Pure Data を用いた音響信号処理に関する高校生向け模擬講義の試み

神澤祐紀子　北村達也
甲南大学知能情報学部

概 要

Pure Data は，一般的なプログラミング言語とは全く異なり，GUI 上で種々の機能をもつオブジェクトを線でつなぐことによって所望の機能を実現する．本稿では，Pure Data を用いた高校生対象の音響信号処理の模擬講義について報告する．Pure Data を用いることによって，プログラミングの知識の有無や技量の優劣にかかわらず，音響信号処理による音の変化を体感させることができた．その一方で，音響信号処理の仕組みの理解につなげるのは困難であった．
【キーワード】Pure Data，グラフィカル言語，パッチ，音響信号処理

1　はじめに

音響信号処理は，学習者自身の耳で処理結果を確認できたり，日頃の音楽鑑賞で使われている効果を体験できたりする点において，学習者に興味を持たせやすい．しかし，その一方で，音響信号処理が数式で記述されることや，それを (ハノイの塔のように) 手作業で確認できるわけではないことから，処理の仕組みを理解させることは難しい．

日本音響学会音響教育調査委員会は，音響教育に関するアンケートと大学におけるシラバス調査を行い，2009 年にその結果を報告した [1]．それによると，「現場見学・体験ができるもの」，「音の大切さを実感できるもの」，「信号処理などを実際に行う演習プログラム」のような，音に興味を持たせるための導入プログラムと実際に信号処理などを体験させる演習的なものを望む意見が目立った．

このような背景から，学習者に音響信号処理のプログラムを作成させることを通して処理の仕組みの理解と体験させる教育が検討されている [2]．しかし，この方法には，プログラミングの知識をもつ学習者しか対象にできないという制約がある．たとえ学習者にプログラミングの知識がある場合でも，クラスの中でプログラミング技量に差があると，そのことへの対応に時間が割かれてしまう可能性がある．そして残念ながら大学の演習でもそういう場面は少なくない．

堀内 [3] は，文法学習が少なく短時間で習得しやすいグラフィカル言語 Pure Data [4][5][6] を用いてフーリエ級数の可視化をしたり，音として実際に聞かせたりした授業が学生から評価を得られた事例を報告している．Pure Data は，Miller Puckette が開発したオープンソースのグラフィカルなプログラミング環境で，音の処理が比較的簡単に実現できる[1]．そこで，本研究では Pure Data を用いて音響信号処理の実習案を立案し，高校生を対象とした模擬講義を試みた．

本稿では，まず Pure Data のプログラミングについて概説した後，模擬講義の内容および講義後に得られたアンケート結果について報告する．

[1] このほか，画像処理や Arduino [7] の制御も可能である [6].

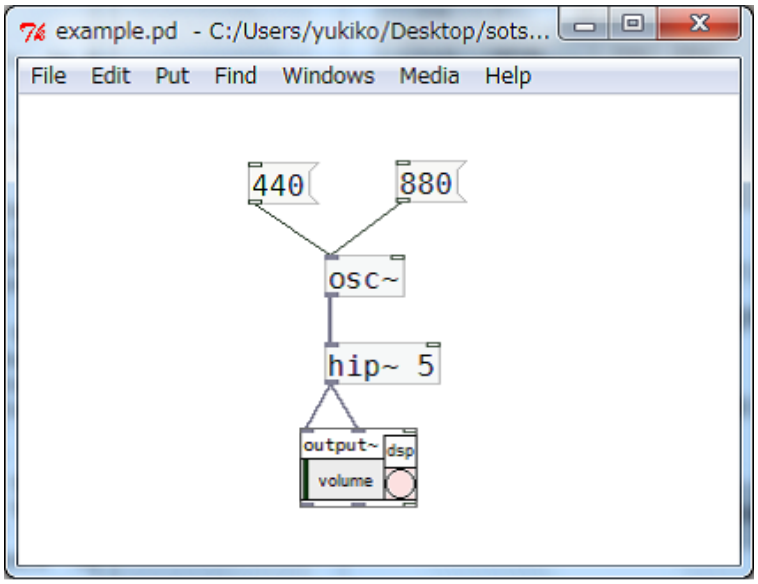

図 1: 440 Hz または 880 Hz の純音を提示するパッチ

2 Pure Data のプログラミング

2.1 基礎

Pure Data で書かれたプログラムは「パッチ (patch)」と呼ばれる．Pure Data のパッチの例を図 1 に示す．このパッチは 440 Hz または 880 Hz の純音を鳴らすものである．440，880 のいずれかのオブジェクトをマウスでクリックすることによって周波数を選択する．`[osc〜]` は正弦波の発振器，`[hip〜 5]` はカットオフ周波数 5 Hz のハイパスフィルタであり，一番下のオブジェクトは音の再生を行う機能をもつ．なお，このハイパスフィルタは，スピーカーやヘッドホンに直流電流 (DC) が長時間加わることによってボイスコイルが発熱するのを避けるためのものである [6].

オブジェクト同士が線で結ばれていると，上から下へデータが流れる．数値や `bang`（データの流れを開始させるスタートボタンのような役目）といったメッセージが流れている接続線は細く，「音」が流れる接続線 (audio connection) はやや太めになっている [6]．図 1 の例では，数値と `[osc〜]` を接続する線は細く，`[osc〜]` から `[hip〜 5]`，`[hip〜 5]` から `[output〜]` を接続する線は太くなっている．それぞれのオブジェクトは数値データを出力したり，2 つの入力を合成したりすることによって計算結果を出力する．最終的に，これを波形に変換して音を出力する．

オブジェクト作成時は編集モードと呼ばれ，カーソルが指のマークになっている．実行時には `Ctrl+e` でカーソルを矢印のマークにし，実行モードにする．実行モードではオブジェクトや数値を変更することはできない．

以上のように，Pure Data はデータや処理の流れを視覚的に表現でき，音符や数式を直接扱わないことから，誰でも簡単に音の制作を始めることができる．

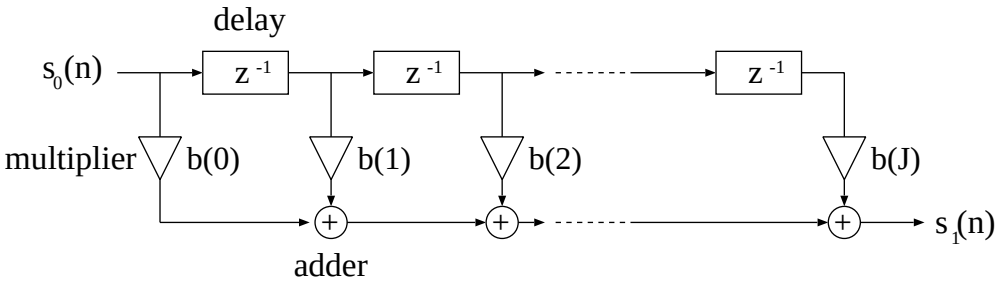

図 2: リバーブのブロック図

2.2 サウンドエフェクタの作成例

2.2.1 リバーブ

ここではリバーブ (reverb) というサウンドエフェクタの作成例を示す．リバーブとは，現在の時刻の音データと，それよりも前の音データを同時に再生することによって，残響効果を作り出す音響信号処理である [8].

現在の時刻の音データを $s_0(n)$，過去の音データを $s_0(n-1), s_0(n-2), \ldots, s_0(n-J)$ とすると，これらを同時に再生した音データ $s_1(n)$ は一般に次のように表される．ここで，J は遅延器の数である．

$$s_1(n) \quad = \quad b(0)s_0(n) + b(1)s_0(n-1) + b(2)s_0(n-2) + \cdots + b(J)s_0(n-J) \qquad (1)$$

ここで，$b(0), b(1), b(2), \ldots, b(J)$ は，それぞれの音データにかけ合わせる重みを表している．この式をブロック図で表すと図 2 のようになる．これは,「乗算機」,「加算機」,「遅延機」という基本三要素によって構成されたデジタルフィルタとなっている．

遅延時間が長いと本来の音と過去の音が分離してやまびこのように聞こえるディレイ (delay) と呼ばれるものになるが，遅延時間を短くするとひとつの音に融合し，コンサートホールのように聞こえるリバーブとなる [8].

指定したファイルから読み込んだ音にリバーブをかけるパッチを図 3 に示す．ファイル名 (ここでは `beat_it.wav`) とデータを保存する配列名 (ここでは `sound`) を指定したメッセージ `[read]` を `[soundfiler]` に与えることによって，音データを配列に読み込むことができる．

パッチの左側にある `bang` (四角の中に円が描かれたオブジェクト) をクリックすると，`[tabplay〜]` が配列の内容を読み出し音データとして再生する．`[delwrite〜]` は，名前 (ここでは `reverb`) と時間 (ここでは 1000 ms) を与えることによって，指定時間分の音データを格納するディレイライン (delay line) が作られる．`[delwrite〜]` に入力された音は，常に最新の 1000 ms 分がディレイライン上に保持される．

一方，パッチの中央にある `[delread〜]` は，ディレイラインから音を読み出すオブジェクトである．`[delread〜]` は，指定された時間だけ遅延した音データをディレイラインから読み出す．そして，`[+〜]` で元の音と遅延した音を加算することによって，多重化されたリバーブを実現している．

このパッチでは，音の遅延時間 (`time`) と，遅延された音の大きさ (`level`) にスライダを使用している．これにより実行時にマウスを用いるて音の遅延時間と遅延された音の大きさを調節することができる．`time` と `level` の可動範囲はそれぞれ 0〜1000 ms，0〜1 である．

このパッチを実行するためは，beat_it.wav という音のファイルを準備する必要がある．そして，まず `[output〜]` の volume を上げ，`[read]` メッセージをクリックしてファイルを配列に読み込む．次

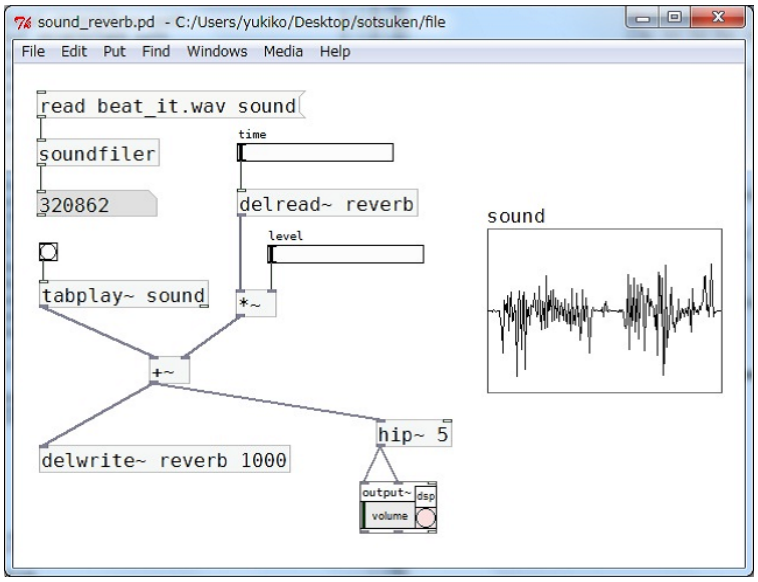

図 3: リバーブのパッチ

に，bang をクリックすることによって，読み込んだ音が 1 度再生される．その間にスライダで time と level を調節すれば，リバーブの効果を変化させることができる．

2.2.2 ハードクリッピング

ハードクリッピング (hardclipping) とは，音を極端に増幅し，音の波形の上端と下端を意図的に潰すことによって歪んだ音色を作り出すサウンドエフェクトのことである [8]．この処理はディストーションと呼ばれる処理の一種である．

ハードクリッピングによるディストーションは次のように定義される．

$$s_1(n) = \begin{cases} 1 & (gain \cdot s_0(n) \geq 1) \\ gain \cdot s_0(n) & (-1 < gain \cdot s_0(n) < 1) \\ -1 & (gain \cdot s_0(n) \leq -1) \end{cases} \tag{2}$$

ここで，$s_0(n)$ は入力信号，$s_1(n)$ は出力信号，$gain$ はギターアンプの増幅率を表す．なお，増幅の限界を 1 で正規化し，$|s_0(n)| \geq 1$ とする [8]．

読み込んだ音にハードクリッピングを施すパッチを図 4 に示す．リバーブのパッチと同様，ファイル名と配列名を指定した [read] メッセージを [soundfiler] に与え，音データを配列に読み込ませる．[tabplay〜] にて配列に読み込んだ音を再生し，[clip〜] によって，振幅を −0.3 から 0.3 の範囲に制限している．

実行時はリバーブと同様，[output〜] の volume を上げ，[read] メッセージをクリックしてファイルを配列に読み込む．次に，[tabplay〜] の上にある bang をクリックすることによって，読み込んだ音にハードクリッピングがかかった音が再生される．

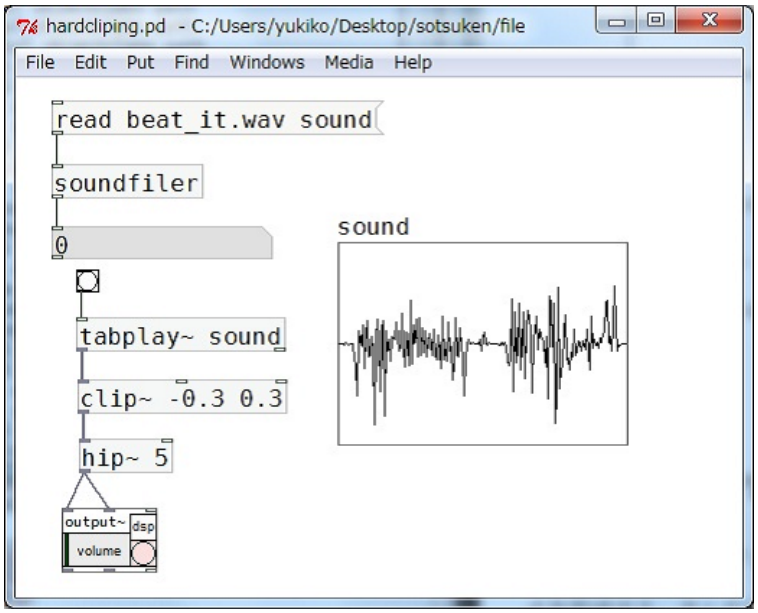

図 4: ハードクリッピングのパッチ

3　高校生を対象にした模擬講義

3.1　概要

Pure Data を用いて音響信号処理に関する模擬講義を実施した．模擬講義は 2012 年 11 月 16 日，鳥取県立倉吉総合産業高等学校の情報科 2 年生 35 名を対象に，本学 14 号館地下多目的レクチャールームにて約 50 分間行った．

3.2　実習環境

実習には Windows XP/7 のノート PC を用いた．3 名を 1 班とし，ノート PC を班に 1 台の割合で配置した．実習の途中で操作を交替させることによって全ての生徒が Pure Data を体験できるよう配慮した．使用した Pure Data のバージョンは 0.42.5 である．当日の様子を図 5 に示す．

3.3　模擬講義の内容

模擬講義は以下のような内容で実施した．1，2 は第 2 著者が担当し，3 以降は第 1 著者が担当した．

1. 学部および研究室の紹介
2. 講義内容の説明
3. Pure Data の概説
    - Miller Puckette が開発したオープンソースのグラフィカルなプログラミング環境
    - オブジェクトと呼ばれる箱を線でつないでプログラムを作成
    - プログラミングを経験したことがなくても簡単に始められる
    - 自宅でもダウンロードして使うことができる

図 5: 模擬講義の様子

4. Pure Data の操作法の解説
    - パッチは上から下へ流れる
    - 編集モードと実行モードがあり，各モードは `Ctrl+e` で切り替える
    - モードの違いはカーソルに現れる
    - パッチを作るときは、編集モード
5. ハードクリッピングのパッチの作成
    - ウインドウを立ち上げ，新規作成
    - 各オブジェクトの作成
    - オブジェクト同士を接続
    - 保存
6. パッチの実行
    - 各オブジェクトの説明
    - 編集モードから実行モードへ切り替え
    - 音を再生し，元の音と比較
7. まとめ
    - Pure Data には他にも様々なオブジェクトが用意されている
    - Pure Data では音だけでなく画像の処理も手軽にできる
    - この機会に少しでも Pure Data に興味を持ってもらえると嬉しい
8. アンケート

3.4 アンケート結果

アンケートのうち選択式の部分の結果を表 1 に示す.

表 1: アンケート結果 (選択式)

No.	質問	回答		
1	Pure Data を今日初めて知りましたか？	はい 35 名	いいえ 0 名	
2	音の信号処理の経験はありますか？	はい 24 名	いいえ 11 名	
3	Pure Data の使い方は理解できましたか？	はい 30 名	いいえ 5 名	
4	一般的なプログラミング言語 (C，Java など) と比べ，理解し易かったですか？	はい 31 名	いいえ 4 名	
5	講義の難易度はいかがでしたか？	簡単 31 名	普通 1 名	難しい 3 名
6	講義のスピードはいかがでしたか？	早い 13 名	普通 15 名	遅い 7 名
7	実習を通して信号処理の仕組みを理解できましたか？	はい 19 名	いいえ 16 名	
8	Pure Data をまた使ってみようと思いましたか？	はい 31 名	いいえ 4 名	
9	信号処理に興味を持てましたか？	はい 32 名	いいえ 3 名	

全ての生徒が Pure Data を初めて体験したが (問 1)，全ての班でパッチを完成させることができた．30 名 (86 %) の生徒が Pure Data の使い方を理解できたと回答している (問 3)．また，参加者たちは高校の授業でプログラミングを学んでいたが，一般的なプログラミングに比べ Pure Data の操作を簡単だと感じた生徒が 31 名 (89 %) いた (問 4)．これらの結果は，Pure Data が修得しやすく時間に制約のある実習でも効果を上げやすいことを示唆している．

さらに，Pure Data を 31 名 (89 %) の生徒がまた使ってみたいと答えている (問 8)．この実習によって少なくとも Pure Data を使うきっかけを作ることはできたといえる．

実習前，音の信号処理の経験があった生徒は 24 名 (69 %) であったが (問 2)，実習後，信号処理に興味を持てたという生徒が 32 名 (91 %) に上ったことから (問 9)，信号処理に関する興味を喚起する効果はあった．一方，信号処理の仕組みを理解できたと答えた生徒は 19 名 (54 %) にとどまり (問 7)，課題があることが明らかになった．

「今日の模擬講義の感想を書いてください」という自由記述式のアンケートに対しては以下のような意見が寄せられた．

- 初めて使ったが，C 言語より簡単にできた．
- 楽しかったので，また使ってみたいと思った．
- 少ない作業で音が鳴るのはすごいなと思った．
- 家でまた使ってみようと思った．
- 思ったよりずっと簡単だった．
- これなら誰でも作れると思うので普及してほしい．

- 短い時間でも出来たので楽しかった．
- 他の楽器の音も作りたいと思った．
- プログラム言語を覚えるより簡単で楽しかった．
- 難しそうな印象が強かったが，実際に作ってみると楽しかった．
- 初めてのことだらけであまり理解できなかった．
- プログラムを作るときの表示方法をもう少しゆっくり教えてほしかった．
- オブジェクトに書かれたことによって，どんな風に変わったのか説明が欲しかった．
- 簡単にできたが，仕組みは理解できなかった．

これらのコメントも上記の選択式アンケートの回答と一致する．すなわち，ほとんどの生徒にとって Pure Data の使用法は簡単で，音も簡単に出るため楽しかったが，処理の中身の理解にまで至るのは困難ということである．今後，この点を改善する実習案を検討する必要がある．

4 おわりに

高校生を対象にして音響信号処理に関する模擬講義を実施した．Pure Data というグラフィカル言語を用いることによって，プログラミングや音響信号処理の敷居を下げ，楽しい実習が実現できた．しかし，音響信号処理の仕組みを理解してもらうことは困難であった．この一因として，2.2 節に示したように，音響信号処理を表す数式やブロック図と Pure Data によるパッチの見た目が対応していないことが考えられる．その意味では，GUI でブロック図を作成することによってプログラムとして動作する Simulink (Mathworks 社の Matlab の機能の 1 つ) の方が適している可能性がある．今後も引き続きよりよい実習の実現のための検討を行っていく．

参考文献

[1] 音響教育調査委員会, 音響教育に関するアンケートと大学におけるシラバス調査の結果, 日本音響学会誌, 65(5), 264–269 (2009).

[2] 青木直史, サウンドエフェクトを題材としたプログラミング演習, 電子情報通信学会技術研究報告 (音声), 109(99), 35–38 (2009).

[3] 堀内泰輔, 応用数学授業における ICT 活用の実践, 長野工業高等専門学校紀要, 41, 57–60 (2007).

[4] Pure Data – PD Community Site, `http://puredata.info/`

[5] Andy Farnell, *Designing Sound*, The MIT Press (2010).

[6] 中村文隆, グラフィカル言語 Pure Data による音声処理, CQ 出版社 (2009).

[7] Arduino – HomePage, `http://www.arduino.cc/`

[8] 青木直史, C 言語ではじめる音のプログラミング, オーム社 (2008).