# Dynamic elevation profile lines as QGIS geometry generator

[15 Replies](https://hannes.enjoys.it/blog/2019/09/dynamic-elevation-profile-lines-as-qgis-geometry-generator/" \l "comments)

**Note**: You need the current master build of QGIS or wait for QGIS 3.10. I’m a cool kid.

Load a raster layer in QGIS.

Add a new Scratch Layer of Polygon type (in any CRS). Set its Symbology to Inverted Polygons. Use a Geometry Generator as symbol layer type. Set it to LineString/MultiLineString. Enter the expression below and adjust the layer ID (best enter the expression editor and find it in the “layers” tree). Then adjust the scaling factor at the very bottom.

-- UPPER CASE comments below are where you can change things

with\_variable(

'raster\_layer',

'long\_and\_complicated\_layer\_id', -- RASTER LAYER to sample from

-- this collects all the linestrings generated below into one multilinestring

collect\_geometries(

-- a loop for each y value of the grid

array\_foreach(

-- array\_foreach loops over all elements of the series generated below

-- which is a range of numbers from the bottom to the top of y values

-- of the map canvas extent coordinates.

-- the result will be an array of linestrings

generate\_series(

y(@map\_extent\_center)-(@map\_extent\_height/2), -- bottom y

y(@map\_extent\_center)+(@map\_extent\_height/2), -- top y

@map\_extent\_height/50 -- stepsize -> HOW MANY LINES

),

-- we want to enter another loop so we assign the name 'y' to

-- the current element of the array\_foreach loop

with\_variable(

'y',

@element,

-- now we are ready to generate the line for this y value

make\_line(

-- another loop, this time for the x values. same logic as before

-- the result will be an array of points

array\_foreach(

generate\_series(

x(@map\_extent\_center)-(@map\_extent\_width/2), -- left x

x(@map\_extent\_center)+(@map\_extent\_width/2), -- right x

@map\_extent\_width/50 -- stepsize -> HOW MANY POINTS PER LINE

),

-- and here we create each point of the line

make\_point(

@element, -- the current value from the loop over the x value range

@y -- the y value from the outer loop

+ -- will get an additional offset to generate the effect

-- we look for values at \_this point\_ in the raster, and since

-- the raster might not have any value here, we must use coalesce

-- to use a replacement value in those cases

coalesce( -- coalesce to catch raster null values

raster\_value(

@raster\_layer,

1, -- band 1, \*snore\*

-- to look up the raster value we need to look in the right position

-- so we make a sampling point in the same CRS as the raster layer

transform(

make\_point(@element, @y),

@map\_crs,

layer\_property(@raster\_layer,'crs')

)

),

0 -- coalesce 0 if raster\_value gave null

-- here is where we set the scaling factor for the raster -> y values

-- if things are weird, set it to 0 and try small multiplications or divisions

-- to see what happens.

-- for metric systems you will want to multiply

-- for geographic coordinates you will want to divide

)\*10 -- user-defined factor for VERTICAL EXAGGERATION

)

)

)

)

)

)

) -- wee