LIBERA AKADEMIO, AN AUTHORING TOOL FOR LOW-COST EDUCATIONAL VIDEOS

Marcello Ribeiro Salomão

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Orientador: Ricardo Guerra Marroquim

Rio de Janeiro
Julho de 2014
LIBERA AKADEMIO, AN AUTHORING TOOL FOR LOW-COST EDUCATIONAL VIDEOS

Marcello Ribeiro Salomão

PROJETO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinadores:

Prof. Ricardo Guerra Marroquim, D.Sc.

Prof. Claudio Esperança, D.Sc.

Prof. Claudio Miceli de Farias, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2014
Salomão, Marcello Ribeiro


Orientador: Ricardo Guerra Marroquim


Bibliography: p. 38 – 41

1. flipped classroom. 2. blended learning. 3. distance education. 4. interactive whiteboard. 5. authoring tool. 6. video representation. 7. education technology. I. Marroquim, Ricardo Guerra. II. Universidade Federal do Rio de Janeiro, Escola Politécnica/ Curso de Engenharia de Computação e Informação. III. Título.
I would like to thank my family for supporting me all these years, specially during the hard times.

My girlfriend Sarah, for the love and affection that sweeten my spirit.

My friends, who often make me laugh and remember that life should not be taken so seriously all the time.

My advisor Ricardo Marroquim, for understanding all my delays, believing in this project and being so nice to me, even when putting up with a lot of stress.

My friend Gustavo Pfeiffer, for also believing in my ideas, to the point his very own final project was motivated by mine. A project that indeed, turned out to be a great complement to Libera Akademio, allowing it to become truly low-cost.

And everyone else from my lab, for creating such a pleasant environment to work at.
Abstract of the Undergraduate Project presented to Poli/COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

LIBERA AKADEMIO, AN AUTHORING TOOL FOR LOW-COST EDUCATIONAL VIDEOS

Marcello Ribeiro Salomão

July/2014

Advisor: Ricardo Guerra Marroquim
Course: Computer and Information Engineering

We present an authoring tool for the creation of high resolution educational videos that can be reproduced even on the most inexpensive computers, such as the Raspberry Pi. Videos are encoded using our own format, that allows edition even after the video has been published. This, along with the use of the Opus audio codec, allows those videos to be streamed over an Internet connection with bandwidth as low as 56kbit/s, with acceptable download times.

Keywords: flipped classroom, blended learning, distance education, interactive whiteboard, authoring tool, video representation, education technology.
Resumo do Projeto de Graduação apresentado à Escola Politécnica/COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

LIBERA AKADEMIO, UMA FERRAMENTA DE AUTORIA PARA VÍDEOS EDUCACIONAIS

Marcello Ribeiro Salomão

Julho/2014

Orientador: Ricardo Guerra Marroquim
Curso: Engenharia de Computação e Informação

Apresentamos uma ferramenta de autoria para a criação de vídeos educacionais de alta resolução que podem ser reproduzidos com computadores de baixíssimo custo, como o Raspberry Pi. Vídeos são codificados usando nosso próprio formato, que permite edição mesmo após o vídeo ter sido publicado. Isso, junto ao uso do formato de áudio Opus, permite que esses vídeos sejam transmitidos através de uma conexão com a Internet de banda tão baixa quanto 56kbit/s, com tempos de carregamento aceitáveis.

Palavras-Chave: flipped classroom, blended learning, distance education, interactive whiteboard, authoring tool, video representation, education technology.
# Contents

## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ix</td>
<td>ix</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Motivation ........................................ 2
1.2 Objectives .......................................... 4
1.3 Related Work ....................................... 6
   1.3.1 Traditional Khan Academy Workflow .......... 6
   1.3.2 Educreations .................................... 6
   1.3.3 ShowMe ........................................... 7
   1.3.4 ScreenChomp .................................... 7
   1.3.5 Doceri ............................................ 7
   1.3.6 Explain Everything .............................. 8
   1.3.7 Our Solution .................................... 8
   1.3.8 Graphics Tablet Cost ............................ 8

## 2 Methodology

2.1 The Processing Prototype .......................... 11
2.2 The Java Prototype ................................ 12
2.3 The OpenGL Prototype ............................. 12
2.4 The QT Prototype ................................... 14

## 3 Implementation

3.1 Video .................................................. 15
   3.1.1 Events ........................................... 15
   3.1.2 Cursor Events ................................... 17
   3.1.3 Stroke, Eraser and Line Events ................ 17
   3.1.4 Event Rendering and Seeking ................. 18
3.2 Video Manipulation ................................ 19
   3.2.1 Copying .......................................... 19
   3.2.2 Pasting .......................................... 19
   3.2.3 Erasing .......................................... 19
3.2.4 Moving ................................................................. 20
3.2.5 Scaling and Moving ........................................... 20
3.3 Audio ................................................................. 21
    3.3.1 Scaling .......................................................... 21
    3.3.2 Opus Codec ...................................................... 22

4 Parser ................................................................. 23
    4.1 Finding the Cursor ............................................. 23
    4.2 Finding Segments - Naive Approach ...................... 24
    4.3 Finding Segments - Elaborated Approach ............... 25
    4.4 Improvements ................................................... 26

5 User Interface ..................................................... 28
    5.1 Menu Buttons ................................................. 29
    5.2 Example Use Cases ........................................... 31

6 Conclusion and Future Work .................................... 37
    6.1 Conclusion ....................................................... 37
    6.2 Future Work ..................................................... 37

Bibliography ........................................................... 38
List of Figures

1.1 Different styles used for recording educational videos.  

2.1 Path’s vertices are rendered twice as opaque as they should be, resulting in visible nodes in every corner of the path.  

2.2 This prototype was created in collaboration with 3 other friends: Andrew Richardson, Jesse Chezenko and Jim Bigland.  

2.3 Illustration of our stroke drawing mechanism.  

2.4 Comparison of rendered strokes. In the left, what we achieved with our connected strokes technique - notice the stroke thickness near the connection point. In the right, what we wanted.  

3.1 Different types of Events and their representation in the video bar.  

3.2 Possible device states for mouses and pens.  

3.3 Overview of how Events are structured.  

3.4 An illustration of how Stroke and Cursor Events are added to the Event vector. Sprites being added to the sprite buffer are also represented. Each circle inside one of the Events, correspond to one SubEvent.  

3.5 Explanation of parameters used in Algorithm [1].  

4.1 Example of an old Khan Academy video, where the only resolution available is 240p.  

4.2 Illustration of the method to locate the cursor’s tip. Area enveloped in blue contains pixels within the brightness threshold. The pixel with its perimeter in brown is the picked one, in this example. And the three red arrows represent the pixel search directions.  

4.3 Results of the naive approach. Extracted segments have low precision, are drawn when they should be not, and are drawn with the wrong color.
4.4 This image shows the result of the new technique in the window “Result”. Notice that stroke paths are being represented here only with their segment extremities, for clarity. Other windows show auxiliary information used by the algorithm: “Difference” is the frame difference and “background” is the known portion of the video (everything not occluded by the cursor).

4.5 Examples of outliers that could be easily removed from the final image (circled in red).

5.1 Current state of our implementation.

5.2 New.

5.3 Upload.

5.4 Hotkeys.

5.5 Options.

5.6 Example use case.

5.7 Audio segment being scaled.

5.8 Applying the desired modification.

5.9 Editing video Events.

5.10 Selecting both audio and video segments at the same time for symmetric editing.

5.11 Copying, pasting and erasing Events.

5.12 Moving Events in space, instead of time.
Chapter 1

Introduction

For most of the last century, the main aspects of the educational system were stagnated: very long lectures with compulsory attendance, schools with strong hierarchical structures, relatively short teacher-student interaction, and an approval system not based on subject mastery, but on students’ performance on a finite set of assessments [1, 2]. During that period, advances in education technology such as the use of personal computers in classrooms, multimedia presentations, and calculators [3], positively affected academic performance [4], but were not sufficient to change these aspects.

However, in the late 1990s [5], the educational system started changing, due to the growing number of schools adhering to a new paradigm known as flipped classrooms [6], a type of blended learning where traditional lectures are replaced by educational videos, that are consumed at home. While “homework”, instead, is done at school. This way, teachers’ time, previously spent on preparing and giving lectures, is reassigned to tutoring students.

As traditional lectures represent a considerable part of teachers’ time allocation, and their associated costs are one of the most cost prohibiting elements of the traditional educational system [7], the flipped classroom enables a significantly more efficient and affordable educational model [8].

Video lectures that support this new system have been produced in many ways, by people ranging from renowned university teachers to anonymous Internet users, using different approaches, as exemplified by Figure 1.1. One of their greatest advantages is that once recorded they can potentially be used by generations of students from every part of the world.
(a) Khan Academy - a disembodied voice narrates a lecture presented over a digital blackboard.

(b) Whiteboard lecture by YouTube user twbaroberts.

(c) CEDERJ - Brazilian program that offers public university access using a blended learning model. Recorded lectures are structured with 3 views: slides, outline and camera.

(d) MIT OpenCourseWare - regular university lectures recorded by a camera in the back of the classroom.

(e) Veritasium - videos do not follow a rigid structure, changing from lesson to lesson. Here, the gyroscopic effect is demonstrated through an outdoors experiment.

(f) Udacity - some lessons are given using an interactive whiteboard solution that attempts to solve stroke occlusion problems.

Figure 1.1: Different styles used for recording educational videos.

1.1 Motivation

This upgrade to the educational system brings many benefits. Among the ones considered the most important, we have:
• True possibility for mass education. For the first time ever it is possible to deliver the best available education to all parts of the world - even the most poor and remote - through digital resources such as videos, documents, images, annotations, e-books, applets and many more. These materials, specially videos, used to be scarce. Also, expensive equipment was necessary to access them. But this situation is being overcome due to all the initiatives that are releasing high quality educational content for free on the Web. Also due to the constant price drops in the tablet market [15], and new projects as Google’s Loon [16] and Facebook’s drone provided Internet solution [17] that promise to deliver cheap wireless connectivity to areas where it would not financially compensate before.

• More efficient use of teacher-student time and self-paced learning. Students can learn from recorded lectures that can be paused as many times as they want without feeling embarrassed for interrupting the lecturer multiple times. These lectures can be watched anytime, according to students’ needs and their time allocation. Also they are usually designed to last for a time that is within the user’s attention span [18] (this kind of lecture has been referred to as a microlecture [19]). This way, the teacher is left with the job of going from student to student during class time, finding out about what problems they are having and offering personalized help, a type of assistance that would have been more expensive in the classical educational system. This could increase the chances of teachers identifying and solving usually unnoticed problems that reduce student performance - such as family conflicts, health problems and bullying [20]. This face-to-face tutoring approach has been regarded as a more humanized way of teaching [2 21].

• Ability to change the approval system to one based on mastery. In contrast to the previous model, where the cost in time and effort of grading and correcting tests, homework and other assignments would limit the possible amount of assessments per student, it is now possible to evaluate every student as many times as needed through automated exercises and tests. This way, it becomes possible to ensure that a student will not move past a certain subject until he or she achieves mastery of it. For instance, Khan Academy does this by requesting students doing exercises on a given topic to get 10 questions right in a row, before moving on to the next one [2].
There are now many initiatives creating high quality educational content for the web, such as the ones seen on Figure 1.1. Khan Academy, however, stands out among them. Created by Salman Khan and financially backed up by the Bill & Melinda Gates Foundation, Google and others, this not-for-profit organization has gone much further than any other initiative, having stretched its influence so far that it is now being tested in some schools as a part of an experimental teaching paradigm [2, 22].

Having in mind the undergoing changing process in education and the success of the Khan Academy model of lecture structuring and recording, we decided to create a tool that would accelerate the flipped classroom phenomena and expand the horizons of distance education. This would be done by empowering more users to create high quality educational content, through the reduction of existing barriers for creating online educational videos, such as adaptation to new technology, dexterity needed to draw, speak and interact with the user interface at the same time, expensive tools, and relatively long setup times for existing lecture recording solutions [23].

1.2 Objectives

We decided to create a platform called Libera Akademio (“Free Academy” in Esperanto) that would make the creation, edition, translation and consumption of educational videos simpler and cheaper, by guaranteeing the following features:

- Creation of content similar to Khan Academy’s videos, where the teacher annotates on an interactive blackboard while he explains a subject, while also providing tools to insert images, text, and lines.

- Allow videos to be uploaded or watched even with very low bandwidth connections (even lower than 56 kbit/s), using state of the art audio compression and a video format based on series of chronologically ordered events (which we will discuss in the implementation section).

- Render the video using the graphics card, to achieve a smooth and pleasant experience during video playback - this specific feature focuses on bringing users of other platforms (such as YouTube and Vimeo) to ours, thus getting closer to the critical mass needed for a self maintaining community. Also, this feature would allow videos to be consumed even with very low end devices like the Raspberry Pi (a 25 dollar development board) [24], unbranded Android powered Chinese Tablets and old netbooks.

- An integrated subtitling and translation tool, that automates and simplifies some of the steps usually needed for those activities, such as slowing down...
the video, and then speeding it up again to its normal speed after it has been translated.

- Make all the videos editable using an intuitive tool and allow anyone to edit any video uploaded to the platform, while maintaining the original version, creating a sort of version control system for videos. This would allow for users from the entire world to correct and/or improve previously uploaded videos, making the collection continuously updated. In this sense, it would work like a Wikipedia for educational videos.

- Allow for many different videos on the same subject, with different points of view, teaching techniques, voices tones, paces, etc. Unlike what Wikipedia does, allowing only one article per subject. This would make the platform more adaptable to different learning styles, that benefit from different presentations of the same content [25].

- Provide incentive for the creation of microlectures [19] through restrictions in the authoring tool itself - for instance restraining the amount of drawing space to work with per video and the time limit per lecture. Also, discourage slideshow presentations, in favor of more dynamic lectures with real time annotations - for example, by limiting the total amount of images that can be used per video. However, allow users to create presentations of one slide per video, in order to encourage teachers that already have slides on a certain subject to reuse them, while changing the pace of the presentation to a more dynamic one.

- Presenting all the available content in a website similar to YouTube, therefore creating a familiar and intuitive interface for video consumption. But improving the comments section to allow more structured discussions, for instance with forums shared across videos on the same subject.

- Allow for videos to be annotated and printed by the final user for offline studying.

- Be entirely free and open source, allowing the community to extend, improve or provide fixes to the software. Also, do not receive funding through contracts that could somehow limit the freedom of expression in our platform - users should be able to teach and express anything that comes to their minds, as long as they comply with local and international laws. For instance, videos containing messages with racism and/or xenophobia would not be tolerated, no matter what the rest of the content is.
1.3 Related Work

After analyzing other projects in the area of “Authoring Tools for Educational Videos”, we came to the conclusion that our proposed solution would be novel in certain aspects that would justify its development. We will discuss these aspects in the subsection “Our Solution”.

Among the most similar initiatives, we highlight five interactive whiteboard screencasting applications: Doceri [26], MorrisCooke’s Explain Everything [27], TechSmith’s ScreenChomp [28], Educreations [29], and ShowMe [30]. In the following subsections, we will compare them with each other and our proposal. We will also comment on the traditional solution to create educational screencasts. A summarized review with feature comparison is available on table 1.1.

1.3.1 Traditional Khan Academy Workflow

The traditional workflow used by Khan Academy and other screencasters is the most cumbersome of all solutions, requiring a screen capture tool like Camtasia Studio [31], and a painting tool as SmoothDraw [32]. Therefore, from the fact that the recording and the painting applications are separated, it can be inferred that there is no possibility of a simple video edition mechanism, with features like deleting unwanted strokes and substituting them with other strokes. Also, there is no support for special types of video files that allow great compression and high performance playback. Additionally, this is the slowest solution in terms of setup speed, requiring multiple, non specialized, software to be installed, tested, learned, filtered and finally used to record the intended lecture.

1.3.2 Educreations

Free and one of the simplest in terms of user interface. It allows users to create educational videos using strokes with 4 different colors, an eraser tool, images and audio capture. Undo and redo feature is available for every kind of input (stroking, erasing or inserting images). The working area for each video is a set of vertically scrolling sheets that are accessed in a similar fashion as slides in a usual presentation software such as LibreOffice’s Impress. The sheets can have a customized background image to imitate surfaces like graph paper, notepad paper or a blackboard. Recording can be paused and resumed at any time and annotations can be made either when paused or recording. Modifications made when the recording is paused will appear as they were instantly made when the video is played.

Both its recorder and its player use the Flash Player technology available for
browsers, a technology that counts with an extensive user base [33], except for iOS devices such as the iPad (for which there is a dedicated version) [29].

Final recorded videos are light in terms of bandwidth requirements, as they do not use a regular video format, but one based on vector graphics, such as the one we proposed. However, even though Flash can render with GPU acceleration, it is considered to be a slow, unstable and insecure platform that is gradually going to be replaced by more robust technologies such as HTML5 [34] - Also, Flash does not work on mobile platforms like the raspberry pi, and never will as Adobe has abandoned Mobile Flash [35].

1.3.3 ShowMe

Similar to Educreations, but only has an iPad version, which is also free - desktop versions are not planned by its developers. Additionally, videos made with this tool are shared as regular videos reproduced with the Flash tool JW Player, therefore not being suited for use cases involving low bandwidth or low cost devices [30].

1.3.4 ScreenChomp

Similar to ShowMe, except it has support for multiple slides per video, has no undo feature, and the final version of its videos is pixalated because of compression [28].

1.3.5 Doceri

The first one to actually support some form of video editing. Doceri features a stop point based animation system, where “each stroke of your finger or stylus is recorded and can be replayed and edited in sequence, or cut, copied and pasted elsewhere in the project timeline using the lasso tool.” [26]

Its videos are handled internally in a different way than regular videos, what can be verified by their claim that “Any saved drawing can be opened again so you can review, edit and set stop points using the timeline so you can replay your drawings and annotations as they were written, or reveal a section at a time.” [26]

However, videos created by Doceri that are shared to people that do not have an iPad with their app are regular video files: “Once you’ve perfected your project, you can add an audio voice over to any section, or the entire project, pausing at any point and resuming recording when you’re ready to go on. The result is a standard .mov video that will be saved to your My Recordings folder.” [26]

It features an iPad version and a “desktop version” (for Windows and Mac OSX), that is an extension for the iPad version.
1.3.6 Explain Everything

The tool with the most similarities with our proposal. Also counting with video and audio editing and advanced features such as annotating over videos and PDF documents, importing from Evernote and Dropbox and exporting to DropBox, Evernote, YouTube or Email. It is paid, though [27].

1.3.7 Our Solution

The last mentioned software apparently allows for special compressed videos to be watched on both Mac OSX and on the iPad, and might use some degree of GPU acceleration. As these features are not declared by its developers and we did not get copies of this program to evaluate whether they really exist, we assumed it already did exactly what we planned for our application regarding video compression and acceleration. Even with those assumptions, we considered it reasonable to continue the development of our application for the following reasons:

1. Their software is not free;
2. Their software is not open source nor based on open standards (for instance, their video format specification is not publicly available);
3. There is no support at all for Linux and no support for video creation on Mac OSX;
4. The timeline editing capabilities of their software did not have features such as changing the tempo of audio segments, without distorting the original pitch (one of the audio editing features we planned for our software).

Table 1.1: Comparative table between evaluated solutions [36]

<table>
<thead>
<tr>
<th>Free</th>
<th>Traditional Method</th>
<th>Educriations</th>
<th>ShowMe</th>
<th>ScreenChomp</th>
<th>Doceri</th>
<th>ExplainEverything</th>
<th>Libera Akademio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-Source</td>
<td>Depends</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Plays Low-Bandwidth Format</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Depends</td>
<td>Yes</td>
</tr>
<tr>
<td>Efficient Rendering</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Depends</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows Video Editing</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows Audio Editing</td>
<td>Depends</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Available for the iPad</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Available for Android</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Available for Mac OSX</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Available for Linux</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Available for Windows</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Available for Browsers</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

1.3.8 Graphics Tablet Cost

A good graphics tablet used to be the greatest cost bottleneck for our application. However, a recent work by Gustavo Pfeiffer [37], allows users to control the mouse
using just a webcam, a white paper and a blue pen. The mouse precision and the ease of use of the system was considered satisfactory, thus it is the recommended low cost alternative for interacting with our software.
Chapter 2

Methodology

Before talking about the prototyping phase, it’s useful to identify the restrictions that imposed the specific frameworks we experimented for prototyping. Succinctly, they were:

1. Availability to mobile platforms;

2. Capacity to work with a preexisting library for interfacing with graphics tablets.

The first one is more simple to understand - working with a language or framework that would not function on mobile devices would mean excess work, as these devices are targeted by our project, as they are potentially more accessible than notebooks and desktops in price terms.

The second restriction was perceived after the effort spent attempting to interface our first Windows prototype with a Wacom tablet using the standard method, suggested on Wacom’s website - modifying a C ANSI file and its header, both provided by Wacom [38]. Even though we made it work, it required some days of effort. Also, integrating it with other libraries was not simple and it only worked for Windows. Solutions for the two other targeted desktop systems (Linux and Mac OS X) were completely different and had reduced documentation and demos available.

Therefore, we identified three cross-platform solutions for interfacing with Wacom tablets: the proTablet [39] library for Processing, the jPen [40] library for Java and the C++ framework, QT [41], which supports Wacom tablets (although broken on some Operational Systems, at that time). This narrowed down the considered languages to Processing, Java and C++. Coincidentally, those are commonly used languages for developing to mobile platforms [42, 43].
2.1 The Processing Prototype

Considering all the proposed features, a first prototype was implemented using Processing and its built-in rendering commands, that allow choosing between both software and hardware (i.e., accelerated by the graphics card) modes. The prototype had many features implemented, including some more advanced ones like audio capture and playback, ability to create multiple slides, seeking and exporting the video file. That was not a good project decision, because we should have tested more basic features before, such as the stroke rendering capabilities of Processing. In fact, the language did not offer options regarding the quality of strokes (see Figure 2.1) and drawing performance that were adequate for our use cases.

Figure 2.1: Path’s vertices are rendered twice as opaque as they should be, resulting in visible nodes in every corner of the path.
2.2 The Java Prototype

Therefore, another prototype was created, this time using Java with the Java2D rendering module for drawing strokes and other GUI elements. Java2D was immediately put to the test of stroke rendering, in order to avoid having issues with this part of the project later in the development, as it happened with the Processing prototype. It performed well, boasting high framerates. However it also had problems, similar to those found in the Processing prototype, when it came to rendering quality of the strokes. This was expected since Processing’s rendering module is very close to Java2D, although less configurable because of its minimalistic API. In fact, Java2D was chosen due to the possibility that having access to a more complete API, the rendering quality and speed could be somehow improved. As it did not work out, we moved to our next attempt.

2.3 The OpenGL Prototype

Then, a third prototype was built, again using Java, but this time using OpenGL ES (Embedded Systems) for the rendering of strokes and the UI. The OpenGL ES API was accessed using the JOGL library as a glue between Java and the native OpenGL ES API. This API is considered by some to be a complex and daunting one.
and indeed a long time was spent mastering its use. The choice for OpenGL ES over regular OpenGL is due to the fact that our software is supposed to work in mobile environments, where the regular OpenGL API is not available.

The first method attempted when drawing with OpenGL consisted of drawing series of stripes with a special setup of texture coordinates (see Figure 2.3). This setup allowed an initial stroke texture, that was created on the fly using shaders, to be mapped to the stripes, creating a rounded corner effect where stripes connected.

However, this method would generate jagged edges on the strokes, due to the lack of antialiasing on the screen and textures. So, an operation called framebuffer multisampling (also know as framebuffer antialiasing) was applied to each final frame rendered, in order to solve this. The problem was solved, but at the cost of a considerable performance loss.

Then we attempted an improvement over the last technique. Instead of using a solid stroke texture, we used one with a radial gradient, going from completely opaque on the center to fully transparent on the borders. In theory, this would make the entire process faster, since the texture itself would solve the aliasing problem, sparing our application from the heavy framebuffer multisampling operation. As we predicted, rendering performance was recovered, but a new problem appeared - the points where the stroke lines connected were perceptively darker than they should be (see Figure 2.4 for an illustration). This problem could be solved by using another blending method in OpenGL during the stripe rendering, however the fact that this alternative blending method was not available for the ES version of OpenGL made us look for another alternative.
Finally, we tried to no longer use connected stripes, but lines made up by tightly packed textured squares or “particles” (as they are often called). As the squares were evenly spaced, the dark corner problem was solved. But again, this had a performance cost, as now, lots of drawing commands were being issued to the GPU and a great number of vertices was being processed and stored in memory. However, the rendering system was finally good enough, running at more than 30 frames per second on an Intel Atom N270 processor with an integrated graphics card.

After this, a few small improvements were made, to achieve better performance, like changing the rendering of quads to the rendering of sized points (a transition from 6 vertices and 6 texture coordinates per square, to 1 vertex and no texture coordinates) and changing the numerical precision of the vertices from float to short (50% reduction in memory use for storing vertex coordinates on the GPU).

### 2.4 The QT Prototype

After that, another problem showed up. The Java version of the application used up too much memory for running on embedded devices like the Cubieboard 2 [47], which we used for testing. As the whole code at that time needed to be cleaned and refactored, and Digia’s QT framework had just solved it’s compatibility problems with the Wacom API, it was a good opportunity to start developing the final version of the program using C++ and QT. This decision wasn’t as time consuming as expected; most of the Java code was easily adapted to C++. Also, profiling OpenGL code created using C++ was proved much easier than using Java, providing a faster development.
Chapter 3

Implementation

We now present the algorithms and data structures used in the main parts of the project.

3.1 Video

In order to allow fast editing of the video and create maintainable code, the data structure we used to represent the animated part of our video was a vector of Events. Other data structures were considered, but they all presented some weak points. We will briefly talk about them and why they were rejected in the end of this section.

3.1.1 Events

Events in our program represent anything that happens visually during video playback, such as the insertion of pen strokes, eraser strokes, lines, images, text, and the manipulation (movement, rotation and scale) of these Events. Also, cursor movement, undos and redos are Events in our system.

Stroke, Line and Text Events are painted in the video bar with their respective color, Eraser Events are painted with the canvas’ background color, Image Events with parts of the inserted image, Manipulation Events with diagonal line patterns, Undo Events and Redo Events with dashed lines, and Cursor Events aren’t painted at all. Refer to Figure 3.1 for an illustration.
Events are structured as it can be seen on Figure 3.3. All of them share some characteristics: a beginning, an end, a type, and an ID, which is used to keep track of Events for use cases such as picking and deletion. Whenever an Event is generated, it is appended to the Event vector.

In the next subsections, each Event type will be seen in detail. For simplicity, from now on, we will refer to a mouse or a graphics tablet’s pen as a pointing device. Also, the released state of a pointing device is simply the state where it is not pressed (see Figure 3.2 for further clarification).
3.1.2 Cursor Events

These Events are generated every time the pointing device is released or moved, given the preceding Event is not a Cursor Event itself. When they are created, their Start Time is set to the current time.

After that, as the pointing device is moved, SubEvents are appended to their SubEvents vector, recording the position and instant of each Cursor position sampled.

When the user presses the Cursor again, then we know another kind of Event will be created and appended to the Event vector. Therefore we close the current Cursor Event, by setting its End Time.

3.1.3 Stroke, Eraser and Line Events

These are generated every time the pointing device is pressed and their corresponding tool is currently active.

Similar to Cursor Events, their SubEvents vector is extended as the pointing device is dragged across the canvas. And when the pointing device is finally released, they are closed.

It is important to notice that SubEvents’ Sprite Pointer is used whenever a Stroke or Eraser Event is being filled. When a new SubEvent is added, a number of sprites are added to the sprite buffer - an OpenGL vertex buffer that holds the
center coordinates of each sprite. We keep track of each SubEvent’s corresponding sprite buffer slice by storing the slice’s range in it. An illustration of this process can be seen on Figure 3.4.

![Figure 3.4: An illustration of how Stroke and Cursor Events are added to the Event vector. Sprites being added to the sprite buffer are also represented. Each circle inside one of the Events, correspond to one SubEvent.](image)

3.1.4 Event Rendering and Seeking

Events can be rendered in two ways: from an arbitrary instant of the video to another (referred to as an Incremental Draw), or since the start of the video to a certain instant (referred to as a Complete Redraw).

The first method is used either in record and playback mode, when normal strokes or eraser strokes are being drawn. It is also used when seeking to an instant of the
video, ahead of the current one. This method simply finds all the existing SubEvents between the two specified instants, and draws their respective point sprites.

As this use case is the most frequent one, having this specialized rendering method means that, most of the time, the software won’t consume too much processing power, thus saving battery time on mobile devices. Also, it guarantees that, even tough seeks may be slow if skipping to a portion of the video where many strokes have been drawn, playing the video will never be slow.

The second method is used for destructive seeks (when seeking to an instant of the video before the present one) and for Event pasting and erasing operations, which we will discuss ahead. To draw the Events, it uses the same technique as the first one, except the initial instant is the start of the video (zero minutes and zero seconds).

3.2 Video Manipulation

With the vector of Events data structure, performing these operations is simple and computationally inexpensive. Since the vector is actually holding pointers to Events, these operations are light on memory access - only pointers are moved around, the Events themselves remain barely untouched, only needing their Timestamps and their SubEvents’ Timestamps to be updated.

3.2.1 Copying

Copying involves finding the Event range to be copied (which is fast, since we use binary search of Timestamps) and copying them to the Event clipboard. Also, the painted section of video bar is copied too.

3.2.2 Pasting

When pasting, we simply move the Events present on the clipboard, if any, to the new designated time slot, which starts on the time cursor’s current position. If Event collision is detected, the old Events are erased to make space for the new ones, being pasted.

3.2.3 Erasing

Erasing, just like copying, involves finding the Event range to be deleted. After that, we simply remove the selected Events. If the Event selection contains incomplete Events, we trim those Events accordingly.
3.2.4 Moving

Moving is the same as adding up the total delta time to the Timestamps of all the moved Events and their respective SubEvents’ Timestamps.

When Stroke, Eraser, Line, Image or Text Events are being dragged through the video bar, we continuously check for collisions with other Events, using a color code to inform the user about any collisions. This will be seen in more detail in the Results Section.

3.2.5 Scaling and Moving

When scaling is performed, we execute Algorithm 1 for each Event, regardless of whether moving was performed as well. This is because Events can be stretched in both negative and positive directions, and stretching them in the negative direction is the same as scaling them in the positive direction by the same amount, and then moving them back accordingly.

| Data: TimeShift, Scale, Pivot |
| Result: The Event and its SubEvents, if any, get scaled and moved. |
| $\text{StartTime} \leftarrow (\text{StartTime} + \text{TimeShift} - \text{Pivot}) \times \text{Scale} + \text{Pivot}$; |
| $\text{EndTime} \leftarrow (\text{EndTime} + \text{TimeShift} - \text{Pivot}) \times \text{Scale} + \text{Pivot}$; |
| if does not have SubEvents then |
| \quad return |
| end |
| foreach $SE$ in SubEvents do |
| \quad $SE \leftarrow (SE + \text{TimeShift} - \text{Pivot}) \times \text{Scale} + \text{Pivot}$; |
| end |
| return |

**Algorithm 1:** Algorithm to scale and move Events. Input parameters are explained in Figure 3.5
3.3 Audio

Audio management is simpler: during the recording phase, the audio is continuously written to a file as a stream of samples. Copying, pasting and moving audio segments is done by transferring parts of this buffer to and from the audio clipboard, while erasing is done by writing zeros to the specified segment to be erased.

3.3.1 Scaling

Audio scaling, however, is not trivially performed. But instead of doing it ourselves, we rely on the SoundStrech Library for this task. This library allows us to change the tempo of a given audio file without changing its pitch. There is no way to demonstrate the result of this operation on this article, but we encourage the reader to visit their website [48] for a demo, or test the audio scaling feature in our program.

Although the video translation feature was not implemented in our program yet, SoundStretch is an essential part of this feature, as the traditional workflow for translating educational videos, involves slowing down the original video and then translating in real-time, but in the slowed down pace. SoundStretch would make this process more comfortable, by providing a more naturally sounding version of the slowed down video and by automatically speeding up the translated audio track,
when the translation is over, while maintaining the original pitch, as opposed to the “chipmunk” voice that naive audio stretching would generate.

### 3.3.2 Opus Codec

In order to achieve the most compressed audio format we used the Opus Tools, acquired from the Opus Codec webpage [48], to convert our audio track to the Opus Format at the end of the video creation and/or editing session. The Opus Codec is a royalty-free, open-source, audio coded that is available at the moment for the Firefox and Chrome web browsers as a native feature (no plugins or extensions must be installed to play audio files of this format). And since it provides good sound quality at small bitrates [48], it is the most adequate audio codec for our application.
Chapter 4

Parser

Parallel to the development of the Libera Akademio authoring tool, we also worked on the creation of a parser that would read Khan Academy style videos in formats such as mp4 or mov, and transform them into our project’s format. This was mainly motivated due to the fact that some old Khan Academy videos are only available in low resolution formats, such as the one on Figure 4.1. In the next subsections we will discuss our approach to find stroke paths in one of these videos.

![Figure 4.1: Example of an old Khan Academy video, where the only resolution available is 240p.](image)

4.1 Finding the Cursor

The first step we took to find the equivalent vector video of the parsed Khan Academy video, was to elaborate an algorithm to verify, for a given frame, if the cursor is present and, in case it is, precisely locate its tip’s position.

To locate the cursor’s tip, we scanned each frame, looking for pixels brighter than a certain threshold. Whenever one was found, we repeatedly jumped to the pixel to its left, if this pixel was within the previously specified brightness threshold.
When not able to jump left anymore, we repeated the same procedure up and down, as is can be seen on Figure 4.2. Then, if the vertical distance between the last pixels found when searching up and down is greater or equal to four, we consider we found our cursor. The cursor tip will be in a fixed distance from the top pixel.

![Figure 4.2: Illustration of the method to locate the cursor’s tip. Area enveloped in blue contains pixels within the brightness threshold. The pixel with its perimeter in brown is the picked one, in this example. And the three red arrows represent the pixel search directions.](image)

This method works with good precision for most videos, since the color white is not used very often for drawing strokes.

We tried some other template matching techniques before, such as SAD and NCC, but they did not give satisfactory results, in the sense that the cursor tip’s found position would oscillate around the correct position.

### 4.2 Finding Segments - Naive Approach

After managing to find the cursor’s tip with precision, we thought that finding the corresponding vector video would be a matter of adding Stroke Events for every two frames, where there were some pixels changing between them. In other words, if the difference between these frames contained some non dark pixels (above a certain brightness threshold), than it would mean that those different pixels are part of a new stroke path that was added on the last frame, whose color would be the mean color of those pixels.

In the next Figure, the results of this approach can be seen. Clearly, it did not work. We believe the reason is that mouse sampling during recording is around 60 frames per second, while the video we have access to has only 10 frames per second. In other words, there can be discrete curves significantly far from a line segment between each two frames, that are composed of up to 6 line segments. The naive approach oversimplified this problem.
4.3 Finding Segments - Elaborated Approach

In this new approach, for each frame difference, where the resulting frame is not entirely black, we start from the bright pixel that is closer to the cursor tip of the older frame. Then, we follow the curve it belongs to, until we get close enough to the cursor tip of the newest frame.

In order to avoid problems caused by the occlusion from the cursor, whenever a pixel from the curve being followed is in a position under the cursor, we stop the curve following algorithm, and only continue it when this hidden position becomes revealed. Then, we substitute its pixel value for the one of the just revealed pixel. The new results can be seen in Figure 4.4.
4.4 Improvements

Our parser is still under development and therefore can be improved in many areas. One of the most visible improvement possibilities is checking the points we found against the final video frame. If their neighborhoods differ significantly, they could be identified as outliers, and removed from the list of found stroke segments (see Figure 4.5 for reference).

Apart from this improvement, several others will be needed until our algorithm can produce satisfactory results. Also, the parser must be able export the processed video to our format, before it can be released at all.
Figure 4.5: Examples of outliers that could be easily removed from the final image (circled in red).
Chapter 5

User Interface

This section will be structured as follows: First the user interface will be illustrated and explained. Then, commons tasks will be explained, exemplified and illustrated. Screenshots will be used when convenient.

The current state of our authoring tool implementation can be seen on Figure 5.1. The main window interface is self descriptive, except for the bottom part, where we have, from just underneath the white canvas: the video bar, the audio bar and the timeline ruler.

![Figure 5.1: Current state of our implementation.](image)
5.1 Menu Buttons

Clicking on the "New" button will open up a window that allows the user to specify the project’s name and background style.

![New Project Window]

Figure 5.2: New.

Buttons "Save", "Open" and "Export" open a file dialog, allowing the user to specify where the project will be saved, where to open a project from, and where to export a webM video to, respectively.

Clicking on the "Upload" button allows the user to complete metadata about the video created, upload it to our servers and, optionally, to YouTube as well.

![Upload Window]

Figure 5.3: Upload.

"Hotkeys" allows the user to setup hotkeys for almost any task within the program.
"Options" allows things such as changing to a dark theme, more convenient for working during the night, different rendering modes and selecting the audio input and output devices.
5.2 Example Use Cases

Here, the common use case of creating a video and performing some basic editing will be exemplified.

After selecting the Pencil Tool, pressing record, and creating a few strokes, the user will notice his strokes are being represented as colored rectangles in the video bar, drawn according to the time they were inputted. Also, his voice is shown in the audio bar, through the use many absolute amplitude bars, displayed in black - this way, the user can know about moments of silence during the video recording and has a way to edit the video’s audio track.

The time Cursor, visually identified as the dotted line marking 35 seconds, shows the total recorded time, until the present moment, but will also be used for seeking, and copying and pasting operations.

The user now decides to alter the speed at which some of his voice is played. In order to do that, he selects the audio segment he wants to alter, by dragging with his pointing device. When hovering over the selection, left and right arrows will smoothly appear. These arrows can be dragged to scale the current selection. To move the selection, he simply drags the selection rectangle, that resides between the two arrows.
To confirm the operation, he right clicks the selection and selects the option “apply” in the context menu that appears. Or “cancel”, if he wishes to reconsider (clicking outside of the selection has the same effect as hitting the cancel option).

The workflow for scaling and moving video Events is very similar to the audio workflow. With the difference that when selecting Events, unless the shift key is pressed, the selection will snap to the closest Event limits, as seen on Figure 5.9.

Also, when moving video Events, they are color coded to inform if the current selection is colliding with other Events. A representation of this behavior can also be seen on Figure 5.9.
If the user wants to select both audio and video at the same time, in order to do a symmetric manipulation of these events, he can achieve this by dragging from the space between the video and the audio bar. Snapping will still apply for the video Events, as it can be seen on Figure 5.10.

Every modification is applied to both selected segments selected, until the current modification is applied or canceled.
In order to copy and paste a certain video or audio selection, the user has to select the piece to be copied and either right click the selection and choose "Copy", or press "Ctrl + C". After that, he chooses where to paste the selection to, by positioning the time cursor where the selection should be pasted (seeking is done by left clicking the timeline ruler area). And finally he right clicks and selects "Paste" or presses "Ctrl + V".

To erase a selection, he simply presses "Delete" or right clicks and selects the "Erase" option. Both these operations are described in Figure 5.11.

Figure 5.10: Selecting both audio and video segments at the same time for symmetric editing.
Finally, if the user wishes to move his Events through space, not time, this can be done by selecting the Pointer Tool and dragging the Event to desired place, as it can be seen on Figure 5.12.
Figure 5.12: Moving Events in space, instead of time.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

We managed to fine tune our editor to achieve a decent rendering performance and work over very low-bandwidth connections, thus completing the most important parts of our proposed work.

However, it should be pointed that developing GUI intensive applications can be very time consuming and prone to the creation of code that is hard to maintain, because of the intense message exchange between classes. Considering this, the QT framework was clearly the best option for developing our application, for its paradigm for message exchange is simple to understand and maintainable in the long range.

Also, developing a cross-platform software proved to be a challenging task. Using Java at first seemed like a better option, as taking our software to another platform required little to no code adaptation, contrary to using C++ with QT. Even with this disadvantage, we believe QT was the right choice for our application development, given its benefits outweighed any drawbacks.

6.2 Future Work

Our system was developed to a functional state, but there are still some improvements needed in order to publicly release it. Most noticeably, implementing the import and export features and a dedicated player.

Also, doing user tests and iterating at least one time over the user interface, would likely improve our chances of causing a good first impression when we release our software.
Bibliography


