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 3
   1.1 🤖 Why QGIS Animation Workbench? 3
   1.2 🎨 Features 3
2. Quickstart 6
   2.1 Installing the QGIS Animation Workbench plugin 6
   2.2 Initial Configuration 7
   2.3 Using the Animation Workbench 8
3. Manual 9
   3.1 Preparing your project 9
   3.2 The workbench user interface 10
   3.3 Under the hood 11
4. Tutorials 12
   4.1 Tutorial 12
5. Library 13
   5.1 🧮 QGIS Expression Variables 13
   5.2 Snippets 15
6. Frequently Asked Questions 20
7. Develop 21
   7.1 Developer Notes 21
   7.2 Design 22
   7.3 Working with documentation 23
8. Contribute 24
   8.1 Contribute 24
9. Credits 28
   9.1 Credits 28

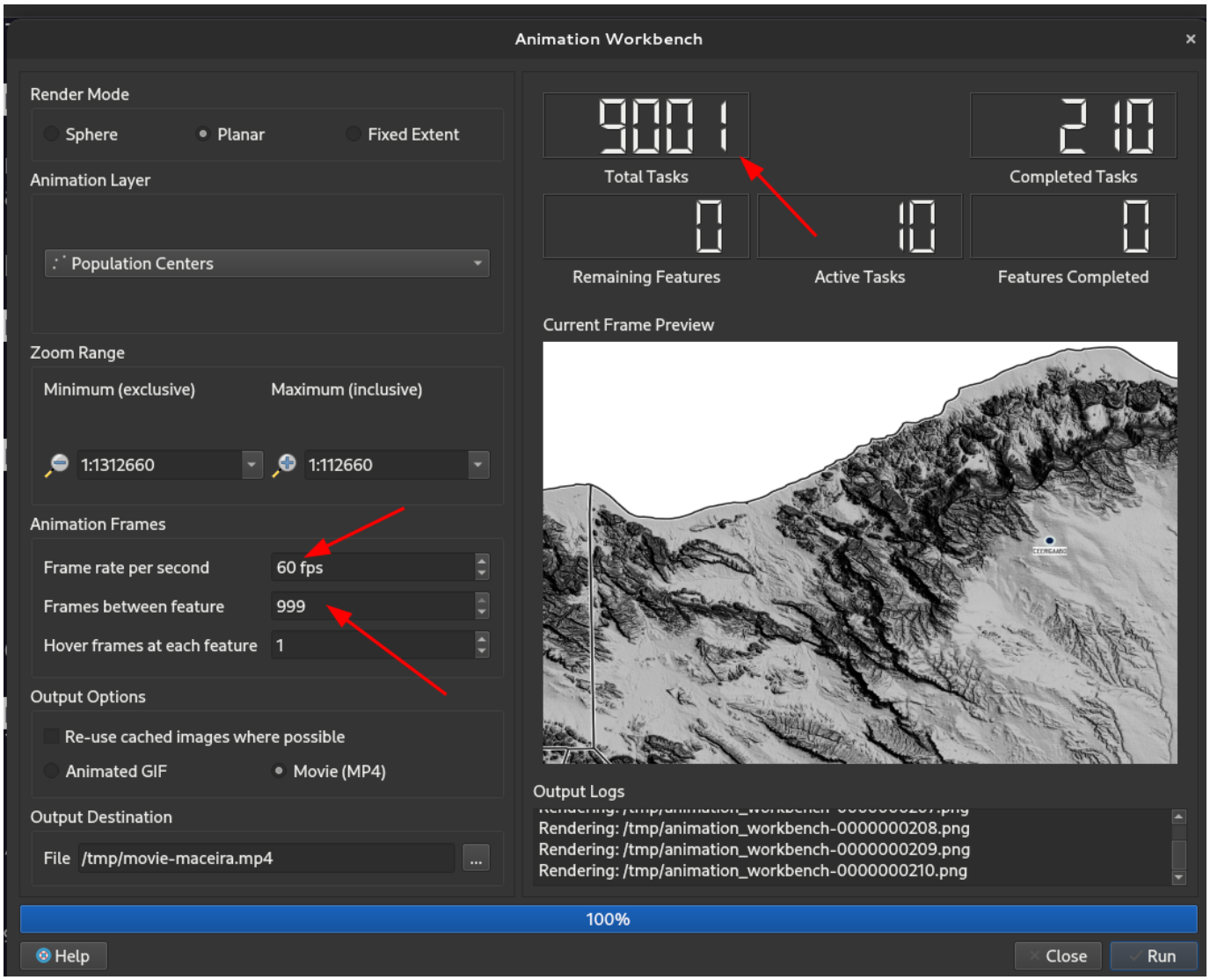
1. Welcome to the Animation Workbench

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. Here 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

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.

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

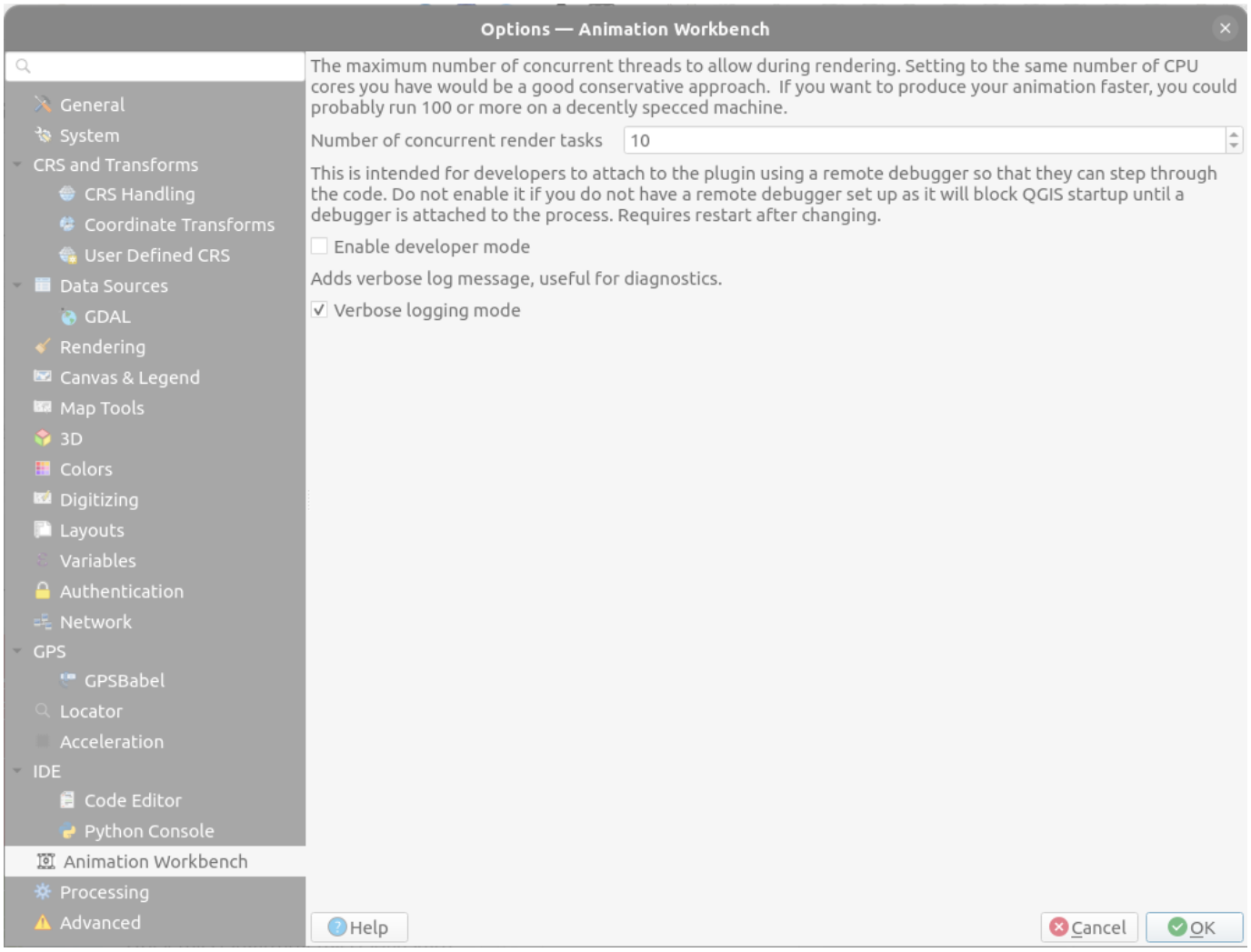
```
sudo apt install PyQt5.QtMultimedia
```

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



Currently there are just three options:

1. **Number of concurrent render tasks:** This is the number of concurrent tasks that will be used to render animations. The default is 10.
2. **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.
3. **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!

3. Manual

3.1 Preparing your project

3.2 The workbench user interface

3.3 Under the hood

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:

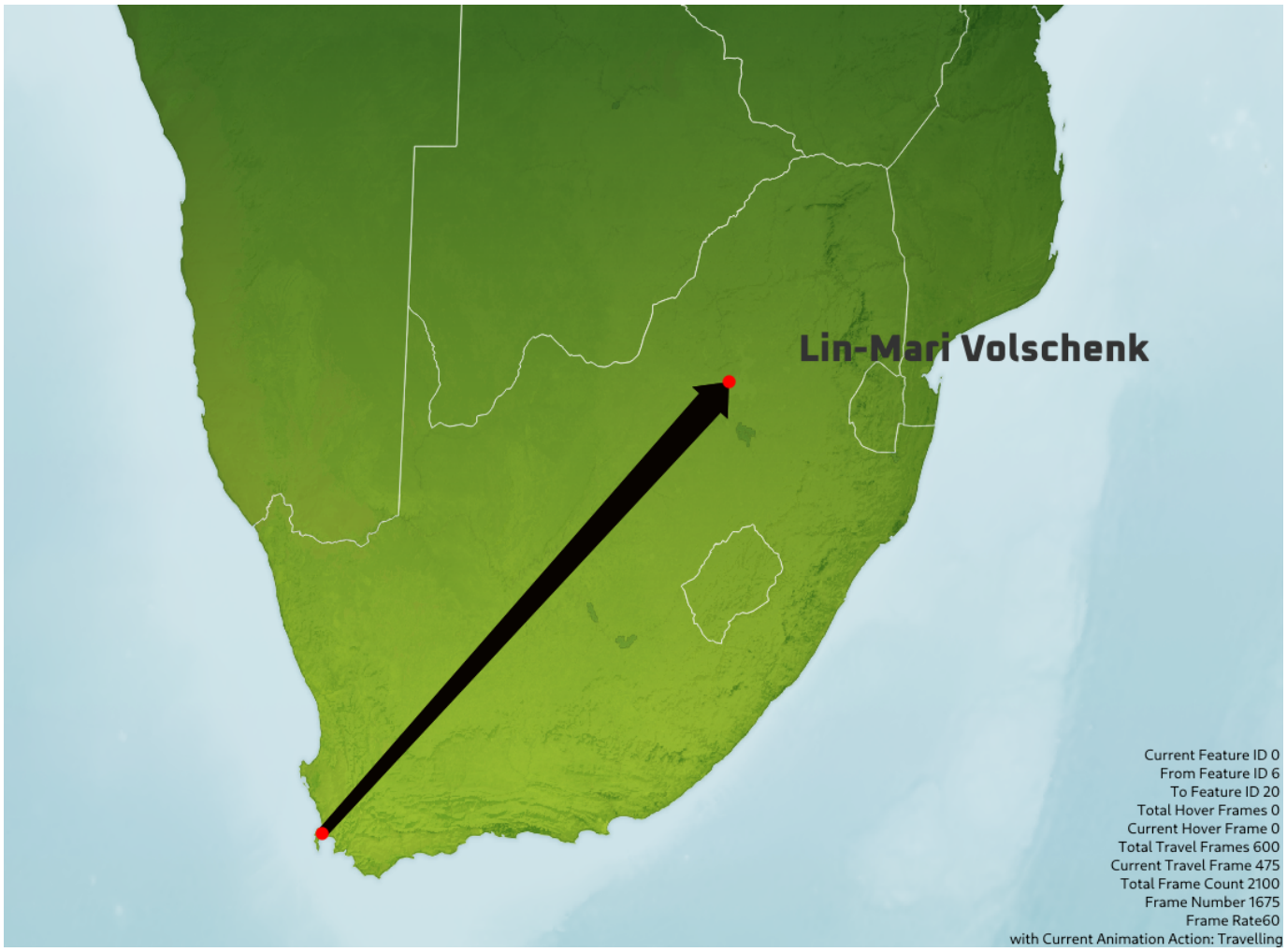
```
@project_home
||
'/gifs/cat_000'
||
lpad(to_string( @frame_number % 48 ), 2, '0')
||
'.png'
```

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)
```

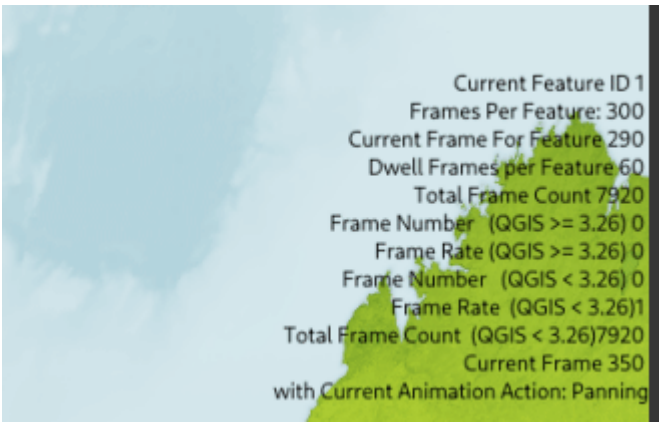


5.2.3 Showing diagnostic info as a copyright label

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:

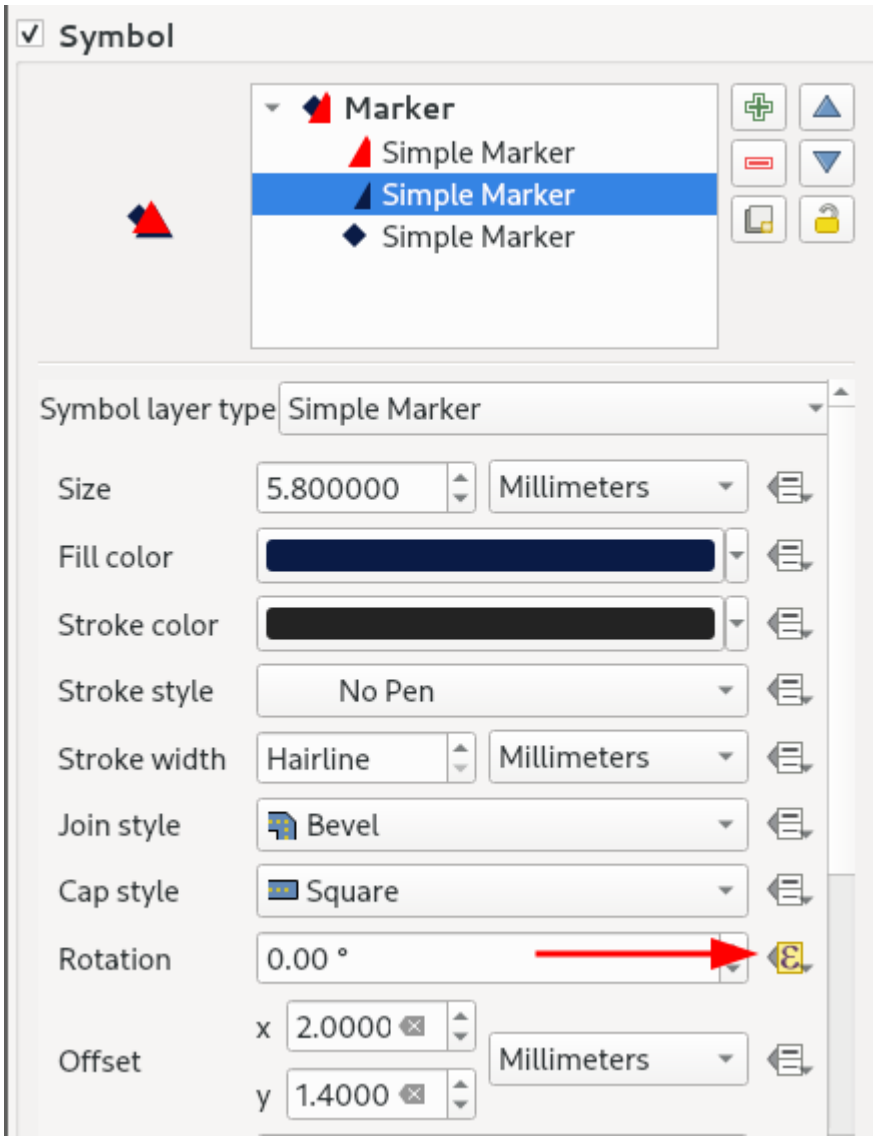
```40 * ((@frame_number % @hover_frames) / @hover_frames)

```
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

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

Commit type	Emoji
Package.json in JS	:package: `:package:`
Merging branches	:twisted_rightwards_arrows: `:twisted_rightwards_arrows:`
Bad code / need improv.	:hankey: `:hankey:`
Reverting changes	:rewind: `:rewind:`
Breaking changes	:boom: `:boom:`
Code review changes	:ok_hand: `:ok_hand:`
Accessibility	:wheelchair: `:wheelchair:`
Move/rename repository	:truck: `: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)

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