

Interactive Cinema
Technical Note

August 1993

LogBoy and FilterGirl:

Tools for Personalizable Movies

Ryan Evans
Interactive Cinema Group
MIT Media Laboratory

Abstract

LogBoy and FilterGirl constitute a toolkit designed specifically for building personalizable movies which use a fluid interaction mode. Fluid interaction is an important interaction metaphor which relies on narrative playout which is uninterrupted by the interface. By avoiding periodic viewer queries, fluid interaction encourages reverie and continuity in storytelling. Fluid interaction requires not only an interactive story structure, but also a machine-readable representation of content. LogBoy and FilterGirl provide tools for these ends. LogBoy is a video database tool which allows the moviemaker to attach descriptions to video clips. LogBoy's interface provides a graphical "overhead" view of descriptions which aids in the creative process. FilterGirl provides a way for creators to quickly and easily create personalizable story structures. FilterGirl implements a filter-based story description language as well as a story structure previewing facility. Functionality of the tool set is discussed as well as interface design and implementation details.

1. Introduction

LogBoy and FilterGirl are two complimentary applications (a video database tool and a story structure editor) which make up a tool kit for the creation of personalizable movies. Personalizable movies are movies which can be tailored to the preferences of a particular viewer using parameters defined by the creator of the movie. LogBoy and FilterGirl are specifically designed to create personalizable movies which make use of a fluid interaction mode. Fluid interaction is an interactive paradigm which stresses reverie and storytelling by presenting the personalizable movie in an uninterrupted format based on user preferences.

Often, essential storytelling conventions (e.g. reverie and continuity) suffer in the context of interactive narratives because the viewer is periodically interrupted by the interface. For example, the typical computer adventure game repeatedly asks the viewer (either explicitly or implicitly) which way he or she wants to go and will not continue until that decision is made. This type of interface makes it difficult to tell a coherent story since the viewer must constantly worry about driving the story on to the next stage rather than the progress of the story itself.

Fluid interaction provides uninterrupted interactive stories by presenting interactive movies in an uninterrupted format based on user preferences. Under this paradigm the burden of driving the story on to the next stage no longer lies with the viewer. Instead, the computer drives the playout of the story guided by parameters set by the user. For example, the viewer might set a violence level parameter before watching a favorite television show. The show would then play out with the amount of violence the viewer was interested in.

Fluid interaction requires that the computer not only use some kind of interactive story structure, but also representation of content. LogBoy and FilterGirl are tools for modeling content and interactive story structures. LogBoy is a video database tool which provides a view of video descriptions suited particularly for interactive narratives. FilterGirl is a interactive story structure editor designed specifically for creating fluid personalizable narratives. This document describes the LogBoy/FilterGirl tool set in detail. Chapter 2 describes LogBoy's video description framework and user interface. Chapter 3 describes FilterGirl's filter language for creating interactive narratives, user interface and debugging options. Chapter 4 briefly outlines the implementation of the tool kit.

2. LogBoy Design

LogBoy is a database tool for creating and editing descriptions which are attached to video clips. LogBoy differs from traditional video loggers and computer database applications in three respects. First, LogBoy is designed specifically as a tool for creating descriptions for personalizable movies which rely on description based content selection. As such it provides the moviemaker with an "overhead" view of the database which can point out database properties in relation to a particular story structure. Second, LogBoy allows the author of an interactive movie to engage in truly iterative design since it is part of a single-platform, integrated toolkit for creating and editing personalizable movies. Finally, LogBoy encourages sketchy descriptive bases for content. Sketchy descriptions, as discussed in Chapter 2, provide an efficient representation paradigm for content designed for limited reuse (e.g. personalizable movies).

2.1. Video Clips

LogBoy's basic unit of content is the video clip. As discussed in Chapter 2, a video clip is a non-overlapping segment of video with a specified beginning and end point. Under this paradigm all descriptive structures associated with a clip are valid for the entirety of the segment. The presentation of a personalizable movie is ultimately made up of a computationally selected sequence of video clips. Video clips are, by definition, always shown in their entirety. One of the major reasons that LogBoy uses the video clip as a database model is to avoid the problem of computationally choosing valid in and out points for a segment of video. This provides a well-defined division of labor between the human filmmaker and the computational clip selector. The human decides what constitutes a good clip while the clip selector (with guidance from human created story structures) decides what constitutes a good sequence of clips.

Video clips are represented graphically within the LogBoy environment as draggable icons. Figure 1 shows a collection of four video clip icons. Each icon shows a key frame from its associated video clip along with two buttons used for obtaining more information. The info button, indicated by the question mark, opens a window which displays important information about the video clip. Currently, this includes the location of the clip on the hard disk, the title of the clip and descriptions which have been attached to this clip in LogBoy. The playout button, indicated by the greater-than sign, opens a digital video playout window in which the video clip can be previewed in its entirety.

2.2. Slots and Values

Descriptions in LogBoy are stored in a simple slot and value structure. This means that each database (collection of video clips) has some associated characteristic types called slots. Each clip has one or more values associated with it for each slot. This means that each clip has the same set of slots as every other clip in the database, while the values within those slots are unique to each clip. Typical slots might include "characters that

appear within the clip", "location where clip takes place" or "part of the story". (Often slots are given shorter, mnemonic names such as "Characters" or "Setting".) Values take on meanings only in relation to these slots. For example, a particular clip might have the value "Betty" within its "Characters" slot or the value "New York" in its "Setting" slot. The slot/value structure is discussed in more detail in Chapter 2. The LogBoy system allows the filmmaker to associate zero, one or multiple values with each slot for a video clip. Figure 2 shows a conceptual model of slots and values attached to video clips.

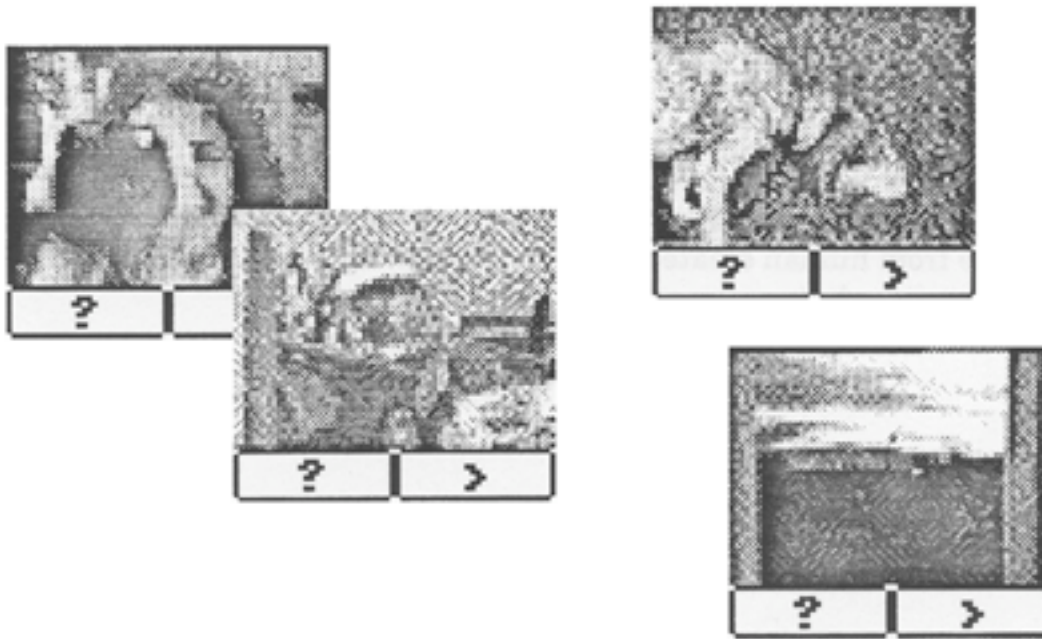


Figure 1: A collection of four video clip icons.

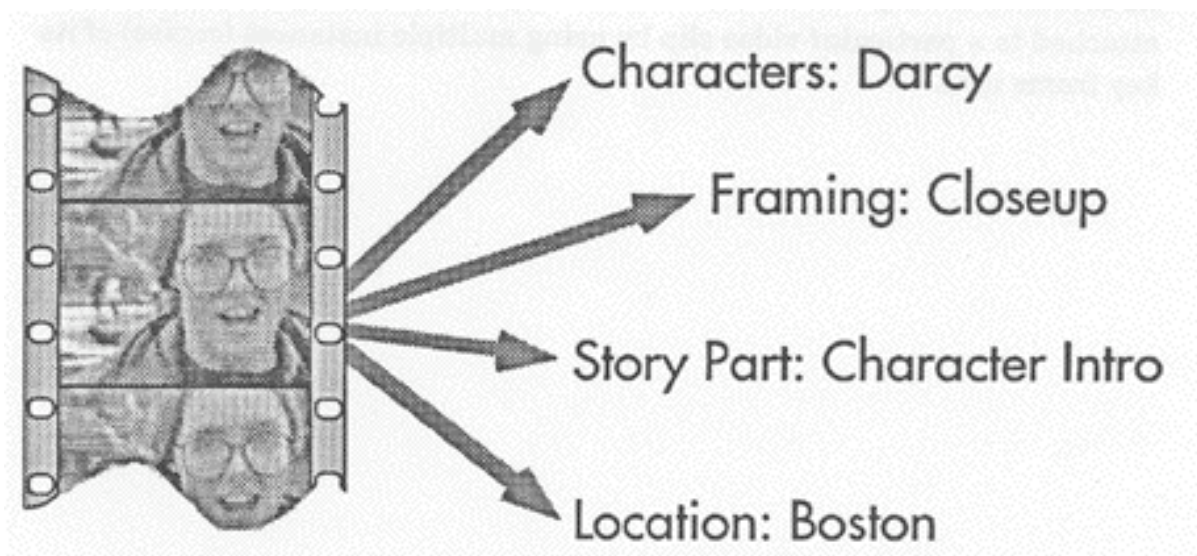


Figure.2: A conceptual model of slots and values attached to a video clip.

The LogBoy application does not limit the moviemaker to a specific set of standard slot types, nor does it limit the number of slots that may be associated with a clip database. This is because LogBoy is designed to create video databases in conjunction with personalizable story structures. In creating content-based personalizable movies, the descriptions in the content database are generally paired closely with the content selection mechanisms and interactive story structures. This means that the creator can use sketchy descriptive bases and highly personalized description spaces that might not make sense within some more generalized content selection scheme.

2.3. A Graphical Description Space

LogBoy represents the three major components of its database structure {video clips, slots and values} graphically to create a description map of the database. As described above, video clips are represented on the screen as key frame icons. The position of an icon on the screen gives the user information about the descriptions attached to it. Additionally, these icons can be repositioned to change their descriptions.

To show the descriptions attached to a video clip slots and values are also shown graphically on the screen. Slots are represented as windows within the Macintosh environment {called "slot windows"}) and values are represented as colored, rectangular areas within slot windows {called "value areas"). If a video clip icon is positioned within a value area which is in turn located within a slot window then that value is attached to that video clip for that slot. Repositioning the clip icon into a different value area will change its attached description for that slot. Multiple slot/value pairs may be attached to a particular video clip by using multiple instances {copies} of its key frame icon.

Figure 3 shows a typical LogBoy session including clip icons, slot windows and value areas. In this example the database currently includes six video clips and three slots. Each of the three windows ("Mood", "Setting", "Characters") depicts the descriptive structures for its associated slot. Each slot window contains value areas and video clip icons. Video clip icons within value areas inherit slot/value pairs based on the window in which they appear and the value area in which they reside. The same icon can appear in multiple slot windows so that multiple slot/value pairs may be associated with it. For example, the icon which appears in the "Alice" value area of the "Characters" slot inherits the "Characters"/ "Alice" slot/value pair. The same icon also appears in the value area "Happy" within the slot window "Mood" and the area "New York" in the window "Setting". Thus the slot/value pairs associated with this clip are "Characters"/ "Alice", "Mood"/ "Happy" and "Setting"/ "New York". Pressing the info button (marked with a question mark) on the bottom of the clip icon opens a description window for that clip which lists all of the associated slot/value pairs (see Figure 4).

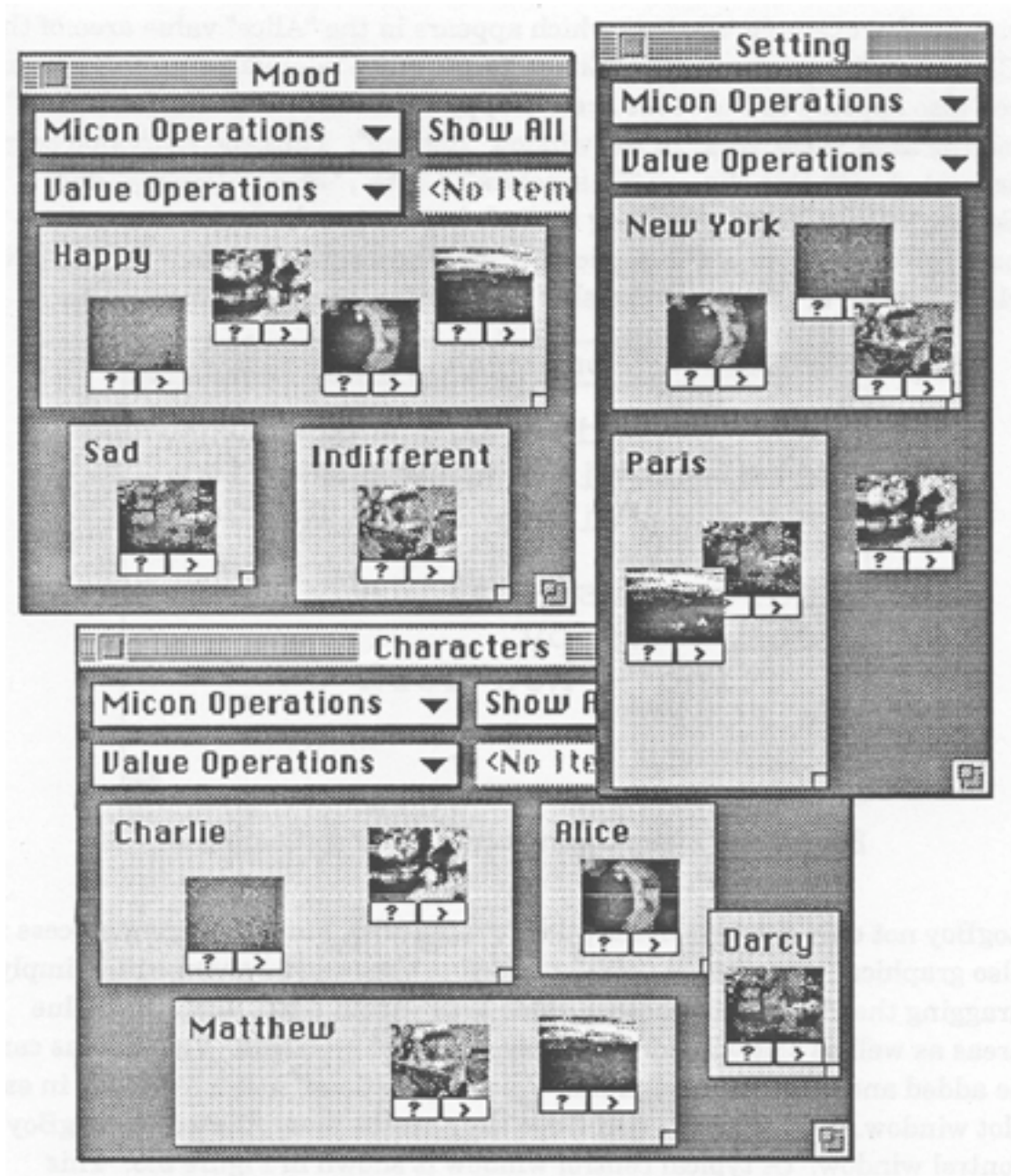


Figure 3: A typical LogBoy scene with three slot windows.

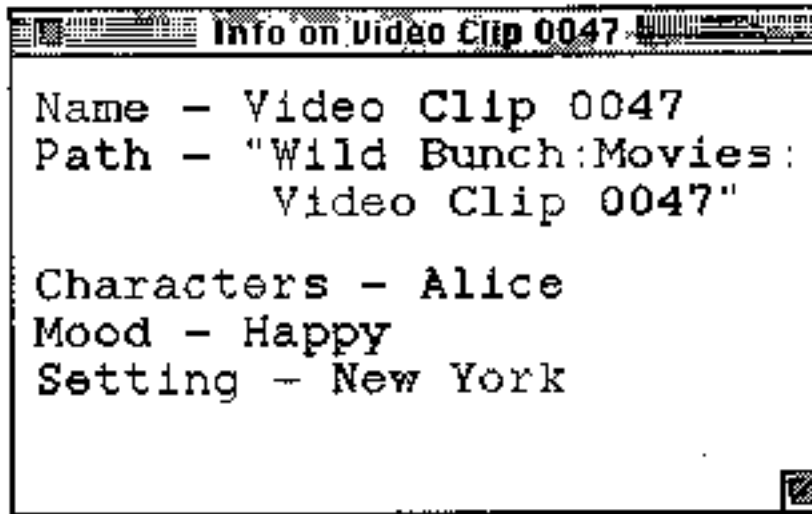


Figure 4: A clip description window containing slot/value pairs.

LogBoy not only displays descriptions graphically, but the logging process is also graphical. Slot/value pairs attached to clips can be changed by simply dragging the clip icon to another value area within a slot window. Value areas as well as slot windows are draggable and resizable. Value areas can be added and deleted through the "Value Operations" menu available in each slot window. Slot windows and video clips can be manipulated via LogBoy's control window. (A typical control window is shown in Figure 5. This control window corresponds to the session shown in Figure 3.) The control window contains a list of all currently defined video clips in the database (listed by their titles) along with a list of all currently defined slots in the database. Slots can be added and deleted via the "Slot Operations" menu. Video clips can be added and deleted via the "Clip Operations" menu. Slot windows can be opened by double clicking on their title in the list. Video clip description windows can also be opened by double clicking on their description in the list.

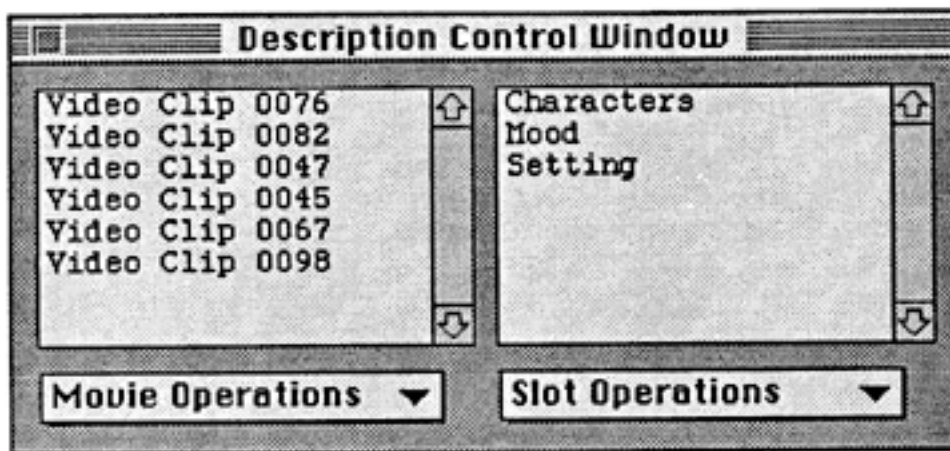


Figure 5: The LogBoy control window showing a list of video clips and a list of slots.

2.3.1. Slot Window Filters

Slot window filters provide a way for the creator to effectively make simple database queries and also manage icons within slot windows by showing or hiding the clip icons in a slot window. Slot window filters work in a similar way to the story filters described in the next chapter, however slot window filters (currently) only come in one flavor while story filters come in about a dozen different types. Slot window filters simply include or exclude clips that have a particular description (slot/value pair) attached to them in LogBoy.

Slot window filters can be applied via the two menus available at the top of each slot window. The top menu allows the user to select between "Show All Clips", "Show Only If" and "Show Only If Not", while the bottom menu allows the user to select a slot/value pair from a list. All possible slot/value pairs are available in this menu even if they are not currently used. Figure 6 shows the effect of different filters on the "Characters" slot window originally shown in Figure 3. The top window in Figure 6 shows the "Characters" window with a "Show Only If" "Mood = Happy" filter applied. This filter hides all clip icons which do not meet this descriptive criterion. Similarly, the bottom window shows the "Characters" window with a "Show Only If" "Setting = Paris" filter in effect. This hides a different selection of icons.

Slot window filters are helpful in two ways. First, the filmmaker can make simple Boolean queries in a slot window. For instance, in the first window it is very easy to pick out all of the clips which both contain the character "Matthew" and are described as being "Happy." Second, the set of video clips within a particular slot window can be simplified to include only clips which are relevant to the characteristic described by that slot. This can be helpful in more complicated databases when a particular characteristic (i.e. slot) is not relevant to an entire set of clips. For instance, imagine a moviemaker creating a personalizable movie which includes both a chase scene and a dialog scene. She might want to describe each of the clips in the dialog scene by the topic of conversation depicted. To do this she would create a "Conversation Topic" slot for the database. Obviously, this characteristic has no bearing on clips intended for the chase scene. If she made a slot called "Scene" with two values "Chase" and "Dialog" and logged the database under these slots and values, she would be able to hide all of the chase scene clips within the "Conversation Topic" slot window. In many cases this greatly increases the understandability of a slot window and reduces visual complexity.

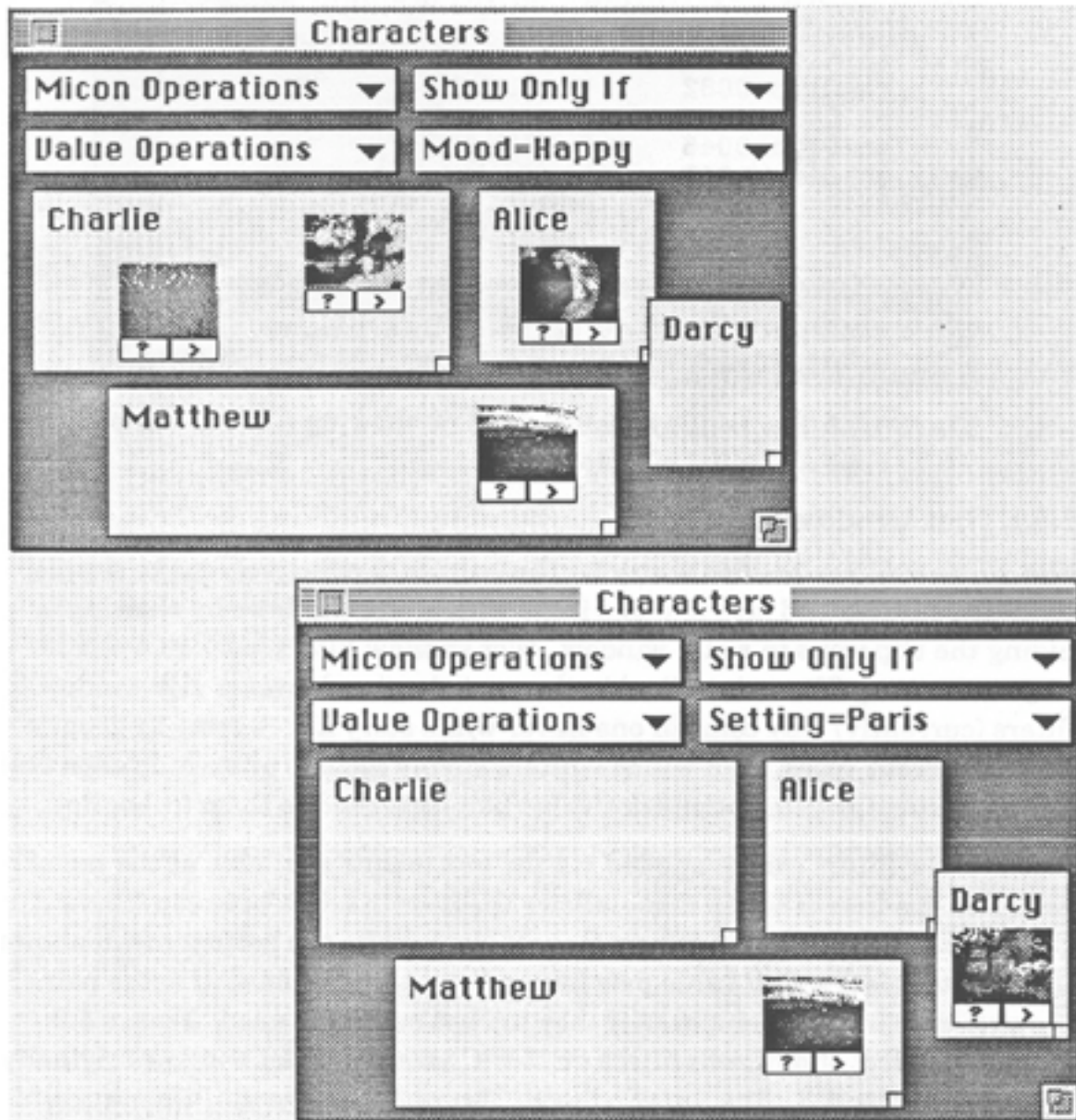


Figure 6: A slot window with slot window filters applied.

2.4. LogBoy and Personalizable Movies

LogBoy's graphical interface to the database provides the filmmaker with a way to visually scan and evaluate a collection of clips for specific descriptions, anomalies and general trends. In LogBoy it is very easy to quickly get an idea for how the database as a whole is divided under a certain characteristic. For instance, in the small database shown in Figure 3 we can see very quickly that there are many more "Happy" clips than "Sad" clips in the "Mood" slot window.

This interface works particularly well for designing databases for computational shot selectors and personalizable moves. This is because the filmmaker must make sure that the shot selector has a fully populated database into which it can make queries. The selector cannot ask the database for a close-up of Alice in New York when none exists. The graphical nature of LogBoy allows the filmmaker to quickly scan the database to make sure it is fully populated. FilterGirl also helps in the task of fully populating a database. This process is described in the next chapter.

These same qualities which make LogBoy ideal for use in creating personalizable movies detract from its use as a more general database tool. The LogBoy interface has a practical limit on the number of clips which may be stored in one database. This is because the screen and windows become very cluttered with icons when working with large databases. This is almost never a problem when creating personalizable movies since the number of clips for a particular movie is generally small because each clip is designed for use in that movie alone. Similarly, there is also a practical limit on the number of descriptive structures (slots and values) that can be built in one LogBoy database. Similarly, this is because the screen real estate will become very cluttered and confusing. Luckily, this is even less of a problem in creating personalizable narratives since descriptions are closely tailored to work with a particular shot selector mechanism resulting in sketchy descriptions.

3. FilterGirl Design

FilterGirl allows a maker of interactive movies to create, edit and preview simple story structures in conjunction with a LogBoy created database of clips. FilterGirl's design centers around filters which come in several different types. The several filter types provide a flexible, hierarchical language within which time dependent and context dependent movie structures can be built. This chapter describes the filter types and their behaviors, how the author can create and edit filter instances, how filter based movies can be previewed and debugging options.

3.1. The Filter Types

At the simplest level a filter takes a set of video clips as input and returns some selected subset of those clips as output (see figure 7). Filters can select clips based on descriptions which have been created in LogBoy (e.g. all of the clips containing the character Nicole). More complicated filters can be constructed by taking advantage of the simple interface between any particular filter and the clip database (i.e. a set of clips as input, some subset of those clips as output). This simple interface allows the creator to combine multiple filters in many interesting ways. Set operations (e.g. set intersection or set union) become possible for clip selection by chaining the inputs and outputs of filters. Dynamic, time-dependent story structures can be modeled by swapping filters with other filters as the viewing experience plays out. Additionally, filters can be selected in real time based on viewer input or more complicated context dependent means (e.g. rule bases).

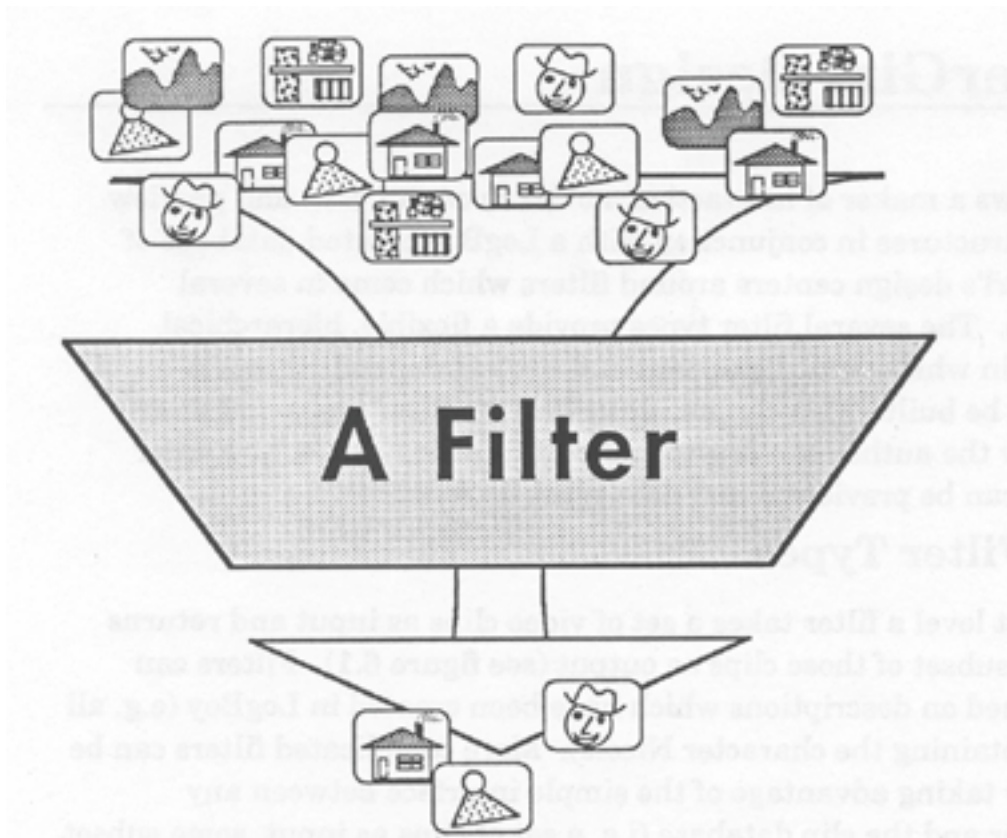


Figure 7: A typical filter.

FilterGirl effects many of these filter combinations by providing the creator with a varied set of filter "types" from which to create story structures. Filter types have been constructed which implement clip selection by set operations, simple templates, time dependent structures, context dependent means and viewer interaction. The more complicated filter types may contain other, simpler filters within themselves and combine their behaviors according to predefined rules. For example, the "multi-filter" filter type (see below) implements traditional set intersection by containing two or more other filters within itself. Each of the defined filter types is described below.

3.1.1. Basic Filter

The basic filter is also sometimes called the identity filter. It outputs exactly the same set of clips which are given to it as input. It is useful as a placeholder in time dependent structure when the creator does not want to filter out clips, but needs to place a filter between other filters. It is also useful when creating new types of filters since the basic filter forms the CLOS super-class from which all other filter types inherit their basic behavior.

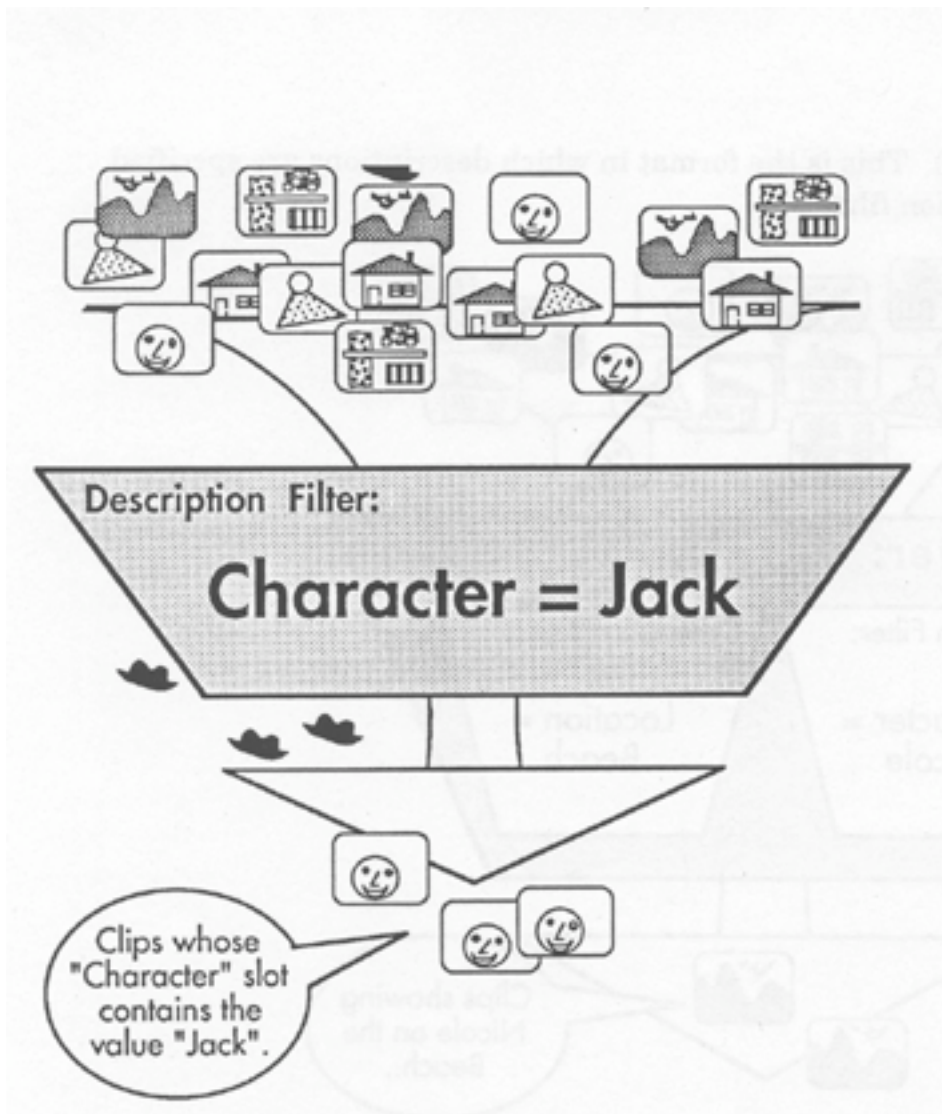


Figure 8: A description filter which selects clips described as containing the character Jack.

3.1.2. Description Filter

The description filter is the simplest and most widely used of all of the filter types. Any particular instantiation of a description filter selects only clips which have a particular description attached in the LogBoy application. Thus a movie maker can create a description filter which selects only clips in which a particular character appears, which take place at a certain location or have some other particular characteristic described in a LogBoy database. Each instantiation of a description filter only selects clips on one specific description. Figure 8 illustrates a description filter which selects only clips in which the character Jack appears. When selecting on combinations of descriptions the creator must rely on more complicated filters to combine description filters (e.g. the multi-filter). As noted in previous chapters, descriptions in LogBoy are annotated as

slot/value pairs (e.g. Character/Nicole). This is the format in which descriptions are specified within a description filter.

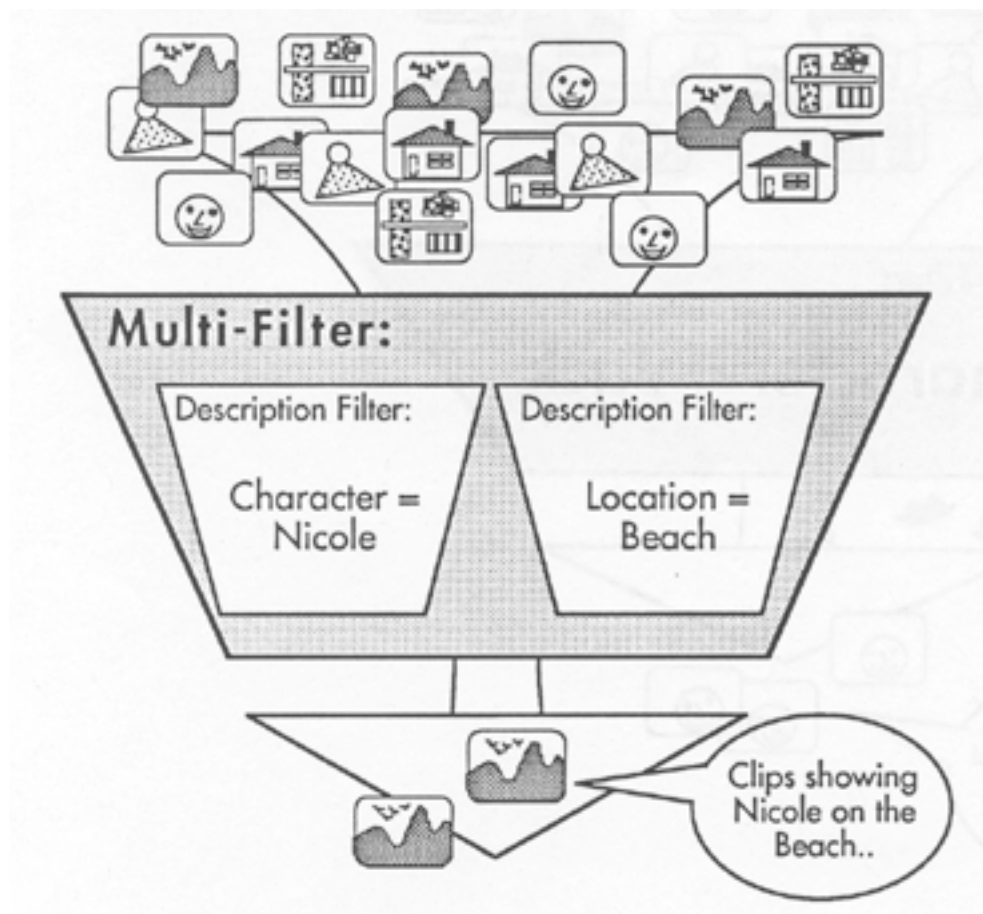


Figure 9: The multi-filter.

3.1.3. Set Operation Filters

The set operation filter family implements simple set operations such as intersection, union, negate, etc. As such, they are not dependent on an internal clock, viewer interaction or the context of their operation. The set operation filter types provide the simplest means of combining other filters.

3.1.3.1. Multi-Filter (Intersection Filter?)

The multi-filter allows the movie maker to combine two or more filters with a set intersection operation. Each of the combined filters is run in turn and the common clips which are selected by all of the filters are gathered as the output of the multi-filter. Figure 9 shows a multi-filter instance which combines two description filters. The first of the description filters selects only clips in which the character Nicole appears. The second

description filter selects clips which are located on the beach. The multi-filter combines the outputs of its contained filters using set intersection so the output of the multi-filter consists only of clips which are described as containing Nicole at a beach location.

3.1.3.2. Union Filter

The union filter implements the set union operation on two or more inner filters. Each of the inner filters are run in sequence and all of the clips returned by the filters are gathered as the output of the union filter (duplicate clips are removed). For example, a union filter could be used to combine a "Character = Jack" description filter with a "Character = Nicole" description filter to return all clips in which either Jack or Nicole appears.

3.1.3.3. Negate Filter

Any instantiation of the negate filter type contains exactly one filter inside of it. The negate filter essentially inverts the behavior of its contained filter by returning all clips given as input to the negate filter which are not returned by the inner filter. A simple example: by putting a "Location = Beach" description filter inside of a negate filter one can create a filter which selects all clips which do not take place at the beach.

3.1.4. Context Dependent Filters

There are two generalized types of context dependent filters: the rule filter and the eval filter. Their behavior can depend on arbitrary aspects of the movie playout state and viewer interaction. Other filter types not in this section also depend on particular parts of the movie context, but rule filters and eval filters allow the movie maker access to any part of the movie state by allowing arbitrary lisp expressions.

3.1.4.1. Rule Filter

The rule filter makes use of an arbitrary lisp predicate (a predicate is a function which returns either true or false) to choose which of two contained filters to make valid. If the predicate returns true the "then filter" is chosen. If the predicate returns false the "else filter" is chosen. The predicate function has access to all parts of the movie state such as viewer interfaces, previously shown clips (the movie transcript), the entire clip database and the state of other filters in the movie. For example, suppose one wanted to build a filter which would introduce the character Jack only if he has not been introduced yet. A rule filter could be built with a predicate which would reference the movie transcript to see if any clips of Jack have been shown yet. If clips of Jack have been shown then it might choose a different character. If clips of Jack have not been shown then it would choose an introductory clip of Jack.

3.1.4.2. Eval Filter

Rather than containing and combining other filters, the eval filter contains arbitrary lisp code which evaluates into a filter object. This allows the movie maker a way to construct

filters on the fly which can depend on any part of the movie state including the clip database itself, the transcript of previously shown movie clips, viewer interfaces and the state of other filters in the movie. Eval filters also provide a way to hook into existing lisp programs which might be useful in creating more complex interactive behaviors (e.g. network applications, expert systems, etc.).

3.1.5. Template Filters

The template family of filters includes filter types which apply their contained filters over time. This makes it simple for the creator to sequence clip playout over time or across a viewing experience.

3.1.5.1. Shot Template Filter

The shot template filter is the most familiar of the template filters. It provides the capability of making filters valid in sequence, one filter per clip, as the viewing experience plays out. This capability allows the creator to guide the playout of the final movie on a clip by clip basis. Figure 10 shows a shot template filter which contains four filters inside of it. The filters represent four steps in the process of a character, Woody, cooking an omelet. In this example, the shot template filter is currently set to the third internal filter, meaning that the first two shots have already been shown in this movie and that only the third filter will be used in selecting a subset of the clips on the input.

Any instantiation of the shot template filter type" can be set to repeat when it reaches the end of its filter sequence. This option can be used to create simple looping structures.

3.1.5.2. Time Template Filter

The time template filter allows the movie maker to specify clocked lengths of time over which specific filters are valid. This is done by associating a "start valid time" and an "end valid time" with each filter inside the time template filter. The time template filter watches the system clock and routes its input through the appropriate internal filters. For example, we could make a movie which starts with 60 seconds of clips featuring the character Nicole, continues with 120 seconds featuring the character Jack and ends with 30 seconds of clips featuring both characters. If two or more filters are scheduled to be valid during the same time period their outputs are combined with set intersection, as with the multi-filter. If other combinations are desired more complicated filter structures can be built with multiple time template filters. (Note: Filter valid times only specify when filters are turned on and off. The clips that are selected may in fact run over these times.)

The time template filter has a repeat mode much like the shot template filter, which zeros its internal clock when filter valid times have been run through. Also, the time template filter can run off of one of two separate internal "clocks". The first choice is the absolute clock provided by the system. This behaves as expected. The second choice is to add up the length of all clips which have already been viewed and match this time against the

filter valid times. This allows the viewer to skip over unwanted clips and move on to later clips without disturbing the normal playout.

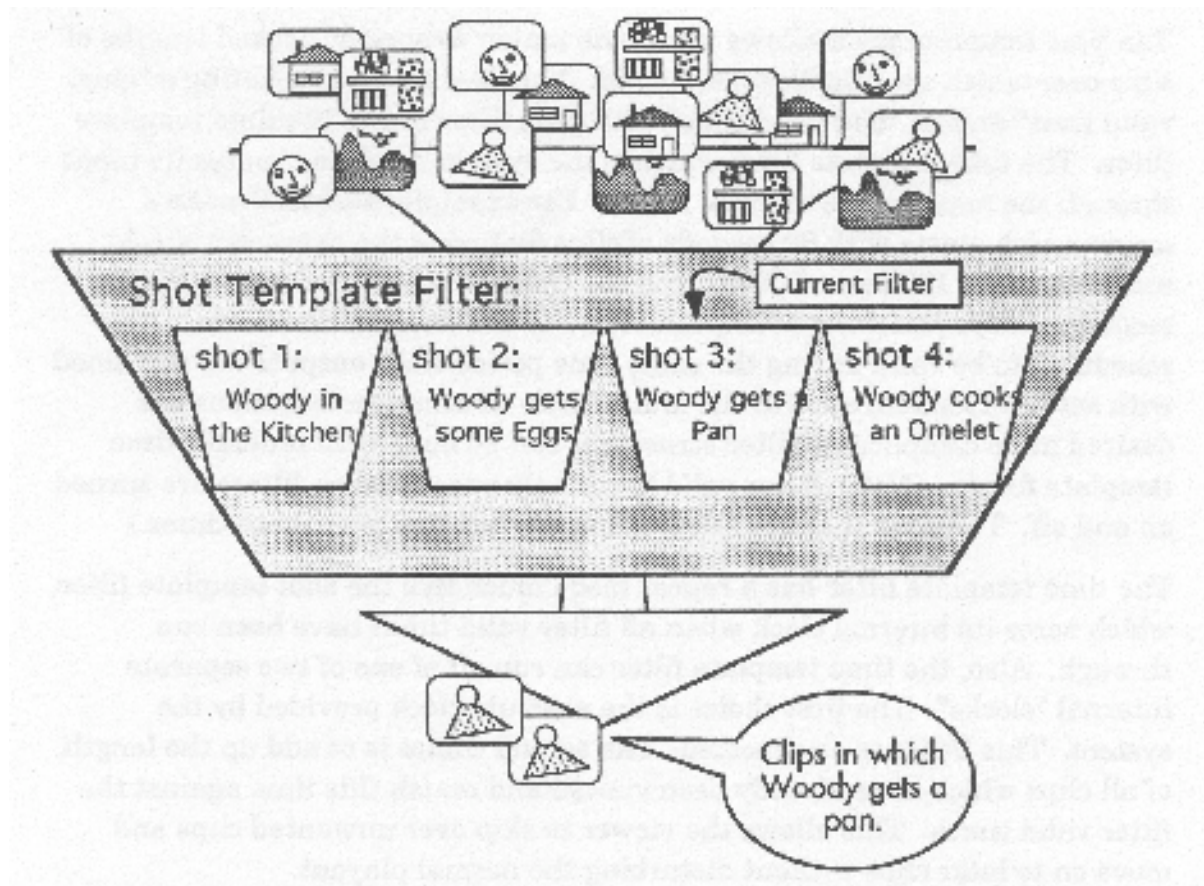


Figure 10: A shot template filter.

3.1.5.3. Advancable Template Filter

The advancable template filter is the most powerful of the template filters. It is useful when the movie maker wishes to abstract the development of a personalizable movie into sections or scenes. The advancable template filter works much like the shot template filter in that it contains a list of filters within itself and makes them valid (turns them on and off) in sequence. Unlike the shot template filter, the advancable template filter does not move to the next filter in its sequence after each shot is played out. Instead, it advances to the next filter when it receives a special signal.

This signal, called the end of filter signal, can be generated by any type of filter which can contain another filter. This is because the end of filter signal can take the place of a normal filter. When an end of filter signal is called upon to filter a set of clips, it instead sends a signal to the advancable template filter which contains it. This signal tells the advancable template filter to turn on the next filter in its sequence to narrow the clip set.

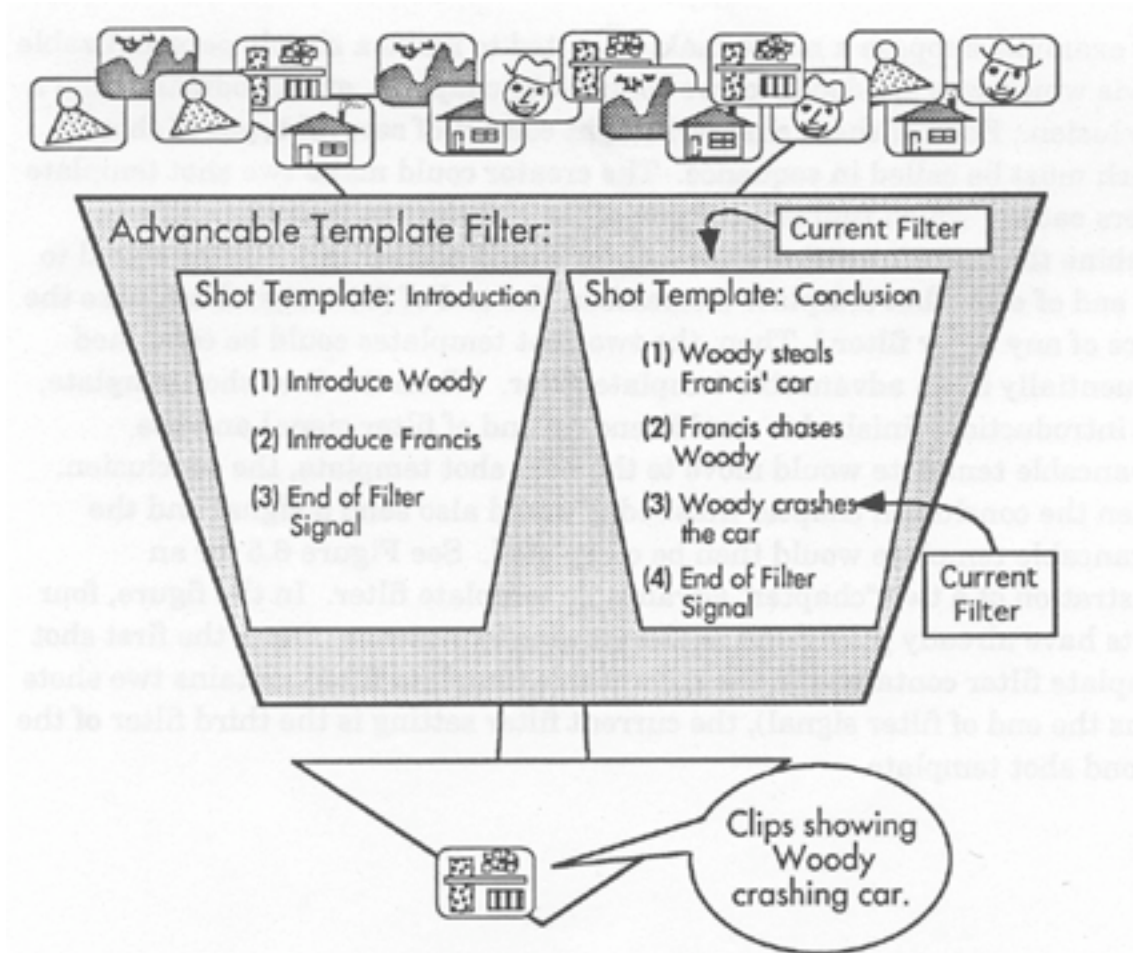


Figure 11: A simple advancable template filter.

Some details about advancable template filters: First, more interesting behaviors can be constructed with the advancable template by embedding end of filter signals inside of context dependent filters or interaction filters. This allows for movies of variable lengths. Second, if an end of filter signal has no advancable template filter "containing" it to catch the signal, then the signal is caught by the top level filter driver, the movie experience ends and the viewer is notified that the experience is over. Third, advancable templates have a repeat flag just like shot template filters, which allow the construction of simple looping structures.

3.1.6. Interaction Filters

Interaction filters provide simple on-screen interfaces to the viewer which allow him or her to easily change the behavior of specific filters, thus affecting the playout. Currently only one simple type of interaction filter has been implemented, but more complicated

interaction filters could be built which were more suited to particular types of personalizable movies (e.g. sliders for ranges of values, graphical dialog items which related to the theme of the personalizable movie).

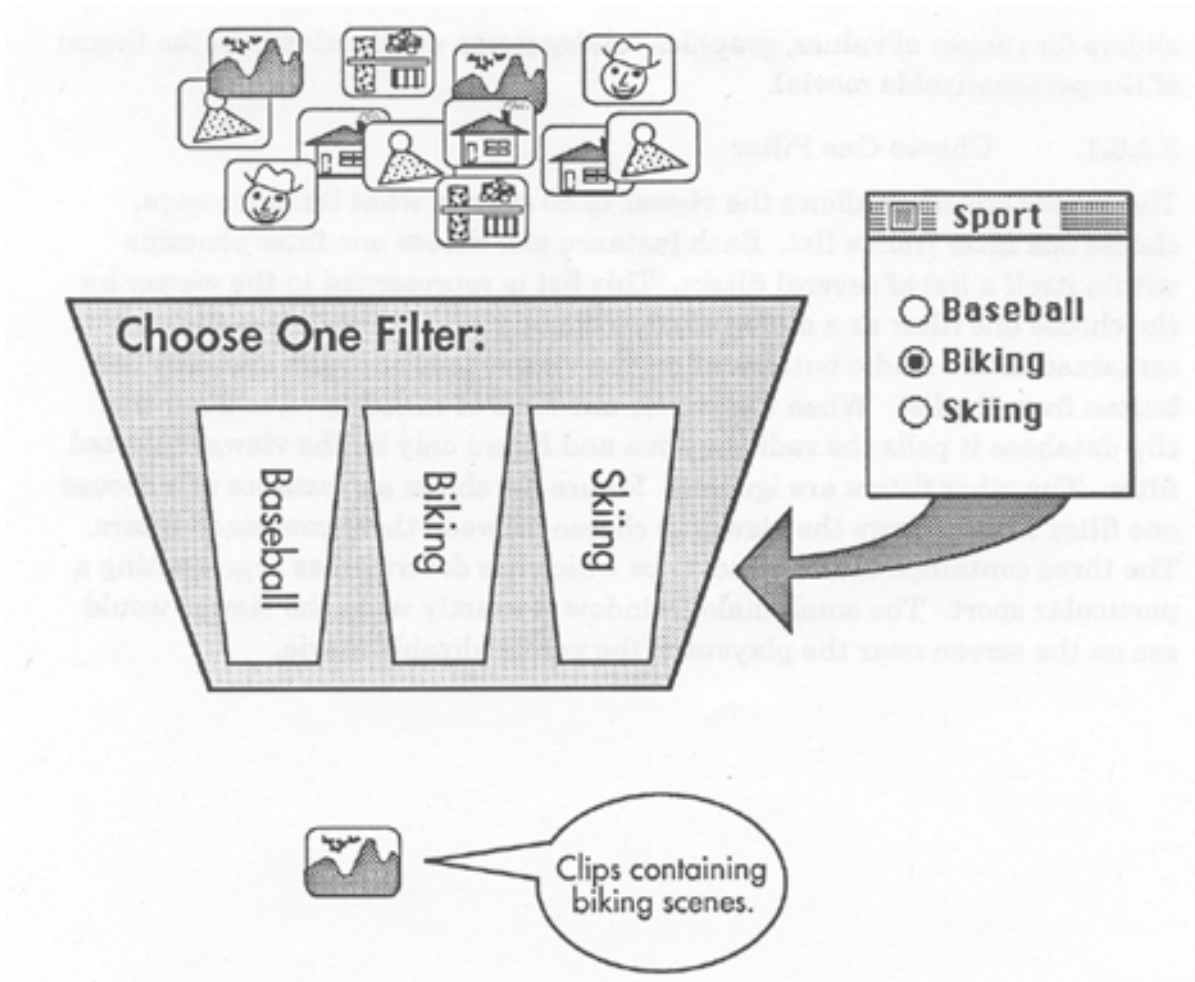


Figure 12: An instance of the choose one filter type.

The canonical filter family includes filters which could be created with the other filters types, but are so commonly used that they are included as part of the provided filter types.

3.1.7.1. Continuity Filter

The continuity filter is a specialized type of context dependent filter which keeps track of descriptive "variables" as the movie plays out. It allows the author to specify one descriptive slot which is to be matched or not matched from clip to clip as it plays out. Options include match slot value to last clip's slot value, match slot value to next to last clip's slot value, match slot value to first clip's slot value. The inverse of each of this is also available (e.g. do not match slot value to last clip's slot value). This provides "continuity" for the movie at the most basic level. For example, a continuity filter could

be used to make sure all the clips in a movie featured the same character. More complicated continuity structures can be built with the eval filter.

3.1.7.2. Suppress Duplicates Filter

The suppress duplicates filter removes clips which have already been presented to the viewer (i.e. appear in the clip transcript). This is often a desired behavior when several similar types of clips might be used in repeated sequences.

3.1.8. Filter Combinations

Most of the power of using FilterGirl as a language for dynamic content selection comes from combining the many different types of filters in interesting ways. Some of the simplest combinations are easy extensions of the filter types mentioned above. For example, the multi-filter can be used to combine two or more description filters and create a more closely specified selection of the database (e.g. clips in which the characters Nicole and Woody appear on the beach flying a kite).

Another interesting use of the multi-filter is to combine the different types of template filters. For example, we could use a multi-filter to combine two shot template filters to create a dialog between two characters. The first template filter would define the structure of the dialog (e.g. a question then a story then a reaction and so on) while the second template would define the cutting back and forth between the two characters (e.g. first show a clip with the character Dave then the character Thomas). The multi-filter would take the intersection of these templates at each point resulting in a question from Dave then a story from Thomas and so on. In this way we have abstracted away the structure of the dialog from the character specification. Now, assuming our clip database is rich enough, we can change the dialog structure without modifying the way the characters appear and we can modify the characters that appear without changing the nature of the dialog (e.g. an argument between the two characters or a normal dialog between new characters).

Many other interesting types of combinations can be imagined such as advancable template filters which advance based on signals from interaction filters, advancable template filters which advance based on signals from rule filters, eval filters which create time template filters (to give the viewer control over playout time) or time templates combined with repeated shot templates.

3.2. Creating and Editing Filters

Instances of all of the filter types can be created and edited interactively with the FilterGirl application. The command center of FilterGirl is the "filter control window". The filter control window contains a list of all currently defined filter instances along with a list of currently running filters and controls for editing and manipulating the filters. Figure 13 shows a snapshot of the filter control window during a typical session with FilterGirl. The list in the upper left corner includes all currently defined filter instances

while the list at the bottom includes all of the currently running filters. Remember, since filters can contain other filters some of the filters in the top list not listed in the running list may be called during a session if they are contained within another running filter. The buttons along the right hand side allow the author, among other things, to start a playout with the currently running filters, create new filters and open editing windows for filters.

To create a new filter instance, the moviemaker selects a filter type from the "new filter" pull down menu and a new instance is created. An editing window is opened for the new instance that allows the author to change the internal parameters which vary the behavior of the filter. Opening edit windows for existing filter instances takes place in a similar way. The filter instance is selected in the top list and the "open filter script" button opens the editing window. Double clicking on a name in any filter list (including lists of filters included within other filters) has the same effect.

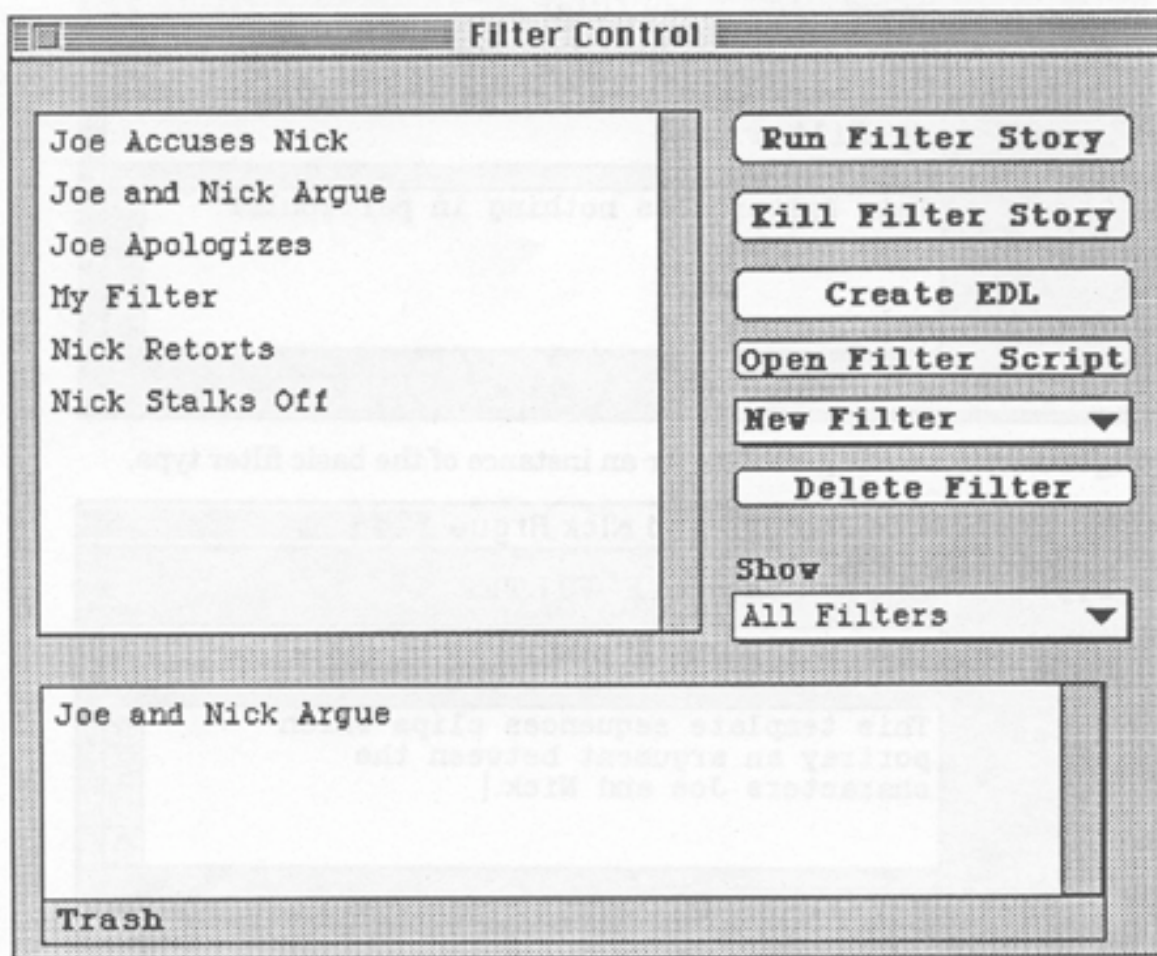
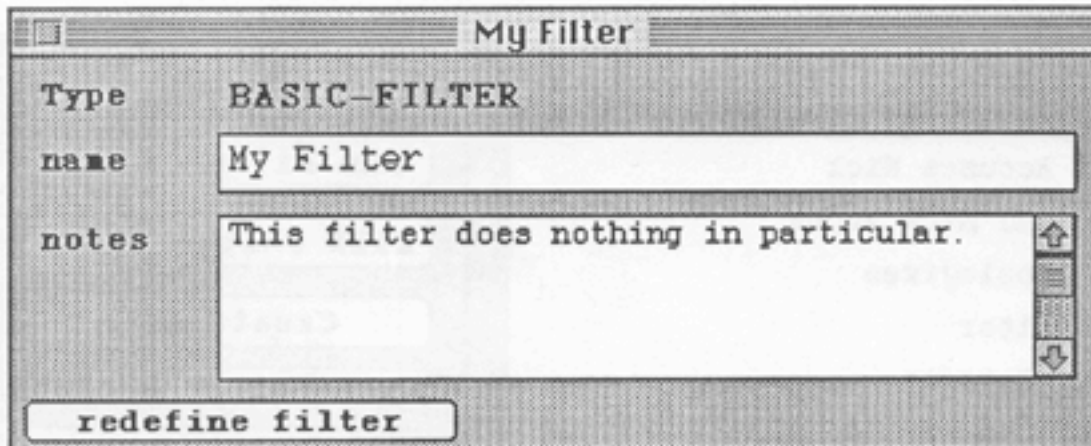


Figure 13: The filter control window during a typical FilterGirl session.

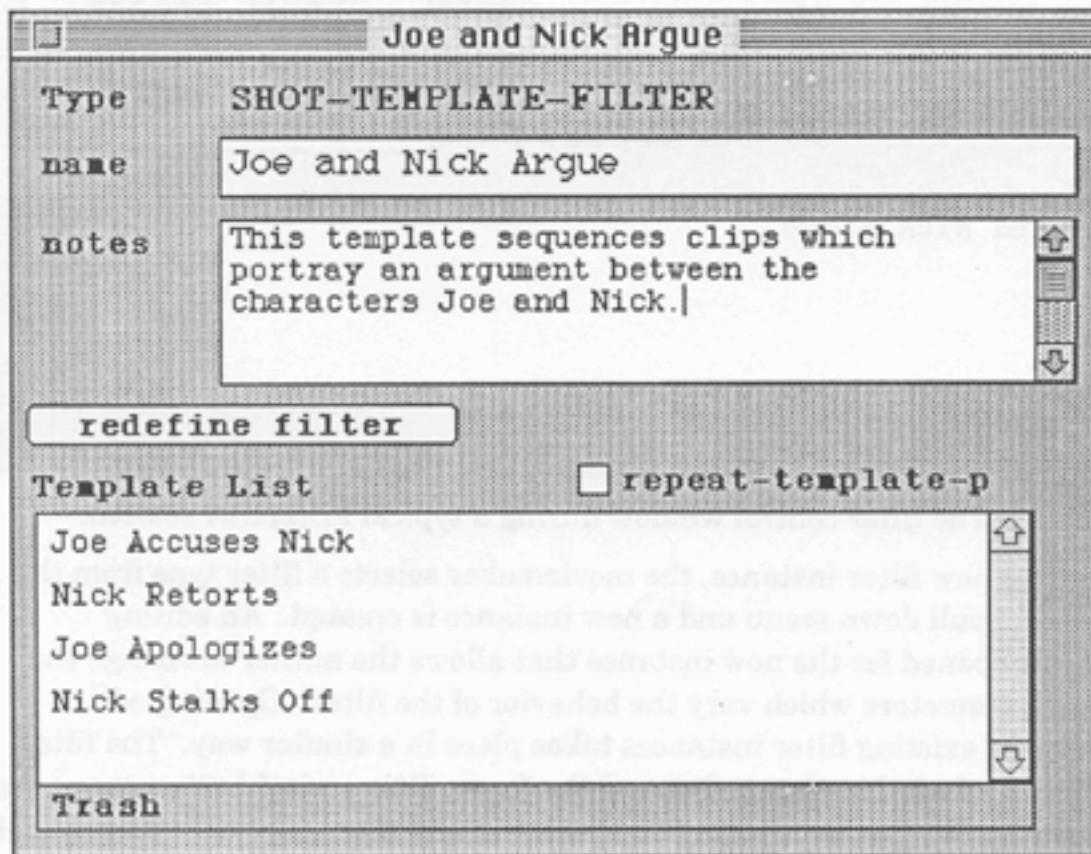
Figure 14 shows a prototypical filter editing window. It is an editing window for the basic filter type and includes features common to all filter editors. All editing windows display the filter type, the filter name and comments about the filter. They also include a button

which allows the user to redefine the attributes at any point. Redefining the filter records all of the currently displayed (perhaps new) information as the parameters of this filter instance.



The image shows a window titled "My Filter". Inside, there are three fields: "Type" with the value "BASIC-FILTER", "name" with the value "My Filter", and "notes" with the text "This filter does nothing in particular." To the right of the notes field are three small icons: an up arrow, a list icon, and a down arrow. At the bottom of the window is a button labeled "redefine filter".

Figure 6.7 An editing window for an instance of the basic filter type.



The image shows a window titled "Joe and Nick Argue". Inside, there are three fields: "Type" with the value "SHOT-TEMPLATE-FILTER", "name" with the value "Joe and Nick Argue", and "notes" with the text "This template sequences clips which portray an argument between the characters Joe and Nick." To the right of the notes field are three small icons: an up arrow, a list icon, and a down arrow. Below the notes field is a button labeled "redefine filter". Below the button is a section titled "Template List" with a checkbox labeled "repeat-template-p" to its right. The "Template List" contains a list of four items: "Joe Accuses Nick", "Nick Retorts", "Joe Apologizes", and "Nick Stalks Off". To the right of the list are three small icons: an up arrow, a list icon, and a down arrow. At the bottom of the window is a button labeled "Trash".

Figure 14: An editing window for an instance of the shot template filter type.

Edit windows for more complicated filters add more parameters below the common parameters. Figure 14 shows an edit window for a typical instance of the shot template filter. In addition to the filter type, instance name and comments there are two more parameters which correspond to the parameters described in the section on shot template filters above. First, the template list contains an ordered list of all of the filters within this shot template. Filters can be added by dragging and dropping from any other filter list (including the filter control window). Filters can be removed from this list by dragging to the "trash" icon at the bottom. Second, the repeat-template-p checkbox is a flag which tells the shot template filter to repeat when it reaches the end (rather than sending an end of filter signal). Edit windows for other filter types work in similar ways by adding interfaces for the various parameters below the standard information.

Filter sets can be saved out through the menu bar. By selecting "Save Filter Set" from the main menu, the author is prompted for a file name to which all currently defined filters are to be saved. Similarly, a filter set can be loaded through the same menu.

3.3. Running a Filter Story

FilterGirl provides facilities for previewing filter based story structures while filters are being built. This requires not only that a set of filters be defined in FilterGirl, but also that a clip database created in LogBoy be loaded. Loading a clip database is accomplished through FilterGirl's main menu. Previewing takes place by pushing the "run filter story" button on the filter control window. This opens a playout window which displays the movie as a viewer would see it. The playout window includes buttons to pause, resume and skip ahead one clip in the current playout.

Previewing the movie is accomplished by applying in sequence the filters in the running filters list to the loaded database. Each application of the filters results in some non-empty subset of clips. One is chosen arbitrarily and presented to the viewer. When FilterGirl has finished presenting the clip, the filtering/presentation process is repeated. This continues until an end of filter signal or an end of movie signal is passed out of one of the filters in the running filters list. Either of these signals will end the current playout.

There is one caveat to the layered filtering process explained in the previous paragraph. The empty set rule (described at the end of Chapter 3) is in effect during the filtering process. This means that the filters in the running filter list are applied in top to bottom order. If any of the filters returns an empty set of clips then FilterGirl stops the filtering process and returns (as its ultimate output) the input to the filter which returned the empty set. This provides a simple way to assure that FilterGirl will never come up empty handed when trying to select a clip to present to the viewer. This rule also means that the filmmaker must prioritize the running filters. In this prioritization scheme, the most important filters (e.g. basic story structure) come first in the list while the less important filters (e.g. user interaction, stylistic structures) come later in the list.

3.3.1. Debugging Output

Used by itself, the playout window simulates the viewer's experience closely, but provides a poor representation of the inner workings of the layered filtering process. To help provide a clearer view of how the filters manipulate the database FilterGirl has a debugging output which presents textual information regarding the input and output of specific filters in the running filter set.

The debugging output feature of FilterGirl can be turned on via the main menu. In the debugging output the filter layers are listed down the left-hand side of the text window while a histogram of the number of clips presented as input to each filter layer is printed on the right-hand side of the window. The two streams of information are interleaved so that the filter layers are synchronized with their input and output on the histogram (i.e. the number of clips fed into the filter layer is printed on the line directly above the filter layer's line and the number of clips output from the filter layer is printed on the line directly below). Additionally, each filter layer name is indented to represent how deep it is within the filter hierarchy. Unindented filter names are at the top level (i.e. the running filters list in the filter control window) while each successively indented filter name is contained within the previous filter name. This interleaved filter and clip histogram output is repeated for each clip selection. When the empty set rule is called into effect (i.e. when one of the filter layers returns an empty set of clips) a warning is printed in the debugging output window.

Figure 15 shows a typical FilterGirl debugging output for two clip selections. There are two top level filters in this scenario. The first is a shot template filter named "Joe and Nick Argue" while the second is a suppress duplicates filter named "Suppress Duplicates". In the first selection the filter "Joe Accuses Nick" has been selected from the shot template filter list. It is a multi-filter which contains three description filters which each successively narrow the selection. The suppress duplicates filter narrows the selection to a single clip which is then presented to the viewer. The second clip selection is very similar, except now the next filter in the shot template filter list has been selected and it contains two description filters.

The debugging output provides a good picture not only of how each filter affects the clip selection process, but also exactly which filters are caned upon during each stage of the movie experience. This provides an interactive previewing and debugging environment for personalizable movies.

```

*****
>> Joe and Nick Argue <<
*****
>> Joe Accuses Nick <<
*****
>> Speaker = Joe <<
*****
>> Utterance Type = Accusation <<
*****
>> Listener = Nick <<
**
>> Suppress Duplicates <<
"

Selected Clip: "Digital Film Clip 23"

*****
>> Joe and Nick Argue <<
*****
>> Nick Retorts <<
*****
>> Speaker = Nick <<
*****
>> Utterance Type = Rebuttal <<
*****
>> Suppress Duplicates <<
***

Selected Clip: "Digital Film Clip 08"

```

Figure 15: Part of a typical debugging output.

3.3.2. Creating an "EDL"

FilterGirl has a second previewing option which can be used to create multiple linear non-interactive edits from a personalizable filter structure. This feature, rather than actually playing each digital movie file, creates a list of the pathnames of each digital movie file. This list can then be used to create a linear digital movie which represents a specific playout of the personalizable movie. The list of pathnames serves a similar function to creating an edit decision list (or EDL) in a traditional video editing suite. Conceivably the pathname based EDL created by FilterGirl could be used as a template for creating higher quality linear analog edits of the same material if a lookup table was created to translate between digital movie pathnames and time code numbers on a physical video tape.

4. System Implementation

This chapter briefly describes the implementation of LogBoy and FilterGirl and the issues involved. There are four sections pertaining to the four major software subsystems: video, database, filters, interface.

4.1. Video

The LogBoy and FilterGirl applications currently use Apple's QuickTime software as a disk-based digital video standard. Disk based digital video storage provides the random access video capabilities essential to computational movie generation. Digital video has several benefits in addition to other random access formats (e.g. videodisc). First, images from the video source material can be easily sampled and manipulated within the screen space. One example of this is LogBoy's video clip icons which incorporate a still frame. Second, QuickTime is a read/ write format. This means that source material can be added to the database quickly and easily. Read/ write formats allow a more flexible creative process as the movie maker is not committed to a particular set of source material at any particular point in the production process. This also opens the door on serialized movies to which the creator is continually adding content. Finally, QuickTime digital video can be edited on the desktop using a Macintosh computer. Software packages allow the creator to digitize, compress, edit and present footage all on the same piece of hardware.

4.2. Database

The video clip database holds the locations of the digital video clips along with the descriptive slots and values attached to each clip. The LogBoy application must be able to create, present, edit, load and save this database structure while FilterGirl must be able to load and access the descriptors. Ken Haase's Framer representation language was chosen for the database implementation because it supports all of these characteristics along with providing other sophisticated features which can be made use of in the future.

Framer's hierarchical structure maps easily onto LogBoy's extensible slot/value database architecture. Framer files are implementation independent and can be read with libraries provided for several platforms in several computer languages. This flexibility allows moviemakers to experiment with data paths other than the intended LogBoy -> FilterGirl. For instance, a Framer database created in LogBoy could be loaded into some other story structure driver or databases could be created in another logger and loaded into FilterGirl. Framer also provides inferencing capabilities which can be taken advantage of by sophisticated story structure editors/ generators.

4.3. Filters

FilterGirl is designed around a hierarchical filter language which supports personalized video sequencing. This filter language is based on Macintosh Common Lisp's (MCL's) implementation of the Common Lisp Object System (CLOS). CLOS provides class inheritance which is useful in situations where there are many types of objects which have similar, but slightly different characteristics. For example, every filter type exhibits

a different filtering behavior, however each filter must implement the empty set rule (explained in Section 6.3). This is implemented with a basic filter type from which all other filter types inherit their basic behavior. Thus, the empty set rule is implemented once for the basic filter type and all other filter types make use of this implementation.

4.4. Interface

Both LogBoy and FilterGirl use a window based graphical user interface to allow the moviemaker to create and edit databases and story structures. Both interfaces are created in Macintosh Common Lisp's (MCL's) implementation of interface objects which is based in CLOS. Many of the interface objects in both LogBoy and FilterGirl make use of class inheritance to reduce the amount of duplicated code when many objects share behaviors. For example, all of the filter editing window types in FilterGirl have several common interface items. The window definitions are built with a class hierarchy such that there is a basic filter editing window class which implements all of the common characteristics and each different type of filter editing window inherits from this superclass. This is analogous to the inheritance scheme for the filters themselves. Each of these specific editing window types can specialize the canonical window to suit its own needs. Similarly, LogBoy has two types of interface items, clip icons and value areas, which share the behavior of dragability. This behavior is implemented with a "draggable-object" superclass from which they both inherit.