

# LAYERS

**View online:** <https://www.construct.net/en/make-games/manuals/construct-3/project-primitives/layers>

---

A **layer** is like a transparent sheet of glass that objects are painted on to. Layers can be used to show different groups of objects in front or behind each other, like the foreground and background. Layers belong to a **layout** and can be added, edited and removed in the **Layers Bar**. Layers can be scrolled at different rates for parallax effects, and also individually scaled and rotated, which makes them a powerful way to make interesting visual effects.

Layers are also important to add non-scrolling content (e.g. HUDs or UIs) to scrolling projects. A layer with its parallax set to **0, 0** will not scroll at all, so any objects placed on this layer will always stay in the same place on-screen. Note that in this case, objects should be placed within the dashed rectangle that appears in the top-left of the **Layout View**.

A common arrangement for layers might be:

- **HUD** (top layer - health bar, UI info etc.)
- **Foreground** (objects appearing on top, e.g. explosions and effects)
- **Middleground** (main game objects such as the player and enemies)
- **Background** (bottom layer - the background)

Note that the Free edition is limited to using two layers only.

Layers can also have **effects** applied, which affects all content appearing on the layer.

## Sub-layers

Layers can also be added as sub-layers of another layer. Sub-layers appear indented in the **Layers Bar** to show they come under another layer.

A layer with both objects and sub-layers will show its objects on top of its sub-layers. In other words, sub-layers come beneath a layer's own objects in the Z order. This also means that sub-layers act a lot like a simple flat list of layers, and so can be used solely for organizing long layer lists, much like layer folders.

However adding an effect to a layer with sub-layers allows for more efficient and more advanced effects. An effect on a layer with sub-layers will alter the appearance of both the layer and all its sub-layers. This is more efficient than adding the same effect to multiple layers, as it ensures the effect is only processed a single time, while affecting the content of multiple layers.

Layer effects involving sub-layers also allow for composition of more advanced effects. For example a group of layers can be combined to make a single lighting layer, which then affects

the appearance of another group of layers beneath it. See the [Shadows: blending multiple lights](#) example for a demonstration of this technique.

## Global layers

Sometimes many layouts in a project have the same content on a particular layer, such as for interface or HUD overlaid on to the project. Changing this content then becomes a chore since changes must be repeated on every layout. **Global layers** are aimed at solving this problem.

If a layer's *Global* property is enabled, then **every layer in the project with the same name** is overridden by that layer. The initial objects, as well as its properties, are used instead of the other layer's own content and properties. Then changes can be made once to the original global layer, and the changes will be applied project-wide.

The layer with the *Global* property enabled is the "master" layer. On other layers in the project with the same name, the *Global* property will be read-only and display *Overridden* to indicate it is being substituted by a different layer. The same layer's content will appear in the editor, and all edits will affect the master layer, no matter which layout it is being edited from.

Whether a layer is the original global layer or is overridden will be shown next to a layer's name in between parenthesis in all relevant places, these includes the *Layers* dropdown in the Properties bar when an [instance](#) is selected and next to each item of the [Layers Bar](#).

## Layer properties

The properties for a layer can be edited in the [Properties Bar](#) after clicking the layer in the [Layers Bar](#). Note this also changes the **active layer**.

### Name

The name of the layer, which can be used to refer to the layer in the event system.

### Initially visible

Whether or not the layer is initially visible **when previewing**. This is different to the *Visible in editor* property which only affects the Layout View.

### Initially interactive

Whether or not the layer is initially interactive when previewing. If disabled, then the content of the layer will not respond to mouse or touch input.

### HTML elements layer

Allow HTML elements to appear above this layer. This allows content on other layers above this layer to render on top of HTML elements on this layer. Layers which enable this are shown with a special icon in the Layers Bar. For more details see [HTML layers](#).

## Use render cells

Optimise the rendering of this layer for extremely large layouts with a large number of static objects spread out across this layer. This is not normally necessary except for certain types of large projects. If this is used incorrectly, it can actually make rendering less efficient, so make sure you can measure a performance improvement before using it. For more information, see the blog post [How render cells work](#).

## Scale rate

Change the rate at which the layer zooms if scaling is applied to the layer or layout, a bit like parallax but for zoom. A scale rate of 0 means the layer will always stay at 100% scale regardless of the scaling applied. A scale rate of 100 means it will scale normally.

## Parallax

Change the rate at which the layer scrolls in the horizontal and vertical directions. A parallax rate of  $100\% \times 100\%$  means ordinary scrolling,  $0\% \times 0\%$  means it will never scroll (useful for UIs),  $50\% \times 50\%$  means scrolling half as fast, etc. Also useful for multi-layer parallaxing backgrounds.

## Z elevation

The Z elevation of the entire layer. By default the camera is at Z = 100, and looking down to Z = 0. The default Z elevation is 0. Increasing it will move the layer upwards (towards the camera) and decreasing it will move it downwards (away from the camera). You can learn more about Z elevation in the tutorial [Using 3D features in Construct](#).

## Transparent

Make the layer have a transparent background. If enabled, the background color is not used.

## Background color

The background color for the layer, if it is opaque (i.e. *Transparent* is disabled).

## Opacity

Set the opacity (or semitransparency) of the layer, from 0% (invisible) to 100% (opaque).

## Force own texture

Force the layer to always render to an intermediate texture rather than directly to the screen. This is useful for some kinds of effects. However it slows down rendering, so it should be disabled unless specifically needed.

## Uses own texture

A read-only property indicating if the layer renders to an intermediate texture. This has a performance overhead. The *Force own texture* setting enables this, but some other properties also cause the layer to use its own texture, including changing the layer opacity from 100%, changing the blend mode, or adding effects.

## Rendering mode

When using 3D rendering mode, this setting can change a layer back in to rendering in 2D mode. This allows projects with 3D content to still use 2D layers as backdrops or overlays which are not affected by depth. For example a HUD layer ought to display on top of all 3D content, regardless of depth, so would typically use a 2D rendering mode for the layer. Otherwise in 3D mode, 3D features may still overlap the layer content if they rise higher than the layer contents. For an example, see [Combining 2D & 3D layers](#), and you can learn more about 3D features and 2D layers in the tutorial [Using 3D features in Construct](#).

*This property only appears for projects using 3D rendering mode. See the [Rendering mode project property](#).*

## Draw order

This setting only appears for layers using a 3D rendering mode. The default draw order is *Z order*, meaning objects are drawn in a back-to-front order according to the Z order of instances on the layer. 3D layers can also be set to *Camera distance* draw order, which instead ignores the Z order and draws instances on the layer according to how far away from the camera they are, from furthest away to nearest. This has no effect on opaque objects, but is important for rendering transparency in 3D. For more information see the tutorial [Using 3D in Construct](#).

## Blend mode

Change the way the layer is blended with the background when it is rendered to the display. See the *Blend modes* example that comes with Construct 3 for a visual demonstration of each.

## Effects

Add and edit [effects](#) that apply to the whole layer.

## Visible in editor

Whether or not the layer is showing **in the Layout View**. Note this is different to the *Initially visible* property which only affects previewing. This setting can also be accessed via the Layers Bar.

## Locked

Whether or not the layer is locked in the Layout View. Objects on locked layers cannot be selected. This setting can also be accessed via the Layers Bar.

---

## Parallax in editor

If enabled, the *Parallax* property will also be applied in the Layout View, allowing you to preview what the effect will look like.

---

## Global

See the section above on *Global layers*. If enabled it will override every other layer in the project with the same name with its own contents and properties. Overridden layers display this property read-only as *Overridden*. If disabled its contents and properties are unique to itself.