Krita in Stores

Windows Store

Instructions for the Windows Store are in packaging/windows. Uploading to the Windows Store:

- Go to https://partner.microsoft.com/en-us/dashboard/products/9N6X57ZGRW96/overview
- Select Update in the latest submission
- Check whether the store listing needs to be updated
- Go to packages and upload the msix that was created with the scripts in packaging/windows.
- When the package has been accepted, don't for get to click the Submit to the Store button.

Steam

Steam is a little bit involved, but I have it pretty much down to a system now. :)
The entire process is documented on https://partner.steamgames.com/doc/sdk, but here's the rundown:

1. Set up a **Steamworks** account, and download the Steamworks SDK somewhere on your system. (Works on both Windows and Linux, and maybe MacOSX too, but I'm not sure about that.)
2. Download stable **Krita** binaries for all Steam platforms (right now that's just Linux and Windows); place the respective content into windows/ and linux/ subdirectories inside sdk/tools/ContentBuilder/content/.
    - **Linux note:** for Steam we currently run Krita through a small launch.sh script. This is used to circumvent Steam's built-in "linux runtime", since we don't need it and it interferes with appimages. launch.sh expects the appimage to be named krita.appimage, so we name it that.
    - **Windows note:** on Windows, Steam is configured to launch Krita at the relative path krita\bin\krita.exe so I usually rename the folder that holds our portable build to just krita. This could be changed, but it works for me.
3. With the SDK downloaded and the content in the right place, we can *almost* build and push, but first we need to set up "**SteamPipe Build Scripts**" (https://partner.steamgames.com/doc/sdk/uploading#3)
    - This is a script that we feed to tools\ContentBuilder\builder\steamcmd.exe that contains, among other things, Krita's **AppID** (which you can find on Steamworks),

the path where our content is located, the path where we want to build to, and the path to a separate build script for each of our **Depots**.
    - A "depot" is basically what Steam calls a package. Each depot has a unique DepotID number. Through Steamworks' web interface we can create depots, and through the SteamworksSDK we can push our built content to them. For Krita we have have 4 depots (Krita Common, Krita Windows, Krita Linux, and Krita MacOSX), but right now we only really use the Windows and Linux ones. (The other ones were made just in case we released on OSX or had some common data that we could share between all platforms. When I got involved with Krita on Steam I think we had 10-12 depots! haha.)
    - Each depot has a script that we referenced in our main App build script, and it's basically just used to tell the SDK where the content for that particular depot lives. (The windows depot, for example, points to the sdk/tools/ContentBuilder/content/windows/ on my machine.)
    - This is the worst part of the entire thing, but it really only has to be set up once--make the depots on Steamworks website, and then write a build script for the App and for each of the 4 depots, keep them on your computer forever.
    - I could share my build scripts here, but I'm not sure if there are any security ramifications to doing so, so I'll avoid it for now.
4. Then run sdk/tools/ContentBuilder/builder_linux/steamcmd.sh +login <STEAM_USERNAME> <STEAM_PASSWORD> +run_app_build_http -desc "Krita Desktop Build" <PATH_TO_APP_BUILD_SCRIPT> (or the Windows equivalent), which will log you into Steamworks, build all of our depots and hopefully push them up to Steam.
5. At this point the newest build has been pushed to Steam--but it's not LIVE yet! In order to send it out to our Steam users we have to set the default branch to the latest build in the "SteamPipe" section of the Steamworks web interface.
    - Note: we also have a beta branch that we can use for pushing beta builds if needed, as well as a rollback branch that I keep pointing to the last minor version build for Steam users that want the option of jumping back a version. (Both are opt-in through Steam's GUI.)
6. Finally we should tell our users what's been updated and thank them for their support, and we can do that through the Post/Manage Events & Announcements section of the Steamworks web interface.

It's kind of *a lot,* actually... but it does get easier as you get used to how it works.

I think that's just about everything. At this point, for me, it's basically as simple as downloading the latest stable build, putting it in the proper content directories, running the steamcmd.sh with my build script, pointing the default branch to the build through Steamworks, and then writing a post about the update.
I also try to take some time to read reviews and respond when it's warranted, and I like to thank people on Steam and let them know that their support helps to drive development. (Especially

since it's not the usual FOSS crowd and many of them may not be fully aware of our development model.)

Also worth noting that Steamwork's web interface is where we do all of our other management, like updating the Store page information, setting up discounts, etc.

That's all I can think of for now, let me know if you have any questions.

Apple Store

Epic

Get the build-patch tool

https://launcher-public-service-prod06.ol.epicgames.com/launcher/api/installer/download/BuildPatchTool.zip

Process the build:

XXX: secrets that can be found in https://epicsupport.force.com/epicgamesstoreknowledgebase/s/group/0F94z000000oM87CAE/boudewijn-rempt-software-krita, for those who can login there.

```
C:\dev\BuildPatchTool_1.4.0\Engine\Binaries\Win64\BuildPatchTool.exe ^

-ClientId="XXX" ^
-ClientSecret="XXX" ^
-OrganizationId="XXX" ^
-ProductId="XXX" ^
-mode=PatchGeneration ^
-CloudDir="c:\dev\epic"
-BuildRoot="C:\dev\krita-x64-4.4.3-setup"
-BuildVersion=""
-AppLaunch="bin\krita.exe"
-AppLaunch="bin\krita.exe" ^
-ArtifactId="XXX" ^
-AppArgs=""
```

Label the build:

```
C:\dev\BuildPatchTool_1.4.0\Engine\Binaries\Win64\BuildPatchTool.exe ^
```

-ClientId="XXX" ^
-ClientSecret="XXX" ^
-OrganizationId="XXX" ^
-ProductId="XXX" ^
-ArtifactId="XXX" ^
-mode=LabelBuild ^
-BuildVersion="4.4.3" ^
-Label="Live" ^
-Platform="Windows"

The build will appear in the dev.epicgames.com portal:
https://dev.epicgames.com/portal/en-US/boudewijn-rempt-software

Play Store

This needs the krita-playstore-key.jks keystore.

We only need to upload a single app bundle from the app bundle builder.

Which should be signed like:

jarsigner -keystore keystore.jks -storepass password -keypass keypass krita-appbundle.aab key

Then go to Google Play dashboard:
https://play.google.com/console/u/0/developers/5087476919189101974/app-list?pli=1#AppDashboardPlace:p=org.krita&appid=4972307855597874233

Click on Krita, go to Release section (it may be hidden, use hamburger menu to open it). Expand the Testing item and then click on Open Testing, Create new release and fill in Release name and additionally the Release notes. Upload the App bundle and then click Review release.