Setting Fire – Documentation

SETTING
FIRE
TEAM VICTOR & BORIS
by BLInformatique
Dynamic fire spread, explosions, rain
wind & fire extinguishing system for Unity

Realistic Fire System with Propagation, Explosions and Weather Integration

---

Table of Contents

---

1. Introduction

Setting Fire is a complete system to simulate dynamic fire behavior in Unity.
It includes fire ignition, spread logic, particle effects, explosions, fade transitions, and environmental integration like rain and wind.

Designed to be lightweight, modular, and artist-friendly, it's perfect for both realistic simulations and stylized games.

What makes it special:

- Automatic fire distribution on any object
- Fire that reacts to rain and wind
- Interactive extinguisher with both input systems
- Full explosion support with debris & audio
- Powerful editor tools to streamline setup
- Compatible with first-person or third-person gameplay

2. Quick Setup

This section guides you through setting up the **Setting Fire system** from scratch in just a few minutes.

<u>Required Setup – New Input System</u>

Important: Before using the Game Tools demo scenes or any components supporting both input systems, you must:

Run :

Tools/BLInformatique/Run Setup Wizard

And follow the instructions.

Or :

Install the New Input System via the Unity Package Manager

In **Project Settings > Player > Active Input Handling**, select **"Both"** to enable compatibility with both **Old** and **New Input Systems**.This setup is **mandatory** for all input features to work correctly with your keyboard, mouse, and gamepad.

□ Step-by-step Setup Instructions

1. Import the Package

Make sure all Setting Fire files are inside your project under:

```
Assets/BLInformatique/SettingFire/
```

`And go to folder`

```
Assets/BLInformatique/SettingFire/
```

`Run your setting graphiq, Built-In or URP`

---

2. Install Tags and Layers

On first import, you'll be go to:

```
Tools > BLInformatique > Setting Fire > Install Tags And Layers
```

To install

- Tag: `"Torch"`
- Layer: `"Fire"`

□ Click **"Yes, install them"**
If you skipped it, you can add them manually in **Project Settings > Tags & Layers**.

---

3. Add the FireManager

Go to:

```
Tools > BLInformatique > Setting Fire > Create Fire Manager
```

➡□ This prefab controls wind, weather detection, and global propagation behavior.

---

4. Prepare Flammable Objects

Use the Fire Systeme tool:

```
GameObject > BLInformatique > Create Fire Systeme
```

Steps:

- Select your mesh parent object (e.g. “House”)
- Choose the Layer to assign (`Fire` recommended)
- Auto-assign colliders where needed
- □ Click **“Add FireEntity to selected”**

Each selected object will now contain a `FireEntity` component.

---

5. Assign a Fire Prefab

In each `FireEntity`, assign your preferred fire prefab (e.g. `MediumFlames`).
You can use any VFX or particle prefab that fits your visual style.

---

6. Test with a Torch

Place a torch in the scene and set:

- Tag: `Torch`
- Layer: `Fire`

When the torch touches a FireEntity, the object will ignite automatically!

---

7. Optional: Add the Fire Extinguisher

Use the menu:

```
Tools > BLInformatique > Setting Fire > Create Fire Extinguisher Object
```

It includes:

- Input System compatibility (old & new)
- Built-in VFX & SFX
- Fire detection & fade-out system

Summary Checklist

- Tags & Layers installed (`Torch`, `Fire`)
- `FireManager` in scene
- `FireEntity` on objects with fire prefab
- Torch or igniter present (tagged correctly)
- Optional extinguisher for gameplay

Next: **FireEntity Detailed Settings**

3. Required Tags & Layers

To ensure proper detection, ignition, and propagation of fire, **Setting Fire** relies on one tag and one layer:

Tag: `"Torch"`

Used to identify objects that can **ignite FireEntities**.
Any GameObject with this tag will be considered a valid igniter (e.g. torches, burning barrels, etc.).

Tip: You can assign this tag to any object that should trigger a fire — just make sure it has a collider and is in the correct layer.

Layer: `"Fire"`

Used for:

- Detecting nearby igniters (via physics checks)
- Extinguishing fire via raycasting
- Propagation & spread detection

Important: Any object that should **catch fire or be detected** (like the torch) must be placed on the `"Fire"` layer.

Auto-Installation Prompt

Go to:

```
Tools > BLInformatique > Setting Fire > Install Tags And Layers
```

A popup asks:

“Would you like to automatically add the required Tags and Layers for Setting Fire?”

□ Click **“Yes, install them”** to set everything up instantly.

If skipped, you can manually create them via:

```
Edit > Project Settings > Tags and Layers
```

Debugging Tip

If your fire is **not igniting**, check:

- Is the torch in the `"Fire"` layer?
- Does it have the `"Torch"` tag?
- Is `FireEntity.canBeIgnited` checked?

4. The FireEntity Component

`FireEntity` is the main component used to make any object flammable and reactive to fire, weather, explosions, and extinguishing systems.

It controls:

- Fire instantiation and visual fade-in/out
- Spread to nearby objects (with wind influence)
- Explosions and chain reactions
- Rain-based extinguishing
- Reaction to extinguishers
- Events on ignition or destruction

How to Add It

You can add the component in two ways:

- Manually:
  Add `FireEntity` to any GameObject via the Inspector.
- Automatically (recommended):
  Use the **Fire Systeme** window:

  ```
  GameObject > BLInformatique > Create Fire Systeme
  ```

Component Structure

`FireEntity` is organized into **4 tabs** to keep things clean and accessible:

Tab	Description
Fire	All settings related to ignition, visual effects, and fire prefab distribution
Weather & Propagation	Controls spread logic, wind influence, and rain-based extinguishing

Tab	Description
Optimization	Auto fade-out, lifetime, light toggling
Explosion	Controls explosion VFX, force, radius, and aftermath prefabs

---

Core Settings Overview

Setting	Description
`Fire Prefab`	Prefab of the fire particles to spawn
`Can Be Ignited`	Enables this object to catch fire
`Auto Ignite`	Checks for nearby igniters at runtime
`Fire Density`	Controls how many fire instances spawn across the mesh
`Always Place Ground Fire`	Ensures at least one fire at the base

When ignited, `FireEntity` distributes fire effects across its surface using its bounds and the parameters you've configured.

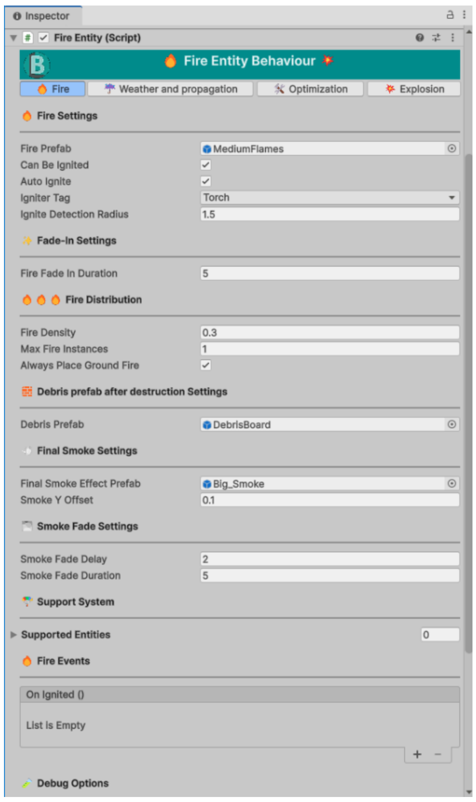
Advanced Features

- **Debris support**: Replace burned object with debris prefab after destruction
- **Smoke fade**: Final smoke effect fades over time for realism
- **Propagation Delay**: Control how fast the fire spreads
- **UnityEvents**: Easily hook custom logic with `OnIgnited()` and `OnExploded()`

In the next sections, we'll deep-dive into each tab with screenshots and explanations:

- **4.1 Fire Settings**
- **4.2 Weather & Propagation**
- **4.3 Optimization**
- **4.4 Explosion**

4.1 Fire Settings



Inspector

Fire Entity (Script)

Fire Entity Behaviour

Fire | Weather and propagation | Optimization | Explosion

Fire Settings

Fire Prefab	MediumFlames
Can Be Ignited	✓
Auto Ignite	✓
Igniter Tag	Torch
Ignite Detection Radius	1.5

Fade-In Settings

Fire Fade In Duration	5

Fire Distribution

Fire Density	0.3
Max Fire Instances	1
Always Place Ground Fire	✓

Debris prefab after destruction Settings

Debris Prefab	DebrisBoard

Final Smoke Settings

Final Smoke Effect Prefab	Big_Smoke
Smoke Y Offset	0.1

Smoke Fade Settings

Smoke Fade Delay	2
Smoke Fade Duration	5

Support System

Supported Entities	0

Fire Events

On Ignited ()

List is Empty

Debug Options

The **Fire** tab contains all the base parameters for controlling how a `FireEntity` reacts when it ignites, what visuals it spawns, and how it behaves visually.

Basic Fire Behavior

Property	Description
Fire Prefab	The particle prefab to instantiate when the object catches fire. Must contain a `ParticleSystem`.
Can Be Ignited	If unchecked, this object cannot catch fire (manually or by igniters).
Auto Ignite	If enabled, this object will regularly check for nearby igniters (tagged “Torch”).
Igniter Tag	The tag used to identify objects that can ignite this FireEntity.
Ignite Detection Radius	The distance within which an igniter can trigger the fire.

Tip: By default, torches should have `Tag = Torch` and `Layer = Fire` to be detected.

Fade-In Settings

Property	Description
Fire Fade In Duration	Controls how long the fire fades in when ignited. Smooth alpha blending is applied to all ParticleSystems.

This creates a **soft and realistic ignition** rather than instant flames.

Fire Distribution

When ignited, Setting Fire intelligently spreads multiple fire instances across the surface of your object, using its bounds.

Property	Description
Fire Density	Controls how many fire instances are distributed across the object's volume.
Max Fire Instances	Maximum number of simultaneous fire effects spawned on the object.
Always Place Ground Fire	Ensures a fire is always placed at the base of the object, even if surface fires are limited.

This ensures **immersive, realistic fire coverage** without needing manual placement.

Debris & Final Smoke

These options allow you to define **what happens when the object is fully burned** or destroyed.

Property	Description
Debris Prefab	The broken version of the object to instantiate after burning.
Final Smoke Effect Prefab	A smoke particle system to instantiate at the debris base.
Smoke Y Offset	Adjusts the vertical offset of the smoke spawn point.

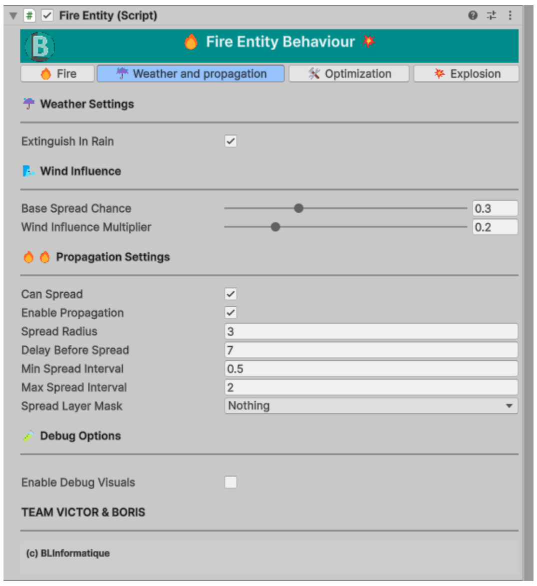
Combine this with `Auto Fade` and `Fire Lifetime` for **complete destruction cycles**.

Smoke Fade Settings

Property	Description
Smoke Fade Delay	Time before smoke starts to fade.
Smoke Fade Duration	Duration for the smoke to fully fade out.

Creates a **natural dissipation effect** after the fire is gone.

4.2 Weather & Propagation


Fire Entity (Script)

Fire Entity Behaviour

Fire | Weather and propagation | Optimization | Explosion

Weather Settings

Extinguish In Rain	✓

Wind Influence

Base Spread Chance	0.3
Wind Influence Multiplier	0.2

Propagation Settings

Can Spread	✓
Enable Propagation	✓
Spread Radius	3
Delay Before Spread	7
Min Spread Interval	0.5
Max Spread Interval	2
Spread Layer Mask	Nothing

Debug Options

Enable Debug Visuals	

TEAM VICTOR & BORIS

(c) BLInformatique

This tab controls how fire **spreads**, how it **reacts to rain**, and how **wind** affects its behavior. It's what makes **Setting Fire** more than just a visual effect — it's a living system.

Weather Settings

Property	Description
Extinguish In Rain	If enabled, the fire will automatically fade out when raining (based on your weather system).

Works with:

- `FireManager` and manual rain toggle
- `WeatherManager.isRaining`

- Full **Enviro** support (if integration is active)

Wind Influence

Wind can make fires spread faster or further, based on direction and intensity.

Property	Description
Base Spread Chance	The default chance that fire will spread to a nearby object (0–1).
Wind Influence Multiplier	Modifies spread chance based on how well wind aligns with the direction of nearby flammable objects.

The closer the target is to the **wind direction**, the higher the spread chance becomes.

Propagation Settings

Fire can propagate **autonomously** to nearby `FireEntity` components if enabled.

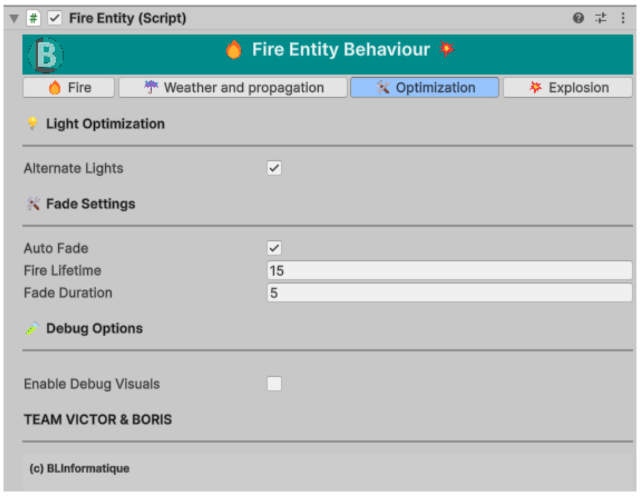
Property	Description
Can Spread	Activates this object's ability to spread fire (controls internal logic).
Enable Propagation	Enables actual fire spreading after ignition.
Spread Radius	The radius within which other `FireEntity` objects will be detected.
Delay Before Spread	Time (in seconds) before the first spread attempt occurs.
Min Spread Interval	Minimum cooldown time before the next spread attempt.
Max Spread Interval	Maximum cooldown between propagation attempts.
Spread Layer Mask	Layer(s) used to detect valid `FireEntity` targets for propagation.

Fires will spread **realistically**, only to valid and igniteable objects nearby.

Behind the Scenes

- Spread logic is evaluated via **Physics.OverlapSphere** and wind alignment using **Vector3.DotProduct**.
- All spreading is **controlled per-object**: full control, no global settings needed.

4.3 Optimization



Light Optimization

Alternate Lights ✓

Fade Settings

Auto Fade ✓
Fire Lifetime 15
Fade Duration 5

Debug Options

Enable Debug Visuals ☐

TEAM VICTOR & BORIS

(c) BLInformatique

The **Optimization** tab lets you configure how the fire fades out over time and how it behaves in terms of performance and light simulation.

Perfect for large scenes or performance-sensitive projects.

Light Optimization

Property	Description
Alternate Lights	Randomly enables/disables the light modules on fire particle systems.

This reduces the number of real-time lights in your scene while still preserving **dynamic and realistic ambiance**.

Fade Settings

These settings control how long the fire **lasts** and how it **disappears** once its time is up (or if extinguished).

Property	Description

Property	Description
Auto Fade	Automatically starts a fade-out timer once the fire ignites.
Fire Lifetime	Time in seconds before the fire starts fading.
Fade Duration	Time it takes for all fire visuals to smoothly disappear.

Combines with debris and smoke to simulate a **full burn cycle**.

Behavior Overview

With these options:

- Fire appears, burns, and disappears **without manual intervention**
- No more infinite fire draining performance
- You can fine-tune how quickly an area becomes quiet and cold again □➜□➜❄□

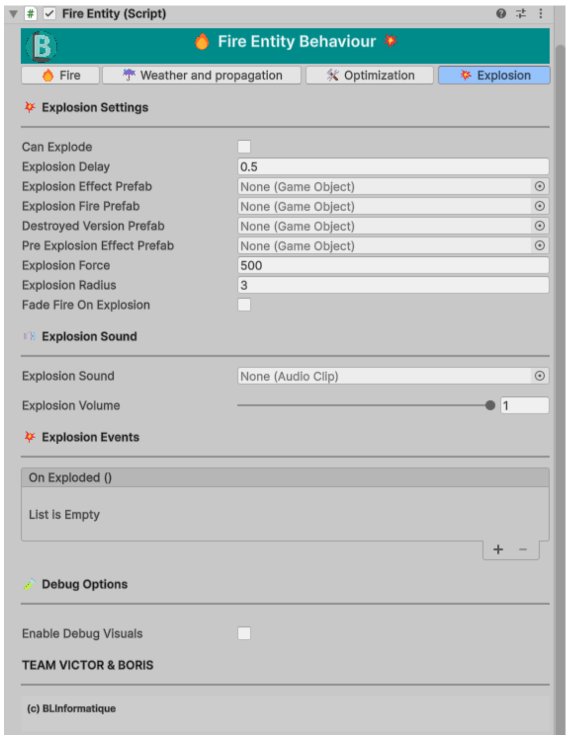
Tip for Large Scenes

If you're burning entire buildings or forests:

- Reduce `Max Fire Instances`
- Enable `Alternate Lights`
- Keep `Auto Fade` active

This allows **realistic burn sequences** without overwhelming the engine.

4.4 Explosion



Explosion Settings

Can Explode ☐
Explosion Delay 0.5
Explosion Effect Prefab None (Game Object)
Explosion Fire Prefab None (Game Object)
Destroyed Version Prefab None (Game Object)
Pre Explosion Effect Prefab None (Game Object)
Explosion Force 500
Explosion Radius 3
Fade Fire On Explosion ☐

Explosion Sound

Explosion Sound None (Audio Clip)
Explosion Volume 1

Explosion Events

On Exploded ()
List is Empty

Debug Options

Enable Debug Visuals ☐

TEAM VICTOR & BORIS

(c) BLInformatique

This tab allows your `FireEntity` to **explode** after ignition or under certain conditions.
It includes **visuals**, **force**, **VFX**, **replacement objects**, and **audio**.

Explosion logic is fully modular and can be triggered **automatically** after a delay or by other scripts.

Explosion Settings

Property	Description
Can Explode	Enables explosion behavior for this object.
Explosion Delay	Time (in seconds) after ignition before the explosion triggers.
Explosion VFX Duration	Defines how long the explosion visual effect (VFX) remains in the scene after being triggered. (This value is crucial to **avoid looping particles)**
Explosion Effect Prefab	The VFX prefab to instantiate at the moment of explosion.
Explosion Fire Prefab	Fire prefab to spawn at the explosion origin. Useful for chain reactions.

Property	Description
Destroyed Version Prefab	Optional broken version of the object to instantiate after explosion.
Pre-Explosion Effect Prefab	Optional effect (e.g., flashing or smoke) shown before the explosion delay ends.
Explosion Force	The force applied to nearby rigidbodies using `AddExplosionForce()`.
Explosion Radius	The radius within which objects are affected.
Fade Fire On Explosion	Smoothly fades existing fires instead of destroying them instantly.

Combine `Fade Fire On Explosion` with a short fade time for a **cinematic chain reaction** effect.

Explosion Sound

Property	Description
Explosion Sound	The audio clip to play at the explosion location.
Explosion Volume	Volume multiplier (0–1) for the explosion sound.

Played using `AudioSource.PlayClipAtPoint()` at the center of the explosion.

Explosion Events

Property	Description
OnExploded	UnityEvent triggered when this object explodes.

Use it to:

- Trigger animations
- Activate post-processing effects

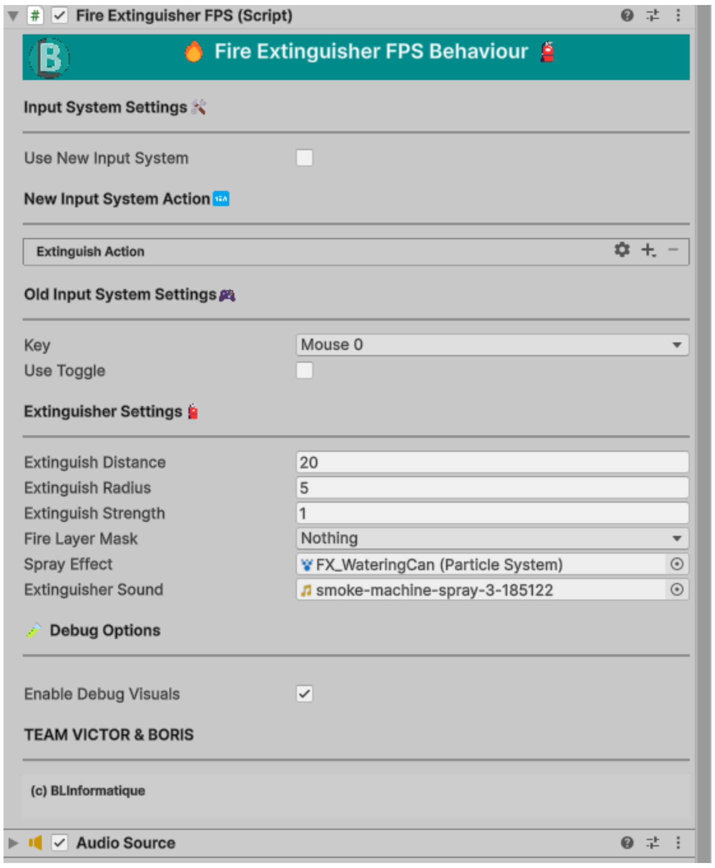
- Start other fires or explosions

Typical Explosion Sequence

1. Fire ignites
2. `Explosion Delay` countdown begins
3. Pre-explosion VFX appears
4. **BOOM** – explosion VFX + sound + damage
5. Debris and smoke remain
6. `FireEntity` is destroyed

Explosion integrates seamlessly with fire spread and debris systems.

5. Fire Extinguisher FPS


Fire Extinguisher FPS (Script)

Fire Extinguisher FPS Behaviour

Input System Settings

Use New Input System: ☐

New Input System Action

Extinguish Action

Old Input System Settings

Key	Mouse 0
Use Toggle	☐

Extinguisher Settings

Extinguish Distance	20
Extinguish Radius	5
Extinguish Strength	1
Fire Layer Mask	Nothing
Spray Effect	FX_WateringCan (Particle System)
Extinguisher Sound	smoke-machine-spray-3-185122

Debug Options

Enable Debug Visuals: ✓

TEAM VICTOR & BORIS

(c) BLInformatique

Audio Source

The `FireExtinguisherFPS` component allows players to extinguish active fires directly from a first-person perspective.
It supports both the **Old Input System** and **Unity's New Input System**, and includes built-in **sound**, **particles**, and **visual feedback**.

Key Features

- Detects fire within a given **range and radius**
- Gradually **fades out fire particles**
- Plays **looping sound** while active
- Includes spray **particle system**
- Compatible with **FireEntity** logic
- Supports both **toggle mode** and **hold to spray**

Setup Instructions

You can create a ready-to-use prefab via:

```
Tools > BLInformatique > Setting Fire > Create Fire Extinguisher Object
```

Or attach the `FireExtinguisherFPS` script manually to any GameObject (preferably the hand or tool in your FPS).

Component Parameters

Input Settings

Property	Description
Use New Input System	If true, uses Unity's new `InputAction` asset. Otherwise uses classic `KeyCode`.
Key (Old Input System)	The key used to activate the extinguisher (default: Mouse0).
Use Toggle	Enables toggle mode: press once to start, press again to stop.

New Input System

Property	Description
Extinguish Action	`InputAction` reference for activation (e.g., Trigger or Mouse).

Extinguisher Settings

Property	Description
Extinguish Distance	How far the extinguisher ray can reach (default: 5 units).
Extinguish Radius	Area around the hit point where particles are affected.
Extinguish Strength	How quickly fire is extinguished (affects alpha, size, emission).
Fire Layer Mask	Layer(s) used to detect fire particle systems (should include `"Fire"`).
Spray Effect	Particle system that plays when extinguisher is active.
Extinguisher Sound	Looping sound played while extinguishing.

Debug Options

Property	Description
Enable Debug Visuals	Shows rays, impact zones, and logged feedback in the console.

Extinguishing Logic

- Casts a **ray** from the center of the screen (like shooting).
- Detects all **ParticleSystems** in range on the `"Fire"` layer.
- Gradually reduces:
  - Particle **alpha**
  - **Size**
  - **Emission rate**
- Adds an `AutoFadeParticles` script if not present.
- If a `FireEntity` is detected and **burning**, it calls `StartFadeOut()`.

Sound Handling

- Uses a `looping AudioSource` on the same GameObject.
- Only plays sound while actively extinguishing.
- Stops when input is released (or toggled off).

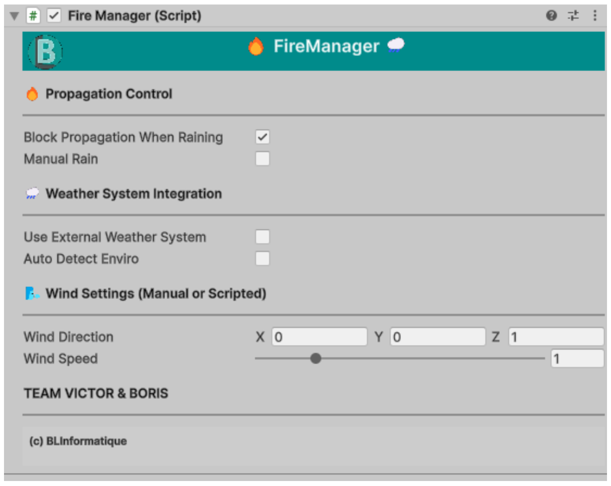
Performance Note

The script uses:

```
FindObjectsByType<ParticleSystem>() // Unity 6000+
```

Optimized and filtered by Layer and proximity for efficiency.

7. FireManager & Weather Integration



Fire Manager (Script)

FireManager

Propagation Control

Block Propagation When Raining ✓

Manual Rain

Weather System Integration

Use External Weather System

Auto Detect Enviro

Wind Settings (Manual or Scripted)

Wind Direction X 0 Y 0 Z 1

Wind Speed 1

TEAM VICTOR & BORIS

(c) BLInformatique

The `FireManager` is a singleton-based controller that manages **weather conditions**, **fire propagation control**, and **wind settings** across the entire scene.

It determines **whether propagation is allowed**, and integrates with **Enviro** (if present) or a **custom rain toggle**.

Global Fire Control

Property	Description
Block Propagation When Raining	If true, disables all fire spread while raining.
Manual Rain	Simulates rain manually without needing a weather system.
Use External Weather System	Enables detection via an external system (like Enviro).
Auto Detect Enviro	Automatically looks for `EnviroManager` in the scene.

If fire spreading isn't working, this is the first place to check.

Wind Settings

Wind affects the **directional spread** of fire when enabled in each `FireEntity`.

Property	Description
Wind Direction	A `Vector3` defining the direction of wind across the map.
Wind Speed	Controls how much influence the wind has on spread chance (0–5).

Spread is boosted when a neighboring object is **aligned with the wind vector**.

Rain Detection Methods

Depending on your setup, the fire system checks rain in one of three ways:

1. Using FireManager Only

- Enable `Manual Rain` from code:

```
FireManager.Instance.manualRain = true;
```

2. Using Enviro

- If `autoDetectEnviro` is true, `FireManager` will try to fetch rain status from Enviro API (customizable).

3. Using WeatherManager

- You can control weather globally via:

```
WeatherManager.isRaining = true;
```

This is a static fallback for simple projects.

FireEntity Integration

Each `FireEntity` internally checks:

```
if (!FireManager.Instance.IsPropagationAllowed) return;
```

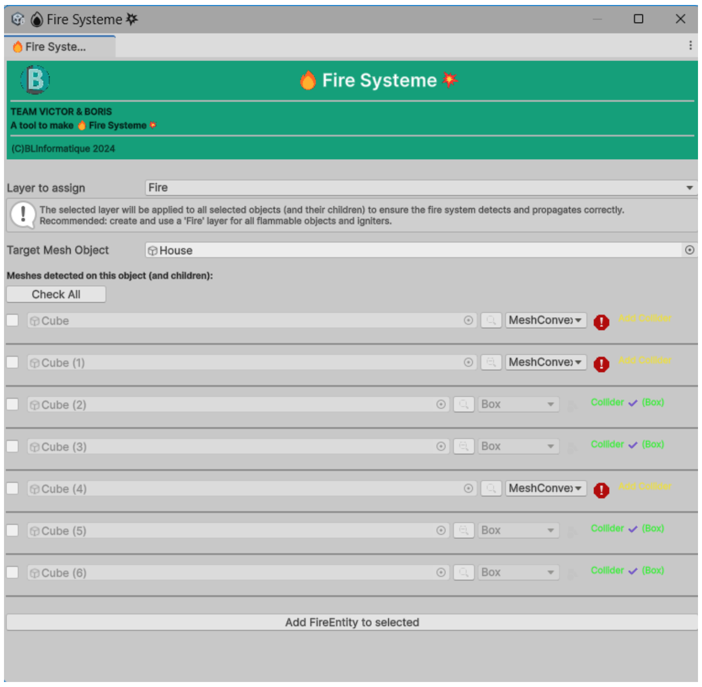
So disabling propagation from rain **automatically affects** all entities in the scene.

Debug Tip

If propagation **never starts**, check:

- `Block Propagation When Raining` is disabled
- Or that it's **not raining** in `FireManager` or `WeatherManager`

8. Fire Systeme Editor Window



TEAM VICTOR & BORIS
A tool to make Fire Systeme
(C)BLInformatique 2024

Layer to assign: Fire

The selected layer will be applied to all selected objects (and their children) to ensure the fire system detects and propagates correctly.
Recommended: create and use a 'Fire' layer for all flammable objects and igniters.

Target Mesh Object: House

Meshes detected on this object (and children):

Cube	MeshConve›	Add Collider
Cube (1)	MeshConve›	Add Collider
Cube (2)	Box	Collider ✓ (Box)
Cube (3)	Box	Collider ✓ (Box)
Cube (4)	MeshConve›	Add Collider
Cube (5)	Box	Collider ✓ (Box)
Cube (6)	Box	Collider ✓ (Box)

The **Fire Systeme** window is a custom Unity Editor tool designed to **prepare multiple GameObjects** for use with the `FireEntity` system — in a few clicks.

It automatically:

- Adds colliders
- Sets the correct layer
- Assigns `FireEntity`
- Adds a proper Rigidbody for interaction/destruction

How to Open It

Go to:

```
GameObject > BLInformatique > Create Fire Systeme
```

This opens a window with a user-friendly interface.

Window Overview

Layer Assignment

- Choose which layer to apply (e.g., `"Fire"`)
- Applies to all selected objects and **their children**

□ Required for propagation & detection to work correctly

Object Selection

- Use the `Target Mesh Object` field to select any GameObject
- All **MeshRenderers** from that object and its children will be listed

---

Mesh List & Colliders

For each mesh:

- Shows current collider status
- Lets you assign a new one (Box, Sphere, Capsule, or Convex Mesh)
- Uses Unity icons and tooltips for clarity

If a collider is missing, a **yellow warning icon** is shown
□ If it's already valid, a **green checkmark** is displayed

Final Step: Add FireEntity

At the bottom, click:

```
Add FireEntity to selected
```

This will:

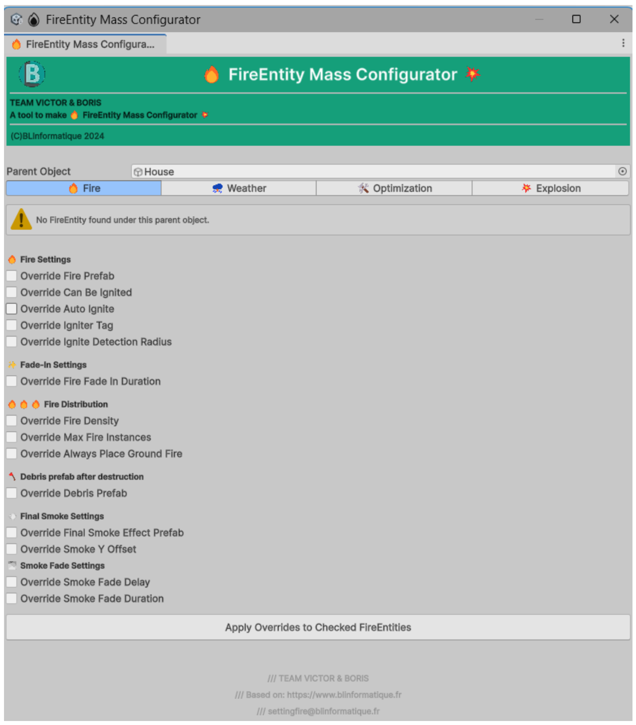
- Add the `FireEntity` component to each checked object
- Set the correct **Layer**
- Add a **Rigidbody** (with `isKinematic = true` and `useGravity = false`)
- Apply selected **collider types**

It's the **fastest way to prepare an entire scene** of objects for fire.

Tip

Use this tool after importing buildings, furniture, barrels, or destructible props.
One click, and they're ready to burn.

9. FireEntity Mass Configurator


FireEntity Mass Configurator

FireEntity Mass Configura...

FireEntity Mass Configurator

TEAM VICTOR & BORIS
A tool to make FireEntity Mass Configurator

(C)BLInformatique 2024

Parent Object: House

Fire | Weather | Optimization | Explosion

No FireEntity found under this parent object.

Fire Settings
- [ ] Override Fire Prefab
- [ ] Override Can Be Ignited
- [ ] Override Auto Ignite
- [ ] Override Igniter Tag
- [ ] Override Ignite Detection Radius

Fade-In Settings
- [ ] Override Fire Fade In Duration

Fire Distribution
- [ ] Override Fire Density
- [ ] Override Max Fire Instances
- [ ] Override Always Place Ground Fire

Debris prefab after destruction
- [ ] Override Debris Prefab

Final Smoke Settings
- [ ] Override Final Smoke Effect Prefab
- [ ] Override Smoke Y Offset

Smoke Fade Settings
- [ ] Override Smoke Fade Delay
- [ ] Override Smoke Fade Duration

Apply Overrides to Checked FireEntities

/// TEAM VICTOR & BORIS
/// Based on: https://www.blinformatique.fr
/// settingfire@blinformatique.fr

The **FireEntity Mass Configurator** is a powerful Editor tool that lets you **batch-edit and override** settings for multiple `FireEntity` components at once.

Perfect for making global changes to:

- Fire behavior
- Spread logic
- Fade timings
- Explosion parameters

How to Open It

Right-click any GameObject in the hierarchy and go to:

```
GameObject > BLInformatique > FireEntity Mass Configurator
```

This opens a dedicated window with a smart tab system.

Main Interface Features

Object Scanning

- Select a **parent GameObject**
- All `FireEntity` components found in children will be listed
- You can **check/uncheck** which ones to apply overrides to

You can also ping or highlight each object directly from the window

---

Tabbed Settings Navigation

Tabs are split into four categories, matching the `FireEntity` structure:

- Fire
- Weather
- Optimization
- Explosion

✔□ Smart Override System

Each property has:

- A **checkbox** to enable override
- The current value (editable)

Once you're ready:
Click **“Apply Overrides to Checked FireEntities”**

This will instantly update all selected objects with the new values.

Quality of Life Features

- Initial values auto-filled from the **first selected FireEntity**
- LayerMask and prefab fields are fully editable
- Override **only what you need**, leave the rest unchanged
- Easily manage **hundreds of FireEntities** across large maps

---

Example Use Cases

- Change all fire prefabs after a VFX upgrade
- Enable `Extinguish in Rain` globally
- Set fire lifetimes and fade durations for optimization

- Enable explosions and set common audio clip or VFX

Debug Note

If no FireEntities are listed:

- Make sure you've run the **Fire Systeme** tool first
- Or confirm that the selected object has `FireEntity` components in children

9. Custom Events – OnIgnited & OnExploded

The `FireEntity` component includes two powerful **UnityEvents**:

- **`OnIgnited`**
- **`OnExploded`**

These allow you to **trigger custom logic** in your game at runtime, **without writing any code**.

`OnIgnited` Event

When it triggers:

- This event is fired when the object **successfully catches fire** (via igniter detection or manual call to `Ignite()`)

Use cases:

- Play a custom animation (e.g., a wood pile catching fire)
- Enable a sound or voice line
- Start a countdown
- Activate a light or shake the camera
- Broadcast to a quest system

How to assign:

1. Select the GameObject with `FireEntity`
2. In the **Inspector**, scroll to the **"Fire Events"** section
3. Click the + button to add a listener
4. Drag and drop the target object with the method to call

`OnExploded` Event

When it triggers:

- This event is invoked right after the object **explodes**,
  either from delayed explosion logic or triggered programmatically.

Use cases:

- Chain multiple explosions (by igniting other `FireEntity`s)
- Play special music or cutscene
- Update objectives (e.g., "Destroy all barrels")
- Spawn AI enemies after a trap is triggered

Combined Example

Burning a gas tank:

- `OnIgnited`:
  Play a warning sound
  Start flashing the object
  Trigger a timer
- `OnExploded`:
  Spawn debris
  Ignite nearby objects
  Update mission status

Pro Tip

You can also trigger `Ignite()` or `Explode()` manually from script:

```
GetComponent<FireEntity>().Ignite();
GetComponent<FireEntity>().StartCoroutine("HandleExplosion");
```

Or chain from another explosion:

```
otherFireEntity.OnExploded.AddListener(() => myFireEntity.Ignite());
```

10. Integration with Enviro Weather System

Setting Fire fully supports **Enviro – Sky and Weather**, one of the most popular weather systems for Unity.

With just a few toggles, fire will react to **Enviro’s weather conditions**, like **rain, storm, or wetness** – and stop spreading if it’s too humid.

□ How to Enable Enviro Integration

1. Go to your scene’s `FireManager`
2. Enable:
    - **Use External Weather System**
    - (Optional) **Auto Detect Enviro**

The system will try to find `EnviroManager.instance` at runtime.

How Fire Reacts to Enviro

When integration is active:

- **Propagation will be blocked** if it’s raining or wet
- `FireEntity` will **auto-extinguish** if `Extinguish In Rain` is enabled

This is determined by:

```
EnviroManager.instance.Environment.Settings.wetness > 0.25f
```

You can tweak this threshold in your own logic if needed.

EnviroRainToggle Script

You can also manually control rain status from any script using our included:

```
EnviroRainToggle.cs
```

This script lets you:

- Trigger a **rain preset**
- Trigger a **clear weather preset**

Useful for **demo scenes**, **testing**, or custom weather triggers.

```
public void SetRain(bool enable)
{
    EnviroManager.instance.Weather.ChangeWeather(enable ? rainWeatherName :
clearWeatherName);
}
```

Fallback Option – WeatherManager

For simple projects, you can use:

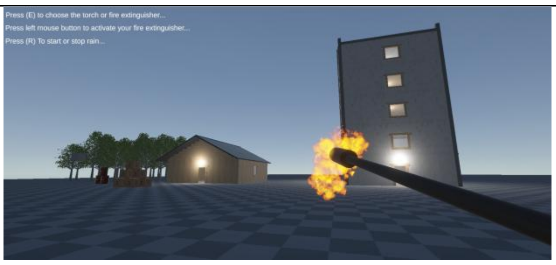
```
WeatherManager.isRaining = true;
```

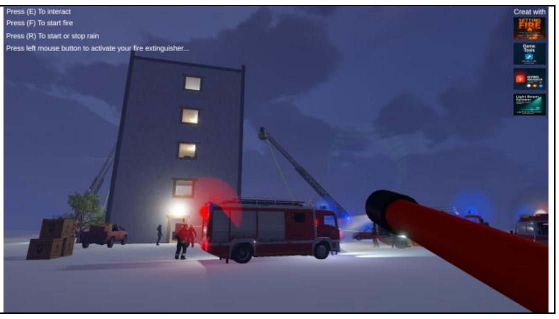
This boolean will be checked by all `FireEntity` components, allowing you to simulate rain **without any external system**.

Summary

Weather System	FireManager Behavior
Enviro (auto)	Uses `EnviroManager` wetness
Manual script	Set `manualRain = true`
Fallback	Uses `WeatherManager.isRaining` static value

10. Demo Scene Overview


Press (E) to choose the torch or fire extinguisher...
Press left mouse button to activate your fire extinguisher...
Press (R) To start or stop rain...


Press (E) To interact
Press (F) To start fire
Press (R) To start or stop rain
Press left mouse button to activate your fire extinguisher...
Creat with

Setting Fire comes with **two complete demo scenes** showcasing the full range of features — from a basic torch ignition to a full firefighter simulation with weather and tools integration.

Demo Scene 1 – Simple & Modular

This scene is designed for **quick testing** and **learning the basics** of the fire system.
There's no dependency on external assets like Enviro.

Key Features:

- □ **Toggle torch or extinguisher** with `E`
- □ **Left mouse click** to activate the extinguisher

- □ **Start/Stop rain** with `R` (via `WeatherManager`)
- Ignite:
  - A **wooden house**
  - A **tall concrete building**
  - A **forest zone**
  - **Explosive barrels** and **flammable crates**
- Objects react with spread, destruction, smoke, and explosion
- Real-time extinguishing using `FireExtinguisherFPS`

This scene is perfect for learning how the system behaves under various conditions without any third-party dependency.

Demo Game 2 – Full Interactive Firefighting Simulation

This playable demo is a **full simulation scene** designed for immersion and realism.

It integrates:

- **Enviro – Sky & Weather** for real-time rain handling
- **Game Tools** for input, UI prompts, and toggles
- **Gyro Manager** for flashing lights and vehicle effects
- **Light Beam System** for realistic URP headlights & spotlights

Key Gameplay Elements:

- **Firefighters** actively extinguishing the building
- You can **replace them** and take control:
  - Climb the ladder platform
  - Extinguish from above or on foot
- Change fire state, interact with objects (`E`)
- Rain dynamically affects propagation
- Fire spreads through floors, crates, and nearby barrels

On-Screen Instructions:

- `E` to interact
- `F` to start fire manually
- `R` to toggle rain
- Mouse to extinguish

This scene is designed to demonstrate **full integration** between Setting Fire and all TEAM VICTOR & BORIS tools.

Folder Locations

Scene	Path
Simple Demo	`Assets/BLInformatique/SettingFire/Scenes/Demo_Simple.unity`
Full Enviro Demo	`Download from https://www.blintormatique.fr`

12. Troubleshooting & Tips

Having trouble getting fire to ignite, spread, or extinguish properly? This section lists common problems, their causes, and solutions.

□ Fire doesn't start

Possible Causes:

- The object does **not have** a `FireEntity` component
- `Can Be Ignited` is **unchecked**
- The igniter **has no Tag or wrong Tag**
- The igniter is **not on the "Fire" layer**
- No **collider** on the object

□ Solutions:

- Make sure you used the **Fire Systeme** tool to prepare the object
- Check that the torch uses:
  `Tag = Torch` and `Layer = Fire`
- Ensure `Auto Ignite` is enabled or call `Ignite()` manually

□ Fire doesn't spread

Possible Causes:

- `Enable Propagation` is **unchecked**
- Spread radius is too small
- No nearby `FireEntity` objects
- It's **raining**, and propagation is blocked

□ **Solutions:**

- Check `FireEntity`’s **Weather & Propagation** tab
- Set `Block Propagation When Raining = false` in `FireManager`
- Increase `Spread Radius` to reach neighbors

□ Fire disappears too fast

Possible Causes:

- `Auto Fade` is enabled
- `Fire Lifetime` is too short
- Rain is extinguishing fire

□ **Solutions:**

- Disable `Auto Fade` or increase lifetime
- Disable `Extinguish In Rain` in FireEntity
- Toggle `manualRain` in `FireManager` to test without weather

□ Fire Extinguisher doesn't work

Possible Causes:

- Incorrect LayerMask (not targeting "Fire")
- `Extinguish Distance` too low
- Extinguisher not active

□ **Solutions:**

- Set **LayerMask** to include "Fire"
- Test with `Enable Debug Visuals` to see the raycast
- Make sure input key or InputAction is correctly bound

□ Explosion never happens

Possible Causes:

- `Can Explode` is **disabled**
- `Explosion Delay` too long
- Fire never ignites

□ **Solutions:**

- Enable `Can Explode`
- Set a short delay (e.g., 0.5s)
- Trigger ignition manually with script or igniter

Resetting the Scene

If your scene is acting up:

- Recreate the `FireManager` via menu
  `Tools > BLInformatique > Setting Fire > Create Fire Manager`
- Re-run the `Fire Systeme` window on your object
- Reassign the `Fire Prefab` if broken

Bonus Tips

- Use **low-poly debris** and optimized smoke for big scenes
- Combine with `Game Tools` for toggle inputs and HUD
- Call `FireEntity.Ignite()` from triggers, enemies, AI, or scripts

13. Support & Credits

Setting Fire was crafted with passion, precision, and a bit of controlled chaos by:

□TEAM VICTOR & BORIS

Two developers united by one mission:
Deliver powerful tools with style, reliability, and fun.

Need Help? Questions? Suggestions?

We're here to help.

Contact Support:

Email: `settingfire@blinformatique.fr`
Website: https://www.blinformatique.fr

Whether you're a beginner or a veteran dev, we're always happy to help improve your fire simulation experience.

Other Tools from the Team

You'll find these tools used in the Setting Fire demo scenes:

- **Game Tools** – Input manager, toggles, key events, and UI etc…
- **Gyro Manager** – Flashing lights, emergency systems
- **Light Beam System** – URP-ready light cones and volumetrics
- And many more to come...

Browse all our assets here:
https://assetstore.unity.com/publishers/53650

Built with Unity 6000.1.7f1

- Fully compatible with URP
- Designed for FPS / Simulation / Sandbox / Training
- Tested in real demo projects
- Organized, clean code

License

A Final Word

"We don't just simulate fire… we make it part of the story."
— Victor & Boris