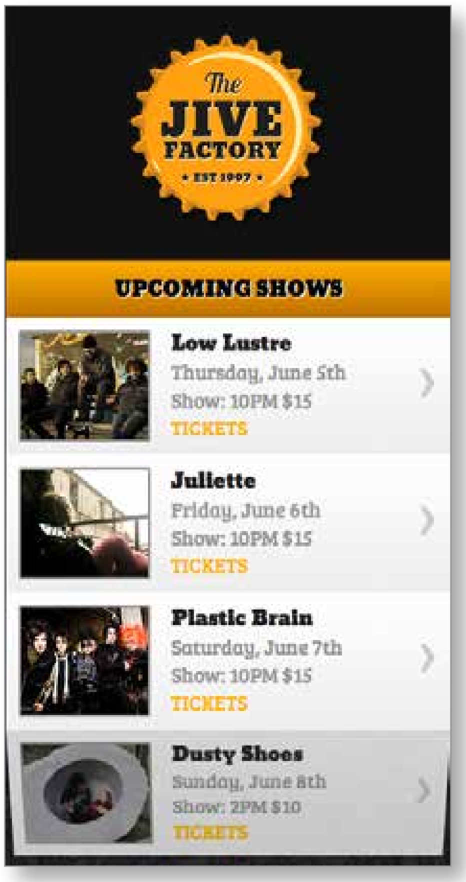
GSAP *Staggered Animation in 3D*

2D EXERCISE

EXERCISE PREVIEW



Photos courtesy of istockphoto: Hakan Çaglav, Image #14393929;
Renee Keith, Image ##7827478.

EXERCISE OVERVIEW

It's easy to create super slick effects with minimal code with TweenMax.
In this exercise we'll use the powerful staggerFrom() method to create a series of animations on multiple elements. We'll also dip our toes into some easy 3D tricks along the way.

PREVIEWING THE FINISHED PRODUCT

1. To take a look at the file we will be building, open up Google Chrome.
2. To open a file, hit **Cmd–O** (MAC) or **Ctrl–O** (WINDOWS).
3. Navigate to **Desktop > Class Files > yourname-GSAP Class > Staggered Animation.**
4. Double–click on **finished.html.** We'll be focusing on animating the elements that contain the information for each show. Notice how they all flip down in sequence with a nifty 3D effect.
5. Reload the page as many times as you need to get a feel for the animation.

EXAMINING THE DOM STRUCTURE AND JAVASCRIPT

1. In a text/code editor open **start.html** from the **Staggered Animation** folder.
2. Take a look at the code from lines 15–42 to familiarize yourself with the DOM. There are four elements that share a class of **show.** Each show is structured like the following show for Dusty Shoes:

```
<div class="show">
   <div class="pic"><img src="img/thumb-dusty-shoes.jpg"></div>
   <h3>Dusty Shoes</h3>
   <div>Sunday, June 8th</div>
   <div>Show: 2PM $10</div>
   <div class="button">Tickets</div>
</div>
```

   NOTE: We will not have to edit any of the CSS for this project but if you'd like to examine the linked CSS file, go into the **Staggered Animation** Folder, click on **CSS** and open **style.css.**

3. Around lines 53–56 notice the following four JavaScript variables. We will tween the jQuery object $shows, so it is shown in bold:

```
   var $header = $("#header"),
       $logo = $("#header img"),
       $upcoming = $("#upcoming"),
       $shows = $(".show");//selects every element with a class of show
```

   Make sure you are aware that the $shows variable selects all four of the elements with a class of show. There's no need to tween each show individually.

4. There are also three basic TweenLite tweens that animate the $header, $logo and $upcoming:

```
TweenLite.from($header, 0.5, {opacity:0});
TweenLite.from($logo, 0.5, {scale:0.5, rotation:-20, ease:Back.easeOut});
TweenLite.from($upcoming, 0.5, {opacity:0, delay:0.5});
```

   They have already been coded for you because we are going to focus on animating the $shows.

5. Preview **start.html** in a browser.

   You'll see the tweens for the header, logo, and upcoming shows. Let's create a complementary tween for the shows.

CREATING A STAGGERED ANIMATION USING TWEENMAX

1. Before we stagger our animation, let's take a quick look at another capability of TweenLite. In your text/code editor, after the last **from()** tween (around line 60), give yourself some space and type the bold code as shown:

```
TweenLite.from($upcoming, 0.5, {opacity:0, delay:0.5});

TweenLite.from($shows, 1, {opacity:0, delay:1});
```

This code makes every show fade in from an opacity of 0.

2. Save the file and preview the page in a browser. There is a delay of 1 second, during which the header animation plays, and then all four shows fade in at the same time, over a duration of 1 second.

3. To create a staggered (evenly-spaced sequential) animation, we will have to use TweenMax. To make room for the new and improved tween, in your text/code editor delete the **TweenLite.from()** tween for the shows object you just typed.

4. Replace the deleted tween with the following new **staggerFrom()** tween, as shown below in bold:

   ```
   TweenLite.from($upcoming, 0.5, {opacity:0, delay:0.5});

   TweenMax.staggerFrom($shows, 0.4, {opacity:0, delay:1});
   ```

5. Save the file and preview the page in a browser. The animation looks virtually the same. We made the duration of the tween faster but, after the header animates, all the $shows are still fading in simultaneously instead of in a staggered manner. The staggerFrom() method needs an additional parameter.

6. In your text/code editor in the **staggerFrom()** method, add the following bold parameter to your code like so:

   ```
   TweenMax.staggerFrom($shows, 0.4, {opacity:0, delay:1}, 1);
   ```

   This new parameter represents the **stagger,** the amount of time that transpires between the start of each tween. Take a moment to note the syntax: the parameter is always listed after the {properties} and it represents a value in seconds.

7. Save the file and preview the page in a browser. The start time of each tween is offset by one second. All four elements with a class of show have the same staggered animation.

8. The stagger is a bit too long. In your text/code editor, change the stagger to **0.2,** as shown below in bold:

   ```
   TweenMax.staggerFrom($shows, 0.4, {opacity:0, delay:1}, 0.2);
   ```

9. Save the file and preview the page in a browser. The shows' tweens overlap. It's a little too fast.

10. Let's make the stagger the same as the duration. In your text/code editor, change the stagger to **0.4,** as shown below in bold:

    ```
    TweenMax.staggerFrom($shows, 0.4, {opacity:0, delay:1}, 0.4);
    ```

11. Save the file and preview the page in a browser. Looks nice, but let's add some more pizazz to this tween!

USING TRANSFORMPERSPECTIVE TO ANIMATE IN 3D

1. We want to tween more than just the opacity. Let's add some 3D flair! In your text/code editor in the **staggerFrom()** method, enter the following bold property:

   ```
   TweenMax.staggerFrom($shows, 0.4, {opacity:0, rotationX:-90, delay:1}, 0.4);
   ```

   rotationX is a 3D property that rotates an element along its X (horizontal) axis. Our start value is -90º.

2. Save the file and preview the page in a browser. While the animation is different, it looks as if each element is just stretching vertically. This doesn't really look like a 3D effect. It lacks perspective.

3. To give our tween 3D distortion, we have to specify a vanishing point by entering a **transformPerspective** property. In your text/code editor, type the following bold code above the staggerFrom tween, like so:

   ```
   TweenLite.from($upcoming, 0.5, {opacity:0, delay:0.5});

   CSSPlugin.defaultTransformPerspective = 100;

   TweenMax.staggerFrom($shows, 0.4, {opacity:0, rotationX:-90, delay:1}, 0.4);
   ```

   NOTE: Make sure to place the CSSPlugin code before the tween runs.

4. Take a moment to review the code you just typed. It's important to point out that this code will apply the same transformPerspective to every element that will be animated in 3D space. If we had other elements we needed to be animated in 3D space and wanted to set separate perspective values for just the $shows, we could have entered the following code:
   TweenLite.set($shows, {transformPerspective = 100});

5. Save the file and preview the page in a browser. Whoa, that sure worked! There's some serious 3D distortion going on. It's a little much, though, because we used a relatively low value of 100.

   The transform perspective allows us to set a distance of where the virtual camera is from the element that's being animated. Imagine that you were standing very close to the object. A low value replicates this proximity—the lower the number you use, the more 3D distortion there is. On the other hand, the greater the value, the further the distance and the less intense the visual effect. We've found that using a perspective value somewhere between 600 to 800 works really well.

6. In your text/code editor, increase the TransformPerspective to a less extreme **600,** as shown below in bold:

   ```
   CSSPlugin.defaultTransformPerspective = 600;
   ```

7. Save the file and preview the page in a browser. The distortion is less jarring to the viewer and more effective overall.

USING TRANSFORMORIGIN IN A 3D TWEEN

By default, the animation originates at the vertical and horizontal center point. As we did earlier in a 2D tween, we can specify a transformOrigin to have more control over the animation. Let's make each show appear as if it's "swinging" down from the top.

1. In your text/code editor, add the following bold code to the **staggerFrom()** method:

```
TweenMax.staggerFrom($shows, 0.4, {opacity:0, rotationX:-90,
transformOrigin:"center top", delay:1}, 0.4);
```

   We are essentially keeping the default horizontal value (center) but modifying the vertical value to **top.**

2. Save the file and preview the page in a browser. Each element flips down from the top! Great job, the tween is finished!

   NOTE: In browsers that don't support 3D transforms (IE9 and lower), the 3D animations will simply be ignored but the opacity animations will still work fine.

TWEENMAX METHODS

TweenMax also has staggerTo() and staggerFromTo() methods.

Find out more at: **api.greensock.com/js/com/greensock/TweenMax.html**.