タイムラインとビジュアルプログラミングを融合した映像制作システム

高橋 弘毅[†1]　宮下芳明[†1]

概要：本稿ではモーショングラフィックスと呼ばれる映像の制作に焦点を当て，タイムラインとビジュアルプログラミングを組み合わせた映像制作システムを提案する．映像の制作手法にはタイムラインを用いるものと，プログラミングを用いるものがある．前者は映像を時間軸上で管理できるが大量の映像オブジェクトを扱うことや，数式に基づいてオブジェクトを移動させることなどが難しい．一方プログラミングを用いればタイムラインの欠点を補うことができるが時間指定での映像制作が難しく，プログラミング未経験者にとってハードルが高い．そこで提案手法では映像プログラムを視覚的にわかりやすいビジュアルプログラミングで構築し，タイムラインで映像プログラムの実行時間を制御することで既存の手法の問題点を解決する．本論文では提案システムの実装，評価を行った．

Video Production System that Integrates Timeline and Visual Programming

KOKI TAKAHASHI[†1]　HOMEI MIYASHITA[†1]

Abstract: In this paper, We propose a video production system that integrates timeline and visual programming. There are two methods for video production. One way is using timelines and the other is using programming. Although the former can manage animation on the time axis, it is difficult to deal with large numbers of objects and to move objects based on mathematical expressions. On the other hand, using programming makes it possible to compensate for the shortcomings of the timeline, but it is difficult to produce animation with time designation, and it is tough for inexperienced users. Therefore, the proposed method solves the problem of the existing method by constructing the video program with visually easy visual programming and controlling the execution time of the video program in the timeline. In this paper, We implemented and evaluated the proposed system and discussed future tasks.

1. はじめに

YouTube やニコニコ動画といった動画共有サービスだけでなく，近年は Twitter や Facebook などの SNS でも動画の投稿が増えていることから，映像制作はより一般的になってきている．映像には多くの種類があるが，モーショングラフィックスと呼ばれる，図形や文字などのグラフィックデザインに動きを加えた映像作品に焦点を当てる．映像の制作手法にはタイムラインを用いるものと，プログラミングを用いるものがある．

タイムラインは映像の再生時間に対応した横軸，映像のレイヤに対応した縦軸から構成され，時間軸上で映像を管理する機能である．タイムライン上に映像オブジェクトを設置し，シークバーが映像オブジェクトの上を通過している間，映像オブジェクトに対応した映像が出力される．タイムラインの強みは図形を表示するタイミングやエフェクトをかけるタイミングなどを視覚的に把握し編集できる点である．しかし大量の図形を扱うことや，ある数式に基づいて図形を移動させることが難しい．大量の図形を扱うためにはタイムライン上に表示したい図形の数だけ映像オブジェクトを設置しなければいけないため，編集画面が圧迫されてしまう．また，図形の移動もユーザが手作業でパラメタを入力しなければいけないため複雑な動きを表現するのが困難である．

一方プログラミングを用いれば繰り返しの処理を行うことで図形の複製や，数式に基づいた図形の移動などが可能になる．しかしプログラミングによる映像制作ではシーンチェンジや図形を表示させるタイミングの細かい編集が難しい．シーンチェンジを行うにはいくつかシーンごとに実行するプログラムを用意し，再生フレーム数で条件分岐などを行うことで実現できるが，タイムラインのように瞬時に表示するタイミングを入れ替えることや，表示する長さを変更するのが難しい．また，プログラミング未経験者にとって映像プログラムの構造把握が難しく，プログラミングを用いて自分の思い通りに映像を制作するのは困難である．そこで，本論文では既存の２つの手法の問題点を解決するため，タイムラインとビジュアルプログラミングを組み合わせた映像制作システムを提案する．

†1 明治大学総合数理学部先端メディアサイエンス学科
Department of Frontier Media Science, Faculty of Interdisciplinary Mathematical Sciences at Meiji University

2. 関連研究

タイムラインは複数のアイテムを時間軸上で管理するのに大変有効であるが，扱うアイテム数が増えすぎてしまうと編集画面がアイテムで埋め尽くされ，機能しなくなってしまう問題がある．そこで栗原らは柔軟に動かすことのできる新しいタイムラインを提案した[1]．このタイムラインではアイテムの位置を別のアイテムと揃えることや，長くなってしまったタイムラインの一部を部分的に表示することが可能で，最初に述べたアイテムが増えすぎるとタイムラインが機能しなくなる問題に対しては，複数のアイテムを一つのグループとして扱うことでタイムラインが埋めつくされることを回避した．また，的場らは音・映像構成のためのロータリーシーケンサを提案した[2]．円状のタイムラインを画面に設置することで音や映像を出力するというものである．タイムラインのシーケンスの周期や長さを変えることで複雑なリズムの音や，それに合わせた映像を作り出すことができる．

視覚的にプログラムの把握，編集ができるビジュアルプログラミングに関しての研究は古くから行われており，Haeberli らはビジュアルプログラミングを用いたグラフィックスシステムを提案した[3].小林らはライブ映像パフォーマンスにおいて映像素材の選別，エフェクトの適用などをデータフロー図によって表現するシステムを提案した[4]．データフロー図は映像ソースやエフェクトの構成を視覚的に把握しやすいため，音楽用のビジュアルプログラミング言語 Max/MSP[5]や映像用のビジュアルプログラミング言語 vvvv[6]などでも用いられている．After Effects や Adobe Flash にはアニメーションを制御するためのスクリプト言語が用意されているが，アニメーションを確認するためにはスクリプトの記述を変更したあとに，レンダリングするまでどのようなアニメーションになったか確認することができない．また，スクリプトの編集画面と GUI が別のウィンドウに分離されている場合があり，プログラム記述と，アニメーションの確認を行うために画面を何度も切り替える必要がある．そこで加藤らはアニメーションのアルゴリズムに対するパラメタを生成し，それをユーザが編集できるコードエディタと，パラメタ調整に適した GUI をもつシステムを提案した[7]．このシステムではキネティックタイポグラフィという歌詞などの文字にアニメーションを加えた映像を制作することができ，歌詞を表示するタイムングを管理するためのタイムライン，アニメーションのパラメタを変更するためのエディタを持つ．物理シミュレーションを用いたアニメーション制作の支援では，アニメーションの軌道を候補の中から絞り込んでいく手法を Twigg らが提案した[8]．絞り込みは出力された映像の上をドラッグすることで行うことができ，数式などを操作することなく直観的な操作でアニメーションを制作できる．

3. 提案システム

提案システムは映像制作を行うためのビジュアルプログラミング環境（図 1 赤枠），構築した映像プログラムの実行区間を管理するためのタイムライン（図 1 緑枠），映像を再生するためのディスプレイから構成される（図 1 青枠）．本章では提案システムの持つ機能について述べる．システムの実装は JavaScript で行い，ブラウザ上で動作する．

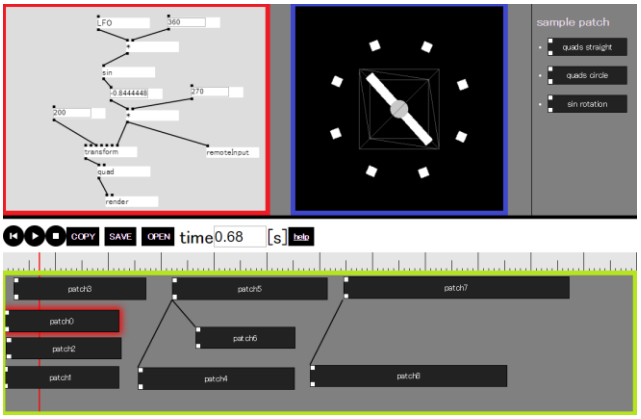

図 1 提案システムの画面

Fig.1 User Interface of proposed system.

3.1 タイムライン機能

提案システムのタイムラインは映像プログラムの実行時間，実行順序を管理する役割を持つ．タイムライン上をダブルクリックすることで映像プログラムを格納するためのブロック（図 2）を配置することができる．このブロックを以後パッチと表記する．シークバーがパッチと重なっている間，そのパッチの内部にあるプログラムが実行される．パッチとパッチはタイムライン上で接続することでパッチ間での連携を実現できる．詳しい説明は 3.3.6 章でノードの説明と併せて述べる．パッチをクリックするとそのパッチ内部のプログラムの編集が可能になる．編集可能状態になったパッチは赤く点灯するため視覚的に確認できる．

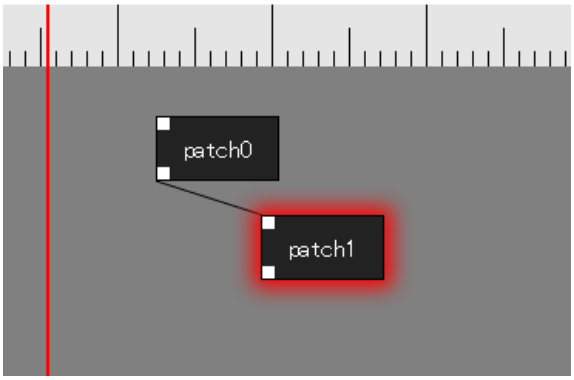

図 2 作成されたパッチ

Fig. 2 Created patches.

パッチはマウスでドラッグすることで上下左右の移動，右端をドラッグすることでパッチの横幅を変更することができる．タイムライン上部の COPY ボタンを押すと選択中のパッチをコピーすることができる． また，タイムラインにはレイヤ機能があり，シークバーが複数のパッチの上を通過する際，タイムラインの上部にあるパッチからプログラムが実行される．例えば patch0 に小さな青い四角を表示するプログラム，patch1 に大きな赤い四角を表示するプログラムが格納されている場合，patch0 が上部にあるケースではまず青い四角が描画され，その後に patch1 の赤い四角が描画される（図 3 左）．同様に patch1 が上部にあるケースでは先に赤い四角が描画されその後に青い四角が描画される（図 3 右）．

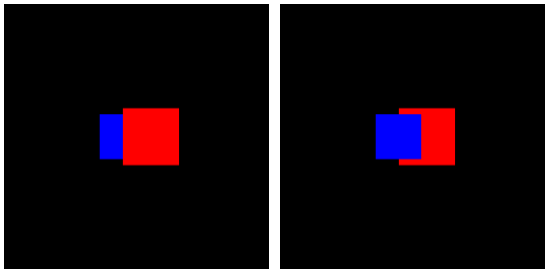

図 3　レイヤ機能を用いた例．

Fig.3　Example of using layer function.

3.2 ビジュアルプログラミングシステム

映像プログラムは図形の情報や図形を変形させる命令などが入ったブロック（以後ノードと表記）を線でつなぎ合わせるビジュアルプログラミングを用いて構築する．ノードには情報を入力するためのピンと，出力するためのピンを持つ．

プログラミングエディタ上でダブルクリックすると生成したいノードの名前を入力するためのテキストエリアが出現する．作成したいノードの名前を入力してエンターキーを押すと入力した名前のノードが生成される．ノードの出力ピンをクリックすることで他のノードの入力ピンと接続するための線がマウスカーソルについてくるようになる（図 4 左）. この状態で接続したいノードの入力ピンをクリックすると 2 つのノードが線で接続される（図 4 右）.

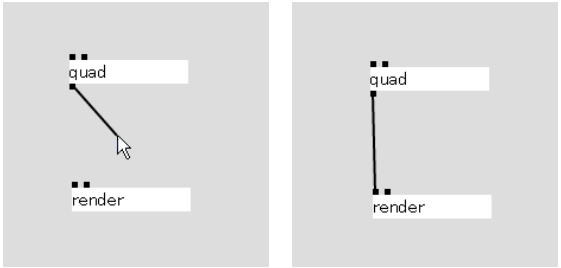

図 4　ノードの接続

Fig.4　Connected nodes.

3.3 ノードの説明

本章ではビジュアルプログラミングに使用するノードの働きについて説明する．

3.3.1 映像出力用ノード

受け取った情報を映像として出力するためのノードが「render」である．入力ピンを 2 つもち左のピンでは映像として出力するメッシュ（3.2.2 参照）を受け取り，右のピンでは映像の背景を変更するための「rgb」ノードから色情報を受け取る．

3.3.2 メッシュノード

メッシュとは四角や円といった図形の情報のことで，「quad」ノードは四角，「cube」ノードは立方体，「circle」ノードは円のメッシュを扱う．入力ピンを 2 つもち，左のピンには図形の変形を行うための「transform」ノードをつなぎ，右のピンには「rgb」ノードをつなぐことができる．

「group」ノードは受け取ったメッシュを一つのまとまりのメッシュとして扱うためのノードである．例えば横一列に配置した四角をまとめて回転させる場合などにはこのノードを使用することで表現可能である．

3.3.3 数値ノード

「IObox」ノードは入力ピンで受け取った数値，もしくはノードの持つテキストエリアに入力された数値を出力することができる．入力ピンから数値を受け取った場合は常にその値を表示するため，どのような値がノード間でやり取りされているかを確認するために使用することもできる．

四則演算は「+」「-」「/」「*」の 4 つのノードを用いて行うことができる．四則演算を行うノードは入力ピンを 2 つ持ち，入力された 2 つの値の演算結果を出力する．

「linearspread」ノードは配列を扱うためのノードである．入力ピンを 2 つ持ち，左のピンで受け取った数値を変化幅とし，中心を 0 として右のピンに入力された数値の数で分割した値を出力する. 例えば左に 30, 右に 4 を入れた場合，出力される配列は-15 , -5 , 5 , 15，となる(図 5).

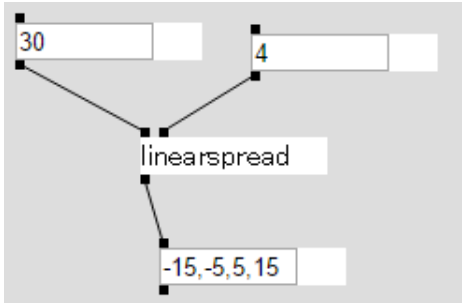

図 5　「linearspread」ノードの出力例

Fig.5　Example of Linearspread node.

3.3.4 物体変形用ノード

「transform」ノードは図形の変形を行うためのノードである. 入力ピンを 7 つ持ち, 左から図形を表示する x 座標, y 座標，図形の横幅，縦幅，xyz 軸に対する図形の回転角度の数値を受け取ることができる.

メッシュノードは「transform」ノードから配列で数値が送られてきた場合，受け取った数値の数だけ図形を表示することができる．そのため「linearspread」,「transform」ノードを用いることで図形を大量に扱えるようになる（図 6）.

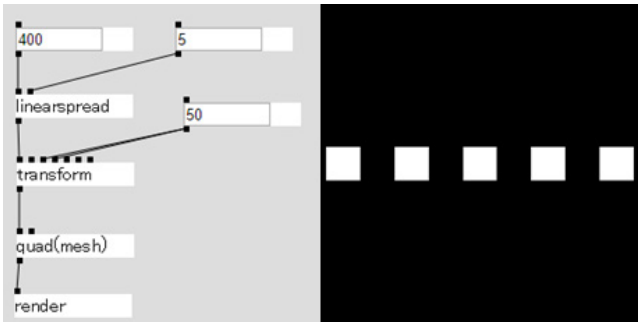

図 6　図形を複数描画した例

Fig.6　Example of rendering multiple geometry.

3.3.5 アニメーションノード

「time」,「LFO」は描画した図形に動きを加えるためのノードである.「time」は入力ピンを持たず，2 つの出力ピンを持つ．左のピンからはパッチ内での再生時間，右のピンからはパッチの最大再生時間を出力する．例えばシークバーが 10ms で 1 フレーム進む場合，パッチの最大再生時間が 970ms,シークバーがパッチの開始点から 700ms 過ぎた位置にあるときは「time」ノードは図 7 のように値を出力する.

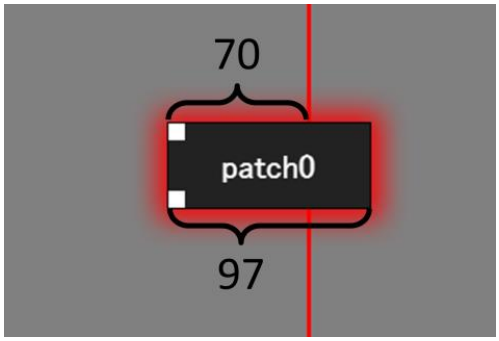

図 7　「time」ノード

Fig.7　Time node.

3.3.6 パッチ間連携ノード

タイムラインとビジュアルプログラミングをよりインタラクティブなものにするために，パッチ間で情報を共有できるノードを実装した．まず連携させたいパッチの入力ピンと出力ピンをビジュアルプログラミングと同様に線で結ぶ. 例えば patch0 の出力ピンと patch1 の入力ピンが接続されている場合，patch0 から patch1 へと値を送ることができる．パッチを接続した後は，それぞれのパッチ内のプログラムに値をやり取りするためのノードを追加する．値を渡す方のパッチには「remoteInput」ノードを追加し渡したい値を入力ピンにつなぎ（図 8 左），値が渡される方のパッチに「remoteOutput」ノードを追加することで出力ピンから値を得ることができる（図 8 右）. このノードを使うことで異なるプログラムから出力された図形の回転周期や座標変化の速度を同期することができる.

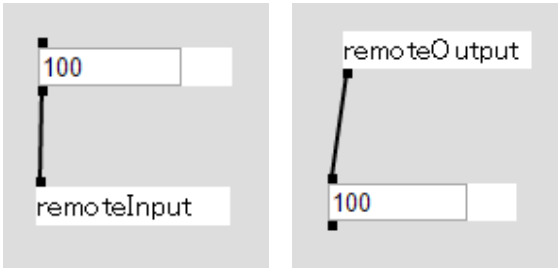

図 8　パッチ間連携ノード

Fig.8　Patch synchronization node.

3.4 サンプルパッチ

映像プログラムの構築を楽にすることと，ビジュアルプログラミングの理解を手助けする目的であらかじめ映像プログラムを組んだパッチを 3 つ用意した.「quads straight」は配列ノードを用いて四角を横一列に並べ,「quads circle」は四角を円状に並べるものである.「sin rotation」は三角関数を用いたアニメーションである．サンプルパッチは映像出力用ディスプレイの右にある 3 つのパッチを指す．サンプルパッチをドラッグしてタイムライン上にドロップするとその位置にパッチが作られ，中にはそれぞれのプログラムを動かすためのノードが配置されている．サンプルパッチの例として「sin rotation」のプログラム構造を図 9 に示す.

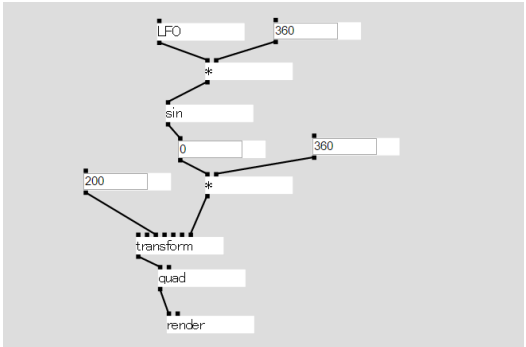

図 9　サンプルパッチの例

Fig.9　Example of sample patch.

4. 実験

4.1 実験内容

提案システムがユーザにとって使いやすいものか，提案システムを用いてユーザがどのような映像を作ることができるかを明らかにするために実験を行った．参加者は映像制作経験のある 20～23 歳までの大学生 4 名であった．このうち 2 名はプログラミングの経験があり，2 名はプログラミング未経験者である．

実験の手順は，初めに提案システムの使い方を説明し，1 時間程度の制作時間を設け被験者に提案システムを用いて映像を作らせた．映像制作が終了した後，提案システムに関するアンケートをとった．本章では制作された映像，アンケートの結果から提案システムの分析を行った結果について述べる．

4.2 実験結果

プログラミング未経験者でもビジュアルプログラミングが行えたかをプログラミング経験者と比較するため，被験者 4 名をプログラミング未経験者 a,b とし，プログラミング経験者 c,d とする．

4.2.1 映像の作例とプログラムの構造例

まずはプログラミング未経験者の例として被験者 a の作例を図 10 上，プログラミング経験者の例として被験者 d の作例を図 10 下，に示す．作例は制作された動画の中から 2 シーンずつ取り上げた．

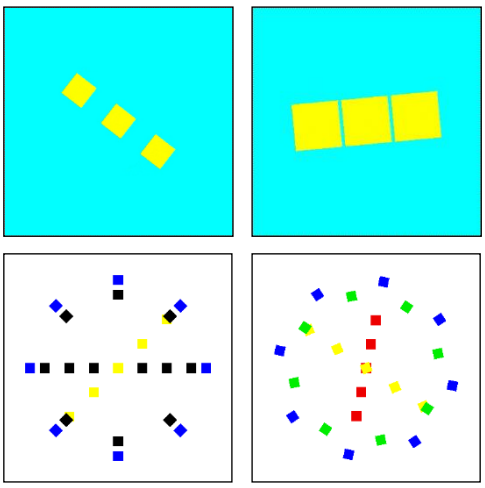

図 10 （上）被験者 a の作例（下）被験者 d の作例

Fig.10 (Top) Example of subject a (Bottom) Example of subject d.

4.2.2 アンケートの結果とプログラムの分析

映像制作後に被験者に対して提案システムに関するアンケート調査を行った．これは提案システムがユーザにとってどの程度使いやすいものかを分析するためのものである．アンケートはタイムラインとビジュアルプログラミングに関して 5 段階評価を行うもので，内容は表 1 に示し．アンケートの結果を表 2 に示す．また，提案システムを分析するために被験者の制作した映像の長さ，映像プログラムに使用されたパッチの数，ノードの数，映像オブジェクトの最大表示数についてまとめたものを表 3 に示す．

表 1 アンケート内容

Table 1 The questionnaire contents.

質問 1	提案システムの持つタイムラインを使うことで映像プログラムの時間管理は容易に行えたか（当てはまれば 5，当てはまらなければ 1）
質問 2	ビジュアルプログラミングによる映像プログラムの構造は把握しやすかったか（当てはまれば 5，当てはまれなければ 1）

表 2 アンケート結果

Table 2 Result of the questionnaire.

	a	b	c	d	平均
質問 1	5	5	5	4	4.75
質問 2	3	3	4	5	3.75

表 3 映像の各要素

Table 3 Element of video.

	a	b	c	d
映像の長さ[s]	3.9	4	2.75	9.5
パッチ数	2	3	2	4
ノード数	36	40	31	74
映像オブジェクトの最大数	3	15	2	26

4.3 考察

アンケートの結果から，タイムラインを用いることで映像プログラムの実行時間の管理を容易に行えることがわかった．ビジュアルプログラミングに関しては，プログラミング未経験者にとってはやや使いやすさの評価点が低いが，全くプログラムが組めないということはなかった．ノードとノードを線で結ぶことでプログラムを構築していくため，視覚的にも構造が把握しやすく，プログラミング未経験者でも映像プログラムを組むことができることが分かった．被験者からはノードに数値を入れるために「IObox」ノードを作ると 1 つのパッチ内に「IObox」ノードがたくさん必要になりエディタが圧迫されてしまう意見があった．この問題を解決するためはノードに直接数値を入力できる機能が必要である．また，パッチ連携ノードで 1 つのパッチから複数の数値を送りたかったという意見もあった．現時点ではパッチ連携ノードで送信できる値は 1 つだけであり，この機能も強化することでより映像制作の幅が広がると考える．

表 3 の映像の各要素を比較すると，被験者 d は多くのパッチとノードを生成している．この差はパッチのコピーを活用したことで生まれた．被験者 d は作成したパッチをコピーして一部分を変え，またパッチ間での連携を行うことで多くのパッチとノードを扱った．

映像の作例を比較すると，被験者全員がサンプルパッチを元にして映像プログラムの構築を行ったため，回転アニメーションを基本とした作例が多くなった．プログラミング未経験者の被験者は図形の回転を反対にする方法や，配列ノードの使い方を理解するのにやや時間がかかった．配列ノードはテキストベースのプログラミングにおける for 文による繰り返し処理の役割を果たすため，プログラミング未経験者にとって繰り返し処理の感覚がつかみにくかったと考える．また，パッチ間連携ノードもプログラミングにおける変数の役割を果たすため，映像プログラムに組み込むことを控えてしまったと考える．

5. 議論

実験の結果を踏まえ，提案システムでどのような映像を作ることができるのかについて述べる．提案システムではビジュアルプログラミングを用いて for 文による繰り返し処理や数値の共有，演算などを行うことで数式に基づいた図形の移動や図形を大量に生成，操作することが可能である．また，テキストベースのプログラミング言語では手間のかかるプログラム実行のタイミングの調整をタイムラインで管理することで容易に編集できる．これらの点から細かく時間調整された，プログラミングの力が必要な映像の制作に向いていると考える．関連として映像用ビジュアルプログラミング言語である vvvv には時間変化に応じて様々な値を出力することができるノードが存在する．このノードはタイムラインで値の出力を管理するであり，プログラムの実行時間を管理する提案システムのタイムラインとは別である．このノードを使用してシーンチェンジを実装するには映像プログラムのパッチをいくつか作成し，それらを条件分岐で表示するタイミングを分ける必要がある．しかし作成するシーンが多くなるほど条件分岐は複雑になり，調整も難しい．時間軸上でのパラメタ操作が難しいというプログラミングの問題を解決できても，シーンの切り替えといった映像プログラミングの問題に対しては有効な手段とは言えず，提案システムのもつタイムラインではこの問題にも対処できる強みがあると考える．Adobe Flash と比較した場合，プログラムを時間軸上で管理できるが複数プログラム間での連携ができない点や，プログラムの構造把握が難しいという点で提案システムに劣ると考えた．

次にこの提案システムでは作成が困難な映像について述べる．図形の動きはすべてシステム側で用意したノードを使って表現しなければいけないため，ユーザの描いた軌道にそって図形を動かすといったことができない．また，壁にぶつかったら四角を反射させる，受け取った配列の中で x 座標が偶数の場合のみ色を変更するなど，細かい条件分岐を用いた表現も困難である．制作できる映像はノードの種類に依存しているため，テキストベースのプログラミングと比較すると映像制作の自由度が下がってしまう．

以上の点から提案システムは細かく時間管理されたプログラミングによる映像制作に有用ではあるが，制作できる映像はノードの種類に依存するため細かいアルゴリズムを用いた映像などの制作には向いていないと言える．

6. まとめ

本論文ではタイムラインとビジュアルプログラミングを用いた映像制作システムを提案した．タイムラインで映像オブジェクトを一つずつ管理する方法では，大量の映像オブジェクトを扱うことや，数式に基づいた図形の移動などが難しい．プログラミングを用いればこの問題を解決できるがプログラミング未経験の映像制作者にとって扱うのが難しく，時間軸上での細かいパラメタ操作も難しい．

提案システムでは映像プログラムを構築しやすくするためにビジュアルプログラミングを用いてプログラムを組み，タイムラインで作ったプログラムの実行時間を管理することで既存の手法の課題を解決した．本論文では提案システムの有用性を評価するための実験を行なった．実験の結果，プログラミング経験の有無にかかわらずビジュアルプログラミングを用いて映像を制作できることが分かり，タイムラインが映像プログラムの時間軸上での管理に有用であることが示された．しかし提案システムでは制作が困難な映像もあり，今後はそのような映像制作の方法について検討したい．

参考文献

[1] Kazutaka Kurihara, David Vronay, Takeo Igarashi. Flexible timeline user interface using constraints. CHI’05, pp.1581-1584, 2005.
[2] 的場 寛，中村 滋延. dial:音・映像構成のためのロータリーシーケンサ. 情報処理学会研究報告音楽情報科学, Vol.89, pp.1-5, 2011.
[3] Paul E.Haeberli. ConMan: a visual programming language for interactive graphics. SIGGRAPH’92, Vol22, pp.103-111, 1988.
[4] 小林 敦友，志築 文太郎，田中 二郎. データフロー図に基づくリアルタイム映像合成システム. 情報処理学会研究報告音楽情報科学, Vol.50, pp.1-6, 2008.
[5] Max/MSP, https://cycling74.com/products/max/(2016-11-20)
[6] vvvv - a multipurpose toolkit, https://vvvv.org/(2016-05-21)
[7] Jun Kato, Tomoyasu Nakano, Masataka Goto. TextAlive:Integrated Design Environment for Kinetic Typography. CHI’15, pp.3403-3412, 2015.
[8] Christopher D. Twigg, Doug L.James. Many-Worlds Browsing for Control of Multibody Dynamics, SIGGRAPH'07, Vol.26, No.3, 2007.