EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2009)
C. Grimm and J. J. LaViola Jr. (Editors)

Sketch-Based Interaction and Calligraphic Tags to Create Comics Online

Ricardo Lopes[1] and Tiago Cardoso[2] and Nelson Silva[2] and Manuel J. Fonseca[1]

[1]Dept. of Information Systems and Computer Science
INESC-ID / IST / Technical University of Lisbon, Portugal
[2]inEvo, I & D, Lisboa, Portugal

Abstract

Over the last years, amateur artists have been using the new Internet technologies to easily produce, share and distribute their comics. However, web applications for this kind of creation are still restricting users by adopting simple and very limiting interaction methods. Therefore, these systems do not allow the creation of visually rich comics and do not support some of the inherent needs of comics, such as the repetition of previous elements. In this paper, we propose a solution for creating comics online which builds upon the traditional principles of creating paper comics. Our approach combines a sketch-based interaction to draw comics with rich edition possibilities, computer assisted drawing techniques and a calligraphic retrieval mechanism for reusing previous elements. Experimental evaluation showed that this approach is better suited for these problems than existent applications, and that users can create more appealing and richer comics with higher flexibility and efficiency.

Categories and Subject Descriptors (according to ACM CCS): H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Information filtering H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles

1. Introduction

Creating comics online is becoming increasingly popular in an online amateur context. Currently there is a growing number of web applications that appeal to a new kind of anonymous comics artists. These authors can easily use this technology as their only mean of production and distribution to a large number of worldwide readers.

However, web applications for comics creation still show some limitations. The most frequent technique for creating comics online is the composition and arrangement of images that can be picked up from a fixed and narrow predefined group. This restricts users' creativity while failing to assist and motivate it, by not allowing users to explore their abilities to create complex and rich comic strips, similar to those created on paper. Moreover, these approaches are not helping authors in automating some needed, but time-consuming, tasks, such as the repetition of elements over one or several comic strips and the creation of simple and common shapes.

To solve these problems, we propose a new approach for creating comics online, which extends the principles of traditional comics creation by combining a rich interaction based on free hand sketching, a shape suggestion mechanism and a sketch-based retrieval method. Our solution enables users to create visually complex comic strips in a more flexible and efficient way, while keeping the easiness and quickness of this process.

The rest of this paper is organized as follows: Section 2 provides an overview of related work in comics creation, suggestion mechanisms and drawing retrieval. In Section 3 we present the overview of our approach, while in Section 4 we start detailing it, by describing the calligraphic rich editor. Section 5 explains our method for reusing previous drawings based on calligraphic tags. In Section 6 we describe the computer assisted drawing technique and Section 7 presents the results from experimental evaluation. Finally we discuss our conclusions and future work.

2. Related Work

In this work we combined three different techniques to achieve our goals. First, free hand sketching and geome-

Sketch-Based Interfaces and Modeling 2009, New Orleans, LA, August 1–2, 2009.
© 2009 ACM 978-1-60558-602-1/09/0008 $10.00

tric transformation of elements, were used to provide users with an interaction based on the principles of traditional comics creation, where there is no restriction to what an author can create, besides their own imagination and skill. Second, automatic suggestion of drawings was included to speed up comics creation. Third, the retrieval of previous elements was used to allow the reuse of visual entities along the comics strips.

From previous work done in compiling and analyzing these type of techniques, we can see that the comics medium has a specific visual nature, defined throughout the years by different artists [Mcc94, Eis96]. They have been using the same visual conventions to represent information, such as, sound, speech, movement, emotions and time, which constitutes the vocabulary of a universal language for comics creation. In these studies, authors describe the narrative structure of comics as a deliberate sequence of juxtaposed images, intended to convey information. Moreover, by analyzing these studies and other narrative structures, we observed that repeating visual elements throughout one or several comic strips is a mandatory process, while creating traditional comics. In summary, the traditional comics nature, besides being visually complex, highlights the need for reusing visual elements.

Regarding comics in digital format, we studied Williams et. al. work [WBS05], where authors propose a comics creation interface to acquire unique story-scripts from casual Internet users, called the ComicKit. This application allows users to create comics by combining multiple visual elements available into the system database. Users can pick the next element to insert into the story by searching images in the database or using keywords. ComicKit uses the most frequent technique exploited in this type of systems, the combination and manipulation of predefined images. Comeks [Sal07] and HyperComics [Gam07] are other examples of this type of solution.

However, all of these systems fail in providing adequate support for the reuse of previously created elements. Users can create elements (*e.g.* , characters, objects, etc.) by combining multiple images, but they cannot reuse them in the same or in another comic strip. Additionally, these systems fail in allowing the creation of visually complex comics, since users' creativity is limited not by their skills, but by the set of available images and by systems functionalities.

Other solutions, like the Comics Sketch [inE08], tries to stimulate users to create richer comic strips by supporting free hand drawing. Results show that users explore this kind of freedom and are able to draw more complex comics than with the previously described solutions. However, Comics Sketch has a very reduced set of editing operations, working as a simple digital sketch-pad without any form of rich interaction or retrieval of drawings.

Some of the solutions we studied are taking the process of creating comics a bit further, by giving suggestions to users. ComicKit suggests the next element to be picked, when the author selects an image to use. This system uses similarity and predefined subject-object relationships as the criteria for suggesting elements. When the user is not pleased with the suggestions made, ComicKit allows a correction through a keyword-based retrieval mechanism. Results show that users found suggestions as an intelligent and helpful method to create comics. Also in the area of calligraphic interfaces, are solutions that use suggestions lists to present possible elements for users to include in their drawings, or to solve ambiguities during the drawing creation, such as Igarashi's Chateau [IH01] and Pereira's GIDeS [PJBF03]. Suggestions list are a kind of non-intrusive context-based dynamic menus, containing the different interpretations of the sketched input. In the area of drawing retrieval there are some recent works which are able to retrieve complex vector drawings, such as, clip arts [SF09], technical drawings [HR08] or free-form hand-drawings [LC03]. These approaches perform drawing retrieval using sketches as queries and by analyzing drawings contents.

Looking at the majority of comics creation systems, we can observe three things. First, these solutions are restricting users interaction and, by consequence, the visual richness of resulting comics strips. Second, few systems offer computer assisted techniques, such as suggestions mechanisms, to increase the efficiency and flexibility of the creative process. Finally, there is no support for the reuse of visual elements, a need strongly inherent to the traditional comics creation process. Our solution overcomes the main drawbacks of these works by offering a richer visual composition and a simpler mechanism to include previous drawings in comic strips.

3. System Overview

Our approach offers a new methodology for creating comics online, by combining free hand drawing and editing, a sketch-based retrieval mechanism based on calligraphic tags and computer assisted drawing. To implement this approach, we defined an architecture for a web application composed by two main components: the editor module, which is on the client side, and the retrieval module, which is a server application.

The editor module is responsible for the interaction with the user and allows the creation and edition of comics. This module has a component-oriented architecture, where each component can be viewed as a tool set, with specific responsibilities and interface. The combination of all of these individual components leads to our solution. In this section, we stress two of these components: the Canvas and the Expectation List. The Canvas is the main sketching area and supports not only the creation of new comics, but also works as the interface with the retrieval and computer assisted drawing mechanisms. The Expectation List is the component that presents possible visual elements produced by these two mechanisms. The List not only shows results, but imple-

ments and supports all the logic behind the computer assisted drawing mechanism. Indeed, it uses the CALI a recognition library [FPJ02] to identify geometric shapes and calligraphic commands from users' sketches.

The retriever module is responsible for supporting the execution of all the mechanisms and algorithms to retrieve previous drawings created by the user. We adopted our previous sketch-based retrieval approach [SF09] to create the retrieval engine. Users interact with the retriever module through calligraphic queries that are matched against drawings in a database. Then, the most similar results are presented to the user in the Expectation List.

Here, we extended our previous approach by defining and implementing a new concept for retrieval of visual elements, called calligraphic tags. Now, users can save and classify drawings by associating simple sketches, the tags, to them. This way, users can find and retrieve previous drawings by sketching the correspondent calligraphic tag. As in the "normal" tagging mechanism, users can associate the same calligraphic tag to several drawings and multiple tags to the same drawing.

4. Calligraphic Comics Editor

To support the cornerstone of our solution, the rich creation and edition of comics, we created an editor module that is responsible for the interaction with a large set of techniques and operations. This combination leads to a new method for creating comics with more flexibility, efficiency and freedom than current web applications. Our editor module supports a set of functional requirements capable of enabling users to create more complex and rich comics in a simpler way, such as, free hand drawing, elements transformations, visual layers, strokes grouping, textual elements, zoom and computer assisted drawing.

We defined a component oriented architecture for the editor module, which distributes these functional requirements by different individual blocks that are independent and configurable. Our prototype for comics creation was built by instantiating and configuring all of these individual components. Due to the flexibility and extensibility of our component-oriented module, we can build other applications with different requirements than our editor, by instantiating a subset of the available components or even extending the editor with new ones.

Figure 1 illustrates our web application for comics creation, showing the different components of the editor. The Canvas captures the user mouse gestures and converts them into drawings. Later on, the position, size or rotation of these drawings can be changed. This component is also responsible for the submission of sketched queries to the retrieval system. The Panel List is the component that allows users to manage the current comic strip. The Canvas and the Panel List communicate in order to update the current scene.

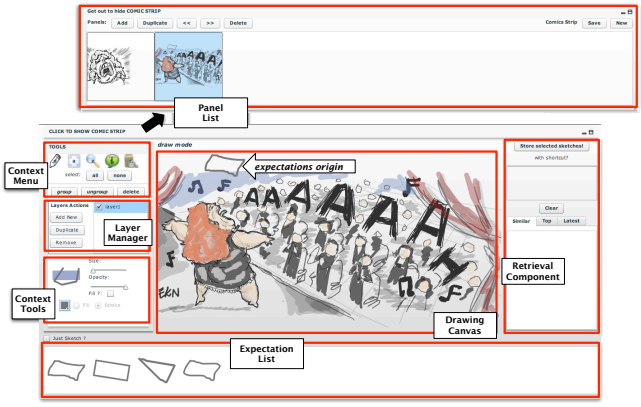

Figure 1: *Comics Rich Editor Interface*

For the internal representation of created sketches we use the SVG format. This way, we maintain the independence between stored drawings and our web application. To achieve this goal, we defined our Canvas component as an SVG editor, with the ability to read, draw and edit SVG individual elements, in real time. This allows the communication between the Canvas, the Panel List or even the Retriever module using exclusively this format. Indeed, with this solution, we can import any SVG element to the Canvas, use it directly or modify it to include in the comic strip.

The Context Menu is responsible for changing the active edition mode. Modes control the application behavior and decide, for example, what to do with sketches (draw, define calligraphic tags or retrieve). We included edition modes for free hand sketching, sketched actions, zoom, text input, calligraphic tags association and retrieval. The Layer Manager allows users to distribute drawings on a pile of overlapped visual layers. Finally, the Expectations List provides the interface and mechanisms for computer assisted drawing through sketch recognition.

5. Retrieving Comics using Calligraphic Tags

In this section, we describe the concept of calligraphic tags for classification and retrieval of drawings, as a more flexible way of reusing previous visual elements for comics creation. Our solution takes advantage of the use of a calligraphic interface, to provide a seamless integration between creation, edition and reuse, since users can sketch to define new visual elements or to retrieve previous drawings.

5.1. Concept

From our previous work on drawing retrieval, we observed that users liked the interaction paradigm very much (use of sketches as queries) in contrast to more traditional approaches, based on query-by-example [SF09]. This previous retrieval solution uses an automatic visual classification

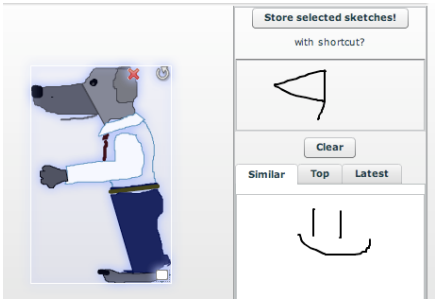

Figure 2: *Example of classifying using calligraphic tags.*

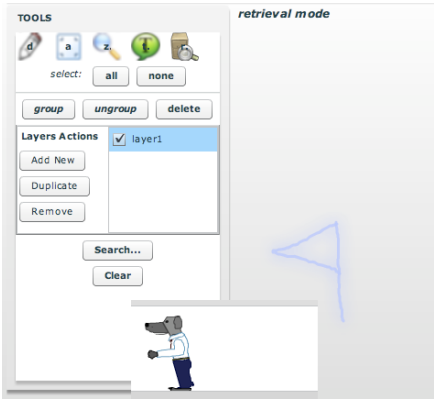

Figure 3: *Example of retrieving using calligraphic tags.*

scheme based on geometry and spatial relationships (topology) to describe drawings and sketched queries. The approach presented in this paper is based on our previous work on drawing retrieval. We still use geometry and topology to describe drawings and queries contents, but we explore a new way of accessing drawings using calligraphic tags.

While analyzing experimental evaluations from previous works, we noticed that users typically sketch a very small set of shapes to define a query, making the comparison between a very simple query and a complex drawing difficult. We believe that this behavior can be explained by users expectations. Typically, users draw the minimal common elements, but on the other hand, they are expecting to get a large set of results semantically similar to the sketched query. For example, if a user is trying to find an air plane, he sketches a query with the basic common features he knows from every air plane. With this query, the user is creating a visual abstraction of all air planes, specifying a personal visual meaning for this kind of objects. Unfortunately the user is not aware that there are a large number of air plane drawings that do not share the geometric or spatial similarity with his mental and visual abstraction.

To overcome this, we decided to extend our drawing retrieval scheme by making the creation of visual abstractions explicit, *i.e.* , allowing the creation of calligraphic tags, somehow similar to what Mark Gross [Gro96] did for the retrieval of architectural buildings. With this scheme, when users store any drawing for future reuse, they can bind a simple sketch to it. Users are free to create a simpler abstraction of the original drawing or any other association that they deem logical. Figure 2 illustrates an example of this behavior, where users save the selected element by drawing an additional shape (top right of Figure 2) and store both. To retrieve the original drawings, users can now sketch a query similar to the calligraphic tag (see Figure 3).

Our application also provides a set of suggestions for calligraphic tags, similar to what users are drawing. This can help users classify drawings using previous tags (bottom right of Figure 2) or to avoid similar tags, thus reducing ambiguity. Our suggestions include previous tags similar to the one being created, the list of most used and the most recently created.

As illustrated in Figure 2, previous calligraphic tags can be used to classify more than one drawing, allowing users to group, under the same tag, any drawing they consider similar visually or semantically (*e.g.* , cars, people, dialog balloons, love, hate, etc.). With the use of calligraphic tags, original drawings are not only described by their geometry and topology, but also by the meaning they have to users.

To retrieve the visual elements stored, users can sketch a query similar to the calligraphic tag (Figure 3) and then select the desired drawing from the returned results (bottom of Figure 3). Retrieved drawings can then be edited using the potentialities of the editor (*e.g.* resize, change head or hands position, etc.). Additionally, this new version of the reused element can be stored for later reuse.

It is important to note that the use of calligraphic tags is an extension to the previous retrieval mechanisms, which describes the content of drawings using topology and geometry. Users can still sketch traditional queries, similar to drawings, (and not to tags) to retrieve past visual elements. When users submit a sketched query, this is used to search in the set of calligraphic tags and simultaneously in the set of drawings, being the final result a combination of both. This way, our solution remains valid for users who do not see advantages in calligraphic tags or are not willing to classify visual elements by sketching yet another drawing (the tag).

5.2. Retrieval Architecture

The tag-based retrieval mechanism is supported by the retriever module, a server side application that interacts with the editor. The retriever module has two layers that support the functionalities described above. One is the retrieval module, which uses our existing system for retrieving clip-art drawings, called Indagare [SF09]. This, offers support for our traditional retrieval mechanism based on topology and

geometry extracted from drawings content. The other is a new layer developed to extend the traditional solution, with calligraphic tags, and is responsible for creating and recognizing tags.

This layer is a service module that offers three functionalities: new drawings classification (with or without calligraphic tags), previous tags suggestion and previous drawings retrieval. It allows *n* to *n* associations between drawings and calligraphic tags. The interpretation of such associations is also supported in this layer, which can filter and discriminate calligraphic tags and original drawings.

Our new service layer uses Indagare to implement the retrieval functionalities for calligraphic tags. It classifies and finds similar calligraphic tags, either in the retrieval and suggestion cases. This layer is invoked by the editor module and by a special retrieval component (bottom right of Figure 2), when users want to classify or retrieve drawings into the current comic strip. Figure 4 illustrates the architecture of our retriever module, integrated with the Indagare server.

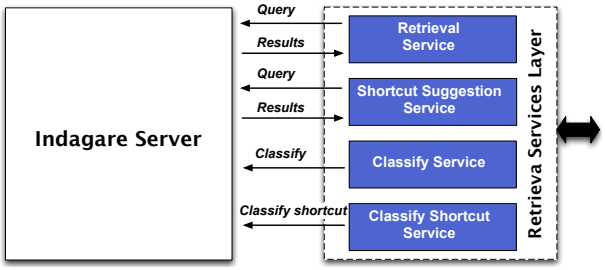

Figure 4: *Architecture of the Retriever Module: Indagare and services layers.*

6. Computer Assisted Drawing

In our solution we included mechanisms to assist users while drawing or transforming new content for the current comic frame. Our goal was to relieve users from some of the time and skill consuming tasks, while providing more natural and intelligent ways of interacting and creating. To accomplish this we adopted computer assisted drawing (CADr) techniques that provide usability gains to our solution. We implemented two mechanisms: recognition of calligraphic actions and an intelligent suggestions list.

6.1. Calligraphic Actions

Calligraphic actions are simple gesture commands that after being recognized and processed trigger specific actions. They provide a natural and continuous way of interacting with the system. Thus, comics creators can perform complex and flexible operations on specific drawings, by using the same interaction paradigm as the one they use while sketching. Our set of calligraphic actions allow the basic operations in the context of a drawing application. For instance, stroking over drawings several times, leads to deleting them; sketching a lasso around elements selects them; drawing a line creates the axis for performing a mirror copy of selected drawings; and finally, sketching a 'V' character creates a copy of the selected drawing. Variations in the 'V' size have a direct relation to the size of the copy. Copy and retrieval operations are compatible due to their different goals and scopes. Copy is for duplicating individual elements inside the same frame. Retrieval is for reusing previously stored content (one or several elements) across one or several different comic stories. Figure 5 illustrates an example of a calligraphic action for selection.

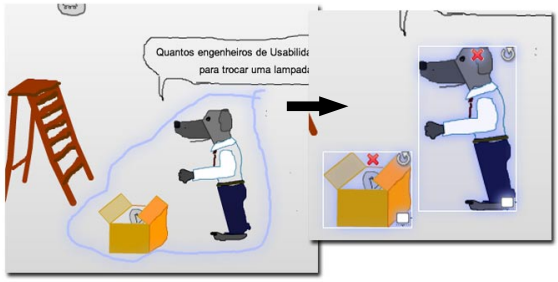

Figure 5: *Calligraphic action: sketching the lasso (left) selects elements (right).*

6.2. Suggestions List

We included a mechanism for shapes suggestion in our solution, which is able to recognize sketches as shapes geometrically similar to what users drew. Identified shapes are presented to users as suggestions in the suggestions list. When accepted, the suggestion replaces the original sketch.

With this method, we aim to assist users in the task of drawing new elements and consequently improve their usability experience. Users can take advantage of this suggestion mechanism to achieve high quality representations of specific shapes, independently of their drawing skills. Although, we are suggesting "perfect" shapes to users, we would like to point out that this suggestion mechanism is not restricting users' freedom or creativity, since these shapes are optional and are only suggested for simple geometric elements (*e.g.* circles, triangles, etc.). Furthermore, if users decide to adopt a suggested shape, they still can transform it as they do on a normal sketch.

Our suggestions mechanism is able to recognize circles, ellipses, lines, rectangles, triangles, closed polygons and bezier curves. When users draw a sketch, suggestions could include several of these elements, depending on the characteristics of the sketch. Suggestions are shown in the Expectations List component of the editor module. Figure 6 illustrates an example of a possible interaction with the Expectations List and the suggestion mechanism.

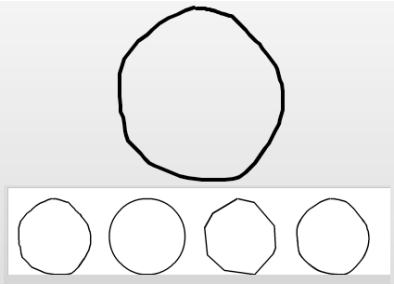

Figure 6: *The sketch (above) is recognized as one of the (bellow) possible shapes: original, circle, polyline and spline*

6.3. CADr Architecture

The Expectations List centralizes all the implementation of the calligraphic actions and suggestions mechanisms. When it is present and activated in the web application, it listens to the sketching events in the canvas and responds to them. This behavior means that all of the CADr functionalities are executed exclusively on the client-side of our solution (editor module). We achieve performance gains by allowing the client application to process all the sketch recognition and actions algorithms.

The Expectations List has a layer responsible for processing and recognizing sketches created on the Canvas. This layer is formed by four agents able to extract geometric features from sketches and computing the appropriate suggestions or actions. The Expectation List invokes the correct agents, depending on the user's intentions or the application's context mode. Available agents are: Cali, Filter, Actions and Retrieval Agent.

The Cali and Filter agents are responsible for generating shapes for the suggestion mechanism. The first implements the recognition of shapes, based on the CALI algorithms, and identifies circles, ellipses, lines, rectangles, triangles and polygons. The Filter agent is based on an open source library for the modelling of parametric curves [Ams08] and is able to recognize bezier curves and closed curve-shaped elements. The Expectations List component is responsible for compiling the suggestions calculated by these two agents, present the results, capture the user's selection and, if necessary, replace the original sketch by the selected shape.

The Actions agent does not update the Expectations List, since the result of the recognition process is an action and not a shape. After identifying the sketched command, the operation is performed on the target element in the Canvas. Identification of this action gestures is also done using our sketch recognition library.

7. Experimental Evaluation

We developed a prototype for creating comics online, which combines our new approach based on calligraphic tags, for reusing visual elements, a free hand drawing editor, to support the rich creation of comics and CADr mechanisms to reduce creation time and improve simple shapes quality.

To evaluate our overall solution, we performed experimental evaluation with users using the resulting prototype (called ReCCO). We intended to evaluate two aspects: first we wanted to test our calligraphic tags mechanism in the context of amateur comics creation, against the most functional solution online. Second, we wanted to evaluate our overall solution and see if our editor module and its CADr solutions improves the flexibility and efficiency during the creation of comics online.

7.1. Experience Description

We conducted a series of user tests, in which participants performed a task in our prototype (ReCCO) and in another application, Comics Sketch [inE08]. The task consisted in the illustration of a predefined scripted story. The plot was created to encourage users to perform the task using all the components and mechanisms present in our solution. We give users a storyboard composed of six frames, containing just the text of the story. Frames in the story included some of the elements from the previous one, but with variations in size or point of view. Before starting the tests, we performed a short demo of both programs, without advising users on which specific features they should use. Each individual test, on the ReCCO application, started with an empty database of visual elements.

Users created the same story in both applications. However, while half of them created first in ReCCO and then in the Comics Sketch (group R/CS), the other half did it in the reverse order (group CS/R). With this we tried to avoid biasing result. After users perform the task, we did a questionnaire and an interview to collect their feedback about the applications evaluated.

We selected 8 participants who represent the adequate user profile of our application. We preferred to perform tests with few "specialized" users, which are used to create comics (in an amateur environment), than to collect a larger number of users without any experience on comics creation.

7.2. Results

Figure 7 (top) shows the time values for creating each individual frame in the comic strip, while Figure 7 (bottom) presents the average of total time needed to create the entire comic strip. These results show that there is a significant difference in times for individual frames and for the complete strip. On average, users spent less 15 minutes (32% of the total time) creating the complete comic strip in the ReCCO prototype than in Comics Sketch. From observations and interviews, we noticed that this difference was due to the suggestions, retrieval and transformation techniques presented

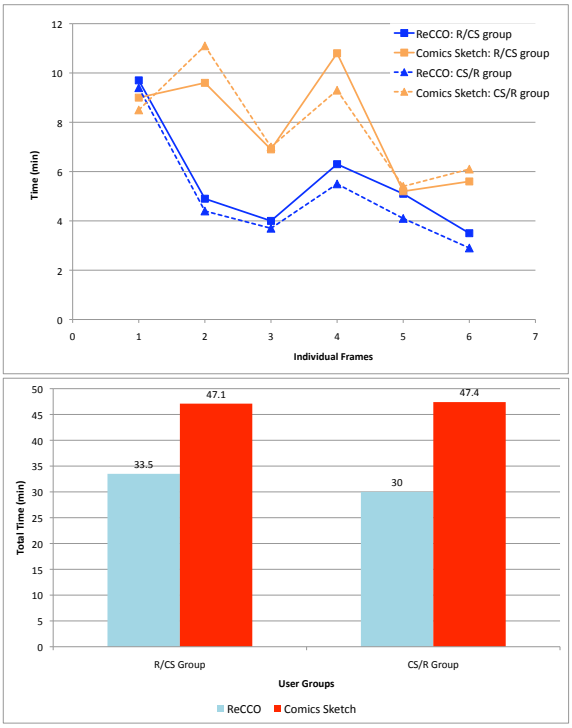

Figure 7: *Creation times, per frame (top) and for the overall strip (bottom).*

in our editor, since these allow a more powerful and efficient interaction.

Looking at the individual frames (Figure 7 top), we can see that, with ReCCO, users spent more time creating the first frame than the following ones. This difference is due to the retrieval mechanism. In the first frame, the majority of users spent time not only drawing the visual content, but also creating the calligraphic tags and storing them for later use. In the following frames, these saved elements were already created, and users only had to retrieve and reuse them, saving time.

We then tried to collect some measures for the retrieval mechanism. To that end, we first defined a set of expected results through conditioned pilot-tests, which were performed to maximize the usage of our retrieval mechanism. Figure 8 shows the results achieved by the eight users during the execution of the task, in our system, in comparison to the expected values. These results account the number of occurrences for storing elements, accepting tag suggestions (when creating a tag), retrieving (and reusing) elements and building tag-based or traditional queries. As we can see, we obtained results similar to the expected, regarding the number of stored and retrieved elements. These experimental values demonstrate that the calligraphic retrieval of previous drawings was understood and fully explored by users.

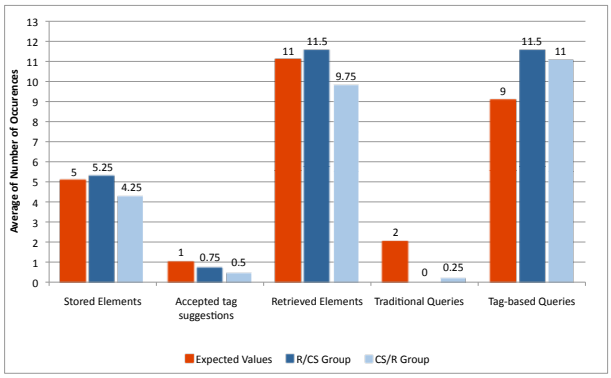

Figure 8: *Results for the evaluation of the retrieval mechanisms.*

In what concerns the comparison between the traditional retrieval mechanism and the calligraphic tags, our results are better than expected. All users chose to always store drawings using a tag and, therefore, the majority of queries were tag related. Users told us that a simpler and more specific sketch is an easier and faster option for creating retrieval queries, specially when one has to repeat the same drawing several times in a comic strip. Answers to our questionnaires showed that the calligraphic tags mechanism was well received, with 88% of users stating that tags made the drawing retrieval easier and 75% agreeing that they allowed a better organization of stored elements. Even the 25% who considered creating tags difficult, chose to do it due to its advantages.

The values obtained for the number of accepted shortcuts suggestions were slightly smaller than expected. We believe that further tests would improve these results.

We also measured users satisfaction through our questionnaires and interviews. In what regards usability, all the participants chose our prototype as their favourite one. The main reasons for that were the acknowledgement of an easier and quicker creation process, supported by the retrieval and CADr mechanisms. 75% of the participants considered the suggestions mechanism as essential and 87.5% of them think that calligraphic actions are an easier way of interaction.

Finally, the vast majority of the participants chose the comic strip created with ReCCO as their favourite and the most visually pleasing. Through observation we also noted that, with our prototype and due to the presence of more operations, users became more demanding and detail oriented with their comic strips. Figure 9 shows the same frame created by the same user, using ReCCO and Comics Sketch.

8. Conclusions and Future Work

In this paper, we presented a new approach for creating comics online. This solution combines a rich editor,

Figure 9: *Example of frames created during experimental evaluation with ReCCO (top) and Comics sketch (bottom).*

based in free hand sketching, with computer assisted drawing techniques and a calligraphic retrieval mechanism. For this mechanism, we proposed a new concept in the drawings retrieval field, the calligraphic tag. Experimental results showed that our approach enables users to create comics in a shorter time than current solutions, while achieving visually richer and more complex scenes.

The rich edition (with calligraphic actions) and the suggestion mechanisms were fully explored, with users quickly adopting this kind of interaction. In fact, the combination of these enabled users to create comics easily while being simultaneously faster and more (visually) ambitious.

We can also concluded, from the results, that the calligraphic retrieval of previous drawings is a valid solution for the inherent need of repeating elements during a comic strip. Indeed, the calligraphic tag-based retrieval method proved to be the best solution for reusing elements.

In what regards future work, we observed that the coexistence of both retrieval mechanisms (traditional and tag-based) caused some problems to users. Sometimes, tag-based queries were recognized as occurring in part of some complex drawings. This happens because tags were normally created as very simple shapes that can easily occur in a complex drawing. To solve this, we can adapt our approach to exclusively use the calligraphic tags mechanism for the retrieval of drawings. However, further studies would have to be conducted to account all the usability issues.

We also have some questions with the scalability and standardization of the calligraphic tags mechanism. With a continuous and long use of the calligraphic tags and its *n*-to-*n* associations with drawings, users might create large sets of information that overloads their memory. We believe that a future direction can be sharing tags and drawings between all users, to try to define one global vocabulary for the most simple and common elements.

References

[Ams08] AMSTRONG J.: Singularity - parametric curve library. *http://www.algorithmist.net/*, 2008. as seen in September 2008.

[Eis96] EISNER W.: *Graphic Storytelling*. Poorhouse Press, 1996.

[FPJ02] FONSECA M. J., PIMENTEL C., JORGE J. A.: CALI: An Online Scribble Recognizer for Calligraphic Interfaces. In *Proceedings of the 2002 AAAI Spring Symposium - Sketch Understanding* (Palo Alto, USA, Mar. 2002), pp. 51–58.

[Gam07] GAMES P.: Hypercomics. *http://www.hypercomics.com*, 2007. as seen in October 2007.

[Gro96] GROSS M. D.: The electronic cocktail napkin - a computational environment for working with design diagrams. *Design Studies 17(1)* (1996), 53–69.

[HR08] HOU S., RAMANI K.: Structure-oriented contour representation and matching for engineering shapes. *Computer Aided Design 40*, 1 (2008), 94–108.

[IH01] IGARASHI T., HUGHES J. F.: A suggestive interface for 3d drawing. In *14th annual ACM symposium on User interface software and technology (UIST'01)* (2001), ACM, pp. 173–181.

[inE08] INEVO: Comics sketch. *http://mainada.net/comics*, 2008. as seen in December 2008.

[LC03] LEUNG W. H., CHEN T.: Hierarchical Matching for Retrieval of Hand-Drawn Sketches. In *IEEE International Conference on Multimedia and Exposition. (ICME'03)* (July 2003), vol. 2, IEEE Press, pp. 29–32.

[Mcc94] MCCLOUD S.: *Understanding Comics*. Perennial Currents, 1994.

[PJBF03] PEREIRA J. P., JORGE J. A., BRANCO V. A., FERREIRA F. N.: Calligraphic interfaces: Mixed metaphors for design. In *10th International Workshop on the Design, Specification and Verification of Interactive Systems (DSV-IS'03)* (June 2003), pp. 154–170.

[Sal07] SALOVAARA A.: Appropriation of a mms-based comic creator: from system functionalities to resources for action. In *SIGCHI conference on Human factors in computing systems (CHI'07)* (2007), ACM, pp. 1117–1126.

[SF09] SOUSA P., FONSECA M. J.: Geometric matching for clip-art drawing retrieval. *Journal of Visual Communication and Image Representation (JVCI) 20*, 2 (2009), 71–83.

[WBS05] WILLIAMS R., BARRY B., SINGH P.: Comic kit: acquiring story scripts using common sense feedback. In *10th international conference on Intelligent user interfaces (IUI'05)* (2005), ACM, pp. 302–304.