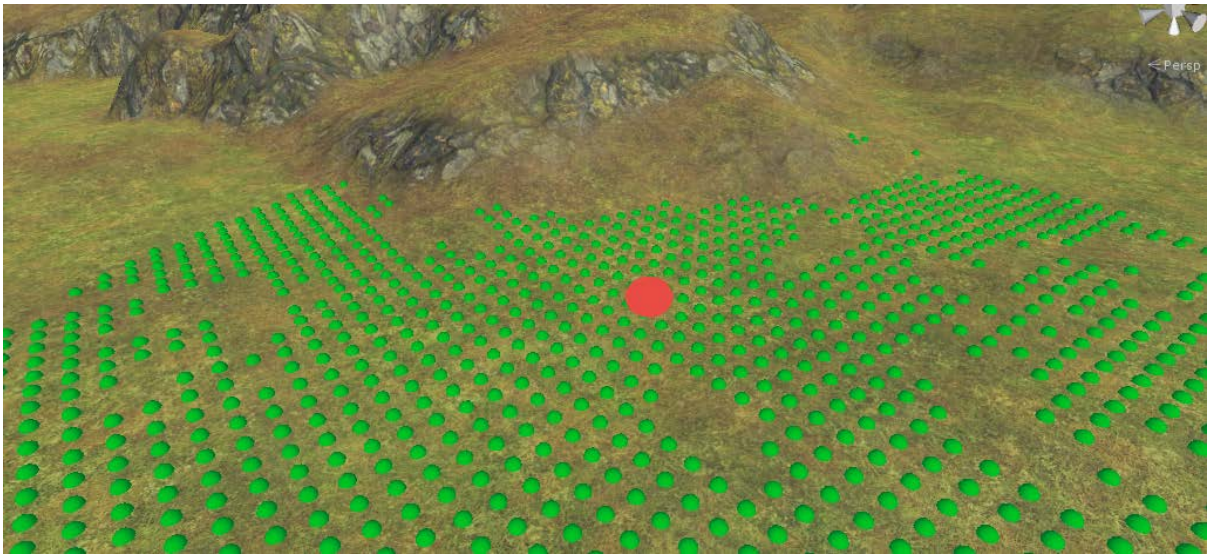
In this image I have clicked near a cliff and you can see that the cliff cannot be spawned on. The areas that are deemed suitable to spawn on are controlled by your Spawn criteria, so let's look there next.

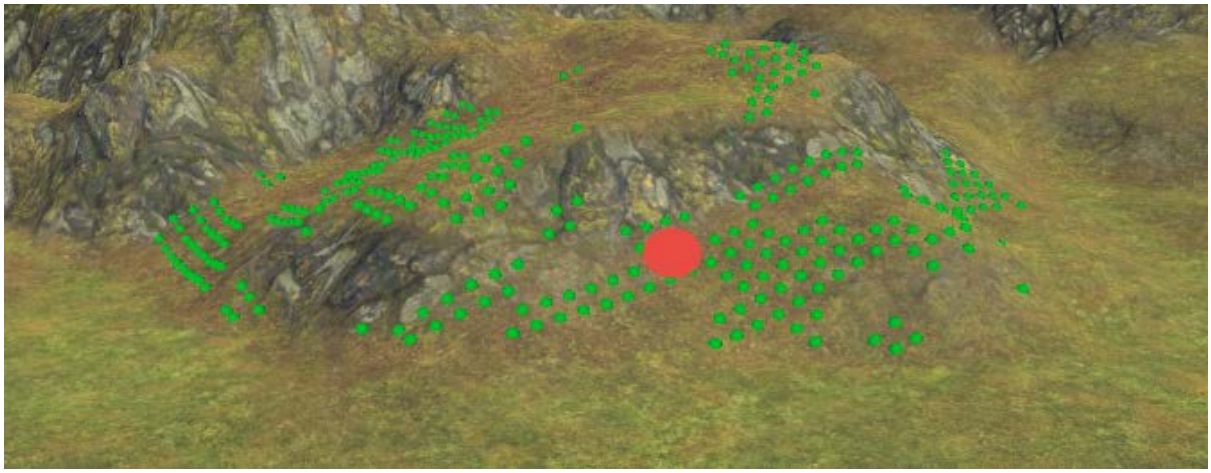
12. Check the Spawn Criteria panel to open it.



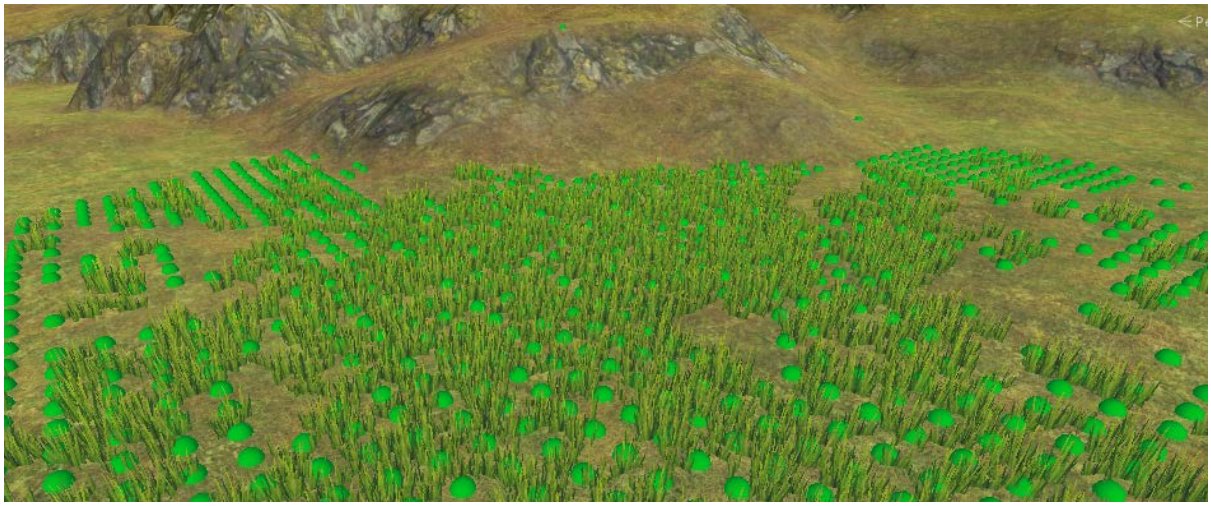
The values here control how the spawner interprets your scene. This is reflected in real time by the visualiser. Go ahead and change the slope range… make sure you are near some cliffs to get the best effect.

Shift click around in the scene to see how the slope range and the actual place you click on affects where the grass will be placed. ***This is a key to way that GeNa works. GeNa will always base its selection on where you click in a scene. Click on a flat area it will select for flat, click on a slope and it will select for slopes.*** This dramatically and intuitively speed up your level design.





13. If there are any good locations then they will show in green and you can then go ahead and Ctrl + Click to place the Prototype into your scene. Make sure you Ctrl + Click in the same place you Shift + Clicked as GeNa always evaluates based on where you clicked. Notice how the grass stays within the bounds indicated by the visualiser.



14. When you are done you can save your spawner as a prefab for later use, and it will retain all the configuration settings you created.

Mesh Based Workflow

Mesh based work flow – i.e. the process of spawning onto meshes is the same as the terrain based workflow with the exception that you cannot spawn terrain grass or terrain trees.

Set up your spawner and then shift click to visualise, and ctrl click to place. The target that you will spawn your prefabs on will be the mesh that you clicked on.

Runtime Workflow

GeNa can also run spawn iterations at run time as well via the sample script provided or the API's provided.

The sample "RuntimeSpawner.cs" script is a good example to see how the spawner API's should be called.

The typical work flow is to configure your spawner first, optionally save it as a prefab, and then instantiate it if necessary and call it at run time.

The Spawning process is a two-step process:

1. First sample the environment at a given location so that the criteria can be set for "like" places on the environment to be used for spawning. This is like the Shift Click operation.

   ```
   SetSpawnOriginAndUpdateRanges(Transform groundObject, Vector3 location, Vector3 normal)
   ```

2. Run either a local or a global spawn iteration based on the previously selected sample. This is like the Ctrl + Click or Shift + Ctrl + Click operations.

   ```
   Spawn(Vector3 location, float rotation, bool subSpawn)
   SpawnGlobally()
   ```

Note: You can pre-configure the first step via the Shift or Ctrl click processes when you are setting up your spawner, so the first step is optional. If not used then the original sample criteria will be used instead.

Have a look at the runtime spawning scenario for a practical example and code that shows this in action.

Prefabbing / Reuse Workflow

GeNa spawners and their configurations can be saved as prefabs, and then re-instantiated and used like the original objects.

The great benefit of this is that you can save your spawner configurations and then re-use them.

Treat them as normal prefabs and the other operations and workflows described in this document will continue to work as normal.

APIs

Gena can be controlled directly via it's API's.

The aspect of Gena that is controlled from an API perspective is implemented in the spawner class.

```
public class Spawner : MonoBehaviour
{
    Spawner variables - public so that they can be accessed via editor

    Pre-Spawning / spawner initialisation methods

    Spawning methods

    Unspawning methods

    General helper methods

    Colour conversion utilties used by image masking

    Gizmos
}
```

The suggested approach is to configure the spawner prior to usage via the API's and then either call it directly at runtime, or save it as a prefab and then instantiate it and call it at run time.

The Spawning process is a two-step process:

1. First sample the environment at a given location so that the criteria can be set for "like" places on the environment to be used for spawning. This is like the Shift Click operation.

   ```
   SetSpawnOriginAndUpdateRanges(Transform groundObject, Vector3 location, Vector3 normal)
   ```

2. Run either a local or a global spawn iteration based on the previously selected sample. This is like the Ctrl + Click or Shift + Ctrl + Click operations.

   ```
   Spawn(Vector3 location, float rotation, bool subSpawn)
   SpawnGlobally()
   ```

Rather than document every API here, where it will be immediately outdated, please open the relevant section in the Spawner.cs file, as every function has been documented.

```
#region Spawning methods

/// <summary>
/// Run a spawn instance across the entire target object
/// </summary>
1 reference | 0 changes | 0 authors, 0 changes
public void SpawnGlobally()...

/// <summary>
/// Run a spawn instance at this point in the terrain with the specified rotation
/// </summary>
/// <param name="location">Location to spawn at</param>
/// <param name="rotation">Overall Y rotation to be spawned at</param>
/// <param name="subSpawn">Set to true if this is a sub spawn</param>
1 reference | Adam Goodrich, 3 hours ago | 1 author, 1 change
public void Spawn(Vector3 location, float rotation, bool subSpawn)...

/// <summary>
/// Run a spawn instance at this point on the terrain
/// </summary>
/// <param name="location">Location to start from</param>
/// <param name="subSpawn">Set to true if this is a sub spawn</param>
6 references | Adam Goodrich, 3 hours ago | 1 author, 1 change
public void Spawn(Vector3 location, bool subSpawn)...

/// <summary>
/// Instantiate the prototype at the location provided
/// </summary>
/// <param name="prototype">The prototype of be spawned</param>
/// <param name="location">The location to spawn at</param>
/// <param name="normal">The normal at the location to spawn at</param>
/// <param name="alpha">The alpha value at the location - can be used to overide some aspects of the spawn</param>
/// <param name="scaleFactor">The scale factor to apply to what has been spawned</param>
/// <param name="rotation">The overal rotation to apply to what has been spawned</param>
/// <param name="spawnAtLeastOneResource">Force at least one instance to be spawned</param>
/// <param name="spawnedInstance">The spawned instance</param>
/// <returns>True if something was spawned</returns>
3 references | Adam Goodrich, 3 hours ago | 1 author, 3 changes
private bool PaintPrototype(Prototype prototype, Vector3 location, Vector3 normal, float alpha, Vector3 scaleFactor, float rot
```

The internal variables and their effect on the spawn output have been documented elsewhere in this document.

For an example of runtime API usage take a look at “RuntimeSpawner.cs”.

Usage Scenarios

This section details some of the common usage scenarios for GeNa, and makes some suggestions on how to get better results.

Video Tutorials

You can view these scenario's in conjunction with the latest video tutorials at:
http://www.procedural-worlds.com/gena/tutorials/

Spawning Grass

1. Add your grasses to your terrain.
2. Add a spawner and add your grasses to the spawner.
3. Go through each prototype and make sure the right grass is selected.
    a. [Optional] Enable a Perlin based mask in your Spawn Criteria panel. Change the frequency to change the sizes of the grass patches. Change the midpoint and range to explore more of the fractal.
4. Spawn as usual.

TIP: The noise generator generates noise in world space, and this means that multiple spawners can use it and expect it will generate the same mask. Try setting up two spawners with the same noise settings, and one, select Invert Mask. This will invert the generated pattern and you can use this to spawn one grass in one section, and another grass in the inverted section. The overall effect of this is a nice grassy field, and the interesting aspect of this is that you will get nice patches of alternating grasses as well. This tip applies to all types of spawning.

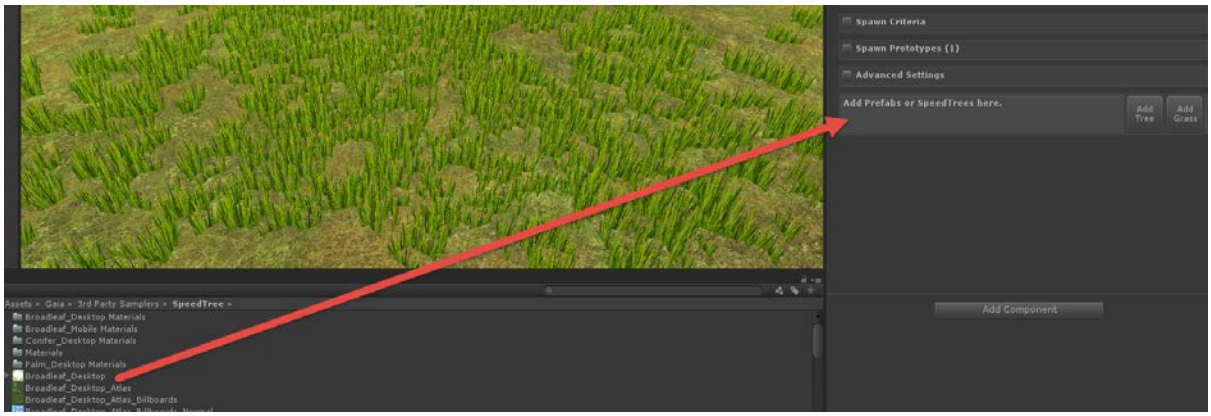
Spawning Terrain Trees

1. Add your trees to your terrain.
2. Add a spawner and add your trees to the spawner.
3. Go through each prototype and make sure the right grass is selected.
    a. [Optional] Enable a Perlin based mask in your Spawn Criteria panel. Change the frequency to change the sizes of the tree patches. Change the midpoint and range to explore more of the fractal.
4. Spawn as usual.

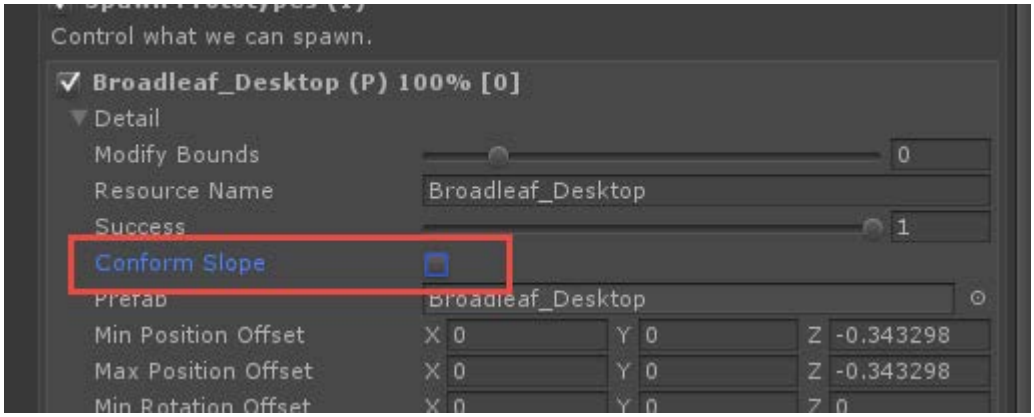
TIP: The noise generator generates noise in world space, and this means that multiple spawners can use it and expect it will generate the same mask. Try setting up two spawners with the same noise settings, and one, select Invert Mask. This will invert the generated pattern and you can use this to spawn one tree in one section, and another tree in the inverted section. The overall effect of this is a nice populated terrain, and the interesting aspect of this is that you will get nice patches of alternating trees as well. This tip applies to all types of spawning.

Spawning SpeedTrees As Prefabs

1. Add a spawner and drag and drop SpeedTree .spm file onto the spawner.



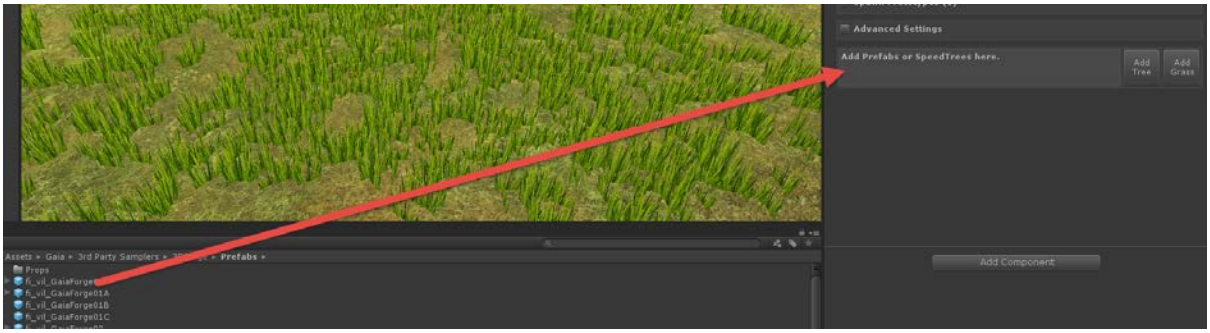
2. Consider turning off Conform To Slope as most trees grow straight up.



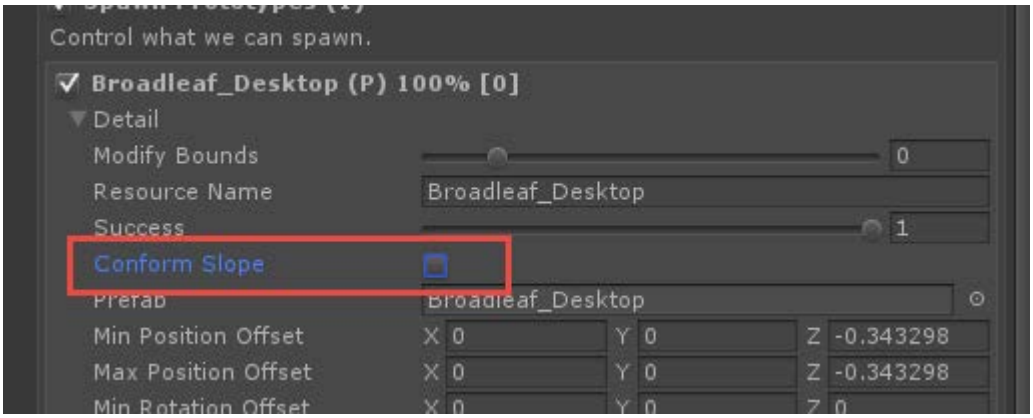
3. Spawn as usual.

Spawning Prefabs

1. Add a spawner and drag and drop a prefab onto the spawner.



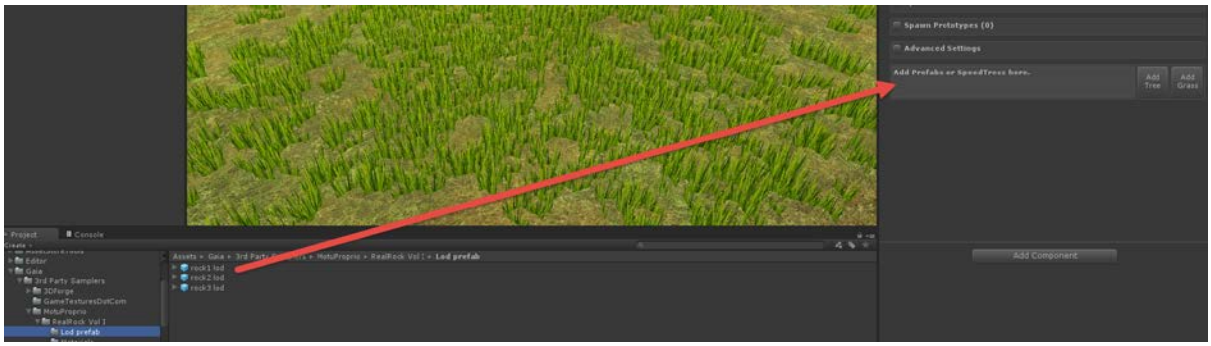
2. Consider turning off Conform To Slope. Depends on the prefab.



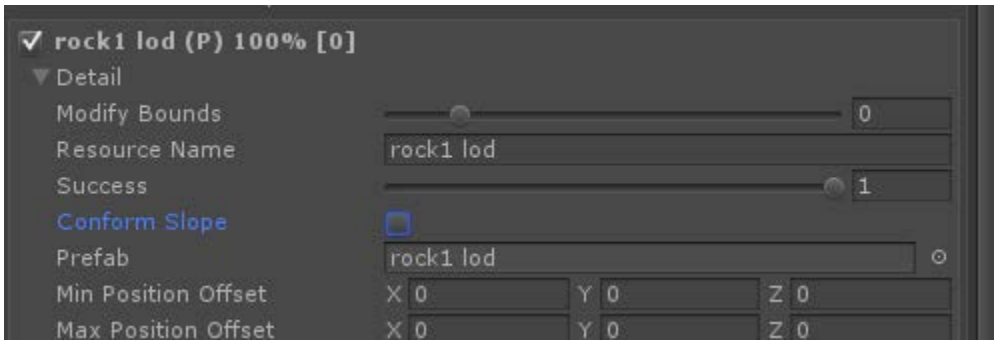
3. Spawn as usual.

Spawning With Gravity

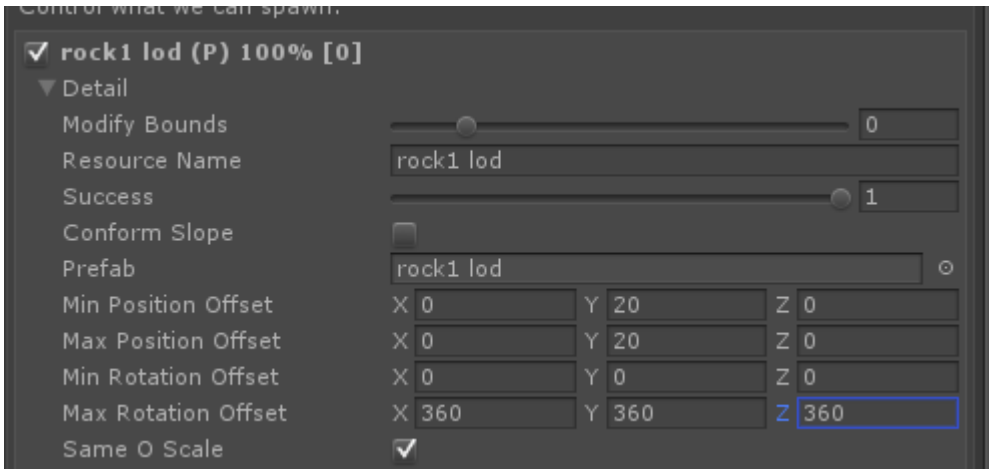
1. Add a spawner and drag and drop a prefab onto the spawner.



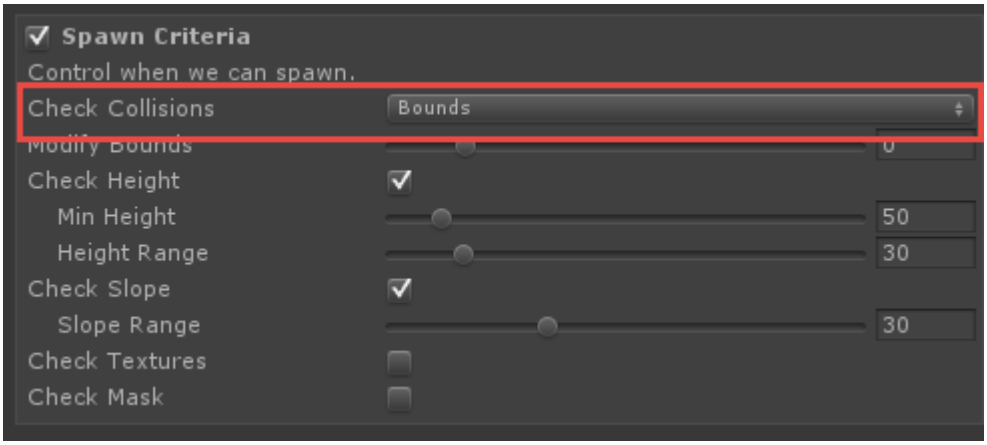
2. Consider turning off Conform To Slope. Depends on the prefab.



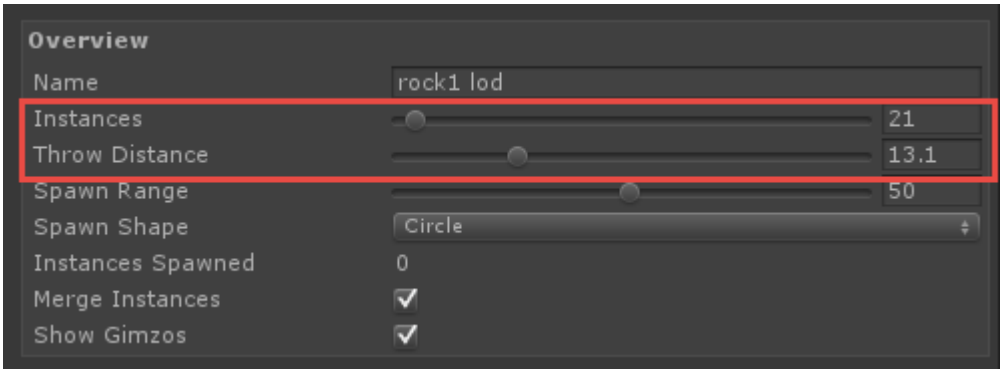
3. Set the Min and Max Position offset to a value that will ensure it spawns above the terrain... let's say 20 meters. Let's also set the max rotation offset so that they spawn with random rotations.



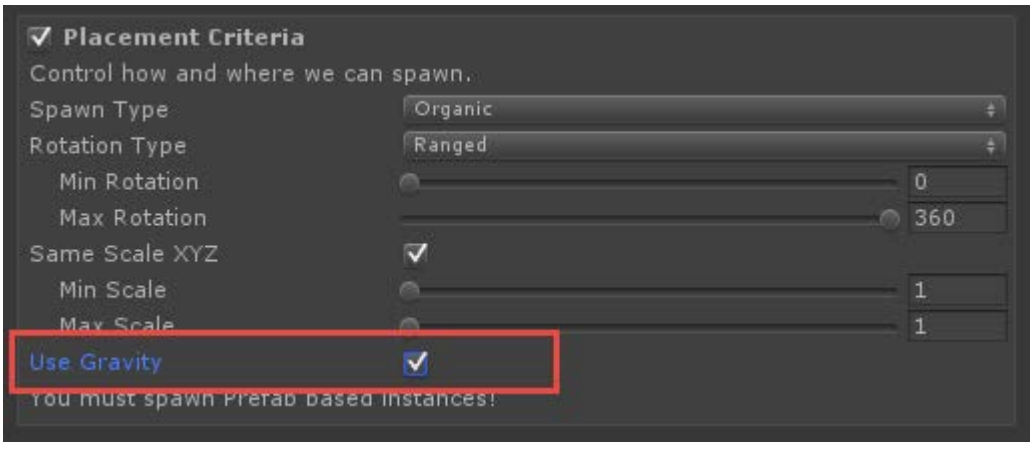
4. Ensure that bounded spawning is selected.



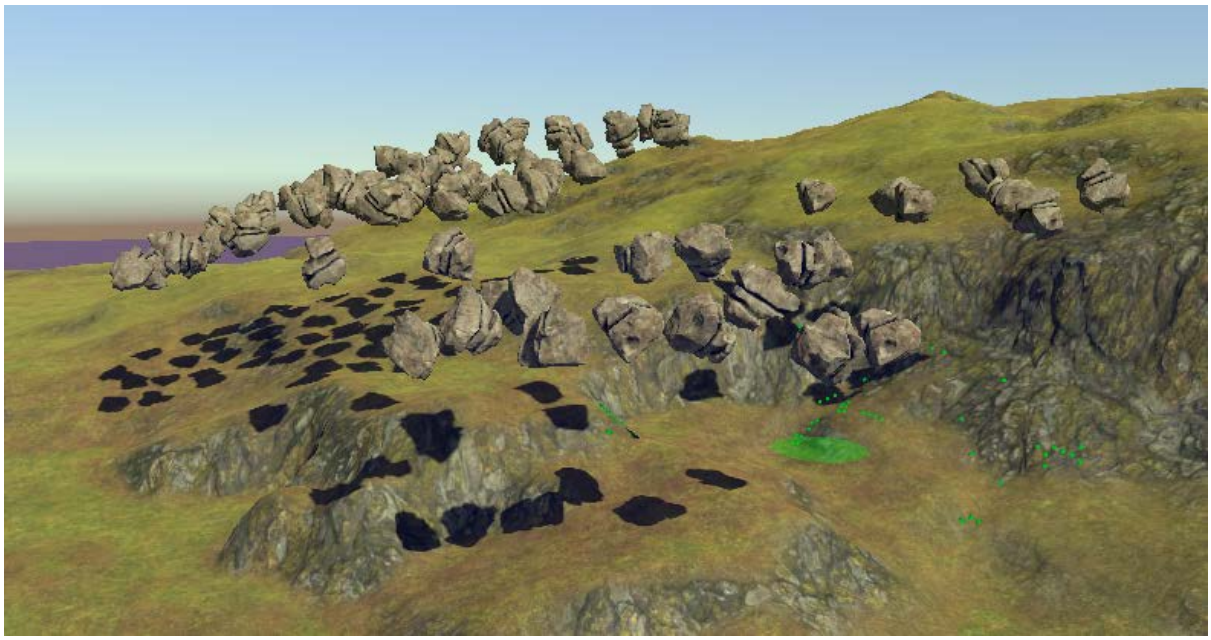
5. Dial in the instances and throw distance so that we can spawn a bunch of them.



6. Make sure you enable Gravity in your Placement panel. This must be done before spawning anything into your scene.

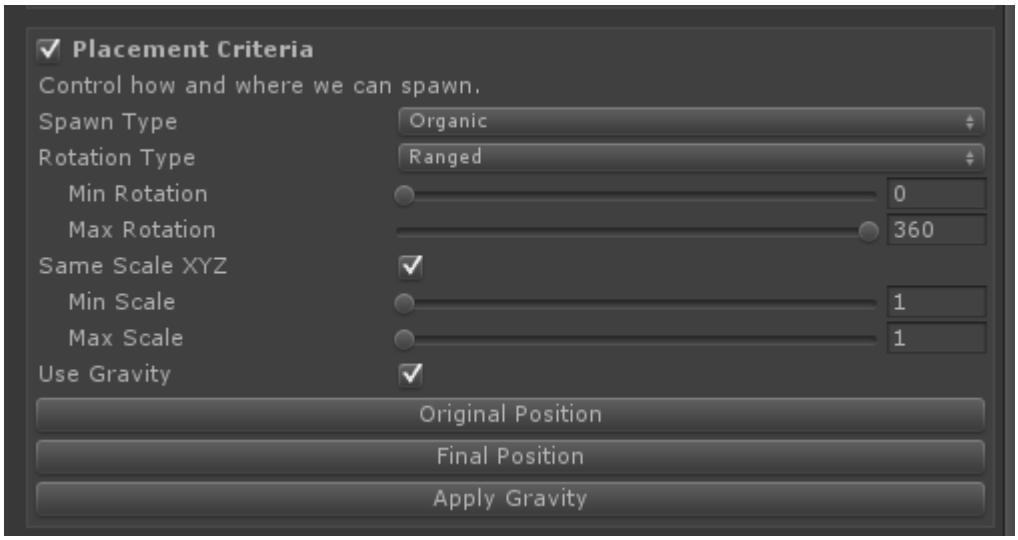


7. Hit Ctrl + Left Click a few times to spawn a bunch of them into your scene. It's more fun if you do this on a hill ☺



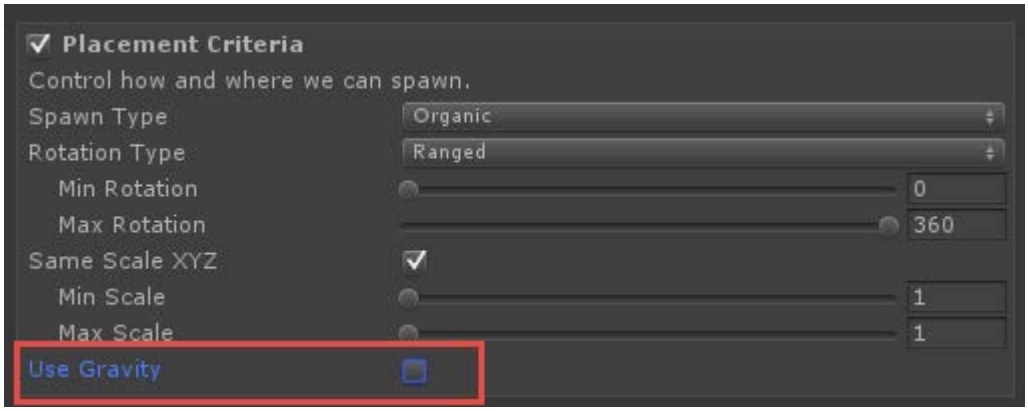
8. Press play and watch the fun. Wait until all the rocks have come to rest before stopping playback.

9. Now click between Original Position and Final Position to see the before and after the application of Gravity.



10. When you are done hit the Apply Gravity button. This will respawn the objects in their final locations without all the things that were added to enable gravity.

11. Finally, unselect Gravity. This will disable the run time gravity system.



Spawning Sophisticated Structures

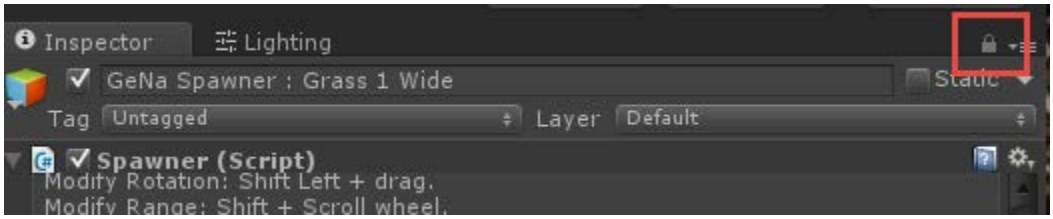
This is one of the more powerful abilities of GeNa. It allows you to take sophisticated structures and spawn them into the environment.

1. Create a plane and set its Y value to -0.05.

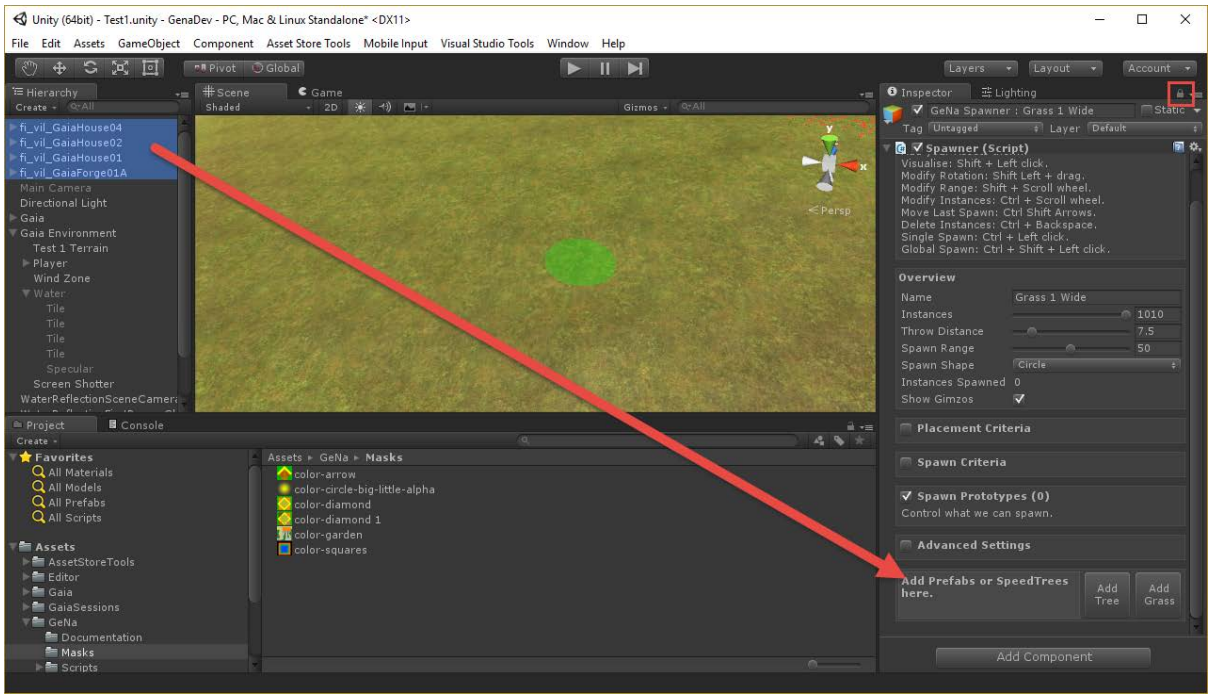
   NOTE: The reason we do this is that Gena spawns all objects relative to the point where the collision hits the terrain or mesh. The Y Offset of the prefab is then added to this, so a value of -0.05 will ensure that it is embedded by 5 centimeters.

2. Lay your prefabs out on that plane in the way that you would like them to be spawned into your scene. As per the previous comment, Gena will take the height the object is placed at and reflect it in its Y offset. In addition to sinking objects into the terrain slightly, you could also make objects float by placing them above the plane and GeNa will reflect this in what it spawns.

3. Select your spawner and lock it in the inspector

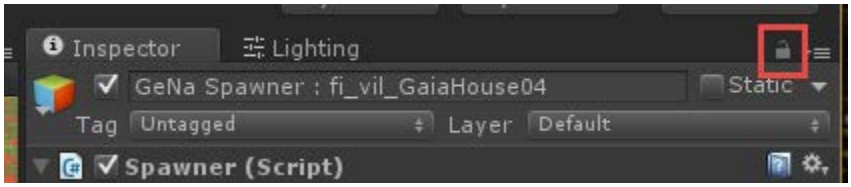


4. Select the objects as a group, and physically drag them all onto the prefab pad.

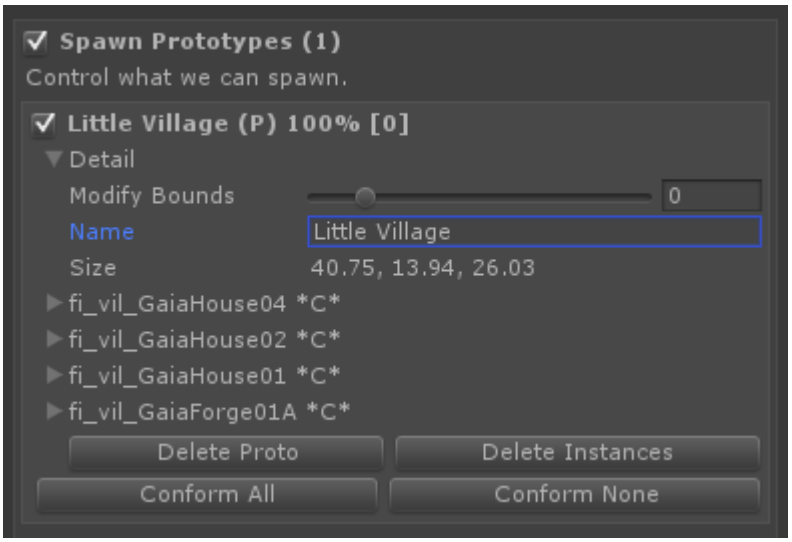


NOTE: Do not child prefabs under other objects. This must be done without any form of hierarchy.

5. Unlock the inspector.



6. Change the name to something that makes sense for when this prototype is spawned into your environment.



7. Go in and fine tune the individual prefab settings. The most common thing you will do will be to set the prefab to conform to the normal of the environment or now. For example buildings would typically be build straight up, whereas paving stones or implements would conform to the slope of the environment they are being spawned in.

8. Choose your location as usual by shift clicking.

9. Place the structure at that location by Ctrl clicking.

Spawning Fences

Gena has the ability to rotate the next object to point to the previous object. This is great for things like fences.

1. Add a fence prefab to your spawner. Make sure it has a collider attached to it.

2. Set Rotation Type in the Placement Criteria to Last Spawn Closest.

3. Ctrl click to place the fence. The first fence pole rotation will be random. All future ones will be aimed at the last one.

   If the fence rotation is incorrect then change the Min and Max Y settings inside of the Prototype settings until it is correct.

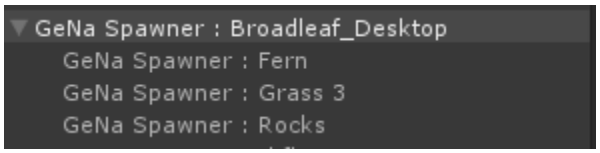
4. Move to where you think the next fence item should be and place the next fence item.

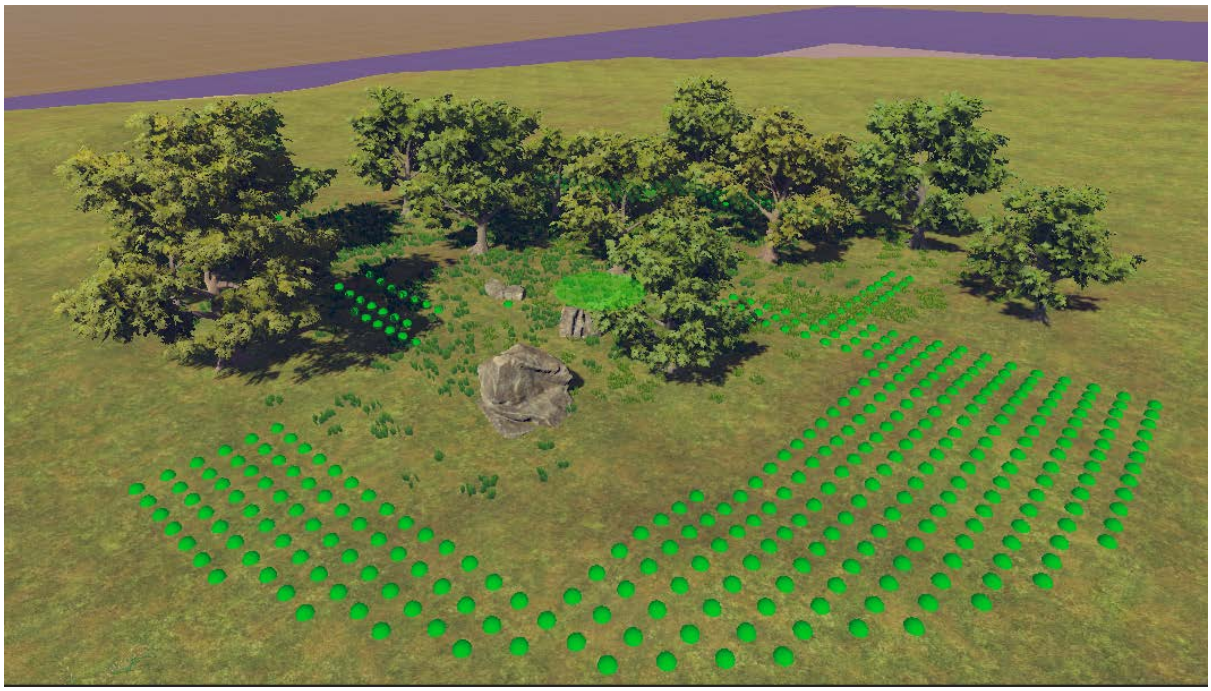
5. Repeat until done.

NOTE: This is never 100% accurate as it is based on colliders. You can use combinations of Shift + Ctrl and the arrow keys to fine tune the location and rotation of the fence.

Grouped / Child Spawners

Gena has the ability run multiple spawns at one time as a single entity. This is great for things that involve multiple objects that can be logically spawned as one entity.





1. Create and test 3 spawners – as tree spawner, and two grass spawners, and a rock spawner.

2. Test each spawner independently and set it up so that it delivers the results you want.

   NOTE: Be sure to unselect “Force Spawn” under advanced settings on each spawner as this will ensure that each spawner will only spawn in contextually correct locations so that things like collisions and other criteria are always obeyed.

3. Child the grass and rock spawners to the tree spawner.

4. Press Ctrl + Left click to run the spawner.

5. Repeat until done or press Shift + Ctrl + Left click to run a global spawn.

Runtime Spawner Script

Gena has the ability run spawn things at runtime, and has provided a simple but functional demonstration script that shows how to do this.

To demonstrate let's perform the following steps:

1. Add a player to your scene.

2. Add and configure a spawner in your scene.

3. Attach the RuntimeSpawner script to your player.

4. Set the spawn interval to the time in seconds that you would like between spawns.

5. Drag the spawner game object into the Spawner slot.

6. Select update spawner settings if you would like the spawn criteria for to be updated based on the players location, otherwise it will use the settings from the last shift or control click.

   Re-using old criteria / settings can be useful as it means you can preconfigure the type of terrain you can spawn on, and when the spawn kicks in it ensure that only the old settings will be used i.e. only spawn on flat, even if the player is standing on a hill at runtime.

7. Press play and run around in your scene. A new spawn iteration will be instigated every "spawn interval" seconds.

NOTE: It is better to have multiple small spawns rather than one large one as big spawns can be cpu expensive and cause dropped frames.

For a video example please see:
https://www.youtube.com/watch?v=mFhECi_6qak&t=11s

Growth Script

As a fun value add, Gena comes with a simple but efficient growth script that can be placed on things like trees, and then used either at design time or run time.

To demonstrate let's perform the following steps:

1. Physically drag a speed tree into your scene.
2. Attach the "GenaGrowthScript.cs" to your tree and set the growth settings.
3. Save the tree as a prefab by dragging it into your assets folder.
4. Delete the tree from your scene.
5. Set the x, y, and z values on the prefab to zero.
6. Create a spawner and drop the new tree prefab onto it.
7. Update the prototype and un check "Conform Slope" under the detail folder.
8. Spawn a bunch of trees into your scene.
9. Click play and watch them grow ☺
10. Optionally, set these up on a runtime spawner and spawn growable trees at runtime.

For a video example please see:
https://www.youtube.com/watch?v=mFhECi_6qak&index=5&list=PLckEMv5tz0E6Gw6PlN1vP0c2M-ZRMG63F

Mouse & Keyboard Controls

GeNa is controlled by both the keyboard and the mouse, and it's the combination of these and the dynamic way that GeNa interprets where you click that makes it so powerful.

You can modify these defaults by changing the settings in the GeNa Defaults file.

Shift + Left Click

Visualise the environment at that location.

Ctrl + Left Click

Run a spawn iteration at the location.

Shift + Ctrl + Left Click

Run a global spawn iteration based on the scene interpretation of that location i.e. find similar locations across the entire scene and run a spawner iteration on them.

Ctrl + Backspace

Delete all instances of the resources managed by this spawner from the scene. USE WITH CAUTION as it will delete every instance of the same resource regardless of whether it was placed by this spawner or not.

Shift + Left Click & Drag

Modify the rotation of the prototype. Only active when Draggable Rotation is enabled in the Advanced Settings panel.

Shift + Scroll Wheel

Change the range of the spawner.

Ctrl + Scroll Wheel

Change the instances of the spawner.

Ctrl + Left, Right, Up, Down Arrow

Move the position of the last prefab spawned.

Shift + Ctrl + Up, Down Key

Change the Y axis height of the last prefab spawned.

Shift + Ctrl + Left, Right, Up or Down Keys

Change the Y axis rotation of the last prefab spawned.

Release Notes

1.0.0.0 – October 2016

Initial release

1.1.0.0 – November 2016

New Features:

- Sub spawners / child spawners
- Runtime spawner sample
- Runtime growth sample
- Added POI colliders option and runtime collider disabler script

Enhancements:

- Updated default embedded setting for prefab prototypes
- Improved the invert perlin noise function
- Fixed the global spawn on mesh function
- Updated location of root component for poi spawns
- General tidy up of spawner class organisation
- Updated default prototype rotation settings to always rotate on Y axis
- Expose prototype rotation settings for trees
- Added version specific physics settings
- Added ability to cancel global spawns

1.5.0.0 – December 2016

New Features:

- Automated Optimization system
- Automated LightProbe placement system
- Defaults system
- Added Forward Rotation for precise rotation when using draggable rotation

Enhancements:

- Improved way in which last spawn can be positioned
- Improved many small editor settings