**Carousel[[1]](#footnote-1)**

Front-end developers have a myriad of choices when it comes to content carousels. [May I suggest Flickity?](https://flickity.metafizzy.co/) Now that we have 3D capabilities in our browsers, why not give a shot at creating an actual 3D carousel?

The markup for this demo takes the same form as the box, cube, and card. Let’s make it interesting and have a carousel with 9 panels.

<div class="scene">

<div class="carousel">

<div class="carousel\_\_cell">1</div>

<div class="carousel\_\_cell">2</div>

<div class="carousel\_\_cell">3</div>

<div class="carousel\_\_cell">4</div>

<div class="carousel\_\_cell">5</div>

<div class="carousel\_\_cell">6</div>

<div class="carousel\_\_cell">7</div>

<div class="carousel\_\_cell">8</div>

<div class="carousel\_\_cell">9</div>

</div>

</div>

|  |
| --- |
|  |

Now apply basic layout styles. Let’s give each cell 20px gaps between one another, done here with left: 10px. The effective width of each panel remains 210px.

.scene {

width: 210px;

height: 140px;

position: relative;

perspective: 1000px;

}

.carousel {

width: 100%;

height: 100%;

position: absolute;

transform-style: preserve-3d;

}

.carousel\_\_cell {

position: absolute;

width: 190px;

height: 120px;

left: 10px;

top: 10px;

}

Next up: rotating the faces. This carousel has 9 cells. If each cell gets an equal distribution on the carousel, each panel would be rotated 40 degrees from the next ( 360 / 9 ).

.carousel\_\_cell:nth-child(1) { transform: rotateY( 0deg); }

.carousel\_\_cell:nth-child(2) { transform: rotateY( 40deg); }

.carousel\_\_cell:nth-child(3) { transform: rotateY( 80deg); }

.carousel\_\_cell:nth-child(4) { transform: rotateY(120deg); }

.carousel\_\_cell:nth-child(5) { transform: rotateY(160deg); }

.carousel\_\_cell:nth-child(6) { transform: rotateY(200deg); }

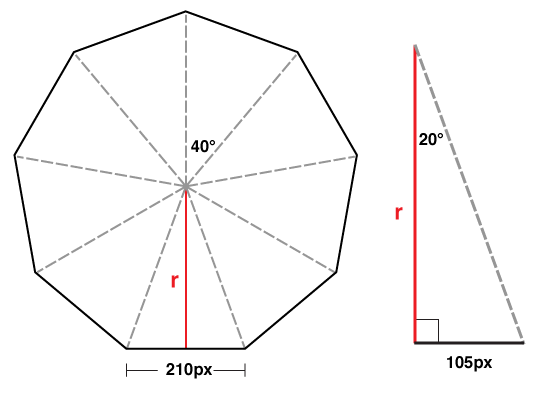
.carousel\_\_cell:nth-child(7) { transform: rotateY(240deg); }

.carousel\_\_cell:nth-child(8) { transform: rotateY(280deg); }

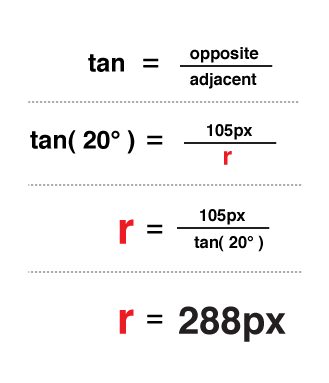
.carousel\_\_cell:nth-child(9) { transform: rotateY(320deg); }

Now the outward shift. Back when we were creating cube and boxes, the translate value was simple to calculate, as it was equal to one half the width, height, or depth of the object. Now with a carousel, there is no size we can immediately reference. We’ll have calculate the distance for the shift by other means.

Drawing out a diagram of the carousel, we see that we only know two things: the width of each panel is 210px and the each panel is rotated 40 degrees from the next. If we split one of these triangles down its center, we get a right triangle, prime for some trigonometry.



We can determine the length of r in this diagram with a basic tangent equation.



There you have it, 288px is the distance to translate the panels out in 3D space.

.carousel\_\_cell:nth-child(1) { transform: rotateY( 0deg) translateZ(288px); }

.carousel\_\_cell:nth-child(2) { transform: rotateY( 40deg) translateZ(288px); }

.carousel\_\_cell:nth-child(3) { transform: rotateY( 80deg) translateZ(288px); }

.carousel\_\_cell:nth-child(4) { transform: rotateY(120deg) translateZ(288px); }

.carousel\_\_cell:nth-child(5) { transform: rotateY(160deg) translateZ(288px); }

.carousel\_\_cell:nth-child(6) { transform: rotateY(200deg) translateZ(288px); }

.carousel\_\_cell:nth-child(7) { transform: rotateY(240deg) translateZ(288px); }

.carousel\_\_cell:nth-child(8) { transform: rotateY(280deg) translateZ(288px); }

.carousel\_\_cell:nth-child(9) { transform: rotateY(320deg) translateZ(288px); }

|  |
| --- |
|  |

If we decide on changing the width of the panel or the number of panels, we only need to plug in those two variables into our equation to get the appropriate translateZ value. In JS terms, that equation would be:

var tz = Math.round( ( cellSize / 2 ) /

Math.tan( ( ( Math.PI \* 2 ) / numberOfCells ) / 2 ) );

*// or simplified to*

var tz = Math.round( ( cellSize / 2 ) / Math.tan( Math.PI / numberOfCells ) );

Just like our previous 3D objects, to show any one panel, we need only to apply the reverse transform on the carousel.

*/\* show fifth cell \*/*

.carousel {

transform: translateZ(-288px) rotateY(-160deg);

}

|  |
| --- |
|  |

[Edit this demo on CodePen](https://codepen.io/desandro/pen/jxwELK)

## 3D Carousel with JavaScript

By now, you probably are thinking how re-writing transform styles for each panel is tedious. And you’re absolutely right. The repetitive nature of 3D objects lend themselves to scripting. We can offload all the monotonous transform styles to our JavaScript, which, if done right, will be more flexible than the hard-coded version.

|  |
| --- |
|  |

[Edit this demo on CodePen](https://codepen.io/desandro/pen/wjeBpp)

Not only can we change the number of cells, we can even change the orientation of the carousel from horizontal to vertical. Perfect for The Price is Right wheel.

1. <https://3dtransforms.desandro.com/carousel> [↑](#footnote-ref-1)