CodeDrummer: リズムに着目したプログラム可聴化システム

佐藤和哉[†] 平井重行[††]
丸山一貴[†††] 寺田実[†]

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，その効果を確認できるようにした．

CodeDrummer: A System of Program Sonification Focused on the Rhythm

Kazuya Sato,[†] Shigeyuki Hirai,[††] Kazutaka Maruyama[†††]
and Minoru Terada[†]

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 rhytm 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.

1. はじめに

Processing[☆] や ActionScript[☆☆] 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ギングを支援する研究がある[2)3)4)]．

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，リズムパターンによる可聴化のデザインを行うことができるシステム CodeDrummer のコンセプトと機能およびその実装について紹介する．また，関数の再帰呼び出しなど，関数呼び出しが重なりあいながら，様々な処理を行うプログラムでは，異なる可聴化手法を適用できることから，どういった効果が得られるか検証したので，その結果についても述べる．

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

† 電気通信大学大学院 情報通信工学専攻
The University of Electro-Communications
†† 京都産業大学 コンピュータ理工学部
Kyoto Sangyo University
††† 東京大学 情報基盤センター
Information Technology Center, The University of Tokyo

☆ http://processing.org/
☆☆ http://www.adobe.com/devnet/actionscript.html

的な理解を支援する可能性がある．

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

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[1] 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，停止等を操作する．

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 4 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 拍のシーケンサが用意され，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ては，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

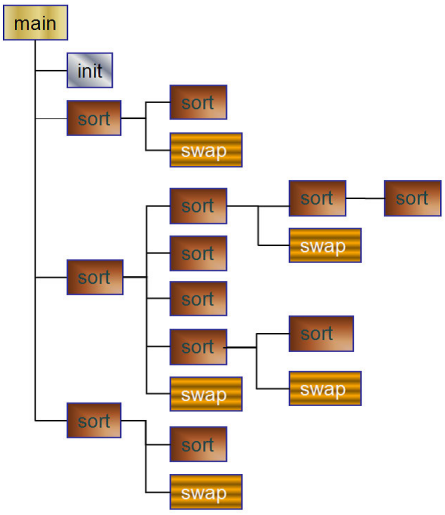

図 2 例とするプログラムの関数呼び出し構造
Fig. 2 Function call tree of the example program

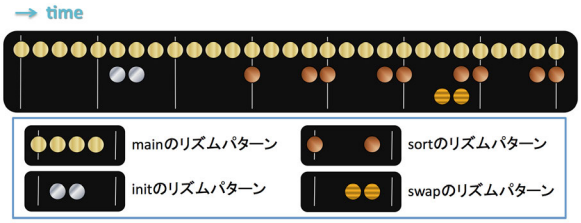

図 3 手法 1 で生成したリズムパターン
Fig. 3 Rhythm pattern generated from method 1

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)

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．

3.5 手 法 3

ルートにあたる関数からリーフにあたる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．

4. 実 装

4.1 インタフェース

音を出力する機構及び，グラフィカルな操作インタフェースは，Adobe Systems 社の Flex☆ を用いて作成した．ドラム音源は BEST SERVICE 社の PS15 DANCE DRUM☆☆ を用いた．

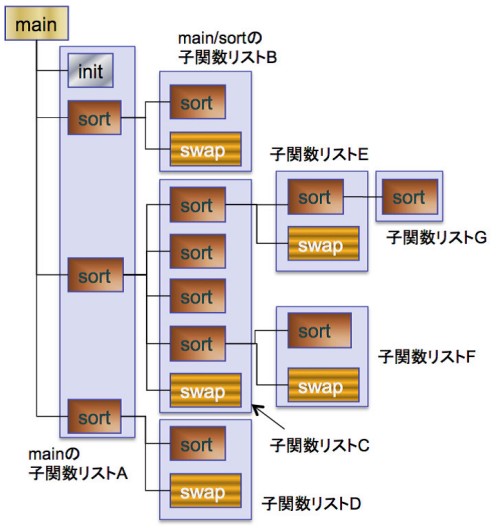

図 5 例とするプログラムの関数呼び出し構造
Fig. 5 Function call tree of the example program

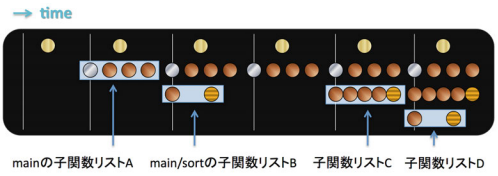

図 6 手法 2 で生成したリズムパターン
Fig. 6 Rhythm pattern generated from method 2

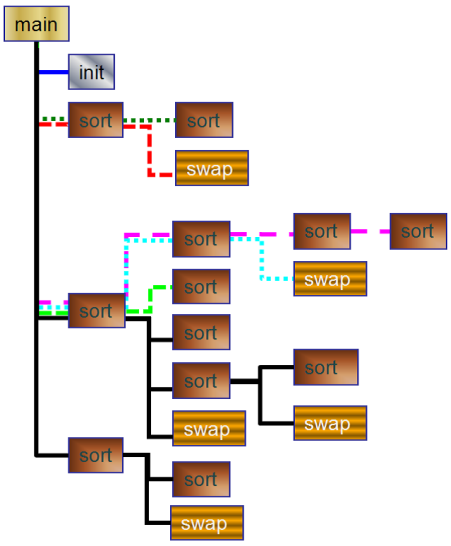

図 7 例とするプログラムの関数呼び出し構造
Fig. 7 Function call tree of the example program

☆ http://www.adobe.com/jp/products/flex
☆☆ http://www.crypton.co.jp/mp/do/prod?id=19750

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

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，音の編集が可能である．

5. 考　　察

著者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 2 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）

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，どちら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．

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．

6. 関 連 研 究

プログラム可聴化に関する研究は古くから行われており[3]，プログラム実行と並行して音を提示する動的なシステム[2] や，プログラムを構成する要素を静的に解析し，要素の判別に応用するシステム[4] など様々である．

6.1 Vickers らの研究[2]

Vickers らは，プログラム言語 Pascal の教育において，音楽を用いたデバッギング支援手法を提案した．条件分岐文，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おいて，それぞれ決められたメロディを並べ，音楽作品を生成し，デバッギングに応用した．

6.2 大澤の研究[4]

大澤は，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．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．

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様々な可聴化手法を適用し，初学者がそれをエンターテイメント的なものとして楽しみながら学習できるシステムを開発した．今後の課題として，複数のユーザに本システムを利用してもらい，使用した感想や意見をいただきたい．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することで，初学者の理解がより向上することが考えられる．

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 回エンターテイメントと認知科学シンポジウム, (2009).
2) Paul Vickers , James L. Alty: Siren songs and swan songs debugging with music, Communications of the ACM, (2003).
3) Sonnenwald, D. H., et al. : Infosound: An audio aid to program comprehension. In 23rd Hawaii International Conference on System Sciences, (1990).
4) 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).