Spinning Diagrams with CSS

Harold Cooper

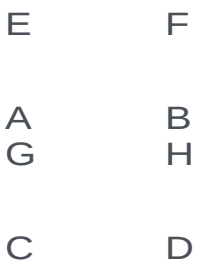
20th April 2023

I wrote a little math thing last year, which featured equations like this:

Several people expressed surprise that the spinning diagrams don't use any JavaScript or animated image formats, just HTML and CSS. So I thought I'd explain how they work before I forget.

A spinning cube

We can build a spinning cube, with a letter at each vertex.



For the HTML, we make a div for each letter and position it with `translate3d`:

```
<div id="cube" style="width: 4em; height: 8em;">
  <div style="transform: translate3d(0em, 0em, 2em)">A</div>
  <div style="transform: translate3d(4em, 0em, 2em)">B</div>
```

```
    …
    <div style="transform: translate3d(0em, 4em, -2em)">G</div>
    <div style="transform: translate3d(4em, 4em, -2em)">H</div>
  </div>
```

(I use `em` units, but `px` or any other unit is fine too. The cube is 4em wide and the vertices are centered around x=2em and z=0em, making it easy to spin about the center.)

For the CSS, we set an `animation` on the parent from `rotateY(0turn)` to `rotateY(1turn)`:

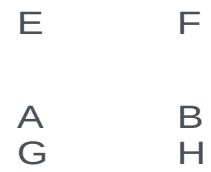
```
#cube {
  position: relative;
  transform-style: preserve-3d;
  animation: spin 20s linear infinite;
}

#cube > div {
  position: absolute;
  transform-style: preserve-3d;
}

@keyframes spin {
  from { transform: rotateX(-0.1turn) rotateY(0turn); }
  to { transform: rotateX(-0.1turn) rotateY(1turn); }
}
```

(Note that `1turn` equals `360deg`. And we add a slight tilt `rotateX(-0.1turn)` to make things look better. Finally, for the 3d positions to work, we need `preserve-3d` and `position: relative` on the parent, and `position: absolute` on the children.)

Put it all together and we get:

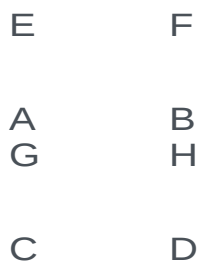


C D

Notice that the letter glyphs themselves are rotating, which is neat, but could make the diagram hard to read.

Un-spinning the letters

To keep the letters facing forwards, we can ‘un-spin’ them in sync with the spinning parent, but in the opposite direction.



To accomplish this we add another div around each letter, where we can perform the un-spinning without interfering with the existing `transform`:

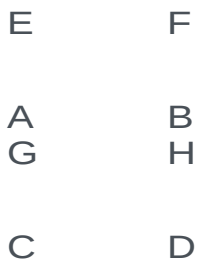
```
<div id="cube" style="width: 4em; height: 8em;">
  <div style="transform: translate3d(0em, 0em, 2em)"><div>A</di
  <div style="transform: translate3d(4em, 0em, 2em)"><div>B</di
  …
  <div style="transform: translate3d(0em, 4em, -2em)"><div>G</d
  <div style="transform: translate3d(4em, 4em, -2em)"><div>H</d
</div>
```

We keep the CSS from before, but give the new inner divs an un-spinning `animation` from `rotateY(0turn)` to `rotateY(-1turn)`:

```
#cube > div > div {
  animation: un-spin 20s linear infinite;
}
```

```
@keyframes un-spin {
  from { transform: rotateY(0turn); }
  to { transform: rotateY(-1turn); }
}
```

All together this looks like:



I was pleasantly surprised that all this spinning and un-spinning seems to perform fine even on mobile browsers.

You can even select the rotating text and your selection will rotate as well—impressive work by the browser builders.

The original math thing involved a few other tricks—to embed the diagrams in LaTeX and generate their geometries—which for completeness I'll describe as a footnote.[1]

1. The source of the original math thing is a Markdown/LaTeX file.

   Each diagram is embedded in the LaTeX as a numeric ID, which is then replaced with generated HTML by a Python script. (The IDs are all at the bottom of the script, after a long mess of typesetting hacks and NumPy geometry.)

   The Python script is run after the Markdown and LaTeX have been rendered because it's specified with `postprocess` in the Markdown front matter, which triggers a `post_convert` Jekyll hook thanks to a bespoke Jekyll plugin.

Lastly, the relevant CSS will look familiar if you've read this far. ↩