Connect STG マニュアル

平成 30 年 2 月 6 日
大阪電気通信大学高等学校
岸本　有生

１．はじめに

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 のマニュアルです。

２．プログラム記述場所

キャラクターの動作は、スクリプトファイルに記述する。各キャラクターの個別の動作は、表１のように、ファイル別に記述する。最大数は、画面上に出る最大の個数を示している。ConnectSTG.exe を実行(ゲームを起動)することにより、キャラクターがプログラム通りに動作する。ゲーム実行時は、ESC キーを押すと終了させることができる。

表１　プログラムファイル名

スクリプトファイル名	最大数	キャラクター
player.txt	1 体	自機
enemy.txt	20 体	敵機
pmissile.txt	100 発	自機弾
emissile.txt	100 発	敵機弾

３．画像ファイル

画像は、表２のファイル名で保存すると使用できる。画像の一部を透明にした透過 png を使用しても表示できる。画像の大きさは、自機、敵機は 32×32 pixel が望ましい。自機弾、敵機弾は 6×6～10×10 pixel の間の大きさを使用するのが望ましい。ファイル名の*には半角数字を'0'から連番でいれる。プログラム内で変数 gp の値を変更すると対応する番号の画像に変更される。

表２　画像ファイル名

画像ファイル名	画像の大きさ(pixel)	キャラクター
player*.png	32×32	自機
enemy*. png	32×32	敵機
pmissile*.png	6×6～10×10	自機弾
emissile*.png	6×6～10×10	敵機弾

４．画面構成

ゲームを起動すると、図１のような 640×480 の大きさのフィールドに自機が配置される。自機の表示位置は、x 軸が 320、y 軸が 450 である。画像ファイルは、x 軸 y 軸を中心として画像が表示される。自機はフィールド外に出ていかないように、x 軸 y 軸の値を自動的に補正してくれる。敵機、自機弾、敵機弾は、フィールド外に行くと消滅する。各パラメータの範囲や役割は、９節の表３を参照すればよい。

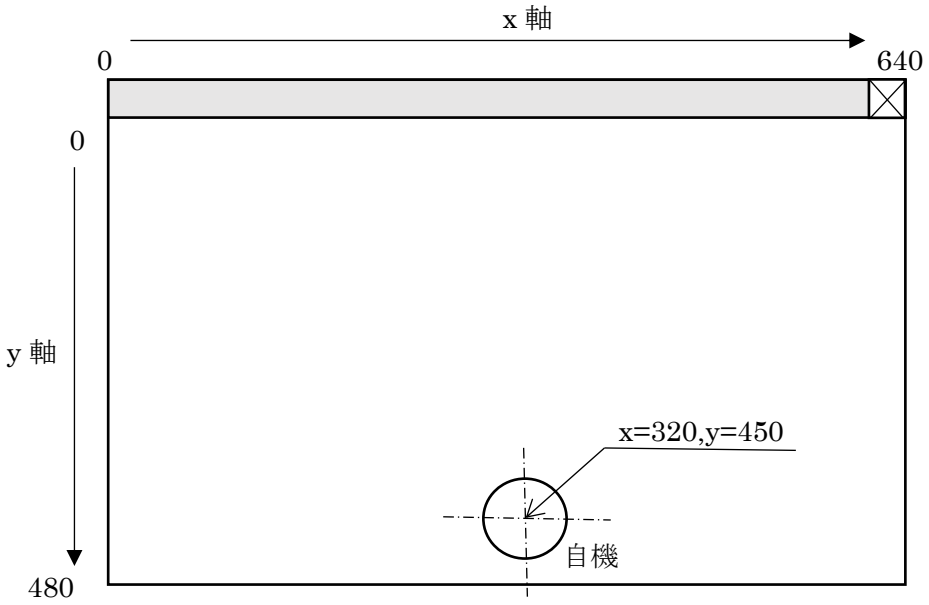
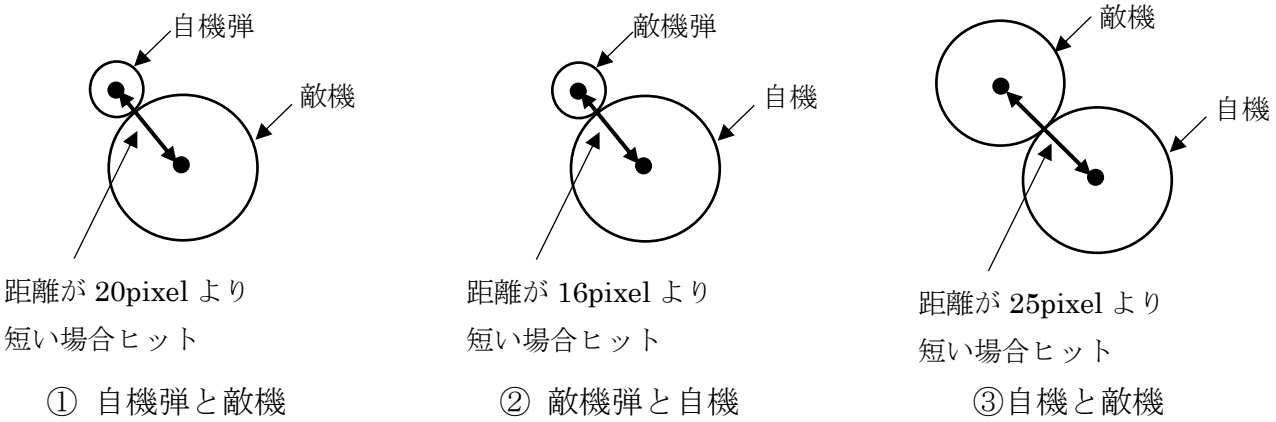


図１　実行画面

５．当たり判定

このゲームでは、①「自機弾と敵機」②「敵機弾と自機」③「自機と敵機」に当たり判定が設けられており、図２のようにキャラクターが近づく事により次の式が適用される。当たり判定が実行され、life が 0 になるとキャラクターは消滅する。自機の life が 0 になると、自機が消滅し、画面に「GAME　OVER」と表示される。



① 自機弾と敵機　　② 敵機弾と自機　　③自機と敵機

図２　当たり判定

新しい自分の体力 ＝ 現在の自分の体力 － 相手の攻撃力　　(1)
新しい相手の体力 ＝ 現在の相手の体力 － 自分の攻撃力　　(2)

ただし、③「自機と敵機」のあたり判定がヒットした場合は式(1)のみが適用される。また、②「敵機弾と自機」もしくは、③「自機と敵機」がヒットした場合、自機は 40 フレームの間、当たり判定が行われない無敵状態となる。

６．プログラムの記述とエラー

プログラムの各命令の最後には必ずセミコロン「;」をつける必要がある。ただし、if,while,for 文に於いて{　}を用いる場合は、その後ろにセミコロンをつける必要はない。プログラムの記述エラーや計算時に型のエラーが生じた場合、メッセージボックスのタイトルにファイル名が表示され、エラーが起きた行や問題点を表示してくれる。

７．プログラムの動作順

図３には、プログラム動作の順番を記述している。ゲームは、自機・敵機・自機弾・敵機弾のプログラム順に実行される。プログラムが全て実行されると、ゲームが１フレーム進んだことになり、変数 frame が加算され、各キャラクターの変数 timer1,timer2 が減算される。ESC キーが押された時や、プログラムの記述にエラーが起きた時に、プログラム終了が Yes となりループから抜ける。

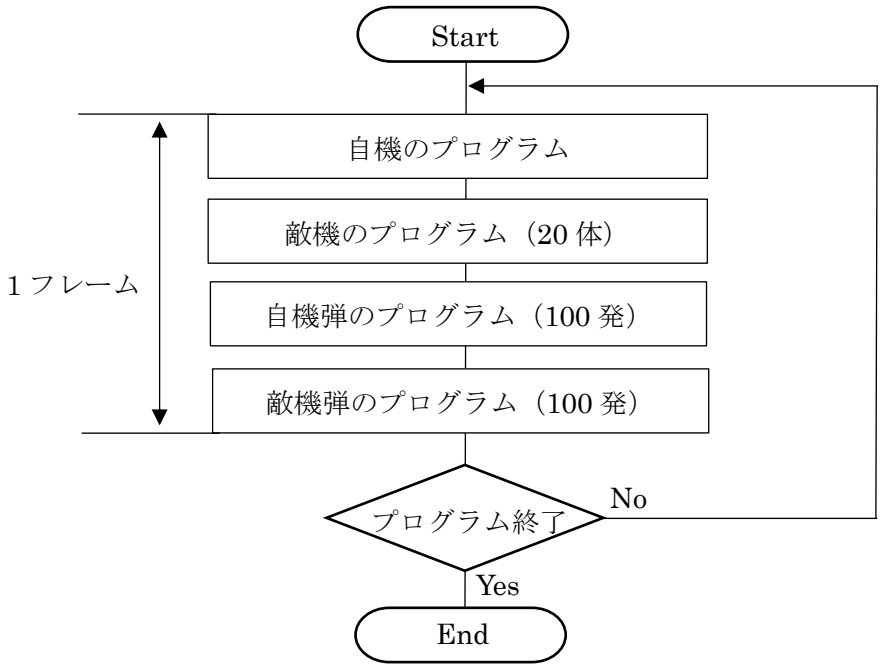


図３　フローチャート

８．定数

この ConnectSTG で、扱う値は下記のものである。

① Int 型　：整数（32bit）　　- 2,147,483,648 ～ 2,147,483,647

② Bool 型　：論理(真-Yes・偽-No）true,false

９．変数名

変数は、データを記憶できることのできる箱(メモリ)である。変数 frame 以外は、各キャラクターが互いに干渉しあわずに独立して変数を宣言して使用できる。変数の宣言は、定数を記憶(代入)させればよい。変数の名前の付け方は、'0'～'9'の数字、'a'～'z'または'A'～'Z'のアルファベットで構成される。ただし、先頭で使用する文字は、必ずアルファベットでなければいけない。また、変数の名前に、命令と同じ物や予約語(if,for,while など)は使用できない。

例)

① a = 0;　　変数 a に 0 を記憶(代入)する。

② ab0 = 0;　　変数 ab0 に 0 を記憶(代入)する。

③ 0a = 0;　　変数名の先頭が数値なので宣言できない。

④ if = 0;　　予約語を変数名にはできない。

各キャラクターに予め宣言されている予約変数が存在する。表３にその変数名と役割を示す。予約変数を使用したプログラミングを行えば、開発がスムーズに行える。ただし、変数 frame は、全キャラクター共通で値を保持しており、値を変更すれば全てのキャラクターに反映される。

表３　各キャラクター予約変数

変数名	数値	役割
x	16～624(自機) -32～672(敵機、自機弾、敵機弾) 自機は、数値が範囲外に行くと補正。敵機、自機弾、敵機弾は、数値が範囲外に行くと消滅。	x 軸 当たり判定に使用。 記憶された場所にキャラクターが描画される。
y	16～464(自機) -32～512(敵機、自機弾、敵機弾) 自機は、数値が範囲外に行くと補正。敵機、自機弾、敵機弾は、数値が範囲外に行くと消滅。	y 軸 当たり判定に使用。 記憶された場所にキャラクターが描画される。
dx dy	-2,147,483,648 ～ 2,147,483,647	x 軸移動量 y 軸移動量 x = x + dx;　このように、 y = y + dy;　スクリプトファイルに記述
gp	0 ～ n(読み込んだ画像番号)	画像番号
num	-2,147,483,648 ～ 2,147,483,647	制御用番号
flag	-2,147,483,648 ～ 2,147,483,647 ただし、実行時の初期値は 0	制御用フラグ
life	0 ～ 2,147,483,647 数値が 0 以下になると消滅	体力
attack	-2,147,483,648 ～ 2,147,483,647	攻撃力 ③ 「自機弾と敵機」、②「敵機弾と自機」、③「自機と敵機」のいずれかの当たり判定がヒットした時、次の計算式が実行される。 自分の体力＝自分の体力－相手の攻撃力 (1) 相手の体力＝相手の体力―自分の攻撃力 (2) ただし、③の当たり判定がヒットした場合は式(1)のみが適用される。また②、③のあたり判定がヒットした場合、自機は 40 フレーム間判定を行わない無敵状態となる
timer1 timer2	0 ～ 2,147,483,647 1 フレームに 1 減算される。0 以下になると、0 に補正される。	タイマー１ タイマー２ ミサイルの発射のタイミングや移動の方向を変えるタイミングに使用する。
frame	0 ～ 2,147,483,640 1 フレームに 1 加算される。 数値が範囲外に行くと自動的に補正される。	現在のフレーム数(共通変数) 敵機の出現パターンを frame で制御する。

１０．初期値

自機は、ゲームを起動したと同時に、予約変数に初期値が与えられる。各変数の初期値は表４のようになる。また、１４節に説明している初期化判定 Init()を用いれば、キャラクター作成時に初期値の再設定や、変数の宣言ができる。

表４　初期値

変数名	数値
x	320
y	450
dx	4
dy	4
gp	0
num	0
flag	0
life	1
attack	1
timer1	0
timer2	0
frame	0

１１．代入

代入は、変数に値を記憶させる動作である。それは、定数だけでなく、変数から読み取った値や算術演算、論理演算、関係演算、等価演算の結果も記憶できる。

例)

① a = 0;　　変数 a に 0 を代入する。

② d = true;　　変数 d に true を代入する。

③ a = b;　　変数 a に変数 b の値を代入する。

④ c = 5 + 4;　　変数 c に計算式(5 + 4)の結果(9)を代入する。

⑤ a = a + 1;　　変数 a に計算式(a + 1)の結果を代入する。

１２．算術演算

数値計算は次の演算子を用いる。ただし、優先順位(2)と(3)の数値計算は、a,b ともに Int 型で行わなければいけない。

表５　算術演算子

優先順位	記号	名前	意味
1	(　　)		括弧
2	a * b	乗算	a に b をかける
	a / b	除算	a から b を割る
	a % b	剰余	a を b で割った時の余り
3	a + b	加算	a に b を足す
	a - b	減算	a から b を引く
4	a = b	代入	a に b を代入する

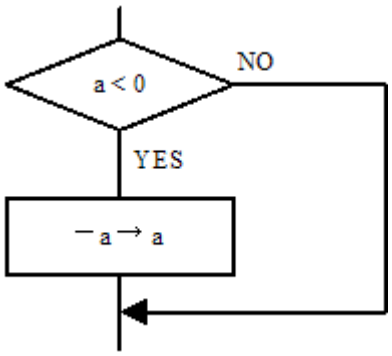
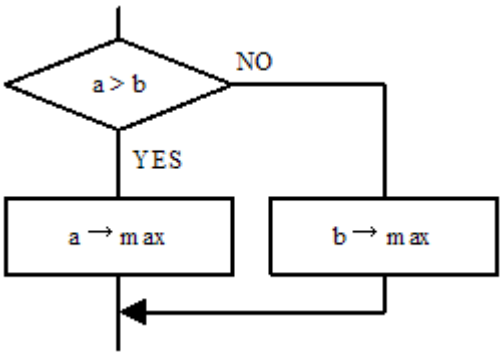
１３．制御文

(1)分岐処理 if

if 文は、表６の流れ図のように、条件が「成立するか(true,Yes)」、「成立しないか(false,No)」によって異なる処理を行う。また、条件が成立しない(false,No)時に何も処理を行わない場合、else を省略することができる。例として、else 無しでは絶対値を求めるプログラム、else 有りでは a と b の値を比較して、大きい方の値を max に代入するプログラムを示している。

ただし、if 文では、実行文が１命令のみの場合は、{　}を省略することができる。

表６　if 文の例

<table>
<tr><th>else 無し</th><th>else 有り</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>if(a < 0) {
　　a = -a;　←　Yes 実行文
}
←　else の省略可</td><td>if(a > b) {
　　max= a;　←　Yes 実行文
}
else{
　　max = b;　←　No 実行文
}</td></tr>
</table>

(2)条件式

if,while,for 文の条件式は、表 7 の演算子を用いる。正しい場合は真(true,Yes)、正しくない場合は偽(false,No)となる。等価演算子は、比べるデータの両方の型を Int 型、Bool 型のどちらかに合わせる必要がある。

表 7　条件式の演算子

種類	演算子	優先順位	意味
括弧	(　　)	1	表 4 を参照
否定演算子 (Bool 型)	!a	2	a の否定
乗算 除算 剰余 (Int 型)	* / %	3	表 4 を参照（算術演算子）
加算 減算 (Int 型)	+ -	4	表 4 を参照（算術演算子）
関係演算子 (Int 型)	a < b a > b a >= b a <= b	5	a は b より小さい a は b より大きい a は b より小さいか等しい a は b より大きいか等しい
等価演算子 (Int 型,Bool 型)	a == b a != b	6	a と b は等しい a と b は等しくない
論理演算子 (Bool 型)	a && b a || b	7 8	a と b の論理積(a かつ b) a と b の論理和(a または b)
代入 (Int 型,Bool 型)	=	9	表 4 を参照

和・差・積・商・剰余を用いた算術演算子は数値計算のみなので、結果が Int 型となってしまい、これだけでは条件式として不十分な状態である。そこで、否定演算子、関係演算子、等価演算子、論理演算子を用いて Bool 型にしてから条件式として適用する必要がある。下記に、条件式に適用した例を記述している。

例)

① a + b > 10　　a＋b の結果が 10 よりも大きければ true、そうでなければ false

② a + b > 10 && a + b < 20　　a＋b の結果が 10 よりも大きい且つ、20 よりも小さければ true、そうでなければ、false

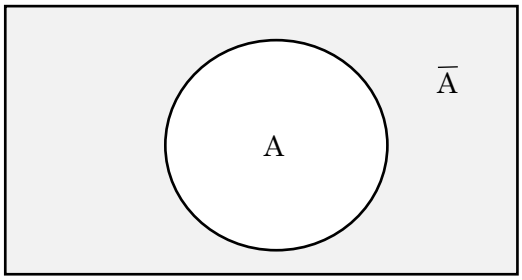
③ a + b < 10 || a + b > 20　　a＋b の結果が 10 よりも小さいまたは、20 よりも大きければ true、そうでなければ、false

(3)否定演算子、論理演算子

否定演算子と論理演算子の入力と出力結果を真理値表とベン図に示す。

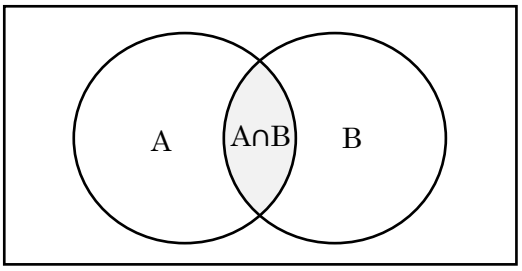
① 否定(a ではない)　!a

a	出力結果
true	false
false	true



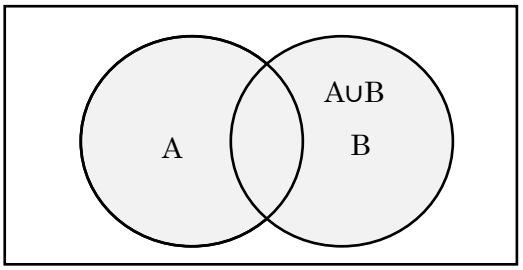
② 論理積(a かつ b)　a && b

a	b	出力結果
false	false	false
false	true	false
true	false	false
true	true	true



③ 論理和(a または b)　a || b

a	b	出力結果
false	false	false
false	true	true
true	false	true
true	true	true



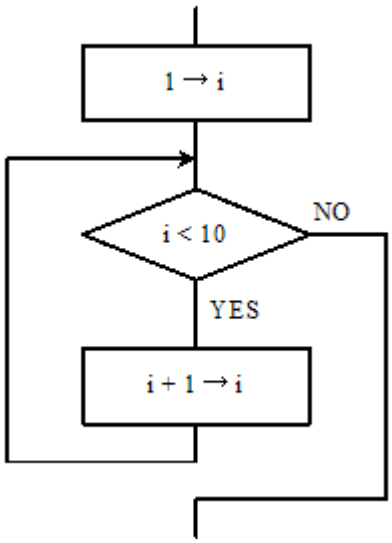
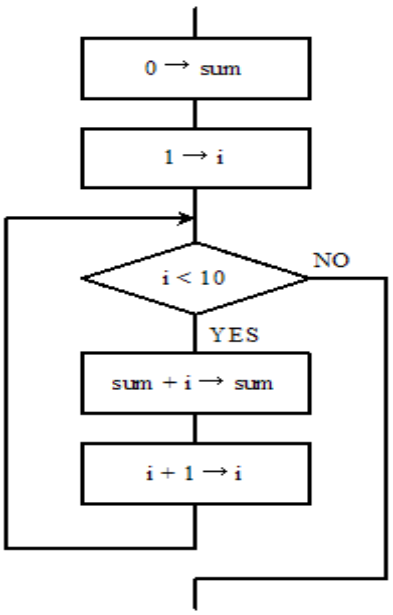
(a)　真理値表　　(b)　ベン図

図４　否定演算子、論理演算子の真理値表とベン図

(4)繰り返し文① while

while 文は、表９の流れ図のように、条件が「成立する(true,Yes)」場合に、繰り返し処理を行う命令である。ただし、while 文では、実行文が１命令のみの場合は、{　}を省略することができる。例１では、１から９まで繰り返すプログラム、例２では、１～９までの合計を求めるプログラムを示している。

表９　while 文の例

例１(１から９まで繰り返す)	例２(１から９までの合計)
	

i = 1;
while(i < 10) i = i + 1;
Yes の時繰り返し実行
While に適応させたい命令が１命令の場合{　}を省略できる
	sum = 0;
	i = 1;
	while(i < 10){
	sum = sum + i;
	i = i + 1;
	}
	Yes の時繰り返し実行

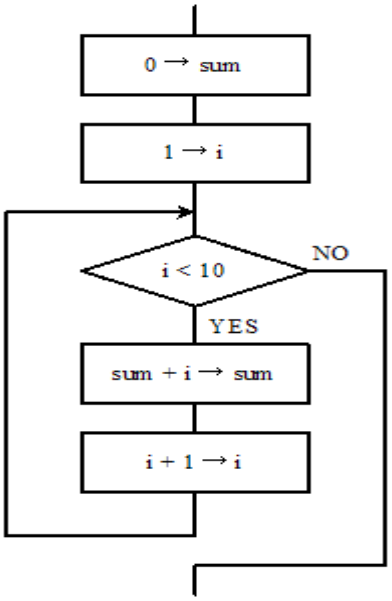
(4)繰り返し文② for

for 文は、表１０の流れ図のように、条件が「成立する(true,Yes)」場合に、繰り返し処理を行う命令である。for 文の特徴は、while 文よりも機能が多く、可読性が高い部分にある。前回の while 文の例２のように、while 文に入る直前に、「変数の初期値の設定(i＝0;)」があり、繰り返しの命令の中に「変数の増減(i＝i＋1;)」がある場合、for 文としてまとめた形で書くことができる。

ただし、for 文では、実行文が１命令のみの場合は、{　}を省略することができる。例は、１～９までの合計を求めるプログラムを示している。

表１０　for 文の例

１から９までの合計	プログラム

	sum = 0;
	for(i = 1;i < 10;i = i + 1){
	　　sum = sum + i;
	}

１４．命令

(1)キーボード入力

次の命令を使用すれば、キーボードからの入力を調べることができる。対応したキーを押しているときは true、そうでないときは false を出力する。下記に例を示す。

表１１　キー入力

命令	内容
UP()	方向キーの↑を押しているとき true、そうでないとき false
DOWN()	方向キーの↓を押しているとき true、そうでないとき false
LEFT()	方向キーの←を押しているとき true、そうでないとき false
RIGHT()	方向キーの→を押しているとき true、そうでないとき false
KZ()	Z キーを押しているとき true、そうでないとき false
KX()	X キーを押しているとき true、そうでないとき false
KC()	C キーを押しているとき true、そうでないとき false
KV()	V キーを押しているとき true、そうでないとき false

例)

①

```
if(UP()){
        y=y-dy;
}
```

方向キーの↑を押したとき、y の値が dy だけ減算される。

②

```
if(KZ() && timer1==0){
        PMissileSet(0,0,x,y,0,-10,1,1,0,0);
        timer1=30;
}
```

Z キーを押したとき、自機弾を x,y の位置から上に 10 の速さで発射する。
次の自機弾の発射まで 30 フレーム待つ。

(2)初期化判定

次の命令は、キャラクターが作成されたその時を調べることができる。それを利用し初期値の再設定や、変数の宣言を行う。

表１２　初期化判定

命令	内容
Init()	キャラクターが作成されたフレームは true、それ以外は false

例)

```
if(Init()){
        life=100;
}
```

キャラクターが作成された時、そのキャラクターの life を 100 にする。

(3)乱数

次の命令は、０から引数に入力した数に１を引いた値までの乱数を生成するものである。生成した値は、変数に代入して使用する。

表１３　乱数

命令	内容
Rand(x)	0～(x-1)までの乱数を生成する

例)

x=Rand(608) + 16;　　変数 x に乱数(0～607) + 16 を代入する。

(4)出力関数

次の命令は、図５のような画面の(x , y)の位置に変数の値や定数を表示する命令である。実行中に変数の中身を確認するために使用する。

表１４　表示命令

命令	内容
Print(x,y,z)	(x,y)の位置に変数 z の値を表示する

例)

Print(x,y,12345);　　(x,y)の位置に 12345 の値を表示する。

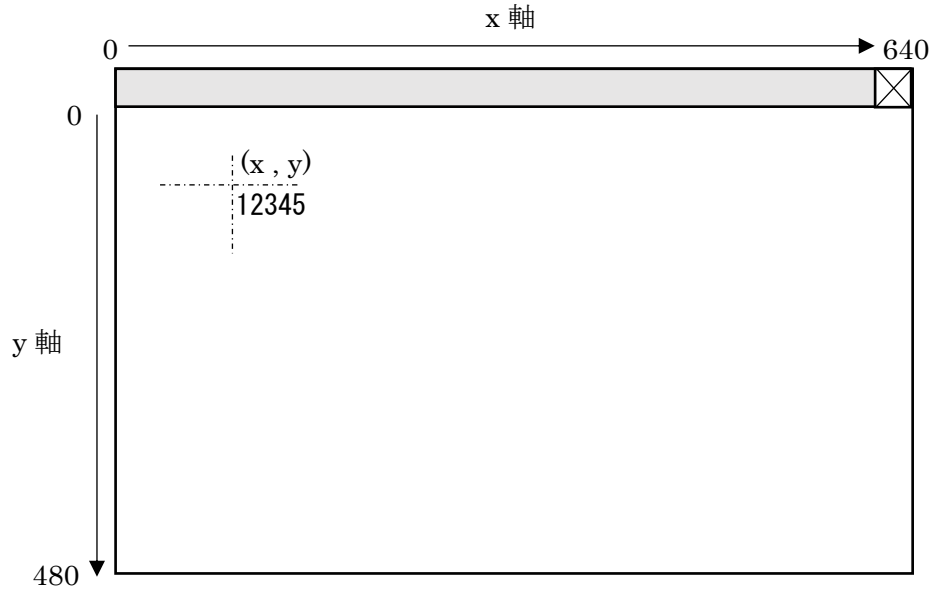


図５　Print 命令

(5)キャラクター出現関数

次の命令は、キャラクターを作成する命令である。ただし、画面に表示できる最大数以上のキャラクターは作成できない。括弧内に定数を与えることで、キャラクターの初期値を与えることができる。それぞれのパラメータは、表１６に記載されている。

表１５　キャラクター出現命令

命令	内容
PlayerSet(gp,num,x,y,dx,dy,life,attack,timer1,timer2)	自機を出現させる(1 体)
EnemySet(gp,num,x,y,dx,dy,life,attack,timer1,timer2)	敵機を出現させる(20 体)
PMissileSet(gp,num,x,y,dx,dy,life,attack,timer1,timer2)	自機弾を出現させる(100 発)
EMissileSet(gp,num,x,y,dx,dy,life,attack,timer1,timer2)	敵機弾を出現させる(100 発)

表１６　引数パラメータ

引数	説明
gp	画像番号
num	制御用番号
x	キャラクターx 軸
y	キャラクターy 軸
dx	x 軸の移動量
dy	y 軸の移動量
life	体力
attack	攻撃力
timer1	タイマー１
timer2	タイマー２

出現命令の引数を表１７のように、４つに簡略化することも可能である。ただし、その他の変数は、表１８の値が自動的に与えられる。

表１７　キャラクター出現命令

命令	内容
PlayerSet(x,y,dx,dy)	自機を出現させる(1 体)
EnemySet(x,y,dx,dy)	敵機を出現させる(20 体)
PMissileSet(x,y,dx,dy)	自機弾を出現させる(100 発)
EMissileSet(x,y,dx,dy)	敵機弾を出現させる(100 発)

表１８　引数パラメータ

引数	数値
gp	0
num	0
life	1
attack	1
timer1	50
timer2	0

(6)自機狙い撃ち弾関数

図６のように(x,y)の位置から自機に向かって、斜めに speed の移動量で移動するミサイルを発射する場合、その x 軸の移動量(ShootX)と、y 軸の移動量(ShootY)を計算することができる。

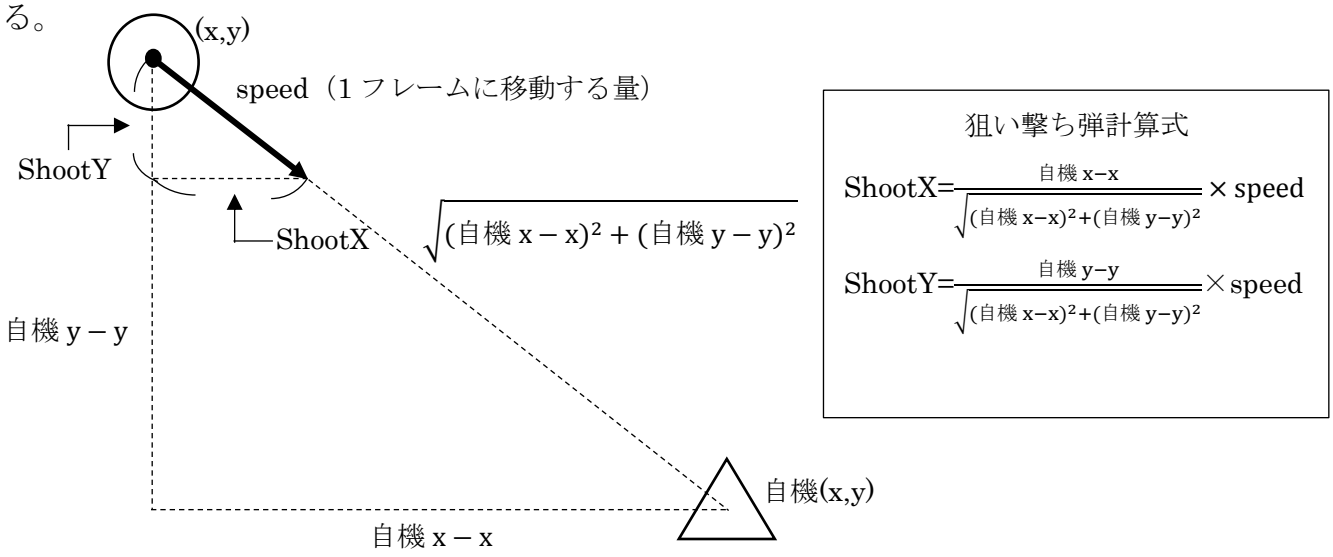


図６　狙い撃ち弾の計算

表１８　狙い撃ち弾

命令	内容
ShootX(x,y,speed)	(x,y)の位置から、自機に向けての speed に対する移動量 x を計算
ShootY(x,y,speed)	(x,y)の位置から、自機に向けての speed に対する移動量 y を計算

例)

```
if(timer1==0){
        tdx=ShootX(x,y,5);
        tdy=ShootY(x,y,5);
        if(tdx==0 && tdy==0) tdy=5;
        EMissileSet(0,0,x,y,tdx,tdy,1,1,0,0);
        timer1=50;
}
```

タイマー１が０になったら、５の速さの狙い撃ち弾の移動量を tdx,tdy に格納し、敵機弾として発射する。

※tdx,tdy ともに値が 0 の場合、ミサイルが動かないので、強制的に下に５進むようにする。

(7)三角関数 Sin,Cos

次の命令は、三角関数の値を求めるものである。式は次のようになる。amp は振幅、deg は角度[°]である。

Sin の命令=amp × Sin(deg)

Cos の命令=amp × Cos(deg)

表１９　三角関数

命令	内容
Sin(deg,amp)	角度 deg[°]、振幅 amp の Sin の値を求める
Cos(deg,amp)	角度 deg[°]、振幅 amp の Cos の値を求める

例)

```
if(timer1==0){
        tdx=Cos(45,10);
        tdy=-Sin(45,10);
        PMissileSet(0,0,x,y,tdx,tdy,1,1,0,0);
        timer1=50;
}
```

タイマー１が０になったら、10 の速さで 45° の移動量を tdx,tdy に格納し、自機弾として発射する。