BLInformatique — Setting Fire V2.0 – User Guide

Create, spread and extinguish interactive fires
Unity 6000.6.3f1

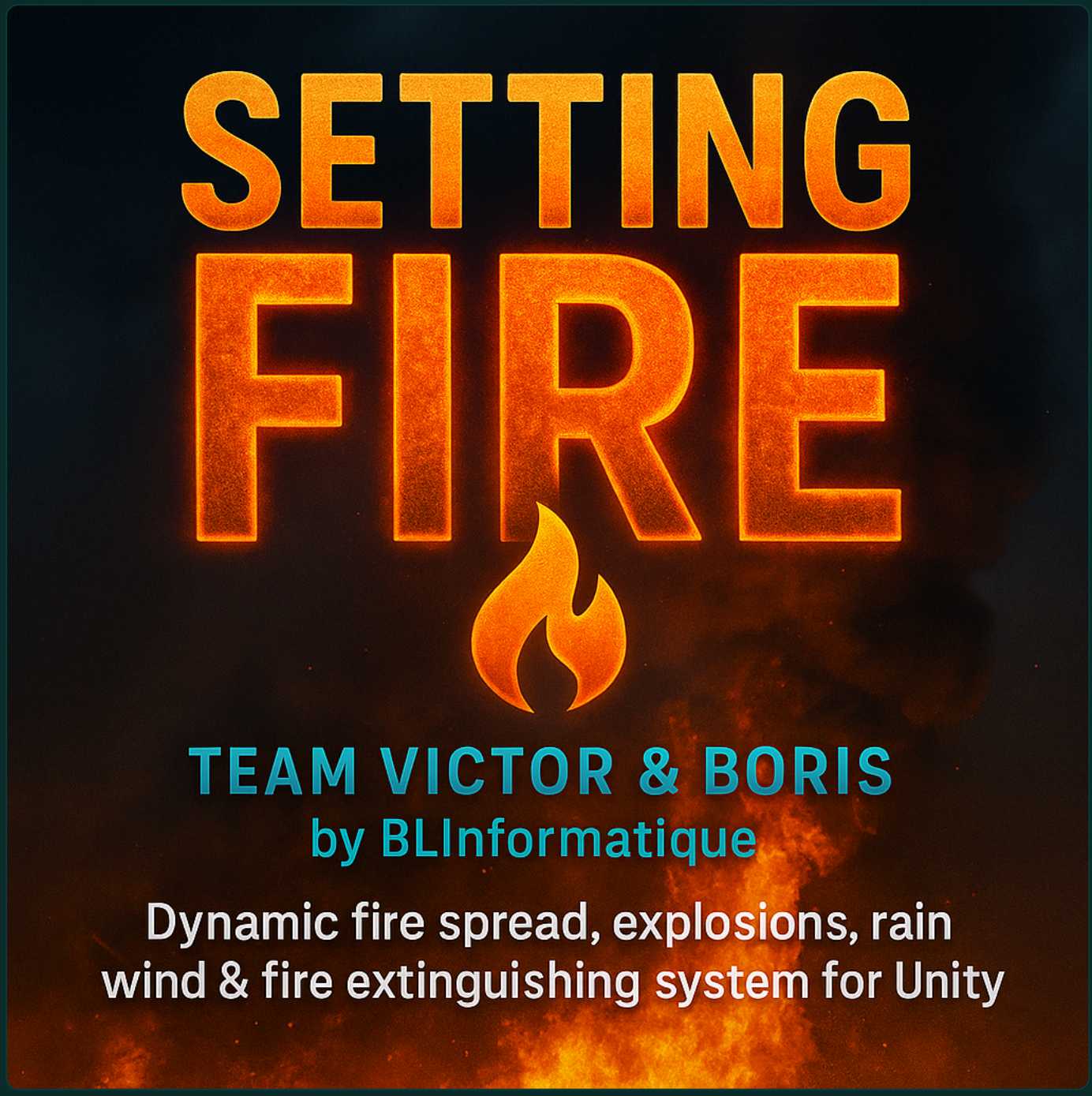


Table of Contents

📖 Introduction

- 📖 Overview
- 🛠️ How To Use
- 🏁 Conclusion

📚 Optional Sections (12)

- 📦 01 — Installation and scene preparation
- 🔥 02 — Your first fire: ignite and extinguish
- 🛠️ 03 — Prepare many objects with the Editor tools
- 🎄 04 — Flame placement and visual effects
- 🌬️ 05 — Propagation, wind and rain
- 🧯 06 — FPS extinguisher
- 💥 07 — Destruction, explosions and events
- 🌱 08 — Terrain burning and optional restoration
- 🎧 09 — Spatial fire audio
- ⚙️ 10 — Scheduling and performance
- ☔ 11 — Optional Enviro bridge
- 🔎 12 — Troubleshooting and validation

📦 Components (12)

- 📦 Components Overview
- 🧩 BLInformatique.SettingFire (12)
  - AutoFadeInParticles
  - AutoFadeOutParticles
  - EnviroRainToggle
  - FireAudioEmitter
  - FireAudioManager
  - FireAudioSetupWindow
  - FireEntity
  - FireEntityMassConfiguratorWindow
  - FireExtinguisherFPS
  - FireManager
  - SettingFireWindow
  - TerrainBurnManager

📖 Overview

Setting Fire adds interactive fire behaviour to scene GameObjects. Objects can catch fire, ignite nearby objects, display distributed flame effects, finish burning with debris and smoke, or explode. Optional systems add terrain texture and grass burning, spatial fire audio and a first-person extinguisher.

This guide starts with a small scene, then explains the setup tools and advanced settings. Component reference tables and your Inspector screenshots remain available below the practical chapters.

Main components

Component	Purpose
FireEntity	Ignition, flame placement, propagation, destruction and explosion
FireManager	Central fire updates, distance scheduling and shared physics buffers
TerrainBurnManager	Burn textures and terrain details; optional delayed restoration
FireAudioManager / FireAudioEmitter	Rank audible fires and blend fire loops
FireExtinguisherFPS	Aim from the main camera and extinguish nearby entities
EnviroRainToggle	Optional commands for Setting Fire rain state and Enviro weather

Setting Fire uses the namespace `BLInformatique.SettingFire` and the BLInformatique EditorTools Inspector attributes. Keep the supplied runtime, Editor and EditorTools files together when importing the package. Enviro is optional and is not required for the basic fire and extinguisher workflow.

🛠️ How to Use

Recommended learning order

1. Install tags and layers, then create one FireManager.
2. Add FireEntity to a visible object with a Collider and assign a fire prefab.
3. Ignite and extinguish that object in Play Mode.
4. Add a second object and configure propagation.
5. Add terrain effects and audio only after the basic scene works.

Use the search field to find a component or setting. Expand component panels to inspect their fields and screenshots. The setup tools are under `Tools > BLInformatique > Setting Fire > Setup`.

The imported preset currently identifies Unity `6000.6.3f1`. That label comes from the supplied project preset; it is not a compatibility test report. For a release targeting Unity `6000.5.3f1`, validate the complete package in that Editor before changing the advertised version.

🏁 Conclusion

Build your first scene with one object, one fire prefab and an explicit ignition command. Add propagation, destruction, terrain and audio in small steps, checking each result in Play Mode.

Use the component reference below alongside the practical chapters. After importing this preset into ManualBuilder, check the resolved script and image references, regenerate HTML, and inspect the PDF produced by your Unity project before publishing.

Support: SettingFire@blinformatique.fr

⚓💪 TEAM VICTOR & BORIS — BLInformatique

📚 Optional Sections

📦 01 — Installation and scene preparation

Import the complete Setting Fire package into your project and wait for compilation to finish. Resolve missing dependencies before opening the setup windows. The custom Inspector requires the supplied BLInformatique EditorTools.

1. Open `Tools > BLInformatique > Setting Fire > Install Tags and Layers` and accept the installer dialog. It adds the `Torch` tag and the `Fire` layer when needed.

2. Create one FireManager using `Tools > BLInformatique > Setting Fire > Create Fire Manager`. This command loads the supplied `Resources/FireManager` prefab. If that resource is missing, create an empty GameObject and add FireManager manually.

3. Use a camera tagged `MainCamera`. This is needed by the FPS extinguisher and can be used automatically by the adaptive scheduler.

4. Start with one visible mesh and one Collider. Place FireEntity and the Collider on the same GameObject for the simplest setup.

5. Assign a fire prefab using materials suitable for your project's render pipeline. Keep additional managers on their own scene objects.

The runtime components do not require a Rigidbody just to ignite. A Rigidbody is needed for physics reactions such as falling supported objects or receiving an explosion force. Keep static scenery kinematic when using a Rigidbody.

For the classic extinguisher controls, the project must support the legacy Input Manager. For new Input System controls, enable the Input System in the project and configure the extinguisher action. The script exposes that action only when `ENABLE_INPUT_SYSTEM` is defined.

🔥 02 — Your first fire: ignite and extinguish

Prepare an object

1. Create a Cube in front of the main camera, above a floor.

2. Select it and run `Tools > BLInformatique > Setting Fire > Add Fire Entity Behaviour`.

3. Assign `firePrefab` and put the Cube on the Fire layer.

4. Keep `canBeIgnited` enabled. For this first test, disable `autoIgnite`, `autoFade`, `canExplode`, `canSpread` and `enablePropagation`.

5. Enter Play Mode. Select the Cube, open `Tools > BLInformatique > Setting Fire > Setup > FireEntity Mass Configurator`, use the selection and check the Cube.

6. Click `Ignite Checked`. Flames should appear while the Cube remains present.

7. Click `Extinguish Fire Only`. The fire state becomes Extinguished and the flames fade while the Cube is preserved.

For gameplay, call `FireEntity.Ignite()` from your own interaction or a UnityEvent. Call `ExtinguishFireOnly()` to stop a burning entity without destroying its object.

Detect a torch automatically

Enable `autoIgnite` before entering Play Mode. In the current implementation, this starts an igniter scan every 0.5 seconds; it does not call Ignite immediately on scene start.

Give the igniter's Collider GameObject the `Torch` tag and the `Fire` layer. Move it into the bounds of the target object's first child Renderer. The current scan uses those bounds; `igniteDetectionRadius` does not control this scan. Keep the flame prefab's own tagging separate from the intended igniter so you do not accidentally create a reusable torch source.

🛠️ 03 — Prepare many objects with the Editor tools

Setting Fire Setup

Open `Tools > BLInformatique > Setting Fire > Setup > Setting Fire Setup`. Select a parent in the Hierarchy and click `Use Current Selection`, then `Refresh` if needed. This tool discovers MeshRenderer objects under the selected parent.

Check the objects you want to configure. Choose the layer, Collider mode, optional Rigidbody settings and effect prefabs before clicking `Apply Setting Fire Setup`.

The available setup presets are Custom, GenericObject, TreeForest, SmallProp and ExplosiveObject. Review the resulting Inspector values after applying a preset. The tool can add FireEntity, assign layers recursively and add missing Colliders. Existing Colliders are kept; Rigidbody and FireEntity settings can be updated according to the selected options. Use the option for keeping existing FireEntity values when preserving a tuned object.

FireEntity Mass Configurator

Open `Tools > BLInformatique > Setting Fire > Setup > FireEntity Mass Configurator` to adjust existing FireEntity components. Choose a parent, refresh the list and check the intended targets.

Use `Apply Preset To Checked` for GenericObject, ForestTree, SmallProp, ExplosiveObject or SafeExtinguishable. Alternatively, edit values in the Fire, Placement, Terrain, Weather, Optimization, Explosion and Debug tabs, then click `Apply Serialized Changes`.

Presets replace several values at once. Review prefab references, propagation masks and destruction settings afterwards. Both tools record Editor changes with Undo. Save the scene and apply intended prefab overrides after checking the result.

The runtime test buttons `Ignite Checked` and `Extinguish Fire Only` are intended for Play Mode. The Setup window prepares MeshRenderer objects; use manual component setup for objects outside that discovery workflow.

🎄 04 — Flame placement and visual effects

FireEntity creates separate fire instances from `firePrefab`. FireFollower maintains their position relative to the burning object. AutoFadeInParticles provides the initial fade using `fireFadeInDuration`.

Setting	Practical use
flamePlacementMode: Auto	Detect tree-like names or tall bounds, otherwise use standard placement
StandardBounds	Distribute flames over Renderer bounds
TreeTrunk	Explicitly use the tree placement algorithm when enabled
fireDensity	Adjust standard placement density
maxFireInstances	Limit distributed instances; review extra ground fires in standard mode
alwaysPlaceGroundFire	Add a flame at the object's base
treeMaxFlameHeightNormalized	Keep trunk flames low instead of high in the canopy
treeTrunkRadiusMultiplier	Adjust estimated flame distance from the trunk

For trees, start with TreeTrunk mode and keep the main flame height below the canopy. Adjust the minimum and maximum trunk radius for thin or wide meshes. Auto mode uses

object names and geometry proportions, so explicitly select StandardBounds for tall props incorrectly treated as trees.

Ground projection uses `terrainBurnGroundMask`. Include the floor layer and avoid projecting onto the object's own Collider when positioning flames at its base.

`alternateLights` randomly enables or disables the lights module on the first ParticleSystem found in each fire instance. It is an optional visual cost reduction, not a complete lighting budget.

The fade helpers modify the alpha of `ParticleSystem.main.startColor` in single Color mode. Gradients and shaders that ignore alpha need their own compatible effect setup. AutoFadeOutParticles can destroy its GameObject when fading completes; attach it to the effect object rather than scenery you want to preserve.

🚒 05 — Propagation, wind and rain

Configure propagation

Add a second FireEntity with a Collider near the first. Put it on a layer included in the first object's `spreadLayerMask`. Assign fire prefabs to both.

The current implementation has two propagation paths:

Path	Control	Behaviour
Periodic spread	canSpread	Searches within spreadRadius using spreadLayerMask and a wind-adjusted chance
Delayed spread	enablePropagation	Schedules a direct neighbour scan after delayBeforeSpread; this scan does not use spreadLayerMask

Use `canSpread = true` and `enablePropagation = false` when testing only the periodic, layer-filtered path. Disable both to prevent outgoing propagation. A neighbour also needs `canBeIgnited` enabled.

Periodic spread uses a random interval between `minSpreadInterval` and `maxSpreadInterval`. Its chance is `baseSpreadChance + alignment × windSpeed × windInfluenceMultiplier`, clamped between 0 and 1. Alignment compares the neighbour direction with FireManager's windDirection. Wind favours objects downwind but does not guarantee ignition.

Weather setup and current limitations

FireManager exposes `manualRain`, `useExternalWeatherSystem` and `SetRainState(bool)`. EnviroRainToggle can update both the manager and the fallback `WeatherManager.isRaining` state.

FireEntity chooses its rain source according to the `FIREMANAGER_PRESENT` scripting symbol. With the symbol, it reads the manager's propagation permission; without it, it reads WeatherManager. Keep the relevant states consistent when integrating a weather controller.

In the supplied scripts, the periodic spread path does not check rain. The delayed path checks rain when scheduled and when executed. Rain does not continuously extinguish an already burning object: `extinguishInRain` only participates in scheduling StartFadeOut when Ignite runs, and that fade destroys the object.

For reliable gameplay extinction that preserves scenery, explicitly call `ExtinguishFireOnly()` on the desired burning entities. Validate the rain behaviour in your scene before using it as an automatic gameplay rule.

🧯 06 — FPS extinguisher

Create the supplied extinguisher using `Tools > BLInformatique > Setting Fire > Create Fire Extinguisher Object`, or add FireExtinguisherFPS to your extinguisher object. The creation command expects the `Resources/FireExtinguisherObject` prefab.

1. Position the extinguisher and its spray effect for your player.

2. Ensure the aiming camera is tagged MainCamera.

3. Assign `sprayEffect`, an optional `extinguisherSound`, and a `fireLayerMask` containing the target Colliders' layers.

4. Keep `extinguishFireEntities` enabled to stop the logical fire state.

5. Start with `extinguishDistance = 5` and `extinguishRadius = 2`, then adjust to your scene scale.

With classic input, `key` defaults to Mouse0. With `useToggle` disabled, hold the input to spray; with it enabled, press once to start and again to stop. For new input, enable `useNewInputSystem` and configure `extinguishAction` before Play Mode.

The spray is aimed by a ray from the centre of the screen. A Collider must first be hit within extinguishDistance and fireLayerMask. Nearby burning FireEntity components are then found around that hit point using extinguishRadius and the same layer mask.

`ExtinguishFireOnly()` stops pending propagation, destruction and explosion processes on a burning entity and preserves the object. It may be ignited again later if `canBeIgnited` remains enabled. An igniter still overlapping it can therefore reignite it.

`fadeLooseFireParticles` also searches nearby ParticleSystems on the Fire layer. This affects visuals independently of FireEntity state. Keep unrelated particles off that layer. `extinguishStrength` affects loose particle visuals; logical FireEntity extinction happens immediately when a burning target is found.

💥 07 — Destruction, explosions and events

Finish burning with destruction

With `autoFade` enabled, Ignite schedules StartFadeOut after `fireLifetime`. StartFadeOut changes the state to Destroying, fades active fire particles over `fadeDuration`, creates optional debris and final smoke, releases supported entities and destroys the original GameObject.

Use `debrisPrefab` for the remains and `finalSmokeEffectPrefab` for smoke. `smokeYOffset`, `smokeFadeDelay` and `smokeFadeDuration` control the final smoke placement and fade. Keep autoFade disabled for an object that must survive until explicitly extinguished.

`supportedEntities` lists FireEntity objects that should fall when this object finishes the destruction fade. Each supported object needs a Rigidbody on its FireEntity GameObject. ForceFall makes that Rigidbody non-kinematic and enables gravity. This support-release step belongs to the fade destruction path; explosion has a separate implementation.

Explosive objects

Enable `canExplode` and set `explosionDelay`. Optional references include a pre-explosion effect, explosion effect, aftermath fire, destroyed replacement and explosion sound. `explosionForce` affects nearby Rigidbodies within `explosionRadius`; it does not implement a character health or damage system.

`fadeFireOnExplosion` fades active flames before exploding and adds the fade duration to the sequence. When it is disabled, active flames are removed immediately at the explosion. The original object is destroyed after the explosion.

Avoid a short autoFade lifetime competing with the intended explosion delay. For an isolated explosion test, disable autoFade and ignite the object.

Events and public methods

API	Use
OnIgnited	Trigger sound, UI, objectives or another gameplay action when ignition succeeds
OnExploded	Notify another system when the explosion occurs
Ignite()	Ignite an eligible object
ExtinguishFireOnly()	Preserve the object while stopping a burning fire
StartFadeOut()	Start the destructive fade sequence
IsBurning() / GetFireState()	Read the runtime state
ForceFall()	Enable gravity on the local Rigidbody

The supplied component has no OnExtinguished or general OnDestroyed UnityEvent. Implement additional notifications in your gameplay integration when needed.

🌱 08 — Terrain burning and optional restoration

Add one TerrainBurnManager to a dedicated scene object. Assign `targetTerrains`, or leave the list empty with `autoFindTerrains` enabled to discover active terrains during initialization.

Burn textures and grass

1. Add a suitable burnt or ash TerrainLayer to each target Terrain.

2. Set `burntTerrainLayerIndex` to that layer's index. A value of -1 selects the last layer.

3. Enable `enableTextureBurn` for texture blending and `enableDetailBurn` for grass/detail removal.

4. On each FireEntity, enable `enableTerrainBurn`, the desired ignition/ongoing burn options, and the texture/detail options.

5. Adjust `terrainBurnRadius`, `terrainBurnStrength` and `terrainBurnInterval`.

`projectTerrainBurnToGround` uses a downward raycast and `terrainBurnGroundMask`. Check that it reaches the intended ground. The manager operates on Unity Terrain texture maps and detail layers; it does not remove Terrain trees or arbitrary grass GameObjects.

`textureBurnSpeed` controls each paint operation. `minimumBurntDetailMultiplier = 0` permits complete detail removal; progressive fading can still require repeated requests. Brush noise softens the shape of the burnt area.

Keep `cloneTerrainDataInPlayMode` enabled to work on runtime copies and preserve the original data. The manager restores original TerrainData references when disabled or destroyed. Disabling cloning lets burn operations modify the original data and removes the untouched baseline needed for restoration.

Optional delayed restoration

Enable `enableRegrowth` on the manager and `allowTerrainRegrowth` on the entity. Choose `regrowthDelay`, `regrowthSpeed`, `regrowTexture` and `regrowDetails`.

The current implementation queues one delayed restoration operation for each eligible burn request. Each operation blends the affected area towards original terrain data once. It does not keep re-enqueuing itself until full recovery, so a single request may produce only partial restoration. Validate the result with your terrain and chosen burn frequency.

`leaveTerrainAsh` is passed through the burn request, but the supplied manager does not process it into a separate ash effect. Use the burnt TerrainLayer for visible ash feedback.

Processing budgets are `maxBurnRequestsPerFrame`, `maxRegrowthRequestsPerFrame` and `maxQueuedBurnRequests`. A full burn queue discards new requests, so monitor busy scenes when lowering budgets.

🎧 09 — Spatial fire audio

Open `Tools > BLInformatique > Setting Fire > Setup > Fire Audio Setup` before Play Mode.

1. Assign Small, Medium and Large Fire Loop clips. Wind Fire Loop and Global Ambience are optional.

2. Choose Max Active Sources, Max Audible Distance and Fade Start Distance.

3. Click `Create / Update Manager`.

4. Click `Add Emitters To Scene Fires`. Existing emitters are updated; missing emitters are added to active FireEntity objects.

5. Assign any random crack clips directly on each FireAudioEmitter Inspector. The current setup window does not expose a crack list field.

Keep one FireAudioManager and a valid listener. Use listenerOverride if automatic listener discovery does not select the intended player. Emitters normally play only while their associated FireEntity is burning.

The manager ranks candidates by intensity and distance. `maxActiveFireSources` limits selected emitters, not individual Unity AudioSource components: an emitter can blend several loop sources and a crack source. Global ambience is additional, and sources fading out may briefly continue playing.

Set `fadeStartDistance` below `maxAudibleDistance`. `baseVolume`, pitch variation and smoothing control the blend. For a stable starting point on ordinary FireEntity objects, disable `estimateIntensityFromParticles` and adjust `manualIntensity`: the emitter caches

child ParticleSystems in Awake, while FireEntity creates its flames as separate objects later. Those spawned flames are not automatically included in that cache.

`windFireLoop` is blended according to computed intensity, not directly according to FireManager.windSpeed. Use manager counters and emitter read-only fields to inspect selection and distance during Play Mode.

⚙️ 10 — Scheduling and performance

Keep `useCentralScheduler` enabled on the FireManager to centralize burning entity updates. With `useAdaptiveScheduler` enabled and a resolved reference camera, distance determines how frequently each entity is processed.

Default range	Default tick interval
Up to nearDistance: 50	nearTickInterval: 0.10 seconds
Up to mediumDistance: 150	mediumTickInterval: 0.50 seconds
Farther away	farTickInterval: 2.00 seconds

If `suspendBeyondFarDistance` is enabled, entities farther than farDistance (300 by default) stop receiving these internal ticks. That suspends periodic propagation and ongoing terrain requests; Invoke timers and explosion/fade coroutines are separate and continue.

Without a reference camera, the manager falls back to fixed scheduling using `propagationTickInterval`. `maxEntitiesPerTick = 0` processes all eligible entities; a positive value limits work per pass. `staggerInitialAdaptiveTicks` spreads first updates over time.

Shared physics buffers reduce repeated overlap-query allocations. Configure `sharedBufferSize`, prewarming and storage limits along with each entity's `maxOverlapResults`. Propagation falls back to an allocating overlap query when its buffer fills, so avoid describing this system as entirely allocation-free.

For dense scenes, first lower unnecessary flame counts and light cost. Then tune tick intervals, terrain budgets and audio emitter limits while observing actual gameplay. The loose-particle extinguisher option searches scene ParticleSystems repeatedly while spraying; disable that option when only logical FireEntity extinction is needed.

Use debug logging for diagnosis, then disable verbose options for normal play. Confirm performance in a player build with your own effects and scene density.

☔ 11 — Optional Enviro bridge

Add the `EnviroRainToggle` component when you need explicit rain commands. The source file is named EnviroRainToggle_BRIDGE.cs, while the class is EnviroRainToggle.

Public commands are `EnableRain()`, `DisableRain()`, `ToggleRain()` and `SetRain(bool)`. They can be connected to UI buttons or UnityEvents. Keep `updateFireManagerRainState` and `updateFallbackWeatherManager` enabled when both rain paths must stay synchronized. Enable `useExternalWeatherSystem` on FireManager, keep manualRain disabled for external switching, and enable blockPropagationWhenRaining to make its permission reflect rain.

If Enviro is installed, the bridge looks for its manager and tries to call `Weather.ChangeWeather(string)` by reflection. Set rainWeatherName and clearWeatherName to names recognized by your Enviro setup. The defaults are Heavy Rain and Clear. This depends on the installed Enviro API and must be tested in that project.

The bridge sends commands when called. It does not continuously monitor weather changes made elsewhere in Enviro, and FireManager's auto-detection code does not implement continuous rain synchronization. `rainAmount` is available as an integration field but is not applied by these commands.

Enable missing-Enviro warnings or debug logs while checking the bridge. Basic Setting Fire use does not require Enviro. See the propagation chapter for the current rain limitations before relying on weather to stop all spread or preserve burning scenery.

🔎 12 — Troubleshooting and validation

Symptom	Check
No visible flames	firePrefab assignment, enabled Renderer, material/render-pipeline compatibility and successful ignition
Torch does not ignite an object	autoIgnite enabled before Start; Torch tag and Fire layer on the igniter Collider; overlap with Renderer bounds
Neighbour never catches fire	Collider, canBeIgnited, spreadRadius, canSpread and spreadLayerMask; scheduled entity updates
Object catches fire outside the chosen spread mask	The delayed enablePropagation path does not use that mask

Symptom	Check
Extinguisher sprays but does not extinguish	MainCamera tag, target Collider layer, fireLayerMask, aim hit, distance and extinguishFireEntities
Object disappears after fire fades	autoFade and StartFadeOut use the destructive sequence
Extinguished object reignites	Remaining torch overlap or another burning neighbour; disable canBeIgnited if permanent prevention is required
No terrain mark	TerrainBurnManager, active target terrain, burnt layer index and both entity/manager texture options
No grass removal	Unity Terrain detail layers, both detail options and sufficient repeated burn requests
Terrain only partly recovers	Current restoration is one blend per delayed request; verify cloning and regrowth permissions
No fire audio	FireAudioManager, listener, assigned loops, FireEntity state and distance budget
Audio intensity ignores spawned flames	Use manualIntensity or an integration that provides the intended child particle systems
Menu cannot create a manager/extinguisher	Supplied Resources prefab missing; restore it or add the component manually

Small validation scene

Test a single fire, safe extinction, reignition, periodic propagation, destructive fade, explosion, terrain burning and audio separately. Then combine them. Verify the selected input backend and optional weather integration in a player build.

Do not infer automatic rain extinction or full terrain recovery from field names alone. Compare the configured result with the behaviour explained in this guide.

CONTENTS

🔎 Filter components…

📚 Table of Contents

📖 Overview

🛠️ How To Use

🏁 Conclusion

📦 01 — Installation and scene preparation

🔥 02 — Your first fire: ignite and extinguish

🛠️ 03 — Prepare many objects with the Editor tools

🌲 04 — Flame placement and visual effects

🌬️ 05 — Propagation, wind and rain

🧯 06 — FPS extinguisher

💥 07 — Destruction, explosions and events

🌱 08 — Terrain burning and optional restoration

🎧 09 — Spatial fire audio

⚙️ 10 — Scheduling and performance

☔ 11 — Optional Enviro bridge

🔎 12 — Troubleshooting and validation

📦 Components (12)

BLINFORMATIQUE.SETTINGFIRE

AutoFadeInParticles

AutoFadeOutParticles

EnviroRainToggle

FireAudioEmitter

FireAudioManager

FireAudioSetupWindow

FireEntity

FireEntityMassConfiguratorWindow

FireExtinguisherFPS

FireManager

SettingFireWindow

TerrainBurnManager

Components

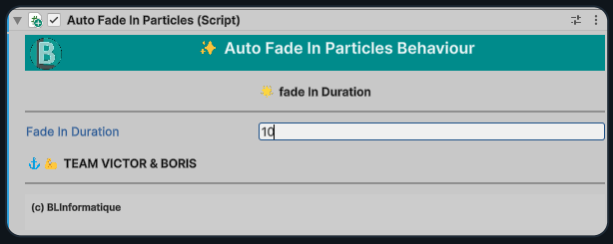
Tooltips are in **English**. Missing tooltips can be completed automatically from field names. Tables list documented fields. Optional notes (Markdown) and images are shown below each component when present.

AutoFadeInParticles

MonoBehaviour

BLInformatique.SettingFire

Fades single-Color particle start alpha in child systems. A shader and particle setup that respect alpha are required.



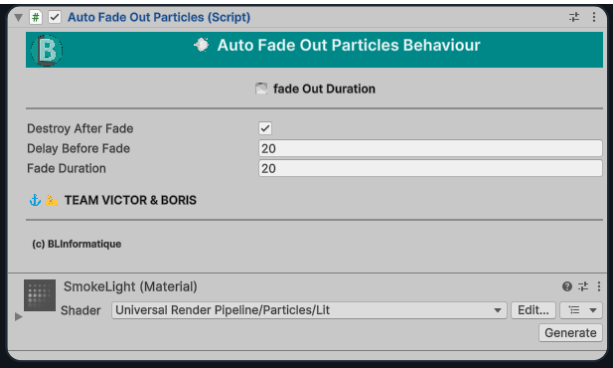
Field	Type	Tooltip	Hints
⬇			
fadeInDuration	`Single`	Duration of the particle fade-in.	—
⬇			
footerDummy	`String`		—

AutoFadeOutParticles

MonoBehaviour

BLInformatique.SettingFire

Fades particle start alpha after a delay. destroyAfterFade removes this effect GameObject when enabled. Avoid placing it on scenery that must survive.

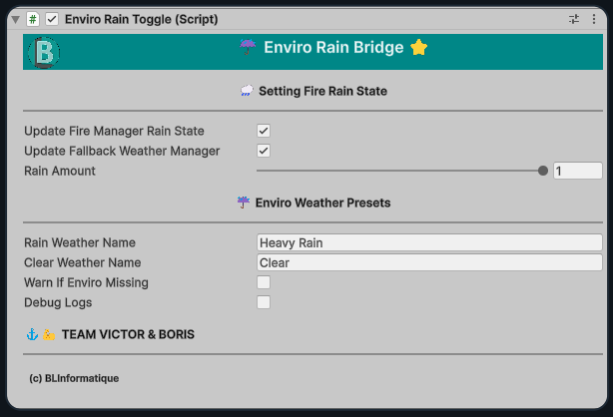


Field	Type	Tooltip	Hints
⬇			
destroyAfterFade	`Boolean`	Destroy this GameObject when the fade-out is complete.	—
delayBeforeFade	`Single`	Delay before the particle fade-out starts.	—
fadeDuration	`Single`	Duration of the particle fade-out.	—
⬇			
footerDummy	`String`		—

EnviroRainToggle MonoBehaviour

BLInformatique.SettingFire

Explicit rain commands for the manager, fallback state and optional Enviro weather. This bridge does not continuously monitor Enviro. See the weather chapter for current limitations.

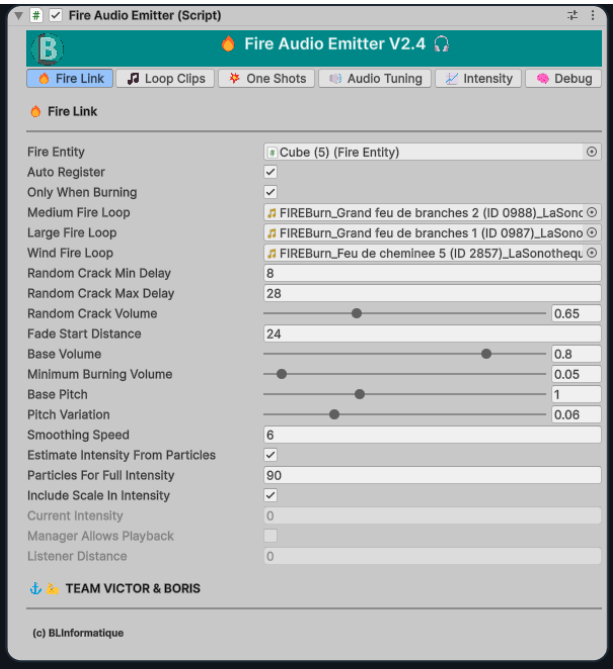


Field	Type	Tooltip	Hints
updateFireManagerRainState	`Boolean`	Also update the Setting Fire manager rain state.	—
updateFallbackWeatherManager	`Boolean`	Also update the fallback static WeatherManager rain state.	—
rainAmount	`Single`	Optional rain intensity value used by custom integrations. • Range [0..1]	Range
rainWeatherName	`String`	Weather preset to activate when enabling rain in Enviro.	—
clearWeatherName	`String`	Weather preset to activate when disabling rain in Enviro.	—
warnIfEnviroMissing	`Boolean`	If enabled, a warning is shown when Enviro cannot be found at runtime.	—
debugLogs	`Boolean`	If enabled, the bridge logs weather changes sent to Enviro.	—
footerDummy	`String`		—

FireAudioEmitter MonoBehaviour

BLInformatique.SettingFire

Assign clips before Play Mode. For flames spawned separately by FireEntity, start with manualIntensity rather than assuming the Awake particle cache includes them.



Field	Type	Tooltip	Hints
header	`String`	Controls the manual header display.	—
⬇			
fireEntity	`FireEntity`	FireEntity used to drive this audio emitter. If empty, the component searches in parents.	Object Reference
autoRegister	`Boolean`	Automatically register this emitter in the FireAudioManager.	—
onlyWhenBurning	`Boolean`	Only play audio while the linked FireEntity is burning.	—
⬇			
smallFireLoop	`AudioClip`	Light crackling loop used for small fires.	Object Reference
mediumFireLoop	`AudioClip`	Medium fire loop used when the fire becomes stronger.	Object Reference
largeFireLoop	`AudioClip`	Large roaring fire loop used for intense fires.	Object Reference
windFireLoop	`AudioClip`	Optional wind or air pressure loop blended with strong fires.	Object Reference
⬇			

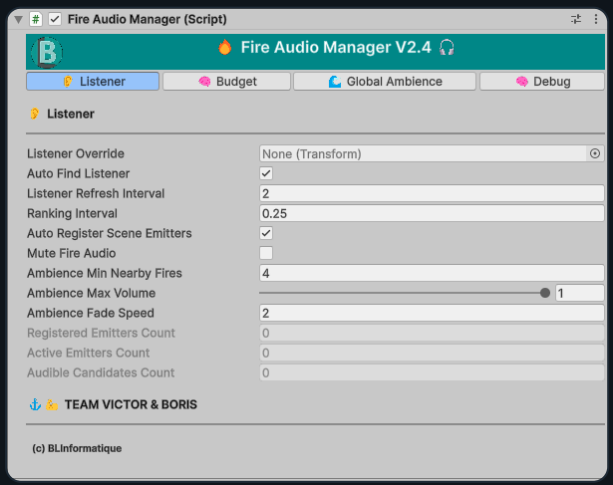
Field	Type	Tooltip	Hints
randomCracks	`AudioClip[]`	Random crack clips played from time to time while the fire is burning.	List / Array
randomCrackMinDelay	`Single`	Minimum delay between random crack sounds.	—
randomCrackMaxDelay	`Single`	Maximum delay between random crack sounds.	—
randomCrackVolume	`Single`	Volume multiplier used for random crack sounds. • Range [0..2]	Range
⬇			
maxAudibleDistance	`Single`	Maximum distance where this fire can be heard.	—
fadeStartDistance	`Single`	Distance where the fire starts fading out before being muted.	—
baseVolume	`Single`	Base volume applied before distance and intensity attenuation. • Range [0..1]	Range
minimumBurningVolume	`Single`	Minimum volume while the emitter is active and burning. • Range [0..1]	Range
basePitch	`Single`	Base pitch of the fire loop. • Range [0,5..2]	Range
pitchVariation	`Single`	Random pitch variation applied to avoid obvious looping. • Range [0..0,25]	Range
smoothingSpeed	`Single`	How quickly volume and pitch react to fire changes.	—
⬇			
manualIntensity	`Single`	Manual intensity used when particle based intensity is disabled. • Range [0..1]	Range
estimateIntensityFromParticles	`Boolean`	Estimate intensity from active ParticleSystems below this object.	—

Field	Type	Tooltip	Hints
particlesForFullIntensity	`Int32`	Maximum particles considered as a full intensity fire.	—
includeScaleInIntensity	`Boolean`	Use the Transform scale as a small extra intensity factor.	—
⬇			
debugAudio	`Boolean`	Display runtime audio values in the Inspector.	—
currentIntensity	`Single`	Current computed intensity.	—
managerAllowsPlayback	`Boolean`	True when the FireAudioManager currently allows this emitter to play.	—
listenerDistance	`Single`	Current distance to the active listener.	—
⬇			
footerDummy	`String`		—

FireAudioManager MonoBehaviour

BLInformatique.SettingFire

The playback budget selects emitters, not individual AudioSources. Each emitter can blend several loops. Global ambience is additional.



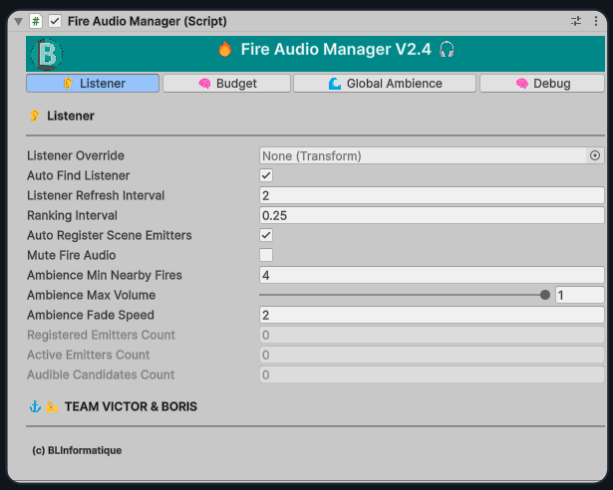
Field	Type	Tooltip	Hints
header	`String`	Controls the manual header display.	—
↓			
listenerOverride	`Transform`	Listener transform used to select the closest audible fires. If empty, the manager uses AudioListener or Camera.main.	Object Reference
autoFindListener	`Boolean`	Automatically search for the active listener when no override is assigned.	—
listenerRefreshInterval	`Single`	How often the listener search is refreshed.	—
↓			
maxActiveFireSources	`Int32`	Maximum FireAudioEmitter components allowed to play at the same time. • Range [1..32]	Range
rankingInterval	`Single`	How often emitters are ranked by distance and intensity.	—
autoRegisterSceneEmitters	`Boolean`	Automatically register all FireAudioEmitter components found in the scene on Start.	—
muteFireAudio	`Boolean`	Mute all fire audio handled by this manager.	—
↓			
globalLargeFireAmbience	`AudioClip`	Optional non-spatial ambience loop used when many nearby fires are burning.	Object Reference
ambienceMinNearbyFires	`Int32`	Minimum number of audible candidates required before the global ambience starts fading in.	—
ambienceMaxVolume	`Single`	Maximum volume of the global ambience layer. • Range [0..1]	Range
ambienceFadeSpeed	`Single`	How quickly the global ambience fades in and out.	—
↓			

Field	Type	Tooltip	Hints
debugAudioManager	`Boolean`	Display runtime manager values in the Inspector.	—
registeredEmittersCount	`Int32`	Number of registered FireAudioEmitter components.	—
activeEmittersCount	`Int32`	Number of emitters currently allowed to play.	—
audibleCandidatesCount	`Int32`	Number of audible fire candidates around the listener.	—
⬇			
footerDummy	`String`		—

📘 FireAudioSetupWindow ScriptableObject

BLInformatique.SettingFire

Editor tool: assign loops, create/update the manager, then add/update emitters on active scene FireEntities. Configure random crack clips on the emitter Inspector.

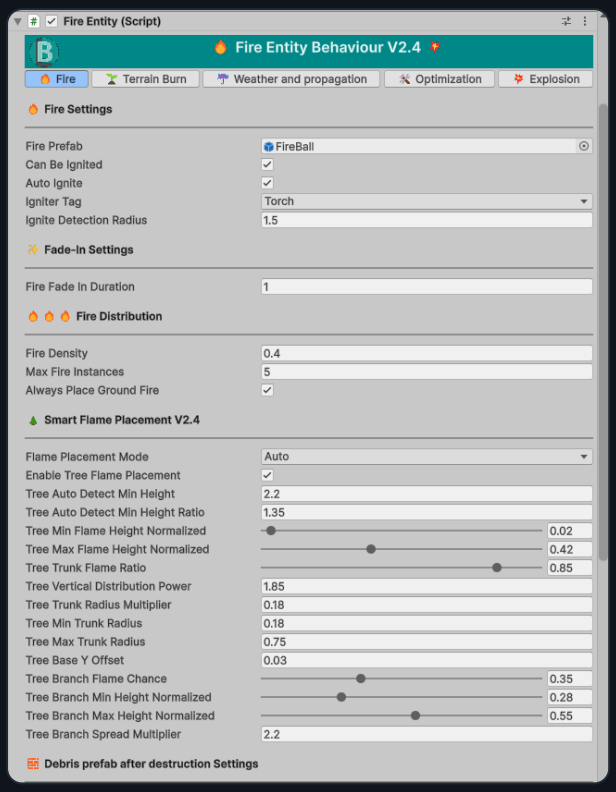


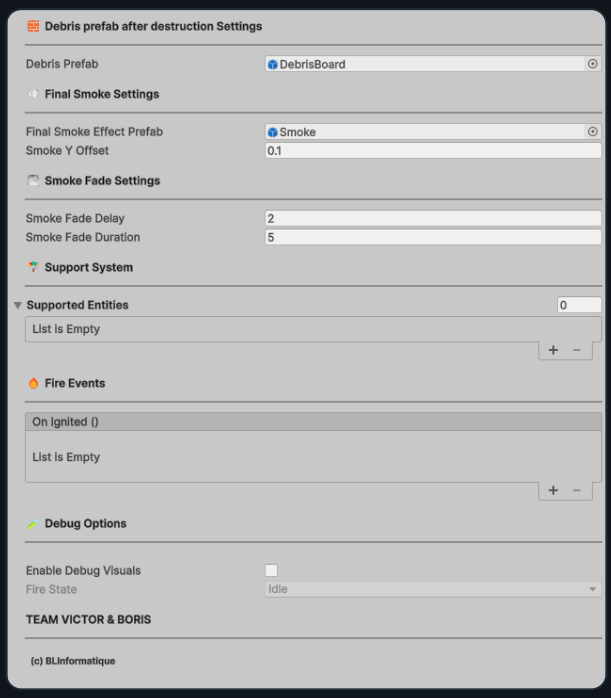
No serialized fields detected.

📦 FireEntity MonoBehaviour

BLInformatique.SettingFire

Auto Ignite starts torch detection rather than immediate ignition. ExtinguishFireOnly preserves the object; StartFadeOut destroys it. Both outgoing propagation options must be disabled for an isolated fire.





Field	Type	Tooltip	Hints
fireState	`FireState`	Current runtime fire state.	Enum, Private Serialized
header	`String`	Controls the manual header display.	—
⬇			
firePrefab	`GameObject`	Prefab of the fire effect.	Object Reference
canBeIgnited	`Boolean`	Can this object be ignited?	—
autoIgnite	`Boolean`	Automatically ignite on start.	—
igniterTag	`String`	Tag of the object that can ignite this.	—
igniteDetectionRadius	`Single`	Radius for ignition detection.	—

Field	Type	Tooltip	Hints
⬇			
fireFadeInDuration	`Single`	Duration of the fade-in effect when fire appears.	—
⬇			
fireDensity	`Single`	Density of fire instances over object bounds.	—
maxFireInstances	`Int32`	Maximum number of fire instances.	—
alwaysPlaceGroundFire	`Boolean`	Always spawn a ground fire under object.	—
⬇			
flamePlacementMode	`FlamePlacementMode`	Select how fire instances are placed on this object.	Enum
enableTreeFlamePlacement	`Boolean`	Enable the dedicated tree placement engine when Auto detects vegetation or when TreeTrunk mode is selected.	—
treeAutoDetectMinHeight	`Single`	Minimum height used by Auto mode to detect a tree-like object.	—
treeAutoDetectMinHeightRatio	`Single`	Minimum vertical ratio used by Auto mode to detect a tree-like object.	—

Field	Type	Tooltip	Hints
treeMinFlameHeightNormalized	`Single`	Normalized minimum height for tree flames. 0 means object base, 1 means top. • Range [0..1]	Range
treeMaxFlameHeightNormalized	`Single`	Normalized maximum height for main tree flames. Lower values keep flames away from the canopy. • Range [0,05..1]	Range
treeTrunkFlameRatio	`Single`	Ratio of tree flames placed around the trunk. • Range [0..1]	Range
treeVerticalDistributionPower	`Single`	Controls how much tree flames stay near the base. Higher values push flames lower.	—
treeTrunkRadiusMultiplier	`Single`	Multiplier used to estimate trunk radius from object bounds.	—
treeMinTrunkRadius	`Single`	Minimum trunk radius used by the tree placement engine.	—
treeMaxTrunkRadius	`Single`	Maximum trunk radius used by the tree placement engine.	—
treeBaseYOffset	`Single`	Small vertical offset applied to the ground fire at the tree base.	—
treeBranchFlameChance	`Single`	Chance to place optional flames on lower branches. • Range [0..1]	Range

Field	Type	Tooltip	Hints
treeBranchMinHeightNormalized	`Single`	Minimum normalized height for optional branch flames. • Range [0..1]	Range
treeBranchMaxHeightNormalized	`Single`	Maximum normalized height for optional branch flames. • Range [0..1]	Range
treeBranchSpreadMultiplier	`Single`	Spread multiplier used for optional branch flames.	—
⬇			
debrisPrefab	`GameObject`	Prefab to instantiate after destruction.	Object Reference
⬇			
finalSmokeEffectPrefab	`GameObject`	Smoke prefab to play after destruction.	Object Reference
smokeYOffset	`Single`	Vertical offset for the smoke position.	—
⬇			
smokeFadeDelay	`Single`	Delay before smoke starts fading.	—
smokeFadeDuration	`Single`	Duration of smoke fade.	—
⬇			
supportedEntities	`List<FireEntity>`	Other FireEntities supported by this one (for gravity).	List / Array
⬇			

Field	Type	Tooltip	Hints
OnIgnited	`UnityEvent`	Event invoked when the object catches fire (Ignite is called).	UnityEvent
⬇			
enableTerrainBurn	`Boolean`	Send terrain burn requests when this entity burns.	—
burnTerrainOnIgnite	`Boolean`	Burn terrain once when this entity ignites.	—
burnTerrainWhileBurning	`Boolean`	Continue sending terrain burn requests while this entity is burning.	—
burnTerrainTexture	`Boolean`	Paint burnt terrain texture when this entity burns.	—
removeTerrainDetailsOnBurn	`Boolean`	Remove terrain grass and details where this entity burns.	—
leaveTerrainAsh	`Boolean`	Leave ash feedback on the terrain burn manager when this entity burns.	—
allowTerrainRegrowth	`Boolean`	Allow the terrain burn manager to regrow this burned zone if regrowth is enabled.	—
terrainBurnRadius	`Single`	Radius used to burn terrain texture and details.	—
terrainBurnStrength	`Single`	Strength used to burn terrain texture and details. • Range [0..1]	Range

Field	Type	Tooltip	Hints
terrainBurnInterval	`Single`	Interval between terrain burn requests while burning.	—
projectTerrainBurnToGround	`Boolean`	Use a downward raycast to place the terrain burn exactly on the ground.	—
terrainBurnGroundMask	`LayerMask`	Layer mask used to find ground when projecting terrain burn positions.	—
terrainBurnProjectionDistance	`Single`	Maximum distance used when projecting burn positions to the ground.	—
⬇			
extinguishInRain	`Boolean`	Extinguish fire when raining.	—
⬇			
baseSpreadChance	`Single`	Base chance to spread fire regardless of wind. • Range [0..1]	Range
windInfluenceMultiplier	`Single`	Multiplier applied to wind influence. • Range [0..1]	Range
⬇			
canSpread	`Boolean`	Enable fire propagation to nearby objects.	—
enablePropagation	`Boolean`	Enable fire propagation to nearby objects.	—
spreadRadius	`Single`	Radius in which the fire can spread.	—

Field	Type	Tooltip	Hints
delayBeforeSpread	`Single`	Delay before fire spreads.	—
minSpreadInterval	`Single`	Minimum interval before spreading.	—
maxSpreadInterval	`Single`	Maximum interval before spreading.	—
spreadLayerMask	`LayerMask`	Layer mask for objects that can catch fire.	—
⬇			
alternateLights	`Boolean`	Alternate light emission randomly for performance.	—
⬇			
autoFade	`Boolean`	Enable fire auto fade after time.	—
fireLifetime	`Single`	Duration before fire fades.	—
fadeDuration	`Single`	Duration of the fade out effect.	—
⬇			
maxOverlapResults	`Int32`	Maximum physics hits requested for NonAlloc propagation queries. FireManager can provide this buffer from its shared pool.	—
⬇			
canExplode	`Boolean`	Can this object explode?	—

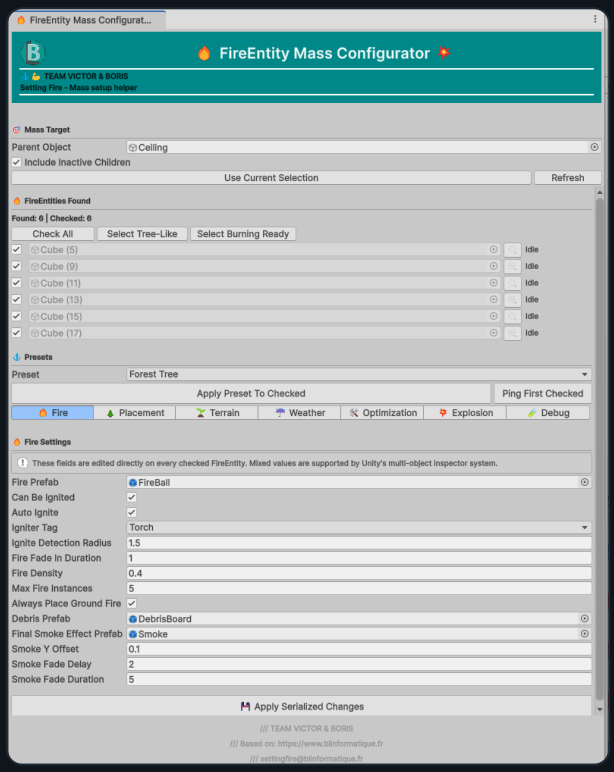
Field	Type	Tooltip	Hints
explosionDelay	`Single`	Delay before explosion triggers.	—
explosionVFXDuration	`Single`	Time before explosion VFX is destroyed.	—
explosionEffectPrefab	`GameObject`	VFX prefab for explosion.	Object Reference
explosionFirePrefab	`GameObject`	Fire prefab for explosion aftermath.	Object Reference
destroyedVersionPrefab	`GameObject`	Destroyed version of the object.	Object Reference
preExplosionEffectPrefab	`GameObject`	Pre-explosion effect to instantiate.	Object Reference
explosionForce	`Single`	Force of the explosion.	—
explosionRadius	`Single`	Radius of the explosion.	—
fadeFireOnExplosion	`Boolean`	Fade fires when exploding.	—
⬇			
explosionSound	`AudioClip`	Audio clip to play when this object explodes.	Object Reference
⬇			
explosionVolume	`Single`	Range [0..1]	Range
⬇			
OnExploded	`UnityEvent`	Event invoked when the object explodes.	UnityEvent
⬇			

Field	Type	Tooltip	Hints
enableDebugVisuals	`Boolean`	Draw debug visuals for this fire entity.	—

📘 FireEntityMassConfiguratorWindow ScriptableObject

BLInformatique.SettingFire

Editor tool for existing FireEntities. Select a parent, check targets, apply a preset or serialized edits. Runtime ignite/extinguish buttons require Play Mode.



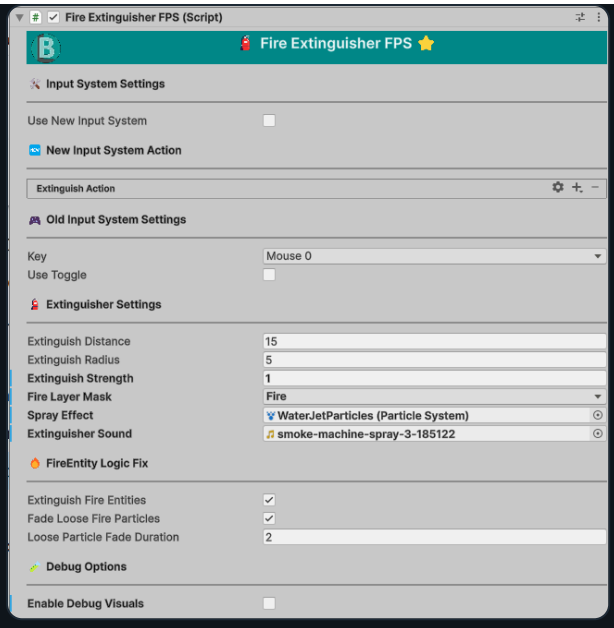
No serialized fields detected.

📦 FireExtinguisherFPS MonoBehaviour

BLInformatique.SettingFire

Requires: AudioSource

A centre-screen ray must hit a Collider in fireLayerMask. Keep extinguishFireEntities enabled to stop the logical fire. Loose particle fading is a separate visual option.



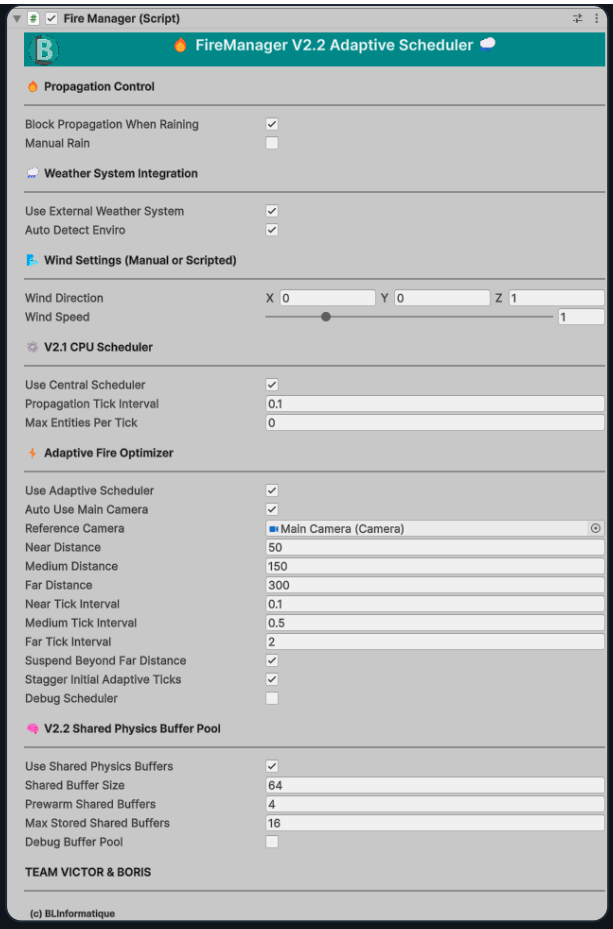
Field	Type	Tooltip	Hints
⬇			
useNewInputSystem	`Boolean`	Use the New Input System instead of the classic Input Manager.	—
⬇			
extinguishAction	`InputAction`	Input action used to activate the extinguisher spray.	—
⬇			
key	`KeyCode`	Key or mouse button used to activate the extinguisher with the classic Input Manager.	Enum
useToggle	`Boolean`	Toggle the extinguisher on/off instead of holding the input.	—
⬇			
extinguishDistance	`Single`	Maximum distance from the camera where the extinguisher can hit fire.	—
extinguishRadius	`Single`	Radius around the hit point used to extinguish nearby FireEntity objects and fire particles.	—

Field	Type	Tooltip	Hints
extinguishStrength	`Single`	Strength used to visually reduce nearby fire particles over time.	—
fireLayerMask	`LayerMask`	Layer mask used to detect fire objects and fire entities.	—
sprayEffect	`ParticleSystem`	Particle system played from the extinguisher nozzle while spraying.	Object Reference
extinguisherSound	`AudioClip`	Looping sound played while the extinguisher is spraying.	Object Reference
⬇			
extinguishFireEntities	`Boolean`	Call FireEntity.ExtinguishFireOnly() so the logical fire state stops, not only the visual particles.	—
fadeLooseFireParticles	`Boolean`	Also fade loose ParticleSystem objects found near the impact point.	—
looseParticleFadeDuration	`Single`	Fade duration applied to loose fire particles when they are hit by the extinguisher.	—
⬇			
enableDebugVisuals	`Boolean`	Draw debug rays, debug radius, and log extinguished fire entities.	—
⬇			
footerDummy	`String`		Hidden

FireManager MonoBehaviour

BLInformatique.SettingFire

Distance scheduling affects internal fire ticks, not every coroutine or Invoke timer. Read the weather chapter before relying on rain to block all propagation.



Field	Type	Tooltip	Hints
header	`String`	Controls the manual header display.	—
⬇			
blockPropagationWhenRaining	`Boolean`	Block fire propagation when raining.	—
manualRain	`Boolean`	Simulate rain manually (useful if no weather system is present).	—
⬇			
useExternalWeatherSystem	`Boolean`	Enable integration with external weather systems like Enviro.	—
autoDetectEnviro	`Boolean`	Automatically detect Enviro if available.	—
⬇			
windDirection	`Vector3`	Wind direction used to influence fire propagation.	—
windSpeed	`Single`	Wind speed used in propagation calculations. • Range [0..5]	Range

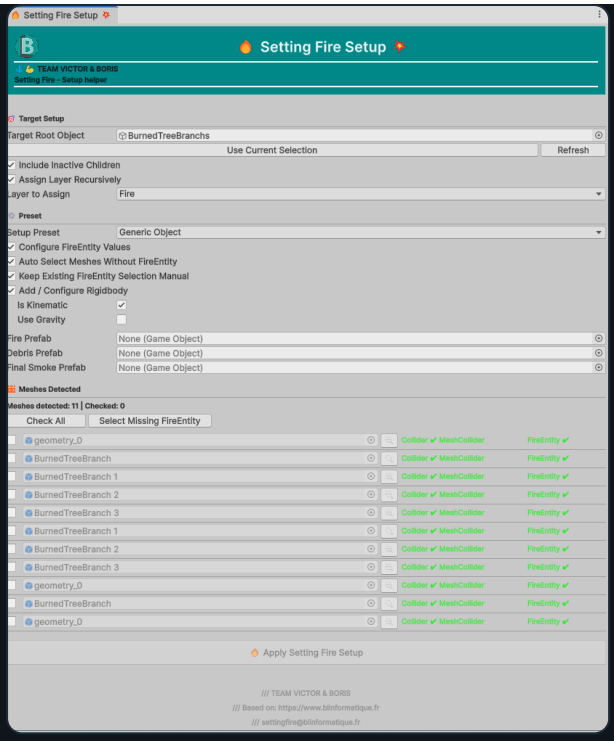
Field	Type	Tooltip	Hints
⬇			
useCentralScheduler	`Boolean`	Use the FireManager centralized scheduler instead of one Update call per burning FireEntity.	—
propagationTickInterval	`Single`	Time in seconds between two propagation scheduler passes when adaptive scheduling is disabled. Lower values are more reactive, higher values are cheaper.	—
maxEntitiesPerTick	`Int32`	Maximum number of burning entities processed per scheduler pass. Use 0 to process all burning entities each pass.	—
⬇			
useAdaptiveScheduler	`Boolean`	Enable distance based adaptive scheduling for burning entities.	—
autoUseMainCamera	`Boolean`	Automatically use Camera.main when no reference camera is assigned.	—
referenceCamera	`Camera`	Camera used to calculate distance based fire update rates.	Object Reference
nearDistance	`Single`	Distance under which fire uses the near tick interval.	—
mediumDistance	`Single`	Distance under which fire uses the medium tick interval.	—
farDistance	`Single`	Distance under which fire uses the far tick interval.	—
nearTickInterval	`Single`	Update interval for nearby burning entities.	—
mediumTickInterval	`Single`	Update interval for medium distance burning entities.	—
farTickInterval	`Single`	Update interval for far burning entities.	—
suspendBeyondFarDistance	`Boolean`	Suspend propagation for burning entities farther than the far distance.	—

Field	Type	Tooltip	Hints
staggerInitialAdaptiveTicks	`Boolean`	Spread first adaptive ticks across time to avoid a CPU spike when many objects ignite together.	—
debugScheduler	`Boolean`	Show scheduler information in the Console for debugging.	—
⬇			
useSharedPhysicsBuffers	`Boolean`	Use a shared Collider buffer pool for fire propagation physics queries.	—
sharedBufferSize	`Int32`	Default size of shared Collider buffers used by fire propagation queries.	—
prewarmSharedBuffers	`Int32`	Number of shared Collider buffers created on Awake to avoid runtime allocations during the first fire ticks.	—
maxStoredSharedBuffers	`Int32`	Maximum number of shared Collider buffers kept in memory after use.	—
debugBufferPool	`Boolean`	Show shared buffer pool information in the Console for debugging.	—

📘 SettingFireWindow ScriptableObject

BLInformatique.SettingFire

Editor setup tool for MeshRenderer objects. Adds missing components and can overwrite configured values. Review Collider, layer and Rigidbody choices before applying.

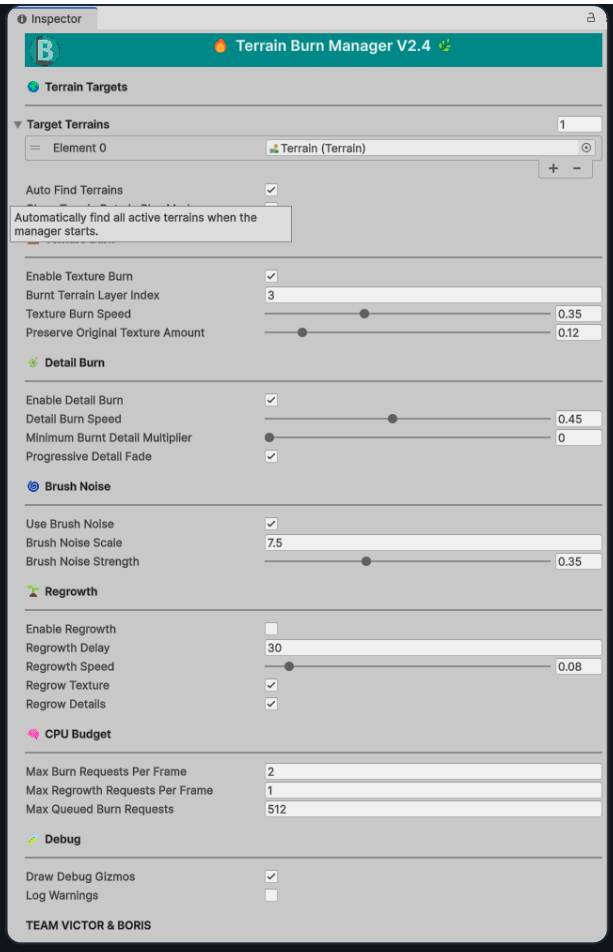


No serialized fields detected.

TerrainBurnManager MonoBehaviour

BLInformatique.SettingFire

Use runtime TerrainData cloning. Regrowth is one delayed blend per eligible request in the current implementation. leaveAsh is not processed as a separate ash effect.



Field	Type	Tooltip	Hints
header	`String`	Controls the manual header display.	—
⬇			
targetTerrains	`Terrain[]`	Terrains affected by the burn system. If empty, terrains are detected automatically at runtime.	List / Array
autoFindTerrains	`Boolean`	Automatically find all active terrains when the manager starts.	—
cloneTerrainDataInPlayMode	`Boolean`	Clone terrain data during Play Mode to prevent permanent terrain modifications after exiting Play Mode.	—
⬇			
enableTextureBurn	`Boolean`	Enable painting burnt terrain textures.	—
burntTerrainLayerIndex	`Int32`	Terrain layer index used as the burnt or ash layer. Use -1 to automatically use the last terrain layer.	—

Field	Type	Tooltip	Hints
textureBurnSpeed	`Single`	How fast terrain textures are blended toward the burnt layer. • Range [0,01..1]	Range
preserveOriginalTextureAmount	`Single`	Keep some original texture under the burnt layer to avoid a flat black look. • Range [0..1]	Range
⬇			
enableDetailBurn	`Boolean`	Enable burning and removing terrain details such as grass.	—
detailBurnSpeed	`Single`	How fast terrain details are reduced when burning. • Range [0,01..1]	Range
minimumBurntDetailMultiplier	`Single`	Minimum detail density multiplier left after burning. Use 0 for full removal. • Range [0..1]	Range
progressiveDetailFade	`Boolean`	Apply progressive detail fading instead of removing grass instantly.	—
⬇			
useBrushNoise	`Boolean`	Add noise to the burn brush to avoid perfect circular marks.	—
brushNoiseScale	`Single`	Noise scale used by the burn brush.	—
brushNoiseStrength	`Single`	Noise influence on the burn brush shape. • Range [0..1]	Range
⬇			
enableRegrowth	`Boolean`	Enable optional terrain regrowth after burn delay.	—
regrowthDelay	`Single`	Delay before a burnt area starts regrowing.	—
regrowthSpeed	`Single`	How fast terrain details and textures regrow toward the original terrain data. • Range [0,01..1]	Range
regrowTexture	`Boolean`	Allow burnt terrain texture to fade back to its original texture during regrowth.	—

Field	Type	Tooltip	Hints
regrowDetails	`Boolean`	Allow terrain details such as grass to regrow during regrowth.	—
⬇			
maxBurnRequestsPerFrame	`Int32`	Maximum number of burn requests processed per frame.	—
maxRegrowthRequestsPerFrame	`Int32`	Maximum number of regrowth requests processed per frame.	—
maxQueuedBurnRequests	`Int32`	Maximum number of queued burn requests. Extra requests are discarded when this limit is reached.	—
⬇			
drawDebugGizmos	`Boolean`	Draw debug gizmos for queued and recently processed burn requests.	—
logWarnings	`Boolean`	Log warnings when terrain burn requests cannot be processed.	—