Animate that icon! Easily create Animated Vector Drawables

Akaita Follow
Mar 27, 2018 · 6 min read ★

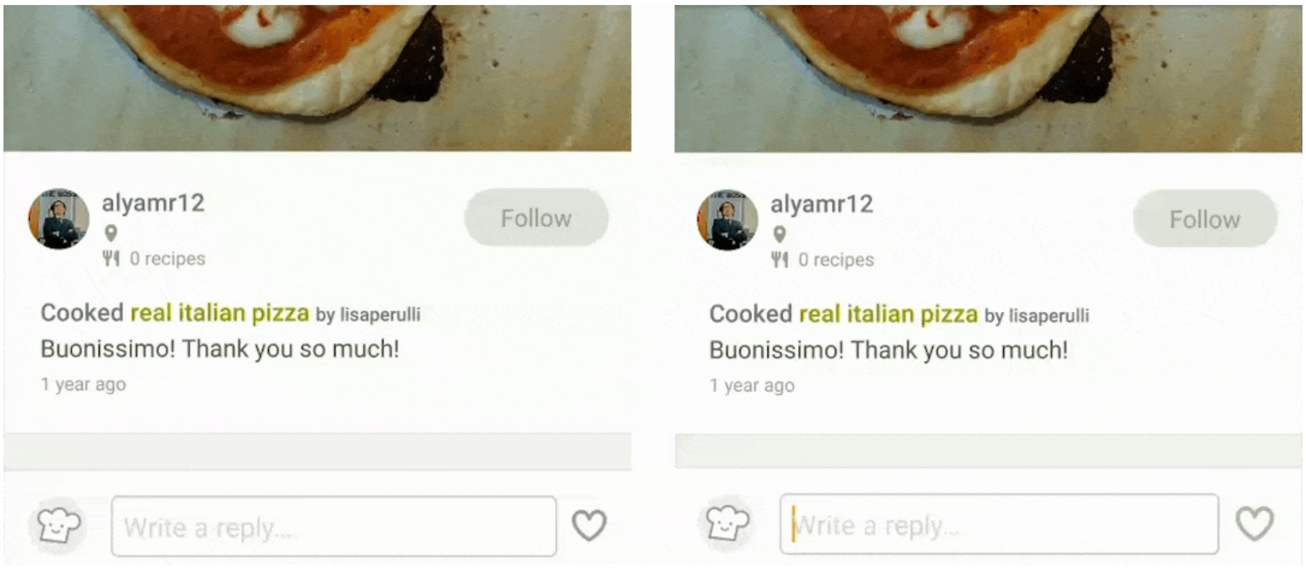
While developing apps, we often find ourselves replacing one icon with another to reflect a change of state, a newly available action, … Let’s find the best way to create a more enticing experience.

TL;DR

A typical solution usually involves changing the visibility of multiple Views, using StateListDrawables, testing your luck with `android:animateLayoutChanges` , using custom Views, … forcing us to often settle with “good enough”.

Then, Android API 22 introduced AnimatedVectorDrawable and Support Library 23.2.0 brought AnimatedVectorDrawableCompat for API 11+. Yay!

replace sudden icon changes with beautiful and engaging transformations🕺 :


alyamr12
Follow
0 recipes
Cooked real italian pizza by lisaperulli
Buonissimo! Thank you so much!
1 year ago
Write a reply...
alyamr12
Follow
0 recipes
Cooked real italian pizza by lisaperulli
Buonissimo! Thank you so much!
1 year ago
Write a reply...

A heart is replaced by a paper plane. Suddenly (left) and with a nice transformation (right)

We will today focus on this type transformations in which an icon “becomes” another icon, an icon “morphs” into another icon.

The problem

There had to be some caveat… 😢

For one, AnimatedVectorDrawables allow you to `start` and `stop` the animation. Once defined, the animation goes from one icon to the other. Therefore, if we want to do the **reverse animation**, we need to code a bit… fair enough, we can easily solve the issue with a custom View, etc.

The big issue, though, lies here: AnimatedVectorDrawables require you to provide two **compatible** SVG images 😕…

Two SVG images are compatible for morphing transformations only if their paths have the same commands, in the same order, and have the same number of parameters for each command.

For example, these two SVGs are built using completely different commands. They are **not compatible**:

```
<vector>
    <path
        android:pathData="M19,6.41L17.59,5 12,10.59 6.41,5 5,6.41 10.59,12 5,17
</vector>
<vector>
    <path
        android:pathData="M12,8c1.1,0 2,-0.9 2,-2s-0.9,-2 -2,-2 -2,0.9 -2,2 0.9
</vector>
```

reduced_incompatible_vectors.xml hosted with ❤ by **GitHub** **view raw**

Incompatible SVGs

These two SVGs, on the other hand, are built using the same commands, in the same order, with the same number of parameters. They are **compatible**:

```xml
<vector>
    <path
        android:pathData="M 2 21 L 23 12 L 2 3 L 2 10 L 17 12 L 2 14 Z"/>
</vector>
<vector>
    <path
        android:pathData="M 1 10 L 1 12 L 1 21 L 10 21 L 17 21 L 1 10 Z"/>
</vector>
```

Compatible SVGs

You will never receive two compatible vectors from a designer. It wouldn't make sense, because we are the only ones who want them to be compatible in order to use AnimatedVectorDrawables 😿😿😿

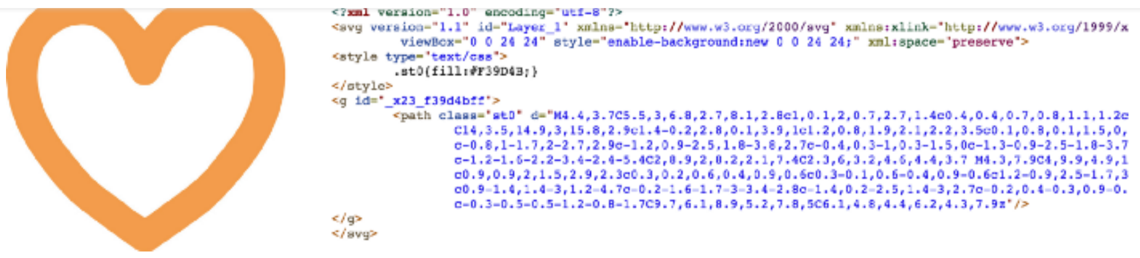
Our only option is to add, remove and modify the paths of the SVGs until they are actually compatible… It will take a very long time to do manually (I did it once, and never again).

This is what this recipe will guide you through. Here you have the process to go from any SVG to a nice morphing transformation in about less than 30 minutes (cooking time may vary).

Step 1: Get two SVG images

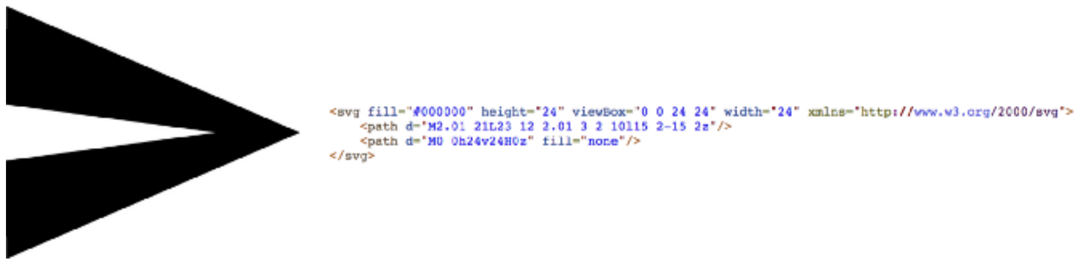
Let's use the same of the example, Cookpad's heart icon and Material Design's send icon. The heart icon will transform into the send icon.

⚠️ Just make sure they are the **same size** (24x24 pixels, for example)



```
<?xml version="1.0" encoding="utf-8"?>
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/x
         viewBox="0 0 24 24" style="enable-background:new 0 0 24 24;" xml:space="preserve">
<style type="text/css">
        .st0{fill:#F39D43;}
</style>
<g id="_x23_f39d4bff">
        <path class="st0" d="M4.4,3.7C5.5,3,6.8,2.7,8.1,2.8c1,0.1,2,0.7,2.7,1.4c0.4,0.4,0.7,0.8,1.1,1.2c
                C14,3.5,14.9,3,15.8,2.9c1.4-0.2,2.8,0.1,3.9,1c1.2,0.8,1.9,2.1,2.2,3.5c0.1,0.8,0.1,1.5,0,
                c-0.8,1-1.7,2-2.7,2.9c-1.2,0.9-2.5,1.8-3.8,2.7c-0.4,0.3-1,0.3-1.5,0c-1.3-0.9-2.5-1.8-3.7
                c-1.2-1.6-2.2-3.4-2.4-5.4C2,8.9,2,8.2,2.1,7.4C2.3,6,3.2,4.6,4.4,3.7 M4.3,7.9C4,9.9,4.9,1
                c0.9,0.9,2,1.5,2.9,2.3c0.3,0.2,0.6,0.4,0.9,0.6c0.3-0.1,0.6-0.4,0.9-0.6c1.2-0.9,2.5-1.7,3
                c0.9-1.4,1.4-3,1.2-4.7c-0.2-1.6-1.7-3-3.4-2.8c-1.4,0.2-2.5,1.4-3,2.7c-0.2,0.4-0.3,0.9-0.
                c-0.3-0.5-0.5-1.2-0.8-1.7C9.7,6.1,8.9,5.2,7.8,5C6.1,4.8,4.4,6.2,4.3,7.9z"/>
</g>
</svg>
```

Cookpad's heart icon



```
<svg fill="#000000" height="24" viewBox="0 0 24 24" width="24" xmlns="http://www.w3.org/2000/svg">
    <path d="M2.01 21L23 12 2.01 3 2 10l15 2-15 2z"/>
    <path d="M0 0h24v24H0z" fill="none"/>
</svg>
```

Material Design's send icon

They are wildly different, so this should be a good example 😅

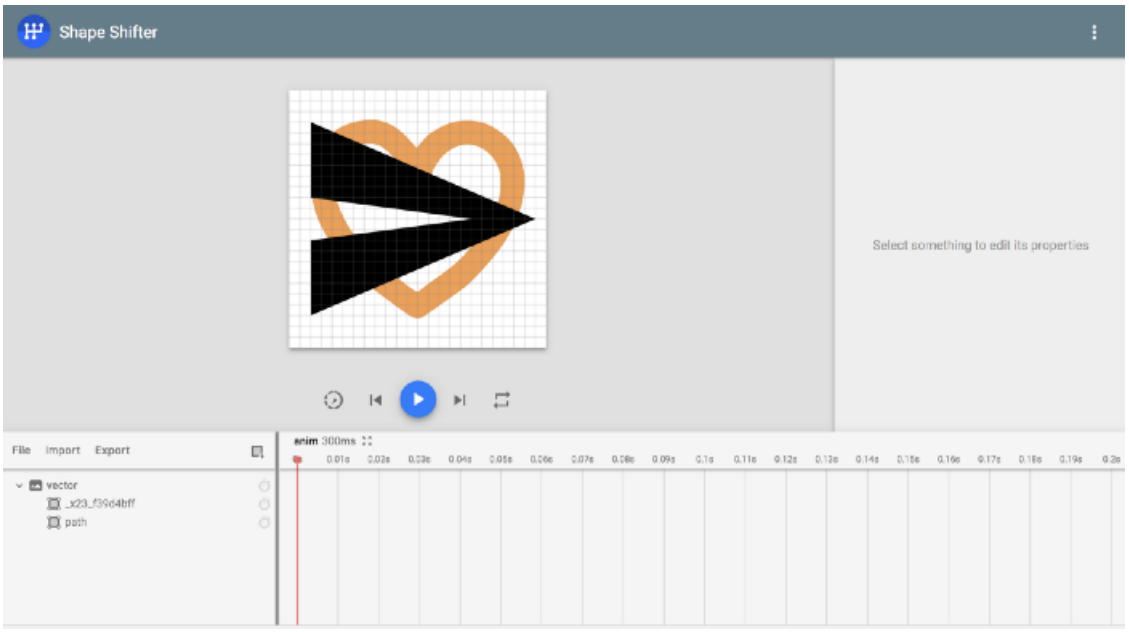
Step 2: Import both SVGs into ShapeShifter

ShapeShifter is a web-app created by Alex Lockwood.

This web-app will take care of making SVGs compatible for you, give you the chance to tweak the morph transformation, add rotations, add color changes, … We can't thank Alex enough for his contribution 👯💃👯

Just open it online in https://shapeshifter.design/ (use Chrome or Firefox, they seem to work best).

Should be able to just do: `Import --> SVG --> Choose file`

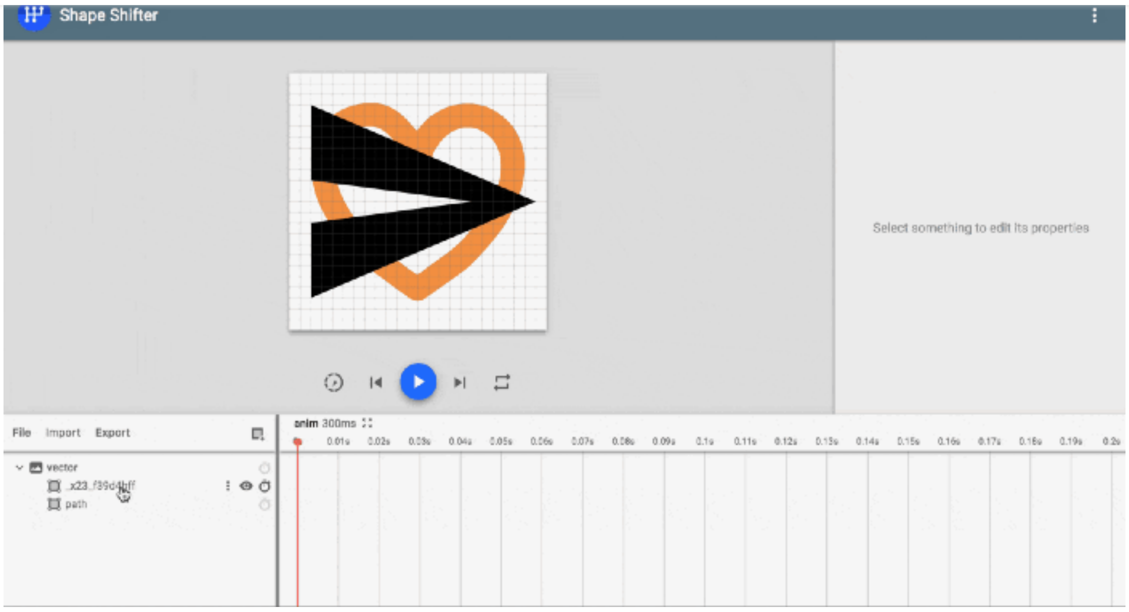


Two SVGs imported

note: there's an open bug which might force you to choose the file twice

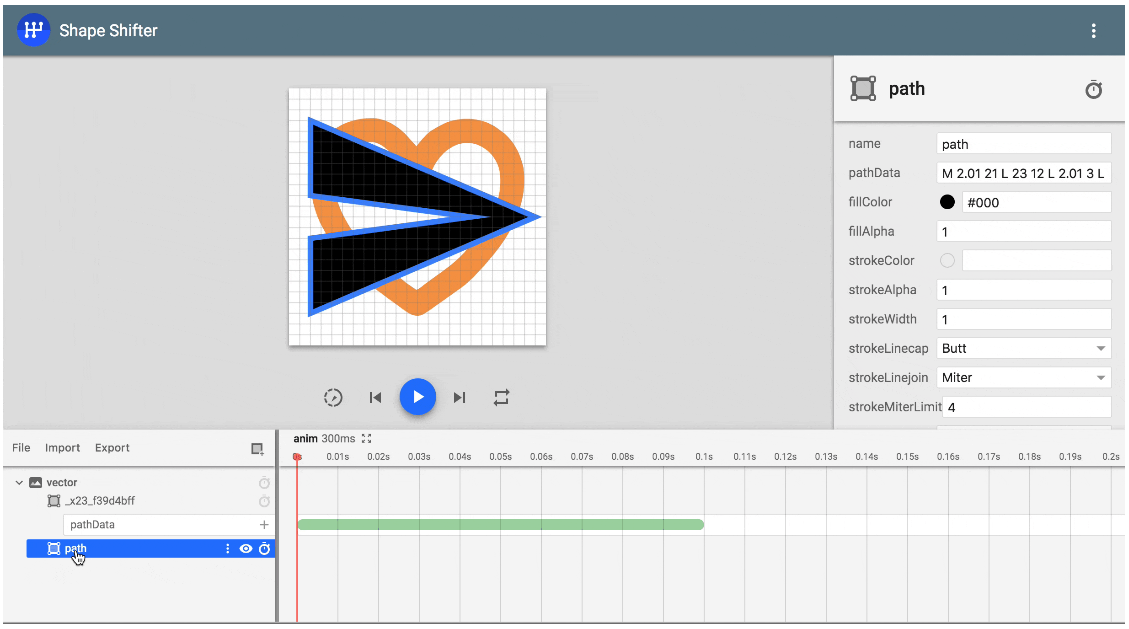
Step 3: Add morph transformation

Choose the `heart` icon and add a `pathData` transformation:



Add a morph transformation

Now, choose the `send` icon and copy it's `pathData` into the `pathData transformation`'s `toValue` field:

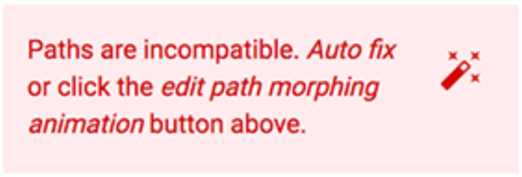


path

name	path
pathData	M 2.01 21 L 23 12 L 2.01 3 L
fillColor	● #000
fillAlpha	1
strokeColor	
strokeAlpha	1
strokeWidth	1
strokeLinecap	Butt
strokeLinejoin	Miter
strokeMiterLimit	4

Set "toValue" for morph transformation

You should now have a red transformation bar and a magic wand



Paths are incompatible. *Auto fix* or click the *edit path morphing animation* button above.

Step 4: Make the SVGs compatible

You probably guessed that the red highlighted **magic wand** is important. Indeed: just press it and ShapeShifter will convert the `toValue` into a SVG that is compatible with the `fromValue`.

The results of this “magic” will vary a lot, even when ShapeShifter gets updated now and then. You will most likely need to **edit** the path morphing animation.

To do so, first remove the `send` icon (use the “delete” key on your keyboard) and then go to the `Edit path morphing animation` window:

Here, you will be able to see a series of numbers when you hover over each icon. These are telling you where each point of the left icon will move to (until the right icon is shown):

- `number 1` on the left will move all the way to wherever `number 1` is on the right
- `number 2` on the left will move all the way to wherever `number 2` is on the right
- etc

You will probably need to play with the tools offered on the top-right area to create new numbers, to change where the numbers are positioned, etc.

Showing the correspondence between points in the path of the source and the target images

Step 5: Improve the transformation

is meant to be black. Therefore, I added a `fillColor` animation the same way I previously added a `pathData` animation.

Even more, I realized that the `heart` icon is pointy on the bottom and the `send` icon is pointy on the right side. Why not rotate the `send` icon 90 degrees to generate a smoother path morphing, and then rotate the result of this path morphing?

These tweaks will easily make a big difference:

Default morph transformation (left) and tweaked transformation (right) at 0.50x speed

Manually making the paths compatible would give you better results, but this is impressively good for a quick win!

Step 6: Bring the results into Android

1. In ShapeShifter: `Export -> Animated Vector Drawable`
2. Take the resulting xml, and add it into your `drawables`
3. Use the new drawable the same way you would use any other drawable
4. To animate it, you will need to obtain the `drawable` and cast it

Step 7: Reverse the animation

AnimatedVectorDrawables only go one way: forward.

There is therefore no easy way out of this AFAIK… we must implement both morphing animations as separate `drawables` 😢.

Understanding the xml generated by ShapeShifter should be easy enough to allow anybody to create the reverse animation without the drawables.

MorphView example taken from https://github.com/akaita/MorphView

note: this also allows us to avoid casts that could crash our app on runtime 😃

Extra step: Make the code reusable

Just made it for you 😉, check the repo:

Android Vectordrawable Animation