制約解消系を用いたタイムライン編集用インタフェース

栗原 一貴　David Vronay　五十嵐 健夫

1　はじめに

オーサリングツールおいてタイムラインエディタの存在は欠かせない。タイムラインエディタは，リニアなシーケンスと同期を扱う上で優れたメタファーと言える．しかし以下のような問題点がある．

1. 時間管理がグローバルな一つの時計に固定されている．よってシーケンスが大規模になったとき，カットや挿入といった作業が煩雑になる．

2. 時間の指定が明示的過ぎる．「何時から何分で」という指定は多くの場合煩雑であるし，再編集する際，開始時刻が重要なのか，実行期間が重要なのかといった情報が不明瞭である．

Flash MX，Wolber などは Hierarchical Chanels, Multiple Synchronous Streems などを導入することによってタイムライン表現を拡張し対応しているが，編集作業が複雑化するという新たな問題を生じている．

2　AfterThought

われわれは従来のタイムラインエディタの優れている点を保存しつつ，弱点を克服されるようデザインされたタイムラインエディタを持つオーサリングツール “AfterThought” のプロトタイプを開発した．その特徴をまとめる．

1. Events happen in time: 実際の作業が実装されているいないにかかわらず，プレースホルダとして次々にイベントを作ることができる．これはユーザに抽象的な創作作業（トップダウンアプローチ）を許す．

2. Time is relative: 過去現在未来の３つのビューを作ることにより，時間を相対化する．ユーザは「以前か，今か，以後か」という単純な選択によりタイムラインを編集することができるようになり，常にグローバルな時間管理を強いられる心理的負荷が解消される．

3. Events can be fuzzy: 開始時間，継続時間，終了時間を指定しなくてもイベントを生成できる．「順番や時間は気にしないがとにかくこれらの仕事を行いたい」という，現実的によくあるユーザの要望に答えるものである．

4. Events can also be specific: イベントの前後関係，つまり相対的な開始時間と終了時間，および継続時間を簡単なクリック動作で指定できる．「これらのイベントはいつ始まってもかまわないが，同時に開始される.」といった柔軟な指定を行える．グローバルな時間ではなく，関係（拘束）を指定することはの重要性は Drapeau によっても示されている．

5. Events can contain their own timeline: イベント外部と内部の時間軸を分離することができる．ほかのイベントとの同期については，同じ時計を持つ同一タイムラインの中でのみ考慮すればよい．

これらの特徴の根底にあるコンセプトは，「スケッチするようなラフなタイムライン編集を可能にすること」である．

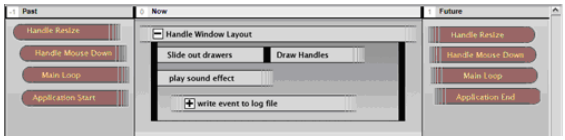

図 1: Timeline of Afterthought

3　実装

実装については，各 event の時間的な拘束を守りつつユーザに操作を行わせるような制約解消系の導入が必要である．本研究では山根らの方法を用いたが，その他の方式を用いても差し支えない．

プロトタイプ版の開発は C#.NET を用いて行った．近年における常識的なスペックのコンピュータで快適に動作する．

4　ユーザスタディと考察

われわれは開発したタイムラインの実用性を調査するために，Macromedia Flash および Adobe Premiere Pro と比較するユーザスタディを行った．18 人のユーザスタディ協力者は過去に Flash か Premiere かその両方を使用した経験を持っている．手順は以下の通りである．

1. まず，協力者にそれぞれのツールの使用方法を説明し，操作に慣れてもらう．
2. すべてのツールに慣れたところで，図に示すような 3 種類のタイムラインを提示し，3 種のツールを用いてそれぞれ指定された場所に 5 つの新しいイベントを挿入する作業を行う．
3. 協力者たちは指定された「これら 3 つのイベントは同時に開始される」といった拘束条件（図の点線で示されているもの）を保守しなければ作業終了とならない．
4. 作業終了後，協力者たちはそれぞれの作業がどれだけ難しかったかを 5 段階（1:難しい　〜 5:簡単）で評価する．

ここで，各協力者について用いるツールの順が異なり，順序効果は相殺されていることを付記しておく．

ユーザスタディの結果を図に示す．図はそれぞれのツールでそれぞれのタスクを行った際の協力者評価の平均をプロットしたものである（3 ツール ×3 タスク）．Scheffe の多重比較法によると，Afterthought はタスク#1 とタスク#2 において他の 2 ツールに比べて有意に作業が容易であったという結果が得られた（タスク#1: $p = 0.036$ 対 Premiere，$p = 0,003$ 対 Flash，タスク#2: $p = 0.045$ 対 Premiere，$p = 0.000$ 対 Flash）．Flash と Premiere においては，協力者は新しいイベントを挿入した際に保ちたい拘束条件が壊れてしまい，それを復帰するのに困難があったことが観察された．Afterthought ではそれらの拘束条件の保存が自動的に行われるため，そのような困難は観察されなかった．

一方，一元配置分散分析の結果タスク#3 においては 3 種類のツールの間に有意な違いは見出されなかった（$F_{2,51} = 2.25, p = 0.115$）．これは現在のわれわれのタイムライン設計において必要な機能が未実装であることに由来するものであると認識している．多くの協力者は「どことどこが拘束されているのか」を示す十分な視覚フィードバックがないことを指摘した．さらにすべての拘束を一時的にすべて解除するモードやボタンを付与したり，強引に拘束を変えようとユーザが思ったときに暗黙的に拘束を緩める機能などが求められる．

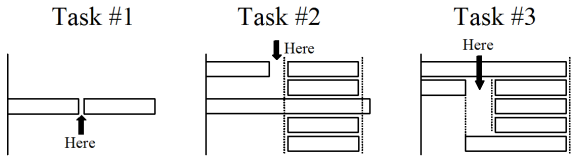

図 2: Timelines for Userstudy

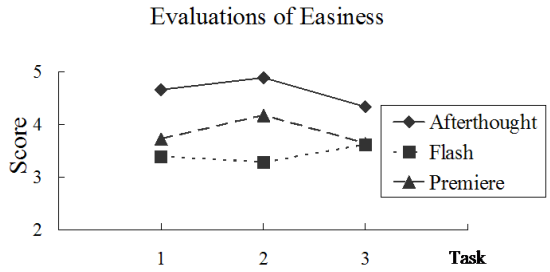

図 3: Evaluations of Easiness

5　おわりに

本研究では従来のタイムラインにおける問題を分析し，その克服のために制約解消系を用いた柔軟なタイムラインを設計した．さらにプロトタイプ版を開発し，従来ツールと比較するユーザスタディを行うことでその有効性と将来への改良点を明らかにした．