ADDING 3D GIS VISUALIZATION AND NAVIGATION TO THE SPARQL QUERY LOOP

W. Beek [a,b,c], E. Folmer [a,d], L. Rietveld [a,c], T. Baving [a], V. van Altena [a,b,e]

[a] Kadaster, Apeldoorn, Netherlands - (tony.baving, vincent.altenavan).kadaster.nl
[b] VU University Amsterdam, Amsterdam, Netherlands - w.g.j.beek@vu.nl
[c] Triply, Amsterdam, Netherlands - laurens@triply.cc
[d] University of Twente, Enschede, Netherlands - e.j.a.folmer@utwente.nl
[e] University of Technology, Delft, Netherlands

Commission IV, WG IV/4

KEY WORDS: 3D visualization, geospatial data, Linked Data, SPARQL, REPL, data analytics

ABSTRACT:

3D environments allow advanced spatial navigation and visualization, but have traditionally provided limited support for performing non-spatial data analysis operations like filtering, joining, and integrating data on-the-fly. Linked Open Data provides advanced support for performing filters and joins over datasets that can be dynamically combined through SPARQL federation. Unfortunately, Linked Data results often lack intuitive visualization capabilities, making it relatively difficult to interpret the data for a data analyst. In this paper we present our integration of 3D visualization into the read-evaluate-print-loop of SPARQL query execution. We show how the inclusion of 3D visualization has concrete benefits for the SPARQL query writing process, and how our integrated solution is used to answer specific use cases that could not be answered before.

1. INTRODUCTION

3D environments allow advanced spatial navigation and visualization, but have traditionally provided limited support for performing non-spatial data analysis operations like filtering, joining, and integrating data on-the-fly. Linked Open Data provides advanced support for performing filters and joins over datasets that can be dynamically combined through SPARQL federation. Unfortunately, Linked Data results often lack intuitive browsing/navigation capabilities and advanced visualizations.

Because of the complementary nature of the two approaches, the combination of 3D GIS and Linked Open Data provides ample potential for data analysis use cases. Unfortunately, not that much prior work on truly combining 3D GIS and Linked Open Data has been performed. There existing prior work on semantically describing 3D objects in Linked Data (e.g., Pittarello et al., 2016), and some viewers are able to display (part of) a Linked Dataset within a 3D viewer (Bosca et al., 2005). However, what is currently lacking is 3D content that
is formatted in a standards-compliant way, is accessed through standardized means, and is visualized in a 3D environment.

The Dutch Cadastre and Triply have collaborated in order to improve the integration of 3D GIS within a Linked Data setting, specifically focusing on enhancing the data analyst experience while writing complex SPARQL queries. This paper presents how 3D GIS functionality was added to the SPARQL query loop, and how its support is beneficial for querying the Cadastre geospatial data assets.

2. THE SPARQL QUERY REPL

Performing complicated data analyses is akin to programming, in the sense that a complex query is not constructed all at once. Rather, query construction is a highly iterative process that consists of repeatedly changing the query until it gives the required result. In programming, this process is widely known as the read-evaluate-print-loop (REPL) is a well-known concept. In data analysis, we observe a similar process:

1. The client writes a query that is **read** by a SPARQL endpoint. We assume that the data analyst has access to a SPARQL editor with syntax highlighting and auto-completion functionality. That way, trivial iterations whose only purpose is to fix an obvious grammatical mistake and/or correct the incorrect spelling of a certain term are mostly ruled out.
2. Once a grammatically correct query is read by a SPARQL endpoint, it is **evaluated** against the underlying triple store. We assume that the endpoint is fully standards-compliant, and that the triple store only contains standards-compliant RDF data. That way, trivial evaluation errors that are due to violations of standards and/or quality issues in the data are mostly ruled out.
3. If a grammatically correct query is evaluated by a standard-compliant endpoint, a SPARQL result set is returned to the client. Since the result set format is standardized, the result set can be **visualized** (a generalization of merely printing the results) by the client. Furthermore, since Linked Data often makes use of shared vocabularies, it is possible to (semi-)automatically custom-tailor visualizations in terms of the most widely used Linked Data vocabularies. For example, result sets over a dataset that uses the GeoSPARQL vocabulary can often be automatically visualized on a map; result sets over a dataset that uses the DataCube vocabulary can often be automatically plotted in a diagram.
4. Based on the visualization a client may determine that some part of the query must be changed, which result in a new iteration through the **loop**.

This read-evaluated-print-loop (REPL) principle is implemented by YASGUI, an integrated SPARQL editor and result set visualizer (Rietveld et al., 2017) that is developed by Triply (https://triply.cc) and used as a component by many Open Source projects and data publishers. In collaboration with the Dutch Cadastre Data Platform (https://data.pdok.nl), YASGUI was previously extended to support GeoSPARQL, the GIS extension to SPARQL that is standardized by the OGC. With this extended support it is possible to query for geospatial relationships, return them in a standard-compliant result set formal, and automatically display them on a 2D Leaflet map (Beek et al., 2017).

3. BENEFITS OF 3D SPARQL

While YASGUI was extended in 2017 to automatically visualize 2D geospatial information on a Leaflet map, no 3D geospatial support was available. In fact, 3D results were treated in exactly the same way as 2D results: the altitude was simply not processed.

At the same time, we can identify several generic benefits of adding 3D support to the REPL principle:

1. Since we live in a 3D world, 3D result set visualizations are able to more closely mimic reality than 2D visualizations of the same results. As such, 3D visualizations are more engaging to the user than their corresponding 2D visualizations.
2. When visualizing statistical data, it is often useful to bind multiple display properties to different result set variables. For example, when geographic regions are displayed on a map, the crime statistics of those regions can be displayed by colouring those regions in an appropriate gradient. When we visualize the same result set in a 3D environment, we can display one more attribute of the respective regions. For example, we can display the average income in terms of height.
3. 3D environments allow multiple perspectives on the data. When a building is displayed on a 3D map, we can only present a birds eye view of the situation (i.e., we can display the shape from above). In 3D environments we can display multiple views: we can look from behind the building, or we can look from the ground up to a building. Each view may show some additional information about the building.

In addition to these generic benefits, we have encountered several use cases in which 3D support is not only convenient but also necessary in order to allow query results to be interpreted correctly. Indeed, the correct interpretation of intermediate query results is required in order to be able to make the correct edits for the next iteration of the query:

1. The Dutch Cadastre stores data about the buildings, businesses, and apartments in The Netherlands. If only a 2D visualization of query result sets is available, building that contain multiple businesses and/or apartments on different floors of the same building are displayed on top of one another. As such, it is not possible to see on which floor a business or apartment is located.
2. The Dutch Cadastre publishes a dataset on zones in The Netherlands where drones are not allowed to fly. These zones are not only expressed in terms of longitude and latitude, but also in terms of altitude. In order to correctly depict the zone where drones are allowed to fly, a 3D visualization is needed.
3. For emergency services like the fire brigade, it is essential to know as much as possible before entering a building during an emergency. How many appartments, where are the entrances, on which level, and so on.

4. IMPLEMENTATION

In order to integrate 3D support in the SPARQL REPL, we will first take a look at the read component, which consists of the data that is stored in the triple store and the query that is written in order to be evaluated over that data. Even though the GeoSPARQL standard does not mention 3D specifically, the datatypes, relations, and functions it defines can also be applied to 3D shapes.

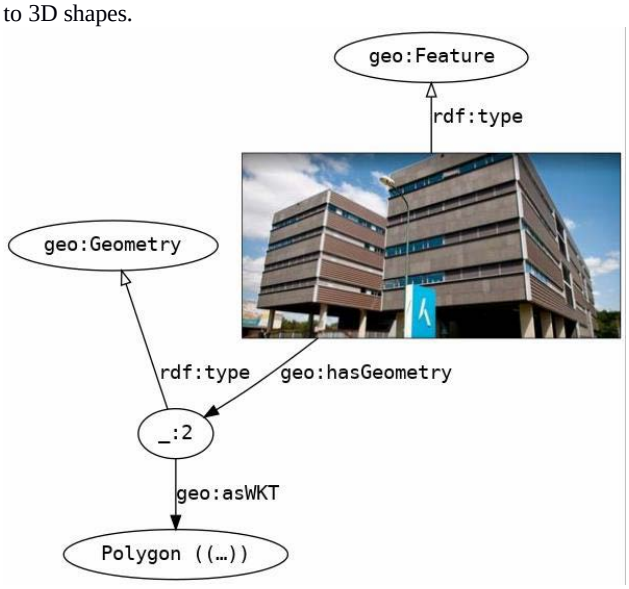

Figure 1. RDF Graph with 3D geometry

Figure 1 shown an example of a small RDF graph that encodes a 3D geometry. Firstly, contains a node representing a particular building, together with a triple that asserts that this building is a feature. Secondly, the graph contains a node that represent the geometry of that building, and a relationship between the feature and the geometry. Thirdly, a node represents a particular serialization of the geometry. In this case, a serialization in Well Known Text (WKT). Such a serialization starts with a keyword that indicates the kind of shape involved, and is followed by nested lists of coordinates. When writing a SPARQL query, the data analyst is able to retrieve the data in various ways. She may first retrieve the feature based on some other criteria (e.g., the address of the building), and then also retrieve its geometry and shape. Alternatively, the data analyst may first retrieve the shape based on some geospatial criterion (e.g., proximity to a point of interest), in order to subsequently retrieve the geometry and feature.

With respect to the **evaluate** component, i.e., the triple store, it is important to choose one that supports 3D. Unfortunately, at the moment there are no good options for this in the marketplace. While most triple stores allow 3D geometries to be stored, some do not allow them to be retrieved through SPARQL. Specifically, such triple stores will actively remove the Z coordinate from 3D shapes. This is worse than not supporting 3D, since that would at least leave the plain WKT string intact. When 3D information is actively purged from SPARQL results, it is impossible for YASGUI to display the data correctly. Other triple stores do preserve Z coordinates, but do not support the GeoSPARQL vocabulary. Some triple strores do support geospatial filters and

relations, but with non-standardized, custom-tailored notation. The very few triple stores that do support GeoSPARQL notation do not always apply effective indexing on geometries, resulting in poor performance for some queries.

The last component that must be present in order to add 3D support to the SPARQL REPL is the **print** or visualization component. Firstly, when YASGUI receives a query result set from the triple store, it must know how to interpret 3D shapes. We focus here on the most common SPARQL SELECT query form. A SELECT query returns results in terms of a fixed number of columns that correspond to a sequence of projection variables. Multiple query results amount to multiple sequences or rows of bindings of RDF terms to these projection variables. Whenever an RDF term in such a binding has the standardized datatype IRI `geo:wktLiteral`, YASGUI is instructed that a 3D shape is present. Secondly, YASGUI must be able to visualize the detected 3D shapes within a 3D environment. Previously, automatic visualization of 2D shapes was implement by including a plug-in that is based on the Open Source Leaflet library (h ttp ://leafletjs.com). For the current extension, a plug-in is added that is based on the Open Source Cesium library (https://cesiumjs.org). Cesium is not directly able to interpret the WKT formatted serializations that are present in SPARQL result sets, but it is easy to transform WKT serializations into GeoJSON, or another format that is supported by Cesium.

(`?varHeight` in Table 1). The height variable can be bound within a SPARQL query, either based on a query variable which derives its bindings from the data itself, or by simply binding the height variable to a static value that will display all shapes at the same height. Since the earth is not a perfect sphere, 3D shapes that are displayed relative to the earth's surface need additional information about the height at which the earth's surface is located for each query result. For this purpose, the `?varZ` variable is recognized by the 3D plug-in. This means that *if* a SPARQL query result includes a detailed 3D WKT literal, *then* the associated 3D shape is immediately drawn correctly. However, in case a 2D WKT literal is included instead, the additional variables `?varHeight` and `?varZ` can be used.

Element	**Variable template**
Shape	`?var`
Colour	`?varColor`
Complex label	`?varLabel`
Extrusion	`?varHeight`
Offset	`?varZ`
Simple label	`?varName`

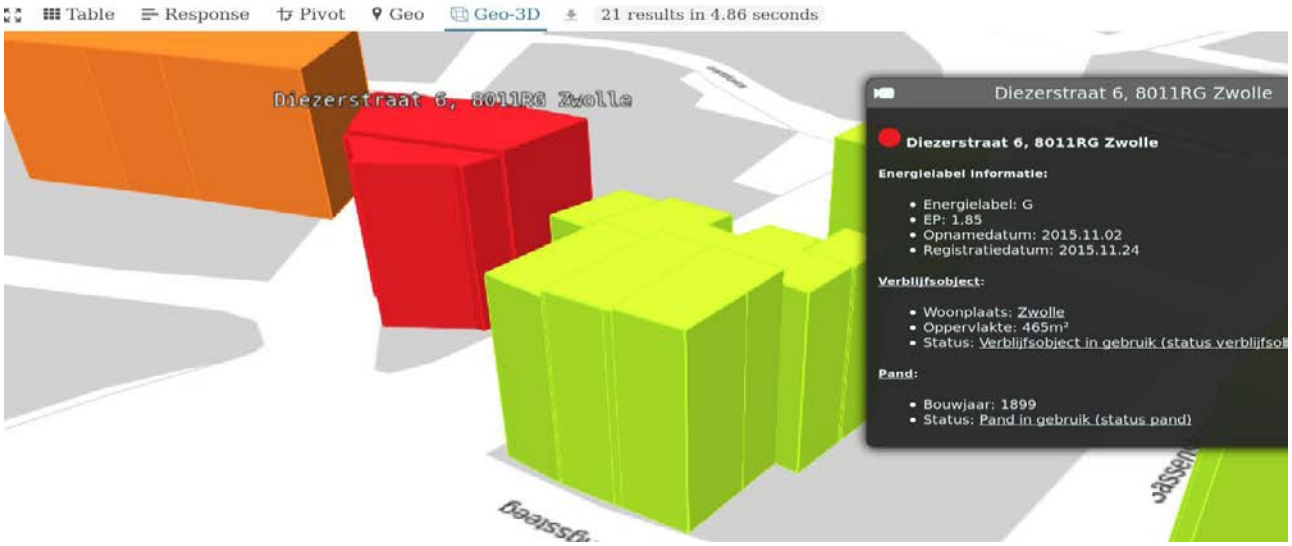

Figure 2. YASGUI result

Besides the ability to display 3D shapes in Cesium, the YASGUI plug-in includes additional support for colouring 3D shapes and for displaying labels. These labels can be displayed within the 3D environment itself (for simple textual labels) and/or in an HTML overlay (for complex labels that can include mark-up and media). In order to associate the right colour and/or label with the right shape, the visualization plug-in recognizes specific patterns in variable names (Table 1). Since variable names are arbitrary, this additional functionality is compatible with the SPARQL standard.

At the moment, very few Linked Datasets contain 3D shapes that are represented by WKT literals and GeoSPARQL properties. As such, the impact of the SPARQL extension would have been quite small. However, there is a lot of 2D Linked Data encoded in datasets today. The plug-in therefore adds specific support for visualizing 2D shapes with an added height property

5. EXAMPLES OF USE

In this section we present some concrete example of using 3D visualization support within the YASGUI REPL. Figure 2 shows the result of retrieving the energy labels of a street in the city of Zwolle. Since the result set contains 3D geometries, these are automatically drawn in the 3D viewer. Energy consumption is expressed in labels that are associated with recognizable colour codes. In our SPARQL query, we are not only binding the geometries of the buildings, but also their energy labels mapped to their respective colour codes. Now it is immediately identifiable which building has a certain energy label. When a building is selected, its textual label (the binding of ?varName in the SPARQL projection) is shown inside the 3D environment, hovering over the building. In addition, the building's HTML labels is shown in the panel to the right hand side. The HTML snippet in this panel contains additional information about the

selected building, such as its Cadastral identifier, it's current status (occupied or not) and use (residence or business). It also contains more information about the energy labels, including when the measurement was performed.

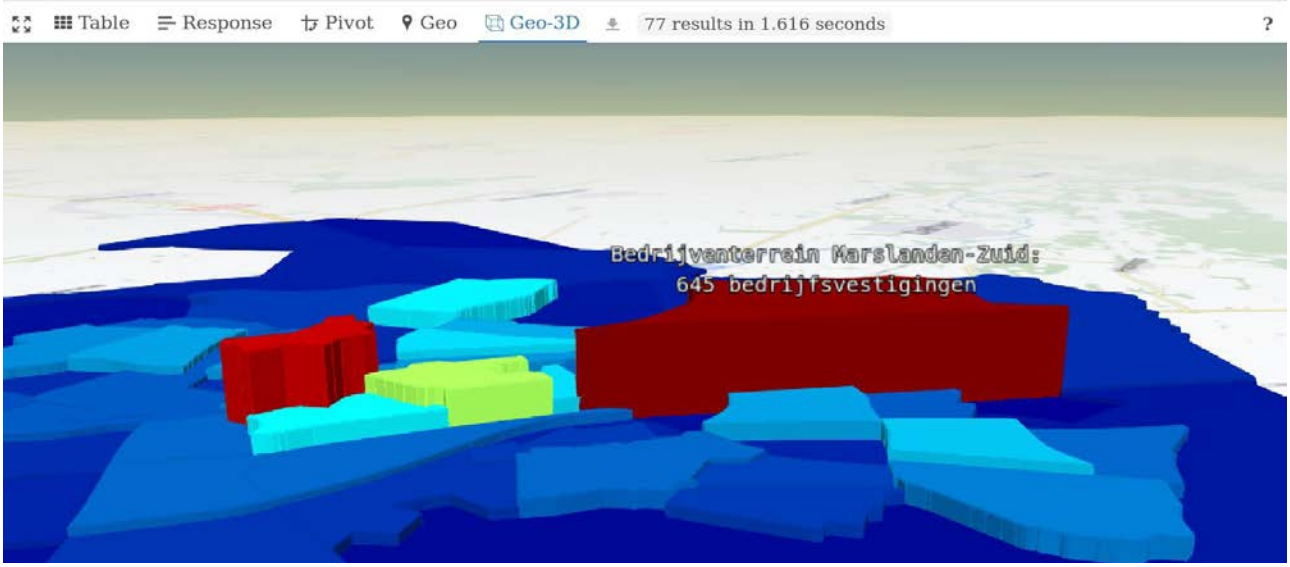

Figure 3. Result with number of businesses as 3D height

Figure 3 shows the result of retrieving the number of businesses for each neighbourhood in the city of Zwolle. In the SPARQL query, we bind the 2D shape of each neighbourhood to the projection variable ?var, and bind the (normalized) number of businesses to the projection variables ?varColor and ?varHeight. The height of the shapes now expresses the number of businesses. This is an example of a query where the Linked Data only contains 2D shapes, but the query visualization is still able to display 3D.

6. CONCLUSION

This paper has described how the SPARQL REPL loop can be extended to include 3D visualization in the 'read' or visualization component. It has outlines generic benefits of D visualization for the data analyst, as well as several concrete use cases that can be supported. By introducing a specific variable template, we are able to associate colours and labels to 3D shapes, improving the visualizations even further. We are also able to associate heights and offsets to 2D shapes, thereby extending the applicability of the 3D plug-in to 2D Linked Data as well.

7. REFERENCES

Beek, W., & Folmer, E. (2017). An Integrated Approach for Linked Data Browsing. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 42, pp. 35-38.

Beek, W., Folmer, E., Rietveld, L., & Walker, J. (2017). GeoYASGUI: The GeoSPARQL Query Editor and Result Set Visualizer. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 42, pp. 39-42.

Bosca, A., Bonino, D., Pellegrino, P. 2005. OntoSphere: More Than a 3D Ontology Visualization Tool. Proceedings of the 2nd Italian *Semantic Web Workshop: Applications and Perspectives (SWAP).*

Pittarello, F., De Faveri, A. 2006. Semantic Description of 3D Environments: A Proposal Based on Web Standards. *Proceedings of the 11th International Conference on 3D Web Technology*, pp. 85-95.