Ignis -Interactive Fire System

support@arctibyte.com

https://support.arctibyte.com/

V 2.1.5+

Updated 10.11.2022

Quick-start

Works at least on

Unity 2019.3+ (Visual effects 7.3.1+)

URP/HDRP (7.3.1+)

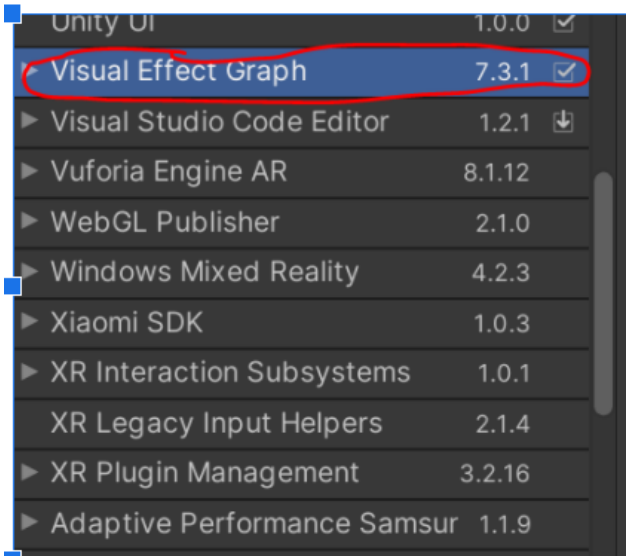
Built-in/Standard RP with exception of one optional/extra shader. (See FPS demo recorded in Standard RP on youtube for confirmation: https://www.youtube.com/watch?v=tZ1p0ilj_CY) (Visual effects 7.3.1+ excluding 10.4.0+ and 7.6.0-7.7.1). Also Ignis currently does not support Unity Editor 2021.2.7+, since Unity dropped the support for downgrading visual effects graph.

ATTENTION: Currently visual effects graph 10.4.0+ and 7.6.0-7.7.1 may not support Standard/Built-in Pipeline. Instructions for downgrading in FAQ/I am using Visual effects Graph NN.NN.NN+... (Everything works in Visual effects 10.2.0 and HDRP/URP works in 10.4.0)

Instructions

Install Visual Effects Graph from Package manager:

- Window/Package Manager -> Unity Registry (upper left corner) -> Visual effects graph
- Install

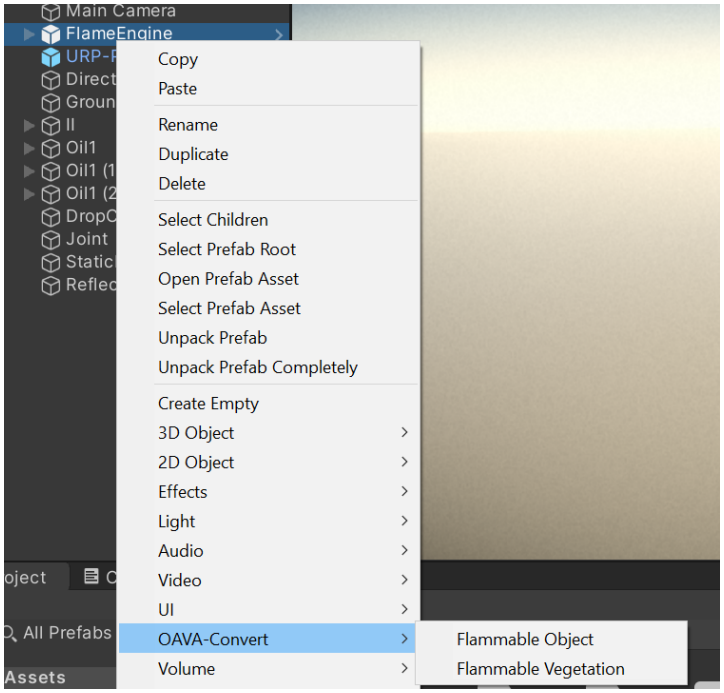


Check that everything is working correctly:

1. Open HDRP/URP/built-in Demo scene from OAVA-Flame/Scenes
2. Press play.
3. IF YOU GET ERRORS IN CONSOLE / FLAMES DO NOT RENDER
    a. Error: Value of name ‘Fire…..” was not found -> Open every OAVA-Flame/VFX/Flame… and close it.
    b. Shader error in ‘Hidden/VFX/…’ redefinition of ‘PackHeightMap’ -> see FAQ/I am using Visual Effects Graph 10.4.0+… OR I am using Visual Effect Graph 7.6.0-7.7.1 with Standard/Built-in pipeline and I cannot see flame VFX
    c. If you are using built-in pipeline see the requirements from the asset store page (Unity editor less than 2021.2.7 and downgraded VFX Graph)

Use the system:

1. Backup materials you want to convert, just in case.
2. Select the objects from the hierarchy you want to convert to flammable:
3. Press right-click
4. Select OAVA-Convert-> Flammable…



5. Untick materials you don’t want to burn.
6. Press yes
7. Voila. Now your object is flammable.
8. Set the right settings:
    a. Add **box colliders** to your object if it doesn’t have them already.
    b. **Or tick** “Calculate flammation area from mesh”. This only works for simple meshes.
9. Test out by ticking “Set Fire on Start” on the “Flammable object” script attached to your object

Usage

Convert a solid object and want to use flammable shader (Burnt and crackling effect)

Example Use Case: You are using HDRP/URP, You want a chair to be able to catch up in flames. You want to use a shader provided in the Ignis package for extra touches.

1. Select object(s) you want convert in object hierarchy (scene or prefab)
    a. Can be a parent which holds many LODs or child objects that you want to act as a single flammable object.
    b. Or can be a single object.
2. Right-click -> OAVA-Convert -> Flammable object
3. Select materials you want to convert and click yes (can be ctrl+z:d)
4. Now objects you had selected have a custom shader (or custom shader is on their children) and script called “Flammable object”
5. Place box colliders on the object/children to have precise flame and spreading positions **OR** Select *Flammable Object -> Calculate flammable area -> Object* for approximate area around object. If the object is not rectangular it’s usually good to tick “Use Mesh Fire” as well to calculate flames around the mesh.
6. Customize the rest of the options.

Convert a solid object and want to other/own shader

Example Use Case: You are using your own shaders/some shader from the asset store or you are using Standard RP.

1. Follow instructions above but Untick the materials as you convert
2. If the shader is supported it should be automatically automated. Otherwise you can still use the flame VFX and the system without the shader.

Convert a vegetation object

Example Use Case: You want your grass/fern to be able to catch fire.

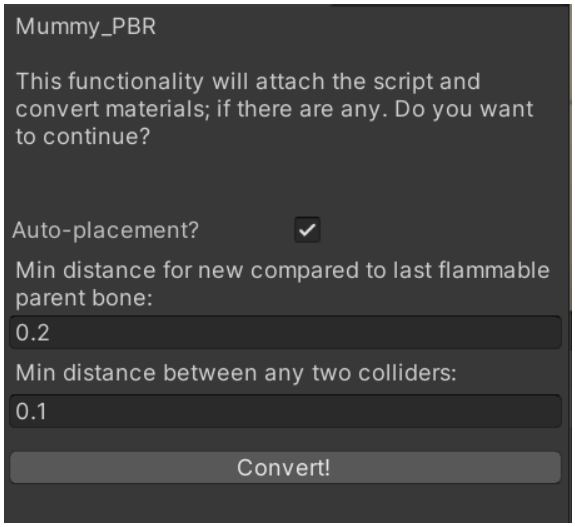
1. Select object(s) you want convert in object hierarchy (scene or prefab)
    a. Can be a parent which holds many **LODs** or child objects that you want to act as a single flammable object. Ignis can handle all renderers in the children.
    b. Or can be a single object.
2. Right-click -> OAVA-Convert -> Flammable Vegetation
3. Now your Vegetation should have script called “Flammable object”
4. You can customize the burning areas by adding box colliders (can be triggers) or just leave the “calculate flammation area” to vegetation for automatic calculation
5. If the vegetation object is large it is usually good to tick “Use Mesh Fire” for a more unique and non-uniform look.

Convert Skinned Mesh

Example Use Case: You have modeled an animated enemy or character. You want the animated object to be able to ignite in flames.

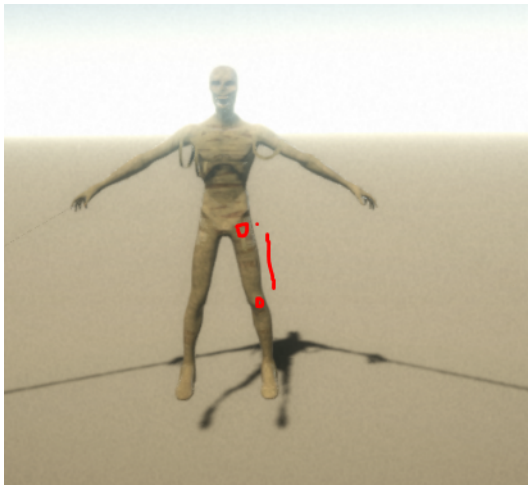
The flames can be attached to the bones of the skinned mesh.

1. Select object(s) you want convert in object hierarchy (scene or prefab)
    a. Can be a parent which holds many LODs or child objects that you want to act as a single flammable object.
    b. Or can be a single object.
2. Right-click -> OAVA-Convert -> Flammable Skinned Mesh



3. Set the Auto-placement variables.

Min distance between bones is the minimum distance of two flame spawns when moving along the bones.



Min distance between any means minimum distance between any flame spawns (Not comparing against the last bone spawn).

4. Click convert and follow the usual setup steps mentioned in “Convert flammable object”
5. The script will create colliders for spawning the flames inside the skinned meshes bones. You can also set up the colliders yourself if you want specific parts of the skinned mesh to burn. Just add box colliders to the desired bones and set them to “IsTrigger” if you want.

Convert a tree

Example Use Case: You want your grass/fern to be able to catch fire.

1. Select object(s) you want convert in object hierarchy (scene or prefab)
    a. Can be a parent which holds many LODs or child objects that you want to act as a single flammable object.
    b. Or can be a single object.
2. Right-click -> OAVA-Convert -> Flammable Tree
    a. The functionality will enable “Use mesh fire”. Tree is a large object which should have many smaller fires within it so this functionality will place them automatically.

   b. If you want to enable/disable it for performance reasons you can do it in “Flammable Object->System->Use mesh fire”
3. Done! Click FlammableObject->Flame VFX->Preview flame and start customizing the flames for your object.

Use with Unity Terrain Trees

Example Use Case: You are using built-in Unity terrain in your project. You want some of the grass or trees or everything in your terrain to be able to catch flames.

Additional information:

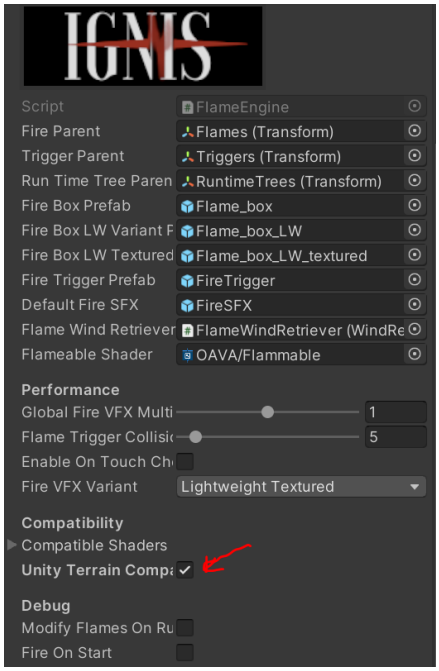
Backup your terrain! Ignis will manipulate trees in runtime, so in case of a rare blue screen on your computer you can lose data!

Ignis can be used with Unity Terrain (Prefabs placed as trees. Can be bushes etc as long as they are placed with tree tool). Ignis checks with predefined intervals if there are any fires close enough of unity terrain trees. If some tree is close enough, Ignis will replace a tree instance with a prefab on the same position/scale/rotation as the tree. After that it will act like any flammable object and is deleted if it is not ignited and the fire moves farther away.

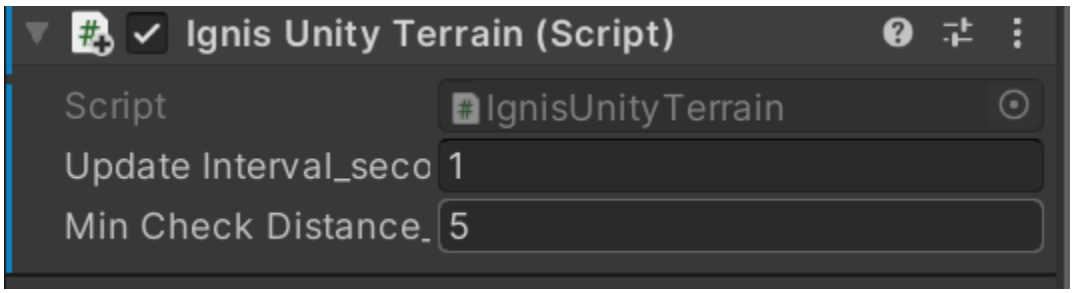
If your scene has a huge amount of trees and huge amount of flammable objects, be sure to cull far away/non-important flammable objects and trees to avoid unnecessary checks and draining the performance. Also this will leave burnt gameobjects (Flame already burnt away, whole life cycle completed). So if you have a large scene you should replace these with burnt tree instances or destroy them when the player is farther away.

How to set it up:

1. **Backup your terrain!** Ignis will manipulate trees in runtime, so in case of a rare blue screen on your computer you can lose data!
2. Enable “Unity Terrain Compatible” from FlameEngine in your scene.

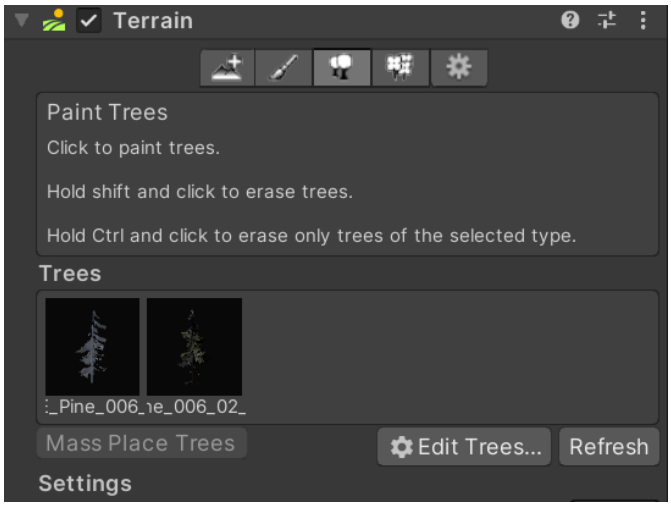


3. This will spawn a new component:



4. You can adjust these variables, if you want to optimize performance/smoothness, but these can also be left alone.
5. Convert your tree prefab you intend to use normally to the flammable object by right-click -> OAVA-Convert->Flammable Object/Vegetation and set up the flames to your liking.

6. Add it normally to your terrain trees:



7. Note that you can add bushes also to trees if you want.
8. Done! Ignis will now spawn invisible flammable objects set up in your tree prefab when flammable objects are close.

Use Vegetation Studio

Example use case: You are using standard Vegetation Studio and want to convert some of the vegetation to flammable objects.

Additional Information:

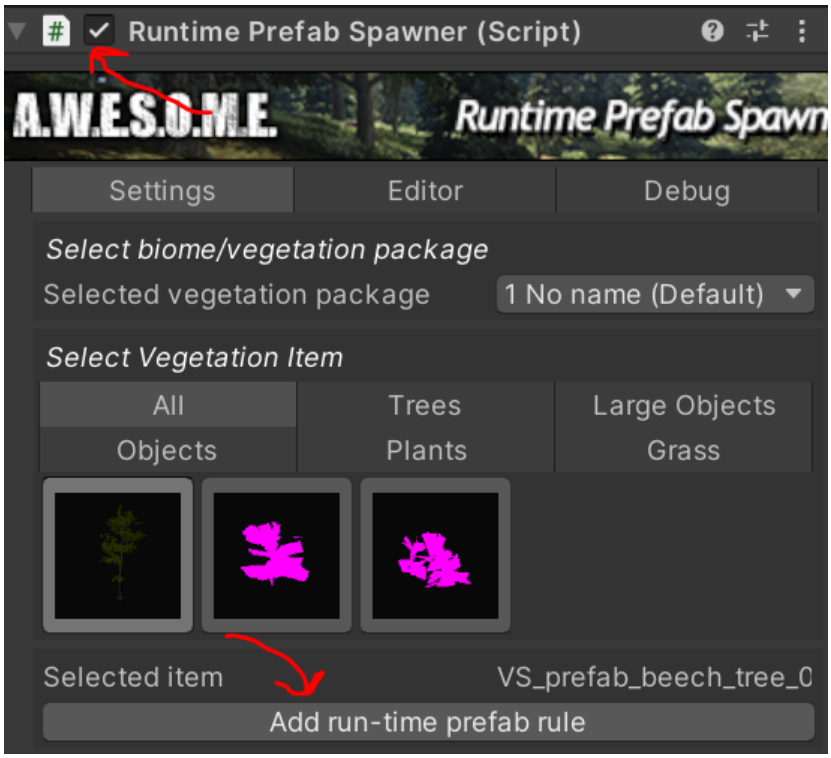
You can use Ignis without shader animation with Vegetation Studio non-pro.

See instructions for **Vegetation Studio Pro** below.

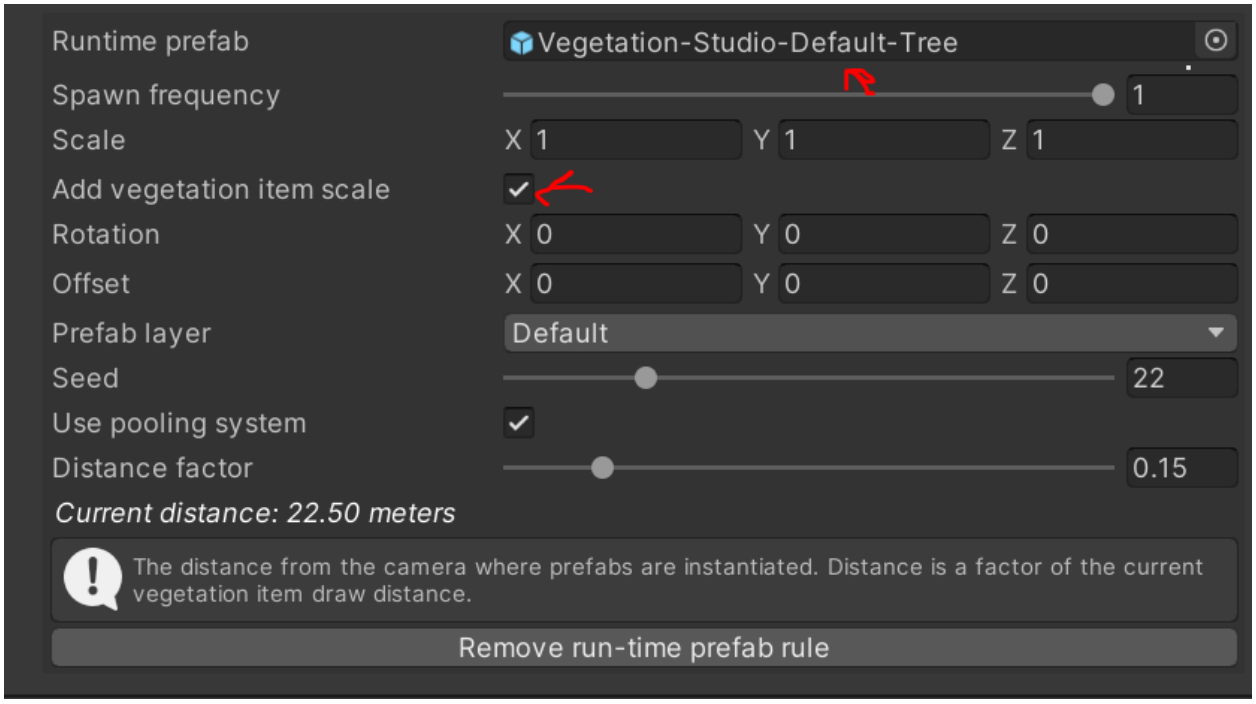
See above “Use with Unity terrain trees” to see how the fire will look without shader animation.

To set up Ignis with objects that you want to burn follow these steps:

1. Enable Runtime Prefab spawner from the Vegetation studio system instance in the scene.
2. Select desired object you want to burn and click “Add run-time prefab rule”



3. Add OAVA-Flame/Compatibility/Vegetation-Studio/Vegetation-Studio-Default… to the Runtime-prefab field. Select bush or tree depending on your object.
    a. You can also make/modify the fire prefab that is assigned here to make better flames specific for your vegetation. Just observe what is inside this prefab and create your own.
4. Tick add vegetation item scale.



5. Done! Now invisible flammable objects should spawn and set your objects to fire.

Use Vegetation Studio Pro

Example Use Case: You are using Vegetation Studio Pro and want to convert some of the vegetation to flammable objects.

Additional Information:

Since objects can be masked with Vegetation Studio Pro, Ignis can mask instanced objects and generate the same prefab as a flammable object when a flammable object would light up the tree/grass/other object.

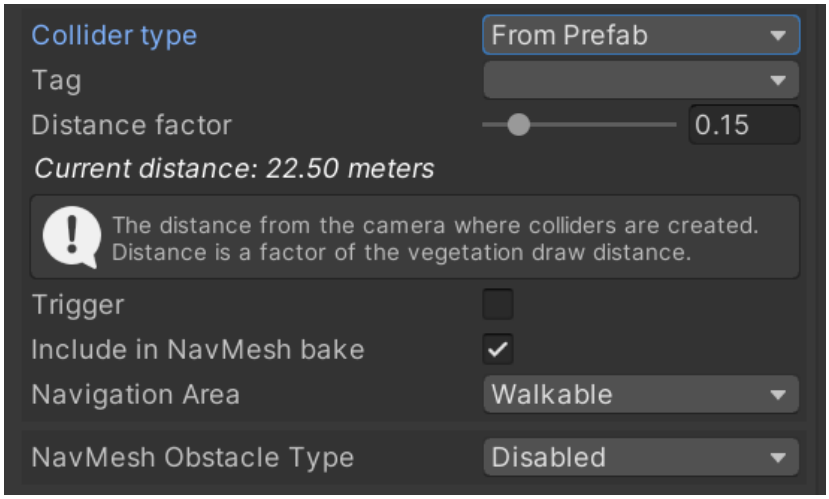
If your scene has a huge amount of trees and huge amount of flammable objects, be sure to cull far away/non-important flammable objects and trees to avoid unnecessary checks and draining the performance. Also this will leave burnt gameobjects (Flame already burnt away, whole life cycle completed). So if you have a large scene you should replace these with burnt tree instances or destroy them when the player is farther away.

How to do this:

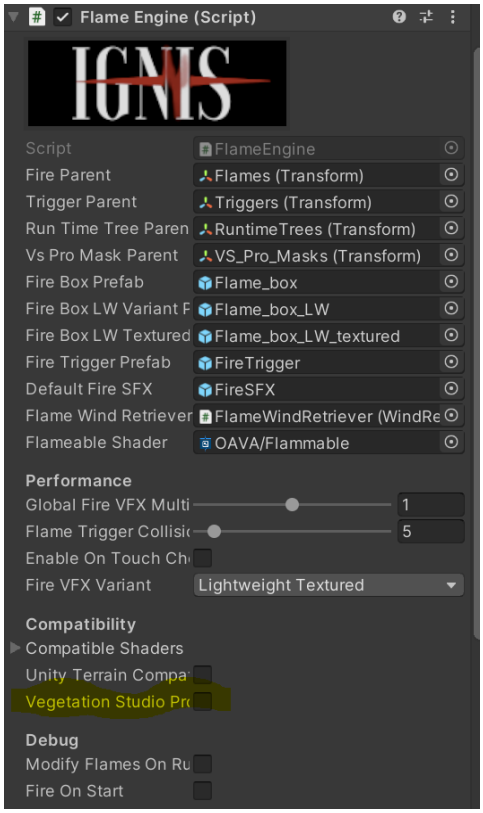
!!! You cannot use the Run-time spawner method described above under “Vegetation studio non-pro” with a masking method since masking a tree will respawn all run-time spawned prefabs and thus delete and reset the burning flames !!!

Trees:

1. Enable a collider for the vegetation object in Vegetation Studio System
2. Collider can be cube, capsule or other primitive or it can come from prefab.

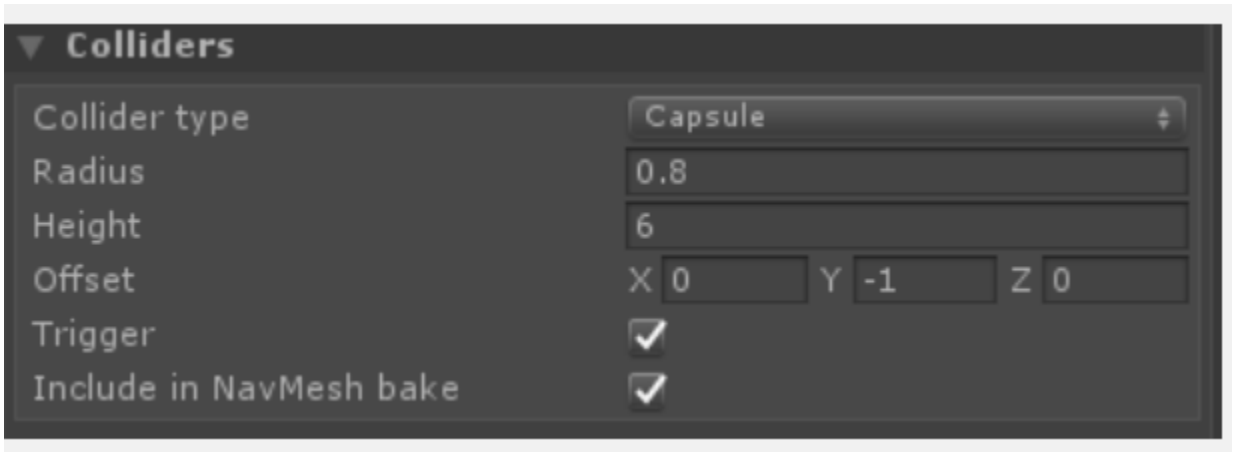


3. Configure the flammable object inside the tree/object/etc prefab like normal.
    a. Grass and foliage - *Convert Vegetation To Flammable* in this document
    b. Trees and other - Convert A solid object to flammable in this document
4. Enable Vegetation Studio Pro Compatibility inside FlameEngine.

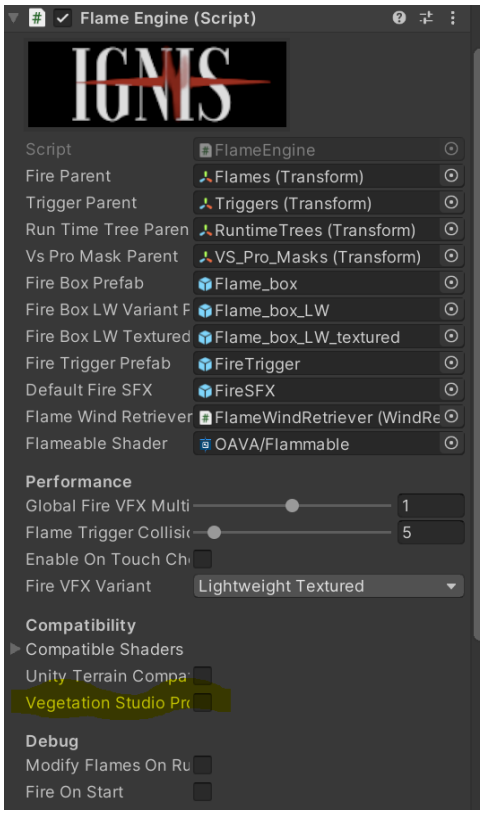


Grass and foliage:

1. Enable a collider for the vegetation object in Vegetation Studio System. For this you need to import grass e.g. as an object (We need a collider to detect if we should ignite it!) You most probably want a capsule or a box collider (or “from prefab”).



2. Tick “Trigger” or Change the collider layer from inside the prefab. We do not want our player to collide with grass.
3. Configure the flammable object inside the tree/object/etc prefab like normal. (*Convert Vegetation To Flammable* in this document)
    a. Grass and foliage - *Convert Vegetation To Flammable* in this document
    b. Trees and other - Convert A solid object to flammable in this document
4. Enable Vegetation Studio Pro Compatibility inside FlameEngine



5. Done! Now your vegetation studio items should flame up.

Manually control flame progress

Example Use Cases: You want to create atmospheric flames or synchronize flames between clients of your multiplayer game. You want manually to control the flame progress and spread.

1. Set Flammable object -> System -> Fire Crawl Speed to 0
2. When Fire Crawl is set to 0, you can see that nothing is happening even if the object should flame up.
3. Even in this mode you need an object to somehow catch flames to tell the system to spawn the culled VFX and start the burn.
    a. You can traditionally use other flames
    b. SetThisOnFireOnStart to spawn the culled flames on start (If you have lot of objects this will eat performance)
    c. Programmatically set the objects on flames in the right moment. Just Call FlammableObject->TryToSetOnFire() with right origin and high ignite power.
4. Now you can control the progress with Flammable Object -> Advanced -> Fire spread and On Fire Timer.

Add wind to affect the fire

Example Use Case: You want flames to be affected by wind like grass.

Additional information: Wind will only affect the VFX and VFX cannot interact with other objects in the CPU world. So if you have a very strong wind and want other objects to catch flame far away according to a wind you should write a component which can manipulate the FireTriggers with wind or add Particle system inside flame which is affected by the wind and attach ParticleIgnite.cs to it.

1. Use TVE Global motion or Unity Wind Zone. They should be supported without any additional setup.
2. Done.

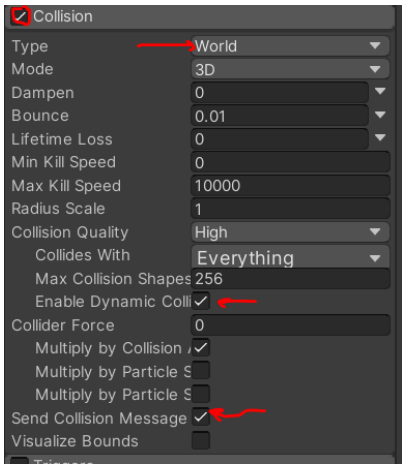
Extinguish flame using your own particle system

Example Use Case: You want fires to be extinguished by your own water particle system for example for firefighter simulation.

Example in Demo Scene: Interact->Water will extinguish one fire after the demo scene has played for a while.

How to Implement:

1. Your own particle systems can be used to extinguish the flame. The extinguish effect is done using particle collisions and can be used with any custom particle system.
2. First you need to make sure that your particle system:
    a. Has enabled Collision
    b. Has collision type-> World
    c. Has enabled Collision->enable dynamic collisions
    d. Has enabled Collision->Send collision message



3. Add “Particle Extinguish” Component to your gameobject which has the particle system.
4. Adjust particleExtinquishRadius and incrementalPower parameters
    a. Should be low with high number of particles
    b. Should be high with a low number of particles.
5. Done. Now you can test the effect and adjust the parameters.
6. Note that if you don’t have any colliders (triggers don’t count) on your object particles cannot detect collisions. In this case:
    a. Use the raycast method explained below
    b. Add a collider with a layer that rigidbodies and other objects do not collide with.

Extinguish flame using raycast/sphere

Example Use Case: You have a fire extinguisher which uses VFX graph to produce the foam effect and thus cannot detect collisions.

Example in Demo Scene: Interact->RayExtinguisher extinguishes one cube in the back after the demo scene has played for a while.

How to Implement:

1. You can extinguish the flame also by raycasting if you don't want to use particle systems which are more expensive for computational power.
2. To do this add a "Raycast extinguish" or "Sphere extinguish" component to your game object and adjust the parameters (explanations in tooltips). Sphere does not care if there are objects in front.
3. After this the extinguishing should happen automatically if the ray hits a collider or trigger which is in the flammable object or a child of a flammable object.
4. You can also call CastRayCastExtinguish() from outside of the component and set Repeating Raycast to false, if you want to raycast only on the specific moments.

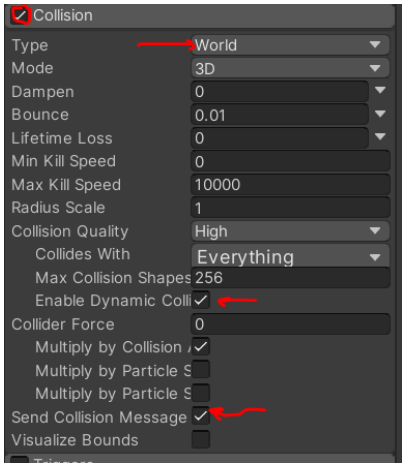
Ignite flame using your own particle system

Example Use Case: You have a mage that can cast a fire spell which uses a particle system. You want the environment to catch fire from the spell.

Example in Demo Scene: Interact->Fire ignites the oil in the front.

How to implement:

1. Your own particle systems can be used to ignite the flame. The ignition effect is done using particle collisions and can be used with any custom particle system.
2. First you need to make sure that your particle system:
    c. Has enabled Collision
    d. Has collision type-> World
    e. Has enabled Collision->enable dynamic collisions
    f. Has enabled Collision->Send collision message



3. Add “Particle Ignite” Component to your gameobject which has the particle system.

4. Adjust IgnitePowerMultiplier and radius parameters

5. Done. Now you can test the effect and adjust the parameters.

6. Note that if you don’t have any colliders (triggers don’t count) on your object particles cannot detect collisions. In this case:

    g. Use the raycast method explained below
    h. Add a collider with a layer that rigidbodies and other objects do not collide with.

Ignite flame using raycast/sphere

Example Use Case: You have a flamethrower which uses a VFX graph to produce the flames and thus cannot interact with objects. You want the flamethrower to be able to ignite objects.

Example in Demo Scene: Interact->RayIgniter can be enabled and it will ignite the oil instead of the “Fire” gameobject.

How to implement:

1. You can ignite the flame also by raycasting/sphere if you don’t want to use particle systems which are more expensive for computational power.

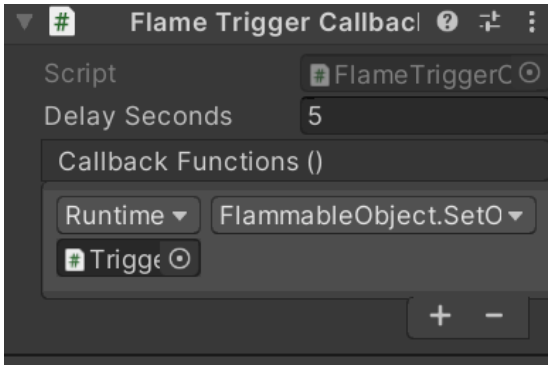
2. To do this add a “Raycast ignite” or “Raycast sphere” component to your game object and adjust the parameters (explanations in tooltips). Sphere does not care if there are objects in front.
3. After this the igniting should happen automatically if the ray hits a collider or trigger which is in the flammable object or a child of a flammable object.
4. You can also call CastRayCastIgnite() from outside of the component and set Repeating Raycast to false, if you want to raycast only on the specific moments.

Trigger event from the flammation

Example Use Case: Explode a barrel after it has burned for some time.

Example in Demo Scene: TutorialObjects -> DropCube has this script which ignites “TriggeredByDropCube” object even when the flames do not touch.

1. You can trigger any function with delay from the moment of flammation of an object.
2. Add Flame Trigger Callback component to your gameobject which has Flammable Object component.

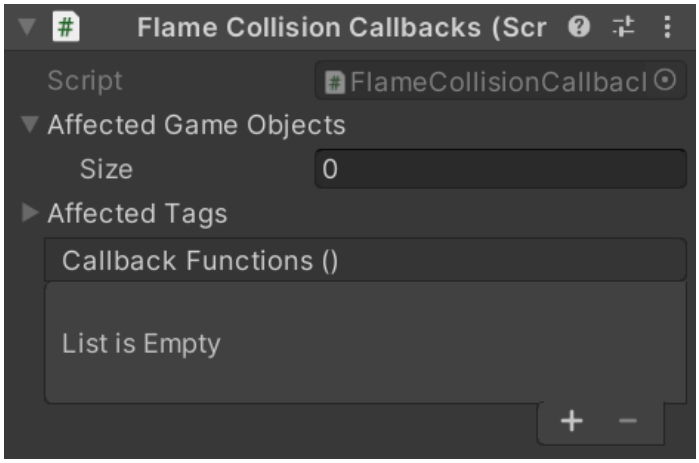


3. Add callback functions that you want to be called after the delay
    a. So if you would want the barrel to explode after burning for 5 seconds: Delay->5.
    b. See https://docs.unity3d.com/Manual/UnityEvents.html for more information on how to use UnityEvents.

[DEPRECATED] Trigger callback when object touches flame

!!!DEPRECATED!!! SEE “Interact when an object touches the flame” below.

1. You can add an event for e.g. damage your player from the FlameEngine Object.
2. Select FlameEngine in your object hierarchy
3. Add FlameCollision Callbacks component or use one on the prefab (there can be multiple of these)



4. Add the gameobject/tags you want to trigger this event
5. Add the functions that you want to be called in case of the event.
6. Done. These are global events and triggered on any fire.

Interact when an object touches the flame

Example use cases: Enemy takes damage when it touches flame, Barrel explodes when it touches flame

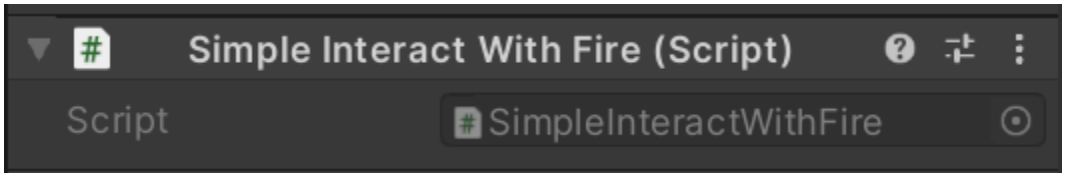
Example in Demo Scene: Scene Setup -> Main Camera has an example script for interacting with flame. If you attach a collider to the main camera and move it to the flames a debug message will pop up.

How to implement:

1. Implement a IInteractWithFire interface to your own class
2. Implement OnCollisionWithFire function with your own functionality. The gameobject provided as a parameter is the burning object with “FlammableObject” script.

```
/// <summary>
/// A template for interacting with fire using IInteractWithFire interface.
/// </summary>
Unity Script | 0 references
public class SimpleInteractWithFire : MonoBehaviour, IInteractWithFire
{
    2 references
    public void OnCollisionWithFire(GameObject burningObject)
    {
        Debug.Log("Object: " + gameObject.name + " Interacted with fire object: " + burningObject.name + " Which had a tag: " + burningObject.tag);
    }
}
```

3. Attach your new class/component to your desired game object.



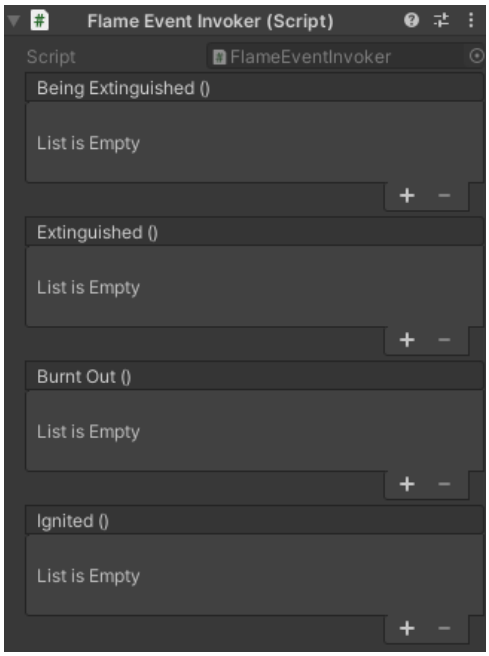
4. Done! Remember to have colliders in your objects when you test this, otherwise flames cannot detect the collisions.

Invoke custom events from flammable object progress

Example Use Case: You want to invoke custom events when the flammable object is ignited, extinguished or burnt out.

How to implement:

1. Add a FlameEventInvoker component to your gameobject which has flammableobject.cs or click FlammableObject->Advanced->Enable Flame Events.
2. Use the invoker like any Unity Event system to call your custom events.



3. More information Unity Events https://docs.unity3d.com/Manual/UnityEvents.html

Pause the flames through script

Example Use Case: You have a custom pause screen, but the flames will keep running if you do not pause them.

How to implement:

1. Call FlameEngine.instance.PauseFlames() to pause
2. Call FlameEngine.instance.ResumeFlames() to resume

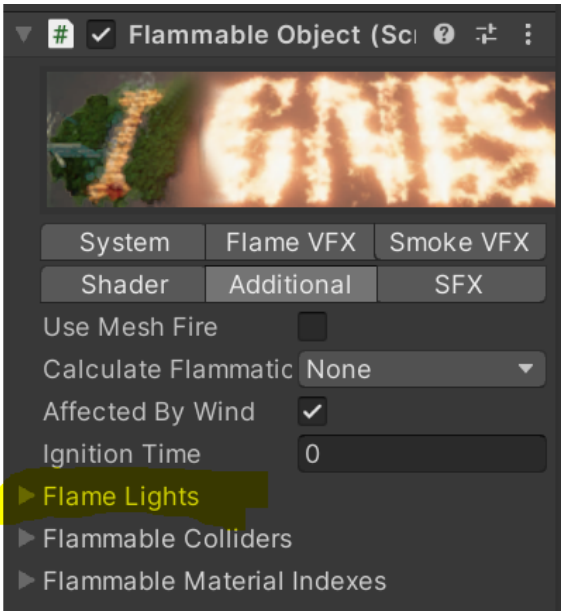
Add Lights to the flame

Example Use Case: You want the flammable object to illuminate a dark area when in flames.

Example in Demo Scene: TutorialObjects->Lights+IgnitedByUser flammable object has illuminating lights attached.

How to implement:

1. You can add flame to trigger global illuminating lights.
2. Add your own flame simulating lights and how you want them to look to scene.
3. (Optional) Add Flame Light Flicker script to the light for automatic flicker and smoothing in.
4. Add the lights to the desired flammable object “Additional/Flame lights” list.
5. Done! Your flames will activate when the object is on fire && fire spread is spread to the point of light.
6. See Sample Scene/Tutorial Objects/Lights + IgnitedByUser object for an example.



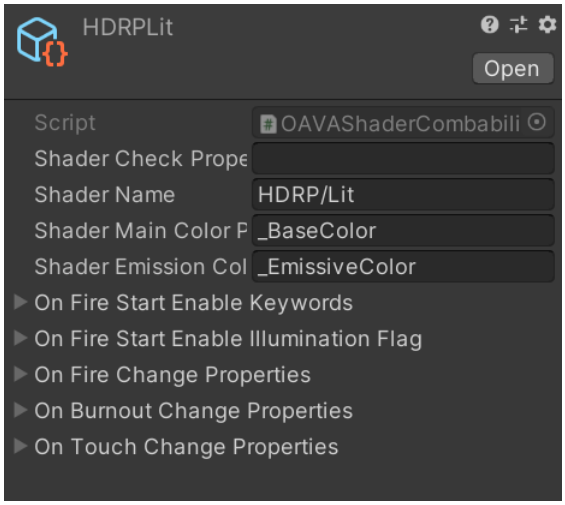
Add compatibility with your/external shader

Example use case: You have an object in your scene that you want to ignite in flames. Your object uses this underground/less-known shader from the asset store, let's call it "Jimbo's awesome party shader" and it is not supported by Ignis out-of-the-box.

Examples in package: There is e.g. The Vegetation Engine shader compatibility created by this same method in the package. You can find many examples in FlameEngine->Compatible shaders.

How to implement:

1. You can add compatibility with any shader that has at least a color parameter exposed.
2. To create a new compatibility object right click on your project assets window-> Create->OAVA-Shader Compatibility.
3. Scriptable object looking like this will be created:



4. It is better if your shader has a property like _IsYourShader, to be checked, but you can also use the name of the shader. These are used to detect your shader.
5. Set the color property names, if you want the colors to be animated. Colors can be later set in FlammableObject component->Shader individually for all the objects.
6. Add float properties to be animated at different life points. E.g. leaves/wind multipliers.
7. Check Combatibility/CompatibleSettings folder for examples.
8. Add your new object, which you created to scene hierarchy->FlameEngine->FlameEngineShaderCompatibility component list.
9. Done. The shader should now be animated automatically. You can also delete the settings from the FlameEngineShaderCompatibility for minimal performance gain.

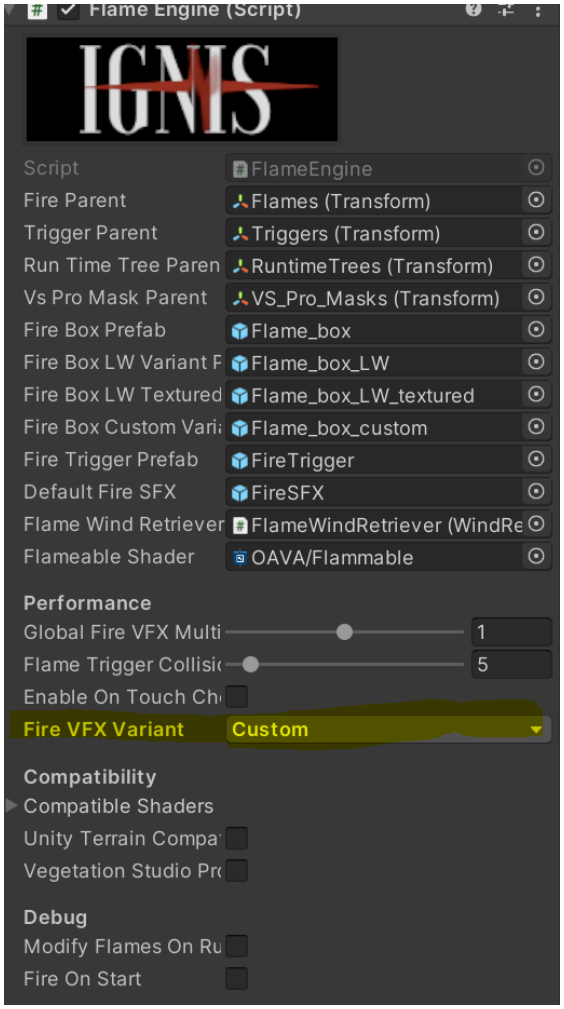
Create custom flame VFX

Example Use Case: You have an extremely stylized game and the flame variants provided in the Ignis package seem out of place. You want to invest your time to create your own VFX graph flame to work with Ignis.

How to Implement:

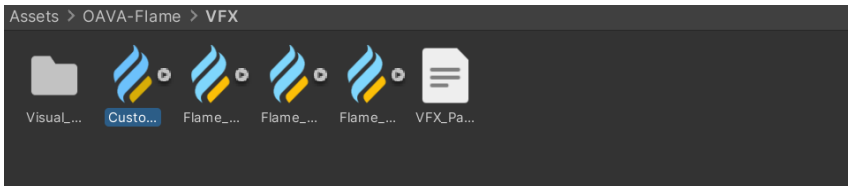
You can create your own custom vfx by modifying the “Custom” VFX inside the Ignis package. Custom VFX is a template which contains only the essentials for Ignis to work. To create your own VFX:

1. Change Flame VFX Variant to “Custom” in FlameEngine gameobject.



2. Modify OAVA-Flame/VFX/Custom_Flame VFX to your liking. Short Crash course to the VFX template:
   - There are 3 systems in this order: Embers, Flame and smoke.
   - The parameters left in the graph are essential for spreading, igniting and extinguishing the flame visually. You can try to disable the blocks to see how it will affect.

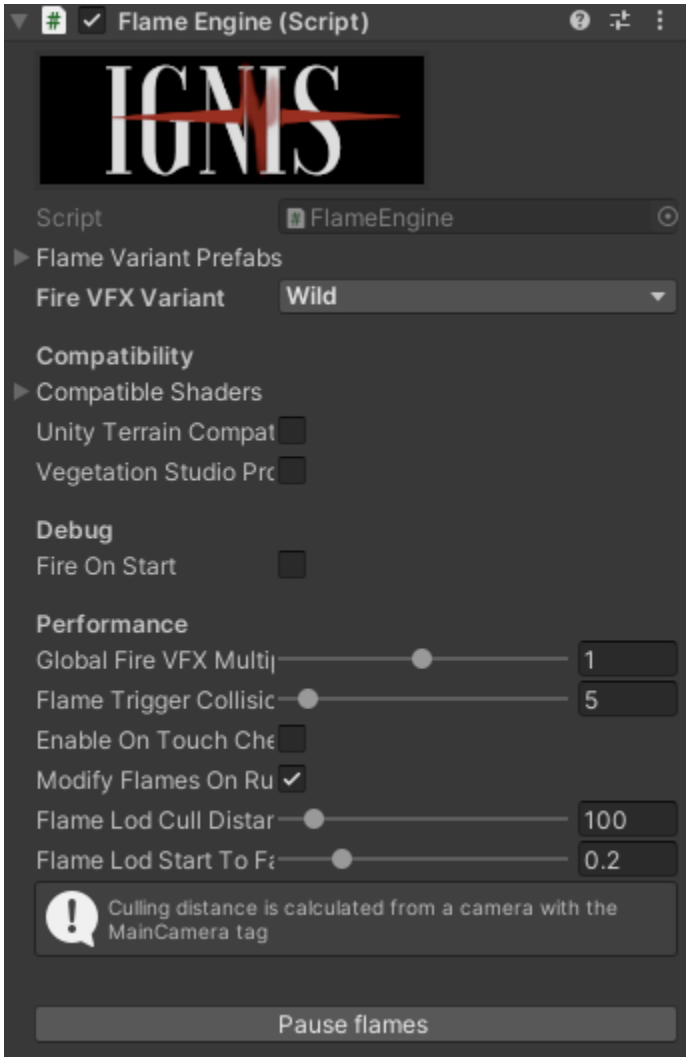
- Inside VFX you will find sticky notes, which will explain my own blocks.



3. Done!

Components

FlameEngine



Here you have the general settings for all the flames.

Flame Variant prefabs: Holds references to different flame variant prefabs

Fire VFX Variant: Here you can choose which flame VFX you want to use.

- **Legacy** (Original vfx with high particle count). Legacy flame from when Ignis was launched.

- **Legacy Lightweight**. Lightweight version of legacy flame.
- **Lively**. Formerly known as Lightweight textured. Produces a slow moving lively flame.
- **Wild.** The default VFX. Most suitable for burning objects without controlled environment/airflow.
- **Gentle.** Most Suitable for stylized games using fire as a calming element.
- **Simple.** Traditional low particle count campfire looking VFX.
- **Old school.** This flame spawns flames as flipbooks. You can easily replace the default flipbook with your own in VFX graph. Navigate to the output node of the flame system.
- **Only Smoke.** Root of this smoke is coloured faking the fire burning. Meant for large fires seen from far away where you cannot see the flame particles.
- **Custom.** Build your own flame! Sticky notes inside the VFX graph

Performance of the graphs from fastest to slowest: Only smoke, Old school, Simple, Legacy lightweight, Lively, Wild, Gentle, Legacy.

All the graphs are pretty fast since they are simulated on GPU.

COMPATIBILITY

Shader compatibility: Holds a list of scriptable objects which can be extended with custom shaders.

Unity terrain compatible: ---Remember to backup your terrain data before using! Ignis modifies terrain data and data can be lost in case of blue screen!--- Is Ignis compatible with Unity Terrain? If you do not want terrain trees to burn enabling this will affect negatively on performance.

Vegetation studio pro compatible: Is Ignis VSPro Compatible? If you don't want to burn Vegetation Studio Vegetation leave this off, otherwise it will negatively impact your performance.

PERFORMANCE

Global Fire VFX Multiplier: Global multiplier for all the flame particle counts

Flame Trigger Collision Check Frequency: How many times in second collisions with flame trigger area is checked? Higher count = more realism, Lower count = More performance.

Enable On Touch Triggers: If you want to melt snow etc from top of the objects enable this (Will affect performance).

Modify Flames On Runtime: If you enable this you can modify the flame parameters such as color and speed on runtime. This is by default false to gain performance.

Flame LOD cull distance: In meters/Units. Flames will be culled when MAIN CAMERA is this far away from flames. If you do not want flames to be LOD culled set this to something very high (frustum culling happens automatically). If this value is e.g. 10 flames will be culled when MAIN CAMERA is 10 meters away from the particular flame.

Flame LOD start to fade percentage: In percent 0-1 = 0-100%. Flames will start to fade away in relation with culling distance from this percentage. E.g. If this value is set to 0.2 = 20% and Flame Culling distance is set to 10, flame will have 100% of the particle count until MAIN CAMERA is 2 meters away and the particle count will linearly decrease until it is culled. This means that with these settings when the camera is e.g. 4 meters away the flame will have 75% of the particles, 6 meters away 50%, 8 meters away 25%, 10 meters away 0%. If you set this value to 1, the flames will not fade, but will be instantly culled when the culling distance is reached.

DEBUG

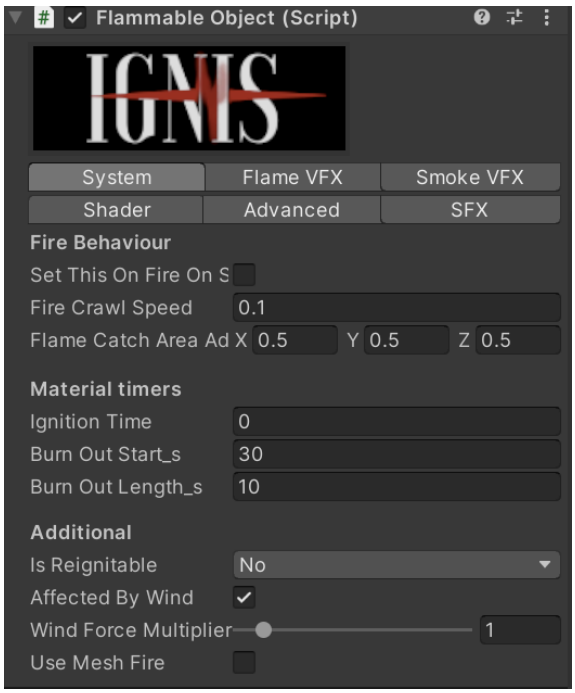
Fire On Start: Sets every flammable item on fire from the start (useful for debugging)

Pause the flames: Pauses the flames

Flammable object

When you convert an object to flammable it adds a flammable object script to it. This is a centerpiece of this system. Each flammable object needs to have this.

System



Set this on fire on start: Sets the object on fire on Start() and OnEnable().

Custom fire origin: Transform to determine custom fire origin. If this is not set object will catch flame on its bounds.min

Fire Crawl Speed: fire spreading speed in m / s

Flame catch area: How big is the area around flames to trigger flame catching (this is added to the initial area)

Ignition time: How long does it take for the object to catch fire. Seconds. Object needs to be under the influence of other fire. If

other fire is taken away before ignition, the object will enter to cool down and slowly return to current ignition progress = 0.

Burn out start: How long does it take for flame to start burning out.

Burn out length: Length in seconds of the burning -> non-burning.

IsReIgnitiable: If you want the objects to be re-ignitable. Useful for example torches that are not meant to be burnt out completely.

- No = Object cannot be re-ignited.
- Only After Extinguish = Object can be re-ignited after extinguishing with particles etc.
- Always = Object can be re-ignited, no matter how it is burnt out.

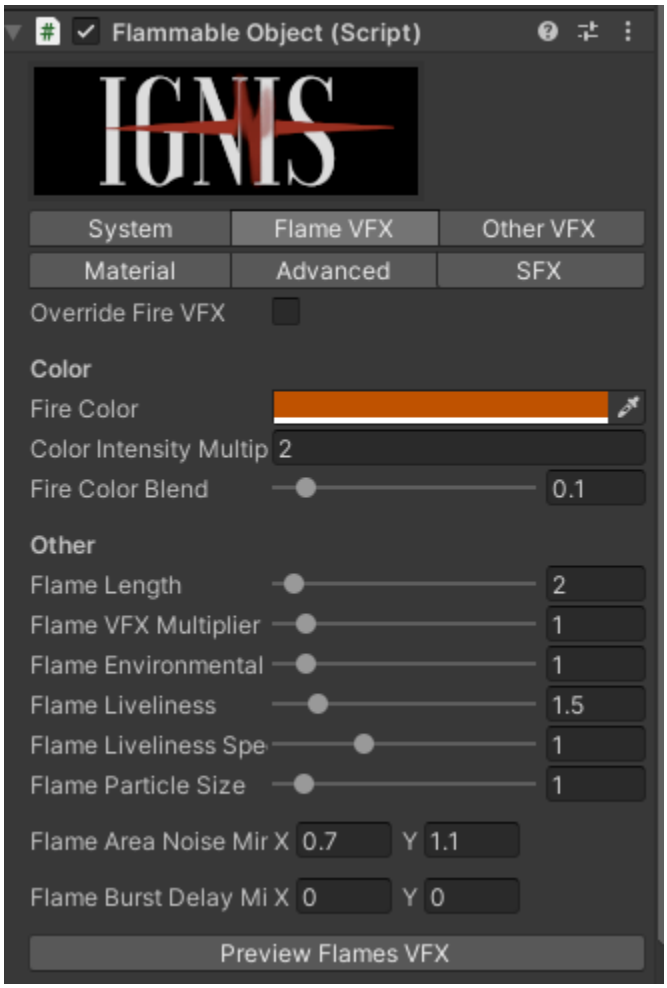
Affected by wind: Does wind affect this object? Untick this object is e.g. inside.

Wind force multiplier: How much the wind will affect the VFX?

Use mesh fire: ONLY FOR NON MOVING OBJECTS. Option to use old-school random flames around the mesh. No box colliders needed. These flames cannot ignite other objects.

- Mesh fire count means actual count of the flames spawned.
- Mesh fire mesh filters: mesh filters you want to include in random point calculation. If you have a multiple LOD object you might want to just reference one mesh filter here.

Flame VFX



Override Fire VFX: Overrides the default fire VFX chosen in FlameEngine. After ticking this you can choose flame VFX just for this instance.

Fire Color: If you want to blend a color to fire (remember to set intensity)

Fire Intensity Multiplier: How intense is the color

Fire Color Blend: How much color should be blended into fire.

Flame length: Essentially sets max lifetime of the fire particles in seconds/4 (this is divided by 4 to keep flames consistent between the earlier versions where fire was much slower).

Local Flame VFX Multiplier: Local multiplier for particle count. Final multiplier is calculated by GlobalVFXMultiplier * local.

Flame Environmental speed: How much upwards motion the flame get? Air generates realistic upwards motion.

Flame liveliness: How lively is the flame? Essentially simulates randomized air flow which keeps the flames particles more together. If this is set to 0 then flame only gains turbulence and upwards/wind force.

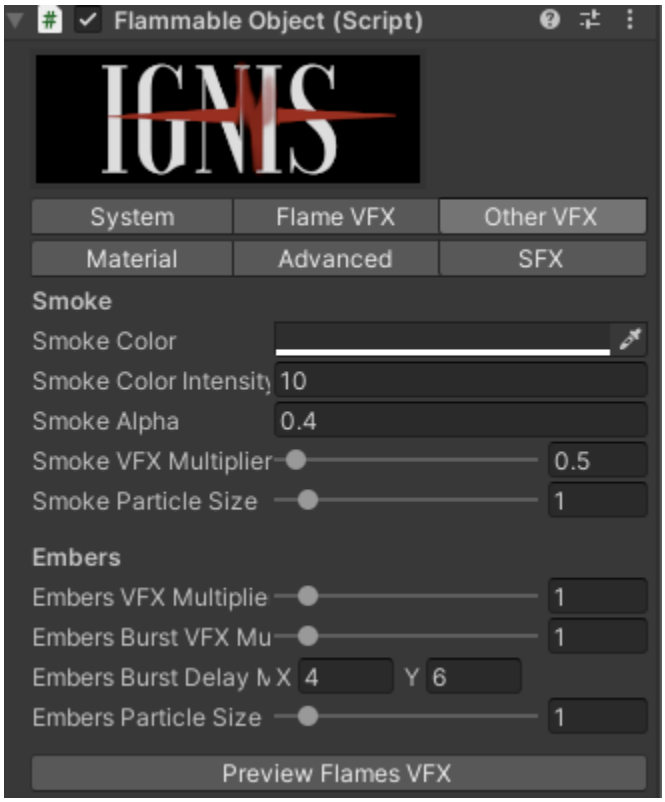
Flame liveliness speed: How fast will the flame random air flows change? Simulates randomness and change in random airflows. 0 = Slow, 3 = fast.

Flame particle size: Simple particle size. If you have large objects this should be bigger than 1.

Flame Area noise min max: *0 - Infinity.* How big is the spawn area noise. X = min, y= max. With vegetation and skinned meshes, flames look better with larger noise.

Flame Burst delay min max: How much there is delay between the bursts. Delay is random between x and y after every burst. X = Min, Y = max.

Other VFX



Smoke

Smoke Color: Smoke color multiplier

Smoke Color Intensity: Smoke Color intensity multiplier

Smoke Alpha: Smoke alpha multiplier

Smoke VFX Multiplier: Smoke VFX local particle multiplier. Total smoke multiplier = Global VFX * Smoke VFX.

Smoke Particle size: Simply the particle size. Adjust according to your object size.

Embers

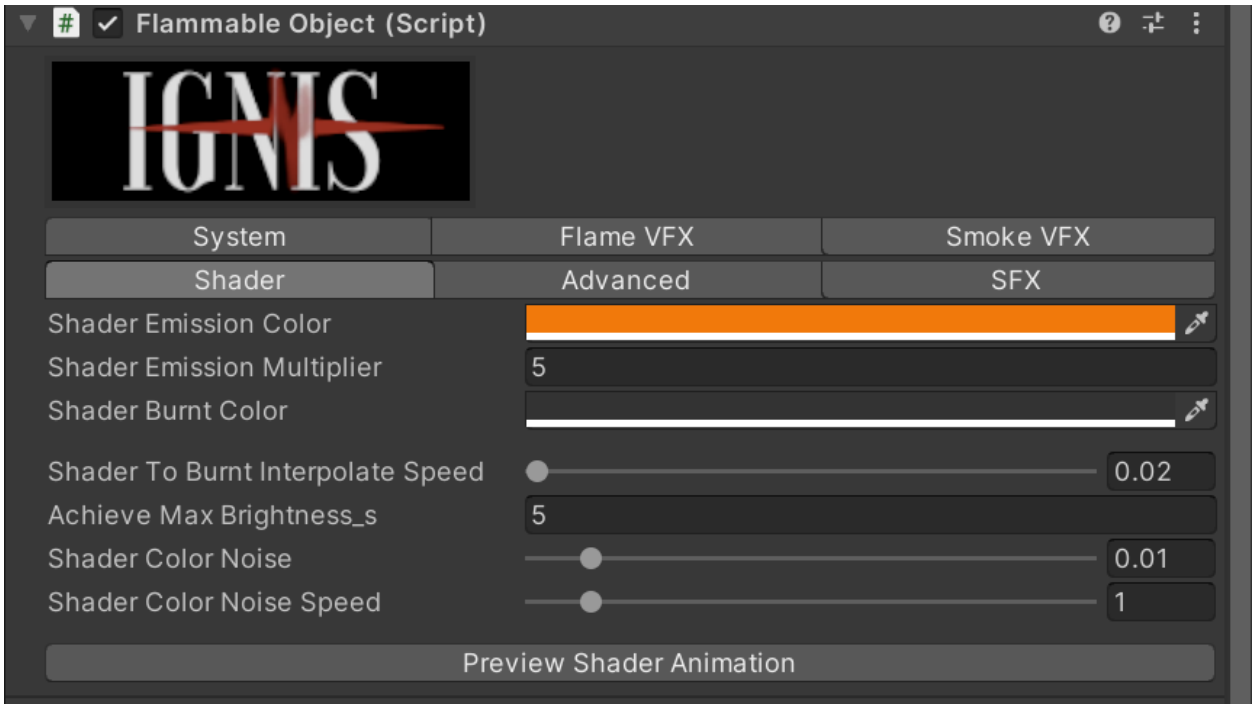
Embers VFX Multiplier: Embers VFX local continuous particle multiplier. Total embers multiplier = Global VFX * Smoke VFX.

Embers Burst VFX Multiplier: Multiplier for delayed ember burst count.

Embers Burst delay min max: x = min, y = max. Minimum and maximum times between ember bursts. The burst delay is randomized between these times.

Embers particle size: Simply the particle size. Adjust according to your object size.

Shader



Shader emission color: Color that is used in shader effect

Shader emission multiplier: Shader emission multiplier. If shader color is invisible consider changing this higher.

Achieve Max Brightness: How long does it take for shader to achieve max brightness after the ignition.

Shader Burnt Color: Only affects the non-Ignis shaders. Burnt color for the shader.

Shader To Burnt Interpolate Speed: How fast the shader will interpolate to burnt color with Lerp. Multiplier. Only for third-party shaders.

Shader Color Noise: Perlin noise multiplier for the shader (Flame flickering effect). If you do not want this set this as 0.

Shader Color Noise Speed: Speed of the perlin noise/flickering.

Preview Shader Animation: Creates an object for debugging the shader animation to adjust the colors etc. Enable Gizmos to force-update in edit mode.

Advanced

Max Spread: If you want to limit the spread to some meters. Leave it high number if you want it to spread infinitely

Delete after burn-out: Deletes the game object after burning out. Use this for invisible objects only if you don't want a harsh "delete effect".

Calculate flammation area from mesh: Calculate approx bounding box from the mesh bounding box. Works best if your object is vegetation or uniformly shaped.

- Object: Adjusts collider to the object

- Vegetation: Adjust collider to the bottom of the vegetation. Vegetation flames look usually better when they come from the root of the object.

Fire spread: How much the fire is already spread (leave this at 0 to use catching effect)

On Fire Timer: How long has the fire been on flames? This is the variable that is checked against the shader animation and burn-out. See usage / Manually control the flame lifetime for more information.

Full Extinguish Toughness: How hard is it to fully extinguish an object AKA start the burn out on the whole object. 0 = single drop of water will start the burn out in the whole object, 1 = Whole object needs to be extinguished before the burn out will start. Formula to start the burn out: if(Extinguish diameter > Object Approximate Size * Full Extinguish Toughness)

Back Spread Cool Down S: Seconds before fire starts to spread back after extinguish attempt.

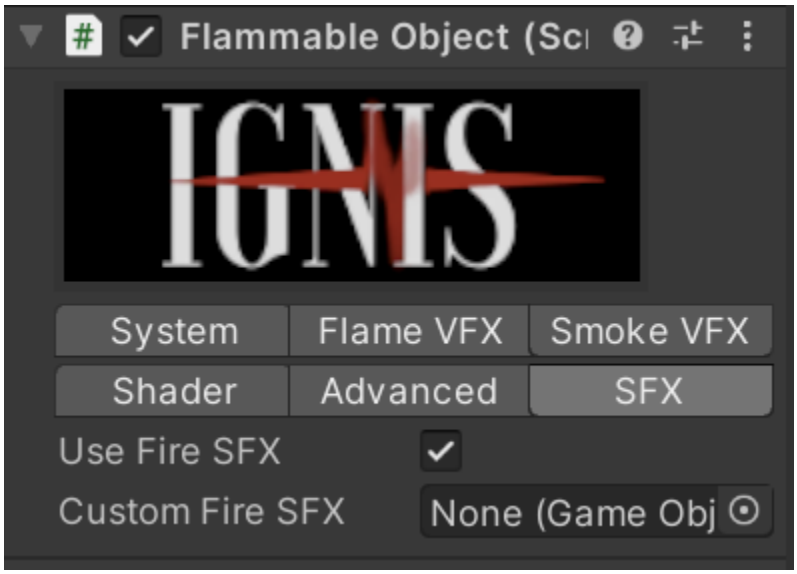
Flame Lights: Lights to be associated with flame. See Usage / Add lights to flame.

Flammable colliders: List of colliders you want to “burn”. If left empty it will use all child colliders by default.

Flammable material indexes: If you do not want all materials to animate even if they are supported, set the indexes you want to animate here.

Enable Flame Events: Enables the flame events. See “Usage->Invoke Custom Events from flammable object progress”.

SFX

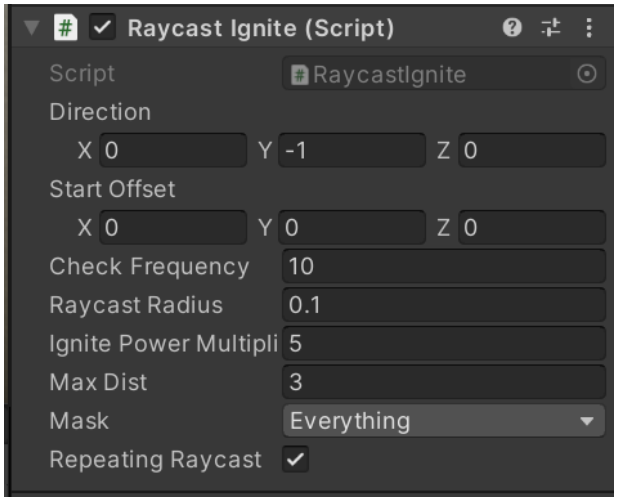


Use Fire SFX: here you can disable the flame SFX if you want.

Fire SFX: If you want to use custom SFX, assign the prefab here. The gameobject is created and moved to the place of the fire at runtime. Leave this empty if you want to use default SFX determined in FlameEngine

Raycast Ignite

With this component you can ignite the flammable objects. Attach this component to your custom object.



Direction: Direction of the raycast (world coordinates)

Start offset: Raycast start offset in world coordinates.

Check frequency: How often the raycast is raycasted (Only if Repeating raycast is checked)

Raycast Radius: Radius of the raycast.

Ignite power multiplier: If an object has Ignite time, how much ignition progress is added with each raycast. Current ignition time += Ignite power multiplier * Time.deltaTime.

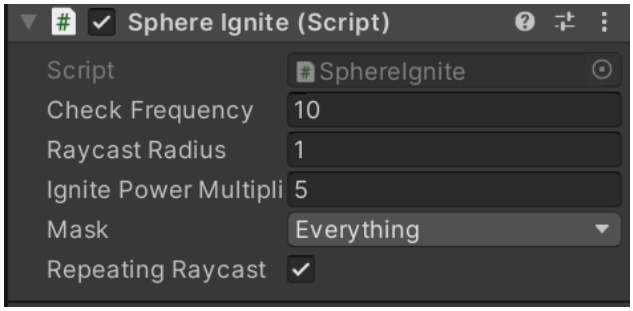
Max dist: Maximum distance of the raycast.

Mask: Raycast mask.

Repeating raycast: Start repeating raycast on Start()

Sphere Ignite

If you want the igniter to not care that there are other objects in front use Sphere ignite.



Check frequency: How often the raycast is raycasted (Only if Repeating raycast is checked)

Raycast Radius: Radius of the raycast.

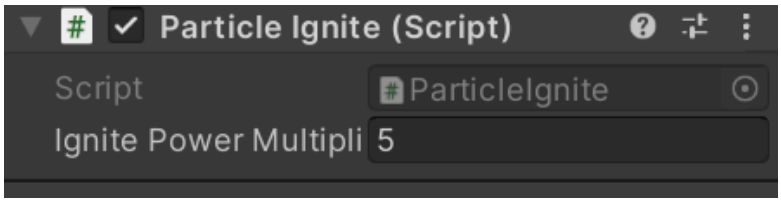
Ignite power multiplier: If an object has Ignite time, how much ignition progress is added with each raycast. Current ignition time += Ignite power multiplier * Time.deltaTime.

Mask: Raycast mask.

Repeating raycast: Start repeating raycast on Start()

Particle Ignite

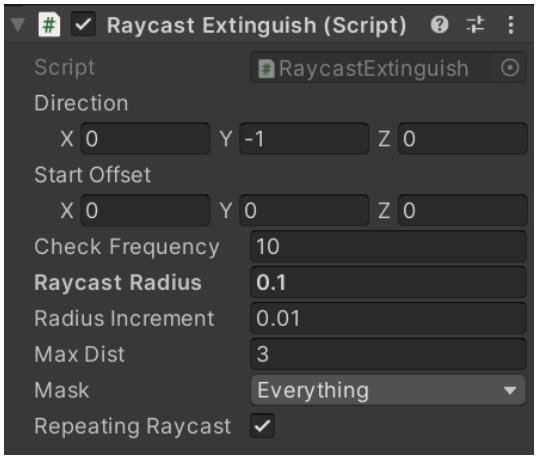
If you want to use your own particle system to collide add this component to your particle system object. Please follow instructions in Using/Ignite using particles to set up your particle system.



Ignite power multiplier: If an object has Ignite time, how much ignition progress is added with each raycast. Current ignition time += Ignite power multiplier * Time.deltaTime.

Raycast Extinguish

With this component you can use raycast to extinguish the flames. Attach this to any gameobject you want to use for this.



Direction: Direction of the raycast (world coordinates)

Start offset: Raycast start offset in world coordinates.

Check frequency: How often the raycast is raycasted (Only if Repeating raycast is checked)

Raycast Radius: Radius of the raycast.

Ignite power multiplier: how much the area is incremented with each hit on the same place (Water flowing effect)

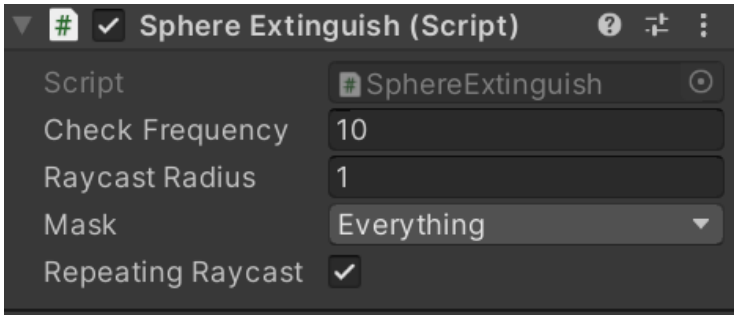
Max dist: Maximum distance of the raycast.

Mask: Raycast mask.

Repeating raycast: Start repeating raycast on Start()

Sphere Extinguish

With this component you can extinguish flames if you do not want to care about colliders e.g. a water bomb.



Check frequency: How often the raycast is raycasted (Only if Repeating raycast is checked)

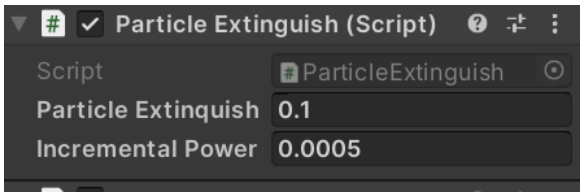
Raycast Radius: Radius of the raycast.

Mask: Raycast mask.

Repeating raycast: Start repeating raycast on Start()

Particle Extinguish

If you want to use your own particle system to collide add this component to your particle system object. Please follow instructions in Using/Extinguish using particles to set up your particle system.



Particle extinguish: area to be affected by one particle

Incremental power: how much the area is incremented with each hit on the same place (Water flowing effect)

Compatibility

The Vegetation Engine

Catching flame system supports The Vegetation engine out-of-the-box. Just add a Flammable object script on the parent/LOD object of your prefabs or use the “Convert” function in the hierarchy menu. Some things are different for different prefabs:

- If the grass prefab is in a large area, you might want to check “Use mesh fire” for better effect, since usually small flame emitters look better.
- To enable snow melt check On Touch Trigger Check from FlameEngine

Vegetation Studio & Vegetation Studio Pro

You can use Vegetation Studio’s awesome runtime prefab spawner to spawn the flammable objects to the root of the bushes/grass/trees. You cannot animate the shaders using this method due to the optimization of the instancer, but you will get Flame VFX. Full instructions how to set this up in Usage->Use Vegetation Studio.

If you have **Pro Version** of Vegetation studio you can have more advanced and better flames with shader animation due to the fact that you can mask objects. Full instructions in Usage->Use Vegetation Studio Pro.

HQ FPS Animated Weapons

Should work without any extra effort. HQ FPS Animated Weapons should include the interaction scripts. If not you can add e.g. SphereIgnite script to your molotov and it should work right away.

NatureManufacture Environments

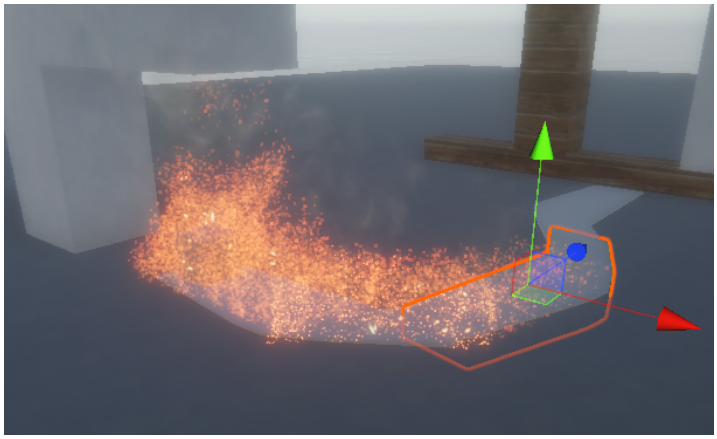
Should work without any extra effort. Shader compatibility settings should be included and applied in the package.

Just add a Flammable object script on the parent/LOD object of your prefabs or use the “Convert” function in the hierarchy menu. Some things are different for different prefabs:

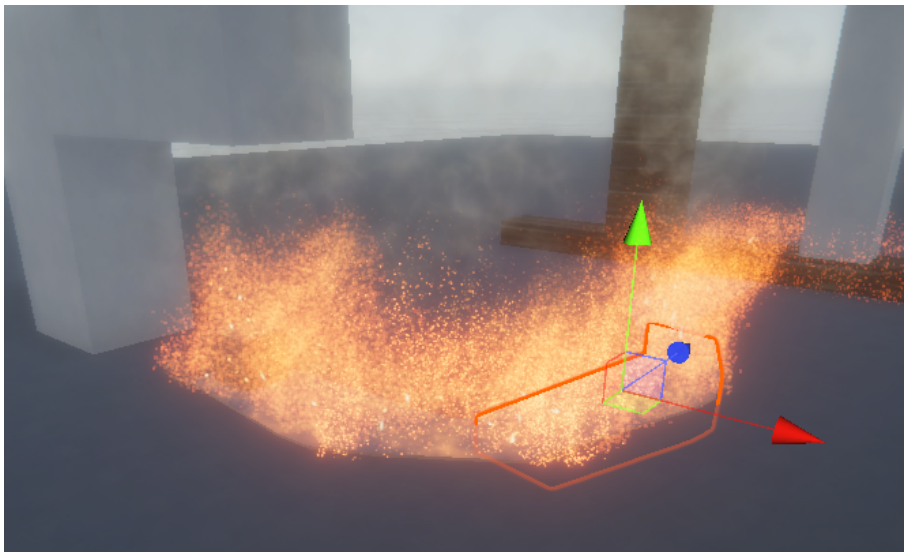
- If the grass prefab is in a large area, you might want to check “Use mesh fire” for better effect, since usually small flame emitters look better.
- In case of a tree you can use manual box colliders to place the flames at certain locations to draw attention (if the player is on the ground it will only harm performance if the flames are in places the player cannot see. Or use automatic mesh fires.

FAQ / Quick fixes

Q: Flame shader does not start to emit flame (HDRP most probably)



A: This is due to the exposure settings in HDRP. Adjust your exposure or “Shader emission multiplier” in the Flammable Object component.



Q: VFX/Shader does not render

A: If flames/shaders are not rendering you may need to Open VFX to compile them.

Open the VFX you are using (or all) OAVA-Flame/VFX/FlameBox... and close it. This will convert the VFX to your Unity Version and Pipeline.

Open OAVA-Flame/Shaders/Flammable and close it

Q: Fire VFX kills my framerate

A: You can adjust the performance settings in the FlameEngine object. If you want to light many fires and render them simultaneously to the camera, please change Fire VFX Variant to lightweight.

Q: My object does not catch flame

A: It probably does not have **COLLIDER or "Calculate Flammation area from mesh" ticked**. Add colliders in places that you want to catch fire (Can be triggers)

Q: My object does not set other objects on fire

A: It probably does not have **BOX COLLIDERS or "Calculate Flammation area from mesh" ticked**. Add colliders in places that you want to catch fire (Can be triggers). The ignition areas are calculated from box colliders. You can still have other colliders (e.g. mesh collider) on your mesh if you want it to catch fire, but mesh colliders cannot ignite other objects.

Q: Snow doesn't melt from my TVE objects.

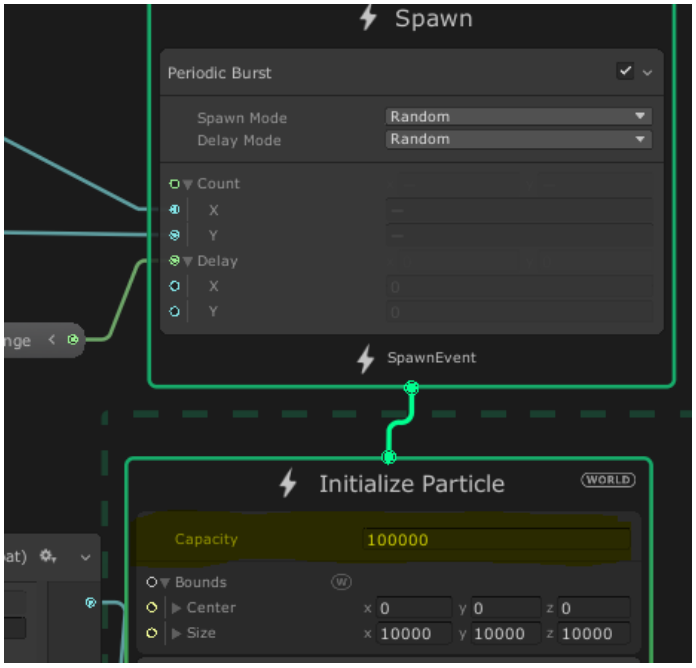
A: You have to enable On Touch Trigger Check from the FlameEngine->Performance. This is by default false to optimize performance.

Q: I want to optimize the fire more

VFX graph system capacity will affect the performance most. The capacity is baked/compiled into the system and cannot be changed dynamically. Therefore, you need to manually adjust it to fit your needs perfectly. See instructions for changing it in the next Q.

Q: I WANT MORE PARTICLES, BUT VFX MULTIPLIER DOES NOT ADD THEM

A: That's because the VFX effect has capacity set to -- to not crash your system and damage your performance. You can add more boxes to your flammable collider/divide them OR scale up the capacity at your own risk. To up the capacity you need to open VFX/Flame_Box and set the capacity of the second from the left system higher:



Q: Flame VFX does not render at certain point / camera position

This can be due to your own occlusion culling, in which case you need to rebake the occlusion culling.

Also this can be due to the LOD culling in the ignis system. Raise FlameEngine->Flame LOD Culling distance to see if this makes any difference. The LOD culling is done by calculating the flame distance

from the MAIN CAMERA so be sure that your camera is tagged as the main camera.

Q: Flammable shader does not render in the built-in RP

Flammable shader is an optional shader provided in the Ignis package and is not compatible with the built-in render pipeline (pointed out in the asset store page). However you can add compatibility with any shader. Follow instructions under “Usage->Add compatibility with your/external shader”

Q: I cannot see the smoke from the flames

This can be due to the smoke alpha settings. Raise the FlammableObject->Smoke VFX->Smoke Alpha value.

Q: Can I use Ignis with Oculus Quest?

Ignis has been reported to be working on Oculus Quest. However you need to build the project to VULKAN instead of OpenGL for it to work. VFX Graph is not yet out-of-preview for mobile platforms so use it at your own risk.

Q: Can I use Ignis on mobile?

The VFX graph has issues with some mobile GPUs so Ignis does not officially support mobile builds. However Ignis works with Oculus Quest, so it could work if you are targeting one specific GPU, but again - use at your own risk and test the compatibility with VFX graph beforehand.

Q: Can I set objects on fire with code?

Yes. Ignis has API documentation included. You can use flammableObject->TryToSetOnFire() function to set objects on fire.

Q: My shader is flashing too much with fire, I don’t want it.

You can change the color intensity and noise from flammableobject->Shader.

Q: Other object does not catch fire even though flame VFX touches it

This is due to the design. Flame VFX particles are simulated on GPU to gain that lightning fast performance and due to that cannot interact with other objects. This means that ignition is done with math. The flammable objects generate boxes that check for other flammable objects around the flame, you can see them under FlameEngine->Triggers. You can change the Ignition area in flammableObject->System->Flame Catch Area addition. This adds an area around your flammable box collider. You can see the ignition area around the flammable object as red gizmos.

If you want the wind to affect the area you can manipulate the ignition triggers or add the flame to emit invisible embers with the Unity Particle system which will catch wind. You need to add ParticleIgnite.cs script to your particle emitter in this case.

Q: I am using Visual Effect Graph 10.4.0+ with Standard/Built-in pipeline and I cannot see flame VFX

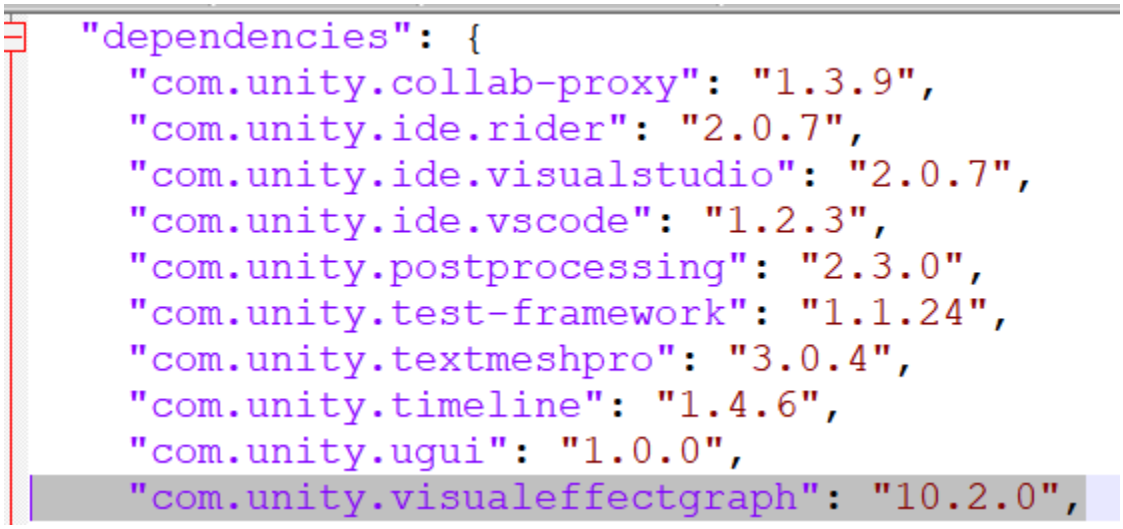
First make sure that you've followed all instructions in README/Quick-start correctly.

The Visual Effect Graph for Standard/Built-in Pipeline in this exact minor version 10.4.0 (Maybe future versions as well) seems to be completely broken and doesn't work at all (even a blank new VFX does not work). I suppose and hope this will be fixed in the upcoming version by Unity. Meanwhile here are instructions for downgrading to previous minor version:

1. Download Visual Effects Graph 10.2.0 from the package manager if you can find it there.

2. If you cannot downgrade from the package manager find this file in your project folder: Your_Project/Packages/manifest.json

3. Change Visual Effects Graph Version to 10.2.0



4. Close and Open your project, wait for the import.

5. Done!

Q: I am using Visual Effect Graph 7.6.0-7.7.1 with Standard/Built-in pipeline and I cannot see flame VFX

First make sure that you've followed all instructions in README/Quick-start correctly.

The Visual Effect Graph for Standard/Built-in Pipeline from versions 7.6.0-7.7.1 seems to be unsupported by Unity and doesn't work at all (even a blank new VFX does not work). Here are instructions for downgrading to previous minor version:

1. Download Visual Effects Graph 7.3.1 from the package manager if you can find it there.

2. If you cannot downgrade from the package manager find this file in your project folder: Your_Project/Packages/manifest.json

3. Change Visual Effects Graph Version to 7.3.1

4. Delete the whole Ignis folder

5. Import Ignis again from the asset store

6. Follow the normal install procedure (Open the VFX graphs etc.)

7. Done!

Q: I can see flames in scene camera, but my player camera cannot see them

A: This is probably a culling issue. Ignis calculates culling from the camera with the “MainCamera” tag, so adding the MainCamera tag to your player camera will probably solve this issue.

Q: I have multiple cameras, how can I use all of them for culling?

You can replace the scene camera in code if you want. Open flammableObject.cs and search for #if UNITY_EDITOR. Here you can see the scene camera position fed into the VFX graph “AdditionalCameraPosition” parameter. Delete everything inside #if UNITY_EDITOR and replace it with this:
fire.SetVector3("AdditionalCameraPosition", your_camera_position);
with your_camera_position being reference to your camera position.

Note that after this the scene camera will not work for culling.

Q: I have a large object and flames look weak

Usually with large objects, such as houses, it is best to design your flames. Use box_colliders in the non-physics layer or “is trigger” to tell the flame positions to Ignis or tick “flammable object->System->Use mesh fire” to automatically place the fires around the object. Also you can play with VFX particle sizes to see the right fit.