SYSTEM CONDITIONS

View online: https://www.construct.net/en/make-games/manuals/construct-3/system-reference/system-conditions

This section describes all the conditions in the built-in System object in Construct. They are listed in the order they appear in the Add Condition dialog.

Angles

Note angles in Construct start with 0 degrees facing right and increment clockwise.

Is between angles

True if a given angle is between the two other angles in degrees. The first and second angles must be in clockwise order. That is, *X is between 0 and 45 degrees* is true if the angle X is in a 45 degree area, but *X is between 45 and 0 degrees* is true if X is in the 315 degree area from 45 degrees through 0 degrees. The first angle is inclusive, but the second angle is exclusive, to ensure adjacent ranges are handled correctly.

Is clockwise from

True if a given angle is clockwise from another angle in degrees, or in other words, if it is 180 degrees or less in a clockwise direction from Angle 2 to Angle 1. Invert to test if anticlockwise instead. For example, 45 degrees is clockwise from 0 degrees, but 0 degrees is anticlockwise from 45 degrees. Angle 1 is the angle to test, and Angle 2 is the reference angle to test whether Angle 1 is clockwise from.

Is within angle

True if an angle is within a number of degrees of another angle. This is more reliable than testing if an angle exactly equals an angle, e.g. *X within 0.5 degrees of 90 degrees* is probably better than X equals 90 degrees, since there are many cases an angle can be very close to, but not exactly, 90 degrees.

General

Compare two values

Compare any two expressions (which can either numbers or text) with each other. They can be compared as *Equal*, *Not equal*, *Less*, *Less or equal*, *Greater* or *Greater or equal*.

Evaluate expression

True if the given expression is a non-zero number, or a non-empty string. This is useful with boolean expressions like `score < 0 | health < 0`.

Every tick

A condition which is always true. Used on its own, this has the same effect as running every time it is checked, which is once per tick, hence the name "Every tick". This is about 60 times a second on most devices; see how events work for more information. Adding *Every tick* to an event with other conditions is redundant and has no effect at all.

Is between values

Test if a number is between two values (greater or equal to a lower value and less or equal to a higher value).

Is group active

Test if a group of events is active or inactive. The name of the group is used to identify it.

Is number NaN

Test if a number is equal to *NaN* (Not A Number), a special value returned by calculations which cannot be represented as a real number, such as the square root of -1.

Is value type

Check if a value is a number or a string.

Object UID exists

Test if an object exists with the given Unique ID (UID). For more information on UIDs, see instances.

Test regex

Test if a given string matches a regular expression with flags. This only returns a true or false result, so to make more advanced use of regular expressions, see the *Regex...* system expressions.

Global & local variables

Compare variable

Compare the value of a *number* or *text* type event variable (a global variable or local variable in scope). The comparison can be made *Equal*, *Not equal*, *Less*, *Less or equal*, *Greater* or *Greater or equal*.

Is boolean set

Test if a boolean event variable is set to *true*. To test if it is false, invert the condition.

Layers

Compare opacity

Compare the opacity (or semitransparency) of a layer, from 0 (transparent) to 100 (opaque). A layer's opacity cannot be outside this range.

Layer is empty

Test if a layer currently has zero instances on it. This counts any objects anywhere at all on the layer, so even one instance far outside the viewport will make this condition false.

Layer is HTML

Test if a layer is currently set to be a HTML layer. For more information see HTML layers.

Layer is interactive

Test if a layer is currently interactive, allowing its content to respond to mouse and touch input. This can be changed with the *Set layer interactive* action.

Layer is visible

Test if a layer is currently set to be visible.

> *This check includes parent layers: if any parent layer is invisible, then its sub-layers count as invisible too.*

Layer name exists

Test if the current layout includes a layer with the given name (case insensitive). This can be useful when using dynamic layers (i.e. the *Add layer* and *Remove layer* actions).

Layout

On canvas snapshot

Triggered after the *Snapshot canvas* system action, when the snapshot is ready. It can then be accessed with the *CanvasSnapshot* system expression.

Loops

Loops can be stopped with the *Stop Loop* system action.

For

Repeat the event a number of times, using an index variable over a range of values. The index can be retrieved with the *LoopIndex* system expression and passing the name of the loop.

For Each

For Each (ordered)

Repeat the event once per picked instance. This only repeats for instances that have been picked by prior conditions. See how events work for more information on picking. *For Each* is commonly mis-used or used redundantly - actions already apply *for each* instance picked by conditions, so it often is simply not needed. However, if you fully understand how the event system works, it can be used to force the event to apply once per instance where the event system would not normally do that. The 'ordered' variant allows the order that the instances are iterated in to be defined by an expression. For example, ordering by `Sprite.Y` ascending will iterate the top instances on the screen first, moving downwards.

Repeat

Simply repeat the event a given number of times. This tests any conditions following it on every repeat, and if those conditions are met also runs the actions and any sub-events on every repeat.

While

Repeat the event until one of the other following conditions in the event becomes false or a *Stop loop* action is used. Be careful not to create infinite loops which will cause the project to hang.

Memory management

Is loading images

True while any of the memory management 'Load' actions are in the process of loading images.

On image loading complete

Triggered when all memory management 'Load' actions that were started have finished loading their images.

Pick instances

Pick all

Reset the picked objects back to all of them. Subsequent conditions will pick from all instances again instead of filtering from only those meeting all the conditions so far. See How events work for more information on how instances are picked in events. Useful in subevents to start affecting different instances again.

Pick by comparison

Pick the individual instances of an object type that meet a comparison. For example, it is possible to pick all instances where `Object.X * 2` is less than `Object.Y + 100`, which is not possible with either the *Compare X* or *Compare Y* conditions.

Pick by evaluate

Pick the individual instances of an object type where the expression evaluates to a nonzero value. In other words, for each instance if the expression is 0, it is not picked, else it is picked. This is most useful with the comparison and logical operators (see Expressions). For example, it's possible to pick instances using the following expression (where & means "and" and | means "or"): `(Object.X > 100 & Object.Y > 100) | (Object.X < -100 & Object.Y < -100)`

Pick by highest/lowest

Pick a single instance of a given object type for which the given expression evaluates to the highest or lowest number. This evaluates the provided expression repeatedly for each individual instance of the object type, and uses the resulting values to determine which instance had the highest or lowest result, and then picks that instance. For example picking the instance with the highest expression `Object.X` will pick the single instance furthest to the right. Where there is a tie, an arbitrary single instance is chosen - it will never pick more than once instance.

Pick last created

Pick the most recently created instance of an object type or family. This is useful with the *Create object (by name)* system action. For example if you know the created object must belong to a family, then you can use *Pick last created* to pick the created instance from the family.

Pick Nth instance

Pick the instance at a given place in the internal list of picked objects. This is most useful used in sub-events to act on separate instances. For example, in a "Sprite collided with Sprite" event, *Pick 0th instance* and *Pick 1st instance* can be used to act on each instance involved in the collision separately.

If all objects are currently picked, this condition can also be used to pick an object by its index ID (IID).

Pick overlapping point

Pick all instances of a given object type that are overlapping a point in the layout. The given X and Y position in the layout will be tested against the instance's collision polygons.

Pick random instance

Pick a random instance from the currently picked objects. In other words, if *Pick random instance* follows another condition, it will pick a random instance from the instances meeting the prior condition. Otherwise it picks a random instance from all the instances.

Save & Load

For more information on using savegames, see the tutorial How to make savegames.

On load complete

Triggered after the *Load* system action successfully completes.

On load failed

Triggered after the *Load* system action fails to complete, usually because the slot has not been saved to yet.

On save complete

Triggered after the *Save* or *Save to JSON* system actions successfully complete. In both cases, a string of the resulting JSON data can be accessed in this trigger with the *SaveStateJSON* system expression.

On save failed

Triggered if the *Save* system action fails to complete. This can occur if the browser has reached the limit of its storage quota, or the user has withdrawn permission for the page or app to write to storage.

Special conditions

Else

Run if the previous event did not run. Note that this condition does not pick any objects: if it follows an event that picks objects, in the *Else* event all instances revert to picked again. *Else* can only follow normal (non-triggered) events. It can also follow another *Else* event with other conditions to make an "if - else if - else" chain.

Is in preview

True when running the project from a preview in Construct, and false when running after being exported. Useful to add debug or diagnostic features for previewing only.

Trigger once while true

Turn an ordinary event (which is tested every tick) in to a trigger. For example, if an event plays a sound when lives equals 0, normally this event runs every tick. This plays about 60 sounds a second and would sound pretty bad. Adding *Trigger once while true* **after** the other conditions makes the event run just once when it first becomes true. This makes the previous example only play a sound once the first time your lives reaches 0. It should be the last condition in an event.

Start & end

Is suspended

True if the runtime is currently suspended, i.e. in between *On suspended* and *On resumed*. This normally means the browser/app is in the background and is not currently running.

On suspended

On resumed

Triggered when the browser/app suspends and resumes execution. Normally when the app goes in to the background (e.g. minimized, or switched back to the home screen), execution of the app is suspended to conserve system resources and save battery power, triggering *On suspended*. When the app is reopened, *On resumed* triggers and execution resumes.

On end of layout

Triggered when the layout is ending. This can happen when the project goes to a different layout or when the project closes.

On loader layout complete

Triggered on a loader layout when the progress reaches 100%. For more information see How to use loader layouts to make custom loading screens.

On start of layout

Triggered when the layout begins. Use this event to run any actions that need to be done on start-up.

Time

Compare time

Compare the time, in seconds, since the project started. For example, events can be set up to run when the time reaches (equals) 10 seconds.

Every X seconds

Run the event regularly at a given time interval in seconds. This can also be used beneath other conditions to only run the event at a given time interval while the other conditions are true, e.g. "Player is holding spacebar AND every 0.5 seconds: fire laser".

> *Every X seconds does not run more than once per tick. This means very short time periods may not work like you expect. For example many displays use a 60 Hz refresh rate, in which case Every 0.01 seconds will run 60 times a second, not 100 times a second. To avoid such unexpected results, don't use this condition with short time periods - prefer to just use Every tick instead, and if necessary check delta-time (dt) to see how much time passed in the last tick.*

On signal

Triggered when the *Signal* system action is run with a matching tag (case insensitive).

Templates

Template exists

Check if an object type has a template with the provided name.