# 自動演奏と映像の同期による音楽表現の拡張

福知山公立大学 情報学部情報学科

32245100　矢野瑞季

指導教員　橋田光代　准教授

提出日　2025 年 1 月 17 日

改訂日　2025 年 1 月 31 日

## — 目次 ———

**1 はじめに 1**

**2 システムの概要 1**

**3 演奏曲について 1**

**4 Max ８で制作したシステムについて 2**
4.1 Max8 からデータを送信する方法 . . . . . 2

**5 映像制作 2**
5.1 Processing について . . . . . 2
5.2 コードの説明 . . . . . 3
5.3 デザインについて . . . . . 3
5.4 処理速度の課題 . . . . . 4

**6 今後の発展 4**
6.1 システムの改善点 . . . . . 4

**7 まとめ 4**

## — 図目次 ———

1 システムの概要図 . . . . . 1
2 「エオリアンハープ」楽譜 . . . . . 1
3 「 エオリアンハープ」楽譜の一部 . . . . . 1
4 演奏者が弾く楽譜の一部 . . . . . 1
5 「Max ８で構築したプログラム . . . . . 2
6 テンポ計測 . . . . . 2
7 Max8 からデータを送る . . . . . 2
8 映像処理についての概要図 . . . . . 3
9 製作した映像 . . . . . 3

## 1. はじめに

私はこれまでの PBL 研究で、Max8 を使用し音源の制作や、コードで奏でる自動伴奏システムの制作に取り組んできた。今年度はこの経験を生かし、自動演奏と映像を組み合わせた作品の制作を行った。具体的には、次の 3 つを研究の目標とした。

(1) Max8 で自動演奏システムを作ること
(2) Processing というプログラミングソフトを使い、音楽に合う映像を制作すること
(3) Max8 で作ったシステムと Processing で制作した映像を連動させ映像音楽を作ること

この研究を通じて、自動演奏システムと映像との連動がもたらす表現の可能性を探ることを目指した。

## 2. システムの概要

本研究で構築したシステムは、演奏者の電子ピアノの打鍵動作を入力信号として受け取り、それをトリガーとして音楽を生成する Max8[1] と、音楽に連動した映像を制作する Processing[2] を組み合わせたものである (図 1 )。演奏者が電子ピアノの特定の鍵盤を押すと、その打鍵動作がトリガー信号として Max8 に伝達され、システムが作動して音楽が自動生成される。同時に、その音楽データは UDP（ユーザデータグラムプロトコル）を通じて Processing に送信され、リアルタイムで映像が生成される仕組みとなっている。

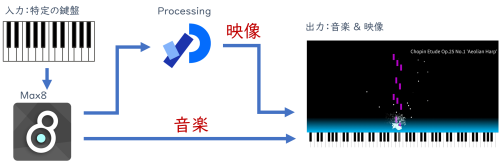


**図 1:** システムの概要図

## 3. 演奏曲について

今回の作品で使用する楽曲は、フレデリック・ショパン作曲のエチュード Op.25-1「エオリアンハープ」[3] である。この楽曲のテンポ（BPM）は 104 であり、演奏には非常に速い指の動きが要求される。しかし、このテンポで滑らかに演奏することから生まれる、流れるような音の美しさこそが、この曲の大きな魅力であると考える。

この曲を選んだ理由は、まず楽譜（図 2）を見ると、ほとんどが十六分音符で構成されており、プログラミングが比較的容易であると判断したからである。また、私自身がショパンのエチュードの中でこの曲を最も気に入っており、絶え間なく奏でられる美しいアルペジオの魅力を最大限に引き出したいと感じたからである。

今回の研究では、楽譜の特徴を活用し、演奏者の負担を軽減するための自動演奏システムを構築した。具体的には、十六分音符が 6 つまとまったフレーズのうち、最初の音（図 3 の赤丸で示された部分）のみを演奏者が弾けばよい仕組みを採用した。システムはこの最初の音を検知すると、残りの 5 つの音を生成し、演奏者の弾いた音に自然に続くように再生される。この方法により、演奏者の指の動きを補助しつつ、楽曲の持つ滑らかさとテンポ感を維持することを目指した。図 4 はシステムを利用する際、演奏者が実際に演奏する楽譜の一部であり、元の楽譜（図 3）と比べ演奏が容易になっている。

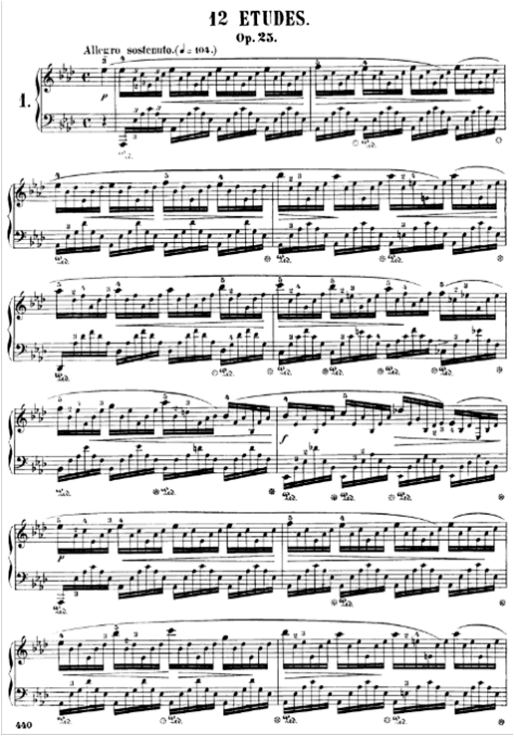


**図 2:** 「エオリアンハープ」楽譜

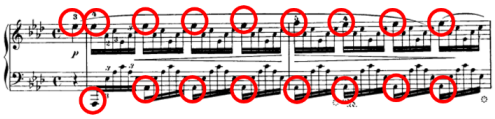

**図 3:** 「エオリアンハープ」楽譜の一部

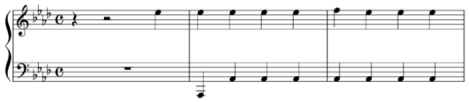

**図 4:** 演奏者が弾く楽譜の一部

## 4. Max 8 で制作したシステムについて

パソコンのキーボードや電子ピアノの鍵盤を使い、赤丸で囲まれた音符に対応するキーや鍵盤を押すと、その音に続くフレーズが自動的に再生されるようにプログラムされている。具体的には、最初の 6 連符「E ♭, A ♭, C, E ♭, A ♭, C」の場合、最初の「E ♭」が打鍵されるとシステムが作動する。Max8 の metro オブジェクトを使用して指定されたテンポに従って、図 5-A の部分でリストに格納された数値が順に処理され、それに対応する音が生成される仕組みになっている。


**図 5:** 「Max 8 で構築したプログラム

このシステムは右手パートと左手パートに分けて設計し、それぞれ個別に制作した。またテンポについては、演奏者の好みや技術レベルに柔軟に対応できるよう工夫した。具体的には、図 5-B で 1 つ前の音が押されてから次の音が押されるまでの時間を計測し、その時間を 6 等分することで、テンポを自然に調整できる仕組みを取り入れている（図 6）。これにより、演奏者は自身の演奏スタイルに合わせて、滑らかな演奏が可能となる。

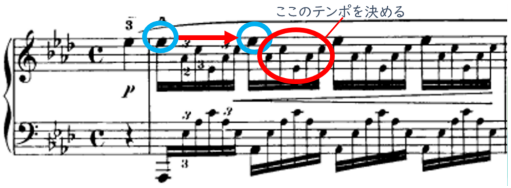


**図 6:** テンポ計測

### 4.1 Max8 からデータを送信する方法

Max 8 から外部のソフトにデータを送信するために「udpsend」というオブジェクトを使用した。「udpsend [IP アドレス] [ポート番号]」と指定することで、UDP を介してデータをネットワーク経由で送信している（図 7）。具体的には、Max8 から Processing に向けて、指定した IP アドレスとポート番号にデータを送る仕組みを構築している。この方法により、Max8 で生成されたデータを Processing が受信し、リアルタイムな連携を実現している。

また、図 7 の「prepend /abc」は、メッセージの先頭に「/abc」というシンボルを付加するオブジェクトである。これにより、Max パッチ内で他のオブジェクトから送られてきたメッセージを「/abc」で始まる形式に変換し、その後にデータや引数を追加することが可能となる。たとえば、他のオブジェクトから「hello」というメッセージが送られた場合、「prepend /abc」を通過すると、Max パッチ内では「/abc hello」という形式のメッセージに変換される。


**図 7:** Max8 からデータを送る

## 5. 映像制作

### 5.1 Processing について

Processing はビジュアルアートに適したプログラミング言語であり、同時に IDE（総合開発環境） でもある。基本的に Java に似ているが、Java より記述がシンプルで、Processing 独自のルールも存在している [4]。

Processing を利用するメリットは、グラフや図形をアニメーションで簡単に実行できる点である。例えば、C 言語や Python などの他のプログラミング言語でアニメーションを実行しようとすると、専用のライブラリやフレームワークをインストールする必要があり、さらに複雑な設定や準備が求められることが多い。しかし、Processing ではそのような手間がなく、単体でアニメーションを実行するための機能が備わっている。そのため、すぐに視覚的に結果を確認することができ、開発の進行が非常にスムーズである。

また、Processing は視覚的な表現に特化しており、図形やグラフ、アニメーションを簡単に作成できる。そのため、コードの記述に集中しやすいという利点がある。さらに、Processing には基本的な描画機能に加え、デフォルトで豊富な機能が備わっている。例えば、画像の読み込み、音声の再生、センサー入力などを簡単に実装できる。これらの機能が最初から備わっているため、複雑なライブラリの追加や、設定を行う手間が省ける。したがって、初心者でもある程度のソースコードを記述するだけで、目的の結果を得ることができる。

一方で、Processing にはいくつかのデメリットも存在す

る。最も顕著な点は、実行速度が遅いということである。Processing はインタプリタ型の言語であり、コンパイル型言語である C 言語や C ＃と比較すると、実行速度が遅くなる傾向がある。また、処理能力が高く求められる場合、特に複雑な計算やリアルタイムでの音声・映像処理においては、パフォーマンスの低下が見られることがある。このため、音楽と映像を同期させるようなシステムにおいて、ラグが発生する可能性がある。音楽と映像がリアルタイムで密接に連携する場合、そのタイミングのズレが視覚的にも聴覚的にも影響を与える可能性があり、これが一つの懸念点となる。

### 5.2 コードの説明

本システムは、Processing を使用してピアノの鍵盤インターフェースを描画し、鍵盤が押された際に視覚効果を生成する機能を提供するものである。映像処理についての概要図は図 8 に示す。コードは主に以下の部分に分かれている。

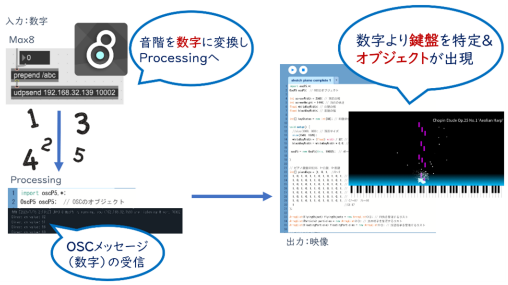


**図 8:** 映像処理についての概要図

まず、setup() メソッドにおいて、画面サイズを 2560x1320 ピクセルに設定し、鍵盤の幅を計算する。さらに、OscP5 ライブラリを使用して、外部ソフトウェアやハードウェアからの OSC(Open Sound Control) メッセージ [5] を受信する準備も行う。

次に、drawKeyboard() メソッドにより 88 鍵のピアノ鍵盤が描画される。この部分では、白鍵と黒鍵の幅を計算し、各鍵を描画している。鍵盤が押されると色が変わり、視覚的に押された状態が示される。

視覚効果については、FlyingObject クラスが担当し、鍵盤が押されると四角形が画面上に現れ、上方向に移動しながら色が変化する。また、Particle クラスは炎のような粒子を生成し、指定された速度で動かしながら透明度を減少させ、視覚的な動きを加える。さらに、FloatingParticle クラスはランダムに浮遊する粒子を生成し、上昇しながら徐々に消えていく効果を生み出す。

oscEvent() メソッドでは、外部から送られてくる OSC メッセージを受信し、方向値を解析して押された鍵盤のインデックスをマッピングする処理を行う。このインデックスに対応した鍵盤が押されると、その鍵盤に関連する視覚効果が生成される。

ユーザーインタラクションは、keyReleased() メソッドによって管理される。ユーザーが鍵盤のキーを離した際、鍵盤の状態がリセットされる。また、mapKeyToIndex() メソッドでは、受信した方向値に基づいて押された鍵のインデックスを計算し、そのインデックスに対応する視覚効果が発生する仕組みとなっている。

全体として、このシステムは、ピアノの鍵盤を押すことで視覚効果を生成し、OSC メッセージに反応してインタラクションを管理する仕組みを提供するものとなっている。

### 5.3 デザインについて

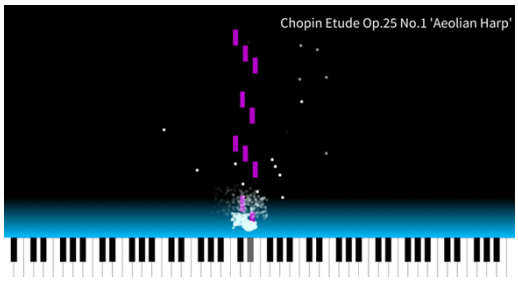


**図 9:** 製作した映像

本システムのデザインは、視覚的に魅力的でインタラクティブなピアノ鍵盤インターフェースを提供することを目標としており、デザインは以下の要素を中心に構成されている（図 9）。

まず、88 鍵のピアノ鍵盤が画面下部に水平に配置され、白鍵と黒鍵が正確に描画される。白鍵の幅は画面幅に応じて動的に計算され、黒鍵は白鍵の 60 %の幅で表示されるため、実際のピアノ鍵盤に忠実なレイアウトが実現されている。鍵盤が押されると、白鍵は少し暗く、黒鍵は明るく表示されることで、どの鍵盤が押されたのかが視覚的に明確に示され、インタラクティブな体験が強調される。

視覚効果としては、鍵盤が押されると FlyingObject クラスによって四角形が画面上に飛び出し、色が虹色に変化する。これにより、音楽が演奏される感覚が視覚的に強調される。また、Particle クラスによって炎のような粒子が放出され、FloatingParticle クラスではランダムに浮遊する粒子が生成され、押された鍵盤に対して動的な反応が表現される。視覚的な特徴として、背景は黒色に設定され、鍵盤や視覚効果は白や虹色、青色などの鮮やかな色で描画される。これにより、高いコントラストが生まれ、視覚的に目を引く効果が得られ、画面上部にはタイトル「Chopin Etude Op.25 No.1 'Aeolian Harp'」が表示されるようになっている。

このシステムでは、音楽演奏に伴う動的な視覚効果を取り入れ、鍵盤操作に連動したインタラクティブな体験を実現することを目指して、デザイン設計を行っている。鍵盤が押されるたびに視覚効果が現れることで、音楽と視覚表

現を組み合わせた新しい表現の形を模索している。

### 5.4 処理速度の課題

最初は背景に動画を使用することを考えていたが、処理が重くなる問題が発生した。試しに画像を背景として使って実行したところ、四角形の動きが非常に遅くなった。画像はできるだけ圧縮して軽くした上で実行してみたが、依然としてうまく動作しなかった。そのため、動画を背景として使用するのは難しいと判断した。その後、背景をグラデーションに変更しようとした際にも、同様に動作が遅くなる問題が発生した。そのため、背景デザインはシンプルなものにしたが、現状の方法では満足できていないため、より効率的な方法を模索している。

## 6. 今後の発展

最終的には、映像をプロジェクターで投影し、それをピアノと連動させた形で演奏を行うライブパフォーマンスを実現したいと考えている。このシステムでは、音楽と映像のシンクロ性を重視し、視覚と聴覚を融合させた新しい表現方法を追求していきたい。演奏者が楽曲を弾くと、それに連動して映像がリアルタイムで変化する仕組みを構築し、演奏と映像が一体となった没入感のあるパフォーマンスを目指している。また、このシステムは多くの人に利用してもらえるものにしたいと考えており、幅広いユーザー層に対応できる設計を目指している。

### 6.1 システムの改善点

システムの改善点として、現時点では自分自身が使用することを前提に設計を行っているため、以下の問題点を詳細に検討する必要がある。

まず、楽譜が読めない利用者でも操作可能なシステムとするか、それとも一定以上の演奏技術を有する利用者を対象とするかという点について、利用者層の明確化が必要である。この点を明確にすることにより、システムの目的や設計方針がより具体的に定まると考えられる。

次に、演奏中に発生するミスへの対応が現状では考慮されていない点が挙げられる。例えば、演奏者が途中で間違えた場合に、その場で修正し再開できる仕組みを整える必要がある。また、演奏の途中から開始したい場合や特定の箇所のみを繰り返して練習する機能についても、現時点では未対応であるため、これらの機能も実装する必要があると考えている。

さらに、現在のシステムでは、演奏中にミスが生じた際に全て最初からやり直さなければならない設計となっており、この点は生演奏の実用性を著しく損なう要因となっている。生演奏に対応するシステムとするためには、リアルタイムでの柔軟な対応や、演奏の進行状況を記録し、途中から再開できる仕組みの導入が必要不可欠である。

以上のように、利用者層の明確化や演奏中の柔軟な対応を含むシステムの設計改善を進めることで、より多くの利用者が快適に使用できる実用的なシステムを目指すことができると考えている。

## 7. まとめ

本研究では、Max8 を用いた自動演奏システムと Processing を活用した映像作品を組み合わせた表現の可能性を探求した。具体的には、ショパンのエチュード Op.25-1 を基に、演奏者の負担を軽減するシステムを構築し、それに連動した映像を制作することで、音楽と視覚が融合する新しいパフォーマンスを提案した。

研究を通じて、自動演奏システムの応用可能性や Processing を用いた映像制作の利便性を確認できた一方で、システムの処理速度やリアルタイム性といった課題も明らかとなった。これらの課題は、演奏者の技術や目的に応じた利用者層の明確化や、演奏中のエラー対応機能の実装など、今後の発展的な取り組みを通じて改善をしていく必要があると考える。

今後は、音楽と映像が完全に同期するリアルタイムシステムを目指し、ライブパフォーマンスに対応できるシステムの実現を目標とする。特に、システムの処理速度や安定性を向上させることで、演奏と映像の同期精度を高め、視覚的な遅延を減少させる必要があると考える。さらに、多様なユーザーが利用できる汎用性の高い設計を追求し、音楽と映像の新たな表現方法を提供することで、幅広い人々に感動を与える作品を制作したい。また、将来的には、インタラクティブな要素を追加し、ユーザーが自身の演奏に対してリアルタイムで反応する新しい形態のパフォーマンスを構築したいと考えている。

### 参考文献

[1] Max 8 | Cycling ’74, https://cycling74.com/products/max8.
[2] Processing: Welcome to Processing!, https://processing.org//.
[3] : 練習曲作品 25-1 (ショパン), *Wikipedia* (2023).
[4] Processing でできること 4 選！メリットや使い方もわかりやすく解説 - WEBCAMP MEDIA, https://webcamp.io/magazine/archives/96159/.
[5] Qiita: OSC とは？ OSC 通信についてまとめてみた, https://qiita.com/generosity_honda/items/904aaeb382f6496ab920 (2023).