

Canvas3D API Use Cases

The point of this document is to record any and all realistic use cases for the Canvas3D API (also known as C3DL), which is being developed at Seneca College.

Other ideas are certainly welcome. Get in touch with one of us if there's something you would like to be able to do on the web that could be done using Canvas3D.

2D ought not be overlooked. The web is quite a useful thing, even though it's almost entirely two-dimensional.

Shapes and Movement

Create simple 2D and 3D shapes

Using the API one ought to be able to create simple shapes such as a rectangle, oval, a cuboid, and a sphere. Perhaps also triangles and pyramids, though writing coordinates for those manually is not realistic.

Load model from a file

Give the API a URL of a file that describes a shape, including colours or textures. The format should be exportable from 3D Studio Max and Blender at least.

If it comes to that, we can provide plugins (separately from the extension and the API) to export models into this format.

In the rest of this document 'shape' includes models as well as simple shapes.

Text

The canvas must be able to deal with text. Not necessarily all kinds of text one can have in HTML, but at least some basic options for size, font, and colour should be available.

The text can be rendered onto a bitmap, which can be used on a shape, to make it easier to implement things like moving around.

Create a straight line for a shape to move on

The API will know to move the shape from one point to another. Ought to have an option of using either an absolute path (move from point A to point B) or a relative path (move this far in this direction).

If the API for creating the path turns out to be pretty complicated because of 3 dimensions, a 2D wrapper should be made as well.

Create a curved line for a shape to move on

There are formulas with only 1 or 2 variables that will give you a curved path. The API ought to accept: a constant identifying the formula + the needed variables + a destination point, so that it will know how to move a shape to a new point given a trajectory.

This can probably be made to work with 3 dimensional curves, but I don't be the one figuring out the math for that.

Append a path to another

The API should allow to easily concatenate path A and path B, so more complicated paths can be created without using ridiculously complicated formulas.

Morph a shape into another

Given either (2 shapes) or (1 shape and a formula), the API ought to be able to turn shape A into shape B. There will almost certainly be a lot of complicated math to implement this feature, but it's something content creators will expect to be able to do.

Chameleon

Give the API a colour, and have it apply the colour to a shape over time, with some reasonable transition in between (should be easy with simple RGB arithmetic).

If the shape has a texture rather than a colour to begin with, the transition would probably be too hard to implement and too CPU-intensive to bother.

Rotate a shape

The API should be able to rotate a shape in any direction. Either n degrees, or perpetually.

Combine operations

The API should be able to do more than one of the above operations at the same time. E.g. move, morph, and change colour at the same time.

Time & Speed

Many operations take time, for example moving. Eventually we'll want a timeline but for now, I think, all the events will be triggered by the user so we just need some way to specify how long they'll take (e.g. move this shape to this point in 5 seconds starting now).

Interaction

Pre-rendered animations is not what Canvas3D is about. The user ought to have the control over Canvas3D content as they do over HTML content.

Sadly the canvas is a black box by design, so we have to emulate, rather than reuse. Note: I could be

wrong about this part.

Mouse events

Using the API one ought to be able to listen to mousedown, mouseup, click, double click, mouse move, mouse over, and mouse out events on shapes in the canvas.

Keyboard events

Preferably the API would allow to listen to keyboard events in two modes: canvas only and global.

In canvas only mode any shapes that are links or are specifically designated become 'tabbable'. From the user's point of view they're inserted into the normal page tab order. We don't actually want to do that (too much integration with the browser's code would be needed). We want to hook into that tab order; when we get ours - prevent the normal tab order from working and do it ourselves inside the canvas; when the list of tabbable elements in the canvas is exhausted - allow the normal tab order to work again.

In global mode it would get keyboard events no matter what element on the page has the keyboard focus. This mode is less important, but it may be a prerequisite to implementing the canvas only mode.

Links

The API ought to allow any shape to be a link. A link can have blank, self, parent, and top targets. This can be done by the library user manually using mouse events and regular javascript, but it's an important use case, so it should have be in the core library.