

Session Expressions

JaThePlayer edited this page on Jan 5 · [7 revisions](#)

Many entities in Frost Helper which accept flags as conditions can instead use **Session Expressions**. (Check tooltips to see if they are).

Instead of simply checking a single flag, **Session Expressions** can check multiple flags and even session counters, and perform arithmetic and logic operations on them.

A condition using these Expressions is considered to be met if the value of the Expression is not 0.

Reading Session State

- `flagName` - Providing a flag name on its own checks if that flag is set, returning 1 if it is, 0 otherwise.
- `!flagName` - Inverts the logic for checking flags, returning 0 if the flag is set, 1 otherwise.
- `#counterName` - Reads the value of a Session Counter instead of a flag.

Commands

All commands start with `$`, and allow you to get access to various values, not just those coming from the session. Mods can add their own commands.

- `$deathsHere` - Returns how many times the player has died in this room, resets after a screen transition.
- `$deaths` - Returns how many times the player has died this session.
- `$hasGolden` - Returns whether the player is carrying a golden berry. 1 if they are, 0 otherwise.
- `$restartedFromGolden` - Returns 1 if the current session started due to a golden death.
- `$coreMode` - Returns the current core mode: 0 if not set, 1 if hot, 2 if cold.
- `$photosensitive` - Returns whether Photosensitive Mode is enabled. 1 if it is, 0 otherwise.
- `$dashes` - Current player dash count.
- `$maxDashes` - Maximum allowed player dash count.
- `$stamina` - Current player stamina. (float)
- `$speed.x` - Player's speed on the X-axis.
- `$speed.y` - Player's speed on the Y-axis.
- `$pi` - The value of Pi
- `$dtime` - The delta time between frames, taking into account Assist Mode options.

Arithmetic

When working with Session Counters, it can be useful to perform arithmetic on them.

You can use basic arithmetic operators: `+` `-` `*` `/` `%` `//` and comparison operators `<`, `<=`, `>`, `>=`, `==`, `!=`

- For example: `#counterA + #counterB < 10` checks if the sum of counters `counterA` and `counterB` is less than 10.

Because flags get converted to 0 and 1, you can perform arithmetic on flags as well.

- For example: `flagA + flagB + flagC == 2` checks if exactly 2 of the 3 provided flags are true.

▼

Pages

14

Find a page...

▶

Home

▶

Activators

▶

Complex Property Types

▶

Custom Colliders

▶

Custom Spinners

▶

Entity Movers

▶

Entity Rainbowifier

▶

Lua Badeline Boss Starter Guide

▶

New Decal Registry Attributes

▶

Rainbow Tileset Controller

▼

Session Expressions

Reading Session State

Commands

Arithmetic

Logic

Handling Inputs

Buttons

Directional Inputs

Functions

Example Usage

Session Counter Trigger

Interactions with other mods

API

▶

Shader Starter Guide

▶

Shader Wrapper for Stylegrounds

▶

Useful Ahorn Scripts

Clone this wiki locally

https://github.com/JaThePlayer/Frost

NOTE: The `/` operator will perform integer division when supplied with integers on both the left and right side. That is, `5 / 2 == 2` . If you wish to force division on floating-point numbers instead, you can use the `//` operator, in which case `5 // 2 == 2.5`

Logic

`&&` and `||` operators can be used to compare 1/0 values, mostly useful for flags:

- For example: `(flagA && flagB) || flagC` is met if either both flagA and B are set, or flagC. (or all of them at once)
- For example: `#money >= #cost && canBuyUpgrades`

Handling Inputs

As of Frost Helper 1.61.0, Session Expressions can also read player inputs.

Buttons

To read vanilla buttons, you may use any of these Commands:

- `$input.esc` , `$input.pause`
- `$input.menuLeft` , `$input.menuRight` , `$input.menuUp` , `$input.menuDown`
- `$input.menuConfirm` , `$input.menucancel` , `$input.menujournal`
- `$input.quickrestart`
- `$input.jump`
- `$input.dash`
- `$input.grab`
- `$input.talk`
- `$input.crouchDash` - Demodash

Modded buttons can be read via `$input.mod.modName.buttonName` , for example `$input.mod.MaxHelpingHand.ShowHints` .

By default, these commands return 1 if the button is held, 0 if it isn't. If you wish to check for something else, you can add a suffix to the command:

- `.check` - default behavior.
- `.pressed` - Whether the button just got pressed.
- `.released` - Whether the button is not held.

For example, `$input.grab.pressed` checks if the player has just pressed the Grab button.

Directional Inputs

To read directional inputs, you can use these commands:

- `$input.aim` - used for dashing.
- `$input.feather` - used for flying in feathers.
- `$input.mountainaim` - used in the overworld.

Using one of those commands alone won't do much, you need to specify which axis you want to get with the right suffix:

- `.x`
- `.y`

These values are floats in the range [-1, 1]. When playing on a controller, these might be non-integer values, but on keyboard, they will always be -1, 0 or 1. Make sure to keep this in mind, and check the *sign* of the values (using `< 0` or `> 0`) in most cases.

For example `$input.aim.y < 0` checks if the player is aiming downwards.

Functions

Functions can be called like `$func(arg1, arg2, ...)` :

- `$min(arg1, arg2, ...)` - returns the smallest value from all provided arguments.
- `$max(arg1, arg2, ...)` - returns the largest value from all provided arguments.
- `$clamp(x, min, max)` - clamps the value of `x` so that its between `min` and `max`
- `$abs(x)` - Returns the absolute value of `x`
- `$sin(x)` , `$cos(x)` , `$tan(x)` - Calculates trigonometrical functions, `x` is assumed to be in radians. (Tip: `$pi`)

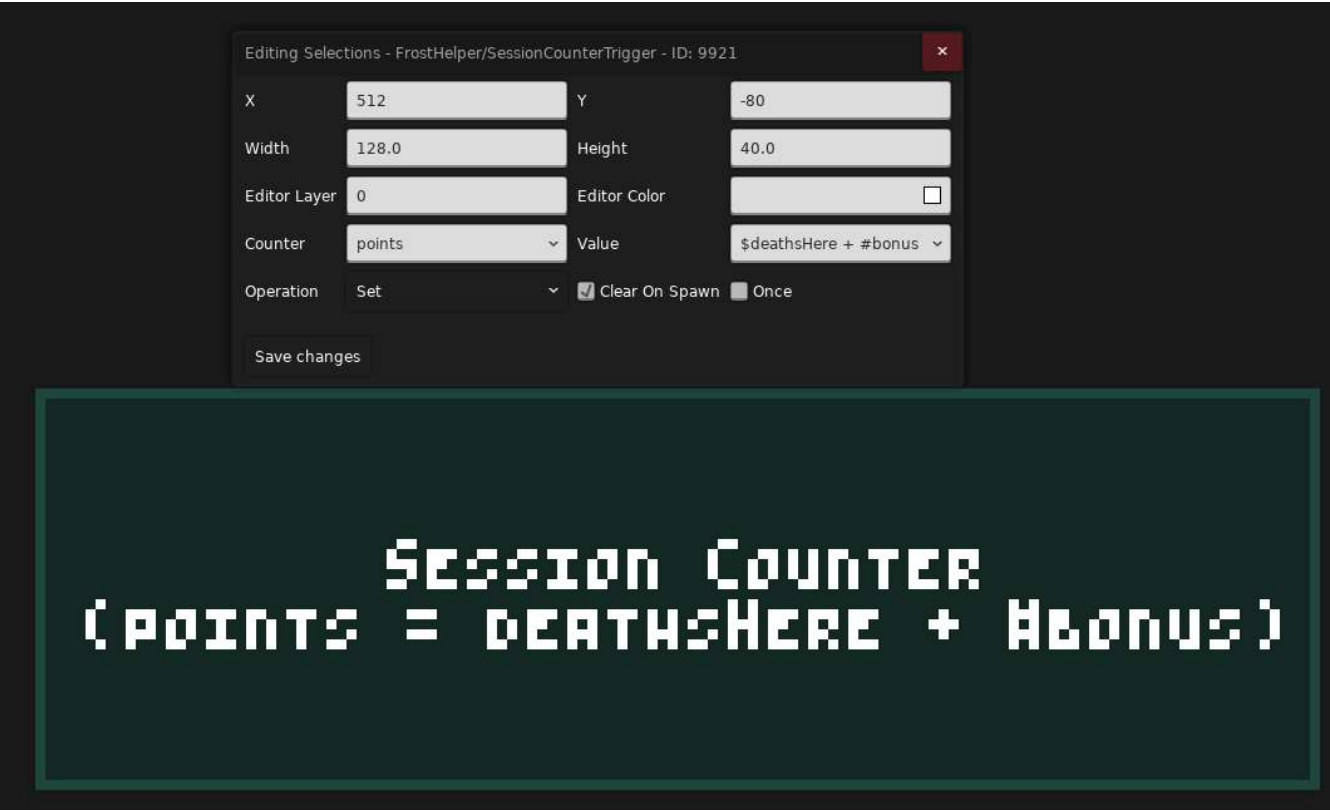
Example Usage

Session Counter Trigger

Session Counter Triggers support Session Expressions in the `value` field (as do many other session counter related triggers) However, there are 2 major differences when using Expressions in them:

- For backwards compatibility, **providing a name on its own treats that name as a Session Counter, not a flag!** (That is, setting Value to `name` will read the *session counter* called `name` , not a flag! However, `name + 2` will read a *flag* called `name` !)
- Expressions are treated as returning a number, not a boolean value.

For example, if you want to set a Counter `points` to the sum of the deaths in the current room plus the value of a Counter `bonus` :

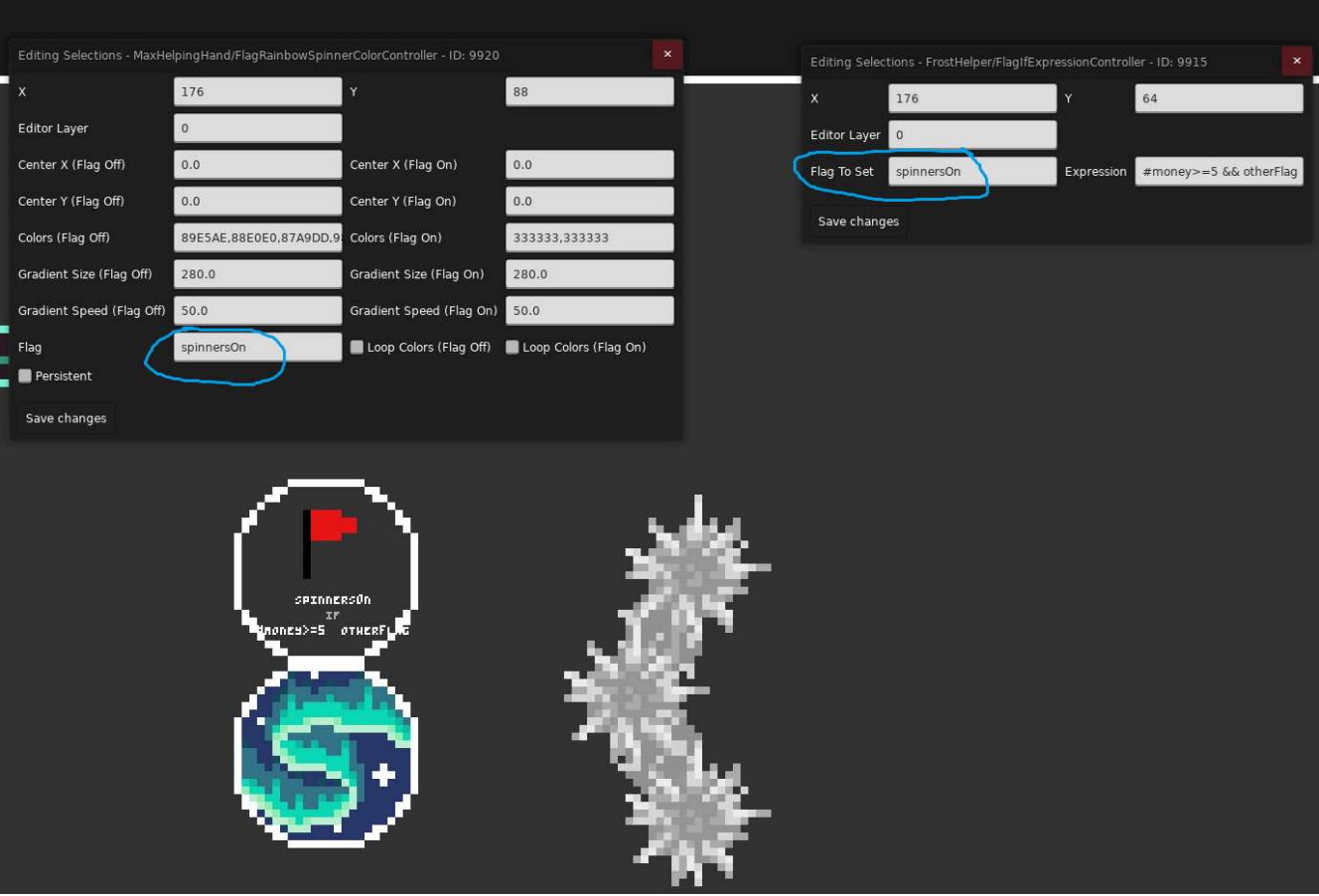


Interactions with other mods

While other mods might not support this feature, you can still make use of it to control other flag-based entities.

Say you want to change rainbow spinner colors when `#money>=5 && otherFlag` using a Flag Rainbow Spinner Color Controller.

You can use a `Flag If Expression` controller entity, which sets a flag if a Session Expression is met, and unsets it immediately when it isn't.



API

Frost Helper provides a MonoMod Export api for creating and evaluating Session Expressions:
[Link](#)