Student Details:

Name: Eliakin Costa de Almeida
Email: eliakim170@gmail.com
IRC Nick: eliakincosta
Location: Bahia, Brazil
Github Username: eliakincosta

Project Title: Develop a showcase of Krita's new scripting support.

Introduction:

Krita is a professional free and open source painting program. The Krita's tools and features are focused on digital painting, concept art, illustration and texturing.

Some steps of working in software for digital painting can be repetitive and consequently boring, like exporting a painting, configuration of the document's properties, applying filters and a set of other tasks. An approach to solve this problem can be to make it scriptable. Making a software scriptable is expose an API for the users and offer some way to execute your own code.

Python is the Krita's script language and the standard language in the VFX industry, but we are talking about artists that in your majority don't code. My proposal is focused in solve this problem working together with the Krita's community to implement a set of scripts and plugins in python, to attending your needs about this repetitive and boring tasks.

Project Goals:

1. Scripts to execute in the interactive GUI

It's an important part of this work, that in the end, we have a set of scripts to be executed in the Interactive GUI (https://phabricator.kde.org/T4551), that's a task in progress, but that works properly like a simple python editor where you can execute python scripts and debug then as well. These scripts are more simple, something to play with the API without PyQt (https://riverbankcomputing.com/software/pyqt/intro).

2. Python plugins to Krita

Sometimes this code can be more difficult to implement and need to be more organized inside a better structure or in more files to make easier to maintain this of some way. In this cases, we need to implement some python plugins that can be executed with a simple click of the mouse. Implement these plugins also can be a start to more developers that don't code C++, but are interested in contributing to the community. They can see these plugins already implemented.

3. Extend libkis

As a natural consequence of implement new plugins using the API (libkis), we will need to extend and adjust some parts of the API to attending this demand. I'm intending to implement this changes with help of my mentor.

Implementation:

First, I have to talk about the importance of the user's participation in this process. We need to investigate what the community really needs and define a set of priorities for this implementations. I believe that the bond period it will be really important for this part. Now we will approach some examples of the implementation:

1. Scripts to execute in the interactive GUI

How I said previously, we need to execute some repetitive tasks that are simple, but when you repeat many times this can be boring. I will show how to apply a filter on all top layers. When you have many layers this can be awful.

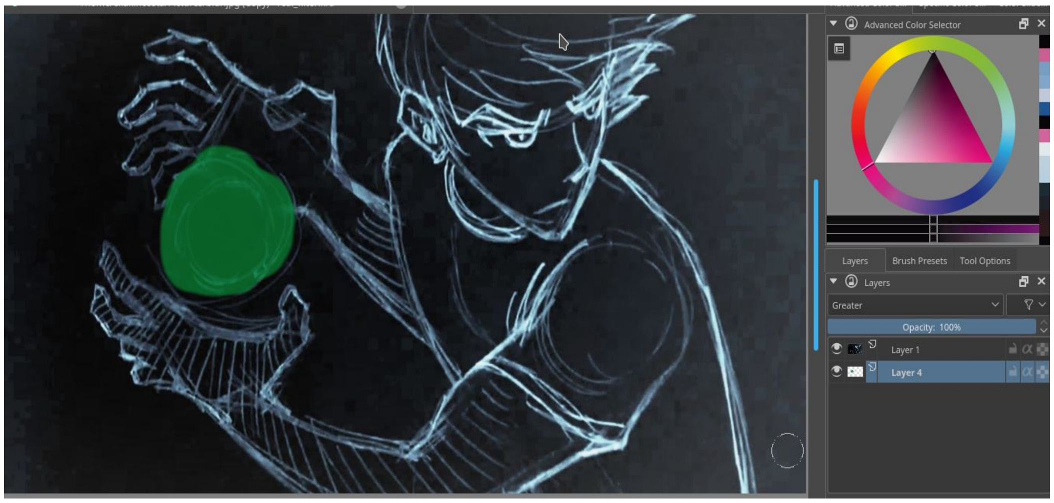


Image1: *It's a document with two layers, that I will apply a filter*

Then with the API that we have, for now, it's possible to apply an *invert* filter with just some lines of code:

```
from krita import *

instance = Krita.instance()
document = instance.activeDocument()
top_level_nodes = document.topLevelNodes()

for child in top_level_nodes:
    filter = Application.filter("invert")
    filter_configuration = filter.configuration()
    filter.setConfiguration(filter_configuration)
    filter.apply(child, 0, 0, document.width(), document.height())
```

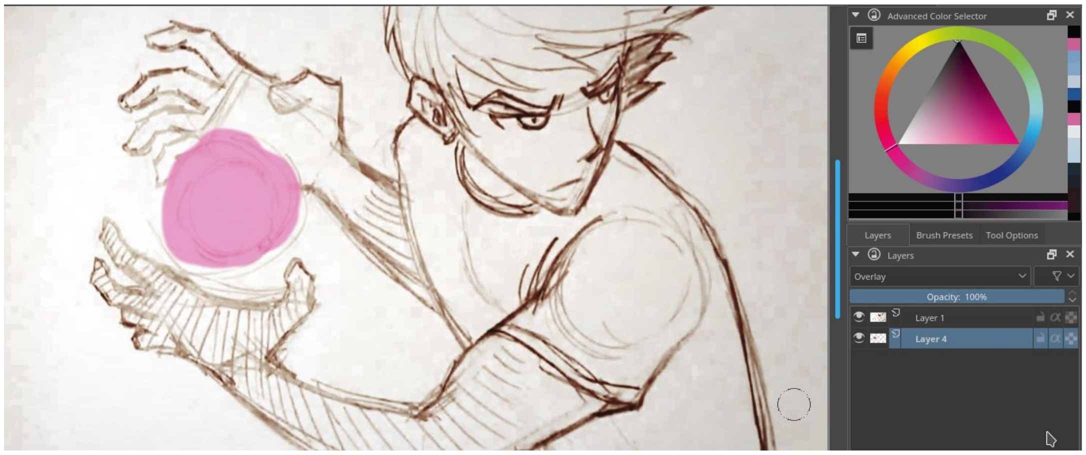
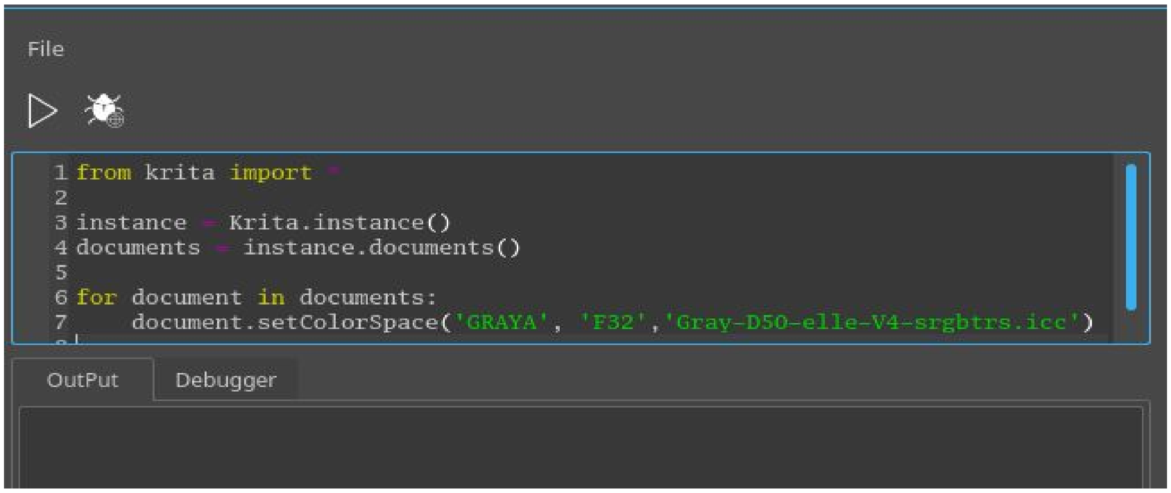


Image2: It's the document of the Image 1 with the invert filter applied.

That's just a simple example among so many others, like set color space for all documents, resize canvas for all documents, export all documents, etc. The main idea here it's that we need to have a good set of examples with key scripts.

2. Python plugins to Krita

The main example that I have today it's the plugin that I implemented in PyQt for SoK (https://season.kde.org/?q=program_home&prg=41). This plugin is called Scripter that can see in the Image 3. It's an Interactive GUI to execute ad hoc scripts with syntax highlight, debugger and code editor features (line number area, output, file menu). This plugin like others it was implemented just with python 3 and PyQt without external dependencies or C++ code.



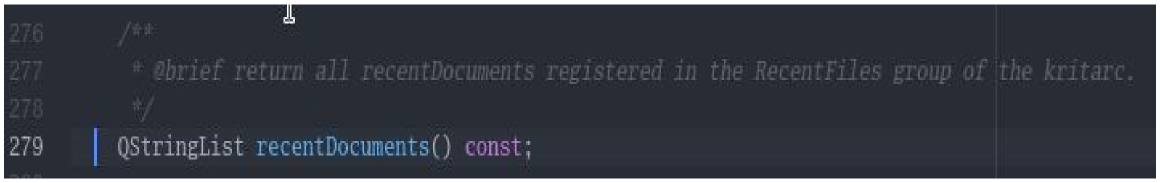
```
from krita import *

instance = Krita.instance()
documents = instance.documents()

for document in documents:
    document.setColorSpace('GRAYA', 'F32','Gray-D50-elle-V4-srgbtrs.icc')
```

Image 3: Scripter with a quite simple example of how to set a color space to all documents.

Talking with quite experienced users, it was possible to identificate other necessary plugins like a high pass filter (https://bugs.kde.org/show_bug.cgi?id=374972), a multifill script (http://davidrevoy.com/article306/tons-of-potions-part-2-multifil) and a docker to show thumbnails of the last ten documents. The main idea behind this last plugin is to give more information about the last opened documents. It's a plugin in progress, but it will be a list of thumbnails with a tooltip showing the document name or path.

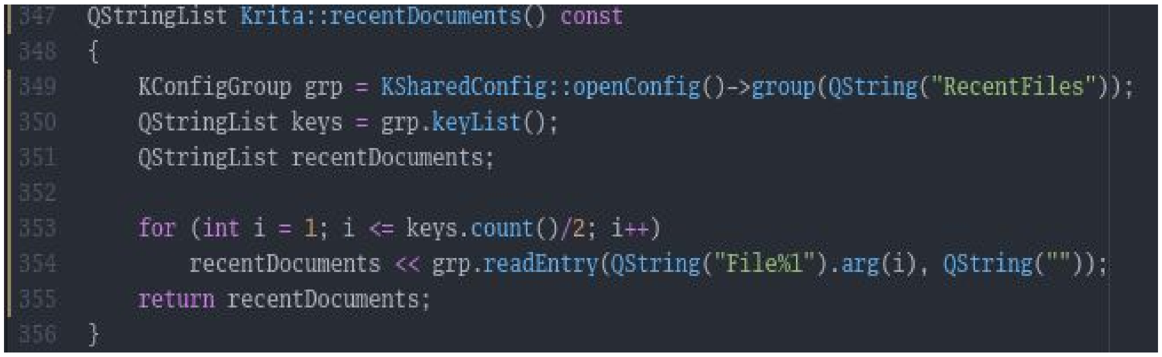
3. Extend libkis

Using the example of the last ten documents thumbnails, I can show how I'm intending to extend the libkis.



```
    /**
     * @brief return all recentDocuments registered in the RecentFiles group of the kritarc.
     */
    QStringList recentDocuments() const;
```

Image 4: Declaration of the recentDocuments method in the Krita.h



```
QStringList Krita::recentDocuments() const
{
    KConfigGroup grp = KSharedConfig::openConfig()->group(QString("RecentFiles"));
    QStringList keys = grp.keyList();
    QStringList recentDocuments;

    for (int i = 1; i <= keys.count()/2; i++)
        recentDocuments << grp.readEntry(QString("File%1").arg(i), QString(""));
    return recentDocuments;
}
```

Image 5: Implementation of the recentDocuments method in the Krita.cpp

<u>All my work to extend is:</u>

- *Investigate the code to understand how a specific feature work. In this case, it was discovered how the KisMainWindow shows the recent documents.*
- *Declare the API and document the C++ header*
- *Implementation in the .cpp file*
- *Expose this in the correspondent .sip file*

<u>Challenges of the Implementation</u>

- I need to talk with the community to understand what they need.
- Define a good set of examples
- Keep it simple - The scripts need to be as simple as possible.
- I need to know how to use every class in the libkis.
- I need to have a good grasp of the structure of the Krita classes

- Code with quality to keep this easy to maintain with the future changes

Roadmap:

I am available for the entire GSoC period I am prepared to work for 36 hours a week, and more, if that is what is required to finish the task. I have a blog now, so I intend to make a weekly post talking about my progress.

Suggestions from the community made me think about the importance of work on the documentation for the Krita's community, then I will work with my mentor and the Krita devs to write the documentation throughout the coding period.

Up to May 29: Bond with the community. How I told previously, talk with the community it's an important part of this work. I will need to understand what they need.

May 29 - July 1: Work on implement all the selected scripts and found a way to make this available for the users in the future. Extensions or changes in the API when necessary.

July 2 - July 9: Test all the scripts with my mentor and other people of the community. Make sure that all scripts are working well.

July 10 - August 11: Implementation of the selected plugins, extending the API when necessary.

August 14 - August 21: Testing the code (both locally and with the help of the community), UI polishing and bug fixing.

After GSoC: Keeping in constant touch with the community and working on the documentation of the API for users like another scriptable softwares.

About Me:

I am a student finishing my 6ª period, pursuing my graduation in System Analysis and Development at Federal Institute of Education, Science, and Technology of Bahia (IFBA). I found in the open source community people that inspired me to keep believing in my work and in the humanity of some way.

Now I'm more active in the Open Source community, promoting KDE and open source in my local community. I'm organizing FLISOL (https://flisol.info/FLISOL2017/Brasil/Salvador) in my city and I will implement a new feature in the Interactive GUI or a new plugin for Krita in the the LaKademy 2017 (https://mail.kde.org/pipermail/kde-community/2016q4/003157.html).

Why me:

- A good grasp over C++ and Qt
- I’m quite experienced with python (around two years)
- I know the libkis and python scripting structure in Krita
- Hard-working person
- Ability to solve problems myself
- Good knowledge about Design Patterns
- I participated in the SoK 2016-2017 (https://phabricator.kde.org/T4551)
- Participating in another open source project created by Sandro Andrade (from the KDE e.V. board of directors) called Émile (https://github.com/sandroandrade/emile-server)
- I love coding

Some Previous contributions:

https://cgit.kde.org/krita.git/commit/?h=eliakinalmeida/T4551-interactive-python-scripting&id=d3a57cfaca9663cbfaa67f20e223512ca1af869e

https://cgit.kde.org/krita.git/commit/?h=eliakinalmeida/T4551-interactive-python-scripting&id=49a904748de833509982717454286d8df96697f8

https://cgit.kde.org/krita.git/commit/?h=eliakinalmeida/T4551-interactive-python-scripting&id=34fd4ca1a70b38aba9d0aad514f9738429b3f9b6

https://cgit.kde.org/krita.git/commit/?h=eliakinalmeida/T4551-interactive-python-scripting&id=c48dc5adf3afea0cde72d38d9bee30ba85c10fb3