ISSN 1882-5370

尚美学園大学芸術情報研究　第 34 号

Journal of Informatics for Arts, Shobi University No.34

研究ノート | Research Notes

ビジュアル・プログラミング環境「Scratch3.0」を用いた音響アプリケーションの開発

Development of Acoustic Application Software Using Visual Programming Environment “Scratch 3.0”

茂出木　敏雄
MODEGI Toshio

尚美学園大学
情報表現学科　講師
Shobi University

2021 年 12 月
Dec.2021

ビジュアル・プログラミング環境「Scratch3.0」を用いた音響アプリケーションの開発

Development of Acoustic Application Software Using Visual Programming Environment “Scratch 3.0”

茂出木 敏雄
MODEGI Toshio

［抄　録］
2020 年度より国内の小学校でプログラミング授業が必修化され、巷に開設された子供向けのプログラミング教室を含めて「Scratch」などのビジュアル・プログラミング開発環境が採用されている。米国 MIT により開発された「Scratch」は、画面上でジグゾーパズルのようにブロックを連結させることによりビジュアルにプログラミングが可能である。一般に「Scratch」は、アニメーションやゲームのアプリケーションを開発するツールとして知られているが、多くの API が満載され、組み込みソフトを含め汎用的なソフトウェア開発が可能である。「Scratch」自体は JavaScript で実装されており、JavaScript の関数に相当する構造化プログラミング機能や、マルチスレッド機能を活用した高度で大規模なアプリケーションを開発できる。マルチメディア機能として、現状の「Scratch3.0」には動画再生や３次元 CG の機能は実装されていないが、波形オーディオ、MIDI、音声合成といった音響処理機能は実装されている。本稿では、「Scratch3.0」を用いた音響分野のアプリケーション開発の基本について紹介する。

キーワード
ビジュアル・プログラミング , スクラッチ , 波形オーディオ , MIDI, 音声合成 , 音響アプリケーション

[Abstract]
Since 2020, in Japanese domestic elementary schools, programming education has become compulsory. In many elementary schools including private programming schools in the streets opened for children, visual programming environments such as “Scratch” have been used for educational tools. Using the “Scratch” developed by the Massachusetts Institute of Technology in the USA, it makes possible a visual programming, by connecting blocks on screen like a jigsaw puzzle. In general, the “Scratch” is known as a development tool for animation or game applications, but it includes a lot of APIs and it can be used as a development tool of generic software including embedded systems. The “Scratch” tool itself has been implemented by the JavaScript programming language. It is possible to develop a high-level and large-scale application, using structured programming and multiple thread functions supported by JavaScript. In the current

“Scratch 3.0” version, video and 3D graphic functions are not supported, but acoustic functions such as waveform audio, MIDI and speech synthesizer are implemented. In this report, we present fundamental programming methods for development of acoustic application, using the “Scratch 3.0” tool.

Keywords
Visual Programming, Scratch, Waveform Audio, MIDI (Musical Instrument Digital Interface), Speech Synthesizer, Acoustic Application

１．はじめに

2020 年度より国内の小学校で、2021 年度より中学校でプログラミング授業が必修化され、文献 1)-3) のように子供向けのプログラミング教科書が数多く出版されるとともに、巷に子供向けのプログラミング教室が数多く開設されている。これらの教室を含めて「Scratch」[4)5)] などのビジュアル･プログラミング開発環境が多くの学校で採用されている。米国 MIT により開発されたビジュアル・プログラミング開発環境「Scratch」は、1972 年にアラン・ケイにより開発されたオブジェクト指向言語「SmallTalk」に端を発しており、その開発環境「Squeak」で開発された「Squeak Etoys」が原点になっている [1)]。この「Squeak Etoys」は、GUI 画面上でジグゾーパズルのようにブロックを連結させることによりビジュアルにプログラミングが可能である。このようなビジュアル・プログラミング手法は、子供向けのプログラミングだけでなく、画像処理のアプリケーション開発 [6)] や、Max/Msp[7)] といった音響信号合成のユーザインタフェース [8)] にも活用されている。

「Scratch」は、スプライト（妖精）と呼ばれるコスチュームと音を備えたキャラクターの動作をスクリプト（台本）上でプログラミングするもので、開発環境はいわば劇場である。「Scratch」は、Web ブラウザ上で動作し、プラットフォームに依存しない構成になっており、英語・日本語に限らず数多くの言語に対応している（2021 年 8 月現在で 71 言語）。日本語版は通常の漢字を混用した「日本語」モードと、子供向けにひらがなのみを使用した「にほんご」モードに切り替えることができる。Windows、Mac のパソコン向けにダウンロード版も提供されているが、Web ブラウザ版はクラウドで動作する API (Application Programing Interface) が提供され、多言語翻訳など高度な機能を活用したアプリケーションを簡単に開発することができる。ただし、現状の「Scratch3.0」では、メール送信などネットワークのアプリケーションを開発する API は用意されていない。

一般に「Scratch」は、子供向けにアニメーションやゲームのアプリケーションを開発するツールとして知られているが、多くの API が満載され、”micro:bit” と呼ばれる小型ボードコンピュータを制御するような組み込みソフトの開発を含め、大人でも十分手ごたえを感じる汎用的なソフトウェア開発ツールになっている。「Scratch」本体は JavaScript で実装されており、子供では理解が困難な配列操作や三角関数など高度な算術関数も使用できるため、統計計算やグラフ表示などのデータサイエンス分野の高度なアプリケーション開発も可能である。また、JavaScript の関数に相当する、独自にブロックを定義して構造化プログラミングを行う機能を備えているため、大規模なアプリケーションを開発するこ

とができる。

特に、「Scratch」が従来の C 言語や JavaScript 言語に比べ優れている機能が、マルチスレッドである。ゲームアプリケーション向けに前述のスプライト（主役、敵、味方、環境など）を複数個定義することができ、各スプライトに独立したスクリプトを定義できるので、複数のスクリプト（スレッド）が同時に実行される。この時、スクリプト間で変数を共有させることができる。また、マルチスレッドは、音楽でいえば合奏である。例えば、複数の音楽ファイルを再生する際、各々個別に再生プログラムを記述し、あたかも指揮者が合図を送るように、メインのプログラムよりメッセージを送るだけで、複数の再生プログラムを同時に実行させる、あるいは一部の再生プログラムを指定時間だけ遅らせて実行させるといった制御が簡単に行える。

マルチメディア機能として、現状の「Scratch3.0」には動画再生や３次元 CG の機能は実装されていないが、波形オーディオ、MIDI、音声合成（音声認識は不可）といった音響処理機能は実装されている。ファイル入出力機能がサポートされていないため、音響信号処理は実現できないが、２次元グラフィックスを用いた波形の表示や、音声、音楽、効果音などの再生や合成の制御は行える。本稿では、「Scratch3.0」を用いたプログラミング手法の基本について述べ、音響分野のアプリケーション開発を行うための基本機能とプログラミング事例について紹介する。

２. 一般的なプログラミングの基本構成と「Scratch3.0」を用いた実装

いかなるプログラミング言語で記述されても、一般的なコンピュータ・プログラムは、上の行から順番に処理される順次処理、指定範囲の処理を繰り返す反復処理、途中で処理内容が複数に分かれる条件分岐のいずれか３つの要素で構成されている [1)]。

この構成は音楽の楽譜でも同様である。楽譜もプログラムと同様に各小節を順番に演奏する順次処理が基本になっている。そして、リピート記号により指定された範囲の小節をリピート演奏したり、D.C.（ダカーポ）や D.S.（ダルセーニョ）と称する記号により、先頭の小節またはセーニョ記号でマークされた小節に戻って（プログラミング言語では GoTo 文といわれる）再度演奏したりするのが反復処理である（音楽では２～３回の繰り返しにとどまるが）。また、リピート演奏時の最終段において１回目カッコと２回目カッコが付与されて、１回目と２回目で演奏対象の小節が異なる場合がある。これが条件分岐である。

図１は与えられた整数が素数であるか否かを判定するプログラムの例である。尚、図のキャプションの末尾に付記されている ” Scratch3_prime_F1.sb3” はプログラムのデータファイル名を示し、以降本稿の図に掲載されているプログラムのデータは、全て後述する著者のホームページよりダウンロードすることが可能である。素数は整数の中で特異な数で、２つの素数 p と q の乗算で与えられる整数 r より、素因数分解により p と q が一意的に定まる。この時、r の桁数が大きくなるとスーパーコンピュータを用いても素因数分解に膨大な処理時間がかかるため、このような整数 r は公開鍵暗号に使用されている。これにより、ビットコインなど暗号資産の安全性も担保されている。

(A)C 言語プログラミング例

```
01 #include <stdio.h>
02 int main(int argc, char* argv[])
03 {
04    int A,M,N;
05
06    printf(" 整数を入力して下さい :");
07    scanf("%d",&A);
08
09     N=2;
10     while (N<=A-1) {
11         M=A%N;
12         if (M==0) {
13            printf(" 素数ではありません ¥n");
14            return 0;
15         }
16         N=N+1;
17    }
18
19    printf(" 素数です ¥n");
20    return 0;
21 }
```

(B)Scratch プログラミング例

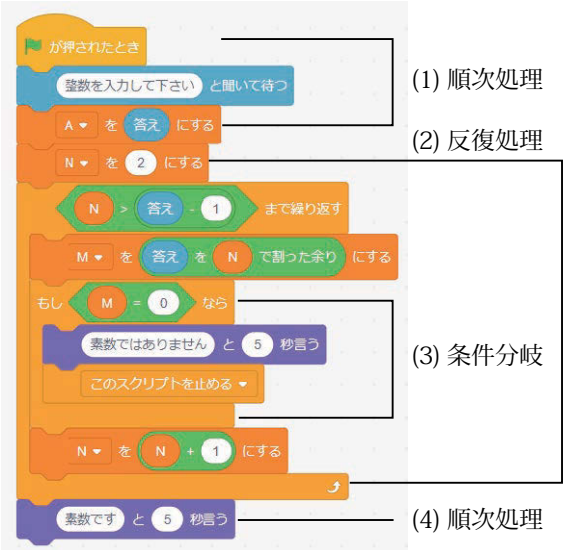


図 1　整数の素数判定：C 言語と Scratch 3.0 によるプログラミング例 (Clang_prime_F1.c, Scratch3_prime_F1.sb3)

図 1 のプログラムは、コード量は少ないが、順次処理、反復処理、条件分岐の 3 要素を備えている。素数とは 1 またはそれ自身の整数でしか割り切れない整数で、与えられた整数 A を 2 以上で A 未満の整数 N で順に割り算して、一個でも割り切れる整数 N が見つかれば、整数 A は素数ではないと判定できる。逆に、2 から与えられた整数 A-1 までの範囲で割り算して、全て割り切れない場合は、整数 A は素数であると判定できる。

図 1 は、素数判定のプログラムを、C 言語および Scratch3.0 で作成した例である。C 言語の 4 行目は使用する変数の宣言で、6・7 行目は整数をキーボードで入力する処理で Scratch の (1) 順次処理に対応する。C 言語の 9 ～ 17 行目は N=2 ～ A-1 の範囲で、N の値を 1 ずつ増加させながら、11 ～ 15 行目の処理を A-2 回だけ繰り返し実行させる処理で、Scratch の (2) 反復処理に対応する。反復処理内の 12 ～ 15 行目は A が N で割り切れる場合（M=0 の場合）に 13 行目で「素数ではありません」と表示を行い、14 行目で反復処理を中断させる処理を行う。これが Scratch の (3) 条件分岐に対応する。C 言語の 19・20 行目は「素数です」と表示を行って一連の処理を終了させる処理で、Scratch の (4) 順次処理に対応する。

続いて、100 以下の整数の中で素数が幾つ存在するかという課題が与えられたとする。この場合は、整数を 2 から 100 まで、 1 つずつ変化させながら、図 1 で記述されたプログラムを 99 回反復させればよい。この時、C 言語では図 1 -(A) の処理を関数で記述することが行われる。関数はプログラミング言語によってはサブルーチンやメソッドとも呼ばれ、一連の処理を繰り返し使用する場合に用いられる。関数を用いたプログラミングは構造化プログラミングや関数型プログラミングとも呼ばれ、現在主流のオブジェクト指向プログラミングではメソッドと称する関数をオブジェクト（クラス）内に定義して、多人数による大規模なプログラム開発を可能にしている。リズム音楽でも関数に対応する既作成の数小節のループ素材を組み合わせて作曲する手法があり、関数やループ素材を数多く集

めたものはライブラリと呼ばれる。

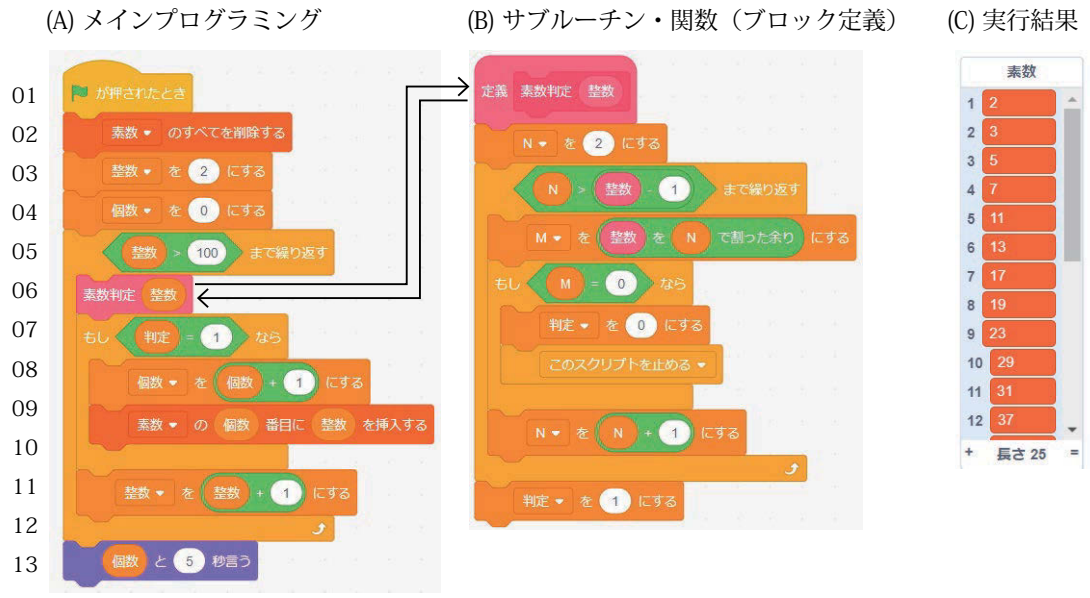


図２ 整数の素数判定：ブロック定義による構造化プログラミング例 (Scratch3_prime2_F2.sb3)

この関数に相当する機能は、Scratch ではブロック定義と呼ばれ、C 言語の関数 f(p) と同様に引数（パラメータとも呼ばれる）p を渡すことができる。図１-(B) を「素数判定」という名称の独自定義のブロックで表現したのが図２-(B) である。判定対象の整数は「整数」という名称の仮引数で定義され、判定結果が素数の場合は「判定」変数を 1、素数でない場合は「判定」変数を 0 にセットする。図２-(A) はメインプログラムで、3 から 12 段目のブロックで、「整数」変数を 2 から 100 まで 1 ずつ変化させながら、6 段目のブロックで「素数判定」独自定義ブロックを繰り返し呼び出している。7 から 10 段目のブロックで、「判定」変数が 1（素数）の場合、「個数」変数をカウントアップさせるとともに、「素数」という名称のリストに素数と判定された「整数」変数の値を登録している。リストは C 言語など一般的なプログラミング言語では配列と呼ばれ、図２-(A) の処理を終了すると、図２-(C) のように判定された 25 個の全ての素数を確認できる。このようにして、C 言語の関数や配列に相当する機能を Scratch3.0 でも使用できる。

３．「Scratch3.0」に実装されている音響プログラミングのための基本機能

図１-(A) に示す従来の C 言語等を用いたプログラミング演習が嫌われる第一の理由として、数式が混在した英語のような非日常的な文章を厳格な文法に基づいて記述しなければならない点が挙げられる。特に、C 言語のようなコンパイラ言語では、スペルミスや文末のセミコロン ";" を忘れるなど、文法上の誤りに伴うコンパイルエラーに悩まされ、初心者にはエラーを完全に除去できるまで多くの時間が割かれる。仮にコンパイルに成功しても、実行時にセグメンテーションフォルト（C 言語では配列やポインター操作のミス）などのエラーが生じることがあり、初心者にはその原因を調べるのが難しい。

第二の理由として、例題が素数判定のように、何の役に立つかわからない、無味乾燥な

題材が与えられることが多いことが挙げられる。仮に初心者に興味を抱かせる例題を準備し、本章で紹介するような視聴覚に訴える音響プログラミングの機能を従来の C 言語等で実装させようとすると、WindowsAPI 関数を用いたかなり煩雑なプログラムになり、とても初心者には歯が立たないため、無味乾燥な例題にならざるを得ないのである。

これに対して、Scratch を用いると、文法エラー等に悩まされることなく、以下で述べるように実用的な音響プログラミングの機能を比較的簡単に実装できる。

3.1.　サウンド再生、音楽音階・和音再生の基本

本節では C 言語等に比べて Scratch の優れた機能として前述のマルチスレッドについて、サウンド再生のプログラミングを題材に紹介する。「Scratch3.0」を用いた音響再生では、任意の音素材を再生できる波形サウンドと、楽器音を再生できる MIDI サウンドの二種を使用することができる。本節では、楽器音で音楽音階「ドレミ」を再生する方法として、図 3 -(A)(C) のように波形サウンドを使用する方法と、図 3 -(B)(D) のように MIDI サウンドを使用する方法の 2 通りについて述べる。

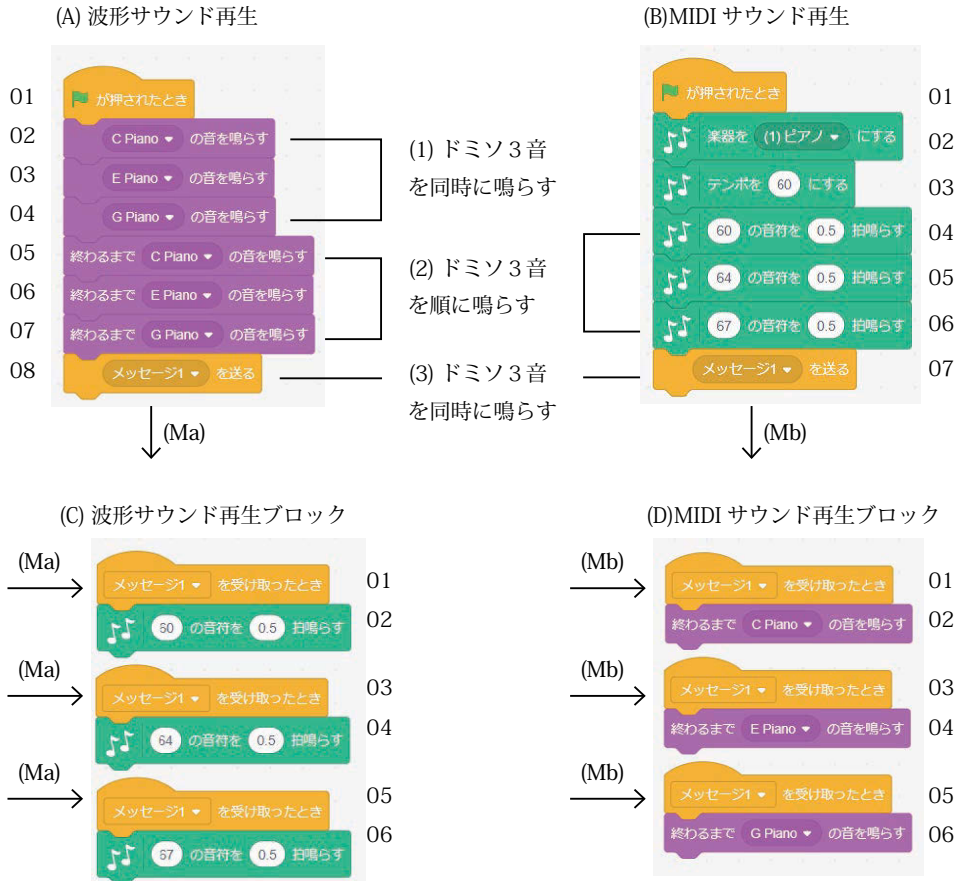


図 3　波形サウンドと MIDI サウンド再生の基本（単音と和音、拡張機能「音楽」使用）(Scratch3_chordw_F3A.sb3, Scratch3_chordm_F3B.sb3)

波形サウンドを使用するには、Scratch 基本機能の「音」を使用し、「音を選ぶ」という音素材ライブラリにデフォルトのキャラクターの猫の鳴き声「ニャー」を含む種々のジャンルの音素材が収録されている。ユーザが録音・編集して新規な音素材をライブラリに追加することはできるが、音声認識のようにプログラム内で録音制御を行うことはできない。

このライブラリには、多くの楽器音も基本音階別に収録されている。図３-(A)(C) は、音素材ライブラリよりピアノのドミソの音に対応する「C Piano」、「E Piano」、「G Piano」の３素材を選択した例である。波形サウンドを再生するブロックとして、図３-(A) の２～４段目のように指定された各音素材の再生を開始した瞬間に、制御がプログラムに戻って次のブロックの実行に進む非同期型と、図３-(A) の５～７段目のように各音素材の再生が終了するまで待機し、再生が終了したら次のブロックの実行が開始される同期型がある。前者の場合は、図３-(A) の２～４段目の音素材を再生する３個のブロックが殆ど同時に実行されるため和音演奏となり、後者の場合は、図３-(A) の５～７段目の３個の音素材が順番に実行されるため単音演奏となる。

続いて、MIDI サウンドを使用するには、Scratch の「拡張機能の追加」で「音楽」を選択し、「音楽」のブロックパレットを追加する必要がある。MIDI サウンドで再生するには、まず図３-(B) の２段目のように楽器音を GM (General MIDI) 音源より選択し、図３-(B) の３段目のようにテンポ（BPM、１分あたりの拍数 beet）を指定する。そして、図３-(B) の４～７段目のように「(N) の音を (T) 拍鳴らす」というブロックを実行させる。N は音符の音高を MIDI ノートナンバーで指定し、T は音価（音の長さ）で、拍の単位で指定する。図３-(B) の４～７段目を実行することにより、「ド」「ミ」「ソ」の音が 0.5 拍（＝ 0.5 秒）ずつ順番に単音演奏される。これは、波形サウンドにおける図３-(A) の５～７段目に対応する。

MIDI サウンドには、図３-(A) の２～４段目に対応する非同期型のブロックは用意されていないので、和音演奏を行ないたい場合には、Scratch のマルチスレッド機能を活用する必要がある。マルチスレッドにおけるスレッドとは、一連のプログラムで指示された処理を指し、マルチスレッドとは、複数の処理を同時に実行させる機能である。例えば、図３-(D) のように、あらかじめ、「メッセージ１を受け取ったとき」と「(N) の音を (T) 拍鳴らす」という２つのブロックで構成されるプログラム（スレッド）を３本作成しておく。この状態で、図３-(B) の７段目の「メッセージ１を送る」というブロックを実行すると、メッセージ１ (Mb) が図３-(D) の 1,3,5 段目のブロックに渡され、2,4,6 段目のブロックが同時に実行される。これにより、図３-(A) の２～４段目と同様に「ド」「ミ」「ソ」の和音が演奏される。

このマルチスレッド機能は図３-(A)(C) の波形サウンドにも適用でき、図３-(C) では図３-(D) と同様に、「メッセージ１を受け取ったとき」と「終わるまで (W) の音を鳴らす」という２つのブロックで構成されるプログラム（スレッド）を３本作成しておく。そして、図３-(A) の８段目の「メッセージ１を送る」というブロックを実行すると、メッセージ１ (Ma) が図３-(C) の 1,3,5 段目のブロックに渡され、2,4,6 段目のブロックが同時に実行される。これにより、図３-(A) の２～４段目と同様に「ド」「ミ」「ソ」の和音が演奏される。

3．2．音声合成、多言語翻訳の基本

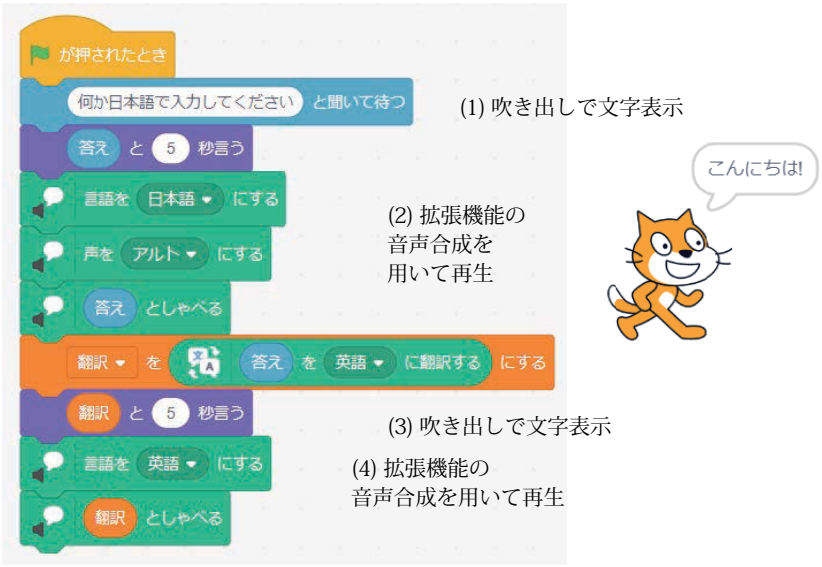


図 4 音声合成の基本（拡張機能「音声合成」と「翻訳」使用）(Scratch3_trans_F4.sb3)

本節では C 言語等に比べて Scratch の優れた機能として音声合成と多言語翻訳の機能を活用したプログラミング手法の基本について紹介する。2021 年 8 月現在の「Scratch3.0」では、音声合成は 23 ケ国語、多言語翻訳は 48 ケ国語に対応している。従って、Scratch は言語翻訳アプリケーションや語学教育教材の制作にも活用できる。これらの機能を使用するには、Scratch の「拡張機能の追加」で「音声合成」と「翻訳」を選択し、各々のブロックパレットを追加する必要がある。現状の「Scratch3.0」では音声認識の機能はまだ実装されていないが、テキスト入力された文章に対して、図 4 に示すような簡単なプログラミングにより音声読み上げや多言語翻訳を実行させることができる。

図 4 の 2 段目のブロックでは日本語でテキストを入力してもらうようにしているが、言語は 23 ケ国語のいずれかから選択できる。3 段目のブロックの変数「答え」は 2 段目で入力したテキストで、「(答え）と（5）秒言う」というブロックは実際に音声合成によりしゃべるわけではなく、図 4 右の猫キャラクターに吹き出しでテキストを表示させるものである。4 ～ 6 段目のブロックで、拡張機能の音声合成を実行しており、4,5 段目で音声合成対象の言語と声の高さを指定し、6 段目のブロックで、実際に音声でしゃべらせている。7 段目のブロックで拡張機能の翻訳を実行しており、変数「答え」に入力されているテキストを英語に翻訳し、翻訳されたテキストは変数「翻訳」に代入される。そして、8 ～ 10 段目のブロックで翻訳されたテキストに対して、3 ～ 6 段目のブロックと同様に、キャラクターに吹き出しで表示し、音声合成でしゃべらせている。

3．3．波形表示の基本

本節では C 言語等に比べて Scratch の優れた機能として 2 次元グラフィックスの機能を活用したプログラミング手法の基本について紹介する。GPU を用いた 3 次元グラフィックスの機能は、ゲームプログラミングにおいて重要であるが、既に Unity [9] といった優れたビジュアル･プログラミング開発ツールが普及していることもあり、現状の「Scratch3.0」

には実装されていない。ただ、今後 Unity の機能が包含されるかもしれない。

図５・６は波形サウンド素材を可視化するための波形表示を行うプログラム例である。ただし、現状の「Scratch3.0」ではファイル入出力機能が実装されていないため、波形サウンド素材である WAV 形式ファイルを読み込んで波形表示をすることはできない。そこで、図６では正弦波（サインカーブ）を生成してグラフィック表示させる事例を示している。

３．３．１．座標軸の表示

図５は波形表示を行うためのＸＹ２次元座標軸の枠線を表示するプログラムである。図５-(A) の２段目のブロックではグラフィック表示画面であるステージ全体をクリアしている。２次元グラフィックスは、キャラクターを動かして絵を描かせる方法をとるため、３段目のブロックではキャラクターの大きさを標準の 50% の小さめに設定している。ステージの座標系は、-240 ≤ X ≤ 240、-180 ≤ Y ≤ 180 の範囲に定義されているため、4,5 段目のブロックでは、「幅」変数を 200 画素、「振幅」変数を 140 画素に設定している。そして、６段目のブロックで図５-(B) に示す「枠表示」独自定義ブロックを実行させている。

図５-(B) に示す「枠表示」独自定義ブロックでは、まず 2,3 段目のブロックにて、描画するペンの太さ（2）と色（黒）を指定している。4 段目のブロックにより、X=-(幅) に設定し、5,6 段目のブロックでは図５-(C) に示す「直線描画」独自定義ブロックを実行させることにより、Ｘ軸（水平線）とＹ軸（垂直線）を描画している。図５-(C) の「直線描画」独自定義ブロックでは、４つのパラメータ X1, Y1, X2, Y2 により座標 (X1,Y1) から座標 (X2,Y2) まで、図５-(B) の 2,3 段目のブロックにて指定されたペンの仕様に基づいて直線を描画している。これにより、図６-(F) の中に描画されている２本の直線が得られる。

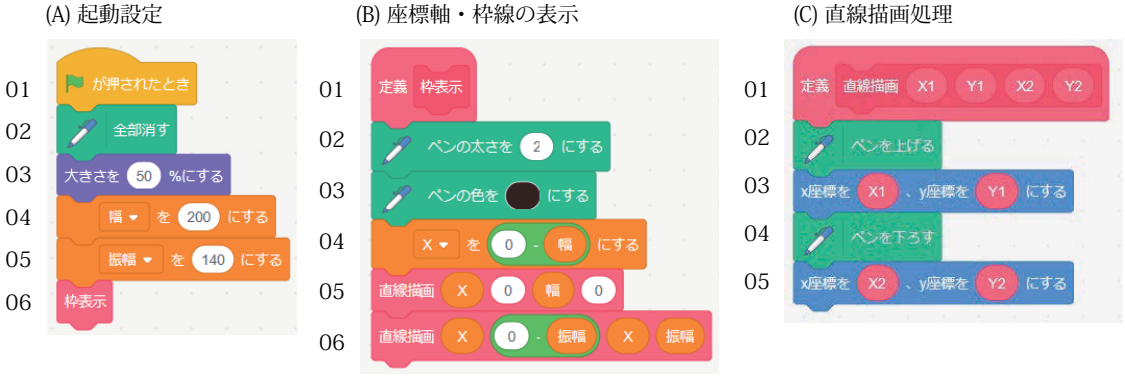


図５ 波形表示の基本１（座標軸の描画、拡張機能「ペン」使用）(Scratch3_frame_F5.sb3)

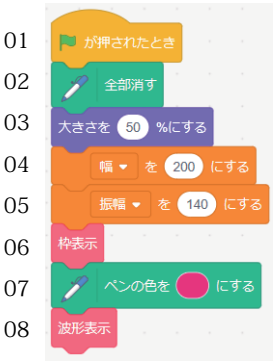
３．３．２．サインカーブの表示

図６は図５で作成した枠線の上にサインカーブの波形を表示するプログラムである。起動設定として、図５-(A) の代わりに図６-(A) を使用し、図５-(B)(C) は本プログラムでもそのまま流用する。

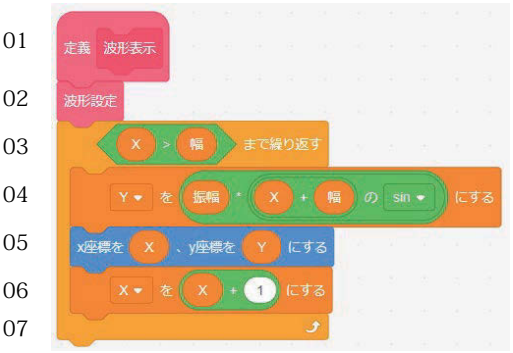
図６-(A) の６段目のブロックまでは図５-(A) と同様である。4,5 段目のブロックで、「幅」変数を 200 画素、「振幅」変数を 140 画素に設定しているので、表示する波形を、Y=(振幅) × sin{2π (X+(幅))/360} なる計算式で与えると、グラフィック表示画面の中心座標

を (0,0) として、-200 ≤ X ≤ 200、-140 ≤ Y ≤ 140 の範囲にプロットされる。7 段目のブロックでは、プロットするペンの色を指定している。図 6 -(A) の最下段の 8 段目のブロックで、図 6 -(D) に示される「波形表示」独自定義ブロックの一連のプログラムを実行させている。

(A) 起動設定



(D) サインカーブの表示



(E) 波形設定

(F) サインカーブの表示結果

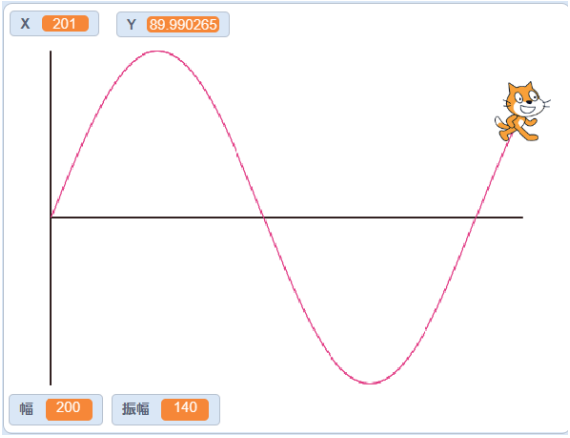


図 6 波形表示の基本 2（サインカーブの描画、拡張機能「ペン」使用）(Scratch3_sine_F6.sb3)

図 6 -(D) の 2 段目のブロックでは、図 6 -(E) に示される「波形設定」独自定義ブロックの一連のプログラムを実行させている。図 6 -(E) の 2 段目のブロックでは、プロットするペンの太さを指定している。図 6 -(E) の 3,4 段目のブロックでは、プロットする (X,Y) 座標の初期位置を指定しており、(-200,0) を起点にしている。図 6 -(E) の 5,6 段目のブロックでは、ペンを上げた状態（描画しない状態）で、キャラクターの位置を 3,4 段目のブロックで定義した初期位置 (-200,0) に移動させている。そして、7 段目のブロックにより、ペンを下した状態（描画する状態）に設定している。

図 6 -(D) の 3 段目のブロックに戻ると、ペンを下した状態（描画する状態）で、3 ～ 7 段目のブロックで定義される反復処理を X ≤ 200 まで繰り返し実行させ、キャラクターの位置を連続的に変化させながらサインカーブを描画する。

具体的には 4 段目のブロックで、現在の X 座標を基に前述の計算式 Y=(振幅) × sin{2 π (X+(幅))/360} に基づいて、Y 座標の値を計算し、5 段目のブロックで、キャラク

ターを (X,Y) の位置にペンを下した状態で移動させている。三角関数の角度の単位はラジアンで与えるのが一般的であるが、Scratch に組み込まれている三角関数では 360 度単位で与えることになっている。そのため、4 段目のブロックでは、Y=(振幅) × sin(X+(幅)) という計算式になっている。そして、6 段目のブロックで、X=X+1 により X 座標を更新し、3 段目のブロックに戻って、X の値が 200 を超えるまで繰り返し実行させている。

4.「Scratch3.0」の基本機能を活用した音響プログラミング事例

4.1.　ネコとピアノの音階合奏（波形サウンド効果）

3.1 節で述べた波形サウンド再生の機能を活用して、デフォルトの猫キャラクターの鳴き声「ニャー」と単一の楽器音「A Piano」のピッチを音楽音階に合わせて変調させ、ネコとピアノの合奏をさせるプログラミング例について紹介する。図 7 -(A) がメインプログラムで、2 段目のブロックの変数「C」は MIDI のノートナンバーと類似した半音階の番号で、3 ～ 10 段目のブロックで、C=0（「ド」の音）から C=12（1 オクターブ高い「ド」の音）まで変化させながら反復させている。8 段目のブロックにより、変数「C」は原則 2 ずつ全音の幅で増加させている。しかし、9 段目のブロックにより、図 6 -(C) に示す「音階補正処理」を実行することにより、C=6（「ファ＃」の音）と C=13（1 オクターブ高い「ド＃」の場合は半音下げる補正を行っているので、結局、「ド」「レ」「ミ」「ファ」「ソ」「ラ」「シ」「ド」の 8 音を鳴らしている。

図 7 -(A) の 4,5 段目のブロックでは、演奏時のピッチの増減率の変数「P」[%] を、$P=100 \times 2^{C/12}-100$ なる計算式で算出している。ただし、Scratch に組み込まれている指数関数の底は 10 またはネイピア数 e のいずれかに限定されるため、$2^{C/12}$ は、$(10^{0.301})^{C/12}=10^{0.301 \times C/12}$ のように底を 10 とする指数関数に置き換えて計算する。また、図 7 -(A) の 4 段目のブロックでは、使用する音素材のピッチが元々高めなので、$10^{0.301 \times (C-6)/12}$ のように 6 半音だけ下げて鳴らすようにしている。

(A) 音階演奏メイン

(B) 合奏処理

(C) 音階補正処理

図 7 ネコとピアノの音階合奏（波形サウンド効果）(Scratch3_audio_F7.sb3)

このようにして、変調させるピッチの増減率が決定したら、図７-(A) の６段目のブロックにより、図７-(B) に示す「合奏処理」を実行させる。図７-(B) の２段目のブロックでは先に算出したピッチの増減率「P」を用いて、「ピッチの効果を（P）にする」の設定を行う。そして、図７-(B) の 4,6 段目のブロックで楽器音「A Piano」と鳴き声「ニャー」を同時に鳴らすと、各々ピッチが指定された増減率だけ変化して再生される。この時、3,5 段目のブロックにて、各々の音素材を再生する際の音量バランスを調整できる。

４．２．メトロノーム（リズムマシン、拡張機能「音楽」使用）

3.1 節で述べた MIDI サウンド再生の機能を活用して、メトロノームあるいはリズムマシンを実現するプログラミング例について紹介する。図８-(A) の 2,3 段目のブロックにて、メトロノームのテンポと拍子のパラメータを指定し、５段目のブロックにて、MIDI サウンドのシステムパラメータにもテンポを設定している。メトロノームの振り子の代わりに、図８-(C) のようにデフォルトの猫キャラクターを時計周りに回転させる方法をとるため、４段目のブロックで、キャラクターの初期表示サイズを大きき目（200%）に指定しているが、後述するように強拍と弱拍に基づいてキャラクターの大きさを２段階に変化させる。６段目のブロックでは、拍をカウントする変数を「C」として、C=1 に初期化して、７段目のブロックにて、図８-(B) のプログラムを起動している。

(A) 起動設定

(B) リズムマシン・メイン

(C) 実行結果

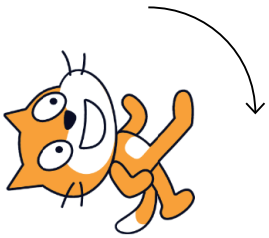


図８ メトロノーム（リズムマシン、拡張機能「音楽」使用）(Scratch3_rhythm_F8.sb3)

図８-(B) は、２～ 15 段目のブロックで無限ループを構成しており、スペースキーをクリックするか、プログラムの停止ボタンをクリックするまで反復実行される。３段目のブロックにてキャラクターの回転角度を計算している。例えば、４拍子の場合は、C=1 を

90 度（キャラクターの初期位置、上向き）として、C=2 で 180 度、C=3 で 270 度、C=4 で 0 度になる。図 8 -(C) の例では、キャラクターの向きは、C=4 で 0 度である。

C=1 の場合は、アクセントとして、4 ～ 6 段目のブロックによりキャラクターの大きさを大き目（200%）に設定して、「（クラッシュシンバル）のドラムを 1 拍鳴らす」、それ以外の拍の場合は、7 ～ 9 段目のブロックによりキャラクターの大きさを小さ目（150%）に設定して、「（スネアドラム）のドラムを 1 拍鳴らす」を実行する。11 ～ 14 段目のブロックにより、変数「C」をカウントアップさせ、C>（拍子）になったら C=1 に初期化する処理を行って、2 段目のブロックに戻り同様な処理を繰り返す。

4．3．　波形合成・うなり（拡張機能「ペン」使用）

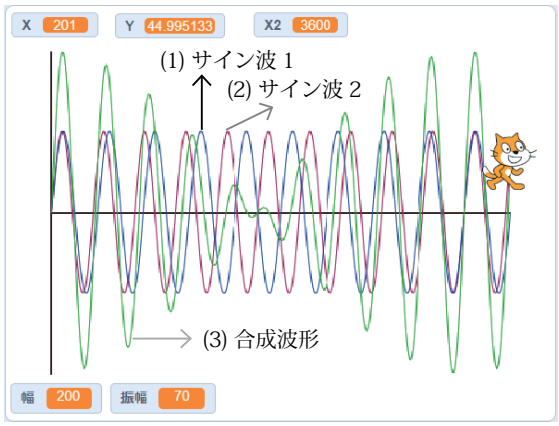
3.3 節で述べた波形表示の機能を活用して、周波数（周期）がわずかに異なる 2 つのサイン波形を合成すると、周波数差に基づく基本周波数をもつうなり（ビート）が発生する様子を表示するプログラミング例について紹介する。

図 9 は、図 5 で作成した枠線の上に 3 本の波形サウンド素材を可視化するための波形表示を行うプログラム例である。起動設定として、図 5 -(A) の代わりに図 9 -(A) を使用し、図 5 -(B)(C) は本プログラムでもそのまま流用する。

(A) 起動設定

(D) 波形表示メイン

(F) 実行結果



(G) うなり音の再生例

図 9 波形合成・うなり（拡張機能「ペン」使用）(Scratch3_beetw_F9.sb3 Scratch3_beet_F9.sb3)

図９-(A) の６段目のブロックまでは図５-(A) および図６-(A) と同様である。図６-(A) からの相違点として、７段目のブロックにて変数「振幅」を５段目のブロックで設定した 140 から 70 に変更している。その理由は合成波形の振幅が最大２倍に膨らむ可能性があり、合成波形を表示枠に収めるためである。また、図６-(A) の 7,8 段目の波形の色指定と波形表示のブロックに対しては、図９-(A) では８～ 13 段目の６ブロックに拡張し、図９-(D) に示される独自定義ブロック「波形表示」を３回呼び出している。この時、「倍率」と「倍率２」なる２種類のパラメータを併せて設定している。

図９-(D) の独自定義ブロック「波形表示」の基本構成は、図６-(D) と同じである。図９-(D) の２段目では、図６-(E) の独自定義ブロック「波形設定」をそのまま呼び出しているので、図９ではプログラムの掲載を省略する。

図９-(D) が図６-(D) と異なる点は、図６-(D) の４段目の三角関数を用いた計算式を、図９-(D) の 4,5 段目に示されるように、Y=(振幅) × sin{(X+(幅)) × (倍率)} と拡張して、「倍率」パラメータを追加して、周波数を変更可能にしたことである。更に、「倍率２」パラメータも追加して、「倍率２」パラメータに０以外の正の値が設定されている場合は、図９-(D) の 7,8 段目に示されるように、Y=Y+ (振幅) × sin{(X+(幅)) × (倍率 2)} なる計算を追加し、「倍率」と「倍率２」で定義される２種のサインカーブの合成値を計算する機能も追加した。

これにより、図９-(A) の９段目のブロックにより「倍率」=10 のサインカーブ、11 段目のブロックにより「倍率」=9 のサインカーブ、13 段目のブロックにより「倍率」=10 のサインカーブと「倍率２」=9 のサインカーブの合成波形からなる３種の波形が、図９-(F) のように、各々 8,10,12 段目のブロックで指定される３色で色分けされて重畳表示される。

図９-(F) に示される波形は、Scratch では WAV 形式等のファイルに収納して音として聞くことができないので、3.1 節で述べた波形サウンド再生の機能を用いて、別途うなり音を合成して再生するプログラミング例を図９-(G) に示す。ただし、図９-(G) のうなり音は、図９-(F) の波形に基づくものではなく、うなり音を生成する事例として別途提示したものである。このプログラムは、3,4 段目のブロックで指定される 220Hz のピアノ音と 6,7 段目のブロックで指定される 233Hz のピアノ音を同時に再生するもので、再生すると２音の周波数差 13Hz で振幅変調されたうなり音が聞こえる。

４．４．　ピアノ鍵盤・電子楽器アプリの開発例（拡張機能「ペン」と「音楽」使用）

本節では本稿での最後の Scratch プログラミング例として、比較的大規模な電子楽器のアプリケーション開発事例を紹介する。２次元グラフィックス機能を用いてピアノ鍵盤を表示し、対話形式に鍵盤をクリックすることにより MIDI 機能を用いて指示された楽器音を鳴らすことができる手動演奏機能と、あらかじめ音符で定義されたメロディーを基に、指定された鍵盤を自動的に鳴らす自動演奏機能のプログラミング例について述べる。

４．４．１．　ピアノ鍵盤の表示（拡張機能「ペン」使用）

電子楽器の基本形態として、3.3 節で述べた波形表示の機能を活用して、１オクターブ強のピアノ鍵盤をグラフィック表示させるプログラミング例について紹介する。

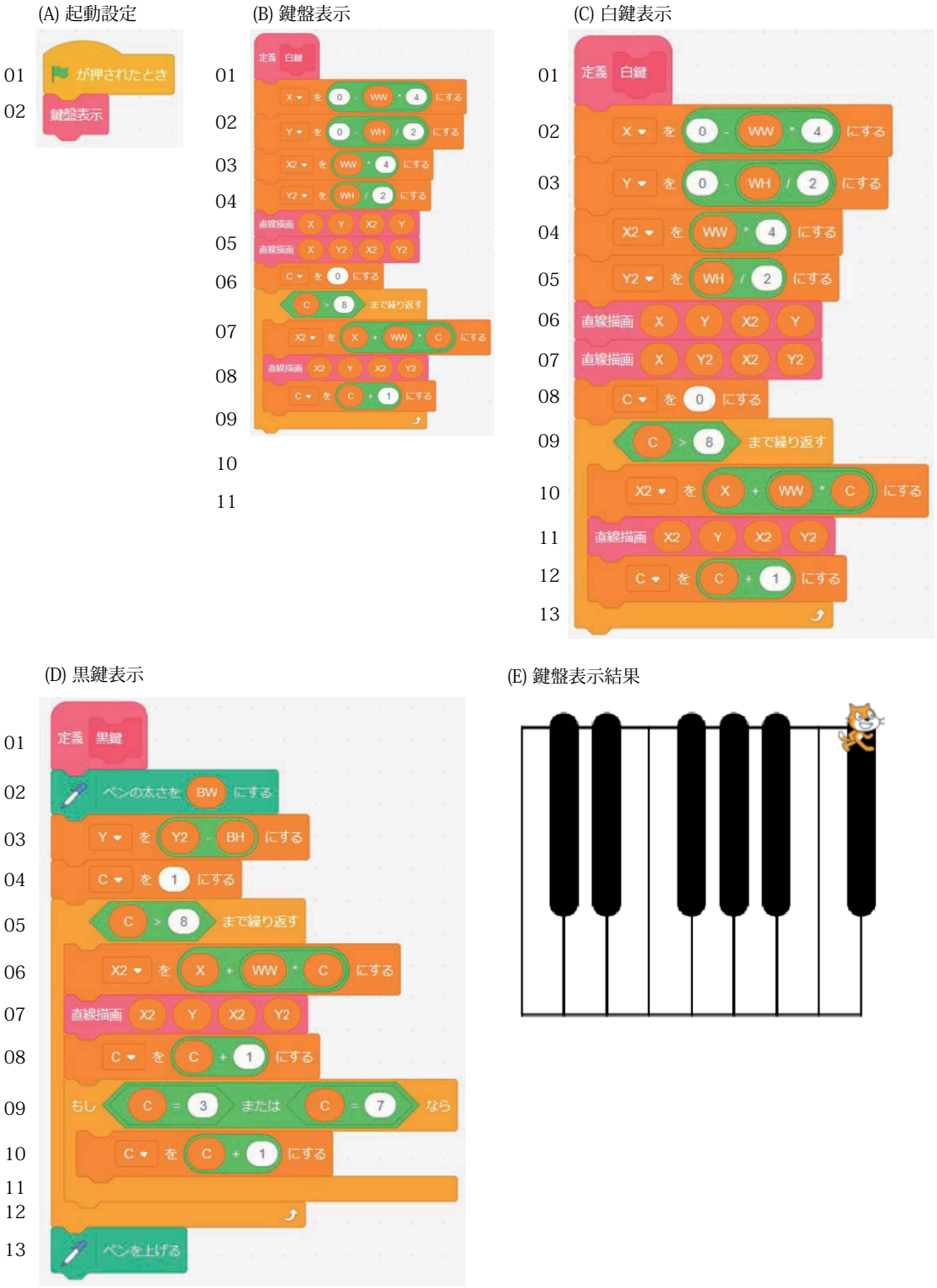


図１０ ピアノ鍵盤の表示（拡張機能「ペン」使用）(Scratch3_piano1_F10.sb3)

図１０-(A) が本アプリケーションの構成で、２段目のブロックで図１０-(B) に示す「鍵盤表示」独自定義ブロックを起動するだけである。図１０-(B) の２段目のブロックでピアノ鍵盤を押下する操作を行うデフォルトの猫キャラクターのサイズを指定している。猫キャラクターはマウスに連動して鍵盤の上を動くカーソルの役目をする。３～６段目のブロ

ックでは、白鍵と黒鍵の縦横サイズを画素単位で指定しており、変数「WW」は白鍵の幅、変数「WH」は白鍵の縦方向の長さ、変数「BW」は黒鍵の幅、変数「BH」は黒鍵の縦方向の長さである。7 ～ 10 段目のブロックでは、画面をクリアした後に、10 段目のブロックで実行する図１０-(C) に示す「白鍵」独自定義ブロックにおけるペンの色と太さを指定している。白鍵を表示した後に、11 段目のブロックで図１０-(D) に示す「黒鍵」独自定義ブロックを実行することにより、図１０-(E) に示すピアノ鍵盤の表示が完成する。

図１０-(C) に示す「白鍵」独自定義ブロックでは、まず２～７段目のブロックにて、図１０-(E) に示すピアノ鍵盤全体の下端の水平線（6 段目のブロック）と上端の水平線（7 段目のブロック）を描画している。水平線を描画するにあたり、２～５段目のブロックにより、X=-WW × 4、Y=-WH/2、X2=WW × 4、Y2=WH/2 に設定し、6,7 段目のブロックでは図５-(C) で作成した「直線描画」独自定義ブロックを実行させている。

図１０-(C) の「白鍵」独自定義ブロックの８～ 13 段目のブロックでは、変数「C」を 0 から 8 まで変えながら、ピアノ鍵盤全体の 9 本の垂直線を変数「WW」間隔で描画している。垂直線を描画するにあたり、11 段目のブロックでは同様に図５-(C) に示す「直線描画」独自定義ブロックを実行させている。

図１０-(D) に示す「黒鍵」独自定義ブロックでは、2 段目のブロックでペンの太さを変数「BW」に変更している。３～ 12 段目のブロックでは、変数「C」を 1 から 8 まで変えながら、ピアノ鍵盤の黒鍵部に太さ「BW」の幅をもつ短い垂直線を変数「WW」間隔で描画している。垂直線を描画するにあたり、7 段目のブロックでは同様に図５-(C) に示す「直線描画」独自定義ブロックを実行させている。９～ 11 段目の条件判断ブロックでは、C=3「ミ #」と C=7「シ #」の場合は白鍵になるため、垂直線の描画をスキップする処理を行っている。従って、変数「C」を 1 から 8 まで変えながら、6 本の垂直線を黒鍵として描画している。

以上のようにして、図１０-(E) に示すピアノ鍵盤を得ることができる。

４．４．２．　ピアノ鍵盤への手動演奏機能の付加（拡張機能「ペン」「音楽」使用）

電子楽器の第 1 のアプリケーションとして前節で作成したピアノ鍵盤に対して、3.1 節で述べた MIDI サウンド機能を活用して、マウス操作により鍵盤をクリックして対話形式に楽器音を鳴らす手動演奏機能を付加する。

プログラムの構成としては、図１０-(A) の起動設定の代わりに、図１１-(A) を使用し、図１１-(A) の２段目のブロックで、図１０-(A) の２段目のブロックと同様に図１０-(B) に示す「鍵盤表示」独自定義ブロックを起動するので、図１０-(B)(C)(D) は本プログラムでもそのまま流用する。

図１１-(A) の３段目のブロックで、変数「移調」に MIDI ノートナンバーへのオフセット値を設定できるようにしている。表示するピアノ鍵盤の左端の白鍵は、初期状態では、ピアノの中央の「ド (C3, MIDI ノートナンバー 60)」に設定されているが、変数「移調」により演奏音の音高を上下させることができる。図１１-(A) の 4 段目のブロックで、演奏する楽器音の楽器「ピアノ」などを指定する。本来、ピアノ鍵盤では、楽器音を鳴らす長さ（デュレーション）を、鍵盤を押してから放すまでの時間で指示するが、マウス操作では難しいため、本アプリケーションでは固定の長さだけ鳴らすようにする。そのため、5

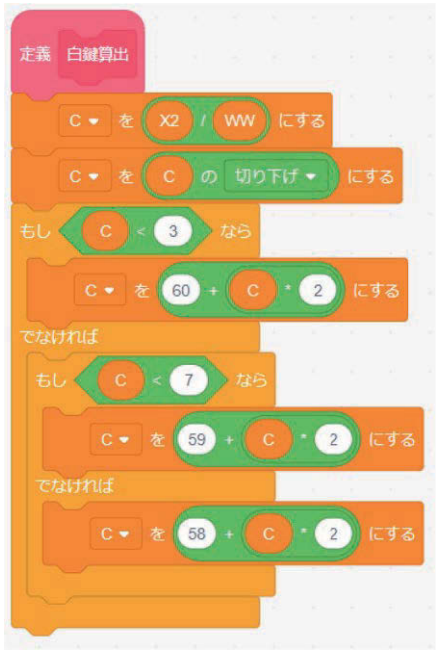
段目のブロックでテンポを指定して、鍵盤をクリックした際に、楽器音を鳴らす長さ（デュレーション）を、図１１-(G) の 17 段目のブロックにて拍単位で固定の長さ（0.5 拍=0.5 秒）で演奏するようにしている。図１１-(A) の６段目のブロックで図１１-(G) に示す「演奏制御」独自定義ブロックを起動している。

(A) 機動設定

(G) 演奏制御

(G) 演奏制御（続）

(H) 白鍵位置算出



(I) 黒鍵位置算出

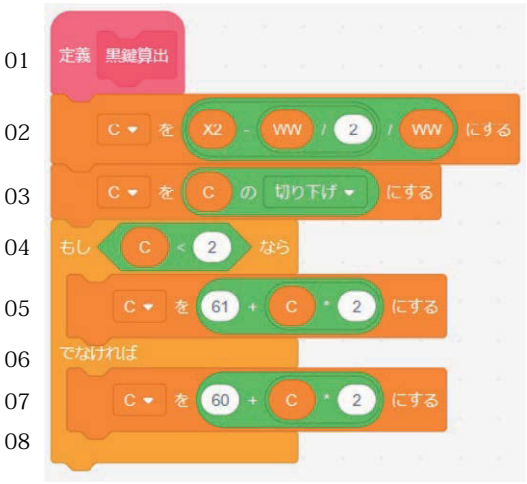


図１１ 手動演奏ピアノ（拡張機能「ペン」と「音楽」使用）(Scratch3_piano2_F11.sb3)

図１１-(G) は、２～ 15 段目のブロックでスペースキーが押されるまで無限に反復するループを構成しており、マウスがクリックされると、３～ 14 段目のブロックが実行され、「マウスの x 座標」と「マウスの y 座標」を基に、MIDI ノートナンバーを算出して変数「C」に格納し、12 段目のブロックで楽器音を鳴らしている。

まず、4 段目のブロックで「マウスの x 座標」と「マウスの y 座標」を読み取り、5 段目のブロックで、変数「X2」に X2=「マウスの x 座標」-X の値を設定し、鍵盤位置を算出するのに使用する。変数「Y」には黒鍵の下端の Y 座標が収納されているので、6 段目のブロックで、「マウスの y 座標」>Y ならば、黒鍵を指示したものと判断し、7 段目のブロックで図１１-(I) に示す「黒鍵算出」独自定義ブロックを起動する。そうでなければ、8 段目のブロックで白鍵を指示したものと判断し、9 段目のブロックで図１１-(H) に示す「白鍵算出」独自定義ブロックを起動する。

図１１-(H) の「白鍵算出」独自定義ブロックでは、2,3 段目のブロックで、白鍵の鍵盤位置を変数「X2」を「WW」で割り算することにより求め、小数点以下切り捨てにより変数「C」に格納する。左端の白鍵の MIDI ノートナンバーを 60（ピアノ中央の「ド」）とすると、C<3 の場合は、5 段目のブロックにより C=C × 2+60 にて MIDI ノートナンバーが得られる。3 ≤ C<7 の場合は、8 段目のブロックにより C=C × 2+59 にて MIDI ノートナンバーが得られ、7 ≤ C<8 の場合は、10 段目のブロックにより C=C × 2+58 にて MIDI ノートナンバーが得られる。

図１１-(I) の「黒鍵算出」独自定義ブロックでは、2,3 段目のブロックで、黒鍵の鍵盤位置を変数「X2-WW/2」を「WW」で割り算することにより求め、小数点以下切り捨てにより変数「C」に格納する。C<2 の場合は、5 段目のブロックにより C=C × 2+61 にて MIDI ノートナンバーが得られ、2 ≤ C<7 の場合は、7 段目のブロックにより C=C × 2+60 にて MIDI ノートナンバーが得られる。

このようにして、算出された MIDI ノートナンバー「C」に対して、図１１-(G) の 11 段目のブロックにて「移調」する補正を行い、12 段目のブロックで楽器音を鳴らしている。12 段目のブロックで、17 段目のブロックに示す「(C) の音符を (0.5) 拍鳴らす」を直接実行させることもできるが、後続の音をクリックして鳴らすまでに 0.5 秒以上待たされて、0.5 秒未満の高速な連打が行えなくなる。そこで、図３-(B)(D) で示したマルチスレッド手法を適用する。即ち、図１１-(G) の左下側の 16,17 ブロックに示されるように、MIDI サウンドを演奏する処理を別のスレッドに分離し、12 段目のブロックでは「メッセージ 1 を送る」を実行して、16,17 ブロックのスレッドを起動して和音演奏させるようにする。これにより、前の音の演奏が終了しなくても、後続の音を鳴らすことが可能になる。

４．４．３．　ピアノ鍵盤への自動演奏機能の付加（拡張機能「ペン」「音楽」使用）

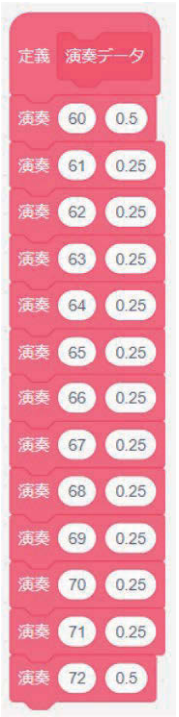
電子楽器の第２のアプリケーションとして 4.4.1 節で作成したピアノ鍵盤に対して、3.1 節で述べた MIDI サウンド機能を活用して、単音メロディーの音符データを与えることにより、鍵盤を自動的にクリックさせて自動的に楽器音を鳴らす自動演奏機能を付加する。単純な方法として、前節で作成した手動演奏機能を付加したプログラムをそのまま実行させた上で、与えられた音符データに基づいて、鍵盤位置を算出してマウスカーソル（猫キャラクターの位置）を移動させ、マウス押下メッセージを送信するプログラムを作成し

て、マルチスレッドで実行する方法が考えられる。しかし、現状の「Scratch3.0」ではマウス押下メッセージを送信する機能が実装されていない。そこで、本節では、与えられた音符データに基づいて、鍵盤位置を算出してマウスカーソルを移動させて、指定されたMIDI ノートナンバーの楽器音を鳴らすプログラムを改めて作成する方法をとる。

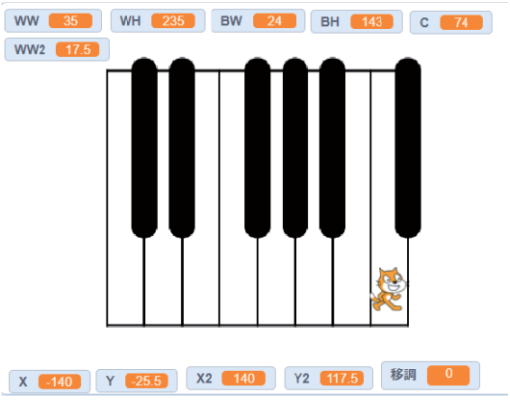
そのため、プログラムの構成としては、図１１のプログラムは使用せず、前節と同様に、図１０のプログラムのみ使用する。図１０-(A) の起動設定の代わりに、図１２-(A) を使用し、図１２-(A) の２段目のブロックで、図１０-(A) の２段目のブロックと同様に図１０-(B) に示す「鍵盤表示」独自定義ブロックを起動するので、図１０-(B)(C)(D) は本プログラムでもそのまま流用する。

(A) 起動設定

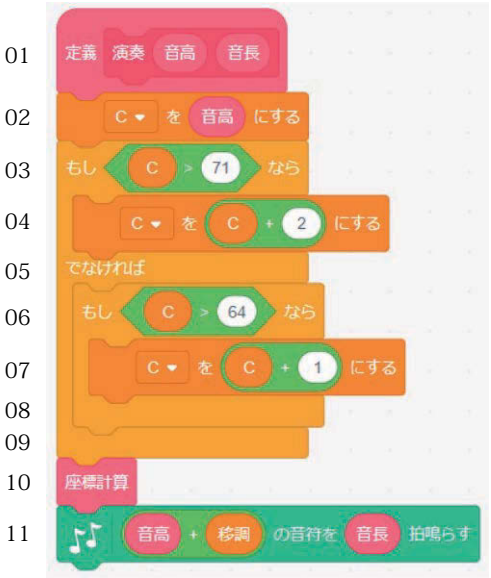
(J) 演奏データ



(K) 実行結果



(L) 自動演奏



(M) 座標計算

図１２ 自動演奏ピアノ（拡張機能「ペン」と「音楽」使用）(Scratch3_piano3_F12.sb3)

図１２-(A) は図１１-(A) の６段目のブロックで「演奏制御」独自定義ブロックの代わりに図１２-(J) に示す「演奏データ」独自定義ブロックを起動している。図１２-(A) の３～５段目のブロックで、図１１-(A) の３～５段目のブロックと同様に、変数「移調」にMIDI ノートナンバーへのオフセット値を設定し、演奏する楽器音の楽器「ピアノ」などを指定し、演奏するテンポを指定する。５段目のブロックでテンポを指定することにより、「演奏データ」独自定義ブロックで、各音符を自動演奏させる際に、楽器音を鳴らす長さ（デュレーション）を、図１２-(J) の各「演奏」独自定義ブロックにて拍単位で指定可能になる。そして図１２-(A) の６段目のブロックで図１２-(J) に示す「演奏データ」独自定義ブロックを起動している。

図１２-(J) の「演奏データ」独自定義ブロックは、自動演奏させるメロディーの各音符の MIDI ノートナンバーと拍単位の音長（テンポ =60 の場合は、拍単位 = 秒単位）をパラメータとして与えた「演奏」独自定義ブロックを上から順に並べた構成になっている。図１２-(J) のプログラム例では 12 半音階を定義したものである。

図１２-(L) の「演奏」独自定義ブロックでは、引数として与えられた「音高」パラメータと「音長」パラメータを用いて、11 段目のブロックで演奏させるのが主であるが、併せて、10 段目のブロックで図１２-(M) に示す「座標計算」独自定義ブロックを起動してピアノ鍵盤上の猫キャラクターの位置を「音高」に対応する鍵盤の位置に移動させる制御を行っている。図１２-(M) に示す「座標計算」独自定義ブロックを起動するにあたり、図１２-(L) の２から９段目のブロックで、「音高」パラメータを基に、鍵盤位置を指示する変数「C」を算出する。２段目のブロックで C=「音高」として、3,4 段目のブロックで C>71 ならば、C=C+2 に補正し、6,7 段目のブロックで 64<C ≤ 71 ならば、C=C+1 に補正する。

図１２-(M) の「座標計算」独自定義ブロックでは、変数「X」には白鍵の左端の X 座標が収納されている。そこで、２段目のブロックで WW2=WW/2 として、３段目のブロックでキャラクターの X 座標を、(C-60) × WW2+X+WW2 で与える。一方、変数「Y」には黒鍵の下端の Y 座標が収納されている。そこで、４段目のブロックで C が奇数（C を 2 で割った余りが 1）の場合は黒鍵であるから、５段目のブロックでキャラクターの Y 座標を、Y+BH/3 で与え、C が偶数の場合は白鍵であるから、７段目のブロックでキャラクターの Y 座標を、Y-WH/4 で与える。

５．おわりに

本稿は、2020 年度よりコロナ禍で開始されたオンデマンド形式のオンライン授業において、本学の情報系科目に導入するために、新規に制作した「Scratch」プログラミング演習の教材を基に、音響分野の例題を中心に整理したものである。

情報系科目において学生が主体的に授業に取り組んで、理解を深めるにはプログラミング演習が不可欠である。既に本学・情報表現学科の演習授業では、Windows ノートパソコンを持参してもらい、プログラミング演習を組み込んでいた。しかし、初心者向けプログラミング例題は、学生には退屈で無味乾燥な題材が多く、学生間でパソコン操作スキルの差が大きく、一斉授業で進めることに幾つかの問題を抱えていた。

また、筆者は情報系教養科目も担当しているが、本学・情報表現学科以外の学生も数多く聴講されるため大教室で授業が行われ、ノートパソコン持参のプログラミング演習を組

み込むことは困難であった。そのため、情報系教養科目では講義主体で進めざるを得ず、どうしても受け身の授業になってしまうという問題を抱えていた。

これに対して、「Scratch」の開発環境は、インターネットに接続された Web ブラウザが稼働する Windows/Mac パソコンがあれば、誰でもインストール不要で無償で利用できる。幸い、オンデマンド形式授業の導入に伴い、全ての学生がインターネットに接続された IT 端末で受講する環境が整った。「Scratch」は、キャラクターと対話しながら進めるビジュアル・プログラミング環境であるため、小学校でも採用されているくらいに、プログラミング初心者にもハードルが低い。更に、初心者向けのプログラミング例題として、本学の多くの学生が関心を示す音響分野の比較的実用的な題材を提供できる。

これらのことから、「Scratch」を用いたビジュアル・プログラミング演習は、大学生が興味をもちそうな例題を作成・提供すれば、各学生のパソコンスキルやプログラミング経験に応じたペースで独習でき、オンデマンド形式授業に向いている。これまで導入が困難であった、大教室で行われる情報系教養科目においても、プログラミング演習を組み込むことが可能である。実際に、2020 年度秋学期から 2021 年度春学期のオンデマンド授業に組み込んでみた。40% 以上の学生は、高等学校までの授業や本学の他の演習科目で既に「Scratch」を経験済であった。それでも、過去の教育と被ることなく、改めて「Scratch」プログラミングを一から学ぶことができたと好評であった。

以上、本稿は Scratch の基本操作に習熟されている方であれば、音響プログミングを学習するためのテキストとして利用できる。また、Scratch 初心者向けには。下記サイトにて入門教材 [Scratch_manual_v3.pdf] も用意している。本稿で紹介したプログラミング例題の全てのデータおよび、その基になった、筆者が担当している本学・情報系科目向けのプログラミング演習教材（テキストとプログラミング例題の全データ）については、以下サイトにて一式を公開しているので、教育・研究・その他にご活用ください。

［独自制作 Scratch 演習教材の公開サイト (2021.9 現在、2021 年 9 月 21 日版を公開)］

以下サイトにて、本稿に掲載されている全ての例題のプログラムデータが収録されているファイル [Scratch_kiyou.zip] をダウンロードできます（各図に明記されている ” Scratch3_prime_F1.sb3” などのファイル名で各データが収録されています）。

また、基本操作と例題 [Scratch_manual_v3.pdf] と高度なプログラミング事例 [Scratch_advance_v3.pdf] の 2 巻からなる独自制作 Scratch 演習教材および掲載プログラムの全データが収録されているファイル [Scratch_example.zip] も以下サイトにてダウンロードできます。

筆者のホームページ：http://www.bekkoame.ne.jp/~modegi/

または　https://sites.google.com/view/hptoshiomodegi

筆者の連絡先：modegi@bekkoame.ne.jp

引用文献

1) 松林弘治『子どもを億万長者にしたければプログラミングの基礎を教えなさい』、（株）KADOKAWA、初版第 1 刷、February 20.2015, 136-148 頁 .

2) 石嶋洋平著、安藤昇監修『子どもの才能を引き出す最高の学び　プログラミング教育』、（株）あさ出版、第 1 刷 July 8.2018, 78-87 頁 .

3) 石井英男 , 吉岡直人 , 赤石昭宏 , 相川いずみ , 森谷健一『小学生からはじめるプログラミングの本』、, 日経パソコン編、日経 BP、電子書籍初版 February 12.2021, 43-98 頁 .
4) Scratch Web 版：https://scratch.mit.edu/、Scratch ダウンロード版：https://scratch.mit.edu/download/ (2021 年 9 月アクセス).
5) 鈴木喬裕『はじめての人も、挫折した人も「Scratch で今からはじめるプログラミング」Scratch だからスラスラわかる』、日経 BP、電子書籍初版 March 29.2021, 23-56 頁 .
6) Ali Moghiseh and Andreas Jablonski, “ToolIP, Visual Programming for Image Processing”, http://www.ipol.im/event/2012_imlib/slides/toolip.pdf (2021 年 9 月アクセス).
7) Max/Msp：https://akihikomatsumoto.com/maxmsp/max.html (2021 年 9 月アクセス).
8) 山内卓哉『「音とコンピュータ」そのプログラミング、サウンドデザインそしてメディアアート』、（株）インプレス R&D、電子書籍 Ver.1.2 版 December 7.2016, 7-47 頁 .
9) Unity のリアルタイム開発プラットフォーム | 3D/2D, VR/AR のエンジン：https://unity.com/ja　(2021 年 9 月アクセス).