Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Martin Indra**

Study programme: Open Informatics
Specialisation: Software Systems

Title of Bachelor Project: **Collab - collaborative raster painting editor**

Guidelines:

Collab is a tool for collaborative painting. It allows to paint and share an image with other users in real-time. The Collab tool is implemented under the open source license by the student. The thesis objective is to improve the architecture and interface of the tool and to create proper software documentation. The main goals of the thesis are:

- to introduce the issue of collaborative painting on computer network in real-time,
- design and document network protocol used in Collab with a regard to the analysis,
- create a library (framework) implementing the network protocol,
- change the existing server so it integrates the network protocol library,
- change the existing client, that is graphical editor, so it integrates the network protocol library and its GUI is improved,
- validate results of the thesis via tests.

Bibliography/Sources:

1. Indra M., Collab [online] http://collab.mgn.cz/ 2013

Bachelor Project Supervisor: Ing. Ondřej Macek

Valid until the end of the summer semester of academic year 2014/2015

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]
doc. Ing. Filip Železný, Ph.D.
Head of Department

[illegible]
prof. Ing. Pavel Ripka, CSc.
Dean

Prague, February 25, 2014

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Bachelor's Project

Collab – Collaborative Raster Painting Editor

Martin Indra

Supervisor: Ing. Ondřej Macek

Study Programme: Open Informatics, Bachelor

Field of Study: Software Systems

January 5, 2015

Declaration

Abstract

Collab is a project which aims to create a full platform empowering users with a graphical editor which could be connected to is's other instances via a computer network in a way that all interconnected users draw on the same "canvas" and see each others work immediately.

The thesis introduces matters of collaborative painting, suggests a solution based on surveys, designs final project architecture and describes actual implementation. The project was divided into multiple independent parts. CRPP – Collaborative Raster Painting Protocol is a network protocol and it's documentation. Collab Desktop is a desktop client users will use for collaborative painting. Collab Canvas is a Java SWING library for graphical editors implementing interface which make it easy to connect it to a network. Collab Server is a server to which the client connects.

The Collab project was in most parts successful. There is designed network protocol, several Java libraries and server and desktop applications ready to use.

Abstrakt

Collab je projekt, jehož cílem je vytvořit celistvou platformu poskytující uživatelům grafický editor, který může být napojen k jiným vlastním instancím přes počítačovou síť takovým způsobem, že všichni propojení uživatelé mohou kreslit na stejné plátno a vidí v reálném čase práci ostatních.

Práce uvádí problém sdíleného kreslení, navrhuje řešení na základě rešerší, navrhuje finální softwarovou architekturu a popisuje vlastní implementaci. Projekt byl rozdělen do několika nezávislých částí. CRPP – Collaborative Raster Painting Protocol je grafický síťový protokol a jeho dokumentace. Collab Desktop je aplikace, kterou uživatelé použijí pro sdílené kreslení. Collab Canvas je Java SWING knihovna pro grafické editory, která implementuje rozhraní usnadňující napojení na síť. Collab Server je server, ke kterému se mohou připojovat klienti přes CRPP.

Collab projekt byl z velké části úspěšný. Vznikl návrh a dokumentace síťového protokolu, několik Java knihoven, server a klientská aplikace ve verzi připravené k použití.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Collab is a digital analogy to a physical place where multiple people have access and can draw on a canvas simultaneously. It provides the possibility of collaborative sketching over geographical distance as if in physical proximity.

There are several ways to represent visual content, this project is concerned about two dimensional raster graphics. That means a matrix of pixels where each is represented by a color defined by a particular number of color channels each stated by a particular number of bits (color depth).

1.1 Motivation

There is an increasing need for virtual connection for both professional and unprofessional collaboration via the Internet. Part of the issue is visual representation and sharing of all kinds of visual material. This need is partially fulfilled by current technologies, but no free software is presenting users with all the crucial features of comfortable visual sharing. These features are described further in this chapter and the free-software issue is described in Open-source (page 21).

1.2 Organization

The project is trying to bring a solution on all levels necessary for a complete, usable software. It concerns a client, server, multiple libraries, network protocol, problem description, design and documentation.

1.3 Specification and requirements

The project should solve the problem of real-time visual connection by sharing gradual changes into the graphics within milliseconds after they occur. Multiple users are connected onto a canvas and are synchronized to its content. Everybody on the canvas can paint and simultaneously receive new data from other collaborators.

Each time somebody paints something the change is immediately displayed to him and within fractions of seconds distributed to the other users. Immediate feed-back is important for the sake of usability, otherwise the user would be distracted by not seeing an immediate response. Application of the changes among other connected people can take perceivable time but no more than several seconds, otherwise it would disturb the cooperation.

There has to be an authority or system guaranteeing accurate synchronization among users. There should be no differences between the data each client has lasting more than a noticeably long period of time.

The graphics and data interchange have to be specified in generic enough terms to enable various clients to share all kinds of raster graphical content. The data interchange protocol should enable users to both remove and add other content and to do so into mutually overlapping layers.

1.4 State of the art

There is no universally accepted and well defined term regarding real-time visual sharing. The most wide spread name under which the technology is regarded in general is *paint chat*. Paint chat is defined as a collaborative painting tool combined with a textual chat.

	multi-platform	real-time	raster	general	free software
Cosketch [Cos14]	yes	no	yes	no	no
Whiteboard [Meg14]	yes	yes	yes	no	no
openCanvas [Por14]	no	?	yes	no	no

Table 1.1: List of some available tools and their features

Most of the tools or software available in December 2014 support only basic features and cannot be used in the sense of a general graphical editor [con14b]. Table 1.1 illustrates the limitations of some available software. According to the research of this thesis there is no feature rich multi-platform general collaborative graphical editor available under free software license. Collab aims to fill this technological gap.

Chapter 2

Architecture

Collab is divided into several parts which are the network protocol, server and desktop client. Collab Desktop and Collab Server use a network library for the connection and data relay and Collab Desktop uses another library for painting. See 2.1 for illustration.

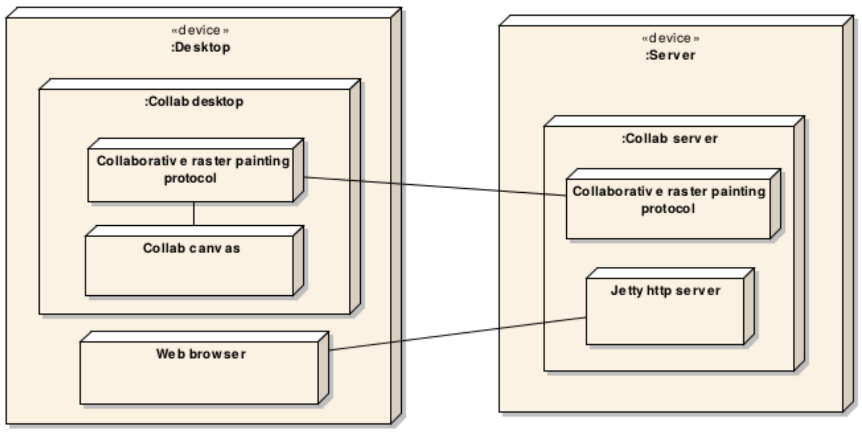


Figure 2.1: Collab deployment diagram

2.1 Protocol

The Collab project is based on a protocol developed for the purpose of real-time graphical data sharing. It's name is *Collaborative Raster Painting Protocol* abbreviated as *CRPP*. The protocol is design to be data and processing time efficient if the graphics are changed gradually.

CRPP is a binary protocol functional over a TCP/IP connection. CRPP is not hardly linked to Collab software so it can be used with any graphical editor.

Protocol performance

The most demanding part of the protocol is graphics distribution. All other data are not sent so frequently and are at least one-fold smaller in size. It requires the transition of image data every time a change occurs from any to all users.

The goal was to fit the transition into a generally available internet connection band-with. The most limiting part of an average internet connection is its upload bandwidth [Tec14]. Because of that 100 KiBps was considered as the maximal upload throughput. 200 KiBps was used as a limiting download speed.

Network requirements were tested experimentally in a painting application prototype which enabled the drawing of thin coloured lines. Users were asked to draw different shapes, pictures and texts. Then every 500 milliseconds of newly created image data were cut-off into squares of 50 pixels. Squares with no change were discarded and the remaining were compressed into PNG images with maximal compression. This PNG data was gathered for about 10 minutes and the maximal and mean pace of data generation were counted.

The measurement results gave a rough estimation of throughput requirements at 45 KiBps maximal and 20 KiBps average per user. The data amount fluctuated around the average most of the time. If we considered eight users as the maximum on a canvas then we get 160 KiBps minimal download throughput. The results are a perfect fit for an average internet connection of end-users.

2.2 Collab Canvas

Collab canvas is a Java framework based on SWING. Its purpose is to put away most of the work needed to create a graphical editor compatible with CRPP. Collab Canvas solves all of the problems connected to graphics rendering and composition so project developers using this framework can focus more on editor features instead of all underlying algorithms.

2.3 Collab Desktop

Collab Desktop is a Java SE graphical editor using Collab Canvas framework. Collab Desktop put together all parts required for a fully functional collaborative editor.

2.4 Collab Server

Collab Server is a Java application which implements network protocol. It communicates with clients and distributes and holds graphical and meta data in memory.

Chapter 3

Network protocol

For the purpose of cross-compatibility among servers and clients there is a TCP/IP based protocol. Its name is *Collaborative Raster Painting Protocol* (abbreviated *CRPP* as it will be referred to later). The protocol is binary and it sends shared information by separate messages.

3.1 Protocol layers

The protocol is divided into two separate layers as is illustrated by figure 3.1.

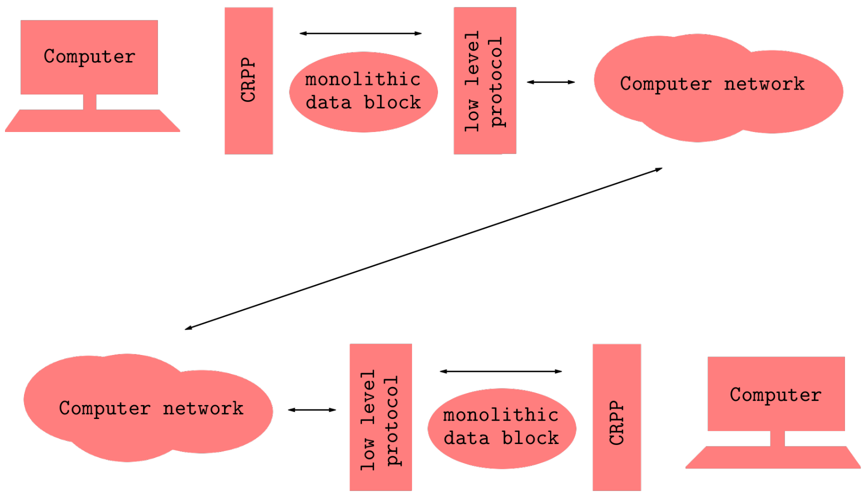


Figure 3.1: Complete data endpoint to endpoint path

Down layer, called *CRPP Binary*, is responsible for message packing and its transmission. It receives message on one side and sends that message on the other side of the connection. It is responsible for delivering messages in the same order as they were sent. CRPP Binary could be in theory substituted by any other message transmitting protocol (e.g. RabbitMQ and its AMQP) but its own implementation was used for the sake of processing speed and bandwidth requirements.

In CRPP Binary every message send as byte sequence and represents independently interpretable information. The messages are structured into a header and body. Header is composed of message ID (first four bytes) and body length (another four bytes). Message structure is illustrated by figure 3.2.

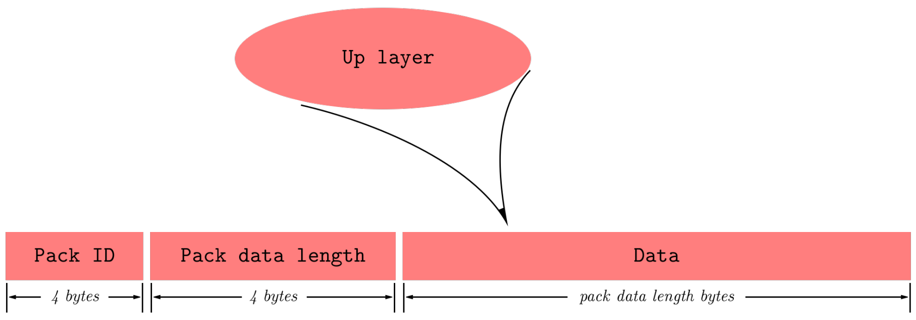


Figure 3.2: Down layer's message structure

The upper layer interprets the data types and communicates with the build on application.

The messages in the upper layer are composed of a command (four-byte long ASCII code) and parameters. Each command represents specific task and could but needn't carry additional data. The command data payload is slitted into parameters, where each one has its own name and body. A parameter's name is defined as four ASCII characters. Then four bytes of the body length precede the body itself. The parameters could contain any binary data which are interpreted in accordance with the command and parameter name. The up layer is illustrated by figure 3.3.

3.2 Image data format

Images are represented as a raster of pixels with 32 bit color depth, where 24 bits represent a color (red, green, blue) and 8 transparency (Alpha channel). 32 bit color depth has been chosen because of its sufficient accuracy and wide spread use among computers. There are a great amount of image data formats supporting either 24 bits (color only) or 32 bits (color and transparency) color depth, some of them supporting lossless or lossy compression [con14a].

There is a requirement for 100 % accuracy in data distribution but because of the constraints of limited network capacity lossless compression is used. Because of easy portability *PNG* (*Portable Network Graphics* [fS04]) was chose for visual data representation. PNG is wide spread and highly supported among a wide variety of platforms.

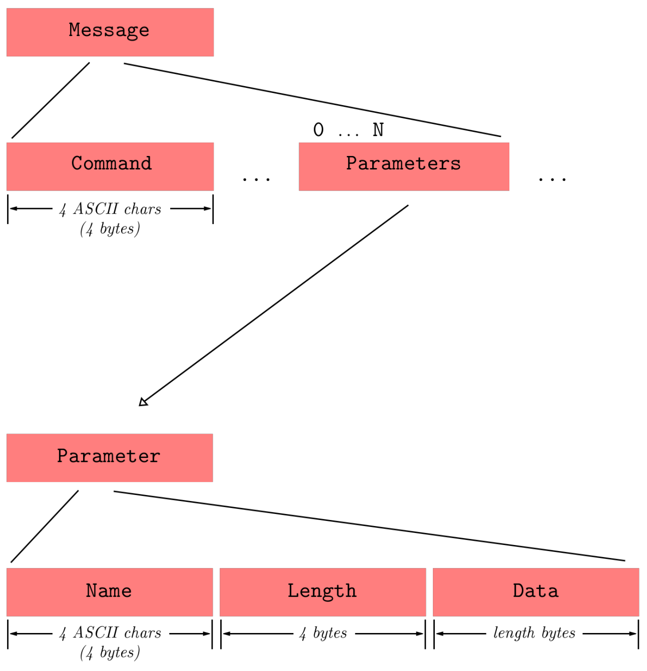


Figure 3.3: Up layer's message structure

3.3 Image changes distribution

Every time, up to 500 ms after somebody makes a change into the graphics the change is distributed to the other clients via the server. The protocol has support for sending only a rectangle cut-out of the edited image which is applied on the same position on the other sides of the connection. The change is distributed as the difference from the original image as an add or erase update. An add update is applied by a standard painting algorithm source over (the weighted average of color with regard to alpha channel). A remove update is applied by erasing (making more transparent) as much as how opaque the update is (see equation 3.1).

$$r_a = o_a \cdot (1 - u_a) \tag{3.1}$$

where r_a is the value of the alpha channel of the resulting image, o_a is the alpha value of the original image and u_a is the value of alpha of the update image. See the illustration of the update process on figure 3.4.

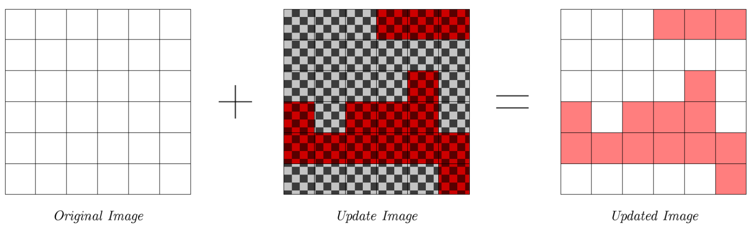


Figure 3.4: Update application example

3.4 Commands

The following is the list of commands specified by CRPP protocol.

3.4.1 Outgoing commands

- GCIN – get connection info – a request for connection info
- AUTN – authenticate – authenticates the client
- GRLI – get rooms list – a request for a list of the rooms on the server
- CROM – create room – a request for a new room
- OJRO – join to room outgoing – a request to connect the client to a particular room
- ODRO – disconnect from room outgoing – to disconnect the client from the room
- ALAY – add layer – a request for a new layer in a canvas

- RLAY – remove layer – a request to remove a layer
- SLAL – set layer location – request for moving a layer to another position (affects the order of the layers)
- ACAN – add canvas – a request for a new canvas in the room
- RCAN – remove canvas – a request for deleting canvas in room
- HTIM – make HTTP image – a request for an HTTP accessible snapshot of a canvas
- OCHA – chat message outgoing – sends a chat message

3.4.2 Incoming commands

- SINF – server info – information about the server
- CINF – connection info – info about the connection state
- SSUC – client connection success – information that a request affecting the connection (e.g. authentication) has been successful
- SERR – client connection problem – information about a problem with the connection (e.g. with authentication)
- RLIS – rooms list – a list of rooms on the server
- IJRO – join to room incoming – the client has been connected to the room
- IDRO – disconnect from room incoming – the client has been disconnected from the room
- ULIS – users list – list of clients (users) in room
- LORD – layers order – the new order of layers
- SRES – set resolution – the new resolution of a canvas
- ICHA – chat message incoming

3.4.3 Duplex commands

- SNIC – set nick – set the nick of the client (user)
- PANT – paint – information about a particular layer update
- SLAN – set layer name

3.5 Paint command

Paint command has been chosen as an example of a message. It carries information about where, which and how an image change should be applied. As has been stated before, an update has one of two types, add and erase.

An update has to be identified by an ID in order to make the clients able to remove it from temporary memory. It solves the problem of latency by giving the clients the possibility to store unreceived changes in a temporary memory which causes the user to see changes immediately.

Then it carries the ID of the canvas and the layer it's been changed from. Coordinates indicating the exact area where the update should be applied and the update data itself.

- PANT – paint
  - UDTY – update type
  - UDID – update ID
  - LYID – layer ID
  - CNID – canvas ID
  - XCOR – X coordinate
  - YCOR – Y coordinate
  - UIMG – update image

Chapter 4

Collab Canvas

Collab Canvas is a general Java library for raster graphical editors. It provides an environment for both local and collaborative painting. It works with 32 bit color depth (RGBA) and layers.

Collab Canvas could be used in a SWING application as it provides JComponnent.

4.1 Interface

All Collab Canvas functionality is hidden behind general interfaces so any future modification to the functionality will not break backward compatibility. The code structure can be seen on figure 4.1.

Zoomable is an interface for zooming the canvas in and out. It provides a function for setting zoom absolutely, relatively and for transforming coordinates from the original zoom to the current zoom and vice-versa.

Visible is an interface for working with cursor related things. The canvas is able to display three types of graphics which move simultaneously with the mouse cursor. They are the mouse cursor itself, the tool cursor and the tool image. All of the three could be set via the interface.

The mouse cursor is a small image representing the mouse pointer as commonly used in other software.

The tool cursor is an image moving with mouse cursor in it's proximity. The purpose of this cursor is to represent the currently used painting tool.

The tool image is graphics illustrating what will be painted to the canvas if the tool is applied. It could be a big image and it is scaled with the canvas.

Selectionable is an interface working with selection. It enables the retrieval of the current selection, selecting the whole canvas area or its parts.

Collab Canvas works with general, pixel-precise and semi-transparent selections. A selection is represented by a raster of pixels which are mapped to the painting image pixels.

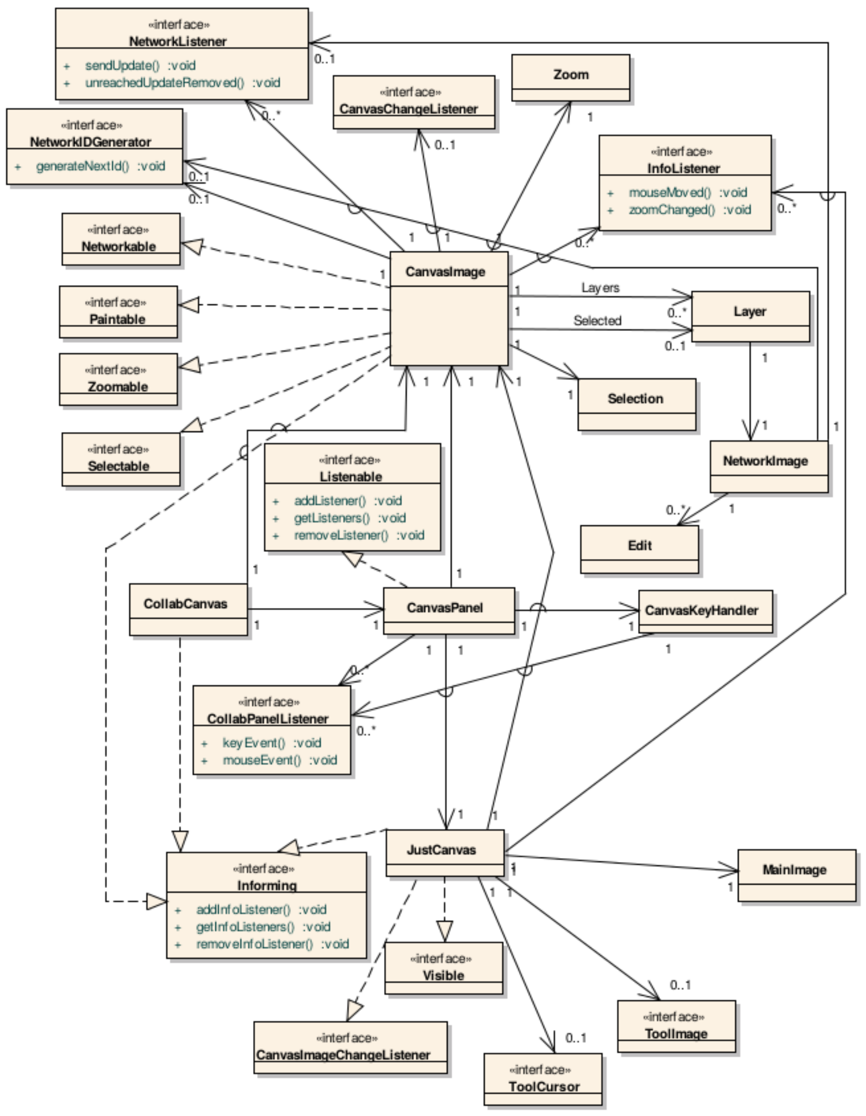


Figure 4.1: Collab Canvas class diagram

Each pixel carries a value between 0 to 255 where one end represents the full selection and the other no selection. Therefore each pixel could be selected with an independent amount of transparency. When a paint or erase update is applied it is applied only to the selected pixels and only to the level of selection opacity.

Paintable is an interface working with image data itself. It works with layers, so it enables the addition, removal and sorting of them. It could set a layer opacity or transparency and get the value. It works with the image so it provides a function for retrieving rectangular cut-off, selection cut-off and/or the whole image from the selected layer only or all layers rendered together. It gives a function for getting and setting a canvas resolution. And it has a function for painting and erasing.

The most important function, that is the paint function, gets one parameter and it is paint data. Paint data is the object compounded from an array of paint images. Paint image has a type which could be add or erase, apply points (an array of two-dimensional points upon which the image will be applied) and image data.

Networkable is an interface which connects the canvas with the outside world. It has a function for adding and removing network listeners (for outgoing updates) and a function for informing about and providing a network (image) update.

A network update is the object of its ID, layer ID, canvas ID, type (add or erase), coordinates and image data.

Informing is an interface binding listeners on changes in mouse position and zoom.

4.2 Usage

Elementary use of Collab Canvas can be seen on the following code example.

```java
// create new canvas with new ID
canvas = CollabCanvasFactory.createNetworkCollabCanvas(
    new NetworkIDGenerator() {

        protected int nextId = 1;

        @Override
        public int generateNextID() {
            return nextId++;
        }
    }, ++lastID);

// listen for user events over canvas
canvas.getListenable().addListener(this);
// sets canvas paintable area size to 300x200
canvas.getPaintable().setResolution(300, 200);
// create new layer in canvas
int layerID = canvas.getPaintable().addLayer();
// select created layer
canvas.getPaintable().selectLayer(layerID);
// set cursor of current "tool"
canvas.getVisible().setToolCursor(generateToolCursor());
// set image of current "tool"
```

```
canvas.getVisible().setToolImage(generateToolImage());
// add canvas to frame
frame.getContentPane().add(canvas.getCanvasComponent(),
        BorderLayout.CENTER);
```

4.3 Network painting

When a user draws something on the canvas it needs to be distributed to other users soon.

Because other collaborating clients could paint simultaneously a dis-synchronization could occur. It is caused by the effect of a switched order of the changes among users. The user who paints something gets his changes immediately but the changes from other users could come with a delay and vice-versa.

So in order to keep all the users synchronized precisely there has to be an authority, that is the server. Therefore Collab Canvas respects incoming data as valid and superior to it's own. But waiting for updates from the server is not a plausible solution because users need to see what they have drawn immediately. Collab Canvas uses temporal painting memory to overcome this problem.

So when a user paints something it is initially stored to the temporal memory, that is an overlaying layer. Ever few dozen milliseconds the Canvas cuts-off the memory into square blocks, each identified by an ID. Then these data are sent to the listening (network) interface and then the delay comes into play. Over the listening interface there is the authority which processes the data and responds back with an update. When an update is received it is applied to the permanent graphics memory and removed from the temporary one with respect to the ID.

In summary the graphical data are split into three layers. These layers are the permanent layer, temporary layer and temporary block layer. The permanent layer stores final, synchronized graphics. The temporary layer stores recently painted graphics. And temporary block layers store graphics which were painted recently but have not been acquired from an authority yet.

Chapter 5

Collab Server

The Collab Server provides interconnectedness among users. It governs information exchanges by acquiring data and deciding which clients should be informed of the change.

The main purpose of the server is to empower clients to share graphics but it integrates additional features such as painting rooms (the digital analogue of an atelier), textual chat and taking screen shots which are then accessible from web browsers via HTTP as pictures.

The server is a compound of four parts, which are core, network layer, painting layer and HTTP server.

5.1 Core

The server's core is responsible for loading the configuration, from a file or command line, and starting server. It is separated from the rest of implementation so it is potentially capable of restarting the server or serving as a watchdog.

5.2 Network layer

The network layer is based on the Collab Network library and it serves as a mediator between computer network (clients behind it) and painting layer. Every piece of information is translated into a coherent form for the opposite side and passed there immediately. For more info about network communication see Network protocol (page 5).

5.3 Painting layer

Painting layer is the most important part of the server. It is responsible for all the logical functions of the server. It connects clients, works with rooms, routes chat messages, composes room images and so on. It stores info about all entities present and processes interactions among them. All the information is stored in memory.

In a room the server continually composes images into layers as clients do. Continuous image processing requires more computational power but has lower memory requirements.

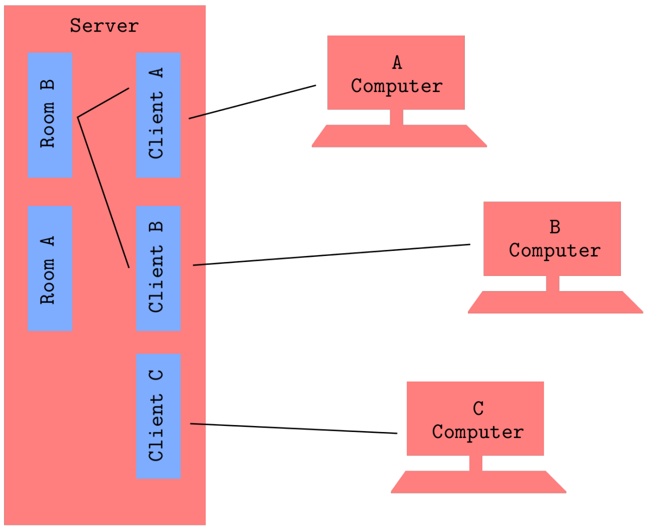


Figure 5.1: Rooms on a server

It is impossible to decide which data have to be stored without any processing which would cause the server to store everything. The fact that there is a lower amount of image data to save is very important because the image data are large in size. Continuous composing makes the server capable of sending the newest version of painted layers to newly connected clients with no need for history or additional computation. Because the server stores already composed images instead of painting history it is able to generate a screen-shot instantly, sending a lower amount of data to new clients and to save the image without extra compilation.

5.4 HTTP Server

Collab Server integrates the feature of taking web-accessible screen-shots of a canvas. It is done through an integrated HTTP server. Whole Collab Server is monolithic where HTTP part is based on Jetty[1]. The feature can be disabled in server configuration.

HTTP server provides a simple HTML page with some information about Collab on root path, not-found page for non-existing URLs and PNG images on specially generated paths. Each image path is compiled from randomly generated string which is coded by SHA-256[2] hash algorithm and then transformed to text by Base64[3].

Screen-shots are stored only in memory and the number of them is limited in server configuration.

[1]Official web of Jetty is http://www.eclipse.org/jetty/

[2]SHS hashes are specified here http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf

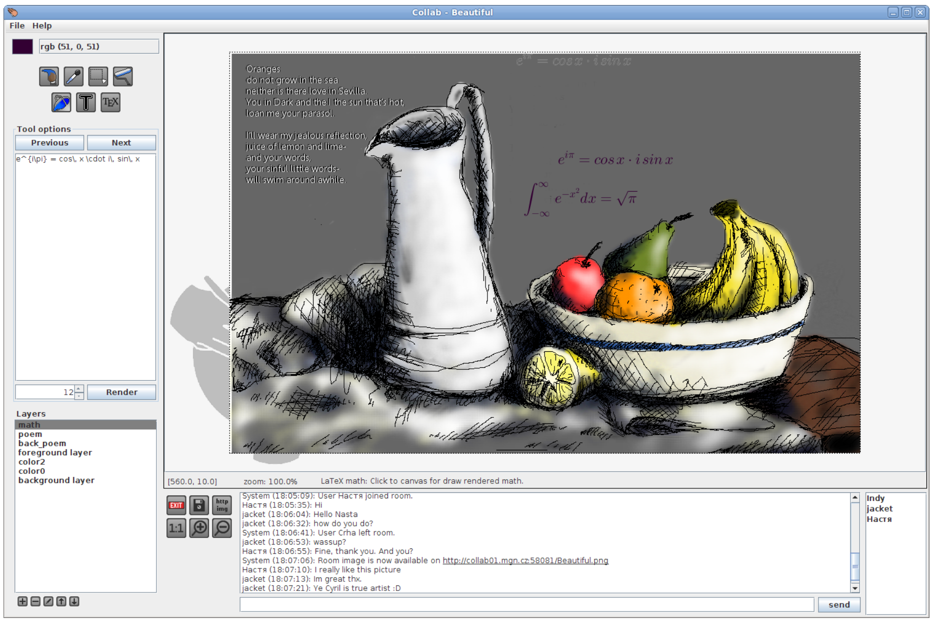
[3]Base64 documentation is available here https://tools.ietf.org/html/rfc4648

Chapter 6

Collab Desktop

Collab Desktop is the actual graphical editor. It is the most visible part of the project. Most Collab users will not learn or directly encounter any other part of the project.

There is a strong emphasis on Collab Desktop's UI, because it is often encountered by non-technical users and because its position as a front liner of the project. See screenshot 6.1.



Oranges
do not grow in the sea
neither is there love in Sevilla.
You in Dark and the I the sun that's hot,
loan me your parasol.

I'll wear my jealous reflection,
juice of lemon and lime-
and your words,
your sinful little words-
will swim around awhile.

$e^{i\pi} = cos\, x \cdot i\, sin\, x$

$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$

```
System (18:05:09): User Настя joined room.
Настя (18:05:35): Hi
jacket (18:06:04): Hello Nasta
jacket (18:06:32): how do you do?
System (18:06:41): User Crha left room.
jacket (18:06:53): wassup?
Настя (18:06:55): Fine, thank you. And you?
System (18:07:06): Room image is now available on http://collab01.mgn.cz:58081/Beautiful.png
Настя (18:07:10): I really like this picture
jacket (18:07:13): Im great thx.
jacket (18:07:21): Ye Cyril is true artist :D
```

Figure 6.1: Screenshot of Collab Desktop

The application guides the user from connecting to a server via room selection or creation to actual communication among users, painting and image persisting.

6.1 Painting Tools

A crucial part of Collab Desktop is called Paint Engine which is an internal interface for the general functionality of painting on Collab Canvas and multiple painting tools are connected to it. The implemented painting tools are listed below.

- Brush – lines painting
- Clearer – erase or repaint a whole layer
- Paint Bucket – fill a bordered area with a color
- Pipette – take a color from a point in a layer
- Selection – select a particular area of a canvas and disable modifications outside of the area
- Text – write a text in a layer
- TeX– render and draw a mathematical formula using TeXsyntax

Every painting tool is configurable so the user can define it's features (e.g. brush thickness, jitter or opacity).

Paint Engine is designed in a way which enables the unlimited addition of more tools with a variety of features. The following is code generating brush lines.

```java
public Brush.PaintBrush paintLine(int x1, int y1, int x2, int y2) {
    x1 -= paintImage.getWidth() / 2;
    x2 -= paintImage.getWidth() / 2;
    y1 -= paintImage.getHeight() / 2;
    y2 -= paintImage.getHeight() / 2;


    Brush.PaintBrush paintBrush = new Brush.PaintBrush(paintImage);
    int size = Math.max(paintImage.getWidth(), paintImage.getHeight());

    float dx = x2 - x1;
    float dy = y2 - y1;
    float steps = (float) Math.sqrt(dx * dx + dy * dy) / (step * scale);
    dx /= steps;
    dy /= steps;

    float x = x1;
    float y = y1;
    Random random = null;
    if (jitter != 0) {
        random = new Random();
    }

    for (int i = 0; i <= steps; i++) {
        float xr = x;
        float yr = y;
        if (random != null) {
            float localJitter = jitter * (float) size * random.nextFloat();
            float adx = ((random.nextFloat() * 2f) - 1f) * localJitter;
            float ady = (float) Math.sqrt(localJitter * localJitter - adx * adx);
            if (random.nextBoolean()) {
                ady *= -1;
            }
            xr += adx;
            yr += ady;
        }
        paintBrush.addPoint((int) xr, (int) yr);
        x += dx;
        y += dy;
    }

    return paintBrush;
}
```

6.2 Configuration

Collab Desktop is configurable so the user can change and save properties such as the default server address and port, default canvas dimensions, etc. The configuration is stored in the user's home directory as an XML file.

Chapter 7

Development

7.1 Open-source

All parts of Collab project are licensed as open source and/or free software. Most parts are distributed under *GNU GPL 3* licence but several other open-source or free software licences were implemented.

7.2 Used development supporting tools

Versioning The Collab project is large in size, it has over 20 000 lines of code. It is free software so other participants are anticipated. For those reasons there was need for versioning.

Git is a new and advanced distributed versioning system. Git is widely known and supported and is very commonly used for open-source projects. There are several free Git hosting, some of them with advanced features such as GitHub.

GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub offers both paid plans for private repositories, and free accounts for open source projects [Git15].

Git is distributed therefore easy migration is possible.

For previous reasons *Git* was chosen as a versioning system and is hosted on *GitHub*.

Issue tracking every non-trivial software project has its bugs and planned features. Because of that an issue tracking system was implemented into the development. Requirements for the system were in basic features, that is to store tickets, comments, users, ticket states and projects. Flyspray is used as the issue tracking system within The Colab project.

Flyspray is an uncomplicated, web-based bug tracking system written in PHP for assisting with software development [Tea12]. It is easy to install on most servers because it is based on common technologies.

Wiki is standard for open-source projects. It functions as documentation, a place for know-how and as a persistent communication canal. DokuWiki is used for its stability and easy maintenance.

DokuWiki is a simple to use and highly versatile Open Source wiki software that doesn't require a database [Tea14].

Chapter 8

Conclusion

The bachelor thesis was written to describe the possibility, limitations and creation of a collaborative graphical editor.

Collaborative graphical editor is a compound of a vast variety of technologies on multiple layers. There are limitations to Internet connectivity, processing power and memory capacity. The solution has to take into account all the technical and practical limitations, feature requirements of users in both unprofessional and professional environments and human habits and perception. The solution therefore has to be complex, rich and use some advanced techniques, designs and algorithms.

Most of the problems of the project were solvable with enough dedication but some of the most difficult required very intense work and a long time to be solved. The work would be smoother with more developers and graphical designers. But regardless of all obstacles I have encountered I consider the results as satisfying.

The main limitation of the Collab project as it is in it's current version is that it has only a desktop client. It would bring more users and comfort if there was an HTML5 client as a perhaps limited substitute for a full featured desktop version. Smartphone versions would bring even more portability and usefulness.

Collab is being developed as free-software. The development will continue with a focus on community around free-software, which include works on more detailed documentation, guides for beginners in the project and improvements into the infrastructure. If more developers took part in the development it could grow into a very advanced, multiplatform and widespread collaborative painting tool.

Technologies such as Collab are still not widely known and used but that can be changed by continued and scaled development. Collab has the potential of spreading a whole new method of digital human to human interaction and even socialization.

Source codes of the project have over 20 000 lines of code spread over five separate Git repositories. There are several hundreds of closed tickets in Collab's issue tracking system.

Bibliography

[con14a] Wikipedia contributors. Image file formats, 2014.

[con14b] Wikipedia contributors. Paint chat, 2014.

[Cos14] Cosketch.com. Cosketch, 2014.

[fS04] International Organization for Standardization. Iso/iec 15948:2004, 2004.

[Git15] GitHub, Inc. About github, 2015.

[Meg14] MegaScopes.com. Whiteboard, 2014.

[Por14] Portalgraphics. opencanvas, 2014.

[Tea12] Flyspray Development Team. Flyspray – the bug killer!, 2012.

[Tea14] DokuWiki Development Team. Dokuwiki, 2014.

[Tec14] Internet Techies. What is average broadband speed in your country? answer is here, 2014.