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

March 27, 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. Use of clickstream analytics will help to learn the popularity of feautures.

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. All the information sent will be completely anonymous for the non-Steam version , and will also be collected with direct user's permission . For users of the Steam version, the collected information necessary to achievements, will be associated with their steam-profiles.

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;`

Presets	KisToolFreehandHelper::initPaintImpl	`string koid;`
		`double paintOpFlow;`
		`string compositeMode;`
		`double savedBrushSize;`
		`int32 hashCode;[1]`

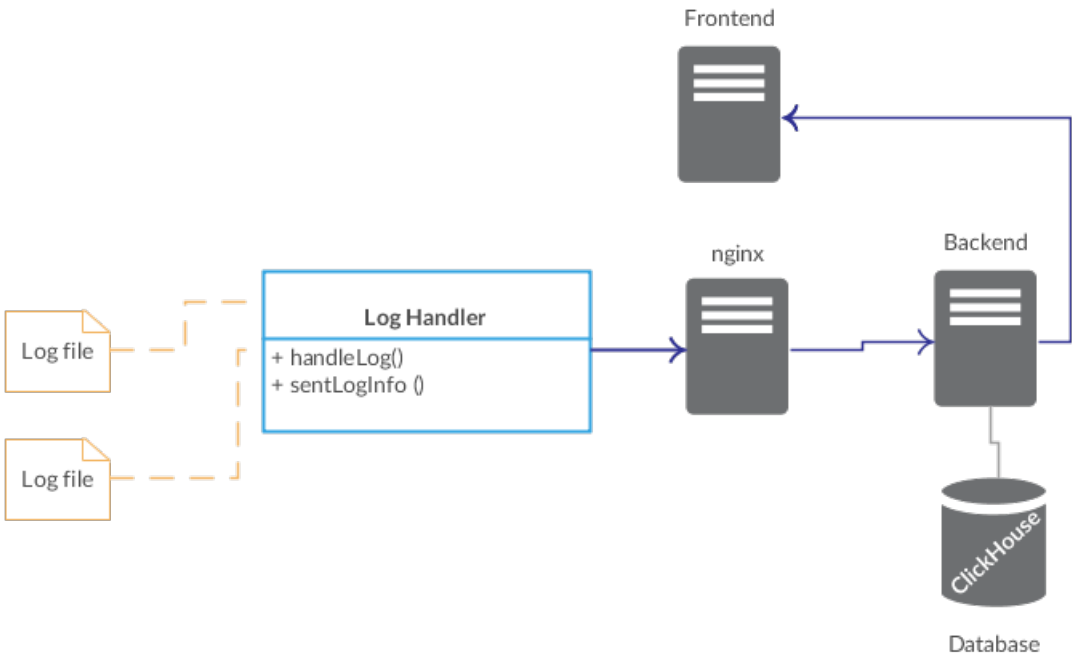
Actions	KisMainWindows actioncollection()	`string name;` `string actionSource;[2]`

Image_Properties	KisDocument::saveFile	`int32 fileFormat;`
		`int32 colorSpace;`
		`string colorProfile;`
		`int32 imageSize;`
		`int32 width;`
		`int32 height;`
		`int32 numLayers;`

Asserts	kis_assert_common()	`string assertMessage;`
		`string file;`
		`int32 line;`

[1]: It is required to save presets into map structure for less size of log. [2]: Hotkey, menu of something else

3.2 Server-side



3.2.1 Own stats

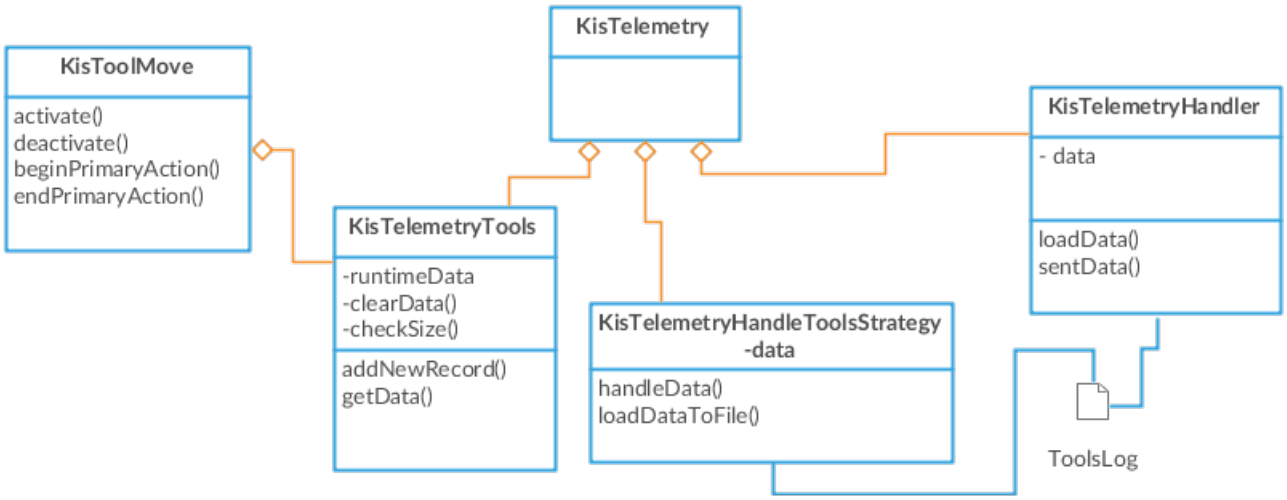
Unfortunately, clickstream services are focused on gathering information from websites. Own implementation allows you to create a convenient api for yourself, and also leaves room for further development and expansion.
Scheme description: The data from the user is serialized and sent to our nginx, which 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 Example

Here is an example of how data storage for a class will be implemented.KisTelemetryTools is singleton.



When events triggered by KisToolMove::activate, KisToolMove::beginPrimaryAction, KisToolMove::endPrimaryAction, information is written to the appropriate structure.When the KisToolMove::deactivate event triggers, information from this structure about usage will be sent to the class KisTelemetryTools. When a certain size is reached, the information stored in KisTelemtryTools::runtimeData is passed to KisTelemetryHandleToolsStrategy, where it is processed and loaded into a file. For example, we have 10 records that KisToolMove was used (one entry is one use interval between KisToolMove::activate() and KisToolMove::deactivate()). This information is analyzed, serialized and written to a file. Record: total usage time, average usage time, average time between activation and deactivation, total time between excitation and deactivation, number of uses (10), name of this tool. It is reasonable to replace the name of the tool with some number to save memory.When Krita is completed, KisTelemetryHandler is called. It reads the file(s) and tries to send it to the server. All data is transmitted over the https protocol. We have a post-request that looks like this (request body):

```
{
"avgUseTime": 111,
"avgADtime": 222,
"useTime": 333,
"useAdTime": 444,
"useCount": 10,
"toolName": 1
}
```

The request comes to nginx whence it is proxied to the backend-server. Comes to the backend server, deserialized, checked for correctness. After that, it records the database in approximately this way. The frontend periodically polls the backend and, based on the state of the database, issues current statistics "online".

3.4 Technologies

Protocol Buffers (a.k.a., Google protobuf) are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data. For C++ language it provides a runtime multi-platform library. 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	
Period	Work
22 May - 4 June	End of the academic year. I can to work 7-10 hours per week.
4 June - 18 June	Exams in university. I can start working 25-30 hours per week.
18 June - 29 Aug.	During this period I can work full-time on the project.
29 June - 15 Sept.	During this period I can work full-time on the project and this period serves as compensation for the first periods.

Implementation timeline	
Period	Work
22 May - 4 June	I study a code of a client part, I begin to do system of logging.
4 June - 15 June	Impelementing logging system for tools.
15 June - 29 June.	Impelementing logging system for asserts and Image_Properties without taking care of memory. Unstable work is possible
29 June - 20 Jul.	End of the implementation of the logging system. Memory optimizations. Investigate communication between server and client
20 Jul - 1 Aug.	Creation of Steam-version. Discussion of achievements with community.
1 Aug - 29 Aug.	Creation of server-side. Load testing.
29 Aug - 15 Sept.	Buffer time for issues that may take longer than expected.

5. About me

I'm Alexey Kapustin, a 3rd year student of Bauman Moscow State Technical University of Software Engineering Faculty. I'm interested in C ++, Python,Javascript, web development. I also study at the Technopark Mail.ru in the 3rd semester. (Russian School of Web Architects).
I did projects using Django, Nodejs. Within the semester project in the second year of education, I made a simple tower-defense game on SFML C++ library. In the third year I made an API for working with the "Forums" database, which withstood high loads. To create this api, I used the libraries Wt, Qt, Libzdb. Since last summer I have commited to Krita. You can also see my githab.