LAYOUT SCRIPT INTERFACE

View online: https://www.construct.net/en/make-games/manuals/construct-3/scripting/scripting-reference/layout-interfaces/ilayout

The `ILayout` script interface represents a layout in the project.

Getting an ILayout

The `ILayout` interface is typically accessed via the IRuntime `layout` property, e.g. `runtime.layout`. This represents the current running layout. Other layouts can be accessed via the IRuntime methods `getLayout()` and `getAllLayouts()`.

Layout events

The following events can be listened for using the `addEventListener` method.

"beforelayoutstart"

"afterlayoutstart"

Fired when the layout starts. `"beforelayoutstart"` fires just before *On start of layout*, and `"afterlayoutstart"` fires just after. In both events, all instances on the layout are created and available to modify.

> *These events can use async handler functions, and the runtime will wait for them to finish before continuing.*

"beforelayoutend"

"afterlayoutend"

Fired when the layout ends due to changing to another layout. `"beforelayoutend"` fires just before *On end of layout*, and `"afterlayoutend"` fires just after. In both events, all instances on the layout are still available, but all non-global instances are destroyed immediately after the `"afterlayoutend"` event.

> *These events can use async handler functions, and the runtime will wait for them to finish before continuing.*

Layout APIs

runtime

A reference back to the IRuntime interface.

---

name

A read-only string of the layout name.

---

index

A read-only number of the zero-based index of the layout in the order it appears in the Project Bar.

---

addEventListener(eventName, callback)

removeEventListener(eventName, callback)

Add or remove a callback function for an event. See *Layout events* above for the available events.

---

width

height

setSize(width, height)

getSize()

Set or get the size of the layout. The methods allow setting and getting both values at the same time.

> *Note a layout cannot have a zero or negative size.*

---

scrollX

scrollY

scrollTo(x, y)

getScrollPosition()

Set or get the scroll position in layout co-ordinates. `scrollTo()` is a shorthand for setting both `scrollX` and `scrollY`, and `getScrollPosition()` returns both scroll co-ordinates at the same time.

---

isUnboundedScrolling

A read-only boolean reflecting the state of the *Unbounded scrolling* layout property.

---

scale

Set or get the layout scale, with `1` being the default scale, `2` being 2x scale, etc. This scales all the layers in the layout, taking in to account their scale rate property.

---

angle

Set the layout angle in radians. This rotates all the layers in the layout.

---

projection

Set or get a string specifying the current layout projection, which must be one of `"perspective"` or `"orthographic"`. For more details see *Projection* in Layout Properties.

---

setVanishingPoint(vpX, vpY)

getVanishingPoint()

Set or get the *Vanishing point* layout property, with each component in the range 0-1. The getter returns an array with two elements in the form `[vpX, vpY]`.

---

effects

An array of IEffectInstance representing the effect parameters of the effects on this layout.

Layer APIs

These APIs relate to the set of layers on the layout.

---

getLayer(layerNameOrIndex)

Get an ILayer interface for a layer on the layout, by a case-insensitive string of its name or its zero-based index. When passing a number, an out-of-range number is clamped to the valid range and the nearest layer returned. When passing a string, if no layer with the given name is found, the method returns `null`.

***allLayers()**

Iterates ILayer interfaces representing all the layers on the layout, in increasing Z order.

---

getAllLayers()

Return an array of ILayer interfaces representing all the layers on the layout, in increasing Z order.

---

addLayer(layerName, insertBy, where)

Create a new layer and insert it to the layer tree at runtime (also known as a *dynamic layer*). `layerName` is a string of the name to use for the added layer, which must be different to all

existing layers already added, including other dynamic layers. `insertBy` is an ILayer of another layer to insert the new layer relative to. `where` specifies where to insert the new layer relative to the `insertBy` layer, which may be one of the following strings:

- `"above"` or `"below"`: insert adjacent to the `insertBy` layer, above or below it in Z order.
- `"top-sublayer"` or `"bottom-sublayer"`: insert as a sub-layer of the `insertBy`, at the top or bottom of its existing sub-layers. `insertBy` may also be `null`, in which case the new layer is added at the top or bottom at the root level of the layer tree.

---

moveLayer(layerToMove, insertBy, where)

Remove and re-insert a layer to a new location in the layer tree. This works similarly to `addLayer()`, except `layerToMove` refers to an ILayer that already exists; otherwise the `insertBy` and `where` parameters are used in the same way.

---

removeLayer(layer)

Remove a given ILayer from the layer tree. This also removes any sub-layers of the removed layer, and all objects on the layer or any of its sub-layers will be destroyed. A layout must have at least one layer, so the last top-level layer cannot be removed.

---

removeAllDynamicLayers()

Removes all layers added using the `addLayer()` method, leaving only the layers added in the editor. All objects on the removed layers will be destroyed. This can be useful to reset the state of dynamic layers.