教育用作曲プログラミング言語に関する考察

中山 心太
EMail: shinta.nakayama@gmail.com

概要 音楽とプログラミングには、いくつかの共通点がある。たとえば、音楽には、同じ旋律を繰り返したり、同じ旋律を一曲の中で数度使われていたりする。そのため、それらは構造化プログラミングにおける反復や、抽象化に対応して考えることができる。加えて、ゲーム産業においてはプロシージャル技術を利用した、動的な音楽生成が行われている。そこで本稿では、教育的な側面と、プロシージャルな音楽生成の両面から、作曲用の言語について検討を行う。

Study of educational music programming language.

Shinta Nakayama
EMail: shinta.nakayama@gmail.com

1.はじめに

音楽とプログラミングには、いくつかの共通点がある。たとえば、音楽には、同じ旋律を繰り返したり、同じ旋律を一曲の中で数度使われていたりする。そのため、それらは構造化プログラミングにおける反復や、抽象化に対応して考えることができる。

加えて、ゲーム産業においてはプロシージャル技術を利用した、動的な音楽生成が行われている。そこで本稿では、教育的な側面と、プロシージャルな音楽生成の両面から、作曲用の言語について検討を行う。

1.1 音楽とプログラミングの共通点

1.1.1 時系列の概念の獲得

日本の高校教育までに行われる数学では、時系列を含んだ概念を取り扱わないでいる。そのため、大学ではじめてプログラミングを学ぶ学生において、すべての文が同時に実行されると思い込んでいたり、状態変化を伴う代入という概念を理解するのに時間がかかるといった事例が見受けられる。

既存のプログラミング言語の問題点として、どの行が実行されているのかわからない、

コンピュータの実行速度が遅かった時代は、逐次実行を肌で感じることができたが、十分に速くなってしまった現代では、デバッガを繋いでステップ実行をしなくては、逐次実行を見ることはできない時代になってきてる。

音楽はそれそのものが時系列を持っており、五線譜等で表現されているため、逐次実行であることが、そのまま伝わると考える。そのため、音楽を経由してプログラミングを教えることで、時系列の概念の獲得が容易になるのではないかと考える。

1.1.2 構造化プログラミングと音楽の類似性

音楽は同じ旋律を繰り返し、再利用するため、構造化プログラミングにおける反復や抽象化との相性が良い。またドラムラインをループで鳴らしたり、同じフレーズを3度鳴らしたら別のフレーズというようなケースでは、ループと分岐が活用される。

そのため、構造化プログラミングの技法を使うことで、より少ない工数で作曲が行えるのではないかと考える。

1.2 音楽のもつ教育的側面

1.2.1 科学的側面

音は空気振動の周波数からなり、和音や不協和音は複数の音の整数比から生み出すことができる。そのため、比や音、振動、周波数といった、基礎的な科学知識の勉強に活用することができる。

1.2.2 楽しさ

現代は様々なエンターテイメントにあふれてるため、コンソールで文字が出るだけでは、面白いと感じる人は少ない。そのため、教育用のプログラミング言語においてはViscuitやScratch、プログラミ

ンといったビジュアルプログラミング言語が多い。

音楽によるプログラミングは、それらに準ずる楽しさがあると考える。

1.2.3 簡易さ

ビジュアルプログラミング言語では、オブジェクトの位置の制御や、あたり判定など、オブジェクトを起点とした考えが必要になってきてしまう。そのため、プログラミング言語だけでなく、オブジェクトの考えが必要になるため、学習のハードルが高くなることが考えられる。

音楽によるプログラミングは、そのようなオブジェクトは登場しにくいため、それらに比べてハードルは一歩下がると考えられる。

1.2.3 恥ずかしくなさ

プログラミングのハードルの一つに、恥ずかしさがある。プログラミングにはバグがつきものであり、正しく動かないことは日常茶飯事であるが、正しく動かない状態であることが恥ずかしいために相談できない、というケースがある。

音楽をプログラミングする場合、バグがあったら変な音が鳴るだけであり、バグがあってもある種楽しいため、恥ずかしさという点は解決されるのではないかと考える。

2.既存事例

2.1 モーツアルト「音楽のサイコロ遊び」

二つのサイコロを振って、出た目の合計にしたがって、2から15の番号が振られた小節を選択し、16小節の楽曲を作りあげ、即興で演奏するという、ピアニストのための遊びである。

小節が音楽として完成しているため、適当につなげても音楽として成立するという特徴がある。

2.2 ChucK

Chuck[1]はGe Wangらが開発した教育用音楽言語である。

例えば次のようなコードでは、サイン波のオシレータオブジェクトを生成し、オシレータの周波数に400Hzをセットし、オシレータをDAコンバータに接続することで、400Hzのサイン波を1秒間鳴らすことができる。

```
SinOsc osc;
400 => osc.freq;
osc => dac;
1::second => now;
```

また、制御構文を用いることができ、次のコードでは、100Hzから1000Hzのサイン波をランダムに0.1秒ごとに鳴らすことができる。

```
SinOsc osc;
osc => dac;
while (true){
  Std.rand2(100,1000) => osc.freq;
  0.1::second => now;
}
```

しかし、ChucKは、オシレータやフィルターを繋ぎ合わせるシンセサイザープログラミング言語の性質が強く、音を鳴らすというよりも、音を鳴らす回路をプログラミングで動的に変更するという性質が強い。

そのため、作曲をするには、シンセサイザーを組み替えながら、音楽を作らねばならず、直観的に記述することが難しい。

2.3 SonicPi

SonicPi[2]はRubyで作られた教育用音楽言語であり、Rubyの文法がそのまま使え、教育用コンピュータであるRaspberry Piの標準OSであるRaspbianにも標準でインストールされている。内部的には音色生成ライブラリであるSuperColliderのラッピングになっている。

次のコードは、C,D,Eのいずれかの音をランダムに選択し、0.1秒ごとに鳴らすものである。

```
loop do
  code = [:C, :D, :E].choose
  play code, release: 0.1
  sleep 0.1
end
```

SonicPiはRubyの構文をそのまま使えるため、ループだけでなく、関数定義や、スレッドによる複数チャネルによる演奏等を行うことができる。また、Rubyをベースとしているため、SonicPiで学習した人は、言語の乗り換えなしに、より難しいことにチャレンジできるという特徴がある。

1. http://chuck.cs.princeton.edu/

2. http://sonic-pi.net/

3 プロシージャル技術を利用したゲームミュージック

ゲーム機の高性能化、大容量化に伴い、作らなければならないコンテンツの質と量が増大した。そのため、コンテンツ制作の工数を削減するためにプロシージャル技術が用いられるようになってきた。

プロシージャルとは、アルゴリズムに基づいて、形状や模様を生成する技術であり、古くはローグのダンジョン生成などが挙げられる。最近では、植物の生成や、地形や都市の生成、物体の破壊や爆発などでプロシージャル技術が用いられている。

プロシージャル技術は、コンテンツを事前に人手で生成する代わりに、アルゴリズムとコンピューティングリソースにより動的に生成するものである。

ゲームは長時間遊ぶするものであり、同じ音楽を何度も聞き続けると飽きてしまうという問題がある、そのため、ゲームのシーンに合わせた音楽をプロシージャル技術を用いて動的に合成するということが行われている。

3.1 ゼルダの伝説 時のオカリナ

任天堂の「ゼルダの伝説 時のオカリナ」では、プレーヤーはハイラル平原という広大なフィールドを何度も歩かなければならない。そのため、プレーやが飽きないように、ハイラル平原のBGMには、プロシージャル技術が用いられている。当時の任天堂の岩田社長が開発スタッフに対して質問する記事[3]では、次のように述べられている。

（横田）
で、『時のオカリナ』の話なんですけど、メインの舞台であるハイラル平原で、ダンジョンの冒険から、そこに戻ってくると、聴こえてくる音楽が毎回違うんです。
曲の流れ自体は、そんなに変わらないんです。
でも、流れてくるメロディが、いつも同じタイミングでこないんです。
しかも、同じ曲であっても、敵と戦ったりすると、少しスリリングな曲調に変化して、戦闘が終わると、また雄大な曲に戻ったり、リンクが立ち止まっていると、静かな曲になったりと、絶え間なく曲が変わっていくんです。
（岩田）
あそこでは決まった音楽をずっと流しているんじゃないんですよね。
（横田）
そうなんです。ハイラル平原では「普通」と「戦闘」、そして「静かな」という、3パターンの曲が切り替わるんです。
（岩田）
サウンドに割り当てることができるメモリの制限が厳しかったあの時代には、一般的に事前につくった音楽をストリーミングという技法で流すことが多かったのですが、『時のオカリナ』で絶え間なく曲が変わっていくようなことができたのは、NINTENDO64でロムカセットの特長を活かして音楽を状況に合わせて合成していたからだったんです。

以上をまとめると図 1のようなオートマトンになる。ゲーム中の状態に応じて、「普通」「静かな」「戦闘」という三つの状態を遷移し、それぞれの状態の中で、モーツアルトのサイコロ遊びのように、小節単位で音楽を組み立て、小さなオートマトンにより音楽が生成されている。

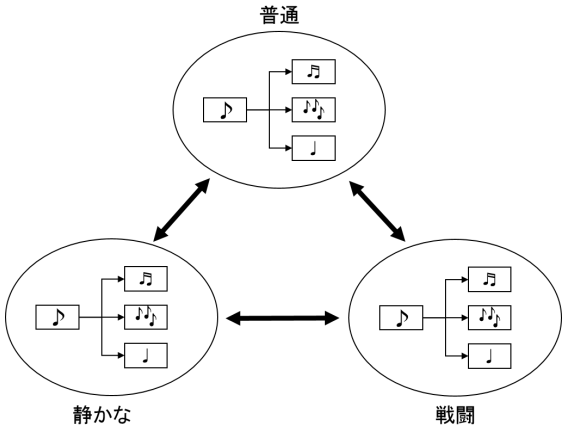

図 1 ゲームのステートと演奏の関係

また、あるステートから別のステートにいきなり遷移すると、音楽が急に代わるため、滑らかに音楽

3. 社長が訊く『ゼルダの伝説 時のオカリナ 3D』 https://www.nintendo.co.jp/3ds/interview/aqej/vol1/index.html

が切り替わるように小節の切れ目で別のステートに遷移ししていると予想される。

3.2 アルトネリコ３

RPGにおいて戦闘は何度も発生するため、同様の曲を何度も聞くことになる、そのため、アルトネリコ３では戦闘中のミュージックの動的生成[4]を行っている。

前述のゼルダの伝説 時のオカリナでは、プレーヤーの状態を大域的にしか取得していなかったが、アルトネリコ３では、キャラクター、装備品、ステータス、場の状況等から、ボーカル、ドラム、コーラス等のトラックをそれぞれ個別に変更することで、状況に合わせた動的なゲームミュージックの生成を行っている。

3.3 ファンタシースターオンライン２(PSO2)

オンラインゲームは長時間プレイし、かつ状況がシームレスに移行する。そのため、プロシージャル技術を用いて、状況に合わせた飽きのこない音楽の生成[5]が行われている。

PSO2では敵の強さや、自分のキャラのHP、クエストの進行度合いから、ゲームの盛り上がりを算出し、盛り上がりのパラメータをBGMの生成ロジックに与えることで、ゲームの状況にあった音楽を再生する。

音楽の最小単位がクリップであり、これが数個集まることでフレーズを形成し、フレーズが数個集まることで、楽章、楽章が数個あつまることで楽章を形成する。そして、それぞれのレベルにおいて、状態遷移のパターンが存在している。

そのためPSO2のプロシージャルゲームミュージックの生成エンジンは、外部入力を受け付ける多層化された確率付き有限状態オートマトンであると言える。

4 作曲プログラミング言語の検討

4.1 最小単位で音楽になる仕組み

モーツアルトの音楽サイコロ遊びや、ゲームでのプロシージャル技術の応用から、ある程度完成しているパーツを組み合わせることで、破綻しない音楽作ることができることがうかがえる。

そのため、教育用に重点を置く場合、楽しさや面白さを担保するためには、小節程度のパーツをプリセットで用意するか、機械学習等で生成することが有効であると考える。

4.2 音楽用の階層型オートマトンの記述

PSO2の事例のように、音楽の自動生成は、実質的には階層型のオートマトンによって記述することができる。したがって、音楽に適した形で状態遷移を記述できるような言語体系が求められる

4.3 暗黙的無限ループ

音楽はリズムに従い音が鳴るものであるため、音楽を鳴らし続けるには、音符に相当するコードを相当量並べるか、これを無限ループ等を利用する必要がある。

初学者にとっては、音を鳴らしたい、というのがゴールであるため、無限ループを理解しないと音楽が鳴らないというのはハードルが高い。Sonic Piは無限ループを最初から要求しており、若干ハードルが高いと考えられる。

例えば、教育用のプログラミング言語であるHMMMML[6]では、インタラクティブな処理を少ない行数で実現するために、暗黙的な無限ループの仕組みが実装されている。

そのため、オプションでもよいので、音楽用のプログラミング言語には暗黙的な無限ループというものが必要であると考える。

5 作曲プログラミン言語の試作

教育用の言語ということで、様々な環境で動くことが求められるため、ポータビリティを考慮して

4. アダプティブミュージックを駆使したゲームならではのBGMの提案 ～ADAMS＆R.A.H.システムによる開発事例～ http://www.cri-mw.co.jp/event/2009/2nt4hm000000qlss-att/2nt4hm000000rugh.pdf
5. ［CEDEC 2012］PSO2の「途切れないBGM」はこうやってできている。セガが語る「BGMのプロシージャル生成」 http://www.4gamer.net/games/120/G012075/20120820076/
6. プログラミングに対するモチベーションを向上させる新言語HMMMMLの開発 http://miyashita.com/?p=971

```
{
    "bpm":300,
    "beats":[
        {
            "source_node": { "type": "noise", "continuity_count": 20, "length":0.10},
            "node_list":[
                {"type":"gain", "gain": 0.8}
            ]
        },
        {
            "source_node": { "type": "noise", "continuity_count": 10, "length":0.02},
            "node_list":[
                {"type":"gain", "gain": 0.3}
            ]
        },
        {
            "source_node": { "type": "noise", "continuity_count": 5, "length":0.02},
            "node_list":[
                {"type":"gain", "gain": 0.3}
            ]
        },
        {
            "source_node": { "type": "noise", "continuity_count": 5, "length":0.02},
            "node_list":[
                {"type":"gain", "gain": 0.3}
            ]
        }
    ]
}
```

HTML5 の Web Audio API[7]を利用した。そのため、HTML5 に対応した近代的なブラウザであれば、どこでも音楽を再生することができる

Web Audio API のアーキテクチャは、シンセサイザーの配線を制御する形式であるため、実質的には ChucK とほぼ変わらない。そのため、楽譜を入力されたら、シーケンサーを動的につなぎ変えながら音を鳴らし、楽譜道理に音を鳴らす仕組みというものを作らなければならない。

試作として、「最小単位で音楽になる仕組み」「暗黙的無限ループ」の二つの検証を行うために、一定のリズムで、入力された音を鳴らすドラムマシンを作った[8]。このドラムマシンは JSON を入力として受け取り、この JSON を解釈することで、ドラムを鳴らすことができる。音色は source_node で基本的な音源が決定され、node_list で様々なエフェクトをかけることができる。上記の例では、音源にホワイトノイズを利用し、ホワイトノイズをどれくらい引き延ばすかによって音の高低を決め、ゲインフィルターによって音量を調整している。

7. https://developer.mozilla.org/ja/docs/Web/API/Web_Audio_API
8. https://tokoroten.github.io/web_drums_machine/

6 考察

ドラムマシンを作成し、「最小単位で音楽になる仕組み」「暗黙的無限ループ」の二つの検証を行った。暗黙的な無限ループは、試行錯誤を容易にし、最小限で音楽になる単位を簡単に作ることができた。

そのため、このドラムマシンで生成したリズムを最小単位として、上位のオートマトンが選択し、実行するといったことが考えられる。

今後の課題としては「音楽用の階層型オートマトンの記述」の検証が挙げられる。今回は音楽用のオートマトンの検討まで行うことができなかった。今後は、音楽用階層型オートマトンの検討、実装を行っていく予定である。