

Dynamic Grass System

Starter guide

Dynamic Grass System allows to add dynamic behavior to any regular foliage mesh and use it easily with default UE foliage system. It also comes with some additional useful game mechanics:

- Dynamic interaction: Trample, bend away, cut, burn, remove... Make your player and other actors interact with grass in many ways! It's easy to add: Just add 'InteractWithGrass' component - to any actor you want. Besides actor interaction, you can also add it to e.g. explosions, so they will affect your grass... Or to UFO actor that leaves marks on your field... There are some examples available in the demo map. UFO included.
- Grass cutting / removing: You can cut or remove desired grass instances in realtime. This is also useful when e.g. spawning an object (like a building, or something smaller like chest) – to remove any grass within the spawning area, to prevent grass meshes overlapping and going through spawned structures.
- Fire propagation system: You can start a fire that will propagate with specified properties like speed or radius. It's possible to stop further propagation (e.g. with provided rain effect), limit its max influence and adjust many other fire behaviors.
- Dynamic wind system: You can change wind direction in game - it will influence fire propagation depending on wind direction & speed, and any wind direction/speed change effect will be reflected in the grass material.
- Grass growing over time: Spawned grass can grow, cut grass can re-grow, trampled grass can un-trample over specified time... And so on.
- Optimized: The system operates on instanced static meshes to reduce drawcalls and provide good performance. Additionally, any dynamic instance transformations are culled/automatically optimized, based on distance from current camera view target. It's also optimized for large amount of instances – the system divides large interactive foliage areas into chunks and interacts only with the nearest ones.
- In-game spawning system: You can spawn dynamic grass in realtime at desired location, as a single instance or as a clusters with specified amount, density, and random offset – to make spawned meadows more natural. It will detect any surface and spawn with rotation that respects the surface normal + add random yaw.
- In-game spawning: Grass Spawn Areas: Special actors that allow for precise grass generation on desired area, with properties like foliage type, density, scale, distribution irregularity, etc. Also included Grass Spawn Blockers.
- Ground texture blending: Grass fits more naturally into your landscape.
- Ready to use with UE Foliage tool: Besides provided dynamic spawning system, you can also paint dynamic foliage using UE Foliage tool and use all of its handy features.
- Many exposed parameters for customizing interaction, grow & burn behavior. You can even customize things like stiffness or burning fire/smoke color for your specified dynamic foliage type.

Besides this guide, there are two demo maps included in the project.
They contain many working examples & demo effects
– be sure to check them, they can serve as a good starting point.

(Maps\DemoMap + Maps\InGameGrassGeneration)

It's recommended to paint your grass/plants with the UE Foliage Tool or to spawn it dynamically in game by using provided foliage spawning areas & functions.
Please note that grass placed via UE Grass Tool (grass from landscape texture layer) works with all the material-based interaction effects, but permanent displacement effects like trample/cutting/burning won't work with it, because it's not possible to use FoliageType (allows for extended logic) with the Grass Tool.

Table of Contents

[Adding dynamic grass to your world](#)

[Adding dynamic grass to your world at runtime \(in game\)](#)

[Adding your own dynamic foliage type](#)

[Foliage type parameters](#)

[Adding interaction for actors](#)

[Interaction effects: Impulse](#)

[Interaction effects: Displacement & Material](#)

[Interaction effects: Bend Away](#)

[Interaction effects: Trample](#)

[Interaction effects: Remove](#)

[Grass un-deforming](#)

[Grass cutting](#)

[Grass growing](#)

[Grass burning & fire propagation](#)

[Dynamic wind control](#)

[Removing grass beneath spawned structures](#)

['Infinite Loop Detected' error with large amount of foliage - FIX](#)

[Custom grass mesh: Recommended shape](#)

[Tip: Grass plane normal](#)

[Ground blending & default grass material](#)

[Optimization parameters](#)

[... Not only grass!](#)

[Tip: Distant flickering fix](#)

[Demo levels & examples](#)

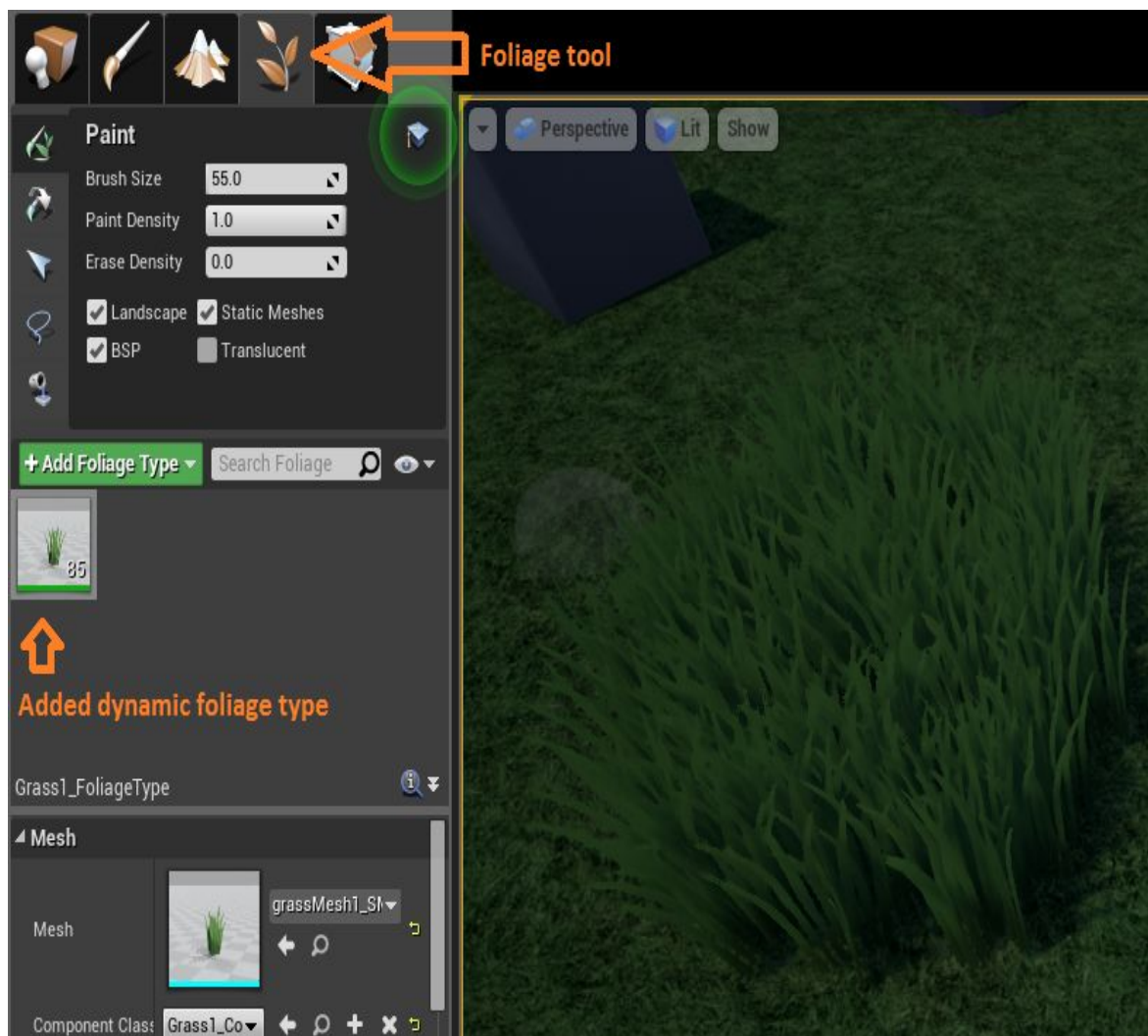
[Ending](#)

Adding dynamic grass to your world

Remember to add the **InteractiveGrassManager** blueprint into your scene (just drag&drop) to use the dynamic features! It's located in the Blueprints folder.

Interaction system uses dedicated foliage mesh component 'INTERACTIVE_FoliageComp' which is located in the Blueprints\InternalLogic\ folder.

You can find an example dynamic foliage types in the Blueprints\FoliageTypes\ folder. To add dynamic grass to your level, simply use the desired foliage type with the Foliage tool, just like with regular foliage meshes:



... This way you can paint your dynamic foliage using all of the handy Foliage tool features.

If you're not familiar with UE Foliage Tool, [HERE](#) you can learn more about all of its features.

Adding dynamic grass to your world at runtime (in game)

You can add dynamic grass during gameplay, using *Spawn Foliage* or *Spawn Foliage Cluster* function.

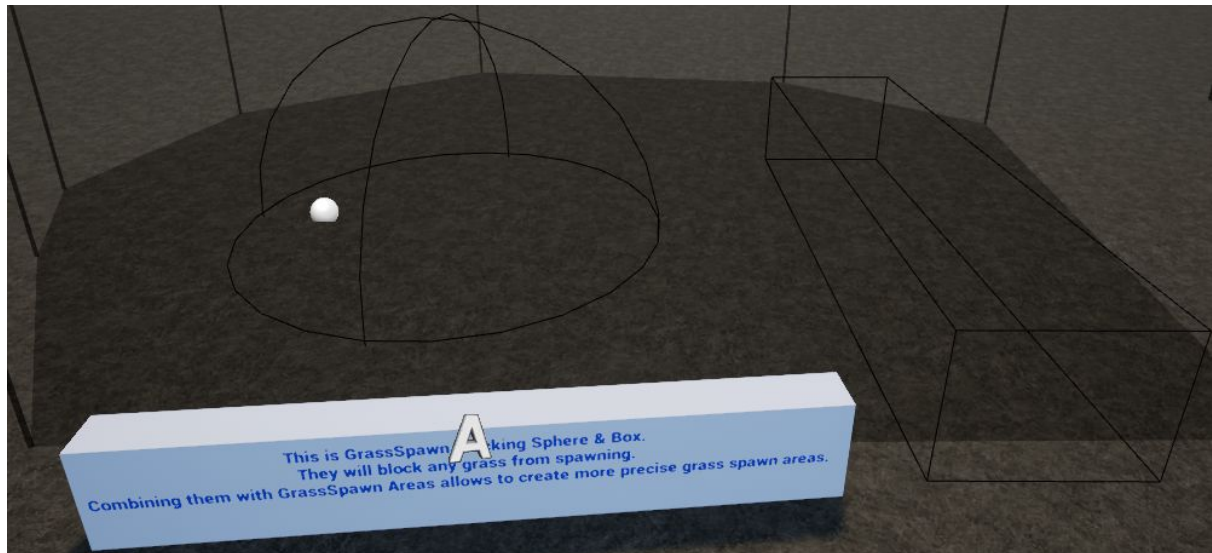
In the demo level, press *V* to spawn foliage instance under cursor or *B* to spawn foliage cluster under cursor. **You can find an example function usage in the demo level blueprint.**

The spawning functions use *FoliageToSpawn* structure. It contains two variables: *Mesh* and *Component*, which should be set to a foliage component that you want to spawn (the same that is assigned to every Foliage Type) This is necessary to be able to spawn foliage with all its custom dynamic properties like e.g. stiffness (transform weights) or fire color.

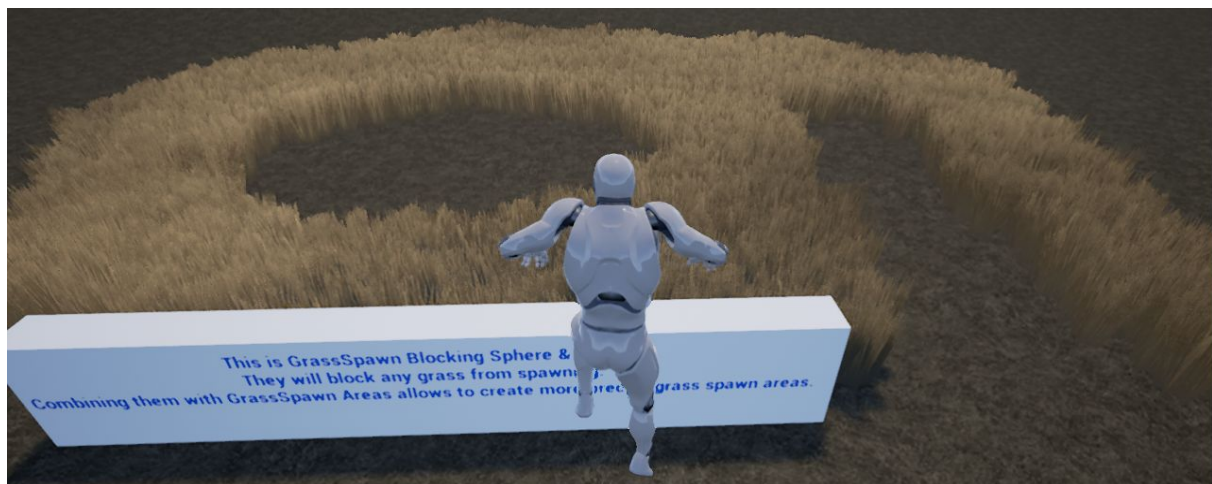
Grass Spawn Areas

You can also use special *GrassSpawnArea* actors. They can be placed in editor or spawned at runtime with ability to adjust their shape using vertices. A demo map named *InGameGrassGeneration* is included in the project - it shows all the usage examples and possibilities of this system.

Grass Spawn Area & Grass Spawn Blockers placed in editor:



... After generating in game:



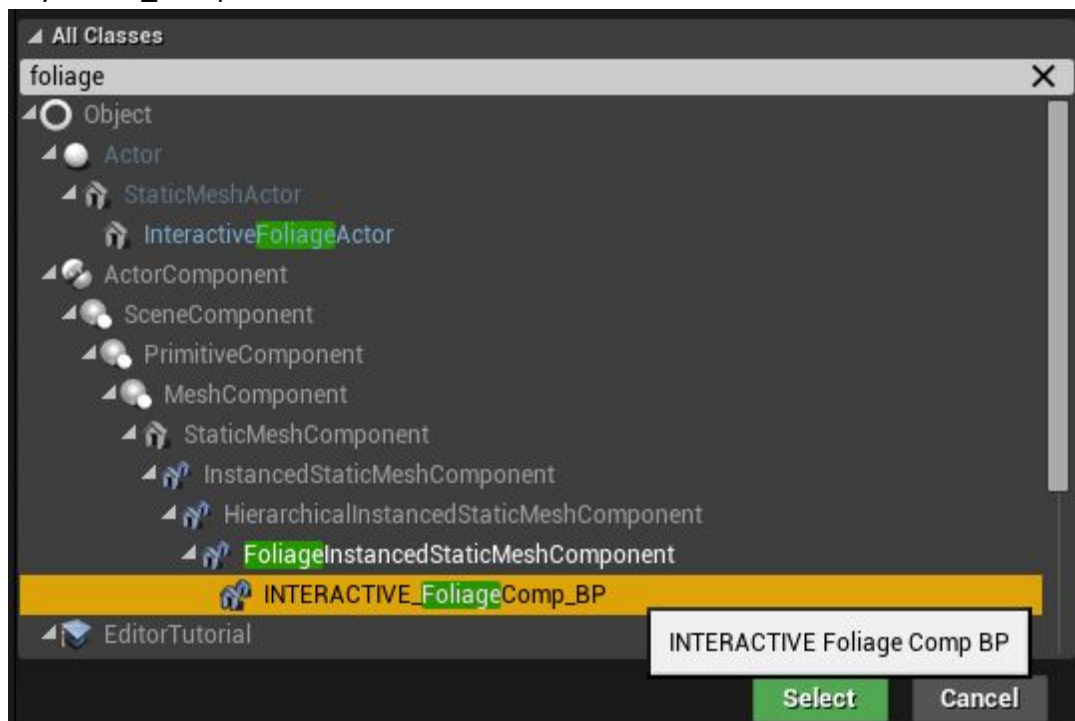
Grass Spawn Area can be used to dynamically spawn at any game moment, either on game start (can be useful with dynamic world generation) or e.g. on key press. For each area you can specify foliage type (it's possible to spawn more than one: For example, within one area you can spawn grass with high density + flowers with less density), density, min & max scale, distribution irregularity, etc.

You can find more examples & explanations in the *InGameGrassGeneration* map.

Adding your own dynamic foliage type

To add your own dynamic foliage type:

1. Create a component for your new dynamic foliage type by creating a blueprint that is child of the *INTERACTIVE_FoliageComp_BP* component. In the Right Click -> Blueprint Class -> 'Search' bar you can type e.g. 'foliage' to quickly find the component. Select it, click *Select* and name your component however you want, e.g. 'MyGrass1_Component'.



2. The newly created component contains all the dynamic properties of your foliage type. Here you can set many foliage behaviors that can be found under *Interaction Reaction*, *Grow Behavior*, *Burn Behavior*, *Burn Behavior – Ground*, *Burn Behavior – Fire* and *Burn Behavior – Smoke* sections. All the properties are explained in their tooltips – just hover over them to see it. You may need to scroll further through the *Details* tab to see all of the sections, because UE doesn't display variable categories correctly – they are not in order. I've created a feature request [HERE](#).
3. Add a Foliage Type (Right Click -> Miscellaneous -> Foliage Type), choose its *Mesh* to desired static mesh and its *Component Class* to the component created in the previous step.

You can also duplicate existing example dynamic foliage types in *Blueprints\FoliageTypes* folder and customize them, to make the process faster.

If you can't find your foliage component on the *Foliage Component* property list, please recompile the missing foliage component – then it should appear correctly. It happens sometimes and it's an UE bug that doesn't display some blueprint classes until re-compiled.

Foliage type parameters

Each foliage type have many parameters that you can adjust. It even includes things like fire/smoke color when burning. In the demo level you can find some examples that use different foliage type parameters. In the default included foliage types you can see that they're adjusted to control how stiff they are (transform displacement weights), how fast they un-deform, what's their burning color (separate values for fire & smoke) and more.

Adding interaction for actors

Grass interaction can work with any actor you want. All you need to do is to add the *Interact With Grass* component to desired actor.

When you have your actor selected in viewport or have its blueprint editor opened, click the green *+Add Component* button and in search bar type e.g. 'interact' to quickly find the *Interact With Grass* component.

Interact With Grass component will check for any dynamic-type foliage instances within specified radius and interact with them in a specified way. You can specify interaction type and properties under *Displacement Interaction* and *Material Interaction* sections of the added *Interact With Grass* component.

When adding interaction for your main character, check the *Always Simulate* option. This will prevent culling its interaction effects due to optimization system which limits maximum number of actors that can interact at the same time.

If your actor doesn't have *CharacterMovement* Component, you might need to check the *Skip Velocity Check* option. An additional explanation is available in the tooltip.

All the other interaction properties are described in their tooltips.

Collision is not necessary for interaction effects. Do NOT add collision to your custom grass meshes, because it can slow down your game when having large amount of foliage.

Interaction effects: Impulse

Thanks to the dynamic system, you can create many interesting interaction effects. One of them is the Impulse – it will simulate a shockwave with desired strength, radius and other displacement parameters. It can be used e.g. for grenades, bombs, magic spells, and so on.

An example usage of *Impulse* actor is demonstrated in *ExampleProjectile* & *ExampleBomb* blueprints that are located in *Blueprints\InternalLogic* folder. In the demo level you can use them using 1/2/3/4/5 keys, just like stated in the on-screen instructions.

[1] Fire a projectile with material interaction only

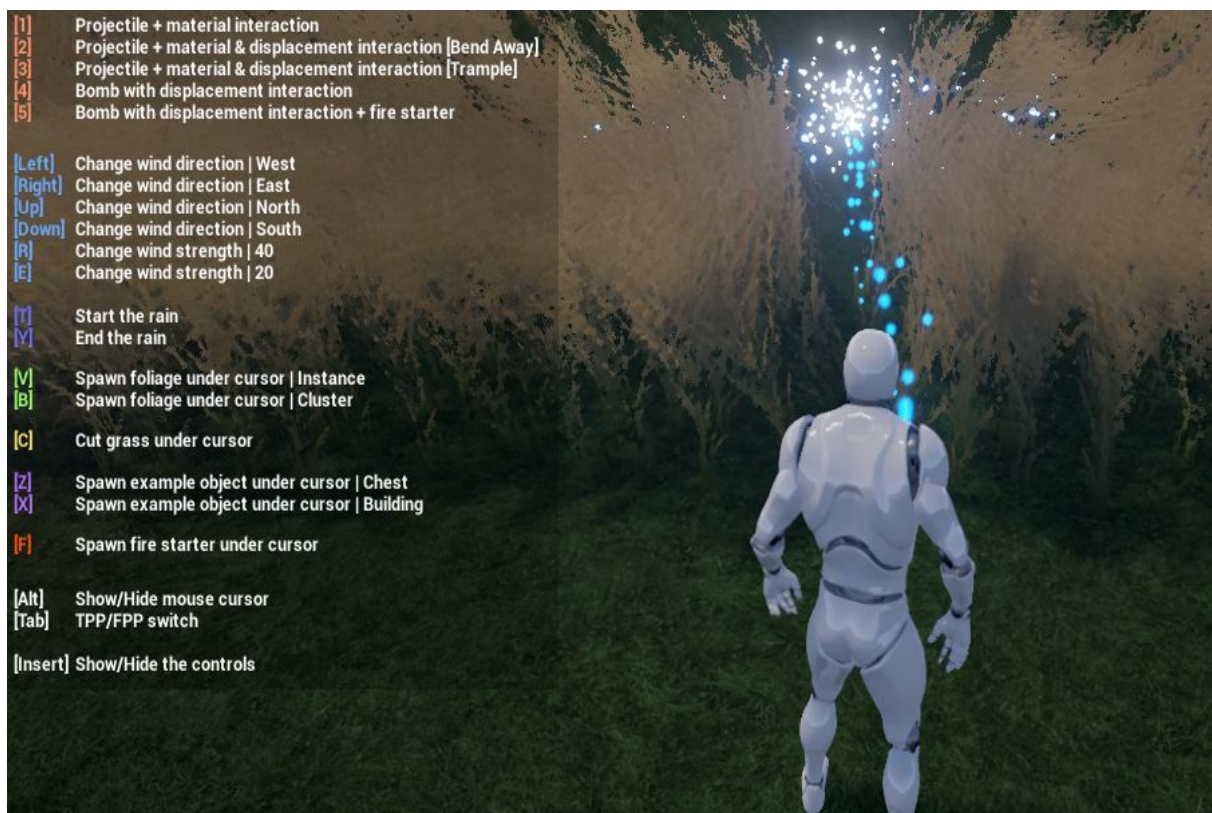
[2] Fire a projectile with material & displacement interaction: 'Bend Away' effect

[3] Fire a projectile with material & displacement interaction: 'Trample' effect

[4] Throw a bomb with displacement interaction

[5] Throw a bomb with displacement interaction + fire starter

The *ExampleProjectile* actor also has the *InteractWithGrass* component. This way it will interact with grass not only via the Impulse function upon exploding, but also while flying through grass.



Interaction effects: Displacement & Material

Each interacting actor can use different interaction parameters. They can use both displacement interaction (effects like trample, bend-away, remove) and material interaction (it will deform grass near interacting actor in appropriate direction, but the effect is not permanent).

Displacement Interaction		Material Interaction	
Use Displacement Interaction	<input checked="" type="checkbox"/>	Use Material Interaction	<input checked="" type="checkbox"/>
How to Interact	Trample	Material Effect Radius	130.0
Check Radius	5.0	Material Effect Strength	90.0
Check Interval	0.08	Mat Interaction Cull Distance	1500.0
Interaction Cull Distance	5000.0	Is for Prioritized Actor	<input type="checkbox"/>
Always Simulate	<input type="checkbox"/>		
Trample Bend Angle	80.0		
Trample Z-Scale	0.5		
Trample Bend Duration	0.25		
Trample Ignore Foliage Type Weights	<input type="checkbox"/>		
Bend Away Bend Angle	22.0		
Bend Away Z-Scale	1.2		
Bend Away Bend Duration	0.25		
Bend Away Ignore Foliage Type Weights	<input type="checkbox"/>		
Skip Velocity Check	<input type="checkbox"/>		

All the parameters are explained in their tooltips.

Interaction effects: Bend Away

This effect bends grass in the direction appropriate to current interacting actor. You can adjust effect parameters individually for each interacting actor.

The example player character use this effect by default. There are also other examples that use this effect, all can be found in the demo map.

Bend Away Angle WEIGHT	1.0
Bend Away Z-Scale WEIGHT	1.0
Bend Away Duration MULTIPLIER	1.0
Bend Away Yaw MULTIPLIER	1.0

Every foliage type can have different bend-away reaction via angle/scale weights & duration/yaw rotation multipliers. The parameters are explained in their tooltips.

Interaction effects: Trample

Trample effect tramples grass in more chaotic way than bend-away effect, which is useful when simulating grass trampling (e.g. when we want to trample it under a heavy object, not just bend away). An example with rolling physics ball is available in the demo level.

Trample Bend Angle WEIGHT	1.0
Trample Z-Scale WEIGHT	1.0
Trample Bend Duration MULTIPLIER	1.0

You can adjust effect parameters individually for each interacting actor. The parameters are explained in their tooltips.

Interaction effects: Remove

Grass instances within the effect radius will disappear.

It is used e.g. by the *GrassRemover* components when spawning structures in game. This feature is explained [here](#).

Grass un-deforming

Every foliage type can un-deform, which means going back to its original shape after deformed, over specified time. You can specify un-deform duration (how fast it will un-deform) or disable it if you wish.

In the demo level you can see a foliage with both un-deform ON and un-deform OFF behavior. For example, the default grass will un-deform, and the wheat field uses wheat foliage type with disabled un-deforming.

Un Deform	<input checked="" type="checkbox"/>
Un Deform Duration	60.0
Un Deform Duration when Trampled	120.0
Un Deform UPS	6
Un Deform UPS when Trampled	4

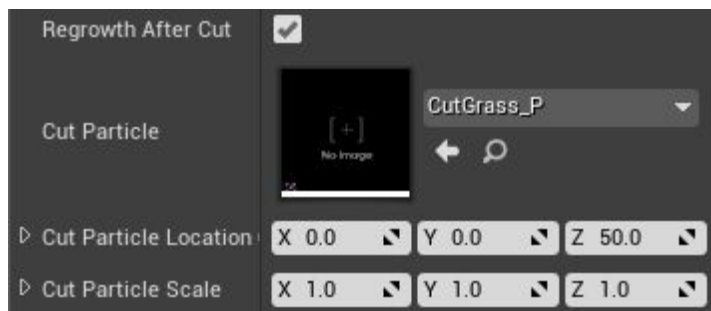
The UPS (Updates Per Second) parameter controls smoothness of the process and it's optimized automatically based on current camera distance, but you can adjust it to your liking. Generally, the longer un-deform duration, the less UPS you need to achieve smooth transition.

Grass cutting

In the demo level, press C to cut grass under mouse cursor. It will spawn an actor that uses cutting function – to see the example logic, check the level blueprint.

Cut grass can re-grow if you keep the 'Regrowth After Cut' property checked.

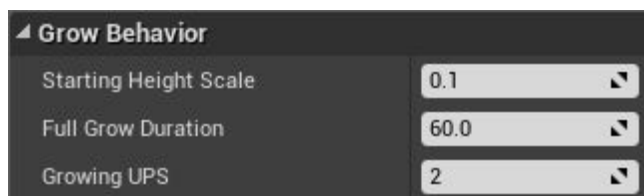
Cutting grass can also spawn a particle system that will simulate freshly cut grass pieces – you can specify individual particle systems for each foliage type.



Grass growing

Grass can grow over specified amount of time after spawning or after cutting it (re-growth).

You can also enable/disable the growing grass globally by switching the 'Growing Grass' parameter in *InteractiveGrassManager*, under General section.



The growth parameters are explained in their tooltips.





Grass burning & fire propagation

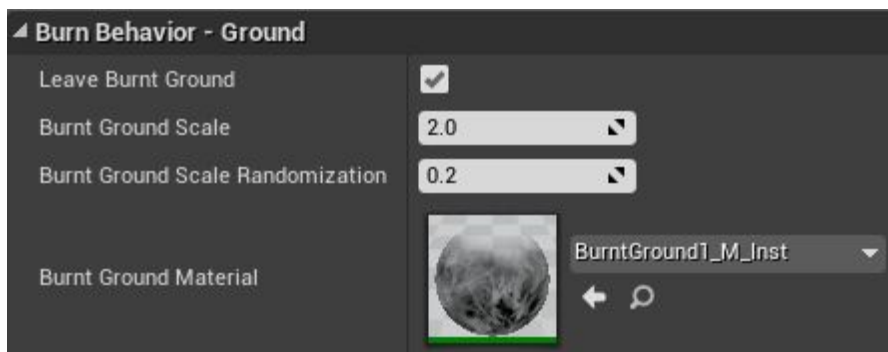
Foliage can be set on fire by spawning the *FireStarter* actor in desired location. The demo level presents a few examples on how to use it in game.

Starting Check Radius	200.0
Initial Fire Catch Radius	50.0
Use Max Propagations	<input checked="" type="checkbox"/>
Max Propagations	20

These are the *FireStarter* parameters and they are explained in their tooltips. You can also control burn behavior for each foliage type individually:

▲ Burn Behavior	
Burn Duration	20.0
Propagate Duration Min	8.0
Propagate Duration Max	15.0
Fire Catch Radius	70.0
Lean Amount when Burning	70.0
Use Dissolve Burn Effect	<input type="checkbox"/>
Dissolve Mask	 BurnDissolveMask1_T
Propagate Only to Same Type	<input type="checkbox"/>

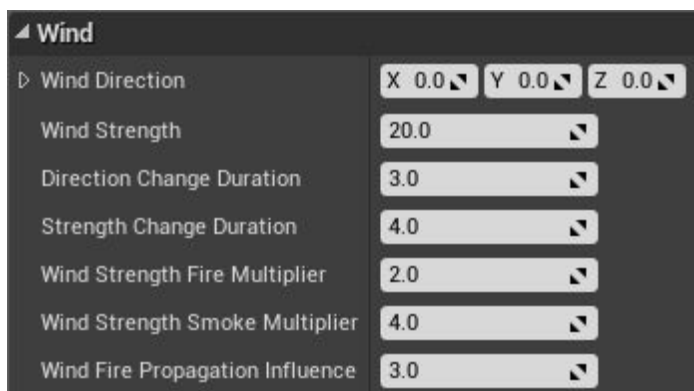
▲ Burn Behavior - Fire	
Enable Fire	<input checked="" type="checkbox"/>
Fire Scale	2.0
Fire Spawn Height	100.0
▷ Flame Start Color	
Flame Start Color Intensity	100.0
▷ Flame End Color	
Flame End Color Intensity	10.0
▷ Spark Start Color	
▷ Spark End Color	
Spark Color Intensity	100.0
▷ Ember Color	
Ember Color Intensity	300.0



All the parameters are explained in their tooltips and you can see some of them in action, for example on the default *PurplePlant* foliage type – it will burn faster, with purple flames & purple-tinted smoke. An example is available in the demo map.

Besides the *Max Propagations* property (more in the tooltip) you can control fire propagation via *Prevent Propagating After Time* function of *InteractiveGrassManager*. The first demo level has an example mechanic that spawns rain – it will stop any further fire propagation after a while.

Fire reacts to current wind direction & current wind strength. Through the manager you can also control wind influence:



Wind affects not only fire propagation, but also all the visual effects like flame behavior and smoke particle direction, all with controllable multipliers.

As usual, the parameters have additional explanations in their tooltips.

Dynamic wind control

Dynamic wind affects grass wind animation and fire propagation effects. Fire propagates faster when wind is strong, it also takes into account current wind direction.

You can change wind strength & direction using *SmoothlyChangeWindStrength* & *SmoothlyChangeWindDirection* functions of the *InteractiveGrassManager*. The first demo level offers an example mechanic that can change strength (R/E keys) and direction (arrow keys) in game.

As you can see in the previous picture, you can control duration of wind strength/direction change process. By default its set to low values, to make the change more visible for demonstration purposes – but in game, it will probably be better to set it to higher values to achieve smoother transition.

Removing grass beneath spawned structures

You can 'tidy up' any grass/foilage when spawning something during gameplay – for example, a building, chest, campfire or any other structure. This prevents grass from overlapping & growing 'through' spawned structures.

There is an example in the demo level – you can spawn a chest or a structure by pressing Z or X.

To add a 'Grass Remover' to your structure:

1. In your blueprint click *Add Component* and search for *Child Actor*
2. In the added *Child Actor*, set the *Child Actor Class* to *GrassRemover_BP*
3. In the Viewport tab of your blueprint, set the position/rotation/scale of the added component (It should appear as a red bound box) to your liking

... You can add as many *GrassRemover_BP*'s as you wish. All foliage instances inside these boxes will be removed after spawning the object.

'Infinite Loop Detected' error with large amount of foliage - FIX

When having a large amount of foliage (many instances) it's very likely to hit the default UE iteration limit which causes the 'Infinite Loop Detected' error. If you encounter this error, just go to *Edit -> Project Settings -> General Settings* and change the '*Maximum Loop Iteration Count*' to a bigger value (e.g. by adding one '0' at the end) until the error is gone.

Custom grass mesh: Recommended shape

The displacement part of the dynamic system works per-instance. This means that any grass mesh used with this system should not be too big. The default included grass meshes have a shape that works quite well with the displacement interaction effects, so they can be used as a guideline for creating your own custom grass meshes for the system:



As you can see, it's recommended to make your grass meshes taller than wider and reasonably small. It's also good to taper the bottom a little bit – this should prevent occasional bottom mesh edges being 'in air' during extreme displacement effects.

Also, when preparing your grass texture, try to fit the mesh plane edges to the texture as closely as possible to minimize 'empty' spaces. The more unused space on the mesh plane, the more overdraw it will generate – and this results in bigger performance hit.

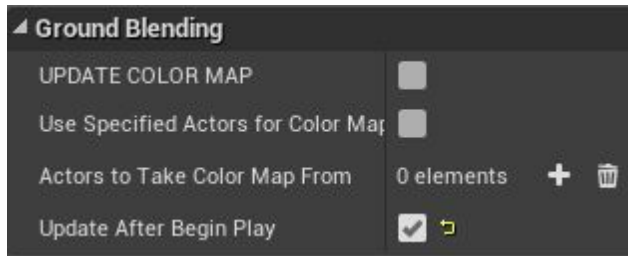
Tip: Grass plane normal

Default included grass mesh planes use edited normals to achieve better look & to play nicely with the 'trampled tint' material feature that enhances the look of deformed grass instances.

To create similar meshes, in your 3D app edit mesh normals to make them face **upwards**. If you have a trouble with setting this up, feel free to ask on the support email & I'll help you.

Ground blending & default grass material

The default included grass material can blend the bottom of grass/foilage meshes with current ground texture. By default it takes first Landscape actor and automatically generates a color map to use with the material. If you want to pick specific actors (e.g. when your level is made out of StaticMesh boxes), there's also an option for that:



... The parameters are available in the *InteractiveGrassManager* actor (make sure to place one in your scene) and have additional explanations in their tooltips.

You can also control ground blending intensity or turn it on/off in the material instance. Default grass materials can be found in the **Materials\Grass** folder.

Besides ground blending, you can adjust things like:

SimpleWind – adds simple wind on top of directional wind

AlternativeSimpleWind – uses even simpler, more optimized wind effect

CloudShadows - turn them on/off

ColorizeTrampledGrass – this effect enhances displacement effects by colorizing instances based on their orientation. For example, trampled grass will appear a little bit brighter

ColorVariation – color variation for grass instances – randomly, slightly changes color of every instance

GroundBlending – turn it on/off

MaterialInteraction – grass deforming around actors with the *InteractWithGrass* component

swayFromStoppingMainActor – it will perform sway animation for grass around main character when he stops moving. It's an additional effect, only for main character

UseNormalMap – check this if you want to use custom normal map for your grass material. By default it uses a constant value for normal, as dedicated normal map is not always necessary, e.g. for stylized grass.

Wind – main wind effect. It's directional and can be affected by wind direction & strength.

CloudShadowTint – tint of cloud shadows

Tint – tint of main grass texture

TrampledTint – tint of trampled grass

SimpleWindIntensity, SimpleWindSpeed – intensity & speed for simple wind

GroundBlendFalloff – falloff for the ground blending effect

MaterialInteractionStrength – how strong the interaction effect will be for this material instance. Useful for defining interaction behavior of individual foliage types

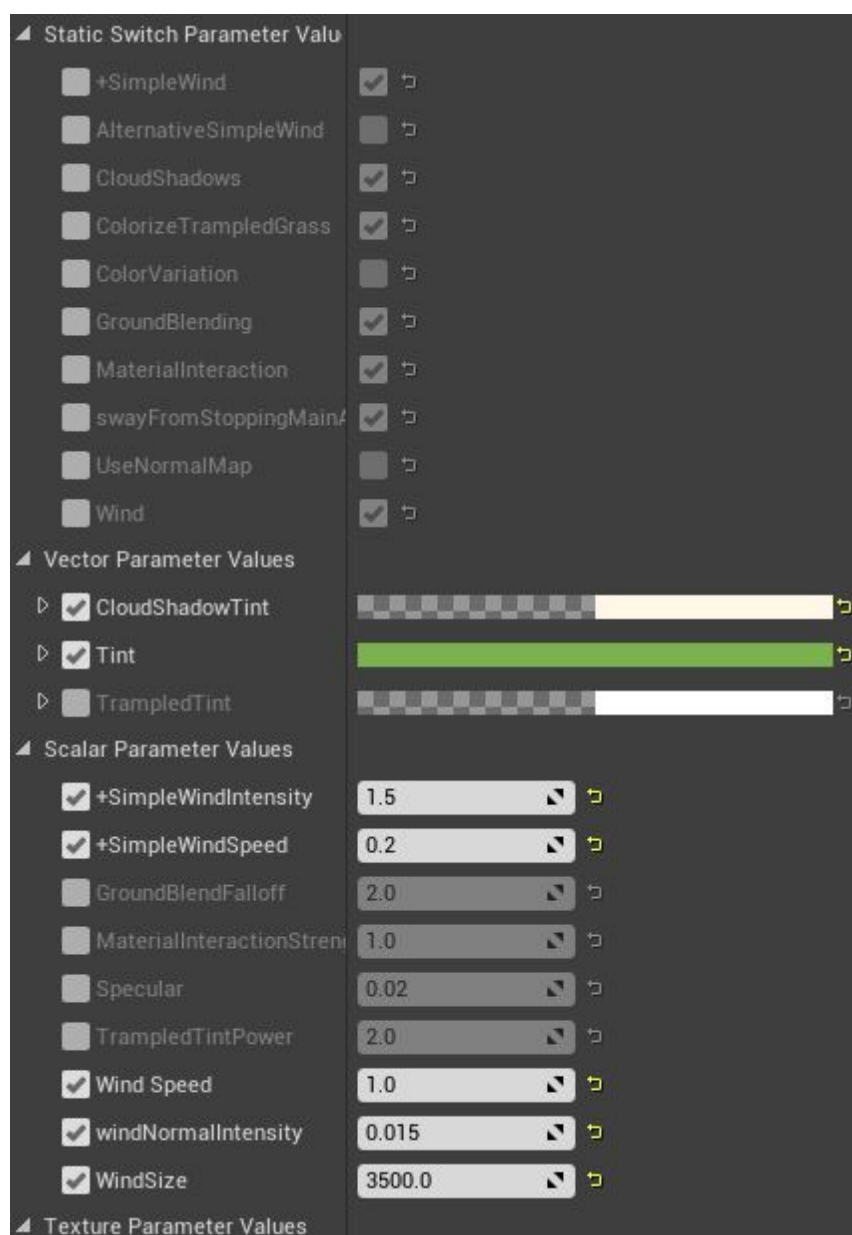
Specular – specular value, higher values tend to add unwanted shine

TrampledTintPower – intensity of the 'TrampledTint' effect

Wind Speed – speed of directional wind

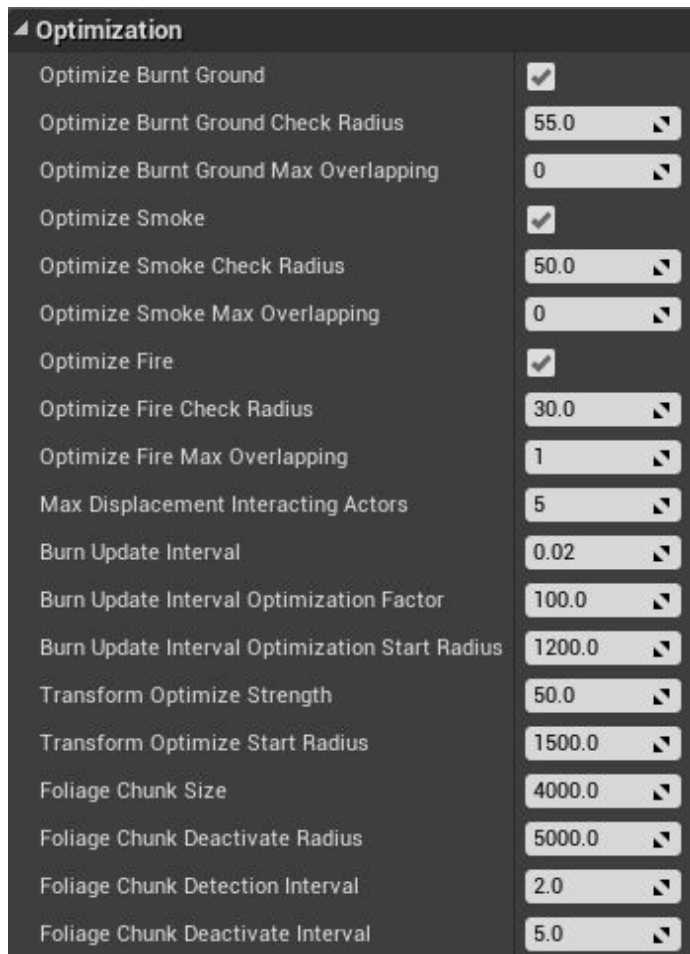
windNormalIntensity – increasing this value will emphasize the wind waves

WindSize – size of wind wave effect.



Optimization parameters

The Dynamic Grass System has some automatic optimization mechanics that prevent CPU lag spikes. It works automatically, but you can adjust its strength in the *Optimization* section of *InteractiveGrassManager*:



Some games require more 'brutal' optimization, some less – it all depends on things like map size, number of interacting actors, grass density, etc. So if you notice that some of the optimization methods are too harsh, or maybe they need to be more strict, you can change it here.

All the parameters listed on this picture are explained in their tooltips.

... Not only grass!

You can use different meshes with the system – small bushes/shrubs will also work with it. There is also a special burning effect for bushes/shrubs that uses dissolve effect instead of top-to-down burn effect which is used for regular grass.

There are some simple shrubs included in the project and you can see that they have different parameters in their Foliage Components. You can also find e.g. a single wheat piece that will act differently than a regular grass – for example, it will un-deform much faster.

Tip: Distant flickering fix

If you experience flickering of your custom grass meshes (mostly visible with large grass draw distance, at great distances) then increase your mesh bounds in Z axis. You can do this in your static mesh Details panel, under Static Mesh Settings -> Positive Bounds Extension.

You can find an in-depth explanation of the problem [here](#).

Demo levels & examples



In the first demo level, walk to the white pillars to enable their example mechanic presentation. Available examples:

- UFO that lands, makes a mark on the wheat field, hovers off and flies away
- Rolling physics ball that tramples grass
- NPC spawner – each NPC will interact with grass
- Fire starter with special foliage type that burns faster and with different flame & smoke color + propagates only to the same foliage type
- In-game grass spawner with additional ground blending showoff
- Example vehicle that tramples grass beneath wheels
- Example harvester vehicle that harvests wheat

Besides that, you can try more example mechanics by pressing the keys that are listed on the screen. Press *Insert* to show/hide the controls.

The demo map is covered in dense grass with very far view distance to demonstrate the optimization – on GTX 760 it maintains 60 FPS with V-sync, 1920x1080, Epic quality settings and of course, full dynamic lighting.

The second demo level named *InGameGrassGeneration* demonstrates mainly the *GrassSpawnArea* & grass spawn blockers in action.

Both the demo maps can be found in the **/Maps/** folder.

Ending

This document covers some basic system mechanics, but it's best to check the included demo levels and play with the working examples.

If you have any questions, feel free to ask on contact@unknown-origins.com or in the forum support thread.

Thank you!