Integrate share.krita.org

Introduction

Krita is a free and open source software that provides a great opportunity to designers, to draw and design their art in a cost effective and in a reliable manner which supports the creation of images from scratch.

The project is aimed at improving the user capability via integrating with share.krita.org, a website used by designers to host their work like drawings, animations, paintings, fonts, brush packs, and several other bundles. The implementation of the project would be focussed on two main tasks:

- Integrate share.krita.org with Krita using the *LibAttica* framework and create a GUI for sharing.
- Improving the support for creating/editing bundles with better GUI.

Sharing with share.krita.org using GHNS or the Get Hot New Stuff was supported by Krita in the previous versions, whose support has now disappeared. The primary tasks would aim at providing the user capability to download/upload resources from/to share.krita.org and later on use it to publish images directly. What's new from the existed GHNS is that we are creating a new and better design from scratch and getting it implemented using *LibAttica*.

Even though Krita offers basic support for creating and editing bundles, it does not work well, since some of the features are broken. To improve on this would be the second part of the project where we can improve the support for creating and editing bundles.

Project Goals

The following are the two major goals of the project:

1. *Integrate share.krita.org*. Here, I intend to integrate share.krita.org with Krita using the KDE API *LibAttica* and later on add a GUI user interface for the same. While the *LibAttica* library implements collaborative data sharing for applications, this framework can be used for downloading and sharing additional application data. We could make use of provider class in LibAttica to do the user authentication as we are dealing with upload functionality, keeping search parameters and much more.

   User Cases after integration:

   - The User can open the collaboration services from the resource manager to download the bundles uploaded by other users based on constraints.
   - If a user has the user account linked in with Krita, then he could upload his data of brushes, designs, animations etc to share.krita.org and publish images as well.

2. The second task aims at improving the support for creating and editing bundles. This bundles could comprise of things such as brush packs, ink packs, gradients, patterns

and even font packs. As this part is not well maintained, I would work with the UI/UX team to implement a better GUI design for the whole bundle resource management. Also, with the existing Manage Resource functionality, it should get support in terms of fixing the existing bugs as well.

User Cases after integration:

- Implementation of a GUI could provide a user-friendly design to create bundles with the existing features more efficiently.
- User can create and edit the bundles in the resource manager.
- Implementation of the search widget inside the Resource Manager can enable the user with better accessibility of bundles.

Implementation Details

1. Collaborative sharing using Attica.

Description: Since Krita no longer has the functionality to directly download items from the share.krita.org, the first task here is to implement collaborative sharing with *LibAttica* within the resource Manager providing the provision for downloading resources. Hence, this deals with creating a user interface to provide the user with the feature to download their preferred resource from krita.share.org.

File-Sharing: *LibAttica* is an open collaborative service which supports communication between an application and a web application.

Before we get started, we need to link and have Attica package already inside the system. So that we will have to make changes with CmakeList.txt.

```
find_package(KF5Attica)
target_link_libraries(kritawidgets KF5::Attica)
```

As our aim is to implement collaborative sharing, the user should be able to download and upload the resource bundles into the system and use it according to their needs when they want to. The below code snippet provides as an example for the implementation of the download and upload functions in the Attica framework.

Upload:

```
void ContentCreation::uploadFile()
    doUpload(QString(),
ui.uploadFile->url().toLocalFile())
}
```

While we are going to work on the upload functionality, we will be taking care of the user-authentication as well using the provider class of *LibAttica*.

Download:

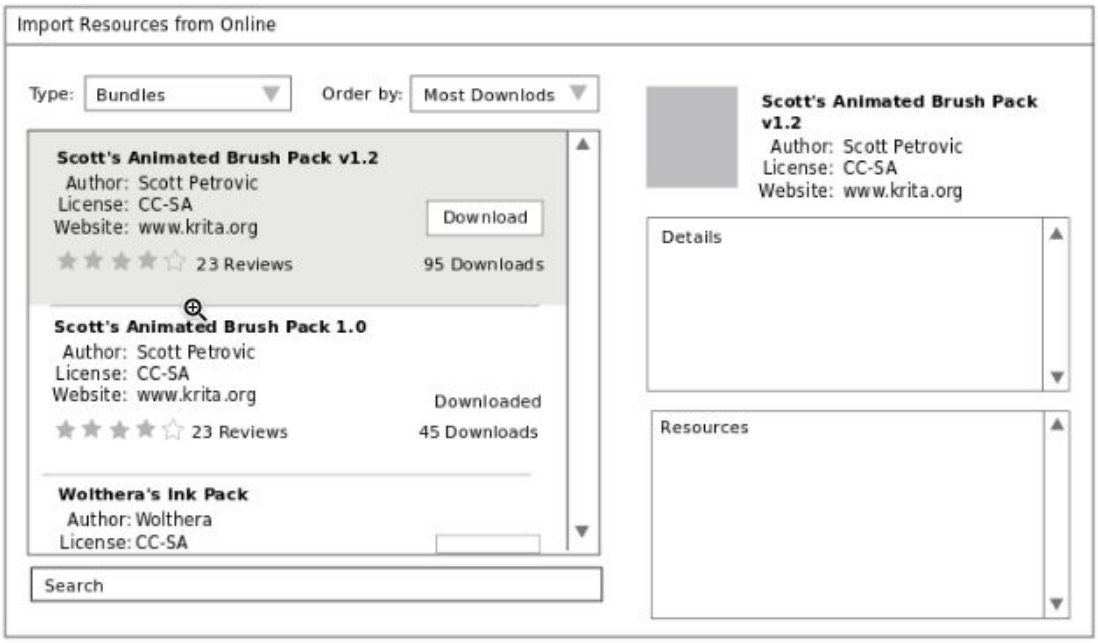
```
void ContentDownload::download() {
  qDebug() << "download";
  QTreeWidgetItem* selectedItem = ui.contentList->currentItem()
  if (selectedItem &&
qVariantCanConvert<Content>(selectedItem->data(0,  Qt::UserRole)))
    {
      Content c = qvariant_cast<Content>(selectedItem->data(0,
Qt::UserRole));
      ItemJob<DownloadItem>* job = m_provider.downloadLink(c.id());
        connect(job, SIGNAL(finished(Attica::BaseJob*)),
        SLOT(downloadLinkLoaded(Attica::BaseJob*)));
        job->start();
   }
}
```

Hence after implementation, the new resources downloaded will available within the resource manager and these resources will be stored inside .local/share/krita/ of the user's device.

GUI Interface: The UI will be implemented working together with the UX/UI design team, referring to the mockup. UI will be using XML and this UI will be connected through defined signals and slots mechanism using Qt.

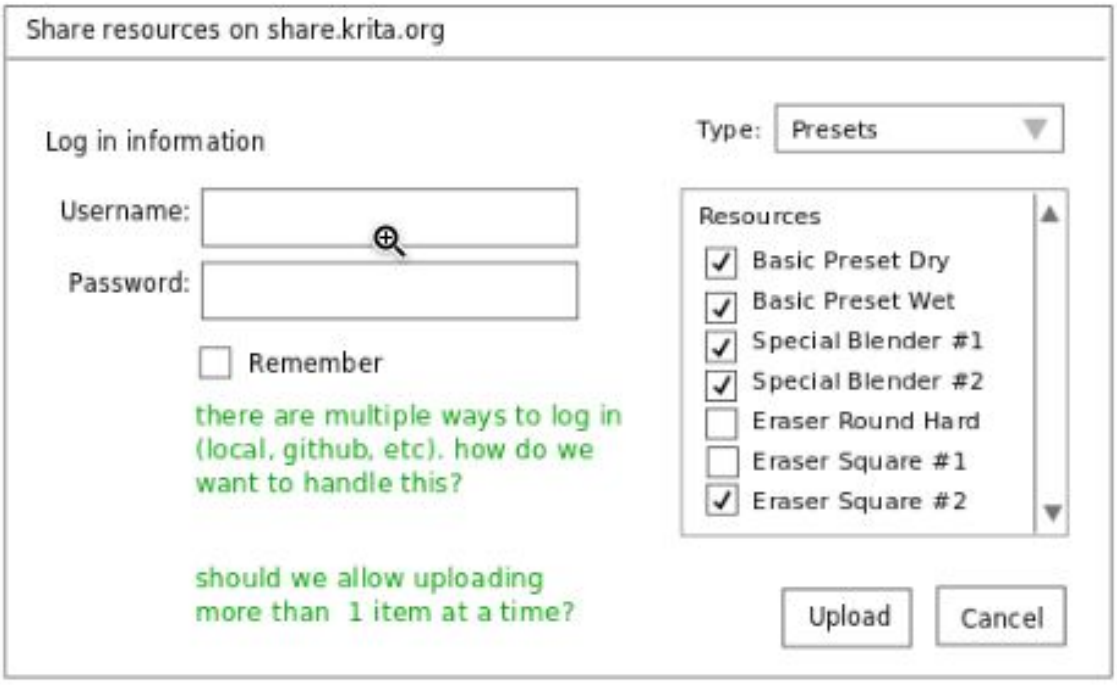
```
connect(ui.upload, SIGNAL(clicked()), SLOT(uploadFile()));
```

For instance, wherever we want to call the GUI to perform the task, we give a call to the “ui” class object to get the signal and slot connected and render the GUI.



*

(Figure 1: Download dialog)



(Figure 2: Upload dialog)

Deliverables: A fully working scenario where the user could interact with the application and get the data from share.krita.org with an interactive GUI.

2. Support for creating and editing Bundles

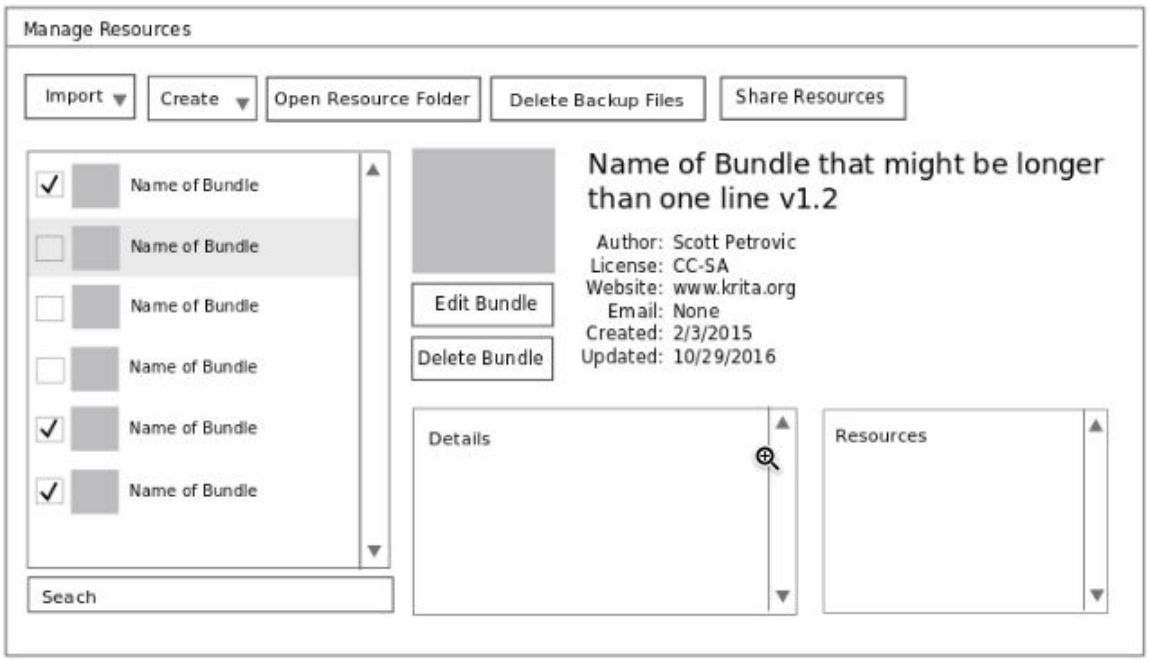
Description: The resource manager that exist is unordered and need a lot of refactoring to be done.

Critical Bugs: There exist a lot of critical bugs which hinder the possibility for the smooth functioning of Krita. A few major bugs are listed below:

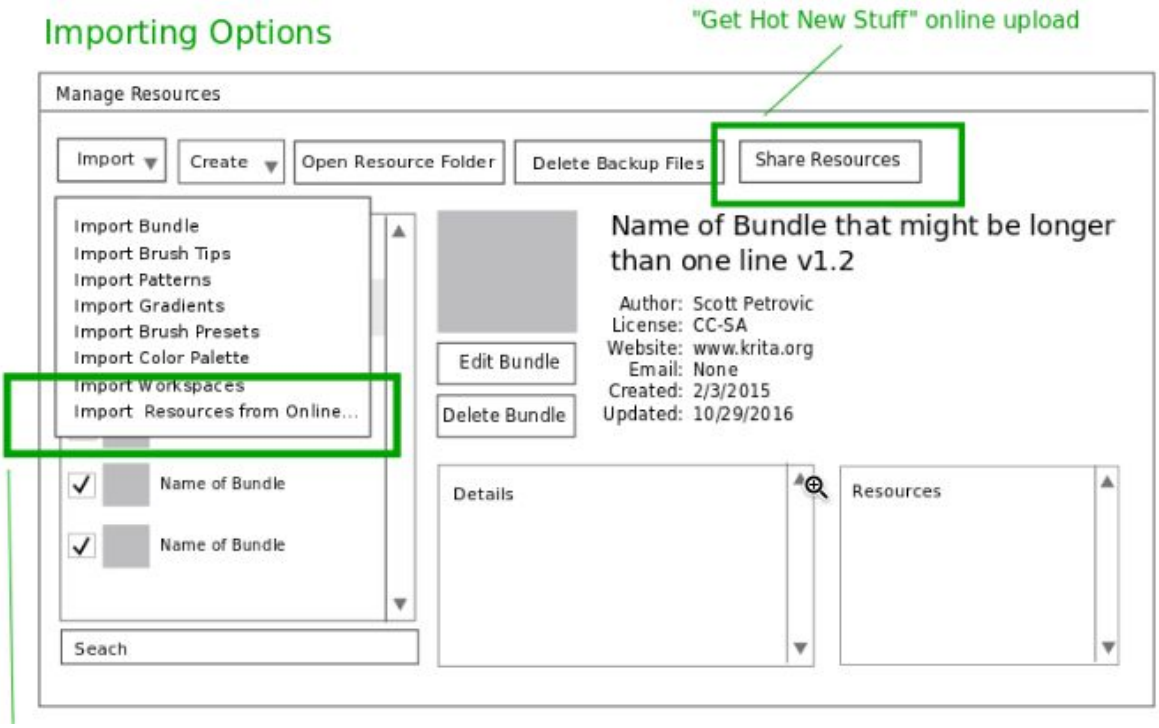
1. There exist problems while deleting brush presets. **Link**
2. We are not able to load palettes stores in CSS file formats. **Link**
3. Also, presets which are duplicate doesn't get overwritten, but rather it stays as a duplicate. **Link**

Adding Features: User capabilities get better with the help of the basic widgets like search, add and delete. Resource Manager in Krita lacks both search and delete widgets which needs to be implemented, which can add on support for creating and editing bundles.

GUI Design: After implementing the necessary features and writing down the required functions for the proper functioning of Resource Manager, the final task is to create a better GUI for the Resource Manager. A mockup of the GUI Design is shown below:



(Figure 3: Resource Manager)



(Figure 4: Expanded list of Resource Manger)

Deliverables: A working resource manager in Krita which would be supported with basic features that provide the user with necessary search/edit functions in presence of a user-friendly GUI.

Files to be Added/changed with the context

/libs/widgets/CMakeLists.txt	CMakeList needs changes in order to add the target files and path.
/lib/widgets/wdgdlgcontentdownloader.ui	GUI file for the Attica download implementation with the UI changes.
/lib/widgets/dlg_content_downloader.cpp	
/lib/widgets/dlg_content_downloader.h	
/lib/widgets/wdgdlgcontentUploader.ui	GUI file for the Attica upload implementation with the UI changes.
/lib/widgets/dlg_content_uploader.cpp	
/lib/widgets/dlg_content_uploader.h	

/plugins/extensions/resourcemanager/resourcemanager.cpp	● Changes due to refactoring the bugs that exist in the resource manager. ● Changes as new features like search existing bundles will be added. ● Changes to create and edit bundle section which needs refactoring.
/plugins/extensions/resourcemanager/resourcemanager.h	
/plugins/extensions/resourcemanager/wdgdlgbundlemanager.ui	As we are redesigning the whole UI/UX for the resource manager this files will have the required changes.
/plugins/extensions/resourcemanager/dlg_bundle_manager.cpp	
/plugins/extensions/resourcemanager/dlg_bundle_manager.h	

Tentative Timeline

Until May 30th	● Community bonding period, get familiar with the community. ● Fix bugs existing in the resource manager and in create/edit bundle section. ● Understand the working of LibAttica API and codebase of Krita. ● Discuss conducting weekly sessions with my mentors.
30th May - 13th June	Implement functions for collaborative sharing using Attica.
13th June - 26th June	Start creating the GUI for sharing resource bundles.
27th June - 8th July	● 1st phase Review ● Fix bugs, in the create/edit bundles ● Testing the implemented features ● Documentation of the classes and functions created.
8th July - 15th July	Start fixing the bugs in the Resource manager
15th July - 29th July	● 2nd Phase review

	● Rewrite the functions needed to be changed and implement new features. ● Documentation and testing.
29th July - 15th August	Work on the GUI part with the UX design for Resource Manager
15th August - 21st August	After a thorough review of the code and testing with mentors and other developers, make the final submission.
Afterwards	Continue testing and bug fixing. Provide support as well.

About me

- Name: Aniketh Girish
- Proposal Title: Integrate share.krita.org
- Email Address: anikethgireesh@gmail.com
- Freenode IRC Nick: aniketh___
- Blog: aniketh01@github.io
- GitHub: Aniketh01
- Country/Time Zone: India (GMT+5:30)
- Will you treat Google Summer of Code as a full-time employment?: Yes

I'm a first-year Computer Science and Engineering student pursuing my bachelor's from Amrita University. I'm an active member of FOSS club in our university(FOSS@Amrita).

I started contributing to KDE since September 2016. I was selected for Season of KDE(KDE-SoK) 2016-17 in which I worked on the project Kstars, mentored by Cojocaru Raphael and was successful in completing it. I was invited as a speaker for KDE India Conference 2017 in IIT Guwahati, where I gave a talk on the topic "Object tracking using OpenCV and Qt". I have also contributed to other projects of KDE like Konsole, KIO and much more. Apart from building desktop applications and contributing to different open Source projects, I like to build web applications as well. I also write for Open Source magazines during my free time.

Following commits have been done by me in various projects of KDE.

1. Commit a965cb
2. Commit d275e9
3. Commit 16c3fe5
4. Commit 5601e99

5. Commit a2aef4c
6. Commit f235818

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

I will be having my university exams in the month of May. Nevertheless, I would be able to devote 40 or more hours a week throughout the program.

Reference

1. https://techbase.kde.org/Development/Tutorials/Collaboration/attica/Introduction
2. https://anongit.kde.org/kdeexamples.git
3. https://phabricator.kde.org/T379
4. https://api.kde.org/frameworks/attica/html/index.html