



QGIS Animation Workbench

Documentation

Tim Sutton, Nyall Dawson, Jeremy Prior
2022

1 QGIS Animation Workbench

Welcome to the QGIS Animation Workbench (QAW). QAW is a [QGIS Plugin](#) that will help you bring your maps to life! Let's start with a quick overview. Click on the image below to view a 14 minute walkthrough on YouTube.



1.1 Why QGIS Animation Workbench?

QGIS Animation Bench exists because we wanted to use all the awesome cartography features in [QGIS](#) and make cool, animated maps! QGIS already includes the Temporal Manager which allows you to produce animations for time-based data. But what if you want to make animations where you travel around the map, zooming in and out, and perhaps making features on the map wiggle and jiggle as the animation progresses? That is what the animation workbench tries to solve...

1.2 Features

- **Modes** : Supports 3 modes: Sphere, Planar and Static.
- **Sphere**: Creates a spinning globe effect. Like Google Earth might do, but with your own data and cartography.
- **Planar**: Lets you move from feature to feature on a flat map, pausing at each if you want to.
- **Static**: The frame of reference stays the same and you can animate the symbology within that scene.
- Add music to your exported videos - see the [Creative Commons](#) website for a list of places where you can download free music (make sure it does not have a 'No Derivative Works' license).

"Bring your QGIS projects to life!"





- Multithreaded, efficient rendering workflow. The plugin is designed to work well even on very modest hardware. If you have a fast PC, you can crank up the size to the thread pool to process more jobs at the same time.

 **Note:**

Supports only English currently - we may add other languages in the future if there is demand.

"Bring your QGIS projects to life!"





1 Quickstart

1.1 Installing the QGIS Animation Workbench plugin

⚠ Please take note: We have not yet published the plugin in the QGIS Plugin Repository, but when we do you will be able to access it simply by clicking on the "QGIS Animation Workbench" option in the QGIS Plugin Manager.

1.1.1 Manual install from GitHub (tagged release)

To install, visit the [Github Repository](#), click on the **Actions** tab, and click on the **Make QGIS Plugin Zip For Manual Installs** workflow (the bottom one).

The screenshot shows the GitHub Actions page for the repository `timlinux/QGISAnimationWorkbench`. The 'Actions' tab is active. On the left, there's a sidebar with 'Workflows' and a 'New workflow' button. Below it is a list of workflows: 'pages-build-deployment', 'Black python code', 'Run Python Plugin Tests', 'Compile MKDocs to PDF', 'Build MKDocs And Publish...', and 'Make QGIS Plugin Zip for ...'. The last workflow is highlighted with a red box. To the right, under 'All workflows', there's a section for 'pages-build-deployment' showing 61 workflow runs. The most recent run is highlighted with a green checkmark and labeled 'Update and rename test_plugin.yaml to RunPythonPl...'. There are two other runs listed below it, also with green checkmarks. The URL for the screenshot is [ps://github.com/timlinux/QGISAnimationWorkbench/actions/workflows/pages-build-deployment](https://github.com/timlinux/QGISAnimationWorkbench/actions/workflows/pages-build-deployment).

Click on the most recent workflow run (the top one).

The screenshot shows the details of a workflow run for the 'Update and rename test_plugin.yaml to RunPythonPl...' workflow. It has a green checkmark and the status 'main'. The run was triggered by a commit 'Make QGIS Plugin Zip for Manual Installs #15: Commit 61e3aba pushed by timlinux'. Below it is another workflow run with a green checkmark and the status 'main', triggered by a commit 'Black python code lint #3: Commit 61e3aba pushed by timlinux'. The URL for the screenshot is [ps://github.com/timlinux/QGISAnimationWorkbench/actions/runs/15](https://github.com/timlinux/QGISAnimationWorkbench/actions/runs/15).

Scroll down on the on the page.

"Bring your QGIS projects to life!"





✓ **Update and rename test_plugin.yaml to RunPythonPluginTests.yaml** 📁 Make QGIS Plugin Zip for Manual
Installs #15

Re-run all jobs ⋮

Summary

Triggered via push 14 minutes ago Status Success Total duration 27s Artifacts 1

Jobs Scroll down

Build Package 🚀

MakeQGISPluginZipForManualInstall.yml on: push

Build Package 🚀 12s

And click on **animation_workbench** to download the most recent build of the plugin

MakeQGISPluginZipForManualInstall.yml
on: push

Build Package 🚀 12s

Artifacts
Produced during runtime Click to download latest build from last commit

Name	Size
animation_workbench	1.62 MB

(animation_workbench)

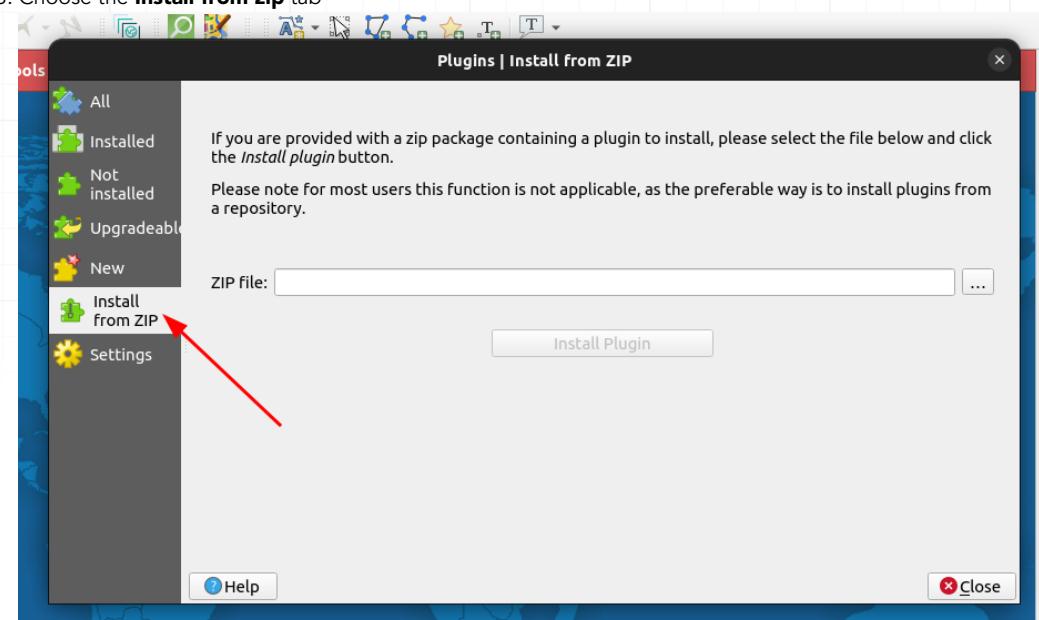
"Bring your QGIS projects to life!"





Download the animation_workbench.zip file and open it in QGIS using the plugin manager as described below.

1. Open QGIS
2. **Plugins → Manage and install plugins ...**
3. Choose the **Install from zip** tab



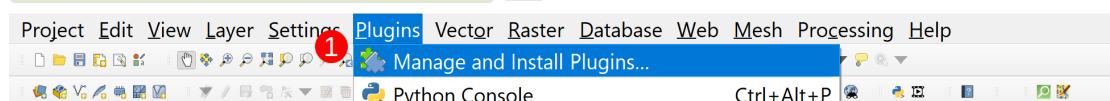
4. Select the **animation_workbench.zip** download
5. Click the **Install Plugin** button.

1.1.2 Install from plugin manager

⚠ Please take note: We have not yet published the plugin in the QGIS Plugin Repository, in the mean time please follow the steps above to do the installation.

To access the QGIS Plugin Manager you simply need to select **Plugins →**

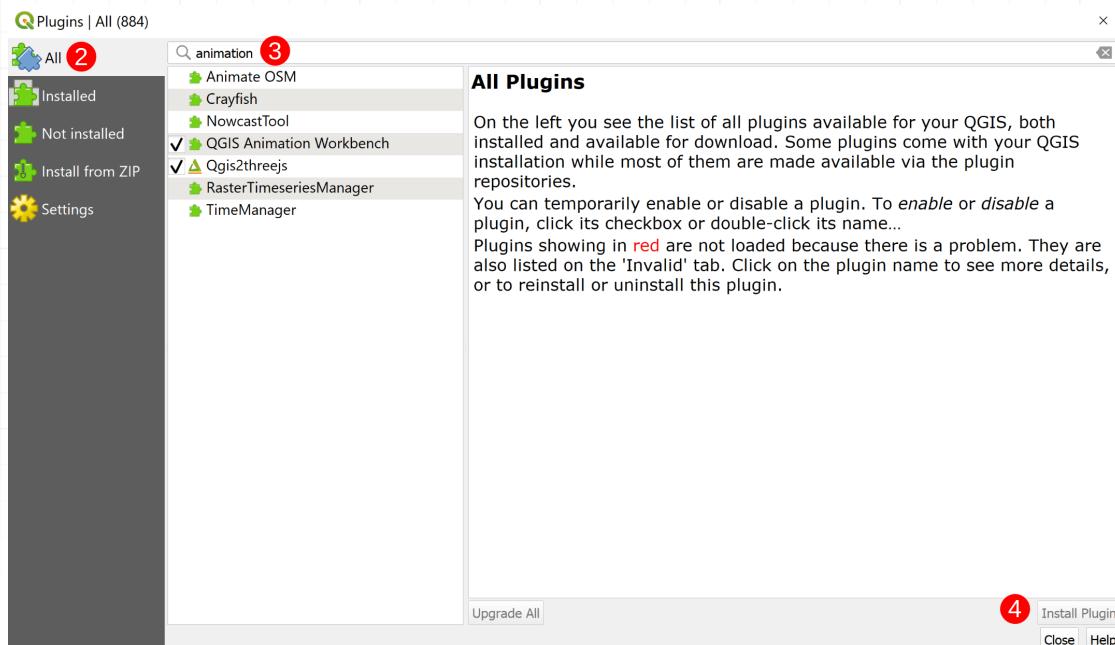
Manage and Install Plugins... (1) in the Menu Toolbar.



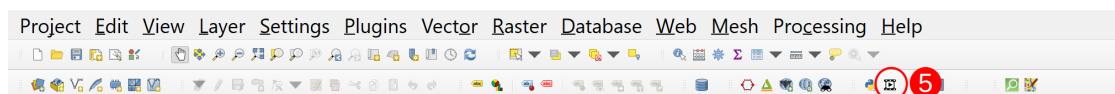
Once the QGIS Plugin Manager loads, you need to navigate to the **All (2)** tab and type "animation" into the search bar (3). Select QGIS Animation Workbench from the list of available plugins and then select **Install Plugin (4)**.

"Bring your QGIS projects to life!"





Once the Animation Workbench is installed, you can access it by clicking on the **Animation Workbench** icon (**5**) in the Plugin Toolbar.



Note:

Note if you are on Ubuntu, you may need to install the Qt5 multimedia libraries.

Code:

```
sudo apt install PyQt5.QtMultimedia
```

"Bring your QGIS projects to life!"





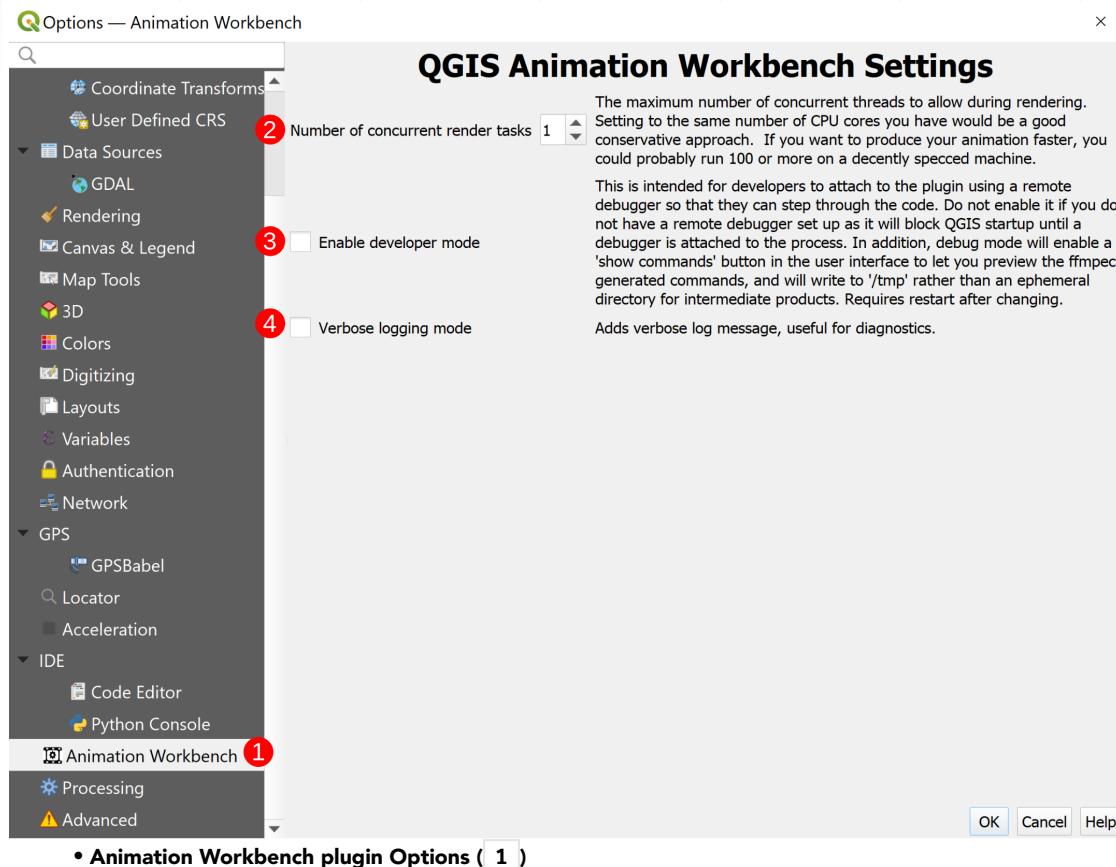
1.2 Initial Configuration

There is nothing really to configure! We do provide a few options in the configuration dialog, but most users should not need to change them.

You can access the QGIS Animation Workbench plugin options by opening the standard QGIS Setting dialog and clicking on the animation workbench tab.

Note:

Settings → Options



"Bring your QGIS projects to life!"





Currently there are just three configuration options:

- **Number of concurrent render tasks (2)**: This is the number of concurrent tasks that will be used to render animations. The default is 1.
- **Enable developer mode (3)**: This is a developer option that enables the developers to see an icon in the toolbar which will start the debug remote server.
- **Verbose logging mode (4)**: This will add extra messages in the logging pane to help you understand what is going on during the rendering process.

"Bring your QGIS projects to life!"





1.3 Using the Animation Workbench

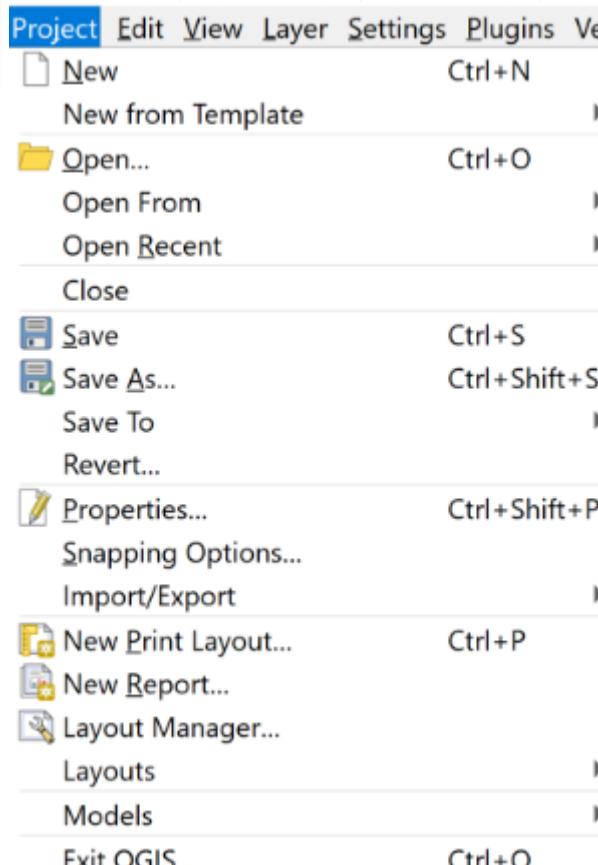
In this section, we describe the general workflow for using the Animation Workbench.

1.3.1 Process Overview

1. Create a QGIS project!
2. Identify features that will be animated.
3. Use the QGIS Expressions system with the variables introduced by the Animation Workbench to define behaviours of your symbols during flight and hover modes of your animation.
4. Open the Animation Workbench and configure your animation, choosing between the different modes and options.
5. Render your animation!

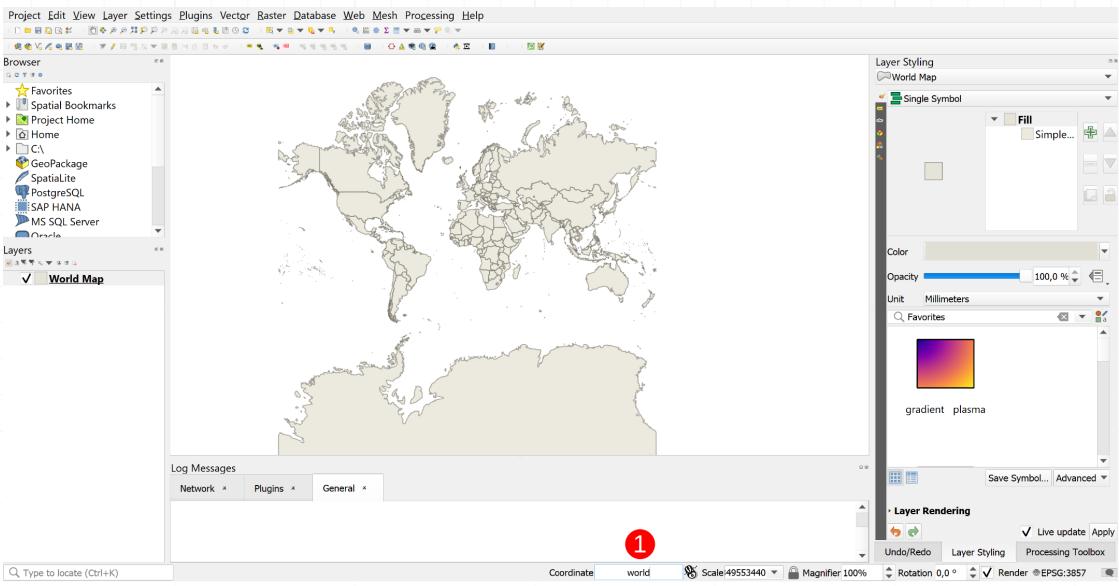
1.3.2 More in Depth Process

1. Create a QGIS Project Open QGIS and click on **Project** → **New**



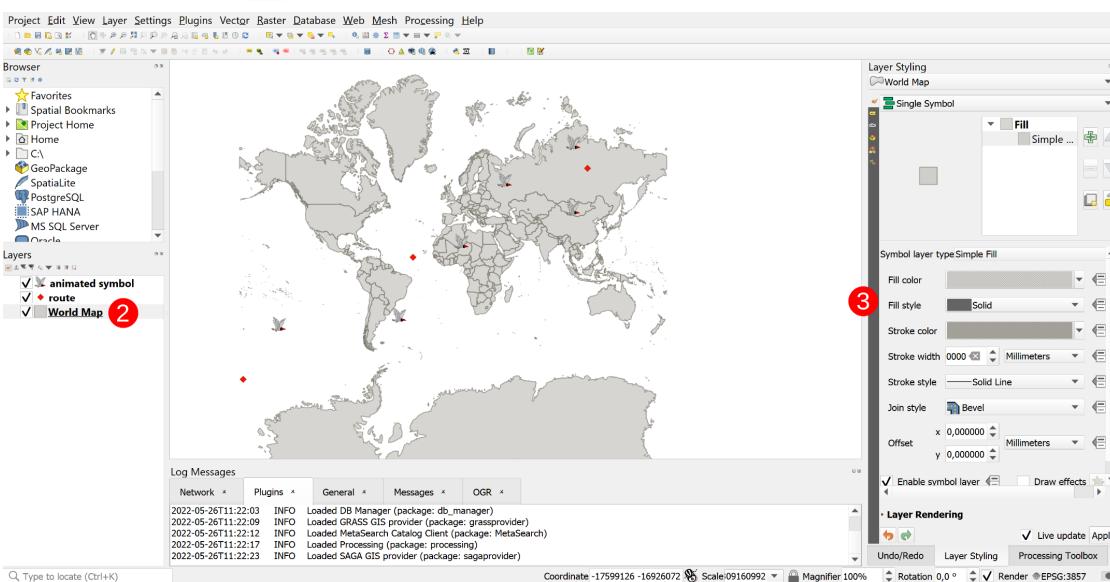
"Bring your QGIS projects to life!"





A simple way to add a base layer is to type "world" (1) into the coordinate textbox

Style the layers you've added to make your project look a bit better. Select the layer (2) you want to style and in the Layer Styling toolbar (3), style the layer to look appealing to you.



"Bring your QGIS projects to life!"





2. Identify features that will be animated.

Pick the layer (or layers) that you want to animate. Then either find or create the animation for the layer. Make sure you have all the correct attribution for any animations you use. Below is an example of an animation split into its frames.



3. Use the QGIS Expressions system with the variables introduced by the Animation Workbench to define behaviours of your symbols during flight and hover modes of your animation.

Select the layer you want to animate and open the Layer Styling toolbar.

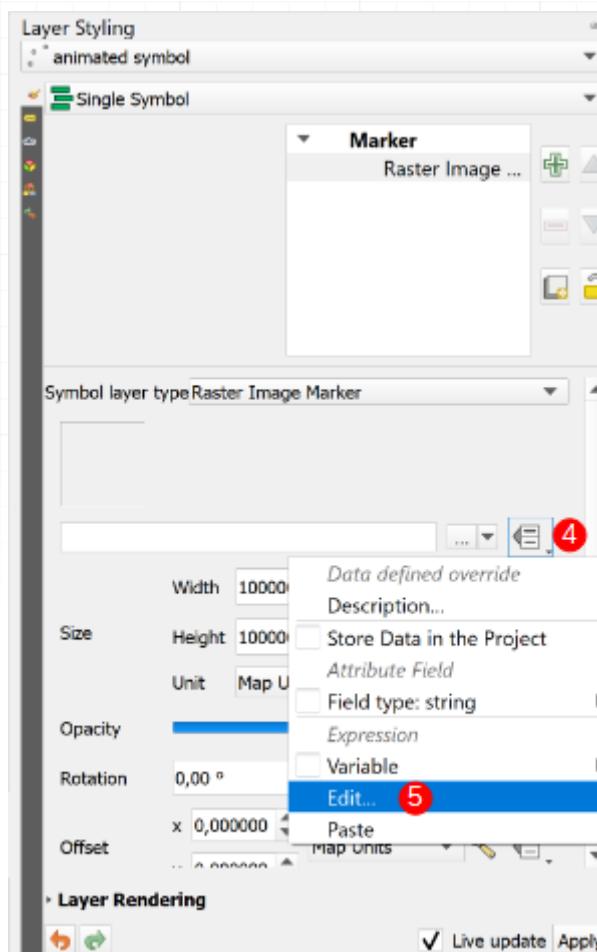
Note:

If you are using [QGIS 3.26](#) you can simply use the new animated point symbol, or if you're using an older version of [QGIS 3.x](#) follow the instructions below.

The layer should be a [Raster Image Marker](#). Once you have selected the image you want to use click on the QGIS Expressions dropdown menu ([4](#)) and click on [Edit](#) ([5](#)).

"Bring your QGIS projects to life!"





Use the [Code Snippets Section](#) for more in depth help. The example below works with the bird animation from earlier

"Bring your QGIS projects to life!"





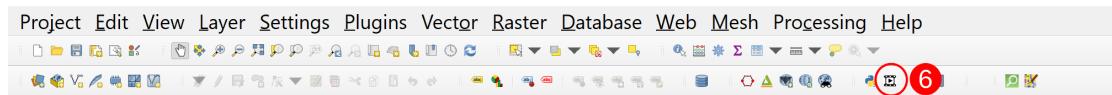
```
@project_home
||
'/bird/bird_00'
||
lpad(to_string(@frame_number % 9), 2, '0')
||
'.png'
```

Code:

```
@project_home
||
'/bird/bird_00'
||
lpad(to_string(@frame_number % 9), 2, '0')
||
'.png'
```

1. Open the Animation Workbench and configure your animation, choosing between the different modes and options.

Open the Workbench by clicking the **Animation Workbench** (6) icon in the Plugin Toolbar.

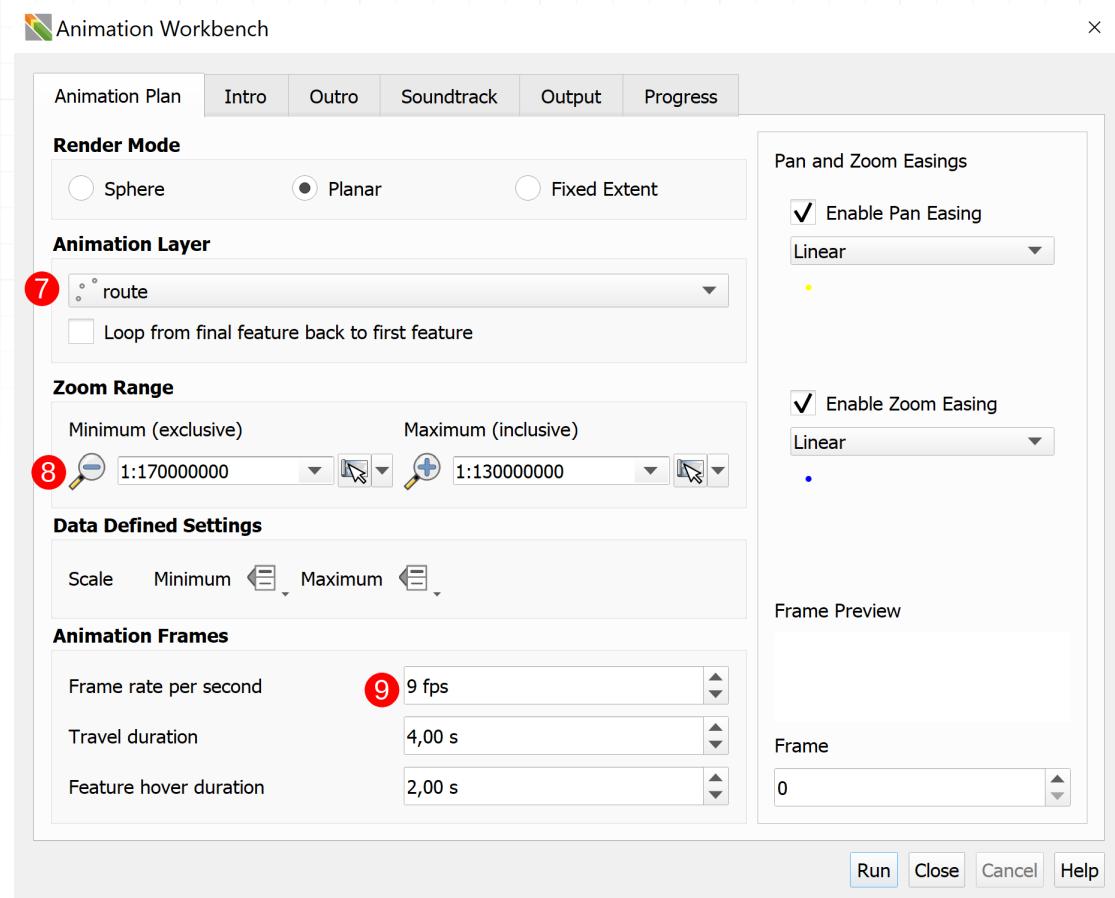


"Bring your QGIS projects to life!"





Configure the settings for your animation. The screenshot below is configured for the example presented in this section. The Animation Layer is selected as route (7) because that is the path the animation will fly along, the Zoom Range (8) was selected from the Map Canvas Extent, and the Frame rate per second (9) was set to 9 to match the bird animation.



Set your desired **Output Options** (10) Select a location for your output (11).

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

10 Output Options

Re-use cached images where possible

Output Format

Note that intro, outro and soundtrack are not generated for GIF.

Animated GIF Movie (MP4)

Output Resolution

720p (1280×720) 1080p (1920x1080) 4k (3840 x 2160) Map Canvas

File qgis_animation.gif 11 ...

Run Close Cancel Help

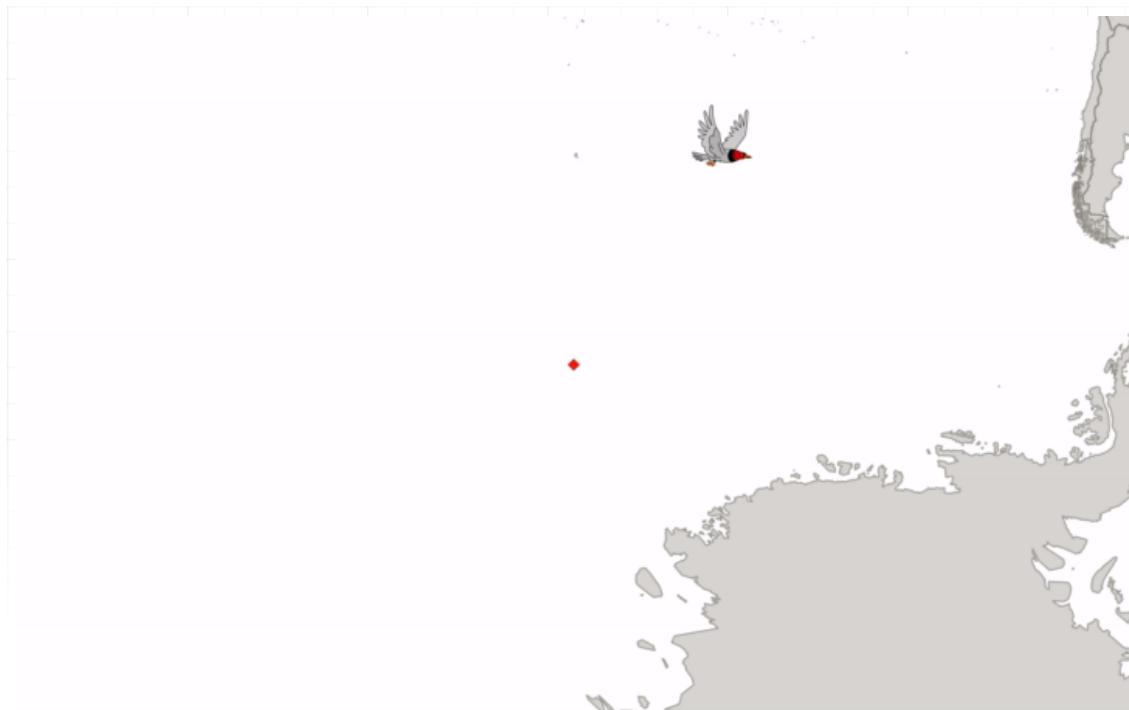
Note:

Refer to the [Workbench User Interface](#) Section for more information about what various settings and buttons accomplish.

2. Render your animation! Click Run and render your output. The output below is the output from the example.

"Bring your QGIS projects to life!"





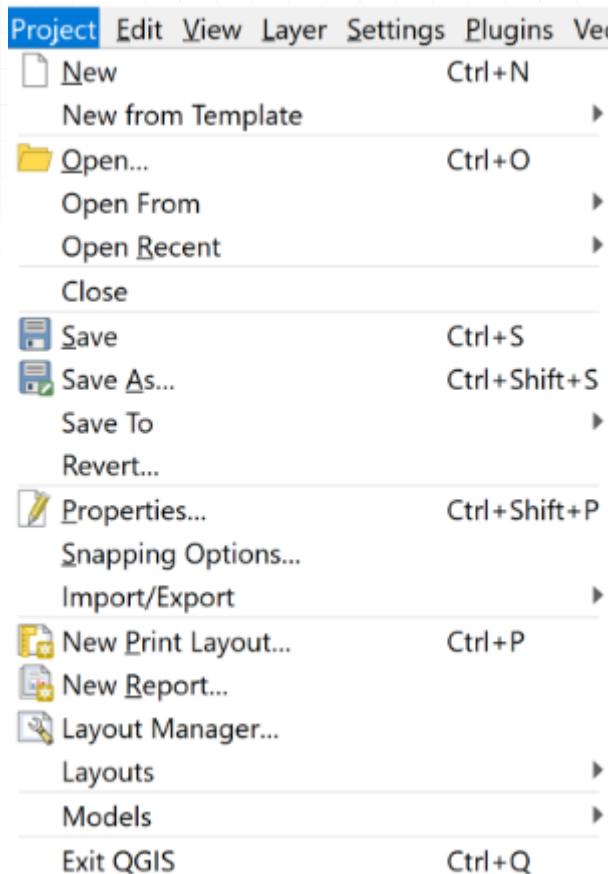
"Bring your QGIS projects to life!"



1 Manual

1.1 How to set up a project to work with the animation plugin

1. The first step for getting an output using the Workbench is to create a QGIS Project. Open QGIS and click on **Project → New**

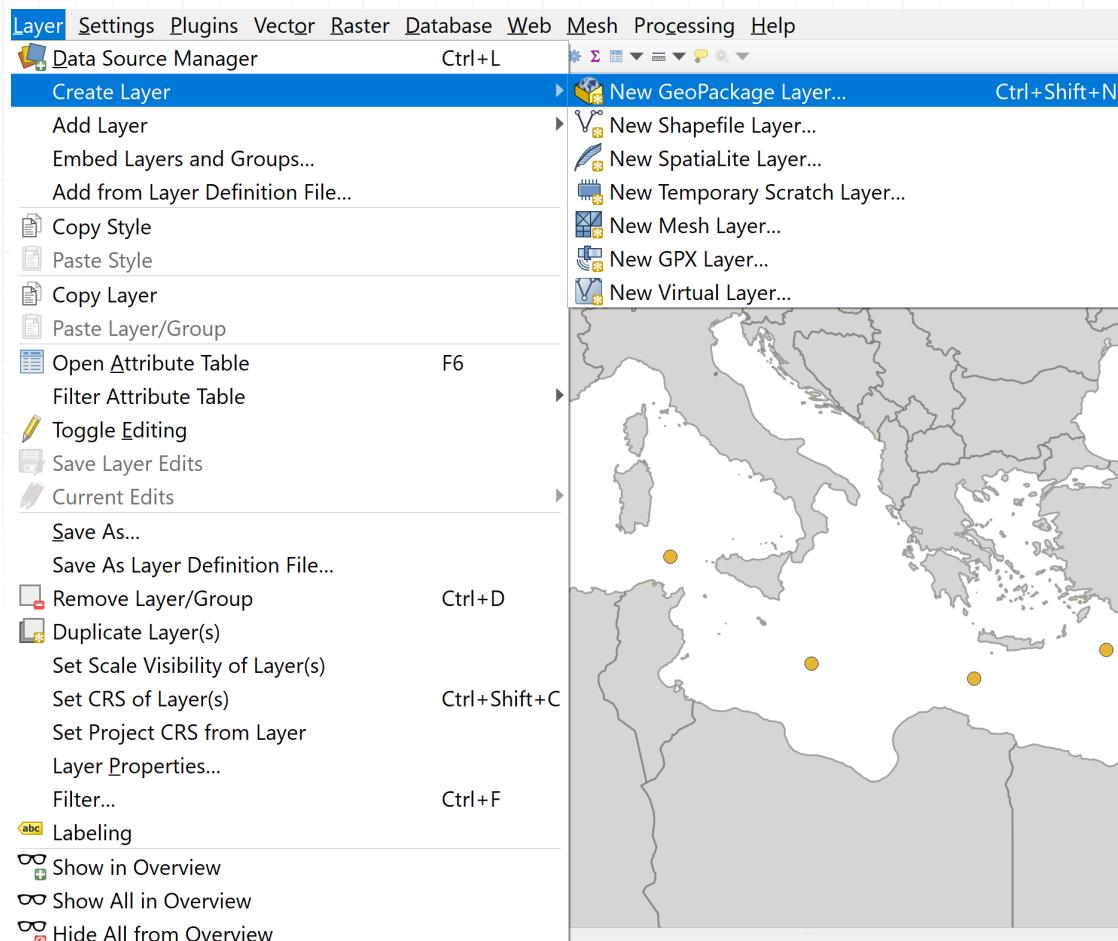


Next, add new layers to your project. You will want a few layers; one, or more, backing layer(s) (vector layers or XYZ Tiles), a layer for the workbench to follow, and one, or more, layer(s) of animated points. The example in this section only has one animated layer.

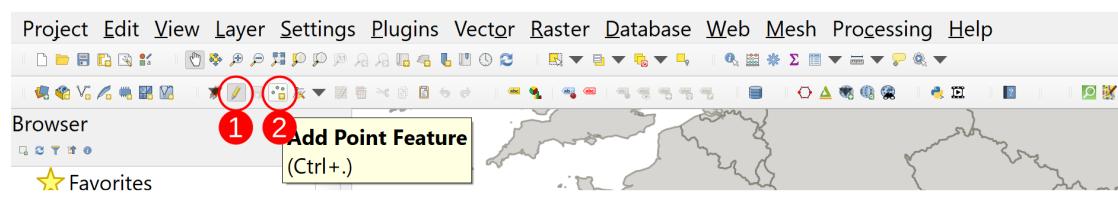
To add a layer, go to **Layer → Create Layer** and then select the type of layer you want to add. The example adds a point layer to a GeoPackage to make the project more portable.

"Bring your QGIS projects to life!"





Once you have added your layers you need to add features to the layers. This is done by selecting a layer and then clicking **Toggle Editing** (1) → **Add PointFeature** (2). Then click around on your map to add as few, or as many, features as you need.



The example project has four layers: two point layers (3) and two backing layers (4).

"Bring your QGIS projects to life!"





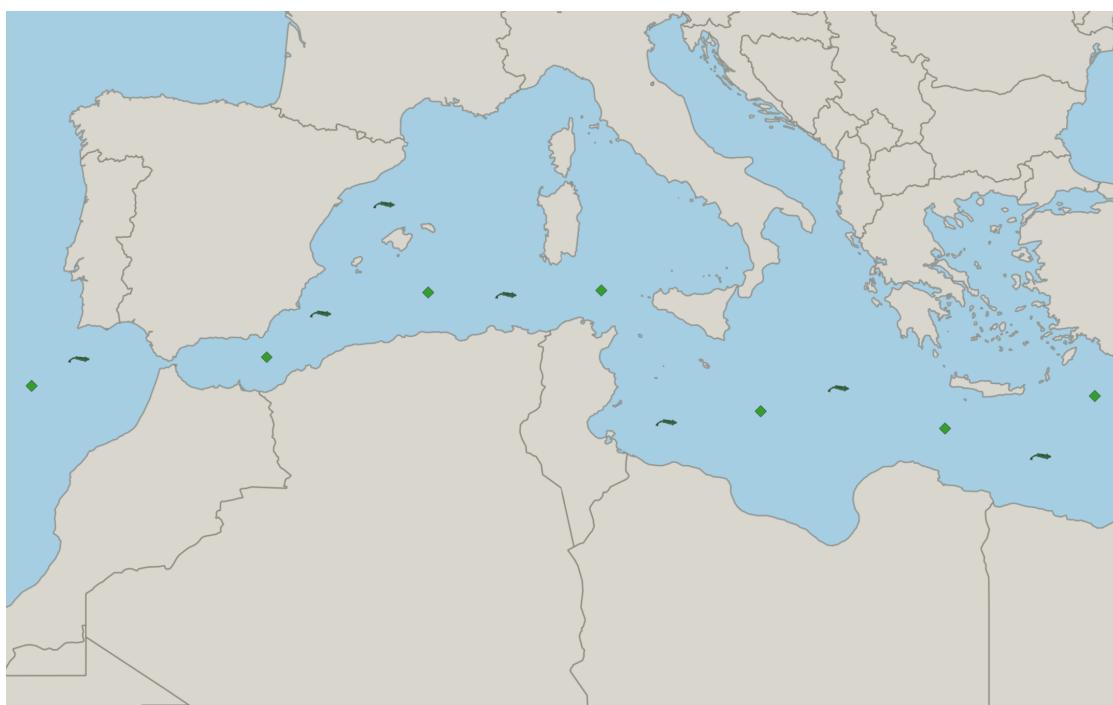
The screenshot shows the QGIS project manager with the following layers listed:

- ✓ fish (highlighted with a red box)
- ✓ route
- ✓ World Map (highlighted with a red box)
- ✓ ocean

Note:

A simple way to add a vector base layer is to type "world" into the coordinate textbox

Finally, style your layers to make your project look aesthetically pleasing. To style your layers you must select the layer you want to style and then using the Layer Styling toolbar, play around with the style of the layer until it suits you. A good practice is to have your backing layers as more muted colours and your desired features as more eye-catching colours.



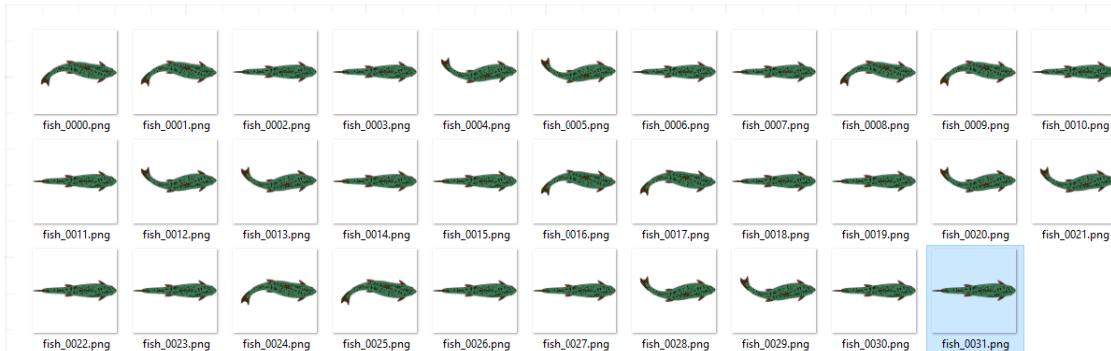
You now have a QGIS Project.

2. The next step is to choose which features you want to be animated.

Pick the layer (or layers) that you want to have animations. Then either find, or create, the animation for the layer. Make sure you have all the correct attribution for any animations you use. Below is an example of a simple fish animation split into its frames. The frames are repeated to slow down the animation's playback speed.

"Bring your QGIS projects to life!"





- Now use the QGIS Expressions system with the variables introduced by the Animation Workbench to define behaviours of your symbols during flight and hover modes of your animation. Select the layer you want to animate and open the Layer Styling toolbar.

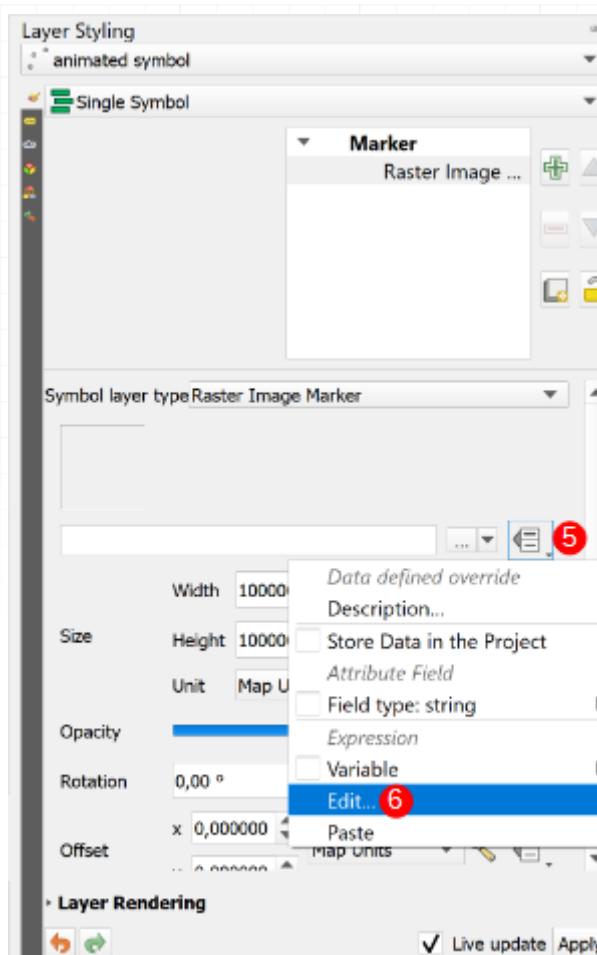
Note:

If you are using [QGIS 3.26](#) you can simply use the new animated point symbol, or if you're using an older version of [QGIS 3.x](#) follow the instructions below.

The layer should contain a [Raster Image Marker](#). Once you have selected the marker you want to use click on the QGIS Expressions dropdown menu ([5](#)) and click on [Edit](#) ([6](#)).

"Bring your QGIS projects to life!"





Note:

You can also make a marker move along a line relative to the frame of the animation.
Use the [Code Snippets Section](#) for more in-depth help.

The example below works with the animation from earlier.

"Bring your QGIS projects to life!"





```
@project_home
||
'/fish/fish_00'
||
lpad(to_string( @frame_number % 32), 2, '0')
||
'.png'
```

 **Code:**

```
@project_home
||
'/fish/fish_00'
||
lpad(to_string( @frame_number % 32), 2, '0')
||
'.png'
```

3. Configure your animation

After animating your markers it's time to configure your animation. Open the Animated Workbench and begin choosing between the different modes and options.

Open the Workbench by clicking the **Animation Workbench** (7) icon in the Plugin Toolbar.

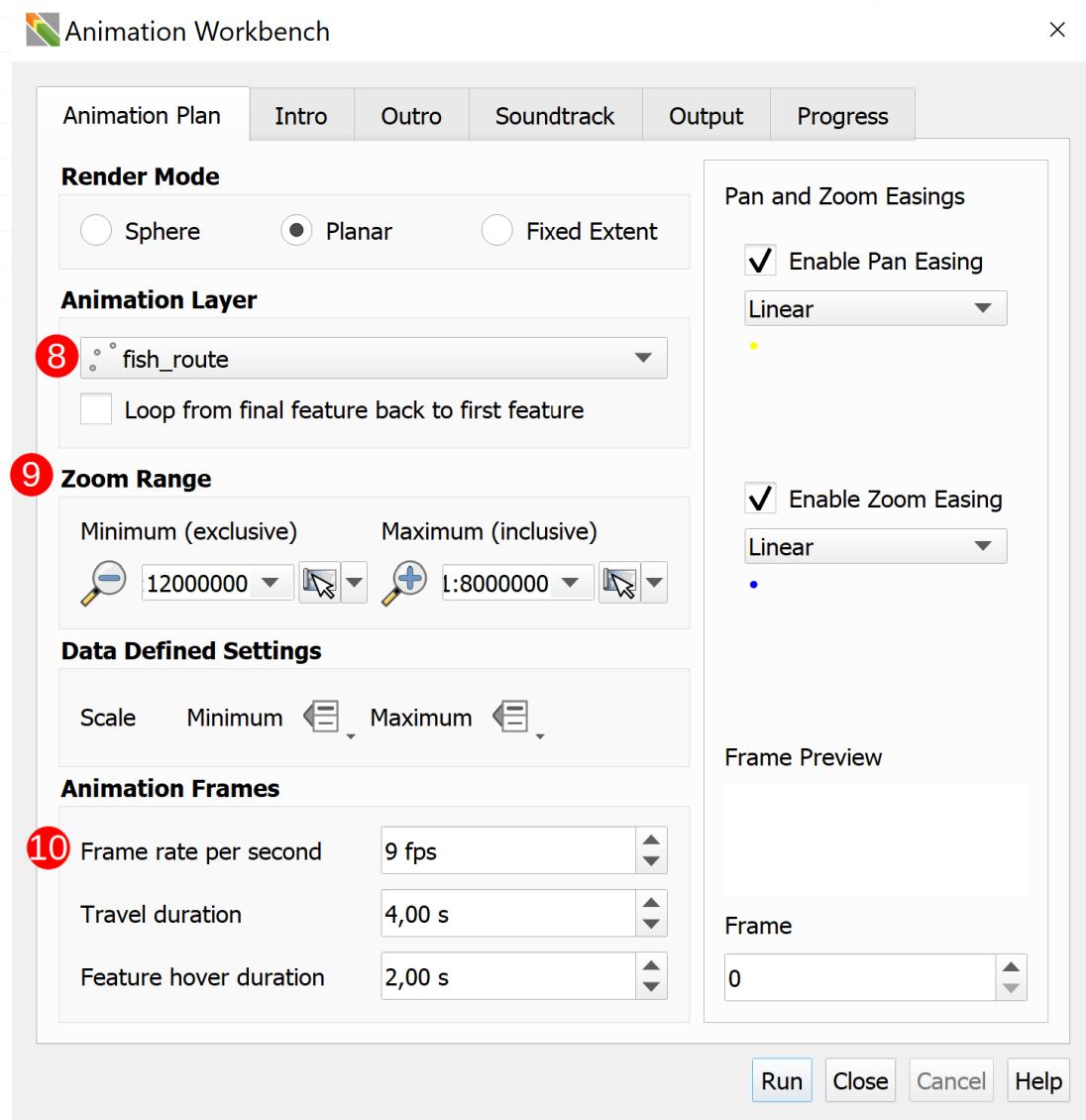


"Bring your QGIS projects to life!"





Configure the settings for your animation. The screenshot below is configured for the example presented in this section. The Animation Layer is selected as **route** (8) because that is the path that the output animation will fly along. The Zoom Range (9) was selected from the Map Canvas Extent, and the Frame rate per second (fps) (10) was set to match the number of frames of the animated markers so that they will play nicely in the output. The other settings were selected as a personal choice.



Select the Output Resolution (11) and a location for your output by clicking on the ellipsis (three dots) or by typing in the desired file path (12).

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

Output Options

Re-use cached images where possible

Output Format

Note that intro, outro and soundtrack are not generated for GIF.

Animated GIF Movie (MP4)

Output Resolution

11 720p (1280×720) 1080p (1920x1080) 4k (3840 x 2160)

File qgis_animation.gif 12 ...

Run Close Cancel Help

Note:

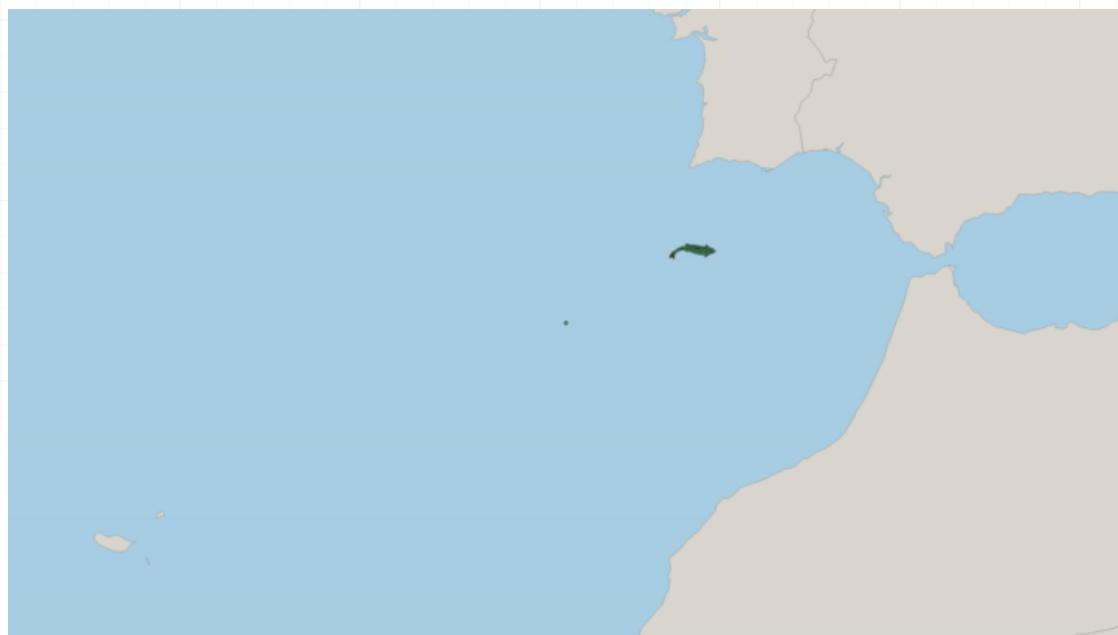
Refer to the [Workbench User Interface](#) section for more information about what various settings and buttons accomplish.

"Bring your QGIS projects to life!"





4. Render your animation! Click **Run** and render your output. The output below is the output from the example.



"Bring your QGIS projects to life!"





1.2 The Workbench User Interface

1.2.1 Animation Plan

"Bring your QGIS projects to life!"





Animation Workbench

Animation Plan Intro Outro Soundtrack Output Progress

1 Render Mode
 Sphere Planar Fixed Extent

2 Animation Layer
route_test
 Loop from final feature back to first feature

3 Zoom Range
Minimum (exclusive) Maximum (inclusive)
Scale 1:10000000 Scale 1:8000000

4 Data Defined Settings
Scale Minimum Maximum

5 Animation Frames
Frame rate per second: 30 fps
Travel duration: 2,00 s
Feature hover duration: 2,00 s

Pan and Zoom Easings
 Enable Pan Easing Linear

Enable Zoom Easing Linear

Frame Preview

Frame
0

Run Close Cancel Help

"Bring your QGIS projects to life!"

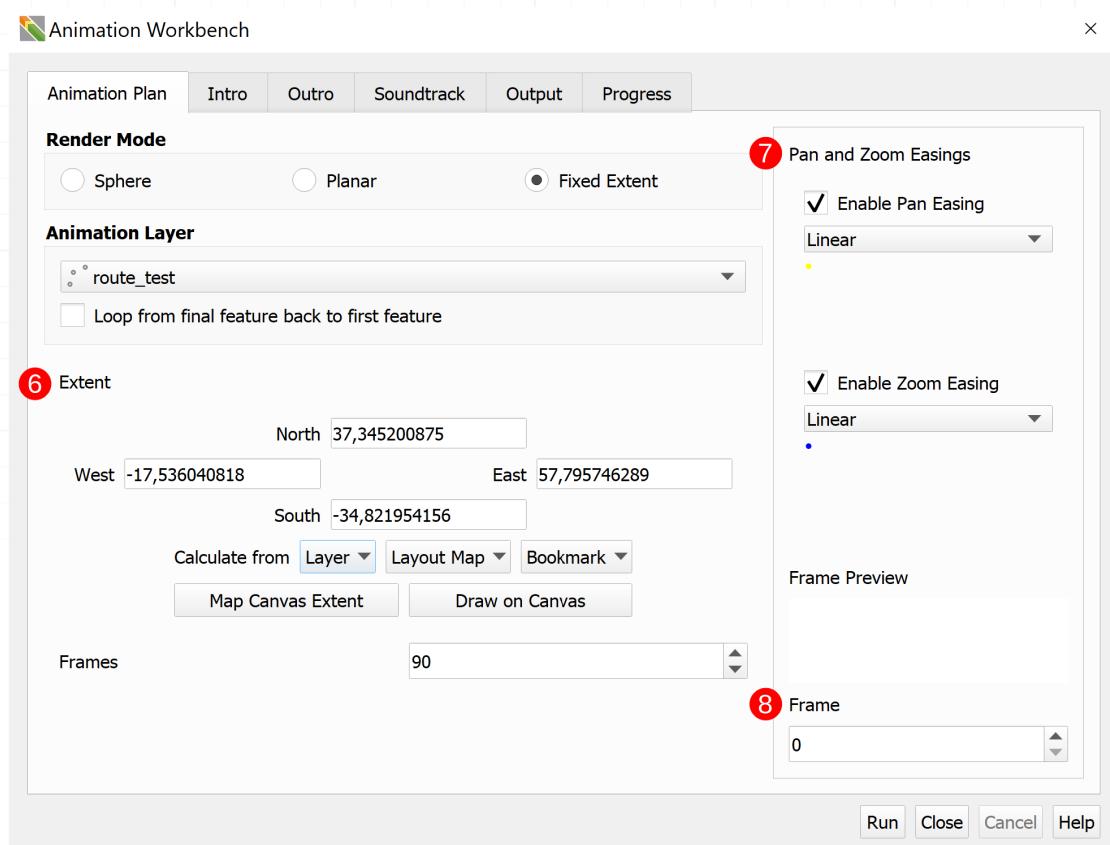




- Render Modes (**1**): These determine the behaviour and type of animation
 - **Sphere** : The coordinate reference system (CRS) will be manipulated to create a spinning globe effect. Like Google Earth might do, but with your own data and cartography.
 - **Planar** : The coordinate reference system (CRS) will not be altered, but the camera will pan and zoom to each point. It lets you move from feature to feature on a flat map, pausing at each if you want to.
 - **Fixed extent** : The frame of reference stays the same and you can animate the symbology within that scene.
- Animation Layer (**2**):
 - **Dropdown menu** : This allows you to select which map layer you want the animation to follow.
 - **Loop from final feature back to first feature** : allows for a seamlessly looping output GIF or movie(MP4).
- Zoom Range (**3**): The scale range that the animation should move through.
- Minimum (exclusive): The zenith (highest point) of the animation when it zooms out while travelling between points, i.e. the most "zoomed out".
- Maximum (inclusive): The scale (zoom level) used when we arrive at each point, i.e. the most "zoomed in".
- Data defined settings (**4**)
- Scale
 - Minimum: User-defined minimum scale
 - Maximum: User-defined maximum scale
- Animation Frames (**5**)
- Frame rate per second (fps): When writing to video or gif, how many frames per second to use.
- Travel Duration: This is the number of seconds that the animation will take during animation from one feature to the next.
- Feature Hover duration: This is the number of seconds that the animation will hover over each feature.

"Bring your QGIS projects to life!"





- Extent (6):
 - Can be manually entered using North, East, South, and West coordinates as limits.
 - Can be calculated from a map layer, the layout map, or a bookmark.
 - Can be set to match the Map Canvas Extent
 - Can be set as a rectangular extent using the **Draw on Canvas** feature.
- Pan and Zoom Easings (7)
 - What are Easings: Easings are transitions from one state to another along a smooth curve. A user can specify the shape of the curve used.
 - Pan Easings (XY): The pan easing will determine the motion characteristics of the camera on the X and Y axis as it flies across the scene (i.e. how it accelerates or decelerates between points)
 - Zoom Easing (Z): The pan easing will determine the motion characteristics of the camera on the Z axis as it flies across the scene (i.e. how the camera zooms in and out of the points)
- Frame previews (8): A preview of what each frame of the animation will look like. A user can decide which **Frame** to view.

1.2.2 Intro Tab

Edit the intro section of the generated movie here.

"Bring your QGIS projects to life!"





Animation Workbench

Intro

Outro

Soundtrack

Output

Progress

Intro Media: Add images and movies for the intro section of the generated movie here. For images you can set a duration for each (in seconds). You can drag and drop items in the list to change the play order.

Media

Total duration for all media:

1 +

2 -

4

Details

3 Duration 0s ▲ ▼

Run Close Cancel Help

• Media: List of the various images or movies selected for the intro section. You can drag and drop items in the list to change the play order.
• Add Media (Plus sign) (1): Add images or movies
• Remove Media (Minus sign) (2): Remove images or movies
• Duration (3): For images, you can set a duration for each image (in seconds).
• Preview Frame (4): This shows what the media will look like.
• Details: Provides details about where the media is stored on your computer.

1.2.3 Outro Tab

Edit the outro section of the generated movie here.

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

Outro Media: Add images and movies for the outro section of the generated movie here.
For images you can set a duration for each (in seconds).
You can drag and drop items in the list to change the play order.

Media Total duration for all media:

1 2 3 Duration 0s

Details

Run Close Cancel Help

- Media: List of the various images or movies selected for the outro section. You can drag and drop items in the list to change the play order.
- Add Media (Plus sign) (**1**): Add images or movies
- Remove Media (Minus sign) (**2**): Remove images or movies
- Duration (**3**): For images, you can set a duration for each image (in seconds).
- Preview Frame (**4**): This shows what the media will look like.
- Details: Provides details about where the media is stored on your computer.

1.2.4 Soundtrack Tab

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack **Soundtrack** Output Progress

Music: Add sound files (.mp3 or .wav) to play during the generated movie.
The cumulative length of your soundtracks should be as long or longer than your movie, including the intro/outro sections.
If the soundtrack is longer than the movie it will be truncated when the movie ends.
You can drag and drop items in the list to change the play order.

Media Total duration for all media:

1 2

Details 3 Duration 0s

Run **Close** **Cancel** **Help**

- Media: List of the various sound files (.mp3 or .wav) to play during the generated movie. You can drag and drop items in the list to change the play order.
- Add Media (Plus sign) (**1**): Add sound files (.mp3 or .wav) to play during the generated movie.
- Remove Media (Minus sign) (**2**): Remove sound files (.mp3 or .wav)
- Duration (**3**): The cumulative length of your soundtracks should be as long, or longer, than your movie, including the intro/outro sections. If the soundtrack is longer than the movie it will be truncated (shortened) when the movie ends.
- Details: Provides details about where the media is stored on your computer.

1.2.5 Output

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

Output Options

1 Re-use cached images where possible

Output Format

Note that intro, outro and soundtrack are not generated for GIF.

2 Animated GIF 3 Movie (MP4)

Output Resolution

720p (1280x720) 1080p (1920x1080) 4k (3840 x 2160) Map Canvas

File **qgis_animation.mp4** 5 ...

Run **Close** **Cancel** **Help**

- Output Options: Select which output format you would like. Regardless of the format chosen, a folder of images will be created, one image per frame.
- Re-use cached Images (1): This will not erase cached images on disk and will resume processing from the last cached image.
- Animated GIF (2): For this export to work, you need to have the ImageMagick 'convert' application available on your system.
- Movie (MP4) (3): For this option to work, you need to have the 'ffmpeg' application on your system.
- Output Resolution (4): Allows a user to specify one of four image resolutions for the output animation. The numbers in brackets for the first three options represent the width and height of the output in pixels (i.e. width x height), and the fourth option matches the output's size to the size of the **Map Canvas** on the screen.
- File selection (ellipsis) (5): This lets a user select the location where the output will be stored.

"Bring your QGIS projects to life!"

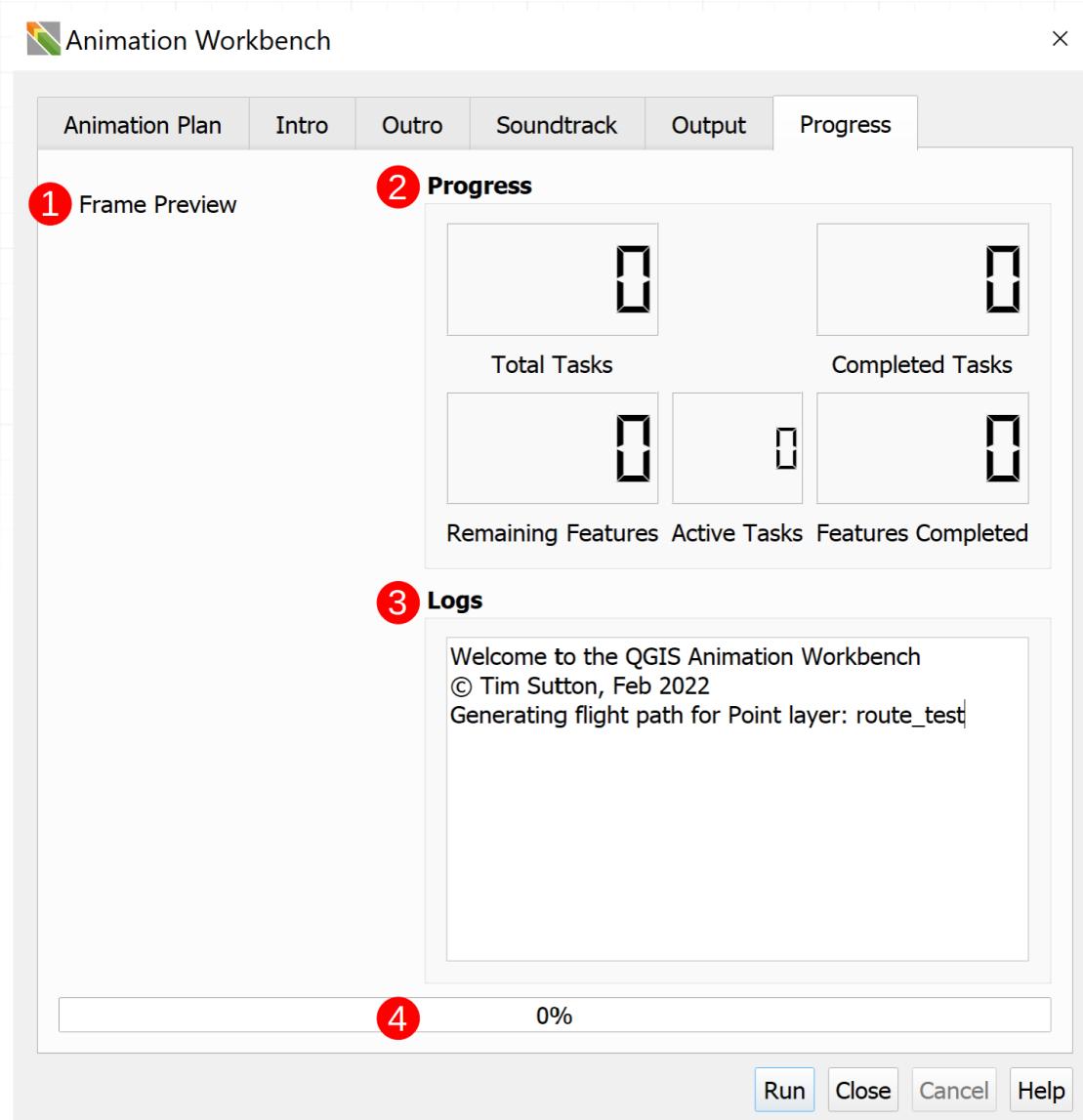




1.2.6 Progress

"Bring your QGIS projects to life!"





- Frame Preview (1): A preview of what each frame of the animation will look like. It changes automatically as the workbench runs.
- Progress (2): This provides a detailed look at what is happening while the workbench runs.
- Total Tasks: This number represents the total number of frames that will be generated by the workbench.
- Completed Tasks: The number of tasks that have completed being processed.
- Remaining Features: The number of features from your animation layer that still need to be processed.
- Active Tasks: The number of tasks (threads) currently being run by the workbench
- Features Complete: The number of tasks that have been processed by the workbench.
- Logs (3): A detailed list of what steps the workbench is doing (a record of processing)
- Progress Bar (4): A visual representation of the workbench's progression as a percentage.

"Bring your QGIS projects to life!"





1.2.7 Other Buttons

- **Run** : Starts the process of getting an output from the workbench. It is greyed out until a user provides a destination for the output file.
- **Close** : Closes the workbench.
- **Cancel** : Ends the workbench processing at whatever point it has reached when the button is pressed.
- **Help** : Opens a link to the Animation Workbench documentation.

"Bring your QGIS projects to life!"





1.3 What is the Workbench doing?

• What does the workbench do?

The workbench creates animations from QGIS by generating multiple static frames (images) and then combining those frames into an animation. The user tells QGIS how the frames should change from one to the other. In **QGIS 3.26** and later the animated markers allow markers to be animated without the use of the expressions system.

• How do the animated markers work?

In the code snippet below, the user tells QGIS that as the frame count increments by one the

Raster Image Marker should change to the next image in the sequence.

```
@project_home
||
'/fish/fish_00'
 ||
lpad(to_string( @frame_number % 32), 2, '0')
 ||
'.png'
```

py

```
@project_home
||
'/fish/fish_00'
 ||
lpad(to_string( @frame_number % 32), 2, '0')
 ||
'.png'
```

The user specifies the path of the image (**@project_home/fish/fish_00**). Then the **lpad(to_string(@frame_number % 32), 2, '0')** tells QGIS to convert the frame number to a string and then modulus the number of frames by the number of animation frames (**32**) (i.e.

"Bring your QGIS projects to life!"





QGIS divides the number of frames by 32 and then repeats the sequence when the remainder is zero). The

`2` and `'0'` in the snippet tell QGIS to pad the `/fish/fish_00` with two zeroes at the end.

Finally the `'.png'` tells QGIS the type of file to finish off the path.

- **Frame Output location on Windows**

For users on a Windows machine who are interested in seeing the frames before they are combined into an animation (GIF or movie) you can find them by going to "C:

`\Users\Username\AppData\Local\Temp\animation_workbench-0000000000.png`". Bear in mind that AppData is a hidden file, so it's preferable to not make changes unless explicitly told otherwise.

- **Frame Output on Linux**

The frames should be in your `/tmp` directory.

"Bring your QGIS projects to life!"





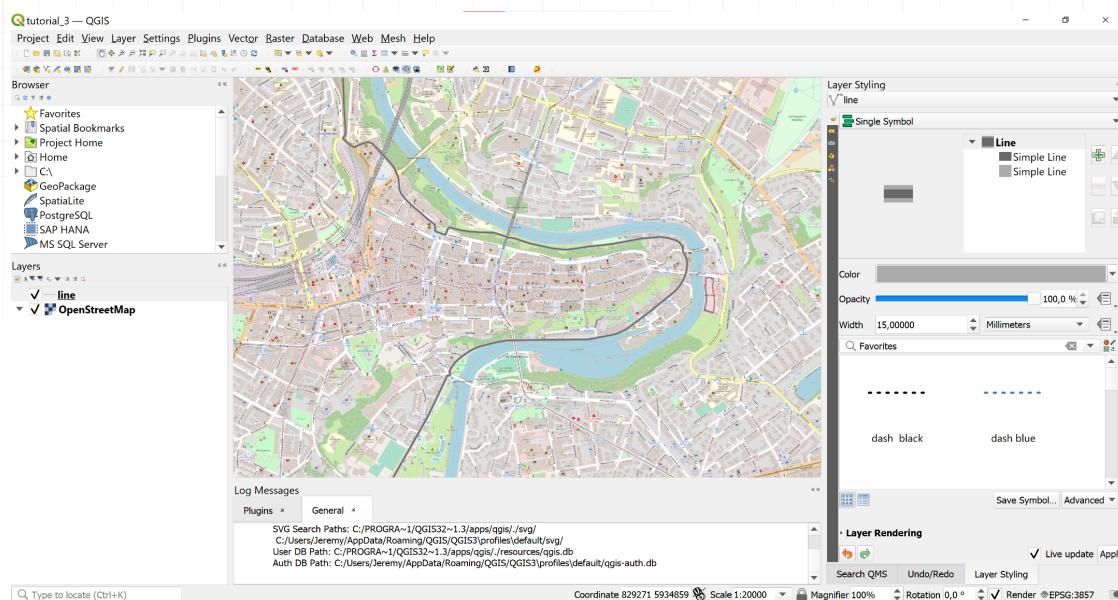
1 Tutorials

1.0.1 Tutorial 1: Point Along A Line

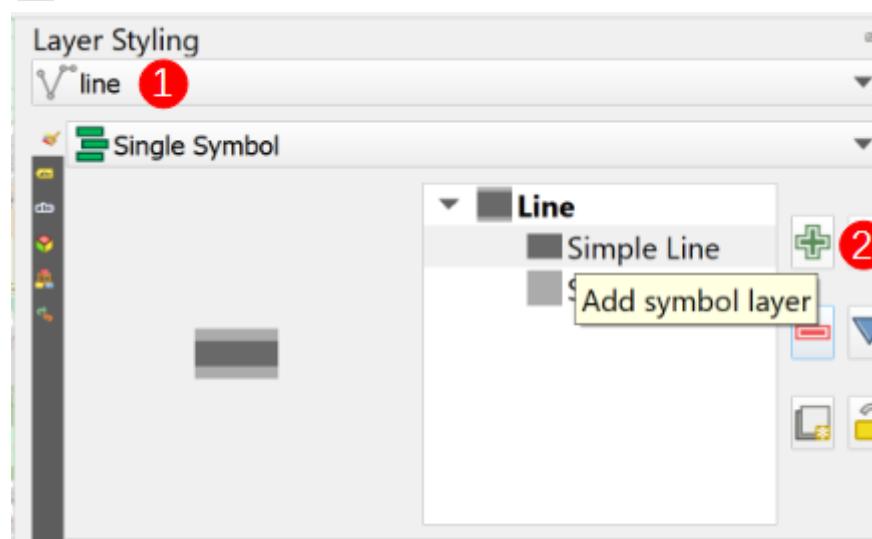
This tutorial introduces the concept of moving a point along a line within your animated map.

1. Download and extract the [Required Tutorial Zip Folder](#)

2. Open the **tutorial_1.qgz** project file that is in the folder. When you first open it you see something like this:



3. Select the premade **line** layer (1), and click on the **Add Symbol Layer** (green plus symbol) button (2) to it.

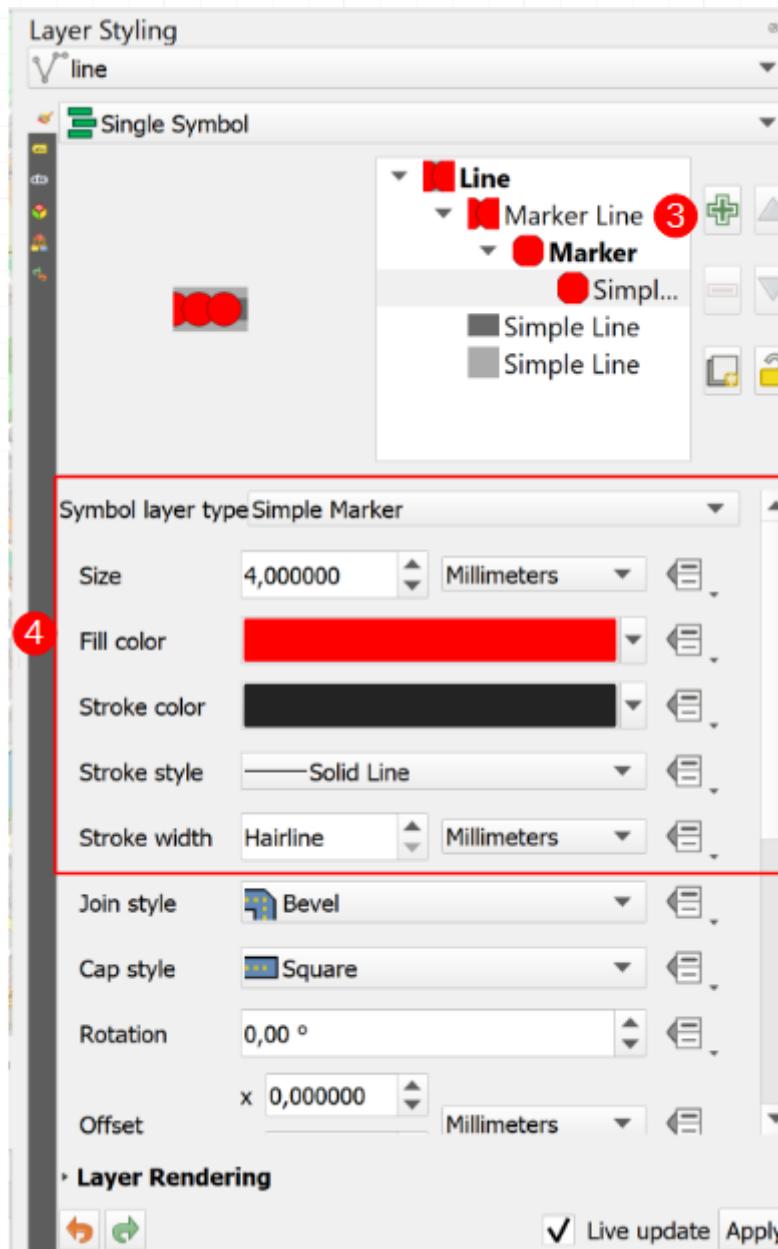


"Bring your QGIS projects to life!"





Change the new **Symbol Layer** (3) type to marker line and then style it (4) so that it is more visible.



4. Change the **Symbol Layer's** settings so that the point is only on the **first vertex** (5) and not at equidistant intervals.

Change the offset along the line to be **Percentage** (6).

"Bring your QGIS projects to life!"





Symbol layer type **Marker Line**

Marker placement

With interval 3,0 Store Data in the Project

On inner vertices Attribute Field

On last vertex Field type: int, double, string

On first vertex Expression

On central point Variable

On central point Edit... **8**

On every curve part Paste

Assistant...

Offset along line 0,0000 7

Rotate marker to follow line direction

Place on every part

Layer Rendering

Click the **Dropdown Menu** (7) → **Edit...** (8) and then add the following code snippet

"Bring your QGIS projects to life!"





Symbol layer type **Marker Line**

Marker placement **Data defined override**

With interval **3,0**

On inner vertices

On last vertex

On first vertex

On central point

On central point c

On every curve p

Store Data in the Project

Attribute Field

Field type: int, double, string

Expression

Variable

Edit... 8

Paste

Assistant...

Offset along line **0,0000** Percentage **7**

Rotate marker to follow line direction

Place on every part

Layer Rendering

Code:

```
-- Point Along Line Code Snippet
(@current_hover_frame/@hover_frames) * 100
```

```
-- Point Along Line Code Snippet
(@current_hover_frame/@hover_frames) * 100
```

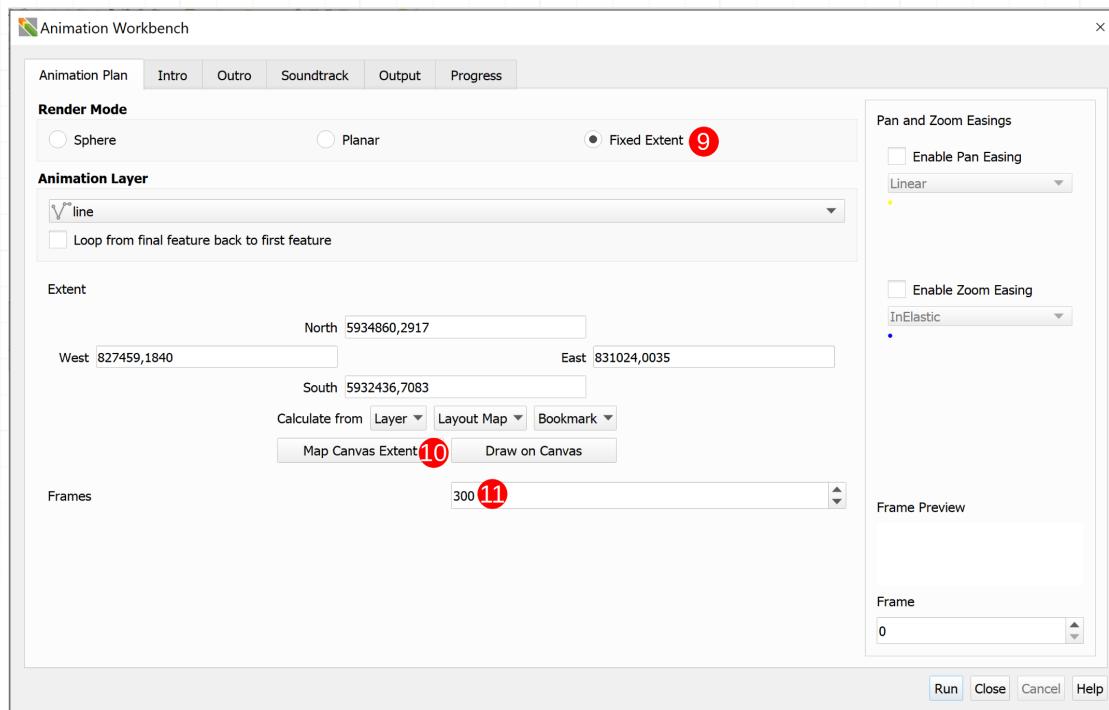
The snippet tells QGIS how far along the line (as a percentage of the line length) to render the point in each frame.

5. Open the Workbench and select **Fixed Extent** (9).

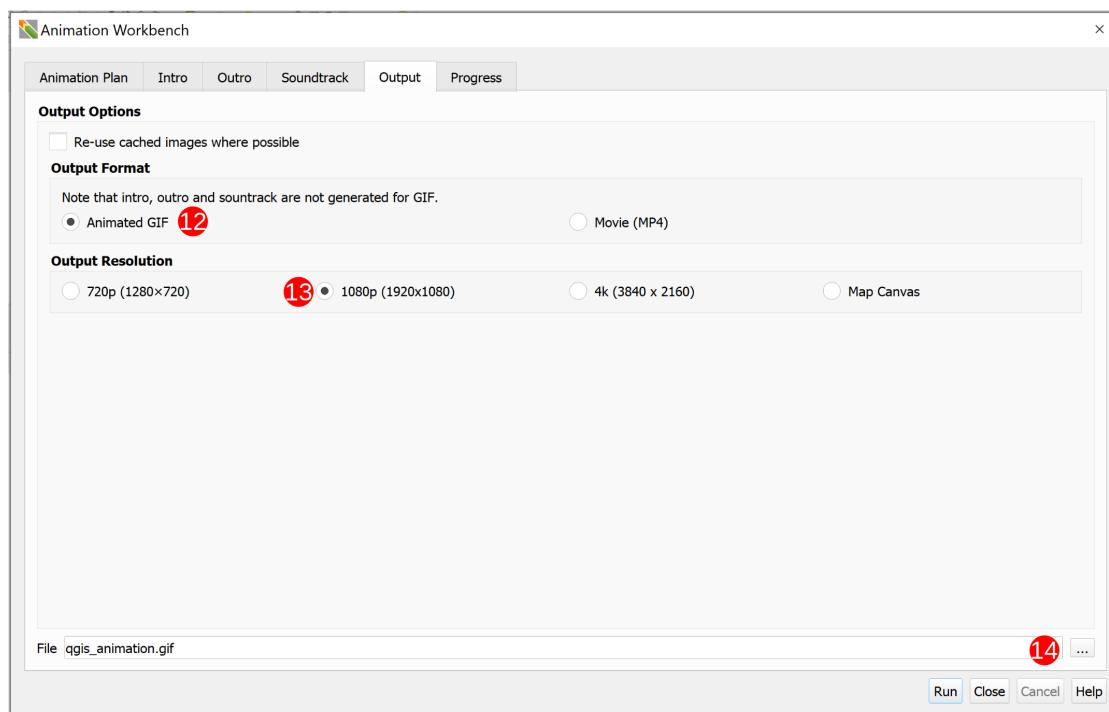
Click on **Map Canvas Extent** (10) and set the the **Frames** to 300 (11) (for a 10 second output at 30 frames per second).

"Bring your QGIS projects to life!"





6. Skip over the **Intro**, **Outro**, and **Soundtrack** tabs. In the **Output** tab, set the output format (12) and resolution (13), and set the output location's path (14).

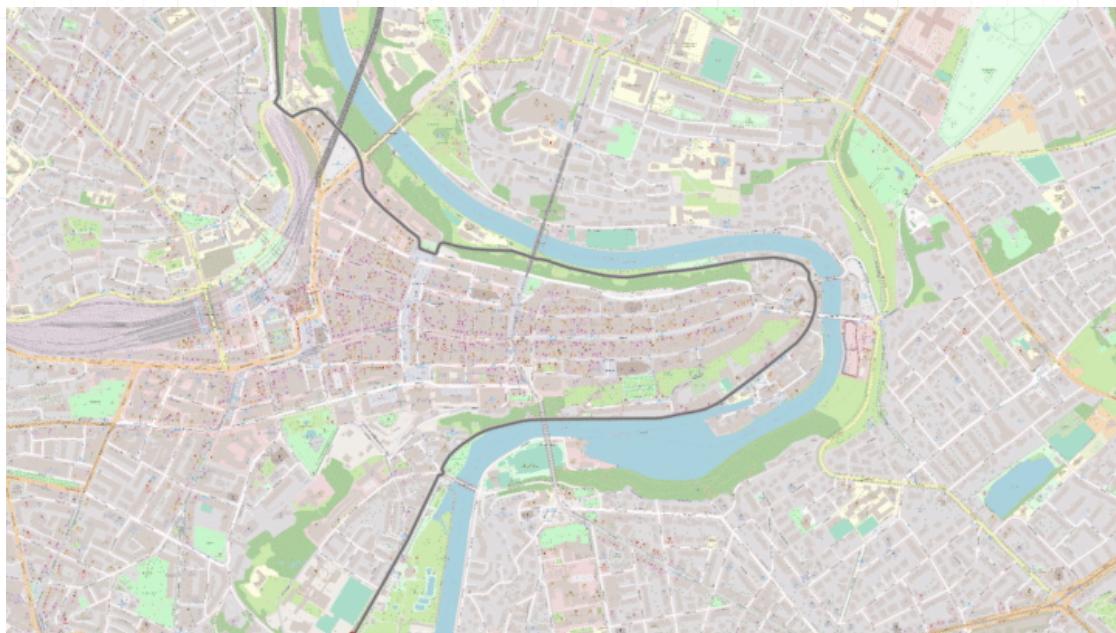


"Bring your QGIS projects to life!"





7. Click **Run** and render your output.



After this tutorial you should have a better idea of how to make a point move along a line. An expansion to this example would be to make the moving point a dynamically changing marker (like the markers in tutorial 1). Go have fun!

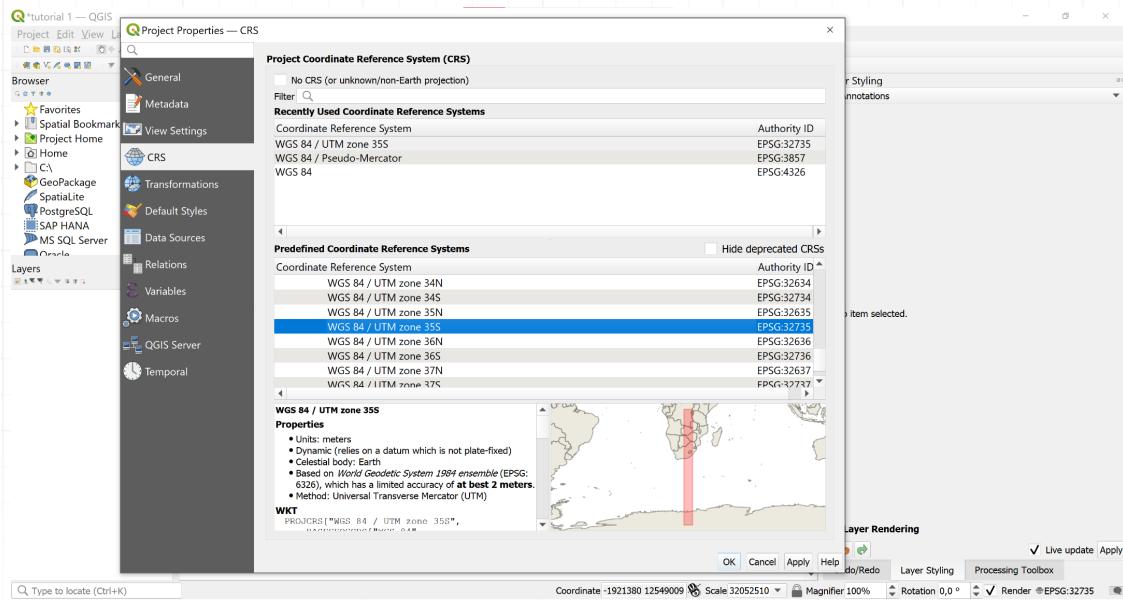
1.0.2 Tutorial 2: Basic Dynamically Changing Markers

This tutorial aims to show you the basics of creating, and animating, a static layer to use with the Animation Workbench. There are three pre-made layers to allow the main focus of the tutorial to be on the Animation Workbench and not on QGIS as a whole.

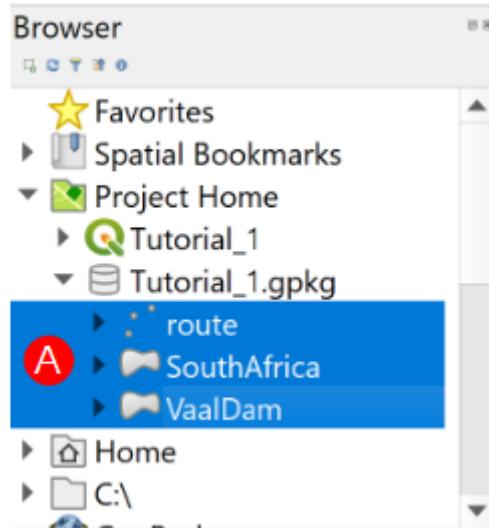
1. Download and extract the [Required Tutorial Zip Folder](#)
2. Open the **tutorial_2.qgz** project file that is in the folder.
3. Set the CRS of your project to **WGS84/UTM zone 35S (EPSG: 32735)**.

"Bring your QGIS projects to life!"





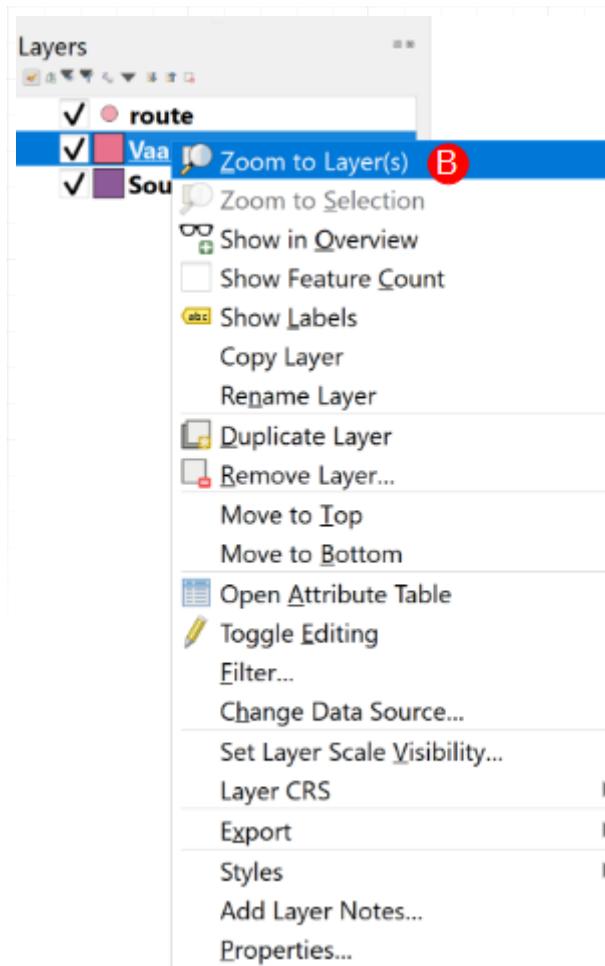
4. In the **Browser**, expand the **tutorial_2.gpkg** and add the three pre-made layers (VaalDam, SouthAfrica, and route) (A) to your project.



5. In the **Layers** Panel, arrange the layers in the following order: **route**, **VaalDam**, **SouthAfrica**. Then right-click on the **VaalDam** layer and **Zoom to Layer(s)** (B)

"Bring your QGIS projects to life!"



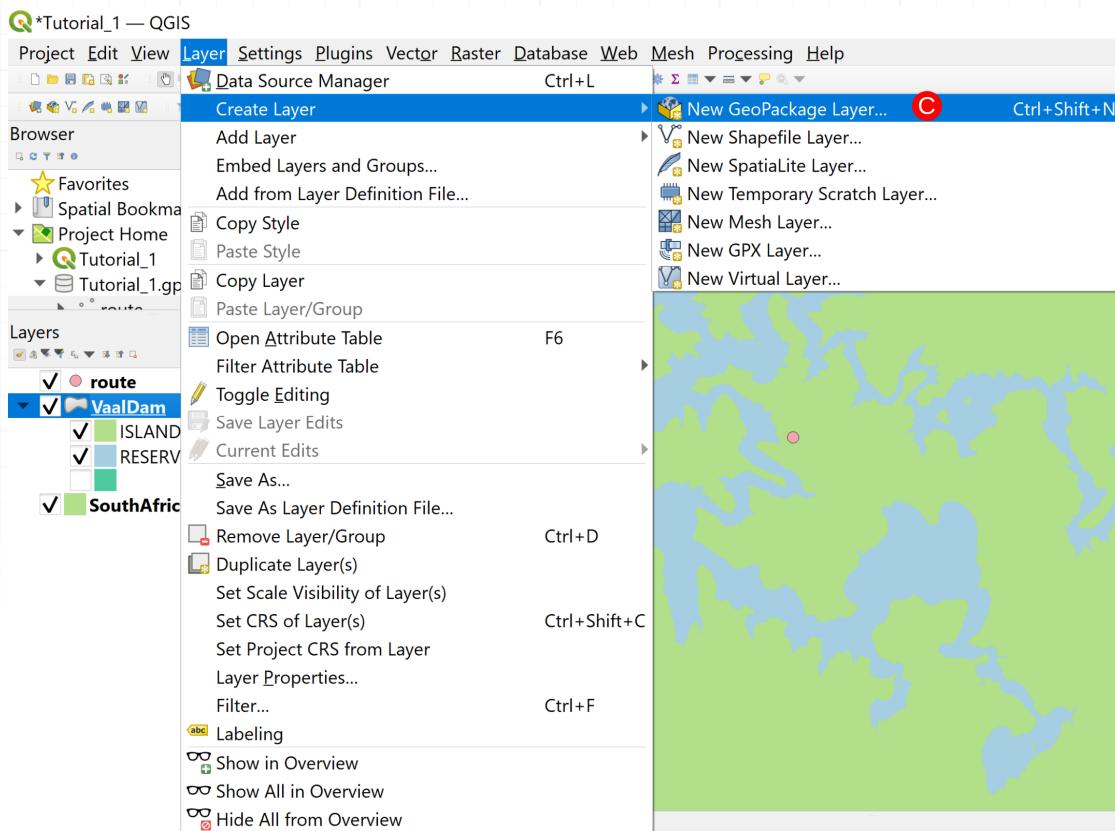


Style the three layers to your preferred style.

6. Now create a new layer in the **tutorial_2.gpkg** by clicking **Layer** → **Create Layer** → **New GeoPackage Layer...** (c).

"Bring your QGIS projects to life!"

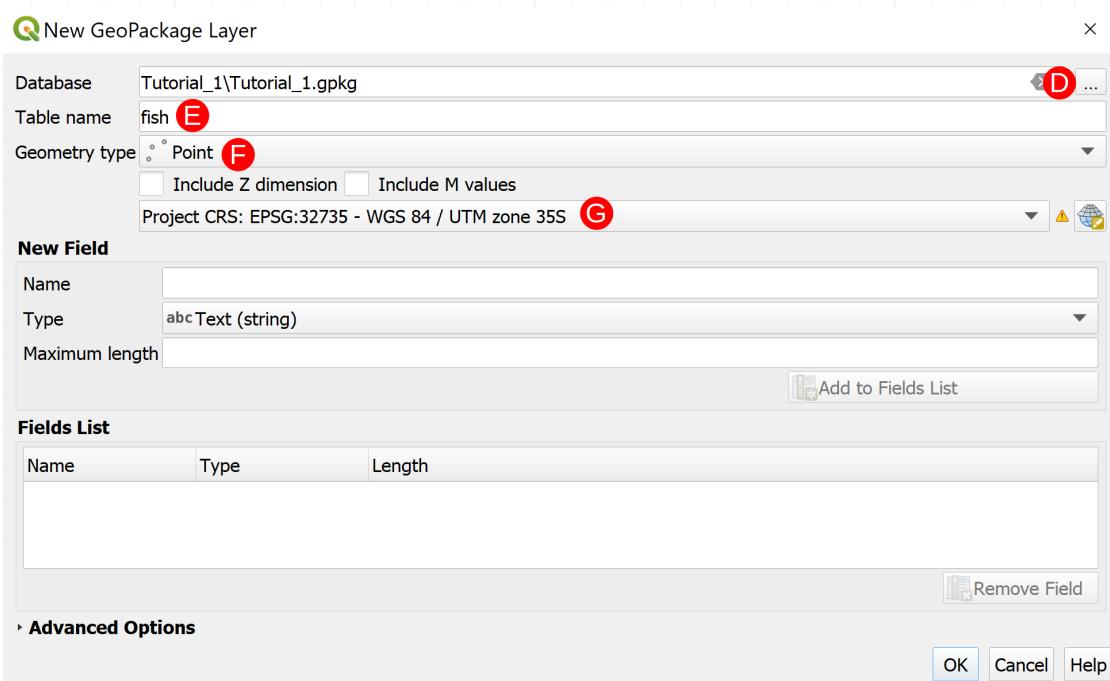




Click on the **Ellipsis** (**D**), navigate to and select the **tutorial_2.gpkg**, and click **Save**. Change the Table name to **fish** (**E**), set the Geometry type as **Point** (**F**), and change the CRS to match the **Project CRS** (**G**).

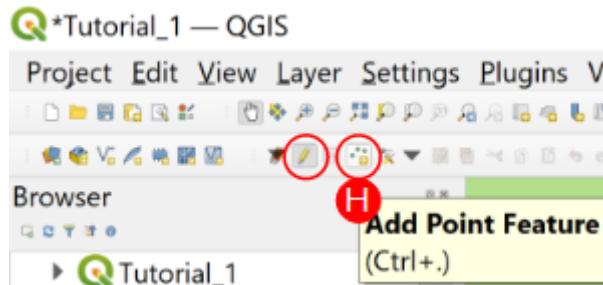
"Bring your QGIS projects to life!"





Click on **OK** and then click **Add New Layer** on the window that pops up.

7. Select the **fish** layer and then click on **Toggle Editing** → **Add Point Feature** (**H**).



Add a few points wherever you feel they should go (Hint: This is a fish layer so adding them above the dam layer would be best). Don't worry about naming the points, just add them.

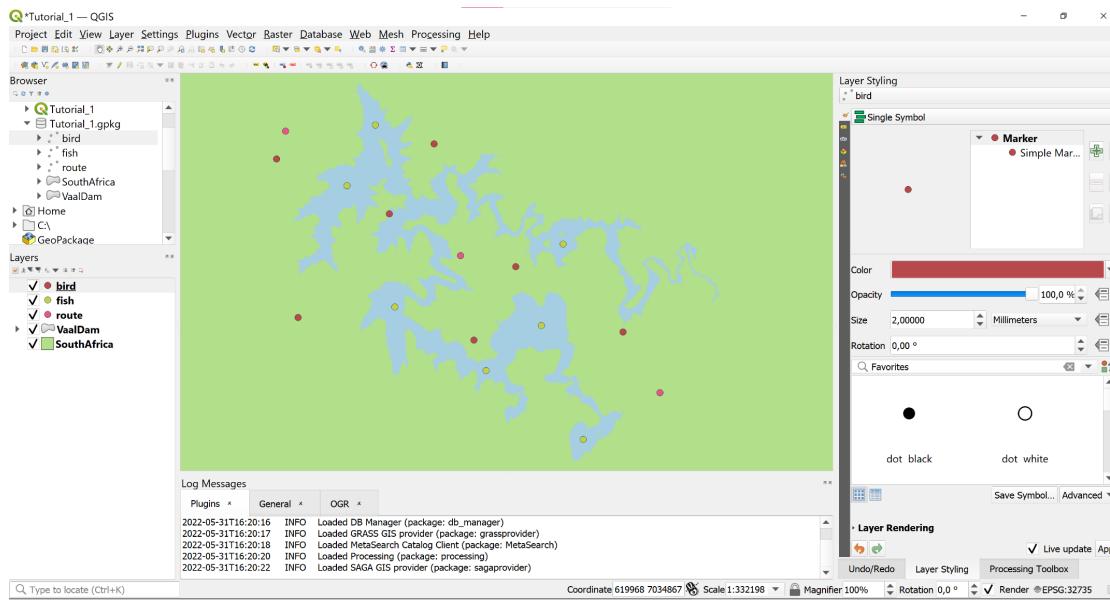
"Bring your QGIS projects to life!"





Save your changes by clicking on **Save Layer Edits** just next to the **Toggle Editing** button.
Then stop editing the layer.

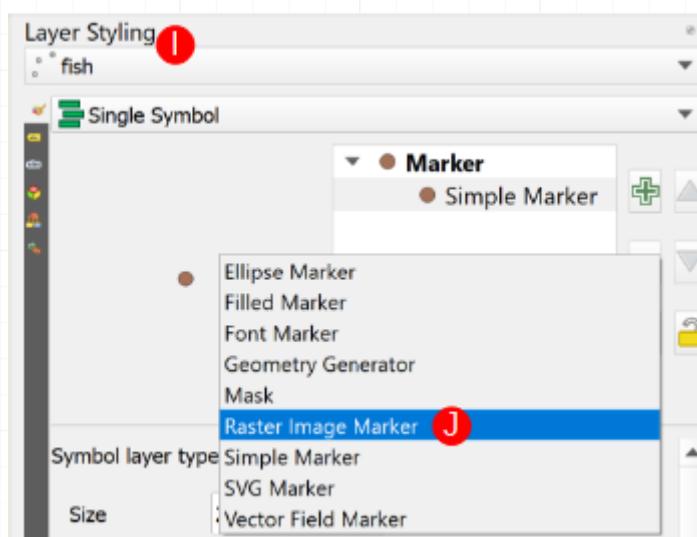
8. Repeat steps **6.** and **7.** but change the Table name to **bird** and add the points over the land areas.



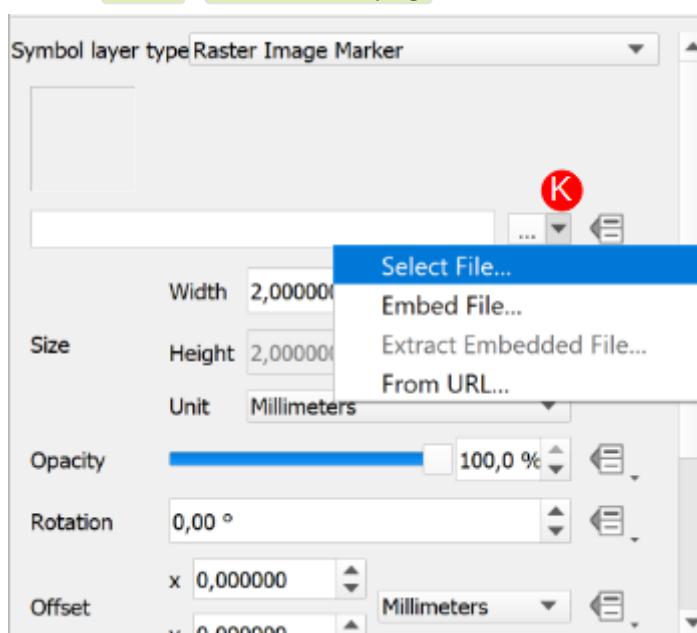
9. Select the **fish** layer and then in the **Layer styling** toolbar (**I**) change the **Symbol layer type** to **Raster Image Marker** (**J**).

"Bring your QGIS projects to life!"





Select the marker image by clicking the **Dropdown menu** → **Select File...** (**K**) and then choosing **fish** → **fish_0000.png** .

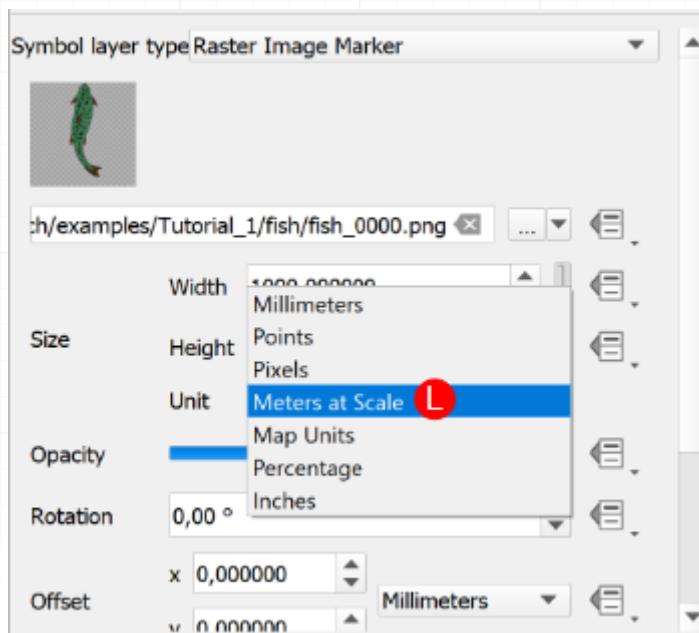


Click **Open**

10. Change the marker's Size Unit to **Meters at Scale** (**L**)

"Bring your QGIS projects to life!"





and set the Width and Height to 1000.

11. Repeat Steps 9. and 10. with the **bird** layer but instead choosing **bird → bird_0000.png** and setting the Width and Height to 3000.

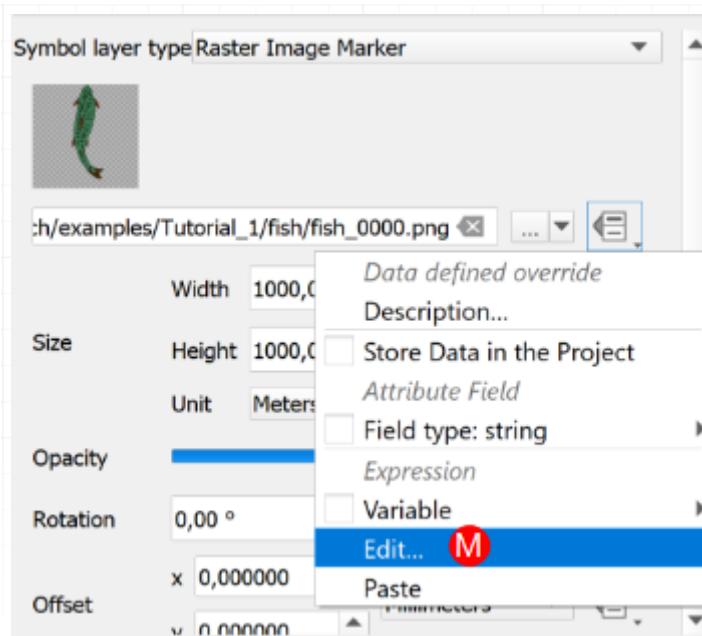
Note:

In **QGIS 3.26**, or later, the **Symbol layer type** can simply be selected as **Animated Marker** and Step 12. can be skipped.

12. To animate the **fish** and **bird** layers using the **QGIS Expressions** system click the **Dropdown Menu → Edit... (M)**.

"Bring your QGIS projects to life!"





For the **fish** layer use the following expression:

"Bring your QGIS projects to life!"





```
@project_home
||
'/fish/fish_00'
||
lpad(to_string( @frame_number % 32), 2, '0')
||
'.png'|
```

 **Code:**

```
@project_home
||
'/fish/fish_00'
||
lpad(to_string( @frame_number % 32), 2, '0')
||
'.png'|
```

And for the **bird** layer use:

"Bring your QGIS projects to life!"





```
@project_home
||
'/bird/bird_00'
||
lpad(to_string(@frame_number % 9), 2, '0')
||
'.png'
```

 **Code:**

```
@project_home
||
'/bird/bird_00'
||
lpad(to_string(@frame_number % 9), 2, '0')
||
'.png'
```

 **Note:**

Refer to the [What is the Workbench doing?](#) section for an explanation about what the above code snippet is doing.

13. Open the Animation Workbench (refer to the [Using the Animation Workbench](#) section if you are unsure how to open the Workbench).

"Bring your QGIS projects to life!"





In the **Animation Plan** tab set:

- the **Render Mode** to **Planar (N)**,
- the **Animation Layer** to **route (O)** using the dropdown menu,
- the **Zoom Range (P)** to 1:270000 for the Minimum and 1:135000 for the Maximum,
- the **Frame rate per second** to 9 fps (Q),
- the **Travel duration** to 4,00 s (R),
- and the **Feature hover duration** to 2,00 s (S)

Enable both the **Pan** and **Zoom** easings and set them to linear.

Animation Workbench

Animation Plan Intro Outro Soundtrack Output Progress

Render Mode

Sphere **N** Planar Fixed Extent

Animation Layer

: route **O**

Loop from final feature back to first feature

Zoom Range (P)

Minimum (exclusive) Maximum (inclusive)

1:270000 1:135000

Data Defined Settings

Scale Minimum Maximum

Animation Frames

Frame rate per second 9 fps **Q**

Travel duration 4,00 s **R**

Feature hover duration 2,00 s **S**

Pan and Zoom Easings

Enable Pan Easing

Linear

Enable Zoom Easing

Linear

Frame Preview

Frame

0

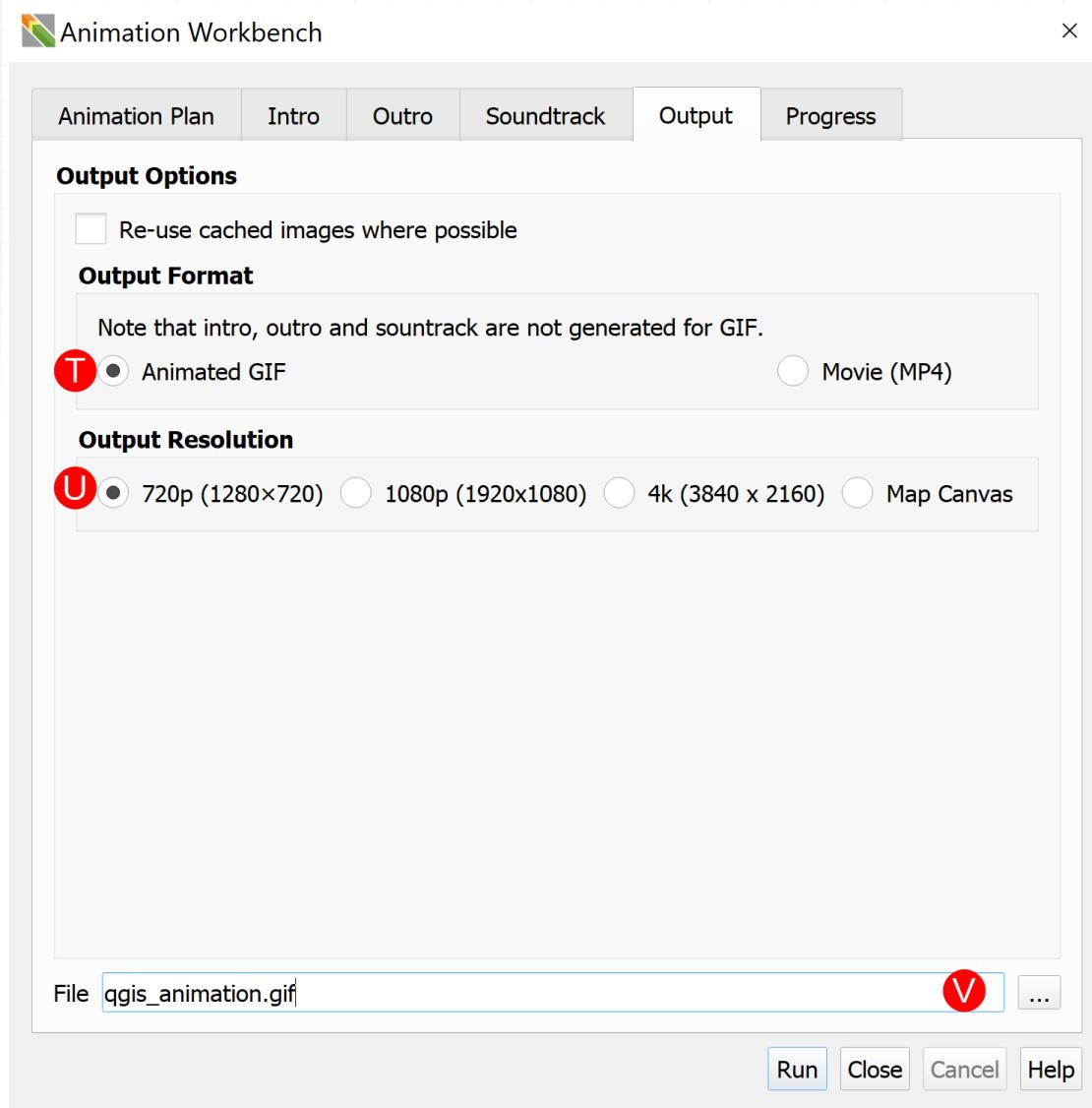
Run Close Cancel Help

"Bring your QGIS projects to life!"





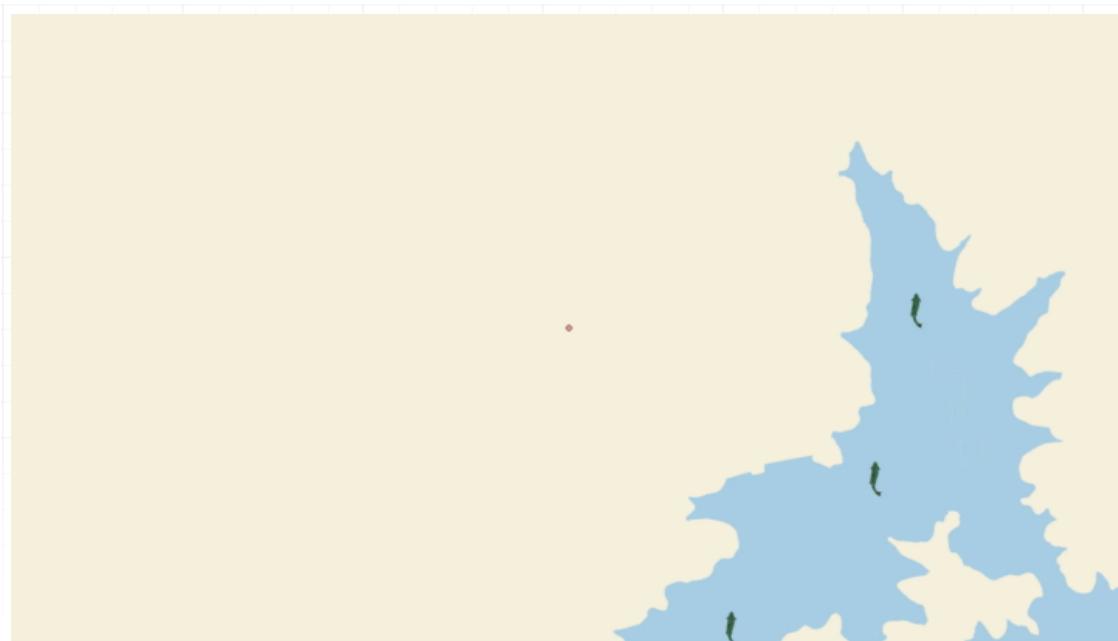
14. Skip past the **Intro**, **Outro**, and **Soundtrack** tabs to the **Output** tab. Set the **Output Format** as **Animated Gif** (**T**) and the **Output Resolution** to **720p (1280x720)** (**U**). The **Output Resolution** can be set as any of the three choices but was set at **720** for this tutorial for the sake of speed. Set the output location to one you can find easily (**V**)



15. Click **Run** and watch what the Workbench is doing in the **Progress** tab. Once the Workbench is finished running, you should end up with an output similar to this:

"Bring your QGIS projects to life!"





After this tutorial you should have a better understanding of how to create a point layer in your project and then to change the **Single Symbol** markers into stationary animated markers. A key focus is the idea that you can tell versions of **QGIS** before **3.26** to dynamically change markers using short code snippets. Versions of **QGIS** post **3.26** allow a user to simply use the **Animated Marker** feature without editing an expression.

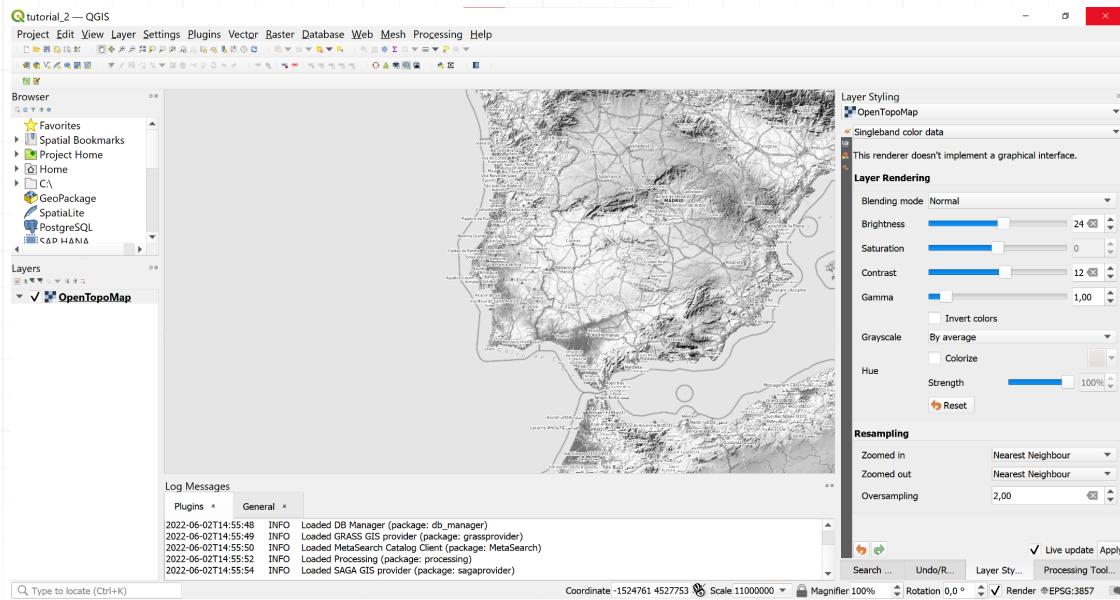
1.0.3 Tutorial 3: Flying Points

This tutorial aims to show you how add a flying point animation to points on your map using built-in QGIS functionalities (The geometry generator line) and introduced variables from the workbench.

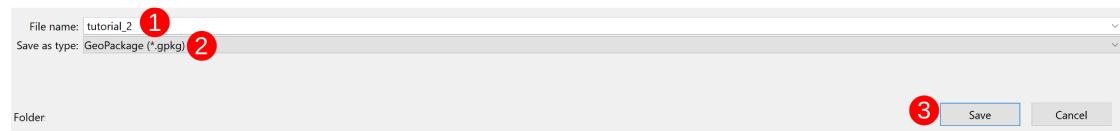
1. Download and extract the [Required Tutorial Zip Folder](#)
2. Open the **tutorial_3.qgz** project file. When you first open the project file you should be greeted with something like this:

"Bring your QGIS projects to life!"





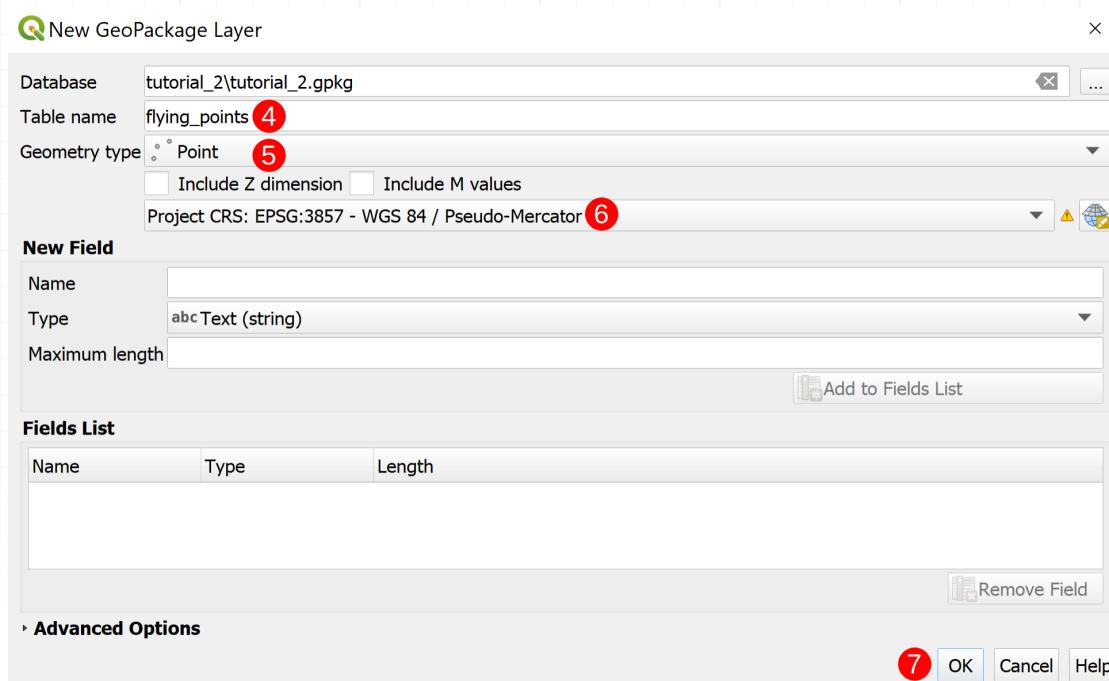
3. Create a new point layer in a new geopackage by clicking **Layer → Create Layer → New GeoPackage Layer...**. Click on the **Ellipsis** (three dots) next to the Database textbox and navigate to the folder that the **tutorial_3.qgz** file is located in. Type the File name "**tutorial_3**" (1) and ensure the file will be saved as a **GeoPackage** (2) and click **Save** (3).



Change the Table name to **flying_points** (4), set the Geometry type as **Point** (5) and change the CRS to match the **Project CRS** (6).

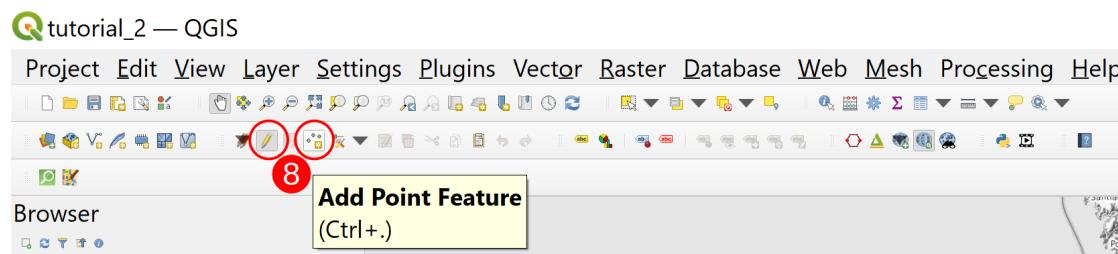
"Bring your QGIS projects to life!"





Click **OK** (7)

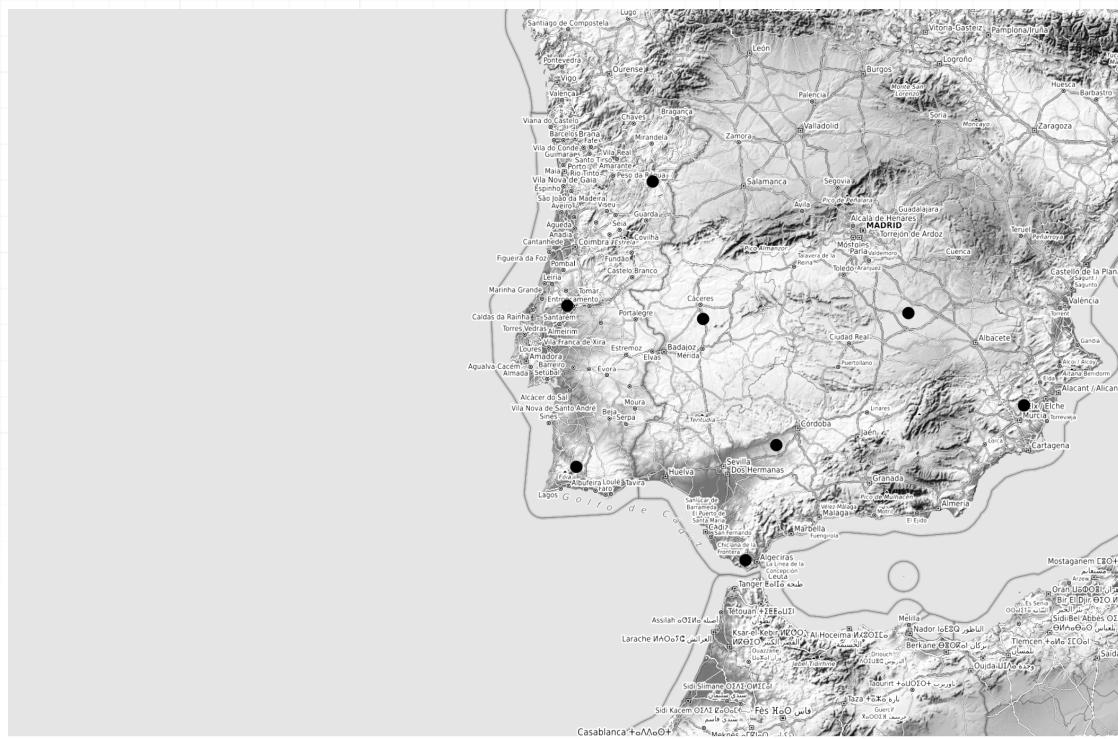
4. Click on **Toggle Editing** → **Add Point Feature** (8).



And randomly add points to your map. Depending on your computer's capabilities, you can add more, or fewer, points than the example below.

"Bring your QGIS projects to life!"

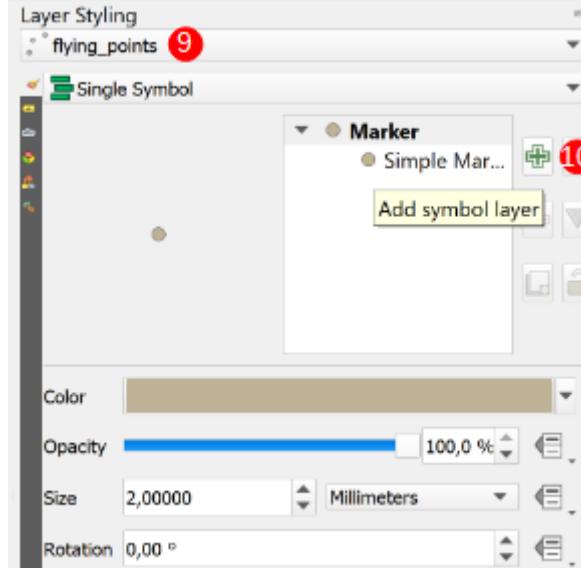




Save your Layer Edits and toggle off the Editing tool.

5. Style the points layer.

Select the **flying_points** (9) layer and in the **Layer Styling** toolbar click on the **Add Symbol Layer** (green plus symbol) button (10).

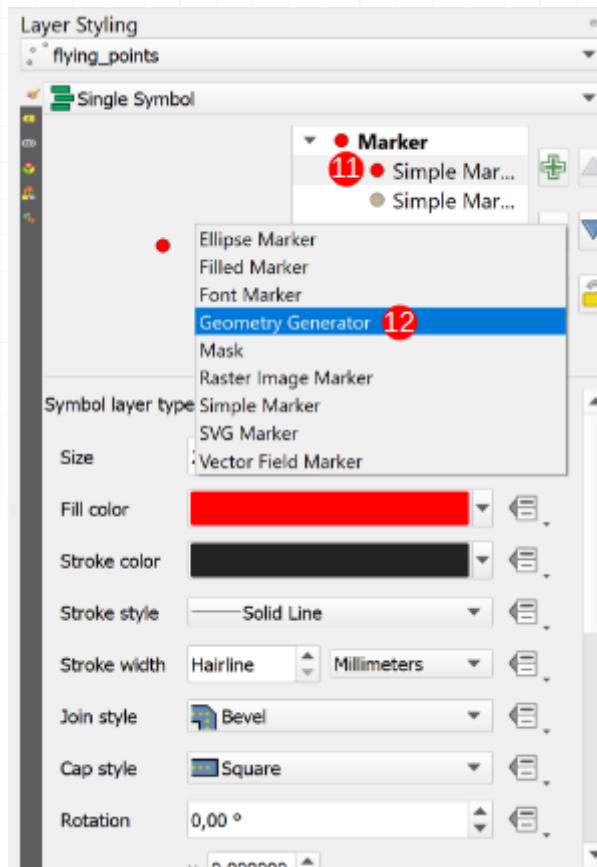


"Bring your QGIS projects to life!"

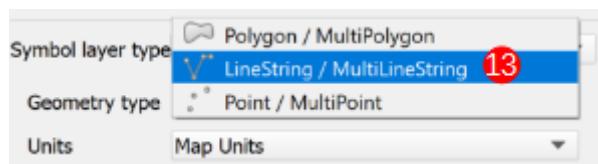




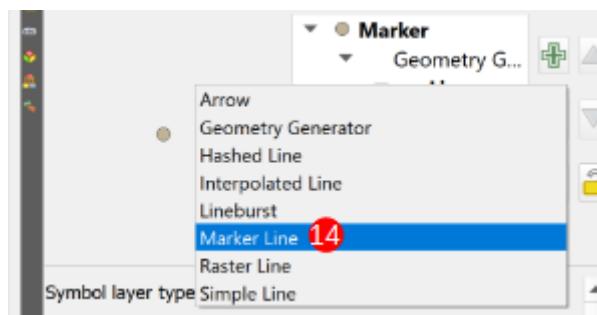
Select the top **Simple Marker** (11) and change its Symbol layer type to **Geometry Generator** (12).



and then set the Geometry type to **LineString / MultiLineString** (13).



Change the line's Symbol layer type to **Marker Line** (14).

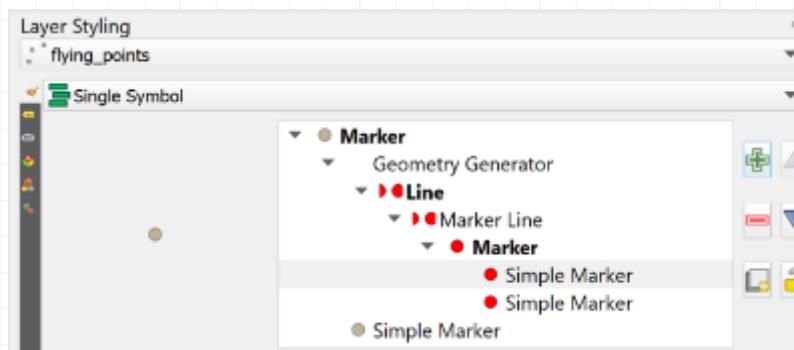


"Bring your QGIS projects to life!"





Add a second **Simple marker** to the marker line so that you end up with something like this:



Style the various **Simple Markers** to your preferred look.

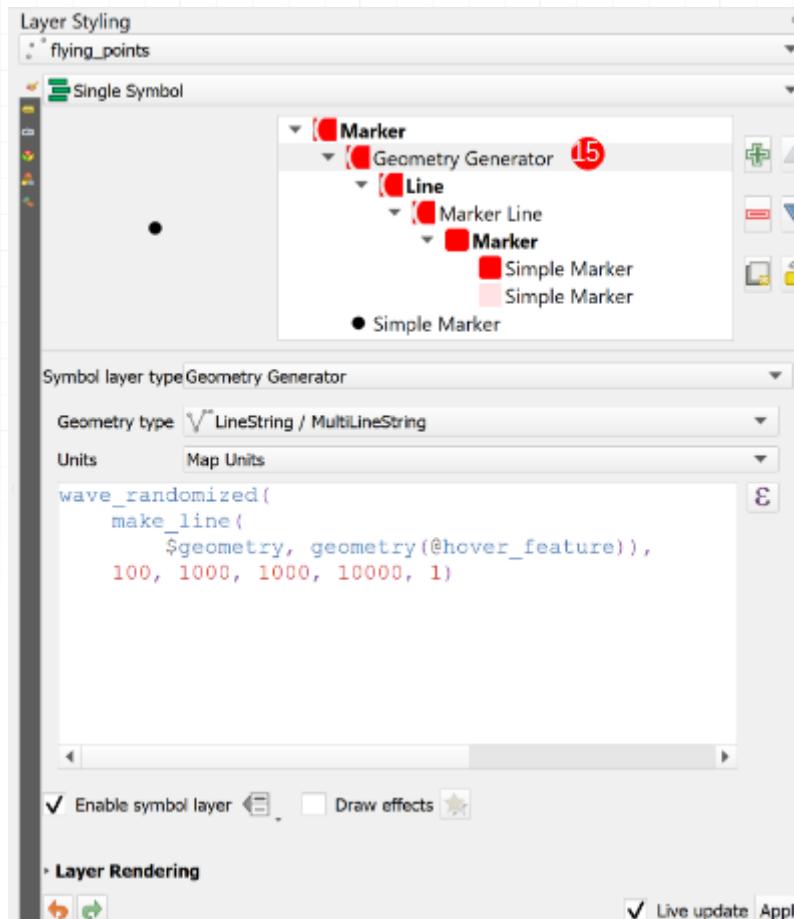
6. Select the **Geometry Generator** symbol layer (**15**) and add this code to it:

Code:

```
wave_randomized(  
    make_line(  
        $geometry, geometry(@hover_feature)),  
        100, 1000, 1000, 10000, 1)
```

"Bring your QGIS projects to life!"





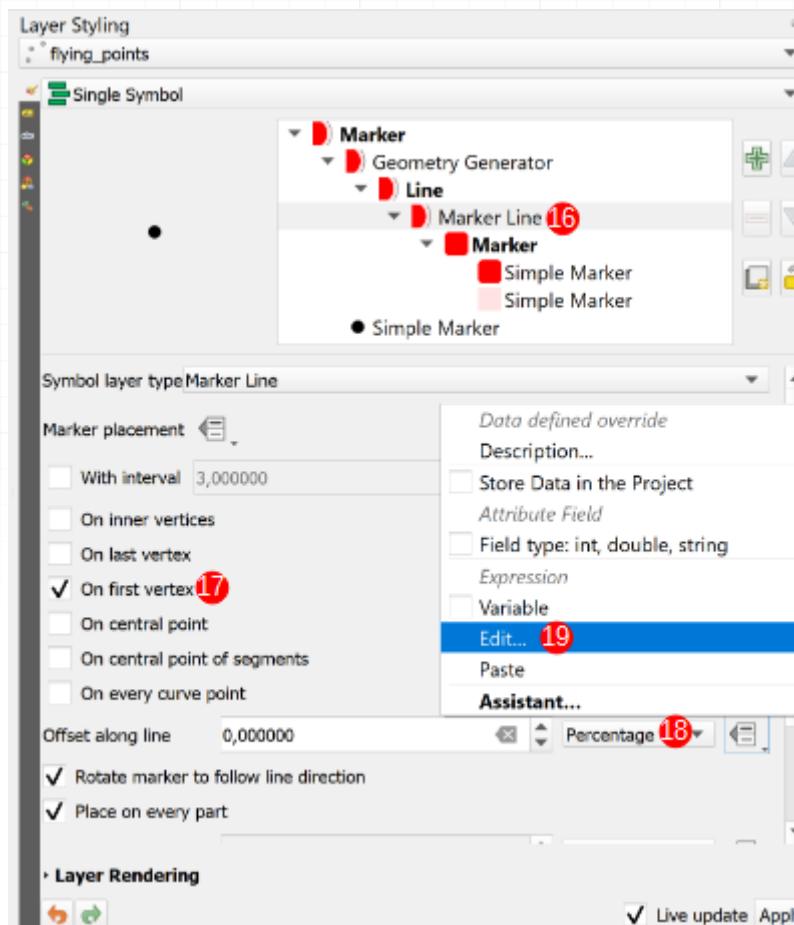
Note:

More information about what changing the numbers will affect can be found in the QGIS expressions editor.

7. A few options need to be changed in the **Marker Line** symbol layer (16): The Marker placement needs to be set to **On first vertex** (17) and, the Offset along line needs to be changed to **Percentage** (18). Then click the **Dropdown menu** next to Offset along line and select **Edit...** (19).

"Bring your QGIS projects to life!"





In the **Expression String Builder** add the following code snippet:

Code:

```
100 - to_int((@current_hover_frame / @hover_frames) * 100 )
```

```
100 - to_int((@current_hover_frame / @hover_frames)  
* 100 )
```

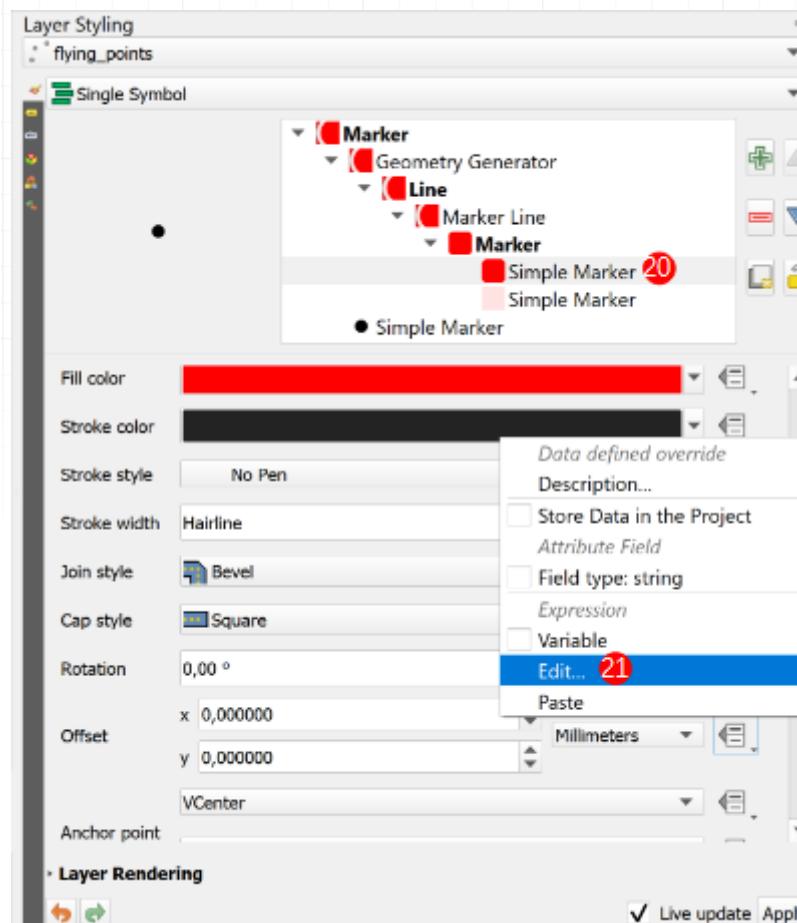
Click **OK**

"Bring your QGIS projects to life!"





8. Select the first **Simple Marker** symbol layer (20) in the **Marker Line** symbol layer. Scroll down to Offset and click on the **Dropdown Menu → Edit...** (21).



"Bring your QGIS projects to life!"





In the **Expression String Builder** add the following code snippet:

Code:

```
-- Taken from https://spicyyoghurt.com/tools/easing-functions
-- t = Time - Amount of time that has passed since the begin
-- b = Beginning value - The starting point of the animation
-- c = Change in value - The amount of change needed to go
-- d = Duration - Amount of time the animation will take. Us
-- Sinusoidal
-- -c / 2 * (Math.cos(Math.PI * t / d) - 1) + b;

-- Use with the animation in static mode
if(@hover_feature_id != $id,
array(
    (-@hover_frames / 2) * (cos( (pi() * @frame_number / @hove
    (-@hover_frames / 2) * (sin( (pi() * @frame_number / @hove
), 
array (0,0))
```

"Bring your QGIS projects to life!"





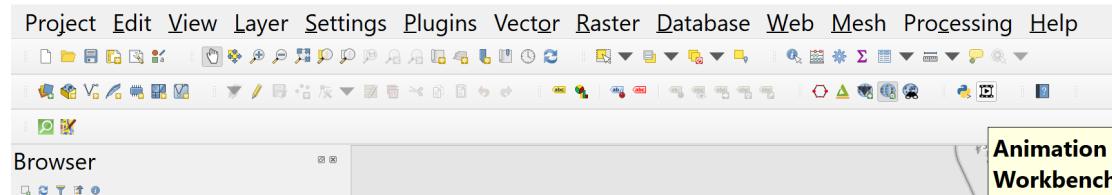
```
-- Taken from https://spicyyoghurt.com/tools/easing-functions
-- t = Time - Amount of time that has passed since the beginning of the
animation. Usually starts at 0 and is slowly increased using a game loop or
other update function.
-- b = Beginning value - The starting point of the animation. Usually
it's a static value, you can start at 0 for example.
-- c = Change in value - The amount of change needed to go from starting
point to end point. It's also usually a static value.
-- d = Duration - Amount of time the animation will take. Usually a
static value aswell.
-- Sinusoidal
-- -c / 2 * (Math.cos(Math.PI * t / d) - 1) + b;

-- Use with the animation in static mode
if(@hover_feature_id != $id,
array(
    (-@hover_frames / 2) * (cos( (pi() * @frame_number / @hover_frames ) - 1
)) ,
    (-@hover_frames / 2) * (sin( (pi() * @frame_number / @hover_frames ) - 1
))
),
array (0,0))
```

Click **OK**

9. Open the **Animation Workbench** (22)

tutorial_2 — QGIS



10. Set up the **Animation Plan** with:

- the **Render Mode** to **Planar** (23),
- the **Animation Layer** to **flying_points** (24) using the dropdown menu,
- the **Zoom Range** (25) to 1:22000000 for the Minimum and 1:11000000 for the Maximum,
- the **Frame rate per second** to 9 fps (26),
- the **Travel duration** to 2,00 s (27),
- the **Feature hover duration** to 2,00 s (28),
- and the **Zoom Easing** as InCirc (29)

"Bring your QGIS projects to life!"





Animation Workbench

Animation Plan Intro Outro Soundtrack Output Progress

23 Render Mode
 Sphere Planar Fixed Extent

24 Animation Layer
flying_points Loop from final feature back to first feature

25 Zoom Range
Minimum (exclusive) Maximum (inclusive)
1:22000000 1:11000000

Data Defined Settings
Scale Minimum Maximum

Animation Frames
Frame rate per second: 9 fps **26**
Travel duration: 2,00 s **27**
Feature hover duration: 2,00 s **28**

Pan and Zoom Easings
 Enable Pan Easing Linear
 Enable Zoom Easing **29** InCirc

Frame Preview

Frame: 0

Run Close Cancel Help

Note:

With a decently specced computer you can up the fps and get the points to fly faster in your output.

"Bring your QGIS projects to life!"





11. Add license-free media to the **Intro**, **Outro**, and **Soundtrack**.

 **Note:**

Make sure your **Soundtrack** is as long as, or longer than, your final animation will be (including the **Intro**, **Animation**, and **Outro**).

12. Set the **Output Format** as **Movie (MP4)** (**30**) and the **Output Resolution** to **1080 (1920x1080)** (**31**). The **Output Resolution** can be set as any of the three choices but was set at **1080** for this tutorial for the sake of speed. Set the output location (**32**) to one you can easily locate.

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

Output Options

Re-use cached images where possible

Output Format

Note that intro, outro and soundtrack are not generated for GIF.

Animated GIF 31 Movie (MP4)

Output Resolution 32

720p (1280x720) 1080p (1920x1080) 4k (3840 x 2160) Map Canvas

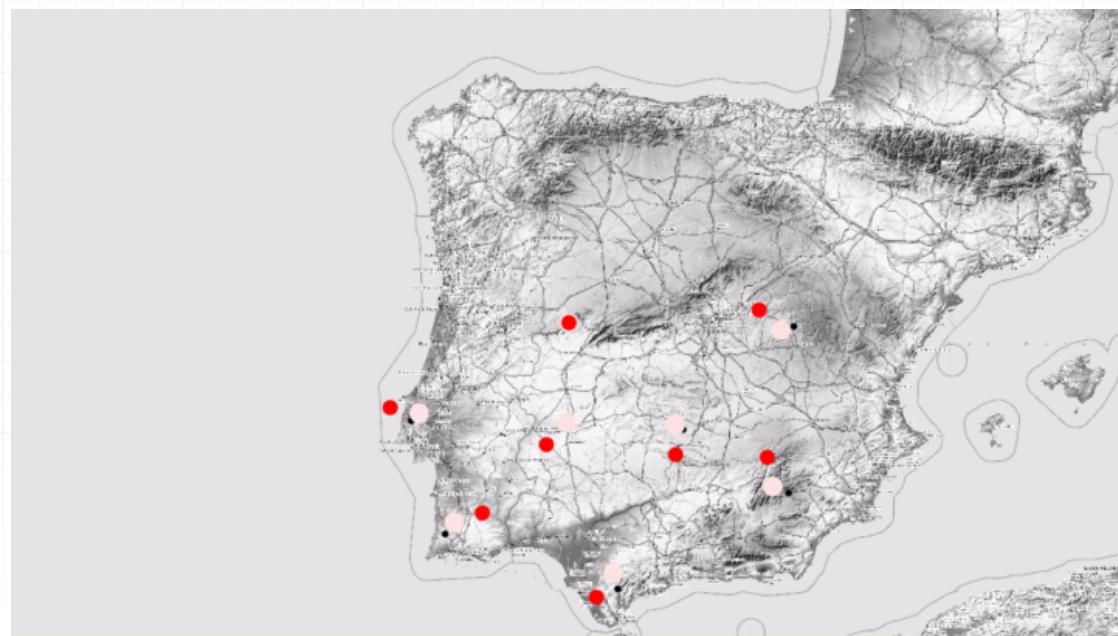
File qgis_animation.mp4 33 ...

Run Close Cancel Help

13. Click **Run** and get an output. The **GIF** below is the visual output of the tutorial if you followed step-by-step and set the parameters to exactly what was stated.

"Bring your QGIS projects to life!"



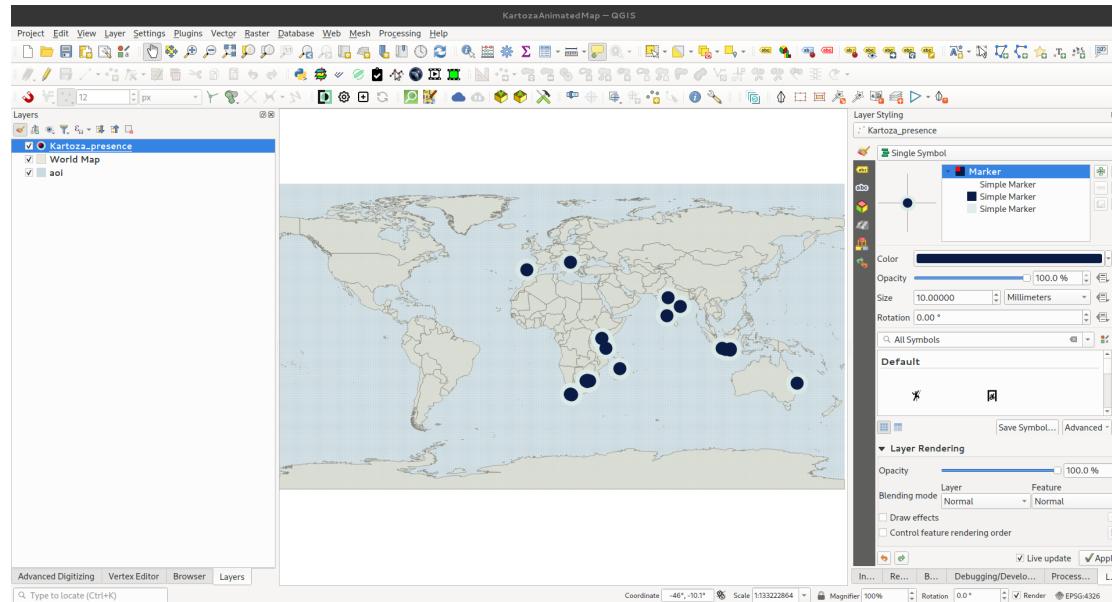


The link to a more complex output (with an [Intro](#), an [Outro](#), and a [Soundtrack](#)) can be found [here](#).

After this tutorial you should have a better idea of how you can use a mixture of built-in QGIS functionalities and the workbench's introduced variables to generate interesting outputs.

1.0.4 Tutorial 4: Spinning Globe

Given a global point layer and countries layer like this:



You can create a nice spinning globe effect like this:

"Bring your QGIS projects to life!"

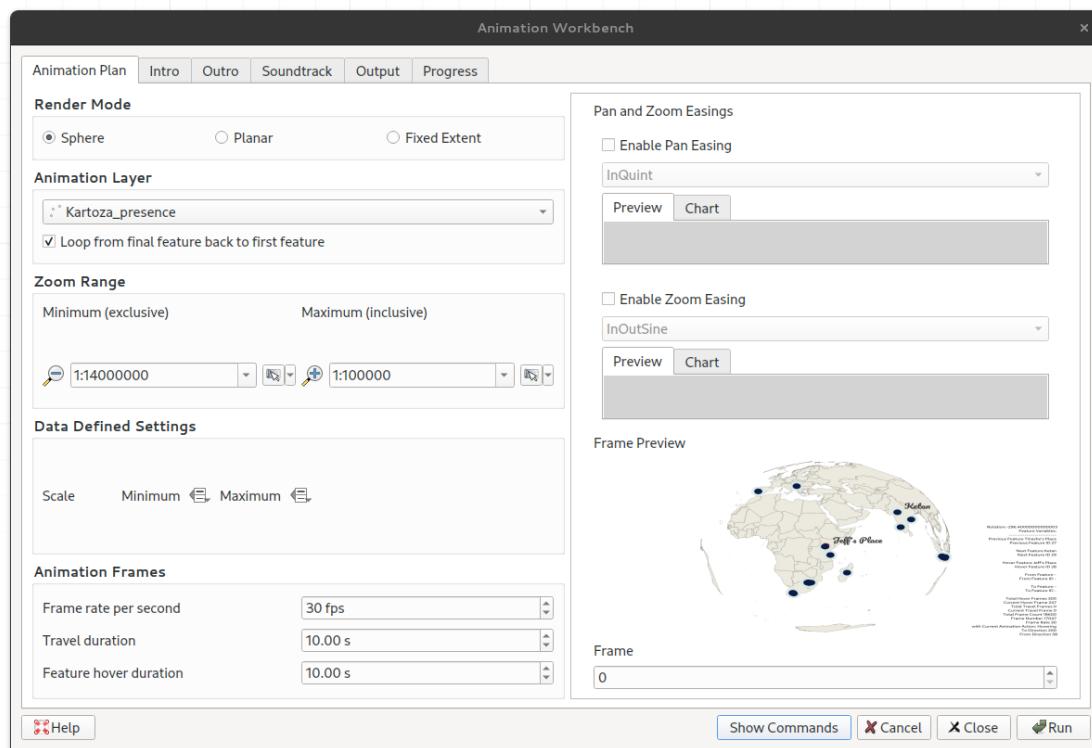




I set up the animation workbench like this:

"Bring your QGIS projects to life!"





For the above animated GIF, I compressed it using imagemagick like this:

 **Code:**

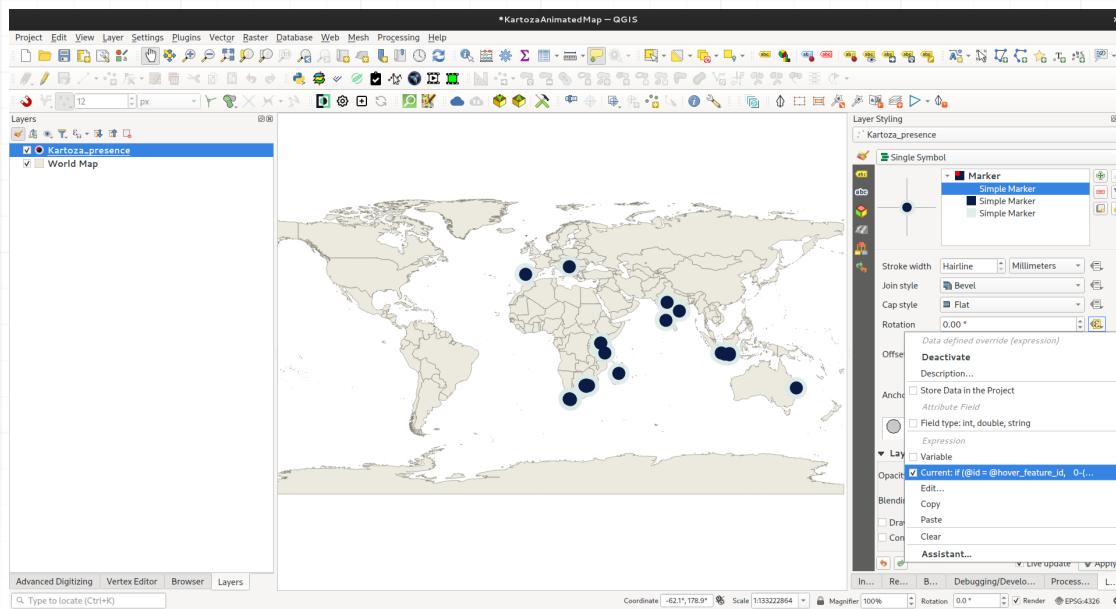
```
convert globe.gif -coalesce -resize 700x525 -fuzz 2% +dither -laye
```

This is a handy technique if you want to generate small file size animations.

For the points I made a red marker using a quarter circle that spins around the points like this:

"Bring your QGIS projects to life!"





The rotation field expression is this:

Code:

```
if (@id = @hover_feature_id,  
    0-((1440 * (@current_hover_frame/@hover_frames)) % 360),  
    0)
```

This will spin around 4 times during the hover cycle.

For the ocean (AOI in the layers list), I generated a grid of 1 degree cells covering the earth. You need to do it as smaller polygons instead of one large polygon because QGIS will run into issues reprojecting a single polygon whose edges lie on the date line.

Here is how the final video came out:

"Bring your QGIS projects to life!"



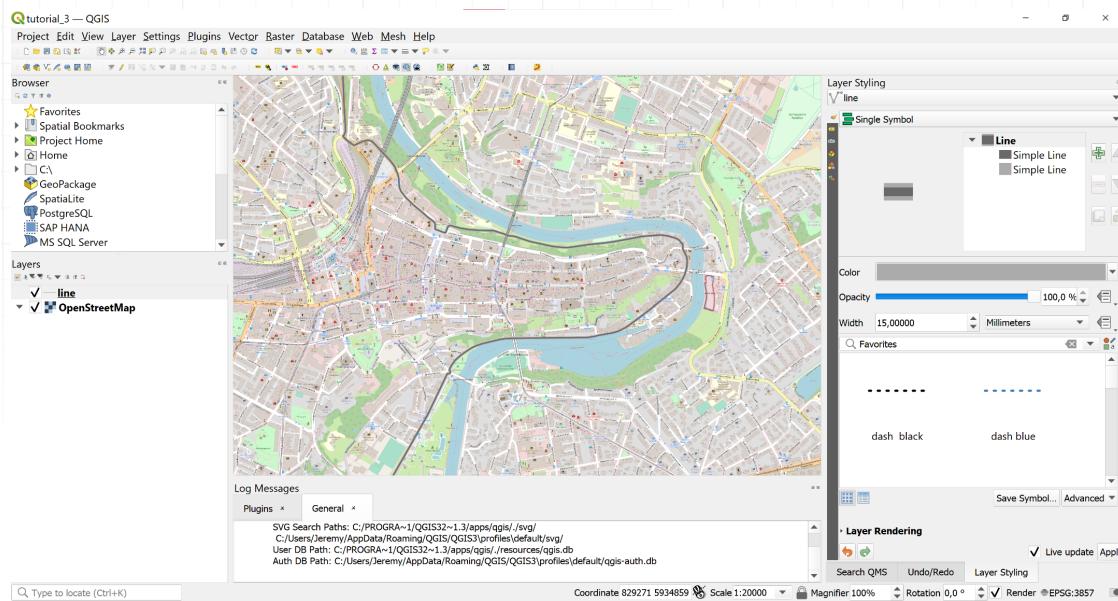


1.0.1 Tutorial 1: Point Along A Line

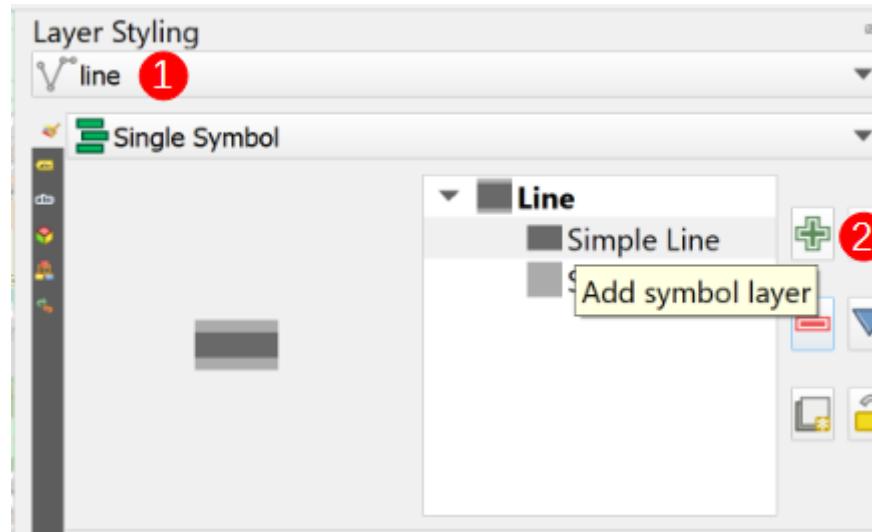
This tutorial introduces the concept of moving a point along a line within your animated map.

1. Download and extract the [Required Tutorial Zip Folder](#)

2. Open the **tutorial_1.qgz** project file that is in the folder. When you first open it you see something like this:



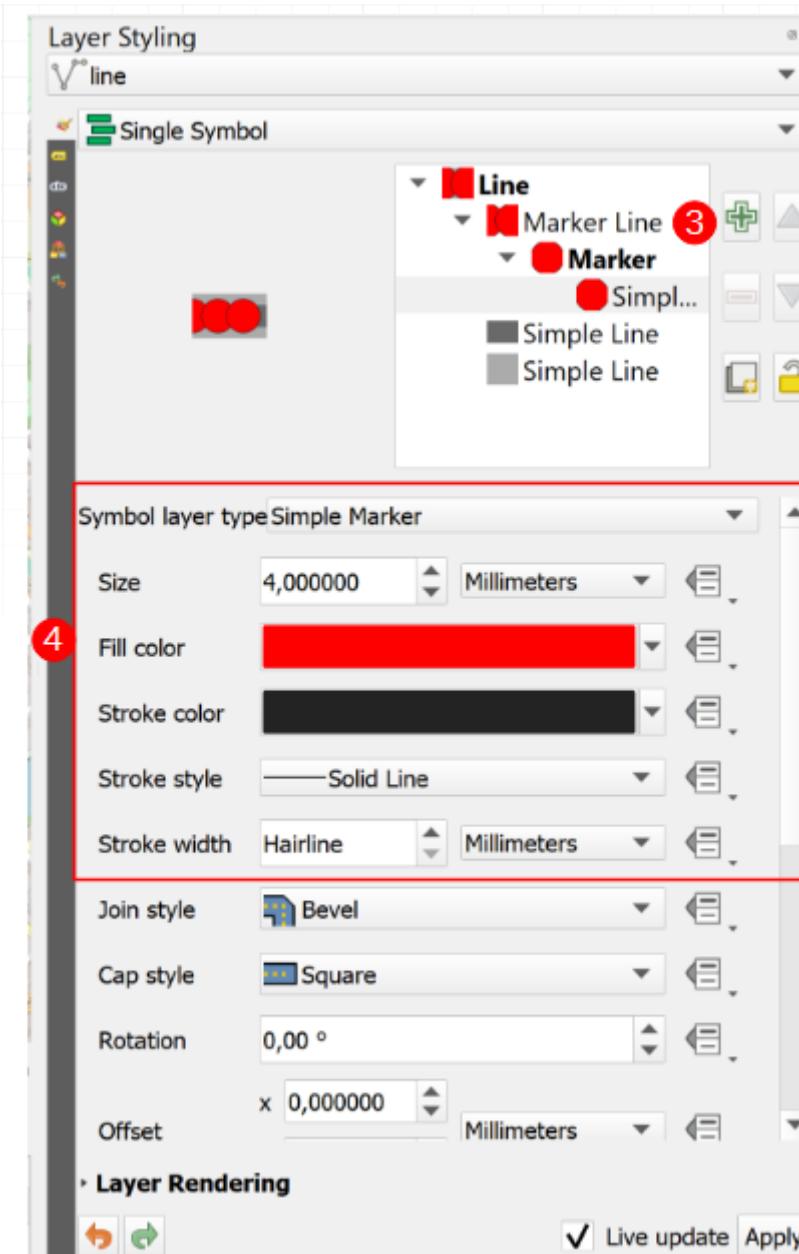
3. Select the premade **line** layer (1), and click on the **Add Symbol Layer** (green plus symbol) button (2) to it.



Change the new **Symbol Layer** (3) type to marker line and then style it (4) so that it is more visible.

"Bring your QGIS projects to life!"





4. Change the **Symbol Layer's** settings so that the point is only on the **first vertex** (5) and and not at equidistant intervals.

Change the offset along the line to be **Percentage** (6).

"Bring your QGIS projects to life!"





Symbol layer type **Marker Line**

Marker placement

With interval 3,0 Store Data in the Project

On inner vertices Attribute Field

On last vertex Field type: int, double, string

On first vertex Expression

On central point Variable

On central point Edit... **8**

On every curve part Paste

Assistant...

Offset along line 0,0000 7

Rotate marker to follow line direction

Place on every part

Layer Rendering

Click the **Dropdown Menu** (7) → **Edit...** (8) and then add the following code snippet

"Bring your QGIS projects to life!"





Symbol layer type **Marker Line**

Marker placement **Data defined override**

With interval 3,0 Store Data in the Project

On inner vertices **Attribute Field** Field type: int, double, string

On first vertex **Expression** Variable

On central point **Edit...** 8 Paste

On central point **Assistant...**

Offset along line 0,000 Percentage 7

Rotate marker to follow line direction

Place on every part

Layer Rendering

Code:

```
-- Point Along Line Code Snippet
(@current_hover_frame/@hover_frames) * 100
```

```
-- Point Along Line Code Snippet
(@current_hover_frame/@hover_frames) * 100
```

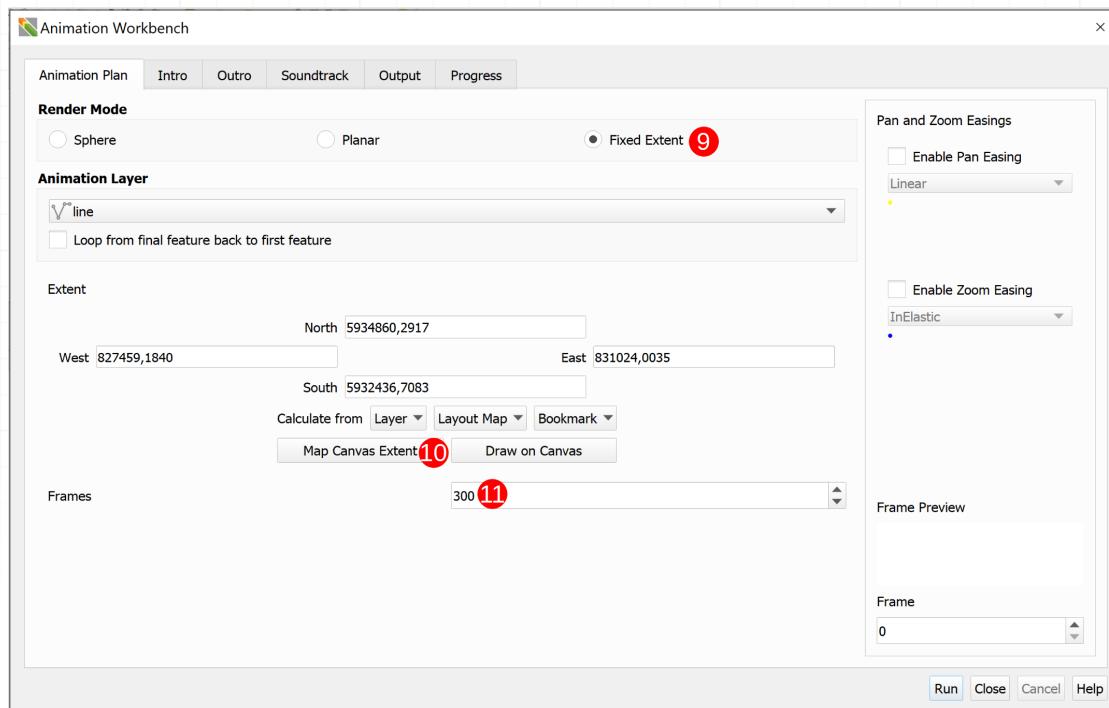
The snippet tells QGIS how far along the line (as a percentage of the line length) to render the point in each frame.

5. Open the Workbench and select **Fixed Extent** (9).

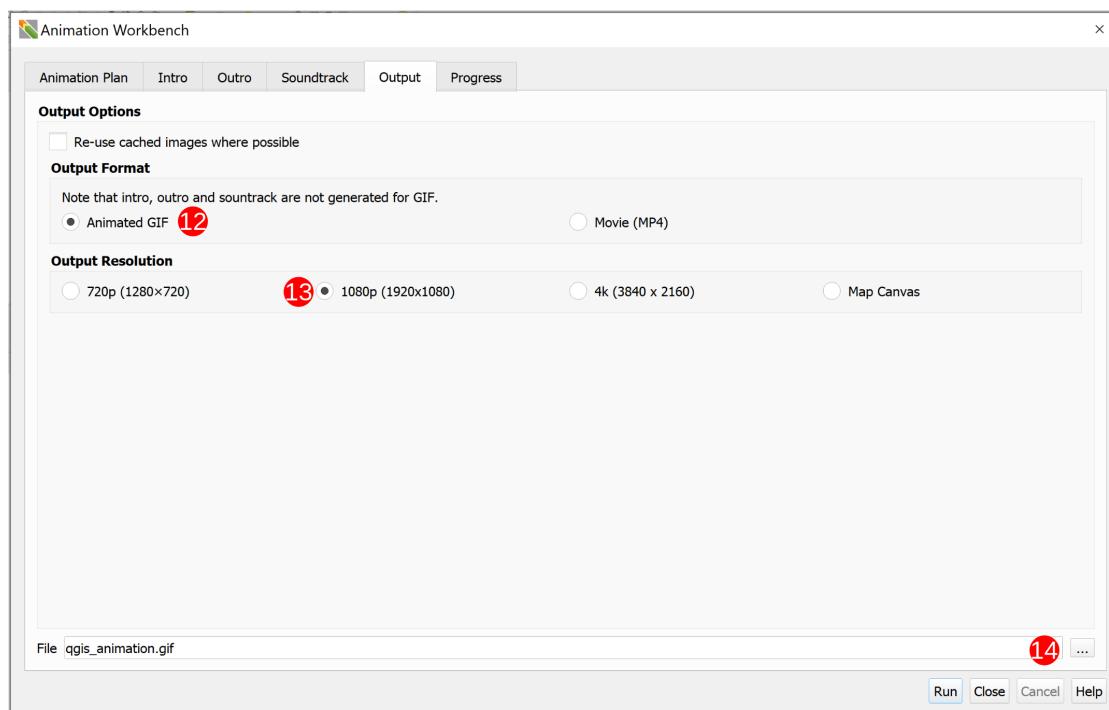
Click on **Map Canvas Extent** (10) and set the the **Frames** to 300 (11) (for a 10 second output at 30 frames per second).

"Bring your QGIS projects to life!"





6. Skip over the **Intro**, **Outro**, and **Soundtrack** tabs. In the **Output** tab, set the output format (12) and resolution (13), and set the output location's path (14).

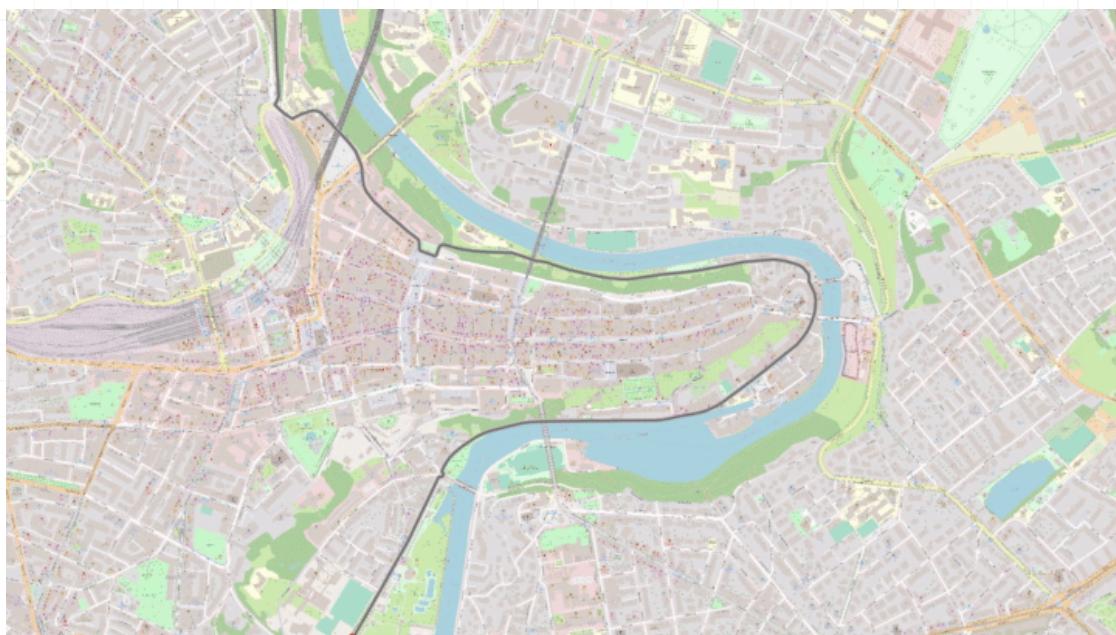


"Bring your QGIS projects to life!"





7. Click **Run** and render your output.



After this tutorial you should have a better idea of how to make a point move along a line. An expansion to this example would be to make the moving point a dynamically changing marker (like the markers in tutorial 1). Go have fun!

"Bring your QGIS projects to life!"

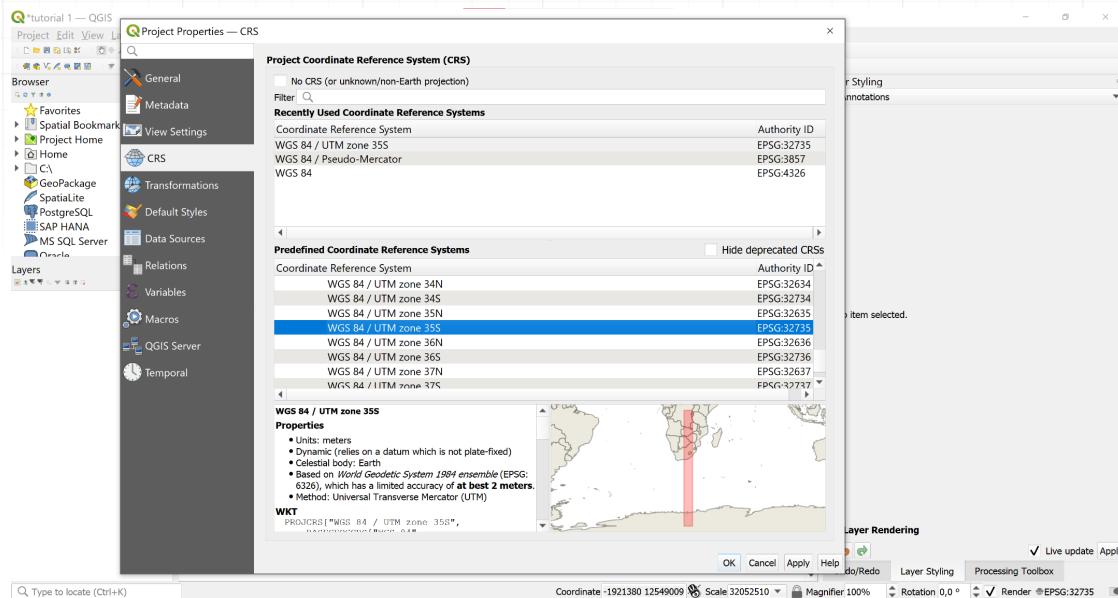




1.0.1 Tutorial 2: Basic Dynamically Changing Markers

This tutorial aims to show you the basics of creating, and animating, a static layer to use with the Animation Workbench. There are three pre-made layers to allow the main focus of the tutorial to be on the Animation Workbench and not on QGIS as a whole.

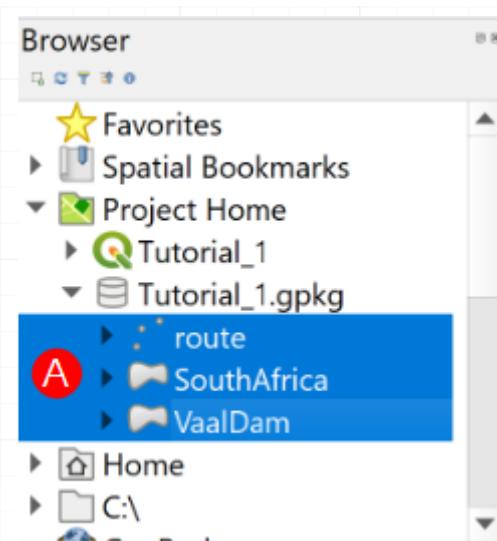
1. Download and extract the [Required Tutorial Zip Folder](#)
2. Open the **tutorial_2.qgz** project file that is in the folder.
3. Set the CRS of your project to **WGS84/UTM zone 35S (EPSG: 32735)**.



4. In the **Browser**, expand the **tutorial_2.gpkg** and add the three pre-made layers (VaalDam, SouthAfrica, and route) (**A**) to your project.

"Bring your QGIS projects to life!"

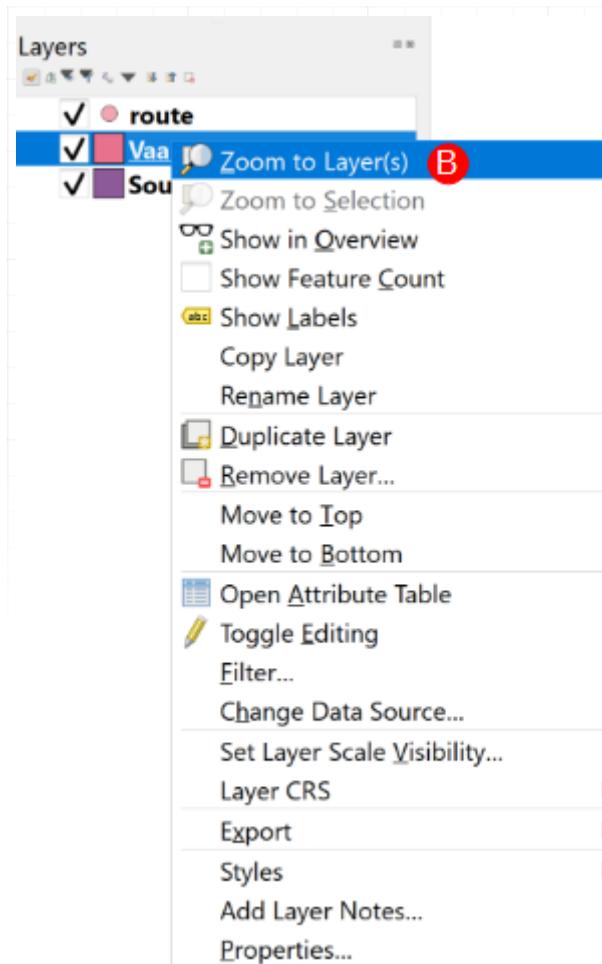




5. In the **Layers** Panel, arrange the layers in the following order: **route**, **VaalDam**, **SouthAfrica**. Then right-click on the **VaalDam** layer and **Zoom to Layer(s)** (B)

"Bring your QGIS projects to life!"



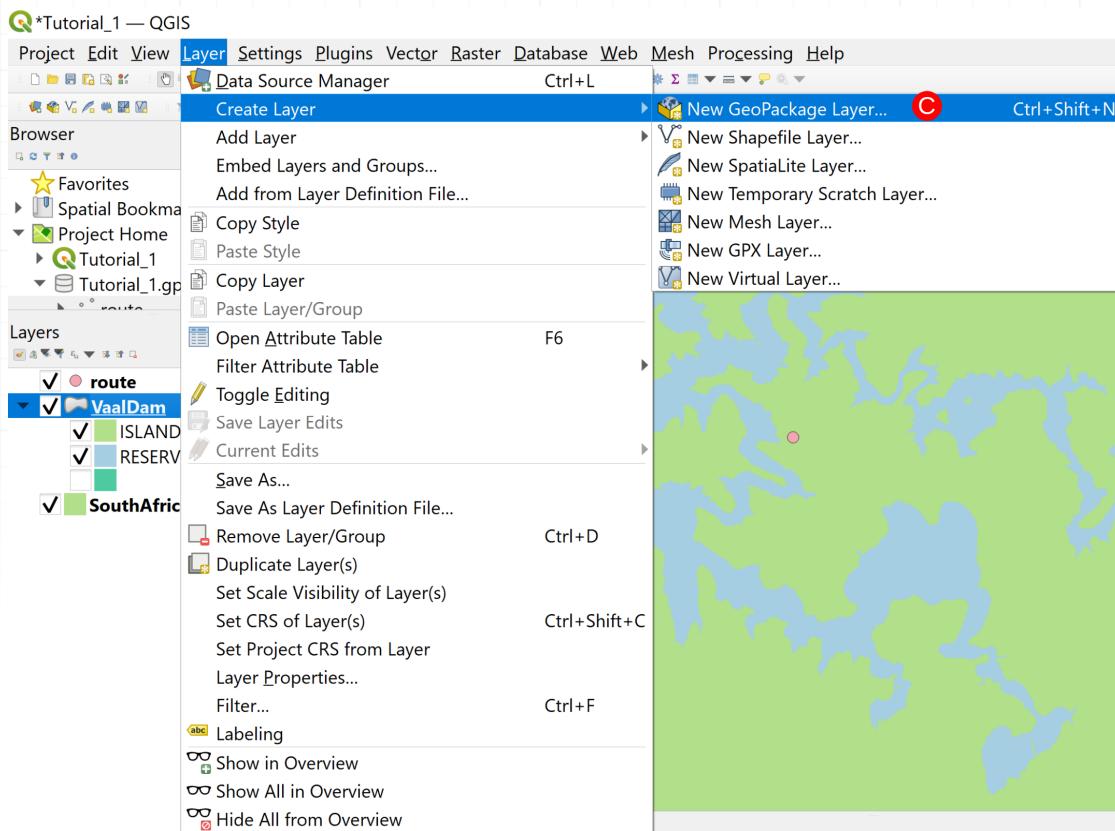


Style the three layers to your preferred style.

6. Now create a new layer in the **tutorial_2.gpkg** by clicking **Layer → Create Layer → New GeoPackage Layer... (c)**.

"Bring your QGIS projects to life!"

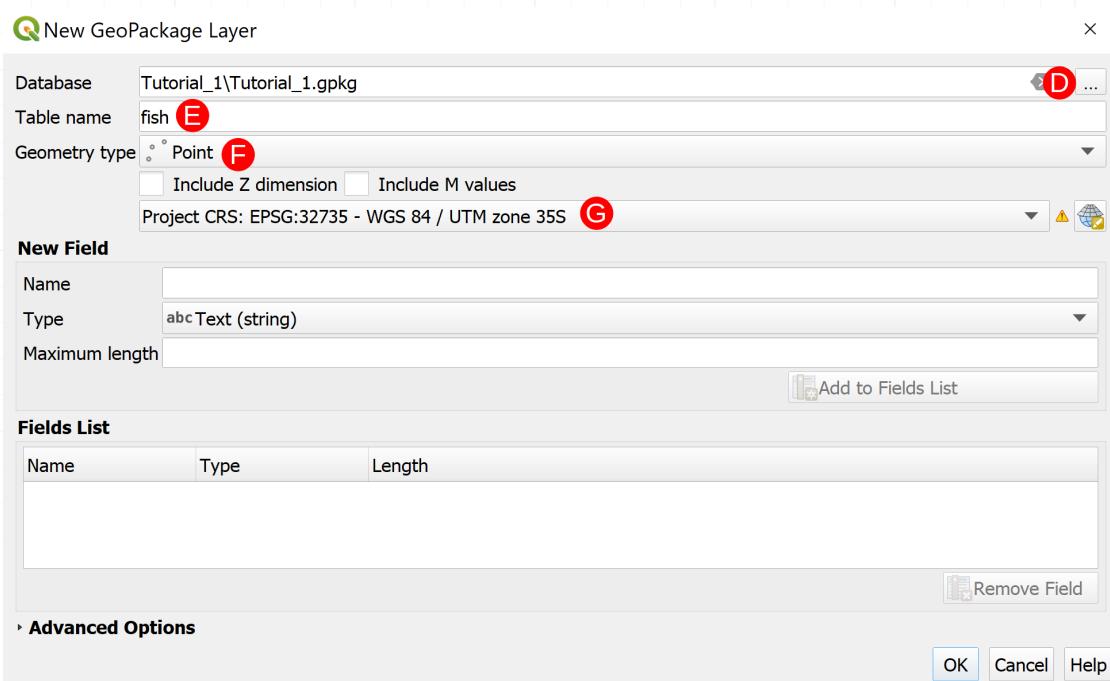




Click on the **Ellipsis** (D), navigate to and select the **tutorial_2.gpkg**, and click **Save**. Change the Table name to **fish** (E), set the Geometry type as **Point** (F), and change the CRS to match the **Project CRS** (G).

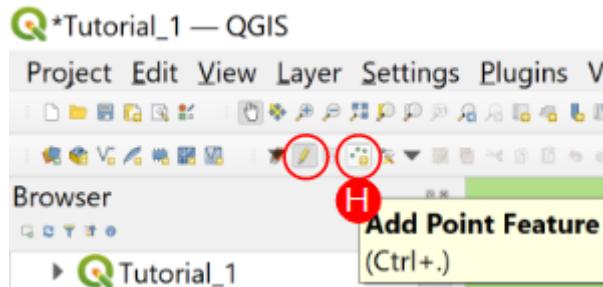
"Bring your QGIS projects to life!"





Click on **OK** and then click **Add New Layer** on the window that pops up.

7. Select the **fish** layer and then click on **Toggle Editing** → **Add Point Feature** (H).



Add a few points wherever you feel they should go (Hint: This is a fish layer so adding them above the dam layer would be best). Don't worry about naming the points, just add them.

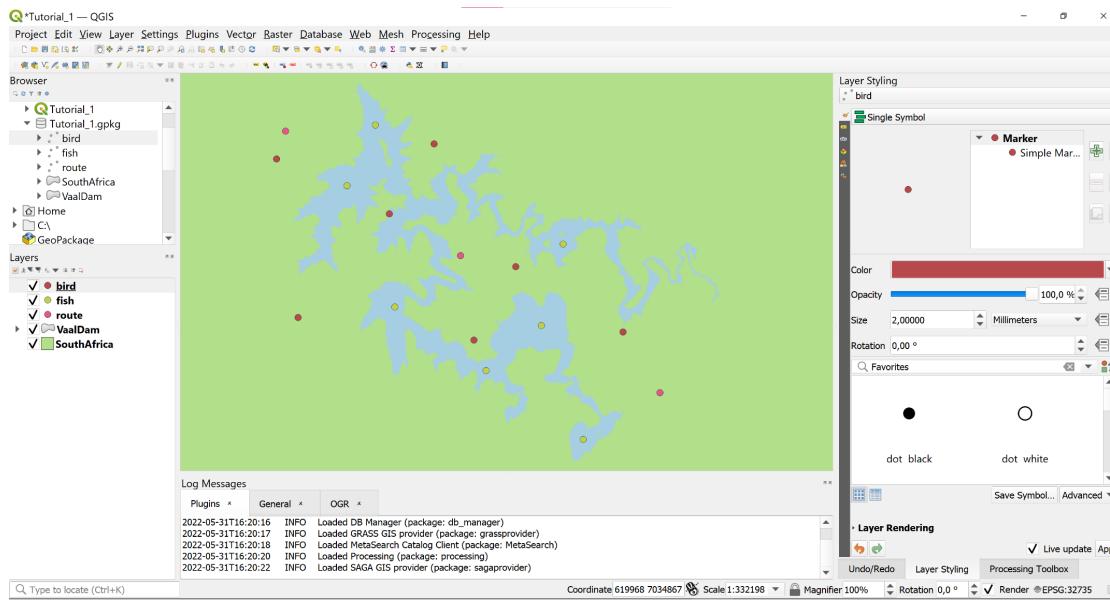
"Bring your QGIS projects to life!"





Save your changes by clicking on **Save Layer Edits** just next to the **Toggle Editing** button.
Then stop editing the layer.

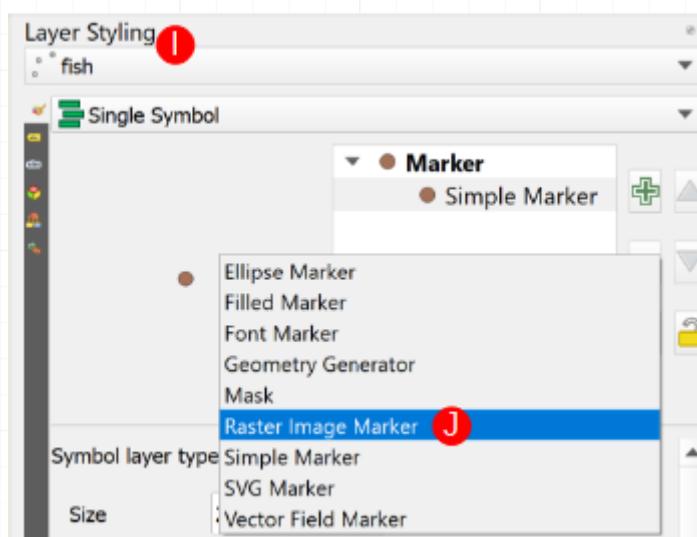
8. Repeat steps **6.** and **7.** but change the Table name to **bird** and add the points over the land areas.



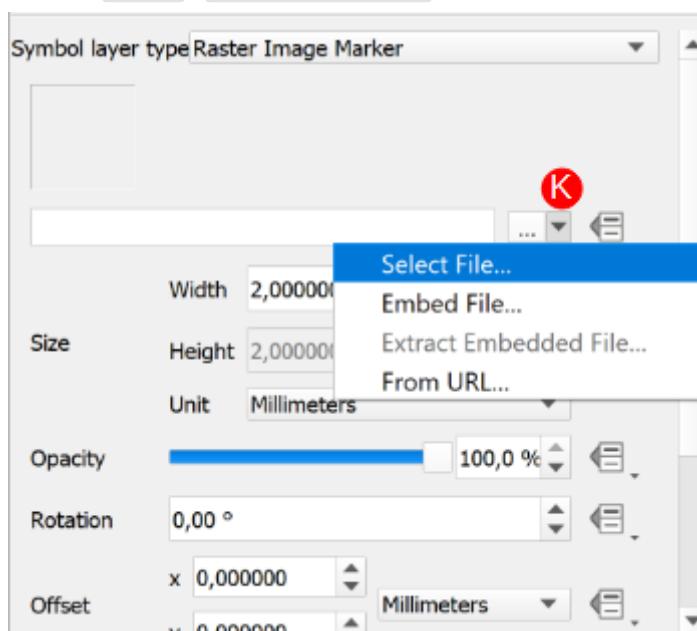
9. Select the **fish** layer and then in the **Layer styling** toolbar (**I**) change the **Symbol layer type** to **Raster Image Marker** (**J**).

"Bring your QGIS projects to life!"





Select the marker image by clicking the **Dropdown menu** → **Select File...** (K) and then choosing **fish** → **fish_0000.png**.

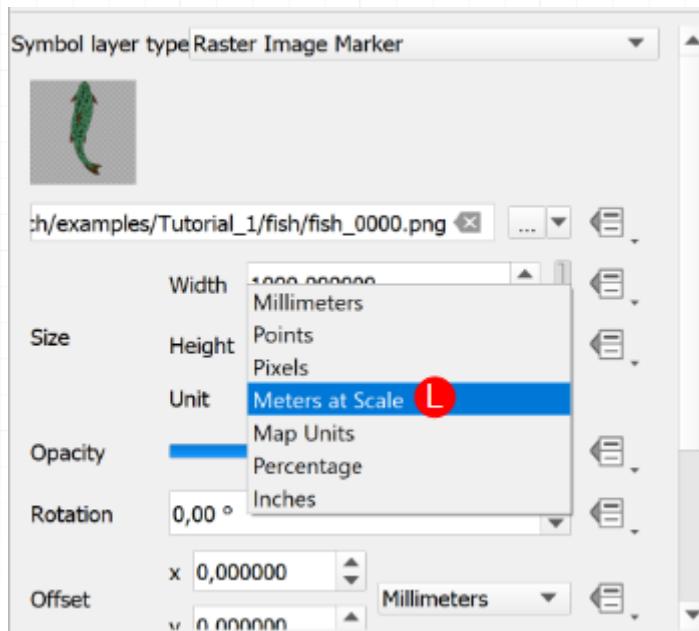


Click **Open**

10. Change the marker's Size Unit to **Meters at Scale** (L)

"Bring your QGIS projects to life!"





and set the Width and Height to 1000.

11. Repeat Steps 9. and 10. with the **bird** layer but instead choosing **bird → bird_0000.png** and setting the Width and Height to 3000.

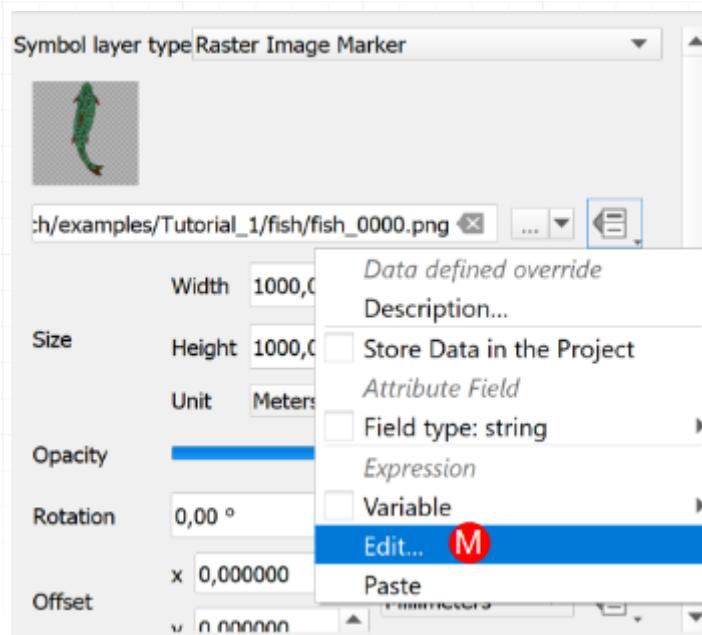
Note:

In **QGIS 3.26**, or later, the **Symbol layer type** can simply be selected as **Animated Marker** and Step 12. can be skipped.

12. To animate the **fish** and **bird** layers using the **QGIS Expressions** system click the **Dropdown Menu → Edit... (M)**.

"Bring your QGIS projects to life!"





For the **fish** layer use the following expression:

"Bring your QGIS projects to life!"





```
@project_home
||
'/fish/fish_00'
||
lpad(to_string( @frame_number % 32), 2, '0')
||
'.png'|
```

 **Code:**

```
@project_home
||
'/fish/fish_00'
||
lpad(to_string( @frame_number % 32), 2, '0')
||
'.png'|
```

And for the **bird** layer use:

"Bring your QGIS projects to life!"





```
@project_home
||
'/bird/bird_00'
||
lpad(to_string(@frame_number % 9), 2, '0')
||
'.png'
```

 **Code:**

```
@project_home
||
'/bird/bird_00'
||
lpad(to_string(@frame_number % 9), 2, '0')
||
'.png'
```

 **Note:**

Refer to the [What is the Workbench doing?](#) section for an explanation about what the above code snippet is doing.

13. Open the Animation Workbench (refer to the [Using the Animation Workbench](#) section if you are unsure how to open the Workbench).

"Bring your QGIS projects to life!"





In the **Animation Plan** tab set:

- the **Render Mode** to **Planar (N)**,
- the **Animation Layer** to **route (O)** using the dropdown menu,
- the **Zoom Range (P)** to 1:270000 for the Minimum and 1:135000 for the Maximum,
- the **Frame rate per second** to 9 fps (Q),
- the **Travel duration** to 4,00 s (R),
- and the **Feature hover duration** to 2,00 s (S)

Enable both the **Pan** and **Zoom** easings and set them to linear.

Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

Render Mode
 Sphere **N** Planar Fixed Extent

Animation Layer
route **O**
 Loop from final feature back to first feature

Zoom Range (P)
Minimum (exclusive) Maximum (inclusive)
1:270000 1:135000

Data Defined Settings
Scale Minimum Maximum

Animation Frames
Frame rate per second: 9 fps (Q)
Travel duration: 4,00 s (R)
Feature hover duration: 2,00 s (S)

Pan and Zoom Easings
 Enable Pan Easing Linear

Enable Zoom Easing Linear

Frame Preview

Frame
0

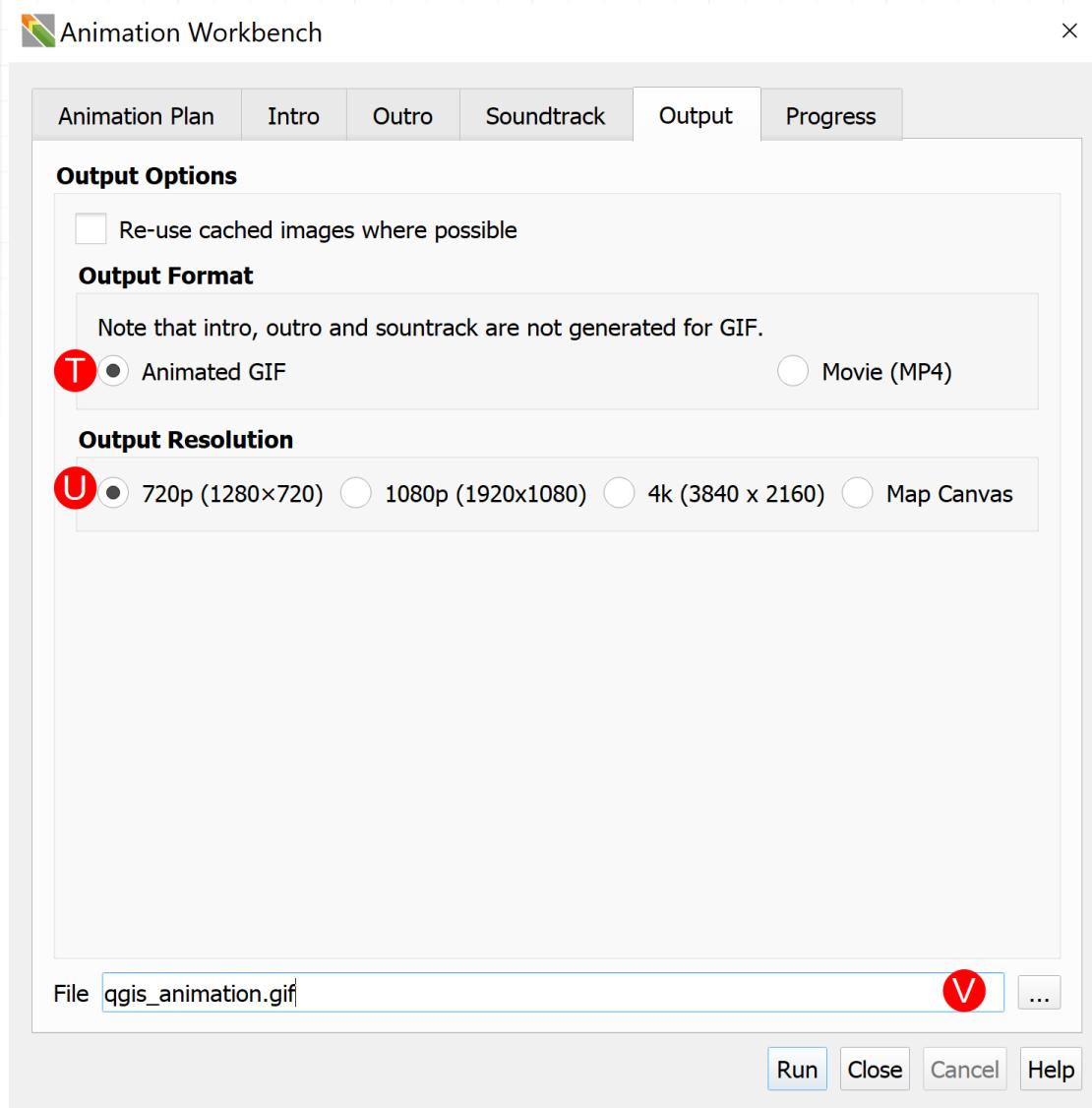
Run **Close** **Cancel** **Help**

"Bring your QGIS projects to life!"





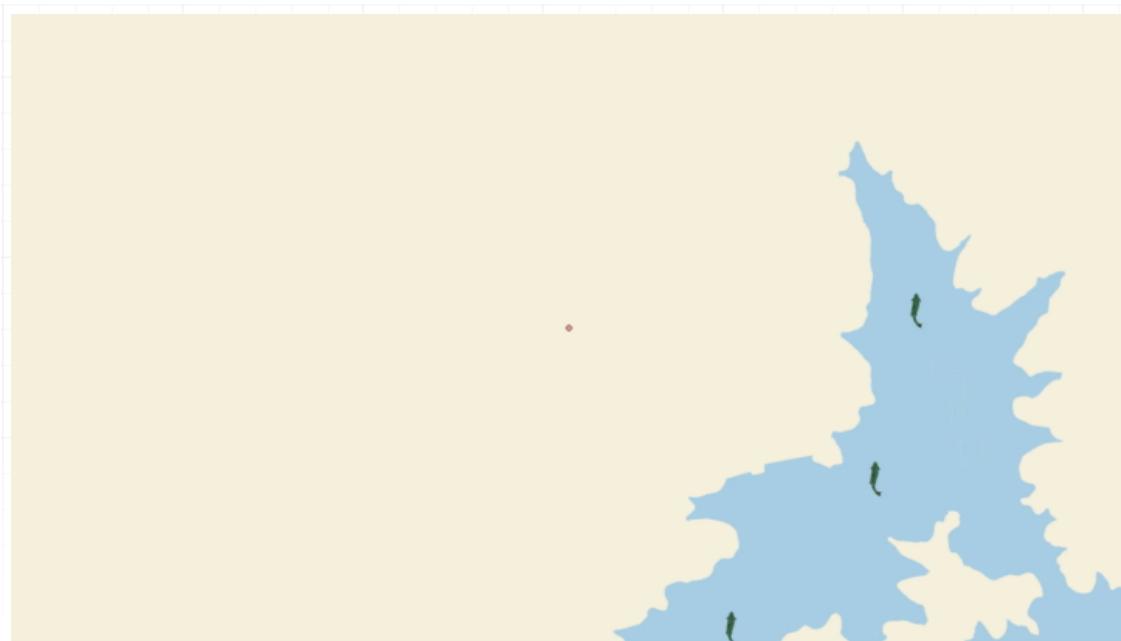
14. Skip past the **Intro**, **Outro**, and **Soundtrack** tabs to the **Output** tab. Set the **Output Format** as **Animated Gif** (**T**) and the **Output Resolution** to **720p (1280x720)** (**U**). The **Output Resolution** can be set as any of the three choices but was set at **720** for this tutorial for the sake of speed. Set the output location to one you can find easily (**V**)



15. Click **Run** and watch what the Workbench is doing in the **Progress** tab. Once the Workbench is finished running, you should end up with an output similar to this:

"Bring your QGIS projects to life!"





After this tutorial you should have a better understanding of how to create a point layer in your project and then to change the **Single Symbol** markers into stationary animated markers. A key focus is the idea that you can tell versions of **QGIS** before **3.26** to dynamically change markers using short code snippets. Versions of **QGIS** post **3.26** allow a user to simply use the **Animated Marker** feature without editing an expression.

"Bring your QGIS projects to life!"



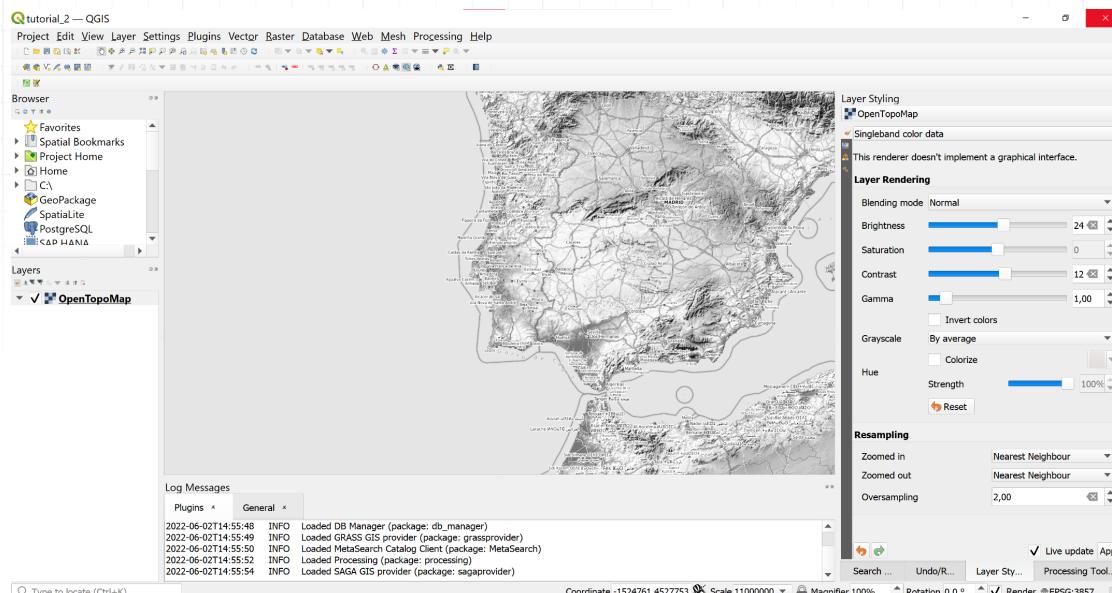


1.0.1 Tutorial 3: Flying Points

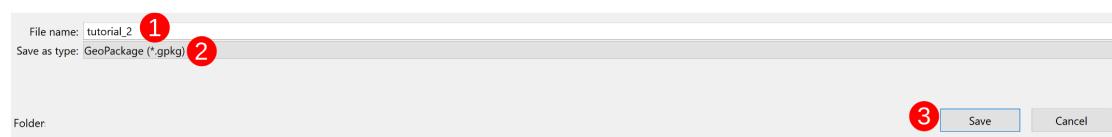
This tutorial aims to show you how add a flying point animation to points on your map using built-in QGIS functionalities (The geometry generator line) and introduced variables from the workbench.

1. Download and extract the [Required Tutorial Zip Folder](#)

2. Open the **tutorial_3.qgz** project file. When you first open the project file you should be greeted with something like this:



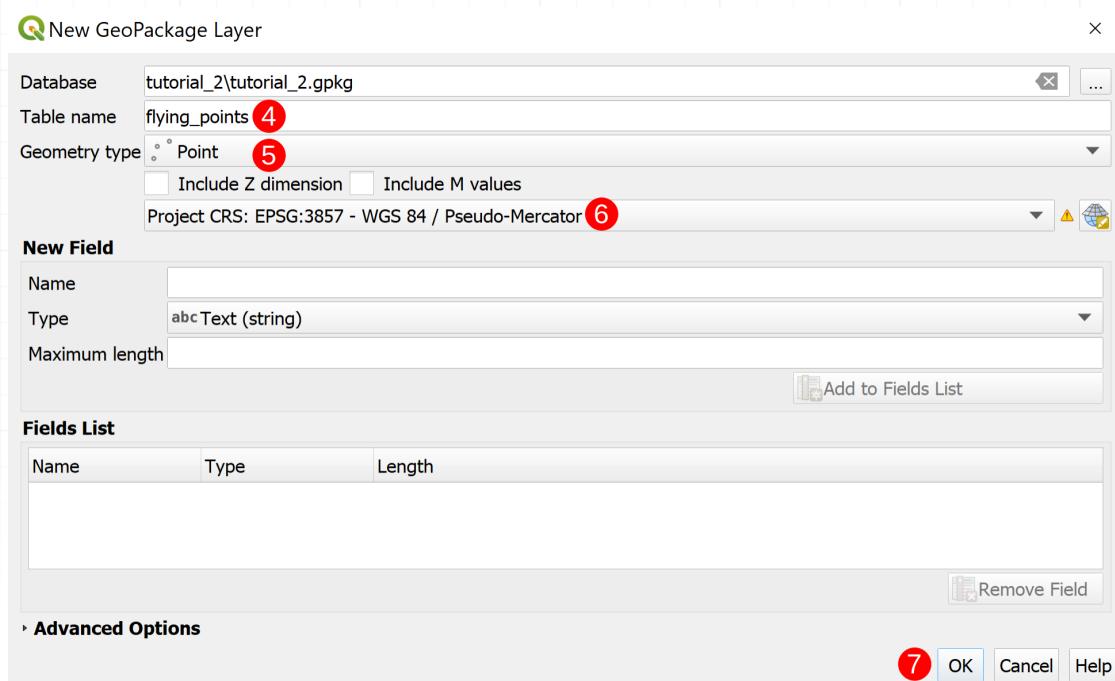
3. Create a new point layer in a new geopackage by clicking **Layer → Create Layer → New GeoPackage Layer...**. Click on the **Ellipsis** (three dots) next to the Database textbox and navigate to the folder that the **tutorial_3.qgz** file is located in. Type the File name "**tutorial_3**" (1) and ensure the file will be saved as a **GeoPackage** (2) and click **Save** (3).



Change the Table name to **flying_points** (4), set the Geometry type as **Point** (5) and change the CRS to match the **Project CRS** (6).

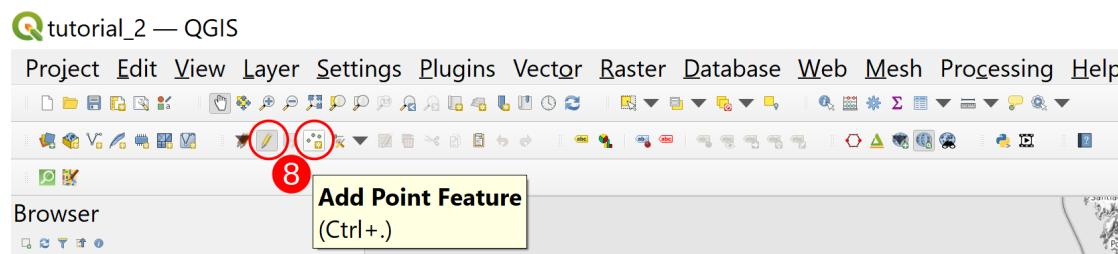
"Bring your QGIS projects to life!"





Click **OK** (7)

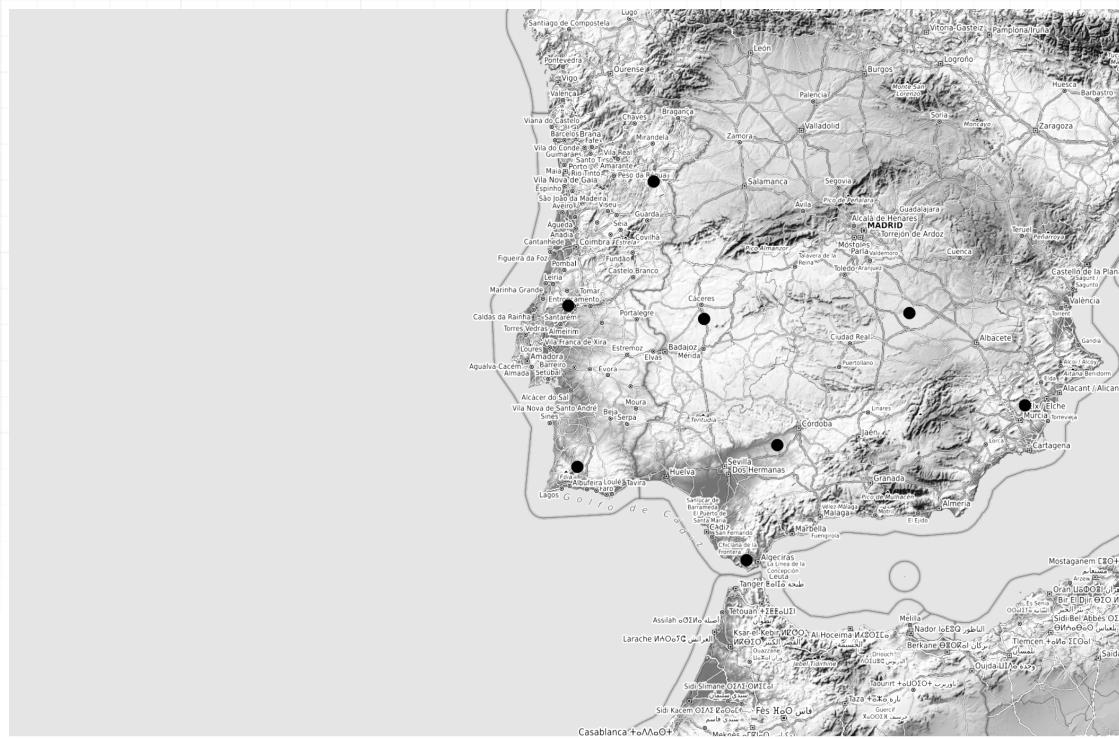
4. Click on **Toggle Editing** → **Add Point Feature** (8).



And randomly add points to your map. Depending on your computer's capabilities, you can add more, or fewer, points than the example below.

"Bring your QGIS projects to life!"

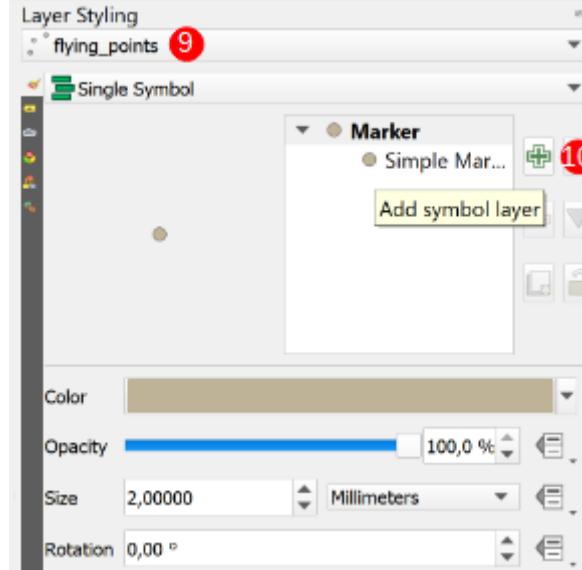




Save your Layer Edits and toggle off the Editing tool.

5. Style the points layer.

Select the **flying_points** (9) layer and in the **Layer Styling** toolbar click on the **Add Symbol Layer** (green plus symbol) button (10).

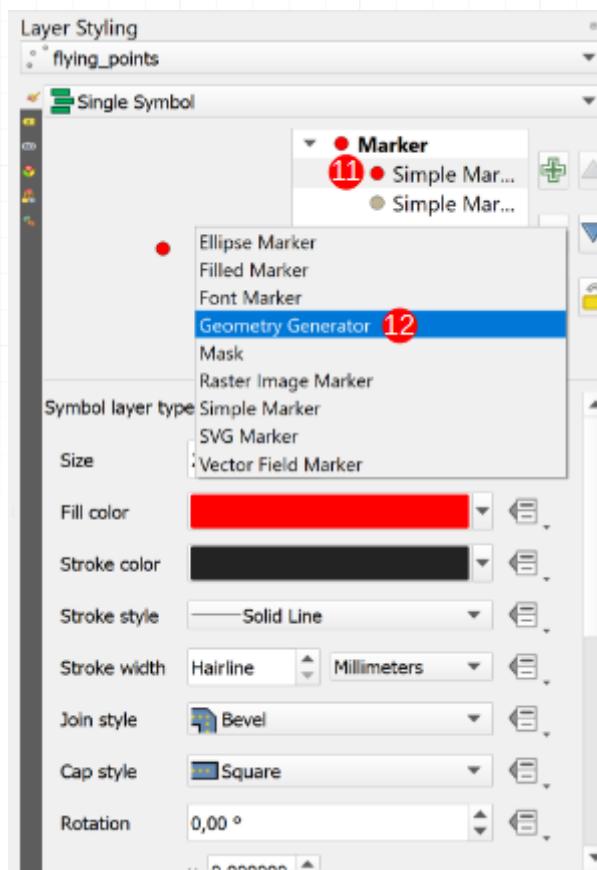


"Bring your QGIS projects to life!"

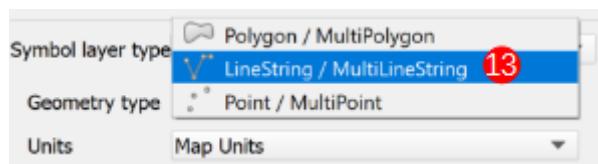




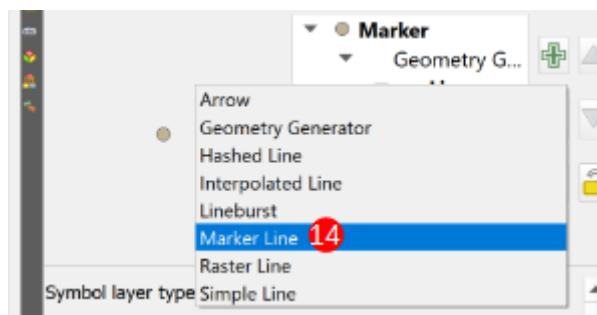
Select the top **Simple Marker** (11) and change its Symbol layer type to **Geometry Generator** (12).



and then set the Geometry type to **LineString / MultiLineString** (13).



Change the line's Symbol layer type to **Marker Line** (14).

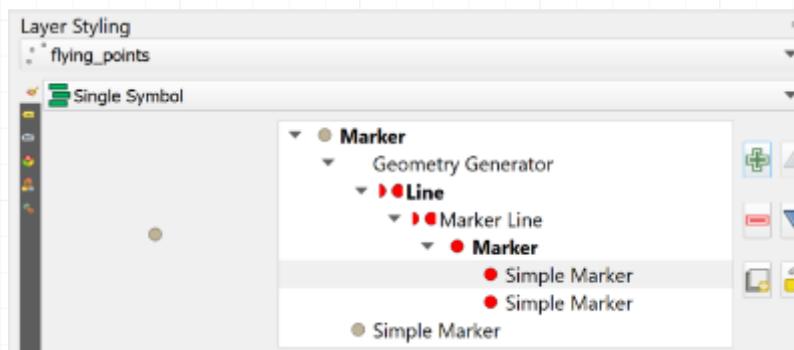


"Bring your QGIS projects to life!"





Add a second **Simple marker** to the marker line so that you end up with something like this:



Style the various **Simple Markers** to your preferred look.

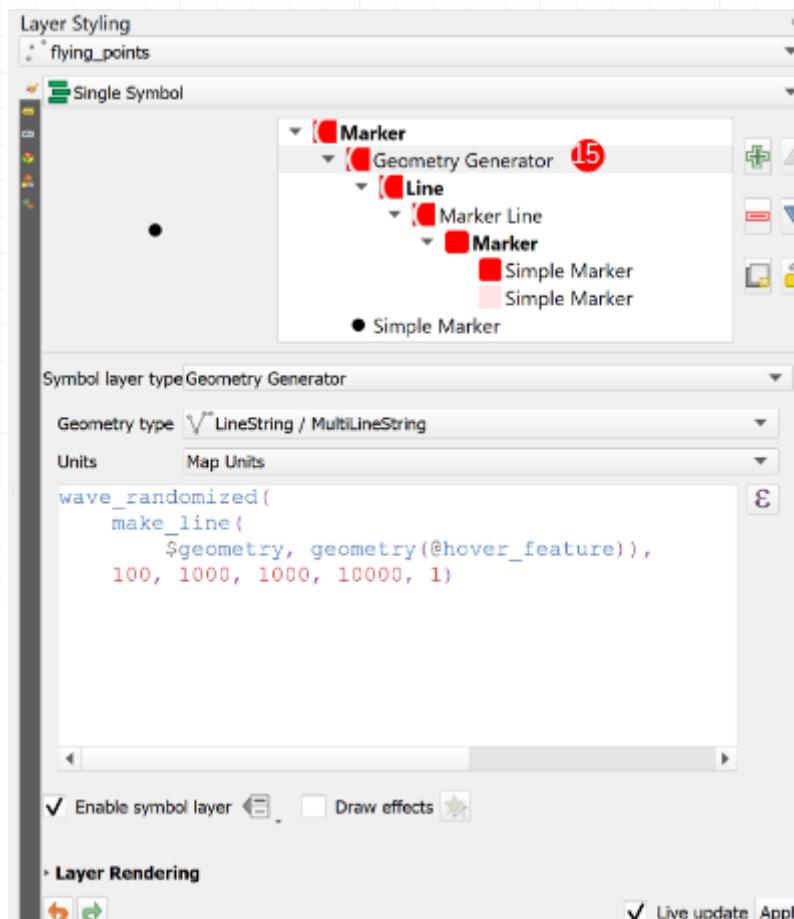
6. Select the **Geometry Generator** symbol layer (**15**) and add this code to it:

 **Code:**

```
wave_randomized(  
    make_line(  
        $geometry, geometry(@hover_feature)),  
        100, 1000, 1000, 10000, 1)
```

"Bring your QGIS projects to life!"





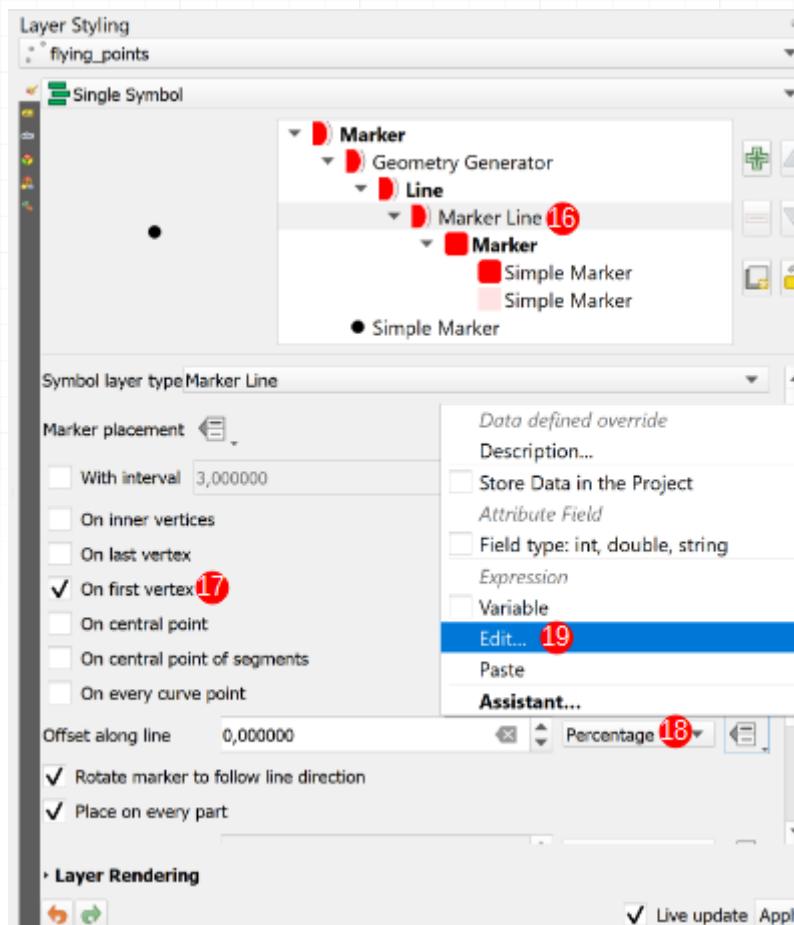
Note:

More information about what changing the numbers will affect can be found in the QGIS expressions editor.

7. A few options need to be changed in the **Marker Line** symbol layer (16): The Marker placement needs to be set to **On first vertex** (17) and, the Offset along line needs to be changed to **Percentage** (18). Then click the **Dropdown menu** next to Offset along line and select **Edit...** (19).

"Bring your QGIS projects to life!"





In the **Expression String Builder** add the following code snippet:

Code:

```
100 - to_int((@current_hover_frame / @hover_frames) * 100 )
```

```
100 - to_int((@current_hover_frame / @hover_frames)  
* 100 )
```

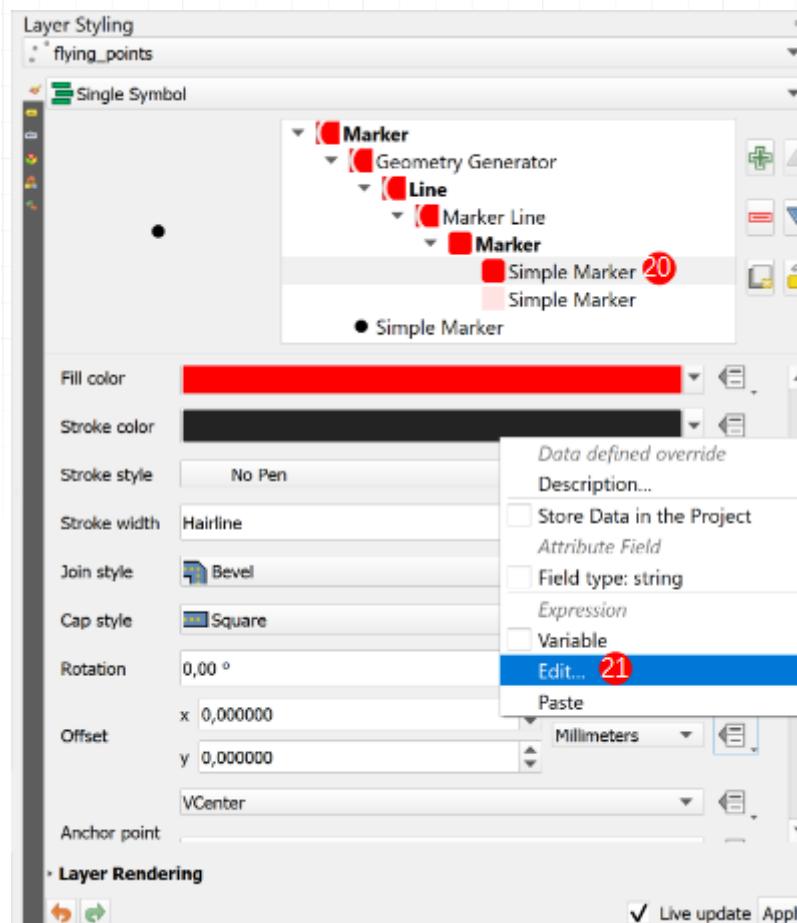
Click **OK**

"Bring your QGIS projects to life!"





8. Select the first **Simple Marker** symbol layer (20) in the **Marker Line** symbol layer. Scroll down to Offset and click on the **Dropdown Menu → Edit...** (21).



"Bring your QGIS projects to life!"





In the **Expression String Builder** add the following code snippet:

Code:

```
-- Taken from https://spicyyoghurt.com/tools/easing-functions
-- t = Time - Amount of time that has passed since the begin
-- b = Beginning value - The starting point of the animation
-- c = Change in value - The amount of change needed to go
-- d = Duration - Amount of time the animation will take. Us
-- Sinusoidal
-- -c / 2 * (Math.cos(Math.PI * t / d) - 1) + b;

-- Use with the animation in static mode
if(@hover_feature_id != $id,
array(
    (-@hover_frames / 2) * (cos( (pi() * @frame_number / @hove
    (-@hover_frames / 2) * (sin( (pi() * @frame_number / @hove
),
array (0,0))
```

"Bring your QGIS projects to life!"





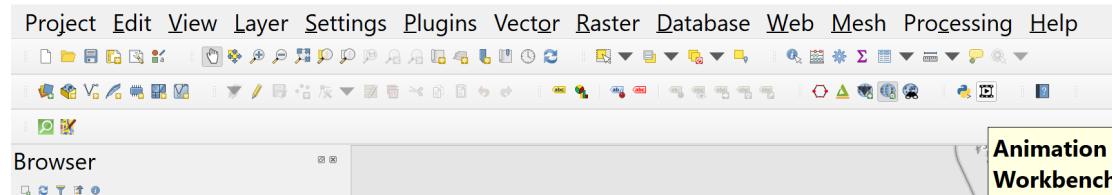
```
-- Taken from https://spicyyoghurt.com/tools/easing-functions
-- t = Time - Amount of time that has passed since the beginning of the
animation. Usually starts at 0 and is slowly increased using a game loop or
other update function.
-- b = Beginning value - The starting point of the animation. Usually
it's a static value, you can start at 0 for example.
-- c = Change in value - The amount of change needed to go from starting
point to end point. It's also usually a static value.
-- d = Duration - Amount of time the animation will take. Usually a
static value aswell.
-- Sinusoidal
-- -c / 2 * (Math.cos(Math.PI * t / d) - 1) + b;

-- Use with the animation in static mode
if(@hover_feature_id != $id,
array(
    (-@hover_frames / 2) * (cos( (pi() * @frame_number / @hover_frames ) - 1
)) ,
    (-@hover_frames / 2) * (sin( (pi() * @frame_number / @hover_frames ) - 1
))
),
array (0,0))
```

Click **OK**

9. Open the **Animation Workbench** (22)

tutorial_2 — QGIS



10. Set up the **Animation Plan** with:

- the **Render Mode** to **Planar** (23),
- the **Animation Layer** to **flying_points** (24) using the dropdown menu,
- the **Zoom Range** (25) to 1:22000000 for the Minimum and 1:11000000 for the Maximum,
- the **Frame rate per second** to 9 fps (26),
- the **Travel duration** to 2,00 s (27),
- the **Feature hover duration** to 2,00 s (28),
- and the **Zoom Easing** as InCirc (29)

"Bring your QGIS projects to life!"





Animation Workbench

Animation Plan Intro Outro Soundtrack Output Progress

23 Render Mode
 Sphere Planar Fixed Extent

24 Animation Layer
flying_points Loop from final feature back to first feature

25 Zoom Range
Minimum (exclusive) Maximum (inclusive)
1:22000000 1:11000000

Data Defined Settings
Scale Minimum Maximum

Animation Frames
Frame rate per second: 9 fps **26**
Travel duration: 2,00 s **27**
Feature hover duration: 2,00 s **28**

Pan and Zoom Easings
 Enable Pan Easing Linear
 Enable Zoom Easing **29** InCirc

Frame Preview

Frame: 0

Run Close Cancel Help

Note:

With a decently specced computer you can up the fps and get the points to fly faster in your output.

"Bring your QGIS projects to life!"





11. Add license-free media to the **Intro**, **Outro**, and **Soundtrack**.

 **Note:**

Make sure your **Soundtrack** is as long as, or longer than, your final animation will be (including the **Intro**, **Animation**, and **Outro**).

12. Set the **Output Format** as **Movie (MP4)** (**30**) and the **Output Resolution** to **1080 (1920x1080)** (**31**). The **Output Resolution** can be set as any of the three choices but was set at **1080** for this tutorial for the sake of speed. Set the output location (**32**) to one you can easily locate.

"Bring your QGIS projects to life!"





Animation Workbench X

Animation Plan Intro Outro Soundtrack Output Progress

Output Options

Re-use cached images where possible

Output Format

Note that intro, outro and soundtrack are not generated for GIF.

Animated GIF 31 Movie (MP4)

Output Resolution 32

720p (1280x720) 1080p (1920x1080) 4k (3840 x 2160) Map Canvas

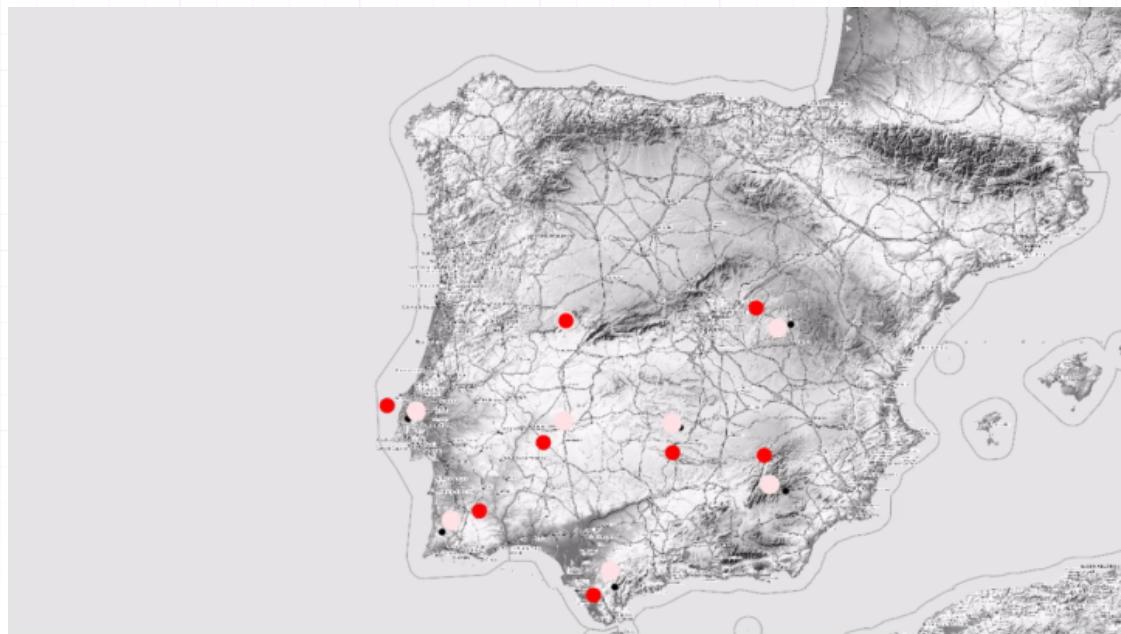
File qgis_animation.mp4 33 ...

Run Close Cancel Help

13. Click **Run** and get an output. The **GIF** below is the visual output of the tutorial if you followed step-by-step and set the parameters to exactly what was stated.

"Bring your QGIS projects to life!"





The link to a more complex output (with an **Intro**, an **Outro**, and a **Soundtrack**) can be found [here](#).

After this tutorial you should have a better idea of how you can use a mixture of built-in QGIS functionalities and the workbench's introduced variables to generate interesting outputs.

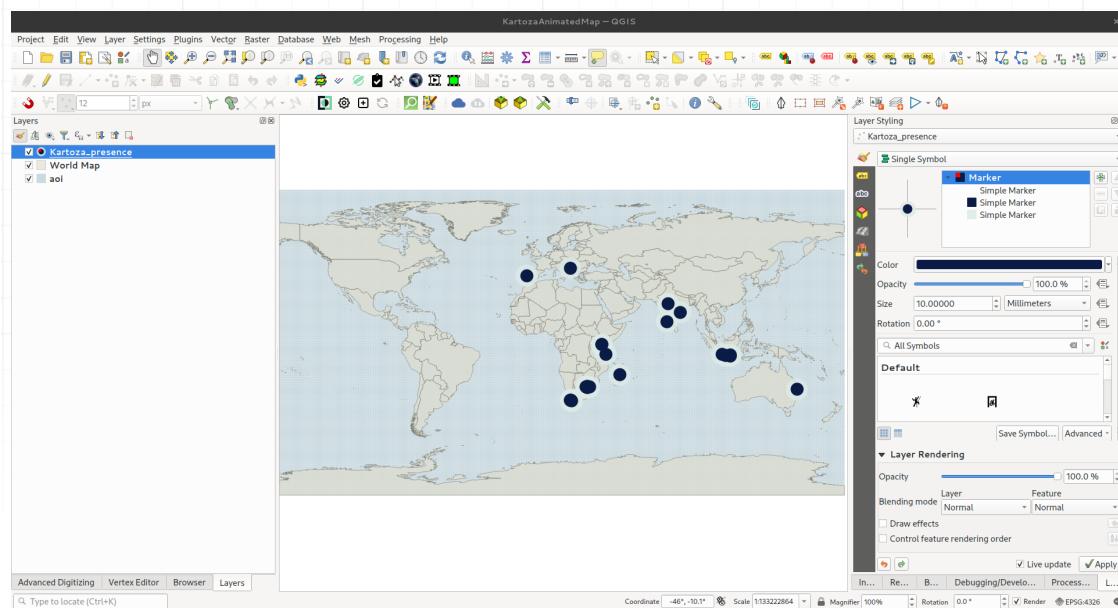
"Bring your QGIS projects to life!"





1.0.1 Tutorial 4: Spinning Globe

Given a global point layer and countries layer like this:



You can create a nice spinning globe effect like this:

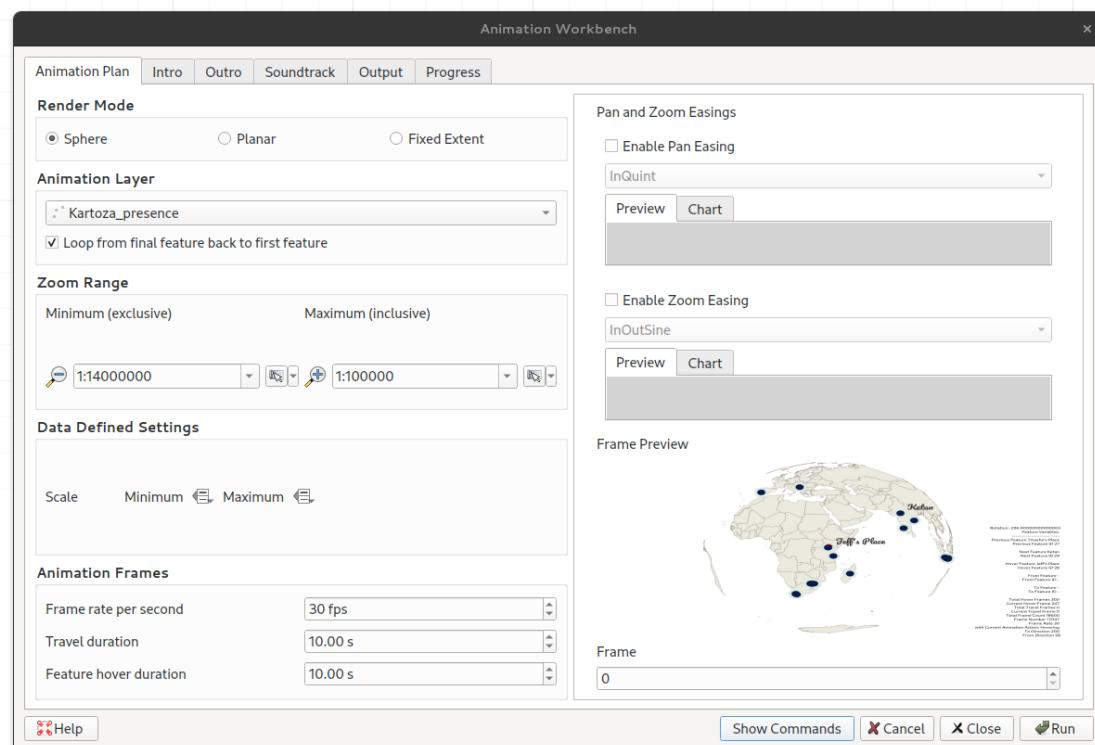


"Bring your QGIS projects to life!"





I set up the animation workbench like this:



For the above animated GIF, I compressed it using imagemagick like this:

 **Code :**

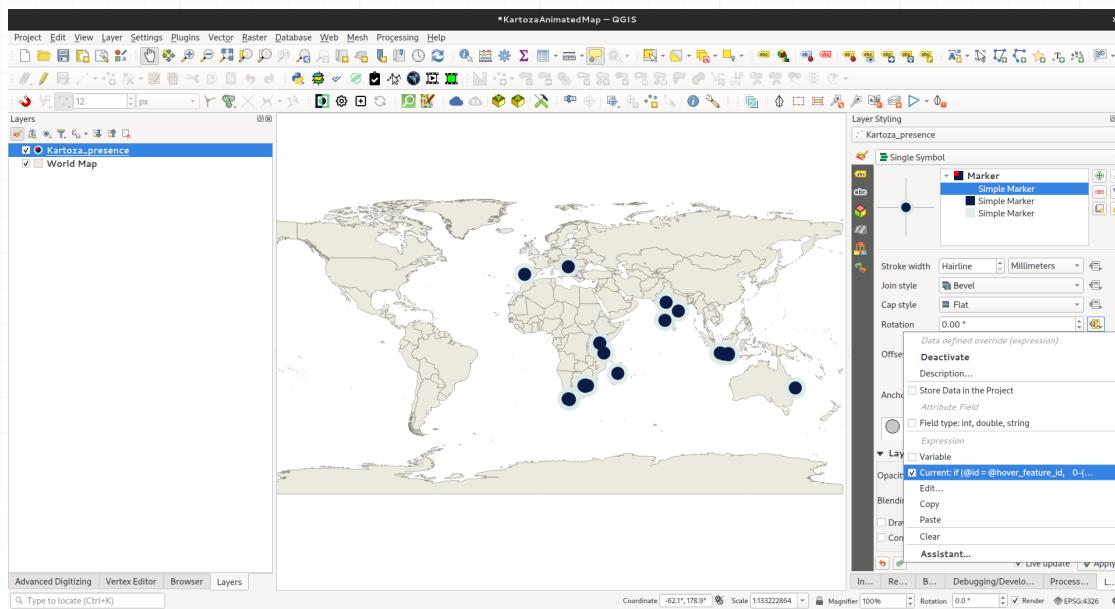
```
convert globe.gif -coalesce -resize 700x525 -fuzz 2% +dither -laye
```

This is a handy technique if you want to generate small file size animations.

For the points I made a red marker using a quarter circle that spins around the points like this:

"Bring your QGIS projects to life!"





The rotation field expression is this:

Code :

```
if (@id = @hover_feature_id,  
    0-((1440 * (@current_hover_frame/@hover_frames)) % 360),  
    0)
```

This will spin around 4 times during the hover cycle.

For the ocean (AOI in the layers list), I generated a grid of 1 degree cells covering the earth. You need to do it as smaller polygons instead of one large polygon because QGIS will run into issues reprojecting a single polygon whose edges lie on the date line.

Here is how the final video came out:

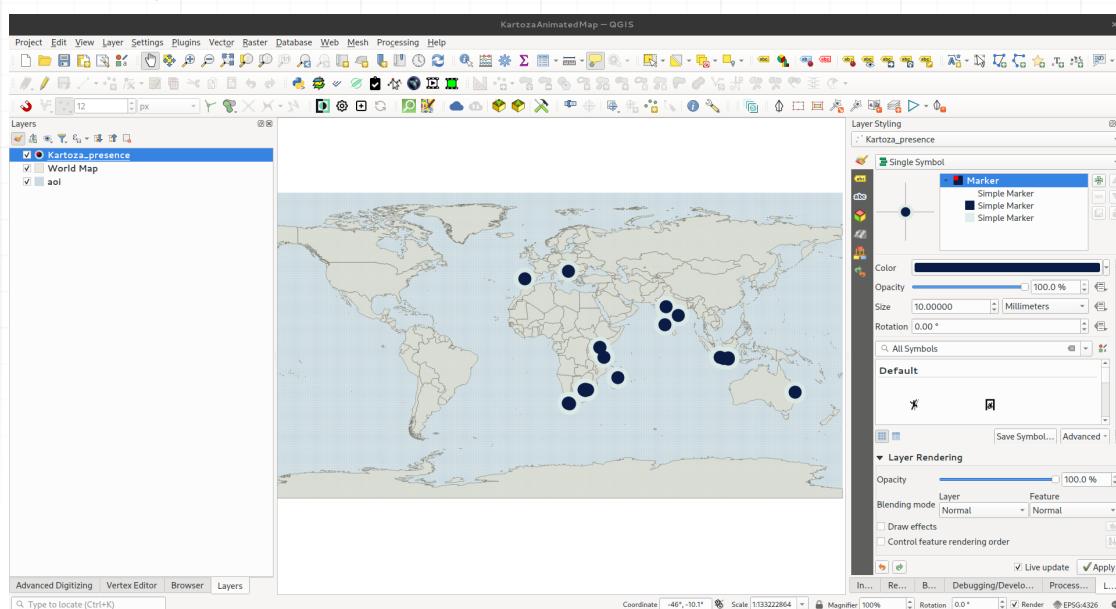
"Bring your QGIS projects to life!"



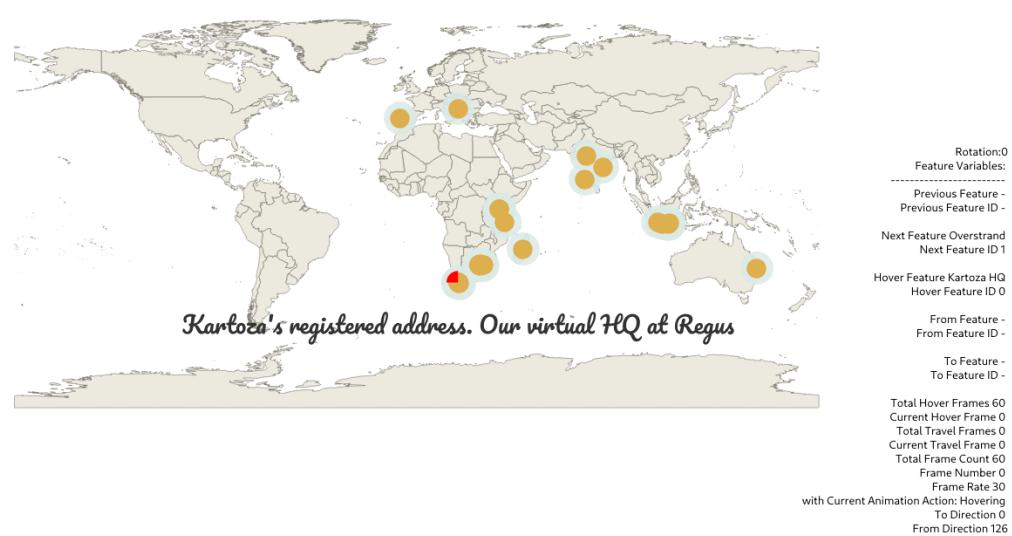


1.1 Tutorial 5: Planar Map Animation

Given a global point layer and countries layer like this:



You can create a nice planar map animation effect like this:



"Bring your QGIS projects to life!"

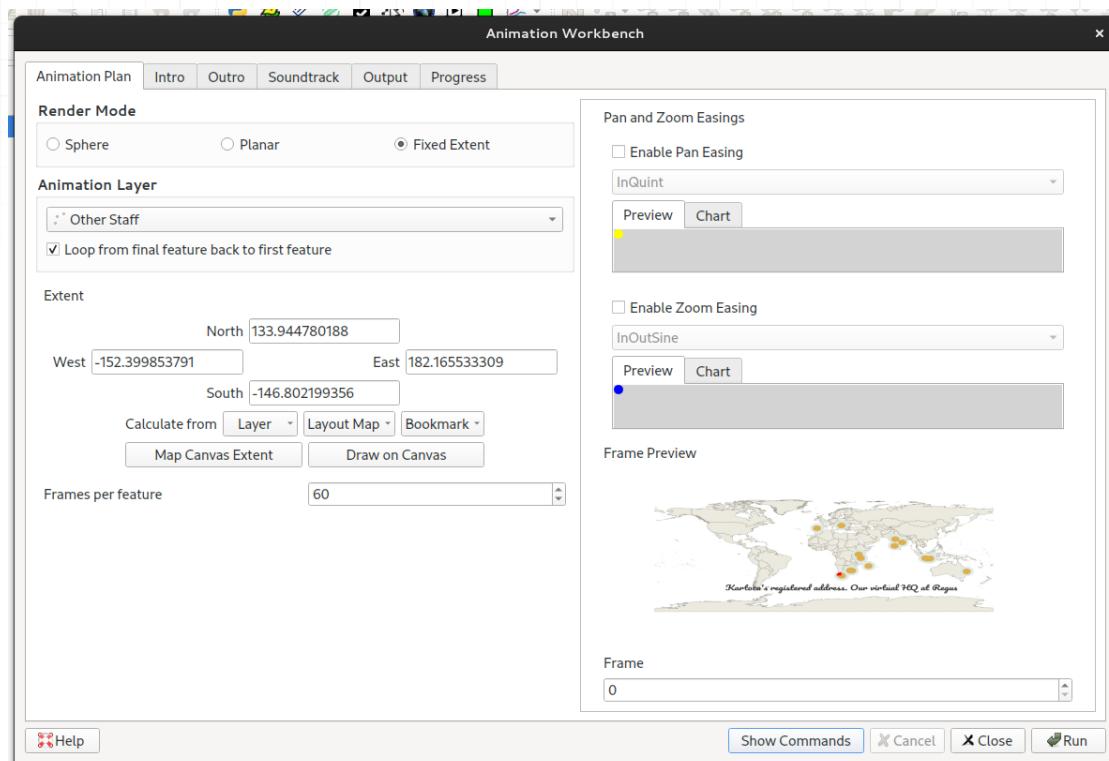


In planar mode, we do not pan and zoom the map from feature to feature. Rather, the map zoom stays constant and the variables for

- current_hover_frame
- hover_frames
- hover_feature_id

are updated as we iterate over the features of your animation layer. In this example project I duplicated the animation point layer twice. The first (lower) copy is used to 'drive' the animation, whilst the second (upper) layer shows only the feature currently being hovered over, with animation effects applied to that feature.

I set up the animation workbench like this:



For the above animated GIF, I compressed it using imagemagick like this:



```
convert globe.gif -coalesce -resize 700x525 -fuzz 2% +dither -laye
```

This is a handy technique if you want to generate small file size animations.

"Bring your QGIS projects to life!"





1.1.1 Expressions Used Copyright Decoration

"Bring your QGIS projects to life!"





Firstly for debugging, we use the following copyright label in View ⇒ Decorations ⇒ Copyright Label. You can use the checkbox in the Copyright configuration dialog to toggle this on and off. This will help you while debugging / tweaking your animations. When you are ready to render your final product, simply turn it off before rendering.

"Bring your QGIS projects to life!"





Code:

```
[%
' \nRotation:' || to_string( 0-((1440 * (@current_hover_frame/@ho
'\nFeature Variables:' ||
' \n-----' ||
' \nPrevious Feature ' || to_string(coalesce(attribute(@previous_f
' \nPrevious Feature ID ' || to_string(coalesce(@previous_feature_
' \n' ||
' \nNext Feature ' || to_string(coalesce(attribute(@next_feature,
' \nNext Feature ID ' || to_string(coalesce(@next_feature_id, '-'))
' \n' ||
' \nHover Feature ' || to_string(coalesce(attribute(@hover_feature
' \nHover Feature ID ' || to_string(coalesce(@hover_feature_id, '-
' \n' ||
' \nFrom Feature ' || to_string(coalesce(attribute(@from_feature,
' \nFrom Feature ID ' || to_string(coalesce(@from_feature_id, '-'))
' \n' ||
' \nTo Feature ' || to_string(coalesce(attribute(@to_feature, 'nam
' \nTo Feature ID ' || to_string(coalesce(@to_feature_id, '-')) ||
' \n' ||
' \nTotal Hover Frames ' || to_string(coalesce(@hover_frames, 0))
' \nCurrent Hover Frame ' || to_string(coalesce(@current_hover_fra
' \nTotal Travel Frames ' || to_string(coalesce(@travel_frames, 0))
' \nCurrent Travel Frame ' || to_string(coalesce(@current_travel_f
' \nTotal Frame Count ' || to_string(coalesce(@total_frame_count, 0
' \nFrame Number ' || to_string(coalesce(@frame_number, 0)) ||
' \nFrame Rate ' || to_string(coalesce(@frame_rate, 0)) ||
' \nwith Current Animation Action: ' || @current_animation_action
' \nTo Direction ' || coalesce(format_number(degrees(azimuth( geo
' \nFrom Direction ' || coalesce(format_number(degrees( azimuth(
```

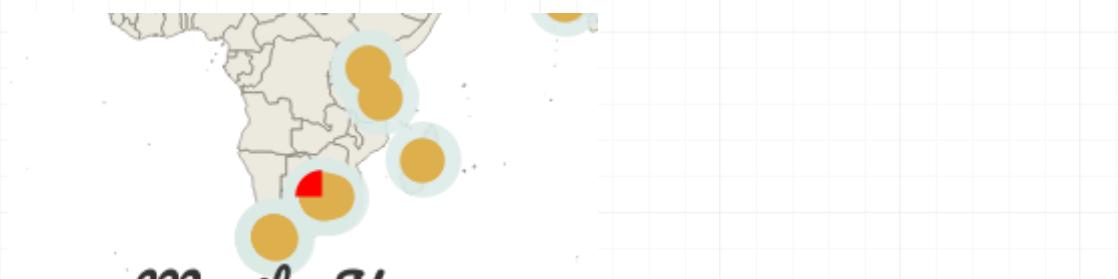
Symbol Rotation

"Bring your QGIS projects to life!"





For the points I made a red marker using a quarter circle that spins around the points like this:



The first line of the listing from the previous section gives you a hint about how we can vary the rotation of a symbol depending on how far through the animation sequence we are. With the addition of an `if` clause, we can apply this rotation only to features that are being hovered over during the planar animation.

Code:

```
if (
    @id = @hover_feature_id,
    0-((1440 * (@current_hover_frame/@hover_frames)) % 360),
    0)
```

This `if` clause has the effect of excluding calculation for any feature that is not the current hover feature.

This will spin around 4 times during the hover cycle. This is because four rotations are $4 \times 360 = 1440$. We calculate the percentage of completion for the current hover frame (`@current_hover_frame/@hover_frames`) and then multiply our rotation product by the current completion percentage. Lastly we calculate the modulus of this (`% 360`) to compute how far along we are in the current rotation. More advanced users could substitute 1440 with a project variable so that it is easy to change the number of desired rotations in a single place.

Symbol Size

"Bring your QGIS projects to life!"





The rotating symbol layer and the other symbol layers in our animation layer are similarly hidden if the feature being rendered is not the `hover_feature_id` using an expression like this:

 **Code:**

```
if ( @id = @hover_feature_id, 10, 0)
```

This has the effect of setting the symbol size to 0 if it is not the feature we are focussing on.

1.1.2 Other Planar Experiments

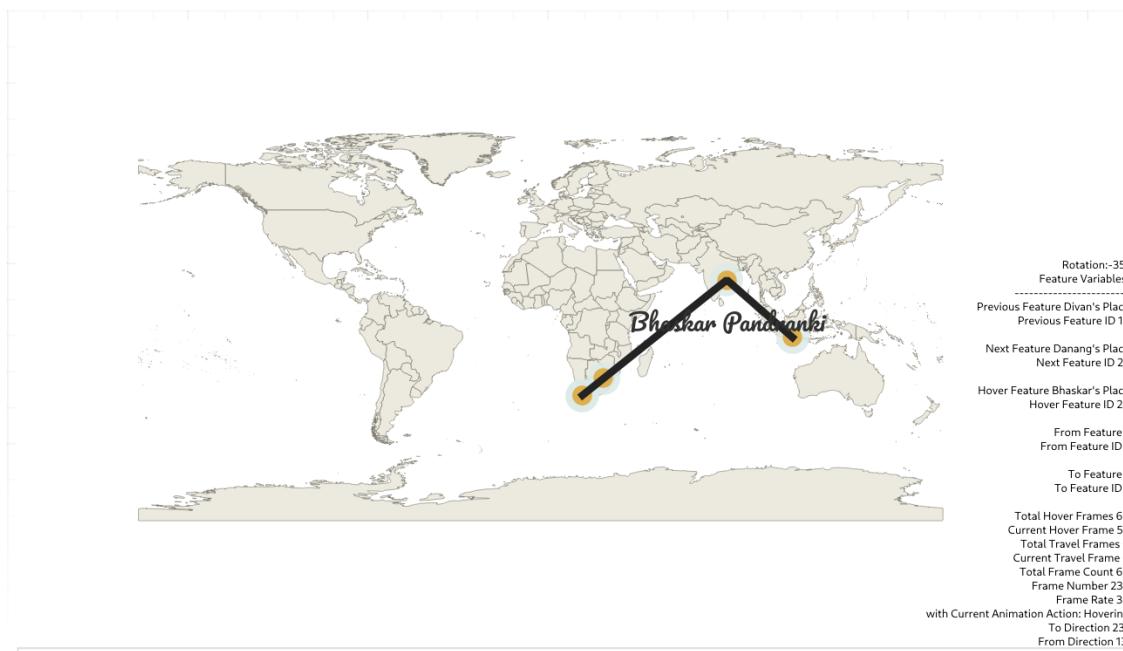
With the basic concepts of working with planar animations covered above, you can do other interesting things.

Generate a line

In this example, we can generate a line using the Geometry Generator function in QGIS. The line will start from the previous point, extend through the current point and terminate at the next point.

"Bring your QGIS projects to life!"





Code:

```
if (
    $id = @hover_feature_id,
    make_line(
        geometry(@previous_feature),
        geometry(@hover_feature),
        geometry(@next_feature)
    ),
    $geometry)
```

We wrap it in an if clause again so that the line is not rendered if the current feature being rendered is not the same as the current animation feature.

There may be some edge cases where there is no previous or next feature. This example does not try to deal with these cases but you could easily add some logic that checks if each of the three components making up the line is null or not.

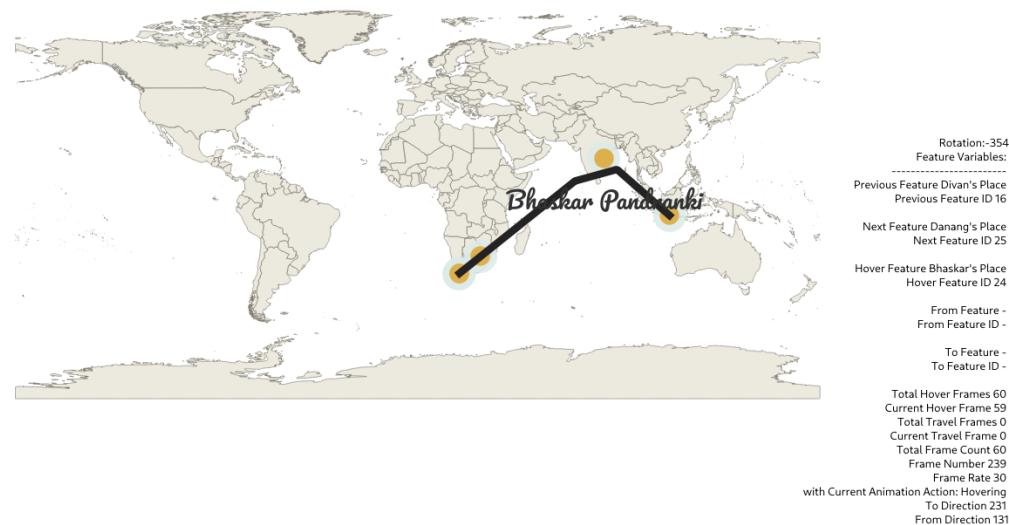
Generate a curve

"Bring your QGIS projects to life!"





We can extend the above example by creating a curve rather than a line, for a more natural looking connection between the hover feature and its previous and following feature.



Code:

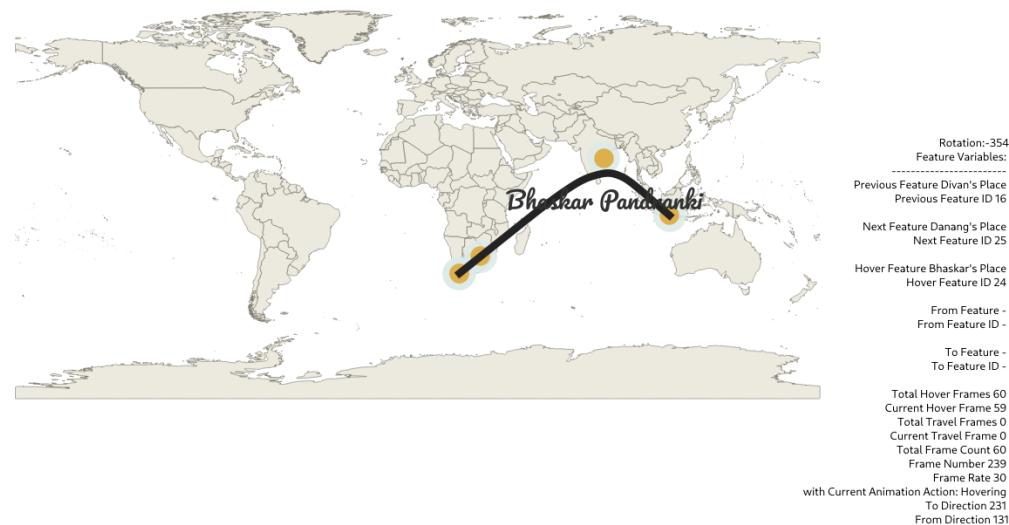
```
if (  
    $id = @hover_feature_id,  
    smooth(  
        make_line(  
            geometry(@previous_feature),  
            geometry(@hover_feature),  
            geometry(@next_feature)  
        ),  
        iterations:=1,  
        offset:=0.2,  
        min_length:=-1,  
        max_angle:=180),  
    $geometry)
```

"Bring your QGIS projects to life!"





If you increase the number of iterations, you can achieve a more and more smoothed out line, at the expense of processing time.



Code:

```
if (
    $id = @hover_feature_id,
    smooth(
        make_line(
            geometry(@previous_feature),
            geometry(@hover_feature),
            geometry(@next_feature)
        ),
        iterations:=5,
        offset:=0.2,
        min_length:=-1,
        max_angle:=180),
    $geometry)
```

"Bring your QGIS projects to life!"





Subtring the Line

As a much more advanced example, you can extract a substring of the smoothed line that connects the previous, current and next features. Don't get put off by the `with_variable` elements - they just allow us to re-use calculations in our expression.

First, let's start with extracting the first half of the smoothed line:

"Bring your QGIS projects to life!"





"Bring your QGIS projects to life!"





Code:

```
if (
    $id = @hover_feature_id,
    with_variable(
        'smoothed_line',
        smooth(
            make_line(
                geometry(@previous_feature),
                geometry(@hover_feature),
                geometry(@next_feature)
            ),
            iterations:=5,
            offset:=0.2,
            min_length:=-1,
            max_angle:=180),
            with_variable(
                'line_length',
                length(@smoothed_line),
                line_substring(@smoothed_line, 0, @line_length / 2 ))),
    $geometry)
```

Animating the substring

If we follow the same approach as above, but vary the start and length of the line clip, we can create some cool line animation effects.

"Bring your QGIS projects to life!"





"Bring your QGIS projects to life!"





Code:

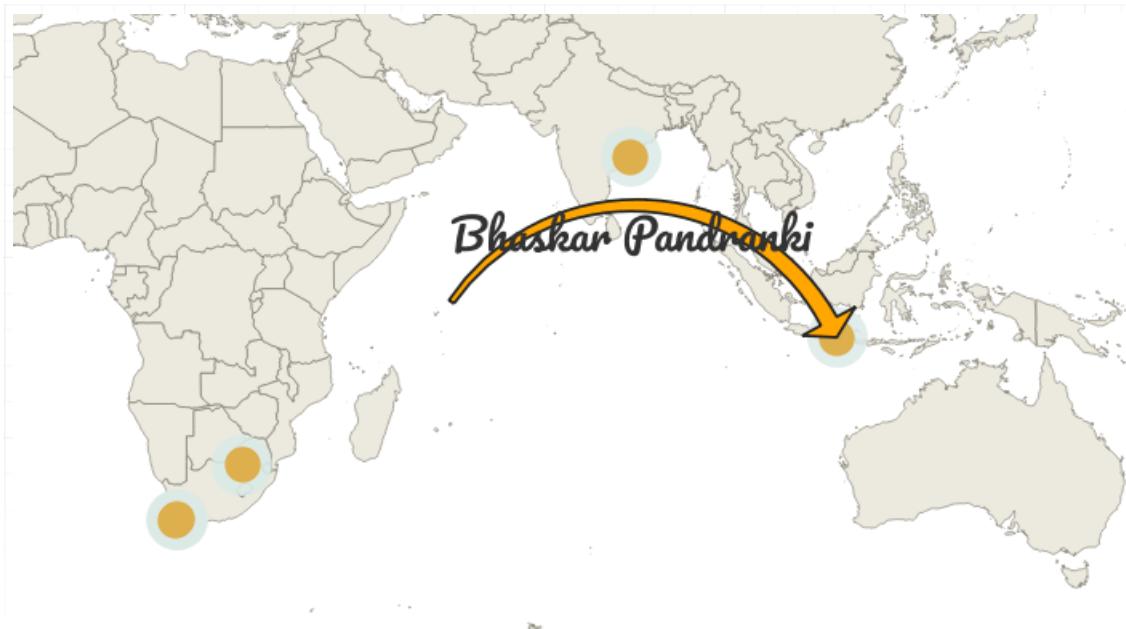
```
if (
    $id = @hover_feature_id,
    with_variable(
        'smoothed_line',
        smooth(
            make_line(
                geometry(@previous_feature),
                geometry(@hover_feature),
                geometry(@next_feature)
            ),
            iterations:=5,
            offset:=0.2,
            min_length:=-1,
            max_angle:=180),
            with_variable(
                'line_length',
                length(@smoothed_line),
                line_substring(
                    @smoothed_line,
                    @line_length * (@current_hover_frame/@hover_fr.
                    @line_length ))),
            $geometry)
```

1.1.3 Final Render

There are still a few details that would need to be taken care of to reach a final solution - in particular taking care of datelines and the like. But here is a little example of what we managed to make thus far (without any debugging text).

"Bring your QGIS projects to life!"





"Bring your QGIS projects to life!"



1 Library

1.1 QGIS Expression Variables

The animation workbench exposes or modifies a number of different QGIS Expression variables that you can use to achieve different dynamic rendering effects.

1.1.1 Common variables

These variables will always be available, regardless of the animation mode

Variable	Notes
frame_number	Frame number within the current dwell or pan range.
frame_rate	Number of frames per second that the video will be rendered at.
total_frame_count	Total number of frames for the whole animation across all features.

1.1.2 Fixed extent mode variables (with layer)

These variables are available when in the fixed extent animation mode when a vector layer has been set

Variable	Notes
hover_feature	The feature we are currently hovering over
hover_feature_id	Feature ID for the feature we are currently hovering over
previous_feature	The previously visited feature (or NULL if there isn't one)
previous_feature_id	Feature ID for the previously visited feature (or NULL if there isn't one)
next_feature	The next feature to visit after the current one (or NULL if there isn't one)
next_feature_id	Feature ID for the next feature to visit after the current one (or NULL if there isn't one)
current_hover_frame	The frame number for the current feature (i.e. how many frames we have hovered at the current feature)
hover_frames	Number of frames we will hover at the current feature for
current_animation_action	Always "Hovering"

1.1.3 Planar/Sphere modes

These variables are available in the Planar or Sphere mode.

Variable	Notes
current_animation_action	Either "Hovering" or "Travelling"

When hovering

"Bring your QGIS projects to life!"





These variables are available in planar or sphere mode, when the animation is currently hovering over a feature

Variable	Notes
hover_feature	The feature we are currently hovering over
hover_feature_id	The feature ID for the feature we are currently hovering over
previous_feature	The previously visited feature (or NULL if there isn't one)
previous_feature_id	Feature ID for the previously visited feature (or NULL if there isn't one)
next_feature	The next feature to visit after the current one (or NULL if there isn't one)
next_feature_id	Feature ID for the next feature to visit after the current one (or NULL if there isn't one)
current_hover_frame	The frame number for the current feature (i.e. how many frames we have hovered at the current feature)
hover_frames	Number of frames we will hover at the current feature for

When travelling

These variables are available in planar or sphere mode, when the animation is currently travelling between two features

Variable	Notes
from_feature	The feature we are travelling away from
from_feature_id	The feature ID for the feature we are travelling away from
to_feature	The feature we are heading toward
to_feature_id	The feature ID for the feature we are heading toward
current_travel_frame	The frame number for the current travel operation
travel_frames	Number of frames we will travel between the current features

1.1.4 Example expressions

Visit the [snippets section](#) of our documentation for example expressions.

"Bring your QGIS projects to life!"





1.2 Snippets

1.2.1 QGIS Support

Should work with and version of QGIS 3.x. If you have QGIS 3.26 or better you can benefit from the animated icon support (see @nyalldawson's most excellent patch [#48060](#)).

For QGIS versions below 3.26, you can animate markers by unpacking a GIF image into its constituent frames and then referencing a specific frame from the symbol data defined property for the image file. Note that to do this extraction below you need to have the [Open Source ImageMagick application](#) installed:

First extract a gif to a sequence of images:

Code:

```
convert cat.gif -coalesce cat_%05d.png
```

Example of how to create a dynamically changing image marker based on the current frame count:

Code:

```
@project_home
 ||
 '/gifs/cat_000'
 ||
 lpad(to_string( @frame_number % 48 ), 2, '0')
 ||
 '.png'
```

Note that for the above, 48 is the number of frames that the GIF was composed of, and it assumes the frames are in the project directory in a subfolder called **gifs**.

1.2.2 Line of travel

"Bring your QGIS projects to life!"





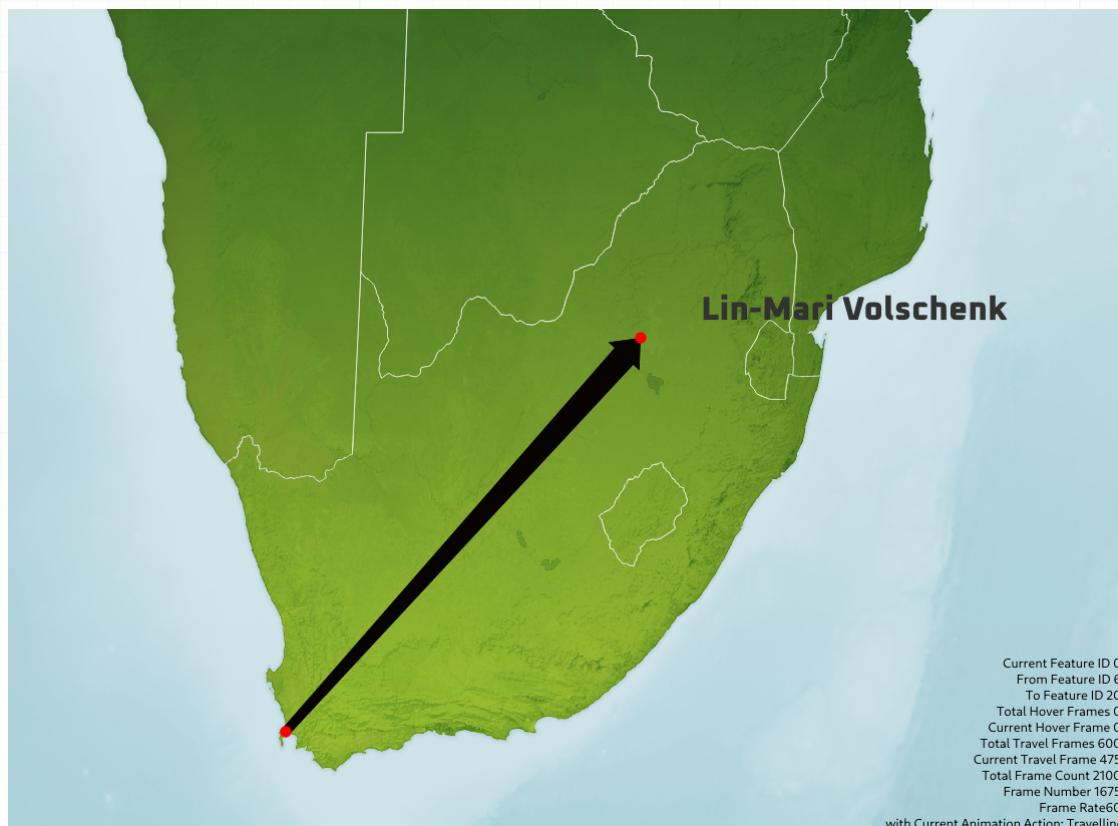
In this example we use a geometry generator to create a line between the origin point and the destination point:

 **Code:**

```
if (@from_feature_id = $id OR @to_feature_id = $id,
    -- read this from inside to out so
    -- last tranform the geometry back to the map crs
    transform(
        -- densify the geometry so that when we transform
        -- back it makes a great circle
        densify_by_count(
            -- move the geometry into a crs that
            -- shows a great circle as a straight line
            transform(
                -- make a line from the previous pont to the next point
                make_line(
                    geometry(@from_feature),
                    geometry(@to_feature)
                ),
                @map_crs, 'EPSG:4326'),
                99),
                'EPSG:4326', @map_crs),
                None)
```

"Bring your QGIS projects to life!"





1.2.3 Showing diagnostic info as a copyright label

"Bring your QGIS projects to life!"





Showing diagnostic information in the QGIS copyright label:

"Bring your QGIS projects to life!"





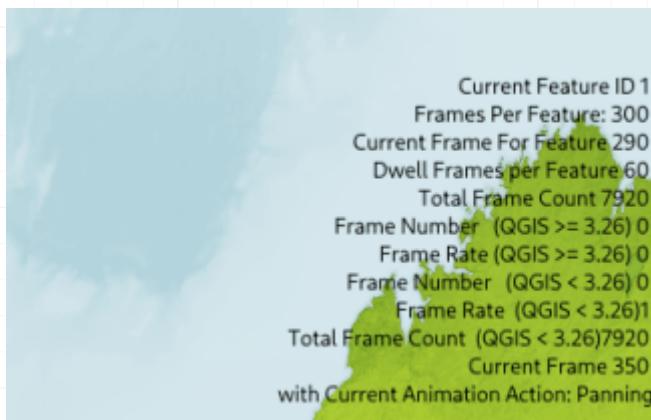
Code:

```
[%
'Feature Variables:' ||
' \n-----' ||
' \nPrevious Feature ' || to_string(coalesce(attribute(@previous_feature),
' \nPrevious Feature ID ' || to_string(coalesce(@previous_feature_id,
' \n' ||
' \nNext Feature ' || to_string(coalesce(attribute(@next_feature,
' \nNext Feature ID ' || to_string(coalesce(@next_feature_id, '-')) ||
' \n' ||
' \nHover Feature ' || to_string(coalesce(attribute(@hover_feature,
' \nHover Feature ID ' || to_string(coalesce(@hover_feature_id, '-')) ||
' \n' ||
' \nFrom Feature ' || to_string(coalesce(attribute(@from_feature,
' \nFrom Feature ID ' || to_string(coalesce(@from_feature_id, '-')) ||
' \n' ||
' \nTo Feature ' || to_string(coalesce(attribute(@to_feature, 'name')) ||
' \nTo Feature ID ' || to_string(coalesce(@to_feature_id, '-')) ||
' \n' ||
' \nTotal Hover Frames ' || to_string(coalesce(@hover_frames, 0)) ||
' \nCurrent Hover Frame ' || to_string(coalesce(@current_hover_frame,
' \nTotal Travel Frames ' || to_string(coalesce(@travel_frames, 0)) ||
' \nCurrent Travel Frame ' || to_string(coalesce(@current_travel_frame,
' \nTotal Frame Count ' || to_string(coalesce(@total_frame_count, 0)) ||
' \nFrame Number ' || to_string(coalesce(@frame_number, 0)) ||
' \nFrame Rate ' || to_string(coalesce(@frame_rate, 0)) ||
' \nwith Current Animation Action: ' || @current_animation_action ||
' \nTo Direction ' || coalesce(format_number(degrees(azimuth( geometry)),
' \nFrom Direction ' || coalesce(format_number(degrees( azimuth( geometry)),
%]
```





Example output:



1.2.4 Variable size of labels

Variably changing the size on a label as we approach it in the animation:

```
```40 * (@frame_number % @hover_frames) / @hover_frames)
```

**Code:**

```
Calculating the angle between points
```

You can calculate the angle between the hover point and the previous point.

```
```python
coalesce(
    format_number(
        degrees(
            azimuth(
                geometry(@hover_feature),
                geometry(@previous_feature)
            )
        )
    ), 0)
```

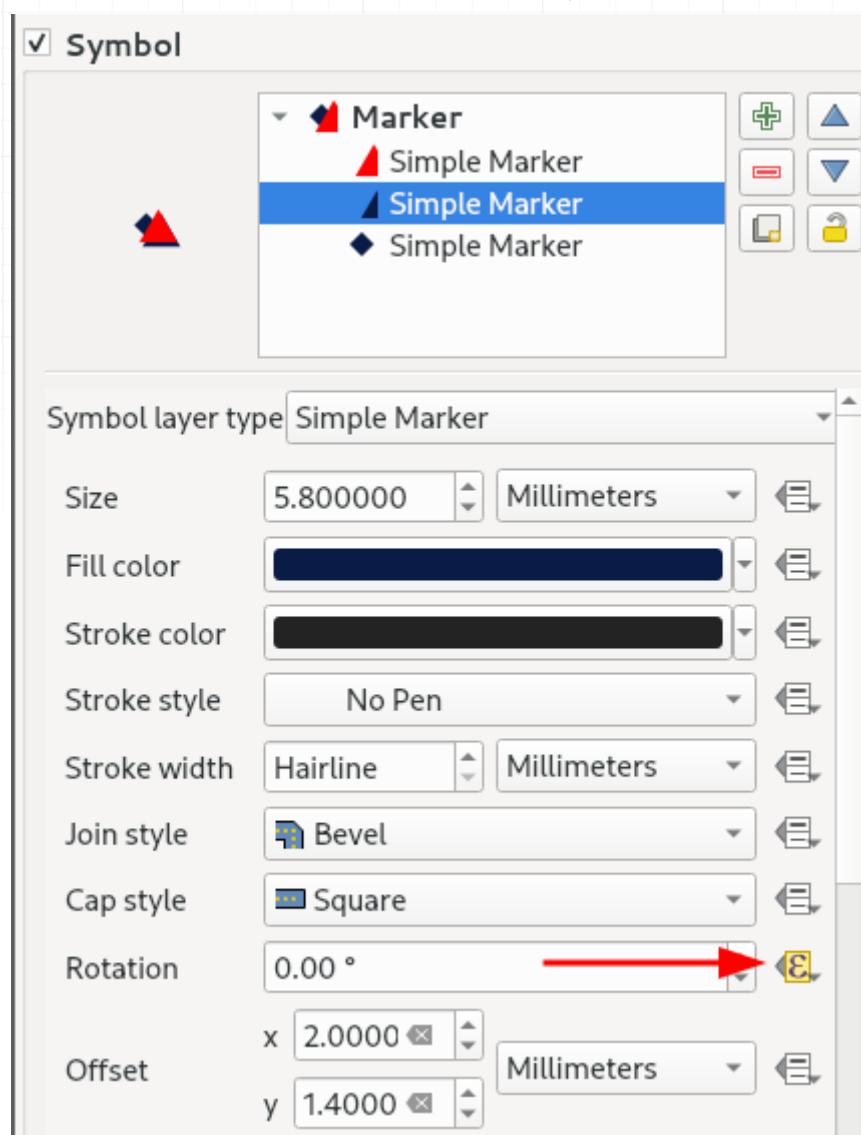
"Bring your QGIS projects to life!"





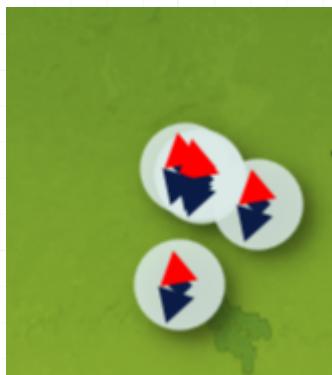
1.2.5 Rotation

You can set the angle of rotation for a symbol using this expression:



"Bring your QGIS projects to life!"





Using this technique you can also create an animation effect showing the source direction of travel and the new destination.

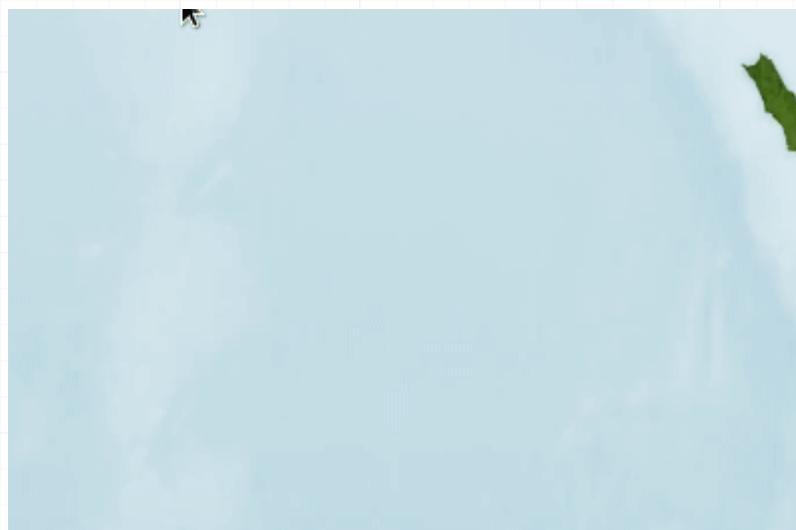
Code:

```
scale_linear (
    @current_hover_frame,
    0,
    @hover_frames,
    degrees(
        azimuth(
            geometry(@hover_feature),
            geometry(@previous_feature)
        )
    ),
    degrees(
        azimuth(
            geometry(@hover_feature),
            geometry(@next_feature)
        )
    )
)
```

Will produce something like this:

"Bring your QGIS projects to life!"





1.2.6 Flying points cluster

Here is an example where we animate all the points in a cluster that are **not** the hover point. We use an easing function to make the animation have an interesting circular motion.

"Bring your QGIS projects to life!"





QGIS
Animation
Workbench



"Bring your QGIS projects to life!"





Code:

```
-- Taken from https://spicyyoghurt.com/tools/easing-functions
-- t = Time - Amount of time that has passed since the beginnin
-- b = Beginning value - The starting point of the animation. U
-- c = Change in value - The amount of change needed to go from
-- d = Duration - Amount of time the animation will take. Usual
-- Sinusoidal
-- -c / 2 * (Math.cos(Math.PI * t / d) - 1) + b;

-- Use with the animation in static mode
if(@hover_feature_id != $id,
array(
    (-@hover_frames / 2) * (cos( (pi() * @frame_number / @hover_fram
    (-@hover_frames / 2) * (sin( (pi() * @frame_number / @hover_frame
),
array (0,0))
```

This function should be applied to the offset X,Y property of the symbol.

"Bring your QGIS projects to life!"





The screenshot shows the QGIS Animation Workbench interface. On the left, there is a map view with several layers visible, including a road network and place names like 'Córdoba', 'Marbella', and 'Fuengirola'. The main window displays the 'Marker' symbol layer configuration for a 'Simple Marker'. The 'Symbol layer type' is set to 'Simple Marker'. The 'Size' is 4.600000 Millimeters. The 'Fill color' is red, and the 'Stroke color' is black. The 'Stroke style' is 'No Pen', 'Stroke width' is 'Hairline' in Millimeters, and 'Join style' is 'Bevel'. The 'Cap style' is 'Square', and the 'Rotation' is 0.00 °. The 'Offset' is set to 0.000000 in both X and Y directions, with units in Millimeters. The 'Anchor point' is set to 'VCenter' vertically and 'HCenter' horizontally. A toolbar at the bottom includes icons for back, forward, save, processing, and debugging. There are also checkboxes for 'Live update' and 'Apply', and a button for 'Layer Rendering'.

Marker

Simple Marker

Symbol layer type Simple Marker

Size 4.600000 Millimeters

Fill color

Stroke color

Stroke style No Pen

Stroke width Hairline Millimeters

Join style Bevel

Cap style Square

Rotation 0.00 °

Offset x 0.000000 Millimeters y 0.000000

Anchor point VCenter HCenter

Layer Rendering

Live update Apply

B... Project... Save Processing... Debugging/Development...

"Bring your QGIS projects to life!"





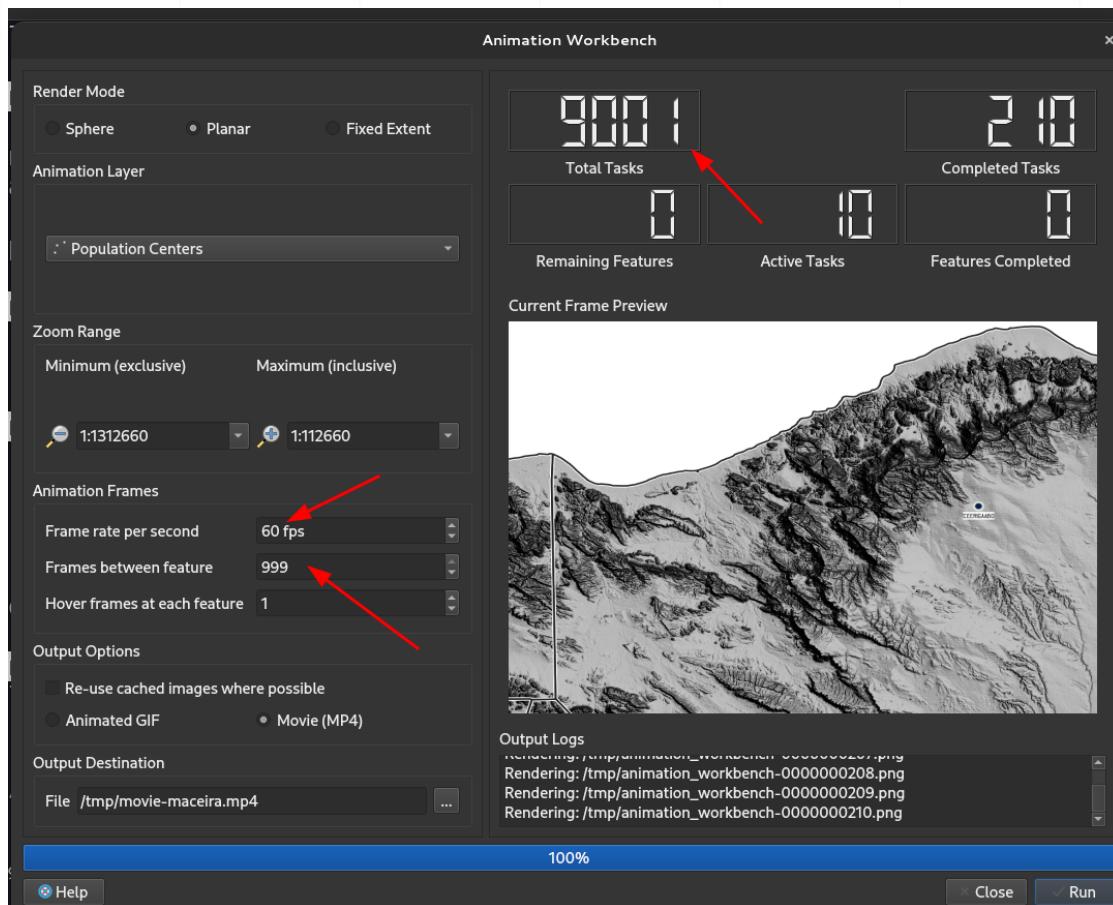
1 Frequently Asked Questions

Can I add any image to the intro or outro?

- As long as you can provide the proper attribution for an image you can use it in your project.

I have an older, less powerful, computer, will it handle running this workbench?

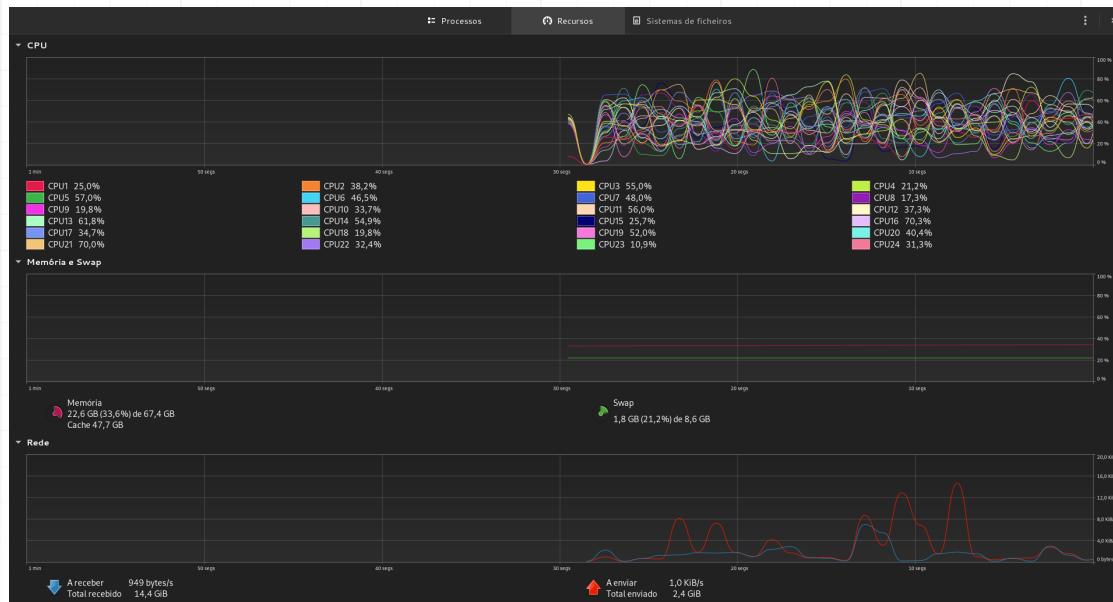
- If you open the standard QGIS settings dialog and select the Animation Workbench options you can follow the advice with regards to lowering the number of threads allowed during rendering to help your computer cope. Rendering shorter movies or GIFs (i.e. fewer frames) will also help. Below is an example of running a job with 9000 frames at 60fps and 999 frames per feature



And the subsequent CPU load during processing:

"Bring your QGIS projects to life!"





After processing:



And here is the resulting video:

<https://youtu.be/1qc3xPdJsU>

I get an error when rendering because of my intro / outro images

Currently your filenames should not contain spaces or special characters (e.g., , [], { }, <, >, /, \, :, *, ?, |, ", &, etc.).

Can I use a movie as the intro / outro media?

"Bring your QGIS projects to life!"





This is planned but not yet implemented. Tim - check.

Can I pay you to add some features?

This is a fun / hobby project, currently we want other contributors who also want to have a fun experience with building this plugin and contribute in-kind efforts to the project. Both [Kartoza](#) and [North-Road](#) offer commercial development services but not for this plugin which is intended to provide an experimental, no-pressure space for us to work on something fun for QGIS.

"Bring your QGIS projects to life!"





1 Develop

1.1 Developer Environment

In this section, we walk you through setting up a development environment and describe common workflows for debugging etc.

1.1.1 Setup

Fork the **main** branch into your personal repository. Clone it to your local computer. Install QGIS and the following dependencies.

- debugpy (python library)
- convert (imagemagick)
- ffmpeg
- vscode (don't use flatpak, debugging will not work with QGIS)

Clone the repo and symlink the **animation_workbench** subfolder into your profile. Remember to change **<profile>** in the line below with the actual name of the profile you will be using.

Code:

```
git clone https://github.com/{your-personal-repo}/QGISAnimationWorkbench -s animation_workbench ~.local/share/QGIS/QGIS3/profiles/<profile>
```

Enable the plugin in the QGIS plugin manager. You should also install the [Plugin Reloader](#) plugin so you can quickly deploy changes to your local session in QGIS as you are working.

1.1.2 Debugging

We use the VSCode remote debugger with **debugpy** in order to carry our debugging workflows such as setting breakpoints, inspecting the application state, stepping through code etc.

To start debugging, you need to put the plugin into developer mode.

Next, open the QGIS Animation Workbench git checkout (as described above), and then active the

Run and Debug tab (1 in the image below). From the list of launchers, choose

Python: Remote Attach and press the green run icon (2 in the image below).

The animation workbench will then resume normal operation, but you will be able to set breakpoints and inspect objects in CSCode. Please refer to VSCode documentation for the actual nuts and bolts of using their debugging tools.

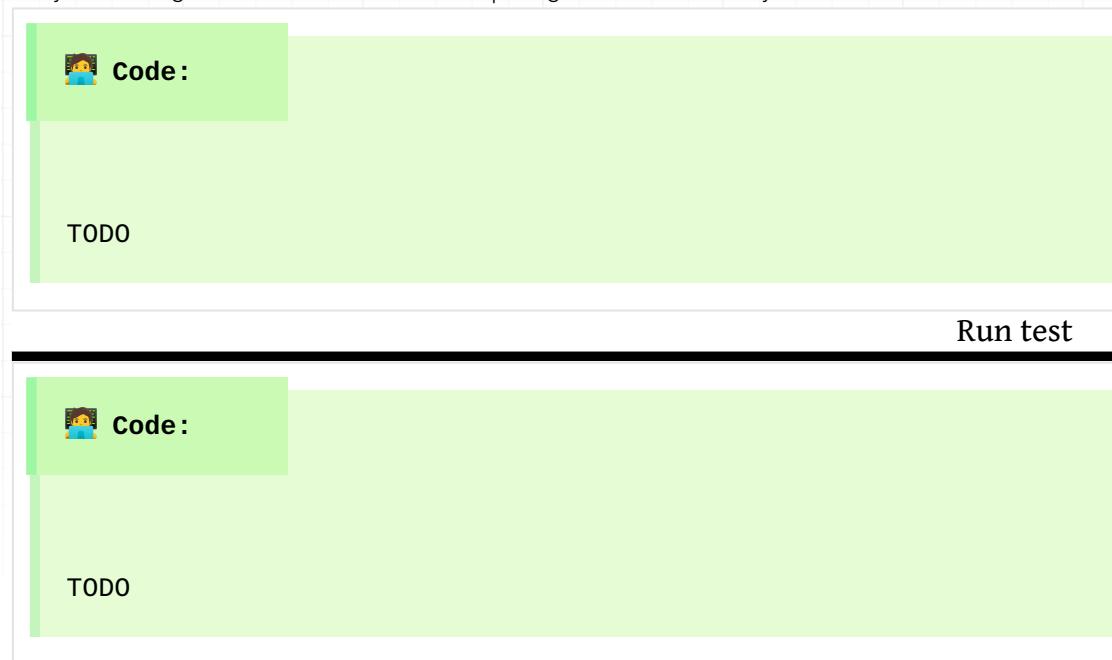
1.1.3 Packaging

"Bring your QGIS projects to life!"





Every time a merge is made to the main branch, a package is built automatically.



"Bring your QGIS projects to life!"





1.2 Design

"Bring your QGIS projects to life!"





1.3 Working with documentation

Documentation is written using [mkdocs](#).

1.3.1 Building documentation PDF

You can build a copy of the documentation as a PDF file using the following steps:

 **Code:**

```
pip install mkdocs-with-pdf
pip install mkdocs-material
pip install qrcode
mkdocs build --config-file mkdocs-pdf.yml
xdg-open pdfs/QGISAnimationWorkbench.pdf
```

"Bring your QGIS projects to life!"





1 Contribute

1.0.1 Pull Request Steps

This project is open source, so you can create a pull request(PR) after you fix issues. Get a local copy of the plugins checked out for development using the following process.

Pull Request

Before uploading your PR, run test one last time to check if there are any errors. If it has no errors, commit and then push it!

For more information on PR's steps, please see links in the Contributing section.

Commit messages

"Bring your QGIS projects to life!"





Please make this project more fun and easy to scan by using emoji prefixes for your commit messages (see [GitMojis](#)).

"Bring your QGIS projects to life!"





Commit type	Emoji
Initial commit	:tada:
Version tag	:bookmark:
New feature	:sparkles:
Bugfix	:bug:
Metadata	:card_index:
Documentation	:books:
Documenting source code	:bulb:
Performance	:racehorse:
Cosmetic	:lipstick:
Tests	:rotating_light:
Adding a test	:white_check_mark:
Make a test pass	:heavy_check_mark:
General update	:zap:
Improve format/structure	:art:
Refactor code	:hammer:
Removing code/files	:fire:
Continuous Integration	:green_heart:
Security	:lock:
Upgrading dependencies	:arrow_up:
Downgrading dependencies	:arrow_down:
Lint	:shirt:
Translation	:alien:
Text	:pencil:
Critical hotfix	:ambulance:
Deploying stuff	:rocket:
Fixing on MacOS	:apple:
Fixing on Linux	:penguin:
Fixing on Windows	:checkered_flag:
Work in progress	:construction:
Adding CI build system	:construction_worker:
Analytics or tracking code	:chart_with_upwards_trend:
Removing a dependency	:heavy_minus_sign:
Adding a dependency	:heavy_plus_sign:
Docker	:whale:
Configuration files	:wrench:

"Bring your QGIS projects to life!"





Commit type	Emoji
Package.json in JS	📦 :package:
Merging branches	🔀 :twisted_rightwards_arrows:
Bad code / need improv.	💩 :hankey:
Reverting changes	⏪ :rewind:
Breaking changes	💥 :boom:
Code review changes	👌 :ok_hand:
Accessibility	♿ :wheelchair:
Move/rename repository	🚚 :truck:
Other	Be creative

1.0.2 ⚬ Contributing

- [Code of Conduct](#)
- [Contributing Guideline](#)
- [Commit Convention](#)
- [Issue Guidelines](#)

"Bring your QGIS projects to life!"





1 Credits

1.0.1 Author

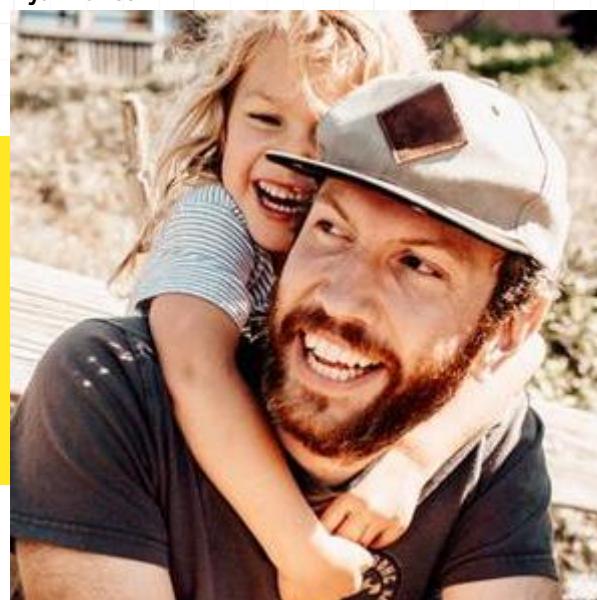
This plugin was developed by:

Tim Sutton



Coder and Ideas Guy
[@github](https://github.com/timlinux)

Nyall Dawson



Genius Guru of Awesomeness
[@github](https://github.com/nyalldawson)

Jeremy Prior



Document and Logo Guy
[@github](https://github.com/Jeremy-Prior)

1.0.2 Contributors

Thanks to:

- Mathieu Pellerin (@nirvn)
- Thiasha Vythilingam (@ThiashaV)

We are looking for contributors, add yourself here!

Also:

- [NHN and Tui Editor](#) for the great README which I based this one on.

"Bring your QGIS projects to life!"





<https://github.com/timlinux/QGISAnimationWorkbench>