This paper has been accepted by the 4th International Workshop on Multimedia Information Retrieval, in conjunction with ACM Multimedia 2002, Juan-les-Pins, France, Dec., 2002.

FLAME: A Generic Framework for Content-based Flash Retrieval

Jun Yang[1,3] Qing Li[1] Liu Wenyin[2] Yueting Zhuang[3]

[1]Dept. of Computer Engineering and Information Technology
[2]Dept. of Computer Science
City University of Hong Kong, Kowloon, HKSAR, China
yangjun@acm.org {itqli, csliuwy} @cityu.edu.hk

[3]Dept. of Computer Science
Zhejiang University
Hangzhou, China, 310027
yzhuang@cs.zju.edu.cn

ABSTRACT
Flash™ is undergoing an explosive growth and has become a prevailing media format on the Web. Unfortunately, no research effort has been dedicated to the retrieval of Flash movies based on content, which is essential to the utilization of the enormous Flash resource. In this paper, we conduct a close investigation of Flash movies and reveal that a typical movie is semantically characterized by means of its heterogeneous media components, the dynamic effects of the components, and the user interactions involved. As the first endeavor in the area of content-based Flash retrieval (CBFR), we propose a generic framework termed as FLAME (*FLash Access and Management Environment*) to address the representation, indexing, and retrieval of Flash movies at different levels of details, including (1) *object* level, which describes the heterogeneous media components in a movie, (2) *event* level, which depicts the movie's dynamic effects, and (3) *interaction* level, which models the relationships between user behaviors and the consequential events.

Keywords
Flash movie, content-based Flash retrieval, multimedia retrieval

1. INTRODUCTION
Flash™ is a new format of vector-based interactive movie proposed by Macromedia Inc. [4], which can be embedded into web pages and efficiently delivered over the Web. Since its advent in 1997, Flash has experienced a phenomenal growth and become one of the most prevalent media formats on the Web. According to statistics [5], by August, 2002 there are over 468 million Internet users that can view Flash movies in Macromedia Flash Player, the presentation tool of Flash. Flash movies are primarily used to enhance the interactive and multimedia feature of web pages, but they are also created as cartoons, commercial advertisements, e-postcards, MTVs, or games, each of which has huge market potentials. The unique features of Flash that contribute to its huge success mainly include its compactness (for fast delivery), ease of authoring, rich semantics (due to its vector-based format), and powerful interactivity, which predict its even greater popularity in the near future.

Due to the popularity of Flash and its promising future, it becomes an imperative task of the multimedia research community to develop effective and efficient retrieval tools for Flash movies. It is foreseeable that Flash retrieval tools will be useful to a variety of user groups, ranging from teenagers looking for Flash games, music fans seeking for MTVs, to Flash developers reviewing the designs of existing movies, and customers searching for Flash advertisements. Some online Flash repositories, such as Flash Kit [1], have provided users with simple search functions by matching user queries against the manual keyword annotation associated with each movie[1]. However, this approach does not investigate the rich content of movies, which contains semantic clues indispensable for evaluating user queries. In the research community, the dilemma is that although extensive work has dedicated to the retrieval of various types of media (text document, image, video, etc), to the best of our knowledge, there is no related work on the content-based indexing and retrieval of Flash movies, despite the fact that nowadays Flash is equally favored as, or even more popular than, other medias on the Web. Therefore, we are motivated to present FLAME, a generic framework for content-based Flash retrieval (CBFR), as the first piece of work in this unexplored area.

A close examination reveals that a typical Flash movie is characterized from three major aspects: (1) *heterogeneous media components* contained in it, such as texts, graphics, sounds, video clips, (2) *dynamic effects* constituted by the spatio-temporal features of these components, and (3) *user interactions* that interfere with the movie presentation. This intrinsic complexity of Flash, as discussed in Section 2, poses a number of nontrivial research issues that are not fully addressed by existing work. As the main contribution of this paper, a generic framework termed as FLAME (*FLash Access and Management Environment*) is proposed to facilitate users to access Flash movies based on their content. As described in Section 3, FLAME features a 3-tier architecture that addresses the representation, indexing, and retrieval of Flash movies at different levels of details, including (1) *object* level, which describes the heterogeneous media components in a movie, (2) *event* level, which depicts the movie's dynamic effects, and (3) *interaction* level, which models the relationships between user behaviors and the consequential events. In fact, the main objective of FLAME is not on providing a "total solution" of CBFR, but to define a comprehensive "skeleton" so that follow-up works in this area can fill into this skeleton as its components. The conclusion of the paper and the future works are presented in Section 4.

2. FLASH MOVIES: AN ANATOMY
By examining the structure of many typical Flash movies, we find that the semantics of a typical movie is mainly synthesized and conveyed through the following three types of devices:

- **Heterogeneous components**. A typical Flash movie usually consists of component media objects in a variety of types. Texts and graphics (i.e., drawings) of arbitrary complexity can be easily created as components using authoring tools of Flash (e.g.,

[1] If not explicitly designated, "movie" is referred to "Flash movie" in this paper.

Macromedia Flash v.4.0). Bitmap or JPEG images and QuickTime video clips can be imported into the movie as well. Compressed audios are embedded into movies in one of the two forms: event sound, which is played in response to a certain event (e.g., mouse-click), and streaming sound, which is played in synchronization with the advance of a movie. All these components are encoded separately, such that they can be easily extracted from Flash data files. This differs fundamentally from pixel-level media formats such as image and video.

- **Dynamic effect**. A Flash movie is composed of a sequence of frames that are played in an order subject to user interactions. With the progression of frames, components can be placed on the current frame, removed from it, or changed in terms of their positions, sizes, shapes, and angles of rotation. The spatio-temporal features of the components, as well as the spatio-temporal relationships among them, make up of some high-level dynamic effects (morphing, motion, rotation, etc), which suggest the semantic meaning of a movie.

- **User interactivity**. Rather than a passive media like streaming video, Flash is an interactive movie format in the sense that a user can interfere with the presentation of a Flash movie. As an example, by clicking a button in a movie the user can let the movie "jump" to a frame prior to or behind the current frame. Thus, an interactive movie has multiple presentations, with each of them resulted from a specific series of user behaviors.

From the above discussion, it is clear that the intrinsic complexity of Flash greatly surpasses that of any traditional media format, such as text document, still images, video clips, to which the extensive multimedia (including text) information retrieval techniques are devoted. Hence, CBFR cannot be simply addressed by any of these existing retrieval techniques. For example, the indexing of dynamic effects (or spatio-temporal features) still has many open issues to be investigated, while the modeling of user interaction is almost an untouched area. Therefore, a set of brand-new techniques needs to be devised under CBFR to index and retrieve Flash movies by their heterogeneous components, dynamic effects, and user interactions. From another perspective, however, since a Flash movie can be viewed as an organic collection of diverse traditional media objects, many existing retrieval methods can serve as the "enabling technologies" of CBFR. For example, text-based information retrieval (IR) techniques [7] can be applied to deal with text components in a movie, and various content-based retrieval (CBR) techniques [2,6,8] can be applied to image, sound, or video components. In conclusion, the framework of CBFR is likely to be a skeleton constituted by a variety of existing techniques together with many new techniques to be developed.

3. THE FLAME FRAMEWORK FOR CONTENT-BASED FLASH RETRIEVAL

FLAME is proposed as a generic framework for indexing and retrieval of Flash movies based on their content. As illustrated in Figure 1, it has a 3-tier architecture constituted by the representation layer, indexing layer, and retrieval layer from bottom to top, whose details are described in this section.

3.1 XML-based Movie Representation

Flash movies are delivered over the Internet in the form of Macromedia Flash data (SWF) files. Each Flash file is composed of a series of tagged data blocks, which belong to different types

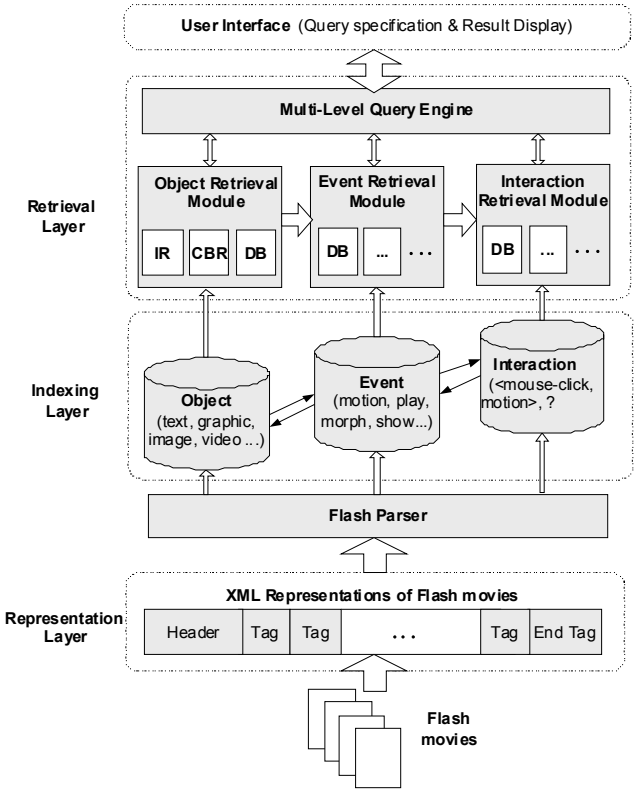

Figure 1: The 4-tier architecture of FLAME

with each type having its own structure. In essence, a Flash file can be regarded as an encoded XML [3] file (a Flash file is binary while a XML file is in ASCII format), and it can be converted into an XML file using tools such as JavaSWF [3]. Each tagged data block in a Flash file is mapped to an XML tag, which usually has attributes and embedded tags to represent the content of the data block. There are two categories of tags in a Flash file: *definition tags*, which are used to define various components in a movie, and *control tags*, which are used to manipulate these components to create the dynamic and interactive effect of the movie. For example, *DefineShape* and *DefineText* are definition tags, while *PlaceObject* (placing a component on a frame) and *ShowFrame* (showing the current frame) are control tags. In the representation layer of FLAME, we convert Flash files into XML formats mainly because they are readable and thus convenient for us to understand the structure of Flash.

3.2 Multi-level Movie Indexing

As discussed in Section 2, a typical Flash movie is semantically synthesized and conveyed through its heterogeneous media components, dynamic effects, and user interactions, which in the indexing layer of FLAME are modeled using the concepts of object, event, and interaction respectively. Specifically, *object* represents movie components such as texts, videos, images, graphics, and sounds; *event* describes the dynamic effect of an object or multiple objects with certain spatio-temporal features; *interaction* models the relationships between user behaviors and events resulted from the behaviors. Naturally, these three concepts are at different levels: an event involves object(s) as the "role(s)" playing the event, and an interaction includes event(s) as the consequence of user behaviors. The features describing the objects,

Table 1. Features for objects, events, and interactions

	Name	Feature
Object	Text	Keywords, font size
	Graphic	Shape, color, number of borders
	Image	Size, color, texture
	Sound	MFCCs (mel-frequency cepstral coefficients)
	Video	Features of a set of key-frames, motion vectors
Event	Motion	Trail, start/end frame
	Rotate	Angle of rotation, location, start/end frame
	Resize	Start/end size, location, start/end frame
	Morph	Start/end shape, number of frame
	Play (for sound and video objects)	Current frame
	Trace (following the mouse point)	Closeness to mouse
	Navigate (going to a specific URL)	Target URL
Inter-action	Button	Event (press, release, mouse-over, mouse-out), position
	Keyboard	Key code
	Mouse	Action (drag, move, click, up), position

events, and interactions in a Flash movie are extracted by the Flash Parser from the XML representation of the movie (see Figure 1). The formal description of each concept and its features are presented below:

- **Objects**. A component object in Flash is represented by a tuple, given as:

$$object = <oid, o\text{-}type, o\text{-}feature>$$

where *oid* is a unique identifier of the object, *o-type* $\in$ {*Text, Graphic, Image, Video, Sound*} denotes the type of the object, and *o-feature* represents its features. Obviously, the particular types of feature vary from one type of object to another. Table 1 summarizes the most commonly used features for each type of object, which are extracted from the corresponding definition tags in Flash files either directly or through some calculations.

- **Events**. An event is a high-level summarization of the spatio-temporal features of object(s), denoted as:

$$event = < eid, \{action\}_n >$$
$$action = <object, a\text{-}type, a\text{-}feature> \ (n=1, \ldots, N)$$

where *eid* is a unique identifier of the event, followed by a series of actions. Each action is a tuple consisting of an *object* involved as the "role" of the action, *a-type* as the type of the action, and *a-feature* as the attributes of the action. Each type of action can be applied to certain type(s) of objects (e.g., morph action is applicable only to graphic objects) and is described by a particular set of features extracted mainly from control tags, as listed in Table 1. This representation of dynamic effects is very powerful in terms of expressiveness, as it supports multiple actions in a single event. For example, a graphic object that is moving and resizing simultaneously over frames can be modeled by an event consisting of two actions describing the motion and resizing of the object respectively.

- **Interactions**. The concept of interaction describes the relationship between a user behavior and event(s) caused by this behavior. Its formal definition is given as:

$$interaction = <iid, i\text{-}type, \{event\}_n, i\text{-}feature> \ (n=1,\ldots,N)$$

where *iid*, *i-type*, and *i-feature* represent the identifier, type, and features of the interaction respectively, and $\{event\}_n$ is a set of events triggered by the interaction. The type of interaction indicates the device through which user behavior is conducted, such as button, mouse, and keyboard. Button is a special component in Flash movies for the purpose of interaction, and it responses to mouse and keyword operation as a normal button control does. Interactions involving buttons are classified as "button" interaction, although they may also involve keyboard and mouse operations. The features for each type of interaction are given in Table 1.

So far, the index of a Flash movie can be represented as a collection of objects, events, and interactions in it, given as:

$$movie = <\{object\}_m, \{event\}_n, \{interaction\}_t>$$

The retrieval of Flash movies is conducted based on such multi-level features, as described in the next subsection.

3.3 Multi-level Query Processing

As shown in Figure 1, the retrieval layer of FLAME consists of three *individual retrieval modules* that are responsible for matching movie features at the object, event, and interaction level respectively. Since user queries usually involve movie features at multiple levels, a *multi-level query engine* is designed to decompose user queries into a series of sub-queries for objects, events, and interactions that can be processed by underlying retrieval modules, and then integrate and translate the results returned from these modules into a list of relevant movies. The functionality of each retrieval module and the multi-level query engine are summarized below:

- **Object retrieval module**. This module accepts the type and features of object as input, and returns a list of objects of the specified type that are ranked by their similarity to the given features. The retrieval process is summarized by the following function:

object-list: **SearchObject** (*o-type, o-feature*)

where *object-list* is a list of <*oid, score*> pairs, with *score* indicating the similarity of each object to the feature specified by parameter *o-feature*. If *o-feature* is not specified, all objects of the given type are returned. The "search space" of this function covers all the objects of every movie in the database and thus the returned objects may belong to different movies. The type of features specified as search condition differs from one type of object to another, and even for the same type of object, it may differ from one query to another. Moreover, different retrieval techniques are needed to cope with different object features. For example, IR approach is used for the keyword feature of text components, and

CBR approach is used for the low-level features of video, image, and sound components.

- **Event retrieval module**. To search events, we need to specify search conditions for both actions and the objects serving as the "roles" of the actions, as illustrated by the function below:

event-list: **SearchEvent** (*a-type, a-feature, object-list*)

This function returns a list of events having at least one action that satisfies all the following three conditions: (1) the type of the action is equal to *a-type*, (2) the feature of the action is similar to *a-feature*, and (3) the object involved in the action is within *object-list*. If *a-feature*, *object-list*, or both of them are not given, the returned events are those with at least one action satisfying conditions, respectively, (1) and (3), (1) and (2), or only condition (1). Since only one action can be specified in **SearchEvent**, the query for multi-action events is handled by firstly performing **SearchEvent** based on each desired action and then finding the events containing all the desired actions by intersecting multiple *event-list* returned from **SearchEvent**.

- **Interaction retrieval module**. The retrieval of interactions is conducted by the following function:

interaction-list: **SearchInteraction** (*i-type, i-feature, event-list*)

The semantics of this function is similar to that of **SearchEvent**. The *event-list* specifies the scope of events, among which at least one must be triggered in every interaction returned by this function. Similarly, to search for an interaction that causes multiple events, we need to perform this function for each desired event and integrating the results to find the interactions causing all the desired events.

- **Multi-level query engine**. The results returned by individual retrieval modules are objects, events, and interactions, whereas the target of user queries is Flash movies. Thus, a primary task of the multi-level query engine is to translate the retrieved objects (or events, interactions) into a list of relevant movies, as defined by the following function:

movie-list: **Rank** (*object-list / event-list / interaction-list*)

The movies in *movie-list* are those containing the objects in *object-list*, and their similarity scores (and therefore ranks) are identical to their corresponding objects in *object-list*. The semantics of **Rank** taking *event-list* or *interaction-list* as parameters is similar. Furthermore, since a user query may specify multiple search conditions, a **Merge** function is devised to combine multiple lists of movies retrieved based on each search condition into a single list giving the final ranking of relevant movies, defined as:

movie-list: **Merge** ($\{movie\text{-}list\}_n$, $\{weight\}_n$)

where $\{movie\text{-}list\}_n$ denotes n movie lists that are obtained based on different search conditions, and $\{weight\}_n$ contains the weight indicating the relative importance of each condition, which is preferably specified by users. Each movie in the returned movie list must appear in at least one input list, and the similarity score of the movie is determined by the weighted sum of its similarity score in each input list.

The usage of these functions can be demonstrated by processing sample user queries. Consider, for example, a query for Flash movies as commercial advertisement of BMW cars. Since a Flash advertisement usually contains the hyperlink to the company website, a likely interpretation of this query is: *Find all movies that have keyword 'BMW' and a button by clicking which the BMW website will be opened in a Web browser*. This can be processed by a combination of all the aforementioned functions:

Merge ({**Rank** (**SearchObject** (text, 'BMW'), **Rank**(**SearchInteraction** (button, 'mouse-click', **SearchEvent**(navigate,'www.bmw.com')))))}

4. CONCLUDING REMARKS

This paper has investigated the problem of content-based Flash retrieval (CBFR), which is essential to better utilization of the proliferating Flash resource but unfortunately overlooked by the research community. As the major contribution of this paper, we have proposed a generic framework for CBFR, FLAME, which addresses the representation, indexing, and retrieval of Flash movies through their heterogeneous media components, dynamic effects, and the forms of user interactions.

Although FLAME may appear to be comprehensive, there remains much room for future research regarding Flash retrieval and management. One interesting direction is to investigate human-computer interaction for better retrieval effectiveness, e.g., applying relevance feedback technique to enhance the retrieval performance. Moreover, the storage, navigation, classification of Flash movies are equally important and promising research directions. On the other hand, the framework of FLAME can be generalized to support the retrieval of other types of multimedia representations, such as PowerPoint, etc.

5. ACKNOWLEDGMENTS

This work has been supported, in part, by Hong Kong UGC Research Grants Council under grant CityU 1038/02E, and partially by a grant from the Doctorate Research Foundation of the State Education Commission of China.

6. REFERENCES

1. Flash Kit. http://www.flashkit.com/index.shtml
2. Foote, J. An overview of audio information retrieval. *ACM Multimedia Systems*, 7: 2-10, 1999.
3. JavaSWF. http://www.anotherbigidea.com/javaswf/
4. Macromedia, Inc. www.macromedia.com.
5. Macromedia Flash Player adoption statistics. www.macromedia.com/software/player_census
6. Rui, Y., et al., Image retrieval: current techniques, promising directions and open issues. *J. of Visual Communi. and Image Represent.*, 10: 1-23, 1999.
7. Salton, G., et al., *Introduction to modern information retrieval*. McGraw-Hill Book Company, 1983.
8. Smoliar, S., Zhang, H., Content-based Video Indexing and Retrieval. *IEEE Multimedia*, 1: 62-72, 1994.