HTML LAYERS

View online: https://www.construct.net/en/make-games/manuals/construct-3/tips-and-guides/html-layers

Construct supports integrating HTML content with your project. This includes the HTML Element object, as well as other plugins in the *HTML elements* category, such as Button and Text Input. Layering HTML objects works differently to other kinds of objects. In particular, to have other Construct objects appear on top of HTML objects, it is necessary to use **HTML layers**. This guide explains how HTML layers work.

Using HTML layers is useful for better integration of HTML content in your project. For example if you want to use HTML for part of your project's user interface, but then have some decoration like a Particle effect appear on top of the user interface, then it is necessary to use HTML layers to get the Particle effect to appear on top of the HTML content.

Canvas vs. HTML objects

Most Construct objects, such as Sprite, Tiled Background and Particles, render in to a `<canvas>` element. These kinds of objects are collectively referred to as *canvas objects*. The canvas element is a single HTML element that effectively acts as a large image that changes every frame. Much like the usual `<img>` (image) element, it can only be placed entirely in front or behind other HTML elements - there is no way for it to appear both partly in front and partly behind another HTML element.

HTML objects like Button and Text Input are themselves represented by other HTML elements (`<button>` and `<input>` respectively). These kinds of objects are collectively referred to as *HTML objects*. They are not drawn in to the canvas. Therefore they can only appear in front of, or behind, a canvas element.

By default Construct creates a single canvas element, and all other HTML elements are placed on top of the canvas. For example if a project uses a Sprite and a Button object, Construct will use the following stack of HTML elements (in top-to-bottom order):

- `<button>` for the Button object (on top)
- `<canvas>` for canvas objects like Sprite (underneath)

With this arrangement, it is not possible for the Sprite object to appear on top of the Button object, even if the Sprite is on a layer above the Button object, or if it is on top in Z order on the same layer. In other words, the HTML layering takes precedence over Construct's Z order.

HTML layers

It is possible to have canvas objects render on top of HTML objects by making use of Construct's *HTML layers* feature. Checking the *HTML elements layer* property of a layer will turn

that layer in to a HTML layer, and indicate this status in the Layers Bar with a special tag icon. Then content on other layers above it will appear on top of HTML objects on that layer.

This works by creating an additional `<canvas>` element per HTML layer. For example consider the following arrangement of layers:

- **Layer 1** with a Sprite object
- **Layer 0** with a Button object

Normally, the Button will appear on top of the Sprite object even though it is layered beneath it, due to the reason described above. However if *Layer 0* is made a HTML layer, it is now possible for the Sprite to appear on top of the Button. This is because Construct now creates the following HTML elements (in top-to-bottom order):

- `<canvas>` for Layer 1, with a Sprite object
- `<button>` for the Button object
- `<canvas>` for Layer 0, with any other canvas objects

Note that HTML objects still appear on top of any other canvas objects on the same layer. However canvas objects on all layers above a HTML layer can then appear on top of the HTML objects.

If multiple HTML objects are on the same HTML layer, then they are ordered in between the two canvas elements, respecting their own relative Z order. In actual fact Construct uses a `<div>` element wrapper to contain all HTML content in between canvases. This also ensures HTML content is clipped to the canvas area, so they cut off when moving to the edge of the screen like canvas objects.

The implicit top HTML layer

The top layer is always implicitly a HTML layer. In other words, if you have no HTML layers, then HTML objects appear on top of all canvas objects, as all HTML elements are placed on top of the `<canvas>` element by default, as illustrated in the first example in this guide.

Multiple layers

HTML objects on non-HTML layers will visually appear on the next HTML layer above them in Z order. To illustrate this, consider this arrangement of layers.

- Layer 7 (normal layer)
- Layer 6 (normal layer)
- Layer 5 (HTML layer)
- Layer 4 (normal layer)
- Layer 3 (normal layer)

- Layer 2 (HTML layer)
- Layer 1 (normal layer)
- Layer 0 (normal layer)

This arrangement uses two HTML layers on Layer 2 and Layer 5. In this case, all HTML objects on Layer 0, Layer 1 and Layer 2 will appear on Layer 2 (above all other canvas objects on layer 2); all HTML objects on Layer 3, Layer 4 and Layer 5 will appear on Layer 5; and all HTML objects on Layer 6 and Layer 7 will appear on top of those, as there is an implicit HTML layer at the top. This is because Construct creates the following HTML elements (in top-to-bottom order):

- HTML elements for layers 6-7
- `<canvas>` for canvas objects on layers 6-7
- HTML elements for layers 3-5
- `<canvas>` for canvas objects on layers 3-5
- HTML elements for layers 0-2
- `<canvas>` for canvas objects on layers 0-2

In this case, layers 0-2, layers 3-5, and layers 6-7 can each be thought of as separate HTML layers. So in this case there are three HTML layers, as there are three options for where HTML objects can appear relative to canvas objects, whereas there are eight canvas layers as there are eight options for where canvas objects can appear (which are distributed across three canvas elements).

Note that HTML objects always appear on top of all canvas objects on the same HTML layer. A HTML layer can be thought of as a canvas element with all canvas objects paired with a layer of HTML objects above it. Within a HTML layer, HTML objects will respect their relative Z order. For example if a Button is layered on top of a Text Input in the same HTML layer, then Construct will ensure the Button object's HTML element is layered on top of the Text Input's HTML element.

Only top-level layers can be made HTML layers. With this method of stacking canvas elements and other HTML elements, it is not possible to support making sub-layers HTML layers.

Effects

When using effects, note they can only process content on the same canvas. This usually affects background-blending effects. Taking the previous example of multiple layers, if there was a background-blending effect on layer 5, then it will only be able to blend with content on layers 3-5. Layers below that are on a different HTML layer and so render to a different canvas, and thus the background blending effect is not able to process them. This is a side-effect of the way the rendering composition works: as each canvas is rendered separately, it is not possible for effects to work across them.

Performance overhead

As illustrated, for each layer which you make a HTML layer, an additional canvas element is created at the size of the viewport. This comes with a performance overhead. It will impact two aspects of performance:

1. Each canvas must be copied to the display every frame. This has the effect of drawing a viewport-sized surface, much like a *Force own texture* layer. This uses up the GPU fill rate (see Performance Tips for more details about fill rate). In some cases the compositing process may require more than one copy, making it potentially two or three times as costly as a *Force own texture* layer.
2. Each canvas must be allocated in memory, which will use at least as much memory as a viewport-sized image. For example a 1920x1080 size canvas will require at least 8 MB of memory (see Memory usage for more details). In some cases the compositing process may require more than one surface, making it potentially use two or three times as much memory.

For this reason, avoid using too many HTML layers. Try to use the minimum necessary number of HTML layers to achieve the layering you need for your project.

Positioning

HTML objects use a size and position based on the Construct layer they are on. In other words they match the size and position that they would have if they were a canvas object. Therefore if HTML objects are on layers with different scroll positions, scales, or parallax, they will match the position of the layer they are on, even though they are displayed on the next HTML layer above them.

Browsers do not always update canvas elements and other HTML elements at the same time. You may find that if you move both canvas objects and HTML objects simultaneously at high speed then a visible difference appears between them as one lags behind the other by a frame or two, due to the browser drawing them at different times.

Dynamic layers

It's possible to dynamically add or change HTML layers using the *Set layer HTML* action or `isHTMLElementsLayer` script property. This will cause Construct to add or remove canvas elements, and rearrange the layering of HTML elements, to reflect the changes. However this is complicated to synchronize, especially in worker mode where changes cannot be made synchronously. This may result in a brief flicker as HTML changes are made. This may be able to be avoided by ensuring sufficient HTML layers are created in advance of filling them with content, or using techniques such as fading in content so it initially starts invisible and so a flicker is not noticable.