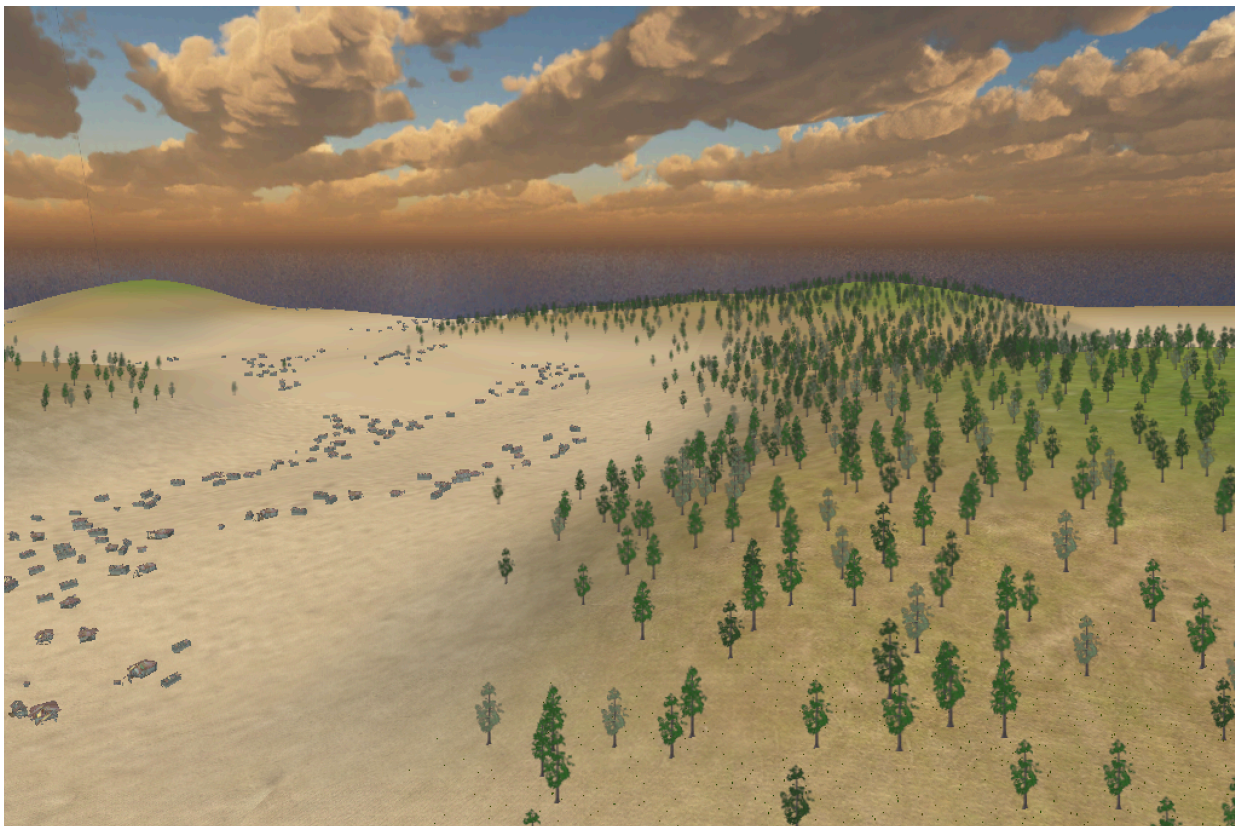
Easy Terrain



1. Introduction

This asset is a wrapper for Unity's built-in terrain system. It is meant to quickly generate several terrain 'tiles', that follow the movement of the player. This creates the possibility to create the feeling that the player moves in an endless environment.

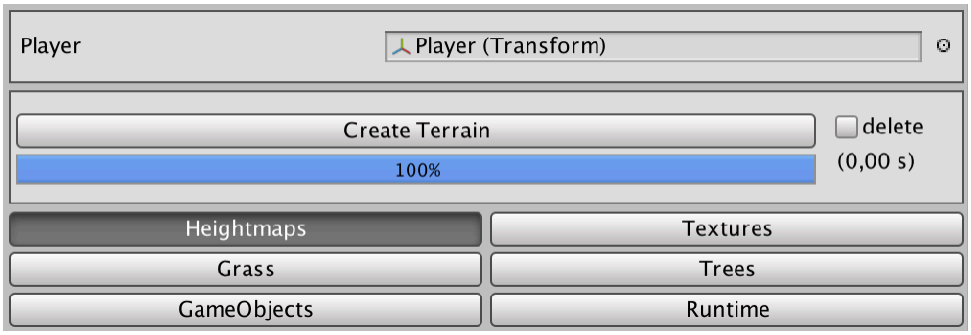
For further info or questions, you can contact me at: MouseSoftware@GMail.com

Happy Coding!

Maurits.

2. How to use

- Put the “EasyTerrain” prefab in the scene Hierarchy and select it:



- Now drag your player transform in the scene to the “Player” field. The terrain around this transform will be continuously updated, effectively following the player as it moves around in the scene.
  Note: At the start of the application, the player transform will always be placed on the terrain, the game world center (0,0,0).

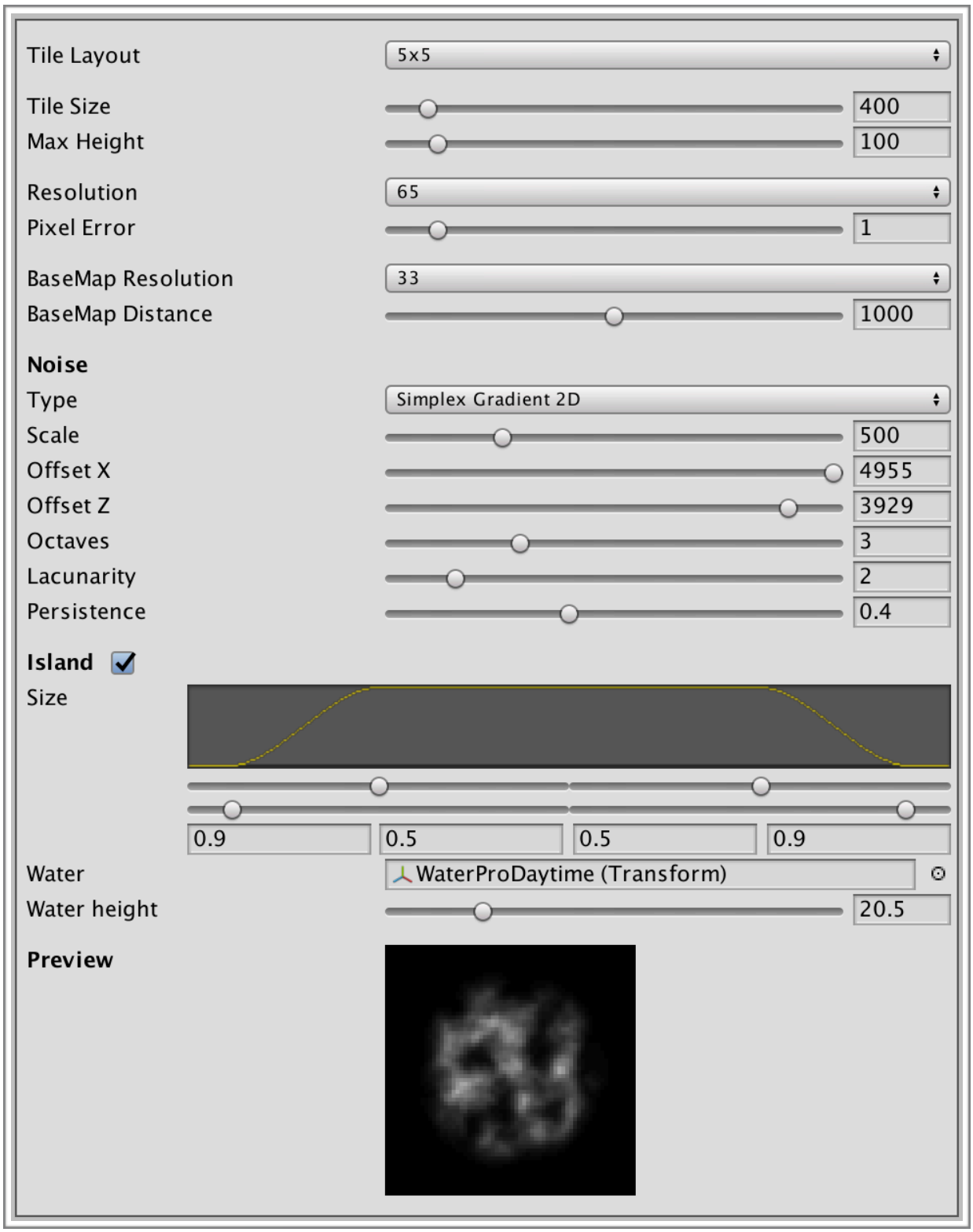
- Select the various settings in the Heightmaps, Textures, Grass, Trees, GameObjects, and Runtime-menu. Each menu-item will be described in more details in the following sections.
  Use the preview texture to quickly determine how your terrain will look.

- Now hit the “Create Terrain” Button and wait for your terrain to finish.

Note: During editing of the terrain in the editor, the script will only update itself when the gameview is visible in the editor.

2.1 Heightmaps

The height of each point on the terrain is represented as a value in a rectangular array. This array can be represented using a grayscale image known as a *heightmap.*



Tile Layout	Several separate terrain ‘tiles’ are combined, to create a larger terrain. These tiles are arranged in a square pattern (3x3, 5x5, 7x7, etc). When the player moves, some tiles are repositioned, to create the illusion of a single, infinite terrain.
Tile Size	The size of a single tile in its X and Z axis (in world units).
Max Height	The maximum height of each terrain tile
Resolution	Pixel resolution of each terrain tile’s heightmap. Higher resolutions result in smoother terrain, but higher amount of vertices.
Pixel Error	The accuracy of the mapping between the terrain maps (heightmap, textures, etc) and the generated terrain; higher values indicate lower accuracy but lower rendering overhead.
Basemap Resolution	Pixel resolution of each terrain tile’s basemap. This is the map to be used on the terrain when viewed from a distance greater than the *Basemap Distance.* *Note: the asset will not allow resolutions that exceed the heightmap (close range) resolution*
Basemap Distance	The maximum distance at which terrain textures will be displayed at full resolution (*‘Resolution’*). Beyond this distance, a lower resolution (*‘Basemap Resolution’*) composite image will be used for efficiency.

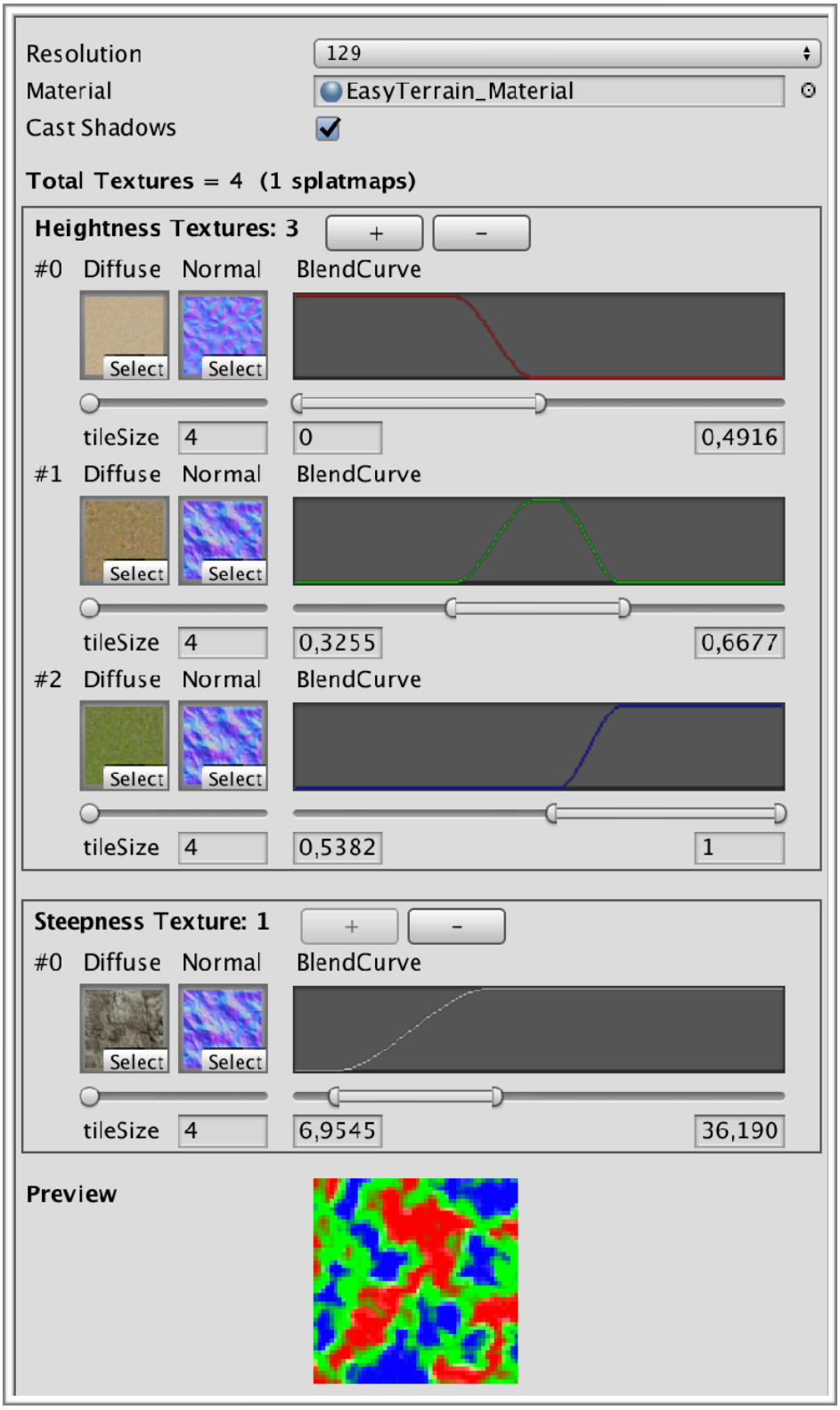
Noise Settings

Type	The noisetype to be used. Available options are Value2D, Value3D, Perlin2D, Perlin3D, SimplexValue2D, SimplexValue3D, Simplex2D. [Note: Simplex3D is by default not available. Uses of implementations of Simplex noise in *3D and higher* for *textured image synthesis* are covered by U.S. Patent 6,867,776, but only if “the algorithm is implemented using the specific techniques described in any of the patent claims". To play safe and to prevent possible infringements of the patent, the selection of Simplex3D is disabled by default. It can be enabled by editing the “public enum NoiseMethodType”-function in the NoiseGenerator.Cs script. Use at your own risk.]
Scale	The largest size of noise-features to be used on the terrain. [Note: The scale is actually 1/frequency, where frequency is the number of cycles per unit length that a specific noise type outputs. In terrain-editing, the use of the term scale is more straightforward]
OffsetX, OffsetY, OffsetZ	The offset of the noise, in it’s X, Y and Z-direction. Alter these to create a different look to your terrain, while keeping all other noise-parameters the same.

Octaves	Several noise-layers can be combined, to create larger and smaller noise-features. Octaves is the number of layers to be used, and thereby defines the *amount of detail* that is visible in the terrain. The scale of each successive octave is determined by the value of the previous octave's scale, the value of the lacunarity and the value of the persistence.
Lacunarity	A multiplier that determines how quickly the scale decreases for each successive octave / noise layer
Persistence	A multiplier that determines how quickly the amplitudes decrease for each successive octave / noise layer.
Island	(optional)
Size	The (circular) size and shape of the island
Water	The transform of the water object in the scene
Water height	The height level of the water

2.2 Textures

You can add texture images to the surface of a terrain to create coloration and fine detail. Since terrains are such large objects, it is standard practice to use a texture that repeats seamlessly and tile it over the surface. The painted textures can be applied with variable transparency so you can have a gradual transition between sandy areas, grassy hilltops and rocky steep hills, for example.



Resolution	Pixel resolution of each terrain tile's "splatmap", that controls the blending of the different terrain textures.
Material	The material to be used on the terrain. If left empty, Unity's default terrain material wil be used.
Cast Shadows	Does the terrain cast shadows?

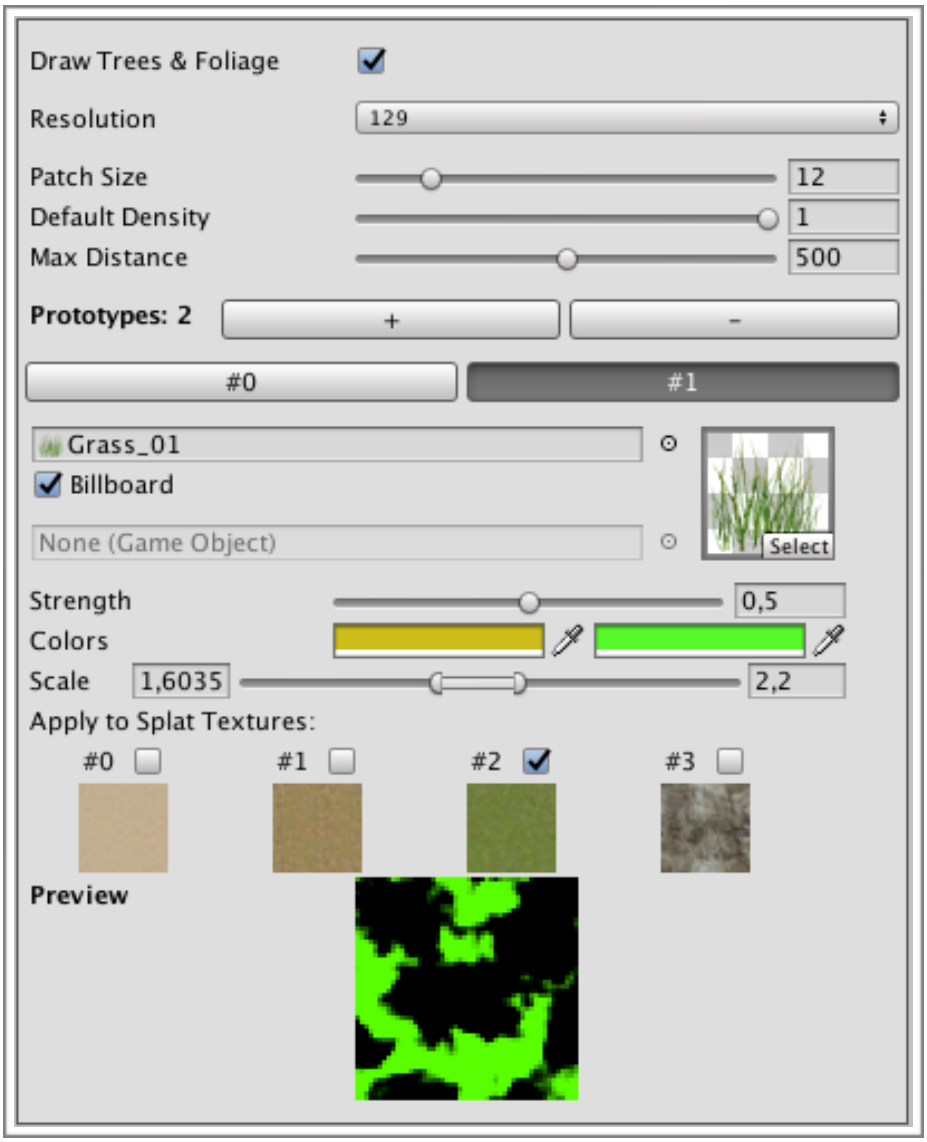
Heightness / Steepness Textures

A total of up to 8 textures can be added by clicking the [+]-button. In the current version only one texture can be used as a steepness texture (leaving a maximum of 7 textures to be used as heightness textures).
A splatmap can hold up to 4 textures, So using more than 4 textures will result in more draw calls, because it requires an additional splatmap

Diffuse Texture	The diffuse (RGB)-textures to be used.
Normal Texture	Optional: The normal texture to be used
Tilesize	The size of the texture (in world units).
Blendcurve	Use the sliders beneath the blend curves to determine where the textures will be used on the terrain. The preview texture will provide a direct feedback of any changes that are made. For heightness textures, values can be in the range of 0 (minimal height) or 1 (maximum height of the terrain). For the steepness textures, values wil be in the range of 0 (horizontal) or 90 (vertical).

2.3 Grass

A terrain can have grass clumps and other small objects such as bushes covering its surface. Grass is rendered by using 2D images to represent the individual clumps while other details are generated from standard meshes.



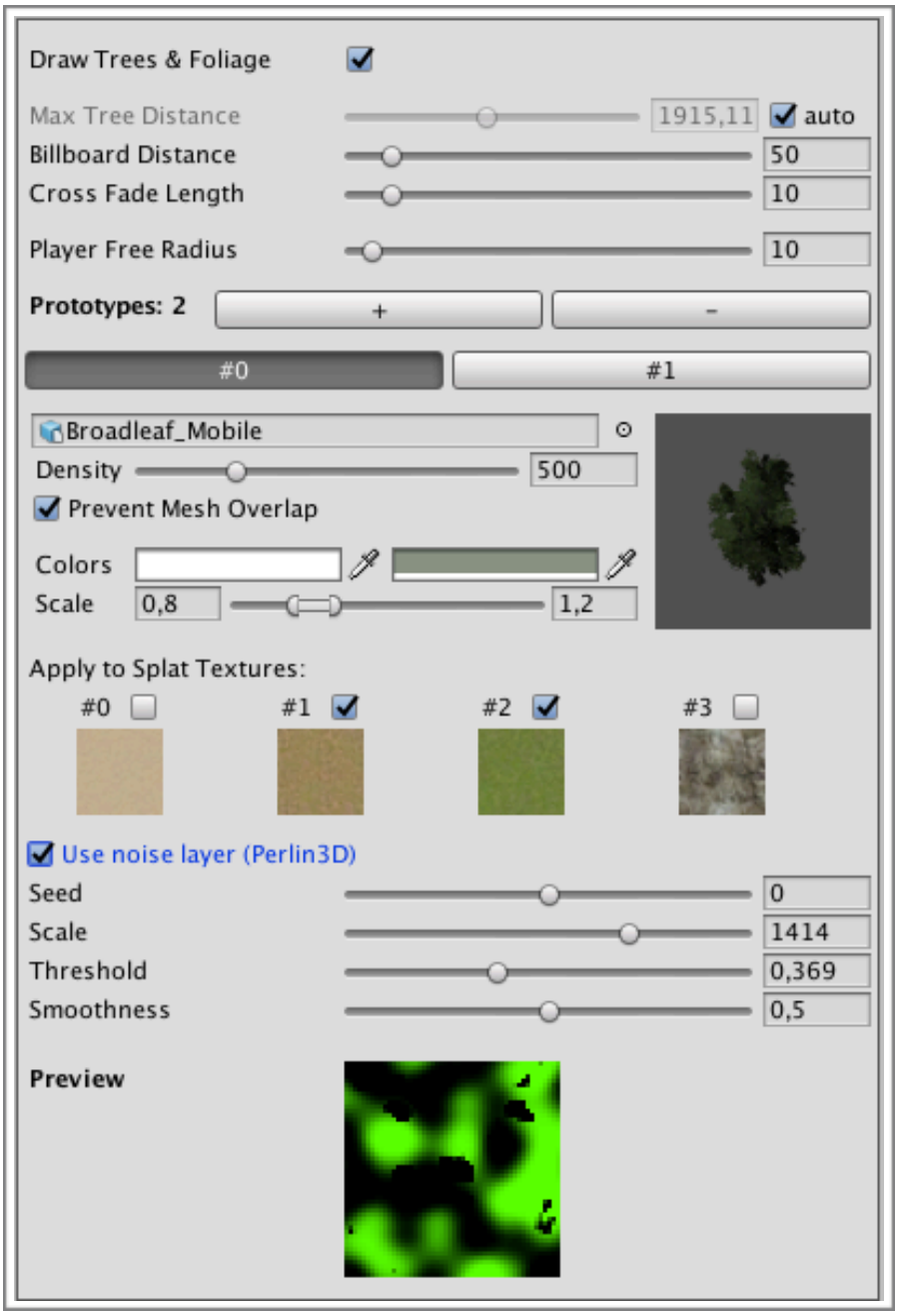
Draw Trees & Foliage	Should trees and grass be drawn on the terrain?
Resolution	Resolution of the map that determines the separate patches of details/grass. Higher resolution gives smaller and more detailed patches. Note: the asset will not allow resolutions that exceed the splatmap's resolution.
Patch Size	Length/width of the square of patches renderered with a single draw call.
Default Density	The number of detail/grass objects in a given unit of area. The value can be set lower to reduce rendering overhead. (1=max density)
Max Distance	The distance (from camera) beyond which details will be culled.

Prototypes
Clicking the [+]-button will add a new prototype. Clicking the [-]-button will remove the last prototype in the list.

Texture2D	The 2D images to be used to represent an individual piece of grass
Billboard	When the *Billboard* option is enabled, the grass images will rotate so that they always face the camera. This option can be useful when you want to show a dense field of grass because there is no possibility of seeing clumps side-on and therefore visibly two-dimensional. However, with sparse grass, the rotations of individual clumps can become apparent, creating a strange effect.
GameObject	The game object to be used to represent an individual piece of grass. Only use meshes with a low poly count.
Strength	De strength of this particular prototype. When set to 1, the maximum density will be used.
Color	Use different colours to distinguish “healthy grass” and “dry grass”
Scale	The grass pieces can be randomly scaled, to allow for differences in size.
Apply to Splat Textures	Select the splat textures where the grass should be placed.

2.4 Trees

Unity terrains can be furnished with *trees.* Unity's terrain supports SpeedTrees made with SpeedTree Modeler from IDV, inc. and trees that were created using unity's own Tree Creator.



Draw Trees & Foliage	Should trees and grass be drawn on the terrain?
Max Tree Distance	The maximum distance where trees should be visible on the terrain. When set to auto, the tree distance is calculated using the player camera's view distance.
Billboard Distance	Distance from the camera where Unity Trees will be rendered as billboards only. Decreasing this value improves performance but makes the transition look worse because the difference between billboards and trees will be more obvious.

Cross Fade Length	Total distance delta that Unity Trees will use to transition from billboard orientation to mesh orientation. Decreasing this value makes the transition happen faster. Setting it to 0 will produce a visible pop when switching from mesh to billboard representation. Note: The settings for Cross fading SpeedTrees should be set on the SpeedTree itself.
Player Free Radius	This will allow a clear spot around the player at startup of the level: The player will always starts at the center of the terrain (0,0,0) and no trees/gameobjects will be placed within this radius.

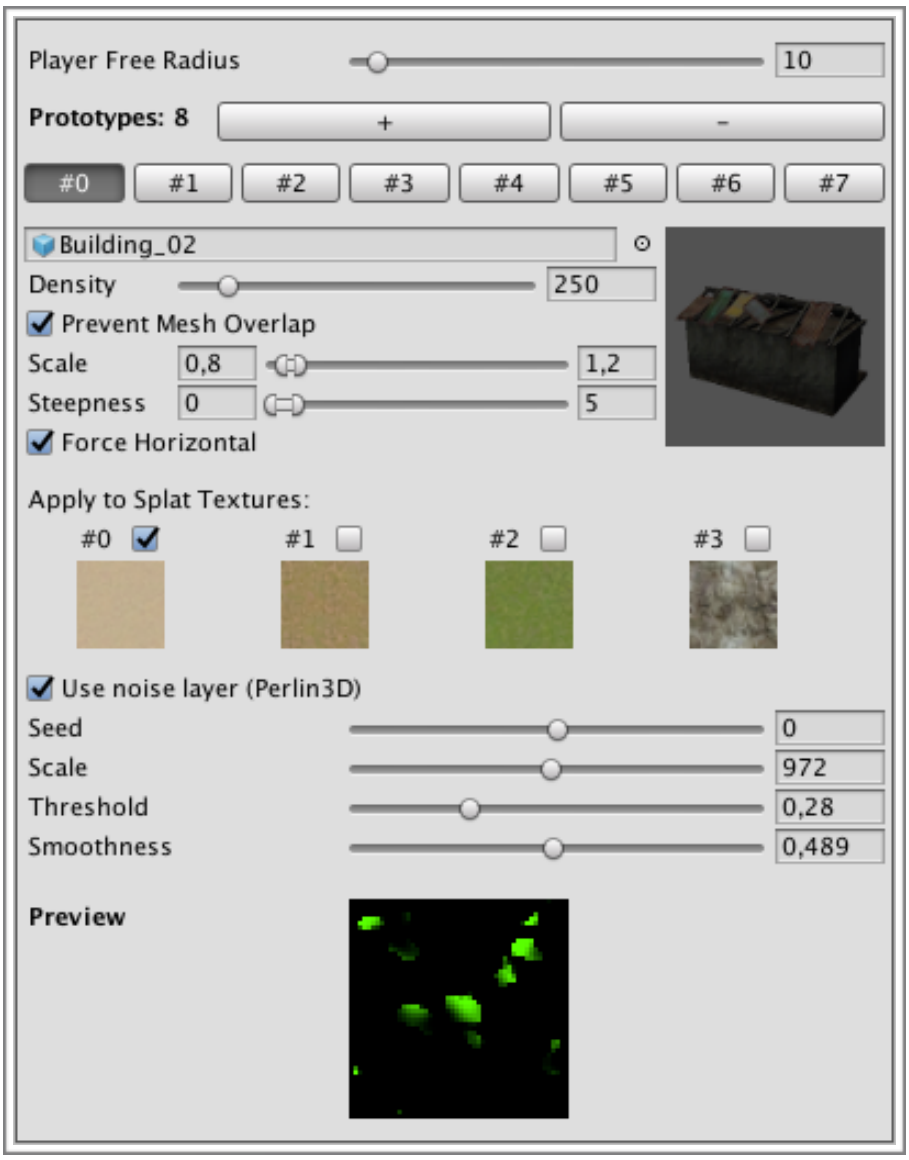
Prototypes

Clicking the [+]-button will add a new prototype. Clicking the [-]-button will remove the last prototype in the list.

GameObject	The prefab game object to be used to represent an individual tree. Note: don't forget to add Colliders to this prefab!
Density	The maximum density (unit: number of trees per squared kilometer)
Prevent Mesh Overlap	When enabled, the game object bounding boxes will be used to create a free range around the tree, where no other object can be placed. This will effectively prevent any meshes to overlap.
Colors	Trees can be drawn using different color variants. This way, much more difference can be achieved with a single tree prefab.
Scale	The trees can be randomly scaled, to allow for differences in size.
Apply to Splat Textures	Select the splat textures where the trees should be placed.
Use Noise Layer	Should an additional noise layer being used?
Seed	The seed to be used. Each seed results in a different 'random' pattern.
Scale	The scale of the random pattern
Threshold	This value can be used to create smaller 'islands' where trees will be placed.
Smoothness	This value determines how smooth the transition will be from areas with no trees and areas with many trees.

2.5 Gameobjects

Terrains can also be decorated with gameobjects, to create features like rock formations, watchtowers, villages, etc.



Player Free Radius — This will allow a clear spot around the player at startup of the level: The player will always starts at the center of the terrain (0,0,0) and no trees/gameobjects will be placed within this radius.

Prototypes

Clicking the [+]-button will add a new prototype. Clicking the [-]-button will remove the last prototype in the list.

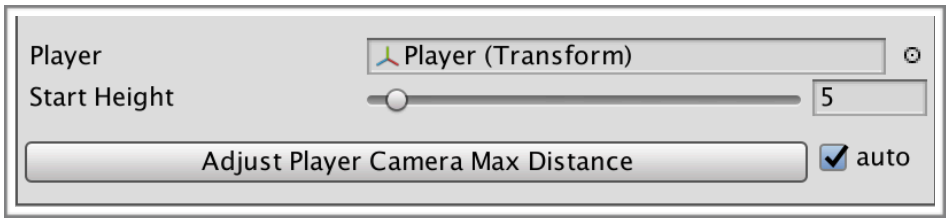
GameObject — The prefab game object to be used to represent an individual tree.
Note: If this game object holds colliders, try to keep it's collider geometry as simple as possible, e.g. use sphere, capsule and box colliders and try to avoid mesh colliders.

Density	The maximum density (unit: number of trees per squared kilometer)
Prevent Mesh Overlap	When enabled, the game object bounding boxes will be used to create a free range around the tree, where no other object can be placed. This will effectively prevent any meshes to overlap.
Scale	The gameobejcts can be randomly scaled, to allow for differences in size.
Steepness	Gameobjects will only be placed on the terrain when the steepness of the terrain lies within these boundaries.
Force Horizontal	Should the GameObject be placed horizontally? This is a good option for buildings. If this setting is disabled, the rotation of the game object will follow the curvature of the terrain, which is suitable for rocks, rubble, etc.
Apply to Splat Textures	Select the splat textures where the game objects should be placed.
Use Noise Layer	Should an additional noise layer being used?
Seed	The seed to be used. Each seed results in a different ‘random’ pattern.
Scale	The scale of the random pattern
Threshold	This value can be used to create smaller ‘islands’ where GameObjects will be placed.
Smoothness	This value determines how smooth the transition will be from areas with no GameObjects and areas with many GameObjects.

2.6 Runtime settings

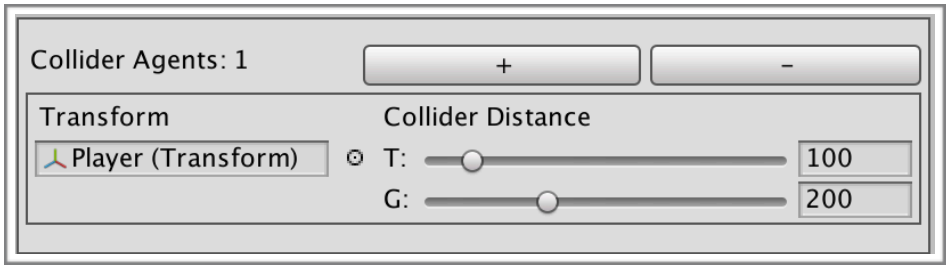
This menu holds various settings that apply to runtime:

Player:



Player	The player transform. The terrain around this transform will be continuously updated, effectively following the player as it moves around in the scene.
Start Height	The starting height of the player transform, relative to the terrain surface. At the start of the application, the player transform will always be placed at the terrain's center (0,h,0), where h is equal to (terrain height + *Start Height)*
Adjust Player Camera Max Distance	The player's main camera can be (automatically) adjusted to fit the number of terrain tiles and tile size. It is chosen in such a way, that it will minimalist the effect of new terrain tiles visually disappearing and reappearing in front of the player.

Collider Agents: A scene filled with many colliders for trees and/or other objects will soon lead to hiccups in gameplay, especially when new colliders are generated. therefor, the script creates a pool of TreeColliders, and uses those on trees that are nearby so-called collider agents. The colliders of GameObjects added via the GameObject Menu in this script will be deactivated by default, and will be reenabled when the player moves within a certain range.

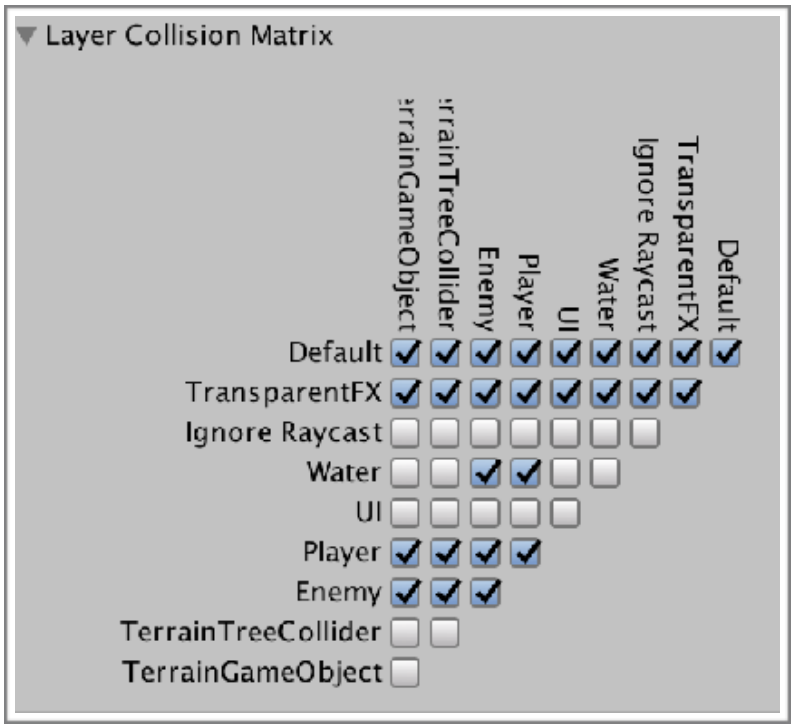


Transform	The transform(s) to be used as a Collider Agent. If the player transform is not selected as a Collider Agent, it will be automatically added during the start of the application or when a new terrain is generated.
Collider Distance	The distances where the colliders for Trees (T) and GameObjects (G) should be activated. Only the colliders of GameObjects added via the GameObject Menu of this script will be influenced by this setting.

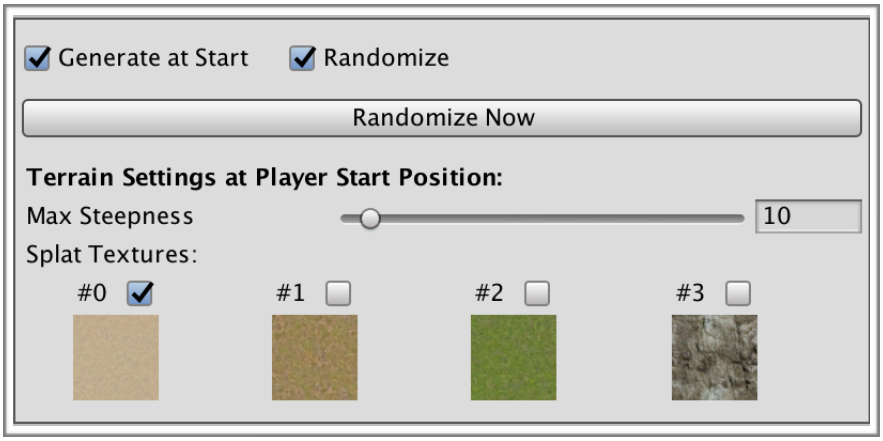
Tree Collider Layer:

The asset will allow you to assign tree colliders to a unique layer. Combined with the use of the layer collision matrix. Especially with dense forests, this might give you better performance, because less collisions means less calculations.

- Create a new layer that will hold all tree colliders: Edit —> Project Settings —> Tags & Layers. For example, create a layer named “TerrainTreeLayer” at slot 13.
- Now edit your layer collision matrix: Edit —> Project Settings —> Physics.
  Make sure that collision detection of “TreeColliderLayer” with itself is disabled. It will look something like this:
- In the runtime menu of the EasyTerrain asset, make sure the *TreeColliderLayer* is set to the correct layer.



Auto-generating and randomizing terrains:



Generate at Start	Should the entire terrain be generated at the start of the game?
Randomize	Should the newly generated terrain be completely randomized?
Randomize Now	Click this button to create a new random terrain, with al runtime settings applied.

Terrain Settings at Player Start Position

Max Steepness	The maximum allowed steepness of the terrain at the players coordinates (X=0, Z=0)
Splat Textures	The player can only be placed on this splat texture.

3. Advanced (additional scripting with API)

The asset includes some static public functions, that can be used during runtime.
Note: always put “using MouseSoftware;” at the top of a script, or else the commands will not work.

`TerrainSample GetTerrainSample(Vector3 coordinates)`
Get information about the terrain’s height, steepness at the provided x,z-coordinates (y-coordinate is being ignored)

Example 1:

```
using UnityEngine;
using MouseSoftware;

public class GetTerrainSampleTest : MonoBehaviour
{
   void Update()
    {
        EasyTerrain.TerrainSample terrainSample =
      EasyTerrain.GetTerrainSample(transform.position);

        Debug.Log("height = " + terrainSample.height);
        Debug.Log("steepness = " + terrainSample.steepness);
        Debug.Log("normal = " + terrainSample.normal);
        Debug.Log("derivative = " + terrainSample.derivative);
    }
}
```

`float GetUpdateStatusPercentage()`
Returns the status of the update progress, from 0% to 100%

`float GetUpdateStatusFactor()`
Returns the status of the update progress, from 0 to 1

`float GetStopwatchElapsedSeconds()`
Returns the amount of seconds that were needed to perform the last terrain update. When the update is in progress, this will return the current value of the stopwatch.

Example 2:

```
using UnityEngine;
using MouseSoftware;

public class GetTerrainUpdateStatusTest : MonoBehaviour
{
   void Update()
    {
      float statusPercentage = EasyTerrain.GetUpdateStatusPercentage();
      float seconds = EasyTerrain.GetStopwatchElapsedSeconds();
      Debug.Log(System.String.Format("{0:F0}% ({1:F2} s)",
      statusPercentage, seconds));
   }
}
```

`DetachTreesFromTerrain(Vector3 point, float radius)`
`DetachTreesFromTerrain(Vector3 point, float radius, Transform parentTransform)`
`DetachTreesFromTerrain(Vector3 point, float radius, string message)`
`DetachTreesFromTerrain(Vector3 point, float radius, string message, Transform parentTransform)`
`DetachTreesFromTerrain(Vector3 point, float radius, string message, object messageValue)`
`DetachTreesFromTerrain(Vector3 point, float radius, string message, object messageValue, Transform parentTransform)`

Trees are not separate gameobject, but are handled internally by unity's terrain system. With this command you can detach all trees that are near the coordinates *point* that are within the range *radius*. They are removed from the terrain and replaced by their individual gameobjects. All colliders will be automatically enabled. If no rigidbody is present, a new rigidbody will be added, in order for the trees to realistically fall down.
By default, detached trees will be added to the root of the scene, but you can provide a *parentTransform*, to keep your scene organized. Once detached, trees are no longer being controlled or updated by the terrain.
Optionally, a *message* can be send, with or without a *messageValue.* This can be used to call a specific function in a script attached to the gameobject. This way you can apply damage, remove a tree after a certain amount of time, etc.

Example 3:

```
using UnityEngine;
using MouseSoftware;

public class DetachTreesTest : MonoBehaviour
{
   public float treeKillRadius = 50f;
   public Transform detachedTreesRootTransform;

   void Update()
   {
      if (Input.GetKeyDown(KeyCode.Backspace))
      {
         Vector3 point = transform.position + transform.forward * 5f;
         float damageValue = 100f;
         EasyTerrain.DetachTreesFromTerrain(point, treeKillRadius,
                  "ApplyDamage", damageValue, detachedTreesRootTransform);
      }
   }
}
```

`AddColliderAgent(Transform agentTransform, float treeColliderDistance, float gameobjectColliderDistance)`

`RemoveColliderAgent(Transform agentTransform)`

Adds a new *colliderAgent*, or removes an existing colliderAgent.
EasyTerrain will only activate a tree collider when it's within a radius of *treeColliderDistance* of one or more colliderAgents. All other trees will not have a collider. Equivalent, only gameobjects within a radius of *gameobjectColliderDistance* will have an active collider.
Note: the player cannot be removed from the colliderAgentList.

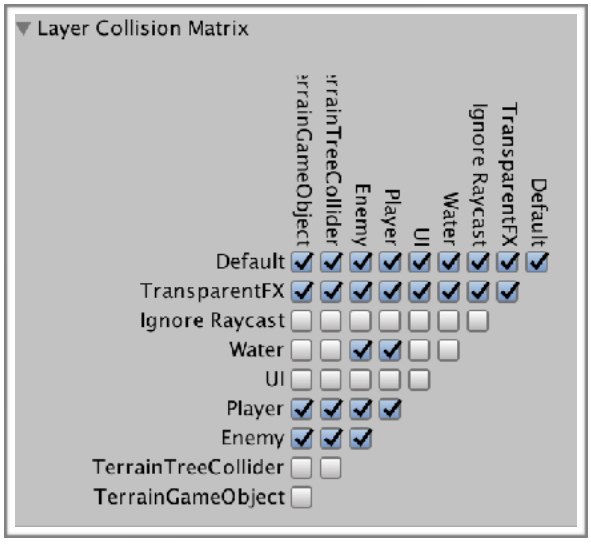
Example 3:

```
using UnityEngine;
using MouseSoftware;

public class AddOrRemoveColliderAgent : MonoBehaviour
{
  void Update()
   {
      // Hit backspace to remove this transform from the collider list
      if (Input.GetKeyDown(KeyCode.Backspace))
      {
         EasyTerrain.RemoveColliderAgent(transform);
      }
      // Hit space to remove this transform from the collider list
      if (Input.GetKeyDown(KeyCode.Space))
      {
         EasyTerrain.AddColliderAgent(transform, 200f, 250f);
      }
   }
}
```

3. Tips

- Keep the number of (visible) triangles as low as possible:
  - Don't use too many GameObjects, and keep there poly count as low as possible.
  - Don't put too many trees on your terrain.
  - Use some kind of LOD-technique (Level Of Detail). Speedtrees have this built in, but you still need to manually configure the settings. When using Unity's Tree Creator trees, try to keep the billboard distance as low as possible.
  - Speedtrees look better than Unity Tree Creator trees, but usually have a higher triangle count.
  - Keep grass density as low as possible.
- Many game objects that share the same material? Enable *GPU Instancing* in the materials inspector window, to dramatically increase performance (Note: *GPU Instancing* is not supported by all shaders. If you don't see the option to enable GPU Instancing, try using a different shader for your material).
- Use simple collider geometry for GameObjects, e.g. box, capsule and sphere colliders. Try to avoid mesh colliders, even with the convex-option enabled, because they are very computational intensive.
- Using many colliders? Keep your GameObjects in Layers and edit your layer collision matrix (Edit —> Project Settings —> Physics). This will give you better performance: less collisions = less calculations. This is also the reason why the asset allows you to automatically set the Layer for all treeColliders.



4. Changelog

v1.0
- Initial release

v1.1
- Don't like to wander off to infinity and beyond? You can now make an island too.
- A complete update to the shaders used in the demo scenes. It now supports multi-uv-mixxing AND triplanar projection AND blend-to-basemap. (fog is not supported, though)
- Package size has been reduced by almost 50% by reducing the number of sample prefabs in the demo scenes, removing some textures or replacing them by lower resolution versions and by replacing the skybox with a default procedurally generated skybox.
- The Unity team made some changes to the terrain engine in 2018.3. The asset has been updated, to reflect these changes, while maintaining backward compatibility.
- Some warning messages in 2018.2 were fixed
- Fixed some typos and made some minor changes to the scripts.

5. Acknowledgment

The noise-algorithm's used in this asset are strongly based on the tutorials by Jasper Flick, which are available on https://catlikecoding.com/unity/tutorials/. His website is full of really good tutorials. Definitely worth checking out!

Models of buildings and trees are taken from Unity's example projects and Unity's standard assets. It is legally fine to use Unity's assets, according to the Unity Licensing. It is recommended not to use them in commercial games and content, since these assets are all well known. They are intended for use in non-commercial content and for educational purposes only. See also https://support.unity3d.com/hc/en-us/articles/211950043-Can-I-use-Unity-assets-in-my-game-.

Some textures belong to the Public Domain. See the file "Third Party Notices.txt" for a further description.