QGIS Animation Workbench

The QGIS Animation Workbench

Bring your QGIS maps to life!

Tim Sutton, Nyall Dawson

Table of contents

1. Welcome to the Animation Workbench

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

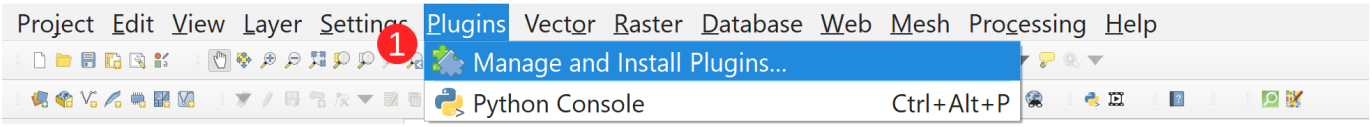
- Modes : Supports a 3 modes: Sphere, Planar and Static.
- Sphere: Creates a spinning globe effect. Like Google Earth might do, but with your own data and cartography.
- Planar: Lets you move from feature to feature on a flat map, pausing at each if you want to.
- Static: The frame of reference stays the same and you can animate the symbology within that scene.
- Internationalization (i18n) : Supports English currently - we may add other languages in the future if there is demand.
- 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).
- 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.

2. Quickstart

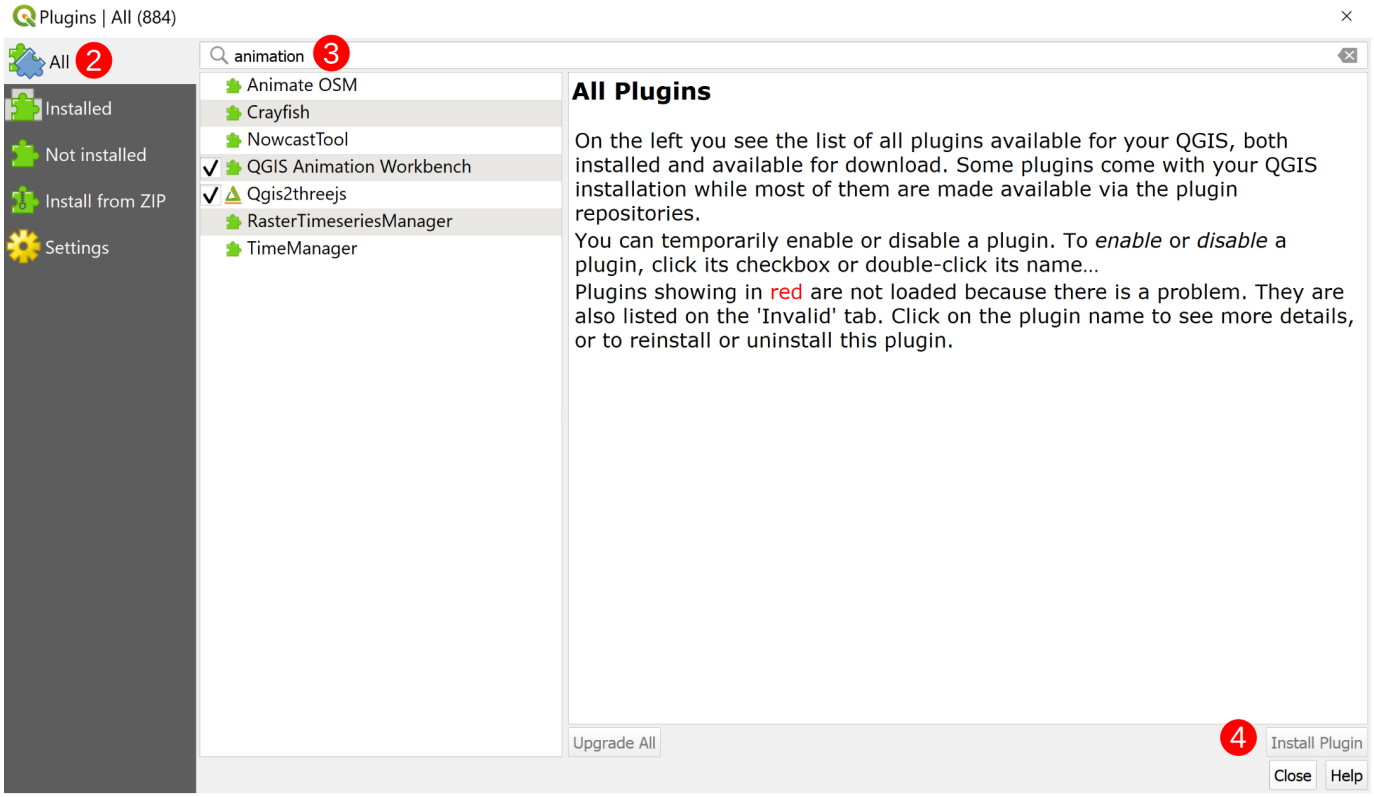
2.1 Installing the QGIS Animation Workbench plugin

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.

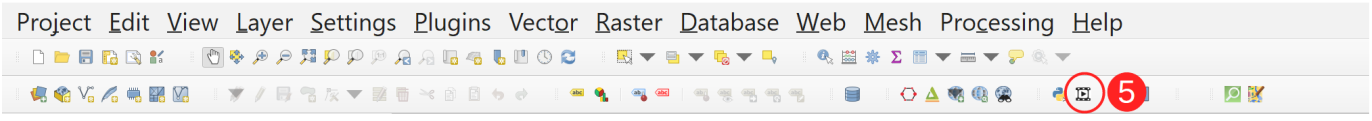
To access the QGIS Plugin Manager you simply need to select `1.` `Plugins` -> `Manage and Install Plugins...` in the Menu Toolbar.



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



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



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

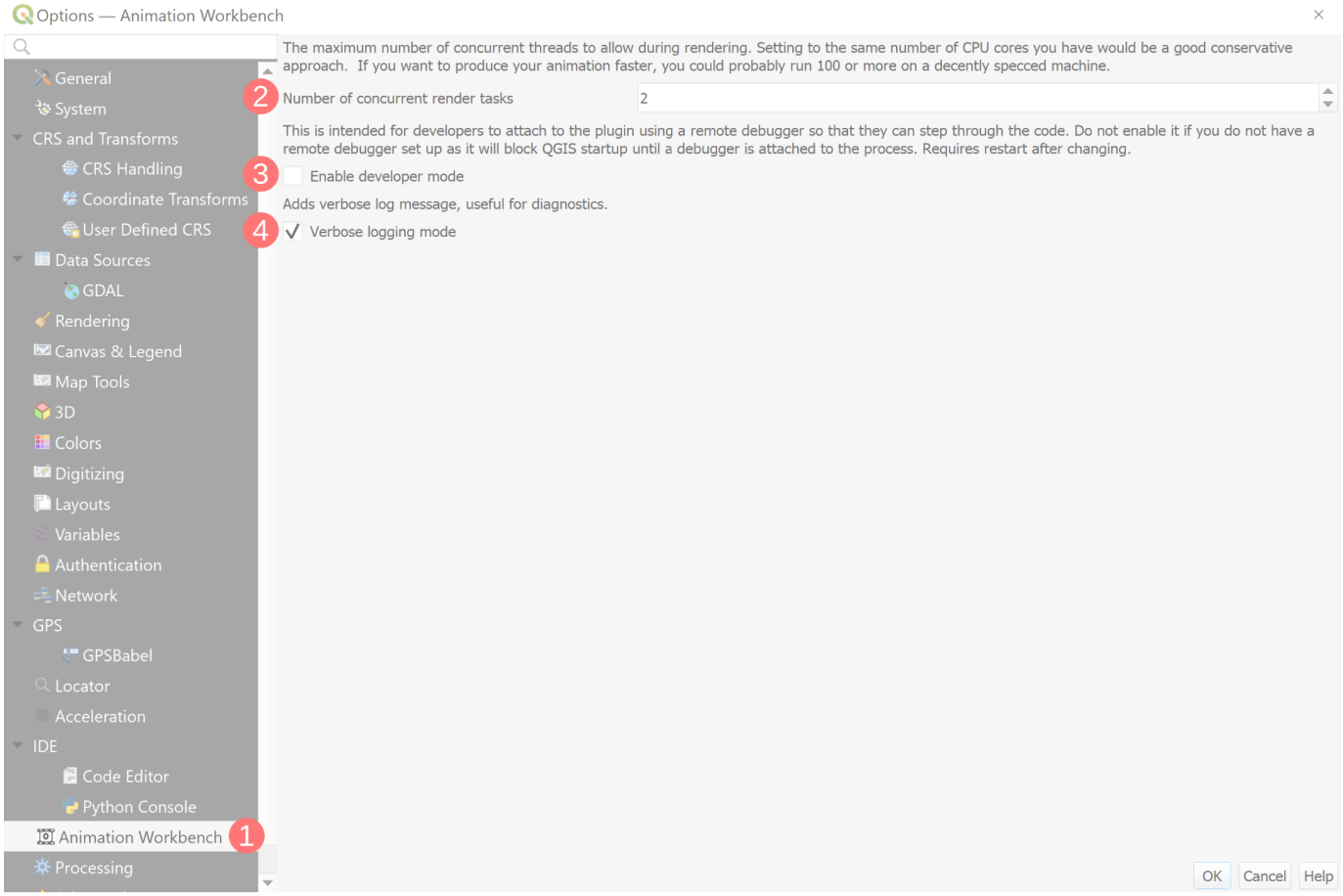
```
sudo apt install PyQt5.QtMultimedia
```

2.2 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.

Settings ➔ Options



- 1 Animation Workbench plugin Options

Currently there are just three configuration options:

- 2 Number of concurrent render tasks: This is the number of concurrent tasks that will be used to render animations. The default is 10.
- 3 Enable developer mode: This is a developer option that enables the developers to see an icon in the toolbar which will start the debug remote server.
- 4 Verbose logging mode: This will add extra messages in the logging pane to help you understand what is going on during the rendering process.

2.3 Using the Animation Workbench

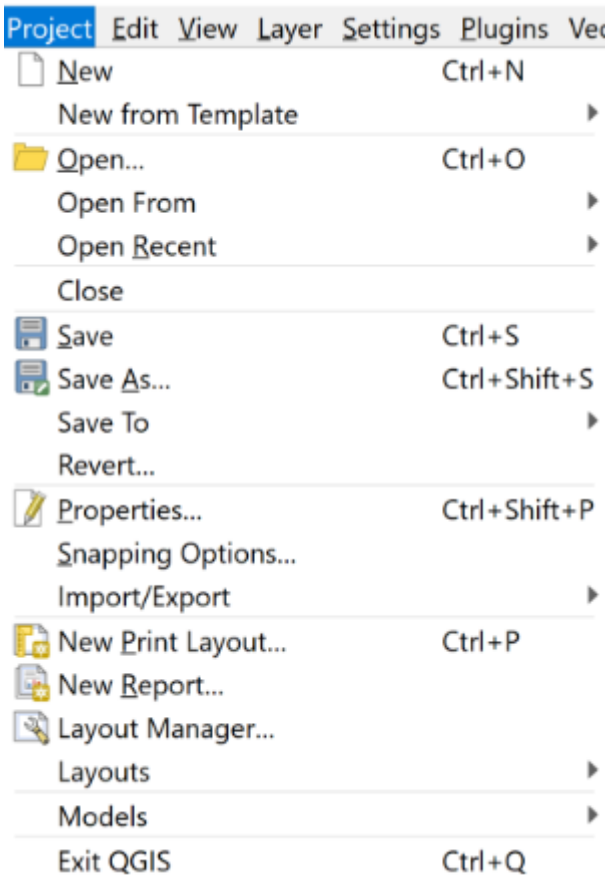
In this section, we describe the general workflow for using the Animation Workbench.

2.3.1 Process Overview

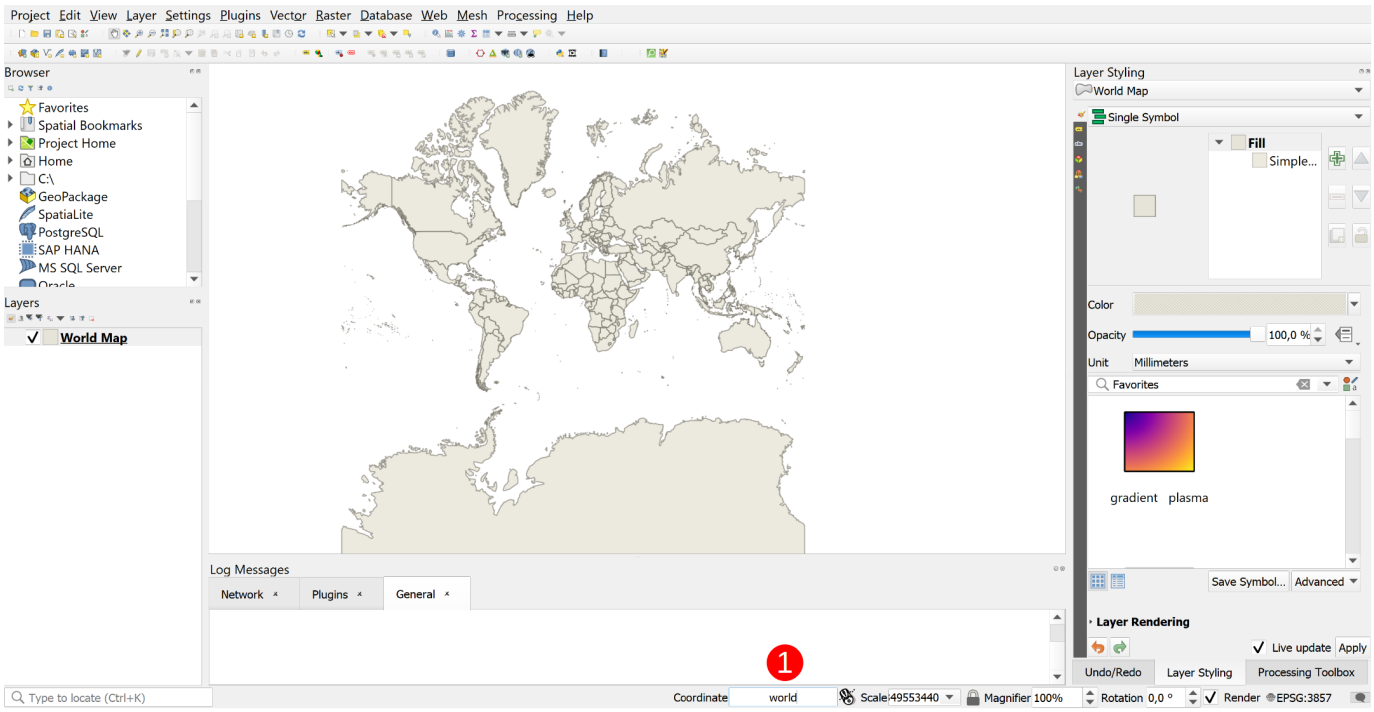
1. Create a QGIS project!
2. Identify features that will be animated.
3. Use the QGIS Expressions system with the variables introduced by the Animation Workbench to define behaviours of your symbols during flight and hover modes of your animation.
4. Open the Animation Workbench and configure your animation,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

2.3.2 More in Depth Process

1. Create a QGIS Project Open QGIS and click on `Project` ➔ `New`

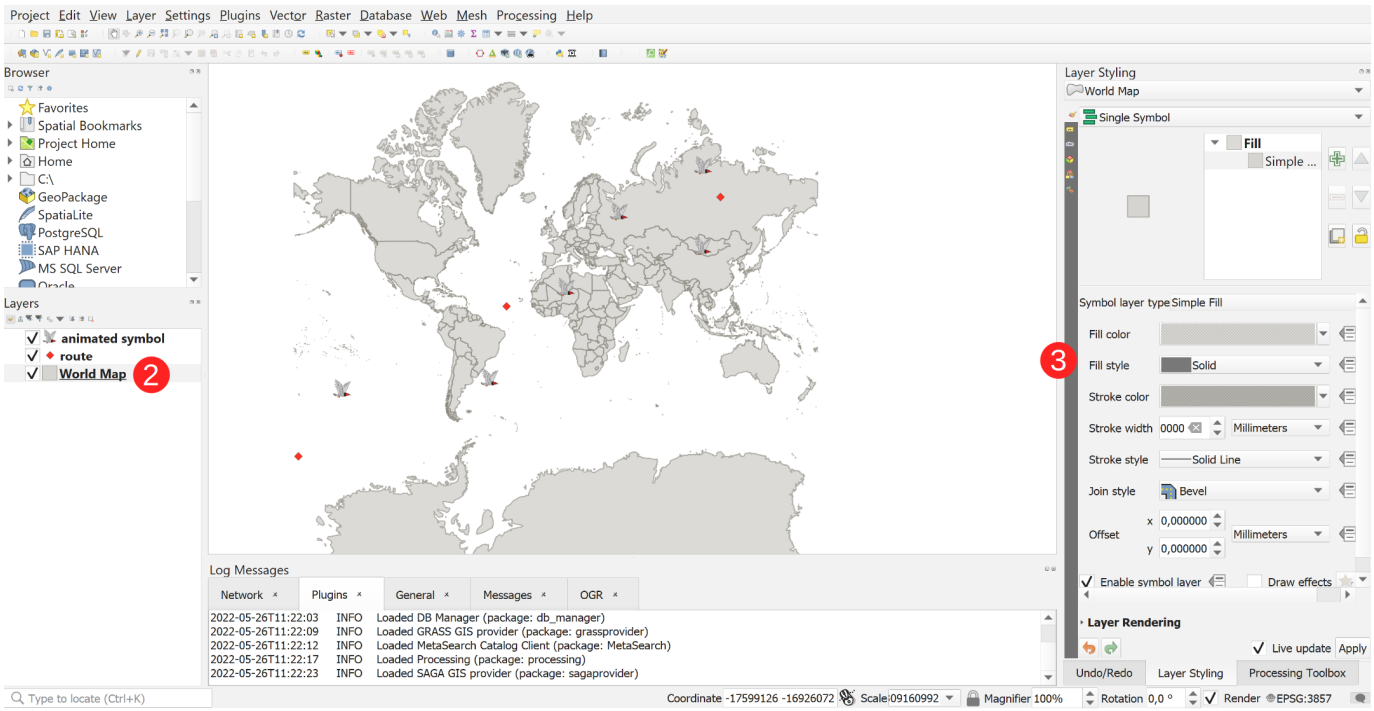


Add new layers to your project



Note: A simple way to add a base layer is to type `1` "world" into the coordinate textbox

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



1. Identify features that will be animated.

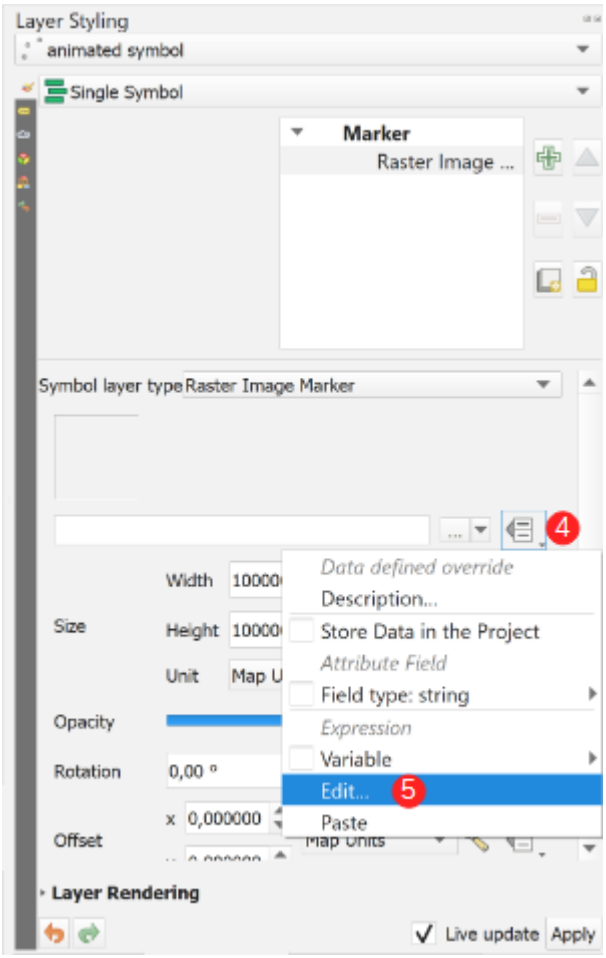
   Pick the layer (or layers) that you want to animate. Then either find or create the animation for the layer. Make sure you have all the correct attribution for any animations you use. Below is an example of an animation split into its frames.

1. 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` 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 `4` QGIS Expressions dropdown menu and click on `5 Edit`.

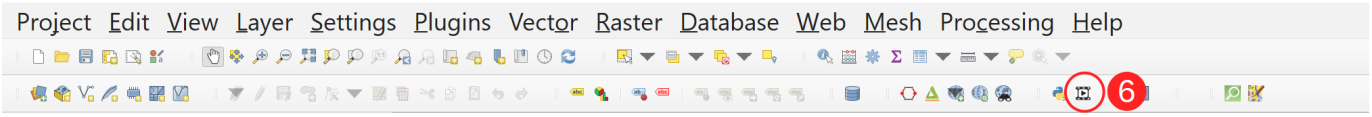


   Use the Code Snippets Section for more in depth help. The example below works with the bird animation from earlier

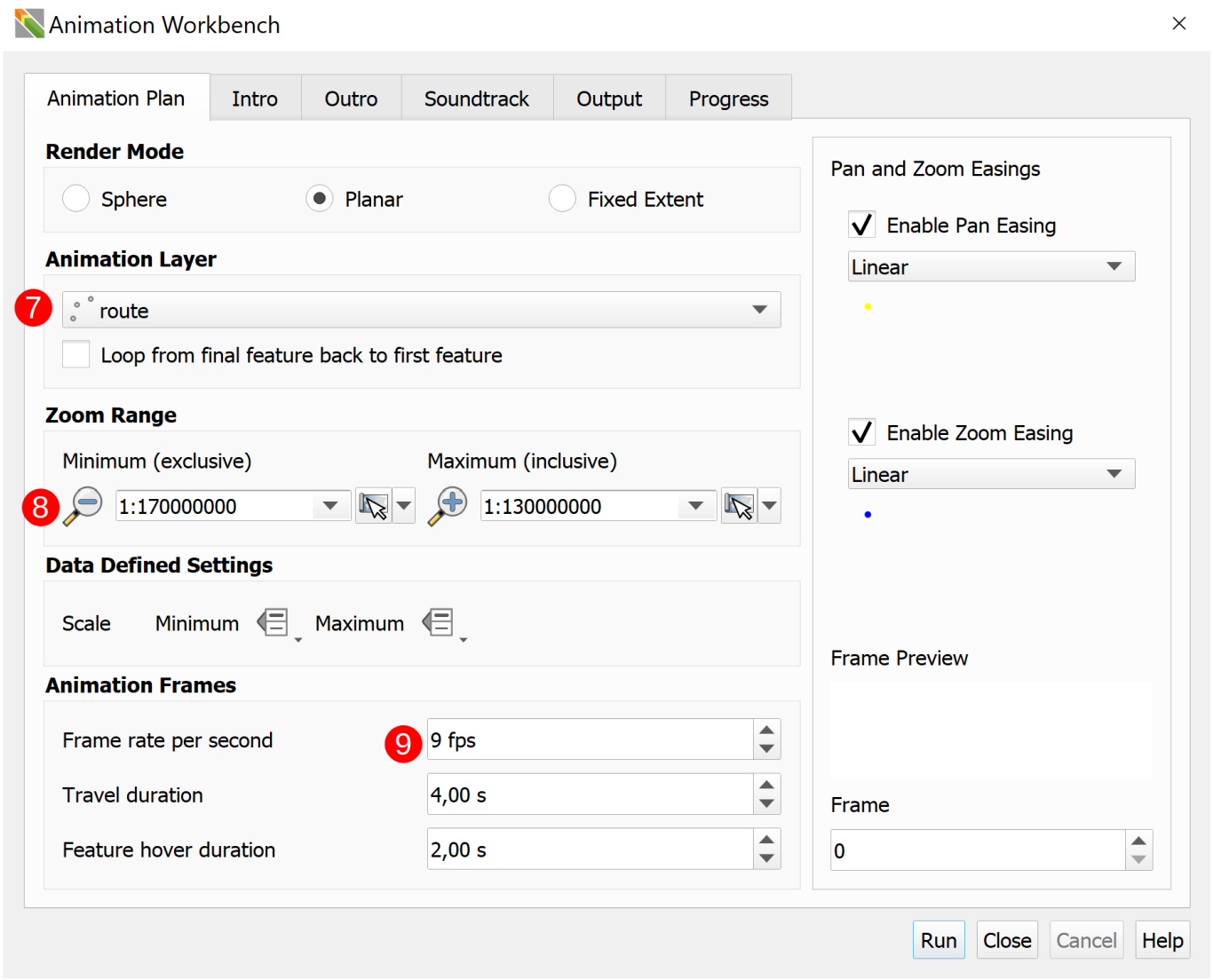
```
@project_home
||
'/bird/bird_00'
 ||
 lpad(to_string(@frame_number % 9), 2, '0')
  ||
  '.png'
```

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

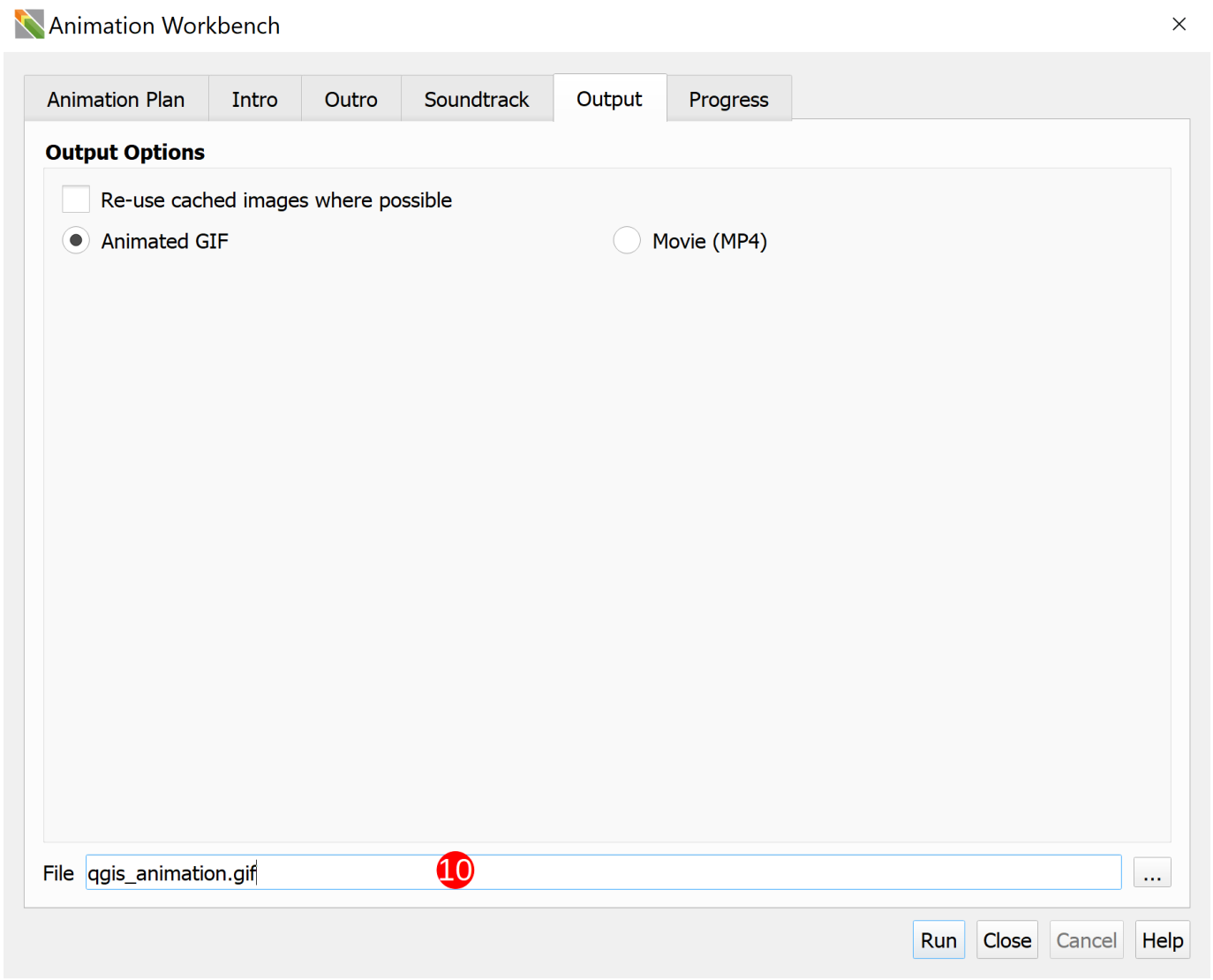
   Open the Workbench by clicking the `6` `Animation Workbench` icon in the Plugin Toolbar.



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



10 Select a location for your output.



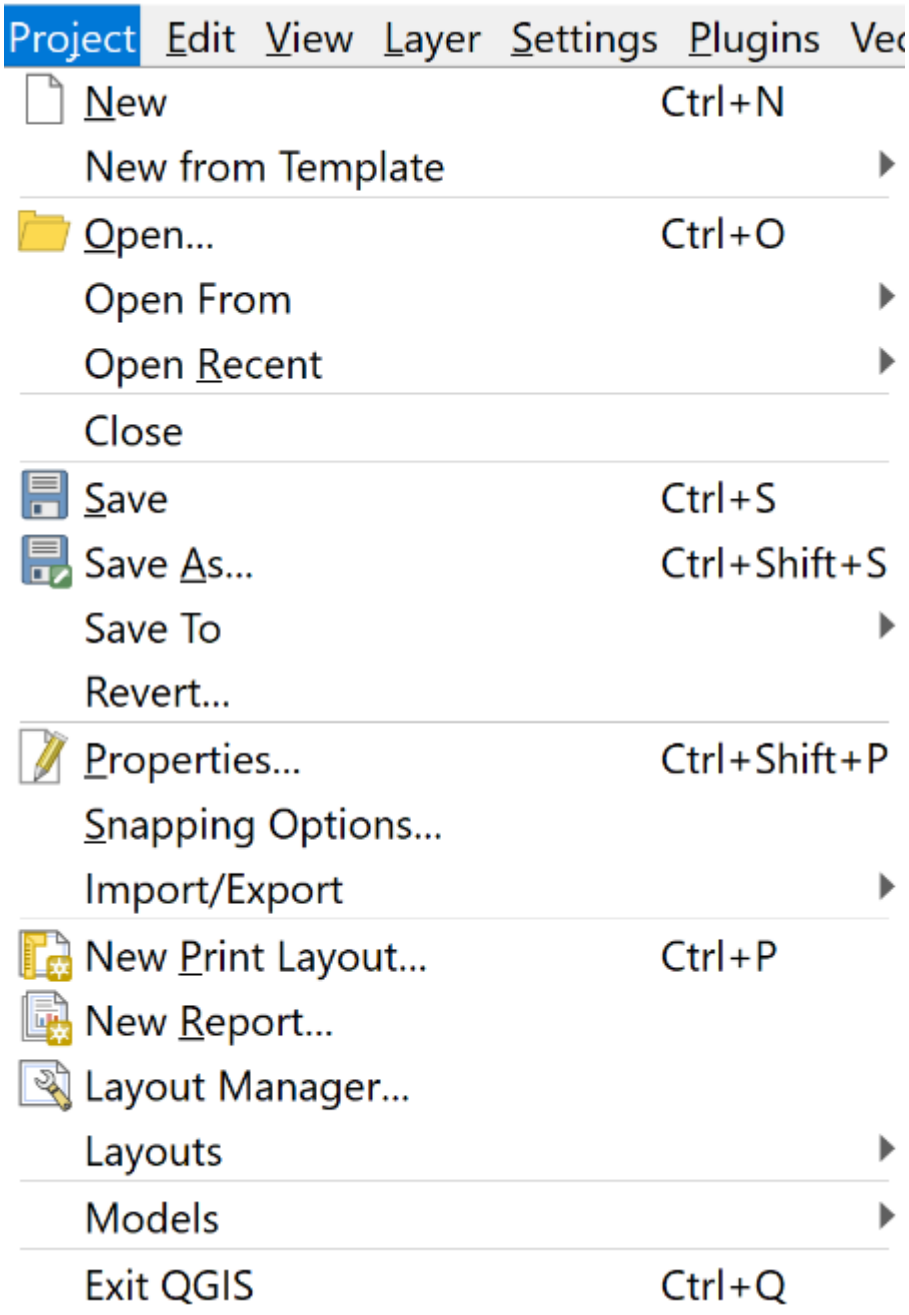
Note: Refer to the Workbench User Interface Section for more information about what various settings and buttons accomplish.

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

3. Manual

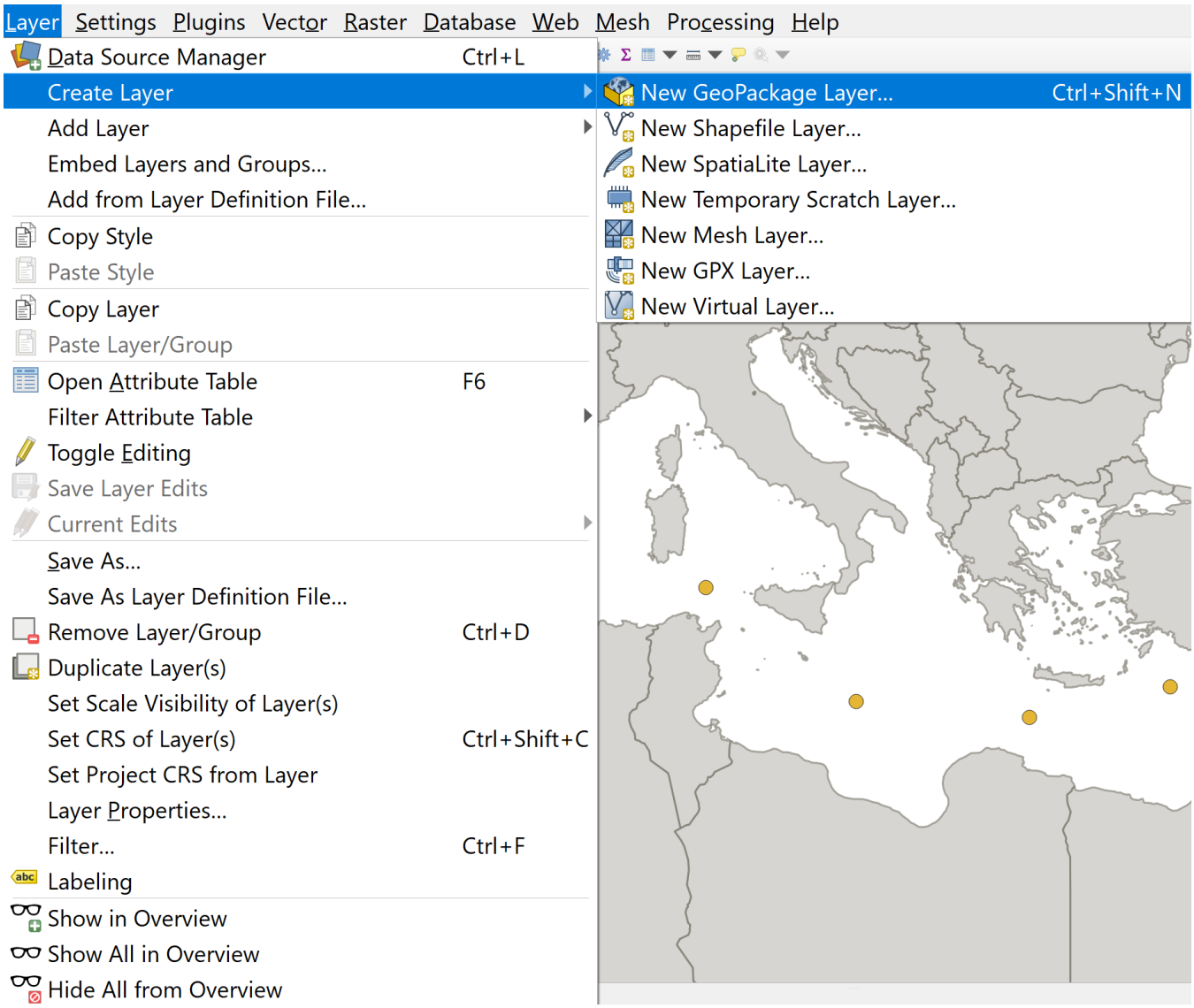
3.1 How to set up a project to work with the animation plugin

- The first step for getting an output using the Workbench is to create a QGIS Project Open QGIS and click on `Project` ➔ `New`

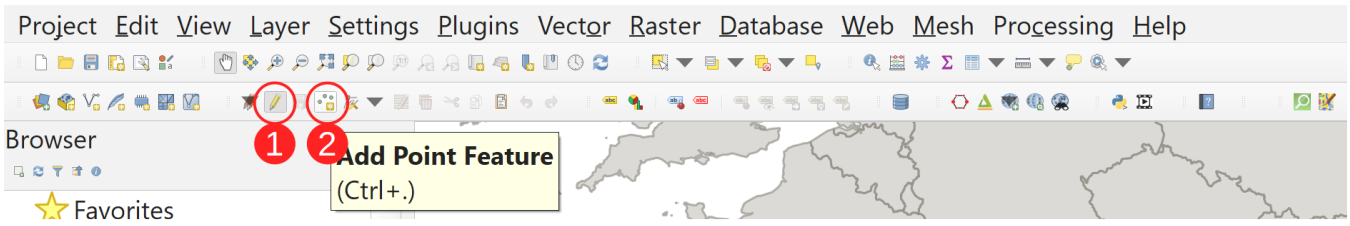


Next, add new layers to your project. You will want a few layers; one, or more, backing layer(s) (vector layers or XYZ Tiles), a layer for the workbench to follow, and one, or more, layer(s) of animated points. The example in this section only has one animated layer.

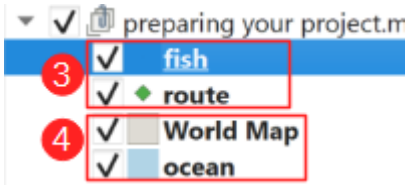
To add a layer, go to `Layer` -> `Create Layer` and then select the type of layer you want to add. The example adds a point layer to a geopackage to make the project more portable.



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

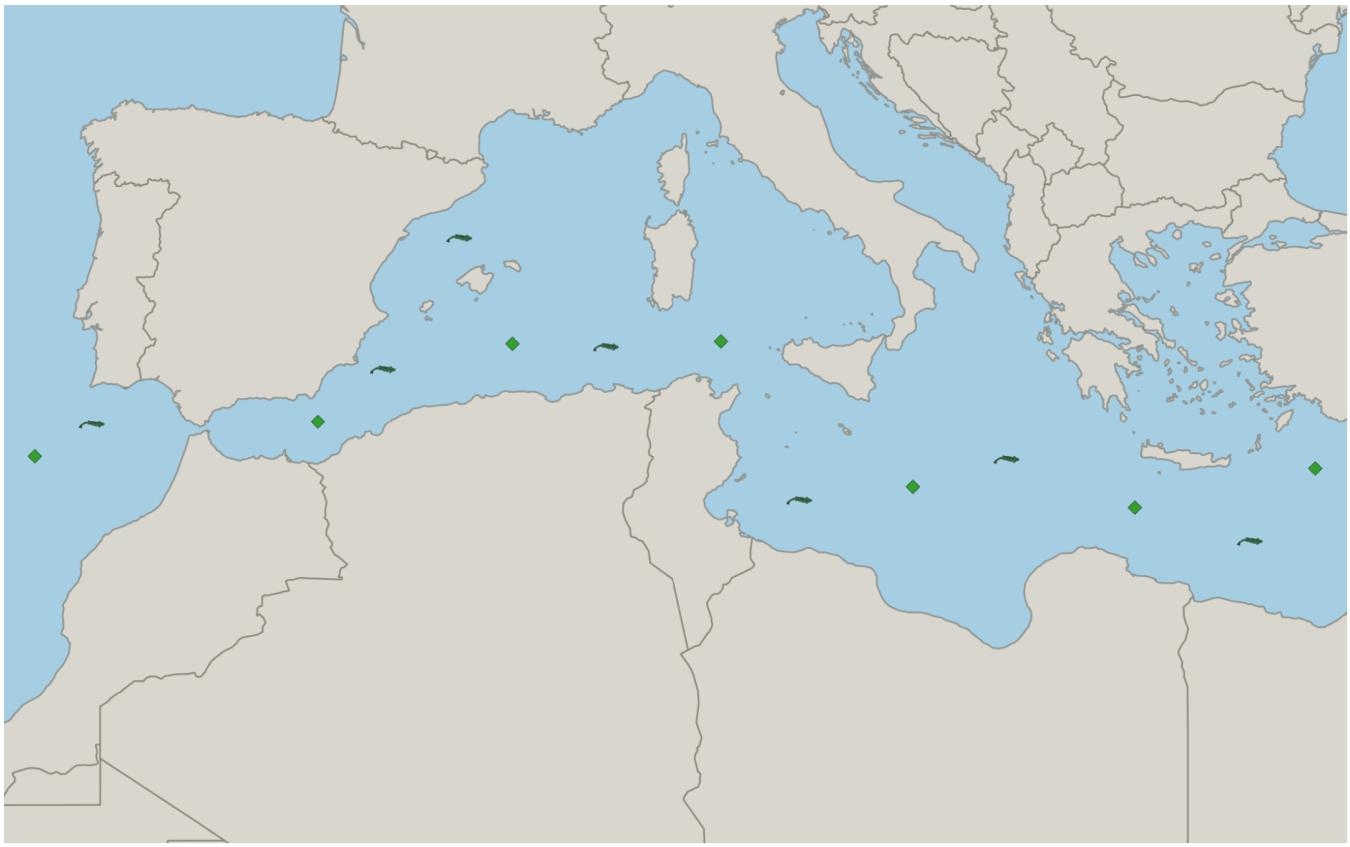


The example project has four layers: `3` two point layers and `4` two backing layers.



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

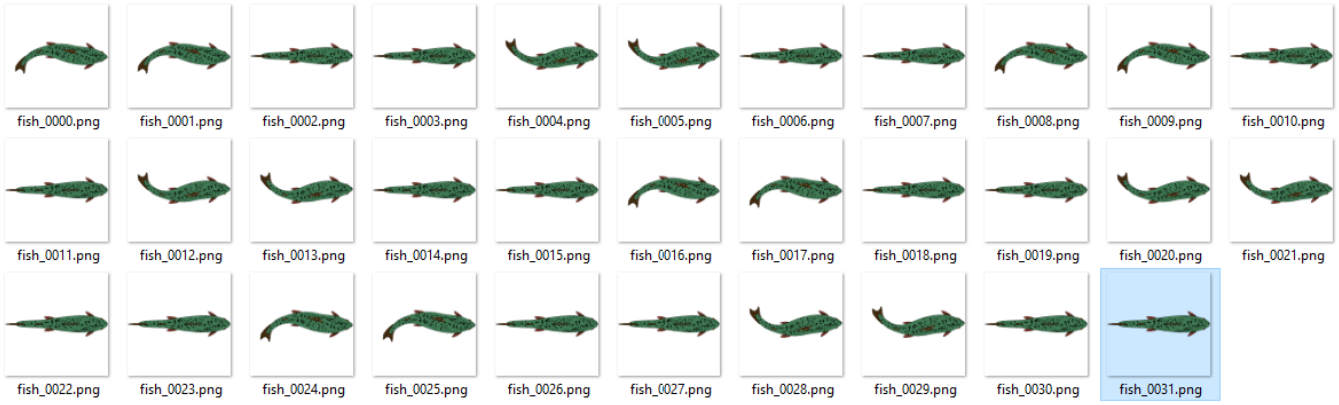
Finally, style your layers to make your project look aesthetically pleasing. To style your layers you must select the layer you want to style and then using the Layer Styling toolbar, play around with the style of the layer until it suits you. A good practice is to have your backing layers as more muted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



You now have a QGIS Project.

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

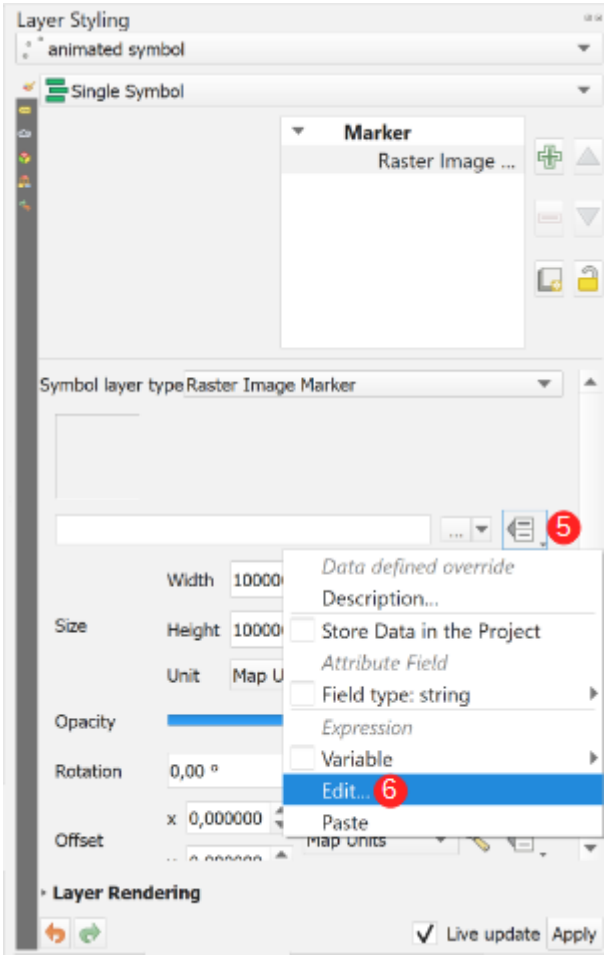
Pick the layer (or layers) that you want to have animations. Then either find, or create, the animation for the layer. Make sure you have all the correct attribution for any animations you use. Below is an example of a simple fish animation split into its frames. The frames are repeated to slow down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


fish_0000.png
fish_0001.png
fish_0002.png
fish_0003.png
fish_0004.png
fish_0005.png
fish_0006.png
fish_0007.png
fish_0008.png
fish_0009.png
fish_0010.png
fish_0011.png
fish_0012.png
fish_0013.png
fish_0014.png
fish_0015.png
fish_0016.png
fish_0017.png
fish_0018.png
fish_0019.png
fish_0020.png
fish_0021.png
fish_0022.png
fish_0023.png
fish_0024.png
fish_0025.png
fish_0026.png
fish_0027.png
fish_0028.png
fish_0029.png
fish_0030.png
fish_0031.png

- 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 `5` QGIS Expressions dropdown menu and click on `6 Edit`.



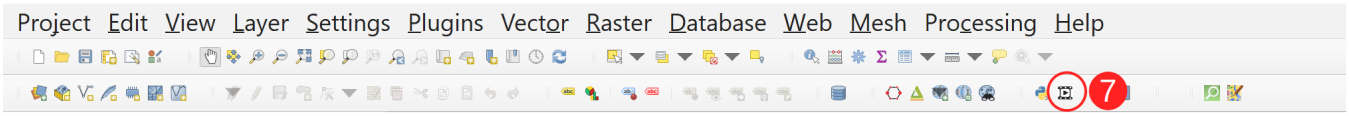
You can also make a marker move along a line relative to the frame of the animation. Use the Code Snippets Section for more in-depth help.

The example below works with the animation from earlier.

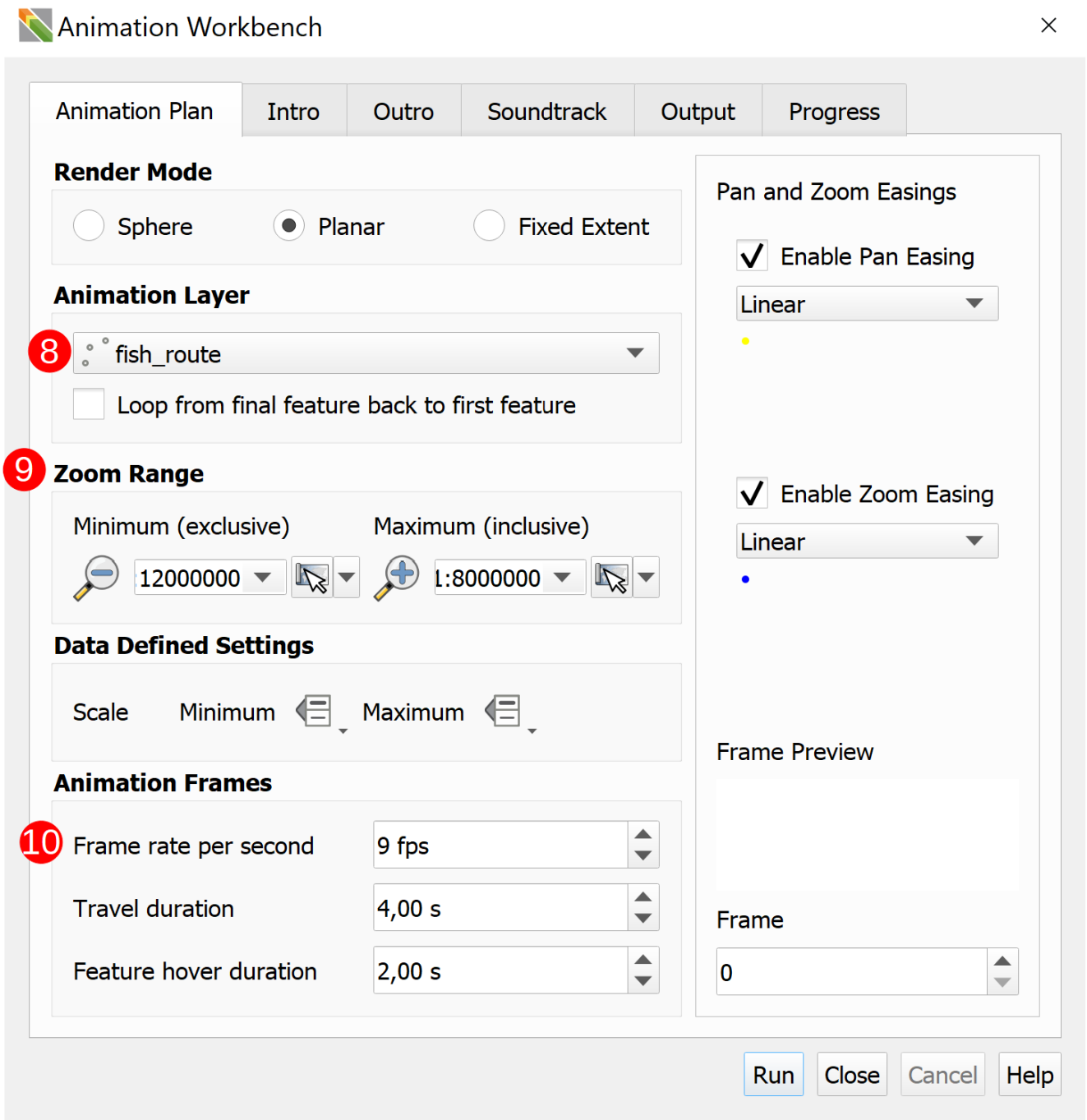
```
@project_home
 ||
'/fish/fish_00'
  ||
  lpad(to_string( @frame_number % 32), 2, '0')
   ||
   '.png'
```

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

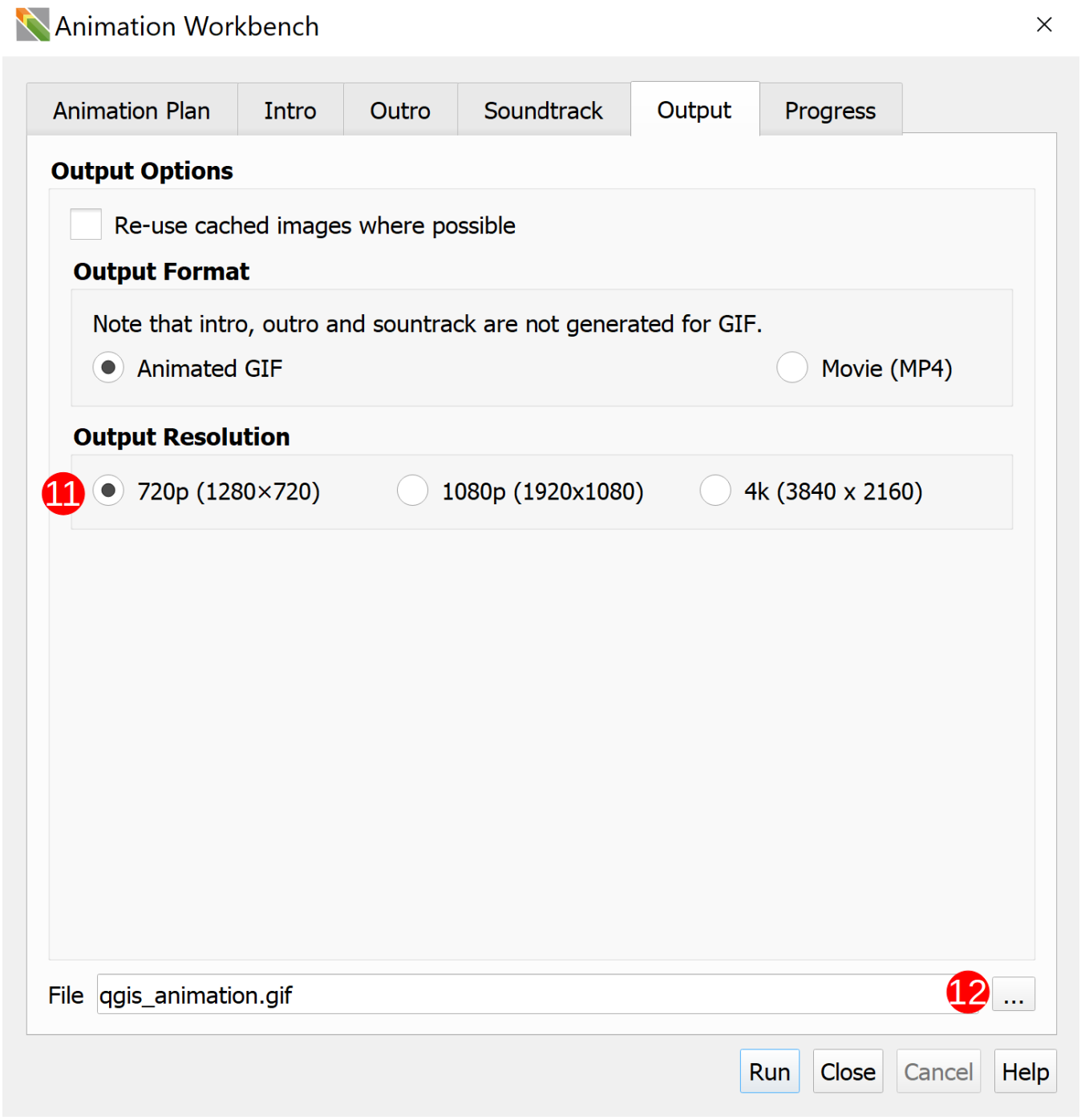
  Open the Workbench by clicking the `7` `Animation Workbench` icon in the Plugin Toolbar.



  Configure the settings for your animation. The screenshot below is configured for the example presented in this section. The Animation Layer is selected as `8` `route` because that is the path that the output animation will fly along. The `9` Zoom Range was selected from the Map Canvas Extent, and the `10` Frame rate per second (fps) 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.

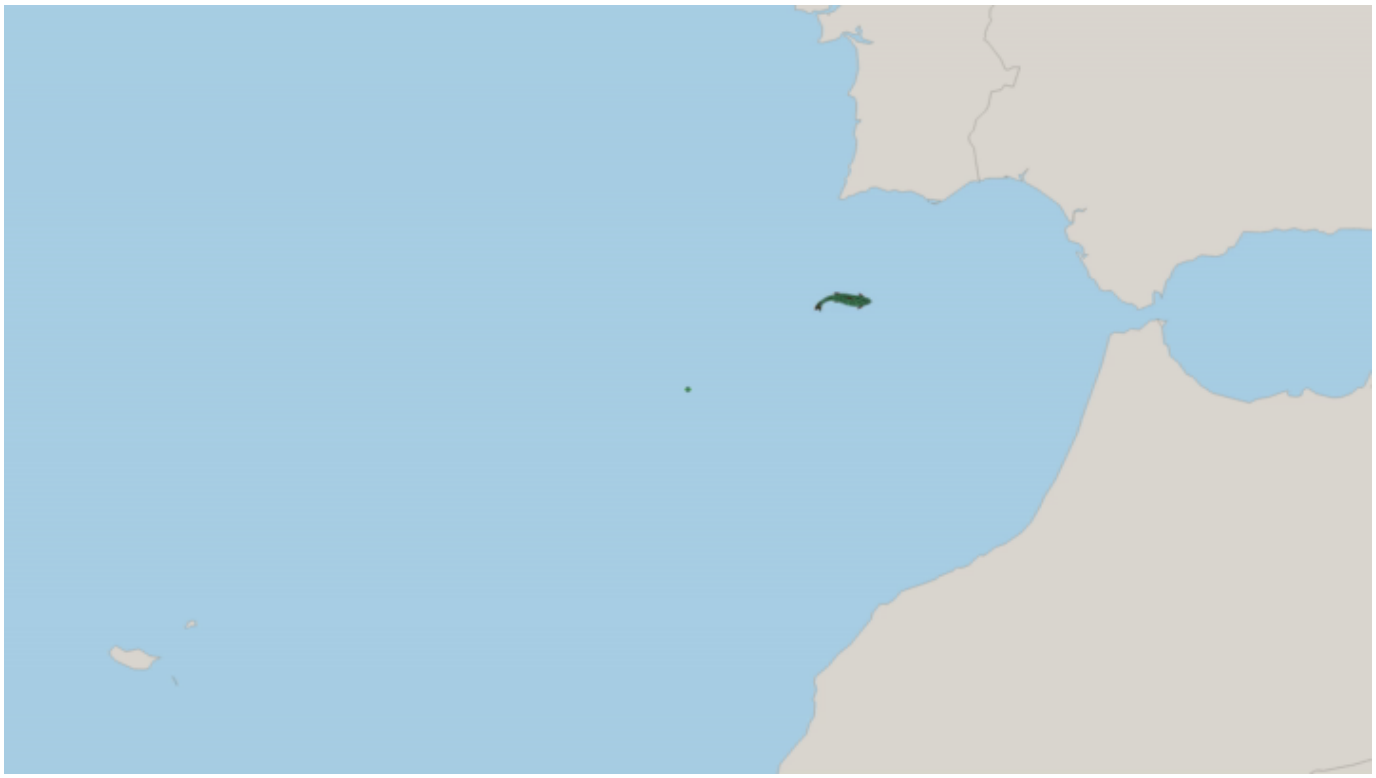


Select the 11 Output Resolution and a location for your output by clicking on the 12 ellipsis (three dots) or by typing in the desired file path.



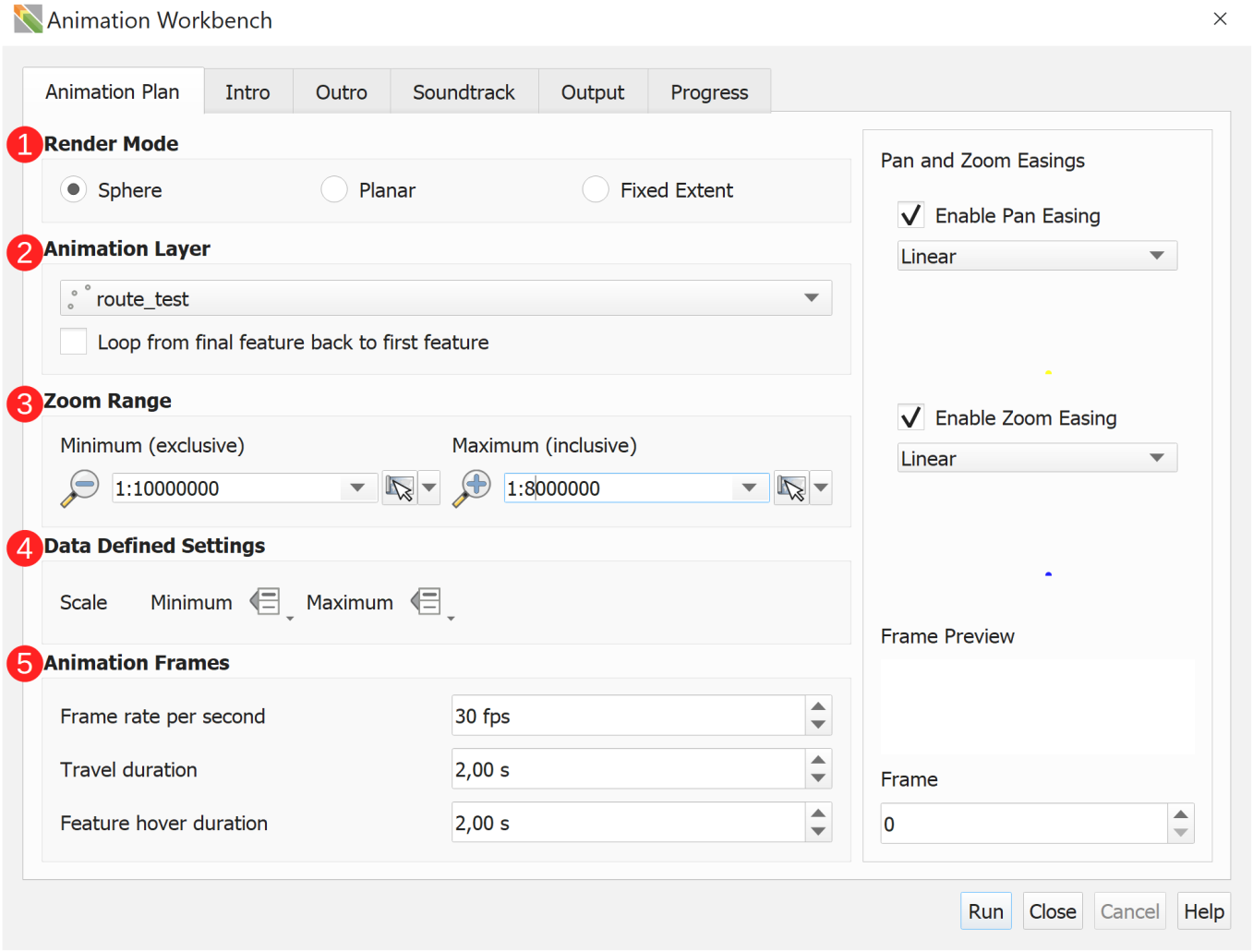
Note: Refer to the Workbench User Interface section for more information about what various settings and buttons accomplish.

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

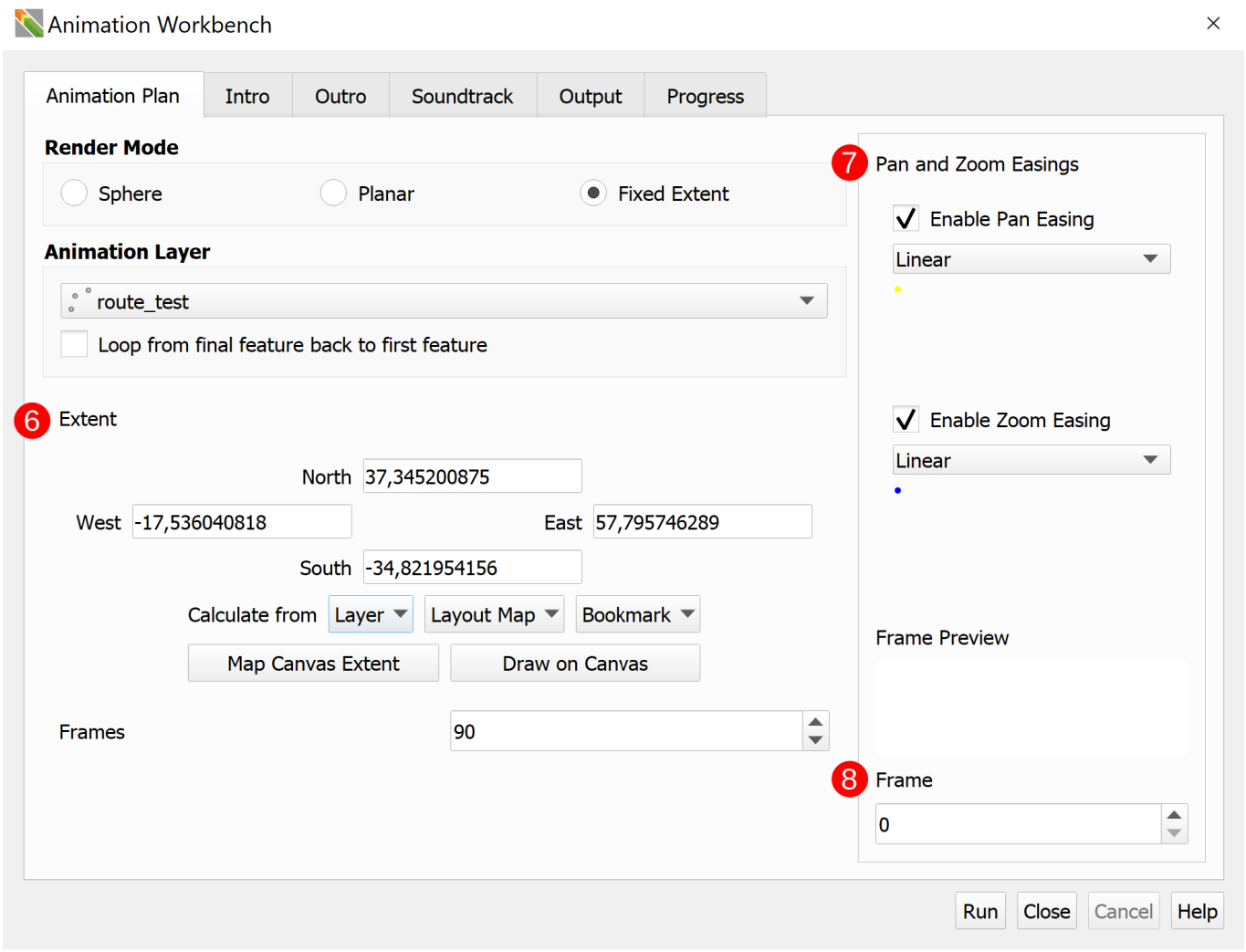


3.2 The Workbench User Interface

3.2.1 Animation Plan



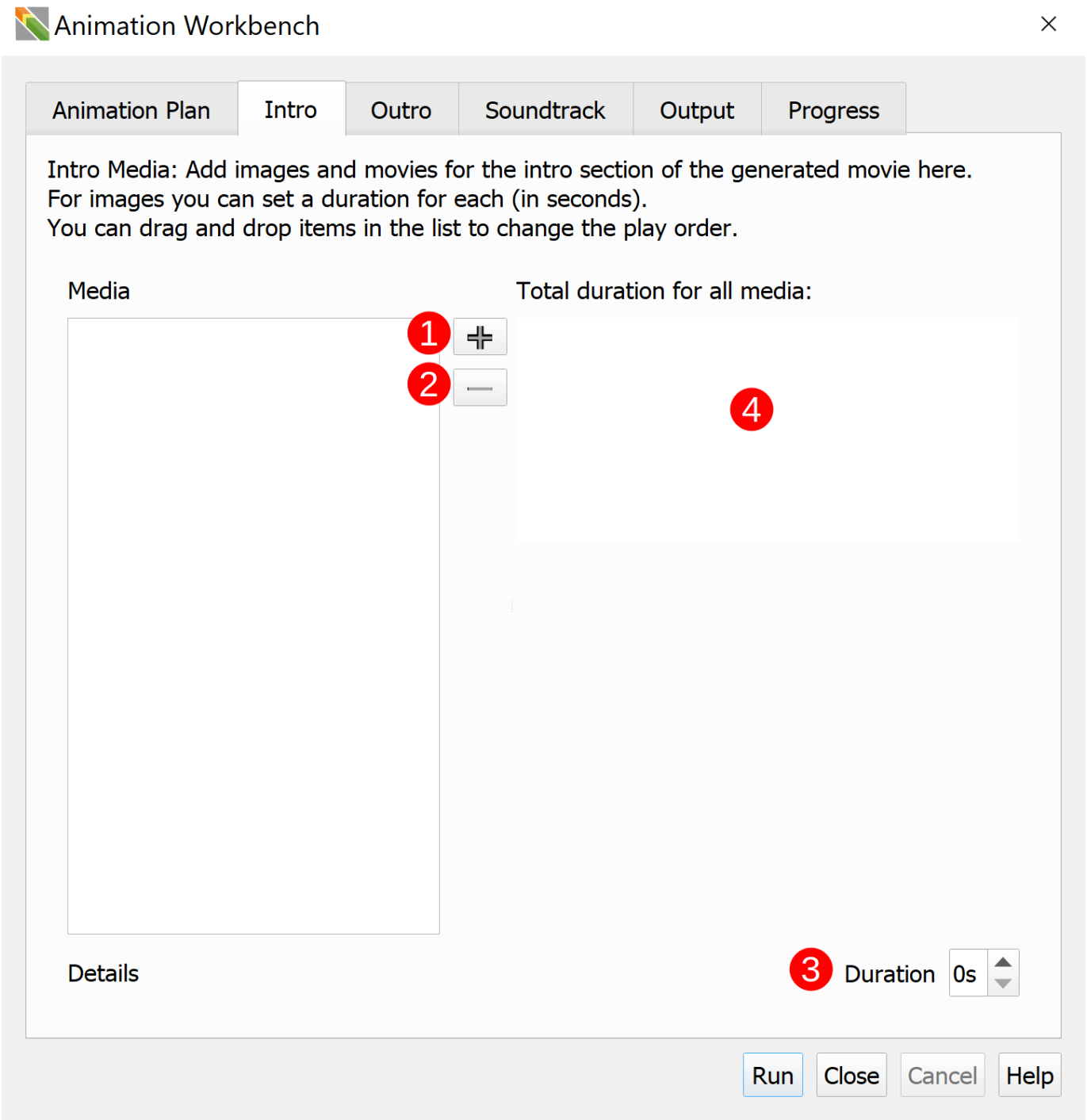
- `1` Render Modes: These determine the behaviour and type of animation
- `Sphere` : The coordinate reference system (CRS) will be manipulated to create a spinning globe effect. Like Google Earth might do, but with your own data and cartography.
- `Planar` : The coordinate reference system (CRS) will 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.
- `2` Animation Layer:
- `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).
- `3` Zoom Range: 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".
- `4` Data defined settings
- Scale
- Minimum: User-defined minimum scale
- Maximum: User-defined maximum scale
- `5` Animation Frames
- 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.



- `6` Extent:
- 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.
- `7` Pan and Zoom Easings
- 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)
- `8` Frame previews: A preview of what each frame of the animation will look like. A user can decide which `Frame` to view.

3.2.2 Intro Tab

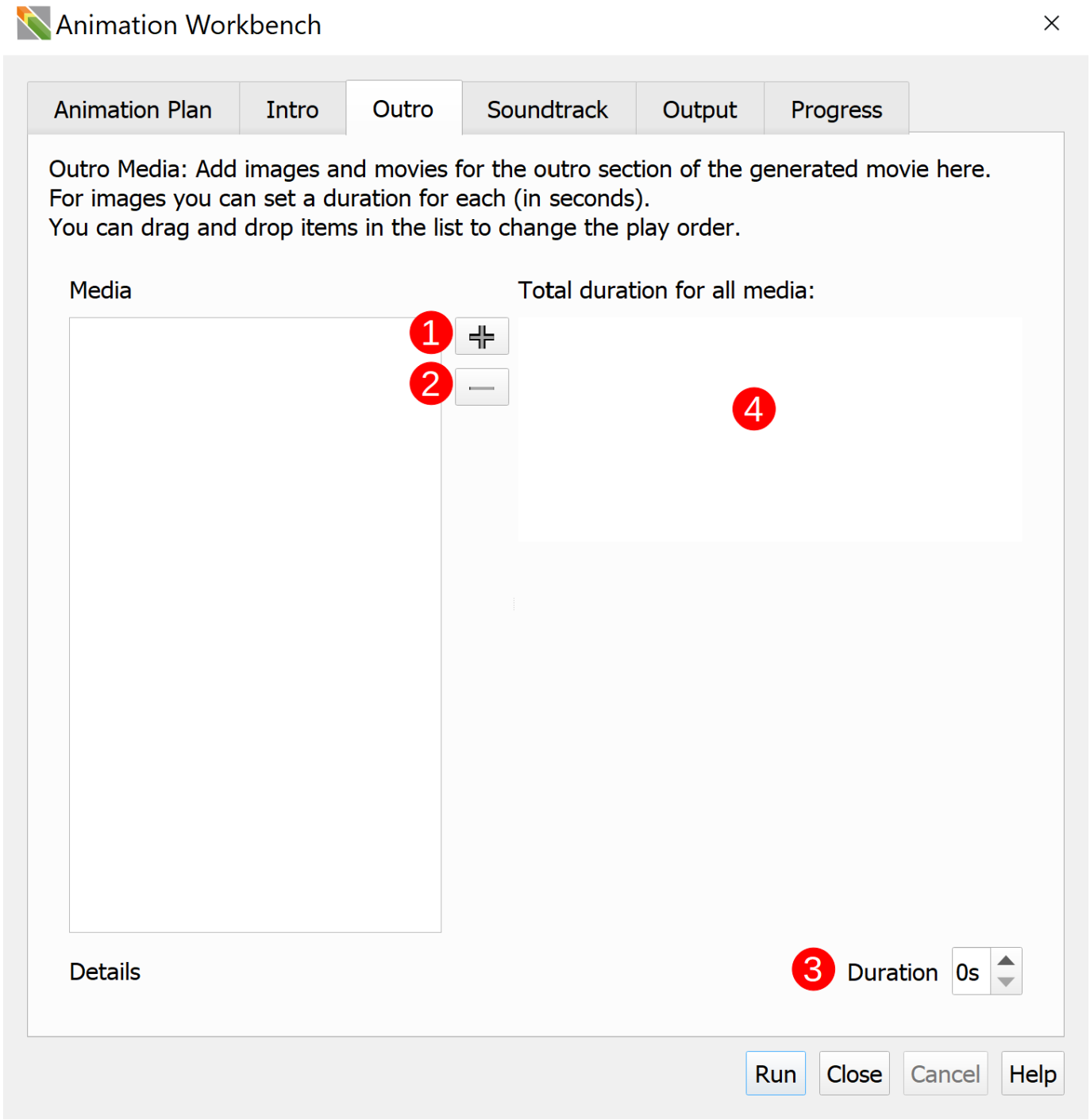
Edit the intro section of the generated movie here.



- 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.
- `1` Add Media (Plus sign): Add images or movies
- `2` Remove Media (Minus sign): Remove images or movies
- `3` Duration: For images, you can set a duration for each image (in seconds).
- `4` Preview Frame: This shows what the media will look like.
- Details: Provides details about where the media is stored on your computer.

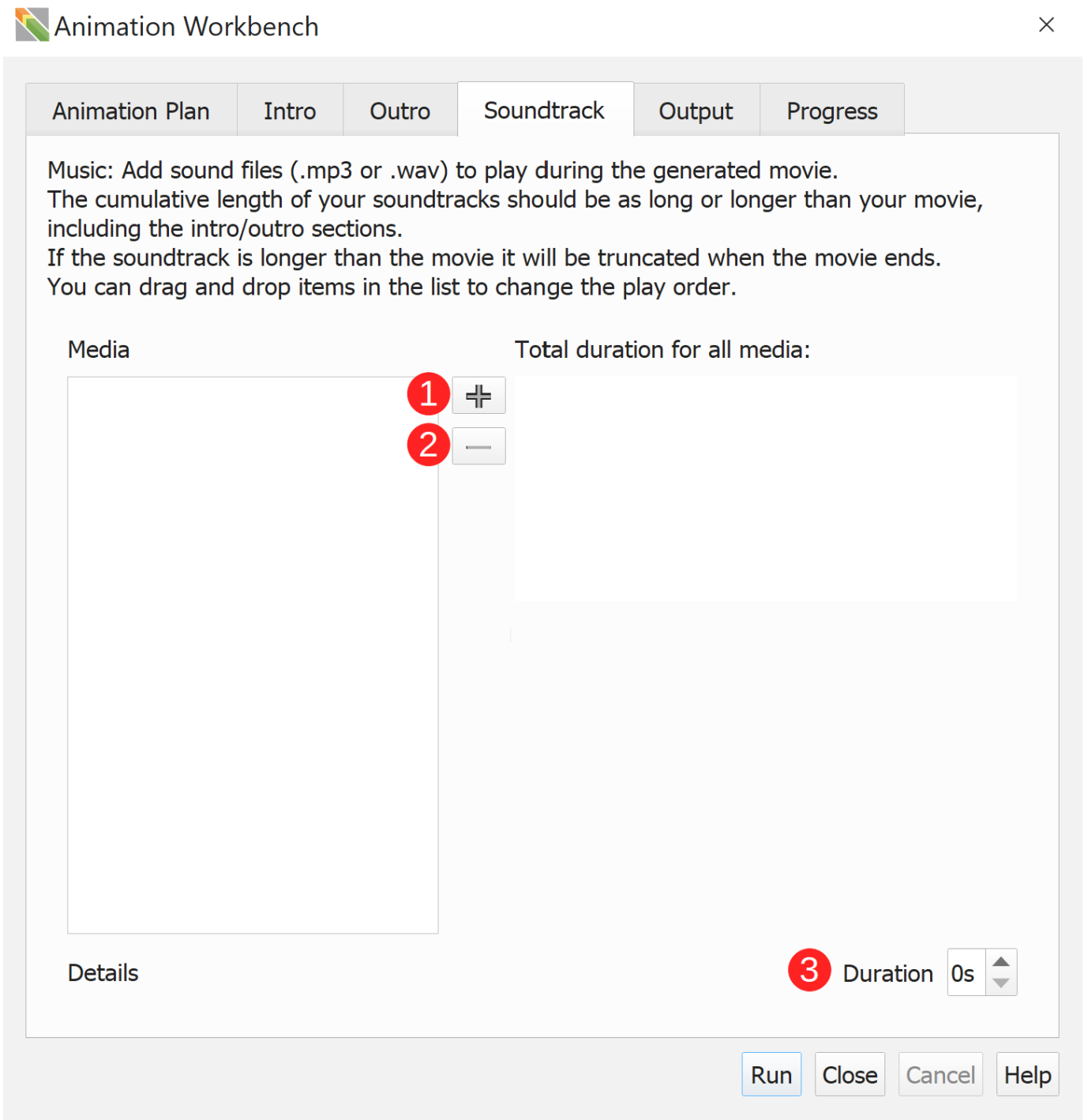
3.2.3 Outro Tab

Edit the outro section of the generated movie here.



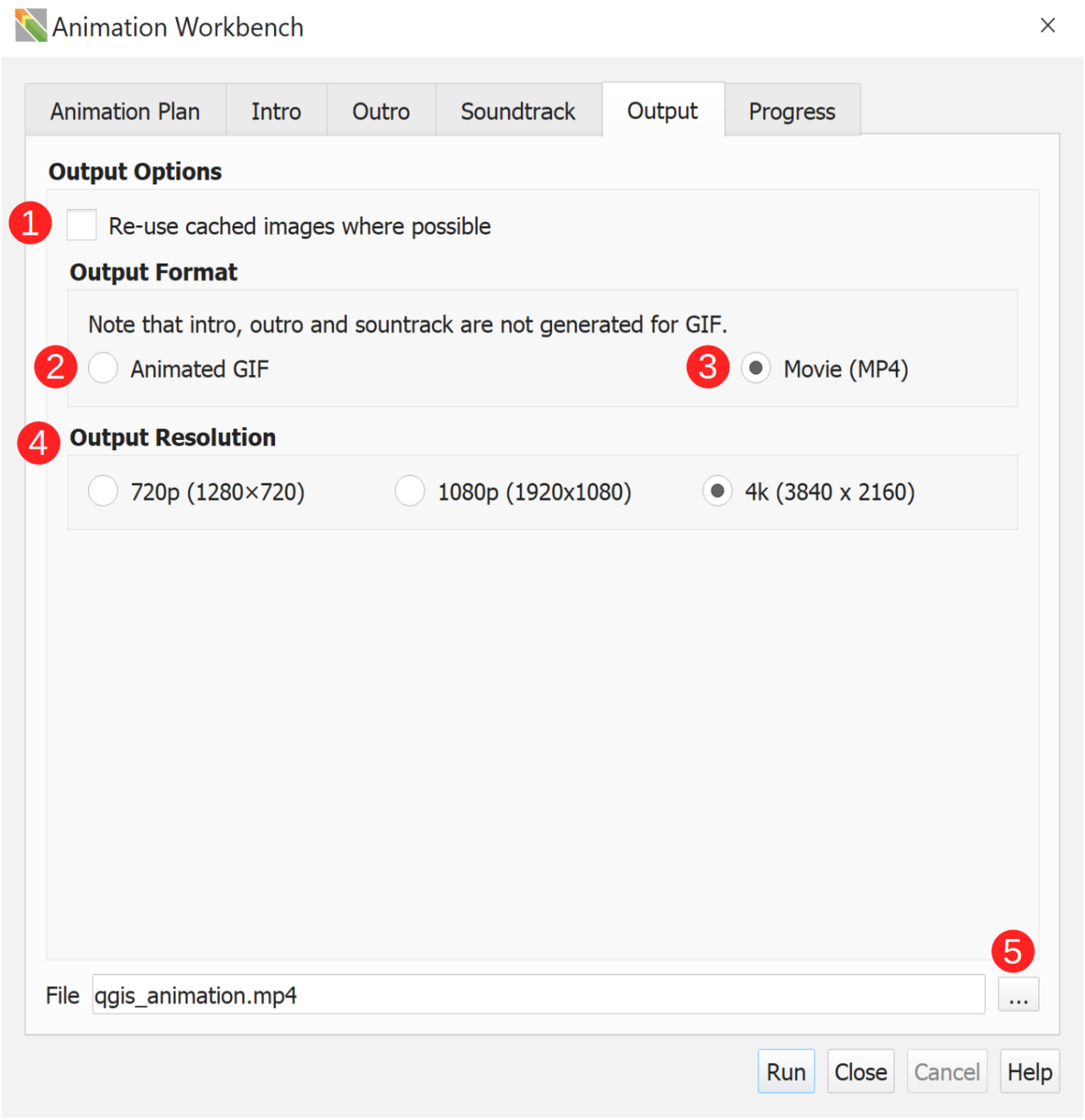
- Media: List of the various images or movies selected for the outro section. You can drag and drop items in the list to change the play order.
- 1 Add Media (Plus sign): Add images or movies
- 2 Remove Media (Minus sign): Remove images or movies
- 3 Duration: For images, you can set a duration for each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

3.2.4 Soundtrack Tab



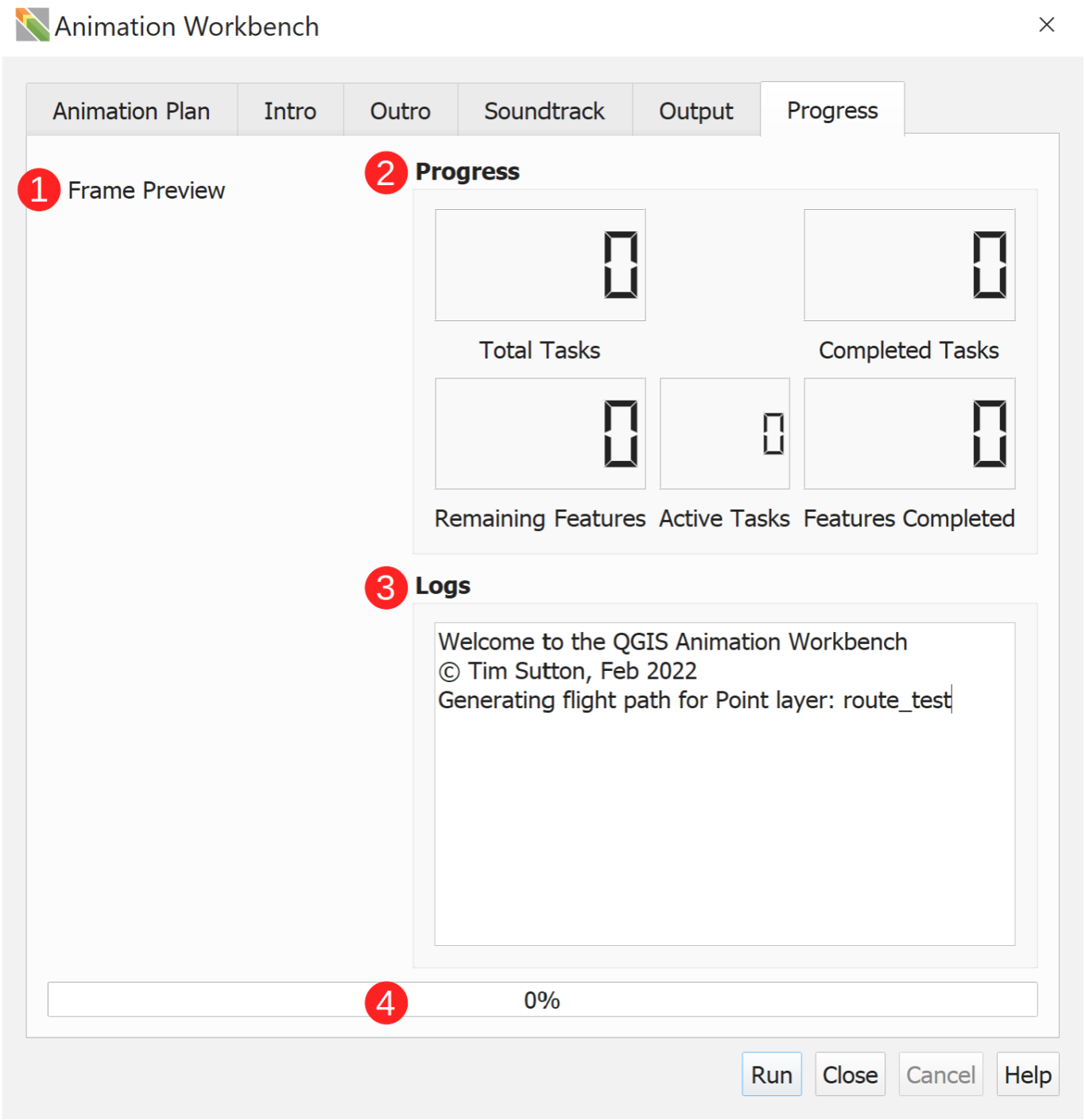
- Media: List of the various sound files (.mp3 or .wav) to play during the generated movie. You can drag and drop items in the list to change the play order.
- 1 Add Media (Plus sign): Add sound files (.mp3 or .wav) to play during the generated movie.
- 2 Remove Media (Minus sign): Remove sound files (.mp3 or .wav)
- 3 Duration: 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.

3.2.5 Output



- 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.
- 1 Re-use cached Images: This will not erase cached images on disk and will resume processing from the last cached image.
- 2 Animated GIF: For this export to work, you need to have the ImageMagick 'convert' application available on your system.
- 3 Movie (MP4): For this option to work, you need to have the 'ffmpeg' application on your system.
- 4 Output Resolution: Allows a user to specify one of three image resolutions for the output animation. The numbers in brackets represent the width and height of the output in pixels (i.e. width x height).
- 5 File selection (ellipsis): This lets a user select the location where the output will be stored.

3.2.6 Progress



- `1` Frame Preview: A preview of what each frame of the animation will look like. It changes automatically as the workbench runs.
- `2` Progress: 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.
- `3` Logs: A detailed list of what steps the workbench is doing (a record of processing)
- `4` Progress Bar: A visual representation of the workbench's progression as a percentage.

3.2.7 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.

3.3 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'
```

  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. 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.

4. Tutorials

4.1 Tutorial

5. Library

5.1 🧮 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.

5.1.1 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.

5.1.2 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 a current 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"

5.1.3 Planar/Sphere modes

These variables are available in the Planar or Sphere mode.

Variable	Notes
current_animation_action	Either "Hovering" or "Travelling"

When hovering

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

5.1.4 Example expressions

Visit the snippets section of our documentation for example expressions.

5.2 Snippets

5.2.1 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:

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

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

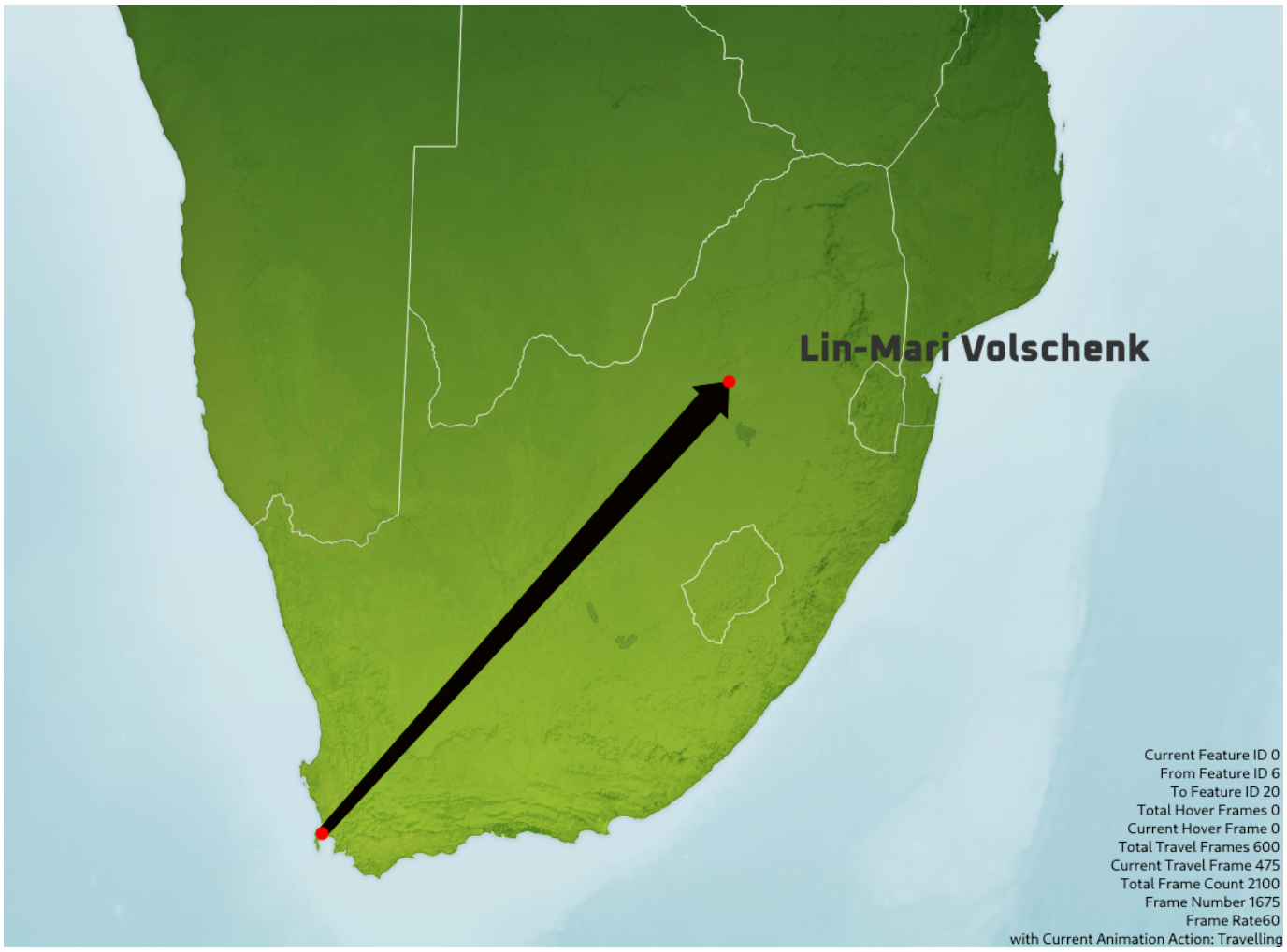
```
@project_home
||
'/gifs/cat_000'
||
lpad(to_string( @frame_number % 48 ), 2, '0')
||
'.png'
```

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`.

5.2.2 Line of travel

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

```
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)
```

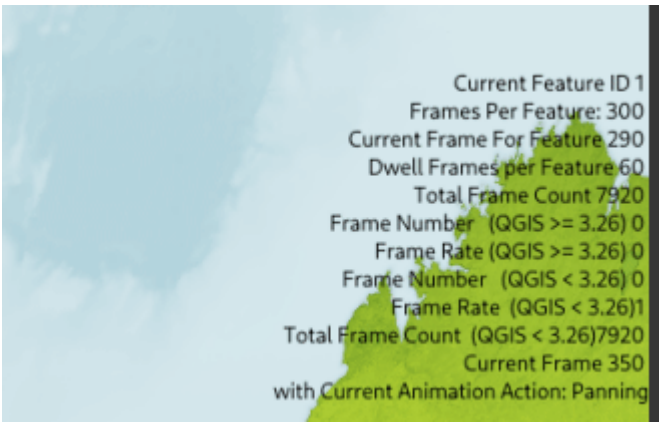


5.2.3 Showing diagnostic info as a copyright label

Showing diagnostic information in the QGIS copyright label:

```
[%
'Feature Variables:' ||
' \n------------------------' ||
' \nPrevious Feature ' || to_string(coalesce(attribute(@previous_feature, 'name'), '-'))  ||
' \nPrevious Feature ID ' || to_string(coalesce(@previous_feature_id, '-'))  ||
' \n' ||
' \nNext Feature ' || to_string(coalesce(attribute(@next_feature, 'name'), '-'))  ||
' \nNext Feature ID ' || to_string(coalesce(@next_feature_id, '-'))  ||
' \n' ||
' \nHover Feature ' || to_string(coalesce(attribute(@hover_feature, 'name'), '-'))  ||
' \nHover Feature ID ' || to_string(coalesce(@hover_feature_id, '-'))  ||
' \n' ||
' \nFrom Feature ' || to_string(coalesce(attribute(@from_feature, 'name'), '-'))  ||
' \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, 0))  ||
' \nTotal Travel Frames ' || to_string(coalesce(@travel_frames, 0))  ||
' \nCurrent Travel Frame ' || to_string(coalesce(@current_travel_frame, 0))  ||
' \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(@hover_feature), geometry(@previous_feature) ) ) ), 0) ||
' \nFrom Direction ' ||  coalesce(format_number(degrees( azimuth( geometry(@hover_feature), geometry(@next_feature) ) ) ), 0)
%]
```

Example output:



5.2.4 Variable size of labels

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

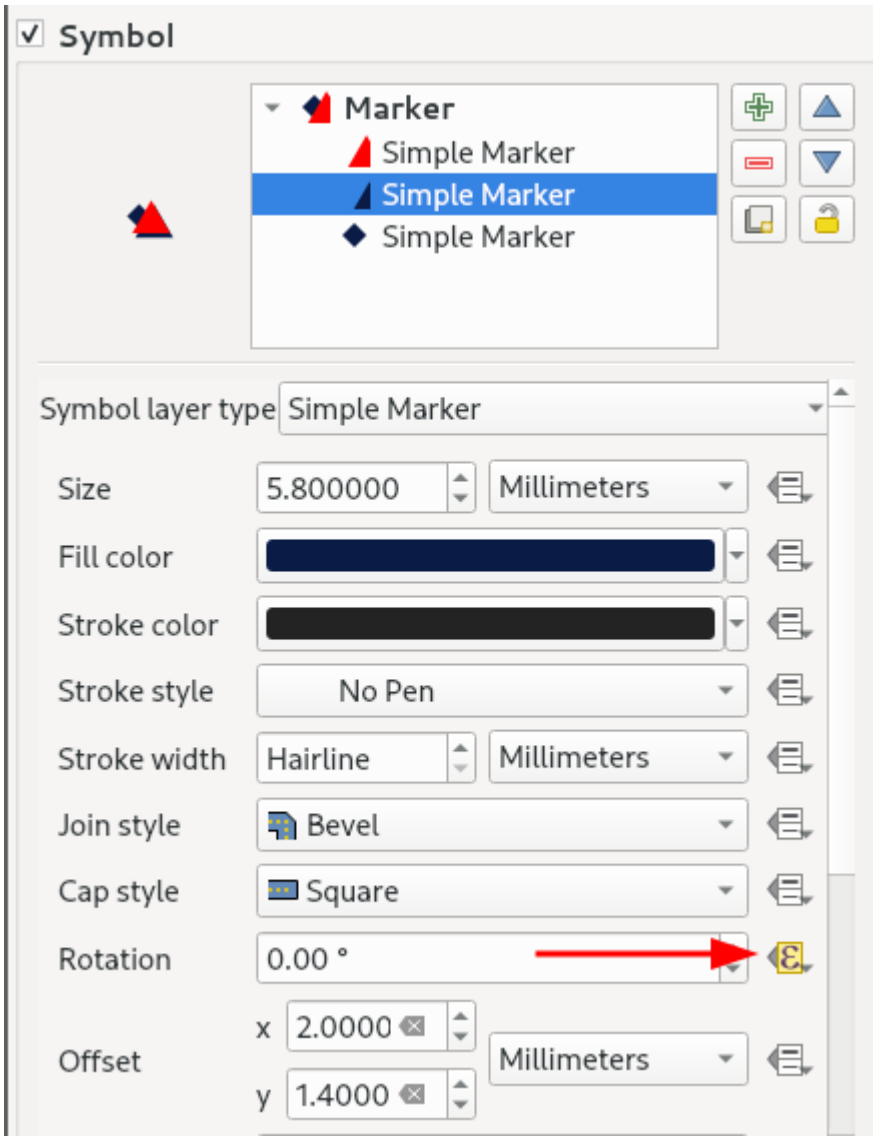
```40 * ((@frame_number % @hover_frames) / @hover_frames)

```
Calculating the angle between points

You can calculate the angle between the hover point and the previous point like this:

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

5.2.5 Rotation

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



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

```
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:

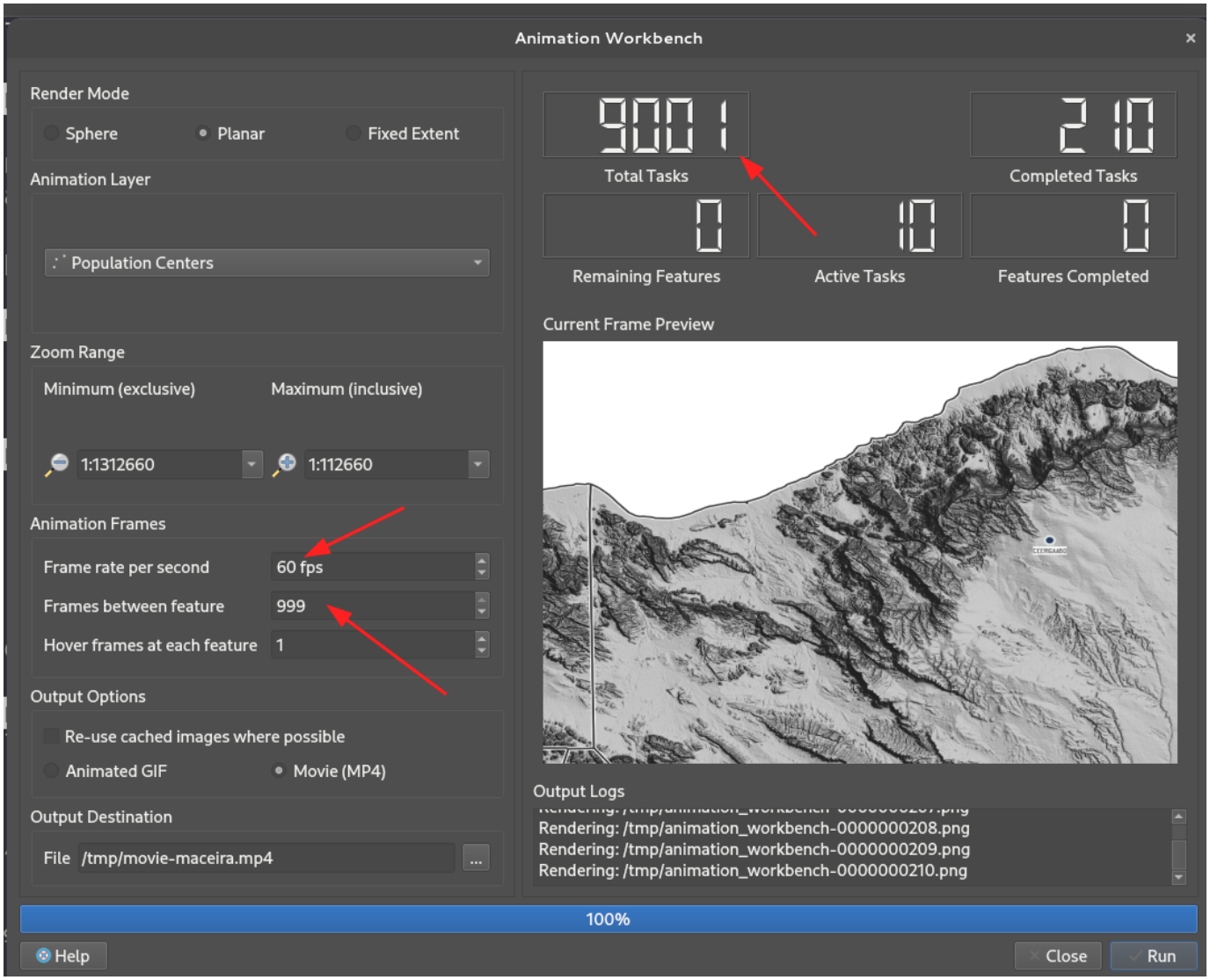
6. Frequently Asked Questions

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 you 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:

After processing:

And here is the resulting video:

https://youtu.be/1quc3xPdJsU

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?

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 a intended to provide an experimental, no-pressire space for us to work on something fun for QGIS.

7. Develop

7.1 Developer Notes

Setup

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

- debugpy
- convert (imagemagick)
- ffmpeg
- vscode (dont use flatpak, debugging will not work with QGIS)

Before starting development, you should check if there are any errors.

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

Enable the python 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.

DEBUGGING

```
TODO
```

PACKAGING

```
TODO
```

RUN TEST

```
TODO
```

7.2 Design

7.3 Working with documentation

Documentation is written using mkdocs.

7.3.1 Building documentation PDF

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

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

8. Contribute

8.1 Contribute

8.1.1 🔧 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

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

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:`

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

8.1.2 💬 Contributing

- Code of Conduct
- Contributing Guideline
- Commit Convention
- Issue Guidelines

9. Credits

9.1 Credits

9.1.1 Author

This plugin was developed by:

Tim Sutton	Nyall Dawson
	
Coder and Ideas Guy	Genius Guru of Awesomeness
timlinux @ github	nyalldawson @ github

9.1.2 Contributors

Thanks to:

- Mathieu Pellerin (@nirvn)
- Jeremy Prior (@Jeremy-Prior)
- 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.

https://github.com/timlinux/QGISAnimationWorkbench