Improving symbols dialog and optimizing performance

Name: Osama Ahmad
E-Mail: osamaahmadcv@gmail.com
(ssssss1233345@gmail.com at GitLab)
IRC Nickname: osama.ahmad
Telephone: +201098197483

The idea

To improve symbols dialog. Improvements are:

- The symbols are loaded from the dialog itself, not form a background process. making it asynchronous will make the start time (if the dialog is opened before the last exit) or the runtime response (if the dialog is opened for the first time while the program is running) faster.
  More details:
  - In case the symbols dialog is present at the startup (which means the symbols will be loaded at the startup), the symbols will run in a background process and let the program open even if the symbols are not loaded yet. This optimizes the runtime of the program.
  - Also, in case the dialog is not loaded at the startup, and the user decided to open it (thus the symbols are loaded while the program is running), the dialog will show and the program will be responsive even if the symbols are not loaded yet since they'll be loaded in a background process making the program usable even if the symbols are not loaded yet.

  Even more details:

  Q: When will the symbols get loaded? At the startup? After the startup?

  A: It depends. The thing is, if the dialog is open at the startup, it's gonna be loaded at the startup anyways. Now, the program is not interactive until all of the symbols load. The user has to wait for all the symbols to load to be able to use the program. By making it asynchronous, it's gonna still load, but in a different thread. In the dialog, there will be a message saying something like "Loading..." or anything to indicate that the symbols are not loaded yet. But the rest of the program should not be unresponsive, just because the symbols are not loaded. The user should be still able to edit and use other tools even if the symbols are not loaded yet, they shouldn't wait for the symbols to load to be able to use other stuff. The tools have nothing to do with the symbols and should not be dependent on it to load. All of what I said describes when the dialog is loaded at the startup, but the dialog might not even be present at the startup. It can be not present if the user has set the program this way. In this case, the dialog and

the symbols will only load when the user opens them manually. This case also has the same consequences. The user will have to wait until the symbols are loaded. Again, the tools and the entire program won't be responsible until the symbols load. By loading the symbols in a different thread, and letting the current thread serve the users and let them be able to keep using the tools even if the symbols are not loaded yet, the program becomes more responsive and interactive. The thing is, the current thread will only be doing one thing at a time, either interacting with the users and letting them edit the document, or loading symbols and not responding with the users. By creating one more thread, we ensure that the original thread will not be loading symbols ever, thus, making the tools usable all the time. And the loading of the symbols will be a job of a different thread dedicated completely for this process.

- To be able to open, edit and save the symbols from the dialog. If the symbols are loaded from a read-only folder, or if the user intends to save as a new copy, the user can save the edited symbols as new ones. By edit, I mean to edit the SVG itself or edit the tags and other properties of the symbols.

- To be able to load/save symbols from/to a custom folder while the program is running directly from the symbols dialog.

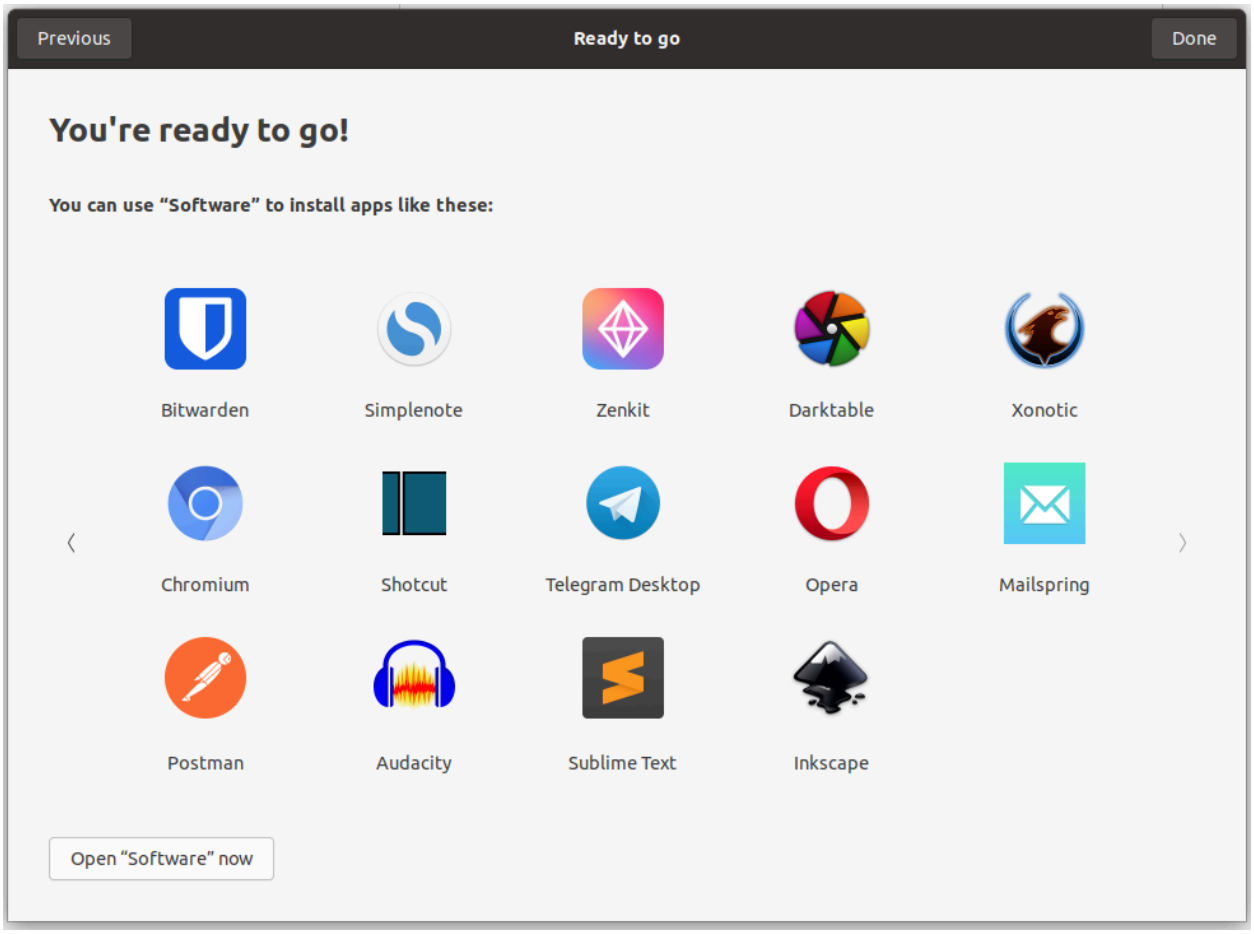
What are your favorite programming tools (editor, etc.)?

Related to C++: CLion, GDB, VIM, CMake

Other: IntelliJ, PyCharm

When did you first hear about Inkscape?

Ubuntu suggests Inkscape as one of the programs that can be downloaded once the OS is installed.



What kind of drawings do you create with Inkscape?

Not too much. Mostly making use of the ability to convert objects to paths and change their shape. Mostly dealing with texts and changing how characters look. Nothing complicated.

Describe your participation in our community (e.g. uploaded drawings, tutorials, bug reports, communication via mailing lists or IRC). Also Describe your contributions to the Inkscape development (e.g. bug fixes, translations, packaging, testing).

Commits on the master branch of Inkscape:

- Refactoring: removing Guide Constraints
- Changed SPStar to derive from SPShape instead of SPPolygon.
- Renamed Inkscape::Algorithms::longest_common_suffix to nearest_common_ancestor.

- Improved Inkscape::Algorithms::nearest_common_ancestor
- Rename getDocumentURI to getDocumentFilename

Open merge requests:

- Improved Verb::get_action to run in O(Log(N)) instead of O(N).
- Moved "Convenience" functions in repr.h into Inkscape::XML::Node as member functions.
- C++fied DrawAnchor (turned it into a class with its own methods).
- Got rid of the duplication in selectorsdialog.cpp and styledialog.cpp
- Simplified ResourceManager. It does only one thing: fix broken links.

What programming projects have you completed?

Most of my time was dedicated to competitive programming (That's why I had to learn C++ in the first place). Still, there are some projects that I've done/doing:

- Implementing various parts of an OS (OS161). The code is unfortunately private since it was asked by the university to not to publish the solutions for assignments online so that no one can cheat. Can provide access if needed. The code is entirely in C. Used bmake to build. Implemented parts include:
    - Synchronization primitives:
        - Lock
        - Mutex
        - Reader-Writer lock
        - Didn't Implement Conditional Variables and Semaphores since they were provided.
    - System Calls:

      fork - execv - waitpid - exit - getpid - open - close - write - read - lseek - dup2 - getcwd
    - File Handles and File Tables
    - Process Tables.
    - Last assignment was to implement virtual memory, TLB, paging, page faults, and caches; but I stopped temporarily because of preoccupations and didn't return since then.
- Working on and responsible for several parts for an ROV for Mate ROV Competition. Responsibilities include:
    - Controlling

        - Dealing with peripherals like joysticks (to control the ROV)
        - Networking (between different devices in the ROV)
        - Managing the motors of the ROV (so that it moves correctly)
    - Computer Vision (not done yet)
        - Controlling the attached cameras to the ROV
        - Processing the input of the multiple cameras to determine the environment and take actions accordingly according to the requirements for the tasks.
        - Lots of algorithms and data structures. Unfortunately, I didn't know how to use git then, couldn't find most of them. Found an implementation for Red-Black Trees.
- Gaussian solver (C++ and QT)
- Sudoku solver (C++ and QT)
- Blockchain project (Java) (not completed yet)
- Space Invaders game (Python and PyGame)
- Simple reddit-like API (Python, Django and Django REST Framework)
- TODO-list website (Python, Django and Django REST Framework)
- Personal Portfolio website (Python and Django)
- Profiles API (Python, Django and Django REST Framework)
- Recipe App API (Python, Django and Django REST Framework)
- Password Generator website (Python and Django)

Describe any work on other open-source projects.

Even though I learned C++, Java, and Python, Git, algorithms, data structures, design patterns, OS... a long time ago, this is my first time contributing to an open-source project. Was using git to only control my own code.

In exactly **two** sentences, why should we pick **YOU**?

Have the required skills. Dedicated and care so much to succeed in this project (since this is my last college year, thus, last chance in GSoC).

List other GSoC projects you are applying to.

Nothing, just this one.

Describe any plans you have for the summer in addition to GSoC (classes, thesis, job, vacation, etc.).

- Graduation Project (as a backend developer using Python, Django and Django REST Framework)
- Responsible for Controlling, Computer Vision, and some electronics for the ROV competition.
- The final exams of the spring semester

Proposed schedule for the coding period (*Tentative*)

- **Community Bonding Period**
- (May 17, 2021 - June 6, 2021)
  - Get fully acquainted to the code base, specifically the components that the symbols dialog is composed of, using, or may use when the features are implemented.
  - Discussing and tailoring the details of the features and finalizing a work plan with the mentor.
  - providing that there's still time available in this period, I will start coding immediately.

- **Code Period**
- (June 7, 2021 - June 20, 2021) (2 Weeks)
  - Goal: to have the loading of the symbols happen independently of the main thread of the program, thus optimize its startup time and the performance in the runtime.

- (June 21, 2021 - July 3, 2021) (~2 Weeks)
  - Goal: To be able to open, edit and save the symbols from the dialog. If the symbols are loaded from a read-only folder, or if the user intends to save as a new copy, the user can save the edited symbols as new ones. By edit, I mean to edit the SVG itself or edit the tags and other properties of the symbols. Due to the final exams, this will be interrupted and will be continued after the exams end.

- The final exams of the spring semester will start on the third of July (**Not official because the semester agenda is not yet posted**). In the period of my exams, I won't be able to actively contribute and work on the project. Nonetheless, I will be active on the IRC participating in discussions about the project. (Supposing exams will last 3 weeks.)

- **Evaluations phase**
- (July 12, 2021 - July 16, 2021)
  - By the time of the evaluation phase, the deliverables would be to have the loading symbols happen asynchronous to the loading of the program (whether it's on startup or while the program is working).
  - To have most of the functionality done for being able to edit the symbols and their properties.

- (July 25, 2021 - Aug 1, 2021) (1 Week)
  - Goal: to finalize the last task. To have the symbols modifiable through the dialog directly. Both editing the actual SVG or editing the properties.

- (Aug 2, 2021 - Aug 16, 2021) (2 Week)
  - To be able to load/save symbols from/to a custom folder while the program is running directly from the symbols dialog.