



All Theses and Dissertations

2011-05-27

Flexible Storylines

Ivan Dmitrievich Romashka
Brigham Young University - Provo

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>



Part of the [Computer Sciences Commons](#)

BYU ScholarsArchive Citation

Romashka, Ivan Dmitrievich, "Flexible Storylines" (2011). *All Theses and Dissertations*. 2731.
<https://scholarsarchive.byu.edu/etd/2731>

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

Flexible Storylines

Ivan Dmitrievich Romashka

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements of

Master of Science

Dan Olsen, Chair
Michael Jones
Charles Knutson

Department of Computer Science

Brigham Young University

August 2011

Copyright © 2011 Ivan Dmitrievich Romashka

All Rights Reserved

ABSTRACT

Flexible Storylines

Ivan Dmitrievich Romashka
Department of Computer Science, BYU
Master of Computer Science

A long-standing goal of computer-based entertainment is the creation of a story where a user is in control of portions of the storyline. These non-linear stories give a user an opportunity to adapt the story to his or her interests, schedule and needs. The Internet has made non-linear video a reality. Different approaches have been taken to create and play non-linear video stories. They suffer from lack of simplicity, smoothness, and television-like experience in story creation and presentation. Flexible Storylines provide a way to easily create and present non-linear video stories. The creation of these stories is done using a time-line based editor that mimics the way video stories are composed by film makers. The viewing experience of flexible stories is very similar to viewing a normal video with an introduction of the choice to see more or less of the current topic. This provides a highly variable experience with a simple, smooth and non-intrusive form of user interaction. We also provide a mechanism that lets a story flow smoothly despite the introduction of user interaction.

Keywords: flexible storyline, interactive television, non-linear story, Hypervideo, video editing, Silverlight, seamless playback.

Table of Contents

Chapter 1 - Introduction.....	1
Chapter 2 - Prior Work	6
Branching structures	7
Hypervideo Systems	9
Flow and structure preserving systems.....	12
Chapter 3 - Overall Approach.....	18
Flexible Storylines Structure.....	20
Origins of Tunguska Explosion Example.....	23
American Idol Example.....	24
Challenges	25
Creator tools	26
Seamless playback.....	26
Viewing experience	27
Chapter 4 - Creator tools.....	29
Production video editing tools.....	31
Flexible storylines creation	33
Video Editor	33
Flexible Creator	34
Validation	41

Preproduction validation.....	41
Postproduction validation.....	43
Conclusion.....	53
Chapter 5 - Seamless playback	55
Flexible Story Data Structures Overview.....	59
Playing and Preloading.....	65
Play and Preload Implementations	67
Preloading Workflow Example	71
Media Players' Resource Pool	73
Conclusion.....	82
Chapter 6 - Viewing experience	84
Flexible Storylines structure	85
User interface	87
Validation	91
Interaction statistics	92
User Feedback	95
Conclusion.....	100
Chapter 7 - Conclusion	102
Bibliography	107

Chapter 1 - Introduction

In order to produce a good story, story tellers, such as writers, film-makers, and theme park designers, focus on controlling the pace and sequence of story elements. The depth of each event or topic covered stays unchanged once the story is produced. Therefore, the goal of the story-tellers is to create a story that will appeal to the largest audience possible. Although, controlled pace and sequence of story may engage with a large amount of consumers, it still uses “one size fits all” principle. There is no way to adapt the story to individuals.

In comparison, storytellers of old used a different approach. They too narrated stories to audiences with diverse backgrounds. While narrating, storytellers watched the audience’s reaction and level of interest and adapted their story accordingly. The content and flow of the story was conditional upon audience’s background and level of interest. This story-telling approach added to “tension, entertainment level, and mental participation” of that particular audience (Vorderer 2003). Bringing this idea into the space of video stories, viewers have a great variety of interests, needs, and schedules. Allowing viewers to be in control of portions of the storyline will add to their “mental participation” and “entertainment level”. We want to create simple web-based interactive video mechanism allowing adaptation of video to interests of individual viewers. We call such video stories Flexible Storylines.

Flexible Storylines will be of great significance to two groups of users: story viewers and story creators. As stated by Hand and Varan (2009), the ongoing process of television digitization is constantly increasing interactive capabilities. This idea is continued by Murray (1997), who explains that increased interactive capabilities will have a greater appeal to the audience. First and foremost, Flexible Storylines will benefit viewers by deepening their immersion into the

viewing process. A sense of agency will be added to the viewing process (Murray, 1997). As viewers are able to adapt the depth of the topics covered, they will feel that their individual interests are targeted and satisfied. On the other hand, story creators are interested in offering a new cutting edge experience to their viewers. Story creators will benefit from a tool that introduces flexibility in story presentation and is a familiar extension of production video-editing tools. Finally, Flexible Storylines will extend the viewing time of the particular video content. Encouraging viewers to watch more of a current story will lead to a higher income for the television company.

In order to address the needs of both story creators and story viewers, we have divided the problem into two parts. The first part is creating an easy-to-use tool to assemble the flexible storyline. Second part is, creating a tool for a smooth viewing experience of flexible storylines.

The most important criteria of story creators is how intuitive and easy to use is our system. The flexible storyline creation tool needs to be a familiar extension of the video editing tools already used in production. First, the flexible storylines creation tool should be time-line based. Second, it should allow film-makers to drag and drop pieces of video content to assemble the story.

The needs of story viewers will be satisfied by fulfilling the following requirements. First - adaptability of the story, which will allow viewers to be in control of portions of the story. Second need is smooth viewing experience. The viewing experience is smooth when the playback is seamless, without stops or glitches, even at the interactive transitions. Third, although flexibility is introduced, the flow and structure of the story needs to be preserved. Fourth is perceived safety, meaning viewers need to feel safe making a choice that will alter the flow of the story. This is achieved by letting a viewer undo their story-altering choice.

We have identified the following three examples that will illustrate the applicability of Flexible Storylines to three different genres of videos: fiction movie, documentary, and reality show.

A regular complaint of avid book readers is that movies that were based on the book leave out important portions. *Harry Potter and the Goblet of Fire* movie depicts the Tri-wizard tournament. The first two tasks in the tournament are covered in detail. As for the final task, going through the maze, the book includes different obstacles such as the Blast-ended Skrewt, Boggard, the Golden Mist, and a Sphinx. The movie while showing Harry going through the maze omits each obstacle listed above. Using Flexible Storylines we would be able to add a longer version of the third task for viewers to choose as an option.

Discovery or National Geographic documentaries are often scientific in nature. Certain topics covered in them can be explained in either a simple or a complex way. Simple explanations can target a general viewer, who is not interested in scientific support and proof of the given facts. Complex versions can target informed viewers, interested in scientific explanations and supporting claims made in the documentary. As an example documentary we'll consider the origins of the Tunguska explosion. It could explore different theories that led to the explosion. The story could start by explaining the meteorite theory, which will lead to a comet theory. The main line of the story will explain comet theory in its simple form. The longer version of the comet theory will also be available for choice. It will cover such facts as the chemical composition of the rocks found at explosion site, physics behind the scattering of light caused by a comets tail etc.

Finally, reality shows can also benefit from Flexible Storylines. Reality shows, as any produced video, has to omit certain portions of the video that could be of interest to some viewers. In

American Idol singers perform several songs for the judges, while only one song is aired. As viewers see a particular singer perform his song, Flexible Storylines might add an option to continue watching the omitted songs by the same contestant.

Current research on interactive video stories has focused on creating desktop applications instead of internet based applications. They developed a linear branching structure that allowed viewers interactively take branches off the main story line, and used bridging video stories to bring viewers back. However, they did not address how does the viewer gets notified of the availability of choice, and when can the branch be taken. They have essentially treated the interactive video story as a graph assembly and traversal problem, instead of a story telling problem. Although, a graph-based approach does give you freedom and flexibility, it does not create a smooth story. The present systems are complicated to create and confusing for viewers to watch.

Our approach is to add flexibility to the story in the form of *Body* clips and *Diversions*. A body clip is the main route through the story (see Figure 1). A diversion is an alternative and usually longer piece of video that viewer may want to see. One can picture a diversion as a bend in the story stream. We provide a standard template mechanism that allows identification of upcoming choice, an interactive way to follow and exit out of a choice without breaking the flow of the story.

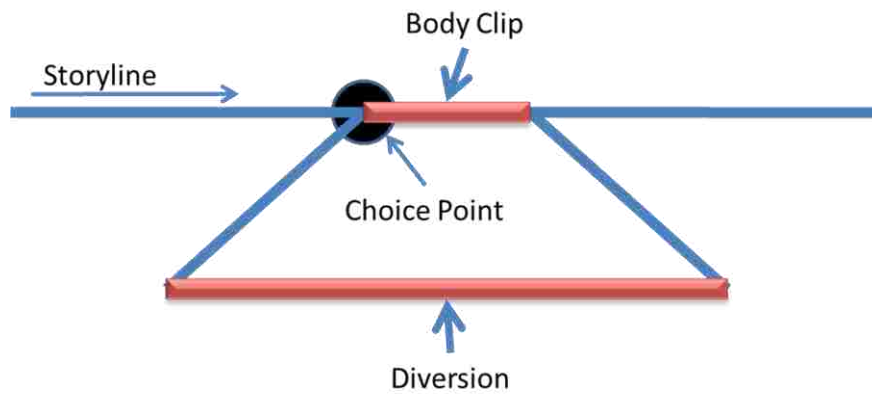


Figure 1
Basic structure of Flexible Story Line

Users will be presented with choices throughout the story. A user may choose to stay with the main storyline or take a diversion. If a user chooses to stay with the main storyline, the body clip will play and the user will be smoothly taken to the next concept. If a diversion is taken, the body clip is omitted and upon completion of a diversion the user is smoothly brought back into the main story. In either case we add the adaptability to the story, while preserving its structure, smoothness and continuity. Finally both creation and presentation of Flexible Storylines are internet based applications, allowing online user access.

Chapter 2 - Prior Work

There are three main categories of prior systems that are distinguished by the structure used to represent non-linear video systems. Branching structures presented various narrative templates for non-linear videos. Each template occasionally allowed viewers to control further development of the story. Hypervideo systems did not follow a structured template approach, but allowed creation of free form stories. The story consisted of video pieces that were interconnected in a graph-like manner. A viewer was then able to “browse” a video story selecting paths of interest through interconnected graph. The last category of prior systems took the branching structures of interactive narrative and extended them to allow preservation of story’s flow and structure.

There are two main groups of people affected by non-linear video systems: story creators and story viewers. We will first examine how well prior systems satisfy the set of viewer needs explained in the introduction, namely:

- adaptability of the story
- smooth viewing experience
- flow and structure preservation
- perceived safety.

We will then examine prior systems from a story creator’s point of view. The main criteria of evaluation will be a level of similarity of authoring tools presented to video editing tools that are currently used in production. In addition, we will evaluate the level of complexity of story structure proposed by prior systems.

Branching structures

Branching structures presented various narrative structures that at times offered choices for viewers. Once a viewer would reach a decision point the story would pause, and he or she had an option to control further development of a story. For example, Figure 2 shows a model that allows viewer to choose the destination of character's journey. Subsequently each destination will alter the story development.



Figure 2
Example of an interactive narrative model

Adaptability of video stories lies in the core of these systems. Garrand (1997), Samsel and Wimberly (1998), and Meadows (2003) have outlined various branching structures. Figure 2 shows a typical example of a branching structure where a choice made by a viewer determines how the rest of the story will play out. The hierarchical branching structure (Garrand 1997), also known as the extended branching structure (Samsel and Wimberly 1998) makes it possible for the story to develop in completely different directions (see Figure 3). Although adding multiple branches adds a great variety in story development, it introduces an “unmanageable number of outcomes” for the story (Samsel, Wimberly 1998). For viewers, the flow and structure of the story does not get preserved, since there are so many endings to the story. For creators, this

structure has a high level of complexity. Story creation process becomes overly complicated and cumbersome, again due to the explosion of possible paths and endings of the story.

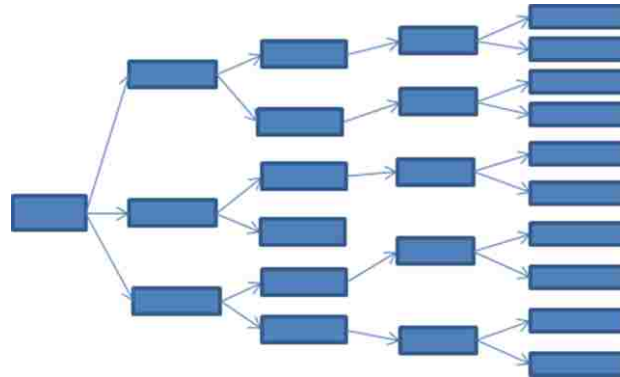


Figure 3
Exponential growth of story ending with branching

The parallel branching structure (Garrand 1997) shown in Figure 4 had several versions of the same story laid out in parallel. The idea was for the viewer to be able to move up and down between the versions of the story. The advantage of the parallel branching structure was that it did not have a problem of branching explosion. Greater adaptability was provided by allowing viewers to jump up or down to a parallel branch of story development. However, this increase in adaptability came with a cost of increased complexity. Transitioning back and forth between parallel versions of a story made it difficult to keep a mental model of a story's flow and structure. In addition, parallel versions of a story could be different enough to not connect in a smooth manner, hindering viewing experience. Finally, neither branching structure discussed a way for a viewer to undo a choice taken, by exiting it. It is common knowledge in User Interface Design that users are more inclined to experiment with software that provides an undo mechanism. Absence of such a mechanism hinders viewers' perceived safety, meaning viewers become more reluctant to explore branches in the story line. Adding a way to undo the choice to take a branch, and be brought back smoothly into the story, will be of great value to viewers.

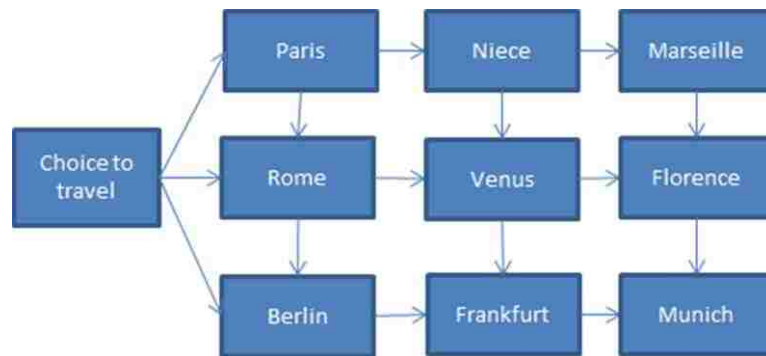


Figure 4
Parallel branching structure

The complexity of this structure also made story creation process convoluted. It would be cumbersome for story creators to create parallel paths and think through ways to connect between these paths.

Hypervideo Systems

Hypervideo, also known as hyperlinked video, is a video story that contains links to other media elements. These links are user clickable, and allow viewers to “browse” a video story.

Great adaptability was provided by various hypervideo systems such as, *Electric Charles*, (BrØndmo et al. 1991), the *Interactive Kon-Tiki Museum* (LiestØl et al. 1993), *HyperCafe* (Sawhney et al. 1997), *Hvet* (Tielllet 2010) and *The Interactive Village* (Ursu 2008). Figure 5 shows a typical example of a hypervideo system. At the top of Figure 5 there is the main story line represented by four video clips connected by a green line. In addition there are links from the clips in the main story line that allow viewers to browse the web of connected videos. One of the browsing paths connects to the subordinate story line (four video clips connected by a green line at the bottom of Figure 5). As a user is browsing hypervideo story he or she can become buried in the structure of the web of links and become disoriented.

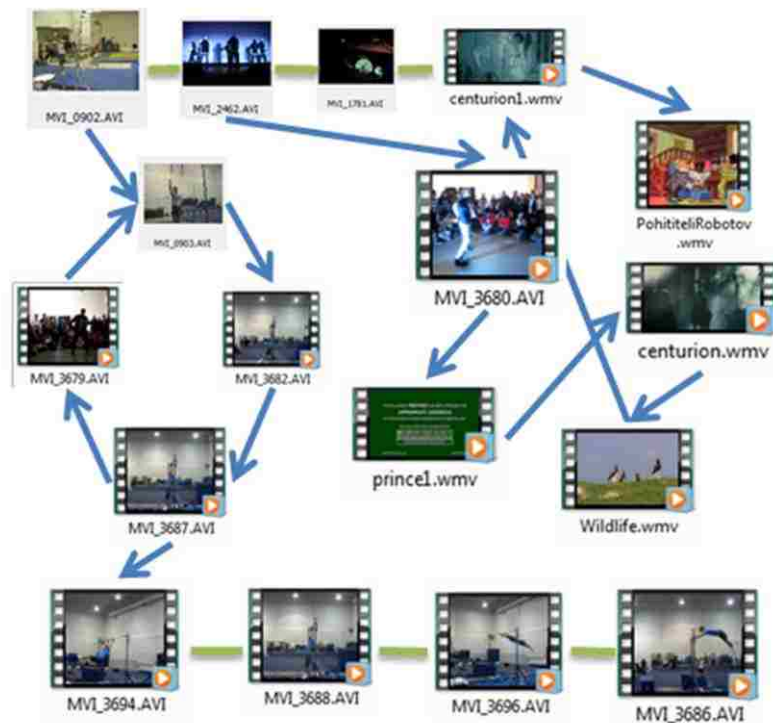


Figure 5
Hypervideo system example

Electric Charles (BrØndmo et al. 1991) and *Interactive Kon-Tiki Museum* (LiestØl et al. 1993) pioneered the development of hypervideo. *Electric Charles* was a hypermedia journal that included any form of media (sound, video, pictures, and text) as its content. *Electric Charles* provided support for links from any media chunk of information in the environment to any other media chunk. Links to media items were shown in correspondence to the topic currently presented. Later, the idea of video-to-video linking was continued in *Interactive Kon-Tiki Museum* (LiestØl et al. 1993). They used hypervideo to expose to and allow navigation through a large collection of filmed documentary materials stored in a museum. Again, as the main video played, linked videos, relevant to the current topic covered, were presented as small buttons below the main video’s playback window. Each button had a picture that showed the still image representing the story it covered.

Other hypervideo systems: *HyperCafe* (Sawhney et al. 1997), *Hvet* (Tiellet 2010) and *The Interactive Village* (Ursu 2008), also used video to video linking either to add flexibility to their documentary, fiction or educational video stories. The main difference between these systems was the interface that was used to present the links. For example *The Interactive Village* (Ursu 2008) in addition to showing thumbnail images for links during a playback, allowed viewers to navigate to a specific story using a map of the village. *Hvet* (Tiellet 2010) was the only hypervideo system that allowed “content access anytime anywhere” through a web-based interface.

HyperCafe (Sawhney et al. 1997) was the only hypervideo system that explained their story creation process. They used a tool called Hypervideo Engine. Story creators wrote scripts that were interpreted by Hypervideo engine to play a story. A script specified the time a piece of content (video, sound or text) was to be played as well as its spatial position on the screen (Figure 6). The author started by specifying the main video script where the narrative may be initiated, plus specific scripts representing content sequences as shown in Figure 6. Temporal links, hypertextual links and temporal videos (small clickable video playing link’s preview) provided a way for user to navigate through hypervideo.

```
on MainVideoScript
  showText "There is no simple way to say this," "bottom," "right"
  playVideo "Andy - Artificial Consciousness," 0, 0, 5, 1, 1
  removeText "bottom"

  showText "Panning," "top," "right"
  tempoVideo 3, "Pan - Shawn to Ian to Kelly," 3, 6, 0, "TempoScript_First"
  removeText "top"

  tempoVideo 5, "Ian Lips - Influencing Outcomes," 3, 6, 0, "TempoScript_Second"
  playVideo "Andy - Death is not an End," 0, 0, random(9), 1, 1
end
```

Figure 6
Hypervideo Engine script example (Sawhney et al. 1997)

The focus of these systems, as well as of most hypervideo systems, was to produce a video story that allowed navigation through a vast number of sub-stories. Links were provided to connect related sub-stories together. The video story became a connected graph, in which nodes were sub-stories and edges were links between them. This graph representation adds great adaptability to the structure of the story. However, it can disorient a viewer in the web of links. Feeling disoriented downgrades the viewing experience. Large amounts of interlinked sub-stories as well as a lack of formal template structure damages the flow and structure of the story. None of these systems provided a way for viewers to undo their story-altering choice. Finally, a common trend among all hypervideo systems is that video stories navigated to additional content immediately after a link was chosen by a viewer. This created a browser-like experience instead of a story flow. Immediate navigation to a connected video story could interrupt the currently playing portion, again hindering the smoothness of a viewing process.

From story creators' point of view, no story structure template is proposed by hypervideo. On the other hand a graph approach of interconnected sub-stories still leads to a high level of complexity in the story structure. Complex structure makes it difficult to plan and manage a story. Finally the authoring tool provided was script-based. Although specifying a video story through a script gives story creators a great freedom, it is inconsistent with the way video stories are currently edited and assembled, and is difficult to use.

Flow and structure preserving systems

Flow and structure preserving systems enabled story creators to add a degree of freedom to their story, while keeping the story's main flow and structure intact.

A model that provided adaptability while preserving the story's structure was linear structure with scene branching (Garrand 1997). It is compared to a highway with a few detours, which always route the travellers back to the main highway. Linear structure with scene branching gave viewers an opportunity to make few limited choices as to how certain scenes will play out but they would always be returned to the main story line. Figure 7 shows an example of a linear structure with scene branching for a portion of Cinderella's tale. A viewer would be able to make a choice whether the prince searches in the palace or the garden before going into the village. Similar structures were defined by Samsel and Wimberly (1998) as "conditional branching with bottlenecking", and by Meadows (2003) as "nodal plot structure".

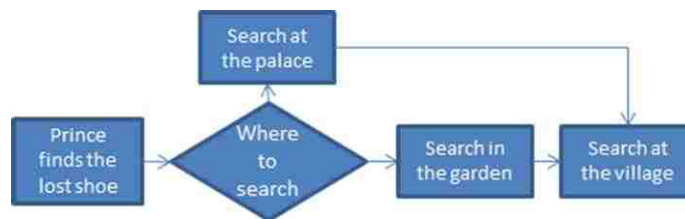


Figure 7
Linear structure with scene branching

For story viewers, adaptability of the story was provided, along with preserving the flow and structure of the story. The downside of these models was a lack of smoothness. During each decision point, the narrative stopped and continued only upon viewers' choice. Stopping a narrative impedes the smoothness of the viewing experience. In addition, viewer needs were not met in the area of perceived safety. Neither model discussed a way for a viewer to undo a choice taken, by exiting it.

For story creators, linear structure with scene branching decreased the level of complexity proposed by a branching model. A video story now had a manageable set of paths and one outcome.

Hyper-Hitchcock (Shipman et al. 2003) was the only hypertext system that added a plausible way to preserve the flow and structure. It allowed authors to specify a behavior associated with each linked video's completion. These behaviors defined what portion of the source storyline would resume playing upon abortion or completion of the link. In this manner links could be merged back into the main story. An example of linked video abortion and completion behavior is shown in Figure 8. While watching *Video 2* a viewer is notified of the link availability and decides to follow it. As the linked video (*Video A*) is playing a viewer decides to abort watching the linked video. In order to keep the flow of the story a viewer is brought back to the time in *Video 2* when the link was followed. On the other hand, if a viewer decides to watch linked videos until completion he is brought to *Video 3* to resume the story.

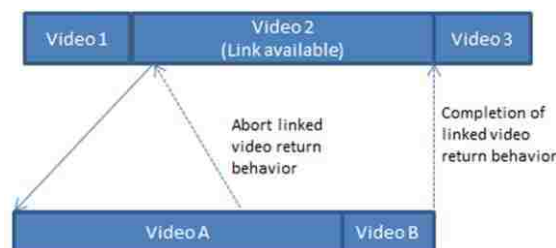


Figure 8
Hyper-Hitchcock linked video abortion/completion behavior

In addition, *Hyper-Hitchcock* (Shipman et al. 2003) provided an authoring tool to create their video stories. The user interface of the authoring tool consisted of a selection pane at the upper left, tree view at the upper right and an authoring workspace at the bottom (see Figure 9). Clips in the selection pane were grouped into piles according to the time of the recording. Tree View

showed an alternative representation of the selection pane. Selection pane allowed dragging clips to the authoring workspace. Clips could then be assembled into a hypervideo web of linked objects. Figure 9 shows a timeline below authoring workspace, which was used for the sole purpose of displaying in and out points of the selected clips.

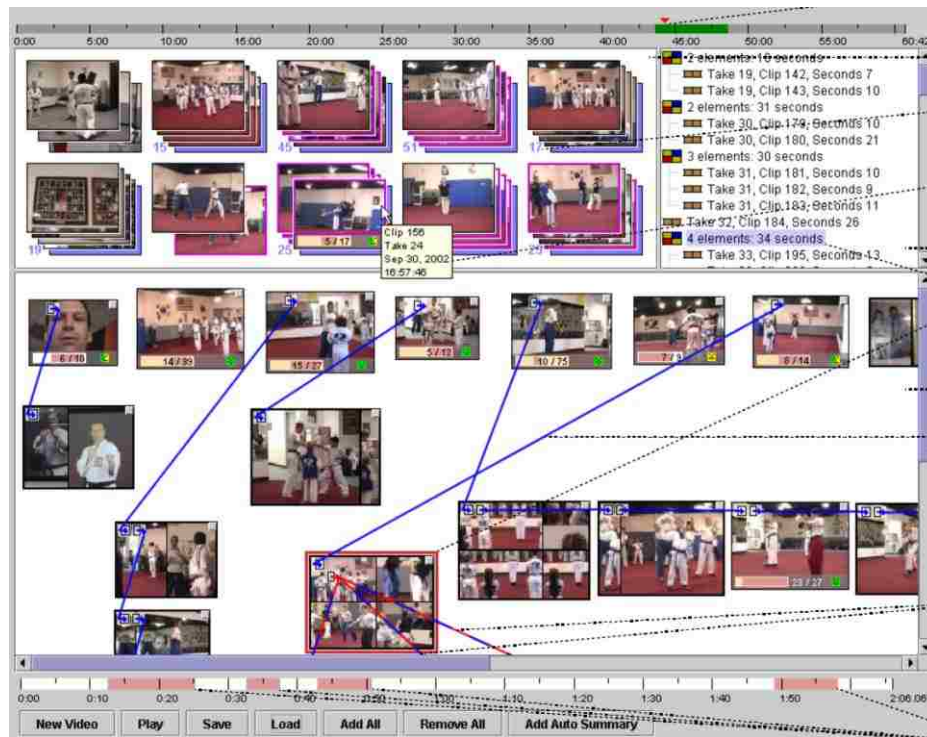


Figure 9
Hyper-Hitchcock Authoring Tool (Shipman et al. 2003)

Hyper-Hitchcock (Shipman et al. 2003) provided great adaptability of video stories for viewers, since they were able to navigate to different sub-stories. In addition, *Hyper-Hitchcock* system made it plausible to preserve the flow and structure through specifying link's abortion and completion behaviors. However, as other hypervideo systems discussed in the previous section, *Hyper-Hitchcock* navigated to links immediately after they were chosen by viewers. Interruption of a currently playing story downgrades smoothness of a viewing process. Nevertheless *Hyper-*

Hitchcock did provide a way to specify link abortion behavior. This provided viewers with a level of perceived safety, since they now could exit from a currently playing link to undo their choice.

For story creators', *Hyper-Hitchcock* (Shipman et al. 2003) had a high level of story structure complexity, since it used a graph approach of interconnected sub-stories. The authoring interface had a degree of similarity to video editing tools in the process of story assembly. Story creators could drag and drop clips into a content pane to assemble their stories. The timeline, provided by the authoring tool, was only used to show in and out points of the selected clips. This is inconsistent with other video editing tools that use a timeline as a chronological representation of the whole story. Overall, *Hyper-Hitchcock's* authoring tool suffered from complexity and lack of similarity to conventional video editing tools.

Flow and structure of the story along with smooth viewing experience was more preserved by the “*Yo-yo*” model (Hand and Varan, 2007), than by hypervideo systems discussed above. Hand and Varan's implementation used the model of linear structure with branching (Garrand 1997) and further extended it. Their main focus was to define a story structure and keep it intact while allowing viewer interaction. They explained that in order to have a successful interactive television experience, producers must encase interactivity within a strong narrative structure. In “*Yo-yo*” model, each branch of interactive choice ended with a bridging video clips.

Figure 10 illustrates the idea of “*Yo-yo*” model for the story of Cinderella. Choices that a viewer takes, a search in the palace or a search in the garden, are followed by a corresponding bridging clip. In the bridging clips, different people suggest to prince that he should look for Cinderella in

the village. Bridging video clips ensure that the audience is smoothly brought back to the main storyline.



Figure 10
Yo-Yo model example

“Yo-yo” model (Hand and Varan, 2007) of branching and bridging also provided adaptability to the story. The flow and structure of the story was kept since a viewer was always brought back to the main storyline. Bridging clips transitioned viewers back to the main story, providing smooth return from a branch. However, Hand and Varan did not address how does the viewer gets notified of the availability of the choice. Also, it is unclear when does the choice take effect. Therefore, smoothness of taking a branch is uncertain. Finally, the absence of a way to undo a choice taken in “Yo-yo” model hindered perceived safety of viewers. From story creators’ point of view “Yo-yo” model presented a simple narrative template structure that simplified story planning process.

Chapter 3 - Overall Approach

Our approach is to add flexibility to the story in the form of *Body clips* and *Diversions*. A diversion is an alternative and usually longer piece of video that a viewer may want to see. One can picture a diversion as a bend in the story stream. A body clip is the main route through the story (see Figure 11). Users will be presented with choices throughout the story. A user may choose to stay with the main storyline or take a diversion. If a user chooses to stay with the main storyline, the body clip will play and the user will be smoothly taken to the next concept. If a diversion is taken, the body clip is omitted and upon completion of a diversion the user is smoothly brought back into the main story. In either the case we provide adaptability to the story, while preserving its structure, smoothness and continuity.

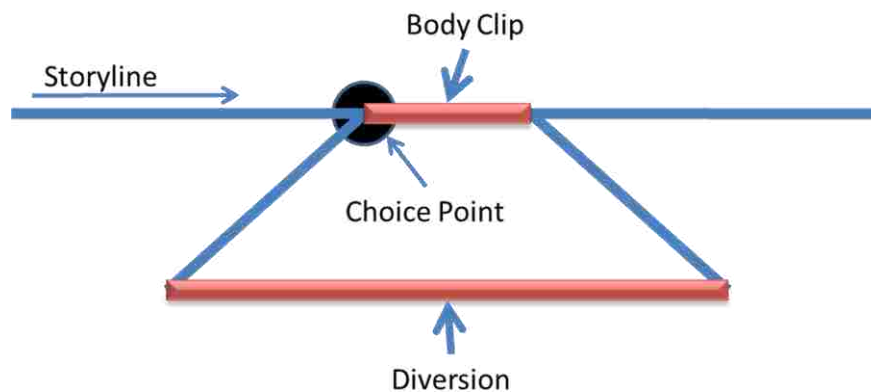


Figure 11
Basic structure of Flexible Storylines

This model adds flexibility but fails to satisfy some of the story viewers' requirements. Figure 11 currently shows a choice point right before the Body clip. Smoothness will be lost if we stop the story, and wait for the user to respond. Second, we would like to add ability for the viewer to exit out of a diversion. This will provide perceived safety, for viewers, since they will be less

reluctant to make a choice, knowing they can exit out of it. Exiting a diversion early could break the continuity of the story.

Figure 12 illustrates the point. A viewer is watching *Harry Potter and the Goblet of Fire* and decides to see an extend version of the third task. Half way through the Diversion, as Harry and Cedric are going through different obstacles, a viewer decides to exit. The story continues with the next clip after the Body Clip, where we learn that Harry and Cedric touched the Triwizard Cup together. Since a viewer did not finish watching the diversion, he does not know that Harry and Cedric became united and arrived at the Cup together.

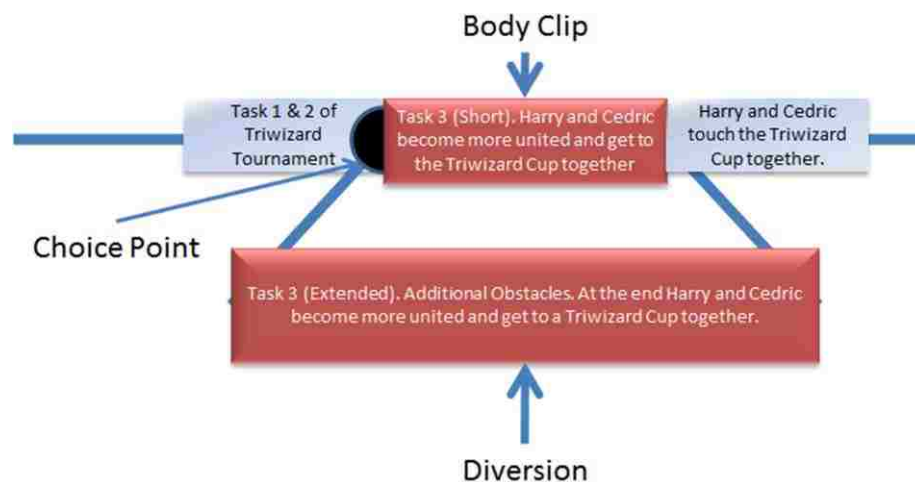


Figure 12
Example of basic structure of Flexible Storylines

Finally we would like to make it possible for the user to postpone a diversion and watch it later by itself. Returning to the last example of Harry Potter, if a viewer decides to watch an extended version of the third task later, it may appear out of context and be confusing. In order to solve these problems we have augmented the basic structure with Build-Up, Context and Merge clips.

Flexible Storylines Structure

Flexible Storyline's elements consist of Pre-story, Buildup, Body, Post-Story, Context, Diversion and Merge (see Figure 13). The *Pre-story* and the *Post-story* are the remainder of the storyline that are not part of this body clip/diversion structure. The *Buildup* will contain a video that informs the user of an available choice. This may be as simple as a banner across the bottom indicating an upcoming choice, or a special video introducing the diversion. The choice point is available throughout the buildup, making it unnecessary to stop the story for the user to make a choice. If a viewer does nothing, a Diversion is omitted and a Body clip starts playing after a Buildup.

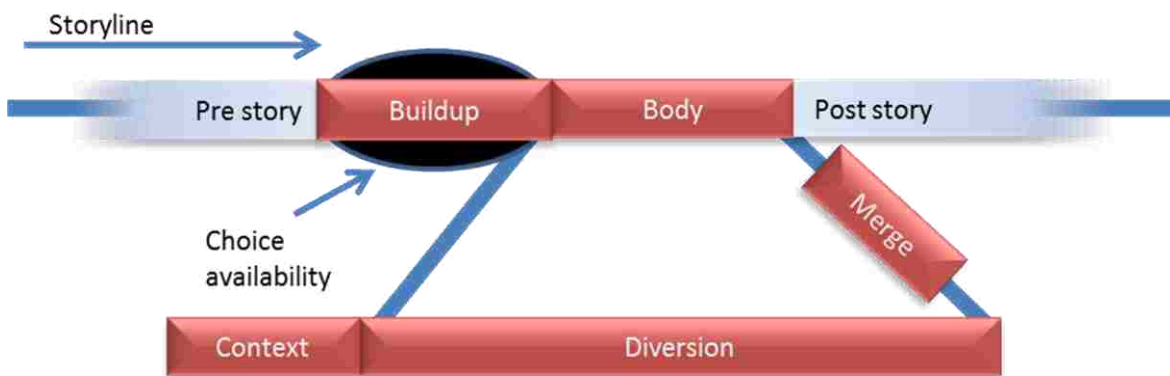


Figure 13
Complete structure of Flexible Storyline

In order to allow early exit out of a diversion we have augmented the basic structure with a *Merge* clip. A merge is an optional video clip that transitions the diversion content back into the main story line. If a viewer chooses to exit early out of a diversion, a merge will smoothly transition a viewer into the main story. Finally, in order to address the last problem of deferring a diversion to play it later, we have added another optional clip - *Context*. A *Context* clip will only

be played before a deferred diversion. Its sole purpose is to summarize the Pre-story bringing a diversion back into context. Viewers watching the diversion at a later time may not remember the pre-story and the buildup. A *Context* clip will help them remember this material before they move into the diversion. The *Body* and the *Diversion* will still behave the same way as described above.

This whole structure is illustrated by our Harry Potter example mentioned in the introduction. In the *Pre-story* Harry finds himself selected as a competitor in a dangerous Triwizard tournament (see Figure 14). The tournament consists of three tasks. The first two tasks are shown in the pre-story. We now enter into a *Buildup*, where the third task, passing through a maze, is introduced to the competitors. The conditions of winning are explained in the buildup as well.

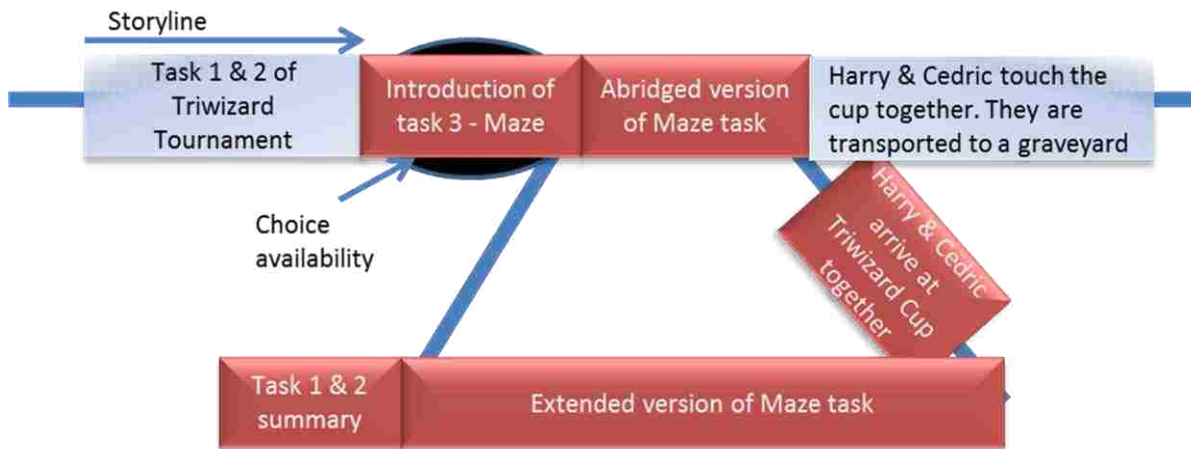


Figure 14
Harry Potter Triwizard tournament Flexible Story structure

The first one to get to a Triwizard Cup, placed in the middle of the maze, becomes the winner. During the time of Buildup there's a banner on the screen saying: "To see an extended version of the third task press X. To watch it later press Y". We will now examine each of the three options that a viewer has, and the effect that his actions take once *Buildup* is done playing.

1. A viewer performs no action.

The *Body* clip shows an abridged version of the maze task. In this version, the main obstacles are vines that pulling competitors into the bushes and the effect which a possessed maze has upon the minds of the competitors. Difficulty of passing through the maze united Harry and Cedric, the second competitor from Gryffindor. They get to the Triwizard cup together.

2. A viewer presses a button to choose a diversion.

The *Diversion* shows an extended version of the maze task. In addition to the vine, and possessiveness of the maze, Harry and Cedric encounter other obstacles. Such as Blast-Ended Skrewt, Boggart and Golden Mist. They have to solve a riddle given to them by sphinx and fight the giant spider, Acromantula.

The *Merge* will now play to bring the diversion back into the main storyline. In the merge, we see that having completed the third and final challenge of passing through the maze, Harry and Cedric become more united. By joining forces they were able to get to the Triwizard cup together. Notice that if we chose to exit out of the diversion early, merge will still help us connect back with the story.

3. A viewer presses a button to defer a diversion.

The *Body* clip is played. The diversion covering the Maze task is now added to the list of deferred diversions and will be available to watch later. When viewer choses to watch it, the *Context* clip will play, summarizing the first and second task of the tournament.

All three options end with the *Post-story*. Here we find out that Harry and Cedric decide to touch the Triwizard Cup together, to both become winners of the Tournament. As they touch the cup, they are transported out of the maze to a graveyard.

We will now show the structure of flexible storylines on two remaining examples given in the introduction: a documentary on the origins of Tunguska explosion and a reality show: *American Idol*.

Origins of Tunguska Explosion Example

In the *Pre-story* viewers learn about the meteorite theory of origin of Tunguska explosion (see Figure 15). As the meteorite theory is covered, and found to have weaknesses we enter a *Buildup*, where the comet theory is introduced. During the time of this *Buildup*, viewers also see a banner, saying: “Would you like to watch a more detailed version of the comet theory?”

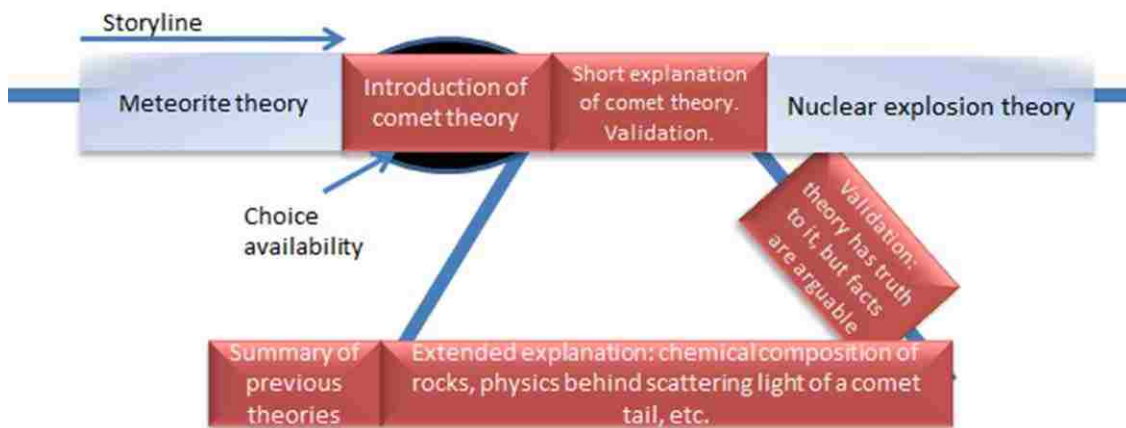


Figure 15
The structure of Flexible Storylines: the Tunguska explosion

The *Body* clip will include a simple explanation of the comet theory, with a few of the strongest facts that support it. It will end by determining that supporting facts are not strong enough to accept this theory as a most probable cause of the explosion.

The *Diversion* will include an extended explanation of the comet theory. For example, it could cover the chemical composition of the rocks found at the sight of the Tunguska explosion. In

addition, the *Diversion* will explain the physics behind scattering of light caused by a comet's tail and how it relates to the accounts of eyewitnesses of the explosion.

The *Merge* will include the validation of the theory and will conclude that the supporting facts are not strong enough. Notice, that if a viewer decides to exit a diversion early he or she will still be transitioned smoothly into a next concept. The *Merge* will wrap up the essential facts regardless of where in the *Diversion* an exit was taken. Viewers will be transitioned back into the main story, where they will learn about the nuclear explosion theory in the *Post-story*. The *Context* will include a summarization of the theories that were discussed before the comet theory.

American Idol Example

We will now show the example of flexible storyline's structure on the reality show, *American Idol*. In the *Pre-Story* different contestants are performing and being judged (see Figure 16). As the last contestant before contestant *N* is finished, story enters a *Buildup* clip, where the first song of contestant *N* is shown. Also viewers see a banner: "Would you like to see two additional songs by contestant *N*?" The *Body* clip will consist of opinions of judges on contestant *N*.

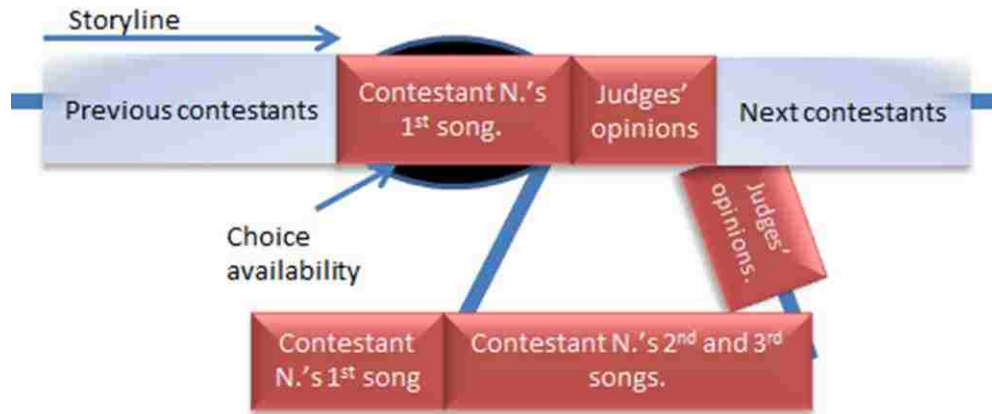


Figure 16
Flexible Storylines structure: The American Idol

The *Diversion* will include two additional songs sung by contestant *N.*, and will end with the *Merge* clip. In the *Merge* viewers will see the opinions of judges on contestant *N.*'s performance. Notice that if viewers choose to exit the *Diversion* earlier, they will still see the opinions of judges. The performances and judging of contestants following contestant *N.* will comprise the *Post-story*. Finally, the *Context* will include contestant *N.*'s first song, in order to remind viewers about this contestant.

Challenges

The challenges that we faced in this project fall into three areas: creator tools, seamless playback, and viewing experience. Challenges for building creator tools entailed building a familiar extension of film-editing that would be easy to use. Viewing experience challenges included providing an easy way for viewers recognize the availability of the choice and interact with it. Seamless playback area focused on providing continuous and unbroken playback to viewers.

Creator tools

Flexible Storylines is closely associated with film production process. Our main challenge is building a tool for story creators that will be first, a familiar extension of production film-editing software and second, easy to use.

In video editing tools a timeline is a chronological representation of the story. Clips can be dragged onto it consecutively to assemble the story. Non-linearity of flexible story has posed a challenge of creation of a timeline structure that could be descriptive of the Flexible Storylines structure shown in Figure 17.

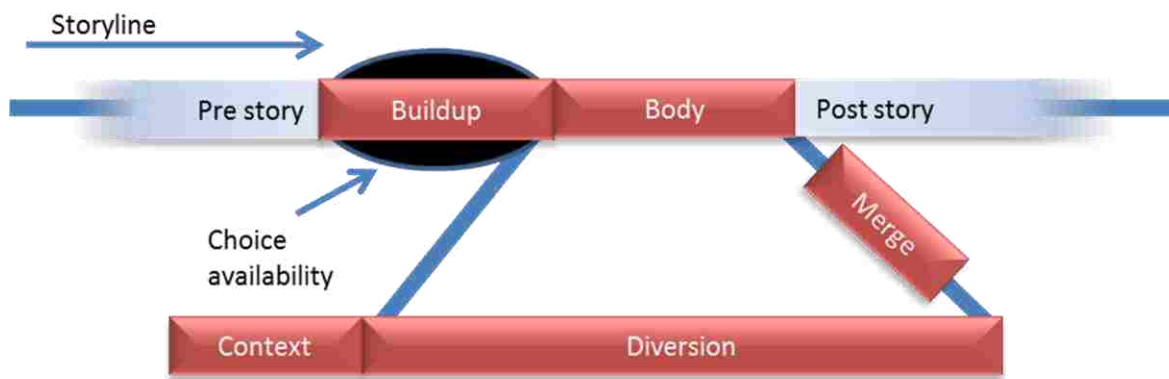


Figure 17
Flexible Storylines structure

Seamless playback

As seen in Figure 17 the structure of Flexible Storylines represents a cohesive story composed of distinct clips. Consecutive playback of discrete video clips over the internet requires halts for loading and buffering. Whether viewer chooses to watch a diversion or to skip it, there is a sequence of distinct clips that are to be loaded and played. For a viewer who skips a diversion, the sequence of clips is: a Pre-story clip, a Buildup clip, a Body, and a Post-Story clip. On the

other hand for a viewer who chooses to watch a diversion the sequence is: a Pre-story, a Buildup, a Diversion, a Merge and a Post-story. We want to avoid playback interruption needed to load and buffer subsequent clips in the sequence. The matter is complicated by allowing viewer to vary the sequence of clips to be played. After a Buildup is done playing there are two variations of subsequent clips: a Body, or a Diversion and a Merge. Story's non-linearity coupled with its segmented structure poses a challenge of seamless playback.

Viewing experience

Flexible Storylines structure (Figure 17) may also introduce several challenges for the viewing experience. The main challenge is how to make interactive viewing experience intuitive and easy. A viewer may not want to make any choices throughout the video story either because of lack of interest in the subject offered or fear of the unfamiliar. Another challenge is the gulf of execution problem, meaning viewer may want to make few story altering choices but will not know how. Viewers may either not notice the availability of the interactive interface or they may notice it but not know how to engage with it. If a viewer does choose a diversion, he or she may become uninterested in the subject and want to be returned to the main story. At the same time, viewers will not want to lose the flow and structure of the story. Viewers may also get annoyed with the way buildups are presented. For example, it will be bothersome if every buildup was a video clip showing an announcer declaring: "To see more of the next subject please press..." A large number of buildups may also cause unnecessary interruptions and be a burden for viewers. The length of buildups and diversion can also pose a challenge for viewing experience. For example, having too short of a buildup, may not give viewers enough time to make a choice. Similarly including diversions that are too long may cause viewers to constantly avoid them by

either not making a choice or exiting out early. Finally since we are adding an option to watch a diversion later, it may appear out of context for viewers.

Chapter 4 - Creator tools

Flexible Storylines is closely associated with the film production process. In order to create a tool that is a familiar extension of production film-editing software we need to understand the process of film production first. Figure 18 shows three main steps in film production: pre-production, production, and post-production (Davis 1975). The preproduction, or planning stage, includes developing film ideas, scripts, and storyboards. Production is where all the scenes for a film are shot, and post-production includes film editing, and distribution. Flexible Storylines affects the preproduction and postproduction stages of film-making process.



Figure 18
Steps of film production process

For preproduction, story creators need to understand the Flexible Storylines (see Figure 19) structure and plan to produce extra content to fill the structure. First, story creators need to identify the scenes in the story that will present viewers with a choice. Second, certain parts of a written script and story boards can no longer be linear. That means that while writing a script and story boarding, story creators will need to create two versions for interactive scenes: one for a Body clip, a shorter version of a scene, and one for a Diversion, a longer version. As each diversion is thought through story creators need to pay special attention to a Buildup and a Merge. A Buildup clip needs to be a good introduction to a Diversion that will catch viewer's interest. A Merge needs to be a smooth transition back into the main story. Story creators also need to remember that in case a viewer decides to exit a Diversion early, a Merge clip needs to contain necessary information for the Post-Story to make sense to a viewer. Finally story creators

need to plan to shoot story summarization clips that can be used for Context and possibly for Body clips.

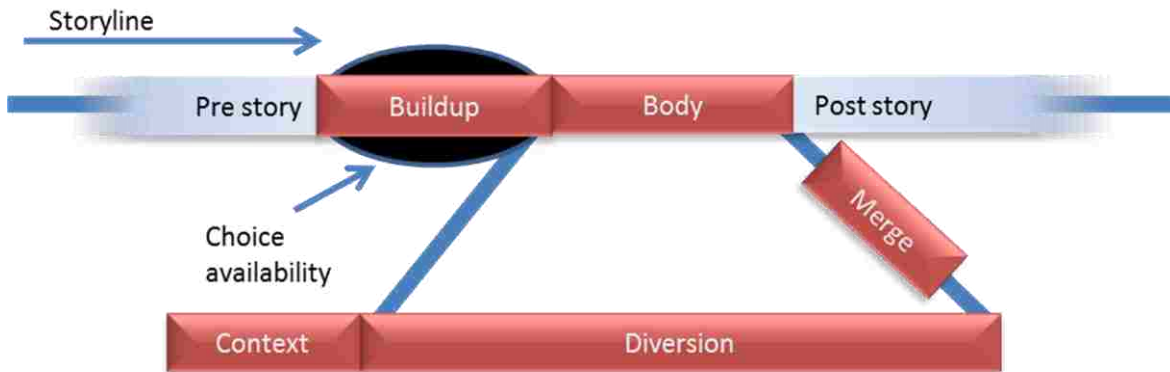


Figure 19
Flexible Storylines Structure

We will use the example of Harry Potter’s tri-wizard tournament in order to illustrate the preproduction planning process. In order to prepare a Flexible Storylines version of the *Harry Potter and the Goblet of Fire*, story creators will need to plan to shoot enough material to produce two versions, a shorter and a longer one, of the third task. Also, they will need to plan to produce a Buildup clip that will introduce a third task to viewers. A Merge will need to be prepared to describe that Harry and Cedric became united and arrived at Tri-wizard cup together. Finally, story creators need to plan to produce a summary of the story that occurred before the third task, to be used for a Context. Aside from planning for additional scenes to be shot, Flexible Storylines have small effect upon preproduction process.

During the post-production stage, story creators will edit their material, assemble it into the story and distribute it to the viewers. Main editing of video clips and audio is done using production film editing software such as the Avid or Final Cut Pro. Current video editing tools provide a way to create and edit clips and assemble them into a story. Stories are assembled by laying out

clips in a sequence onto a timeline. The process of creation of Flexible Storylines is also comprised of two main steps: creating story components and assembling them into a sequence.

We have provided two simple timeline-based editors to accomplish each step respectively – the Video Editor and the Flexible Creator. The main challenge for creating these tools was to make them a familiar extension of production video-editing software.

Production video editing tools

As an example of video production software we used Avid. This application offers three main modules: a clips bin, a source editor, and a timeline (see Figure 20). After video content is imported it is edited in a source editor. The source editor allows story creators to cut video pieces and place them into a clips bin. The cutting is done by seeking precisely to frames that will be a start and an end of a clip and marking them. Seeking is made simpler by allowing story creators to clicking on a video's timeline to jump to that spot and using “seek ten frames” or “seek one frame” buttons for a more precise seeking.

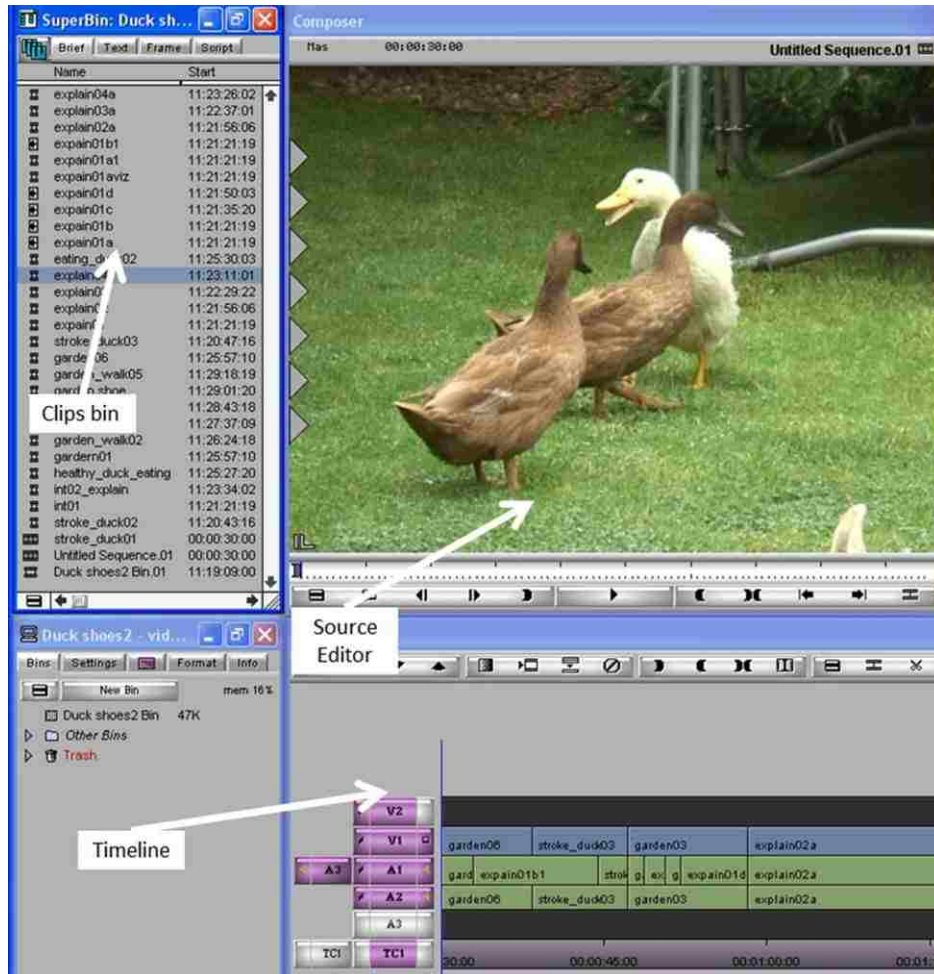


Figure 20
Avid screen shot

After the clips have been edited they are placed into a clips bin (see Figure 20). Story creators can then drag them from a bin onto a timeline. A timeline shows a graphical representation of all the clips in story's sequence. The length of a graphical representation of each clip is proportional to the time length of that clip. There is also a way to add audio clips to the audio tracks marked with letters A.

Flexible storylines creation

Creation of Flexible Storylines consists of two steps: creating story components and assembling them into a sequence. We have built two simple timeline-based editors to accomplish each step—the Video Editor and the Flexible Storylines Creator.

Video Editor allows story creators to cut video material into clips and label them. These clips will later be used to assemble a flexible story. After the clips are spliced and labeled they story creators will use Flexible Creator to assemble into a sequence that will form a story.

Video Editor

The Video Editor is a tool that is used for cutting clips that will compose Flexible Storylines structure shown in Figure 17. The video repository (Figure 21), allows story creators to upload and browse and select videos for editing. The video cutting panel includes the player with a timeline and an editable table of clips data on the right.



Figure 21
Video Editor tool. Left: clips repository, right: slicing tool and clips data table, bottom: timeline

After uploading a video, story creator will cut it into multiple clips. The cutting is done by placing a tracker at the desired position on a timeline and pressing a “New Marker” button that places a cut at the position of the tracker. In order to aid seeking to a specific frame we have added the following buttons for seeking forward and backward: “30 seconds seek”, “10 seconds seek”, “1 second seek”, “5 frames” and “1 frame”. Story creators cut the main video into clips and label them in accordance with the corresponding components of Flexible Storylines structure in Figure 17. Labeling is done through an editable table (see Figure 21-3) by double-clicking on a specific clips title and editing it. It is advised that story creators adopt a labeling scheme for clips to identify them with a scene and an item in the structure of Flexible Storylines.

For example in Figure 21 a story creator working with a video named “centurion1.wmv”. Clips created are shown in the clips table data, as well as in video repository (Figure 21). Looking at the titles of clips in clips data table we can notice that a story creator is planning an interactive scene that he identified with a letter “C”. The clip named C_Context is a Context clip for scene “C”, C_Buildup and C_Body are a Buildup and a Body for that same scene “C”. The rest of clips shown in the table are portions of a Diversion for scene “C”.

Flexible Creator

Non-linearity of flexible stories has posed a challenge for building a creator tool. It was important to construct a timeline template that matches Flexible Storylines structure shown in Figure 19 and is intuitive for story creators. We will first show the linear portion of Flexible Storyline structure. Figure 22 shows the structure of Flexible Storylines without a diversion. It is linear and therefore easy to represent on a timeline (see Figure 23).

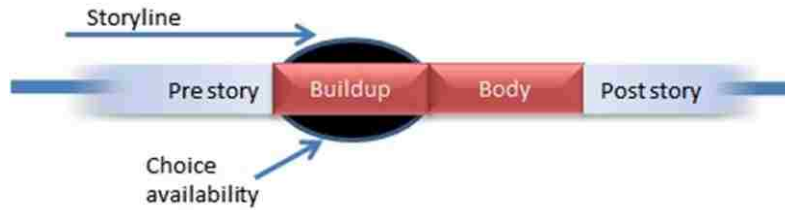


Figure 22
Flexible Storylines Structure without diversion. Timeline representation

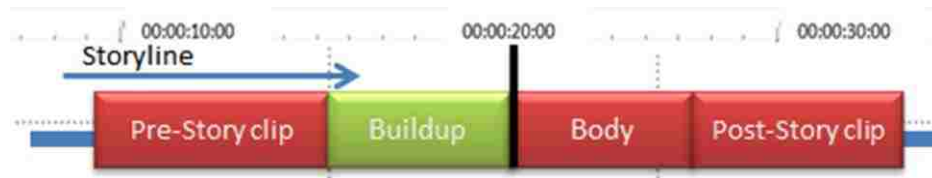


Figure 23
Timeline representation of Flexible Storylines without diversion

Adding non-linearity into the structure we now have two main paths through the story which is represented by two levels of timeline (Figure 24). The top level is the main storyline, represented by a Body clip. The bottom level is the extended version of the storyline, represented by a Diversion. Timeline slider will follow a viewer's choice and show video playback's position in the level that is currently being played.



Figure 24
Two paths through the story

Figure 25 shows the first step in transformation of Flexible Storylines structure into its timeline representation. We divide the components of Flexible Storylines structure into levels shown in Figure 24: the main storyline and the extended storyline. The content of a viewer's choice (either

a Body clip or its longer version - a Diversion) will only play immediately after the Buildup is finished. Therefore, it makes sense to align the start of a Diversion in the bottom level with the start of a Body in the top level of story (Figure 26).

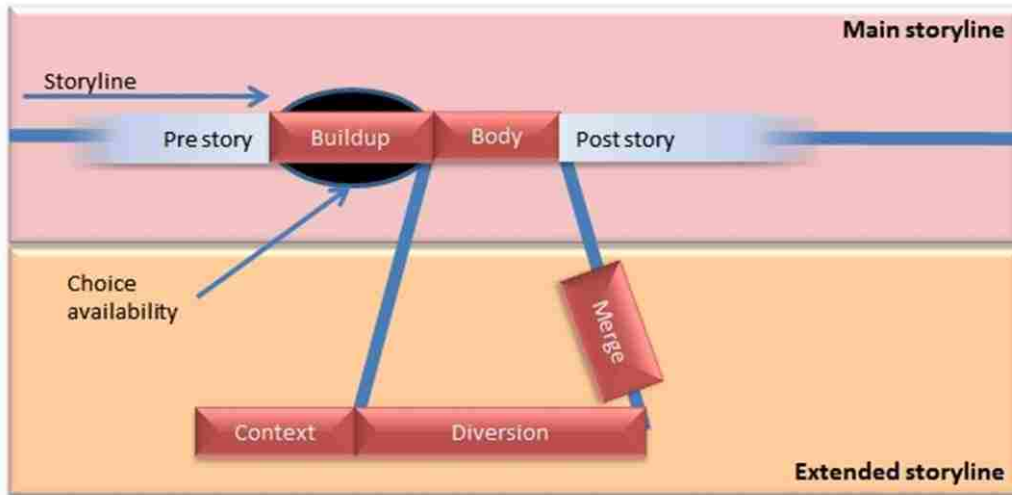


Figure 25
Timeline representation of Flexible Storylines structure. Step 1

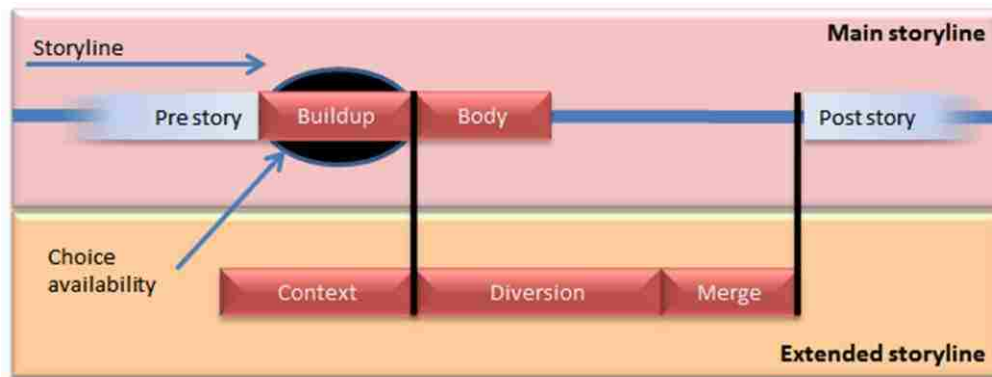


Figure 26
Timeline representation of Flexible Storylines structure. Step 2

After a Diversion is finished playing a Merge clip is played before returning to Post-story. Therefore in a timeline representation a Merge clip follows a diversion (Figure 26). A Context does not need to be aligned with any clip except that a Diversion must be placed immediately

after it. Figure 27 shows the final step in representing a structure of Flexible Storylines on a timeline.

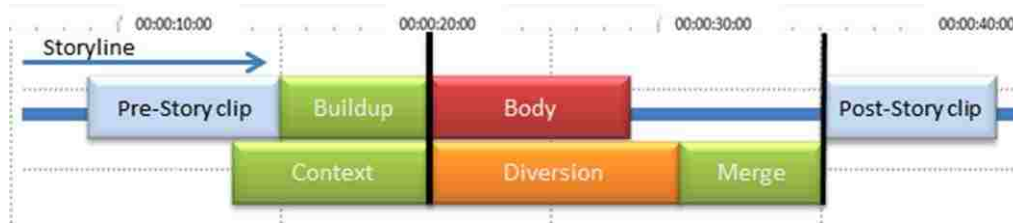


Figure 27
Timeline representation of Flexible Storylines. Step 3

Figure 28 shows a complete interface for the Flexible Storylines Creator. Flexible Storylines Creator consists of two components: video repository on the left and a story's timeline panel on the right. Both components have a video preview window associated with them. Selecting a clip in video repository shows its preview in the small preview window on the left. Story's timeline preview window, in turn, is able to play the complete story from a timeline.

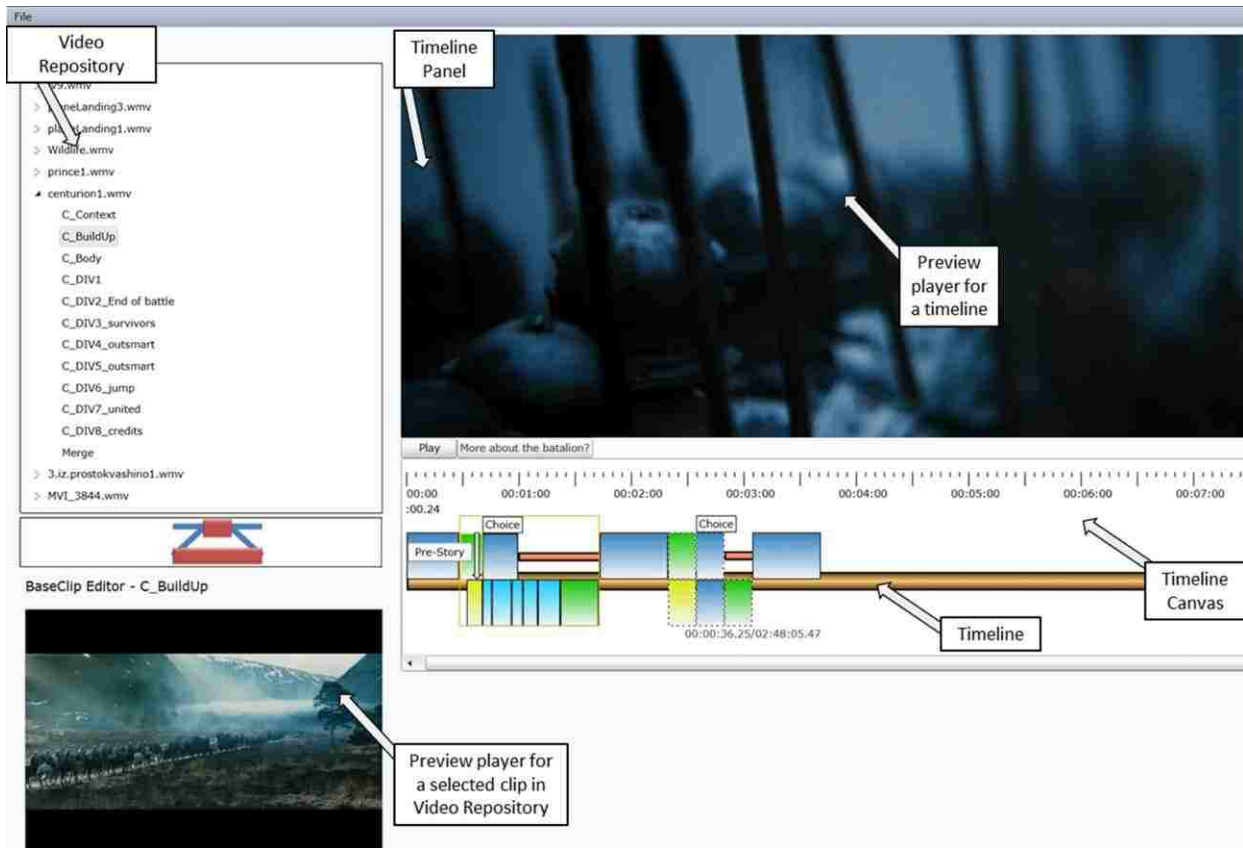



Figure 28
Flexible Storylines Creator

In order to assemble a story, clips are dragged from video repository onto a timeline. There are two types of scenes that can be added to flexible story: non-interactive Clips, and interactive Choices. A clip represents a piece of video content that is to be used as a part of Pre-story or Post-story. A choice is a template representing Flexible Storylines structure.

In order to add a choice a story creator drags this icon:  onto a story's timeline. As it is dropped on a timeline a template consisting of five placeholders is added. Each placeholder is a drop target for video clips for Buildup, Body, Context, Diversion and a Merge (see Figure 29).

Flexible Storylines added variability to the length of a story. In order to assist story creators with planning of the length of the story we have added a time scale above the timeline. Variability in the length of a story is introduced through the choice templates. We have provided two ways to show a choice: an expanded view (Figure 30) and a collapsed view (see Figure 31). An expanded view shows the length of a story if after a Buildup for that particular choice, a viewer chooses to watch a Diversion, which will be followed by a Merge. A collapsed view shows the length of a story if after a Buildup of that choice, a viewer decides not to watch a Diversion and will only see a Body clip. A story creator can switch between views for each choice by right-clicking on that choice scene (see Figure 32).

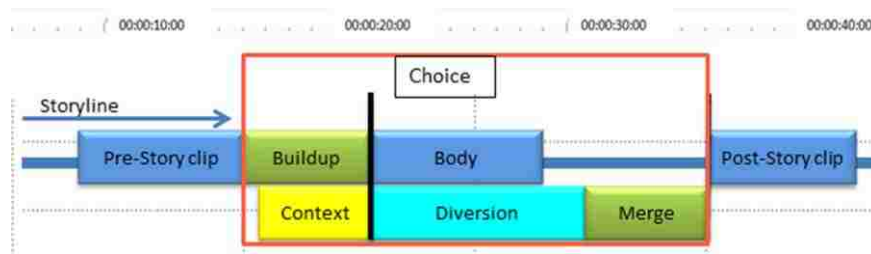


Figure 29
Choice template expanded view

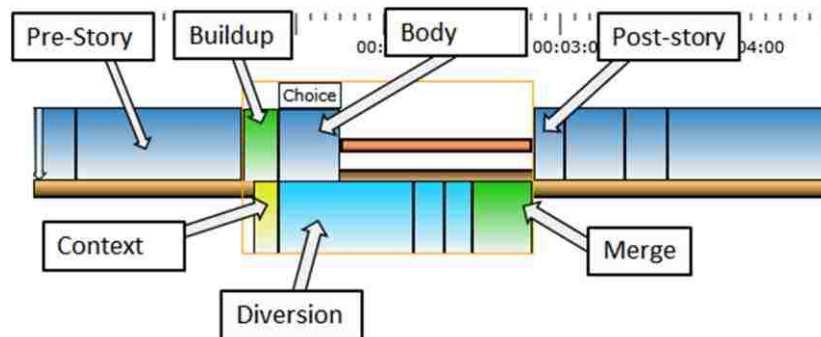


Figure 30
Choice template full view screenshot

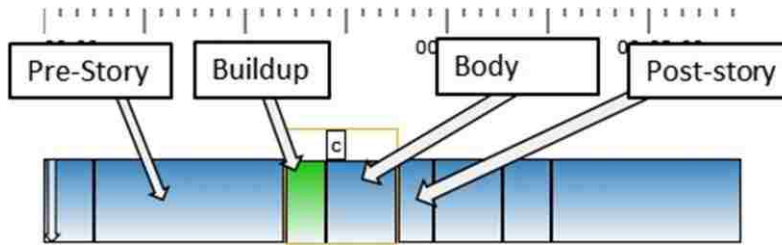


Figure 31
Choice template collapsed view screenshot

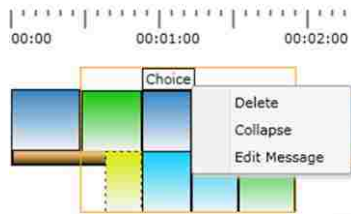


Figure 32
Choice's context menu

During a Buildup, viewers will see a transparent button on the left of their screen (see Figure 33) offering them a choice of taking a diversion. Story creators can edit a message that appears on that button to catch the interest of story viewers. This is done by right clicking on a choice scene (see Figure 32).



Figure 33
Player view, buttons on the left offering viewer a choice

Validation

We have approached the evaluation of creator tools from preproduction and postproduction perspectives. For preproduction evaluation we have tested the ability of story creators to plan and construct a flexible story. For postproduction evaluation, we have tested the usability of creator tools that we have implemented.

Preproduction validation

Flexible Storylines introduced a new concept of a story structure to story creators. For successful production of flexible stories, story creators must understand overall structure and the usage of its components. We have asked twelve Broadcast Journalism upper classmen to listen to a

presentation on Flexible Storylines. After the presentation we have asked them to plan and construct a flexible story.

The demographics of study participants consisted of three males and ten females. All thirteen participants were undergraduate degree seeking students in Broadcast Journalism. Every participant had a class standing of Junior. Finally, all participants had one year of experience in creating and producing visual stories.

Study participants were not familiar with the concept of Flexible Storylines previously. We gave a ten-minute presentation explaining the structure of Flexible Storylines and the usage of each component. We concluded the presentation with a demonstration of the creator tools that we implemented. At the conclusion of our presentation we have given each participant a task to construct two flexible stories.

We took two well-known fairy tales, Cinderella and Little Red Riding Hood, and divided each tale into scenes. Participants received these scenes in an unordered list. Their task entailed matching each clip with a component of Flexible Storylines. Figure 34 shows Little Red Riding Hood flexible story-building task. Notice that scenes are listed out of order. Flexible Storylines structure is provided for matching each scene to a single component.

Scene	Description
L1	Little Red Riding Hood decides to go to take food to her grandma. While going through the woods, wolf sends her on a longer path.
L2	Wolf gets to grandma's house, eats her, changes in her clothes
L3	Wolf is chasing little Red Riding Hood to heat her.
L4	Little Red Riding Hood comes in and sees a wolf, a wolf is chasing Little Red Riding hood to eat her.
L5	Woodsmen save Little Red Riding Hood and grandma.
L6	Little Red Riding Hood comes in and sees her grandma. "What big ears that you have" "What big eyes that you have" "What big teeth that you have"
L7	Short version of Little Red Riding hood story until she gets to the hose of grandma

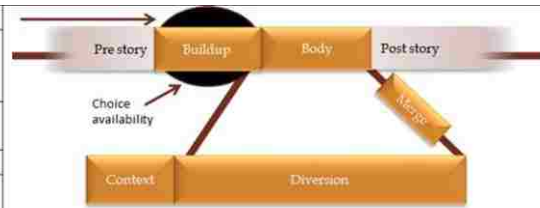


Figure 34
Little Red Riding Hood Flexible Story task

The task was complicated by the absence of actual video story in front of participants. Even with this complication, nine out of thirteen students (69%) were able to perform the task successfully. That means they correctly matched scenes in the table with the components of Flexible Storylines structure (Figure 34). Many students have commented that the task would be much easier to perform if they had an actual video story in front of them. Overall, this result signifies a success in preproduction evaluation.

Postproduction validation

We also evaluated how well our solution satisfies the needs of story creators in the postproduction step of producing a story. We asked Broadcast Journalism upper-classmen to use our system to create and distribute flexible storylines. The demographics of study participants consisted of twelve females. All twelve participants were undergraduate degree seeking students in Broadcast Journalism. Three participants had a class standing of Senior and two years of visual story telling experience. Nine participants had a class standing of Junior and one year of visual story telling experience. We specifically invited students that have experience in editing and production of visual stories.

To each participant we explained the concept of Flexible Storylines and showed them a demonstration of Creator tools. Next we have asked them to create a flexible story on their own using Video Editor and Flexible Creator tools. The first task was to cut a source video into clips using the Video Editor. The second task was to assemble a flexible story from these clips using the Flexible Creator. Five participants used one of two documentaries produced at BYU for their source video, since they were familiar with it. Remaining participants used movie trailers for *The Centurion* and *The Prince of Persia* for their source video.

We wanted to perform a preliminary validation first to obtain early feedback and fix current problems with our system. We have used three participants for a preliminary validation. Next we continued with user experience validation using nine remaining students as participants.

We recorded each participant's task execution. At the conclusion of the task, instead of filling out a survey, we asked participants to describe their experience (good and bad) and recorded each response. We have gathered and organized user experience data from recordings of task executions and participant's responses. User experience data was organized into three groups: negative feedback, positive feedback, and issues (or what when wrong with our system).

Preliminary validation

The negative feedback that we obtained from preliminary validation helped assimilate our system to current video editing tools such as Avid. We have made the following changes to creator tools:

1. Video Editor
 - a. Use space bar as an input for playing and pausing (same as Avid).

b. Enabled clicking on a timeline (see Figure 35) to seek a video position to that point.

c. Added clearer labels to seek buttons and increased buttons' size (see Figure 35).

2. Flexible Creator

a. Added a tooltip for each place holder in the choice structure, such as Buildup, Merge and Context (see Figure 36). This will help story creators know where to place a certain component of a flexible story.

b. Made a brown timeline which holds all the clips (see Figure 36) wider. This will simplify dropping clips onto a timeline.



Figure 35
Video Editor tool

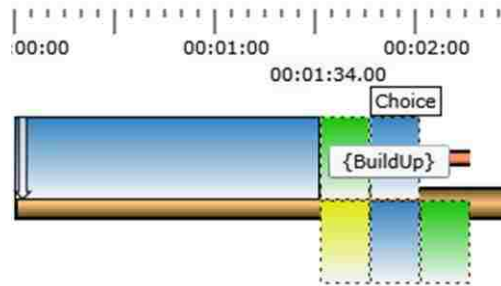


Figure 36
Flexible Creator hover hint

Two participants also wished that our system would add black out fades between clips. We have chosen not to implement this feature, since fades can be added to a video source in video editing tools such as Avid. The purpose for fades was to smooth out the edges of clips that did not have a clear separation. If a story is produced with the structure of Flexible Storylines in mind, there will be a clear separation between clips. Lastly, two participants stated that it would help them to have a picture of Flexible Storylines structure for reference while performing this task. For each participant in user experience validation we have made this picture available.

For positive feedback all participants in preliminary validation have reported our system to be user friendly, simple and easy to use. And two out of three participants have mentioned that our system was easier to use than Avid.

User Experience Validation

Having implemented the changes mentioned above we performed actual validation of creator tools using nine remaining participants. We will cover all four groups of responses: negative feedback, positive feedback, issues or what went wrong, and questions that participants asked.

Starting with the negative feedback three out of nine (33%) of participants expressed that the source video that they were given was not produced with Flexible Storylines structure in mind. That made it harder to cut into pieces that would smoothly fit together. Three out of nine participants (33%) explained that they would have done a better job, had they been more familiar with the source video beforehand.

Five participants (55%) explained that the hardest part of the task was to cut video into clips in Video Editor. Three of these five explained that this difficulty was caused by a novelty of concept of offering viewers a choice. These participants were used to create stories that are linear and do not offer viewers a choice. The two remaining participants explained that they needed more time to identify and cut the source video into clips to make the final story smoother.

The rest of negative feedback dealt with problems with interface of Video Editor and Flexible creator. For instance, two out of nine (22%) of participants could not find a way to undo a clip creation until it was shown to them. One participant (11%) found it inconvenient that we have provided only a way to drag a scene onto a timeline to place it between any two elements or after the last one, but not before the first element. This is an implementation defect that can be easily fixed. Also, one participant (11%) stated that Flexible Creator also needs to enable playing and pausing a video with a Space bar key.

For positive feedback, all nine participants were able to build interactive stories that made sense. Next, all nine of participants stated that our system was very simple, easy to use and user friendly. All participants expressed that our system was very similar to Avid. In addition five (55%) participants stated that our creator tools were easier to use than Avid. As stated by one participant: “As long as you have a basic idea of video editing software, this is easy to use”.

Two out of nine participants (22%) expressed that once they have understood the concept behind Flexible Storylines structure it was simple to create a flexible story. Two other participants stated that starting a preproduction of a film with Flexible Storylines in mind, would simplify greatly the usage of Creator tools.

We have also witnessed a few issues with our system while participants were working on their tasks. These issues can be grouped into two groups: first, implementation errors and hardware failures, and second, user interface inadequacies. We have had singular instances of the following issues caused by implementation errors and hardware problems:

1. Our system froze and needed to be reloaded.
2. Network connection dropped and since our system is a web-based system participants had to re-establish the network connection to continue
3. Mouse was not responding due to weak battery charge.
4. A participant was not able to use Video Editor because specifications for database table were not set correctly.
5. After the ending of one of the Merge clips, Flexible Creator did not play the clip from the Post-story.

No issues caused by implementation errors occurred more than once.

User interface inadequacies caused issues that were more repeatable in nature. We will first discuss the Video Editor's interface inadequacies, and Flexible Creator's next. Video Editor allowed labeling the clips for easier identification of them during story assembly. Labeling was done using clip data table (see Figure 37). Three participants (33%) were off by one in labeling their clips. That means that while labeling C_DIV2 (see Figure 37) which is a fifth clip in the

sequence they thought that they were labeling the fourth clip in sequence since timeline tracker was on it.

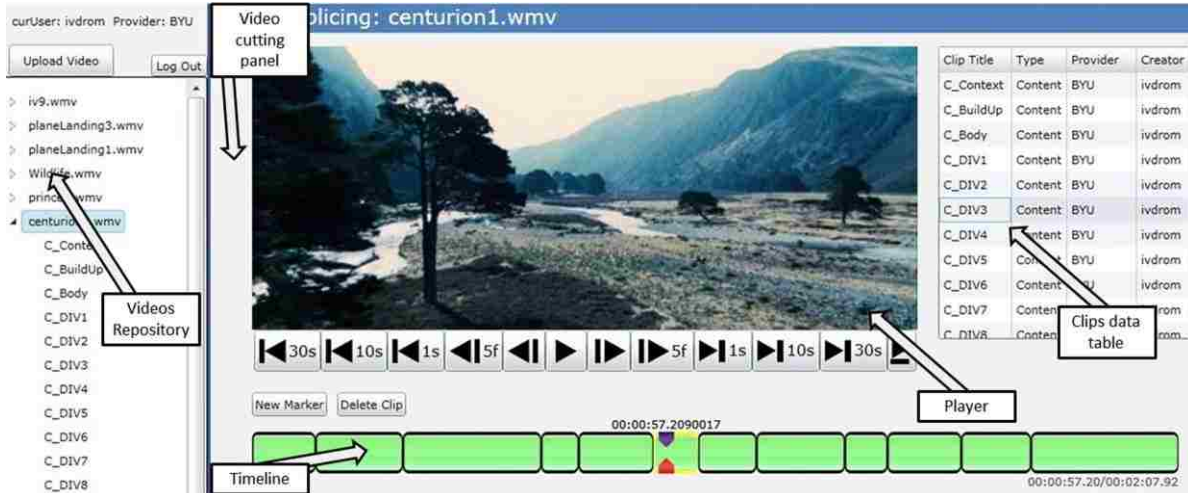


Figure 37
Video Editor Interface

This problem came from a few user interface design errors on our side. Upon selection of a clip in clips data table it was highlighted with a yellow border on a timeline (see timeline in Figure 37). The rest of clips on a timeline had a black border. Yellow as opposed to black has a lower contrast, and therefore does not catch story creator's attention. We also did not put label above a selected portion of a timeline to show the current label of the selected clip. These user interface design errors hindered story creators' perception of correlation between selected clips in a data table and timeline. These participants had to go back and rename each clip upon realizing that they were off by one.

Three participants (33%) were unclear as to what they need to do if they have created a cut at the incorrect place. Although they were informed of a Delete Clip button, it was unclear to them which cut will be deleted (clip's start offset or clip's end offset).

Flexible Creator had few of user interface inadequacies as well. Three participants did not realize that they are able to add multiple clips to a diversion. Almost certainly the reason for that was identical color coding for Body and a Diversion placeholder in the Choice template (see Figure 38). The color coding does differentiate a Body from Diversion but only after you add a clip to a Diversion (see Figure 39).

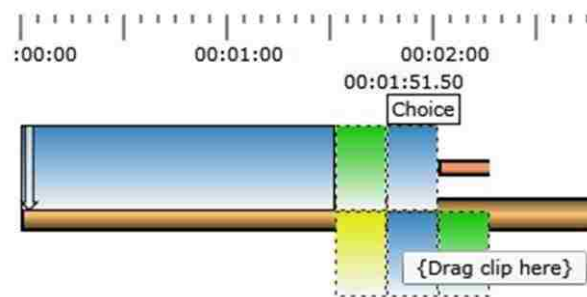


Figure 38
Choice template before dragging clips to a Diversion

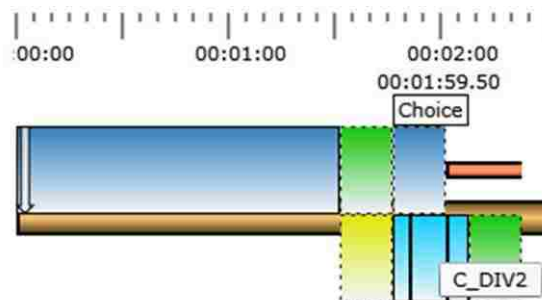


Figure 39
Choice template after dragging clips to a Diversion

There was yet another user interface problem with the Choice template. Since Buildup appears to be on the main story line in Flexible Storylines Structure (see Figure 40) three participants out of nine assumed that when they would first place a build up onto a timeline and add a Choice template second. A Buildup actually is a part of a Choice template and is colored with green. This was mostly likely caused by the error in a design of the Choice template icon (see Figure

41) that is used by story creators to add a Choice template to a timeline. Comparing Flexible Storylines structure (see Figure 40) to a Choice template icon (Figure 41), it appears that a Choice template will only add a Body and a Diversion place holders. A fix for this is to make a Choice template icon show a Buildup also (see Figure 42).

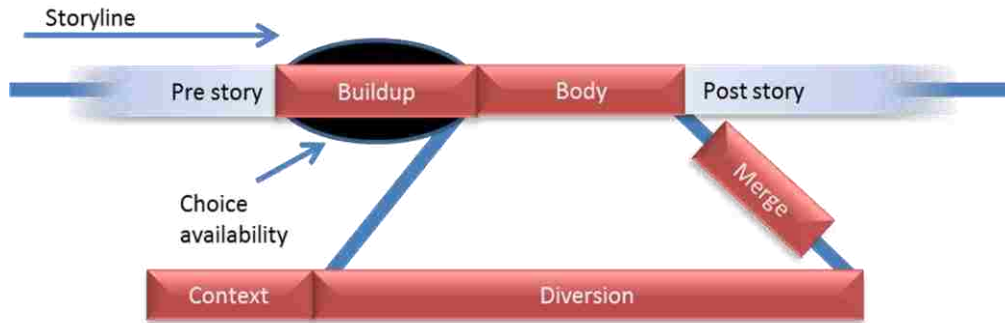


Figure 40
Flexible Storylines Structure

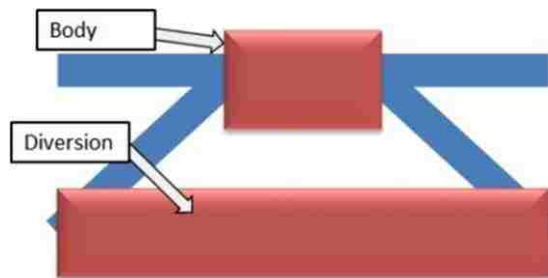


Figure 41
Choice template icon

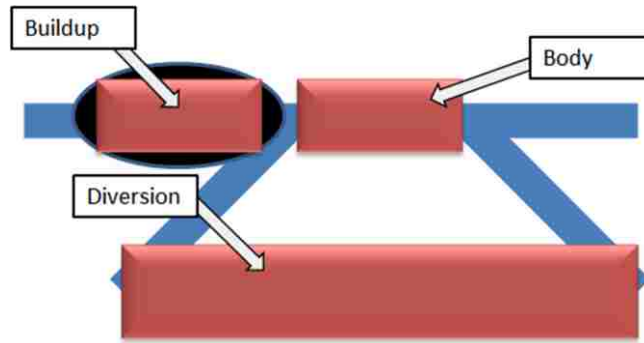


Figure 42
Extended version of a Choice template icon

Finally four out of nine participants had difficulties placing scenes onto a timeline. Instead of dropping them directly on timeline (see Figure 43) they dropped them anywhere at Timeline Canvas. A fix for this issue is allowing story creators drop a scene anywhere in the Timeline Canvas, and place a new scene after the closest scene on a timeline.

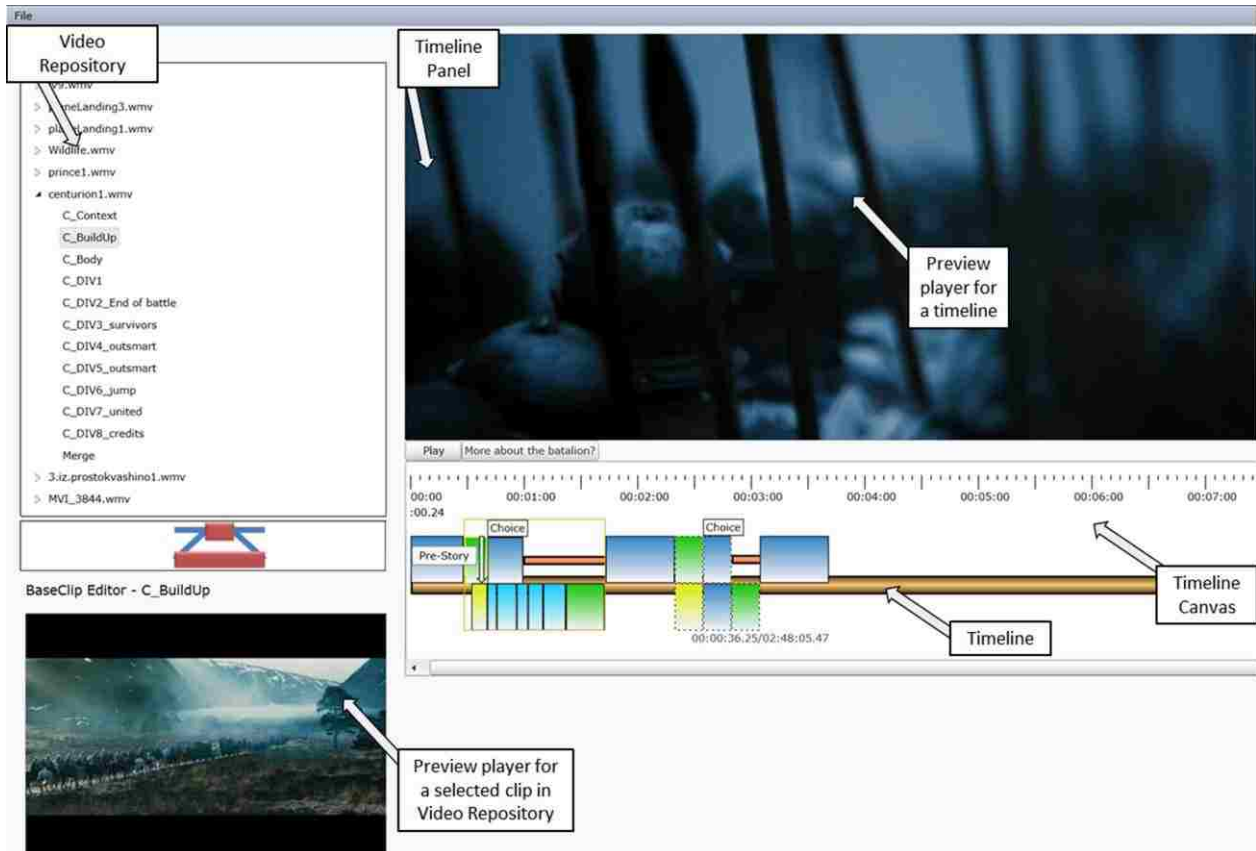


Figure 43
Flexible Creator interface

Conclusion

Flexible Storylines affects two stages of film-production process: preproduction and postproduction. Preproduction involves planning a film with the thought of Flexible Storylines structure in mind. Postproduction involves creation of flexible stories using Flexible Storylines Creator tools. Story creators use video editing software such as Avid, to edit video clips, combine them with audio and create self-contained components of a story. Next, story creators use Flexible Storylines Creator tools to edit these components if needed and label them using

Video Editor. Finally, Flexible Creator tool is used to assemble labeled components into a flexible story.

We presented story creators with first a new concept of story's structure, and second a new tool. We have performed evaluation of story creator's comprehension of a new concept in preproduction evaluation. Preproduction validation has shown that 69% of story creators were able to comprehend this new concept of story's structure. Postproduction validation evaluated first, our software usability, and second, its similarity to video editing software. 100% of participants were able to build interactive stories that made sense. Also, 100% of story creators have found our software simple, user friendly and easy to use. Finally, 100% of story creators stated that our system was very similar to Avid. These results signify great success of our system.

Chapter 5 - Seamless playback

Flexible Storylines is Internet-based application that allows story creators assemble flexible stories, and story viewers to view them. Video delivered over the Internet is subject to several constraints such as the time needed to load and buffer, and very slow seeking within a video. Playback may be interrupted to buffer the next portion of the video.

Usually one can avoid delays by watching a video of a really low quality. If a video of a higher quality is chosen a solution is to allot a certain time, based on the speed of internet connection, for video to load and then do playing and seeking. The matter is complicated if one is trying to play a video play list over the internet. Taking YouTube as an example, there is, on average, 2-2.5 seconds delay between each video on a play list.

The Flexible Storylines structure embodies a cohesive story composed of discrete clips (see Figure 44). Consecutive playback of discrete video clips over the Internet requires halts for loading and buffering. Whether a viewer chooses to interact or not there is a sequence of distinct clips that are to be loaded and played. This is comparable to a video playlist, with one major difference: all items in a list are to be stitched together into a single story. An ability to interact with a story adds additional complexity to seamless playback. The sequence of discrete clips to be played is no longer linear. We need to ensure that all paths through a story will play seamlessly. The goal of Flexible Storylines is to play our cohesive interactive stories consisting of distinct clips seamlessly, with no interruptions or halts.

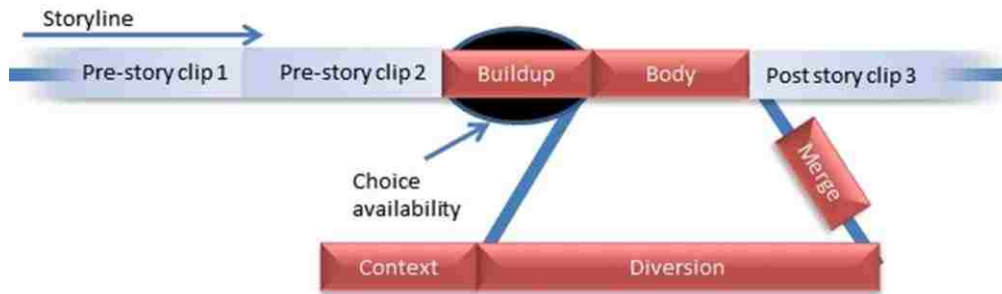


Figure 44
Flexible Storylines Structure

Seamless playback was achieved by preloading the immediate successor to the currently playing video clip. Interactive points introduced alternative video clips that could follow a current one. By accounting for all possible successors we were able to achieve seamless playback of interactive stories.

Figure 45 shows all possible paths through a flexible story. Looking at the structure of Flexible Storylines we see that the story is linear until the Buildup (Figure 45). At the time of the Buildup there is a “fork” in the sequence of clips to be played. If a viewer takes no action, a transition (3) takes a viewer to the Body. After viewing the Body, transition 10 will take a viewer to Post-story. If a viewer decides to watch a diversion, transition (4) takes him or her to Diversion clip 1. While watching Diversion clip 1, a viewer has an option to watch a diversion until the end or exit out. Watching a diversion to the end will take a viewer to Diversion clip 2 through transition (5). Exiting early, through a transition (6) will take a viewer to Merge. After Merge a transition (9) will take a viewer to a Post-story.

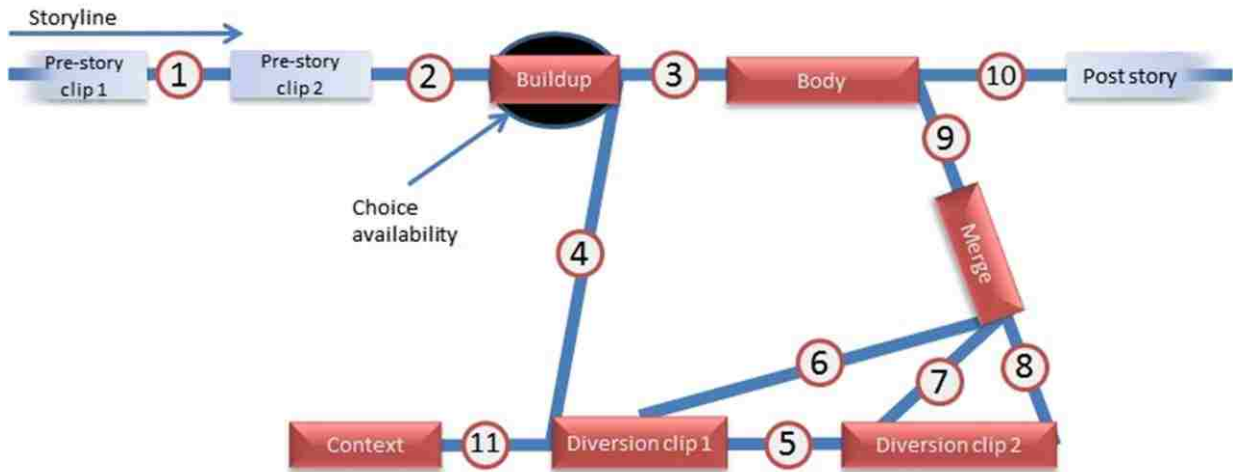


Figure 45
Two sequences of playback order

Completely pre-loading all possible sequences will take too long and will be a waste of resources. However, while playing a clip it does make sense to preload its possible successors. For example the simplest case is playing a sequence of linear clips: we are playing Pre-story clip 1, and the next transition (1) will take us to Pre-Story clip 2 (see Figure 45). As Pre-story clip 1 starts its playback it tells Pre-story clip 2 to preload. Similarly Pre-story clip 2 starts its playback and the next transition (2) will take us to Buildup. Therefore, Pre-story clip 2 tells Buildup to preload.

Listed below are all other possible cases of a need to preload:

1. After starting to play Buildup there are two possible transitions (3) and (4) that take viewer to the Body and the Diversion respectively. Therefore, as the Buildup starts its playback it tells the Body and the Diversion clip 1 to preload.
2. After starting to play a first clip in a Diversion, there are two possible transitions: (5) and (6) (Figure 45). Transition (5) goes to Diversion clip 2. Transition (6) is taken when a viewer decides to exit the Diversion early. Therefore, upon starting a playback of

Diversion clip 1, both Diversion clip 2 and the Merge need to get preloaded. In general, every clip inside a Diversion will need to preload its successor after starting to play.

3. After starting to play Diversion clip 2, there are two possible transitions: (8) upon completion of Diversion clip 2's playback, and (7) upon early exit out of Diversion clip 2. Both of these transitions end in the Merge. There is no need to preload the Merge, since we preloaded it during the playback of Diversion clip 1.
4. After starting to play the Merge, the only next transition is (9). Therefore, Post story clip needs to get preloaded after Merge starts its playback.
5. After starting to play the Body, transition (9) is the only possible transition. It takes us to the Post story. The Post story needs to get preloaded.
6. If a story has been deferred, the Context clip will be played before a diversion. As the Context clip starts playing, Diversion clip 1 is reached through transition 11 and needs to get preloaded (see Figure 45).

In order to accomplish all the preloading logic we have created a set of data structures to represent a flexible story. We have also shown that while playing a video we might have to preload several other videos. This preloading logic required a concurrent use of several media players: one to play a current video, and others to preload immediate subsequent transitions.

Figure 45 shows that there are at most two transitions available from any portion of a story.

Hence, all interactions presented by Flexible Storylines never create more than two successors to currently playing video. We have chosen to use a resource pool of Microsoft Smooth Streaming Media Elements to accomplish concurrent preloading and playing.

This chapter will first give an overview of all data structures and how do they fit together. Next, we will describe in detail the implementation of playing and preloading of a flexible story.

Finally, we will examine the implementation of the resource pool of media players that allowed concurrent playback and preload. We will conclude with a description of validation process and its results.

Flexible Story Data Structures Overview

The need to preload raised a question of ownership and connection between objects. A current video upon the start of its playback needed to preload videos in each possible immediate transition. A first option would be to give current video access to the subsequent videos. That means that Pre-Story clip 1 will need access to Pre-Story clip 2 (see transition (1) in Figure 45). As we continue looking at Figure 45, Pre-Story clip 2 needs to have access to Buildup. Listed below are all the remaining connections for each clip shown in Figure 45:

1. Buildup needs access to Body and a first clip in a Diversion.
2. Each clip in a Diversion needs to have access to the next clip in a sequence and a Merge.
3. A Body needs to have access to a Post-Story clip.
4. A Merge needs to have access to a Post-Story clip.
5. A Context needs to have access to a first clip in a Diversion.

This approach makes video objects highly integrated and connected with each other. This results in a set of data structures where most of the objects are interconnected and rely on each other.

This approach is a bad design practice and will complicate code's maintenance.

Another option is to have a centralized monitor to keep track of preloading. A centralized monitor could hold a collection of clips, keep track of the current index of a video playing, and ask the next video in a collection to prepare. This approach would work great for a linear

sequence of objects. For example while playing Pre-story clip 1, the first element in the list of clips, (see Figure 45) a monitor could preload a second element, Pre-story clip 2.

Non-linearity of our sequences introduces extra state information to be kept in a monitor. For example while playing Buildup, a monitor needs to discern that preloading the next clip in a sequence, Body, is not enough. Now a monitor also needs to preload a first clip in the Diversion collection (see transitions (3) and (4) in Figure 45).

In turn, once a viewer enters a diversion a monitor needs to preload the next video in Diversion, as well as Merge. Upon starting to play Merge or Body, a monitor needs to preload Post-story (see transitions (9) and (10) in Figure 45). This also leads to a tight interconnectivity between classes since monitor will need to keep a large amount of state information to identify the next video in a sequence.

While our implementation uses an approach of a centralized monitor, we were able to avoid keeping a large amount of state information in it. In order to make the relationship between classes clear, we will first describe the structure of each class. Next, we will cover the details of each class's implementation.

We have created a notion of **FlexClip** interface (see Figure 46). **FlexClip** represents a playable and preloadable scene. A scene can be either interactive or non-interactive.

Implementations of **FlexClip** interface need to implement a **Play** method and a **Preload** method. Execution of a **Preload** method must prepare a **FlexClip** for playing.

```
1. interface FlexClip
2. {
3.     void Play();
4.     void Preload();
5. }
```

Figure 46

In order to accomplish managing of playback and preload we have added a **FlexClipHolder** base class (see Figure 47). **FlexClipHolder** is a base class for each class that serves as a container for a number of FlexClips. It holds a reference to the currently playing FlexClip – **currentFlexClip**. To manage FlexClips' playback we need a mechanism to know when a current **FlexClip** has finished playing. Notification is done through **ClipFinishedPlaying** method. Since this method is abstract, each subclass of **FlexClipHolder** will need to implement it.

```
1. class FlexClipHolder
2. {
3.     FlexClip currentFlexClip;
4.
5.     //will be implemented in subclasses
6.     abstract void ClipFinishedPlaying();
7. }
```

Figure 47
FlexClipHolder class

Flexible Story is an ordered collection of scenes, or **FlexClips**, that can be preloaded and played. Below we are showing a timeline version of a Flexible Storyline (Figure 48 **Error! Reference source not found.**) along with a visualization of a **FlexStory**'s data structure (see Figure 49). **FlexStory** serves as a container for an array of **FlexClips**. Non-interactive

FlexClips are called **Clips** and interactive – **Choices**. Figure 49 shows a Flexible Story that consists of three **FlexClips**.

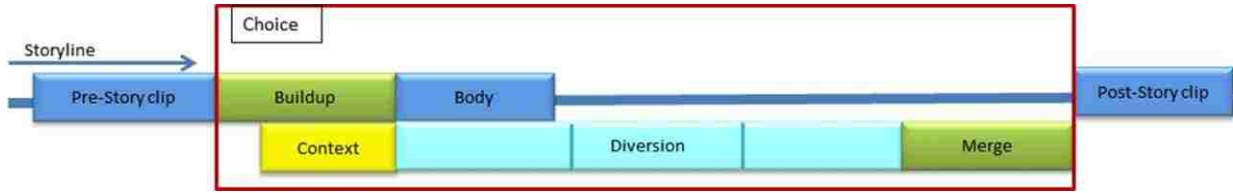


Figure 48
Flexible Story Timeline Version

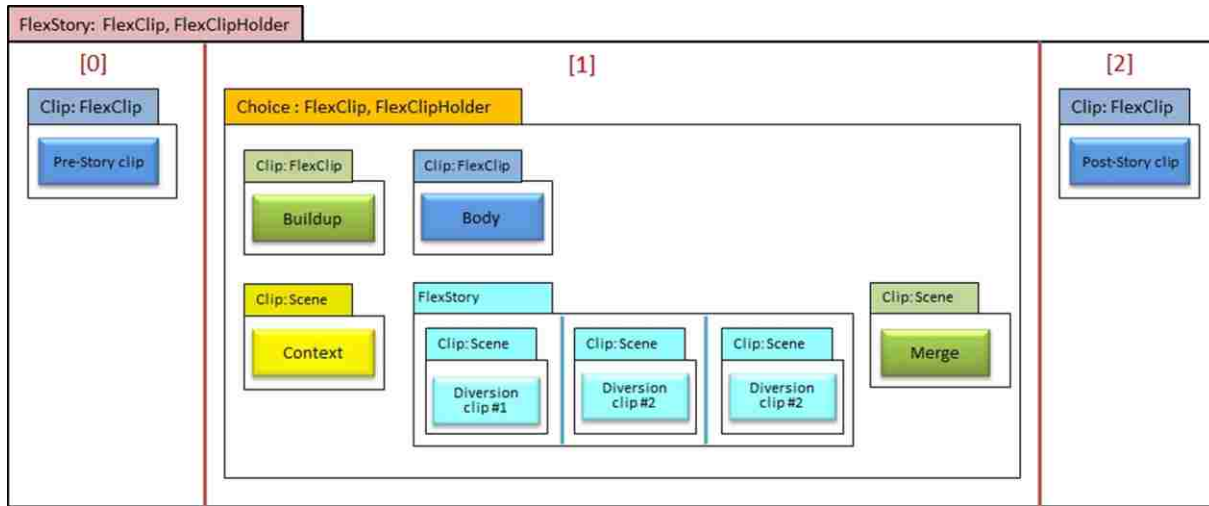


Figure 49
Flexible Story Data Structure

FlexStory class implementation is shown in Figure 50. **FlexStory** implements the **FlexClip** interface because it can be played and preloaded. The details of **Play** and **Preload** methods will be covered in the next section. The **FlexStory** class also extends **FlexClipHolder**, and needs to implement the **ClipFinishedPlaying** method. Through the **ClipFinishedPlaying** method, **FlexStory** will be notified when each of its containing **FlexClips** finishes its playback. This will direct the playback and preload of a story and will

also be discussed in the next section. The ordered collection of scenes in a story is held in the **flexClips** array (Figure 50 line 4). **FlexStory** keeps track of a currently playing **FlexClip** through a **currentClipIdx** variable (Figure 50 line 7).

```
1. class FlexStory implements FlexClip extends FlexClipHolder
2. {
3.     // an array of scenes, interactive and non-interactive
4.     FlexClip flexClips[];
5.
6.     // index of the current scene playing
7.     int currentClipIdx;
8.
9.     void Play() {}
10.
11.    void Preload() {}
12.
13.    void ClipFinishedPlaying() {}
14. }
```

Figure 50
FlexStory class

The simplest unit of flexible story's ordered collection is a non-interactive scene, called **Clip** (Figure 51). The **FlexStory** in Figure 49 has two Clips, first under index 0 and second under index 2. **Clip** implements **FlexClip** since it can be played and preloaded. **Clip** class stores information about the video that it will play: the video's url, as well as starting and ending time of a clip within that video (Figure 51). **Clip** interacts directly with resource pool of media players for preloading and playing. We will describe the interface of the resource pool of media players and **Clip**'s interaction with it in the next section.

```

1. class Clip implements FlexClip
2. {
3.     URL videoUrl;
4.     long startOffset;
5.     long endOffset;
6.     int playerId;
7.
8.     void Play()
9.     {
10.         ask Media Players' Resource Pool start player at index = [playerIdx]
11.     }
12.
13.     void Preload()
14.     {
15.         ask Media Players' Resource Pool to preload this clip
16.     }
17. }

```

Figure 51
Clip class

The other unit of flexible story's ordered collection is an interactive scene, called **Choice** (Figure 52). The **FlexStory** in Figure 49 stores Choice at index 1. As a scene that can be played and preloaded, **Choice** implements the **FlexClip** interface. **Choice** contains five **FlexClips**: a **BuildUp Clip**, a **Body Clip**, a **Context Clip**, a **Merge Clip**, and a **Diversion**. A **Diversion** is of type **FlexStory**, because it can hold multiple **Clips**. According to viewers actions **Choice** will direct the playback sequence of containing **FlexClips**. Notice that since **Diversion** is of type **FlexStory**, it appears to allow for nested Choices. Nested Choice occurs when one of the containing elements of **Diversion** is a **Choice** of itself. This situation can be confusing for viewers and requires additional players in the resource pool. For this reason, we placed implementation restrictions to prevent nesting of Choices.

```

1. class Choice implements FlexClip extends FlexClipHolder
2. {
3.     Clip BuildUp;
4.     Clip Body;
5.     FlexStory Diversion;
6.     Clip Merge;
7.     Clip Context;
8.
9.     void Play() {}
10.
11.    void Preload() {}
12.
13.    void ClipFinishedPlaying() {}
14.}

```

Figure 52
Choice class

Being a class that is a container for a number of **FlexClips**, **Choice** extends **FlexClipHolder**. As with **FlexStory**, **ClipFinishedPlaying** method assists the playback and preloading of **Choice**. We will discuss the implementation of **Play**, **Preload**, and **ClipFinishedPlaying** methods in the next section.

By using **FlexClip** interface we have created a linear abstraction of non-linear story. A **FlexStory** is a linear collection scenes, or **FlexClips**. Each non-interactive scene, a **Clip**, plays from start to finish. Each interactive scene, a **Choice**, offers viewer choices, and handles interaction appropriately. Both **Clip** and **Choice** implement their own playback and preload functionality that we will discuss in detail later. A **FlexStory** simply keeps track of a current scene's index and calls a **Play** method on a current scene to play a story.

Playing and Preloading

Our preloading logic requires a concurrent use of several media players: one to play a current video, and others to preload immediate subsequent transitions. We have chosen to use a resource pool of Microsoft Smooth Streaming Media Elements to accomplish concurrent preloading and playing.

The resource pool is represented by a class called **SSMHolder** (Smooth Streaming Media Elements Holder) shown in Figure 53. **SSMHolder** contains an array of media players of type **SmoothStreamingMediaElement**. **SSMHolder**'s interface offers three methods to be used by **Clip**: **Preload(Clip c)**, **PlayIdx(int playerId)** and **ReleaseIdx(int playerId)**. **Preload** (Figure 53 line 5) preloads a **Clip** into a free slot in the **players** array. It uses **Clip**'s starting time to prepare a video by seeking to that time. It also sets a watch point for ending time of a **Clip** to know when a **Clip** has finished playing. Before **Preload** returns, it saves the index of assigned player in **Clip**'s **playerIdx** (Figure 51 line 6). **PlayIdx** (Figure 53 line 13) simply starts the playback of a player under the index **playerIdx**. **ReleaseIdx** (Figure 53 line 18) marks a player at **playerIdx** as free by setting it to null.

```
1. class SSMHolder
2. {
3.     SmoothStreamingMediaElements players [];
4.
5.     void Preload(Clip c)
6.     {
7.         take a free player
8.         load video into player
9.         seek player to the startOffset
10.        set c's playerId
11.    }
12.
13.    void PlayIdx(int playerId)
14.    {
15.        players[playerIdx].Play();
16.    }
17.
18.    void ReleaseIdx(int playerId)
19.    {
20.        players[playerIdx] = null;
21.    }
22. }
```

Figure 53
SSMHolder class

Play and Preload Implementations

We will look at the details of **Play** and **Preload** methods of each implementation of **FlexClip**, starting with class **Clip**. **Clip**'s **Preload** calls **Preload** method on the resource pool of media players (Figure 54 line 13). **Clip**'s **Play** method calls **PlayIdx** method on the resource pool passing **playerIdx** as a parameter. **Clip**'s **Release** method calls **ReleaseIdx** method on the resource pool also passing **playerIdx** as a parameter. The implementation of **Preload**, **PlayIdx** and **ReleaseIdx** methods of the resource pool were discussed in the beginning of this section.

```
1. class Clip implements FlexClip
2. {
3.     URL videoUrl;
4.     long startOffset;
5.     long endOffset;
6.     int playerIdx;
7.
8.     void Play()
9.     {
10.         ssmHolder.PlayIdx(playerIdx);
11.     }
12.
13.     void Preload()
14.     {
15.         ssmHolder.Preload(this);
16.     }
17.
18.     void Release()
19.     {
20.         ssmHolder.ReleaseIdx(playerIdx);
21.     }
22. }
```

Figure 54
Clip class implementation

Next two implementations of **FlexClip** are **FlexStory** and **Choice**. Both **FlexStory** and **Choice** serve as a container for a number of **FlexClips**, and therefore extend **FlexClipHolder**. The playback of these classes is initiated by their **Play** method. The rest of their playback depends upon **ClipFinishedPlaying** method. **ClipFinishedPlaying** method, called every time a contained clip finishes playback, determines the next action to be taken.

FlexStory's **Preload** method, simply asks the first scene to preload itself (Figure 55 line 17). FlexStory's **Play** method first checks if a story has more scenes to play. If so, the current scene in **flexClips** array initiates its playback (Figure 55 line 11), and the next scene preloads itself (see Figure 55 line 12). If there are no more scenes to play than FlexStory raises **Finished** event. **Finished** event notifies any class that is listening to this event that FlexStory has finished playing.

```
1. class FlexStory implements FlexClip extends FlexClipHolder
2. {
3.     // an array of scenes, interactive and non-interactive
4.     FlexClip flexClips[];

5.     // index of the current scene playing
6.     int currentClipIdx;
7.
8.     void Play()
9.     {
10.         if (currentClipIdx < flexClips.length)
11.             flexClips[currentClipIdx].Play();
12.             flexClips[currentClipIdx+1].Preload();
13.         else
14.             Raise Finished Event
15.     }
16.
17.     void Preload()
18.     {
19.         flexClips[0].Preload()
20.     }
21.
22.     void ClipFinishedPlaying()
23.     {
24.         currentClipIdx++
25.         Play();
26.     }
27. }
```

Figure 55
FlexStory Class Implementation

After initial call to play **ClipFinishedPlaying** method directs the playback of **FlexStory**. **ClipFinishedPlaying** method gets called whenever an element of **flexClips** array finishes its playback. The index of the current scene is incremented (Figure 55 line 24), and the **Play** method is called to continue story's playback.

Choice class's **Preload** method preloads **Buildup**, since **Buildup** is the first scene in **Choice**'s playback (Figure 56 line 18). The **Play** method initiates the playback of **Buildup** (Figure 56 line 10) and tells **Body** and **Diversion** to preload. As mentioned before, the **Play** method tells the resource pool to start a playback of a specific player. The players of the resource pool perform their playback on the separate thread. **Choice**'s playback is initiated with a **Play** method but continues to be driven through **ClipFinishedPlaying** method. This method gets called when **Body**, **Diversion**, **Merge**, or **Context**, finish their playback.

```

1. class Choice implements FlexClip extends FlexClipHolder
2. {
3.     BodyClip BuildUp;
4.     BodyClip Body;
5.     FlexStory Diversion;
6.     BodyClip Merge;
7.
8.     void Play()
9.     {
10.         BuildUp.Play();
11.         Body.Preload();
12.         Diversion.Preload();
13.     }
14.
15.     void Preload()
16.     {
17.         BuildUp.Preload();
18.     }
19.
20.     void ClipFinishedPlaying()
21.     {
22.         if BuildUp finished playing
23.             if a user chose to take a diversion
24.                 Diversion.Play();
25.                 Merge.Preload();
26.             else
27.                 Body.Play();
28.
29.         else if Diversion finished playing
30.             Merge.Play();
31.
32.         else if Body finished playing
33.             Raise Finished Event
34.
35.         else if Merge finished playing
36.             Raise Finished Event
37.
38.         else if Context finished playing
39.             Diversion.Play();
40.     }

```

Figure 56
Choice class

ClipFinishedPlaying method first gets called when Buildup has finished its playback (Figure 56 line 22). If by this time a viewer has chosen to watch Diversion, Diversion’s playback is initiated and Merge preloads (Figure 56 line 24). Otherwise Body’s playback is initiated.

Listed below are the rest of cases when **ClipFinishedPlaying** gets called:

1. Diversion’s playback is finished, initiate Merge’s playback.

2. Body's playback is finished, viewer did not take a Diversion, Choice's playback is over.
Raise **Finished** event.
3. Merge's playback is finished, raise **Finished** event.
4. Context's playback is finished, that means viewer is watching a deferred diversion.
Initiate Diversion's playback.

Preloading Workflow Example

We will now explain preloading workflow upon a sequence of steps needed to play a Flexible Story shown in Figure 57. Preloading starts upon opening **FlexStory a** (Figure 57). We preload the first scene in the array, namely Pre-Story clip (b). When a viewer initiates a play command **a.Play()** is called. **FlexStory a's Play** method tells a **FlexClip** at index 0 to play itself, and asks scene at index 1, to preload. A **FlexClip** at index 0 (b) is of type **Clip**. Its **Play** method plays the containing video. A scene at index 1 (c) happens to be a Choice. Choice **b's Preload** method preloads Buildup (e) since it is the first video that will play upon Choice's playback.

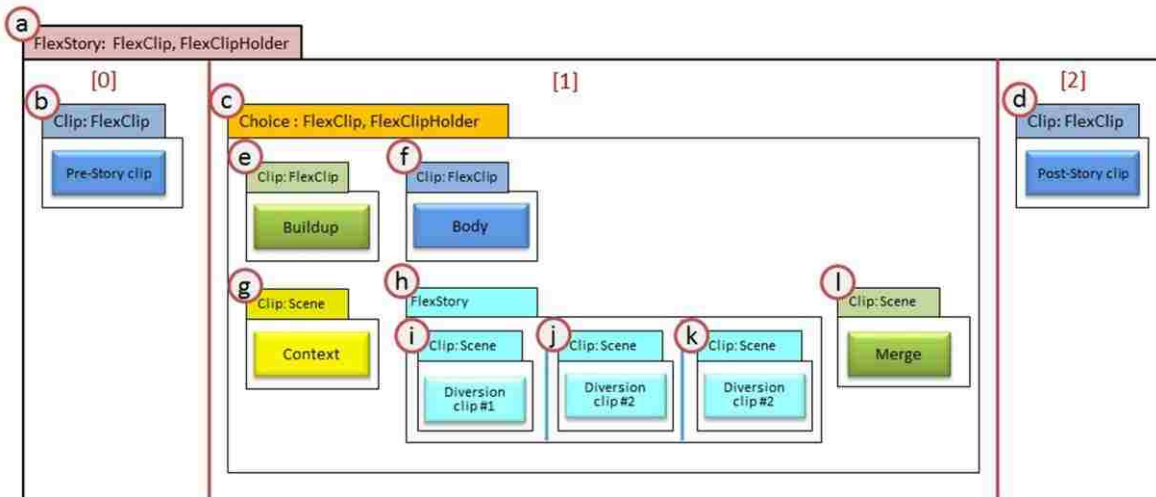


Figure 57
FlexStory Data Structure

As Pre-Story clip, Clip **b**, is finished playing, **a.ClipFinishedPlaying()** is called (see Figure 55 line 24). A **currentClipIdx** is now at 1, therefore **a.Play()** invokes **flexClips[1].Play()** which happens to be of type **Choice** (Figure 57 c). We are treating a Choice as single self-contained video scene therefore we initiate the preload of the next **FlexClip** in a collection, a Post-Story clip (d).

A Choice playback workflow is performed by two methods **Play** (see Figure 56 line 8) and **ClipFinishedPlaying** (see Figure 56 line 20). A **Play** method is only called at the beginning of Choice **c'**s playback. It tells a Buildup (Figure 57 e) to play itself and ask a Body (f) and a Diversion (h) to preload. Since Body **f** is of type **Clip**, its **Preload** method causes the video to load and seek to the beginning. A Diversion **h** which is of type **FlexStory** (see Figure 55) performs pre-loading by asking a first scene in its array of **FlexClips**, namely Diversion clip #1 (i), to preload.

ClipFinishedPlaying method gets called for the first time when Buildup **e** finishes playing. This method from here on directs the rest of playback order. If by the time Buildup **e** is finished a viewer did not choose to watch a Diversion, Body **f** starts playing (see Figure 56 line 26). When Body **e** finishes playing, Choice **c**.**ClipFinishedPlaying()** method raises **Finished** event (see Figure 56 line 33). Finished event causes **a**.**ClipFinishedPlaying()** to be called (see Figure 55 line 24). The next **FlexClip** in line after Choice **c** is a Post-Story clip, Clip **d**. No preloading is needed at this point since Clip **d** was already pre-loaded at the beginning of a Choice playback.

The second option is that a viewer chooses to watch a Diversion. In this case, Choice **c** asks a Diversion **h** to play itself, and Merge **l** to preload (see Figure 56 line 24). Since Diversion is of type **FlexStory** its playback workflow is identical to that of a FlexStory. Diversion **h** asks a first scene in its collection, Diversion clip#1 (Figure 57 i), to play, and a second scene in its collection, Diversion clip#2 (j), to preload. When Diversion **h** finishes playing all of its scenes, Merge **l** plays. When Merge **l** finishes its playback **c**.**ClipFinishedPlaying()** method raises **Finished** event (see Figure 56 line 29). At this time no more pre-loading is needed since Post-Story clip (d) was preloaded at the start of Choice's playback. FlexStory now plays Post-story clip.

Media Players' Resource Pool

The scenes in the Flexible Storylines are preloaded into a resource pool of media players. For our media players we have used Microsoft Smooth Streaming Media Elements. We have chosen Microsoft Smooth Streaming framework for high speed of video streaming and short seek time. Taking into consideration all possible paths through a story, we only need four media players in

the resource pool. We will first explain the workflow of switching between two consecutive non-interactive scenes. Next, we will show why four players is sufficient for our purposes.

SSMHolder (Smooth Streaming Media Elements Holder) class stores an array of four media players. Each Clip that is played calls the **ssmHolder.Preload** (Figure 58 line 15) which preloads this Clip to a free slot in the array of media players. **playerIdx** field has been added to class **Clip** to keep track of preloaded player's index (Figure 58 line 6).

```
1. class Clip implements FlexClip
2. {
3.     URL videoUrl;
4.     long startOffset;
5.     long endOffset;
6.     int playerIdx;
7.
8.     void Play()
9.     {
10.         ssmHolder.PlayIdx(playerIdx);
11.     }
12.
13.     void Preload()
14.     {
15.         ssmHolder.Preload(this);
16.     }
17. }
```

Figure 58
Clip class implementation

Preload of a story shown in Figure 59 starts by clip A's preload (step 1). Clip A asks Resource Pool of Media Players to preload clip A. As preload is complete, Resource Pool allocates player at index 0 to clip A and sets clip A's **playerIdx** to 0 (Figure 59 step 2).

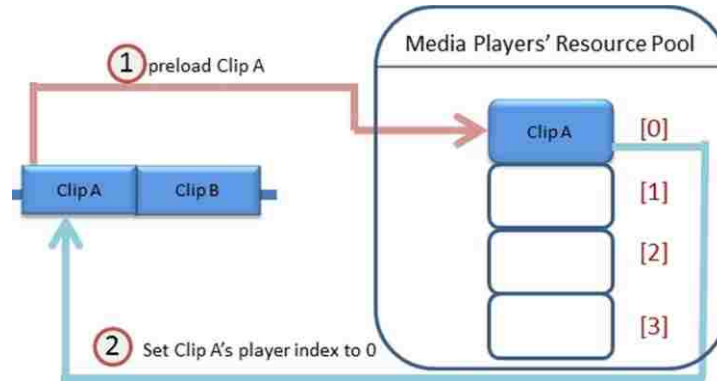


Figure 59
Clip A's preload

When clip A has been preloaded the story starts playing. Clip A tells the Resource Pool to play a player at index 0 (Figure 60 step 1). Resource Pool makes player at index 0 visible and starts its playback. Meanwhile clip B initiates its preload (Figure 60 step 2). Clip B's **playerIdx** gets set to 1 (Figure 60 step 3).

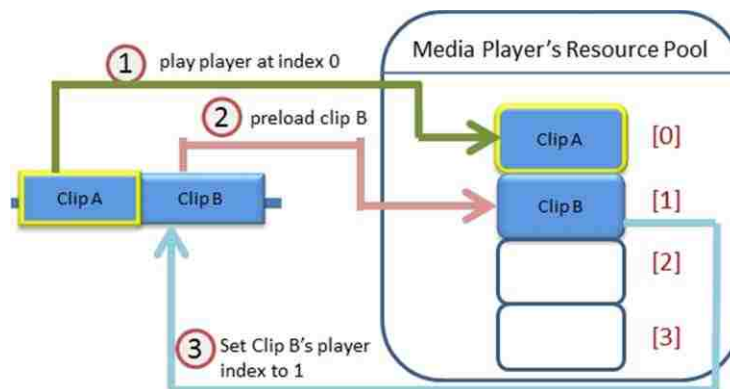


Figure 60
Clip A's playback

Figure 61 shows what happens when a player in the Resource Pool reaches clip A's end offset. The Resource Pool signals to clip A that player at index 0 has finished playing (Figure 61 step 1). Clip B is signaled to start playing and tells the Resource Pool to play player at index 1 (Figure

61 step 2). The Resource Pool stops player at index 0, and starts player at index 1. Next, clip A releases player at index 0 from the Resource Pool (step 3).

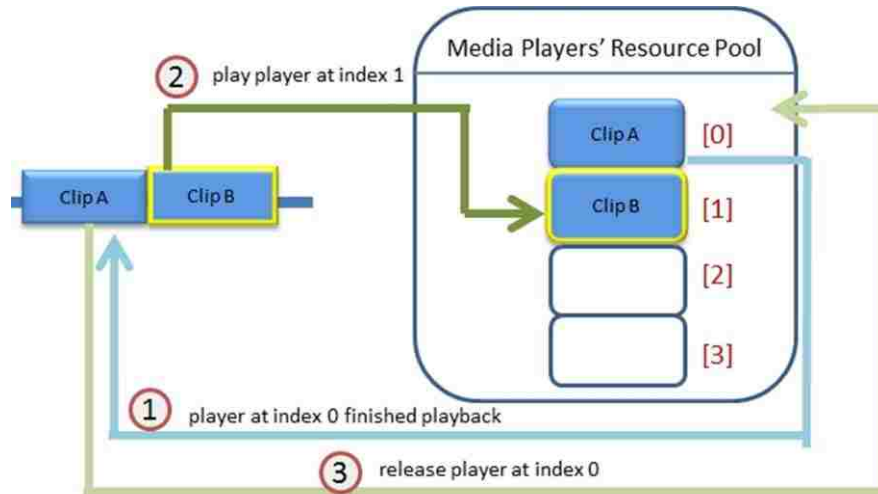


Figure 61
Clip B's playback

Although this setup worked great, we did run into a problem with one specific scenario. Two clips, A and B, are from the same video source (see Figure 62). Clip A's end offset is 10 and clip B's starting offset is 10. When clip A and B are placed in sequence on a timeline, they should play as a single video clip with start offset of 0 and end offset of 20. We saw that clip A's playback ended slightly after its end offset. In order to provide seamlessness the Resource Pool plays a player associated with clip A right until the point when it switches over to player for clip B. There is a slight delay between Resource pool detecting clip A's end offset and beginning clip B's playback. This created a small overlap between the end of clip A and the start of clip B. Overlap in the playback introduces a small but unpleasant glitch. We have fixed this problem by adding functionality to the Media Players' Resource Pool. It was set up to detect if the next clip in sequence starts where the previous clip left off. In this case Resource pool does not switch from currently playing player and continues its playback. Clip B's **playerIdx** now gets

assigned to clip A's `playerIdx`. This additional functionality fixed the problem of seamless playback for this scenario.

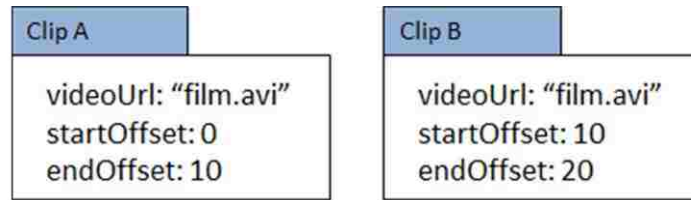


Figure 62
Clip A's and clip B's fields

We will now examine all possible paths through a flexible story that does include interactive scenes. Examining all paths will show the maximum amount of media players that need to be stored in the Resource Pool.

Figure 63 shows a playback of the first element in a story, namely Pre-Story clip (step 1).

Resource pool currently stores a Pre-story clip for current playback, and a preloaded Buildup (step 2). Total count of players is equal to two.

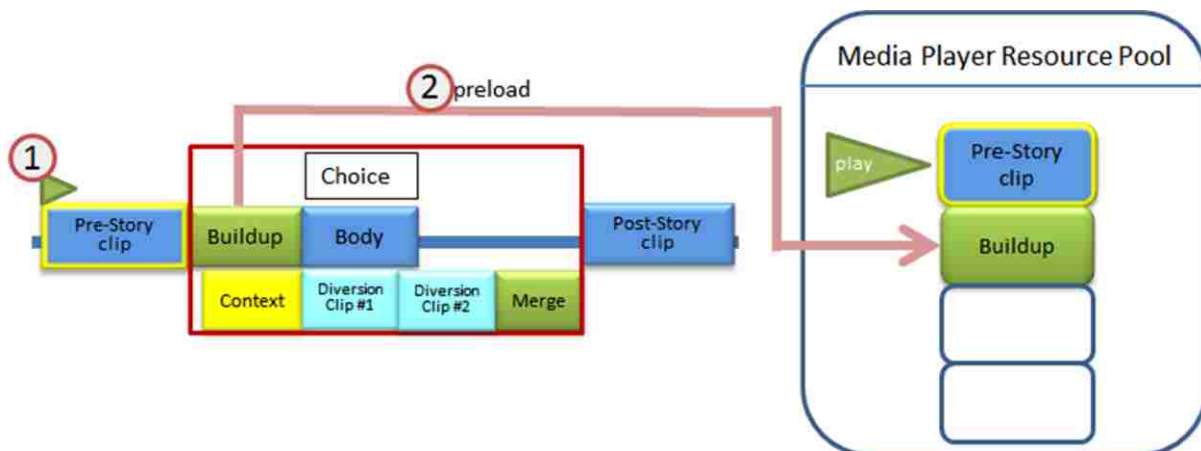


Figure 63
Playback of Pre-Story clip

Figure 64 shows the first step of playback of a Choice (step 1). Pre-Story clip was released (step 2) then Post-Story clip, as the next scene after Choice, was preloaded (step 3). Buildup started to play (step 4), then Body and a Diversion Clip #1 was preloaded (step 5 and 6). Total count of players is equal to four.

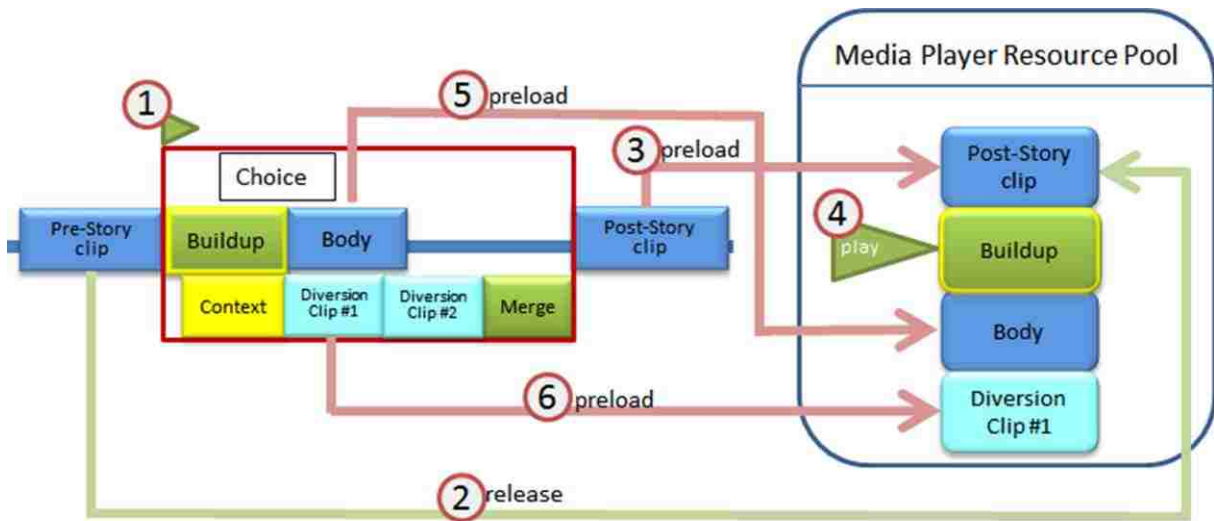


Figure 64
Playback 1 of Choice: Buildup

We will examine the count for media players in the Resource pool for both paths that can follow a Buildup. We will first examine a path when viewer takes no choice and will watch a Body.

Figure 65 shows a playback of a Body. First, a Buildup gets released (step 1). Second Body starts playing (step 2). Since we no longer need a Diversion, Diversion Clip #1 get released (step 3).

Total count of players is equal to two.

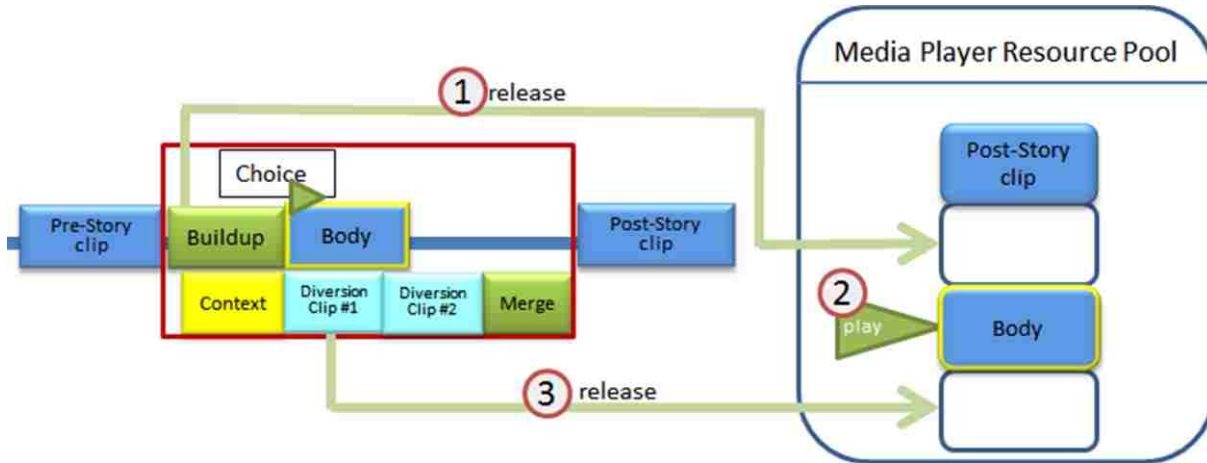


Figure 65
Playback of a Body

Figure 66 shows a playback of a Post-story clip. First, a Body gets released (step 1). Then Post-Story clip starts playing (step 2). Total count of players is equal to one.

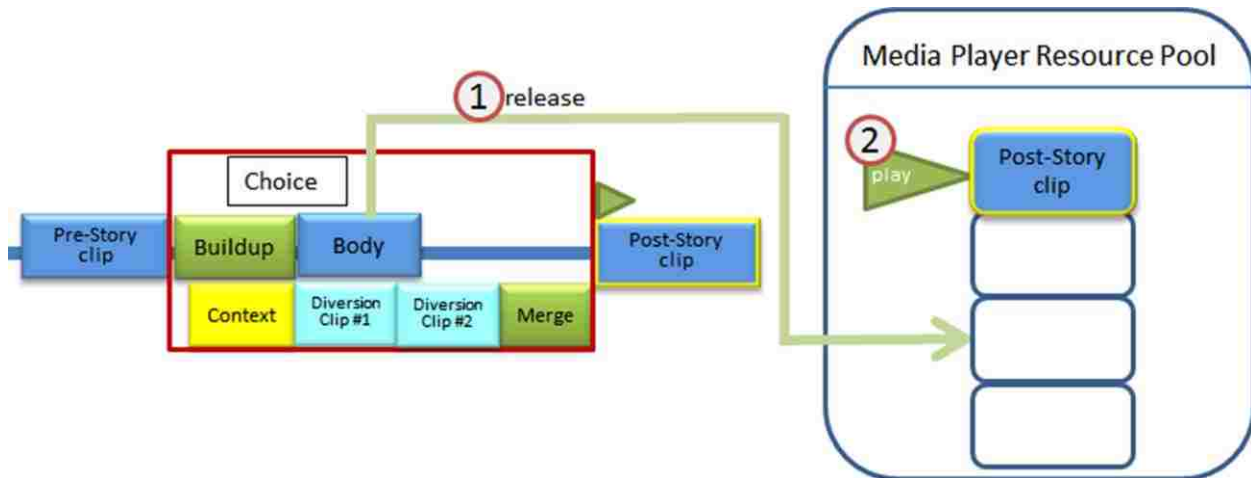


Figure 66
Playback of a Post-Story clip

We will now examine the second path of a story, when a viewer chooses to watch a Diversion. Figure 67 shows a playback of Diversion Clip #1. Although a Buildup and a Body were released (steps 1 and 2), their spots were taken by Diversion Clip#2, and a Merge (steps 4 and 5). Notice

that we still need to hold on to Post-Story clip, since it will be played after a Choice is finished playing. Total count of players is equal to four.

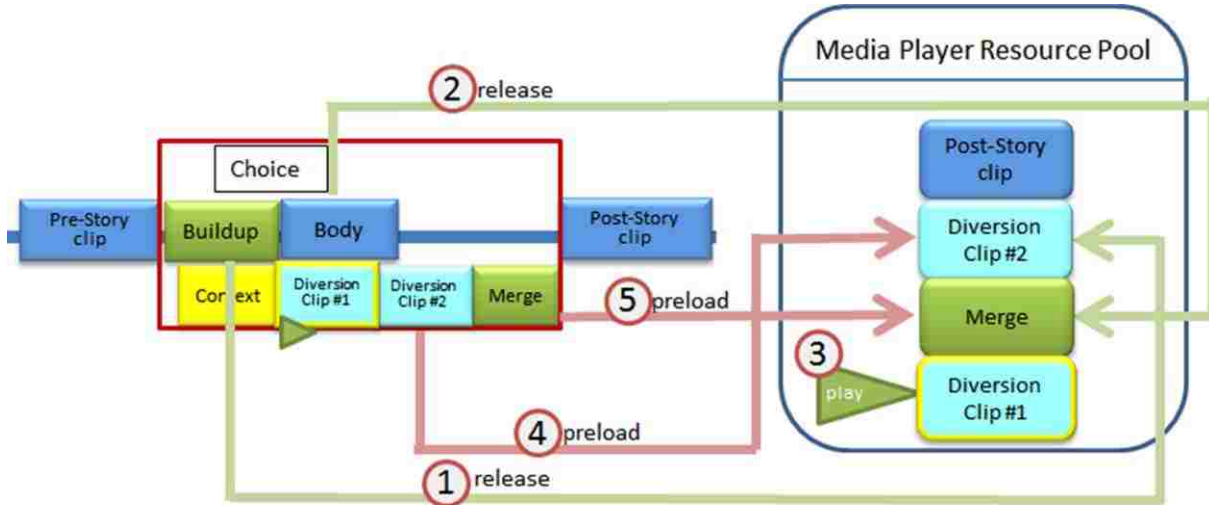


Figure 67
Playback of a Diversion Clip #1

Figure 68 shows a playback of Diversion Clip #2. Diversion Clip #1 was released (step 1), which leaves us to a total count of three players.

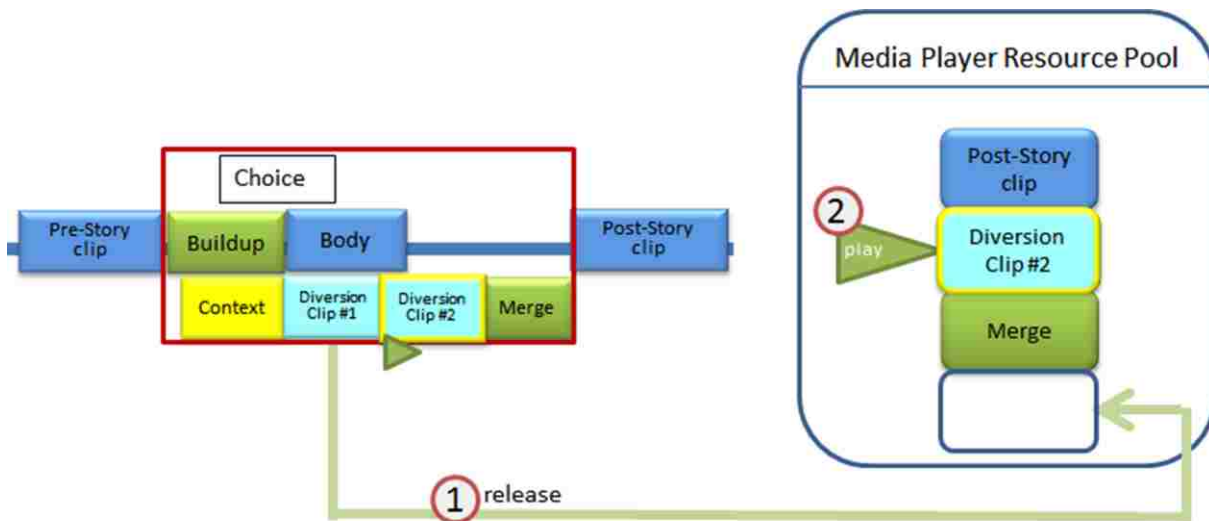


Figure 68
Playback of Diversion Clip #2

Figure 69 shows a playback of Merge. Diversion Clip #2 was released (step 1) leaving a total count of two players used.

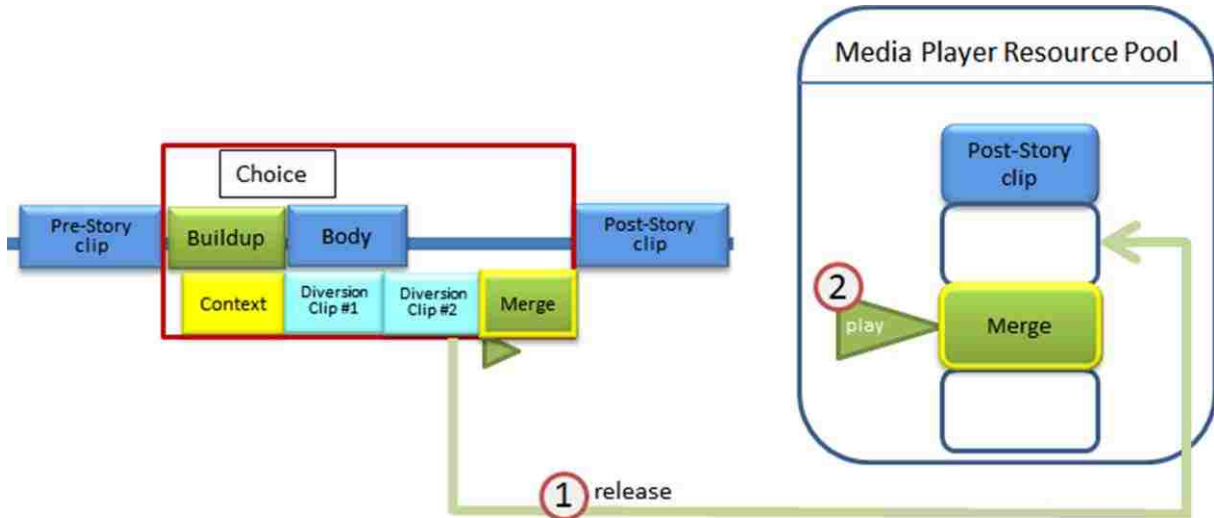


Figure 69
Playback of Merge

Figure 70 shows a playback of Post-Story clip after a Merge. First a Merge was released (step 1), then Post-Story clip starts its playback (step 2). Total count of players is equal to one.

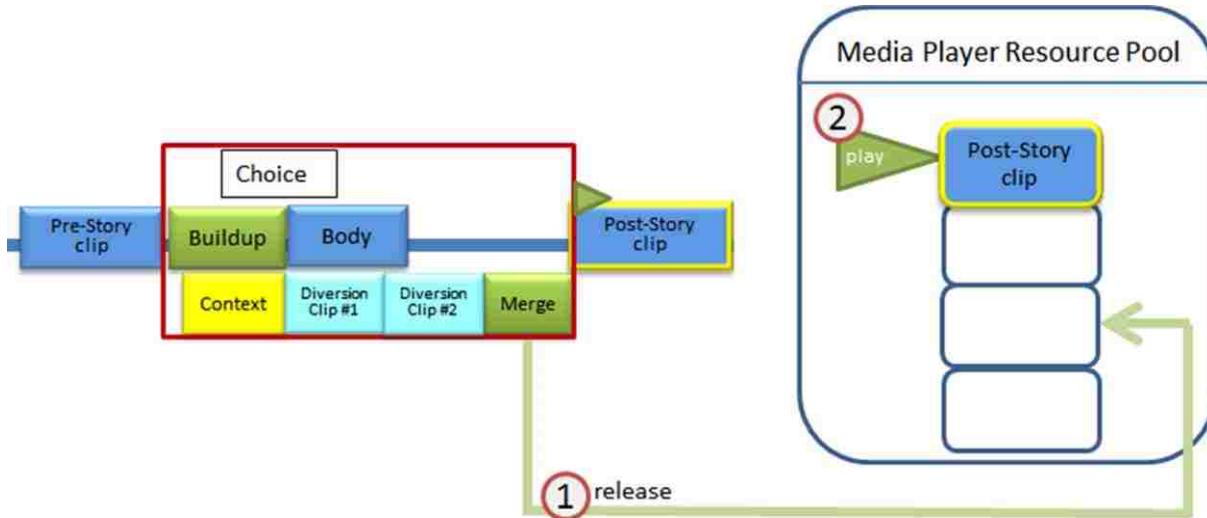


Figure 70
Playback of Post-Story clip

Figure 63 through Figure 70 show all possible paths through a Flexible Story except for a path of deferred Diversion. In that case only two players at a time will be used: one for Context, and the second for Diversion Clip #1. The maximum number of media players used was four as seen during playback of a Buildup (Figure 64) and a Diversion Clip #1 (Figure 67). Therefore we only need a total of four media players in our Resource pool.

Conclusion

Flexible Storylines is a web-based application that required playing of multiple video sources back to back over the network. This can be compared to delivering a playlist of videos over the network. This setup offers a set of time constraints for video delivery, such as time to load and buffer. We were able to overcome these constraints by using preloading architecture and a resource pool of four media players.

The validation for seamless playback was first done by us. We created a Flexible Story and have exercised all possible cases of transitions. In all cases, while playing over the network, there

were no interruptions or delays between distinct videos. In addition we have performed a user study with six participants. Participants were asked to watch a Flexible Story over the network and exercise different paths in a story. All six participants did not observe any interruption or delays while switching between different videos. Having validated our results, we conclude that we were successful in delivering seamless playback.

Chapter 6 - Viewing experience

This chapter will discuss the challenges and solutions of viewing experience. Our goal was to provide intuitive and simple interaction to viewers. We have outlined a set of challenges that are part of viewing experience. First four were defined as the viewer needs in the introduction.

1. Adaptability of the story
2. Smooth viewing experience
3. Flow and structure preservation
4. Perceived safety

5. Watching a deferred Diversion later
6. Balancing the amount of interactive scenes
7. Length of Buildups and Diversions.

The first challenge, adaptability of the story, is to allow viewers to be in control of portions of the story. The second challenge is smooth viewing experience. Viewing experience is smooth when the playback is seamless, without stops, even at the interactive transitions. Third challenge is preserving the flow and structure of the story while allowing viewers' interaction. Fourth challenge, perceived safety, means that viewers need to feel safe making a choice that will alter the flow of the story. The fifth challenge is caused by allowing viewers to defer watching Diversion. When they watch it later, it may appear out of context for viewers. Sixth challenge is the balanced amount of interactive scenes. If interactivity is offered too frequently it may become a burden for viewers. Seventh and final challenge is the length of Buildups and Diversions. For example, having too short of a Buildup, may not give viewers enough time to make a choice. Similarly including diversions that are too long may cause viewers to constantly

avoid them by either not making a choice or exiting out early. The structure of Flexible Storylines paired with a simple interface helped satisfy first five challenges. The last two challenges are to be solved by story creators.

We will first review how the structure of Flexible Storylines satisfies the needs of story viewers and solves viewing experience challenges. Next we will explain how this structure is exposed through the user interface of Flexible Storylines. We will conclude this chapter with a description of validation process and its results.

Flexible Storylines structure

The first challenge is adaptability of the story. It is solved by offering viewer a choice to watch a Diversion (see Figure 71). A Diversion offers an alternative, usually longer version of Body.

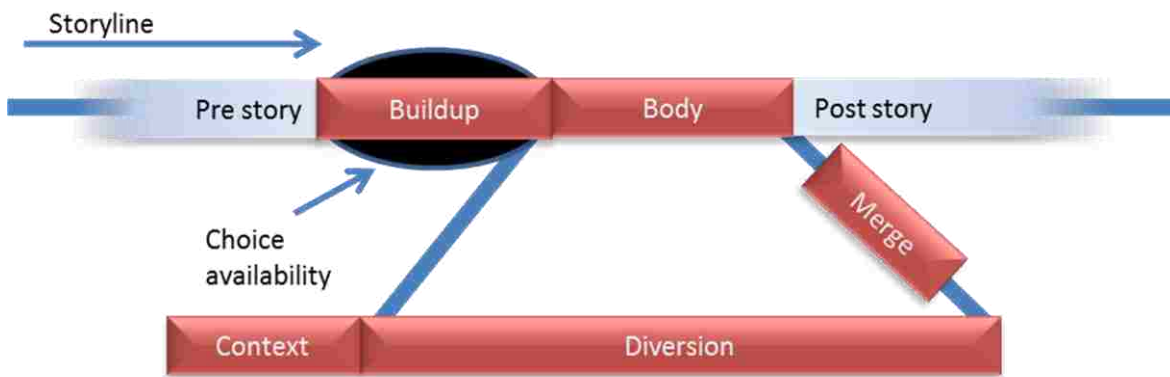


Figure 71
Flexible Storylines Structure

The second challenge is smoothness of viewing experience. The choice is available through the duration of Buildup, making it unnecessary to stop the story for user to make a choice. This ensures smooth viewing experience without interruptions. In addition, if a viewer chooses to watch Diversion, it will start playing after Buildup is finished. If we navigated to Diversion immediately upon viewer's choice, it would create a browser-like experience.

The third challenge is flow and structure preservation. The flow and structure of a story is preserved by bringing each Diversion back to the main storyline after its end. The transition into the main storyline is made smooth through Merge (see Figure 71).

The fourth challenge is perceived safety. As a viewer enters Diversion, he or she is able to exit out it. In case of early exit out of Diversion, video story plays Merge before returning to the main storyline (see Figure 71). First, Merge will ensure smooth transition back to the main storyline. Second, Merge will guarantee that no vital information was missed by exiting Diversion early.

The fifth challenge is that Diversion may appear out of context if deferred and watched later. We solve this by providing Context as a part of a Flexible Storylines structure (see Figure 71).

Context will only be played before a deferred diversion. Its sole purpose is to summarize the Pre-story bringing a diversion back into context.

We give story creators tools and freedom to create flexible stories. We put no restrictions as to amount of interactive scenes that story creators can insert. Similarly no restrictions are placed upon the length of Buildups and Diversions. Therefore, the avoidance of the last two challenges is under the responsibility of story creators.

As story creators produce flexible stories they need to ensure that there is a balanced amount of interactive scenes. This amount may vary with the length and nature of a story. We expect a longer story to allow for more interactive scenes. We also expect a documentary or training video allow for more interaction than a fiction film. The length of Buildups and Diversions is also under the discretion of story creators. Story creators need to ensure that Buildups are long enough for a viewer to make a choice. Next, story creators need to make sure that Diversions' length does not become a burden to viewers.

User interface

Because of novelty of concept of Flexible Storylines, our interface had to be simple and intuitive.

Figure 72 shows the screen shot of our player upon opening of a flexible story. The story is currently paused as shown by Paused Indicator in Figure 72. Once playing the non-interactive portions of a video story our player offers three actions to viewers: play, pause and seek.

Playback is initiated by clicking anywhere on the screen. Same click on the screen, while story is playing, pauses a story. Seeking within a video is done by dragging the timeline tracker on a timeline (see Figure 72). The video is shown in a playback area, and the timeline tracker shows story's progress on a timeline. Time counter shown in the bottom right corner of Figure 72 shows the current time of a video's playback, with a total time. Viewer can open a different story through a File menu.

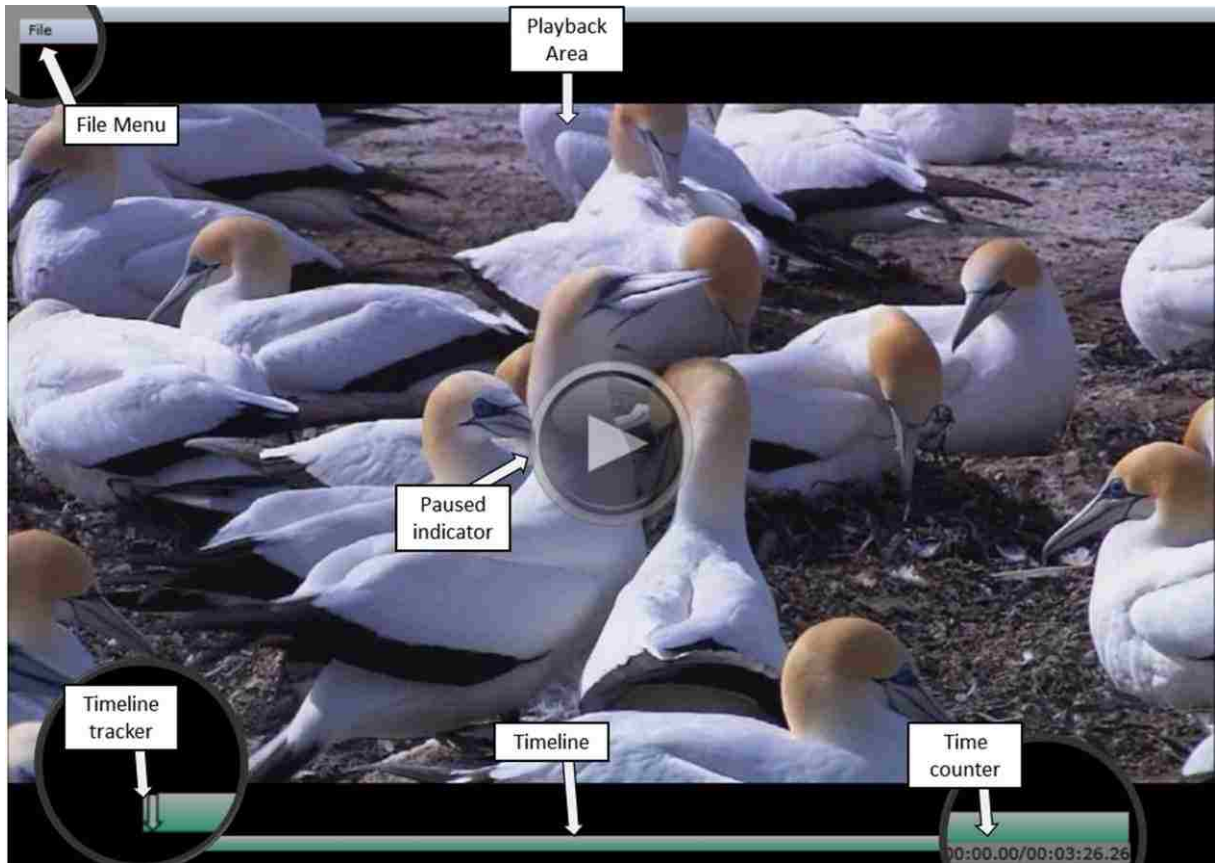


Figure 72
Flexible Player Interface

The first video in the interactive scene is a Buildup. During the duration of a Buildup a viewer is offered a way to adapt the story. This satisfies the first need of story viewers. When Buildup starts its playback, two buttons appear on the left side of the screen (see Figure 73). “Take a diversion” button offers viewers a choice to watch Diversion. It will include a prompt with an enticing message that will describe the contents of Diversion. For example in Figure 73 a buildup is offering viewers a choice to learn more about koala-bears. Diversion will only play after Buildup is finished. This provides smoothness of viewing experience. “Defer a diversion” button gives viewers an option to defer watching this Diversion until later (Figure 73).

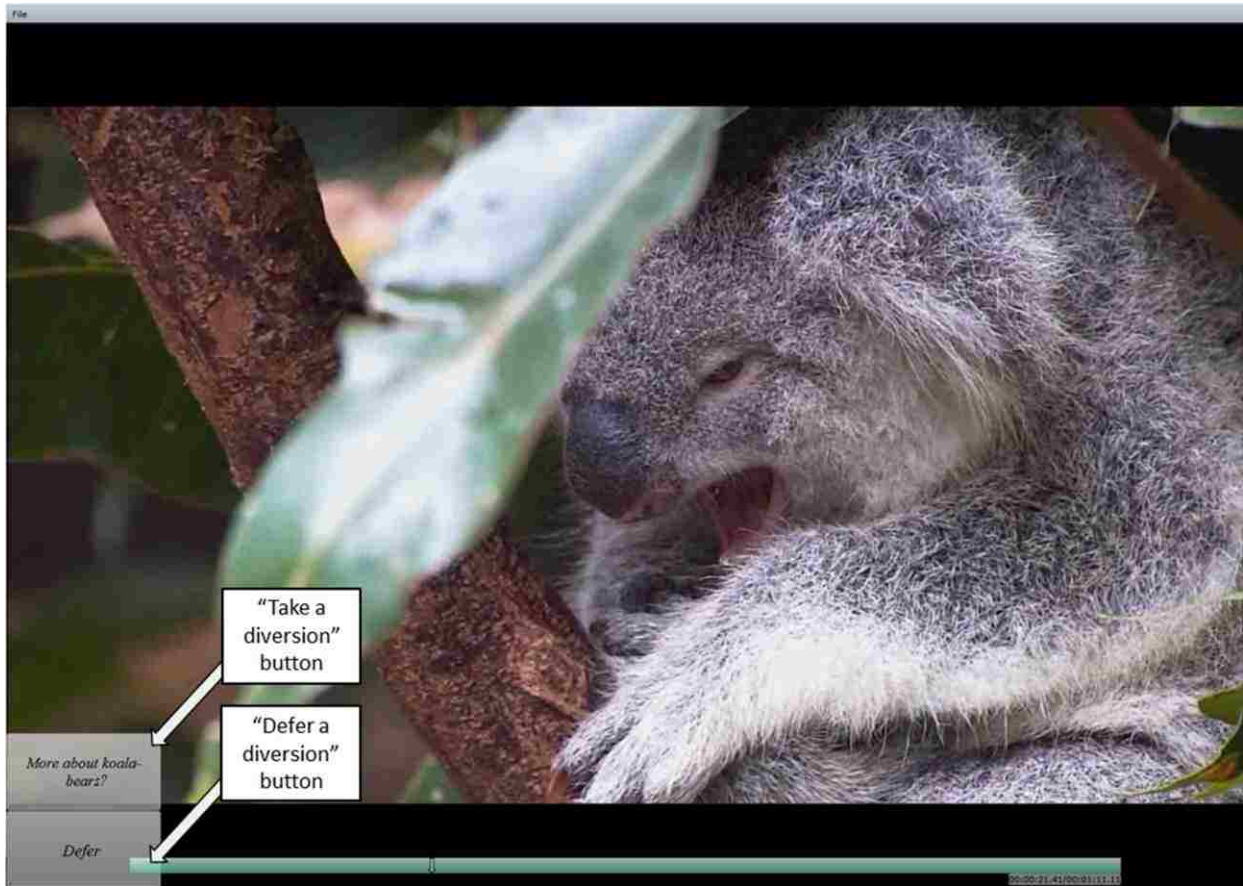


Figure 73
Interface during Buildup's playback

When a viewer presses “Defer a diversion” button, both buttons disappear, and Deferred Diversions notification label shows (see Figure 74). This is to signify that there are deferred diversions that are available for watching. If a viewer performs a right-click on the label, a Deferred diversions context menu show. A viewer can expand the label to show the diversions that have been deferred (see Deferred diversions expanded in Figure 74). Expanded Deferred diversions box allows a viewer to select a diversion, by its prompt. A viewer will be able to load and watch a selected diversion by pressing “Load Deferred” button (Figure 74). “Return to Main” button will become enabled as a viewer starts watching a Diversion, and can be used to load back the main story.

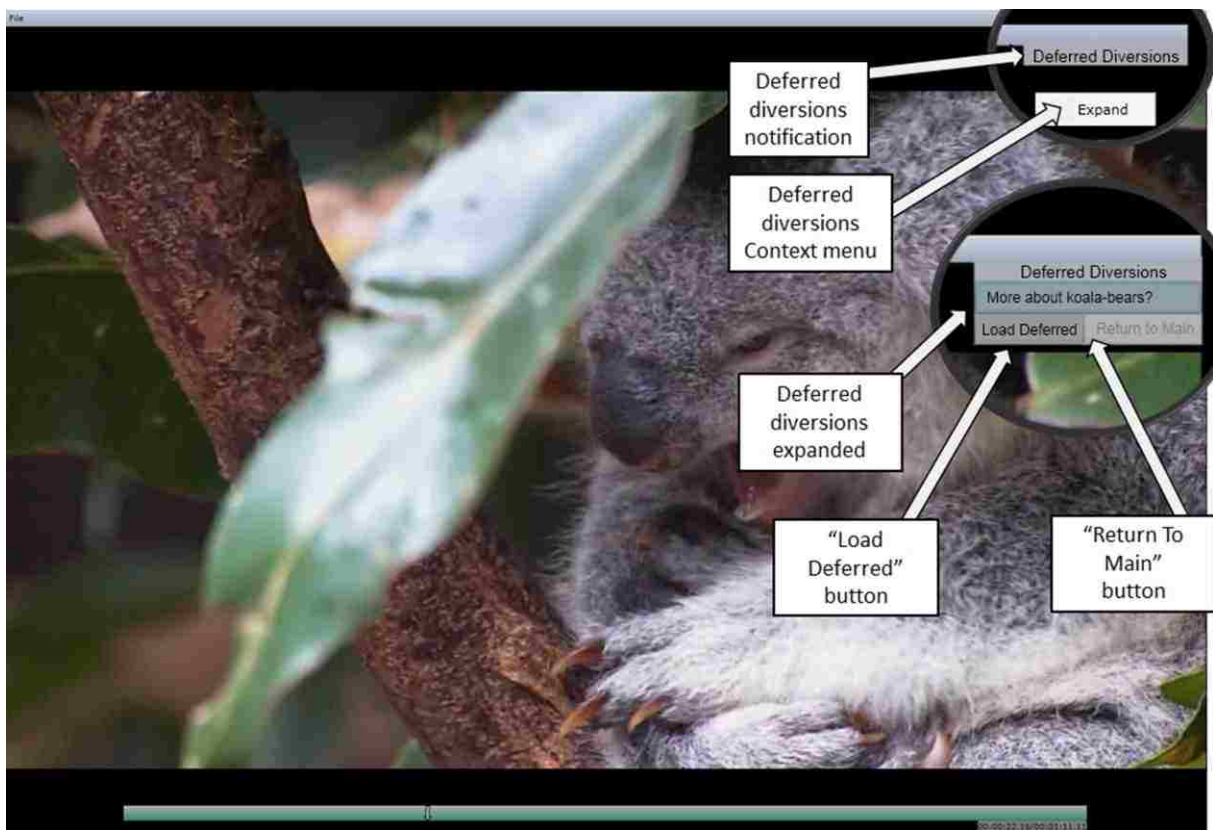


Figure 74
Viewer pressed Defer Button

Looking at the second case, a viewer presses “Take a diversion” button (Figure 73). Immediately after Buildup is done playing, Diversion starts to play. The timeline now represents the length of Diversion’s story plus the length of Merge (Figure 75). “Exit Diversion” button appears on the screen (see Figure 75). Upon pressing this button, a viewer is taken to Merge. This ensures that viewers will feel safe taking Diversions, since there is a way to exit. When Merge finishes its playback, Post-story starts playing and the timeline is updated to represent main story’s timeline. Notice how flow and structure of a story is preserved by bringing viewers back to the main story after each Diversion.

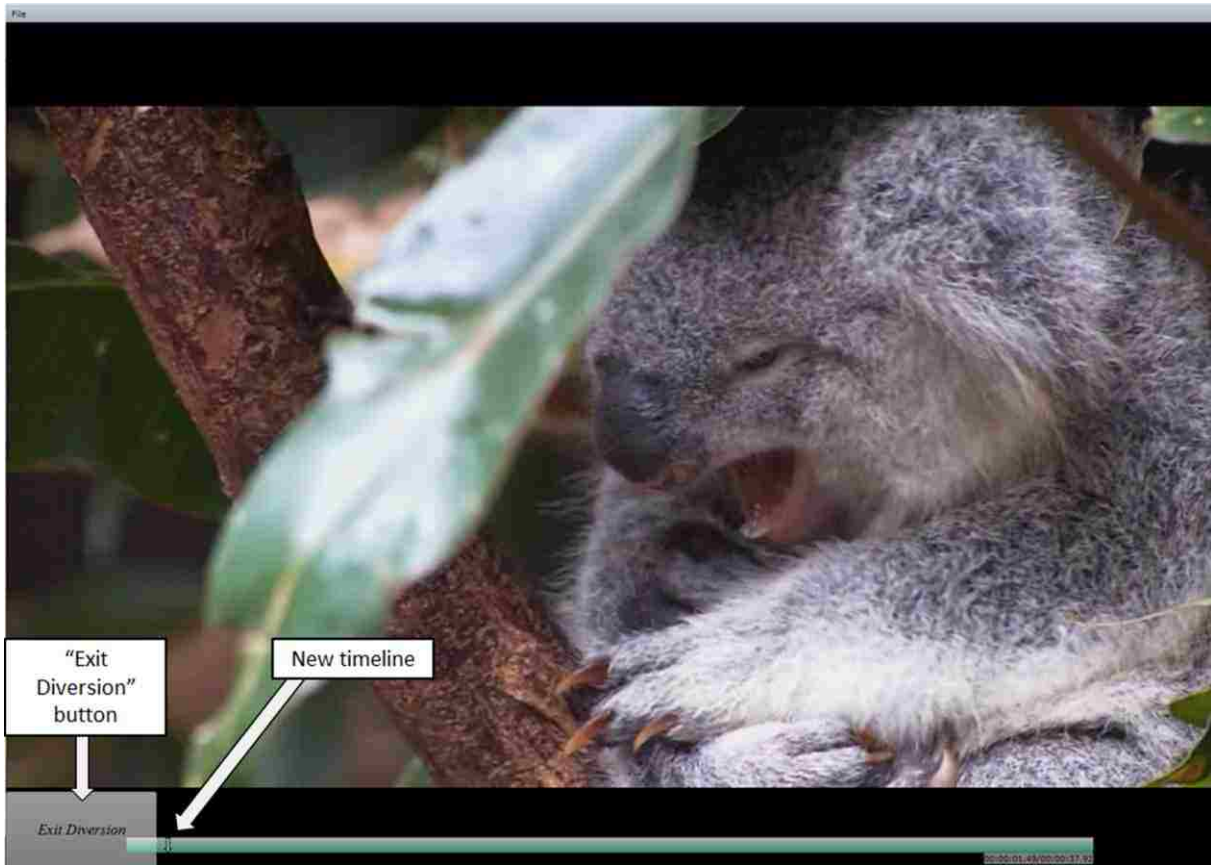


Figure 75
Diversion's playback

Validation

We have validated the quality of the viewing experience, through a user study. Study participants were asked to watch a flexible story. We have built a flexible story that was done in Russian language. The demographics of study participants consisted of four females and two males, ages 23-30. All participants were Russian speakers.

To each participant we explained the concept of Flexible Storylines and showed a demonstration of the Flexible Player. Next we asked them to watch a 15-minute flexible story. The flexible story was built using source video of a cartoon that participants were familiar with. The

familiarity with a cartoon helped participants see a contrast between non-interactive and interactive versions of a story.

We recorded each participant's task execution. Also in addition, our player logged interaction actions performed by participants. At the conclusion of the task, we asked participants to describe their experience (good and bad) and recorded each response. We have gathered and organized user experience data from recordings of task executions and participant's responses. User experience data was organized into two groups: negative feedback and positive feedback. We will first describe the statistics on participant's interaction. Next we will cover the negative feedback, and finally end the positive feedback.

Interaction statistics

We have offered a total of three interactive scenes to viewers. Figure 76 is a graph showing the number of people who took Diversions and the percentage of diversions that they took. Three participants have taken 66.67% of diversions offered to them. Two participants took all the diversions that were offered to them. Finally only one participant did not take any diversions at all. This tells us that the majority of participants (83%) chose to adapt a story to their interest.

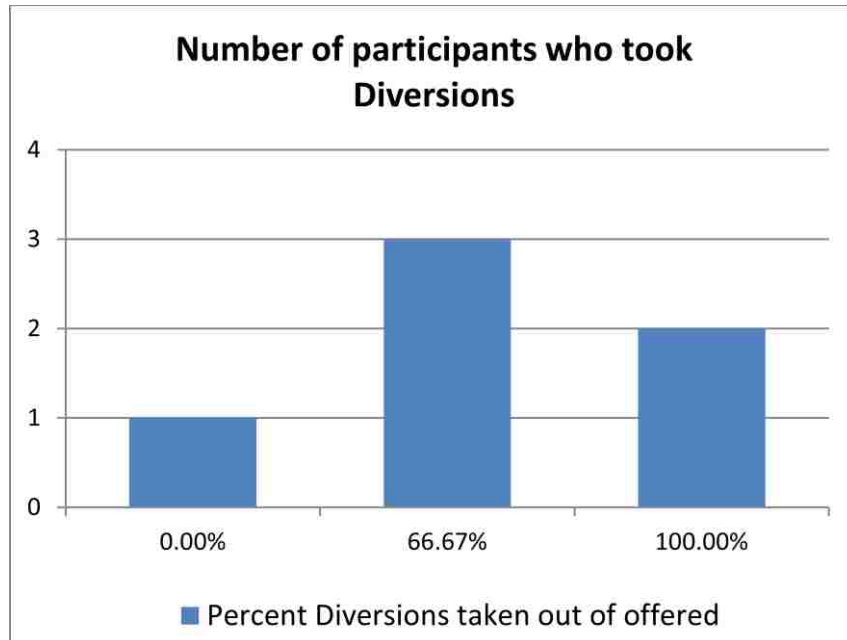


Figure 76
Participants who took Diversions

Figure 77 shows how the number of participants who chose to exit a diversion along with the percentage of diversions exited. The percentage of diversions was calculated based on the amount of diversions that a participant took. For example if participant have exited 1 out of 2 diversions that he took, he or she has exited 50% of diversions. In Figure 77 we see that 2 people exited every diversion that they took. The other two participants have exited 50% of diversions that they took. Finally one person has watched all the diversions taken until completion. From this result we can conclude that a large portion of diversions was not interesting enough to the majority of our participants.

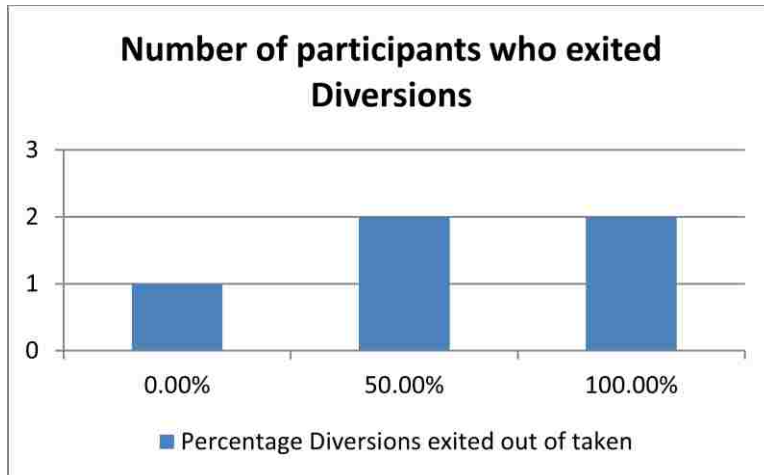


Figure 77
Participants who exited Diversions

Figure 78 shows the number of participants who deferred Diversions and percentage of diversions that they deferred. One participant chose not to defer any diversions. On the other hand, five participants have chosen to defer one diversion out of three offered. From this data we can conclude that five out of six participants (83%) were interested in a Diversion but did not want to watch it immediately.

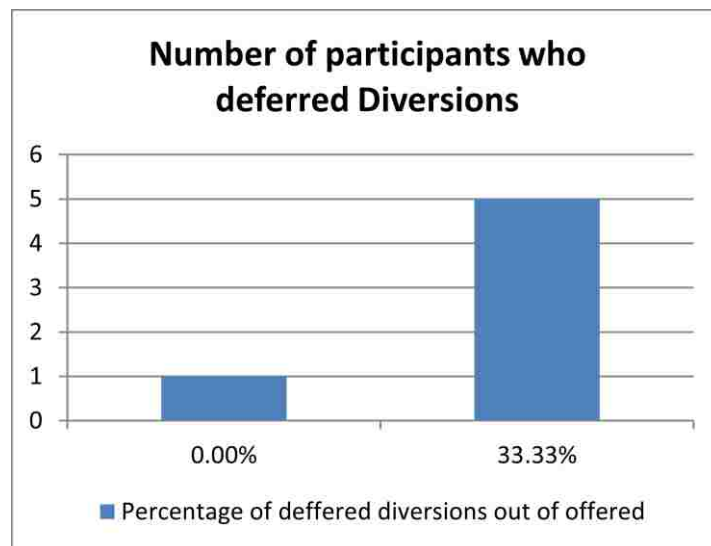


Figure 78
Participants who deferred Diversions

It is difficult to draw specific conclusions about viewers' preference and behavior based on this data. Nevertheless, we saw that five out of six participants (83%) took at least two Diversions out of three offered and deferred one. We consider this behavior to be active interaction. This shows that five out of six participants felt safe interacting with our system and knew how to do it.

User Feedback

We will first cover the negative feedback, and then proceed to cover the positive feedback. From the point of User Interface design all six participants offered suggestions for improvement. Three out of six participants (50%) found that "Take a diversion" and "Defer a diversion" shown in (Figure 79) were not subtle enough. It was suggested to show them, and make them more transparent after a few seconds. Once a viewer chooses to read the contents of buttons he or she would hover over them and they would be made more opaque. This is a good suggestion, but depends highly upon viewers' preference. A viewer may want to read the contents of the buttons without having to hover over them. One more participant has suggested that after a viewer chooses to take a diversion and presses the button, both buttons disappear. This also may vary with a preference of a viewer. "Take a diversion" button is a toggle button, and allows viewer to toggle it through the time of Buildup playback. We have designed the interface this way so that a viewer who can change his or her choice before the end of Buildup.

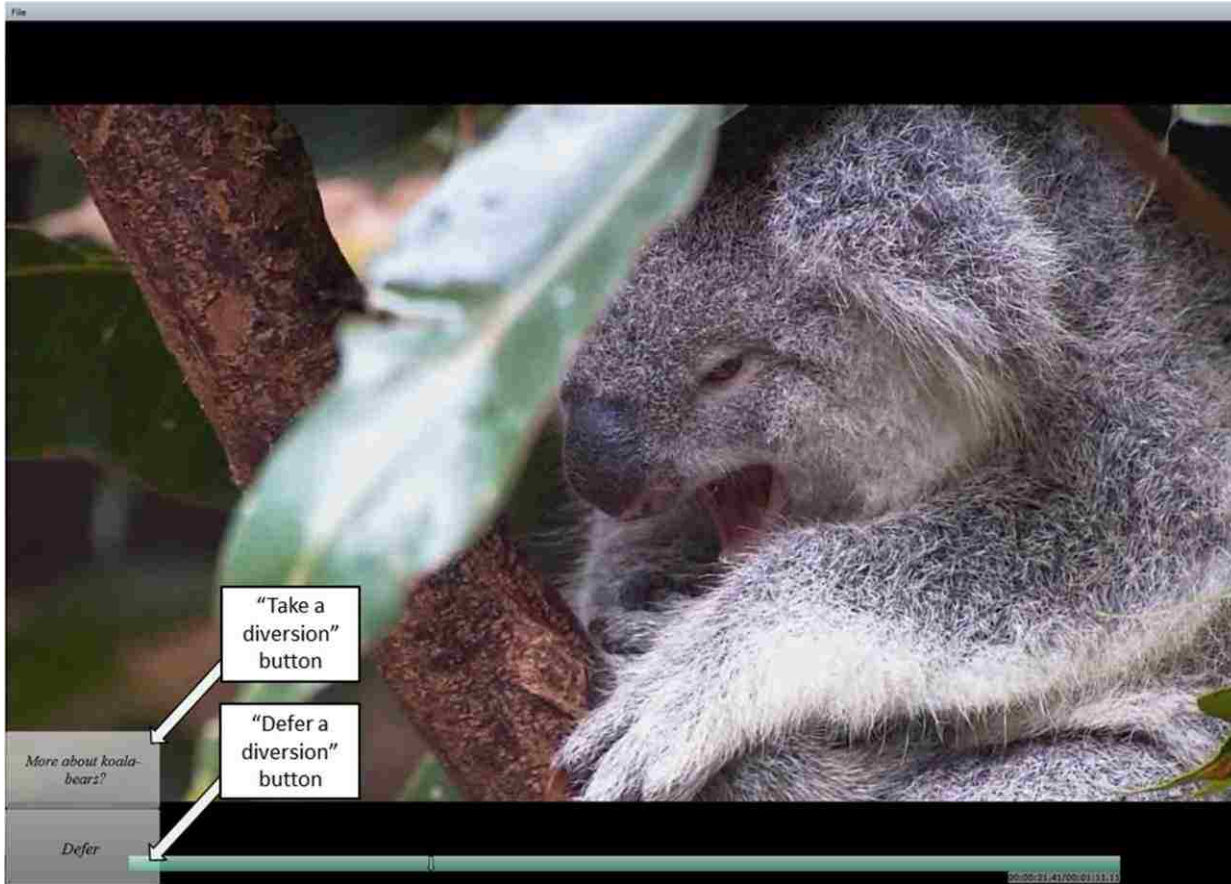


Figure 79
Player's Interface during Buildup

Another participant has asked to move Deferred diversions interactive component (Figure 80) to the bottom left corner of a player. The argument was that other interactive buttons, such as “Take a diversion” and “Defer a diversion” already appear in the bottom left corner. It does make sense to gather all the interactive components into one spot on the screen. However, we believe that interactive component of deferred diversions will be used less frequently, and should be moved to a separate part of the screen.

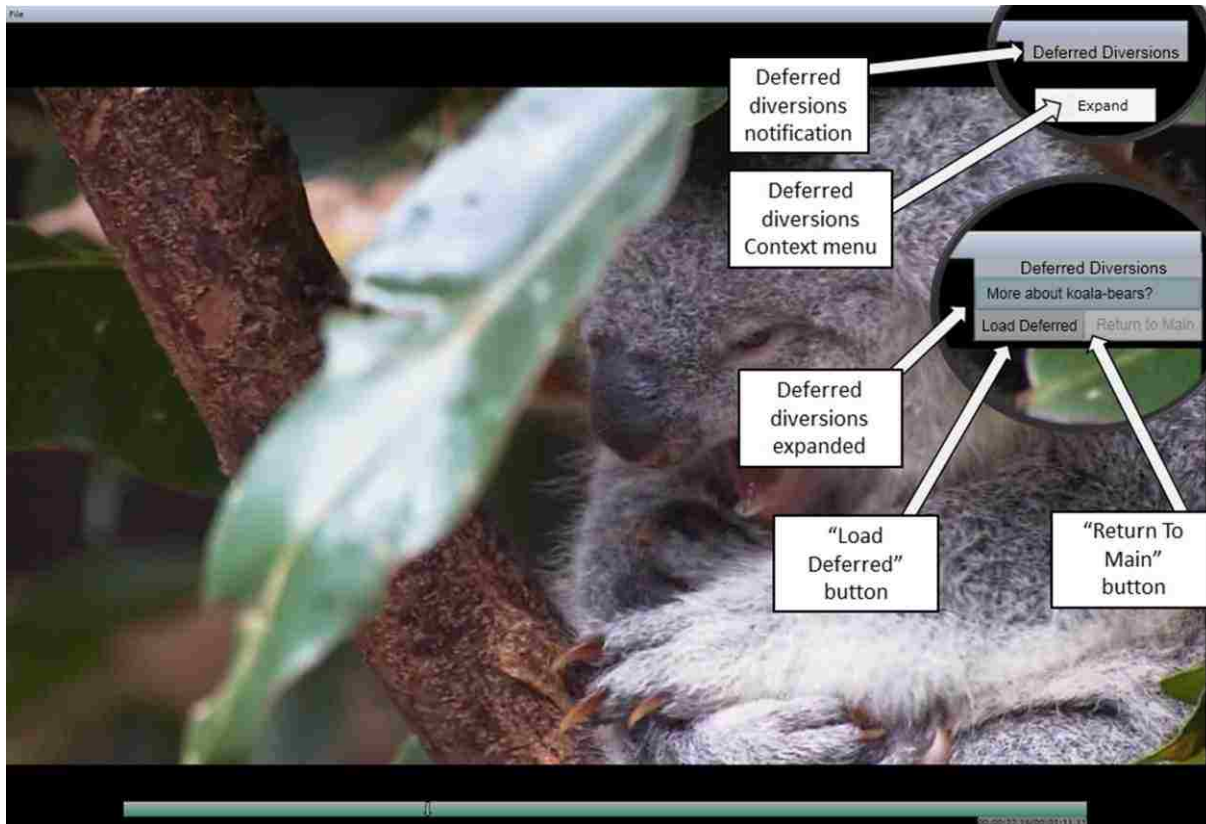


Figure 80
Viewer presses Defer a diversion button

Two participants found problems with the Timeline (Figure 81). One participant has asked to allow viewers to hide a timeline. The moving Timeline tracker was viewed as a potential distraction to some viewers. Although we have this feature implemented we failed to explain it to this participant before task execution. One participant has found that the Time counter was broken. This is an implementation error that can be easily fixed.



Figure 81
Flexible Player interface

There was also negative feedback as to the way story was created. One participant has found that one of Buildups was too short. The problem was that this participant wanted more time to think and the button disappeared. Another participant asked for “Take a diversion” button text to be made shorter. This will require less time to read it. Both of these comments require no change to our interface. On the other hand, they are important for story creators to keep in mind during planning of a story.

The last portion of negative feedback is associated with a conceptual understanding of Flexible Storylines. Three out of six participants (50%) performed the following actions during an interactive scene:

1. Defer a diversion
2. Immediately after interactive scene passed, load and play a deferred scene
3. Return to main story

Returning to main story caused the story to start from beginning. These participants found burdensome to seek back to their position in a story. This problem came from lack of understanding for a purpose of a deferred diversion. Deferred diversion was to be watched after a viewer has finished with a story. In contrast these participants wanted to see deferred diversion during the playback of the main story.

This issue occurred from our lack of understanding of viewers' needs. One of the needs of story viewers, as defined in the Introduction chapter is perceived safety. We did provide perceived safety for taking a Diversion. Whenever viewers have chosen to take a Diversion they were able to undo their action and exit it. This result showed that viewers also needed perceived safety for not taking a Diversion. This means that when a viewer decides to skip a Diversion and watch a Body, they want a way to undo their choice and watch that skipped Diversion if they change their mind. These participants deferred a Diversion because they wanted to watch a Body, and see if it covers enough information. If watching a Body was not enough for them they wanted to see what was included in a Diversion immediately. There are two ways that we can fix this issue. First is to allow viewers to take a Diversion while a Body is playing. In that case if a viewer presses "Take a Diversion" button during Body's playback he or she will be navigated there immediately. Another way is to change the behavior of returning to main story. Instead of starting the story from the beginning, we will continue the story from the point where a viewer chose to watch a deferred Diversion.

There were also three feature requests from participants. One participant has asked that we would make interactive scenes appear visible on a timeline. The argument was that if a viewer would like to seek to the choice point, he or she would be able to do it quickly. We see that this can be very useful to some viewers. A second participant has asked us to include the time duration of a Diversion as a part of "Take a Diversion" button text. This is also a valuable suggestion since it provides additional information for viewers to make a choice. A third participant expressed that there can be a time when a viewer wants to relax during a viewing process. This participant asked us to provide a way to disable interactive controls. This is a great suggestion that will accommodate more viewers.

We will now describe participants' positive feedback. All six participants expressed that they deem Flexible Storylines to be very valuable for viewers. Each has expressed that being able to adapt the story to personal schedule is very useful. For example when a viewer is on a tight schedule he or she would prefer not taking diversion. If they do chose to take a diversion, they could exit it after they get an idea of what kind of material it offers. Also, it was expressed that Flexible Storylines lets you adapt the video story to your level of familiarity with a story. For example, if a viewer is watching a story for the first time, he or she may want to take every diversion. When they are watching the story for the second time the viewing experience may be changed. They may want to skip certain diversions, and defer or watch the ones they want to see again. All six participants reported increased satisfaction in viewing experience from being able to adapt the story. While adaptability was present, every participant said that the playback was very smooth and seamless.

All six participants expressed that Flexible Storylines preserved the flow and structure of a story, regardless of a route they took. As one participant said: "Flow and structure of a story were great every time I exited a diversion. Main moments were not lost, but unimportant nuances were omitted". Finally, Flexible Player was characterized as being user friendly and simple to use. Each participant said that they had no problem interacting with Flexible Storylines player.

Conclusion

Flexible Storylines allows viewers' to adapt video story to their needs. The structure of Flexible Storylines had provided the means to meet story viewers needs explained in the introduction:

1. Adaptability of the story
2. Smooth viewing experience

3. Flow and structure preservation
4. Perceived safety

Adaptation of a story is a novel concept for story viewers. Considering this novelty, we have made a simple and intuitive user interface for watching flexible stories. During the playback of interactive portions Flexible Player offered additional interactive controls to viewers.

User study has shown that five out of six participants (83%) were actively interacting with a story. They took diversions, deferred and exited, despite of using Flexible Player for the first time. Most of the negative feedback was motivated by viewers' personal preferences. Through positive feedback we found out that all six participants experienced greater satisfaction from being able to adapt the story. Each participant thought that the flow and structure of the story was preserved despite interactivity. Finally, the viewing process was smooth and seamless for all six participants. Having validated our solution we conclude that we were successful in satisfying the needs of viewers.

Chapter 7 - Conclusion

Members of video story audience have a great variety of interests, needs, and schedules.

Nevertheless, after video stories have been produced, the depth of each event or topic covered stays unchanged. The goal of story creators is to produce stories that will appeal to the largest audience possible. This approach uses a “one size fits all” principle, and does not provide a way to adapt a story to individuals. Allowing viewers to be in control of portions of a storyline will increase their satisfaction with a viewing process. We created a simple web-based interactive video mechanism allowing adaptation of video to interests of individual viewers.

The core of our solution is the Flexible Storylines structure shown in Figure 82. This structure allows viewers to adapt stories by choosing whether to watch Body or Diversion. The Buildup component provides a smooth way to offer viewers a choice. Context provides missing background information for viewers who decide to watch a Diversion at a later time. Merge clips transition diversion content back into the main story line. Merge also allows viewers to exit early out of Diversion without becoming disoriented in the events covered in Post-story. Exiting early provides perceived safety to viewers, meaning they can undo their story altering choice. Flexible Storylines structure provides adaptability to the story while preserving smoothness of viewing experience and flow and structure of a story.

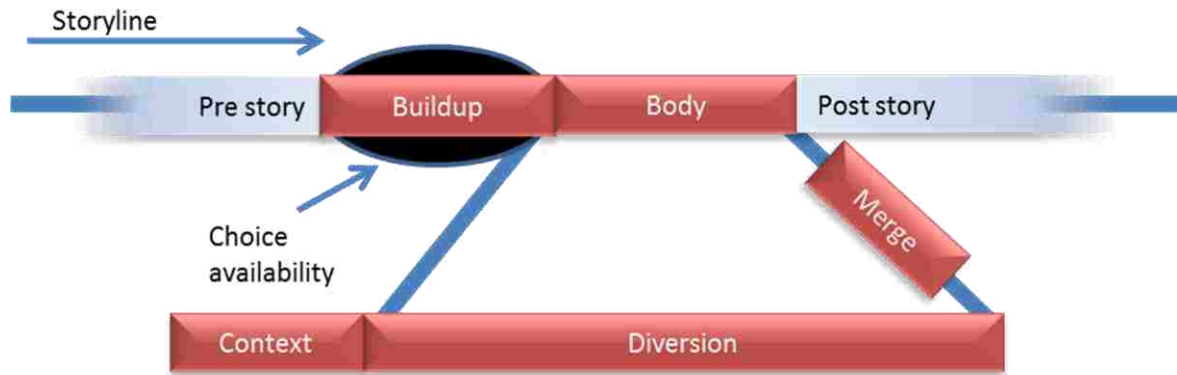


Figure 82
Flexible Storylines structure

In order to make Flexible Storylines easy and familiar to use, we have built both creation and presentation tools for Flexible Storylines around the concept of story and time line. The workflow of story creation process starts with video editing software such as Avid. Story creators use it to edit video clips, combine them with audio and create self-contained components of Flexible Storylines structure (Figure 82). Next, story creators use Flexible Storylines Creator tools to do further editing, label and assemble components into a flexible story. Video Editor is a tool used to upload, edit and label videos. Flexible Creator is a tool used to assemble labeled components into a flexible story. It provides a timeline that can be filled with non-interactive and interactive scenes. Interactive scenes are placed on the timeline with the help of Choice template (see Figure 83). Validation of Video Editor and Flexible creator has shown that these tools were user friendly and similar to current video editing software.

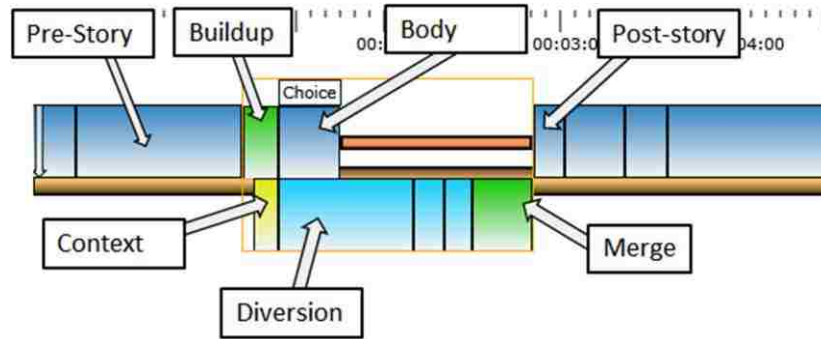


Figure 83
Choice template screenshot

Flexible Storylines required consecutive playback of multiple videos over the network. Delivery of a sequence of discrete videos over the network added time constraints to load and buffer each video. We were able to overcome time constraints by using a preloading architecture and a resource pool of four media players. Validation of our solution has shown that this approach provided smooth and seamless playback.

We offered story viewers a novel concept of being able to adapt the story to their interest. In order to aid viewers, we built a simple and intuitive user interface for watching flexible stories. Flexible Player interface is akin to an interface of an online video player. While playing interactive scenes Flexible Player offers additional interactive controls to viewers. User study has shown that viewers found Flexible Player's interface to be simple and user friendly. Increased satisfaction from being able to adapt a story was reported.

Future work can be done in adding a way to specify a default path through the story. A default path is the path through a storyline that will be taken without viewer's interaction. If a viewer chooses to watch a default version of a story no interaction will be offered to him or her. Listed below are ways to specify the default path:

1. Fixed by story creators
2. Fixed by story viewers
3. Viewer's time constraint
4. Viewer's interest

Default choices fixed by story creators can be specified by story creators at the time of a production of a story. This can be a setting to take or to skip all diversions. Similarly story creators can specify a default choice for each diversion individually.

Default choices fixed by story viewers can be specified by story viewers before they start the playback of a story. Similar to story creators, viewers may choose to take or skip all diversions. Also, examining the list of list of all diversions, viewers may preset a choice for each one separately.

Choices based on viewer's time constraint will be based on time a viewer has allotted to watch a story. First option will be to ask a viewer whether he or she would like to watch a short or a long version of a story. According to the option chosen an algorithm will go through each choice point and chose the option that has the shortest or longest duration. Another option would be to allow viewer set a fixed time that he or she has to watch a story. In that case, for each choice we will choose the shortest duration until we have filled up the allotted time.

Lastly, choices can be based on viewer's interest. Each Diversion will have a zero or more tags associated with it. Tags describe Diversion's content. In a military documentary, there might be tags for types of weapons, battle scenes, war officers or political leaders. All of the tags across a story are collected and presented to a viewer for selection. A viewer may choose to see

additional material about “tanks” and “Stalin”. A viewer may also want to block certain tags such as “language” or “violence”.

Taking this idea further, we can provide an automatic way to generate a default path through a story. A viewer defines a set of tags that are important to him or her regardless of a story.

Furthermore a viewer specifies a weight to each tag. Each tag will be weighted according to the level of viewer’s interest in that subject. This vector of tag weights termed as viewer’s profile

vector will be associated with each viewer. Each Diversion will have a vector of tags with weights associated with them as well. These weights would characterize how deeply this

Diversion covers a particular subject. Whenever a viewer chooses a story to watch, viewer’s profile vector will be compared to a vector of each Diversion. The similarity of these vectors will

produce a priority. A priority of each Diversion will be used to decide whether this Diversion will be taken or skipped. The result is a set of default choices for all Diversions in a story.

Bibliography

1. Brøndmo, H. P. and Davenport, G. Creating and viewing the Elastic Charles: a hypermedia journal. In McAlesse, R. and Green, C., eds., *Hypertext: State of the Art*. Intellect, Oxford, 1991.
2. Davis, R. *Introduction to film making*. Speech Communication Association, Falls Church, 1975.
3. Garrand, T. Scripting narrative for interactive multimedia. *Journal of Film and Video*, 49 (1/2), 66-79.
4. Hand, S. and Varan, D. Exploring the effects of interactivity in television drama. In Cesar, P. et al., ed., *Interactive TV: a Shared Experience*. Springer, Berlin, 2007.
5. Liestøl, G. Aesthetic and rhetorical aspects of linking video in hypermedia. In *ACM Hypertext'94* (Edinburg, UK 1994), 217-223.
6. Meadows, S. *Pause & effect: the art of interactive narrative*. New Riders, Indiana, USA, 2003.
7. Samsel, J. and Wimberly, D. *Writing for interactive media: the complete guide*. Allworth Press, New York, 1998.
8. Sawhney, N. and Balcom, D. Authoring and navigating video in space and time. *IEEE Multimedia Journal*, 4 (4), 30-39.
9. Shipman, F., Girgensohn, A., and Wilcox, L. Hyper-Hitchcock: towards the easy authoring

of interactive video. In *Human-Computer Interaction INTERACT '03* (Zürich, Switzerland 2003), IOS Press, 33-40.

10. Tiellet, C. A. B., Pereira, A. G., Reategui, E. B., Lima, J. V., and Chambel, T. Design and evaluation of a hypervideo environment to support veterinary surgery learning. In *ACM Hypertext'2010* (Toronto, Canada 2010), 213-222.

11. Ursu, M. F., Thomas, M., Kegel, I. C., Williams, D., Tuomola, M. L., Lindstedt, I., Wright, T., Leurdijk, A., Zsombori, V., Sussner, J., Myrestam, U., and Hall, N. Interactive TV narratives: opportunities, progress and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 4 (4), 25-64.