

ArtEvolver:

batch-convert images with ImageMagick



Sean McManus

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

Prepare to build an ever-evolving artwork using Python code that merges images endlessly. Start by discovering how to bulk process images with ImageMagick

Brian Eno's *77 Million Paintings* is a video installation that merges slides together to create endless variations of abstract art. I created my own version, called ArtEvolver, which runs on a Raspberry Pi and uses a Pimoroni 8-inch LCD screen. To work well, it needs a large library of images. This issue, you'll see how to curate them and prepare them using ImageMagick's (imagemagick.org) powerful batch processing. You'll learn how a single command can resize, crop, or transform hundreds of pictures. There's a short Bash script to automate image rotation, and we'll take a tour through some of ImageMagick's special effects.

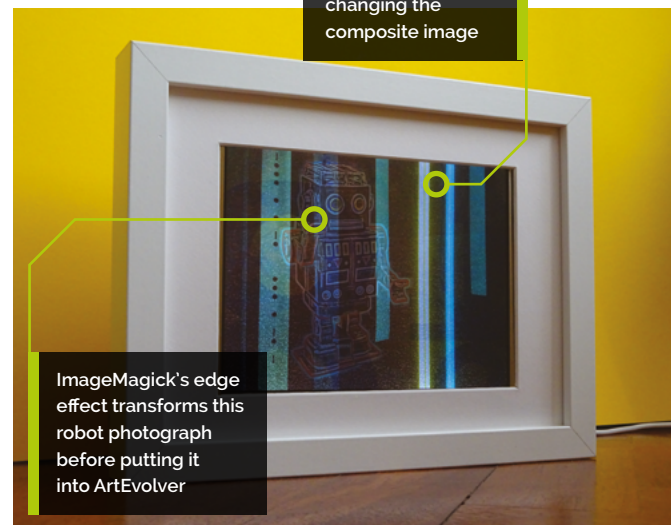
01 Collect your images

In Add/Remove Software (Raspberry Pi menu > Preferences), search for 'ImageMagick' and select the 'image manipulation programs – binaries' option. Click Apply to install the software. Alternatively, open a terminal window and enter:

```
sudo apt update
sudo apt install imagemagick -y
```

In this tutorial, you will make ArtEvolver unique by curating the images that you feed into it. Scour

ArtEvolver blends the robot with other pictures, constantly changing the composite image



ImageMagick's edge effect transforms this robot photograph before putting it into ArtEvolver

your personal photo archive and pick images that resonate. You can bulk these up with free images from sites such as unsplash.com, pixabay.com, and pexels.com. Your images will be layered in unpredictable ways, so search for colours, textures, and shapes that could be part of an abstract artwork. Textures like paper, stone, and paint make the art feel more organic; illustrations often work well. We've collected about 1000 images for this version, but you only really need a hundred or so. Put all your images (and nothing else) in a folder, and keep a separate, safe copy as a backup.

You'll Need

- > Raspberry Pi
- > Raspberry Pi OS
- > Some images
- > ImageMagick imagemagick.org



▲ You can use ImageMagick from the desktop, but the command line enables powerful batch processing capabilities

02 Experiment with the desktop app

ImageMagick installs into a desktop menu's **Graphics** folder. Start ImageMagick and click the splash screen. The splash screen is actually an image you can edit, but you'll get better results by loading a photo using the File menu. Try the various options in the Effects and F/X menus. These include emboss, sharpen, blur, sepia tone, and oil paint. The Enhance menu has options for changing the colour and tone of your image. You'll find the option to resize your image in the View menu. The Image Edit menu enables you to draw. You choose an element such as a filled circle, a fill colour, and 'stipple.' The stipple is a pattern for the fill, such as brickwork, waves, or fish scales. Drag on the canvas to draw your shape, but be warned that it can be slow on high-resolution images.

03 First steps with the terminal

Go into your images folder on the desktop and press **F4** to open a terminal window in that directory. There are two main ImageMagick commands: **convert** and **mogrify**. Convert is good for changing individual images or experimenting with effects. For example, you can mirror an image vertically with the **-flip** operation like this:

```
convert image_file.jpg -flip new_image_file.jpg
```

Use **-flop** to mirror it horizontally. With **mogrify**, you can process many images at the same time, which is a huge plus over the desktop app. You use wild cards, where ***** represents every file, and ***.jpg** would be every file ending with **.jpg**. Here's an example:

```
mogrify -flip *
```

Beware: **mogrify** overwrites your image files.

04 Make all your images landscape format

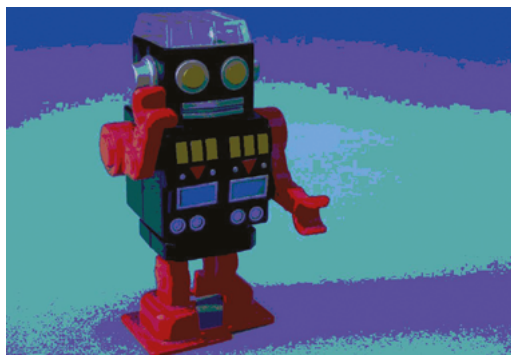
We are assuming you want to display your artwork on a landscape format screen (wider than it is tall). It's OK to have some portrait shape images in ArtEvolver, but it works best if most images fill the screen. With abstract images, it doesn't matter if you rotate them. The **landscapify.sh** listing shows a Bash script that rotates all the portrait images in a folder by 90 degrees. You can download it at **magpi.cc/artevolver** or create it using Text editor. Save it in the same folder as your images, with the name **landscapify.sh**. Open the folder in the terminal and enter:

```
chmod +x landscapify.sh
```

This will make the script executable. Then run it with **./landscapify.sh**. The script uses ImageMagick's **identify** command to get the dimensions of each image, and its **convert** command to rotate any pictures where the height is greater than (**-gt**) the width. Unchanged versions of the rotated images are saved in the new **original_images** subfolder.

05 Combining images

For some images, you could instead make a landscape file by joining two or more portrait images, side by side. The **montage** command enables you to join images together. You list the images you want to combine and specify the layout with the **-tile** parameter. We are using **2x1** to place the images side by side. You could create a **4x2** grid of images as well, and give the command eight files to combine. To remove any gaps between the images, you use the **-geometry** parameter with two zero values. Try this (replacing the 'image_file.jpg' names with your image files):



◀ The posterize effect (here used with a value of 4) gives your image the style of a vintage PC palette

Top Tip

View progress in the desktop

Use ImageMagick's **display image_file.jpg** command to see an image. It's easier to use the Image Viewer in the desktop to quickly review image batches.



▲ The implode effect, here used with a value of 0.5, distorts the image

```
montage image_file1.jpg image_file2.jpg
-tile 2x1 -geometry +0+0 new_file.jpg
```

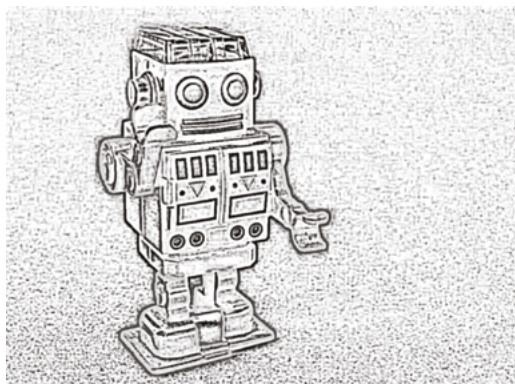
This will create a **new_file.jpg** file from the two images you supply.

06 Resize your images

The Pimoroni display we are using has a resolution of 1024×768 pixels, but camera images are typically much larger. To resize all the images in one go, use ImageMagick's **mogrify** command, like this:

```
mogrify -resize 1024x768^ *
```

The **^** symbol after the resolution sets the images to fill the screen, with some spilling over. If you leave the **^** off, the resized images will fit the screen. In that case, you see the entire image, but the empty spaces at the top and bottom won't work well in our final project. Beware: this command might take a while, and it will overwrite your original images.



Top Tip

Experiment with small batches

Image transformations can be slow, so experiment with test batches before running a command on a big batch.

► Charcoal images like this can work well when blended with coloured textures by ArtEvolver. The line thickness here is 5

07 Crop your images

For typical landscape camera images, any overmatter will be on the width of the image. Let's use **mogrify** to crop it off. The **-gravity** parameter specifies which part of the image you want to keep, using compass points. To keep the left and trim the right, use **west**, for example. Set the gravity to **center** to trim both sides equally. The best-looking crop depends on the image. I manually sorted my images into three folders for cropping left, right, and centre, and then ran a version of this command in each folder.

```
mogrify -gravity center -crop 1024x768+0+0 *
```

If you have images with unusual dimensions, you may need to crop **north** or **south** instead. (You can also use a north crop to extract the top of a portrait-shaped image before you resize it.)

08 Convert to greyscale

Some photos will blend better with other images if you convert them to greyscale. I use **convert** for changes like this so I can keep and compare the results of different effects. You can convert an image using:

```
convert -colorspace Gray image_file.jpg
new_image_file.jpg
```

You can convert to sepia (a browned-out photo style) using **-sepia-tone**, where a higher number makes the image darker:

```
convert -sepia-tone 75% image_file.jpg new_
image_file.jpg
```

09 Adjust the colours

Swapping colours often makes striking images. Use **-negate** to switch black and white, and swap complementary colours (e.g. blue and yellow). Try this:

```
convert -negate image_file.jpg new_image_
file.jpg
```

You can also try negating only the red, green, or blue channel:

```
convert -channel blue -negate image_file.jpg
new_image_file.jpg
```

Posterize reduces the number of colours in the image. Use a value of 2 for an 8-colour palette, 3 for 27 colours, and 4 for 64:

```
convert -posterize 2 image_file.jpg new_
image_file.jpg
```

10 Add visual effects

There are a number of special effects you can apply to images, including `-emboss`, `-charcoal`, `-edge`, `-paint`, and `-spread`. Experiment with them to transform photographs creatively. Spread gives you a frosted glass effect. The value for charcoal is a line thickness. Start here:

```
convert -emboss 2 image_file.jpg new_image_
file.jpg
```

11 Distort images

The `-wave` effect adds ripples to your image. You give it the height of the wave (amplitude) and the distance between two waves (wavelength), like this:

```
convert -wave 5x20 image_file.jpg new_image_
file.jpg
```

You can also use the `implode` effect to collapse an image, like this:

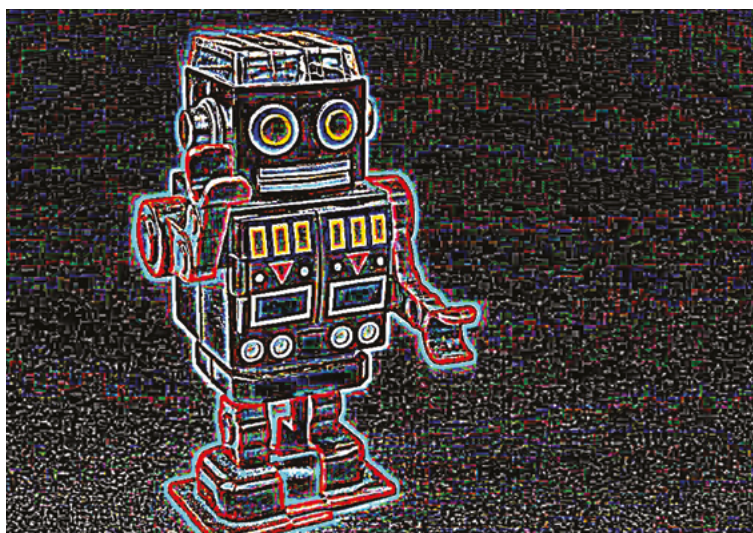
```
convert -implode 1 image_file.jpg new_image_
file.jpg
```

Use a negative number for the implode value to explode the image instead.

12 Combine effects


You can combine multiple transformations in one command. They're carried out in the order you list them. Here's an example that resizes, mirrors, and negates an image in a single command:

```
convert -resize 1024x768^ -flop -negate
```



```
image_file.jpg new_image_file.jpg
```

▲ The edge effect creates striking results like this

Using ImageMagick enables you to batch-convert images into a range of different styles. In the next ArtEvolver tutorial, we will collate these into a physical project that uses these transformations. 

landscapify.sh

DOWNLOAD
THE FULL CODE:

► Language: Bash

 magpi.cc/artevolver

```
001. #!/bin/bash
002. # Rotates portrait images (only) in the current folder
003. # From ArtEvolver Tutorial in The MagPi - by Sean McManus
004.
005. mkdir original_images
006.
007. # Remove any extensions in the list below that you're not
    using to avoid error messages
008. for image_file in *.jpg *.JPG *.png *.PNG;
009. do
010. # Make sure there is no space around the = below
011.     width=$(identify -format "%w" $image_file)
012.     height=$(identify -format "%h" $image_file)
013.     if test $height -gt $width
014.     then
015.         echo "$image_file is portrait shape [$width x
$height]. Rotating..."
016.         new_name="rotated-{$image_file}"
017.         convert -rotate 90 "$image_file" "$new_name"
018.         mv $image_file original_images
019.     else
020.         echo "$image_file is landscape already [$width x
$height]."
021.     fi
022. done
```

ArtEvolver:

Build an abstract art installation



Sean McManus

WRITER

Author of *Mission Python*, *Scratch Programming in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

You'll Need

- ▶ Raspberry Pi OS
- ▶ Pimoroni HDMI 8-inch IPS LCD Screen Kit magpi.cc/8inlcd
- ▶ Picture frame (optional)
- ▶ Prepared images (see *The MagPi* 118) magpi.cc/118

- ▶ This IKEA Ribba picture frame is the right size for the screen, and deep enough to contain your Raspberry Pi

Transform Raspberry Pi into an abstract artist, with ArtEvolver. It blends images together to create surprising, stunning, and surreal artworks that constantly change

Say goodbye to boring old posters, and hello to ArtEvolver. Every time you look at this digital art, it's different. Fill it with patterns, paintings, and textures. Add photos of street art, structures, and neon lights. Personalise it with your children's drawings or monochrome shots of your favourite places. While you can view this on the desktop, it looks most impressive on a small screen in a picture frame. The program, based on Pygame, constantly cycles through your images and includes a safe shutdown function. To get started quickly, you can download the code at magpi.cc/artevolver.

01 Prepare your images folder

The art starts here! Store your images in a folder called **images_folder**. Ideally, the images should be resized and cropped to fit your screen size (1024×768 in my case). Last issue, we



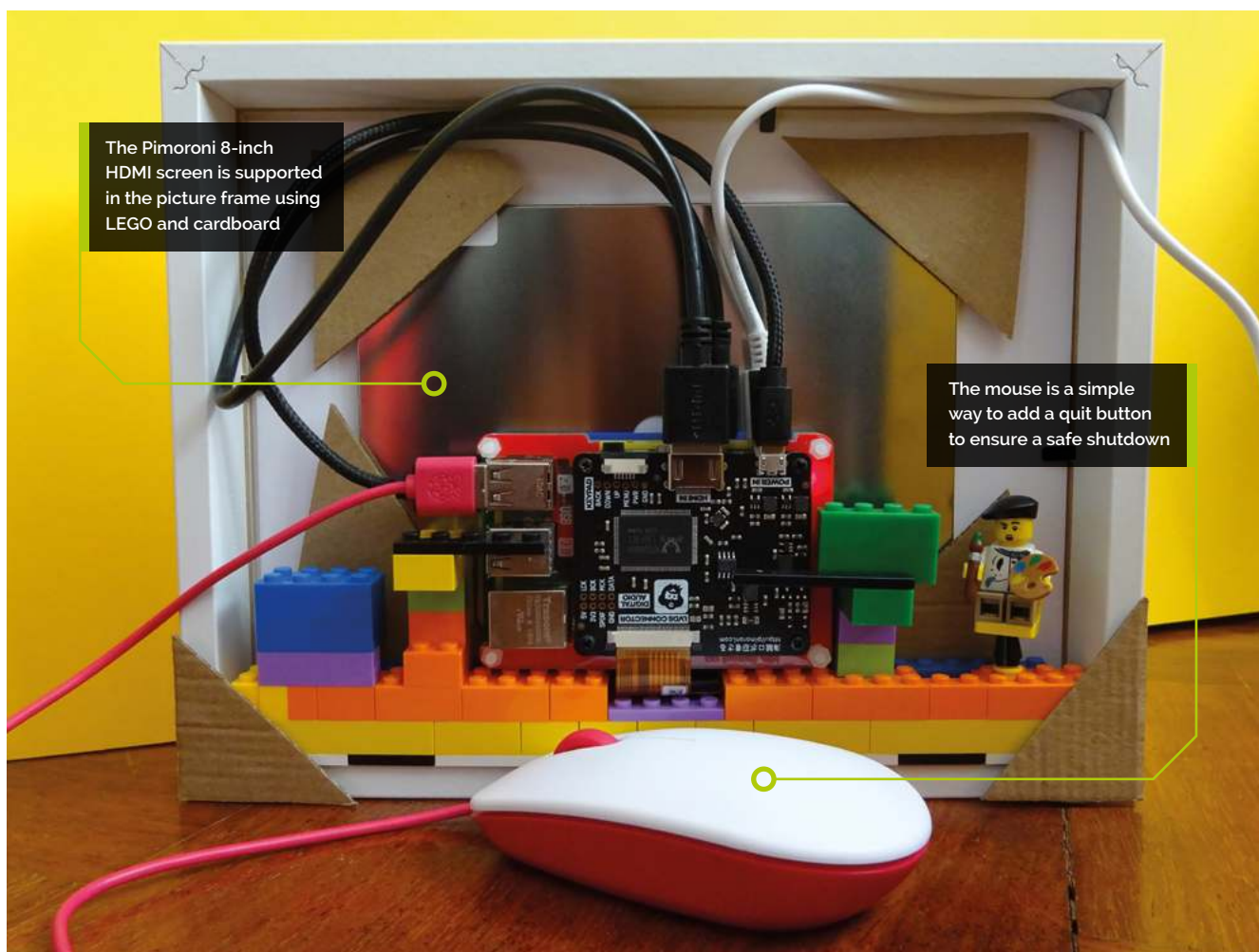
showed you how to do this with ImageMagick. It's OK to include smaller (e.g., square) or larger images, and ArtEvolver will scale them. Remove any work-in-progress images from the folder, including the **original_images** folder if you used my **landscapify.sh** script from last issue. Feel free to organise your images in subfolders.

02 Displaying the images

ArtEvolver uses Pygame to display the images. The **display_prototype.py** listing shows the simple demo created to test how the images would look, layered on top of each other. It loads three images, sets their opacity (alpha values), and then uses `windowSurface.blit()` to add them to the screen buffer. That function takes the image object and the position co-ordinates. Finally, `pygame.display.update()` refreshes the display and makes the changes visible. Change the file names in the **images** list to your own.

03 Building the pictures list

To make this work well, it needs to be able to handle hundreds of images without you needing to add them in the code. Take a look at **ArtEvolver.py**. The `index_images()` function discovers the images in the **images_folder** for you, and puts their file names (including paths) into the **pictures** list. The function is recursive. You call it with the name of the directory you want to index. When the function finds a subdirectory (using



The Pimoroni 8-inch HDMI screen is supported in the picture frame using LEGO and cardboard

The mouse is a simple way to add a quit button to ensure a safe shutdown

`os.path.isdir`), it calls itself to index that directory too. This code only indexes `.png` and `.jpg` images, so rename any images that have upper-case extensions, or modify the code accordingly.

“ The alpha (opacity) value is changed to make the images fade in and out ”

04 Understanding the data structure

The `pictures` list is the primary list of all the image paths and file names. At the start of each run, it's copied to the `sequence` list, which is shuffled into a random order. Images are pulled out of the `sequence` list, and cycled through opacity values from 0 to 150, and back to 0. Each visible image is represented by a `Slide` object, containing its file name and current opacity. The `current_slide_list` stores the five `Slide` objects

that are currently on screen. When a slide fades from view, its file name is replaced in the `Slide` object with the next one from the `sequence` list.

05 Understanding opacity

Pygame is used to display the images and overlay them on each other. The `alpha` (opacity) value is changed to make the images fade in and out. We're using a maximum opacity of 150 to stop one image blocking the rest out totally. At first, we had code that either increased or decreased the opacity depending on whether a slide was fading in or out. We've simplified that by using opacity values from -150 to +150. Now, we just add 1 each time around the loop. Any negative values are made positive using Python's `abs()` function before the opacity value is used. Values go from maximum opacity (-150) to totally transparent (0) and back to peak opacity (150). At 0, the slide's image changes. At 150, the opacity value is changed to -150, and the cycle repeats.

Top Tip

Make a digital photo frame

You can turn this project into a digital photo frame. Simply change the `starting_opacities` list to `!0!` to make it display one image at a time.

▶ Here, two pieces of street art have been merged with ink clouds in water to create a textured, colourful image



Top Tip

Shorter is better

Search eBay for the shortest USB and HDMI cables you can find, so they fit more easily into the picture frame.

06 Setting up the slides

If all the slides start with the same opacity, you have a single composite image that fades in and out. It's more interesting to have each slide fade in and out separately, so the art is in flux. The `starting_opacities` list sets the initial values for opacity. Add or remove values in this list to change the number of simultaneous slides. We found these values by experimentation.

07 Scaling the images

Although it's a good idea to resize your image files, ArtEvolver uses the `pygame.transform.scale()` function to scale images to fit the screen. For portrait images, the scaling factor will be the image height divided by the window height. For example, if your image height was 1000 pixels and the window was 500

pixels high, the scaling factor would be 2. When the image height is set to the window height, the image width is divided by the scaling factor so it remains proportional. Landscape images are scaled to fit the window horizontally, with the height scaled proportionally. Your images don't all have to fill the window. Some of my most effective ones appear in a stripe across the middle.

08 Centring the images

Images are positioned in the middle of the window. You have to give the `windowSurface.blit()` function the co-ordinate for the top-left corner of the image. To calculate what the x co-ordinate should be, we take the middle of the window (the width divided by 2), and subtract half the image width. We do something similar for the y co-ordinate. Images that fill the window will in any case end up positioned in the top left at (0,0). However, this method neatly places images that don't fill the window.

“ We decided the simplest solution was to plug in a spare mouse ”

09 Add a quit function

We wanted to add an off button to ensure there is a safe shutdown, rather than just pulling the plug to turn it off. You could look at adding a HAT or wiring up your own button to the GPIO pins. However, we decided the simplest solution was to plug in a spare mouse. The frame is on our

display_prototype.py

▶ Language: Python

```
001. #ArtEvolver prototype - layers three images
002. import pygame
003. pygame.init()
004. windowSurface = pygame.display.set_mode((1024, 768))
005.
006. images = ["images_folder/image1.jpg",
            "images_folder/image2.jpg", "images_folder/image3.jpg"]
007. opacities = [45, 90, 135]
008.
009. windowSurface.fill((255,255,255))
010. for i in range(3):
011.     image_to_show = pygame.image.load(images[i])
012.     image_to_show.set_alpha(opacities[i])
013.     windowSurface.blit(image_to_show, (0, 0))
014. pygame.display.update()
```



▶ The program centres portrait-shaped images, such as this one of a man at a window

ArtEvolver.py

> Language: Python

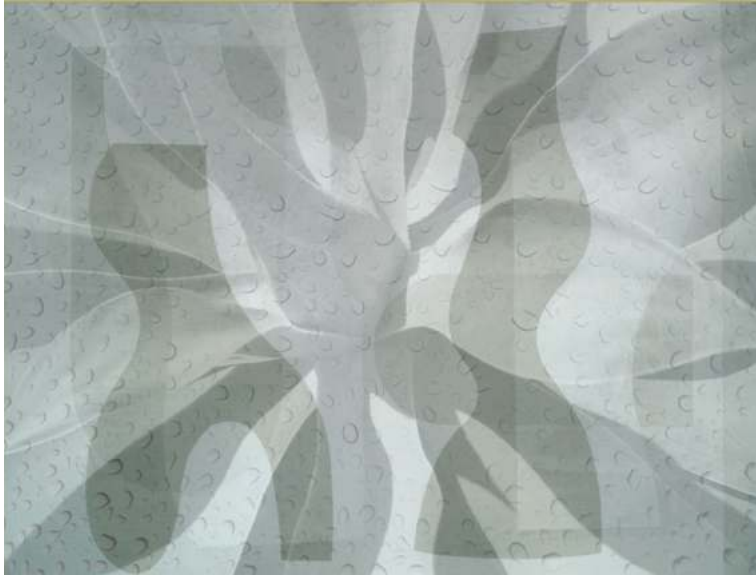
DOWNLOAD
THE FULL CODE:

 magpi.cc/artevolver

```

001. # ArtEvolver - by Sean McManus - www.sean.co.uk
002. import pygame, random, os
003. pygame.init()
004.
005. win_width = 1024
006. win_height = 768
007. windowSurface = pygame.display.set_mode((win_width,
008. win_height))
009. pygame.display.set_caption('ArtEvolver')
010. pygame.mouse.set_visible(False)
011.
012. class Slide:
013.     def __init__(self, filename, opacity):
014.         self.filename = filename
015.         self.opacity = opacity
016.
017. def index_images(path, images_list):
018.     for dir_or_file in os.listdir(path):
019.         path_plus_dir_or_file = os.path.join(
020. path, dir_or_file)
021.         if os.path.isdir(path_plus_dir_or_file):
022.             index_images(
023. path_plus_dir_or_file, images_list)
024.         elif dir_or_file.lower().endswith('.png') or
025. dir_or_file.lower().endswith('.jpg'):
026.             images_list.append(path_plus_dir_or_file)
027.     return images_list
028.
029. pictures = index_images("images_folder", [])
030.
031. while True:
032.     sequence = pictures.copy()
033.     random.shuffle(sequence)
034.
035.     # Set up initial list of current slides
036.     current_slide_list = []
037.     starting_opacities = [-90, -45, 45, 90, 135]
038.     for layer_opacity in starting_opacities:
039.         this_image = sequence.pop(0)
040.         this_slide = Slide(this_image, layer_opacity)
041.         current_slide_list.append(this_slide)
042.
043.     while len(sequence) > 0:
044.         windowSurface.fill((0,0,0)) # Black
045.         for this_slide in current_slide_list:
046.             image_to_show = this_slide.filename
047.             new_opacity = this_slide.opacity + 1
048.             if new_opacity == 150:
049.                 new_opacity = -150
050.             elif new_opacity == 0:
051.                 this_slide.filename = sequence.pop(0)
052.                 # replace image in this slide
053.                 this_slide.opacity = new_opacity
054.
055.             image_to_show = pygame.image.load(
056. this_slide.filename)
057.             image_width = image_to_show.get_width()
058.             image_height = image_to_show.get_height()
059.
060.             # Images are scaled for the long side
061.             (fit, not fill, the window)
062.             if image_height > image_width:
063.                 scaling_factor = image_height /
064. win_height
065.                 new_width = int(
066. image_width / scaling_factor)
067.                 image_to_show = pygame.transform.
068. scale(image_to_show, (new_width, win_height))
069.             else:
070.                 scaling_factor = image_width /
071. win_width
072.                 new_height = int(
073. image_height / scaling_factor)
074.                 image_to_show = pygame.transform.
075. scale(image_to_show, (win_width, new_height))
076.
077.             # Remove # on next line if your screen is
078. upside down
079.             #image_to_show = pygame.transform.
080. flip(image_to_show, True, True)
081.
082.             # get new height and width
083.             image_width = image_to_show.get_width()
084.             image_height = image_to_show.get_height()
085.
086.             image_to_show.set_alpha(abs(new_opacity))
087.             windowSurface.blit(image_to_show,
088. (int(win_width/2) -
089. int((0.5*image_width)),
090. int(win_height/2) -
091. int((0.5*image_height))))
092.
093.             pygame.display.update() # Shows composite
094. after all slides have been blitted
095.             pygame.time.wait(30) # Adjust timings here if
096. necessary
097.
098.         for event in pygame.event.get():
099.             if event.type == pygame.MOUSEBUTTONDOWN:
100.                 pygame.quit()

```

▲ Three monochrome images are blended here to make an abstract image, with an overlaid texture of water on glass

desk, so it's easy to hide the mouse behind it. The final lines in **ArtEvolver.py** use Pygame to check for a mouse click and quit the program if one is detected. A mouse click is the only way to close the program window.

“ A mouse click is the only way to close the program window ”

Top Tip

Image credits

Thanks to Jon Tyson, Parrish Freeman, Pawel Czerwinski, Sasha Freemind, Jen Theodore, Jr Korpa, Birmingham Museums Trust, Hermann Wittekopf, and Andrii Leonov who provided the images in the ArtEvolver composites shown here. Find these images, and many more, at unsplash.com.

10 Connect the screen

You can run ArtEvolver on any screen, or even just run it on your desktop. We're running it using a Pimoroni 8-inch HDMI IPS LCD Screen Kit. The display driver board connects to the top of the screen with a short cable, and connects to your Raspberry Pi computer using USB and HDMI cables. The display driver board doesn't need any GPIO pins.

11 Build the frame

The screen is inside an IKEA picture frame, which looks great. The display driver board is mounted above our Raspberry Pi board using standoffs, and the computer is in a Pibow case to insulate it from the screen. This arrangement keeps everything tidy and compact. To support the weight of the computer more easily, we mounted the screen upside down, and used LEGO to support the computer at the right height. There's a line in


the code that flips the images upside down so they look right. Delete the # at the start of that line if your screen is also upside down.

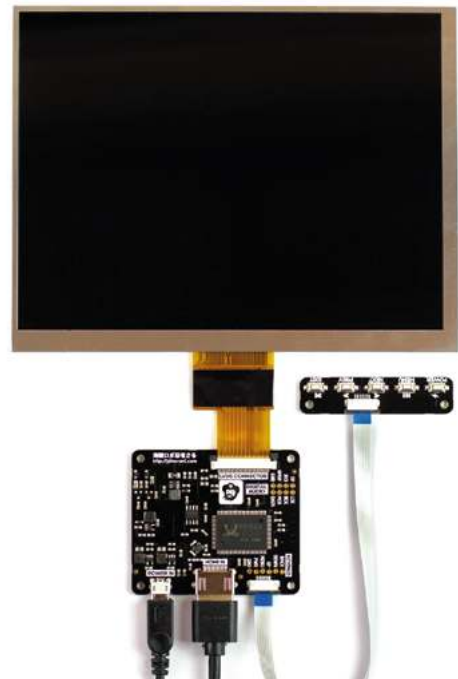
12 Make it autostart

We'll configure the **.bashrc** file to run ArtEvolver when the computer powers up and to shut down the machine safely when ArtEvolver finishes. We are using a delay of three minutes, so there's time to cancel the shutdown (using **sudo shutdown -c**) if I need to make changes, and to make sure we don't lock the machine. Note that these commands will also run if you open a Terminal window in the desktop.

Open a Terminal window and enter **sudo nano ~/.bashrc**. Add these two lines to the end of the file, save with **CTRL+O**, and exit with **CTRL+X**:

```
python ArtEvolver.py
sudo shutdown +3
```

On the desktop, go into Preferences > Raspberry Pi Configuration, and set the computer to boot to the CLI (command-line interface). (To get back to the desktop, you can enter **startx** at the command prompt). Reboot, and enjoy your digital art! 



▲ The Pimoroni 8-inch HDMI IPS LCD Screen Kit includes a display driver board, which sits between the computer and the screen