

Tonyu System 2

ゲーム制作を通じたプログラミング学習に適したフレームワーク

長島和平^{†1} 長慎也^{†1}

著者らはウェブベースのプログラミング環境「Tonyu System2」を開発し、それを用いて学生にゲームを制作させる授業を行った。それ以前の同授業では、JavaScript とそのフレームワークである enchant.js を使用していた。Tonyu System2 ではキャラクタごとに動作を記述できることや、フレームごとに繰り返す処理に while 文などを使用できることから、プログラミングにおける制御構造の学習に向いていると考えられる。本研究では、授業内で学生が制作したゲームのコードから、Tonyu System2 を用いることで学ぶことができたと思われる概念について報告する。

Tonyu system 2: Game development framework suitable for programming learning.

KAZUHEI NAGASHIMA^{†1} SHINYA CHO^{†1}

We developed web-based programming environment "Tonyu System 2". We applied "Tonyu System 2" to programming class in which students learn programming through game developing. In the former programming class, we used JavaScript and "enchant.js", one of the frameworks for JavaScript. "Tonyu System 2" is good for learning control structures. Because the environment make students write codes separately for each character, and make them use while statements frame-by-frame processing. In this article, we report concepts which students could learn through developing games with "Tonyu System 2".

1. はじめに

近年、大学の情報科学系の学部ではゲーム制作をプログラミングや情報学の学習に取り入れることが増えてきた。明星大学においても、C や Java を用いたゲーム制作を通じてプログラミング学習を行う授業が存在する。ゲーム制作を通じてプログラミング学習を行うことは学生の興味・関心を惹きつけるために大きな役割を果たす[1]。

一方で、学生が自主的に学習することに目を向けた場合、C や Java を学習するためには環境を学生自身で整える必要がある。その環境を自宅のパソコンなどにそろえる手間は学生の学習への壁になりうる。そこで、環境をそろえるという壁を取り除くため、著者らは Web ブラウザ上で動作する環境に着目した。Web ブラウザ上でプログラミングを学習することができれば、インターネット環境さえあればどこにいても、どのパソコンからでも学習することができる。

そこで 2013 年度、Web ブラウザ上でプログラミングができる jsdo.it[2] という環境を用いて実験授業を行った。JavaScript とそのゲーム制作用フレームワークである enchant.js[3] を使用して行った実験授業は、授業時間外や、実験期間終了後のコードの更新が見られ、学習意欲と自主学習のしやすさに手ごたえを感じることができた。しかし、JavaScript+enchant.js を用いたプログラミング学習には、制御構造の学習に向かない点や、これまで行われてきたプログラミング教育の順序を入れ替えなければならないなど、いくつかの問題点が認められた。

そこで著者らは、Web ブラウザ上でゲーム制作をするこ
とで、制御構造を含め順序立ててプログラミング学習がで
きるよう配慮した環境「Tonyu System 2[4]」を構築した。
「Tonyu System 2」は、前身である「Tonyu System[5]」に
Web ブラウザ上でのプログラミングや、スマートフォン上
で動作するためのタッチ機能など、近代的なプログラミング
を行うことができるよう改良を施したものである。2014
年度の実験授業では Tonyu System 2 を用いたゲーム制作を
通じたプログラミング学習を行った。

本研究では、jsdo.it と Tonyu System 2 の 2 つの環境を用
いて行った実験授業について報告する。また、それぞれの
実験授業のなかで学生が記述したコードを比較することで、
環境の違いにおける制御構造の使われ方の違いについて述
べる。

2. これまでの研究

著者らは、2013 年度の明星大学の実験授業において、3
年次後期に開講される「情報学実験 II」のなかでゲーム制
作を通じたプログラミング学習をテーマとした教材を作成
した。

情報学実験 II は、4 年次の卒業研究に向けたプレゼミ的
な位置づけであり、この実験を通じて、卒業研究に必要と
なるプログラミングの復習ができることが望ましい。

2.1 JavaScript を用いたゲーム制作

情報学実験 II は授業回数が 4 回しかなく、その時間でゲ
ームを完成させることは難しいため、授業時間外にもゲー

^{†1} 明星大学大学院情報科学研究科
Graduate School of Information Science, Meisei University.

ムを制作できることが望ましい。しかし、学生が自宅で自主的にプログラミング学習を行う際には、CをWindows上で動かすための環境であるCygwinや、Javaに対応した統合開発環境であるeclipseを各自インストールする必要があり、手間がかかる。著者らは学習者がこのような特別な準備を必要としない環境でプログラミング学習を行うことがよいと考えた。そこで、Webブラウザ上でJavaScriptのコードを作り、動作確認や保存、また他のユーザのプログラムを閲覧・コピーして編集することができるjsdo.itという環境を扱うこととした。jsdo.it上でJavaScriptとそのゲーム制作フレームワークであるenchant.jsを使用したプログラミング学習を行った。

この実験を通じて、学生が授業時間外に自宅などでコードの更新を行ったり、実験が終わった後にもコードを追記したりするなど、学習意欲と自主学習のしやすさについて一定の手ごたえを感じることができた。

2.2 JavaScriptを用いたゲーム制作の問題点

しかし、JavaScriptはマウスやキーボードからの入力やタイマーの作動を受けて実行されるイベント駆動であるために、ゲーム制作における継続的な処理を記述することに向いていない。

ゲームに限らず、プログラムは本来離散的な表現しかできないが、ゲームプログラムにおいてはキャラクターの移動を表現するために、移動の過程を1秒間あたり30~60コマ(=フレーム)程度に分けて、キャラクターが滑らかに動いているように見せている。各フレームにおいては、キャラクターの移動と描画を繰り返し行う。例えば、キャラクターが画面左端から画面右端に向かって移動するとき、一回でキャラクターの座標を指定するのではなく、1フレームで少し右へ動かし、次のフレームでまた少し右へ動かしという動作を「繰り返し」行うことで実現される。この処理は、プログラム上でも繰り返し処理で記述することが自然であると考えられる。

しかし、JavaScript+enchant.jsを用いてゲームを制作した場合、フレームごとの繰り返し処理の記述にwhileやforは使用できず、onenterframeというフレームごとに呼び出されるイベントハンドラにフレーム1回当たりの処理を記述しなければならない。例えば、図1はキャラクターを四角形の軌跡に沿って動かすプログラムである。JavaScript+enchant.jsでは図1に示すように、ある地点まで右へ動かし続け、その地点に到達したら下に動かし続け、そのあとは左へ…と繰り返す処理を記述するために、今キャラクターがどちらへ動いているかを保持する状態変数を使う必要があり、繰り返し行われる処理であるにも関わらず繰り返し部分をwhile文などで記述することができない。

プログラミング学習に使用するという点を考慮すると、

```
game.rootScene.onenterframe=function() {
    if(chara1.state==0) {
        chara1.x=chara1.x+3;
        if(chara1.x>300) chara1.state=1;
    }
    if(chara1.state==1) {
        chara1.y=chara1.y+3;
        if(chara1.y>300) chara1.state=2;
    }
    if(chara1.state==2) {
        chara1.x=chara1.x-3;
        if(chara1.x<50) chara1.state=3;
    }
    if(chara1.state==3) {
        chara1.y=chara1.y-3;
        if(chara1.y<50) chara1.state=0;
    }
}
```

図1 キャラクタを四角く動かすプログラム
(JavaScript+enchant.js)

繰り返し処理にwhile文やfor文といった制御構造を使用することができないことは不自然であり、学習者にとっては負担になりうる。また、jsdo.itでは一つのファイルにゲーム一つのコードを全て書き込むため、プログラムが長くなるにつれて学習者が動作を追加する際にどこに書き足せばよいかが分かりにくくなることや、予想と違う動作が起きた場合に修正すべきプログラムがどこに書いてあるのかを探すことが難しいということもあった。

また、これまで大学で行われてきたプログラミング教育では、Hello Worldを出力させるところからはじめ、変数の扱いやキーボードからの入力を序盤に教えている。続いて条件分岐や繰り返し処理といった制御構造、関数、配列や構造体といったデータ構造と順に進め、最後にオブジェクト指向の概念について学ぶ。このように以前から行われてきたプログラミング教育では、一度に多くの学習項目が出てきて学習者が躓くことがないように配慮されている。この流れを極力崩すことなく、ゲーム制作をすることでプログラミング学習を行いたい。しかし、JavaScript+enchant.jsを用いた授業では、フレームごとの処理を行う場合にメソッドのオーバーライドを行う必要があったり、関数を値として扱う必要があったりと、本来終盤に出てくる学習項目が多く出てきて、繰り返しがめったに使われないので、今まで学習してきたプログラミングを復習することには向いていない。

3. 関連するプログラミング環境

近年行われているゲーム制作を通じたプログラミング学習では、様々な環境が使用されている。

3.1 Scratch[6]

制御構造を学ぶ際に障害となりうるのが、制御構造の範囲をあらわすための記号を入力し忘れたり、インデントを正しくつけないために構造がはっきりしなくなったりするなどの文字入力に起因する問題である。これらの問題を気にすることなく制御構造を学べる環境として、「Scratch」がある。Scratch はビジュアルプログラミング環境で、ブロックを組み立てることでプログラミングを学習することができる。ブロックの種類には条件判断や繰り返しがあり、繰り返しブロックを組み合わせることで画面内のキャラクタを継続的に動かすことができるので、制御構造を直感的に学ぶことができる。また、Scratch を用いて授業を行った報告例もある[7]。しかし、Scratch はブロックに日本語で命令が書かれていたり、ブロックを組み合わせるだけで動作したりするなど、構文等を理解していないプログラミング初学者に向けて作られたものである。したがって、すでにプログラミングを学習したことのある情報科学系学部の学生が使用することで、適当な効果が得られるとは考えにくく、これまで学んできたテキストベースのプログラミング環境を扱うことが望ましいと考える。

3.2 Processing[8]

Processing は、簡単に図形などを書き、動かすことができるプログラミング環境である。Processing を用いた授業実践の例もある[9][10]。グラフィックを動かすことによってプログラミングを学習することができるが、Processing におけるプログラムは、`setup` と `draw` という二つのメソッドからなっている。`setup` には初期設定などが記述され、`draw` にはフレームごとの繰り返しの処理が記述される。この点において、2.1 で紹介した JavaScript+enchant.js と同じように、フレームごとの繰り返しの処理を表すために `while` 文や `for` 文を使用することができず、フレームごとに呼び出される `draw` に1フレームで行う処理を記述することになる。もし、`draw` メソッドの中で `while` 文や `for` 文を用いたとすると、その繰り返しは1フレーム中に一気に実行されることになり、複数のフレームにわたる動作を表現することにはならない。制御構造を学ぶ上で繰り返し処理に `while` 文や `for` 文を使用できないことは学習者にとっての負担であると考えられる。

3.3 Unity[11]

Unity は、2D と 3D のゲームを制作できる環境で、スマートフォン上で動かすゲームも制作することができる。

Unity ではフレームごとの処理を `while` 文で書く方法も用意はされているが、デフォルトでは Processing の `draw` メソッドのようにフレームごとに呼び出される `Update` メソッドが使われる。また、ゲームのオブジェクトとその動作がコンポーネント単位に分割されており、オブジェクト指向の知識を前提としたプログラミングが必要になる。したがって、オブジェクト指向を熟知していないと利用することが難しいと考える。

4. Tonyu System 2

著者らはゲーム制作に適したプログラミング環境である「Tonyu System 2」を開発した。Tonyu System 2 はフレームごとの繰り返し処理に `while` 文を特別な手続きなしに使用することができる。図 2 は、Tonyu System 2 で書かれたプログラムの例である。このプログラムはゲームにおける 1 つのキャラクタの動作を表現している。`x` と `y` はキャラクタの座標を表し、`update` メソッドはそのフレームの処理をそこで終了させる働きをもつ。したがって、図 2 のプログラムはキャラクタの `x` 座標が 300 より小さい間は `x` を 3 増やし、処理をやめて画面に描画することを繰り返し、次はキャラクタの `y` 座標が 300 より小さい間は `y` を 3 増やし処理をやめて画面に描画する…という動作をする。なお、図 2 のプログラムは図 1 で示した JavaScript+enchant.js で記述したプログラムと同様の動作を行うものである。図 1 のプログラムとは異なり、フレームごとに繰り返す処理に `while` 文を使用しており、ステート変数は用いられていない。

```
while(true){
    while(x<300){
        x=x+3;
        update();
    }
    while(y<300){
        y=y+3;
        update();
    }
    while(x>50){
        x=x-3;
        update();
    }
    while(y>50){
        y=y-3;
        update();
    }
}
```

図 2 キャラクタを四角く動かすプログラム
(Tonyu System 2)

また, Tonyu System 2 は種類の異なるキャラクタの動作は別のプログラムファイルに記述することを強制する. これにより 1 つのプログラムが長くなることを抑止することができる. Tonyu System 2 の編集画面は図 3 に示す通り, 画面左側にファイル一覧が表示され, そこから選択したファイルのコードが画面中央に表示される.



図 3 Tonyu System 2 のコード編集画面

Tonyu System 2 は, 以前アクションゲーム製作に適した開発環境として開発された「Tonyu System」の新バージョンとして開発した. Tonyu System は, Web 上からアプリケーションをダウンロードし, デスクトップでそれを実行, プログラムの記述やゲームの実行などを行うものであった.

Tonyu System 2 はアプリケーションのダウンロードの必要がなくなり, Web ブラウザ上でプログラムの記述やゲームの実行を行えるものとなった. また, 公開したゲームはスマートフォン上でも動かすことができる. スマートフォンが普及した現在, 学生がプログラミングを学ぶにあたり, PC 上だけでなく普段から使っているスマートフォン上でも動作するゲームを制作できることは学生にとって良いモチベーションになりうる.

5. Tonyu System 2 を用いた実験授業

著者らは, 2014 年度後期の「情報学実験 II」で Tonyu System 2 を用いて実験授業を行った. この実験授業は週に 3 コマの授業を 4 週間, 計 12 コマ行う. 1 週目には Tonyu System 2 の使い方をチュートリアル形式で教え, その後学生が作りたいゲームを作品計画書に記入する. 2~4 週目は学生がそれぞれ作品計画書に書いた内容になるようなゲームを制作し, 4 週目の最後には制作したゲームについてレポートを書き提出する. この実験授業には, 3 クールで合計 30 人の学生が参加した.

また, 2014 年度の夏に二日間にわたり開かれたオープンキャンパスにおいても, Tonyu System 2 と

JavaScript+enchant.js で一日ずつ模擬授業を行ったが, 授業時間が短く, 完成作品がこちらで与えたサンプルプログラムとほとんど同じだったため, 比較の対象から外した.

5.1 完成コードの比較

実験授業で, JavaScript+enchant.js を用いて制作されたゲームのプログラムと Tonyu System 2 を用いて制作されたゲームのプログラムとの比較を行った. このうち JavaScript+enchant.js で制作されたプログラムは, 8 人分が 2013 年度後期の情報学実験 II で制作されたもの, 残る 37 人分は 2014 年度前期の情報学実験 I で制作されたものである. なお, 情報学実験 I では, Android で動作するアプリケーションの制作の一環で, JavaScript で動作するアプリケーションの制作を行わせた.

JavaScript+enchant.js で制作された 45 人分のプログラムをすべて見たところ, for 文を使用したプログラムを書いたのは 13 人, while 文を使用したプログラムを書いたのはたったの 1 人だった. それぞれ for 文と while 文は, 敵やアイテムなどのキャラクタを一括生成するとき, キャラクタを一括で移動させるとき, シューティングゲームなどで複数の敵との衝突判定を行うとき, パズルゲームにおいて同じ色のブロックが一定数以上つながっているかどうかの判定で使用されていた.

Tonyu System 2 で制作された 30 人分のプログラムを見ると, for 文を使用したプログラムは 11 人, while 文はすべての学生が使用した. for 文は敵やアイテムなどのキャラクタの一括生成, 複数のキャラクタを一括で移動させるとき, すべての敵へダメージを与える処理, 複数の敵との衝突判定, 複数フレームの待機, といった用途に使用されていた.

表 1 Tonyu System 2 で使用された for 文の用途と回数, JavaScript+enchant.js で同じ目的での for 文の使用の可否

用途	使用回数	JS+enchant
キャラクタの 一括生成	12	使用可
キャラクタの 一括移動	1	使用可
キャラクタの 一括削除	2	使用可
すべての敵へ ダメージ	1	使用可
複数の敵との 衝突判定	2	使用可
複数フレーム 待機	3	使用不可

表 1 では, Tonyu System 2 で使用された for 文の目的と回数, そして JavaScript+enchant.js で同じ目的のために for 文の使用が可能であることを表で示している. 最も多く使用さ

れたのはキャラクタの一括生成であった。キャラクタの一括生成、一括移動、一括削除、複数の敵との衝突判定は、1フレーム内で行う処理であるため JavaScript+enchant.js で記述することができる。しかし、複数フレームを待機することを JavaScript+enchant.js で記述する場合には for 文を使用することはできない。

図4には学生が実際に記述した for 文で複数フレームの待機を行っているプログラムの構造を示す。初期位置などの設定を行った後、図中★の箇所で 120 フレーム待機を行いその後で移動などの処理を行っている。フレームの処理をその場で中断する update メソッドがあるため、Tonyu System 2 ではフレームごとの繰り返し処理を for 文や while 文を使用して記述することができる。

```
x=$screenWidth/4+rnd($screenWidth/2);
y=-100;
for(i=0;i<120;i++){// ★
    update();
}
while(y<$screenHeight){
    y+=vy;
    // 略
    update();
}
die();
```

図 4 Tonyu System 2 で学生が記述した複数フレーム待機するコード

```
game.rootScene.onenterframe=function(){
    charal.counter++;
    if(charal.counter>120){
        charal.y=charal.y+charal.vy;
        // 略
    }
};
```

図 5 図 4 の複数フレームの待機部分を JS+enchant.js で書き換えたコード

図 5 は、図 4 と同様に複数フレーム待機してからキャラクタが動き始めるようなコードを JavaScript+enchant.js で書き換えたコードである。複数フレームの待機を JavaScript+enchant.js で記述しようとすると、待機フレーム

の変数(charal.counter)を毎フレームカウントアップし、そのカウントが一定数に達するまでその後の処理を始めないようにしなければならない。

表 2 では、Tonyu System 2 で使用された while 文の目的と回数、そして JavaScript+enchant.js で同じ目的のために while 文の使用が可能であるかを表で示している。最も多く使用されたのはフレームごとの繰り返しであった。これは単体で使用する分には、JavaScript+enchant.js でも onenterframe で代用することができる。また、アイテムの一括生成は JavaScript+enchant.js でも同様に記述することができる。しかし、2 番目、3 番目に使用回数の多かった 1 ファイル内に 2 つ以上の連続した繰り返し処理や、全体の繰り返しは JavaScript+enchant.js では記述することができない。

表 2 Tonyu System 2 で使用された while 文の用途と回数、JavaScript+enchant.js で同じ目的での while 文の使用の可否

用途	使用回数	JS+enchant
フレームごと 繰り返し	230	onenterframe
1ファイル内に 2つ以上の連続	44	使用不可
全体の繰り返し	24	使用不可
複数の状態から なる動作	12	使用不可
アイテム生成	2	使用可
複数フレーム 待機	2	使用不可

```
while(true){
    // A
    update();
}
while(true){
    // B
    update();
}
```

図 6 Tonyu System 2 で学生が記述した 1 ファイル内に 2 つ以上の繰り返しが連続して使われているコードの概略

図 6 には、1 ファイル内に 2 つ以上繰り返しが連続して使われている例を示す。図中 A の部分では、ある地点に到達するか、またはキーボードからの入力を受けるまでの処理が書かれてあり、ある地点に到達するかキーボードからの入力を受け取った段階で break される。続いて図中 B の部分で、先ほど break された地点からの処理が記述されて

いる。こうした A という動作を行った後に B という動作を行うことを, Tonyu System 2 では while 文を連続して記述することで直感的に表すことができる。

JavaScript+enchant.js で同様のコードを記述した場合の例を図 7 に示す。図 7 のように, onenterframe 内に A の動作と B の動作を両方記述し, どちらの動作を行っているかを管理する状態変数を用いると, A の動作を行った後に B の動作を行っていることが分かりにくい。

```
Chara.onenterframe=function(){
    if(this.state=="A"){
        // A
    }
    if(this.state=="B"){
        // B
    }
};
```

図 7 図 6 のコードの概略を JS+enchant.js で書き換えたコード

```
while(true){
    // A
    while(true){ // タイトル画面
        // B
        update();
    }
    // C
    while(true){ // ゲーム中
        // D
        update();
    }
    while(true){ // ゲーム終了後
        // E
        update();
    }
}
```

図 8 Tonyu System 2 で学生が記述した全体の繰り返しの使用したコードの構造

図 8 には, 学生が全体の繰り返しを使用して記述したプログラムの構造を示す。このプログラムは, while 文のなかに 3 つの while 文が連なる形で書かれている。図中 A と書かれた箇所で, ゲーム開始前のタイトル画面を表示し, その直後の while 文 (図中 B) でゲームを開始させる入力があるまで待機する。入力を受けると break され, 図中 C と書かれた箇所で, プレイヤーの表示やスコアなどの初期化を行い, 直後の while 文 (図中 D) でゲーム中の動作を行う。これはシューティングゲームのメインプログラムであり, この図中 D の部分では, 敵をランダムで出現させたり, 残り時間を計測したりし, プレイヤーが倒されるか, 時間が切れると break する。続く while 文 (図中 E) で, ゲーム終了後の画面を表示し, 入力があるまで待機する。ここで入力を受けると, 外側の while 文によりまた図中 A に戻るため, 繰り返しゲームを行うことができる。このプログラムのように while 文を駆使してゲームの動作を記述することで, 学習者が分かりやすく繰り返し動作を学習できる。

```
game.rootScene.onenterframe=function(){
    if(state==0){ // タイトル画面
        // A
    }
    if(state==1){ // ゲーム中
        // B
    }
    if(state==2){ // ゲーム終了後
        // C
    }
};
```

図 9 図 8 のコードの構造を JS+enchant.js で書き換えたコード

図 9 には図 8 のコードの構造を JavaScript+enchant.js で書き換えたコードを示す。図中 A ではタイトル画面を表示させ, 入力があったらゲーム開始準備をし, state を 1 にする。図中 B ではゲーム中の動作を記述し, 終了条件になったらゲーム後の画面を準備し, state を 2 にする。最後に図中 C では入力があるまで待機し, 入力があり次第 state を 0 にする。このように, Tonyu System 2 では while 文を 4 つ使用して実現していたコードを JavaScript+enchant.js で記述するためには状態変数を使用する必要があり, while 文や for 文を使用することはできない。

図 10 には, 学生が記述した, 複数の状態からなる動作を用いたプログラムの構造を示す。このプログラムで動作するキャラクタは, ボタンが押されたときにジャンプをすることができる。図中 A の部分では, 左右への移動などのジャンプしていないときの動作が記述されている。図中 B の部分で, ジャンプをするボタンが押されたかどうかを判定し, ボタンの入力を受けていたら中の while 文の処理が行われる。図中 C では, キャラクタがジャンプしてから地面に着地するまでの動作が記述されている。キャラクタが

着地をすると break され、図中 A の動作をするように戻る。

```
while(true){
  // A
  if(// B){
    while(true){
      // C
      update();
    }
  }
  update();
}
```

図 10 Tonyu System 2 で学生が記述した複数の状態からなる動作を用いたコードの構造

このプログラムも通常のフレームごとの繰り返しの中にジャンプをしたときのフレームごとの繰り返しが入っているため、JavaScript+enchant.js では、ジャンプ状態であるときとそうでないときという状態変数を使用することで実現しなければならない。

これらの点から、JavaScript+enchant.js でプログラミングを学習することに比べ、Tonyu System 2 でプログラミングを学習することは制御構造の理解に向いていると考えられる。

5.2 同一の動作を行うコードの比較

実験授業において、JavaScript+enchant.js 書かれたゲームと Tonyu System 2 で書かれたゲームは、その内容に差異がみられる。そこで、同じゲームを両方で記述した場合の繰り返しの使われ方を検証するために、Tonyu System 2 で制作されたゲームを著者が JavaScript+enchant.js に移植した。元となるゲームは 2014 年度後期の情報学実験 II で受講者の一人が作成したシューティングゲームで、Tonyu System 2 のプログラム中には 15 回 while 文が使用され、for 文は使用されていなかった。一方で、JavaScript+enchant.js に移植したプログラムでは、while 文が 1 度も使用されないのに対し、for 文が 4 度使用された。ゲーム中の繰り返し処理に使用されていた while 文はすべて状態変数などを使用して onenterframe に移植されたためなくなったが、Tonyu System 2 では使用されなかった for 文が使われた。この for 文は複数の敵との衝突判定をするために使用されている。

図 11 に、元となったゲームに記述されていた衝突判定部分を抜粋して示す。Tonyu System 2 における衝突判定は crashTo メソッドで行われる。crashTo メソッドへ渡したクラスのオブジェクトのうちどれかと、自分自身が衝突していれば、衝突しているオブジェクトが戻り値として返ってくる。つまり、図 11 に示した例では、Chara1 クラスのキ

ャラクタが複数体画面上にいても、crashTo メソッドでそれらの中から自らと衝突しているキャラクタを取得することが可能である。

```
c1=crashTo(Chara1);
if(c1){
  c1.die();
}
```

図 11 Tonyu System 2 のゲームの Bullet(弾)クラスで使用されていた衝突判定（弾と Chara1 が衝突したら Chara1 が消える）

```
for(var in charals){
  if(charals[i].intersect(this)){
    group.removeChild(charals[i]); // A
    charals.splice(i,1); // B
  }
}
```

図 12 図 11 の衝突判定を JavaScript+enchant.js で書き換えたコード

図 12 に JavaScript+enchant.js で記述した衝突判定部分を抜粋して示す。enchant.js には衝突判定を行う intersect メソッドが用意されているが、このメソッドでは crashTo メソッドのようにクラスを渡して衝突しているオブジェクトを取得する方法がなく、自分自身と他の 1 つのオブジェクトとの衝突判定を得ることしかできない。そのため、衝突判定の対象となるキャラクタは生成時に配列へと入れておく必要がある。そして、フレームごとの処理を行う際、その配列に格納されたオブジェクトの分だけ for 文で衝突判定を行う。そこで衝突していたオブジェクトに対し、画面から消す（図中 A）、配列から削除する（図中 B）という動作を行うことで、複数の敵との衝突判定を行うことができる。

JavaScript+enchant.js でゲームを制作すると、繰り返し処理がほとんど使用されない。ここで移植したコードで使われた 4 つの for 文はすべて複数の敵との衝突判定のために使用したが、それ以外には while 文も for 文も使うことはなかった。複数の敵との衝突判定を行いたいという題材に繰り返し処理を利用させること自体は妥当といえるが、同時に配列の概念も理解しなくてはならないことは学習者にとっては大きな負担になると考えられる。

Tonyu System 2 ではクラスを指定して衝突判定を行う。クラスという概念もまた、プログラミング学習では終盤に

出てくるものであるため、学習者への負担となりうるが、キャラクタの種類ごとにコードを記述すると考えることで、学習者はクラスであることを意識せず書くことができる。ゲームのキャラクタには、プレイヤーや敵、弾など、いくつかの種類があることは自然と考えられることなので、「ある種類のオブジェクトのどれかとぶつかっていれば」ということは容易に理解できるはずである。

6. 考察

JavaScript+enchant.js を用いた学習では、メソッドのオーバーライドや、繰り返しと配列を同時に学ばなければならない点、複数の状態を持つ繰り返しや、プログラム全体の流れを繰り返し処理で書くことができない点などがあることから、学習者が繰り返しの基本を復習するための環境に適しているとは言えない。

一方 Tonyu System 2 は、多くの学習者が繰り返し構造を使用していたので、制御構造について学習する機会を多く与えることができる。

しかし、ほぼ同じ内容のファイルを複数作成していたり、同様の処理をコピーアンドペーストで繰り返し行っていたり、きれいなコードにはなっていない。今回は、コピーされたコードについても while 文の使用回数を個別に数えたため、実際にはあまり繰り返し構造を学習できていない可能性もある。今後は関数化や継承などを使ってファイルをまとめるなど、コードを整理させるような指導もすべきであると考えられる。

7. まとめ

本研究では、ゲーム制作を通してプログラミング学習をすることができる Tonyu System 2 を開発した。その Tonyu System 2 を使用して実験授業を行い、以前 JavaScript+enchant.js を用いた実験授業で作られたゲームプログラムとの比較を行った。その結果、Tonyu System 2 では JavaScript+enchant.js に比べ繰り返しが多く使用されていたことがわかり、制御構造の学習に適していることが示唆された。

今回は、制御構造に焦点を絞って使用頻度とその用途の比較を行った。今後は学生のプログラムの質の向上のために手続きの抽象化や、コードを効率よくまとめることなどを含めた指導を考えていく。具体的には、配列や関数、継承などの学習項目についても教えられるような教材の作成を目指し、それら学習項目が学習できているかの検証を行う。

また、本研究で Tonyu System 2 を扱ったのは、すでにプログラミングを学習済みの 3 年生であったが、今後は入学前教育や入学後の演習において使用することを目標とする。

参考文献

- 1) EL-NASR,M,S and SMITH,B,K.: Learning Through Game Modding, ACM Computers in Entertainment (2006).
- 2) jsdo.it
<http://jsdo.it/>
- 3) enchant.js
<http://enchantjs.com/ja/>
- 4) Tonyu System 2
<https://github.com/hoge1e3/Tonyu2/>
- 5) Tonyu System
<http://hoge1e3.sakura.ne.jp/tonyu/>
- 6) Scratch
<https://scratch.mit.edu/>
- 7) 伊藤一成:Scratch を用いた授業実践報告,情報処理 Vol.52 No.1(2011)
- 8) Processing
<https://processing.org/>
- 9) 菊池誠:Processing によるプログラミング教育,情報処理 Vol.52 No.2(2011)
- 10) 三浦元喜:Processing Web IDE を用いたプログラミング基礎教育の試み,情報処理学会情報教育シンポジウム 2013 論文集(2013)
- 11) Unity
<http://unity3d.com/>