Undergraduate Research Opportunity Program
(UROP) Project Report

SLIME: A Tool For Composing Live And Stored Media

By

Liu Bing

Department of Computer Science

School of Computing

National University of Singapore

2004/05

Undergraduate Research Opportunity Program
(UROP) Project Report

SLIME: A Tool For Composing Live And Stored Media

By

Liu Bing

Department of Computer Science

School of Computing

National University of Singapore

2004/05

Project No: U020190
Advisor: Dr. Ooi Wei Tsang
Deliverables:
Report: 1 Volume
Source Code: 1 CD

Abstract

One of the three grand challenges in multimedia research highlighted recently is the ability to make authoring complex multimedia content as easy as word processing. Most of the video editors which target average user do not support spatial video editing well and only can only edit stored video. To take up these challenges, we designed an easy-to-use video editing tool named SLIME that allow user to compose heterogeneous live and stored video together. It has powerful features on spatial editing and real-time editing (e.g. webcast production). It makes use Plasma language as its processing engine, and provides a user-friendly GUI by combining many existing interface designs for video editing tools. Those designs and implementation are presented in this report.

Subject Descriptors:

D2.2 Design Tools and Techniques
H2.1 User/Machine System
H5.1 Multimedia Information Systems
H5.2 User Interfaces

Keywords:

Video editing, user interface, spatial editing, real-time editing

Implementation Software and Hardware:

Linux Redhat 9, Tcl/Tk 8.4.7, TkZinc 3.3.0, Snack, MPlayer 1.0.6pre, FFMpeg 0.4.8, Dalí 1.0, Plasma 0.2, Visual Tcl 1.7, PC P4 1.6 G

Acknowledgement

First of all, I would like to thank my advisor Dr. Ooi Wei Tsang for his guidance, patience and constant support during my Undergraduate Research Opportunity Program. Working with him is unforgettable experience for me. He introduced me the field of multimedia, and gave me a direction for my research. Not only had he taught me many research skills such as literal review and paper writing, but also I leant very much from his excellent personal characteristics. Without his kind help, I would not be able to complete this project.

I am also grateful to my colleague Zhu Tao in Networked Media Research Group. His valuable support on Plasma language eased my life and made my project better.

I also would like to thank my friends Liu Ning and Wu Hua Yu. They contribute suggestions and participate in my user study.

Finally, I am forever indebted to my parents and my lover Han Zheng for their love and encouragement.

List of Figures

Table of Contents

Chapter 1

Introduction

1.1 Motivation

One of the three grand challenges in multimedia research highlighted recently is the ability to make authoring complex multimedia content as easy as word processing (Rowe & Jain, 2005). Due to the development of digital video equipments such as video camera and web camera, digital video is becoming increasingly ubiquitous. As a result, easy-to-use video editing tools are needed by more and more average users. For example, a teacher may need video editing tools to prepare educational material that includes video demonstration. A travel agent may need tools to prepare video material showing places and experiences. A family member may need tools to prepare video material that document a significant life event such as a birthday, wedding, or birth of child. However, although some excellent commercial tools which allow creation of high-quality video exist, such as iMovie (Apple, 2004b), Movie Maker (Microsoft, 2003), and Premiere Pro (Adobe, 2004), video editing is still a difficult, tedious and error-prone activity. This is because video editing has several unique challenges not found with other media which make those tools hard to use, especially for novices.

Video composition plays an important role in video editing and is widely used in multimedia productions such as television programs, academic video, video conferencing, and surveillance (See Figure 1.1). However, after reviewing many current commercial video editing software,

Figure 1.1: Applications of video compostion. (A) is from a television program; (B) is from an academic webcast; (C) is from an video confrences between universities; and (D) is from an surveillance video.

we find that most of the tools whose target customer are average users are time-line based. Their functions are focus on those most simple and common temporal editing tasks such as the cutting and pasting of video segments interspersed with transitions, special effects and titles. The spatial editing tasks such as video composing and overlay can only be perfectly done by some complex software which target experts, such as Final Cut Pro (Apple, 2004a), Premiere Pro. Since there is trade-off between expensiveness and ease of use, those expert-user tools require too much learning and high system performance. Therefore researches on finding simple and efficient ways for spatial editing have been conducted in the recent years.

Another limitation of traditional video editing tools is that most of them only can process stored media. With the exploding of Internet usage and the prevalence of personal computers, network multimedia technology became more and more important. With the development of IP Multicast, RTP, Internet Mbone Tools (McCanne, 1999), and commercial streaming media systems (e.g. Microsoft Windows Media, Real Network, Apple QuickTime Streaming), the

Figure 1.2: Picture-in-picture effect in webcast production

network media applications used for distant learning, remote collaboration, video on demand system, and interactive game became more and more popular. As a result, there is a need to real-time adapt and manipulate live streaming media. For example, when producing a live webcast for a lecture, several video and audio stream form different cameras and microphones are sent to a central studio. Then the director needs to real-time select and edit those media and stream to the distant audience. When the speaker start the presentation, the director may to place text on an image to describe who is talking; and if an audience asks a question, the director may need to switch to the camera which towards that audience. Alternatively, the director can show the speaker and audience at the same time with the picture-in-picture effects (See Figure 1.2). In television industry, to perform the above task, it may require expensive equipments and many people to operate the system. Therefore researches on internet broadcast system had been conducted in the recent years and many softwares used to produce webcast has been developed for example BIBS (Rowe, Harley, & Pletcher, 2001).

The above challenges motivated us to design and implement **S**tored and **LI**ve **M**edia **E**ditor (SLIME), a prototype of an easy-to-use video editing tool that can real-time compose both live and stored media.

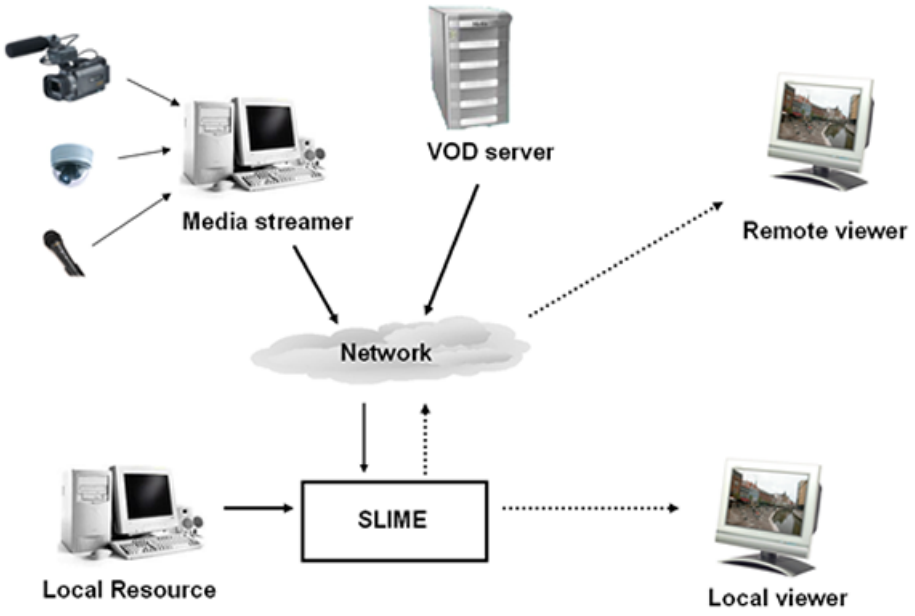

Figure 1.3: A overview of SLIME video editing environment

1.2 Contributions

One of the contributions of SLIME is that it provides an intuitive graphic user interface that exploits the powerful features in Plasma scripting language (Zhu, 2003), which is a high-level scripting language currently actively under development in SoC[1]. The novel user interface of SLIME allow user to compose different video together by freely moving, resizing, cropping, fading and titling video clips on a scratch panel. As a result, even without training, an average user can easily produce high quality videos with various special effects, which usually only appear in the video produced by professional video editing software. Furthermore, due to the streaming feature of Plasma, SLIME can receive, process, and broadcast heterogeneous streaming media as the popular RTP/RTSP stream. The efficient way for real-time editing is a unique feature of SLIME. It allows to produce webcast with common PC and only requires one director to operate. With the network feature, user also can use SLIME for video conferring, video surveillance and even as a multi-channel network TV.

[1]School of Computing, National University of Singapore

Another contribution is that the user interface of SLIME combines some existing human computer interaction designs for example playback highlight technique, frame representation, drag and drop, layer, box cropping, and action history. These designs bring user more convenience and can reduce the difficulty during video editing and webcast production. We also propose a novel design of video timeline which is used to present the temporal relation of video clips. Our timeline is composed by thousands of colored slice, and each slice represents a frame. This technique allow use to easily retrieve the information such as shot transitions so that user can know what is going on.

1.3 Organization

The rest of the report is organized as follows. Chapter 2 presents some background information, and describes recent work related to our contributions. Chapter 3 describes the SLIME application, and the design principles and implementation issues is presented in Chapter 4. Next we address the results and limitation in Chapter 5. Finally, we discuss future work and conclude in Chapter 7.

Chapter 2

Background and Related Work

In this chapter, we present the technological backgrounds needed to understand our work. This is followed by an overview of related research work.

2.1 Software Tools

2.1.1 Tcl/Tk and TkZinc

Tcl (Tool Command Language) (Ousterhout, 1994) is an open source scripting language used by over half a million developers worldwide. It has a simple and programmable syntax and can be either used as a standalone application or embedded in other applications. As an extension of Tcl, Tk is a graphical user interface toolkit. It can quickly create powerful GUIs with a few lines of code. Tcl and Tk are highly portable. It supports many popular platforms (e.g. Linux, Solaris, BSD, Window, Macintosh and more). TkZinc (Zinc Is Not Canvas!) (CenaSoftware, 2004) is a powerful widget for Tcl/Tk, Perl/Tk and Python/Tkinter. It is very similar to Tk canvas in that they support structured graphics. Like the canvas, TkZinc implements items used to display graphical entities. Those items can be manipulated and bindings can be associated with them to implement interaction behaviors. But unlike the canvas, TkZinc can structure the items in a hierarchy, has support for scaling and rotation, clipping can be set for sub-trees of the item hierarchy, supports muti-contour curves. Moreover, TkZinc has optional OpenGL features including color gradient, anti-aliasing and transparency.

Due to these desirable features, Tcl/Tk 8.4.7 (TclCoreTeam, 2004) and TkZinc 3.3.0 was chosen to be the base language for SLIME.

2.1.2 Dalí software library

Dalí (Ooi & Smith, 1999) developed by Cornell is a high performance software library of routines for manipulating video, audio and image data.

Being different from conventional multimedia libraries such as ooMPEG (Holden, 1996), Rivl (Swartz & Smith, 1995) and ISIS (Agmanolis, 1997), it provides low-level abstractions and operations for manipulating media data. To avoid the drawback of high-level libraries, Dalí allows programmers to control the resource such as I/O and memory so that optimized and high performance code can be produced. On the other hand, unlike conventional low-level libraries, programming with Dalí do not need to totally from the bottom since the primitives and operations Dalí provided make complex operation to be decomposed into simple ones. As a result, using Dalí, programmers can easily write efficient code for manipulating media data. For example, we are able to write an MPEG decoder in about 100 lines of code, and it performs about 20 percents faster than the standard Berkeley MPEG decoder (Patel, Smith, & Rowe, 1993).

Since the source code of Dalí includes support for both Tcl and C, SLIME can easily use Dalí API to write efficient functions for resizing and cropping images.

2.1.3 Plasma Language

Plasma (scri**P**ting **L**anguage for real-time manipulation and **A**daption of textbfStreaming **M**edi**A**), currently actively under development in SoC , is a concise and robust scripting language that supply operations for processing heterogeneous streaming media. It is implemented as a Tcl/Tk extension. Currently it uses FFmpeg multimedia system to decode and encode various formats of media (ie. It uses libavcodec so it supports most popular codecs) and use Dalí software library to adapt and manipulate each frame image. Live Media (LiveNetworks, 2004) library is

used to handle the network transmission and SDL library (SDLProgect, 2004) is used to display video.

Plasma is designed to perform real-time mixing and manipulation. It support RTP/RTSP streaming media and can simultaneously receive, process without interruption of decoder and encoder, and real-time stream out the output data.

The current version of Plasma support manipulations for spatial composition of videos, temporal composition of audios and video streaming. Since all the operators are implemented as Tcl commands, its simple object-oriented syntax is very easy to understand. It has only 13 operators which treat media as first class data type. A constructor operator called *media* can be used to create video, audio, image, and text objects. Each media object has several properties such as size, position, transparency, sound volume, and visibility. A dozen of options can be configured as user like. The spatial relation of different medias are abstracted using TAO model (Zhu, 2003) which is quite similar as Tk widgets. Operator *mpack* and *munpack* is used to pack and unpack a media object to its parent. Parent object can be considered as a container, and child object can be a component inside this container or as a new layer on its parent. Plasma also has event support. Event operator *mbind* can bind some build-in events (e.g. time, packet loss, keyboard, mouse, motion detect, etc.) to a media stream. For temporal editing, *mconcat* operator can temporally concatenate a child object to its parent. Finally, *mplay* and *mout* operator is used to encode and output the resultant media steam to disc, device or network. The difference of *mplay* and *mout* is that *mout* will hold the control until the encoder finishes its job, but *mplay* will release the control after the command is executed. The following two examples illustrate how to use Plasma perform spatial editing tasks.

Side-by-side composition

Program 1 is a script to compose three videos together (side by side relation). In Plasma language, the usage of *mpack* is as the same as it in Tcl/Tk's *pack*. In other words, "Plasma looks for the minimal empty space available every time to accommodate a video." (Zhu, 2003)

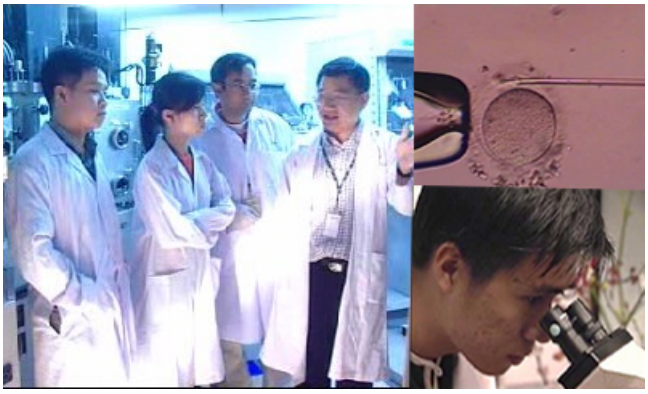

Figure 2.1: Result video snapshot for Example 1

The spatial relations specified by the below script are shown in Figure 2.1. Note that *mpack* operator has some limitations. For example, the order of pack is significant to determine the result layout, and some pattern can not be produced by *mpack*. The output video has been encoded to Windows Media Video format, and been displayed in a SDL window at the same time for monitoring.

Program 1 Side-by-side composition example

```
media .v
media .v.left ../video/students.mpg
media .v.right.top -mpeg1 ../video/cell.mpg
media .v.right.bottom -mpeg1 ../video/operator.mpg
mpack .v.left -side left
mpack .v.right.top -side top
mpack .v.right.bottom
mout .v "dev result.wmv"
```

Picture-in-picture composition

Picture-in-picture effect often appears in TV advertisements and movie. Program 2 shows a Plasma script to produce such a video (See Figure 2.2) with only seven command lines. The pos option make the foreground video clips to position (10, 180), (110, 180) and (210, 180). The

Figure 2.2: Result video snapshot for Example 2

output video is sent as a RTSP stream whose URL address is rtsp://(localhost):5454/out.mpg

Program 2 Picture-in-picture composition example

```
media .bg ../image/blue.jpg
media .bg.fg1 ../video/violin.mpg
media .bg.fg2 ../video/audience.mpg
media .bg.fg3 ../video/doctor.mpg
mpack .bg.fg1
mpack .bg.fg2
mpack .bg.fg3
.bg.fg1 scale 96 72
.bg.fg2 scale 96 72
.bg.fg3 scale 96 72
.bg.fg1 pos 10 180
.bg.fg2 pos 110 150
.bg.fg3 pos 210 150
mplay .bg rtsp://out.mpg
```

2.2 Related Work

In this section, we overview the research study related to stored and live media editing.

Figure 2.3: Apple iMovie

2.2.1 Stored Video Editing

Most modern digital video editing systems are adaptations of classical film editing systems. Their primary function is to cut raw footage into a series of clips, and then assemble these clips along some timeline into a produced video. In general, the interface of commercial video editor consist three components which are browsing panel, preview panel and timeline.

Figure 2.3 shows the interface of Apple iMovie. The video clips represented by their first frames were displayed in the browsing panel. And the timeline where the user edits clip sequence provide several tracks for video and audio. The user can drag the clip into timeline, specify its position, and apply special effects. The preview panel can play back current results or source clips. Since this simple structure is very easy to understand, it widely adopted by many software. SLIME also followed this design, but adds a scratch panel to allow user perform spatial editing task. This will be described in detail in next chapter.

Browsing panel

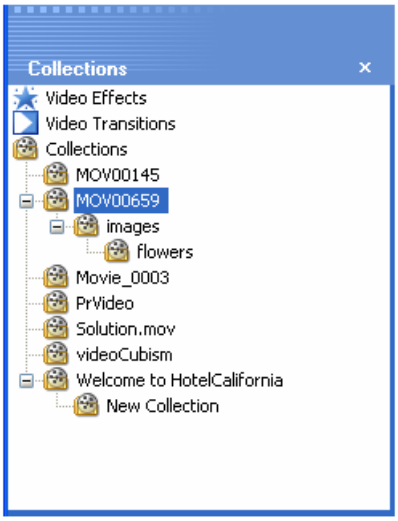

Figure 2.4: Movie Maker Collections

Browsing panel is used to browse video clips. Some systems present the clips in a hierarchical structure. For example, Yang et al (Yang, Mu, & Marchionini, 2003) propose a video indexing and visualization organizer VIVO that present every clip in segment/frame levels. And some system like Windows Movie Maker (Microsoft, 2003) allows user to group the clips into collections (See Figure 2.4). Hitchcock proposed by Boreczky et al (Girgensohn, Boreczky, Chiu, Doherty, Foote, Golovchinsky, Uchihashi, & Wilcox, 2000) even can cluster all clips by the similarity of their color and put them together into a pipe. SLIME intends to adopt the approach similar to Movie Maker's approach, since this design is a simple and efficient way to help user organize mass clips.

Preview window

Preview panel is used to preview source video or resultant video. Most of the softwares embed the preview window in to the application window, and support play/pause, stop functions. Some software like Premiere also support non-linear forward/backward playback. For example, in Premiere, a non-linear scroll bar is used to simulate film editing system.

Figure 2.5: Hitchcock timeline

MyVideo (Wang, Zhao, Zhang, Li, & Zhang, 2002), a home video management system developed by Microsoft Research Asia, use a technique called playback highlight. It also called video summary means play a small portion (automatically extracted by system) of representative content form a long video sequences or a group of clips. During the process of video playback, the user can dynamically adjust the percentage of video skimming to be played.

The above features have been selected and adopted by SLIME as well as other preview features such as unfix size preview window and frame extracting.

Timeline

The temporal relation between clips can be represented by the timeline. On timeline, user can cut, paste, resize video clips, and add transition effect or titles. Timeline can shows the precise time or just be a storyboard. Silver is a video editor proposed by Casares et al(Casares, Long, & et al., 2002) that support a hierarchical timeline. Its timeline has three levels which are clip, shot and frame level; a sophisticated scrollbar can zoom in/out all levels at the same time. This hierarchical structure allow user to work at a high level of detail without losing the context. Figure 2.5 shows a novel storyboard design provided in Hitchcock (Girgensohn et al., 2000). The image size corresponds to the length of clip (log relation). A handle in cornel allow user to resize image so that the length of clip will be modified. Since specify precise time is tedious work, SLIME propose a novel timeline design that adapt and combine above two approach with a new concept of presentation. The detail will be discussed in Chapter 3.

2.2.2 Live video editing

Live internet broadcasts is one of the research of Berkeley Multimedia Research Center. They have proposed Director's Console (Yu, Wu, Meyer-Patel, & Rowe, 2001) which is a live webcast production application based on a general webcast production model composed of three stages (i.e. sources, broadcast, and transmission). The main window contains four panels: sources, preview, broadcast, and transmission. Streams in the webcast are displayed in the broadcast panel. The transmission panel will control webcast transmissions. They output two video streams and one audio stream. Each stream can be switch to other one in the source during production.

Another webcast production system proposed by Machnichi et al (Machnicki & Rowe, 2002) is called Virtual Director. It automates the tasks such as recording equipment control, stream broadcasting, and content decisions. It allow user to specify the automation using an automation specification language, and uses a question monitor service to identify questions and switch the cameras.

Ooi et al (Ooi, Pletcher, & Rowe, 2001) has developed a middleware for distributed media environment named Indiva. It use a file system metaphor to access resources and provide a unified set of abstractions and high-level operations for hardware devices (e.g. microphone, camera, speaker, etc.), software processes, and media data in a distributed audio and video environment.

Webcast is one of the features of SLIME on live video editing. It adapt dc's webcast production model and provide more spatial effects. Invida can be integrated into SLIME to allow the user to operate devices. This will be described in the future work section.

Chapter 3

SLIME Application

This chapter presents the details of our application. We begin with an overview of features, and then we describe the user interface of SLIME. As shown in Figure 3.1, We divide the user interface into browsing panel, editing panel, crop panel, layer panel, property panel, history panel (other components such as timeline, preview window, title panel, music panel, and make video window are not shown in the main window). After discussing above component one by one, we address the effects issues.

3.1 Overview

To solve the problems of video editing mentioned in chapter 1, we developed SLIME, a video editor implemented with Tcl/Tk and TkZinc. Currently, SLIME use Plasma beta version 0.2 as processing engine, use Dalí software library to retrieve video information and manipulate images, use MPlayer (MPlayerTeam, 2004) to achieve highlighted playback, and use FFmpeg multimedia system to convert resultant video files from to various popular formats. It can process not only the stored media but also the live RTP/RTSP streaming media on the air. The real-time or static-time manipulations it provides include resizing, cropping, fading, composing (side-by-side or overlay), swapping, etc. These powerful features allow user to easily produce stored videos or live webcast with professional effects.

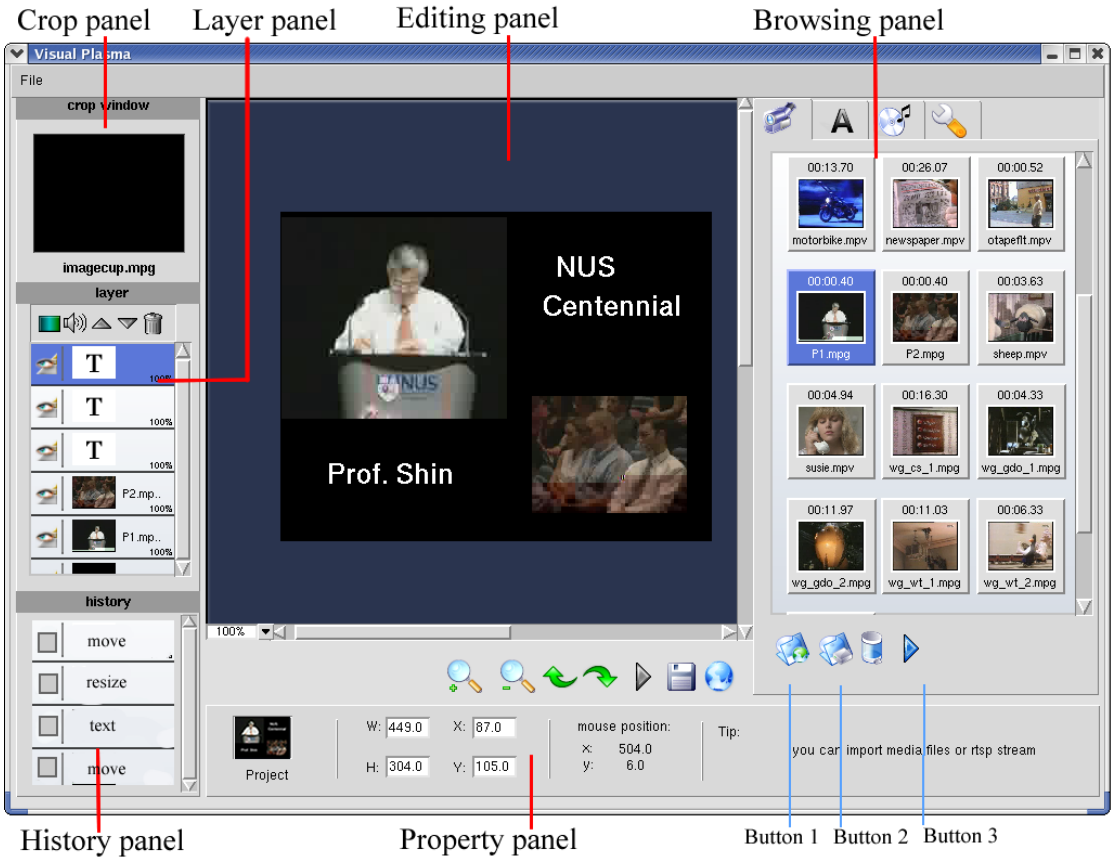

Figure 3.1: SLIME interface

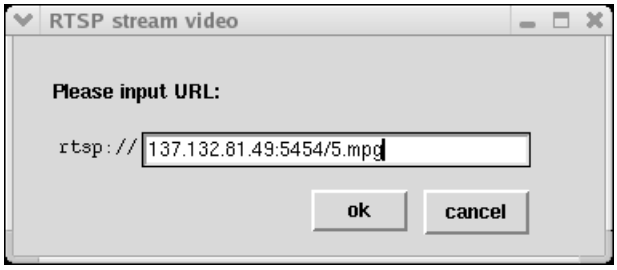

Figure 3.2: Live stream selection dialog

3.2 User Interface

Browsing panel

The Browsing panel serves as a starting point for video editing. It allows user to import, preview or delete stored and live media streams by clicking corresponding button. After user press *Button 2*, a file selection dialog will appear. User can import video clips and images with many popular formats including MPEG, WMV, RM, MOV, JPEG, GIF, PNG, etc. Importing video files one by one is a tedious work. So SLIME supports the feature that allow import multiple files once. Alternatively, user can start to receive a RTP/RTSP media stream by pressing *Button 1* and type in the URL address in the live stream selection dialog (See Figure 3.2).

After importing, all the media will be represented by a thumbnail with other information such as duration and file name. For stored video, the thumbnails are generated by extracting the first frame of a clip. For live stream, the thumbnails are represented with a temp image. After the connection was setup, the thumbnails can dynamically update every second. It describes the current frames so that user can know what is going on. We achieve this using a feature of Plasma and the details will be described in next chapter. To preview stored video clips, user can click *Button 3* to open the Preview Window which will be discussed in the later part of this section. To help user organize mass clips, our design of Browsing panel simulates the file folder styles (e.g. Windows XP, Nautilus) and allow user to group file together into collections.

However, we haven't implement this feature due to the time constraint.

Title panel and Music panel

The Title panel and music panel is source panels as the Browsing panel. As shown in Figure 3.3, text effects (e.g. still, fade in, pup in, and fly in) are represented wish animated GIF images (we use TkImg extension to support animated GIF) so that user can preview those effects. To add a title, user can drag and drop a text thumbnail as same as they can do in the Browsing panel. The default duration of a title is 5 second. User can specify the exact time duration or let the title always appear using the entries below. To avoid exceptions, those entries has programmed with validity checking functions which means they only accept valid input. After drop a title onto the Editing panel, user can type in the text with in a text box. That text box can will confine the width of the text lines which means when the length of a line reach the bound, it will automatically been wrapped. User can move the text box to any position and use the 8 handler to resize the text box. The usage of those handlers will be described in the Editing panel part. Moreover, user can choose the font and size of text. Currently, we only support Arial font.

The interface of Music panel is quite similar with Browsing panel and the Editing panel. It allows user to select and pre-listen audios as background music. Audios are also represented as thumbnails and user can drag them onto the background music pool. Currently, we only support audio playback. Since we use Snack library to dealing with audio, we will continue exploit the powerful feature of Snack and implement narration record and audio visualization features for SLIME.

Editing panel

The Editing panel provides a place for editing. User can easily drag source media thumbnail from the Browsing panel and drop onto it. SLIME has WYSIWYG (what you see is what you

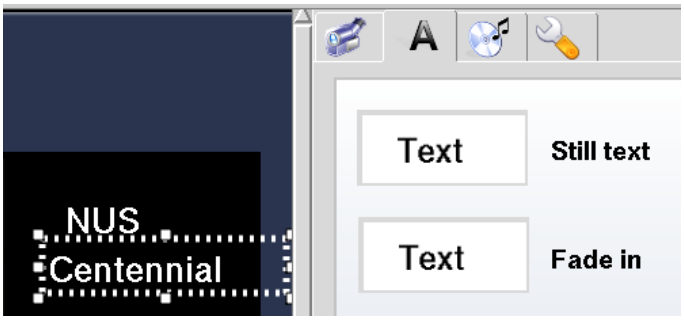

Figure 3.3: Title panel

get) feature, since in editing panel user can freely move and resize a media and the resultant video will be as the same as the user specified.

SLIME adopts the standard CHI design widely used in commercial software (e.g. Microsoft PowerPoint, Adobe Photoshop, Macromedia Fireworks, etc.) To resize a media clip (represented as an image) or an editable text, user can drag and drop the handler of eight directions. The east(resp. west) and north(resp. south) handler only change the width(resp. height) of a media object, and the corner handlers can change the object to any shape. Note that we do not support flipping, so invalid resize action will be ignored. During resizing, user can keep the width-height ratio by holding *SHIFT* or *CTRL* key. Holding *SHIFT* key and dragging an object also can help user keeping move a clip on a strait line.

The *Zoom in/out button* and the pull-down menu can scale of the users' view. This facility allow user to edit at a high level of detail without losing the context. In other word, when user is concentrating on editing small part, he can zoom in the view. On the other hand, user can zoom out the view and get a glance of the whole project. We achieve this by translating the display coordinates. At the meanwhile, we keep a zoom factor that is used to make encoding consistent.

Using *Undo/Redo button*, user can relax and edit as they like without worrying making mistakes. This function has the same underground implementation as the History panel. So we will present the details in that paragraph. The *Preview button* of the Editing panel is different

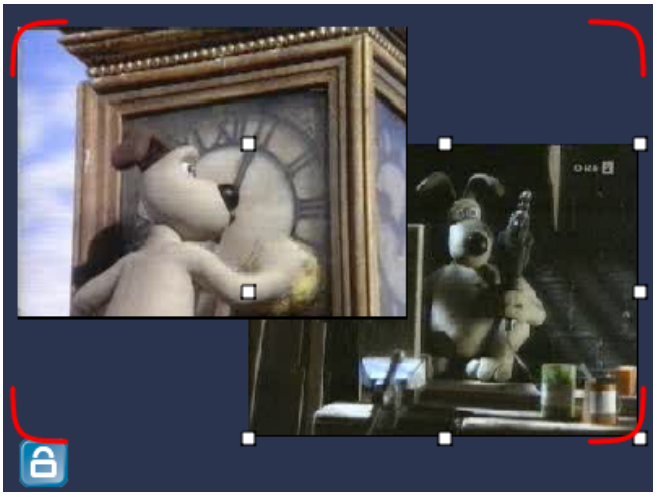

Figure 3.4: The size of video has been locked

from the one in Browsing panel. It will open a Plasma window to show the resultant video. The *Saving button* will call functions to open the Make Video Window that will ask user to specify the codec and output filename. Finally, the Streaming button will be used to broadcasting the resultant video out as a RTSP stream (the default port is 5454). Then the remote user can receive and play this stream by any software with RTSP support such as FFplay, MPlayer, VLC (VideoLan, 2004) and so on. Once the streaming starts, the size of output video will be unchangeable. As shown in Figure 3.4, a red frame with a lock will inform user which region for editing is available. During streaming, the Streaming button will be replaced by the *Update button*. This button is designed for the real-time editing support which is unique feature of SLIME. Whenever user make changes on the editing panel and press the Update button, those changes will reflect on the output steam at once. This feature is very useful for producing a webcast. For example, a director can real-time add videos, titles or logos onto the screen, and he also can switch two streams by dropping one's thumbnail in the Browsing window onto other's (See Figure 3.5).

Crop panel and Layer panel

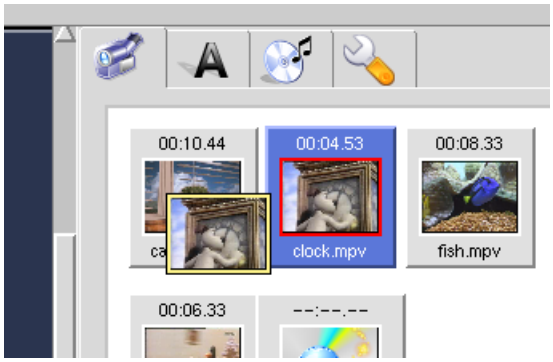

Figure 3.5: Swith two streams by drag and drop

The Crop and Layer panel (See Fugure 3.6) are the helper panels for the Editing panel. The Crop panel can help user to crop a video clip or image selected in the Editing panel. Instead of specify the exact coordinates (traditional way); user can simply draw a crop box by mouse in the Crop panel (See Figure 3.6 left part). Then corresponding part outside that box will be trim out which will also reflex in the Editing panel. We achieve this feature using a Dali function and the details will be presented in next chapter.

The Layer panel is used to organize the spatial relations of clips on z-axis direction. There are five layer control buttons. User can use emphUp/Down buttons to change the order of layer. Note that the highest layer means top media clips on the Editing panel. The *Alpha button* and *Sound button* allow user to adjust the transparency and sound volume. After adjusting, the transparency degree will display at the corner of each layer. The *Trash button* is used to delete a layer. The Eye icon inside each layer is used to hide or show this layer.

Property panel and History panel

The Property panel and History allow user to monitor the editing action. Property panel can show the detail information of each selected clip, and it also allow use to specify the size, position, color and transition effects (e.g. fade in, slide in, curtain in, fly in, etc.). User can change these parameters to make more precise editing. For example, if user wants the output

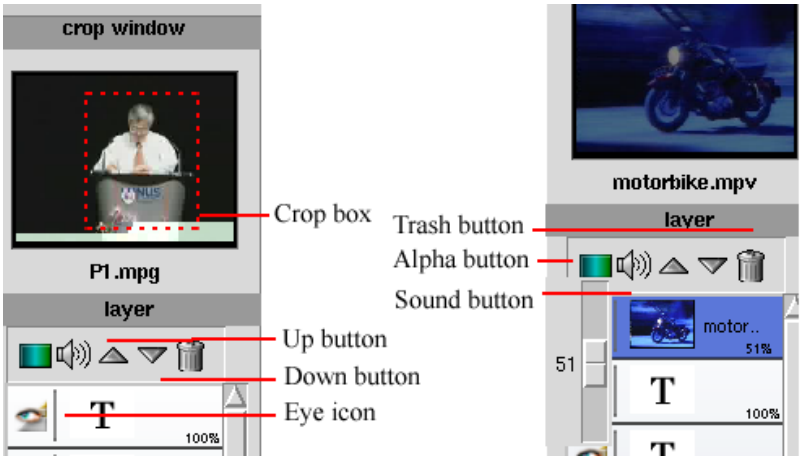

Figure 3.6: Crop panel and Layer panel

video size to be 352x288, resizing by dragging the handler require precise work and may not be a good idea. Instead, user can simply input 352 and 288 to the corresponding entry and the size of output video will be changed. Furthermore, user can specify a background color of the whole project. This is faster than put a colored image as the background picture. Transition effects are similar to the text effect. Our system allow user to select one appearance effect for each media object. During processing, we will use template Plasma code (using event binding which will be described in next chapter) to achieve those effects.

The History panel records all actions user performed so that user can recover to any pervious point as they like. The *Undo button* in the Editing panel can only restore one action once. However, the history panel allow user to recover many actions once. In details, each action inside history panel has a box in front. Clicking this box will recover the system to that state; in other words, the system will undo not only that action but also all the actions below it. To achieve this feature, we record the current state and old state of the target object for every valid action (i.e. those actions can be undo) on a log, and display the corresponding list in the canvas. During recover process, we take each entry of log and use the old state to recover. Note that to support redo function, we use pointers instead of delete log entries during undo.

Preview window

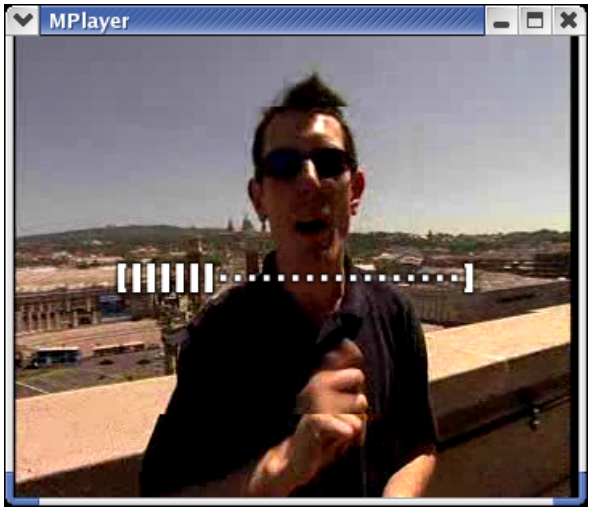

Figure 3.7: Preview window

When user play a source clip in browsing video or the result video, the preview window (See Figure 3.7) will appear. Currently we use MPlayer for playback. User can press *P key* to pause or play the video and press *Q key* to exit. For random access, user can press arrow keys. *Right arrow key* and *Left arrow key* can forward or backward video for 10 seconds, and *Up arrow key* and *Down arrow key* can forward or backward video for 10 minutes. This feature can be consider as a kind of manually highlight playback (Wang et al., 2002). The highlight is a small portion of representative of a long video sequence. During playback, user can play use arrow key to dynamically adjust the highlight portion. This video summary technique is very useful for user to quickly go through the video. However, our current approach is only a simulation of highlight playback, so we will try to embed MPlayer into a Tk window and use pipeline to implement a scroll bar (we have successfully done this with mtv (MpegTV, 2002) application, however since that is not a open source software, we give it up) and also provide options to allow user specify the portion of highlight.

Timeline

Timeline is very useful for stored video temporal editing. Most traditional commercial video editors allow user to drag and drop video clips onto the timeline/storyboard. Then they will

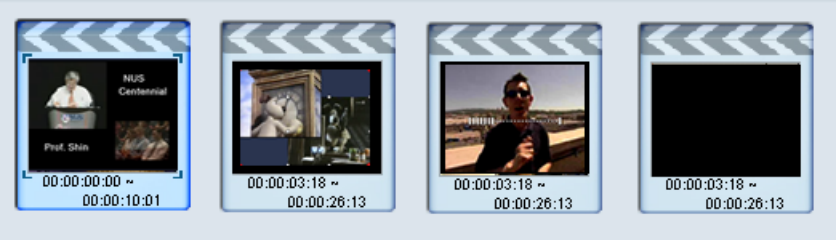

Figure 3.8: A design of scenes

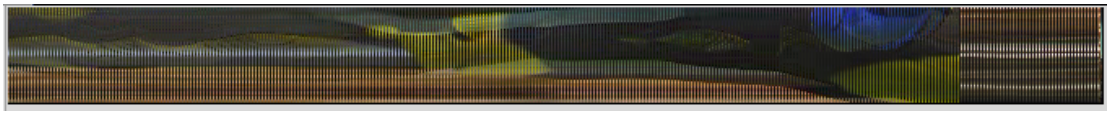

Figure 3.9: A novel timeline bar

be temporal concatenate to a video sequence. Base on traditional design, we propose a novel timeline. Our timeline has two levels. At high level, the video sequence is a series Scenes (See Figure 3.8). Each Scene is a composition of many video clips. User can switch to each Scene to the low level views. In the low level view, each video clips in a Scene has its own timeline. However, in stead of using a bar to represent timeline as traditional approach, our timeline bar is formed of thousands of colored slice. Each slice represents a video frame at particular time point. We extract the dominant colors from a frame and form a slice. Then from the whole bar, user can get the general information such as shot transition (See Figure 3.9). Currently, our timeline haven't been integrated with SLIME since we are improving our algorithm to get better performance.

Make video window

Finally user may want to save the result video to a file. Make movie dialog (See Figure 3.10) allow user to select the location and the format (e.g. MEPG-1, MPEG-2, MPEG-4 DivX, Windows Media Video, Real Media Video, MJPEG, H.263+, FFmpeg's Lossless Video etc.). Then the result file will be produced after clicking start button.

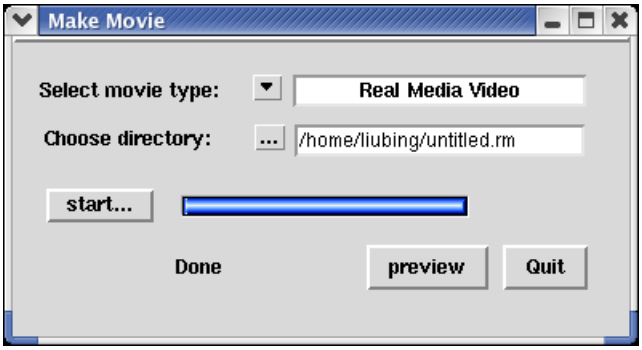

Figure 3.10: Make movie Dialog

Chapter 4

Design and Implementation

This chapter presents the design and implementation details of SLIME. We first address the user interface design issues and then describe the system design and implementation.

4.1 User Interface Design Issues

Bruce Tognazzini has introduced many guidelines about user interface design in his book *Tog on Interface* (Tognazzini, 1992). We adopt those guidelines as our design principles.

Firstly, we use *user-centered design*. One of the contributions of SLIME is to ease video editing task. Therefore, while we are designing this application, we continually remind ourselves that we are designing for an audience. We think about them, think about their problems, and concentrate on how we are going to communicate our ideas to them. To achieve this, we "use scenarios to define and develop a sense of the user space". For example, we have describe a webcast production scenarios in chapter 2 that is director may need to switch the speakers and audiences video steam for some event. For this scenario, we implement a function called *swap* which allow user to drag and drop one stream thumbnail onto other. This simple action will switch those two steams.

Secondly, our avoid *abstract objects*. Tognazzini said that, "All Macintosh interface objects can be seen, heard, felt, or otherwise directly sensed by the user. There should be no abstract, invisible objects in the interfaces, ever." Since people don't want the most abstract interface. They want multiple channels of information. They want neither just words nor just pictures. They want both. The more visual, verbal, vocal, tactile the interface is, the more natural it feels, the more feedback and response it provides, the more confident the user becomes. For example, when user import some video clips, we extract the first frame and make a thumbnail for user so that he can know what this clip is about. To allow user know about it, we also provide duration and filename information. Moreover, we display help tips in the Property panel which can make user more comfortable.

Finally, our interface is *consistent*. "The same action on the part of the user upon the same class of object should result in the same behavior on the part of the system." In our system, no matter adding what type of objects (e.g. video, picture, text) to the Editing panel, it provide the consistent behavior which allow user to drag and drop object from the source panel. Another example is that, during editing a stream, we use dynamic thumbnail to represent videos on both Browsing panel and the Editing panel.

4.2 System design and Implementation

The prototype of SLIME is currently implemented as a Tcl/Tk 8.4.7 application with approximately 8000 lines of code. We use vTcl (Gavin & Courtney, 2002) which is a GUI builder to ease the programming task. TkZinc extension is used to achieve the features Tk dose not support such as transparency and editable text. Exacting frame, information from video clip and images manipulation rely on Dalí library. To handle the music playback and visualization, we use Snack library. Our processing engine is Plasma language described in chapter 2. Since it support limited encoding codecs, we transcode the resultant video using Mencoder. The remainder of this section will describe implementation issues on importing, editing and processing.

4.2.1 Importing

Different types of source require different import method. For stored video, we use Dalí code to parse video header to get information about width height and framerate. Then we count the frames to get the duration. We also exact RGB bytes of the first frame and image resize method to generate a thumbnail. Lastly, we write the RGB bytes of first frame to a temporal PPM file for future use. For live RTSP stream, in order to get the dynamically updated frames, we open a connection to the source stream and start decoding using Plasma. With the support of generate current decoded frame to a Tk photo, we success to make the thumbnails in Browsing panel and Editing panel to be live. Since we cannot get the duration of a live stream, we just live it as blank.

To preview a stored media ,we current use MPlayer to achieve the highlight playback and resizable preview window features. The highlight portion is defined by the framerate. For random access, user can use the arrow keys to forward or backward.

4.2.2 Editing

To achieve WYSIWYG feature, we have to face many graphic user interface challenges. Drag and drop functions can be easily done with Tk code. However, resizes image task is hard to perform with Tk. So we write a Dalí function for resizing Tk photo. Moreover, another Dalí function is written to perform crop task (See Program 4.2.2). It is 10 times fast than the Tk function we write which do the same thing. Both Dalí and Tk cannot make a Tk photo transparent. We solve this problem using TkZinc. Since TkZinc is not a normal Tk canvas (i.e. it has its own widget and syntax), the Editing panel and Crop panel was totally written in TkZinc. With the support of TkZinc, we can also write function to make text in *canvas* to be editable as the same as the text in a Tk *entry*.

Program 3 A dali example for cropping

```
proc ::edit_panel_crop2 {filename x1 y1 x2 y2 img_id} {
    set f [read_ppm $filename]
    set hdr [lindex $f 0]
    set r [lindex $f 1]
    set g [lindex $f 2]
    set b [lindex $f 3]
    set w [pnm_hdr_get_width $hdr]
    set h [pnm_hdr_get_height $hdr]
    set w2 [expr $x2 - $x1]
    set h2 [expr $y2 - $y1]
    set y [byte_new $w $h]
    set u [byte_new $w $h]
    set v [byte_new $w $h]

    set tempy [byte_new $w2 $h2]
    set tempu [byte_new $w2 $h2]
    set tempv [byte_new $w2 $h2]
    rgb_to_yuv_444 $r $g $b $y $u $v
    set clipy [byte_clip $y $x1 $y1 $w2 $h2]
    set clipu [byte_clip $u $x1 $y1 $w2 $h2]
    set clipv [byte_clip $v $x1 $y1 $w2 $h2]
    byte_copy $clipy $tempy
    byte_copy $clipu $tempu
    byte_copy $clipv $tempv
    set r2 [byte_new $w2 $h2]
    set g2 [byte_new $w2 $h2]
    set b2 [byte_new $w2 $h2]

    yuv_to_rgb_444 $tempy $tempu $tempv $r2 $g2 $b2
    write_ppm $hdr $r2 $g2 $b2 "$tmp_crop_$img_id.ppm"
    byte_free $y
    byte_free $u
    byte_free $v
    byte_free $tempy
    byte_free $tempu
    byte_free $tempv
}
```

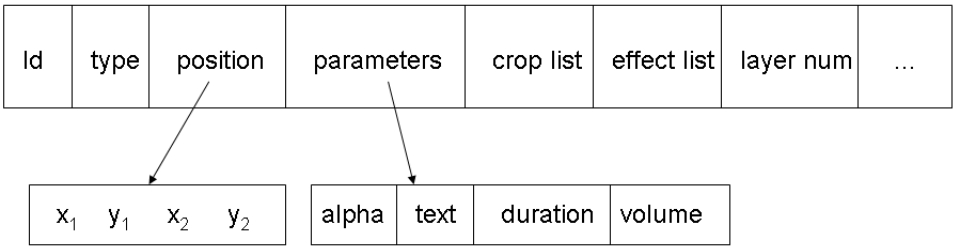

Figure 4.1: Object structure

We have successfully implement most of the features described in chapter 3 except the timeline. Since the main contribution of SLIME is about real-time spatial editing of a live stream, we have focused our research on the spatial editing and network part. For temporal editing part, due to the time constraint, we are still improving our algorithm for represent the frame as a slice. So the timeline haven't been integrated with SLIME. In other words, the current prototype of SLIME dose not support temporal editing.

4.2.3 Processing

We use Plasma beta 0.2 as processing engine. For producing stored video we generate a plasma script and then execute it in a different process. For producing live media supporting real-time editing, we have to execute plasma command inside our application. The drawback is if user terminates this live media, the application will also be terminated. However, for the webcast producing model, we do not have to worry about this drawback since after the termination of all live streams the director has noting to do. To generate plasma script or command, we abstract media object as following structure (See Figure 4.1). After editing by user, the objects are stored in a database. Then we use the following algorithms to parse the database and generate plasma script or command.

Parsing algorithm for side-by-side composition

This algorithm is used to parse the side-by-side spatial relation as shown Algorithm 1. Algorithm 2 shows the *Divide procedure* used by Algorithm 1.

Algorithm 1 Side-by-side parsing algorithm

```
1: let list be media clips database
2: if list length is 0 then
3:     return
4: else if list length is 1 then
5:     get information of clip
6:     generate the Plasma construction command
7: else
8:     call Divide procedure to divide list into two groups so that all clips in one group is on the
       top(bottom) or the right(left) of clips in the other group
9:     generate the corresponding Plasma script
10:    recursively parse group 1
11:    recursively parse group 2
12: end if
```

Algorithm 2 divide procuture algorithm

```
1: sort the list by x1 (if x1 are the same use y1) in ascending order
2: for all clip i under the 1st clip do
3:     find a clip k such that k is on the right of i;
4:     if k dose exist then
5:         divide the list into group1 and group2 such that group1 contains all clips on the left of
           k and group2 contains the remaining clips
6:         return group1 and group2
7:     else
8:         divide the list into group1 and group2 such that group1 contains all clips on the top of
           clip i and group2 contains all the remaining clips.
9:     end if
10: end for
```

Parsing algorithm for overlay composition

This algorithm (Algorithm 3) is used to parse the picture-in-picture spatial relation.

Algorithm 3 divide procuture algorithm

1: sort the database by $x1$, $y1$, $x2$, $y2$
2: get the minimum x1, y1 and maximum x2, y2
3: compute the width and height of result
4: sort the database by layer number
5: **for all** media object in database **do**
6: get the media clip information
7: get correspingding Plasma command
8: **end for**

Parsing algorithm for real-time update

This algorithm is nearly as the same as previous algorithm. The difference is if the object is already created in the database, it only changes its parameters.

Chapter 5

Evaluation

In this chapter, we will describe how we conduct a user study and show the results. Then we analysis the result, compare with other editing tools, and discuss the limitation.

5.1 User Study

Since SLIME is design for making video editing as easy as word processing, we conduct an informal user study on a small group of SoC students. Due to the time and fund constraint, we only have three participants which are all our friends. Two of them have experience of using video editor. One has been a member of Hall video production team, and his experience is to use Adobe Premiere for video post processing. The other one haven't used any expert video editing tools like Premiere, but has ever use Microsoft Movie Maker to produce some game demonstration videos. The remaining one doesn't have any experience of video editing.

To help them to get start, we first show them a about 5 minute long demonstration on how to use SLIME to compose two video tighter, and how to move, resize, crop, transparent a single video clip. Then we let them to separately perform three tasks with different requirements. The first task is an easy one which is similar as the demonstration. The second task is harder which require student to compose 5 videos with complex spatial relation and video effects. The third task is hardest which require student to add image, titles, music and transition effects

that haven't been demonstrate. After half an hour we collect the results video and begin a discussion to get their feedback.

5.2 Results

All the three students roughly finish the three tasks although some results are not exactly the same as requirements described. The student with video studio working experience first finishes all the tasks. During the discussion, he said "SLIME is cool! It's very easy to master." He think that he can also archive the same effects with Premiere, but that will be a tedious work and cost him a lot of time, and he cannot imagine he can do that with in half an hour. The student who have use Movie Maker also give the positive comments such as "The interface is very simple and user friendly. SLIME is very powerful." He describes what he did during producing his game demo with Movie Maker was concatenating several video clips together and adding some transition effects between them. The spatial editing feature of SLIME is very useful and easy to learn. He can easily make more interesting demo videos with this feature. Although the one without any experiences spend the longest time to finish tasks, she seems to be very exciting. During the discussion, she explains that since it's the first time to edit video, she did everything too carefully and slowly at first. But soon she found that using SLIME is quite similar as using PowerPoint or Word. Then she knew what she should do and successfully done the job. She also said that she never thought video editing is so easy, and she may consider buying a digital video camera and produce some short film about her campus life for her parents.

From the above result, we can see that SLIME make video editing as easy as word processing. Moreover, the friendly user interface encourages the participants to explore new features.

5.3 Comparison

Most of the current commercial video editor which targets to average users only cannot perform complex spatial editing task. Although expert software such as Adobe Premiere can perform

the same task as above, it requires user to spend many time to learn and make video editing to be a tedious work. For example, to five video together with a picture-in-picture effect, user have to input the precise size and position for all the five clips. Some research project can perform spatial editing task in a relative easy way. For example Hyper-Hitchcock [24] proposed by Shipman use a tree structure to organize the composite videos. To construct the sample pattern, a 4-level tree is needed. So it is not enough easy to use for novice. Comparing with the above tools, SLIME has powerful feature and a fast and easy-to-use user interface.

The webcast or video conferencing software usually requires hardware support, so we haven't compared the performance on real-time editing live stream feature. However, those softwares such as Director Console only can broadcast one video in a stream. Therefore, SLIME can be considered to be more powerful tool.

5.4 Limitation

The main limitation of SLIME is the on its network part. For the source of streaming media, due to the limitation of Plasma, SLIME only supports the standard RTSP sessions and dose not support Real Network RTSP session which is more popular.

Basically, there are two type of streaming server: live streamer and video-on-demand server. When execute Plasma command to receive stream from a live streamer, the more streams receive the long connection time it require. For example, we run three live streamers written by Plasma on different servers (sentasa, ubin, and kusu) of Multimedia Information Lab. Receiving these streams together and start encoding will cost more than one minute. Therefore, during webcast production, the director has to connect all the live source streams before the lecture begin. Using video-on-demand servers instead of live streamers can solve the performances problem. For example, we start three FFserver on the above three island server. It will only cost less than 10 second to begin encoding. However, since video-on-demand server is not live, we cannot use it to produce live webcast. To solve this problem, we can use capture device (e.g. TV card) to

feed the FFserver. We will talk about the detail in next chapter.

Another performance limitation is due to the Plasma encoder. During encoding and output the resultant video as a stream, the Plasma encoder will consume more and more memory which will slow down the whole system. As a result, producing for example two-hour long webcast require high performance workstation for processing.

There are also some limitations on the user interface. So far, we haven't implemented our timeline. In other words, the current version of SLIME doesn't support temporal editing. Moreover, the folder view collections haven't be implemented either.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We have presented SLIME, an easy-to-use and video editing tool that allow user to real-time compose live and stored video together. In summary, our SLIME has two main features which are powerful spatial editing feature and live stream real-time editing feature. Moreover, the user-friendly GUI design of SLIME makes video editing as easy as word processing. In this report, we begin with introducing some software tools. We also exploited the powerful spatial editing features in Plasma, and review the existing CHI designs on video editor user interface. After we described the intuitive GUI of SLIME, we discussed the design and implementation issue of a prototype. Then we show and analysis the results our informal user study and give a comparison between SLIME and other related works. After address the limitations, we will finally recommend the future work.

6.2 Future work

We have addressed the limitation of SLIME in the preview chapter. Also there are several features could be added to SLIME such equipment control and QoS event handling. In this section, we will recommend our future works in following aspects.

6.2.1 User interface

First and foremost, we will complete our GUI with those features we haven't implemented. Those designs include the folder view collection and timeline. For the timeline, we will improve our algorithm to get better performance, and then integrate it into the main interface to support temporal editing such as cutting and concatenating. Currently our preview window and highlight preview feature relay on the keyboard function of MPlayer, so we will implement our own preview window using SDL library. Moreover, we will propose more CHI designs and bring more convenience to the user. For example we can allow user to select and group the clips together and edit and whole group. We will conduct a formal user study and get more feedbacks on the interface.

6.2.2 Indiva integration

As we mentioned in chapter 2, Indiva (Ooi et al., 2001) is a middleware for distributed media environment which use a file system metaphor to access resources and provide a unified set of abstractions and high-level operations for hardware devices. We will integrate Indiva into SLIME so that our application can manage resource and control the equipments in the lecture rooms. For example, it will allow user to move up or down the camera and adjust the microphone's volume.

6.2.3 Event handling

Plasma has powerful feature on event handling. Currently, our application only uses this feature to achieve transition effects. We will exploit other event features such as QoS event and remote controlling. For example, it will detect the packet loss and allow user to reduce the frame rate of a video steam.

6.2.4 Performance

To improve the performance, we will enhance the memory management of both Plasma and SLIME. To solve the streaming source problem, we will try to make use the RTPtv system (Delco, 2001) developed by Berkeley Multimedia Research Center. RTPtv is an inexpensive system for transmitting production quality television over IP-based network, and can send audio and video streams using the IETF RTP standard multicast or unicast media streaming.

References

[1] Adobe (2004). Premiere Pro, http://www.adobe.com.

[2] Agmanolis, S. (1997). ISIS: A multilevel scripting environment for responsive multimedia, http://isis.www.media.mit.edu/projects/isis/.

[3] Apple (2004a). Final Cut Pro, http://www.apple.com.

[4] Apple (2004b). iMovie, http://www.apple.com.

[5] Casares, J., Long, A., & et al. (2002). Simplifying video editing using metadata. *Proceedings of the conference on Designing interactive systems: processes, practices, methods, and techniques* (pp. 157–166), London, England, June, 2002.

[6] CenaSoftware (2004). TkZinc, http://www.tkzinc.org.

[7] Delco, M. R. (2001). *Production quality internet television* (Technical report). Berkeley Multimedia Research Center.

[8] Gavin, C., & Courtney, D. (2002). Visual tcl, http://vtcl.sourceforge.net.

[9] Girgensohn, A., Boreczky, J., Chiu, P., Doherty, J., Foote, J., Golovchinsky, G., Uchihashi, S., & Wilcox, L. (2000). A semi-automatic approach to home video editing. *Proceedings of the 13th annual ACM symposium on User interface software and technology* (pp. 81–89), San Diego, California, November, 2000.

[10] Holden, L. (1996). ooMPEG: an Object oriented MPEG video decoder. software online, http://www.cs.brown.edu/software/oompeg/.

[11] LiveNetworks (2004). LIVE.COM Streaming Media, http://www.live.com.

[12] Machnicki, E., & Rowe, L. A. (2002). Virtual director: Automating a Webcast. *Proceedings of 2002 SPIE Multimedia Computing and Networking* (pp. 56–58), San Jose, California, January, 2002.

[13] McCanne, S. (1999). Scalable multimedia communication with internet multicast, light-weight sessions, and the mbone. *IEEE Internet Computing, 3(2)*, jun, 1999, 33–45.

[14] Microsoft (2003). Movie Maker, http://www.microsoft.com.

[15] MpegTV (2002). MpegTV Player v1.2.5, http://www.mpegtv.com.

[16] MPlayerTeam (2004). The Movie Player of Linux, http://www.mplayerhq.hu/.

[17] Ooi, W. T., Pletcher, P., & Rowe, L. A. (2001). Indiva: A middleware for managing distributed media environment. *Proceedings of Multimedia Computing and Networking*, San Clara, California, January, 2001.

[18] Ooi, W. T., & Smith, B. (1999). Dali: A mutimedia software livrary. *Proceedings of 1999 SPIE Multimedia Computing and Networking* (pp. 25–27), San Jose, California, January, 1999.

[19] Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit.* Addison Wesley.

[20] Patel, K., Smith, B. C., & Rowe, L. A. (1993). Performance of a software MPEG video decoder. *Proceedings of First ACM International Conference on Multimedia* (pp. 75–82), Anaheim, California, August, 1993.

[21] Rowe, L., Harley, D., & Pletcher, P. (2001). *BIBS: A lecture webcasting system* (Technical report). Berkeley Multimedia Research Center.

[22] Rowe, L., & Jain, R. (2005). ACM SIGMM Retreat report on future directions in multimedia research. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, *1*, February, 2005, 3–13.

[23] SDLProgect (2004). Simple Directmedia Layer, http://www.libsdl.org.

[24] Swartz, J., & Smith, B. C. (1995). Rivl: A resolution independent video language. *Proceedings of 3th ACM International Conference on Multimedia* (pp. 235–242), San Francisco, California, November, 1995.

[25] TclCoreTeam (2004). Tcl/Tk 8.4.7, http://www.tcl.tk.

[26] Tognazzini, B. (1992). *Tog on Interface.* Addison Wesley.

[27] VideoLan (2004). VideoLan Client, http://www.videolan.org/.

[28] Wang, Y., Zhao, P., Zhang, D., Li, M., & Zhang, H. (2002). Myvideos: a system for home video management. *Proceedings of the 10th ACM international conference on Multimedia* (pp. 412–413), Juan-les-Pins, France, December, 2002.

[29] Yang, M., Mu, X., & Marchionini, G. (2003). Vivo: a video indexing and visualization organizer. *Proceedings of the third ACM/IEEE-CS joint conference on Digital libraries* (pp. 403–403), Houston, Texas, May, 2003.

[30] Yu, T. P., Wu, D., Meyer-Patel, K., & Rowe, L. A. (2001). dc: A live webcast control system. *Proceedings of 2001 SPIE Multimedia Computing and Networking* (pp. 111–112), San Jose, California, January, 2001.

[31] Zhu, T. (2003). Design of a scripting language for processing streaming media. *Proceedings of NUROP Conference*, Singapore, October, 2003.