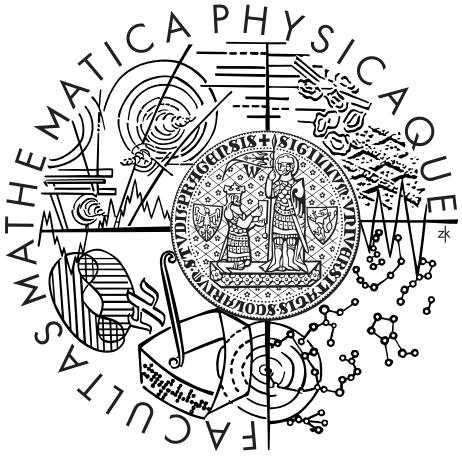
Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Šimon Rozsíval

Vector Screencast

Department of Distributed and Dependable Systems

Supervisor of the bachelor thesis: Mgr. Martin Děcký
Study programme: Computer science
Specialization: Programming and software systems

Prague 2015

Dedication.

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Vektorový screencast

Autor: Šimon Rozsíval

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Martin Děcký

Abstrakt: Cílem bakalářské práce je vytvořit software pro záznam a přehrávání výukových videí pro potřeby Khanovy školy. Na rozdíl od běžných videí nejsou obrazová data uložena ve formě bitmap, ale jako vektory, což umožní snížit datovou náročnost a vykreslit obraz ostře při libovolně velkém rozlišení obrazovky uživatele. Přehrávač videa i nástroj pro nahrávání běží ve webovém prohlížeči. Součástí práce je také návrh a implementace vhodného formátu pro uchovávání obrazových a zvukových dat a implementace v softwarové architektuře klient/server.

Klíčová slova: screencast, vektory, video, on-line

Title: Vector Screencast

Author: Šimon Rozsíval

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Martin Děcký

Abstract: The goal of this bachelor thesis is to create a software for recording and playback of educational videos for Khanova škola (Czech clone of Khan Academy). Contrary to common videos the visual data is not stored as a sequence of bitmaps, but as vectors. This allows to reduce the data bandwidth and playback sharp images in any target resolution. The player and also the tool for recording the videos runs in a web browser. The thesis also focuses on designing and implementing a suitable file format for storing the visual and audio data and implementing the software according to the client/server paradigm.

Keywords: screencast, vector, video, on-line

Contents

Introduction

Each and every person on earth explores the world from the day of and continues to learn new things all his life. Education in the so-called developed world is essential for later employment.

@todo

Khan Academy

Khan Academy is an online tool providing free access to instructional videos and exercises covering various subjects including math, history, programming, economics, and more.

@todo

Screencast A screencast is a video created by recording computer screen output, often accompanied with an audio commentary. This process can be used in many different ways, for example to record a tutorial explaining how to use a specific computer program. The quality of the recorded video depends mainly on the resolution of the user's device resolution and the recorded area of the screen. Khan Academy uses screen capturing tools to record the virtual canvas of a bitmap editor, onto which the author draws using the tools of the editor and talks about the covered subject.

Vector Screencast project

A year ago two members of "Khanova škola", the Czech brach of the Khan Academy, came with an idea of improving the current technical solution of recording and displayling educational videos on their website. It turns out that some of the videos recorded only a few years ago don't look very well or the hand-drawn text in these videos is not even legible on large displays. The other extreme are small displays of tablets and smartphones, where the downscaled letters are too small to read and can't be zoomed well in most video players. Their idea was to create a vector-based animation instead of classical bitmap-based screencasts. This animation could be scaled to any display resolution without any loss of information. The process of recording can be simulated in any distant future based on the recorded data of user's behaviour. The animation can be rendered using a different algorithm and the author would not have to record his screen again. As a result, the video will never become obsolete because of its poor visual quality.

One of the other reasons for this type of solution was the possible decrease of the size of data transfered over the Internet, as most of the image does not change between each two frames and very often the image doesn't change at all. The file size of a vector-based format does not correspond to the quality of the video. In regular bitmap formats, the higher the resolution of an image is, the the larger the data file is. Vector-based format has only one version for every display resolution.

Thesis structure

@todo

1. Distance education

Distance education is not only a phenomenon of the last few decades, but can be traced at least back to the 18th century, when Caleb Phillipps posted an advertisment called "Teacher of the New Method of Short Hand" in Boston Gazette, saying "Persons in the Country desirous to Learn this Art, may by having the several Lessons sent weekly to them, be as perfectly instructed as those that live in Boston." [1]

With the development of the Internet and its general accessibility in the developed world, providing distance education has become much easier and has spread widely. In some countries tuition rates are high and young people take loans so they can affoard to study. This topic is covered in a fitting way by John Oliver in his show [2]. The flexibility and low cost of distance education over the Internet gives people, who wouldn't be otherwise able to attend a traditional university, an opportunity to gain knowledge and train skills from their homes [3] spending much less money or even for free.

Students in the developing world are also taking the advantage of educational content available on the Internet. Several of the top U.S. universities, like Harvard, Stanford or MIT, put some of their materials on so called MOOCs (Massive Open Online Course) like Coursera, edX or Udacity. This content is then available to anyone with a computer and Internet connection and knowledge of the language. A great example is *Kepler* - a non profit university project in Rwanda [5]. The goal of this project is to "provide an American-accredited degree, a world class education, and a clear path to good jobs for thousands of students for around $1,000 tuition per year." [6]

A concept of teaching often reffered to as *Flipped Classroom* uses distance education. The idea is to let students study lecture materials at home at their own pace and then apply gained knowledge at school the next day by doing activities to illustrate the concepts. The teacher can then help them or explain details in person. These materials are often in the form of videos either from an open source, like the Khan Academy, or created by the teachers themselves.

1.1 Current systems

In the next few paragraphs I will try to pick some of current distance education services and tools available on the Internet. This list is not complete and is only meant to give the reader a notion of available technologies and their paradigms, on which the project of Vector Screencast is based.

1.1.1 Coursera

The mission of *Coursera* is to "provide universal access to the world's best education." [9] Anyone can, for free, go through materials published by universities and other organizations aimed at education.

Courses at Coursera consist mainly of video lectures commonly with a transcript and an attached presentation document. These videos can be viewed directly in web browser on demand or downloaded to user's computer. After study-

ing the materials, students can submit assignments' solutions and take quizzes and receive a Verified Certificate for the accomplishment of the course. These certificates are not for free.

Most of the courses are in English and only a few of the courses are also translated into other languages. It is not possible for everyone to publish his materials through Coursera.

1.1.2 Youtube.com

Youtube.com [10] is not an educational service by design. Youtube allows people to create their own channels and upload their video content to share it with other Internet users for free.

Youtube was launched in 2005 and has became one of the most frequently visited websites on the Internet according to Alexa Internet [11]. Uploading video to Youtube is free, advertisment is displayed to the user while watching videos though.

The ease of making original-created videos available for wide audience makes Youtube a perfect place for all individuals and organizations, who want to share their ideas or any video materials. Many educational channels can be found here, for example *Numberphile* [12], *Veritasium* [**?**], and *Khan Academy* [14].

Youtube videos can be viewed only online in a web browser or a specialized mobile app. There are only unofficial tools for downloading these videos.

The form and content of the video is practicaly unlimited, as long as it does not violate the terms of the service. The maximum file size of a video is 128 GB in size and 11 hours in length. To upload larger or longer videos, user must split them into several parts. Video can have a text description which might contain the transcription of the video content and any subtitles can be attached to a video. Youtube also downscales videos to multiple resolutions so they can be viewed with a low speed Internet connection, for the cost of reduced quality.

1.1.3 Moodle

Moodle [7] is an open-source project used by millions of users [8] providing a robust platform providing custom learning materials to students. The source code of Moodle can be downloaded and deployed on any server. Teachers use Moodle mainly for publishing study materials and assigning homework.

1.1.4 Educreations and ShowMe

Educreations is a service for creating and sharing educational videos similar to Khan Academy videos. The idea is that teachers create their own videos by drawing onto a digital whiteboard and then share these videos with their classes.

Educreations is designed for an iPad, but can be used also as a web application from any web browser, which supports Adobe Flash Player. Web application supports playing and recording of videos and the video player can be easily embedded into any website and watched without registration. Content of the video is scaled appropriately to the output screen and the lines are drawn smoothly.

The iPad app is free and users download it through the Apple AppStore. Users must create an Educreations account to use the app. There is a free usage

Figure 1.1: Khan Academy lesson

plan, which allows users to watch, create, and share videos. The free plan allows only a limited use of some of the features, some are not even unavailable. For more storage capacity for user recorded videos, and advanced tools, users must pay monthly fees. The software is proprietary and so is the file format of the video.

ShowMe[1] is a very similar service. An iPad app is available for recording and viewing videos. Recorded videos be played in a web browser, but not recorded. Pricing is very similar to Educreations.

Both *ShowMe* and *Educreations* target mainly tablets users. The advantage of tablets is their touchscreen, which can be used for drawing in a natural way with fingers or a special stylus.

1.1.5 Khan Academy

A similar service to Coursera is *Khan Academy*. The idea of Khan Academy originated in 2003 when Salman Khan began tutoring his cousin over an instant messenger via drawing pictures with a computer mouse. Salman then started to record these videos and put them on his Youtube channel, so someone could watch them later. This channel became the basis of Khan Academy.

Khan Academy became known and has grown a lot, but the style of Khan Academy videos remained the same. A person draws lines and diagrams using a bitmap editor on his computer and talks about the subject aloud while recording his computer screen and recording his voice using a microphone. This style is sometimes called the "Khan-style video" (KSV) [2]. These videos are then uploaded to Youtube and embedded in the Khan Academy website [15].

Apart from the video lectures, the website also contains exercises and quizzes to encourage students in learning. The pace of the lesson depends on the student. He can pause the videos or watch them multiple times before continuing with the lesson. An example of Khan Academy lesson is shown in figure 1.1.

Most of the videos are recorded in English, but many of the videos are translated into other languages - by replacing the audio track with a different one or with subtitles. One of the projects working on the localization of Khan Academy videos is an official Czech branch called Khanova Škola[16].

[1] http://www.showme.com/

[2] https://www.youtube.com/watch?v=Ohu-5sVux28

2. The Vector Screencast project

The overall project consists of two separate tools — the *recorder* and the *player*. Each of these tools behaves differently and will be used by different people. While the video player will be used by general audience, the recorder will be used by a much narrower group of content creators.

Thanks to the sparse nature of vectors, in comparison to bitmaps, data consumption might also be reduced or at least similar. This animation would also be linked to an audio track of authors voice commentary forming a complex KSV suitable for educational purposes.

2.1 Screencast recording tool requirements

User, who wants to create a new screencast, should enter a website in his web browser and without installing any additional software start recording a video using his mouse, touchscreen or digital stylus and a microphone. This video should then be uploaded to a server.

If the user uses a pressure-sensitive stylus, then the applied pressure during drawing should be recorded too. The more pressure the user applies, the thicker the line will be and vice versa. Using this specific hardware might require additional drivers or specific software installed.

Different brush sizes, brush colors and background colors are available to the user. User must be able to erase certain parts of the canvas or the whole canvas at once. User must be able to pause and continue recording at any time.

Lines are immediately rendered on the screen as the user draws them, so he sees exactly the same output as the viewer will, when he is playing the video later. All the raw data collected from the user should be stored, including the data that have no effect for video playback, like recording cursor movement while recording is paused. This will allow the user to simulate the process of video recording and use it for further post-processing in any distant future. Recording of this extra information should be optional.

2.2 Screencast player requirements

Any user should be able to play vector video on-line in all major modern web browsers without any special software or plugin installed, including mobile browsers.

Video should be scaled appropriately to the size of the player and device's screen resolution. The lines should have the same shape they had as the author saw them while he was recording the screencast.

User can pause and continue with the playback of the video at any time. User can skip to any point of the video either forward or backward.

If the author of the video had recorded his voice, it will be played along with the video. Audio must be synchronized with the video whenever user plays or pauses the video or when he skips to a different point of the timeline.

User interface should be intuitive and easy to use either on a desktop computer using mouse and keyboard, but also with touchscreens on mobile devices.

2.3 Goal of the thesis

The result of this thesis should be an open-source library suitable for extending any web application with the abilities of recording and playing KSVs in all modern web browsers. An appropriate vector-based file format should be chosen or defined to store screencast data.

The library should be easily adjustable and configurable for different purposes. User interface should be fully translatable.

3. Analysis

3.1 Available techonologies

Web is a huge and fast growing environment. In only a few years, it has become a universal place for presenting and exchanging information. This put web in the focus of many software companies and organizations and as a result, many different technologies for developing *Rich Internet Applications* (RIA) have been created. Some of them have already faded into obscurity, other are just emerging. One of the main limitations in the selection of the right technology for developing web application is their compatibility with different operating systems and web browsers.

3.1.1 Java applets

Java applets are used for creating interactive applications within web browsers [18]. Java applets meet all the specified technical requirements of both player and recording tool.

Java applets are written in any language, that can be compiled into bytecode, this bytecode is then downloaded to the web browser and then runs using Java Virtual Machine (JVM). This means that to be able to run a Java applet, user needs to have JVM installed on the device and an installed and allowed Java plugin in his web browser. There is no support for mobile operating systems such as iOS and Android [19] and Google Chrome has Java disabled by default in most recent versions for security reasons.

3.1.2 Adobe Flash

Adobe Flash is a multimedia and software platform used for creating vector graphics, animations and games. Flash has all required features: vector graphics manipulation, working with Extensible Markup Language (XML)[1], mouse input capturing, microphone input, and audio streaming [20].

To view Flash animations or to execute Flash applications, Adobe Flash Player is needed. Adobe Flash Player is available and being developed for all major operating systems, although that is not true for mobile platforms. There was never any support for Apple iOS [21] and in 2012 development of Flash for Android was discontinued [**?**]. Using Adobe Flash would mean to exclude most users of tablets and smartphones [24], which is a large disadvantage of this technology.

3.1.3 Microsoft Silverlight

Microsoft Silverlight [25] can be used for creating web applications. It is based on the .NET Framework and it is similar to Java applets and Adobe Flash. It was Microsoft's attempt to compete Adobe Flash, but has not gained much popularity. Development of Silverlight was also discontinued by Microsoft in 2012 and it will

[1] http://www.w3.org/XML/

not be supported in Microsoft's new web browser Microsoft Edge[26]. These facts make Silverlight unsuitable for this project.

3.1.4 HTML5

HTML5 is the fifth revision of *HyperText Markup Language* (HTML) standard by the *World Wide Web Consortium* (W3C). It has been given the Recommendation status in the end of 2014 and all crutial aspects of both tools can be implemented using the proposed standard. Tracking mouse was long supported even in HTML and ECMAScript/JavaScript[2] through *Document Object Model* (DOM) Events[31].

Vector graphics are supported through the *Scalable Vector Graphics* (SVG) format [29] and it can be manipulated through DOM API [30], as well as any other XML content. MediaStream API [] enables access to audio input from user's microphone. The `<audio>` tag can be used to stream audio files and play them in a web browser.

While HTML5 is a new standard, many of the most important features have already been implemented in some web browsers, like Google Chrome and Mozilla Firefox. More specifically, all the features needed to implement screencast player are supported in the latest versions of all major web browsers. The only catch might be disagreement on supported audio file formats among the creators of web browsers. This can be overcome by converting the audio into several used formats and passing them to the browser, which will then choose the one it supports.

Some of the features needed by the recording tool, like the MediaStream API, are not supported by all web browsers yet, but this won't be an issue for content creators, who can easily install a supported web browser on all desktop platforms and even some mobile ones. Fast development in this area is expected and these features will be most likely soon implemented in the next versions of all major browsers.

Scalable Vector Graphics (SVG) and Synchronized Multimedia Integration Language (SMIL)

SVG SVG is an XML based file format designed for describing two-dimensional vector images[29]. It is an open format developed and maintained by the W3C SVG Working Group []. Current W3C Recommendation is SVG 1.1 (Second Edition).

SMIL *SMIL* is an estabelished standard for animation of SVG images using a declarative approach. It is an extension of SVG specification that lets the user to define key-frame animations of SVG elements' attributes. SMIL is not supported by all major browsers[3]. SMIL's future is unsure at the moment, as Blink rendering engine development team has announced to deprecate the technology in Chromium and Google Chrome in favor of CSS animations and the *Web Animation API* [] [4].

[2]ECMAScript is familiarly known as JavaScript after it's most common implementation[28]
[3]https://status.modern.ie/svgsmilanimation
[4]https://www.chromestatus.com/features/5371475380928512

Web Animation API

A very promising technology is the *Web Animation API* [5], which will provide a very effective way of declaring and controlling animations in a web page through a JavaScript API. This technology is currently (3rd July 2015) in the state of Editor's Draft. This means that it is provided for discussion only and may change at any moment [] [6].

Unfortunatelly, at the time of writing this thesis, is not supported by several major browsers[7][8] and is only partially supported in the remaining few [9][10].

3.1.5 Conclusion

When looking at table 3.1, the bottleneck of most of the technologies is their support in mobile devices. These devices don't allow some of the above mentioned technologies to run inside them. As the popularity and market share of mobile devices grow, supporting them is a high priority. HTML5 APIs features needed to create the screencast player are implemented all major web browsers, including browsers for mobile devices. Audio recording is not supported by all major browsers at the moment, but this will change with the release of Microsoft Edge [27]. After considering these facts, HTML5 was selected to implement both the recording tool and the screencast player.

3.1.6 Possible issues and known limitations

As mentioned earlier, not all browsers implement all features described in the HTML5 specification in their latest versions. But the trend is to implement as many technologies as possible to bring better browsing experience to users to gain bigger market share, so this situation will get better soon.

Another fact, that might cause trouble, is that many users still use an old version of their browser. This is mainly the case of Microsoft Internet Explorer (MSIE), which does not have automatic update mechanism. Fortunatelly, the number of MSIE users is dropping and Microsoft Edge, the ancestor of MSIE, will have automatic updates built in (@todo: some relevant source).

So far, there hasn't been consensus on formats and codecs support in HTML5 *WebAudio API*. The most supported format across web browsers is MP3. This format isn't supported by some browsers on some platforms due to licence reasons. An up-to-date table with media formats' support can be found at *Mozzila Developer Network* (MDN) website [] [11]. This inconvenience can be easily solved by providing several `<source>` elements to the `<audio>` element with different formats. Browser will then select the format it supports and it will download and play this audio track. The flaw is the need of having the data stored in several formats on the server and the video as a whole takes more disk capacity.

[5] http://www.w3.org/TR/web-animations/
[6] http://w3c.github.io/web-animations/
[7] https://status.modern.ie/webanimationsjavascriptapi
[8]
[9] http://caniuse.com/#feat=web-animation
[10] https://birtles.github.io/areweanimatedyet/
[11] https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats

	Adobe Flash	Java applets	Microsoft Silverlight	HTML5
drawing vector primitives	Yes	Yes	Yes	Yes
microphone input capturing	Yes	Yes	Yes	Yes
reading and creating XML documents	Yes	Yes	Yes	Yes
Wacom tablet	Yes	Yes	Yes	Some[12]
player features supported by desktop browsers	Yes	Some [13]	Some [14]	Yes
player features supported by mobile browsers	None	None	None	Yes
recorder features supported by desktop browsers	Yes	Some [13]	Some [14]	Some[15]
recorder features supported by mobile browsers	None	None	None	Some[16]

Table 3.1: Comparison of available technologies

3.2 Drawing algorithm

One of the key aspects of success of this library is the way it draws the lines. The lines must be nice, smooth and feel natural, as if somebody physically drew them on a blackboard with a chalk, and must not be disturbing or distracting. This library cannot improve the contents of the author's video, but the way it is displayed to the user can have possitive impact on viewer's impression of the video. It is also required that stylus pressure has direct proportion to the width of the drawn line.

After a research, the algorithm called "Dynadraw" by Paul Haeberli [**?**] from 1989, seems appropriate for this project. It is for example used in *Inkscape*[17] for the calligraphic tool[18].

This algorithm models the tip of the brush as a physical object and simulates its movement. User applies force on the brush by moving the mouse or any other pointing device. The original algorithm is designed for calligraphic strokes and uses mouse speed to calculate dynamic line widths. This technique makes lines drawn by a mouse or other pressure-insensitive device more interesting and can

[12] Browsers supporting *Wacom WebPAPI* or *Pointer Events API*

[13] Java is disabled by default in Google Chrome

[14] Only some browsers and operating systems are supported.

[15] MS Edge will support *getUserMedia*, Safari does not support *getUserMedia* yet

[16] *getUserMedia* is implemented only in Google Chrome for Android

[17] Inkscape: http://inkscape.org

[18] http://bazaar.launchpad.net/ inkscape.dev/inkscape/trunk/view/head:/src/ui/tools/calligraphic-tool.h

be combined with pressure level of pressure-sensitive device.

For details of the implementation see section 5.4.1 on page 24.

3.3 Audio recording

Audio input from the microphone will have the form of raw uncompressed *Pulse-code modulated* (PCM) data[49]. The sample rate is chosen by the web browser and cannot be changed by the programmer. The sample rate must be between 22.05 to 96 kHz. If we choose the bit depth of 16 bits per sample, then we can calculate the minimum data load of uncompressed audio recording. Therefore we receive at least 44.1 kB of data per second, which 2.646 MB of data per minute of recording. From experience, desktop browsers target 44.1 kHz sampling rate, which even doubles the data load. This data must be uploaded to the server so users can watch the screencasts with voice commentary.

Buffering microphone input The first possible approach is to buffer all data from the microphone, create an uncompressed audio file and upload this file to the server at the end of the recording. Creating a *Waveform Audio File Format* (WAVE or WAV) file is relatively simple because it contains only a simple header and LPCM data.

Compression of buffered data in the browser The uncompressed audio is relatively large and the upload will take much more time, then uploading a compressed file would take. Audio compression is a complex process. There are several JavaScript libraries, that implement audio compression inside a web browser[19][20], but they have shown to be large, slow and unstable.

Streaming microphone data to the server in background A different method is possible with the use of the *WebSocket protocol*. The WebSocket protocol enables two-way communication between a client and a server [51]. It is built on top of *Transmission Control Protocol* (TCP) sockets, which means it provides ordered and reliable transmission of messages. Either textual or binary messages can be sent over a WebSocket. If the data is transfered over the WebSocket at the time of recording in background, then a considerable part of the data might be already uploaded to the server as soon as the user stops recording, depending on his Internet connection upstream speed.

3.3.1 Conclusion

Streaming the recorded audio data in background over a WebSocket seems to be convenient for the user. It will fasten the uploading process at the end of recording, as part of the audio data is already uploaded before the user even clicks the upload button.

[19] http://bgrins.github.io/videoconverter.js/

[20] https://github.com/akrennmair/libmp3lame-js

Server must implement WebSocket connections and serve clients. Server might also use a native program, like the open-source FFmpeg[21] to compress the audio and convert it to different file formats.

3.4 File format

All the information of the cursor movement and the pre-drawn lines must be stored on the server, including the time at which every single segment of every line is drawn. For this purpose an appropriate file format must be used. There are a few existing formats, which were designed for vector graphics and animation.

EVA The *EVA* by SHARP[41] is binary file format developed in 1996. It is used to create vector animations and it is popular in Japan. Unfortunately the documentation and resources are available in Japanese only[42].

SWF Another well-known format is Adobe's *SWF*. It is a binary format with a wide range of usage. It can contain bitmap-based video, vector primitives, and ActionScript[43]. It is possible to manipulate contained vector primitives and morph their properties to create a key-frame animation[23]. Documentation of the format is open since 2008.

SVG *Scalable Vector Graphics* was already mentioned earlier. This format can contain SMIL animation definitions, which makes it also a vector animation format. SVG is based on XML and it is therefore a human-readable textual format. As discussed earlier, SMIL animations are not considered to be a good option for this project.

Custom extension of SVG JavaScript is not very good at manipulating binary files. Until the specification of typed arrays[22], manipulating binary files in JavaScript was very ineffective and unintuitive. Even with these new technologies, working with XML in JavaScript is more straighforward. The fact, that SVG is based on XML means, that it can be extended with custom namespace and define custom animation, that will be better for this project.

3.4.1 Conclusion

After considering available options, creating a custom format based on SVG seems like the best option for its human-readability, simplicity, versatility, ease of manipulation using JavaScript, and the possibility of previewing static scenes of the animation in existing software. This format will serve well as a proof-of-concept and can be replaced with a more sophisticated format in the future and the format must not be therefore hard-coded into the library.

For the specification of this new format, see chapter 4 on page 16.

[21] https://www.ffmpeg.org

[22] https://www.khronos.org/registry/typedarray/specs/latest/

3.5 Backend

An important part of every website is its backend. Implementation of the backend is not a part of the Vector Screencast library, but this library needs to have compatible processes running on the web server – an *HTTP web server* and *Audio Recording Server*.

3.5.1 HTTP Web Server

An *HTTP web server* is a server process listens for incomming HTTP requests on well-known port 80 or 443 in case of secured connection. The library should be easily integratable into any web server as it has only a few requirements.

Serving static files The server should respond to HTTP GET requests, whose URL corresponds to a file location in the public section of the file system, by serving this file. To enable replaying and skipping parts of audio files, this server must understand the *Content-Range* header of HTTP requests [17]. This is a basic feature of common web servers like *Apache*, *IIS* or *Express.js*. Any of the mentioned servers or a similar one might be used.

Handling upload of the created SVG file When the recording is finsihed, the result needs to be stored on the server. The web server must therefore be able to handle HTTP POST request with the uploaded file and information about its type on a specific URL. It is usually possible to define this behavior in some scripting language supported by the server. Handling file uploads is a typical task in web applications and the process should be well documented.

3.5.2 Audio Recording Server

The Vector Screencast library will try to upload as much audio data it receives from the microphone to the server as possible to fasten the process of uploading at the end of recording. A compatible server process must receive this stream of PCM data and store them into an uncompressed WAV file.

WebSocket specification is relatively new and there might not yet be existing libraries for all programming languages. Audio recording server can be a separate process running on the server and listening on a different port. This program can be written for example in JavaScript and run on multiplatform *Node.js*. There are several WebSocket packages available for Node.js.

3.5.3 Conclusion

Creating a backend for Vector Screencast is very easy and it can be integrated basically into every existing web application. Administrators of this website must be able to create and run a WebSocket server for audio recording, which will run on their platform.

4. Vector Screencast file format

SVG format is common and is implemented in web browsers and in many programs. These programs can open an SVG image and draw its contents. The idea is to create an SVG file, which contains all the information needed using custom namespace attributes and elements and shows the state of the blackboard at the end of recording.

A valid SVG document must have an *svg* root element with specific namespace attributes and specified *width* and *height* attributes. For the purpose of extending the document, a new namespace *http://www.rozsival.com/2015/vector-screencast* is added with the prefix *a*. An example of an empty SVG document with this namespace looks as follows:

```
1 <?xml version="1.0"?>
2 <svg version="1.1"
3         width="470"
4         height="100"
5         xmlns="http://www.w3.org/2000/svg"
6         xmlns:a="http://www.rozsival.com/2015/
            vector-screencast">
7
8 </svg>
```

SVG specification allows inclusion of elements and attributes from foreign namespaces anywhere with the SVG context. Attributes from foreign namespaces might be attached to any element. Both elements and attributes will be included in the DOM by the SVG user agent, but will be ignored otherwise [**?**].

Vector Screencast `<svg>` root element must have two child elements: `<metadata></metadata` and `<g a:type="chunks"></g>`

SVG `<g>` element[1] is intended for grouping related graphics elements. These groups might be also nested.

4.1 Video Metadata

The first child element of the `<svg>` element must be a `<metadata>` element. This element must contain these child elements:

<a:width> and <a:height> The content is the original width and height of the blackboard. These numbers will be used to correct coordinates when playing the video with different resolution. The `width` and `height` attributes of the `<svg>` element may contain different values.

<a:length> The content is the duration of the video in miliseconds.

<a:audio> Contains the list of audio sources as child elements.

[1] http://www.w3.org/TR/SVG/struct.html#GElement

<a:source> Defines one audio source. It has two attributes: a:src – containing the URL of the audio source, and a:type – containing the MIME type of the audio source.

4.2 Video Chunks

Video is divided into smaller consequent parts, called *chunks*. Chunks have variable time duration based on user's behaviour. These chunks are logical units of the video, each of them can be rendered at once as one graphics primitive. This helps optimize skipping parts of the video. Some chunks can be skipped entirely as the canvas will be cleared at some point after the very chunk is rendered, but before the target time point is reached.

There are three types of chunks: *path*, *erase*, and *void* cuhnk. Each chunk is stored as one SVG group element with a specific a:type attribute (*path*, *erase*, *void*) and an a:t attribute, which is the time, when this chunk starts to be processed, in miliseconds. Chunks contain the prerendered graphics primitive and a list of animation commands.

"Path" This type of chunk represents one line the user draws. The first child element is an SVG <path> element. The fill attribute should correspond to the real color of the path, but is not neccessary for later screencast playing. The data attribute d includes serialized information about the segments of the path in the form of valid SVG path instructions[2]. For more details about the serialization and deserialization of this data, see section @todo on page @todo. One or more child elements might follow after the <path>, all of which must be animation command elements.

"Erase" This type of chunk represents clearing the whole canvas with one color. The first child element is an SVG <rect> element. The fill attribute should correspond to the real color of the new background. One or more child elements might follow after the <rect>, all of which must be animation command elements.

"Void" This type of chunk does not render anything on the canvas. It might contain one or more child elements, all of which must be animation command elements.

4.3 Animation Commands

Animation commands are necessary for correct actions' timing during the video. Chunks contain the information of how the resulting primitive looks like, commands contain user's gradual forming of these primitives.

Animation command must be a child element of a chunk element. Every command must have an a:t attribute with the time of the action in miliseconds. The value of the time attribute of an element must be greater or equal to the values of preceding sibling action elements.

[2]http://www.w3.org/TR/SVG/paths.html#PathData

<a:m> *Move cursor* command tells the player to move cursor to a specific position defined by attributes `a:x` and `a:y`.

<a:c> *Change color* command tells the player to switch current brush color according to the value of `a:c` attribute. Value of the attribute must be a valid CSS color value[3].

<a:s> *Change brush size* command tells the player to change current brush size to the value of `a:w` attribute. The units of this value are pixels and the size must be corrected according to the width and height stated in *metadata*.

<a:d> *Draw next segment* will render the next segment of the current path with current brush color and size. This command doesn't carry any additional information, all the information about the segment is carried by the *path* chunk. *Draw next segment* commands can be present only in a *path* chunk. There must not be any *change color* or *change brush size* command between the first *draw next segment* command and the last one inside one *path* chunk – color or size cannot be changed in the middle of a line.

[3]http://www.w3.org/TR/CSS2/syndata.html#value-def-color

5. Implementation

5.1 ECMAScript and JavaScript

ECMAScript [1] is a standardized scripting language widely used in website development. Latest approved edition of ECMAScript is *ECMAScript 5.1*, which is implemented in most major web browsers, and its implementations in web browsers are commonly called JavaScript [2][3].

JavaScript is a dynamic programming language [4], which combines multiple aspects of imperative, functional, and object-oriented programming.

Functions are so called first-class citizens. This means they can be stored in variables, passed as function parameters, returned as results of functions, and included in data structures.

JavaScript doesn't provide any means of static type checking. All types are created during runtime and they are also checked only during runtime. The lack of static type checking during compilation might lead to rarely occurring errors and it is important to take this in mind while writing JavaScript code. A good practice is to write documentation comments[5], where the types are stated.

JavaScript is an interpreted language, some implementations also use a Just-In-Time compilation (JIT) [6] for better performance. Another very important aspect of ECMAScript which affects its performance is the absence of direct control over memory usage – memory is released when it is not needed any more. This mechanism is called *Garbage collection*. As of 2012, all modern browsers use mark-and-sweep garbage-collector [**?**].

Object oriented programming (OOP) [7] in JavaScript differs from class-oriented OOP in the way inheritance is implemented. While in class-oriented languages, for example in C++ or C#, inheritance is achieved by declaring classes of objects. In JavaScript, OOP is implemented through object prototypes. Prototype is just a link to another object, which has another prototype. A prototype chain is created this way. The last object in this chain has a `null` prototype. When trying to access a property of an object, it is searched in its own properties. If it is not found, then it is searched in its prototype's properties and so on until the end of the prototype chain is reached [33].

5.1.1 TypeScript

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript[35]. It is an open-source project developed by Microsoft. It is, as its name suggests, a strongly typed programming language compatible with JavaScript.

TypeScript extends capabilities of JavaScript by static type checking at the time of compilation, which helps the programmer to find errors in source code

[1] http://www.ecmascript.org/
[2] https://developer.mozilla.org/en/docs/Web/JavaScript
[3] https://msdn.microsoft.com/cs-cz/library/d1et7k7c(v=vs.94).aspx
[4] http://en.wikipedia.org/wiki/Dynamic_programming_language
[5] Commonly used syntax in JavaScript projects is JSDoc (http://usejsdoc.org/)
[6] For example Google V8 engine used in Google Chrome. See https://code.google.com/p/v8/
[7] http://en.wikipedia.org/wiki/Object-oriented_programming

sooner, and speeds up the process of coding. TypeScript also introduces *class* semantics. These classes are transpiled into JavaScript prototypes. TypeScript also includes concepts of interfaces and polymorphism, which makes programs written in TypeScript more understandable to programmers familiar with other object-oriented programming languages like Java, C# or C++.

TypeScript uses several features of ECMAScript 6, but it is transpiled into ECMAScript 5.1, and therefore does not bring any new functionality. The reason for choosing TypeScript is the clarity of code and more convenient development process for the programmer.

5.2 Event driven programming

The idea behind event driven programming is to break direct references between objects and to communicate with *events* instead of calling object methods directly. The advantage of the this approach is the so called *loose coupling* [36] – features might be added or removed without breaking the core of the application.

Each object handles only it's own task without knowing anything about the other objects it's collaborating with. When an object completes it's task, it publishes the result with a specific event. Objects subscribed for this event are notified and are given the outcomes of the previous task.

This mechanism is sometimes called the *Event Aggregator* design pattern. It is implemented through the `VideoEvents`[8] static class, which makes it a simple singleton object. This class provides an interface for registering and triggering callbacks for specified events. Callbacks executed by a triggered event are called asynchronously. It is worth mentioning that web browsers execute all scripts in a single thread.

```
1 VideoEvents.on(VideoEventType.Message, function(message)
      {
2   console.log("received message:", message);
3 });
4
5 VideoEvents.on(VideoEventType.Message, function(message)
      {
6   console.log("received message backwards:",
        message.split("").reverse().join(""));
7 });
8
9 VideoEvents.trigger(VideoEventType.Message, "Hello
    world.");
```

The list of all event types can be found in the enclosed API documentation of the `VideoEventType` enumeration.

5.3 HTML5

HTML5 is a term used to refer to modern web technologies. The core is the *HyperText Markup Language*, designed to describe semantics of structured docu-

[8] Implementation can be found in */public/js/app/Helpers/VideoEvents.ts* file

ments [37]. HTML5 extends the semantics of HTML 4.1 with new HTML tags such as `<header>` or `<footer>`, but also defines a huge set of new APIs, which allow web developers to create much richer websites and web applications. Some of these new technologies are needed by the Vector Screencasts project.

Working with XML documents Working with XML data is very similar to working with regular website DOM[9] in JavaScript. `Document`[10] interface is used for traversing the XML tree, modifying it or for creating a new one. In HTML5, XML files can be opened using a HTTP GET request via `XMLHttpRequest`[11] object, which parses the data and returns a `Document` instance in its `responseXML` property, if the document meets specified criteria [39].

WebSockets WebSockets allow applications to open bidirectional communication channels with server-side processes [][12]. WebSockets are built on top of the *Transmission Control Protocol* (TCP) connection. This means that the delivery of data is reliable and ordered [][13].

Web Workers JavaScript code of a website normaly runs in a single thread. It is common to run functions asynchronously, for example callback functions and event handlers, but all these functions still run in the same thread. Web Workers are a means of running heavy tasks in background without affecting the user interface. A real OS-level thread is spawn for each `Worker` object instance.

Web Workers have several limitations. They cannot change the DOM and have direct links to objects in the main thread. Web workers receive messages from the parent thread through an `onmessage` event and can send a response via the `postMessage` function. These messages contain serialized JavaScript objects, which cannot contain any references. As a result, concurrency problems are not typically an issue. Web Workers can perform HTTP requests using the `XMLHttpRequest` object, but the content is not parsed, if the target is an XML or HTML file. For more details see the specification of the Worker object[] [14]

Rendering graphics using HTML5

Displaying text and static visual content is the main purpose of older versions HTML and *Cascade Style Sheets* (CSS). Creating complex polygons or curves would be very hard, involve various tricks [15] or would be even impossible.

The new HTML specification takes this in mind and brings ways of creating more rich and dynamic content within a web page. There are two technologies that should be taken into account – *Canvas 2D Context* and *SVG*.

[9] Document Object Model, http://www.w3.org/DOM/
[10] https://dom.spec.whatwg.org/#interface-document
[11] https://xhr.spec.whatwg.org/
[12] https://html.spec.whatwg.org/multipage/comms.html#network
[13] https://tools.ietf.org/html/rfc6455
[14] https://html.spec.whatwg.org/multipage/workers.html#worker
[15] http://nicolasgallagher.com/pure-css-gui-icons/

Canvas 2D Context The `<canvas>` element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly [38].

Using canvas seems appropriate for this project. Canvas could be created with respect to user's resolution and web browser window size. All elements can be scaled to fit this viewport. This will make them look sharp and there will not be any artefacts, noise and blur caused by interpolation used for scaling regular bitmap video content.

When the dimensions of the `<canvas>` element change, the image is scaled, but the output will not be very sharp. Canvas contains a bitmap image, onto which the graphics primitives are drawn. All these primitives must be redrawn by the programmer to achieve smooth scaling.

SVG @todo

Audio capturing, processing and upload

HTML5 provides only one way to access microphone data at the moment and it is through the *getUserMedia API* [45]. *navigator.getUserMedia* function prompts the user to for permission to use their audio input (this function is also used to access webcam stream in other applications). The *navigator.getUserMedia* function is well documented on Mozilla Developer Network (MDN) [16] website.

If user's device has a connected microphone and user gives his permission to use his audio input, then a *MediaStream* [17] object is provided by the browser and from this time on audio can be processed. Error callback with *MediaStreamError*[18] instance as a parameter is called otherwise.

Current browsers implement *ScriptProcessorNode* according to the W3C Working Draft from 10 October 2013 of *Web Audio API*. Unfortunately, the *ScriptProcessorNode* interface is deprecated in newer drafts of the specification and should be replaced with *AudioWorkerNote* in the future[47]. This new approach is not yet standardised, implemented and documented in web browsers. Both of these methods provide access to *AudioBuffer* instance, which contains *linear pulse-code modulation* (LPCM)[50] sampled data from the microphone [49].

This data is then sent to the server using a WebSocket.

High Resolution Timer

To make the video look as good as possible, we need to store as precise data as possible. The *Date.now()* function[19] returns the number of milliseconds elapsed since 1 January 1970 00:00:00 UTC. The millisecond accuracy might seem enough, but modern browsers provide even more accurate data via the *High Resolution*

[16] MDN: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

[17] MediaStream: http://www.w3.org/TR/mediacapture-streams/#idl-def-MediaStream

[18] MediaStreamError: http://www.w3.org/TR/mediacapture-streams/#idl-def-MediaStreamError

[19] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/now

Time [20] via the *window.performance.now()* function[21] [22] with the accuracy of microseconds. The *window.performance.now()* function doesn't provide data related to current time, but the milliseconds elapsed since page was loaded as a floating point number. This makes it more suitable for animation purposes.

Timing functionality is wrapped in *VideoTimer* class in the Helpers module[23] with a method for getting the number of milliseconds elapsed since last timer reset with the best precision provided by the web browser.

Input from pointing devices

Detecting mouse movement and the state of it's buttons is a very common task in web development. Users navigate through web pages mainly by clicking on hypertext links with their computer mouse. Therefore mouse events are well specified and work across all desktop web browsers and desktop platforms.

Unfortunately, the situation among other pointing devices other than computer mice is much less uniform. With the boom of smartphones and tablets, touchscreens are very common. Also computer graphics tablets are used by artists and many people use them when creating a Khan Academy style video.

Mouse input Event handler is passed the *MouseEvent* object[24]. The relevat information are the *clientX* and *clientY* properties, stating current mouse posititon relative to the position of the element it is attached to.

Wacom Webplugin pen API The Wacom Webplugin pen API (WebPAPI) is a browser plugin interface for pen data access from all Wacom consumer and professional tablets [][25].

Unfortunatelly support for this plugin was discontinued by Chromium and Google Chrome [26] and therefore it should not be relied on.

Touch Events API Touch Events API is an API for handling touch input from touch screens. The standard is proposed by Apple and is implemented across many platforms and in many mobile web browsers [27] [].

This API supports multiple touches at once, but this feature is not needed and neither implemented in this project. Unfortunately, this API provides no touch pressure information.

Pointer Events API Pointer Events API is an open API created by Microsoft. It's purpose is to unify the way mouse events, touch screen events, stylus and other (i.e. Kinect) similar ways into one API. This technology is implemented in Internet Explorer and will be also present in the final version of the Microsoft

[20] https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/HighResolutionTime/Overview.html
[21] https://developer.mozilla.org/en-US/docs/Web/API/Performance/now
[22] http://updates.html5rocks.com/2012/08/When-milliseconds-are-not-enough-performance-now
[23] implementation can be found in */src/VectorVideo/Helpers/VideoTimer.ts* file
[24] http://www.w3.org/TR/DOM-Level-2-Events/events.html#Events-MouseEvent
[25] http://www.wacomeng.com/web/WebPluginReleaseNotes.htm
[26] http://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.htm
[27] http://caniuse.com/#feat=touch

Edge browser. Firefox implements this API, but it is so far accessible only if a specific hidden flag is enabled. Google has announced the intent to also implement this functionality in upcoming releases of Google Chrome across all platforms.

This API also provides pressure information for pointing devices, that support this feature, including Wacom graphics tablets.

5.4 Drawing lines

Khan Academy videos are known for their consistent and simple style. A person draws on a virtual canvas (evoking a school blackboard) with a brush (or possibly a chalk) of a round shape.

Tutor has a pointing device, typically a computer mouse or a digital stylus, and its position on the canvas is marked with a moving cursor. When the tutor clicks, a dot is marked on canvas in the current position of the cursor. Tutor can produce a line (typically a curve) following this cursor when he presses a mouse button or increases digital stylus pressure while moving the cursor. The curve ends when he releases the button or lowers stylus pressure. The color and size of the dot or line corresponds to current settings. Pressed mouse button corresponds to maximal applied pressure and released mouse button to no pressure.

Rendering at the time of playback gives us the opportunity to adjust the outcome to the environment of the end user. This means that the result can be sharp on every display resolution without the need of having many versions of the same video for each resolution.

5.4.1 Dynadraw algorithm

Paul Haeberli has created a simple algorithm called *"Dynadraw"* in 1989, which is suitable for calligraphy. Brush is modeled as a physical object with its *mass*, *velocity* and *friction coefficient* [28] []. Mouse movement is interpreted as a way of exerting force on the brush – the faster you move the brush, the greater the force applied on the brush is. Acceleration is then calculated according to Newton's second law of motion considering brush's mass and velocity of the brush is derived with respect to the value of friction coefficient. This velocity is then applied and brush is moved. The trace brush should leave behind is then drawn onto the virtual canvas.

The advantage of this algorithm is its simplicity and the possibilities of configuring the brush with different values of mass and friction (the author of the algorithm refers to this constant also as *drag*, which better fits the purpose of slowing down the brush).

Heavier brushes move slower than the light ones, but the path they leave behind is much smoother, as the hand shaking is eliminated by composition of forces in different directions.

Light brushes move faster and are often very close to the cursor during the movement. When the cursor stops abruptly, light brushes with little friction tend to keep moving past the cursor and wrap around it. This produces little curls at the end of lines.

[28] Dynadraw: http://www.graficaobscura.com/dyna/index.html

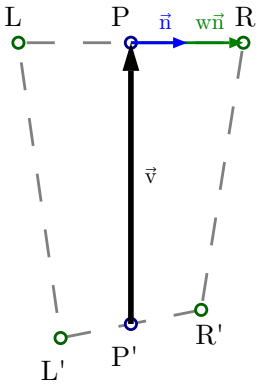


Figure 5.1: Drawn quadrilateral segment of a line

Brush movement simulation

One step of simulation applies force on the brush according to current mouse position and thus moves it in the direction of the cursor. This process of applying force must be done periodically, at the frequency of 60 Hz in ideal case[29]. Implementation of this simulation is not identical to the original *Dynadraw* algorithm, but all of its key aspects are preserved. Pseudocode 1 describes the algorithm used in this thesis.

The main difference between the original implementation and the one used in Vector Screencast project is brush width calculation. The original algorithm calculates the width of the line in a specific point by measuring its velocity – the faster the brush is moving, the thinner the drawn line is. This width dynamics gave the algorithm its name.

In our implementation, this effect is preserved, but it is much more subtle. Brush dynamics relies mainly on the pressure of a digital stylus on a graphics tablet. Since the exact value of pressure in the point of current brush's location is not always known, the value is linearly interpolated between the value of pressure in the previous position of the brush and the value of pressure in the current mouse position according to the distance from each of these points.

As this simulation is not deterministic, this process cannot be reconstructed afterwards. When the video is being played, only the precomputed values of path segments must be used. The precise value of pressure is therefore irrelevant for later playback of the screencast and does not have to be stored. The advantage of storing this values is the possibility of reconstructing the process of recording later and rendering the lines again. Even a different algorithm might be used in this case.

Rendering of one line segment

Line consists of many segments that are drawn after every simulation step, which moves the brush. There are several ways to draw the segment. The most straightforward is to draw a simple quadrilateral as shown in figure 5.1 and pseudocode 2.

The curves drawn using quadrilaterals are not very smooth. For smoother curves, straight lines must be replaced with interpolation splines. Both SVG

[29]HTML5 provides requestAnimationFrame function, which is intended for animations and which targets 60 frames per second

Pseudocode 1 One step of brush movement simulation

```
function OneStep(M, Δt)                    ▷ M - mouse position, Δt - elapsed time
    if M ≠ ∅ then
        brushMoved ← apply(M⃗, Δt)
        if brushMoved = true then
            DrawSegment
        end if
    end if
end function
function apply(M, Δt)
    F⃗ ← M − P                              ▷ P - current position of the brush
    a⃗ = F⃗/m                               ▷ m - mass of the brush
    if ‖a⃗‖ ≤ C_a then                      ▷ C_a - minimum acceleration constant
        return false;
    end if
    v⃗ ← v⃗ + a⃗
    if ‖v⃗‖ > C_v then                      ▷ C_v - minimum velocity constant
        P ← P + μΔtv⃗                       ▷ μ - coefficient of friction
        return true
    end if
    return false
end function
```

Pseudocode 2 Draw one segment of a line

```
function DrawSegment
    n⃗ ← (−v⃗_y, v⃗_x) / ‖v⃗‖
    w ← CurrentBrushPressure · b           ▷ b - brush size
    L ← P − wn⃗
    R ← P + wn⃗
    BeginPath
    MoveTo(L′)                             ▷ L′ and R′ - previousely drawn point
    LineTo(R′)
    LineTo(R)
    LineTo(L)
    ClosePath
    Fill(c)                                ▷ c - current brush color
    L′ ← L
    R′ ← R
end function
```

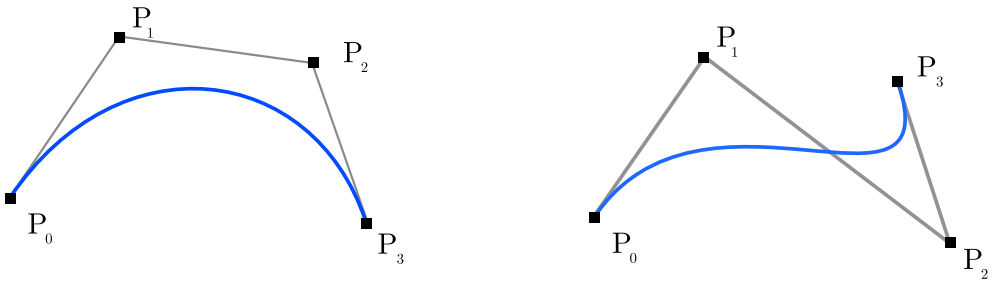


Figure 5.2: Bézier curve interpolation examples

and Canvas 2D Context implement *cubic Bézier curves*. Cubic Bézier curves are defined by four control points, $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. Path starts in $\mathbf{P}_0$ and ends in $\mathbf{P}_3$, it does not usually go through points $\mathbf{P}_1, \mathbf{P}_2$. The interpolation forumla of the curve is

$$\mathbf{B}(t) = (1-1)^3\mathbf{P}_0 + 3(1-t)^2t\mathbf{P}_1 + 3(1-t)^2t^2\mathbf{P}_2 + t^3\mathbf{P}_3, 0 \leq t \leq 1$$

[][30]

To calculate the control points of a cubic Bézier curve for segment between points $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$, calculated by the DynaDraw algorithm, we can also take points $\mathbf{X}_{i-1}$ and $\mathbf{X}_{i+2}$ and look at these four points as *Catmull-Rom spline* control points. Catmull-Rom is a special type of *Cardinal spline*, with the tension parameter $\tau = 0$[][31]. This approach gives us a nice smooth curve calculated just from the consequent points calcuated by the Dynadraw algorithm and also guarantee of C^1 continuity (@todo: citation) of the whole path.

A special conversion matrix between Catmull-Rom spline and Bézier curve is defined [][32] and so the conversion is very straighforward. The formula for calculating cubic Bézier control points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, from the points $\mathbf{X}_{i-1}, \mathbf{X}_i, \mathbf{X}_{i+1}, \mathbf{X}_{i+2}$ is

$$\begin{pmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3\end{pmatrix} = \frac{1}{6}\begin{pmatrix}0 & 6 & 0 & 0\\ -1 & 6 & 1 & 0\\ 0 & 1 & 6 & -1\\ 0 & 0 & 6 & 0\end{pmatrix}\begin{pmatrix}\mathbf{X}_{i-1}\\ \mathbf{X}_i\\ \mathbf{X}_{i+1}\\ \mathbf{X}_{i+2}\end{pmatrix}$$

The first and the last segments must be processed in a special way, as there is no preciding point, or following respectively.

For an example of cubic Bézier curve, see figure 5.2.

5.5 Building the library

Source code of the project can be found either among the attached files, or in git repository *https://github.com/simonrozsival/vectorvideo*. Copy or clone this code to your computer.

[30] An introduction to splines for use in computer graphics and geometric modeling, Bartels, Richard H, p. 160, https://cs.uwaterloo.ca/research/tr/1983/CS-83-09.pdf

[31] http://people.cs.clemson.edu/ dhouse/courses/405/notes/splines.pdf

[32] http://therndguy.com/papers/curves.pdf

`gulp`	
`gulp clean`	
`gulp lib`	
`gulp demo`	Builds the whole example project.
`gulp doc`	Generates API documentation from the TypeScript source files and comments

Table 5.1: Available gulp commands

To be able to compile the library, it is necessary to install *Node.js*[33] on your computer. After installing Node.js, install all dependencies of this project by running the `npm install` command in the root directory of the project's source files. You must also install *Gulp* streaming build system[34] globaly by running the `npm install -g gulp` command.

Build all code by running `gulp` command in the same directory. This will compile all TypeScript source files into JavaScript. You can then use the compiled and minified library located in *./release/vector-screencast/vector-screencast.min.js* JavaScript file. Along with the library, a web worker file *./release/workers/RecordingWorker.js*, which is necessary for audio recording. Default CSS theme of the player and recorder was compiled from LESS source to *./release/themes/default-/theme.min.css*.

A demo audio recording server was also compiled in this process and is available in the *./release/audio-server/AudioServer.js* JavaScript file. This server can used in a Node.js program.

For details on how to use the library in your project, see chapter 6 on page 29.

[33] https://nodejs.org/
[34] http://gulpjs.com

6. Integration of the library in an HTML web page

Using Vector Screencast library is intended to be as simple as possible. All the user needs to do is include a JavaScript file of the library and a CSS file of a theme into his HTML code and configure the player in a few lines of code.

6.1 Obtaining the Vector Screencast library

Prepared library files can be obtained either from the attached files or from the GIT repository https://github.com/simonrozsival/vectorvideo in the */release/VectorScreencast* and */release/themes/* folders. You only need to copy files *vector-screencast.min.js* and *theme-default.min.css* into your project.

The library files must be linked to your document and you should make sure that your website will be displayed properly on mobile devices by specifying the *viewport* meta tag in the *head* section of your document[1]. Also create an empty element with a specific *id* attribute – this will be the container, into which either the screencast player or the recorder will be placed.

An example of a HTML5 template with correct setup can look similarly (irrelevant parts of the document were let omitted and replaced by suspension points):

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      ...
5      <meta name="viewport" content="width=device-width,
           initial-scale=1">
6      <link rel="stylesheet" type="text/css" href="/path/
           to/theme-default.min.css" media="screen,
           projection">
7      ...
8    </head>
9    <body>
10     ...
11     <div id="some-specific-id"></div>
12     ...
13     <script src="/path/to/vector-screencast.min.js">
           </script>
14   </body>
15 </html>
```

The initialisation scripts must be executed afte all web page resources are downloaded and the DOM is ready – this can be achieved by putting this code inside a handler of the *window.onload* event.

[1] https://developers.google.com/speed/docs/insights/ConfigureViewport

6.2 Vector Screencast Player

Inside your scripts, create a new instance of *VectorScreencast.Player*. The constructor takes two arguments, first of them is the *id* attribute of a container element and the second is a configuration object. The only obligatory property of the configuration object is the *Source* property – the URL of the source SVG file.

There are several other interesting optional settings, that will help you customize the screencast player. One of them is the *Localization* property, which takes an object implementing the *VectorScreencast.Localization.PlayerLocalization* interface. To see the complete list of all configuration options and further details, please refer to the *VectorScreencast.Settings.PlayerSettings* interface in the API reference of the project in the */docs/* folder of the attached files. You can see an advanced example of *VectorScreencast.Player* usage */demo/public/play.html* in the attached files.

6.3 Vector Screencast Recorder

Inside your scripts, create a new instance of *VectorScreencast.Player*. The constructor takes two arguments, first of them is the *id* attribute of a container element and the second is a configuration object. The only obligatory property of the configuration object is the *Source* property – the URL of the source SVG file.

There are several other interesting optional settings, that will help you customize the screencast player. One of them is the *Localization* property, which takes an object implementing the *VectorScreencast.Localization.PlayerLocalization* interface. To see the complete list of all configuration options and further details, please refer to the *VectorScreencast.Settings.PlayerSettings* interface in the API reference of the project in the */docs/* folder of the attached files. You can see an advanced example of *VectorScreencast.Player* usage */demo/public/play.html* in the attached files.

6.4 Embedding Vector Screencast Player

@todo

6.5 Custom theme

@todo

7. Users' documentation

Users' documentation with screenshot similar to the one I have already done.

Conclusion

The goal of this thesis was to create an open-source library, which will allow users to create educational websites for creating and playing KSVs. The library uses a vector-based approach. Video is rendered with respect to user's screen resolution and the size of the viewport.

The project of Vector Screencast is available online as a public Git repository[1]. The library is written in a programming language, which should be easy to learn for all web developers. The library uses an object oriented design and is very modular, which makes it easy to extend with a custom rendering method, file format support or user interface.

File format Vector Screencast data is stored in a specificaly formed SVG document extended with a foreign namespace. The data contains information of author's cursor movement and applied pressure and prerendered lines. This type of animation cannot be viewed in any other video player, but a preview of the final state of the blackboard is displayed when the file is opened in a regular SVG viewer. Audio is recorded as an uncompressed WAV file. This file should be converted into one or more common binary audio formats, with respect to web browsers support of audio formats.

File size

7.1 Future work

The library is ready and anyone can use it today in his website for creating and playing KSVs. There are tools for drawing lines of different widths and colors onto a virtual canvas, erasing parts of the canvas and clearing the whole canvas. Some authors might be missing some tools they use in their bitmap editors when recording KSV – including bitmap images or photos, using a text tool or drawing straight lines. It would not be hard to implement these tools, but it would have to be thought over, whether they do not breach the idea of simple, natural look of KSV.

Binary file format SVG has proven to be a sufficient container for Vector Screencast data. A binary format should be used to achieve more data savings. Most of the data consists of coordinates and time values. Numbers are stored as strings in XML. Each digit character in an UTF-8 document takes up 1 B of data storage. Each component of a coordinate is a number with typical value in the range of 0 to 9999 with several decimal places – the library allows three decimal places at maximum. A typical coordinate component value, like "123.5", takes up more than 4 B, with the decimal dot character included. Time values, which express time in miliseconds, consist of more than four digits as soon as they represent a time value of more than ten seconds. If these numbers were saved

[1] https://github.com/simonrozsival/vectorvideo

binary as a 32-bit integer, or 32-bit floating point number, it would take up only 4 bytes of memory in total while maintaining the same precission in most cases.

This project was designed to be independent on a specific file format. The library can be easily extended to support different file structures without editing its source code.

Known issues

Audio recording *ScriptProcessorNode* interface is deprecated in W3C Editor's Draft of *Web Audio API* from 21 June 2015 [47]. This interface is used in this project and should be replaced using *Audio Workers*, when the specification is finalized[**?**].

Demo audio server *Audio recording server* program, which is included in the project as a demo, should be rewritten. This tool is not very robust and sometimes fails to save the data, it receives through a WebSocket, to a file on the server. Users must write this server program themselves at the moment.

Bibliography

[1] Bower, Beverly L. Hardy, Kimberly P. *From correspondence to cyberspace: Changes and challenges in distance education* [online]. New Directions for Community Colleges, Volume 2004, Issue 128, pages 5–12.

[2] John Oliver: Student Debt, HBO [online] 9/7/2015, available at https://www.youtube.com/watch?v=P8pjd1QEA0c

[3] Bill Gates: Online, All Students Sit in the Front Row [online], posted on November 18, 2014, seen on 3/12/2015, available at http://www.gatesnotes.com/Education/Colleges-Without-Walls-Arizona

[4] Jessica Leber: In the Developing World, MOOCs Start to Get Real [online], MIT Technology Review, published on March 15, 2013, available at http://www.technologyreview.com/news/512256/in-the-developing-world-moocs-start-to-get-real/

[5] Kepler [online], http://kepler.org/

[6] Generation Rwanda [online], http://www.generationrwanda.org/

[7] Moodle, https://moodle.org

[8] Moodle statistics, https://moodle.net/stats/

[9] Coursera [online], 3/13/2015, available at https://www.coursera.org/about/

[10] Youtube.com, https://www.youtube.com

[11] Alexa Internet, http://www.alexa.com/topsites

[12] Numberphile Youtube channel, https://www.youtube.com/user/numberphile

[13] Veritasium Youtube channel, https://www.youtube.com/user/veritasium

[14] Khan Academy Youtube channel, https://www.youtube.com/user/khanacademy

[15] Khan Academy, viewed July 8, 2015 [online] https://www.youtube.com/user/khanacademy

[16] Khanova škola, https://khanovaskola.cz

[17] Hypertext Transfer Protocol (HTTP/1.1): Range Requests, Request for Comments: 7233 https://tools.ietf.org/html/rfc7233

[18] Java applet, Wikipedia, the free encyclopedia, viewed July 8, 2015 [online] https://en.wikipedia.org/wiki/Java_applet

[19] Oracle, Java.com, How do I get Java for Mobile device? [online] viewed April 16, 2015 http://www.java.com/en/download/faq/java_mobile.xml

[20] Features, Adobe Flash Player, Adobe.com, viewed July 8, 2015 [online] http://www.adobe.com/cz/products/flashplayer/features.html

[21] Steve Jobs, Apple.com, Thoughts on Flash, published April 2010, viewed April 16, 2015 [online] http://www.apple.com/hotnews/thoughts-on-flash

[22] Adobe Blog, An Update on Flash Player and Android, published June 28, 2012, viewed April 16, 2015 [online] http://blogs.adobe.com/flashplayer/2012/06/flash-player-and-android-update.html

[23] SWF and AMF Technology Center, Adobe.com, SWF and AMF Technology Center, viewed July 8, 2015 [online] http://www.adobe.com/devnet/swf.html

[24] Bosomworth Danyl, SmartInsights.com, Mobile Marketing Statistics 2015, published January 15, 2015, viewed April 16, 2015 [online] http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[25] Microsoft Silverlight, Get Microsoft Silverlight, viewed April 16, 2015 [online] http://www.microsoft.com/silverlight/

[26] Moving to HTML5 Premium Media, Microsoft Edge Dev Blog, viewed July 8, 2015 [online] http://blogs.windows.com/msedgedev/2015/07/02/moving-to-html5-premium-media/

[27] http://blogs.windows.com/msedgedev/2015/05/13/announcing-media-capture-functionality-in-microsoft-edge/

[28] A brief history of ECMAScript versions (including Harmony/ES.next), Axel Rauschmayer, dzone.com, viewed July 8, 2015 [online] https://dzone.com/articles/brief-history-ecmascript

[29] Scalable Vector Graphics (SVG) 1.1 (Second Edition), The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/SVG/

[30] Document Object Model, Wikipedia, the free encyclopedia, viewed Junde 8, 2015 [online] https://en.wikipedia.org/wiki/Document_Object_Model

[31] Document Object Model (DOM) Level 2 Events Specification, The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/DOM-Level-2-Events/

[32] SVG Extensibility, The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/SVG/extend.html

[33] Inheritance and the prototype chain, Mozilla Developer Network (MDN), viewed July 8, 2015 [online] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain

[34] Memory Management, Mozilla Developer Network (MDN), viewed July 8, 2015 [online] https://developer.mozilla.org/cs/docs/Web/JavaScript/Memory_Management

[35] TypeScript, viewed July 8, 2015 [online] http://www.typescriptlang.org/

[36] Coupling (computer programming), Wikipedia, the free encyclopedia, viewed July 14, 2015 [online] http://en.wikipedia.org/wiki/Coupling_(computer_programming)

[37] HTML Living Standard, The Web Hypertext Application Technology Working Group (WHATWG), viewed July 8, 2015 [online] https://html.spec.whatwg.org/#is-this-html5

[38] The canvas element, The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/2010/WD-html5-20100624/the-canvas-element.html

[39] XMLHttpRequest, HTML Living Standard, The Web Hypertext Application Technology Working Group (WHATWG), viewed July 8, 2015 [online] https://xhr.spec.whatwg.org/#document-response

[40] Dynadraw algorithm, Paul Haeberli, viewed July 8, 2015 [online] http://www.graficaobscura.com/dyna/

[41] EVA animater plaza, SHARP, viewed July 8, 2015 [online] http://www.sharp.co.jp/sc/excite/evademo/evahome.htm

[42] Extended Vector Animation, Wikipedia, the free encyclopedia, viewed July 8, 2015 [online] https://en.wikipedia.org/wiki/Extended_Vector_Animation

[43] SWF, Wikipedia, the free encyclopedia, viewed July 8, 2015 [online] https://en.wikipedia.org/wiki/SWF

[44] WAVE Specifications, Prof. Peter Kabal, viewed July 9, 2015 [online] http://soundfile.sapp.org/doc/WaveFormat/

[45] Media Capture and Streams, W3C Last Call Working Draft 14 April 2015, viewed July 9, 2015 [online] http://www.w3.org/TR/mediacapture-streams/#dom-mediadevices-getusermedia

[46] The ScriptProcessorNode Interface, Web Audio API, W3C Working Draft 10 October 2013, viewed July 9, 2015 [online] http://www.w3.org/TR/webaudio/#ScriptProcessorNode-section

[47] The ScriptProcessorNode Interface - DEPRECATED, Web Audio API, W3C Editor's Draft 21 June 2015, viewed July 9, 2015 [online] https://webaudio.github.io/web-audio-api/#the-scriptprocessornode-interface—deprecated

[48] AudioWorkerNodeProcessor Interface, Web Audio API, W3C Editor's Draft 21 June 2015, viewed July 9, 2015 [online] http://webaudio.github.io/web-audio-api/#the-audioworkernodeprocessor-interface

[49] The AudioBuffer Interface, Web Audio API, W3C Working Draft 10 October 2013, viewed July 9, 2015 [online] http://www.w3.org/TR/webaudio/#AudioBuffer

[50] Pulse-code modulation, Wikipedia, the free encyclopedia, viewed July 9, 2015 [online] https://en.wikipedia.org/wiki/Pulse-code_modulation

[51] The WebSocket Protocol, Internet Engineering Task Force (IETF) RFC 6455, viewed July 9, 2015 [online] https://tools.ietf.org/html/rfc6455

[52] The WebSocket API, W3C Candidate Recommendation 20 September 2012, viewed July 9, 2015 [online] http://www-mmsp.ece.mcgill.ca/documents/AudioFormats/WAVE/WAVE.html

List of Tables

List of Abbreviations

Attachments