平成 28 年度　卒業論文

作曲技術を持たない人のためのゲーム音楽自動作曲アプリケーションの開発

函館工業高等専門学校 情報工学科

5 年東海林研究室 古川 浩気

目次

1 章　序論

第 1 節 英文アブストラクト

In this research, we tried to develop an application that automatically generates game music using our own composing algorithm for those who do not have composition skills. This application automatically generates the game music that the user wants by inputting the parameters related to the song. As the parameters of the input, the length (seconds) of the song, the tempo (BPM).,and the atmosphere(tone) were used. Furthermore, we examined the difference of the song atmosphere when changing the parameters. As a result, we found that this application can create game music that the users want.

Key words : game music, mididata library, composition algorithm

第 2 節 研究目的

本研究の目的は、ユーザが望むゲーム音楽を自動で生成するアプリケーションを開発することである。

第 3 節 研究背景

私が所属しているゲームプログラミング研究会[1]において、音楽を担当する人がいなくなってしまった。しかし作曲技術というのは一朝一夕で身につくものではないため、作曲技術がない人でも手軽に音楽を作成できるアプリケーションを開発する必要があると考えた。

自動作曲アプリケーションは多くあるがゲーム音楽の自動作曲に特化しているものはない。そこで本アプリケーションではゲーム音楽のための独自のアルゴリズムを考案して使用する。

第 4 節 開発環境

開発環境は以下のとおりである。

使用 PC	Windows10 home 64bit version
	Intel Core 2Duo CPU E7200 2.53GHz
	RAM 4.0GB
使用ソフト	Visual Studio 2015
	Visual Studio 2017
	Domino[2]
使用ライブラリ	MIDIData ライブラリ[3]
使用言語	C++

2 章　ゲーム音楽自動作曲アプリケーションの開発

第 1 節 システムの概要

はじめに、ユーザは生成する曲のパラメータとして曲の長さ(秒)、テンポ(BPM)、雰囲気(調)[4]を入力する。

その後本アプリケーションはこの後説明するゲーム音楽作曲アルゴリズムを使用して MIDI 形式データを生成する。

なお本アプリケーションはメイン構築部、サブ構築部、コード構築部、ドラム構築部から構成される。

実際の内部処理なども含め簡易的に表すと、図1のような構成になっている。

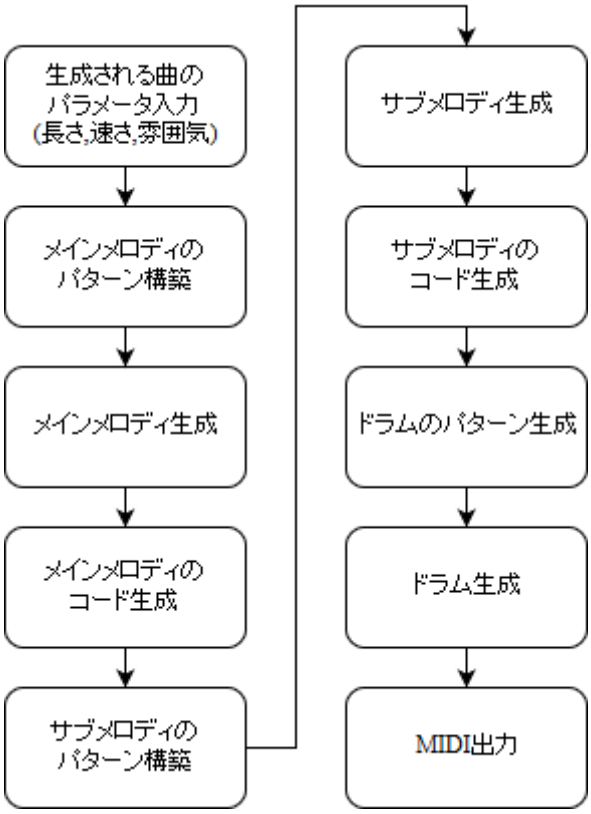

図 1　システムの処理の流れ

第 2 節 入力パラメータについて

今回入力するパラメータは曲の長さ(秒)、テンポ(BPM)、雰囲気(調)の 3 つである。本研究では小節数は長さと時間に比例させることにし、小節数の計算方法を下記の通りとした。

$$\text{小節数 n} = \frac{\text{BPM} \times \text{秒}}{240}$$

例として、BPM120、時間 60 秒が入力された場合

$$\text{小節数 n} = \frac{120 \times 60}{240} = 30$$

なので、したがって小節数は 30 となる。これは後述の作曲アルゴリズムに使用される。

また、本アプリケーションではランダムに音符を配置して音楽を生成しているが、これらは全て入力された調に修正される。例として、入力されたパラメータがハ長調である場合の修正前後の図を図 2 と 3 に示す。

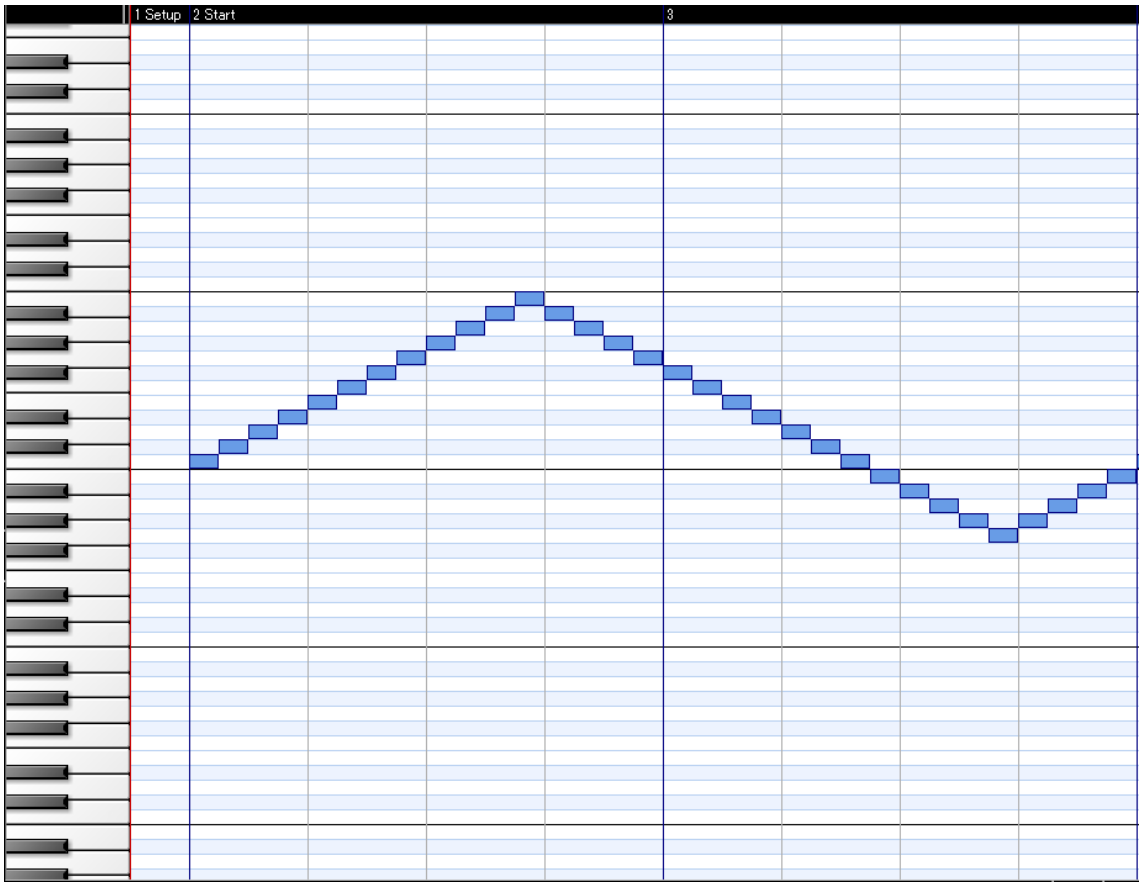

図 2　調修正前の図(ランダムに音符を配置)

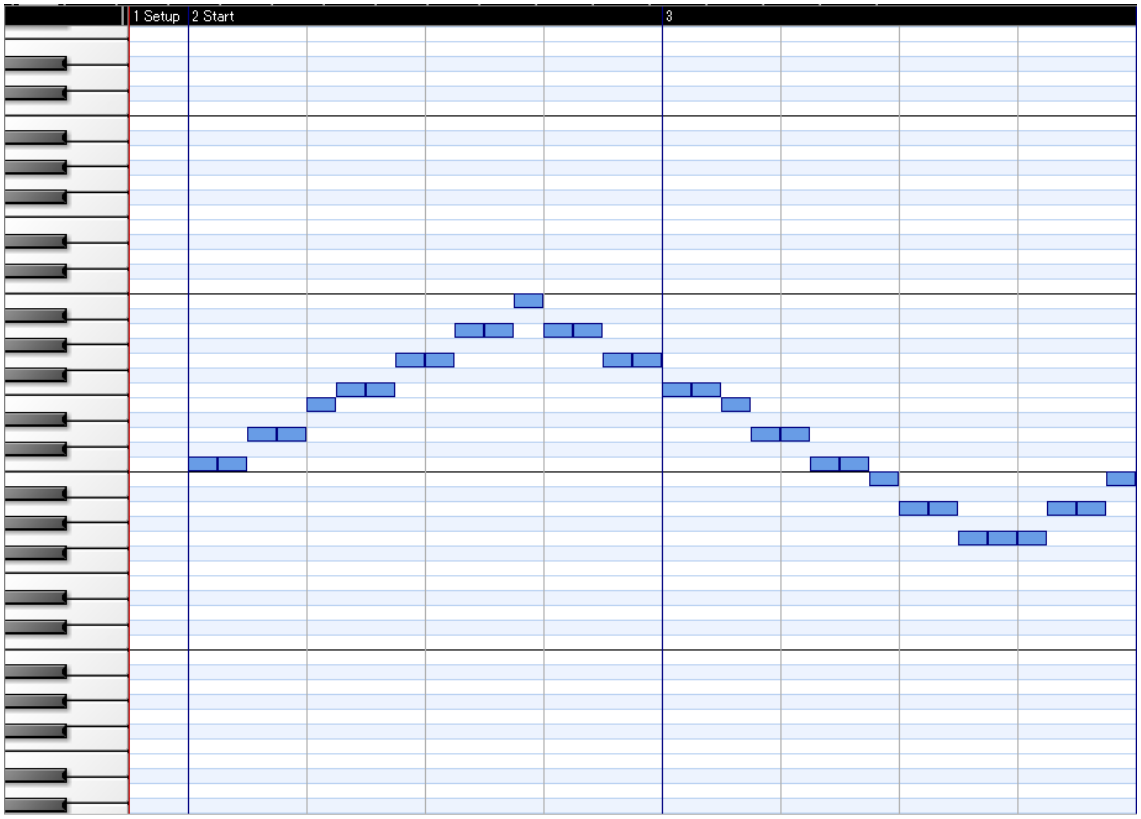

図 3　調修正後の図（ハ長調に修正）

第 3 節 作曲アルゴリズムについて

作曲に関してのアルゴリズムはいくつかあったのだが、ゲーム音楽のアルゴリズムは見当たらなかったため、本研究では独自に用意した。

このアルゴリズムは、ゲーム音楽には繰り返しが多いという点を参考にした[5][6]。各説明と、BNF 記法で表した図を図 4～10 に示す。

$$< Track >::=< Pattern > | < Pattern >< Track >$$

図 4　Track についての定義: ゲーム音楽(Track)は一つ以上のパターン(Pattern)から構成される

$$< Pattern >::=$$

$$< Original >< MinorChangeCollection >< Connector >$$

図 5　Pattern についての定義: パターン(Pattern)は原音(Original)、マイナーチェンジの集合体(MinorChangeCollection)、接続音(Connector)から構成される

$$< Original >::=< Sound > | < Sound >< Original >$$

図 6　Original についての定義: 原音(Original)はパターンの元となる一つ以上の音(Sound)の集合体であり、n(0<n)小節分の長さと m(0≦m)回の繰り返しを持っている

$$< MinorChange >::=< MinorChange > |< Original > |\varepsilon$$

図 7　MinorChange についての定義:マイナーチェンジ(MinorChange)は原音(Original)に変更を加えたものであるが、原音と同一になる場合もある。

$$< MinorChangeCollection >::=$$

$$< MinorChange > \quad |$$

$$< MinorChange >< MinorChangeCollection >|$$

$$\varepsilon$$

図８　MinorChangeCollection についての定義: マイナーチェンジの集合体(MinorChangeCollection)は 0 個以上のマイナーチェンジ(MinorChange)で構成される

$$< Connector >::=< Sound > | < Sound >< Connector > |\varepsilon$$

図９　Connectorについての定義: 接続音(Connector)はパターンとパターンをつなげる役割を持つ一つ以上の音(Sound)の集合体である。構造が原音と同じものであるかのように思えるが、担っている役割、生成に当たっての実装部分、繰り返しが存在しない点や長さが固定されている(小節一つ分)点、存在しない場合もある等、違いが存在する

$$< Sound >::=$$

$$< Channel >< Key >< Time >< Velocity >< During >$$

図１０　Sound についての定義: 音はチャンネル(Channel、値は 0 から 15)、高さ(Key、は 0 から 127)、位置(Time、値はユーザの入力に依存する)、大きさ(Velocity、値は 0 から 127)、長さ(During、値はユーザの入力に依存する)から構成される

第 4 節 メイン構築部

メイン部(main)とはメインメロディのことである。メイン構築部では入力パラメータと上述の作曲アルゴリズムを用いてランダムにメイン部を作成する。

今回は全音符一個が選ばれる確率と 16 分音符 16 個が選ばれる確率は同一とし、休符が出現する確率は 5%とした。図 11、12 に生成されたメイン部の出だし 4 小節の例を示す

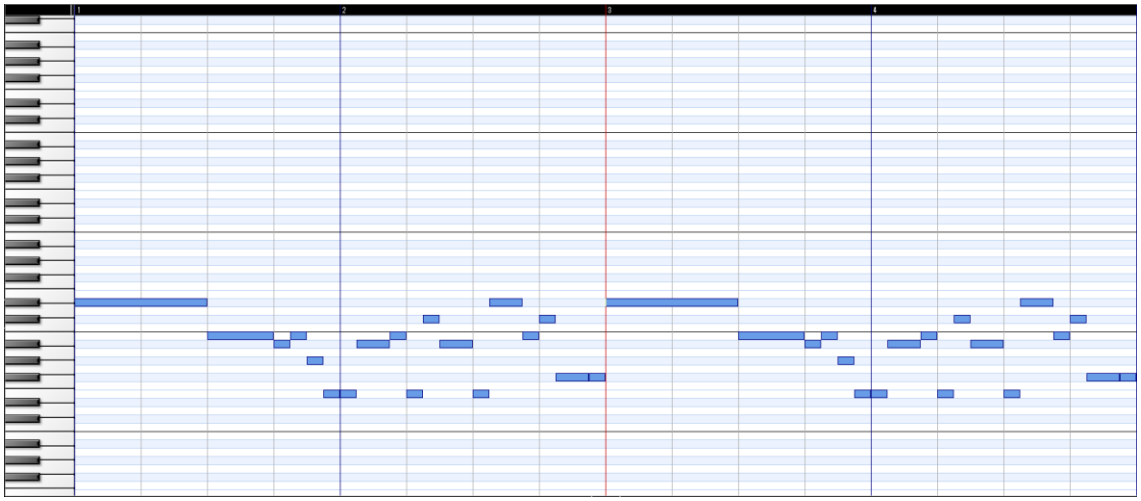

図 11　生成されたメイン部の出だし 4 小節 例 1

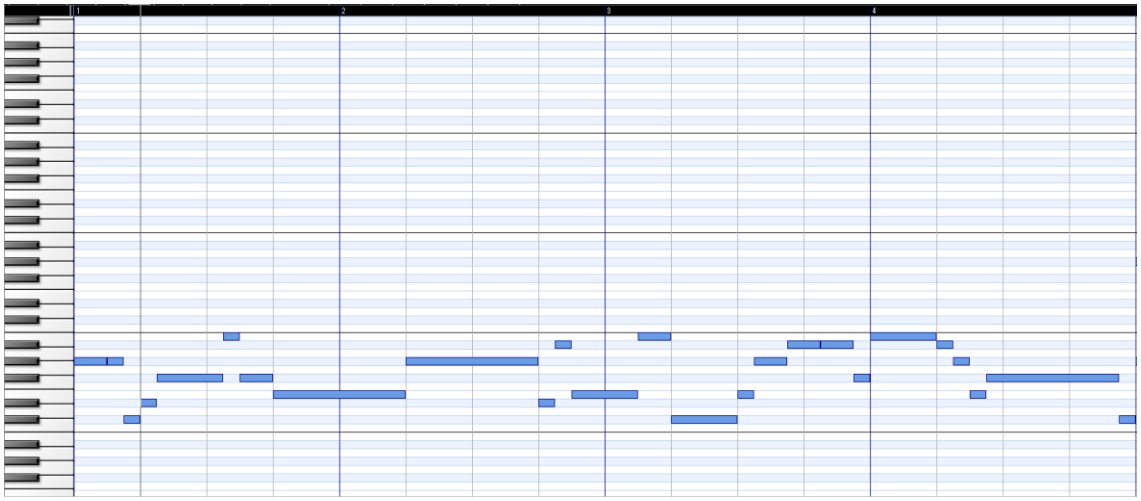

図 12　生成されたメイン部の出だし 4 小節 例 2

第 5 節 サブ構築部

サブ部(sub)とはベース部分のことである。名前をベースではなくサブとしたのは、低い音だけで構成されるわけではないためという理由がある。サブ構築部ではメイン構築部と同様に入力パラメータと作曲アルゴリズムを用いてメロディを作成する。

今回は 2^n ビートの音を生成することにし、休符が出現する確率を 50%とした。図 13、14 に生成されたサブ部の例を示す

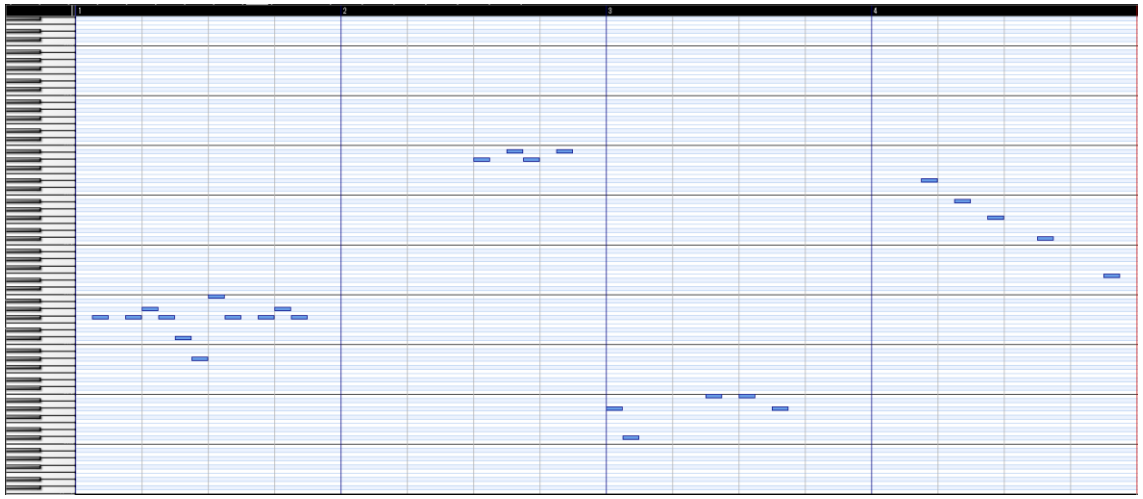

図 13　生成されたサブ部の出だし 4 小節　例 1

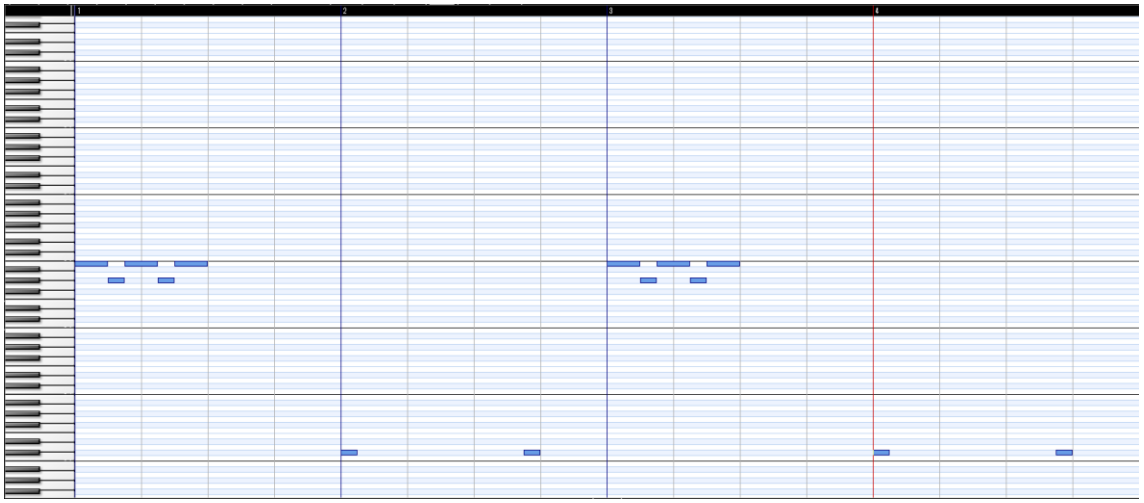

図 14　生成されたサブ部の出だし 4 小節　例 2

第 6 節 コード構築部

コード部(chord)とは和音のことである。メイン部及びサブ部の 1 音 1 音に対して適用して生成する。生成時には不協和音を避けるため音と音の高さの間を 2 つ以上空けた[7][8][9]。

今回はコード部をランダムに生成したため、繰り返しが多い場合にも多少の違いを生み出すことができた。図 15～18 に生成されたコード部の例を示す

図 15　図 11 にコード部を適用後

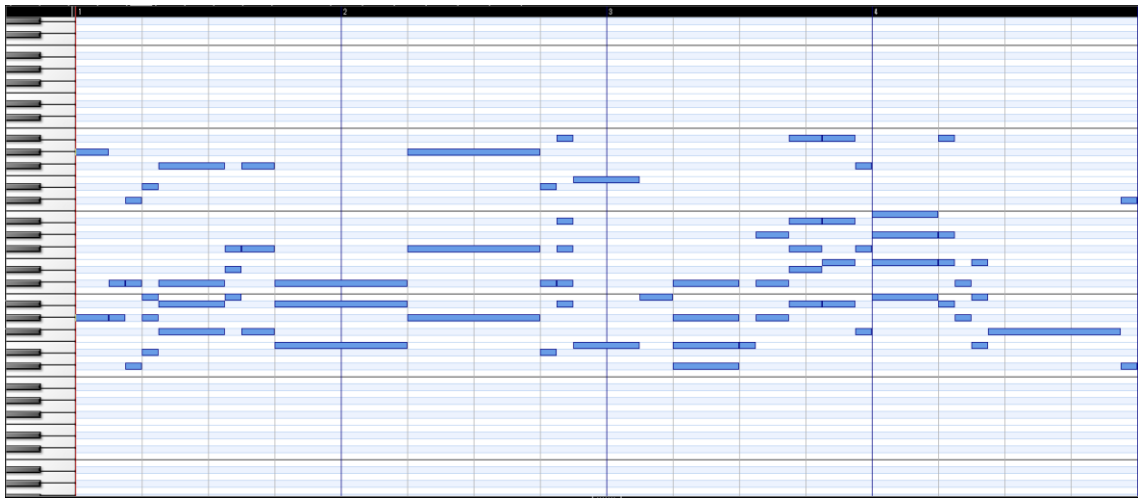

図 16　図 12 にコード部を適用後

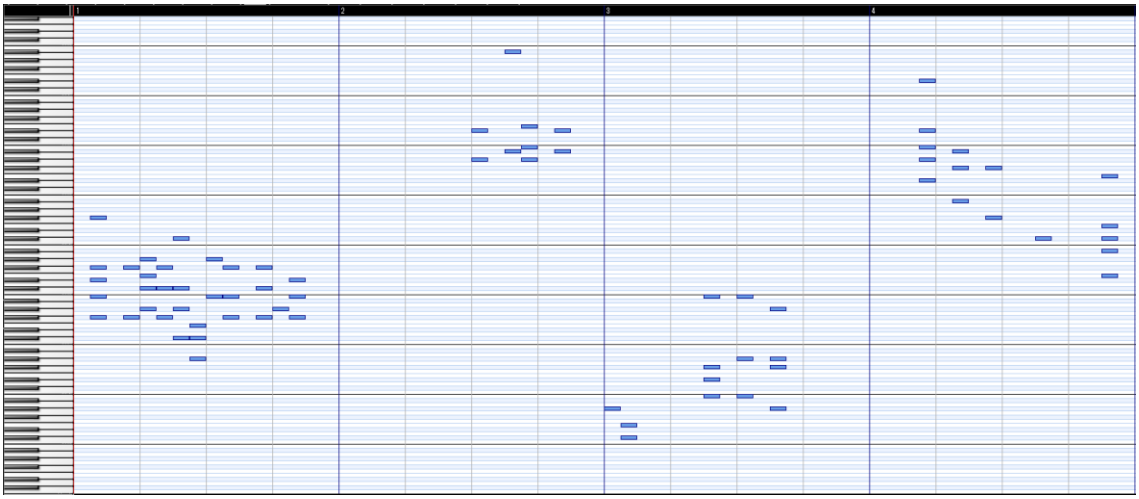

図 17　図 13 にコード部を適用後

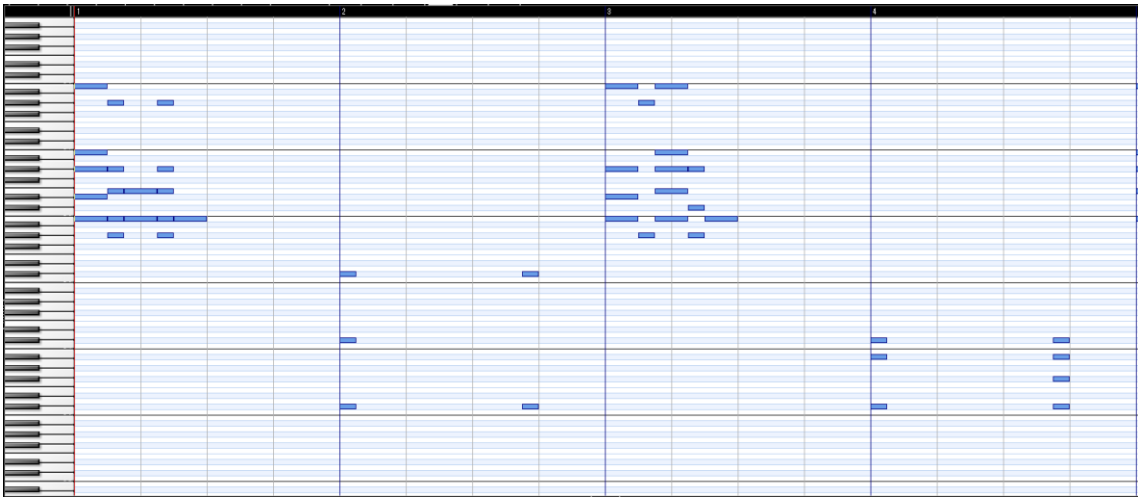

図 18　図 14 にコード部を適用後

第 7 節 ドラム構築部

ドラム部(drum)とは打楽器部分のことである。

今回は 30 種類のパターン［10］を用意し、これらをランダムに組み合わせて打楽器部分生成することにした。図 19、20 に生成されたドラム部の例を示す

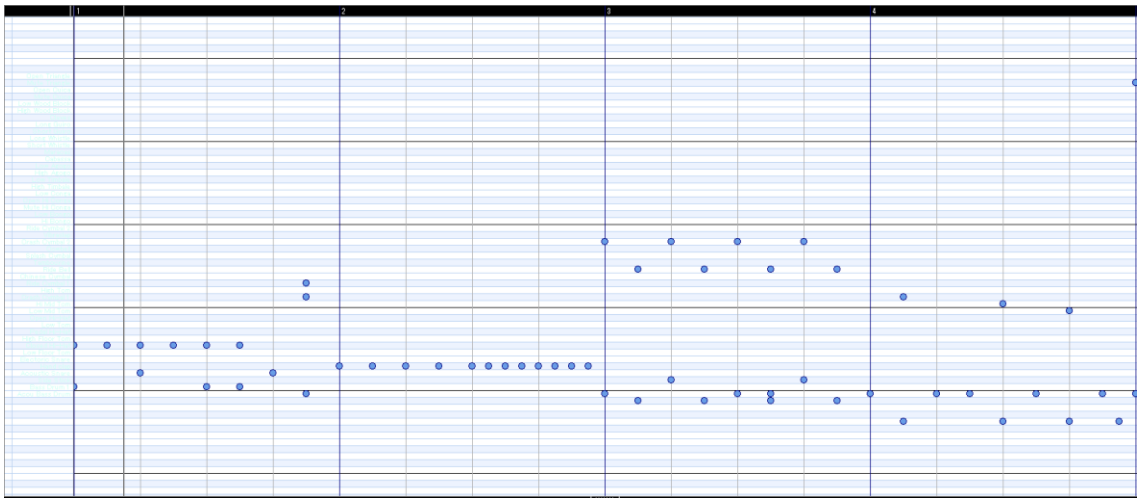

図 19　生成されたドラム部 例 1

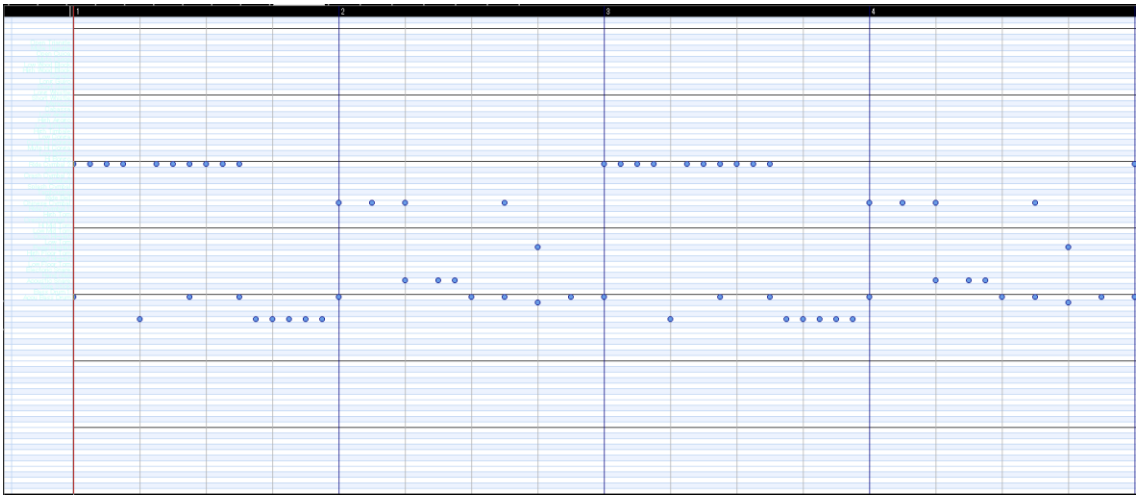

図 20　生成されたドラム部 例 2

第 8 節 出力ファイル

ゲーム音楽作曲アルゴリズムを使用して生成された MIDI 形式データを新規ファイルとして出力する。テンポや分解能は入力 MIDI データと同様とし、複数の MIDI トラックを使用するため SMF1（Standard MIDI Format1）を使用した。出力ファイルを図 21 に示す。

図 21　生成された MIDI 形式データファイル

3 章　出力例

実際に本アプリケーションを使用して生成された MIDI 形式データファイルの例を 2 つ示す。

(例 1) 入力パラメータを 60 秒、BPM120、ハ長調として生成した場合

$$\text{小節数 n} = \frac{60 \times 120}{240} = 30$$

上記の式から小節数は 30 となる。図 22 に生成された MIDI 形式データファイルの全体図を示す。図 22 から小節数が 30 であることがわかる。

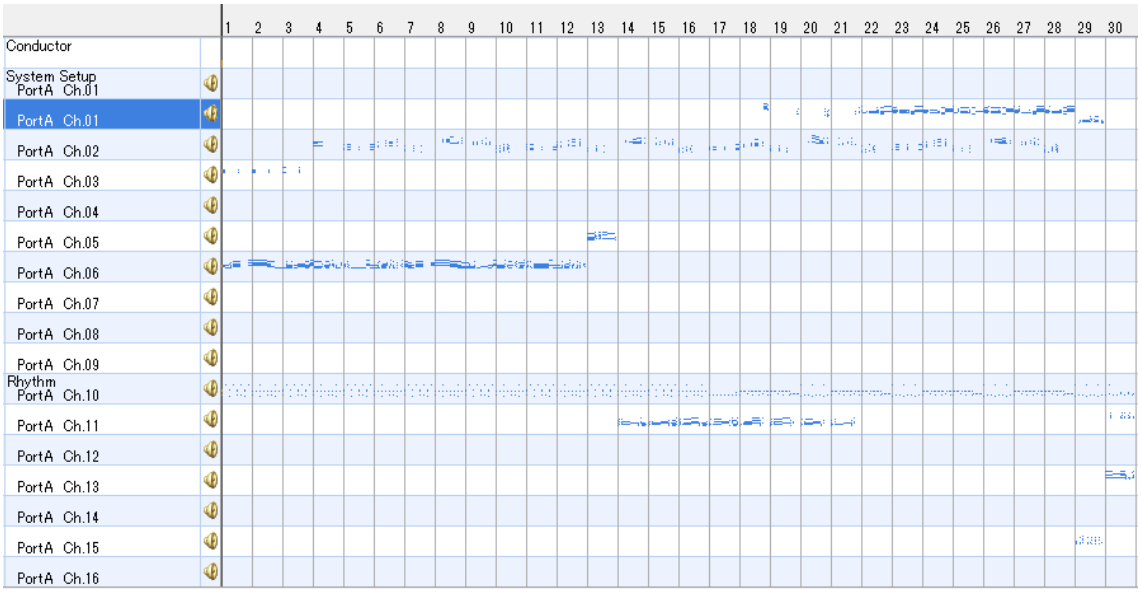

図 22　生成された MIDI 形式データファイルの全体図

生成された MIDI 形式データファイルの出だし4小節を図 23 に示す。図 23 から全てハ長調であることがわかる。

図 23　生成された MIDI 形式データファイルの出だし 4 小節

(例 2) 入力パラメータを 30 秒、BPM140、ロ短調として生成した場合

$$\text{小節数 n} = \frac{30 \times 140}{240} = 17.5 = 17$$

上記の式から小節数は 17 となる。図 24 に生成された MIDI 形式データファイルの全体図を示す。図 24 から小節数が 17 であることがわかる。

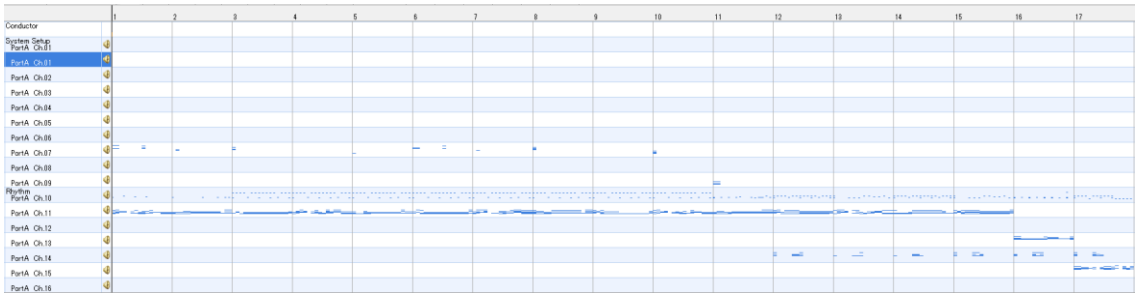

図 24　生成された MIDI 形式データファイルの全体図

生成された MIDI 形式データファイルの出だし4小節を図 25 に示す。図 25 から全てロ
短調であることがわかる。

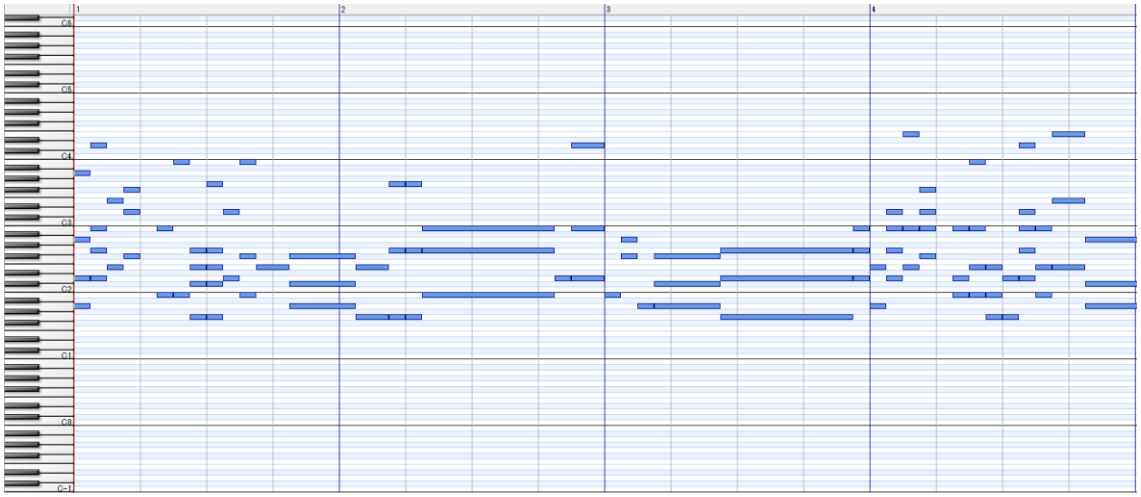

図 25　生成された MIDI 形式データファイルの出だし 4 小節

4 章　生成時間計測

入力パラメータを変化させたときの、ゲーム音楽生成速度の変化を計測した。

入力パラメータごとにゲーム音楽生成を 100 回行い、平均時間、最大時間、最小時間を計測する、という方法で計測した。表 1～11 に入力パラメータと計測時間を記す

表 1 入力パラメータ 時間 60 秒 BPM80 ハ長調

平均時間	最大時間	最小時間
3.767232	7.43436	1.20186

表 2 入力パラメータ 時間 60 秒 BPM100 ハ長調

平均時間	最大時間	最小時間
5.37661	14.0046	1.72557

表 3 入力パラメータ 時間 60 秒 BPM120 ハ長調

平均時間	最大時間	最小時間
6.750663	13.7862	2.39212

表 4 入力パラメータ 時間 60 秒 BPM140 ハ長調

平均時間	最大時間	最小時間
8.962021	19.8652	3.76759

表 5 入力パラメータ 時間 60 秒 BPM160 ハ長調

平均時間	最大時間	最小時間
12.67269	27.5426	4.46169

表 6 入力パラメータ 時間 30 秒 BPM120 ハ長調

平均時間	最大時間	最小時間
2.137765	4.08221	0.836814

表 7　入力パラメータ 時間 40 秒 BPM120 ハ長調

平均時間	最大時間	最小時間
3.187651	6.53786	0.914579

表 8 入力パラメータ 時間 50 秒 BPM120 ハ長調

平均時間	最大時間	最小時間
5.70351	15.4509	1.98809

表 9 入力パラメータ 時間 60 秒 BPM120 ロ長調

平均時間	最大時間	最小時間
7.879718	17.346	2.3711

表 10 入力パラメータ 時間 60 秒 BPM120 ハ短調

平均時間	最大時間	最小時間
7.786187	17.023	2.88134

表 11 入力パラメータ 時間 60 秒 BPM120 ロ短調

平均時間	最大時間	最小時間
6.926476	15.2549	1.61899

上記の結果から、調の変化に関わらず、時間か BPM のどちらかが 10 変化するとゲーム音楽生成の平均時間が 1 秒上がるということが分かった。

5 章　考察

当初はコード進行等の音楽理論を多く使うことを考えていたが、似たような曲が多く出来てしまったことから幅を持たせるためにランダムな生成方法を選択した。

結果として、一曲当たりの生成速度の高さと生成できる音楽の幅がこのアプリケーションの最大の強みとなった。例えば 30 秒の曲を 1 つ作るのに 3 秒かからないため、大量の音楽を高速に作成することができる。

また、実際にゲームプログラミング研究会の人たちに使っていただいたところ、使えるという意見と音量調整をもう少しすると良いという意見を頂いた。

今後は入力パラメータとして、ゲームのジャンルやシチュエーション、場面なども取り入れたいと考えている。これらのパラメータが取り入れることができれば更にユーザの望む曲が生成されやすいアプリケーションになると考えられる

6 章　結論

本研究では独自に用意した作曲アルゴリズムを使用してゲーム音楽を自動生成するアプリケーションの作成を行った。更に実際にユーザに使用してもらい、簡単にゲーム音楽を生成することができるということが分かった。

今後は、ゲームのジャンルやシチュエーション、場面などを入力パラメータとして取り入れていきたい。

参考文献

[1] 函館高専ゲームプログラミング研究会ホームページ

http://hnctgpgk.web.fc2.com/index.html

[2] MIDI 音楽編集ソフト「Domino（ドミノ）」| TAKABO SOFT

http://takabosoft.com/domino

[3] MIDI データ作成・編集用ライブラリ

http://openmidiproject.osdn.jp/MIDIDataLibrary.html

[4] 続きのスクロール・1「調」の種類, その性質

http://pledge.s54.xrea.com/music/dur_moll.html

[5] ゲーム音楽の簡単な作り方～ゲームとかの個人サイト～

http://damedamepg3q.web.fc2.com/column_15.html.

[6]【作曲】戦闘曲の作り方【バトルミュージック】の作曲方法とは？【後編】

http://andy-hiroyuki.hatenablog.com/entry/2016/03/06/192720

[7] 平均律と純正律の和音の比較 - 平野拓一

http://www.takuichi.net/hobby/edu/sonic_wave/temperament_just_intonation/doc_html/temp_just.htm

[8]♦ 平均律について ♦

http://stby.jp/heikinritu.html

[9] 音楽理論を多少数学を使って

http://joewakano.sakura.ne.jp/MusicTheory/music_theory.html

[10] ドラムの打ち方

http://www.circletempo.net/ganriki/kouza/drums.html