Telemetry for getting statistics for which features are used the most in Krita

March 24, 2017

1. Contact Information

Full name: Alexey Kapustin
Email: djkah11@yandex.com
IRC nickname: akap
Telegram nickname: aluka1
Phone: 8-999-986-73-81
Location: Moscow, Russia

2. Problem

Krita contains a large number of tools for painters, a large number of settings and features. Their number increases every year, but not all of them are popular. More precisely, the critics want to know which of the options are popular. Based on this information, you can determine the vector of further work, finalize popular, but not too well-implemented features, remove old, useless functions. The collection of statistics will help to create achievements to the Steam. Achievements for Krita is good marketing solution, which help us to find new users. It would be reasonable to make two versions: Steam version and non-Steam version.

3. Implementation

3.1 Client-side

The main idea is to use Google Protobuf logs to write out necessary information. All information is stored in the logs and as necessary (too large log file, not sent for a long time, etc.) is sent to the server. Sending in the original form the information is irrational, so before sending, the information from the log is analyzed and only the "squeeze" from this information is sent to the server.
Main data that we want to analyze:

Information about	Source	Data

Tools	KisTool::activate, KisTool:deactivate, etc, overloaded in specific tools	`int32 activationTime;`
		`int32 useTime;`
		`int32 kindTool;`

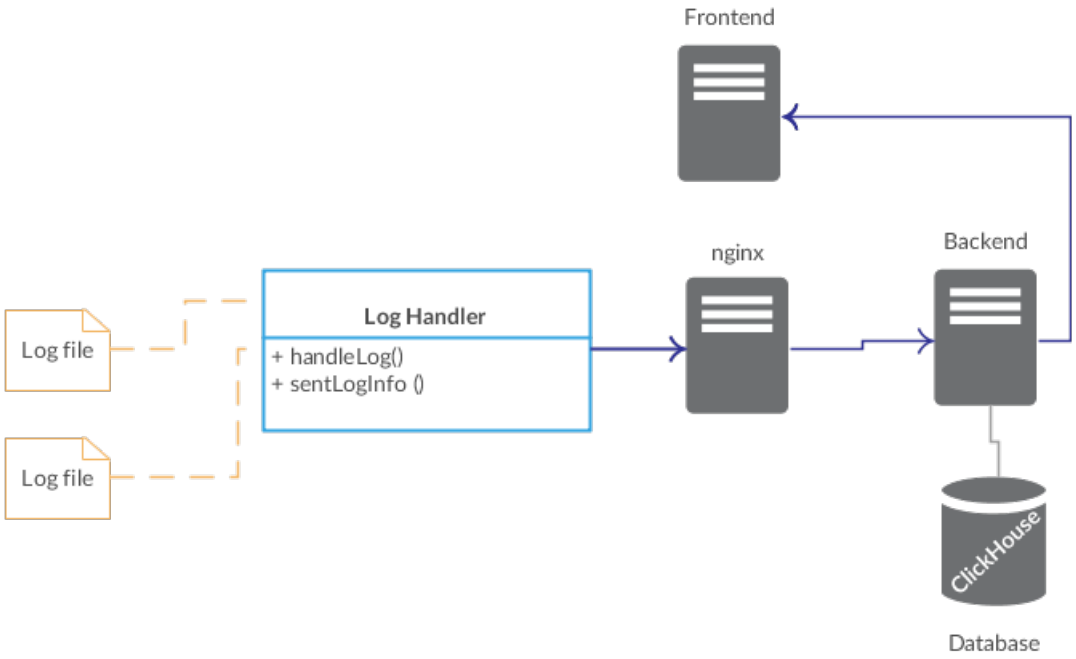
Strokes	KisToolFreehandHelper::initPaintImpl
	KisToolFreehandHelper::endPaint etc
		`int32 distance;`
		`int32 time;`
		`int32 opacity;`
		`string currentFgColor;`
		`FillStyle fillStyle;`
		`StrokeStyle strokeStyle;`

Presets	KisToolFreehandHelper::initPaintImpl	`string koid;`
		`OutlineMode outlinemode;`
		`double paintOpFlow;`
		`string compositeMode;`
		`double savedBrushSize;`
		`int32 hashCode;`[3.1]

Actions	KisMainWindows actioncollection()	`string name;` `string actionSource;`

[3.1]: It is required to save presets into map structure for less size of log.

3.2 Server-side



3.2.1 Own stats

Unfortunately, clickstream services are focused on gathering information from websites. Own implementation allows you to create a convenient api for yourself, and also leaves room for further development and expansion.
Scheme description: The data from the user is serialized and sent to our nginx, which proxy requests to the backend.On the backend, there is deserialization and recording of information in the database. The information from the database returns to the backend, where it then goes to the front-line. This scheme seems somewhat redundant, but it has an extensible architecture and is capable of withstanding highloads. Nginx is an optional element, but with increasing load, this architecture will allow you to scale horizontally

3.2.2 Steam achivments

It is required to add Steam Sdk.We should to create achievements via Steam web interface. Filling in the list of achievements will be discussed with community. Before any functions can be used, first *SteamAPI_Init()* must be called.After that we can call *SteamUserStats()->SetAchievement("Draw_the_length_of_the_equator")* and set specific achivment.

3.3 Technologies

Nginx is widespread server, which is used by nearby 70% percents of top-100 worldwide side. ClickHouse allows you to perform analytical queries interactively on real-time data. The system is scalable to very large amounts of data.

4. Timeline

Work schedule Implementation timeline

Period	Work	Period	Work
22 May -4 June 2,3153	End of the school year. Opportunity to work 7-10 hours a week 1,3644	1,2733 0,9496	5,3763 5,3763