JEmblazoner: Konstruktion von Wappenbildern aus textuellen Beschreibungen

JEmblazoner: Construction of Coats of Arms Images from Textual Descriptions

Christian Breiteneder[1], Horst Eidenberger[1] and Manuel Wasinger[2]
Vienna University of Technology, Institute of Software Technology and Interactive Systems
Favoritenstrasse 9-11 – 188/2, A-1040 Vienna, Austria
Tel.: +43-1-58801-18802, Fax: +43-1-58801-18898
[1]E-mail: {eidenberger, breiteneder}@ims.tuwien.ac.at, Internet: http://www.ims.tuwien.ac.at/
[2]E-mail: a9402833@unet.univie.ac.at

Zusammenfassung: In diesem Beitrag wird die Softwarekomponente JEmblazoner beschrieben. JEmblazoner ist eine Java-basierte Benutzerschnittstelle für die inhaltsorientierte Suche in Datenbanken mit Wappenbildern. Mit JEmblazoner ist es möglich, aus textuellen Beschreibungen (sogenannten Blazons) Wappenbilder abzuleiten, die dann für die inhaltsorientierte Suche verwendet werden können. Unter inhaltsorientierter Suche (englisch: Content-based Image Retrieval, CBIR) versteht man die automatische Suche in Bilddatenbanken anhand qualitativer Bildmerkmale. Solche Bildmerkmale (englisch: Features) können sein: Farben, Farbverteilungen, Muster, Umrisse, etc. Die Suche erfolgt zumeist indem der Benutzer ein oder mehrere Beispielbilder angibt, für die er möglichst ähnliche Bilder in der Datenbank finden möchte. Da ein solches Beispielbild nicht in allen Fällen verfügbar ist, stellen wir für den Anwendungsbereich der Wappensuche mit JEmblazoner auch eine textorientierte Benutzerschnittstelle zur Verfügung. Der Beitrag beschreibt den Aufbau und die Implementierung von JEmblazoner ergänzt durch eine Reihe von Beispielen.

Abstract: This paper describes the free software component JEmblazoner. JEmblazoner is a Java-based user interface for Content-based Image Retrieval (CBIR) in databases of coats of arms images. JEmblazoner allows to create coats of arms images from textual descriptions (so-called Blazons) that can be used as input for CBIR queries. In CBIR image databases are queried by qualitative features without the assistance of human annotations. Such features can be color, color distributions, textures, shapes, etc. Usually, the user issues a query by selecting example images that look similar to those he wants to retrieve. Since in some situations suitable example images may not be available, we provide a textual interface from which search images may be generated. In this case JEmblazoner is – for the application domain of coats of arms – an extension to perform content-based retrieval from textual input. The paper describes the design and implementation of JEmblazoner as well as a number of rendered examples.

1. Introduction

The increasing number of digital libraries and repositories with visual content requires powerful solutions for content-based image retrieval (CBIR). In most approaches image features are extracted, stored in a database and compared with the features of a particular search image. The result set of a query should only contain images that show a minimal difference in similarity to the example image given (nearest neighbor searches). Research usually addresses topics such as reduction of feature space dimensionality and multidimensional data structures and search methods.

The approach presented in this paper addresses a problem that is very rarely attacked: Almost all approaches assume that a search image is easily available which of course is not the case.

This paper presents an approach in which a search image can be generated from a textual description and describes JEmblazoner, a text-based renderer for coats of arms images. JEmblazoner can be used as a user interface for the Content-based Image Retrieval (CBIR) system we developed for coats of arms (see [1]). The work summarized in this paper was performed by Manuel Wasinger in preparation of his diploma thesis [8].

The remainder of this paper is organized as follows: Section 2 gives an overview over background and related work, Section 3 discusses the blazoning process and the user interface and Section 4 shows some examples. In Section 0 the software design of Jemblazoner is described and Section 6 concludes the paper.

2. Background

In this section we give a short introduction to heraldry and blazonry as well as CBIR – the intended major application domain for JEmblazoner.

Basically, heraldry is a set of rules for the creation and description of coats of arms (see [6]). These rules define the colors that may be used (so-called tinctures and furs, essentially white, black, yellow, red, blue, green and purple), the allowed shapes of shields and layouts of fields (often divided by so-called ordinaries and sub-ordinaries) and the types and shapes of icons (so-called charges, e.g. lions, crowns, etc.). Additional rules define the texture substitutes of tinctures in seal prints. Blazonry is a standardized language with a very strict grammar and therefore ideal for textual input in JEmblazoner.

As pointed out above, JEmblazoner is intended to be used for CBIR. CBIR approaches try to retrieve those images from a database that are semantically similar to a given query image. Usually, queries are defined by selecting one or a group of example images. CBIR uses feature extraction functions to extract (visual) properties from images. These properties are stored in numerical feature vectors (so-called Descriptors). Feature extraction functions can be split into methods for color extraction (color histogram, dominant color), texture extraction (statistical and structural methods), shape recognition (e.g. faces, sketches, etc.) and other, probably more domain-dependent image properties. During a query, the feature vectors of images in the database are compared to the feature vectors of the given example image(s). Widely applied distance functions are the city block distance and the Euclidean distance. Figure 1 depicts the principal architecture of a CBIR system. See [2] and [7] for more information on content-based image retrieval.

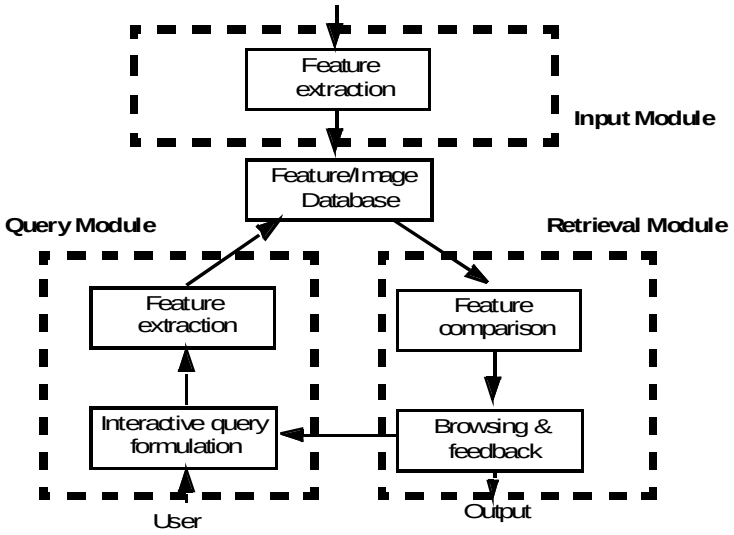

Figure 1. General architecture of a CBIR system.

In earlier work, we have implemented a CBIR system for the retrieval of coats of arms images. It should help heraldry experts in their daily work (this profession still exists, for example in the Austrian National Library). The system is based on the standard CBIR system QBIC developed by IBM [4]. For coats of arms retrieval QBIC was extended by tailor-made methods for visual feature extraction from images and similarity measurement. See [1] for more information. Unfortunately, the QBIC user interface supports just Query by Example (QbE) as the only querying paradigm. In QbE a query is defined by selecting one or a group of images that look similar to the images the user is interested in. Other querying paradigms, e.g. Query by Sketch and Query by Text are not supported. To overcome this limitation for the application domain of coats of arms we developed the Query by Text user interface JEmblazoner. As pointed out above, JEmblazoner uses Blazons to create example images that are used for the actual querying process.

3. Blazoning and the User Interface

Before we actually start discussing the user interface of Jemblazoner we define four terms that are used in heraldry and seem to be important in the context of this paper: blazonry, blazon, blazoning and emblazon. The name of the language heralds used in the middle ages to describe the arms of knights in tournaments is *Blazonry* (see [5] and Section 2 for details). *Blazon* as noun represents words that are used to describe coats of arms. The verb *to blazon* stands for the process of describing a coat of arms, whereas *emblazon* refers to the image (coat of arms) that is specified by a blazon (hence the name of the software tool).

A standardized form of blazonry began to develop in the 13th century. Prior to this, blazons *were* simple descriptions of armory with few details. The main reason for the standardization of blazon was to make the heralds' task easier. In many ways blazonry is like a foreign language: it has vocabulary and grammar, both of which contribute to the meaning of a blazon. We simplified this language in order to be able to process blazons automatically and to generate the according emblazons.

The front-end of the program (see Figure 2) is supplied both as a stand alone application and as an applet runnable in a Java enabled browser. The user interface is of very simple structure. It provides a text field for the input of the text to be emblazoned and a few controls for user defined parameters. There are two radio buttons for the choice of creating a coloured blazon or a blazon of black and white using the system of hatching by Silvester Petra Sancta. In addition the user has also the choice whether he wants to use antialiasing. This creates a smoother image using more colours with the one small disadvantage that the image can be saved only in JPEG format.

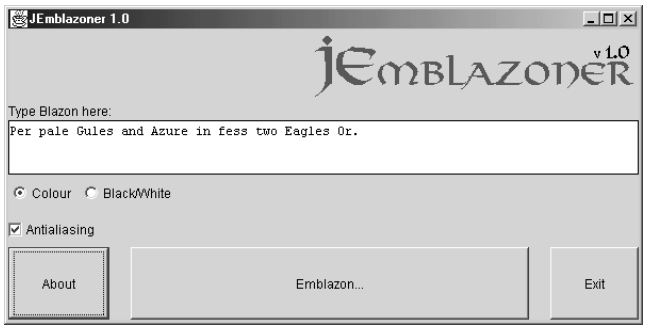

Figure 2. User interface of JEmblazoner (stand alone application).

Pressing the “Emblazon...” button or using the return key starts the process of emblazoning. After the analysis of the textual input the blazon is drawn and a new window opened that contains the resulting image (emblazon) and possible warnings or error messages (see Figure 3). The menu of this window offers the possibilities of printing or saving the created blazon. Printing is done by the printing system of the operating system. Available image formats to save the blazons are JPEG and GIF.

Figure 3. JEmblazoner's user interface containing the result of a user request.

A major advantage of JEmblazoner is the fact that there is no limitation to charges and patterns explicitly provided by the program. JEmblazoner is able to use any images that are supplied by the user in certain program directories and that meet some requirements in size and colour.

4. Some Examples

The following examples show different coats of arms illustrating the possibilities of blazoning. All images were created with JEmblazoner. The example in Figure 4 shows the coat of arms for the Blazon "*Per fess Or and Gules in chief a Mullet on the second*" in color and as seal print. "Per fess" defines the layout, "Or" (yellow) and "Gules" (red) are tinctures, a "Mullet" is a star and "in chief of the second" defines the position of this star.

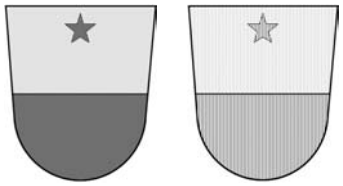

Figure 4. Coat of arms for the Blazon „Per fess Or and Gules in chief a Mullet of the second" in color and as seal print.

In addition to using specific layout information (as for example, "per fess") it is also possible to make use of patterns for the design of the blazon. Figure 5 shows the blazon "*Lozengy Azure and Argent three roses proper*." where "*lozengy*" is the pattern used for the shield.

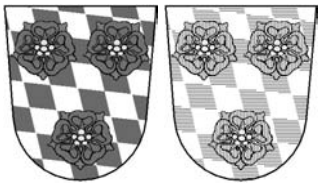

Figure 5. Coat of arms for the Blazon „Lozengy Azure and Argent three roses proper" in colour and as a seal print.

There are two more points worthy to mention in this example. One is that JEmblazoner does not only support explicit colour but also understands heraldic terms like "*proper*" or "*counterchanged*". The other point is that Jemblazoner also takes notice of default positions for charges. The example only says "*three roses*". As there is no information about positioning the roses they are supposed to be in their default position which for three charges is "two and one".

A coat of arms that is made up of two shields is shown in Figure 6. "*Impaled fesswise enhanced 1st Or two bars Sable 2nd gules a pall Sable a lion statant Or langued Gules*." blazons a coats of arms consisting of two shields put together horizontally ("*fesswise*") where the dividing line is somewhat above the middle as the blazon says "*enhanced*". Each shield is blazoned separately.The upper shield is of simple yellow (Or) containing two bars in black (Sable). On the second shield the black ordinary "*pall*" is drawn upon a red (Gules) ground. In addition there is a yellow lion with a red tongue ("*langued Gules*") standing on all four of his paws ("*statant*").

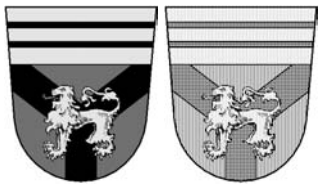

Figure 6. Coat of arms for the Blazon "Impaled fesswise enhanced 1st Or two bars Sable 2nd gules a pall Sable a lion statant Or langued Gules." in colour and as a seal print.

5. Software Design

Figure 7 describes the essential elements of the software design of JEmblazoner. The main class *JEmblazoner* draws the graphical user interface and manages user interaction. *BlazonFrame* is the main class for text analysis and the rendering process. It uses an instance of *TinctureManager* and *Tincture* objects for color management. Tinctures are derived from class *Color*. Rendered coats of arms images can be stored as JPEGs with *SaveJPG*.

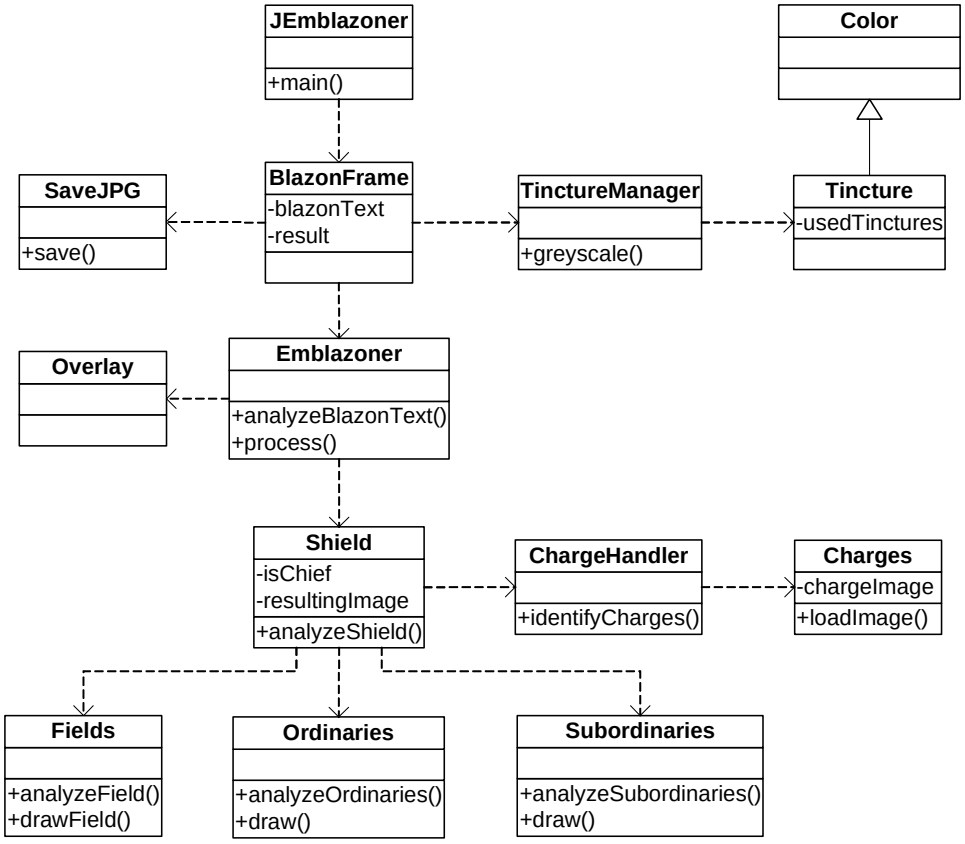

Figure 7. JEmblazoner class diagram.

Each element of a shield can easily be interpreted as a layer of the image. The implementation of JEmblazoner makes use of this fact. Blazons can either be a representation of one single shield or may be a combination of two or more shields. Each of these shields can again be divided into various forms of fields which again may contain ordinaries, subordinaries or charges.

Emblazoner is the main class of the rendering process. Each subclass – *Shield* for shield drawing, *Fields* for field layout, *Ordinaries* and *Subordinaries* for rendering of ordinaries resp. subordinaries and *ChargeHandler* for the management of charges – is responsible for text parsing of its relevant *Blazon* elements. The *ChargeHandler* uses instances of *Charges* for charge loading and rendering. Instances of *Overlay* are used to manage some special forms of charge layers. Each shield object implements an *analyze()* method for text analysis and several *draw()* methods for rendering. The *TinctureManager* is responsible for color-to-greyscale conversions. The final coats of arms image is stored as a resource of *BlazonFrame*. This framework can be extended by additional classes for tinctures and charges.

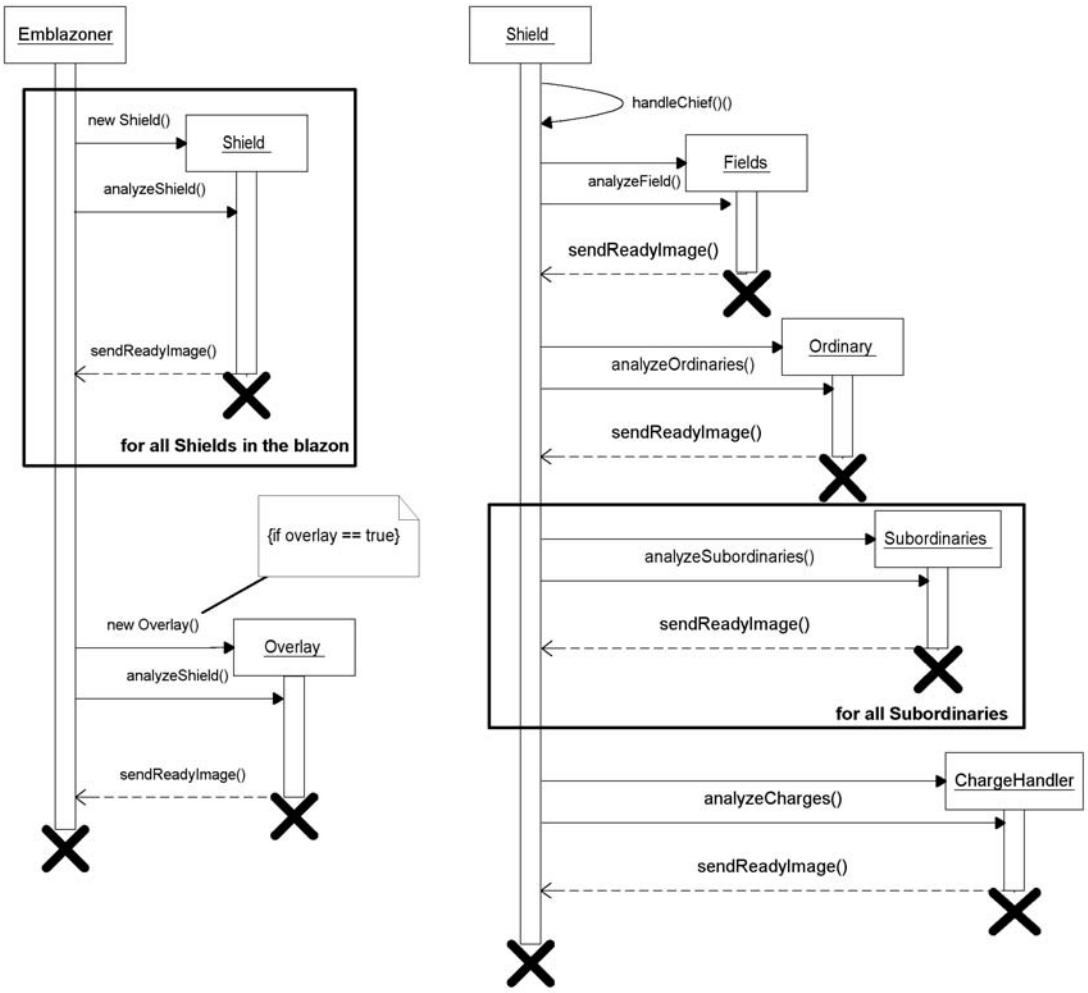

Figure 8. Sequence diagram of the emblazoning process.

Figure 8 visualizes the more or less sequential progress of the creation of a blazon in an UML sequence diagram. As shown on the left side the general composition of the blazon is set by class *Emblazoner*. Each single shield that has to be placed upon the blazon is represented by an instance of class *Shield* which is responsible for the creation of a single shield's image which finally is returned to *Emblazoner*. After all shields have been calculated and a possibly existing overlaid charge has been drawn *Emblazoner* arranges the single images, thus creating the final

image of the desired blazon.

As said above Shield creates the desired image by creating instances of classes that are responsible for different elements. At first, the base of the blazon is drawn by *Fields*. Secondly, ordinaries are placed on this base by *Ordinaries* which is followed by the creation of the subordinaries. Finally *ChargeHandler* calculates the positions and layout of possible charges and again places them upon the image.

JEmblazoner was developed in Java, image operations were implemented with the optional Graphic Layers Framework (GLF) package from SUN. The user interface can be used as a plug-in to our CBIR system for coats of arms.

6. Conclusion

JEmblazoner is a text-based renderer for coats of arms images. It was implemented as a plug-in for a coats of arms CBIR system but may be used as a stand-alone tool for other applications as well. In the future, the system will probably be re-implemented based on our Visual Information Retrieval Framework VizIR. The VizIR project aims at the following major goals:

- Implementation of a modern, open class framework for content-based retrieval of visual information as basis for further research on successful methods for automated information extraction, definition of similarity measures and new, better concepts for the user interface aspect of visual information retrieval.
- Implementation of a working prototype system that is fully based on the visual part of the MPEG-7 standard.
- Development of integrated, general-purpose user interfaces for visual information retrieval.
- Support of methods for distributed querying, storage and replication of visual information and features and methods for query acceleration.

More information on the VizIR project can be found in [3]. Interested research groups are invited to contact the authors and to participate in the design and implementation of the open VizIR project.

7. References

[1] Breiteneder, C. and Eidenberger, H. A Retrieval System for Coats of Arms, in Proceedings of the International Symposium on Multimedia Application and Distance Education (Baden-Baden Germany, 1999).

[2] Del Bimbo, A. Visual Information Retrieval. Morgan Kaufmann Publishers, San Francisco CA, 1999.

[3] Eidenberger, H., and Breiteneder, C. A Framework for Visual Information Retrieval, in Proceedings Visual Information Systems Conference (HSinChu Taiwan, March 2002), LNCS, Springer Verlag, 105-116.

[4] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. Query by Image and Video Content: The QBIC System. IEEE Computer, 28/9 (1995), 23-31.

[5] Grammar of Blazonry. http://www.sca.org/heraldry/laurel/bruce.html

[6] Introduction to Heraldry. http://www.sca.org.au/lochac/scribes/hrld_int.html

[7] Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., and Jain, R. Content-Based Image Retrieval at the End of the Early Years. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22/12 (December 2000), 1349-1380.

[8] Wasinger, M. Jemblazoner – eine Java-Applikation zur Generierung von Wappenbildern, Institut für Softwaretechnik und Interaktive Systeme, Technische Universität Wien, Diplomarbeit, Wien 2001.