# Section : . Penyimbolan

## *Module : . Rule Based Symbology*

### Rule Based Symbology in Context

*"Rule based symbology allows you to to adapt cartographic representation of features according to their attributes"*

In this module, we will dig into how to make your map symbology adapt on a case-by-case basis for each feature as it gets drawn. Rule-based symbology uses expressions and scale levels to decide which treatment a vector feature should be given when drawing it to the map. Since rule-based symbology is driven by data, you should always familiarise yourself with the attribute table first so that you understand the data you have available to define your rules. Defining these rules is quite easy - let's dive in and see!

## You try:

***Goal: To show disputed areas on the map with different symbology for claimant countries and whether the claim is a break-away state.***

- Add the 10m disputed areas to your map.
- Use rule-based symbology to create two top level rules ('break away' and 'not break away')
- For the breakaway, draw an offset line which is coloured according to the left admin boundary's country.

**Check your results:**
Once you have your rules created, you should see borders around Georgia, Armenia etc. coloured according to the dispute type.

See if you can move the country coloured line to the other side of the dispute symbol line.

| Name | Expectation |
|---|---|
| Backdrop Layer | ne_10m_admin_0_countries |
| Disputed Areas Layer | ne_10m_admin_0_boundary_lines_disputed_areas |

| | |
|---|---|
| Renderer | Rule based |
| Break away rule | "featurecla" = 'Breakaway' |
| Not break away rule | "featurecla" !='Breakaway' |
| Refine rules | Add categories |
| Category field | adm0_left |
| Color ramp | Random colours |
| Line style | Dash line |
| Line width | 2mm |

**More about**

It is important to understand that when composing rules, you are creating a branching tree structure. The rule renderer does 'greedy' matching - it applies all of the matching rules to a feature. This allows you to either create a single symbol per rule match (if you construct your rules so that they are uniquely matched) or to 'layer' multiple symbols on top of each other as different rules are encountered and matched.

Because there is the possibility of multiple matching rules, you should take care to set your symbol levels if needed to ensure that as each rule is matched the symbology for that rule is drawn in the correct z-order.

You really need to understand how expressions work to make the most of rule-based rendering, so spend some time learning about them.

A good workflow is to start with a single symbol, categorised or graduated symbol for your layer, apply your changes and then change the renderer to Rule-based. Your symbology will be carried over to the rule-based renderer saving you some setup time.

One very good use case for rule-based rendering is to use scale based rules. For example, when displaying a layer at large scale, you might show lots of detailed classes in the symbology, then when zooming out to a small scale, you simplify the symbology to a single symbol renderer.

**Check your knowledge:**

1. Experiment with the rule based render to see if you can render a polygon layer as a point layer at large scale and a polygon layer at small scale:
   a. *This is not possible*
   b. *You can do this using python programming*
   c. *You can do this using virtual layers*

*d. You can do this using using a centroid fill*

### Module video tutorial:

- https://youtu.be/OMcEnKqkUQk

### Further reading:

- Rule-based-classification : https://docs.qgis.org/3.16/en/docs/user_manual/working_with_vector/vector_properties.html#rule-based-renderer

Download the sample data for the lesson from http://changelog.qgis.org/media/images/lesson/worksheet/external_data/02cf982f0e64a0689732f926c318872e939a1e82.zip.