QGIS Grant Programme Applications

March 2017

Contents

Items are listed in order the applications were received (i.e. the order is not significant).

1 IMPLEMENT QGIS METADATA STORAGE SUPPORT

Proposer: Joana Simoes

Amount: €7991,38

Details: This development is part of QGIS enhancement #91: “QGIS Enhancement: Overhaul Metadata Management In QGIS”.

https://github.com/qgis/QGIS-Enhancement-Proposals/issues/91

The overall goal of this enhancement is to provide the infrastructure in QGIS to author, consume and share standards-based metadata. The proposal in-hand, focuses on the specific task of adding support to an external format for storing metadata, the “metadata store”. The goal is to support portability, enabling users to share their metadata, even in offline scenarios.

Although enhancement #91 aims to support both, remote and local stores, in this proposal we will focus on local stores, only.

The idea is to implement an abstract metadata store, which will have a polymorphic behavior, according to the data format. For instance in the case of a PostgreSQL DB, the method “save” will create a table on the database, whether in the case of a Shapefile, it would create an XML file.

Some formats, such as text files, can be more limited than others. For that reason, we will create a “prime” format, the “QGIS metadata store”, which can accompany more restrictive formats.The prime format will be an SQLite database, because of its lightweight, and because it is well-known within the QGIS community.

As the goal is to support a diversity of formats in the future, we will design an infrastructure to accommodate that, but in the scope of this proposal we will focus on the simple use case of creating an xml file, and creating the SQLite data store.

Along with this development, we will implement a user interface to allow the user to configure serialization/deserialization behavior, e.g.: in which format we should write metadata, and where.

The QGIS metadata store will be synced with any changes that we apply to the metadata. In the moment that we export metadata into XML, it will write those changes to the XML file.
Metadata search will also be polymorphic, according to the format. In this proposal we will implement a basic text search in the QGIS metadata store.

History: This proposal will build on other Work Packages (WP) which are part of the QGIS enhancement #91. Specifically, it will build on the results of WP's 1: "Schema Definition/Selection:", and 2: "QGIS Metadata API".
The good news is that funding was already allocated for these WP's, and the development is scheduled to start in March. This proposal will also build on the work of the geocat fork of the gpkg plugin, which aims at storing a QGIS project in a geopackage, using OGC standards.

https://github.com/GeoCat/qgpkg

Qualifications: I have written the GeoCat fork of the QGIS gpkg plugin, which is designed to support the offline portability of a QGIS project, including data, styles and context. This plugin adds support for reading a geopackage extension, which is entirely based on OGC standards (OWC, SLD).

https://github.com/GeoCat/qgpkg

I have more than 10 years of professional experience with C++, designing, and leading the development complex applications, mostly related to GIS and databases. I have more than eight years of experience in developing cross-platform applications with the Qt framework.
Github repo: https://github.com/doublebyte1

I have been an earlier adopter of QGIS, have been to two HackFests and have published one plugin, in the official QGIS repositories. During my period at the Food and Agriculture Organization (FAO-UN), I was an evangelist of QGIS, and adopted it as primer desktop GIS, when creating the FAO training materials.
Implementation Plan: Since the "QGIS Metadata API", upon which this proposal is dependent on, won't be ready until the 15th April, we envision the developments to start anytime after that. If this timetable holds, the QGIS Hackfest in Essen (17th April) would be a good opportunity to have the development kick-off. The efforts to complete this task are estimated in 15 days of work, which could be spread over 2 months. We are targeting release 3 for these core developments, and although we are not completely sure that they will be ready before the official release, it is still a possibility on the table and something we aim for.

Proposal Link: http://doublebyte.net/2017/03/03/welcoming-the-qgis-metadata-store/

2 IMPLEMENT GUI AND IO HANDLERS FOR PROJECT METADATA TEMPLATES

Proposer: Joana Simoes

Amount: €2142,97

Details: This development is part of QGIS enhancement #91: “QGIS Enhancement: Overhaul Metadata Management In QGIS”.

https://github.com/qgis/QGIS-Enhancement-Proposals/issues/91

The overall goal of this enhancement is to provide the infrastructure in QGIS to author, consume and share standards-based metadata. The proposal in-hand focuses on the specific task of easing the authoring of metadata, by allowing users to create a metadata template, which would then be reused across the project, enabling the automated population of metadata.

This proposal adds the support to two events:

- Filling metadata the template, which would then be associated to the project; this can happen in one tab of the project settings.
- Automated population of metadata for a layer, based on this template; this can be triggered through the layer properties, or when the user loads/creates a layer.

This template is based on the QGIS internal schema, developed on another WP.
One optional enhancement would be to support the import/export of this template, so that it could be shared across an organization. One user could also have multiple templates, according to the layers he was working on (see image bellow). Both these scenarios would require detaching the template from the project file and storing it in an external format.

History: This proposal will build on other Work Packages (WP) which are part of the QGIS enhancement #91. Specifically, it will build on the results of WP’s 1: “Schema Definition/Selection:“, and 2: “QGIS Metadata API”.

The good news is that funding was already allocated for these WP’s, and the development is scheduled to start in March. This proposal will also build on the work of the geocat fork of the gpkg plugin, which aims at storing a QGIS project in a geopackage, using OGC standards.

https://github.com/GeoCat/qgpkg

Qualifications: I have written the GeoCat fork of the QGIS gpkg plugin, which is designed to support the offline portability of a QGIS project, including data, styles and context. This plugin adds support for reading a geopackage extension, which is entirely based on OGC standards (OWC, SLD).

https://github.com/GeoCat/qgpkg

I have more than 10 years of professional experience with C++, designing, and leading the development complex applications, mostly related to GIS and databases. I have more than eight years of experience in developing cross-platform applications with the Qt framework.
Github repo: https://github.com/doublebyte1

I have been an earlier adopter of QGIS, have been to two HackFests and have published one plugin, in the official QGIS repositories. During my period at the Food and Agriculture Organization (FAO-UN), I was an evangelist of QGIS, and adopted it as primer desktop GIS, when creating the FAO training materials.

Implementation Plan: Since the “QGIS Metadata API”, upon which this proposal is dependent on, won’t be ready until the 15th April, we envision the developments to start anytime after that. If this timetable holds, the QGIS Hackfest in Essen (17th April) would be a good opportunity to have the development kick-off. The efforts to complete this task are estimated in 15 days of work, which could be spread over 2 months. We are targeting release 3 for these core developments, and although we are not completely sure that they will be ready before the official release, it is still a possibility on the table and something we aim for.

Proposal Link: http://doublebyte.net/2017/03/03/welcoming-the-qgis-metadata-store/

3 EXTEND UNIT TEST COVERAGE FOR GEOMETRY CLASSES

Proposer: Nyall Dawson

Amount: €2000

Details: Since QGIS 2.8, there has been an increased focused on creation of quality automated regression ("unit") tests designed to flag issues in code before the code is introduced to the QGIS codebase. The increase in stability of recent QGIS versions can be directly attributed to this growth in unit testing. Despite this, many areas of the QGIS codebase remain with little or no unit test coverage.

One critical area which has insufficient unit tests is the geometry classes. The geometry classes form the basis of all geometry interpretation, algorithms, and rendering within QGIS. In order to provide stable QGIS releases, it is crucial that these fundamental classes are rock-solid, efficient, and do not suffer regressions between releases.

Some years ago (shortly after the introduction of the new geometry engine, in which support for Z/M values and curved geometries was added) I added full test coverage for the Point and Linestring classes, and partial coverage for the Polygon class. Unfortunately, writing geometry tests is tricky and time consuming. There's many corner cases with unusual or invalid geometries which need to be tested. The time commitment required prevented me from writing additional tests, and to date the remaining classes (including multi geometries and all curved geometry types) have little or no test coverage.

This proposal covers writing additional unit tests to cover all the remaining geometry classes.

It is important to note that unit tests do NOT ensure bug free software. Unit tests only protect existing logic and avoid regressions when the covered parts of the code base are changed in future releases. Despite this disclaimer, the process of creating unit tests usually stress-tests existing code and in itself CAN reveal existing bugs. This was certainly the case when the existing tests for Point and Linestring classes were added - creation of the tests alone resulted in many fixed bugs and stabler Point and Linestring geometry handling.

History: This work would continue on from work I begun a number of years ago to provide 100% unit test coverage for the base geometry classes.

Qualifications: I have a long history of quality contributions to the QGIS project, and am currently one of the most prolific committers to the QGIS codebase. I have a long history with adding unit tests to QGIS and advocating for their increased usage amongst developers.

Implementation Plan: This work would be targeted to the QGIS 3.0 release, and would be committed to the codebase prior to the feature freeze/bug fixing period leading up to the 3.0 release.

Proposal Link:

4 REMOVE QGSWXSPROJECTPARSER

Proposer: DHONT René-Luc

Amount: €3000

Details: https://github.com/qgis/qgis3.0_api/issues/57

History: Two enhancements needed has been already done: QgsProject is a singleton https://github.com/qgis/qgis3.0_api/issues/8 and QEP 74: QGIS server code refactoring for QGIS 3.0 https://github.com/qgis/QGIS-Enhancement-Proposals/issues/74
The QgsWCSProjectParser has already be removed. The QgsWFSProjectParser removal is a work in progress. The big part of the work will be to remove the QgsWMSProjectParser, because a big part of the code has to be rewrote.

Qualifications: I have made some work on QGIS Server: Implementing WFS and WCS, fixing bugs in WMS, and started removing QgsWxSProjectParser.
https://github.com/qgis/QGIS/pull/4157

Implementation Plan:

- Removing QgsWFSProjectParser: mi-april 2017
- Removing QgsWMSProjectParser: end of June 2017

Proposal Link: https://github.com/qgis/qgis3.0_api/issues/57

5 IMPROVE DEEP RELATIONS WITH POSGRES EDITING

Proposer: Régis Haubourg

Amount: €6000

Details: QGIS has reached a mature level and offers now a very good framework to create professional applications. One of the main reasons is that QGIS is a very strong client for spatial databases, and in particular with PostgreSQL and postGIS for which it was initially created.

Since version 2.14, QGIS offers the not-so-well-known ability to handle transaction groups, which means it can instantly evaluate triggers on database side, and refresh all layers in the same transaction. This is a big win for usability, but some drawbacks glitches remain, such as the lack of the undo/redo edit buffer, a very raw way of saving (ie quitting edition session) or having the legend cluttered by so many edit symbols (a pen symbol). Current proposal is to go a step beyond to make QGIS even better for PostgreSQL by achieving the following targets:

1 Restore an undo /redo feature by taking advantage of PostgreSQL. If possible we will try to take advantage of PostgreSQL named Savepoints.

2 Allow to have some layers not switching to edit mode in QGIS, even if they belong to the same connection. These layers will still benefit from the instant refresh, but won't clutter the legend with the edit pen symbol everywhere, nor risk to load QGIS snapping cache for nothing. A UI for those settings could be an evolution of the current "identify layer" list in the project properties.

3 We would like to submit a mechanism to allow converting error messages raised by the provider, like a RAISE from postgres, into custom user oriented message. Say for instance, instead of a "provider error - duplicate key for... ", QGIS project could be tuned to display first "You tried to insert a feature using the same identifier as another one".
The error message list and regexp rules would be optionally stored in qgis project or read from a datasource table (for instance when error messages rewrites are shared by other applications). The original error message would be still avaiblable by expanding the details of the messageBar and in the general message log.

4 Cherry on cake point, we wish to have QGIS take advantage of PostgreSQL NOTIFY signals to trigger behavior in QGIS when something changes in the database (see https://www.postgresql.org/docs/9.5/static/sql-notify.html) . A first implementation proposal is to allow a map canvas refresh, but we can imagine really dynamic applications driven by the database events by converting NOTIFY messages into QGIS signals (oh yeah!).

History: In our team, we already use transaction groups for production tools and that is much appreciated. We already use some python logic to catch error message and convert them to more user oriented ones. We frequently develop applications where QGIS is linked with heavy database containing most of the intelligence. Having a really interactive edition process, speaking the same langage as average users, and being able to be triggered from database process will unleashed many possible applications.

Qualifications: Oslandia has three QGIS developers, two of them being core comiters and high skills with Postgres and Postgis (core comiter too). We believe that developing using thick databases is a major strength of QGIS, and we have great fun getting involved in that area of the code ;-)

Implementation Plan: We currently are quite involved in QGIS 3 server refactoring and major changes such as Auxiliary Storage for project or core solutions for label connectors. We also are involved in applications build on top of QGIS for Water management like QWAT or QGEP. Such changes would benefit immediatly to those project. Our target is to provide those improvements with all necessary unit tests for 3.0 release.

Proposal Link: coming soon..

6 PROCESSING ALGORITHM DOCUMENTATION

Proposers: Matteo Ghetta, Alexander Bruy

Amount: €4000

Details: This proposal aims to improve the existing Processing algorithms documentation. With the pull request https://github.com/qgis/QGIS/pull/3911 it is possible to add external links for the documentation (both local and remotes). However the effective use of the pull request is not yet included in Processing.

With this proposal the existing code will be incorporated in Processing, allowing to have a Short Help tab (the existing one on the right of the Processing algorithm window) and a Long Help tab (next to the Log tab).

The Short Help tab will be collapsible in order to have a bigger window for the algorithm parameters, while the Long Help tab will point to the on-line existing documentation of Processing for each algorithm (http://docs.qgis.org/testing/en/docs/user_manual/processing_algs/index.html).

The default link of the on-line documentation will be added in the QGIS Settings (thanks to the pull request already merged) in order to have the standard documentation visible but to let the user the choice to overwrite it and load custom paths.

In addition to the code part, this proposal aims also to document the GDAL/OGR provider and the QGIS core algorithms. Existing documentation will be reviewed and pictures will be added when useful, while for algorithm not yet documented, the help will be written from scratch with description and additional pictures.

Currently there are:

- 49 GDAL/OGR total algorithms, 35 to enhance with pictures, 14 to write from scratch
- 154 QGIS algorithms, 38 to enhance, 116 to write from scratch

This means a total of 73 algorithm to enhance and 130 to write from scratch.

History: The pull request https://github.com/qgis/QGIS/pull/3911 is already merged and it is worth to make it effective to have nice, rich and translatable documentation for the Processing algorithms.

Qualifications: Matteo Ghetta: working since the release 2.0 on the documentation and made several improvements and pull request to both documentation and Processing code.
Alexander Bruy: core developer since 2010, co-maintainer of the QGIS Processing framework.

Implementation Plan: The code and the documentation will be ready for the QGIS 3 release.

Proposal Link:

7 QGIS 3 ENHANCEMENT GEOREFERENCER

Proposer: Mark Johnson

Amount: €5000

Details:

- Resolvement of known problems
- Incorporate a Spatialite based support, replacing .points file
- Use of existing QgsVectorLayer logic for the placement of Gcp-Points

Full details at:
https://github.com/mj10777/QGIS/wiki/QGIS-Enhancement:-Georeferencer which contains a link to a Pdf showing, step by step, the usage of the new functionality.

History: Based on present Georeferencer in QGIS 2.18.
Although there 2-3 problems yet to be resolved, the coding is otherwise completed.
The major task yet to be done is the porting to QGIS 3.

Qualifications: For many years I have worked with the Spatialite project. During this time I have assisted in the testing of new features as well as analyzing reported problems, leading to corrections of the code. Also for the, still under development, RasterLite2 project of Spatialite, extensive work has been done which as lead to a fine tuning of the project during it's development. Also patches have been offered which will extend the import/export functionality to work in a similar way as GDAL, that will be taken into consideration when the development of RasterLite2 resumes.

I maintain the present Android version of 'libjsqlite', which supports the present-day version of Spatialite/RasterLite2 and is used in the Geopapparazzi-Project.
https://github.com/geopaparazzi/libjsqlite-spatialite-android
I also developed the mbtiles and administration portions of Spatialite-geometries and RasterLite2 Databases support for Geopapparazzi.

Implementation Plan: To be completed for the first QGIS 3 release.

Proposal Link: https://github.com/mj10777/QGIS/wiki/QGIS-Enhancement:-Georeferencer

8 QGIS 3D

Proposer: Martin Dobias

Amount: €10000

Details: I would like to propose a project that introduces 3D rendering capabilities in QGIS.

To summarize the planned work, the following features can be expected:

- 2D view of map canvas rendered on the graphics hardware (GPU) allowing smooth zooming and panning of map view
- 3D perspective view of the map
- generation of 3D terrain model from DEM (digital elevation model) layers
- map layers rendered as a texture on top of the 3D terrain
- support for true 3D rendering of vector layers rather than having just flat appearance
- map view widget that is dockable in the main window and synchronized with the main map canvas
- support for picking (identification) of objects in 3D view and X/Y/Z coordinate display
- support for 3D map view in map composer

The overall target is to introduce an extensible framework for 3D map view within QGIS, so that in the future developers can add various 3D rendering techniques for map data, using custom geometries and materials (which may involve writing own vertex/fragment shaders), possibly even allowing multi-pass rendering for advanced effects (e.g. to render shadows cast by buildings with a particular sun position).

3D support in QGIS is not only about adding the extra dimension to the rendering: it is also about making it possible to use graphics hardware for rendering of map in 2D - making map browsing even more pleasant and faster at the same time. Rendering 2D maps with OpenGL also opens the door to various new graphical effects that would be otherwise very expensive to achieve by using just CPU for map rendering.

This proposal does not assume addition of new geometry types like polyhedral surface (with read support for those) into QGIS - the aim of the work is to get 3D rendering engine running and new geometry types may be added at some point later.

State of the art

QGIS features very good 2D rendering capabilities, however its 3D support has been very limited. Prior work on 3D in QGIS includes:

- Globe plugin - a C++ plugin developed by Matthias Kuhn and Sourcepole based on OpenSceneGraph and osgEarth libraries. OpenSceneGraph is a generic toolkit that provides higher-level abstraction on top of OpenGL, making it easier to develop 3D applications than directly using low-level OpenGL interfaces.OsgEarth project then builds on top of OpenSceneGraph and provides a toolkit for working with geographical data: it has a terrain engine that combines elevation layers into a terrain, applies textures from "image" layers and adds feature layers with true 3D objects.The plugin acts as a bridge from QGIS environment and feeds scene data into osgEarth to do the 3D rendering.
- Qgis2threejs plugin - a Python plugin developed by Minoru Akagi. It is able to export QGIS project (with various configuration options) into a HTML page that uses three.js library to render map in 3D within web browser using WebGL.
- Horao - developed by Oslandia. It is a standalone 3D viewer based on OpenSceneGraph that may be controlled by a QGIS plugin to display map from QGIS in 3D environment. It has explicit support for true 3D geometries in PostGIS.

While these projects solve some use cases for 3D rendering of map data, each of them have their own limitations. For example, osgEarth library used by Globe plugin has its own data access and rendering of vector features implementation, duplicating QGIS code and not having parity in their capabilities. Moreover it has been difficult over time to keep the build working on all platforms supported by QGIS. The main limitation of Qgis2threejs plugin is the fact that the 3D view is exported to web browser, so the user cannot use benefits of having 3D view tightly integrated with the rest of QGIS. The fact that Horao has a standalone viewer application results in similar limitations as when using Qgis2threejs (although it has some degree of integration with QGIS application).

Proposed approach

Now that QGIS 3.0 is based on Qt5, we can use some of the great new functionality added recent releases of Qt5. In version 5.5, a new framework for working 3D graphics has been introduced and every major Qt5 release since has been adding more functionality, improving performance, compatibility and stability. The 3D support nicely integrates with the rest of the Qt framework, providing a familiar API and at the same time staying very generic and highly efficient.

The 3D framework provides high level abstractions just like other libraries (e.g. OpenSceneGraph, three.js). 3D scene is built with nodes (called entities) with various components (e.g. transformation, mesh, material).

The idea is to build 3D support in QGIS on top of the Qt 3D framework. From my initial tests of the framework this looks feasible and it will allow us to stick with Qt APIs without requiring extra dependencies.

The work can be divided into the following chunks of work:

1. Rendering engine core: develop a framework that will do rendering of the map scene in 3D. The engine will have the responsibility of processing raster layers with elevation into a mesh geometry and texturing the mesh with map images rendered by the existing QGIS 2D rendering engine. The engine will support levels of detail (LOD) and tiling in order to be able to display high-resolution data in real time without having to load all the data into memory at the time of scene creation (which may be prohibitively expensive with more complex layers). 3D scene will be dynamically updated as user browses the map, keeping the amount of rendered triangles low while appropriate quality of the terrain for given zoom level.

All of the processing needs to be done in the background, so the user may freely browse the map and the scene will be continuously updated with data (changing between higher/lower detail when zooming, loading more data when moving map).

2. Handling of user input: controller for camera that will make the camera fly on top of the map. Support for picking will be added to allow identification of objects in the map and display of coordinates at the mouse position.

3. Integration with QGIS environment: dockable 3D map widget for the main window, synchronization with 2D map canvas, support for printing of 3D views in map compositions.

4. Advanced 3D rendering techniques: interface that will allow adding new methods for data visualization in 3D and exploration of methods for rendering. By default map layers will be rendered into map image with the existing 2D map renderer - this interface will allow map layers to instead have 3D renderer associated which will provide entities with custom meshes and materials. As a result we will be able to achieve true 3D appearance of objects (e.g. point clouds, trees as 3D models, tesselation of polygons, buildings with extruded geometry and custom texture). Implementation of the advanced techniques is a task with nearly unlimited scope, so the idea is to develop a suitable interface and as the time will allow, implement some techniques.

History: For this proposal I have studied various sources:

- looked into existing 3D viewer projects related to QGIS
- explored Qt 3D framework
- researched some academic papers regarding terrain generation and vector data display

As a proof of concept, I have created a simple prototype in C++ using Qt 3D framework. It displays aerial imagery on top of a terrain model created from a raster layer (DEM) and allows simple camera control. The code is available here: https://github.com/wonder-sk/qgis3d

Qualifications: I have been a core QGIS developer for more than 10 years and I have a very good knowledge of QGIS codebase, especially the existing 2D map rendering pipeline.

Previously when working at the university, on a project for stereo matching (creation of point cloud out of a pair of images) I worked on visualization of 3D data using OpenGL.

Implementation Plan: The plan is to work on the project between May and July 2017. As of now, the plan for QGIS releases (according to the mail from Paolo) is that QGIS 3.0 will have feature freeze in July 2017 and final release in September 2017. If nothing changes in the QGIS release schedule in meanwhile, the 3D support could be integrated into QGIS master branch before the feature freeze and thus released in QGIS 3.0.

If the project would be accepted, the first step will be to develop a prototype of the 3D rendering engine, then prepare a more detailed architecture proposal as a QEP and continue the implementation once the QEP gets accepted by the community. The work progress should be available on a branch in GitHub for anyone interested.

Proposal Link: https://github.com/wonder-sk/qgis3d

9 ADD CONSISTENCY TO UI CONTROLS

Proposer: Nyall Dawson

Amount: €1800

Details: Across the QGIS UI, numerous inconsistencies exist in the way different properties like opacity and rotation are exposed to users. These inconsistencies make QGIS harder to use, as behavior from one dialog differs to the behavior in another dialog. Some examples of this include:

- Rotation of labels is done in the opposite direction in labeling to symbology. Accordingly, an equal rotation value will result in different rotation between labels and symbols.
- Scales are inconsistently presented, with use of both the scale numerator and denominator in different dialogs. "Minimum" and "Maximum" scales also vary between dialogs, with some dialogs using "minimum" scale as the largest scale and some using "minimum" as the smallest scale. The labeling scale based visibility controls are the biggest offenders here.
- Controls vary between specifying "opacity", "transparency" and "alpha". While these all have similar results, users must adopt values to map "opacity" to "transparency" in different dialogs. This is further compounded by different ranges used for each (eg 0-100%, or 0-255).

Due to the usual API freeze, it has not been possible to fix these discrepancies. The current API break introduced with version 3.0 allow a window for addressing these issues and standardizing behavior and API.

Despite the benefits in providing a consistent UI, the work involved in standardizing is fiddly (careful attention must be paid to not breaking existing projects) and repetitive, and unlikely to be undertaken by developers on a volunteer basis. Furthermore it is highly unlikely that a commercial organisation could justify sponsoring UI standardisation efforts. Without grant funding it is unlikely that these issues will be addressed during the 3.0 development cycle, and the inconsistencies would remain for the lifetime of QGIS 3.x.

In this proposal I will:

- convert all "transparency" controls to "opacity" controls, and consult with the community to determine the ideal value range presented (0-100% or 0-255) before making all opacity controls use the same range.
- Ensure that rotation always operates in the same direction.
- Fix the labeling scale ranges to use the same scale range definitions as layer visibility

- As much as possible, automatically upgrade existing projects so that they open in QGIS 3.0 without any loss of transparency/rotation/scale settings
- (As much as possible without large refactoring), adapt the PyQGIS API so for consistent naming and use of opacity/rotation and scale setting/getting methods. Making the API consistent makes scripting QGIS and writing plugins easier.

This proposal relates to the issues described at:

- https://github.com/qgis/qgis3_UIX_discussion/issues/30
- https://github.com/qgis/qgis3.0_api/issues/18
- https://github.com/qgis/qgis3.0_api/issues/19

History: No work has currently been undertaken in this regard.

Qualifications: I am currently one of the most active QGIS developers, with a long history of quality contributions to the project. I'm passionate about seeing QGIS 3.0 address these kinds of long standing UI issues which detract from QGIS' otherwise professional image and ease of use.

Implementation Plan: The work will be undertaken prior to the QGIS 3.0 feature freeze period.

Proposal Link:

10 MULTILINGUAL QGIS PROJECTS

Proposer: Matthias Kuhn

Amount: €7000

Details: QGIS Projects contain much text information (Legend, layer names, field aliases...). This information is saved inside the QGIS project file in one language. To create a translated project, one needs to create a separate project file (and keep it up to date with the first project file) or resort to some sort of XML parsing or tinker with the API. This grant proposal provides a way to implement a streamlined approach to translate QGIS project files using tools like transifex or QtLinguist that have proven to facilitate the translation process.

This proposal covers the generic system to create translation source files (.ts) based on existing project files. It also covers the loading of translated files (.qm) as well as implementing this for the most important translatable parts of a QGIS project file.

Details can be found in the following QEP

https://github.com/qgis/QGIS-Enhancement-Proposals/issues/90

History: Technical outline in a QEP

Qualifications: We have implemented plenty of features and participated in QGIS bugfixing programmes several times. Just explore https://github.com/qgis/QGIS/

Implementation Plan: This will be implemented for QGIS 3.0.

Proposal Link: https://github.com/qgis/QGIS-Enhancement-Proposals/issues/90

11 SHARING QGIS PROJECTS AND RESOURCES WITHIN GEOPACKAGES

Proposer: Joana Simoes

Amount: €7200

Details: The OGC Geopackage format [1] was designed with the goal of being interoperable across a range of environments, in particular mobile devices operating in offline and near offline scenarios. Technically, it is an SQLite database, which stores spatial data and allows extensions to encode other types of information.

As part of the QGIS gpkg plugin, we are suggesting two extensions to this format: the qgis[2] and owc context[3] extensions; these extensions address two important requirements of GIS users: a) sharing QGIS projects, including style and resources, and b) decoding data, style, and context information from an interoperable format, in order to support migration from other Desktop, or even server side GIS.

In both cases, the user is able to recreate a project, including style information, from a geopackage; while in the former, a QGIS project file is stored in a table within the database, in the latter, style and context information are encoded in different tables, using a standards based approach (OGC:SLD, OGC:OWC).

Currently, the plugin already supports decoding a geopackage in any of the extensions, and our first activity proposal is to extend it, in order to support also the OGC encoding (currently, only the QGIS extension is supported).

We also propose to integrate this functionality with another QGIS plugin: the QGIS Resource Sharing [4]. This tool allows users to share resources (symbols, svgs, images, or processing scripts). These resources can be stored in local, or remote repositories, collections, which follow a certain template. We believe the geopackage is, by definition, the ultimate support for sharing geographic information, enabling to package data and associated items in a portable database. The second activity of this proposal is to extend the Resource Sharing plugin, in order to support the geopackage extensions that we created, for sharing projects among QGIS users. In order to do this, we would add another resource handler, geopackage_handler, which would extend the base handler class. This handler would use much of the code we already created for loading projects from geopackage, but the advantage of having it in this plugin, is that we can enhance the shareability of projects, by supporting the concept of repositories. Our final goal is

to enable users to have repositories of projects (in addition to styles, scripts, and symbols), which can easily be shared across organizations and even within the overall QGIS community.

References:
[1] http://www.opengeospatial.org/standards/geopackage
[2] https://github.com/GeoCat/qgpkg/blob/ows-spec/qgis_geopackage_extension.md
[3] https://github.com/GeoCat/qgpkg/blob/ows-spec/ows_geopackage_extension.md
[4] http://www.akbargumbira.com/qgis_resources_sharing/

History: The main motivation to start developing this plugin was to illustrate the usefulness of the geopackage format for sharing information, and the powerfulness of its extension model, to support specific use cases. We also aimed to demonstrate, how a standards-based approach can be key to connect “silos” of information in the GIS world. When we started researching about this topic, we stumbled into the pka/ qgpkg plugin[1], from Pirmin Kalberer, which implemented a similar concept, except that it stores the project information in a format which is specific to QGIS (the project file). The geocat/qgpkg plugin[2] started as a fork of the pka/gpgk plugin, supporting a different geopackage extension. Both plugins, as well as the OWS GeoPackage Extensions, are available on the project’s GitHub repository, with an MIT license. Our current vision is that both plugins will be merged, in order to have a geopackage encoder/decoder, supporting both extensions.

The QGIS Resource sharing plugin in which we would like to integrate the geopackage functionality, was initially implemented as Google Summer of Code 2016 project for QGIS under OSGeo organization by Akbar Gumbira (student), Alessandro Pasotti (mentor), Anita Graser (mentor), and the help of Richard Duivenvoorde, Tim Sutton, and QGIS community. The code is also available on GitHub [3].

References:
[1] https://github.com/pka/qgpkg
[2] https://github.com/GeoCat/qgpkg
[3] https://github.com/akbargumbira/qgis_resources_sharing

Qualifications: I have written the GeoCat version of the QGIS gpkg plugin: https://github.com/GeoCat/qgpkg

I have more than 10 years of professional experience with C++, designing, and leading the development complex applications, mostly related to GIS and databases. I have more than eight years of experience in developing cross-platform applications with the Qt framework.
Github repo: https://github.com/doublebyte1

I have been an earlier adopter of QGIS, have been to two HackFests and have published one plugin, in the official QGIS repositories. During my period at Food and Agriculture Organization (FAO-UN), I was an envangelist of QGIS, and adopted it as primer desktop GIS, when creating the FAO training materials.

Implementation Plan: Depending on funding availability, our intention is to kick-start the development in the QGIS Hackfest (Essen, 29 April - May 1), in order to have the opportunity to discuss our ideas with the QGIS developers, and in particular with the developers of the "QGIS Resource Sharing" plugin, if present. After that, we planned around two weeks of work to complete the two key activities:

- Implement the OWS Extension geopackage decoder.
- Add support to the the geopackage project extensions, in the collections of the resource sharing plugin.

This includes a basic level of testing and documentation. We are targeting version 3.0 of QGIS with this plugin, and we are aiming to have it ready before the scheduled release date.

Proposal Link: https://www.geocat.net/announcing-the-extended-geopackage-qgis-plugin/

12 ENHANCE INTEROPERABILITY BETWEEN QGIS AND X3D

Proposer: Andreas Plesch

Amount: €900

Details: The geospatial component of the X3D international standard (http://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/geodata.html) provides stable and well defined format for visualizing and interacting with geospatial data in 3D (Plesch & McCann, 2015, http://dl.acm.org/citation.cfm?id=2775315). X3D is platform and encoding independent and X3D viewers are available on a number of platforms including the web (http://x3dom.org, http://titania.create3000.de/cobweb/). It has extensive and formalized metadata capability and is often considered an archival quality format. However, currently most geospatial X3D content is handcrafted or generated in custom solutions (eg. STOQS, http://www.mbari.org/products/data-repository/upper-ocean-data/spatial-temporal-oceanographic-query-system-stoqs-data/). There is a lack of tools to assist with geospatial X3D scene development. On the other hand QGIS just started to provide 3D functionality and would benefit from broader support of 3D visualization, especially on the web platform. The popular Qgis2Threejs plugin is a great example of how QGIS can widen its appeal in this way. In this proposal I suggest to take advantage the open and extensible nature of QGIS, and in particular of the Processing framework, to provide X3D export functionality of the main geospatial data types. As a functional prove of concept I developed a series of Processing scripts and models which convert a digital elevation model in a raster layer to geospatial X3D and use x3dom.js to interactively display the result in 3d on a web page (https://github.com/andreasplesch/QGIS-X3D-Processing). Further steps would include support of color contouring, draping of registered imagery and extruded vector layers. I believe such functionality provided in a standardized and reusable format could enhance QGIS greatly.

History: The proposal leverages the Processing framework and x3dom.js to do the heavy lifting of geo-data processing and 3d visualization. It also takes advantage of the well defined X3D international ISO standard to enable interoperability and reuse of generated content. Because of this existing infrastructure I could relatively quickly develop a functional proof of concept to demonstrate the feasibility and the likelihood of a successful outcome of this proposal.

Qualifications: I am the primary developer of the geospatial component support provided in x3dom.js and advised in the development of the geospatial component in cobweb.js. Many years ago I developed perhaps the first plugin for hill shading in QGIS using numpy (which I believe got lost although I may resurrect it now that numpy comes with QGIS). As functional proof of concept I developed a series of Processing scripts and models

(https://github.com/andreasplesch/QGIS-X3D-Processing) which convert a DEM raster layer to X3D for interactive display on a web page.

Implementation Plan: The implementation schedule will follow the TODO list posted https://github.com/andreasplesch/QGIS-X3D-Processing#todo .
There are three main components:

- DEM,
- Imagery and
- Vector Extrusion

Anticipated milestones are:

- Early Summer 2017: DEM complete with documentation and tutorial
- The following milestones may be better addressed in a followup proposal:
    - Fall 2017: Imagery draping complete with documentation and tutorial
    - Spring 2018: Vector Extrusion complete with documentation and tutorial

Proposal Link: The language provided above is the main body of the proposal. Updates on https://github.com/andreasplesch/QGIS-X3D-Processing

13 UPDATE MACOS CMAKE BUNDLING SCRIPTS

Proposer: Larry Shaffer

Amount: €1800

Details: Currently, the macOS bundling routines (to create a self-contained QGIS.app application) in CMake scripts where created by William Kyngsburye many years ago. Since then, CMake has added many features for bundling, e.g. BundleUtilities (https://cmake.org/cmake/help/v3.0/module/BundleUtilities.html), that handle similar functionality to what has been arduously maintained in the CMake scripts. While the current setup does function, it is quite antiquated and adding any new QGIS dependencies to be bundled is an error-prone ordeal. I propose to fully update the bundling routines to leverage modern CMake capabilities, since building on macOS usually uses the latest CMake versions. Once completed, anyone with appropriate dependencies should be able to produce a production-ready QGIS.app bundle, including the QGIS project itself.

History: I will first build upon the existing work to ensure there is a minimal bundled QGIS.app, then completely refactor the same functionality using a modern CMake code workflow.

Qualifications: I have extensive knowledge in CMake and frequently utilize it in my work for my employer, Boundless Spatial. I have already completed a fully bundled QGIS.app distribution by my employer (similar to the first phase of the proposed work here), though Boundless now uses a different installation approach. I have been working for years on the OSGeo4Mac project in anticipation of producing better CMake bundling routines, to ensure the QGIS project can independently produce its own macOS distributions.

Implementation Plan: Basic work will follow these steps:

- Append minimal bundling to existing CMake setup, so there is a least a functioning bundling routine, regardless of whether the proposed work is accomplished in time for the major next release.
- Ensure the QGIS.app bundle is code-signed
- Create a new methodology, based upon CMake's BundlesUtilities, *in-line* next to the existing CMake routines, so both can be used, until there is a valid replacement.
- Focus on minimal bundling, then add GRASS
- Continue extending bundling routines to include major Processing providers, e.g. OTB, Saga, TauDEM, etc.
- Ensure new method's QGIS.app is properly code-signed

- Enable bundling on Travis CI infrastructure, via Travis' cron jobs, thereby adding the capability for the QGIS project to produce fully bundled nightlies of macOS builds.
- Once new method represents a full replacement, old method will be removed, not just deprecated

Intended completion is in time for QGIS 3.0 release and packaging efforts.

Since the new method does not affect any existing code, as soon as useable functionality is achieved, it will be merged directly into master, then further code committed as work progresses.

Proposal Link: None at this time. Should I consider a QEP? Not many developers beyond the few existing packagers and experimenters would be involved. I would prefer to write a blog post after the work is completed, though post to the QGIS dev mailing list the intention to do the work, if this proposal is granted.