社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－CE－82（12）
2005／12／10

ドリトルと「情報教育の音楽化」

辰己 丈夫 (東京農工大学)　兼宗 進 (一橋大学)　久野 靖 (筑波大学大学院)

概 要

筆者の一人である辰己は 2001 年の CE 研究会で「情報教育の音楽化」という題名で発表を行ない、情報教育の中で楽譜処理を利用できることを指摘した。一方、兼宗・久野は、タートルグラフィクスを土台にオブジェクト指向を取り入れた教育用プログラミング言語「ドリトル」を開発していた。ドリトルには以前より音楽演奏機能が追加されていたが、本格的な教育への応用は行っていなかった。これを利用したプログラミング教育を実践してみたいという辰己とドリトルを開発している兼宗・久野の 3 名が、ドリトルの演奏機能に大幅な演奏機能の拡充を行なった。本発表では、「情報教育の音楽化」の概要、ドリトルにどのような拡充が行なわれたのかなどについて説明をするとともに、教材例・授業例の提示などを行なう。

Dolittle and Musical methodlogy in Programming Education

TATSUMI Takeo (Tokyo Univ. of Agriculture and Technology)
KANEMUNE Susumu (Hitotsubashi University)
KUNO Yasushi (Graduate School of Systems Management, Tsukuba Univ.)

abstract

In 2001, Tatsumi pointed out that the musical methodology is useful in computer literacy education. Kanemune and Kuno developed “Dolittle” which is object oriented turtle graphics programming language for beginner. In 2005 September, Tatsumi wrote new specification in playing musical scores with Dolittle. Kanemune add new command and messages to Dolittle. In this paper, we describe our works and report about elementary class exercises in this November.

1　はじめに

本論の冒頭で、「ドリトル」と「情報教育の音楽化」の二つのプロジェクトがどのように進んできたかについて説明する。

1.1　情報教育の音楽化

1997 年頃から、辰己は「音楽教育を深化させることで情報教育に寄与させることができる」との意見を述べ、1998 年の情報処理学会「夏のプログラミングシンポジウム」において、楽譜処理とスクリプト教育の類似性について指摘 [1] を行なった。その後、「プログラミング教育に役立つ楽譜処理ソフトウェアを作成し、プログラミング教育を実践すべきである」との意見を何ヶ所かで述べていた。

この内容を整理しながら、「情報教育の音楽化」[2] という研究テーマ名をつけ、2000 年〜2002 年頃に何回かの研究資金公募に出願したが、採択されることはなかった。MIDI を利用するソフトウェア開発技術を持っていなかった辰己は、構想を持ったが、開発者に出会えなかったまま、2003 年頃から研究を停滞させていた。

1.2　ドリトル

Logo などの教育用言語を研究してきた兼宗は、教育用言語にオブジェクト指向を取り入れることの必要性 [3] を感じ、2000 年にオブジェクト指向を簡単に学べる初心者用言語「ドリトル」[1]を開発 [4] した。Logo の流れを組むタートルグラフィックス言語の流れを組むように見えるが、Logo のような手続き的な考え方を用いず、プログラムはタートルオブジェクトに対する操作で作成される。

兼宗は、久野と共同でドリトルを完成させ、ドリトルを用いた様々な教育実践 [5, 6] が始まった。

1.3　ドリトルへの演奏機能の追加

その後、ドリトルはタートルグラフィックスに加えて、ロボットなどの外部機器操作 [7] や、ネットワーク [8] でオブジェクトを交換できるようにする

[1] ドリトルの詳細は、http://kanemune.cc.hit-u.ac.jp/dolittle/ に詳しい。

など、教育の中でさまざまな形でオブジェクトを扱う試みを進めて来た。2004 年 6 月には、MIDI のデバイスをオブジェクトとして扱うための拡張を行った。プログラムから音楽を演奏できるようになったことで、生徒の作品表現の幅が大きく広がった [9]。しかし、MIDI の機能面を重視した設計を行ったため、後述するように音楽の記述方法に改善の余地が存在した。

1.4 ドリトルへの演奏機能の充実

1998 年から「情報教育の音楽化」[2]に関わってきた辰己は、音や楽器、楽譜などをオブジェクトと捉えることで、「ドリトルを利用する音楽プログラミング教育が可能である」と気がつき、ドリトルに付加された演奏機能を利用して教材作成を試みた。

その過程で、授業実践に十分な楽譜記述能力や演奏制御機能を持たないことが明らかになった。そこで、辰己は 2005 年 9 月にドリトルの演奏機能を充実させるための仕様提案を行なった。その内容にしたがって、兼宗はドリトルの演奏機能を拡充し、2005 年 11 月に V1.25 を発表した。

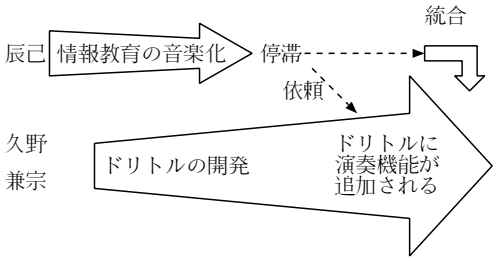

図 1: 開発の時間的変遷

以上が、「情報教育の音楽化」と「ドリトル」の関係である。

2 情報教育にプログラミングを取り入れることの是非

初等中等教育における情報教育の中に、プログラミングを取り入れることの是非については、情報教育の中心がアプリケーションの操作方法の習熟を目的とする「情報リテラシー教育」に移行した 1990 年代後半頃から、いくつかの議論があった。

最近では、2005 年 10 月の情報処理学会情報教育委員会シンポジウム「高校教科「情報」の現状と将来」[10] で大きなテーマとして取り上げられ、大学教員と高等学校の教員らが熱心な議論を行なった。

筆者らは、初等中等教育にプログラミングを取り入れるべきであるとの立場で、情報教育の目標に対する深化、プログラミングを取り入れる理由の考察、プログラミング教育用の適切な言語設計、言語処理系の開発、授業方法論の提案、授業案の提示、いくつかの学校における授業実践など、これまでにもさまざまな研究・実践・開発・議論を行なってきた。

2.1 情報教育の目標

1997 年、情報化の進展に対応した初等中等教育における情報教育の推進等に関する調査研究協力者会議が提出した「体系的な情報教育の実施に向けて」[11] によれば、情報教育の目標は「情報の活用の実践力」「情報の科学的な理解」「情報社会に参画する態度」の 3 つであるとされている。

我々は、「コンピュータの動作を理解する」ことが、これら 3 つの目標を達成するにあたり、非常に重要な役割を果たすと分析している。ここでいう「コンピュータの動作」とは、ハードウェアの構成・動作原理、ソフトウェアの仕組み、プロトコルなどの存在意義などである。

2.2 プログラミング学習と情報教育の目標

情報教育の 3 つの目標に関わる「コンピュータの動作」を学ぶに当たり、一見遠回りに見えるかもしれないが、プログラミングを体験することが重要であると、我々は考えている。

例えば、バグや不正な使用により情報社会を混乱させることがないように、コンピュータを適切に利用することで、「情報社会に参画する態度」を適切に身に付けようとするならば、

> 人間は、同じ動作を繰り返して行なうと疲れてしまったり、間違えてしまったりするが、ハードウェアは常に期待されたように動作する。常に期待されたように動作するならば、その動作手順を文書にして表すことができる。それゆえに、操作手順も文書にして表すことができる。
>
> 人間は、与えられた手順書に不備があっても、自分で考えて適当に補ったり、誤りを修正したりするが、コンピュータにはもともとはそのような機能が付加されていない。現在も、コン

[2]「情報教育の音楽化」の研究は、2005 年度は文部科学省科学研究費補助金基盤研究 (B)(2) （課題番号 16330178）（研究代表者：船越俊介（神戸大学・発達科学部・教授））の研究の一環として行なわれている。

ピュータが関わる事故の多くが、人間が作成した手順書の誤りが原因であるのだから、そのような状況を適切に予想するには、「手順書に誤りがあって動作がおかしくなる」という体験をしておくことも必要である。

そこで、実際に自らプログラムを作成し、その通りに機械が動作するという状況を体験するべきである。

というストーリーが成立すると考えている。

2.3 反対意見

初等中等教育における情報教育にプログラミングを取り入れることに対する反対意見についても触れておく。

2.3.1 他の方法でもよい (必要ではない)

例えば、新聞社などのサイトを利用し、プログラムのバグや不正な使用が原因で起こった事件について調査を行ない、それをプレゼンテーションソフトなどを利用して発表するという方法も、「情報社会に参画する態度」を身に付けるひとつの方法である。

同様に、プログラミングを用いずとも、情報教育の目標を達成させることが可能である。

しかし、プログラミングを利用する方法の場合は、実体験を伴うことで、学習者に深い印象を残すことが可能であり、「情報社会に参画する態度」を効果的に育成することができるといえる。

学ばなくてもいいことを学ぶのは一見遠回りに見えるが、より多くのことを短い時間で学ぶことが可能であるともいえる。小説を読むために必要な単語や文法だけを覚えることよりも、必要でない単語や文法まで覚えておく方が学習内容の俯瞰が可能になるのと同じことである。あるいは、旅行のとき、地図がなくてもたどり着ける場所は、地図があればより早くたどり着けるといえる。

2.3.2 実習環境がない

1980 年代後半から、PC 利用の中心がワープロソフトなどのアプリケーションソフト利用に移るにつれて、パソコンにおける言語学習の環境が急速に減少してしまった。特に、多くのパソコンに装備されていた ROM BASIC が消滅してしまったため、手軽な実習環境を構築できないという問題が生じてしまった。

この問題を解決するには、

- 特別な準備をしないでもプログラミング教育に利用できるものを探す
- 授業に最適な実習環境を作成する

のいずれかの方法が有効である。

まず、JavaScript に代表されるブラウザ組み込み言語などは、前者の候補としてふさわしい。

一方、兼宗・久野らが開発した「ドリトル」や、長が開発した「Tonyu」[12] などは、初心者用のプログラミング言語であり、かつプログラミング環境でもある。後者の候補として十分な資格を持つ。

2.3.3 初等中等教育には内容が難し過ぎる

「プログラミングは難解であり、初等中等教育で取り入れるには適切ではない」という意見がある。しかし、既に多くの初等中等教育における実践例がある。授業方法の工夫、教材の工夫さえ十分に達成できるならば、小学生でも十分にプログラミングを理解することができたという報告がなされている。

2.3.4 実技は教えにくい

「プログラミングは実技的な側面があるので教えにくい」という意見をいう人がいる。「実技的な側面があること」を否定はしないが、体育や音楽のように実技を教えることは可能である。また、プログラミングのすべてが実技で構成されるわけではない。プログラミングには実技の他にも様々内容が含まれている。英語などの語学と共通する要素も多く含まれている。

2.3.5 担当教員に十分な知識がない

教科「情報」の教員として 15 日間の研修で促成された教員の中でも、もともとプログラミングなどに全く興味がなかった人にとっては、安心して授業を行なう知識がない。このことは事実である。

しかし、例えば数学や英語などの教科において、担当教員が「完璧な知識」を持っていなかったとしても、教科書や課題を生徒たちに取り組んでもらうことによって、授業が成立し得るのも事実である。生徒の中には、教員よりも進んだ能力や知識を得る機会となることもある。

さらに、情報処理学会を始めとするいくつかの研究会や、いくつかの web サイト、「ドリトル」「Tonyu」などのサイトをみて、教員が自らプログラミングの知識を深めることも可能となった。インターネットの普及と発達は、教員の自己研修にも変化を与えている。

2.4　初等中等教育にプログラミングの体験を

前項で指摘したように、初等中等教育におけるプログラミング教育への反対意見は、適切な条件、関係者の努力をもって反論することが可能である。もちろん、反対意見にも十分な論拠がある。特に難解な言語を選んだり、特に珍しい言語を選んだり、担当教員があまりにも未熟過ぎたりしている場合には、今まで取り上げた反対意見の方に説得力がある。

我々は、授業成立の「良い例」を積み上げ、さらなる実践を行なう必要があるといえる。

3　初等中等教育に必要なプログラミングの体験の要素

初等中等教育においてプログラミングを扱う際に、一体どんな要素が必要となるだろうか。

初等中等教育におけるプログラミングの体験は、専門的な技術者の育成を目標とするものであってはならない。「プログラミングを学ぶ体験」は、ほぼすべての学習者にとって、コンピュータやネットワークの動作原理を、より印象深く、より短い時間で（効率的に）学ぶことに寄与する。その目的を達成するために必要な構成要素について議論する。

3.1　「作成→動作確認→修正」の手順化された作業

コンピュータプログラムは、機械によって作られるものではなく、人間が作成するものであるということを実感させるために、「人間がプログラムを実際に書いて、それを実行させては修正をし、また実行させては修正をする」という手順化された作業を実際に経験するとよい。

3.2　テストとデバグ

上に述べた「手順化された作業」の合間に、動かないプログラムを作成してしまったり、プログラムが突然動かなくなってしまったり、予期しない動作をしてしまうという体験をすることになる。そこで行なわれる作業はプログラムの動作テストとデバグである。

3.3　動作エラーに対する取り扱い

もし、システムプログラム開発や、高価な実験機器のプログラミングを扱うとするならば、エラーは大きな被害に直接的に結び付く。

一方、タートルグラフィックスや音楽演奏などのプログラムにエラーがあっても、それは「画面の書き間違い」「演奏の間違い」という状況にのみ現れる。そのとき、教室に笑い声がすることはあっても、エラーを作成してしまった人を責める雰囲気は生じない。だからこそプログラムにエラーがあるかどうかを気軽に試すことができる。この性質は、初心者用プログラミング環境に必須のものであるといえる。

3.4　コンパイラ・インタープリタによる解釈の幅

例えばC言語では、各命令をセミコロンで終了することになっている。これをピリオドで終了させることはできない。この部分を書き間違えてしまうとプログラムを正確に動作させることができなくなってしまう。

このように解釈の幅が非常に狭いプログラミング言語は多い。一方、コンピュータの仕組みをよく理解できていない学習者は「なぜセミコロンにする必要があるのか」も理解できない。そこで実行環境系は、これらの小さなミスを吸収する解釈の幅が必要である。

ただし、あまりに多様なプログラム解釈を行なえるようにしてしまうと、「コンピュータは文字通り（杓子定規に）しか動かない」という性質の体験をせずに体験が終了してしまう可能性があることは注意を要する。

3.5　プログラムの構成要素の体験

プログラム作成に当たっては、「データ構造の析出」「アルゴリズムの意識化」「連接」「繰返しと脱出」「制御」「部品化と再帰」「同期」などの概念が必要となる。ただし、これらの概念のすべてをプログラミングの授業にとり入れることは容易ではない。

初等中等教育におけるプログラミングの授業では、上記の概念の内いくつかが取り入れられれば十分であろう。そこで興味・関心を持った一部の児童・生徒が学習を進めればよい。

4　「ドリトル」（旧版）

本章では、初心者のためのオブジェクト指向プログラミング言語「ドリトル」の旧版の仕様の一部を簡単に述べる。

4.1　タートルオブジェクト

ドリトルは、日本語の漢字などを利用することが可能である。例えば、

```
カメ太 = タートル！作る。
カメ太！100 歩 歩く。
```

のように、プログラムの基本系は以下のようなものである。

1. オブジェクトを生成し、
2. オブジェクトにメッセージを送る

4.2 メロディオブジェクト

タートルオブジェクトの他に、さまざまなオブジェクトが最初から用意されている。例えば、メロディオブジェクトを利用すると、次のようなことが可能となる。

```
チューリップ = メロディ！ 作る。
チューリップ ！ 『ドレミードレミー』追加。
僕の楽器 = 楽器！『ピアノ』 作る。
僕の楽器 ！ （チューリップ）設定。
楽器！演奏。
```

これは、メロディオブジェクトに「チューリップ」という名前をつけ、そこに旋律を追加し、楽器を指定して演奏させるものである。

プログラムの「連接」概念は、ここで取り上げられているといっても良い。

4.3 プログラムのブロック化

プログラムのある部分をまとめて一つの繰返し単位などに利用する「ブロック化」ができる。利用例は次項で取り上げる。

4.4 繰返しの利用

繰返しを利用してオブジェクトにメッセージを送ることも可能である。以下の例では、「楽譜！『ドレミー』追加。」というブロック化されたプログラムもまたオブジェクトであり、そのオブジェクトへ「2 繰り返す」というメッセージを送っていることがわかる。

```
「楽譜 ！『ドレミー』追加。」!2 繰り返す。
```

プログラムの「繰返し」概念は、ここで取り上げられているといっても良い。

4.5 制御構造

条件判定をして動作を変化させる「制御」は、次のように書く。

```
x = 1。
「
楽譜！『ドレミードレミー』 追加。
楽譜！『ソミレドレミ』 追加。
「 x == 1 」！ なら 「楽譜！『レー』 追加」
  そうでなければ「楽譜！『ドー』追加」実行。
x = 2。
」 ！ 2 繰り返す。
```

このように、メロディーが持つパターンをプログラムを使って構成できる。

4.6 ブロックの利用

次のようにすると、8つの異なる音が演奏される。

```
「
楽譜 ！ （音階 ！ （random(6)）見る）追加。
」 ！ 8 繰り返す。
```

すなわち、「random(6) よりも「! 8 繰り返す。」の方が先に評価される。同じ音が 8 つ続くようにしたい場合は、

```
次音 ！ （音階 ！ （random(6)）見る）追加。
「
楽譜 ！ 次音 追加。
」 ！ 8 繰り返す。
```

とする。この場合は、「次音」が先に確定しているので、同じ音が 8 つ続く。

これは、ブロックを利用することで内容評価のタイミングを制御できる[3] ことを示している。

5 旧版の問題点と改良

ドリトル旧版には、実際の音楽教育と情報教育の橋渡しをするにあたり、いくつかの不具合があった。ここでは、その不具合と、V1.25 までに変更された改良点について述べる。

5.1 「メロディ」の定義

音楽の基本構造は、これらの「メロディ」を組み合わせて作られている。ここでいうメロディは「いくつかの音（音程と音長）が並んだもの」である。

とても短いメロディ
　動機 (モチーフ)
短いメロディ
　楽句/楽節 (フレーズ)
　主題/主旋律 (テーゼ)
長いメロディ
　楽譜/総譜 (スコア) に近いもの

一方、旧版では

```
チューリップ = メロディ！ 作る。
```

のように「メロディ」を楽譜という意味で使っていた。そのため、音楽に知識を持つ教員にとっては違和感がある言葉使いとなってしまっていた。

先ほどの繰返しや制御構造を利用した楽譜記述の場合は、「モチーフ」あるいは「フレーズ」を繰り返すことで「スコア」を作り出すように言語仕様が作られるべきであった。そこで新版では「譜面」オブジェクトが自動的に生成されるようになった。これは音文字か、通常の文字列で作られるオブジェクト

[3] この点が教育上どのような効果をもたらすのかについては、今後の検討課題である。

であり、譜面に基準音程を設定することで、転調を簡単に実現することができる。また、メロディという言葉を使わずに直接譜面に文字を設定することもできるので、より音楽の専門家に馴染み易い記述をすることができる。

さらに、楽器名を設定せずに譜面に対して演奏メッセージを送ることもできる。

5.2 単独の楽器とオーケストラ・バンド

旧版では、

1. 楽譜はメロディとして作る。
2. 「楽器」という『もの』に「楽譜」を設定する。
3. 楽器に演奏メッセージを送る。

という手順であった。

音楽の観点でいえば、ソロコンサートでないかぎり楽器は合奏されるものである。合奏の単位は、クラシックならオーケストラ、ポピューラーミュージックならバンドである。また、全体のメロディーを記した「総譜（バンドスコア）」と、個々の楽器が演奏するメロディを記した「パート譜」の 2 種類があり、特にクラシックでは指揮者は総譜を見て指揮を行ない、演奏家はパート譜を見て演奏をする。

ドリトル旧版は、このような状況に合致する用語でなかったため、この部分を変更する必要があった。そこで新版では「バンド」オブジェクトが作られた。バンドには楽器や譜面を登録して演奏することができる。またデフォルトの別名として「オーケストラ」「オケ」「指揮者」「演奏者」「スコア」「総譜」が用意された。

また、旧版では楽器オブジェクトはプロトタイプでありながら、それに演奏メッセージを送ることができたが、新版ではより自然な形に近付けるために、楽器のプロトタイプオブジェクトを元に生成された個別の楽器オブジェクトに演奏を指示できる。

具体的には、旧版では

```
僕の楽器 = 楽器！『ピアノ』 作る。
楽器！演奏。
```

としていたが、新版ではこれは許されず、

```
僕の楽器 = 楽器！『ピアノ』 作る。
僕の楽器！演奏。
```

とすることになる。また、楽器単独での演奏は

```
ピアノ１！演奏。
```

という記述で演奏が可能になった。

5.3 演奏速度の調整

バンドオブジェクトに対して演奏速度の調整を行なうことができるようになった。これで、ゆっくりした曲、早い曲を書くことができるようになっただけでなく、演奏中にテンポを落したり上げたりすることもできるようになった。

5.4 楽器の音量

新版では個々の楽器の役割が現実の演奏に近付いた。そこで楽器オブジェクトに音量をメッセージとして設定できるようになった。具体的には、

```
ピアノ１！ 40 音量。
```

とする。音量は 0 から 100 までの数値である。

5.5 メロディ記述（音程）

旧版ではオクターブ上げる／下げるといったメロディ記述が仕様に入っていなかったが、新版では「^」と「__」を利用することで、柔軟に記述ができるようになった。

5.6 メロディ記述（音長）

旧版では、一つの音の長さは固定されており、それを伸ばす場合は『ー』を後置修飾して「音を伸ばす」という意味で利用していた。

新版では、一つの音に数を後置記述をして音長の設定ができるようになった。また、&による音長変化を認めることができるようになった。例えば、

```
ソ２ラ４＆８
```

は「ソの 2 分音符と、ラの付点 4 分音符で」あり、

```
ド１＆１＆１＆１
```

は 4 小節連続する『ド』である。

5.7 演奏終了を『待つ』

プログラムの構成の基本要素として連接、繰返し、制御、部品化、再帰、同期を挙げるならば、旧版において再帰と同期以外は取り入れられているといってよい。しかし、音楽演奏において「待つ」の概念がないと、演奏と表示の同期をとることが不可能である。

そこで新版には「演奏終了を待つ」という記述が可能になった。これで、再帰を除くプログラムの基本概念のすべてが、ドリトルにも取り入れられているといえるようになったのである。

演奏終了を待つことが可能になったので、例えばゲームプログラムを作成してファンファーレを鳴らす場合でも、演奏終了を待ってから次の面に進むことができるようになった。また、文字列表示と組み

合わせることで、ドリトルをを使って簡単なカラオケシステムを記述できるようになった。

6 実験授業

次の実験授業[4]を行なった。

参加者 千葉私立おゆみ野南小学校と、近隣の小学校から、5 年生と 6 年生の児童 18 人

協力者 佐藤和浩（おゆみ野南小学校教諭)、西ヶ谷浩史（藤枝市立青島中学校教諭）、青木浩幸（東京学芸大学大学院学生)、紅林秀治（静岡大学)、原久太郎（NPO ゆーらっぷ）

表 1 に、実験授業のカリキュラムを示す。

表 1: **実験授業のカリキュラム**

回	日付	テーマ	音楽演奏機能の利用
1	7/27	入門	(なし)
2	7/28	作品製作	一部の児童が旧版を使用した
3	11/5	ロボット制御	(なし)
4	11/6	音楽演奏	新版で音楽演奏機能を説明した

7 月に行われた 1,2 日目の授業では、「僕も私もゲーム作家」というタイトルを掲げ、タートルグラフィックスを用いたプログラミングを扱った。参加した児童はオブジェクトや繰り返しなどの基本的な概念を学んだ後、ゲームなどの作品を作成した。音楽機能を扱う予定はなかったが、児童からゲームの BGM として音楽を使いたいという要望があったため、旧版の音楽機能を説明した。

6.1 具体例 1

旧版では、短いメロディを演奏するために、必ず 5 行以上のプログラムを書く必要があったため、音楽演奏を自分の作品に取り入れた生徒は数人にとどまった。

```
チューリップ＝メロディ！作る。
チューリップ！『ドレミードレミー』追加。
僕の楽器＝楽器！『ピアノ』作る。
僕の楽器！(チューリップ) 設定。
楽器！演奏。
```

11 月に行われた 4 日目の授業では、午前中に新版の音楽機能を説明し、午後にゲームなどのプログラム作品で音楽を利用した。新版では「楽器を作って演奏」の考え方で

```
楽器！『ピアノ』作る『ドレミードレミー』追加 演奏。
```

とすることが可能になった。また「譜面を作って演奏」の考え方ならば、

```
譜面！作る『ドレミードレミーソミレドレミレー』
    追加 演奏。
```

と簡単なメロディを 1 行のプログラムで演奏することが可能になり、全員が自分のプログラム作品に音楽演奏を組み込むことができた。

6.2 具体例 2

まず、次のプログラムを利用して「繰返し」の概念の確認を行なった。

```
チューリップ=メロディ!作る。
「チューリップ!『ドレミー』追加。」!2 回繰り返す。
チューリップ!『ソミレドレミレー』追加。
「チューリップ!『ドレミー』追加。」!2 回繰り返す。
チューリップ!『ソミレドレミドー』追加。
チューリップ!演奏。
```

次に、場合分けによる分岐を取り扱った。

```
チューリップ=メロディ!作る。
「|x|
チューリップ!『ドレミードレミー』追加。
    !2 回繰り返す。
チューリップ!『ソミレドレミ』追加。
「x==1」!なら
    「チューリップ!『レー』追加」
そうでなければ
    「チューリップ!『ドー』追加」実行。
」!2 回繰り返す。
チューリップ!演奏。
```

この例を体験することで、プログラムの部品化と繰返しなどの処理について体験することができた。

6.3 具体例 3

次に、楽器名の設定を行なった。

```
僕の楽器=楽器!『クラリネット』作る。
チューリップ!(僕の楽器) 設定。
チューリップ!演奏。
```

このようにして楽器設定を体験した後に、

```
僕の楽器=楽器!  124 作る。
チューリップ!(僕の楽器) 設定。
チューリップ!演奏。
```

および、

```
僕の楽器=楽器!(random(128)-1) 作る。
チューリップ!(僕の楽器) 設定。
チューリップ!演奏。
```

を利用したところ、児童達は夢中になって何回も実行を繰り返した。「譜面と楽器の独立性の実感」「乱数の性質」が体験された例である。

6.4 具体例 4

次の例は、配列（データ構造）を参照して、自動作曲を行なう例である。

```
琉球音階 = 配列 ！  作る。
琉球音階 ！  『ド』 入れる 『ミ』 入れる
```

[4]本稿で述べる「ドリトル」の実験授業は、通商産業省「IT クラフトマンシッププロジェクト」の 2005 年度の補助で行われた。

```
  『ファ』 入れる 『ソ』 入れる 『シ』 入れる。
僕の楽譜=メロディ!作る。
「僕の楽譜!(琉球音階 ！ (random(5)) 見る) 追加」
    !16 回繰り返す。
```

これらの授業を行なったところ、教室では

- 「不思議な感じ」「沖縄っぽい」「他の国のメロディも作れるのかな」などと子供同士でわいわい話していた。
- その後、1 人の児童が、「沖縄の音階はレとラがないけど、日本の音階は『ミ』と『シ』がないはずですよね」と言いながら、あれこれ実験をしていた。

との状況が見られた。

結果として、新版では音楽を簡単に記述できるようになり、小学校高学年であれば容易に演奏プログラムを扱えるようになったことを確認した。

7 まとめ

本論では、「情報教育の音楽化」の考え方と、「初心者用プログラミング言語『ドリトル』」の関係について述べた。今回のバージョンアップによってドリトルの演奏表現能力が著しく向上し、今までは不可能だった豊かな演奏表現力がドリトルに取り入れられることになった。また、演奏に必要な記述も減少すると共に、「音楽演奏におけるオブジェクト指向での記述」を実現することにもつながった。

今後は、タートルを利用したゲーム効果音の高度化、「校歌をドリトルカラオケにしよう」といったテーマ学習の教材を作成や、タートルグラフィックスを全く学んでいない学習者を対象に音楽演奏から入るドリトル入門の授業実践などを行ない、初等中等教育におけるプログラミング体験を更に普及させたいと、筆者らは考えている。

謝辞

実験授業は通商産業省「IT クラフトマンシッププロジェクト」の補助で行われました。実験に協力いただいた、おゆみ野南小学校の佐藤和浩先生、藤枝市立青島中学校の西ヶ谷浩史先生、東京学芸大学の青木浩幸さん、静岡大学の紅林秀治先生、NPO ゆーらっぷの原久太郎さんに感謝いたします。

資料リスト

[1] **「高等学校におけるプログラミング教育で何を教えるべきか」**、辰己 丈夫、 筧 捷彦, 情報処理学会 1998 年度夏のプログラミングシンポジウム pp.55-66, (1998)

[2] **「情報教育の音楽化」**、辰己 丈夫, 情報処理学会 コンピュータと教育研究会 第 61 回研究会, 2001-CE-61(情処技報 Vol.2001,No.101),pp.39-46,ISSN 0919-6072(2001)

[3] **「オブジェクト指向言語による初中等プログラミング教育の提案」**、中谷多哉子、兼宗進、御手洗理英、福井眞吾、久野靖. オブジェクトストーム：情報処理学会論文誌, Vol.42, No.6, pp.1610–1624, 2002.

[4] **「学校教育用オブジェクト指向言語「ドリトル」の設計と実装」**、兼宗進, 御手洗理英, 中谷多哉子, 福井眞吾, 久野靖. 情報処理学会論文誌, Vol.42, No.SIG11, pp.78–90, 2001.

[5] **「小学校におけるプログラミング活用の現状と課題」**、佐藤和浩, 紅林秀治, 兼宗進. 情報処理学会 コンピュータと教育研究会, 2005.

[6] **「初中等教育におけるオブジェクト指向プログラミングの実践と評価」**、兼宗進、中谷多哉子、御手洗理英、福井眞吾、久野靖. 情報処理学会論文誌, Vol.44, No.SIG13, pp58-71, 2003.

[7] **「プログラミング学習についての一考察：ロボット制御のプログラミング学習とソフトウエア作りのプログラミング学習を比較して」**、紅林秀治, 兼宗進. 情報教育シンポジウム (SSS2004), 2004.

[8] **「プログラミングを利用したネットワーク学習の試み」**、西ヶ谷浩史, 紅林秀治, 兼宗 進. 情報教育シンポジウム (SSS2005), 2005.

[9] **「プログラミングを題材とした国際交流授業の提案」**、兼宗進, 李元揆, 久野靖. 情報教育シンポジウム (SSS2004), 2004.

[10] **「高校教科「情報」の現状と将来」**, 情報処理学会 情報処理教育委員会 シンポジウム報告集、2005

[11] **「体系的な情報教育の実施に向けて」**[5]、文部省・情報化の進展に対応した初等中等教育における情報教育の推進等に関する調査研究協力者会議, 1997

[12] **「Tonyu - アニメーション作成に特化したプログラミング言語と開発ツール」**、長慎也, 情報処理学会プログラミングシンポジウム、2001 年 8 月.

[5] http://www.mext.go.jp/b_menu/shingi/chousa/shotou/002/toushin/971001.htm