Vol.2018-CE-146 No.5
2018/10/20

シューティングゲームに特化したプログラミング言語 Connect STGの設計と評価

岸本 有生[1,a)]　兼宗 進[2]

概要：シューティングゲーム（STG）を題材にした、初学者向けのプログラミング学習環境 ConnectSTG を提案する。ゲームは従来から学習者のモチベーションを向上させる題材として認められてきたが、STG についてはプログラムが複雑になることから初学者の学習は困難とされてきた。ConnectSTG では STG のプログラムの構造を検討することで、初学者が記述可能な部分と、システムがサポートすべき部分を整理した。そして、初学者が STG を作成可能にすることと、段階的に機能を増やすことでプログラミングの基礎を学習できる学習環境を開発した。ConnectSTG の学習で、初学者は自機、敵機およびそれらの発射する弾の動作を、変数、条件分岐、ループ、座標などを学習しながらオブジェクトのメソッド定義に相当する形で記述する。ゲームループによるフレーム生成や、複数の敵機や弾の生成と管理などはシステムが自動的に行うことで、学習者がこれらの本質的なプログラミングに注力できるようにした。本発表では ConnectSTG の概要を説明し、高校で実施した 2 つの授業による実践を報告する。

キーワード：プログラミング教育, STG, ゲーム

Design and Evalution for Connect STG: Programming language specialized in Shooting Game

Tomonari Kishimoto[1,a)]　Susumu Kanemune[2]

Abstract: Game programming is an attractive example for students. However, the program of shooting game (STG) is too dicult for students. So, we propose a programming learning environment Connect STG. By using it, students can write STG programs easily. To make STG programs, they have to understand only some basic programming concepts such as variables, conditional branches, loops and screen coordinates. In this paper we will explain the outline of Connect STG and report lessons in two high school classes.

Keywords: Programing edhucation,STG,Game

1. はじめに

プログラミングを学習するには、学習者に対してプログラミング言語の知識だけでなく、論理的思考能力やコンピュータの知識などの能力が必要となってくる [1]。しかし、殆どの学習者が、開発環境のインストールや基礎的なプログラミングの知識を理解する段階で挫折している [2]。次期学習指導要領 [3] の共通必履修科目「情報 I」では、プログラミングが含まれており高等学校では必須となる。授業では、興味・関心を惹きつけ、個々の能力に合った課題が設定できる環境が必要となってくる [1][4]。その中でも、興味・関心を惹かせるために、出力結果にグラフィックスを取り入れている教材がある [5][6][7]。

本研究では高校生が、シューティングゲーム (以下 STG) の制作を通して容易にプログラミングの基礎が学習できる

[1] 大阪電気通信大学高等学校
Osaka Electro-Communication University High School, Moriguchi, Osaka 570-0039, Japan
[2] 大阪電気通信大学
Osaka Electro-Communication University, Neyagawa, Osaka 572-8530, Japan
[a)] t-kishimoto@dentsu.ed.jp

環境 Connect STG を提案する。ここで STG を選んだ理由は、次の通りとなる。まず、ゲームが学習者に対してなじみの深いコンピュータの一例であり [8]、出力結果もグラフィックスで表示されることから、学習者が興味を持ち易い。次に STG は、自機、敵機、自機弾、敵機弾の各キャラクターの絵を一方向しか用意しなくても良いことから、絵の量が少なくでき、学習者は、絵を描くことよりもプログラミングに集中できる。次にルールが、各キャラクターの移動と衝突の単純なアルゴリズムである為に、初学者でも理解し易い。最後に、既にプログラミングを理解している生徒は、弾幕 STG のような複雑な弾の処理を作ることができる為、初学者以外の生徒も個々の能力に合わせて課題を楽しめるという多くのメリットを持つからである。

以下は各節で述べられている内容である。まず第 2 節では、プログラミング言語の工夫点について詳しく述べられ、それによる学習者のメリットが述べられている。第 3 節では、本環境の動作原理や使用方法を、サンプルプログラムを使って述べられている。第 4 節では、高等学校での実践授業について、指導者の指導方法やと学習者の学習工程が述べられ、第 5 節で、学習者のプログラミングに対する学習効果の評価を、確認テストの平均点や各問題の得点率から、そして、興味・関心をアンケート結果から考察している。

2. 従来の言語を用いた STG 制作

これまで筆者らは、教育用 C 言語 Study C[9] を用いて STG 制作の授業を行ってきた。Study C では、ゲーム制作が簡単に出来るようなライブラリが備わっており、キー入力、画像表示、音楽再生ができるのが特徴である。

2.1 制作における問題点

図 1 は、Study C を用いて自機を動作させるプログラムである。ここで、STG を制作する上での問題点を整理する。まず、繰り返し文を理解できていない生徒は、ゲームループに使用している for(;;) の意味が判らずに、ゲームの動作の特徴である 1 フレームずつ移動するアルゴリズムを理解できないまま実習を行う形となる（問題 1)。次にこのプログラムだけで 30 行かかってしまう為にアルゴリズムを理解するだけでなく、動作確認の為に記述をするだけで大変な作業となり、さらに、自機弾、敵機、敵機弾のプログラムを加えると合計 133 行の大きなプログラムになってしまう（問題 2)。図 2 には、STG を制作した時のプログラムの全体像を示している。STG を制作するには、ゲーム画面上にキャラクターが存在するかを判定するフラグ、x 軸、y 軸の位置や x, y 軸の増分といった変数が必要である。自機は 1 つずつ用意すれば大丈夫であるが、敵機、自機弾、敵機弾は複数用意することから、プログラム内に繰り返し文が多用され、変数に配列を用いなければいけない

```
01:HBITMAP player;
02:int px=320,py=440,pdx=4,pdy=4;
03:unsigned long tick;
04:player=gl_loadbitmap("player.bmp");
05:gl_openwin(-1, -1, 640, 480, 0);
06:tick = GetTickCount();
07:for(;;){
08:  if(GetTickCount() >= tick &&
09:  GetTickCount() < tick + 16) continue;
10:  tick = GetTickCount();
11:  gl_fillrect(0,0,640,480,RGB(0,0,0));
12:  if(gl_getkeystate(VK_LEFT) < 0){
13:    px-=pdx;
14:    if(px<0) px=0;
15:  }
16:  if(gl_getkeystate(VK_RIGHT) < 0){
17:    px+=pdx;
18:    if(px>640-32) px=640-32;
19:  }
20:  if(gl_getkeystate(VK_UP) < 0){
21:    py-=pdy;
22:    if(py<0) py=0;
23:  }
24:  if(gl_getkeystate(VK_DOWN) < 0){
25:    py+=pdy;
26:    if(py>480-32) py=480-32;
27:  }
28:  gl_drawbitmap(player,px,py,32,32,0,0);
29:  gl_refresh();
30:}
```

図 1 自機プログラム記述例（Study C）
Fig. 1 Example code of player (Study C).

(問題 3)。配列のようなデータ構造は、計算過程が見えなくて、特に入れ子が入ると添え字の間違いも生じる。そして、敵機の出現や弾の発射プログラムは、ゲーム画面上にキャラクターが存在するかを線形検索で調べるアルゴリズムを用いるため、さらにハードルが高くなる (問題 4)。当たり判定のプログラムは、3 回記述しなければいけないため量が多い (問題 5)。他にも、関数の引数が多いと、各引数の意味や順番を全て覚えるのに苦労し、入力ミスも起こすために、よく使うものだけを整理して少なくした方が望ましい (問題 6)。結果、予想と違う動作が起きた場合に修正すべき箇所がどこに書いてあるかを考えられず、プログラミング学習に使用するという点を考慮すると、プログラムの計算過程が直接結果に表れるものだけを採用し、記述するのが効果があると考えられる。それぞれの問題点と解決方法を表 1 にまとめた。

2.2 先行研究

ゲームを題材にしてプログラミング言語を学習する環境は、Tonyu system 2[7] が報告されているが、Connect STG のように、STG に特化した報告例は少ない。STG に特化されていない従来のプログラミング言語を使用すると、ゲームシステムの部分から制作を行わなければいけない。

```
01://変数宣言
02://初期設定
03:for(;;){
04:   //待ち時間設定（16 ミリ秒）
05:   //画面削除
06:   //自機の移動①
07:   if(ボタンを押す){
08:     for(i=0;i<100;i++){
09:       //自機弾の発射⑤
10:     }
11:   }
12:   for(i=0;i<20;i++){
13:     //敵機の移動②
14:     //敵機の描画
15:     //敵機と自機の当たり判定
16:     if(時間の経過){
17:       for(j=0;j<100;j++){
18:         //敵機弾の発射⑥
19:       }
20:     }
21:   }
22:   for(i=0;i<100;i++){
23:     //自機弾の移動③
24:     //自機弾の描画
25:     for(j=0;j<20;j++){
26:       //自機弾と敵機の当たり判定
27:     }
28:   }
29:   if(時間が経過){
30:     for(i=0;i<20;i++){
31:       //敵機の出現⑦
32:     }
33:   }
34:   for(i=0;i<20;i++){
35:     //敵機弾の移動④
36:     //敵機弾の描画
37:     //敵機弾と自機の当たり判定
38:   }
39:   //自機の描画
41:   //タイマー変数の制御
40:   //画面に反映
42:}
```

図 2 ゲームループによるフレームの実現（Study C）
Fig. 2 Frame structure using game loop (Study C).

表 1 従来の問題点と解決方法
Table 1 Problems and solutions.

番号	問題	解決方法
1	ゲームループ	ゲームループの自動化
2	ソースコードが長くなる	動作のみを記述
3	キャラクターに配列使用	クラスの用意
4	キャラクター出現に線形検索	専用の関数を用意
5	当たり判定の記述が多い	当たり判定の自動化
6	関数の引数は短くしたい	引数の省略

3. Connect STG の言語的な工夫

Connect STG では、2 章の問題に対応するために、以下に述べる工夫を行った。

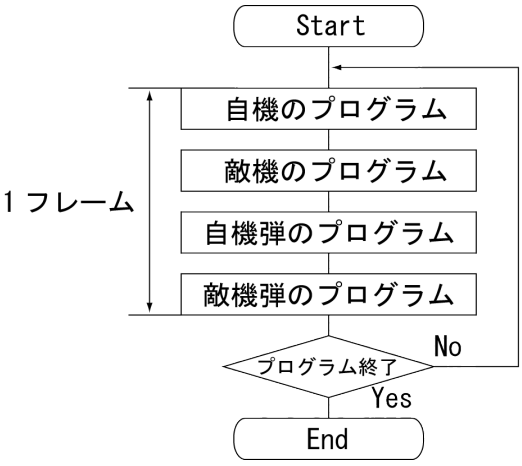

図 3 ゲームループによるフレームの構造
Fig. 3 Frame structure using game loop.

3.1 ゲームループの自動化

ゲームループを内部システムに組み込むことで、図 3 の流れ図ように自動的にループが行われるようにした。これにより、ゲームループの記述が省略でき、従来の言語での問題 1 が解決できる。例えば、学習者が 2 つのサンプルを組み合わせて新たなプログラムを制作する時に、ゲームループごとコピーしてしまう失敗がなくなる。また、今回の実装では、1 秒間に 60 フレーム動作する仕組みにした。

3.2 各キャラクターのクラスを用意

内部システムの中に自機、敵機、自機弾、敵機弾のキャラクターの各クラスを準備し、STG に必要な変数を予め用意した。学習者は、図 2 の①～④の部分を、オブジェクトのメソッド定義に相当する形で記述する。これは、Scratch[10] や Unity[11] といったプログラミング言語の仕様と似ている。それに加え、何度も記述しなければいけなかった当たり判定や描画のプログラムも内部システムが自動的に実行する。これにより、全体のソースコードは短くなり、全てを合わせても 32 行で STG が制作可能となり問題 2 を解決できた。また、クラスごとに変数を用いることから、問題 3 であった配列も意識しなくて良くなった。

キャラクターを出現させたい場合、専用の関数を記述すれば、自動的に使用されていないキャラクターを検知して使用してくれる。これにより、図 2 の⑤～ ⑦の部分が繰り返し文を含めて、一行のみで記述できる形となった。これにより問題 4 が解決できた。今回の実装では、各キャラクターの最大数を、自機は 1 体、敵機は 20 体、自機弾は 100 体、敵機弾は 100 体としている。

3.3 座標系の学習

学習者は、ゲームの座標系を理解することができる。ソフトウェアを起動すると、自機が自動的に生成される。この時、x 軸、y 軸の位置の初期値は 320, 450 が与えらる。

```
01:if(UP()){
02:  y=y-dy;
03:}
04:if(DOWN()){
05:  y=y+dy;
06:}
07:if(LEFT()){
08:  x=x-dx;
09:}
10:if(RIGHT()){
11:  x=x+dx;
12:}
13:if(frame==50){
14:  EnemySet(320,-32,0,2);
15:  frame=0;
16:}
17:if(KZ() && timer1==0){
18:  PMissileSet(x,y,0,-10);
19:  timer1=10;
20:}
```

図 4　自機プログラム記述例（以下、Connect STG）
Fig. 4　Example code of player.

```
01:x=x+dx;
02:y=y+dy;
```

図 5　自機弾プログラム記述例
Fig. 5　Example code of player missiles.

次に、x, y 軸の増分 dx, dy は共に 4 が与えられる。ここで自機が出現しているで、図 4 のプログラムに注目する。指導者は 1~3 行目の上キーが押されたのを判定する UP() の命令に対して、自機が上に移動するように、自機の位置座標 y を少しだけ減少させるプログラムを説明する。そして、学習者には、下キーを押されたのを判定する DOWN() を用いて、自機を下に移動させるプロクラムを制作させる。この時、位置座標 y が、減少するのではなく加算されなければ移動しないことに学習者は気付く。同様に左右の LEFT(), RIGHT() に対しても制作すると、次は座標位置 y ではなく、座標位置 x に対して増減を行わなければいけないことを知ることができる。

3.4　タイマー変数の用意

タイマー変数として timer1 を用意している。タイマー変数の動作は、1 フレーム経過すると 1 カウントダウンし、0 になると停止する仕様である。待ちたいフレーム数の値を代入するだけで実装することができる。同様に現在の経過フレームを調べる変数として frame を用意している。これは、実行開始時に 0 の値から始まり、1 フレーム経過するとタイマー変数とは逆に 1 加算する仕組みとなっている。これらを使い、自機弾の発射の待ち時間と敵機の出現の待ち時間に使用した例が図 4 の 13 行目 ~20 行目である。まず、13 行目では、実行開始時から 50 フレーム経過した時、14 行目の EnemySet の関数を実行し、敵機を出現させている。敵機が出現した後、15 行目のように frame に 0 の値を代入することで、50 フレーム経過まで敵が出現するのを待つことができる。17 行目の KZ() は Z キーを押しているかを判定しており、Z キーが押されていて且つ timer1 が 0 の時に PMissileSet の関数を実行し、自機弾の発射される。自機弾が発射された後、19 行目のように timer1 に 10 の値を代入することで 10 フレーム待つことができる。また、EnemySet, PMissileSet の引数は、x 軸、y 軸の初期位置と x, y 軸の増分 dx, dy を入力する必要がある。ここでのサンプルでは、敵機は上端の中心位置から出現させ、垂直下方向に移動させるために EnemySet の引数に 320, -32, 0, 2 が入力されている。自機弾は、自機と同じ座標から発射させ、垂直上方向に移動させるために、PMissileSet の引数は、x, y, 0, -10 が入力されている。また、自機弾のプログラムを図 5 に示している。それぞれの初期値は、PMissileSet で、x 軸、y 軸の位置と x, y 軸の増分 dx, dy として与えられたものを使用する。1, 2 行目の弾の座標 x, y を増分 dx, dy だけ移動させるプログラムが記述されている。

3.5　当たり判定の自動化

キャラクター同士の当たり判定は、図 6 のように①自機弾と敵機、②敵機弾と自機、③自機と敵機が 16pixel~25pixel 近づいた時の検知プログラムとダメージの計算プログラムが必要になる。今回の実装では、各距離は、キャラクターの大きさである自機、敵機は 32pixel と自機弾、敵機弾は 6pixel を考慮して、敵機には自機弾が当たり易く、自機には敵機弾が当たりにくいように調整して決めた。Connect STG では、当たり判定のプログラムを内部システムに組み込むことで、自動的に計算させ省略させている。これにより、問題 5 の 3 回当たり判定のプログラムを記述しなければいけなかった部分を解決した。キャラクターに体力 (変数名:life) と攻撃力 (変数名:attack) を与えれば、キャラクター同士の近づきを検知した場合、次の式でダメージが計算される。ただし、これらの変数の値も、特に指定しなければ、両方共に 1 が代入され、一撃で破壊できる仕組みとなっている。

$$\text{新しい相手の体力} \leftarrow \text{相手の体力} - \text{自分の攻撃力} \quad (1)$$

$$\text{新しい自分の体力} \leftarrow \text{自分の体力} - \text{相手の攻撃力} \quad (2)$$

当たり判定の検知にヒットしたキャラクターが、①の自機弾と敵機の場合、式 (1) が適用され、②敵機弾と自機もしくは、③自機と敵機の場合は、式 (2) が適用される。自機は、連続でダメージが減らないように、ダメージの計算が行われたのち 40 フレームの間は無敵状態となる。体力が 0 以下になるとキャラクターは自動的に消滅する。自機が消滅した場合、画面に「GAME OVER」と表示される。

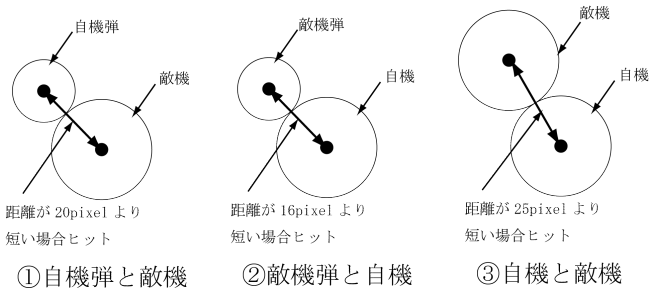

図 6 当たり判定処理
Fig. 6 Collision detection.

3.6 出現関数の引数の省略

敵機は、図 4 の 14 行目のように、EnemySet の関数を実行すると生成される。この場合の敵機の初期値はは、x 軸、y 軸の位置が 320, -32、x, y 軸の増分 dx, dy が 0,2 となる。これらの値は EnemySet の引数から入力され、画面の上方向から出現し垂直下方向に移動する。動きに必要な位置情報と移動情報のみを入力できる工夫が行われており、問題 6 が解決されている。それ以外の初期値は、自動的に予め用意された値が代入される。例えば、タイマー変数である timer1 の初期値には 50 が与えられている。これにより、特に指定しなくても、出現してから 50 フレーム後に敵機弾を発射させるプログラムを記述することができる。敵機の体力 (life) と攻撃力 (attack) は、それぞれ 1 が与えられている。これらの値は、3.5 節でも記述したように、特に指定しなかったために一撃で破壊できるように自動的に設定されている。ただし、いつまでも設定できないままでは困るので、プログラムに慣れてくると引数を多くすることで設定が可能となる。これは、敵機だけでなく、自機弾、敵機弾の生成でも同様の考えで使用できる。

3.7 自機の狙い撃ち弾

STG を制作する上で、敵機が自機を狙い撃つ弾を発射しなければ、プレイヤーに行動パターンを簡単に読まれてしまい面白みが無くなってしまう。特に Connect STG では、各キャラクターはプライベートな変数を持つため、他のキャラクターの変数にアクセスできない。このことから、狙い撃ち弾を自分で記述するのが難しい。そこで、図 7 のように、自機を狙い撃ち、その増分 dx, dy を計算するための関数 ShootX, ShootY を用意した。関数を実行すると内部システムが自動的に計算することで実装を簡単にしている。それぞれの引数には、x 軸、y 軸の位置、弾のスピードを入力する。この関数は、内部で自機との位置の差を取り、単位ベクトル化し、スピードを掛けてそれぞれの値を返してくれる。7 行目は、次に敵機弾を出すまでの待ち時間として timer1 に 50 の値を代入し、50 フレーム待つようにしている。また、図 8 には敵機弾の移動プログラムが記述されているが、弾の座標 x, y に増分 dx, dy だけ移動させるだけで済むことから、自機弾と同様であることが判る。

```
01:x=x+dx;
02:y=y+dy;
03:if(timer1==0){
04:  tdx=ShootX(x,y,4);
05:  tdy=ShootY(x,y,4);
06:  EMissileSet(x,y,tdx,tdy);
07:  timer1=50;
08:}
```

図 7 敵機プログラム記述例
Fig. 7 Example code of enemies.

```
01:x=x+dx;
02:y=y+dy;
```

図 8 敵機弾プログラム記述例
Fig. 8 Example code of enemy missiles.

図 9 ゲーム画面と座標系
Fig. 9 Game screen and coordinate axes.

4. Connect STG の設計

4.1 画面

ゲーム画面は、図 9 のように x 軸が 640pixel、y 軸が 480pixel の大きさで構成されている。座標軸は、x 軸が右方向に行くほど数値が大きくなるが、y 軸は下方向に行くほど数値が大きくなることに注意する。キャラクターに x 軸、y 軸の位置 (変数 x, y) を設定すると、その座標を中心として画像が表示される。

4.2 エディタ

Connect STG は、自機、敵機、自機弾、敵機弾が存在し、それぞれのキャラクターに必要な動作のプログラムを記述する。図 10 には、プログラムを記述するエディタが示されている。プログラムを作成するには、①からキャラクターファイルを選択し、②の部分にソースコードの記述を行う。補助機能として③の行番号や、④のフォントの大きさを変更することができる。これにより、生徒に的確な指示を与え、スムーズに実習を進めることが出来る。そして、⑤の実行ボタンで記述したプログラムを実行する。プ

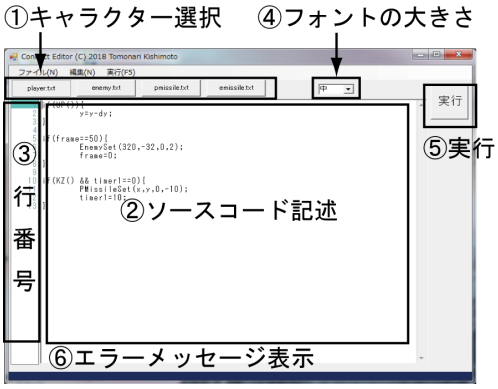

図 10　プログラムエディタ
Fig. 10　Program Editor.

表 2　用意した関数とシステム変数
Table 2　System functions and variables.

関数名, 変数名	説明
PlayerSet(x,y,dx,dy);	自機を出現させる
EnemySet(x,y,dx,dy);	敵機を出現させる
PMissileSet(x,y,dx,dy);	自機弾を発射させる
EMissileSet(x,y,dx,dy);	敵機弾を発射させる
ShootX(x,y,sp);	速さ sp の狙い撃ち弾 dx を計算
ShootY(x,y,sp);	速さ sp の狙い撃ち弾 dy を計算
Print(x,y,z);	x,y の位置に変数 z の値を表示
Rand(x);	0〜x-1 までの乱数を発生
Sin(d,a);	d[゜]、振幅 a の sin の値を計算
Cos(d,a);	d[゜]、振幅 a の cos の値を計算
timer1,timer2	タイマー変数
frame	経過フレームを調べる変数

ログラムにエラーが生じた場合、⑥のエラーメッセージに記載された行番号を用いてその前後を修正する。エディタで作られたプログラムは、テキストファイルとして保存される。

4.3　言語の実装

開発は VisualStudio を用い C#で行った。字句解析と構文解析は、いまどきのプログラム言語の作り方 [12] のサンプルを利用して改変した。動作環境は Windows8 以降であり、実行ファイルを起動するだけで動作する。管理者権限は不要でありインストールを用意にした。

言語は C の構文を参考に設計した。データ型は数値（整数）、論理型を宣言なしに扱える。制御構文は if, for, while を用意した。関数定義、構造体、浮動小数点数、文字列、配列は用意していない。システムが用意した関数を呼び出す際は、可変個の引数を渡せるようにした。表 2 に関数と特殊な変数一覧を示す。

5.　Connect STG の授業実践

5.1　提案する授業モデル

Connect STG を用いた学習方法の特徴は、STG の機能を増やしていくことでプログラムを学ぶ学習方法である (表 3)。STG 完成までの時限は 6 時限で構成されており、最初から順番に完成させなければいけない。課題 1〜4 は、指導者が示したサンプルを写すような課題ではなく、図や矢印で書かれた行動パターンを学習者に示し、自分で考えさせてコーディングできるような課題を用意する。学習者は、課題の仕様を満たすためにキャラクターを動かしながら考えることで論理的な思考を深められる。自由課題に関しては、学習者の能力に合わせたアルゴリズムを教えることも可能である。

表 3　提案する授業モデル
Table 3　Sample syllabus for high schools.

時限	概要	主な学習活動
1	自機の移動	変数、条件分岐、座標 x, y の学習
2	自機弾の発射 課題 1	タイマー変数、増分 dx, dy の学習 自機弾の発射方向の変更
3	自機弾の複数発射 課題 2	繰り返し文の学習 自機弾の発射方向や数の変更
4	敵機の出現 課題 3	フレーム変数の学習 出現位置や移動方向の変更
5	敵機の旋回 課題 4	条件分岐、オブジェクト指向の学習 旋回位置、移動方向の変更
6	敵機弾の発射 自由課題	狙い撃ち弾の学習 自主作品の制作

表 4　実践した授業
Table 4　Syllabus of two lessons.

時限	概要	主な学習活動
1	(1) 自機の移動 (2) 自機弾の発射 (3) 自機弾の複数発射	変数、条件分岐、座標 x, y の学習 タイマー変数、増分 dx, dy の学習 繰り返し文の学習

5.2　実施した授業

本報告では、実際に高等学校にて授業を行った。授業内容は、本来ならば 3 時限かかる表 3 の 1〜3 の内容を、表 4 のように、課題 1,2 を実施しない形として圧縮し、1 時限 (50 分間) で行った。理由としては、実践授業に多くの時間を取れなかったことと、短い時間でも多くの機能を生徒に体験してもらいたかったからである。授業は実習を伴うため、補助の先生が 1 人サポートしている。本対象の生徒は、普通科 3 年生 29 人と電子工業科 3 年生 22 人の生徒である。2 つの被験者の大きな違いは、プログラミング経験の有無となる。普通科の生徒の多くは、プログラミングの初心者であり、知識をほとんど持っていない。それとは逆に電子工業科の生徒の多くは、プログラミングの経験者であり、2 年間、高等学校で C 言語の学習をしているが、基礎的な部分で理解できずに挫折している生徒が多い。この授業では、変数の代入や演算、if 文、for 文を学習することを目標としている。次節では、授業の詳細を述べる。

```
01:if(UP()){
02:  y=y-dy;
03:}
```

図 11　自機が上に移動するプログラム
Fig. 11　Example code of moving up.

```
01:if(DOWN()){
02:  y=y+dy;
03:}
04:if(LEFT()){
05:  x=x-dx;
06:}
07:if(RIGHT()){
08:  x=x+dx;
09:}
```

図 12　自機が下左右に移動するプログラム
Fig. 12　Example code of moving in four directions.

5.2.1　自機の移動 (1)

上下左右の方向キーを押すと自機が上下左右に移動するプログラムである。まず、指導者が、図 11 の方向キーの上を押すと自機が上方向に移動するプログラムを学習者に説明する。この例では、x, y の初期値は 320, 450 で図 9 の自機のように、x 軸が真ん中で y 軸が下の位置に出現する。dx, dy の初期値は共に 4 が与えらる。1 行目の UP() により、方向キーの上キーを押しているかを判定し、押している場合は 2 行目を実行され、y の値が dy の値だけ減少し上の位置に移動する。学習者は、学んだことを活かして、下の場合は DOWN()、左の場合は LEFT()、右の場合は RIGHT() と用いることで、他の方向の移動を 図 12 のように完成させる。この時、自機が上に行くときは dy を減少させていたのが、下に行く時には、dy を増加させなければいけないことに学習は気付くことができる。このプログラムには、変数の代入と演算、if 文が利用されて、自機を実際に動かすことでプログラムの記述の意味を理解できる。さらに、似た記述が続くために、初学者がつまづきやすい、特殊文字のキーボードの配置も学習できる。

5.2.2　自機弾の発射 (2)(3)

繰り返し文を利用した自機弾の発射のプログラムである。図 13 には、自機弾を発射するプログラムが記述されている。指導者は Z キーを押すと自機弾が発射されることや、タイマー変数の数値を変更しながらタイマーの仕組みを説明をする。そして、PMissileSet の関数の第 3 引数と第 4 引数の 0, -10 の数値を変更し、x 軸、y 軸の増分が変化すると、弾の動きがどのように変化するのかを理解させる。次に、図 14 は、図 13 の自機弾を発射するプログラムを改良し、2 発同時に発射できるようにしたものである。まず、2 行目に繰り返し文を追加することで `i=0` と `i=1` の時の 2 回 PMissileSet が実行される。この時、PMissileSet の x 軸の入力値を `x-10+20*i` に変更しておくと、`i=0` の時は `x-10` が入力され、`i=1` の時は `x+10` が入力され、自機弾

```
01:if(KZ() && timer1==0){
02:  PMissileSet(x,y,0,-10);
03:  timer1=10;
04:}
```

図 13　自機弾を発射するプログラム No.1
Fig. 13　Example code of shooting(1).

```
01:if(KZ() && timer1==0){
02:  for(i=0;i<=1;i=i+1){
03:    PMissileSet(x-10+i*20,y,0,-10);
04:  }
05:  timer1=10;
06:}
```

図 14　自機弾を発射するプログラム No.2
Fig. 14　Example code of shooting(2).

を 2 発違う位置から同時に発射できるようになる。繰り返し文の初期値や繰り返し条件を変えることにより、自機弾の個数を変更することも可能となる。

5.2.3　確認テストの実施

授業の事前と事後にプログラミングの確認テストを行うことにより、プログラムの文法やアルゴリズムの理解度を調査した。試験内容は、各 1 点方式で全部で 10 問用意した (付録参照)。問 1 ～問 3 は、変数の代入と演算を行っている。問 4 は、if 文の書式、問 5, 6 は if 文の条件式を記述する問題である。特に問 6 は、剰余の演算や論理演算子などを出題しているため難しい問題となっている。問 7~10 は、for 文の初期値の設定と繰り返し条件を記述する問題が数値を変更して出題されている。事後テストは、事前テストと数値を変更して出題しているため、プログラムを理解していないと点数がとれない形となっている。

5.2.4　アンケートの実施

授業終了後に、授業への満足度と難易度を測る目的でアンケートを行った。本報告では 1 時限しか授業を行わなかったが、たとえ授業の難易度が高くても、満足度が高ければ次回の授業につなげることができる。授業回数を重ねれば、プログラミングに慣れて、授業難易度も低くなると期待できる。満足度では「授業は楽しいか」、「ソフトウェアは使いやすいか」、「プログラミングに対して興味を持ったか」の 3 項目を、難易度では「授業は難しいか」の 1 項目を、それぞれ 5 段階に分けて評価をしてもらった。

5.3　普通科での実践

5.3.1　授業の様子

普通科の授業では、初学者が多くキーボード入力に慣れていない生徒が多かった。全角と半角キーの打ち間違えやセミコロンや括弧などの記号をみつけるのに苦労をしていた。その結果、課題（1）の自機の移動の課題を完成するのに時間がかかってしまい、全員が課題（3）の繰り返し文を利用した自機弾の発射の課題まで完成することはでき

表 5　試験結果（普通科の概要）
Table 5　Test results (overall of general class).

	平均 M	不偏標準偏差 SD
事前テスト	1.41	2.72
事後テスト	4.48	2.57

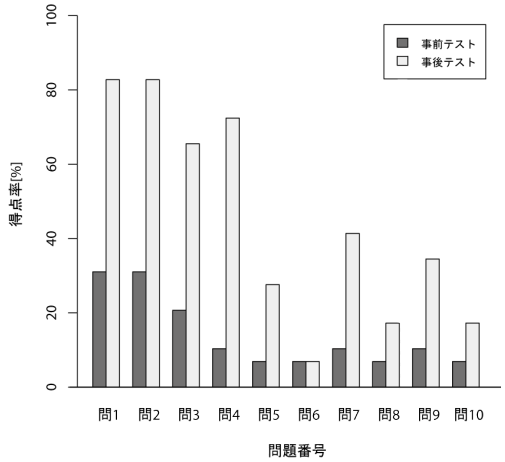

図 15　試験結果（普通科の詳細）
Fig. 15　Test results (each test of general class).

なかったが、理解の早い生徒もいた。早い段階でプログラムを完成させた生徒は、「自機弾の x 軸の増分を変更して斜めに発射させる」、「タイマーを減らしてレーザーに変更する」、「for 文の回数を多くして自機弾を多く発射させる」など、自ら能動的な学習を行っていた。

5.3.2　試験結果

表 5 には、確認テストの結果を示している。標準偏差に変化がないことから、全員が同じような学習効果が得られたといえる。事前テストが平均点 1.41 点に対して事後テストは平均点 4.48 点となった。平均点に差があるか t 検定を行ったところ、有意性が見られた (t=8.06, df=28,p<.05)。平均点の上昇が 3.07 点と初めてのプログラミングにもかかわらず高かった。

図 15 には、それぞれの問に対しての得点率を示している。どのような文法やアルゴリズムを理解したかを検証を行った。事前テストでは、得点率が全体的に低いことから、殆どの生徒が初学者であるということが判る。自機の移動の課題は全員完成させることが出来た為、変数の代入と演算、if 文の書式（問 1～ 問 4）といったプログラムの得点率が、事前テストに比べて 45%～62%の上昇があった。

5.3.3　アンケート結果

表 6 は、それぞれの平均値と標準偏差と評価 4 以上の割合を示している。ここでは、評価 4 以上の生徒は、質問に対して肯定的な意見を持っているとしている。まず、「授業の楽しさ」に対して評価 4 以上の割合が約 69%と大きく、平均値も 3.72 と高いことから、殆どの生徒が授業が楽しいと感じている。さらに、全体の中で「プログラミングに対して興味を持った生徒」が、評価 4 以上の割合を調べると約 31%いる。しかし、その反面、「授業は難しいか」に対しては、評価 4 以上の割合が約 66%あり、殆どの生徒が授業を難しいと感じており、「ソフトウェアの使いやすさ」の評価 4 以上の割合が約 28%と低くでていることが原因である。

表 6　アンケート結果 (普通科)
Table 6　Questionnaire results (general class).

	平均 M	評価 4 以上の割合
授業は楽しいか	3.72	68.97%
使いやすいか	3.03	27.59%
興味を持ったか	3.03	31.03%
授業は難しいか	3.72	65.51%

表 7　試験結果 (電子工業科の概要)
Table 7　Test results (overall of electricity class).

	平均 M	不偏標準偏差 SD
事前テスト	4.45	1.82
事後テスト	6.50	1.73

5.4　電子工業科での実践

5.4.1　授業の様子

電子工業科の授業は、キーボードの入力に慣れており、全員が課題 (3) の繰り返し文を利用した自機弾の発射まで完成できた。

5.4.2　試験結果

表 7 に確認テストの結果を示している。事前テストが平均点 4.45 点に対して事後テストは平均点 6.50 点となった。平均点に差があるか t 検定を行ったところ、有意性が見られた (t=14.69, df=21,p<.05)。事前テストの結果は、プログラミング経験者にも関わらず点数があまり高くないことから、基礎的な部分の理解が不十分であることが判る。そして、事後テストの平均点が 6.5 点と非常に高く、殆どの生徒がプログラミングを理解している状態である。

図 16 には、それぞれの問に対しての得点率を示している。事前テストの結果を見ても問 1 ～問 3 までの得点率が 80%以上と高いことから、変数の代入と演算まで理解できている生徒が多いことが判る。特に問 7～ 問 10 までの繰り返し文の得点率が事前テストに比べて 23%～45%上昇していることから、繰り返し文の理解が深まったといえる。

5.4.3　アンケート結果

表 8 は、それぞれの平均値と標準偏差と評価 4 以上の割合を示している。まず、「授業は楽しいか」に対して、評価 4 以上の割合を見てみると、約 91%の生徒が楽しいと感じている。他にも、「ソフトウェアは使いやすいか」の評価 4 以上の割合は約 45%、「プログラミングに対して興味を持ったか」に対しては約 64%と、ほとんどの生徒が満足している。それとは逆に、難易度に対しては、「授業は難し

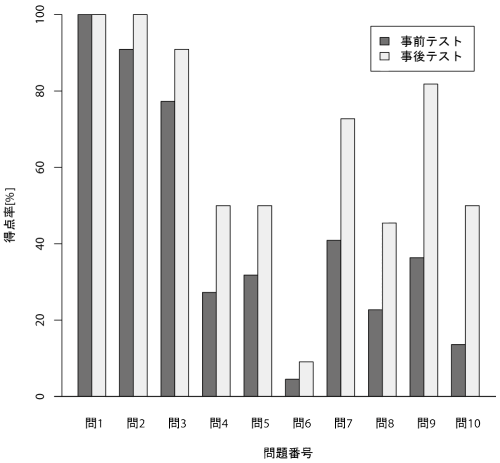

図 16　試験結果 (電子工業科の詳細)
Fig. 16　Test results (each test of electricity class).

表 8　アンケート結果 (電子工業科)
Table 8　Questionnaire results (electricity class).

	平均 M	評価 4 以上の割合
授業は楽しいか	4.64	90.91%
使いやすいか	3.64	45.45%
興味を持ったか	3.86	63.64%
授業は難しいか	3.14	22.73%

いか」に対して、評価 4 以上の割合が約 23%と低かった。

6. 考察

高等学校で実践授業を行ったところ、事後テストは、事前テストに比べて点数が高かった。以上のことから、Connect STG を用いた授業に学習効果があったといえる。また、アンケート調査では、ほとんどの生徒の満足度が高く、興味・関心を惹く題材であることを確かめることができた。授業の難しさに関しては、プログラミング初学者は、約 66%の生徒が難しいと感じていたが、初心者特有のキーボード入力によるものであり、Connect STG そのものが原因ではないことがわかった。

7. まとめ

本研究では、STG を題材にしたプログラミング学習環境 Connect STG を提案した。そして、段階的に機能を追加することで、変数、条件分岐、繰り返し、ゲーム画面の座標などを学ぶことができることを示した。実際に、高等学校で実践授業を行い、その学習効果や興味・関心を確かめた。プログラミング初学者は、簡単な変数の演算や命令の構文を理解することが出来た。プログラミング経験者は、今まで理解が困難であった繰り返し文の記述を理解することが出来た。興味・関心については、どの生徒に対しても高い教材であった。

参考文献

[1] 野口 孝文, 千田 和範, 稲守 栄: 初心者から上級者までシームレスにプログラミングを学ぶことができる持続可能な学習環境の構築, 教育システム情報学会誌, Vol.32, No.1, pp.59–70 (2015).
[2] Michael McCracken, Vicki Almstrum, et al. : A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, ACM, SIGCSE Bulletin, Vol.33, No.4, pp125–180 (2001).
[3] 文部科学省:高等学校学習指導要領 (オンライン), 入手先 <http://www.mext.go.jp/a_menu/shotou/new-cs/1384661.htm> (参照 2018-09-28).
[4] 田口 浩, 糸賀 裕弥, 毛利 公一, 山本 哲男, 島川 博光: 個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援, 情報処理学会論文誌, Vol.48, No.2, pp.958–968 (2007).
[5] 兼宗 進, 御手洗 理英, 中谷 多哉子, 福井 眞吾, 久野 靖: 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, 情報処理学会論文誌 プログラミング, Vol.42, No.SIG 11(PRO 12), pp.78–90 (2001).
[6] 長島 和平, 長 慎也, 間辺 広樹, 兼宗 進, 並木 美太郎: Web ブラウザを用いたプログラミング学習支援環境 Bit Arrow の設計と評価, 情報処理学会論文誌 教育とコンピュータ, Vol.4, No.1, pp.57–69 (2018).
[7] 長島 和平, 長 慎也: 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, 情報処理学会研究報告, Vol.2015-CE-129, No.2, pp.1–8 (2015).
[8] 長瀧 寛之: コンピュータゲームを通して情報科学を概観する 一般情報教育の授業手法の提案と評価, 情報処理学会論文誌, Vol.54, No.1, pp.2–13 (2013).
[9] 株式会社オーキッド : Study C (オンライン) , 入手先 <http://www.orchid.co.jp> (参照 2018-09-28).
[10] MIT Media Lab : Scratch (オンライン) , 入手先 <https://scratch.mit.edu/> (参照 2018-09-28).
[11] Unity Technologies : Unity (オンライン) , 入手先 <https://unity3d.com/jp> (参照 2018-09-28).
[12] randy : いまどきのプログラム言語の作り方 , 毎日コミュニケーションズ (2005).

付　　録

A.1　事前テスト

1. プログラムを実行した時の、A 点での各変数の値を答えなさい (問 1:ten1, 問 2:ten2, 問 3:ten3)。

```
ten1=100;
ten2=50;
ten1=ten1+10;
ten2=ten2*2;
ten3=ten1+ten2;
(A 点)
```

2. 次のプログラムは ten1 の値が 60 ならば「合格」と出力するものである。空欄を埋めなさい。

```
[    問 4    ](ten1[    問 5    ]){
  printf(“合格”);
}
```

3. 次のプログラムは、変数 i を 1~5 まで繰り返し「こんにちは」を 5 回出力するものである。空欄を埋めなさい。

```
for(i=[    問 7    ];i[    問 8    ];i=i+1){
  printf(“こんにちは\n”);
}
```

4. 次のプログラムは、変数 i を 1~50 まで繰り返し偶数を取り出し「こんにちは」と出力するものである。空欄を埋めなさい。

```
for(i=[    問 9    ];i[    問 10    ];i=i+1){
  if(i\%2[    問 6    ]){
    printf(“こんにちは\n”);
  }
}
```

A.2　事後テスト

1. プログラムを実行した時の、A 点での各変数の値を答えなさい (問 1:ten1, 問 2:ten2, 問 3:ten3)。

```
ten1=20;
ten2=200;
ten1=ten1+10;
ten2=ten2/2;
ten3=ten1+ten2;
(A 点)
```

2. 次のプログラムは ten1 の値が 100 ならば「100 点です」と出力するものである。空欄を埋めなさい。

```
[    問 4    ](ten1[    問 5    ]){
  printf(“100 点です”);
}
```

3. 次のプログラムは、変数 i を 6~10 まで繰り返し「こんにちは」を 5 回出力するものである。空欄を埋めなさい。

```
for(i=[    問 7    ];i[    問 8    ];i=i+1){
  printf(“こんにちは\n”);
}
```

4. 次のプログラムは、変数 i を 20~70 まで繰り返し 25 以上かつ 40 以下を取り出し「こんにちは」と出力するものである。空欄を埋めなさい。

```
for(i=[    問 9    ];i[    問 10    ];i=i+1){
  if([    問 6    ]){
    printf(“こんにちは\n”);
  }
}
```