

## CodeDrummer: プログラム実行における 関数呼び出しの可聴化手法

佐藤 和哉<sup>†1</sup> 平井 重行<sup>†2</sup>  
丸山 一貴<sup>†3</sup> 寺田 実<sup>†1</sup>

プログラム動作を理解させるための、実行時における関数呼び出しの構造を音楽的な表現手法を導入して可聴化するシステムを提案する。ここでは、音楽のリズム表現を導入することで、可聴化した結果がエンタテインメントとして楽しみながら、かつプログラム動作の理解を支援できるようにすることを目指している。今回は、関数呼び出し構造に対し、異なる観点でリズムパターンの適用ができるようにし、その効果や利点を確認できるようにした。

### CodeDrummer: Sonification Methods of Function Calls in Program Execution

KAZUYA SATO,<sup>†1</sup> SHIGEYUKI HIRAI,<sup>†2</sup>  
KAZUTAKA MARUYAMA<sup>†3</sup> and MINORU TERADA<sup>†1</sup>

We propose a program sonification system focused on the nesting structure of function calls at the runtime. In this paper, we apply several sonification methods to various programs, and compare their results. We use musical expression, especially rhythm patterns for the sonification. This enables the user to enjoy listening to the sound as an entertainment in addition to understanding the processing flow of the program.

---

<sup>†1</sup> 電気通信大学大学院 情報通信工学専攻  
Graduate School of Information and Communication Engineering, The University of Electro-Communications

<sup>†2</sup> 京都産業大学 コンピュータ理工学部  
Faculty of Computer Science and Engineering, Kyoto Sangyo University

<sup>†3</sup> 東京大学 情報基盤センター

### 1. はじめに

Processing<sup>\*1</sup> や ActionScript<sup>\*2</sup>などによって、本来はプログラマではない人がプログラミングを行う場面が増えている。ただ、プログラミング初学者にとっては、デバッガや開発環境の扱いに不慣れであったり、コンピュータそのものに対する知識が乏しいなど、種々の理由によりプログラムの内部動作を把握したり確認することは難しい。アルゴリズムやその動作の理解に対しては苦痛を強いられる場面も多いと言える。このような初学者の課題に対し、プログラム動作の可視化や可聴化といった手法でプログラミングの学習やデバッグを支援する研究がある。

本研究では、特にプログラムの実行時における関数やメソッド呼び出しの構造の可聴化に注目し、様々なプログラムに対して異なる可聴化手法を適用して、その効果を確認するためのシステム開発を行っている。ここでは音楽的な表現手法を導入した可聴化を主に考えており、本稿のシステムでは特にリズム演奏のパターンに着目した手法について扱うものとなっている。音楽のリズム表現を導入することで、可聴化した結果をエンタテインメント的に楽しみながらプログラム動作の理解を支援することが本研究の目的である。

本稿では、リズム表現を割り当てる機能とそのインターフェースについての検討と考察を行ったので、その点を重視して述べる。

### 2. CodeDrummer の概要

コンピュータプログラムは、関数呼び出しの順序や親子関係などによって、様々な動きが起こりうる。関数呼び出しを可聴化することは、プログラムの規模に依らず、処理順序や呼び出しの深さなどに対して直感的な理解を支援する可能性がある。

著者らは、プログラミング初学者がプログラム実行を理解する際に重要な要素は、構文処理の順序より、ソースコード上における実行点の移動具合や変数の値の遷移情報であると考えており、可聴化においてもそれらの情報を重視したデザインを検討している。

著者らが以前に開発した CodeMusician<sup>1)</sup> では、1行毎の実行をリズム楽器の発音に割り当て、関数呼び出しの際に関数名を読み上げる形で可聴化を行っていた。また、変数値の大小を音階に割り当てることで、特定の変数値の変化の様子を音の流れとして知覚することが

---

Information Technology Center, The University of Tokyo

\*1 <http://processing.org/>

\*2 <http://www.adobe.com/devnet/actionsript.html>

可能となっている。ただ、関数名の読み上げについて、1ステップずつプログラムを実行する場合には判りやすさの観点で良いが、1ステップずつ可聴化せずとも良い場合や、テンポを上げて実行したい場合などでは、たくさんの読み上げが発生するために実行の様子を把握することが難しくなるなど課題があった。

本稿で新たに提案する可聴化システム CodeDrummer では、基本的には CodeMusician と同様に関数呼び出し構造に注目した可聴化を行うものである。ただ、単に1行毎実行の発音を行うのではなく、各関数呼び出しや関数の呼び出し深さ、関数呼び出しの木構造のパスをリズムパターンとして設定できるなど、呼び出し構造に依存した可聴化のデザインができる機能の設計を行っている。ここで、リズムパターンとして可聴化を行うのは、関数呼び出しの重なり合う様子がリズムパターンによって設定・提示することで、身近なドラム演奏やリズムを連想させ、聴き取りやすくなるのではないかと考えたことによる。図1にその画面構成を示す。

### 3. CodeDrummer の機能

#### 3.1 ユーザインタフェース

ユーザインタフェースは以下が用意されている。

- 関数呼び出し構造の表示  
現在実行中の関数呼び出し構造を階層的に表示する。
- シーケンス編集インタフェース  
3.3.1 章にて後述する。
- 楽器割り当てインタフェース  
各関数に割り当てる楽器を選択する。
- コントロールインタフェース  
可聴化を行うプログラムの選択、再生、停止等を実行する。

#### 3.2 可聴化手法

本システムでは、以下の3種類の関数呼び出し単位を用いた可聴化を試すことができる。

- 手法1** 各関数呼び出しに数拍のステップシーケンスを割り当てる
- 手法2** 各関数から深さ1の距離にある関数呼び出し集合を用いる
- 手法3** ルートにあたる関数から各関数に至るまでの関数呼び出し列を用いる

#### 3.3 手法1

プログラム中で処理される関数毎にドラムセットに属する楽器と数拍のリズムパターンを



図1 提案システムの画面構成  
Fig.1 Overview of proposal system

割り当て、該当する関数が呼び出されている間、その関数に対応する楽器音を発する。ここで呼び出しの深さが2以上となり、複数の関数呼び出しが重なる場合は、それぞれの楽器音を同時に発する。

図2のような関数呼び出し構造をもつプログラムを例とすると、図3に示すようなリズムパターンが生成できる(関数名が記された四角形の色と、リズムパターンで記された円形の色が対応している)。

#### 3.3.1 関数ごとのシーケンス編集

図3.3.1に示す画面上で、関数ごとの楽器とその楽器音が発するタイミングを選択できる。初期状態では8拍のシーケンスが用意され、その8拍の中から鳴らしたい拍を選択する。

#### 3.4 手法2

各関数から深さ1の距離にある関数呼び出し集合を用いる可聴化手法である。以降、この関数呼び出し集合を子関数リストと定義する。また、深さが2以上の関数呼び出しについ

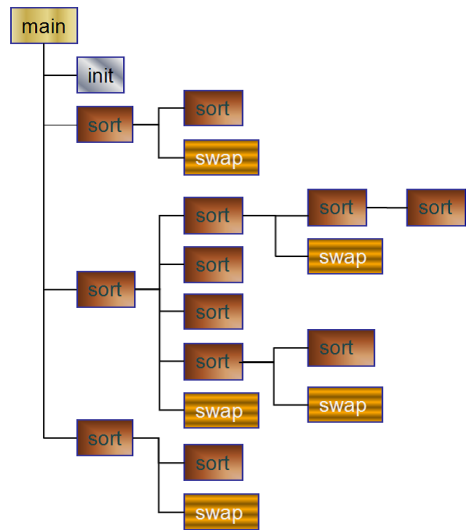


図 2 例とするプログラムの関数呼び出し構造  
Fig. 2 Function call tree of the example program

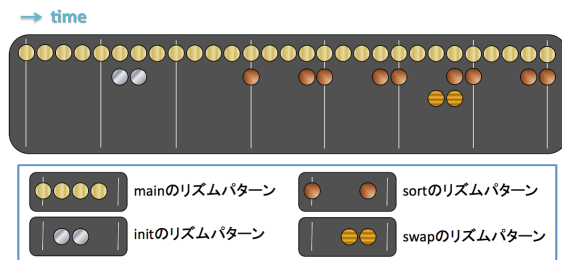


図 3 手法 1 で生成したリズムパターン  
Fig. 3 Rhythm pattern generated from method 1

では、その先祖にあたる関数から生成される子関数リストによる音を重ね合わせることにした。図 5 のような関数呼び出し構造をもつプログラムを例とすると、図 6 に示すようなリズムパターンが生成できる。

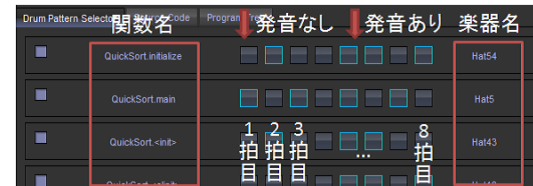


図 4 シーケンス編集インターフェース (QuickSort.main 関数では 1,3,5,7 拍目に鳴らす)  
Fig. 4 Sequence editor (Method of QuickSort.main has 1st, 3rd, 5th and 7th beats)

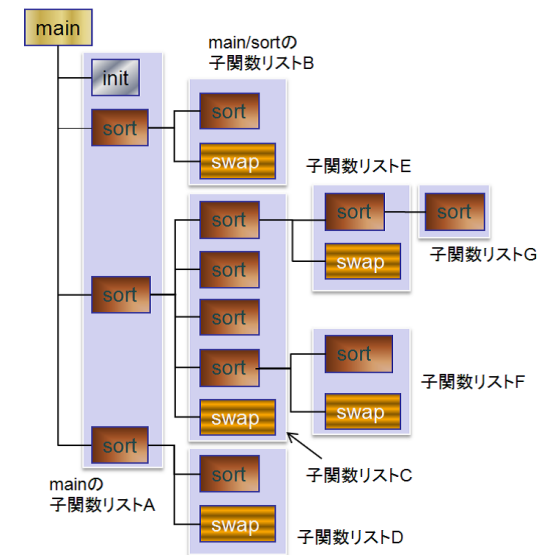


図 5 例とするプログラムの関数呼び出し構造  
Fig. 5 Function call tree of the example program

### 3.5 手法 3

ルートにあたる関数から各関数に至るまでの関数呼び出し列を用いる可聴化手法である。ここでも図 7 のような関数呼び出し構造をもつプログラムを例とすると、図 8 に示すようなリズムパターンが生成できる (関数名が記された四角形の中に引かれた破線の色と、リズムパターンで記された円形の中に引かれた破線の色が対応している)。

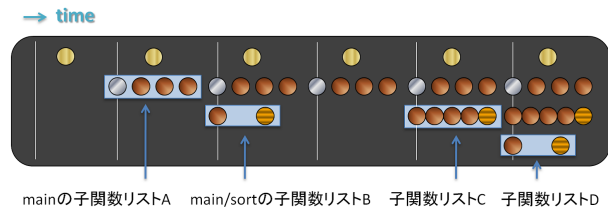


図 6 手法 2 で生成したリズムパターン  
 Fig. 6 Rhythm pattern generated from method 2

## 4. 実 装

### 4.1 インタフェース

音を出力する機構、及び GUI は、Adobe Systems 社の Flex<sup>\*1</sup>を用いて作成した。ドラム音源は BEST SERVICE 社の PS15 DANCE DRUM<sup>\*2</sup>を用いた。このサンプリング CD には、バスドラムやスネアドラムといった 1300 種類以上の単発ドラム音が収録されている。この音源を Flex で読み込み可能な MP3 ファイルに変換し、CSS ファイルに埋め込んで使用した。

### 4.2 コラボレーション機能

本システムは、Web ブラウザ上で C や Java で記述されたソースコードをアップロードし、その場ですぐに可聴化できる機能も含まれており、他のユーザが可聴化したプログラムの試聴、音の編集・保存が可能である。

#### 4.2.1 ソースコード投稿機能

PHP のファイルアップロード機能を利用し、ユーザの自作プログラムをサーバに保存することを可能とした。アップロードが完了し次第、コンパイルを行い、最後に可聴化する上で必要なプログラム実行ログを取得する。これには寺田が開発した実行トレースシステム<sup>2)</sup>を用いた。取得した実行ログからプログラム上で発生した関数呼び出しを解析し、音を生成する。

#### 4.2.2 編集情報の保存

データベース管理システムとして MySQL を使い、各ユーザが編集したリズムパターン

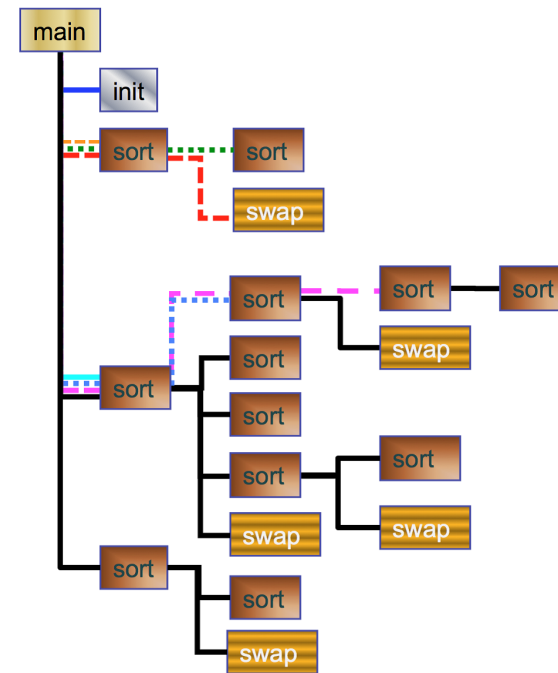


図 7 例とするプログラムの関数呼び出し構造  
 Fig. 7 Function call tree of the example program



図 8 手法 3 で生成したリズムパターン  
 Fig. 8 Rhythm pattern generated from method 3

や割り当てた楽器の種類などの情報をサーバに蓄積し、後から再編集したり、他のユーザにも再生・編集することを可能にした。

\*1 <http://www.adobe.com/jp/products/flex>  
 \*2 <http://www.crypton.co.jp/mp/do/prod?id=19750>

## 5. 考察

著者らが本システムを試用し、主観評価に基づく考察を行った。

### 5.1 評価対象プログラム

評価対象とするプログラムは、プログラムの規模が大きく異なる以下の3つのプログラムを選択した。

**例 1** HelloWorld プログラムをコンパイルするときの gcc の内部動作 (関数呼び出し数:456)

**例 2** Java で記述したクイックソートプログラム (関数呼び出し数:30)

**例 3** C で記述したヒープソートプログラム (関数呼び出し数:14)

### 5.2 各手法に関する考察

手法 1 については、すべての関数が等しい拍数をもったリズムパターンを用いるため、例に示したどのプログラムについてもドラム音として聴くには満足のいくものとなった。しかし、ユーザ自身でリズムパターンを設定する必要があるため、適切に設定しないと、プログラムごとの音の違いが分かりにくいということがあった。また、呼び出しの深さが深くなると、各関数の音が重なり合い個々の関数に対応する音が聴きづらくなることがあった。

手法 2 については、音の種類が多様で、子関数同士の音の重なりあうタイミングが頻繁に出現するため、飽きが来ず、特に関数呼び出し数の多い例 1 のプログラムは興味深い音となった。さらに、処理が混み合っているタイミングとそうでないタイミングを容易に判別することができた。

手法 3 については、関数呼び出しの親子関係に関する判別は容易になったが、似たようなシーケンスが連続することが多く、拍数も音節ごとに変更が生じることが多いため、ドラム音として聴くには少々聴きづらいものとなった。また、現在の手法では複数の関数呼び出し列の音が重なることがないため、全体的に落ち着いた感じの音となった。例 2 のプログラムの場合には再帰的な関数呼び出しが連続するため、呼び出しの深さが深くなっていけばいくほどテンポが速くなっていく効果が得られた。

## 6. 関連研究

聴覚ディスプレイ (Auditory Display) や可聴化 (Sonification/Auralization) の研究は、コンピュータの歴史からすると比較的浅く 1990 年頃から始められており、聴覚ディスプレイの国際会議 ICAD は 1992 年から開催されている。プログラム可聴化に関する研究もその頃から行われてはいるものの<sup>3)</sup>、プログラム実行時のコンピュータの動作を AM ラジオの

混信する音で聞くということは 1950~1960 年代に行われており<sup>4)</sup>、音でプログラムの動作の情報を知る行為は昔から存在していた。プログラム可聴化の研究自体は、デバッグを目的とするものやプログラム動作の理解を支援するもの、プログラムの構成要素の理解を支援するものなど様々である。またその表現手法も音の高さや変調を利用するもの、効果音を利用するもの、メロディモチーフなどの一般的な音楽表現を行うものなど、様々な適用方法が試みられている。これらの研究されてきたプログラム可聴化システムは、Vickers らが比較表としてまとめている<sup>7)</sup>。以下にプログラム可聴化システムの研究例を挙げる。これらのうち CAITLIN だけがプログラム初学者向けであり、その点では本研究と似ているが、他は初学者対象ではない。また、本研究が関数呼び出しの階層構造に着目してリズム表現による可聴化を試みている点などで他の研究とは違うアプローチを取っていると言える。

### 6.1 CAITLIN

Vickers らは、Pascal の教育において、音楽を用いたデバッグ支援手法を提案し、可聴化システムの CAITLIN<sup>5)</sup>を開発した。“Selection(条件分岐文)”, “Iteration(繰り返し文)”といったプログラミング構成子の開始時点、終了時点や条件判定時点に対し、それぞれ決められたメロディ (leitmotif) を設定することで音楽作品を生成し、デバッグに応用した。

### 6.2 大澤の研究

大澤は、プログラム構成要素である継承関係を静的に解析し、構成要素の判別に応用する例として、Java 言語の 1500 以上あるクラスの継承関係において、スーパークラスとサブクラスの継承関係の理解に音 (楽節) を用い、クラスの呼び出しの深さを単音の重ね合わせにより可聴化した<sup>6)</sup>。また、継承関係の判別について評価を行い、テキストによる提示よりも音による提示の方が判別時間が短いという結果が出ている。

### 6.3 InfoSound

Sonnenwald らは、プログラム実行時の様々なアプリケーションイベントに応じてメロディなどの音楽シーケンスを鳴らす InfoSound システムを開発した<sup>3)</sup>。ここでは、電話のネットワークサービスのシミュレータや、並列計算のシミュレータの動作を可聴化する例を示している。

### 6.4 Sonnet

Jameson は、様々なプログラミング言語に対応して、音高や変調でプログラム動作を可聴化する Sonnet を開発した<sup>8)9)</sup>。このシステムは、プログラム実行可聴化用のビジュアルプログラミング環境を含むもので、C 言語などのプログラムソースコードの特定場所のプ

レークポイント設定の代わりにトリガーを発生させ、ビジュアルプログラミング環境でそのトリガーをハンドリングしてサウンド処理を行う点で特徴がある。

### 6.5 ADSL

Dale は、C 言語のプログラムの動作を可聴化するための定義言語 ADSL<sup>10)</sup> を作成した。ここでは音高のほか cranking や stamping などの効果音を用いた可聴化が可能であるものの、定義言語であるためユーザがテキストで記述する必要がある。

### 6.6 その他

Christopher らによって、Logo のプログラム動作を音高や効果音で表現するシステム LogoMedia<sup>11)</sup> が、David らによって音高で様々な言語のプログラム可聴化を行う LISTEN<sup>12)</sup> などが開発されている。

## 7. おわりに

本稿では、プログラムの実行時における関数呼び出し構造に 3 種類の可聴化手法を適用し、各手法の効果や利点などを考察した。プログラム動作理解のための可聴化研究は幾つか存在しているが、リズム楽器のリズムパターンに着目した手法はこれまでになく、この点で本研究は特徴がある。ただ、ドラムやパーカッションなどが複数音鳴っていてその状態を聞き分けることがどの程度可能か、という認知的な課題も考えられる。音楽経験などに関連して聞き分けや可聴化の有効性について検証を行っていくことが今後の課題と言える。また、どのような関数呼び出し集合から音が生成されているのか視覚的にも理解しやすいユーザインタフェースを導入し、初学者の理解をより向上させることも必要と考えている。

一方で、本研究システムの可聴化機能の発展として、1つの楽器でも様々な演奏法を加味した設定を可能にすることが考えられる。現在は1つの関数に対し1つの音色しか設定できないが、ある関数に「スネアドラム」と設定した際に、スネアドラムに関する様々な音色を設定できるようにする。これにより、単に叩く音だけでもリムショットの有無や強弱を設定できたり、フラムやロールなどの演奏技法も設定できるようになり、より人間の演奏に近い表現ができるようになる。単なるリズムパターンという枠に囚われず、ドラムのフィルインのような設定も可能になり、関数の呼ばれ方によって演奏の盛り上がり表現できるよう可聴化ができるようになると言える。

これらの課題や方向性を含めて、今後もリズム楽器やそのリズム演奏パターンなどを用いたプログラム動作の可聴化を提案していきたい。

## 参考文献

- 1) 佐藤和哉, 丸山一貴, 寺田実: CodeMusician: プログラム実行可聴化の試み, 第3回エンターテインメントと認知科学シンポジウム, (2009).
- 2) Minoru Terada: ETV: a program trace player for students, in Proc. of the 10th annual SIGCSE conference on Innovation and technology in computer science education, pp.118-122 (2005).
- 3) Sonnenwald, D. H., Gopinath, B., Haberman, G. O., Keese, William M, I., and Myers, J. S.: InfoSound: An audio aid to program comprehension, in 23rd Hawaii International Conference on System Sciences, (1990).
- 4) Paul Vickers, James L. Alty: Siren songs and swan songs debugging with music, Communications of the ACM Vol.46, No.7, pp.86-92 (2003).
- 5) James. L. Alty and Paul Vickers, The CAITLIN Auralization System: Hierarchical Leitmotif Design as a Clue to Program Comprehension, in Proc. The Fourth International Conference on Auditory Display, pp.89-96 (1997)
- 6) 大澤範高: 継承関係の可聴化, 日本ソフトウェア科学会第8回インタラクティブシステムとソフトウェアに関するワークショップ, 近代科学社, (2000).
- 7) Paul Vickers: Program Auralization: Author's Comments on Vickers and Alty, ICAD 2000, ACM Transactions on Applied Perception, Vol.2 Issue 4, pp.490-494 (2005).
- 8) Jameson D. H.: Sonnet: Audio-Enhanced Monitoring and Debugging, Auditory Display, Vol.XVIII, pp.253-265 (1994).
- 9) Jameson D. H.: The Run-Time Components of Sonnet, Proc. of the Second International Conference on Auditory Display ICAD'94 (1994).
- 10) Dale S. Bock: ADSL: An Auditory Domain Specification Language for Program Auralization, Proc. Second International Conference on Auditory Display, pp.251-256 (1994).
- 11) Christopher J. DiGiano, et.al.: LogoMedia: a sound-enhanced programming environment for monitoring program behavior, in CHI'93 Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems, pp.301-302 (1993).
- 12) David B.Boardman, et.al.: LISTEN: A Tool to Investigate the Use of Sound for the Analysis of Program Behaviour, in Proc. 19th International Computer Software and Applications Conference, pp.184-189 (1995)