Title: KStars: Additional Catalogs and Community Integration

Student: Victor Carbune

Abstract: Collaboration features are of great importance for amateur astronomers. The project I am proposing extends the current implemented support for the OAL format already found in KStars, ensuring that it is fully compatible. Support for additional catalogs in KStars is also considered.
Implementing real time collaboration features (using QWaveClient) using OAL XML format will make KStars a lot different. Astronomy specific collaboration features will be available through a carefully designed GUI.

Name: Victor Cărbune

Email Address: victor.carbune@gmail.com

Freenode IRC Nick: carbonix

IM Service and Username: irc.freenode.org

Location (City, Country and/or Time Zone): Bucharest, Romania (GMT + 2)

Proposal Title: KStars: Additional Catalogs and Community Integration

Motivation for Proposal / Goal:

The features I'm willing to implement in KStars may be of high demand in any astronomy software. Starting from the important base features (extra catalog support and OAL full compatibility), the goal is to help KStars get more popular and demanded in the world of amateur astronomers, by making it different from the others. This project will bring specialized collaborative features (handling directly astronomical objects and events and ensuring support for the kind of interaction that amateur astronomers are willing to have with each other while planning their observations or directly from the field) straight in KStars!

Content:

Implementation Details:

I. Catalog support for KStars: This will be a very straight forward implementation and also involves use of the code written in GSoC 2008 (star catalog loading and binary file format of KStars).

Documentation, done for this step involves the following:

I.a. How KStars stores catalogs in its own format - I have carefully read README.stars (skycomponents) in conjunction with README.binfileformat and README.indexfileformat (data). I understood when and how the stars are loaded in KStars (named/unnamed, static/dynamic, StarComponent and DeepStarComponent classes, etc).

Other additionally documentation read: how the tools in /data work and how the existing catalogs were created, how KStars uses indexing (partially) to speed up drawing and loading and what exactly the Hierarchical Triangular Mesh presumes for star drawing.

I.b. I am an experienced C++ developer, having no problems on working with binary data (detailed information about my experience is provided below).

The first step to do, is to define the method of efficiently storing and working with data. As discussed on the mailing list [1] I'm planning to implement this differently for deep sky objects and stars:

- information regarding the **deep sky object catalogues** will be stored in a database using **SQLite in C++** [2]. This will use the power of SQL queries and solve most of the problems encountered, specially multiple name designations for objects (instantly obtaining all the names for an object, the only match / queries are only made when adding a new dso catalog). [Entity Relationship Diagram]
- **star catalogues** will remain stored in binary files (maybe in some different structure - TBD), but the major improvement will consist of using the **STXXL** Boost Library [3], similar and compatible with the classic STL but optimised for extra large data sets.

What will be the steps of adding a new catalog?

In order to easily add another catalog to KStars (a massive catalog, not a custom created one) I suggest developing an abstract class with standard reading and data parsing methods (probably using stxxl, for star catalogs). After this is made, the following steps should be done (by a developer) in order to succesfully load a new catalog in KStars:

- extend the abstract class and implement a particular reading class for the catalog he is willing to add (actually, specifying the header should be enough)
- call the parsing method with the new defined class and the file where the catalog is stored. This will handle all the operations needed to insert the catalog (either in the database - for dso, either as a binary file - for stars catalogue)

We will use stxxl mostly because of its explicit support for asynchronous read / write. This means that even if we continue with the same catalogue organization that KStars has, we will encounter some performance enhancement.

Supporting multiple star designations (actually matching catalog numbers) could be made possible sorting the catalog information using declination, right ascension and magnitude of the star (in this order). We might do this directly with the stxxl sort algorithm [4]. Through this we may directly use the stxxl find algorithm [4] (using the vector container) and thus, this data can also be used to paint the sky (as we need the stars at the coordinates where the trixel is, and we need the list to a specific magnitude).

Matching the stars designations that refer to the same object might be done (as a supplementary method) on the fly, while initially painting the sky. Supossing that regions of sky are painting one by one, catalogs could be simultaneously read with information for that region (and maybe store it in memory, depending on the region size) and information will be further processed and matched. However, this can also be done using external memory, with stxxl.

STXXL's performance should not be a problem for star catalogs, as it has been optimised to work with terabytes of data.

[1] http://lists.kde.org/?l=kstars-devel&m=127031934811788&w=2

[2] http://sqlite.org/

[3] http://stxxl.sourceforge.net/

[4] http://algo2.iti.kit.edu/dementiev/stxxl/trunk/examples.html

I.c. I have read about the following catalogs, for which I intend to add support in KStars (these can be downloaded from the sites below). I believe my experience as an amateur astronomer will also be of help, as I will be able to analyse and interpret correctly everything found in the catalogue:

Smithsonian Astrophysical Observatory Star Catalog, Principal Galaxies Catalogue, Morphological Catalogue of Galaxies, PK (Perek and Kohoutek, 1967), now Catalogue Of Galactic Planetary Nebulae (CGPN), Saguaro Astronomy Club Catalogue

Other catalogues will also be considered for testing purposes as decided at the completion moment.

II. Community Integration:

The features added through the Observation Planner last GSoC represent the starting point for succesfully support the Open Astronomy Log (OAL) 2.x schema. I have further investigated the code found in the comast/ directory and I believe that I got familiar and confident with what happens there. The Observation Planner will represent the main starting point for the collaborativity feature to be introduced in KStars. Basically, astronomers will be able to plan their observation *together* !

II.a. Intensive research and xml validation based on the OAL 2.x schema will be done in the first phase and check all the things that are not currently compliant with the schema. I believe this will not represent a very hard challenge, as adjustments can be made easily on the resulting XML.

II.b. The GUI that will be implemented is divided in two main parts:

- **the collaborative view** - this will actually be the current Observation Planner widget to which collaborative support will be added using the QWaveClient API and code. Another chat and friend widget might be added (as can it actually be seen in the qwaveclient) and astronomers will be able to plan their observations together. Technically, this relies on generating partial OAL XMLs while using the Observation Planner and sharing it with others
- **the retrieve / submission of OAL specific logs view** - this will be a new designed graphical user interface, specially used for handling observation logs retrieved or submitted to the internet. Filters will be able to show the user only the requested

information (particularly, observations regarding a specific list of objects, constelations, locations or times - these currently come into my mind, but other filtering criteria can be added). Also, URL address can be specified as where to receive or submit logs.

A first step means actually prototyping the retrieve / submit / filter GUI and further enhancing it with the help and feedback of the community. The log management could actually make use of the fact that SQLite support will be available in KStars. Many of the queries describe above could be done using SQL queries which will actually do the filtering, if many logs are available at the client side.

If in the logs are received objects that do not currently reside in the catalogues of KStars a search option might be implemented - e.g. sending the user to a specific site, with automatic search / query parameters for the object. The other possible part would be that the user does not have the current needed catalog installed, and from a query to a KStars web server a catalog plugin / download will be prompted to the user.

Other collaborativity based features might include real time reports about:

- who else is observing at the same site or nearby (by geographical coordinates)
- who else is observing the current object and what he has reported about it
- who else is interested in doing observations in the same night planned as the current user of KStars

III.c. I'm very enthusiast implementing everything above in a realtime manner, by integrating the QWaveClient [1] API support in KStars. Currently I'm thinking on doing this in the third phase of the project.

What QWaveClient offers is the actual the part of synchronizing the same XML to multiple users. This means that if someone does a change at the XML, QWaveClient generates the differences for the XML and sends them to all the clients. When received the differences are operationally transformed and the information is updated and synchronized. Besides this, QWaveClient already offers wave specific features [2] such as a chat and friend widget.

Using the QWaveClient, actually this extends the OAL implementation to a collaborative experience for users with just some extra lines of code. Besides this, the GUI specific features are also important, as they can enhance the user experience (e.g. adding special comment fields depending on the type of the object observed). After the OAL support is refined, a wave should be created for a planning instance and other users should be added to the wave, and all the other part should be handled by the QWaveClient mechanisms.

I have inspected the current status of the QWaveClient and I'm also trying to get in touch with the developer. The main features seem very functional, but indeed, one of the reasons I'm planning this in the third phase is in the hope that the project will get in even a more stable state. I've sent an email to the main developer of the API, just to see it's oppinion regarding this use of it's client (if I will get a reply, I will post relevant information in the proposal or as a comment).

Astronomers using their laptops at astronomy observations could simultaneously see who else is observing what, exchange information or
simply get enthusiast by seeing that so many other are out there observing and get out fast in the yard with the telescope.

I think this would be a unique feature, the first of it's kind in an astronomy software, and since there already exists a wave
implementation for Qt I believe it is a really feasible idea for the summer, attaching it to the full OAL support.

[1] http://code.google.com/p/qwaveclient/

[2] http://wave.google.com

Tentative Timeline:

The project represents for me a full time commitment, not intending to do anything else during the summer (besides school activities until 3rd July, clearly detailed in their section below).

I have been a previous GSoC Student last year and I have worked on a timeline similar to the one below. I'm highly interested on working efficiently on the project, in order to avoid as much as possible inaccuracies in the timeline. However, I know that changes may be done after replanning with the mentor or if else requested during the coding period. I'm also trying to design the timeline by using the community bonding period to, in order to relax some periods, such as the exam period at my university. The last 1.5 months means double working time.

Now - April, 26 - **Continuous documentation on the KStars Code.** Prototyping and detailing (if requested): abstract class implementation for the catalog support, the database entity relationship diagram (ERD) and the graphical user interface in Qt for the community integration part. Other catalogs may be taken in view.

April, 26 - May, 1 - Working with SQLite and implementing the database designed in the prototyped ERD. Implementing the particular methods in the abstract class that can be used to add a new Deep Sky Catalog to the database.

1 - 8 May - Adding support in KStars for some new Deep Sky Catalogs and painting them in the current mode. This will help me get an insight for the harder part, efficiently painting stars.

8 - 15 May - Reviewing the DSO Catalog support and further documenting about the STXXL library that we will use to work with large binary files used for stars catalogs.

15 - 22 May - Deciding the final structure of the binary files, holding stars catalogs (either the current one used in KStars, either with some changes, if needed). Implementing the

abstract class methods needed for parsing a stars catalog and write them in the required binary format. Trying to import some star catalogs using this format.

22 - 29 May - Rewriting / Changing the StarComponent and DeepStarComponent classes, to make use of the asynchronous read / write method avaiable in the STXXL. After this is done, adding the new stars catalog should not be a difficult task.

29 - 15 June - At this step we should have new catalogs integrated with KStars. This 2 week period will be used to review the written code, enhancements and other code related documentation. This will also be a period when final exams are held at my university, so I will be working a bit less, but I will however dedicate sufficient time so that any other request regarding the features implemented above will be finished.

15 - 26 June - Project revision and milestone discussions about the further development of the project. Intensively reviewing the code in comast/ and adjusting the changes in order to make the XML fully compliant with OAL 2.x Schema.

26 June - 3 July - Further tests will be conducted with software that have been announced to fully support OAL in order to enhance that the validation is correct. (+ some other exams). Prototyping and starting to implement the GUI for the OAL import/export using Qt, and filter options. (school ends)

3 - 12 July - Implementing the Qt GUI that supports importing and exporting of the astronomy logs received from others. Basic functionality will be given, at the beginning, and right after the model will be enhanced with filter capabilities on object specific parts. This period will also be used to make sure that by midterm the catalog support is fully delivered and (hopefully) half of the community integration is done.

12 July - 19 July - Integrating full real time collaboration features using QWaveClient based on the OAL XML. Working on the Observation Planner to add partial XML building as the user enters details. Sending / receiving the XML using the QWaveClient and a server supported by QWaveClient.

19 July - 24 July - Correctly interpreting the XML received (update the observation planner view) and maybe adding chat / friend widget.

24 July - 31 July - Releasing an alpha version of this collaborative version of KStars, handing it to astronomy communities related to the KStars developers and be opened to feedback regarding user the experience.

31 July - 7 August - Spending time on enhancing the real time collaboration features based on the feedback received from the community. Documentation, code reviews, tests and other activities related to these will be done.

7 August - 16 August - Enhancing the documentation and other relevant information in order to succesfully complete the project. Further work will be done, but after 16 outside GSoC ;)

Do you have other obligations from late May to early August (school, work, vacation, etc.)?

My only obligations are school related, until 3rd of July. It is worth mentioning that I last year I have participated in GSoC under the Joomla! Organization (project page) and these have not interfered heavily with the project. I have also been working part time (project based, similar to GSoC) right after beginning of school, starting in October and ending a month ago and successfully completed the tasks.

As requested, I am detailing these: mandatory laboratory activities (10 hours / week), homework assignments - (2-3 coding projects every 2 weeks) and exams (5 exams in a month, from June to July - 2 hard, 3 easy - there is plenty of extra time between them, though).

About Me (let us know who you are!):

I am second year student at "Politehnica" University of Bucharest, Romania. I'm passionate about technologies in general, I have worked on a variety of projects from low level programming to higher level. I am currenty in the search of a C++ project under which I can further develop my programming skills and use the knowledge in other fields too. KStars is a project at which I would really enjoy working, as it's one of the really small opportunities where I can combine astronomy with my technical skills.

I am using KDE as my standard desktop environment for a couple of (good and long) years. Also, I would be glad to become a member of your great community! KStars represents a personal interest too, since I'm using it sometimes, while planning observation nights. I'm passionate about science and technology and I'm detailing my relevant experience below.

Technical abilities: I consider myself as an experienced C/C++ developer for more than 2 years. Even though I have not worked on very big projects, I have worked on a simple image processor in C++ and lots of other interesting school assigments (algorithms, data structures). I have previously worked with Qt by creating a GUI for a university project last year.

I also have a strong background on database design and implementation - in my 12th grade I have participated at an international database modeling organized by Oracle and the Entity Relationship Diagram we designed won the first place. We got invited to the Redwood Shores to present our project there. I am willing to value this experience during this project, while working with the deep sky catalogs and SQLite.

Astronomy: I'm passionate about astronomy since 9th grade. I have participated at various local and national contests which helped me to further develop my knowledge in this field. I believe being familiar with astronomy helps on being creative when implementing features to KStars :-)! I definitely hope that I will be able to value all my astronomy related skills and knowledge. I'm also passionate about mathematics and even though, at a first glance, math might seem not so useful for this project, it's better to be prepared :).

Organizational abilities: I am very organised when it comes to tasks I have to do. I have been working during my last year only on a project basis, with clear deadlines and weekly reports. Even if sometimes I did not exactly met the deadlines, I have discussed this in advance with the people I've worked. My GSoC experience last year also helped me on learning how to interact with communities, solve issues on time and efficiently plan each milestone.

I'm also thinking on keeping a blog strictly related to my GSoC project progress. Last year, I have enjoyed reporting every 1-2 weeks my development status and getting feedback from the users community on the features I've implemented.

Other files relevant to the application (if any) will be posted here:
http://swarm.cs.pub.ro/~victor/kde/

As last year, I am having in view a full commitment of 35 hours per week and once enrolled in a job, I'm dedicating most of my energy to keep everything on the right track.