日本ソフトウェア科学会第 28 回大会 (2011 年度) 講演論文集

データフロービジュアル言語におけるデータ調整ノブ

小林 敦友　志築 文太郎　田中 二郎

我々は、データフロービジュアル言語において、ノードへの入力値を調整するユーザインタフェースを開発した。従来のデータフロービジュアル言語においてユーザがデータフロー上を流れるデータ値の作成や調整を行うには、定数ノードを作成し、別のノードと結線することが必要となる。これらのデータ値の作成や調整という操作は、データフローの記述において多く発生する。また結果として記述されるデータフローは、ノード数とエッジ数が多くなるため視覚的に複雑になる。これらの問題に対し我々は、ユーザが値を調整するためのユーザインタフェースを開発した。本インタフェースはノードの入力部分に配置されており、ユーザに指定された値に応じて、エッジが結線されていない場合は定数を、結線されている場合はその入力されたデータ値を変換し、それぞれノードへの入力値とする。本インタフェースにより、ユーザはデータフローをより簡潔かつ速く記述できるようになる。

1　はじめに

データフロービジュアル言語は、データの流れをエッジ、データに加える処理をノードとして視覚的に表現するビジュアル言語である。ユーザはノード間をエッジで結線することによって、データフロー、すなわちデータの流れを記述する。広く使われているデータフロービジュアル言語は、特定のドメインに特化したものが多い。代表的なものとして、信号解析のための LabVIEW、音楽情報処理のための Max、音響処理のための MSP などが挙げられる。これらのドメイン特化型のデータフローで扱われるデータ型は 2 種類に分けることができる。すなわち、その言語の処理対象である主なデータ型と、ノードのパラメータ設定に使われる副次的なデータ型である。例として MSP にて、ノコギリ波を生成し、その信号にローパスフィルタを適用する場合を考える。ノコギリ波を生成するノードは、周波数として実数値の入力を持ち、音響信号を出力する。また、ローパスフィルタのノードは、音響信号の入力と、カットオフ周波数として実数値の入力を持ち、ローパスフィルタの適用結果を音響信号として出力する。この例で主なデータ型とは音響信号であり、副次的なデータ型は実数値である。

MSP を含む、多くのデータフロービジュアル言語では、副次的なデータ型も、主なデータ型と同じデータフロー上で扱われる。このことは、操作における手間を増やす上に、データフローの見た目を複雑にする。上記の例では、実数値の定数ノードを 2 つ作成し、ノコギリ波生成ノードの周波数とローパスフィルタのカットオフ周波数にそれぞれ入力する必要がある。

さらに、ノコギリ波生成ノードから出力される音響信号のボリュームを調整するとき、ユーザは乗算ノードと新しい定数ノードを作成し、それらを結線し、ノコギリ波生成ノードとローパスフィルタノードを結ぶエッジ上に挿入する。この音響信号のボリューム調整は頻繁に行う必要があるが、そのたびにユーザはこれらの操作を行うことになる。視覚的な面においても、これらの副次的なデータフローは MSP の主なデータ型である音響信号のデータフローを把握しにくくする。このように、データフロー上に主なデータ型のデータフローと副次的なデータ型のデータフローが

Data Adjustment Knob on a Data-flow Visual Language

Atsutomo Kobayashi, Buntarou Shizuki and Jiro Tanaka, 筑波大学, University of Tsukuba.

混在することは、主なデータ型のデータフローの把握を阻害する。

そこで我々は、ノードの入力部分（入力ポート）に入力される値を調整するデータ調整ノブを開発し、我々が開発してきたライブ映像パフォーマンス向けの映像処理データフロービジュアル言語 ImproV [1] に導入した。このデータ調整ノブは、入力ポートに配置されており、その入力ポートに何も結線されていない場合は定数ノードが結線されているのと同じ役割を果たし、その入力ポートにエッジが結線されている場合は入力される映像の透明度を変更する。データ調整ノブを導入することにより、定数ノードや透明度変更ノード、さらに、それらのノードを結線するためのエッジが必要なくなる。ユーザはノードのパラメータ調整や、透明度の変更を簡潔に記述できるようになり、主なデータ型のデータフローを把握し易くなる。

以降、第 2 節ではまず、ImproV について説明する。ここではデータ調整ノブ導入前の ImproV を使って説明する。その後、第 3 節にてデータ調整ノブについて説明する。第 4 節ではデータ調整ノブの評価実験について述べる。第 5 節で関連研究について述べたのち、第 6 節にてまとめる。

2 ImproV

ライブ映像パフォーマンスとは、コンサートやクラブにおける DJ イベントにおいて映像を流すパフォーマンスである。ライブ映像パフォーマンスの演者は、流れている音楽やその場の雰囲気に合わせた映像を、途切れることなく流すことが求められる。従来のライブ映像パフォーマンスは、演者があらかじめ用意した映像素材を映像ミキサを使って切り替えるという手法によって行われてきた。

我々が開発した ImproV は、映像ソース、映像ミキサ、映像エフェクタなどをノードとして持つデータフロービジュアル言語であり、映像を再生したままデータフローの編集が可能である。演者はライブ映像パフォーマンスの最中に、ノードを組み合わせることによって映像や映像エフェクタの組み合わせを変更し、多彩な映像を作り出すことができる。これにより演者は、従来より柔軟に音楽や雰囲気に合わせた映像を流すことが可能になる。

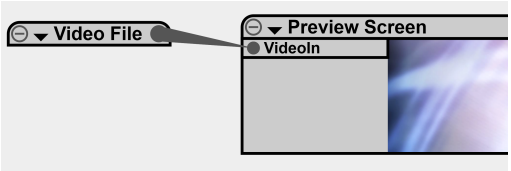

図 1　ImproV のデータフローの例：Video File からの出力を Preview Screen で表示している。

図 1 は ImproV のデータフローの例である。図 1 には Video File と Preview Screen の 2 つのノードが置かれている。各ノードの右上の円は出力ポートであり、Preview Screen の VideoIn と書かれた長方形は入力ポートである。図 1 では Video File からの出力が Preview Screen の VideoIn に結線されている。Video File は作成時に指定された映像ファイルをループ再生し、その映像を出力ポートから出力する。Preview Screen は VideoIn に入力された映像を表示するノードである。すなわち、図 1 の意味は、Video File からの出力を Preview Screen で表示しているということになる。

図 2 はより実際の使用に即したデータフローの例である。ここでは Video File によって読み込まれた二つの映像が Addition によって加算合成されている。また、二つの映像うち一方、花の実写映像（図 2 左下段の Video File に読み込まれる）には Addition に入力される前に Brightness が挟まれている。Brightness は明るさを調整するノードであり、明るさが Slider によって半分程度に調整されている。

ImproV のデータ型は映像型のみである。映像型のデータは実際には映像のフレーム画像であり、ImproV のデータフローは各フレームタイミングに要求駆動によって評価される。つまり、ImproV は画像の処理を連続して繰り返すことによって映像の処理を行う。映像型は、ImproV の処理対象である映像と、映像エフェクトのパラメータなどの副次的なデータの両方に使われる [2]。映像エフェクトのノードはパラメータとして入力された映像型データの各ピクセルの輝度値を取得し、その輝度値をそのピクセル位置におけるエフェクトのパラメータとしてエフェクトを適用する。

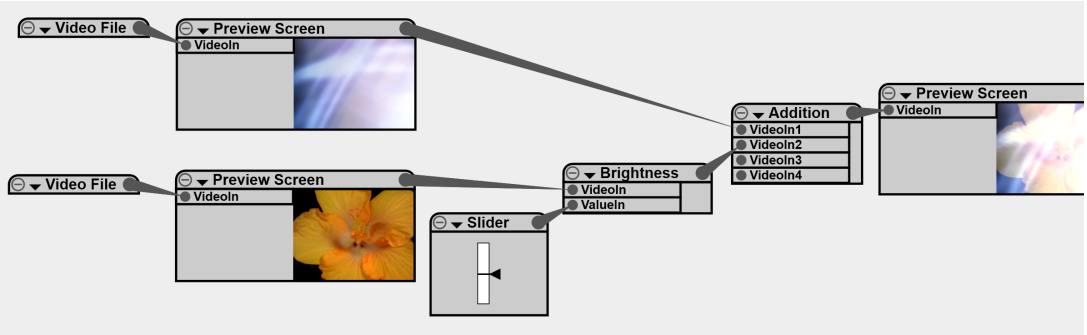

図 2　ImproV の実際の使用に即したデータフローの例：Video File によって読み込まれた二つの映像が Addition によって加算合成されている。また、片方の映像は Brightness によって明るさを調整されている。

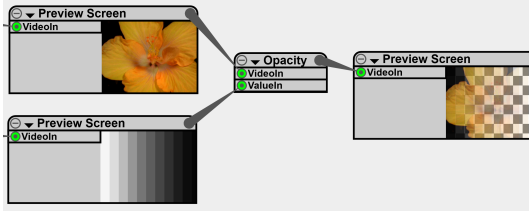

図 3　映像型によるパラメータ指定の例：Opacity の出力は ValueIn に入力された映像の輝度値に応じて透明度が変更されている

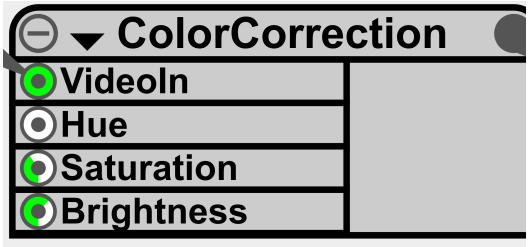

図 4　データ調整ノブ導入後のノード

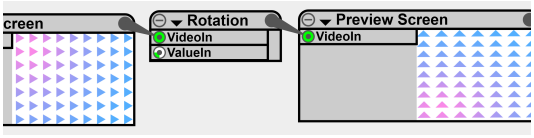

図 5　入力ポートに何も入力されていないデータ調整ノブの例

図 2 では、透明度を変更するノードである Opacity の ValueIn にグレースケールの映像を入力している。Preview Screen の表示において、格子模様は透明であることを示す背景である。ValueIn に入力された映像の暗い部分が透明になっていることが確認できる。

パラメータを映像型によって指定することにより、フレーム内の位置によってエフェクトのかかり具合を変えることができ、パラメータを時間軸に沿って自動的に変化させることも可能である。

フレーム全体に同じパラメータでエフェクトを適用するには全ピクセルが同じ輝度値の画像を入力する。図 2 に出てきた Slider は、全ピクセルが白 (RGB=1.0,1.0,1.0) でありアルファ値がスライダーによって指定された値である画像を出力する。

3　データ調整ノブ

図 4 はデータ調整ノブ導入後のノードである。入力ポートの円が二重になり、二つの円の間に緑の扇型が描かれる。

データ調整ノブは実数値を持っており、その値はユーザが入力ポートを上下にドラッグすることにより 0.0 から 1.0 の範囲で変化する。緑の扇型の角度はこの値を示し、0°から 360°まで変化する。図 4 では、VideoIn、Hue、Saturation、Brightness のデータ調整ノブそれぞれが 1.0、0.0、約 0.33、約 0.66 を示す。

ImproV におけるデータ調整ノブは、エフェクトのパラメータ調整と、映像の透明度の調整の 2 つの役割を果たす。入力ポートに何も結線されていない場合は、データ調整ノブは全ピクセルが白 (RGB=1.0,1.0,1.0) でありアルファ値がノブの値である画像をその入力

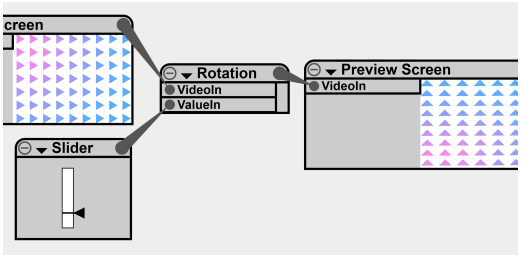

図 6　データ調整ノブ導入前の ImproV における図 5 と等価なデータフロー

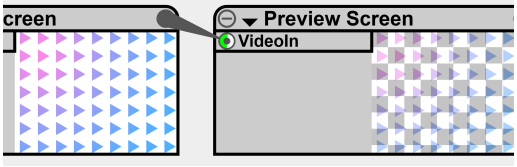

図 7　入力ポートにエッジが結線されているデータ調整ノブの例

ポートに入力する。図 5 に例を示す。Rotation は、VideoIn に入力された映像を ValueIn の値に応じて 0°から 360°まで回転し出力する。図 5 では、ValueIn が約 0.25 に合わせられているため、出力は約 90°回転している。

これはデータ調整ノブ導入前の ImproV において、約 0.25 の値に合わせた Slider を入力ポートに入力したことと同じ意味である。図 5 のデータフローは、データ調整ノブ導入前の ImproV における図 6 のデータフローと等価である。

データ調整ノブを導入することによって、Slider の作成と Slider から Rotaion の ValueIn への結線、すなわち 1 回のノード作成と 1 回の結線が削減された。この削減はパラメータ数の多いエフェクトに対して特に有効となる。図 4 をデータ調整ノブ導入前の ImproV で表したものが図 8 である。

入力ポートにエッジが結線されている場合、データ調整ノブは入力された映像型のアルファ値とデータ調整ノブが持つ実数値を掛け合わせ、もとの映像型のアルファ値を更新し、入力ポートに入力する。図 7 では、右側の Previw Screen の VideoIn が約 0.5 に合わせられている。この結果、入力された映像が半透

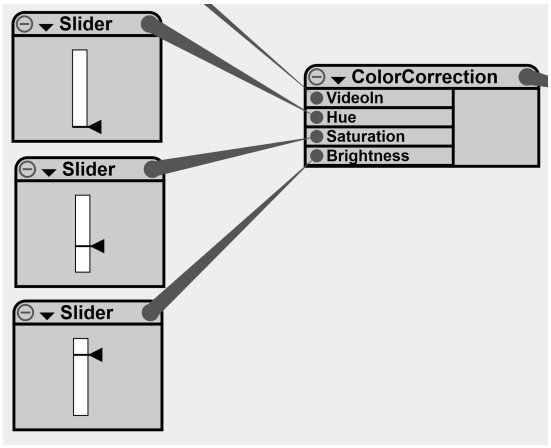

図 8　データ調整ノブ導入前の ImproV における図 4 と等価なデータフロー

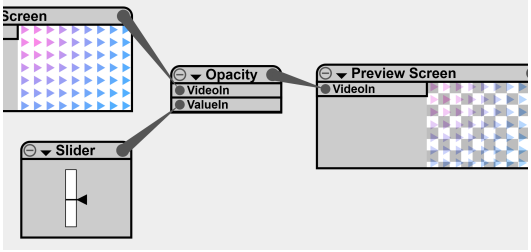

図 9　データ調整ノブ導入前の ImproV における図 7 と等価なデータフロー

明になって表示されている。

この場合データ調整ノブは、データ調整ノブ導入前の ImproV において、結線されているエッジに透明度を変更する Opacity を挟み、Opacity の ValueIn に Slider を結線した事と同じ役割を果たす。図 9 はデータ調整ノブ導入前の ImproV における図 7 と等価なデータフローである。この場合、データ調整ノブを導入することによって、Slider と Opacity の作成、Slider から Opacity の ValueIn への結線、Opacity から元の入力ポートへの結線、すなわち 2 回のノード作成と 2 回の結線が削減された。

他の多くのドメイン特化型のデータフロービジュアル言語においては、エフェクトのパラメータ調整とは、定数ノードをエフェクトのパラメータに結線し、定数ノードの値を調整することである。そして、映像

の透明度の調整とは、乗算ノードと定数ノードの組み合わせにおいてその定数ノードの値を調整することである。調整したいデータを乗算ノードの一項に結線し、定数ノードの出力をもう一方の乗算ノードの入力に結線する。この状態で定数ノードの値を変化させると、もとのデータと定数ノードの値が掛け合わされたデータが乗算ノードから出力される。例えば音響信号を扱うデータフロービジュアル言語では、乗算は音量の調整となる。この様にデータ調整ノブは、他分野のデータフロービジュアル言語においても有効であると考えられる。

4 評価実験

データ調整ノブの評価実験として被験者実験を行った。この実験の目的は、データ調整ノブを導入することにより、データフロー記述の所要時間がどのように変わるかを確かめることである。

そのために、被験者にデータ調整ノブ有りと無しの２つの ImproV を使ってタスクを行ってもらい、それぞれのタスク所要時間を計測した。

タスクでは、計算機の画面が上下に分割され、下画面に目標とするデータフローが表示される。被験者はそれを見ながら上画面でデータフローを完成させる。データ調整ノブはノード作成と結線の回数を削減するが、ノード作成の時間はシステムによって大きく異なる。例えば ImproV では、ノード作成をプルダウンメニューから選ぶことで作成するが、このときメニューに表示される項目が多いほどノード作成に時間がかかるだろう。このためこの実験ではノード作成の時間は測らなかった。あらかじめ必要なノードは用意しておき、被験者には結線とデータ調整ノブ、若しくは Slider の値の調整を行ってもらった。

被験者は 21〜24 歳のコンピュータサイエンスを専攻する学生 8 人であった。被験者にはデータ調整ノブ有りの ImproV とデータ調整ノブ無しの ImproV の両方を説明し、十分に練習してもらった。その後被験者は、データ調整ノブ有り／無しの ImproV を使って 10 種類のタスクを行ってもらい、その後さらに、データ調整ノブ無し／有りの ImproV を使って 10 種類のタスクを行ってもらう。それぞれのタスク間では自由に休憩を取ってもらった。データ調整ノブ有りと無しの順番は被験者間でバランスをとり、タスクの順番はランダマイズした。データ調整ノブ有りと無しのそれぞれ 10 種類のタスクのうち 5 種類づつは等価である物を用意した。

この実験では、以下の 3 つの作業の所要時間をデータ調整ノブ有りと無しの両条件下で測定し比較することとした。

作業 1 単純な結線

作業 2 入力ポートにエッジが結線されていない場合の値の編集

作業 3 入力ポートにエッジが結線されている場合の値の編集

このため、10 種類のタスクのうち 5 種類のタスク（タスク 1〜タスク 5）にこれらの作業を含めるようにした。それぞれのタスクに含まれる作業の回数を表 1 に示す。 なお、他のタスク（タスク 6〜タスク 10）は、実験の意図を被験者から隠蔽するためのディストラクタであり、後の解析にも用いられていない。

表 1 それぞれのタスクに含まれる作業の回数

	作業 1	作業 2	作業 3
タスク 1	1 回	無し	無し
タスク 2	2 回	1 回	無し
タスク 3	1 回	無し	2 回
タスク 4	3 回	2 回	無し
タスク 5	4 回	2 回	無し

作業 1 の所要時間を計測する目的は、単純な結線に要する操作が、データ調整ノブ無しの ImproV においては結線のみであるのに対して、データ調整ノブ有りの ImproV においては結線に加え、ノブを 0.0 から 1.0 に調整する必要があることから、データ調整ノブ有りの ImproV では単純な結線の所要時間が多くなると考えられ、その増分を調べるためである。図 10、図 11 それぞれにデータ調整ノブ無し、有りのタスク 1 の目標データフローを示す。図 10 に示すデータ調整ノブ無しのタスク 1 を完遂するには結線が一回だけなのに対し、図 11 については結線とノブ

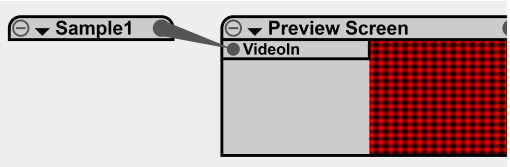

図 10　データ調整ノブ無しのタスク 1

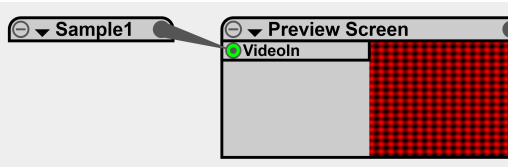

図 11　データ調整ノブ有りのタスク 1

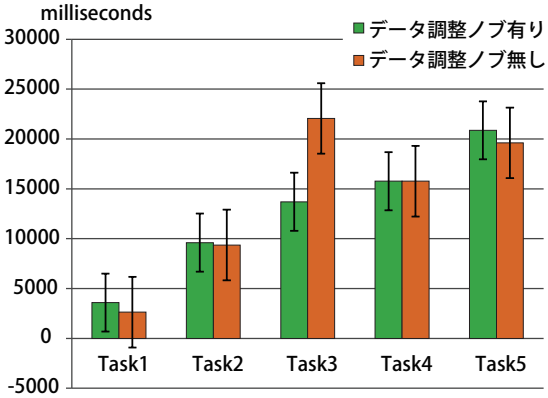

図 12　実験結果：データ調整ノブ有りと無しのそれぞれのタスクにかかった平均時間

の調整が一回づつ必要である。

作業 2 と作業 3 の時間計測の目的は、第 3 節にて述べた、データ調整ノブの有効性を検証するためである。作業 2 はデータ調整ノブ有りの場合はノブの調整が 1 回、データ調整ノブ無しの場合は結線とスライダーの調整が 1 回づつからなり、作業 3 はデータ調整ノブ有りの場合は結線とノブの調整が 1 回づつ、データ調整ノブ無しの場合は結線が 3 回とスライダーの調整が 1 回からなる。

実験結果を図 12 に示す。図 12 では、緑はデータ調整ノブ有り、橙はデータ調整ノブ無しのそれぞれのタスクにかかった平均時間をミリ秒で示している。エラーバーは標準偏差である。分散分析の結果、タスク 3 にのみ有意な差が見られた ($F(7,1) = 77.449, p < 0.01$)。

当初、我々はデータ調整ノブ導入によって作業 1 の所要時間が伸びると考えていた。しかし、作業 1 が 1 回のみからなるタスク 1 には有意な差は見られなかった ($F(7,1) = 0.637, p > 0.1$)。このことから、0.0 から 1.0 へのノブの調整は問題では無いと考えられる。

有意差が見られたタスク 3 は作業 3 を 2 回含んでおり、作業 2 は含んでいないことから、データ調整ノブは作業 3 において有効であるといえる。また平均値も、データ調整ノブ有りの場合 22.1 秒、データ調整ノブ無しの場合 13.7 秒と 10 秒近く差があり、データ調整ノブの有用性を示している。

5　関連研究

平井らは、データフロービジュアル言語 Max において音声によってノードを作成する手法を開発した [3]。データフローの編集効率の改善という点で本研究と類似しているが、本研究のデータ調整ノブはノードとエッジの作成の機会そのものを減らすというアプローチを採っている点で異なる。

ライブパフォーマンス向けのデータフロービジュアル言語はいくつか開発されており、パラメータの調整やデータフローの編集において様々な工夫がなされている。

Jordà らの reacTable [5] や、Taylor らの VPlay [6] はそれぞれ、音楽と映像のライブパフォーマンス向けデータフロービジュアル言語であり、ノードを近づけると自動で結線するというデータフロー編集手法を持つ。これは従来のポインタによるターゲッティングによる結線より、手間が軽減されているが、一方でノードのレイアウトを制限する上、ユーザの意図しない結線が生じてしまう。データ調整ノブはノードとエッジの作成の機会を減らすが、そのような副作用は生じない。

Bencina の開発した AudioMulch [4] はライブパフォーマンス向け音響処理データフロービジュアル言語である。AudioMulch のデータフローが扱うデータ型は音響信号のみであり、パラメータは別のウィンド

ウにノブやスライダとして配置される。データ型を一種類に限定する点は本研究のアプローチと類似するが、ImproV の映像型とデータ調整ノブの組み合わせは、処理対象のデータとパラメータのデータを同じデータフロー上にて処理できる点が大きく異なる。

6 まとめ

我々は、我々が開発したライブ映像パフォーマンス向けの映像処理データフロービジュアル言語 ImproV において、データフロー編集操作を削減するために、データ調整ノブを開発した。

データ調整ノブを導入した ImproV においてユーザは、エフェクトのパラメータ調整と映像の透明度の調整の２つの操作を、定数ノードや透明度変更ノードの組み合わせを使って調整する従来の手法よりも簡潔に行うことができる。また、我々は被験者実験を行い、その結果、映像の透明度の調整にはデータ調整ノブが有効であることを確かめた。

このデータ調整ノブの導入は、一般のデータフロービジュアル言語においても、ノードの作成や結線の操作を行う機会を削減することができ、有効であると考えられる。

参 考 文 献

[1] Atsutomo Kobayashi, Buntarou Shizuki and Jiro Tanaka, “ImproV: A system for improvisational construction of video processing flow”, *in Proceedings of 13th international conference on Human-computer interaction*, Part IV, LNCS 5611, 2009, pp. 534–542.

[2] 小林敦友, 志築文太郎, 田中二郎, GPU を利用したライブ映像パフォーマンス向け映像合成システム, 情報処理学会論文誌 プログラミング (PRO), Vol.4, No.1, 2011, pp. 76–89.

[3] 平井重行, 田中秀明, 音声入力によるビジュアルプログラミング環境操作支援, インタフェース情報処理学会研究報告 (128 回ヒューマンコンピュータインタラクション研究会), Vol.2008, No.50, 2008-HCI-128, 2008, pp. 19–24.

[4] Ross Bencina, “Oasis Rose the composition - real-time DSP with AudioMulch,” *in Proceedings of the Australasian Computer Music Conference*, 1998, pp. 85–92.

[5] Sergi Jordà, Günter Geiger, Marcos Alonso, Martin Kaltenbrunner, “The reacTable, exploring the synergy between live music performance and tabletop tangible interfaces,” *in Proceedings of the 1st international conference on Tangible and embedded interaction*, 2007, pp. 139–146.

[6] Stuart Taylor, Shahram Izadi, David Kirk, Richard Harper, Armando Garcia-Mendoza: “Turning the tables, an interactive surface for VJing”, *in Proceedings of the 27th international conference on Human factors in computing systems*, 2009, pp. 1251–1254.