A collaborative video annotation and sketching tool

Lienzo

Vrije University of Amsterdam
Javier Quevedo Fernández
Supervisor Anton Eliens

1. Introduction

Lienzo is a Video Annotation tool focused on letting users express their creativity in a collaborative way enhancing the media experience. You can think of lienzo as a sum of *youtube* with a paintbrush, mixed together in a seamless experience. Lienzo allows users create timestamped sketches overlaying them on top videos, it is meant to be used in educational environments or just for fun.

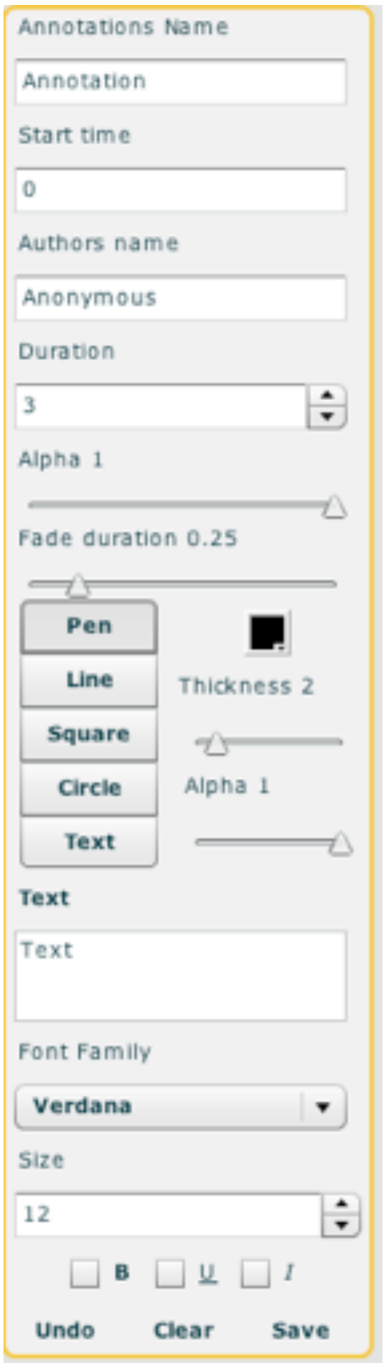
2. User’s manual

Create a new video

- Click on New Clip
- Enter the name and description
- Enter the clips youtube url or select the file to upload (Lienzo supports FLV, MP3 and any other format natively support by Adobe’s flash plugin
- Done!

Create annotations

- Select the clip to annotate.
- Jump to the moment in time in which you want the animation to start.
- Enter your name, the name and the annotation, the amount of time it should be visible, the transparency (alpha) and the fade in/out duration
- Draw the annotation using the tools.
- Click on Save.
- The annotation will be saved and the Datagrid with annotations will be refreshed including the new annotation.


Annotations Name

Annotation

Start time

0

Authors name

Anonymous

Duration

3

Alpha 1

Fade duration 0.25

Thickness 2

Alpha 1

Text

Text

Font Family

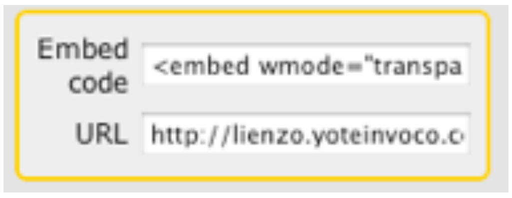
Verdana

Size

12

Share a Video

- Click on the video you want to share.
- Copy to the clipboard the Embed code (on the top right part of the page). Paste the code on your website, facebook, blog, etc.
- Alternatively you can just copy and paste the link code instead.
- Done!


Embed code	<embed wmode="transpa
URL	http://lienzo.yoteinvoco.c

3. Deployment

The following text describes how to install and deploy Lienzo on Windows systems.

3.1 Setting up the Ruby on rails environment:

1. Browse www.ruby.org and download the latest version of the Ruby language interpreter
2. Install it
3. Install Rails version 2.3.2. This can be done by opening windows command line tool (cmd.exe) and running the following command:
   - “gem install –v=2.3.2 rails”
4. Install the UUIDtools gem with this command:
   - “gem install uuidtools”
5. Install a database manager engine, Ruby on Rails includes adaptors for *sqlite3*, *mysql*, *postgresql*, and so on. Install the appropriate gem, for instance if you are using sqlite3 you should run the command:
   - “gem install sqlite3-ruby”
6. Unzip the Lenzio-XXX.zip file
7. Edit the file *config/database.yml* to set up your database details.
   For example, if you are using sqlite3 your *database.yml* file should look like:

```
# SQLite version 3.x
#   gem install sqlite3-ruby (not necessary on OS X Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

8. Run the database migrations
   - “rake db:migrate RAILS_ENV=production”
9. Start the server
   - “script/server –e=production –p=80”
     If you wish to run Mongrel rails instead of Webrick , use the following line instead
   - “mongrel_rails start –e production –p 80 –d”

10. You are now ready to browse http://localhost

Note: If you wish to use the Youtube video importer you will also need to have installed a Python interpreter, please browse www.python.org for it.

4. Overall application design

Lienzo is composed of two different programs. The Ruby on Rails backend and the Flex Player Application.

The Rails backend defines three RESTful resources: Annotations, Clips and Users.
The *user*s resource is generated with the *rails RESTful authentication plugin*.

```
<embed wmode="transparent" bgcolor="#FFFFFF" src="/flash/
Videoannotations.swf?clipUrl=http://lienzo.yoteinvoco.com/data/868102203/
video.flv&annotationsServerUrl=http://
lienzo.yoteinvoco.com&clipId=1&autostart=false" type="application/x-
shockwave-flash"  width="100%" height="665">
```

A clip contains a set of data that describes it such as the name, the description and UUID and also a filename which is the name of the *flv* file which contains the actual clip.
Each annotation includes its name and the name of the author that created it. Additionally, it also includes a *start time* that determines when exactly the annotation should be shown in the

```
<?xml version="1.0" encoding="UTF-8"?>
<annotation>
  <alpha type="float">1.0</alpha>
  <author>Javier</author>
  <clip-id type="integer">1</clip-id>
  <content>PNG ENCONDED DATA (base 64)</content>
  <contenttype nil="true"></contenttype>
  <created-at type="datetime">2009-09-03T14:48:18Z</created-at>
  <duration type="float">4.0</duration>
  <fadeduration type="float">0.689189189189189</fadeduration>
  <id type="integer">1</id>
  <name>Annotation</name>
  <starttime type="float">5.867</starttime>
  <updated-at type="datetime">2009-09-03T14:48:18Z</updated-at>
</annotation>
```

screen, an alpha value which defines its transparency, a fade duration that defines the amount of time (seconds) that the annotation should fade (in or out) until it is completely visible (or invisible). The content field holds the annotations pixels data, encoded in PNG format (base 64).

```
create_table "annotations", :force => true do |t|
 t.string  "name"
 t.string  "author"
 t.string  "content"
 t.string  "contenttype"
 t.float   "starttime"
```

```
    t.float    "duration"
    t.float    "alpha"
    t.float    "fadeduration"
    t.integer  "clip_id"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "clips", :force => true do |t|
    t.integer  "user_id"
    t.string   "name"
    t.string   "description"
    t.string   "filename"
    t.string   "uuid",              :limit => 36
    t.datetime "created_at"
    t.datetime "updated_at"
    t.boolean  "processed"
    t.string   "original_source_url"
    t.boolean  "agreed"
  end

  create_table "users", :force => true do |t|
    t.string   "login"
    t.string   "email"
    t.string   "crypted_password",        :limit => 40
    t.string   "salt",                    :limit => 40
    t.datetime "created_at"
    t.datetime "updated_at"
    t.string   "remember_token"
    t.datetime "remember_token_expires_at"
  end
```

When a clip webpage is loaded an instance of the clip player is embedded passing it needed the parameters so that it can fetch the clips annotations.

When the lienzo (*Flash)* clip player is completely loaded, it queries the *Lienzo Webservice* to obtain an Xml description of the set of annotations that belong to the clip. With this information, the *player* creates a set of queue points that trigger events when the annotation should be displayed or removed.

The *player* includes a canvas editor which is displayed on top of the video where the user can draw using the drawing tools. When the user clicks the *Save* button, the *flex* application takes a screenshot of the content of the canvas and calls the *Lienzo web service* controller to store it under the specified clip.

5. Further improvements

Many improvements could be done to *Lienzo*. Here is a small list of some of them:

- Allow the user to upload images and use them as annotations
- Use vectorial objects instead of bitmaps so the user can move them around easily and scale them without the need to erase and redraw
- Include a simple way to create animations such as moving text or shape morphs
- Allow the user to input voice annotations
- Improve the annotation list so that a user can automatically enable or disable all of the annotations at once or quickly select just the annotations of one or more authors
- Create a different standalone version of the player which can scale so that it can be embedded on external webpages using less space
- Add full screen functionality for playback or annotation creation

More importantly, if *Lienzo* was going to be deployed on a wide scale, a distributed scheme would have to be studied in order to scale the *Lienzo webservice.*