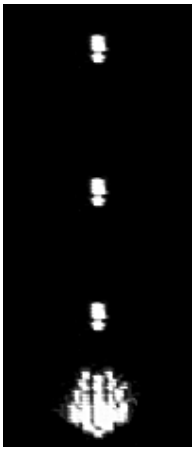
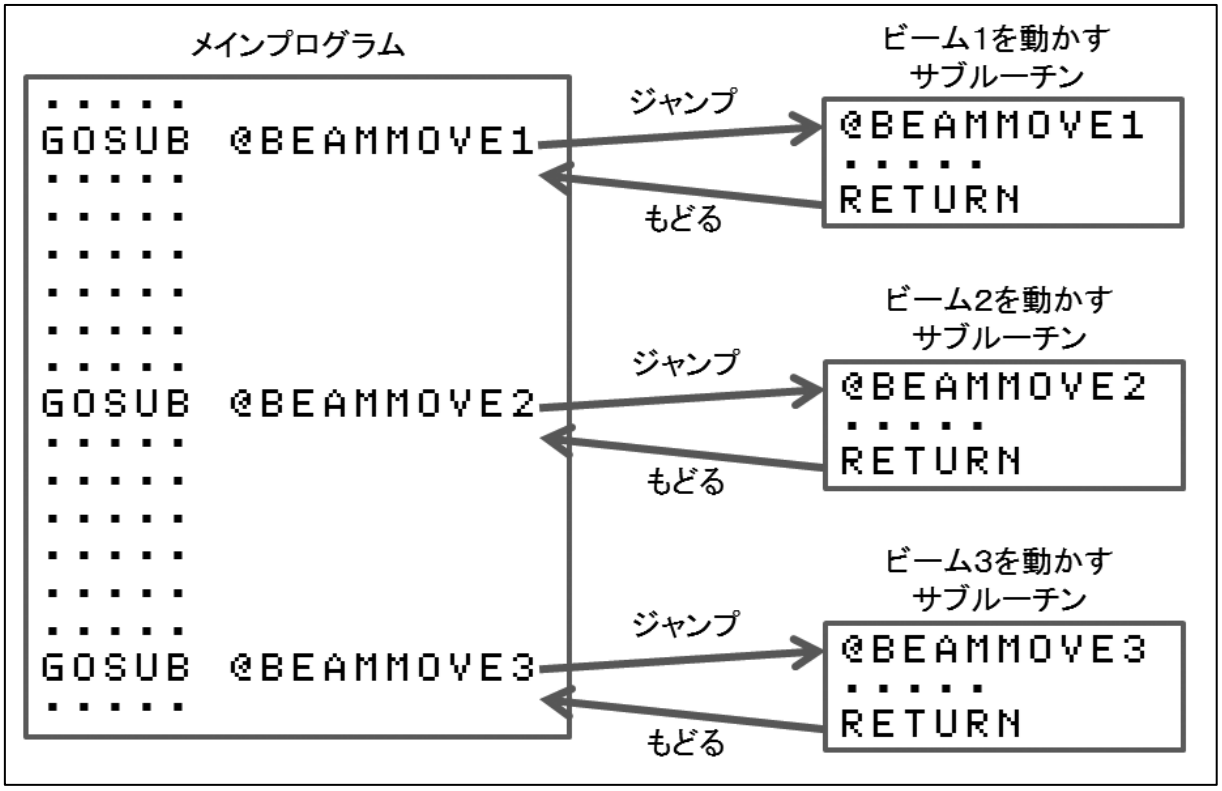
プチコンでシューティングゲームを作る（３）

●ビームを連射する



これまでは、ビームが 1 発ずつしか発射できませんでした。
ビームを 3 発まで連射できるようにしましょう。

ビームを発射したり、動かしたりするプログラムは、
メインプログラムからサブルーチンを呼んでいます。
単純に考えると、3 発のビームを動かすには、
「ビーム1のサブルーチン」「ビーム 2 のサブルーチン」
「ビーム 3 のサブルーチン」と3つのサブルーチンを作って、
それぞれ動かせばいいように思えます。

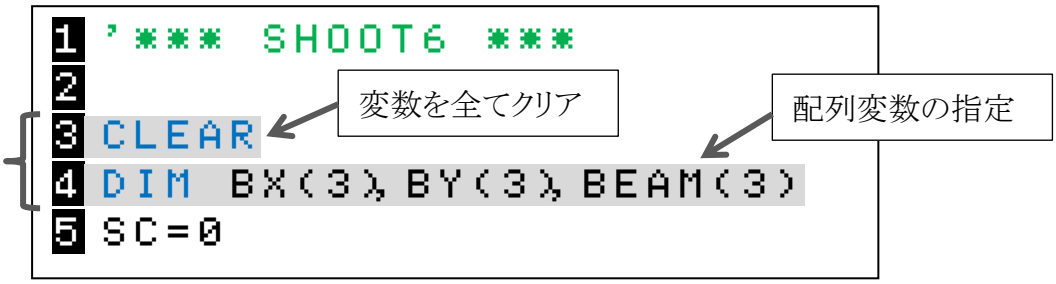


しかし、同じような内容のサブルーチンを3つ作るのは労力の無駄ですし、例えばあとで「連射を 5 発にしたい」「10 発にしたい」などと思うと、数が増えて大変です。

このように、同じような処理を多数行う時に便利なのが、**配列変数（はいれつへんすう）**です。
以下で説明します。

★配列変数（はいれつへんすう）

ビームを 3 発動かすには、ビームのx座標 BX・y座標 BY などの変数を、ビームの数の分だけ用意しないといけません。
同じような変数をたくさん使うのに、配列変数を使います。
まず、プログラムの最初の方で、変数を一度クリアするための「CLEAR」（クリア）命令と、配列変数を使うための「DIM」（ディム、ディメンジョンの略）命令を追加します。



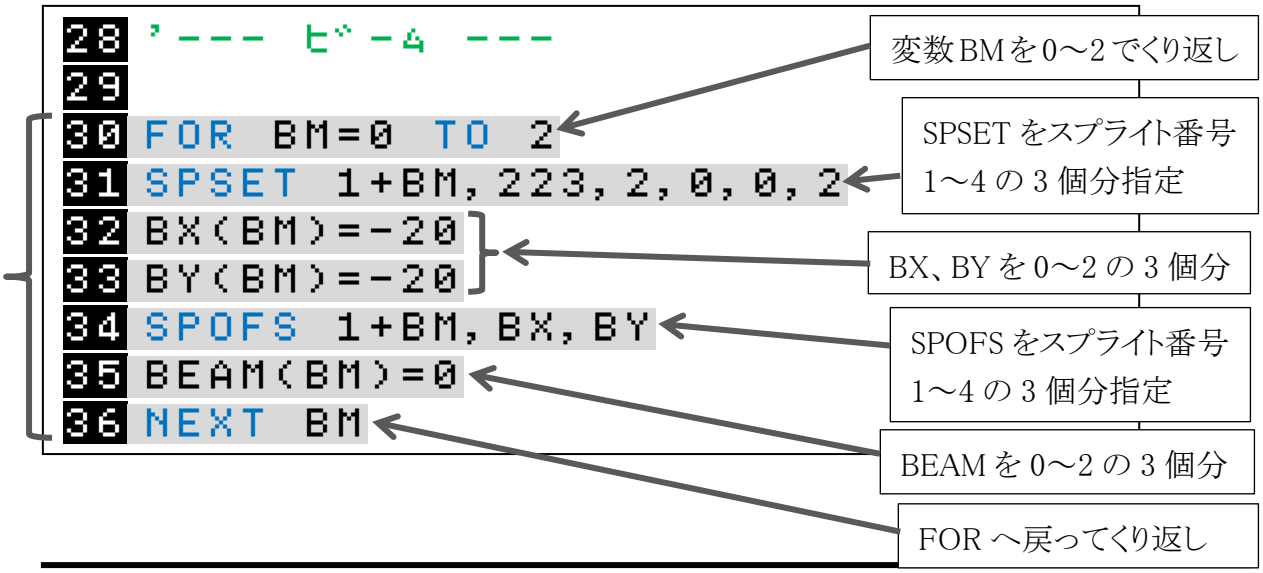
DIM 命令の文法は以下のとおりです。

DIM　　A(3)
　　変数名
　　(配列の個数)

配列の個数	例えば A(3)と指定すると、A(0)、A(1)、A(2)の 3 個の配列変数が使える。 ()内の数字（添え字）は必ず 0 から始まる。 複数の配列変数を使いたい時は、「, 」（カンマ）で区切って書く。

CLEAR 命令は、エラーでプログラムが止まった後、再度実行する時に、変数を一度全てクリアするために入れます。（入れないと DIM 命令でエラーが出ます）。

次に、ビームの初期設定をしているプログラムを、配列変数を使って 3 個分の初期設定をするように書き換えます。



★くり返し命令

配列変数 3 個で同じ設定をくり返すので、くり返し命令「FOR」（フォー）～「NEXT」（ネクスト）を使っています。
FOR 命令、NEXT 命令の文法は、以下のとおりです。

```
FOR    BM=0      TO    2      STEP    1
       ループ変数の      ループ変数の      ループ変数の
       初期値            終値              増分
```

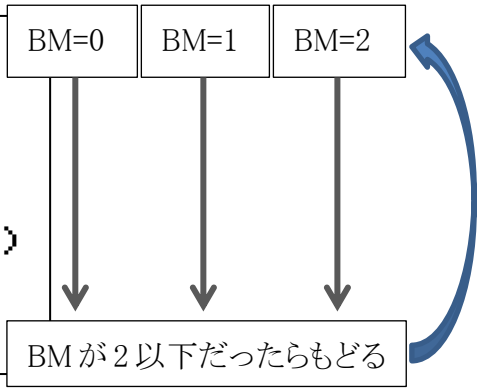
ループ変数の初期値	ループ変数の最初の値。
ループ変数の終値（しゅうち）	ループ変数の最後の値。
ループ変数の増分（ぞうぶん）	ループ変数が 1 回ごとにどれだけ変化するかを指定。 STEP 以下を省略すると、1 ずつ増える。

```
NEXT      BM
          ループ変数
```

ループ変数	指定したループ変数の FOR 命令まで戻ってくり返す。

くり返し用に使う変数を**「ループ変数」**といいます。
ここではループ変数として「BM」を使っています。
FOR 命令と NEXT 命令に挟まれた部分を、最初は BM を 0 で処理→次に BM を 1 にして処理→次に BM を 2 にして処理→くり返し終わり、という動作をします。

```
30 FOR BM=0 TO 2
31 SPSET 1+BM,223,2,0,0,2
32 BX(BM)=-20
33 BY(BM)=-20
34 SPOFS 1+BM,BX(BM),BY(BM)
35 BEAM(BM)=0
36 NEXT BM
```



これで、3 個のビームが以下のように初期設定されます。

	BM	BX(BM)	BY(BM)	SPOFS	BEAM(BM)
ビーム 1	0	-20	-20	1,-20,-20	0
ビーム 2	1	-20	-20	2,-20,-20	0
ビーム 3	2	-20	-20	3,-20,-20	0

次に、メインループのプログラムを改造します。
A ボタンが押されていたらビーム発射サブルーチンを呼ぶ所、ビームが存在したらビーム移動サブルーチンを呼ぶ所を、それぞれ FOR～NEXT で 3 回（ビーム 3 個分）くり返します。

```
58 '--- ﾙｰﾌﾟ ---
59 @LOOP
60
61 B=BUTTON()
62 IF (B AND 4)==4 AND PX>8 THEN PX=PX
-1
63 IF (B AND 8)==8 AND PX<248 THEN PX=
PX+1
64 FOR BM=0 TO 2
65 IF (B AND 16)==16 AND BEAM(BM)==0 T
HEN GOSUB @BEAMFIRE
66 NEXT BM
67 SPOFS 0,PX,PY
68 (改行を入れる)
69 FOR BM=0 TO 2
70 IF BEAM(BM)==1 THEN GOSUB @BEAMMOVE
71 NEXT BM
72 (改行を入れる)
73 IF RND(100)==0 AND TEKI==0 THEN GOS
UB @TEKIDERU
74 IF TEKI==1 THEN GOSUB @TEKIMOVE
75 (改行を入れる)
76 FOR BM=0 TO 2
77 IF BEAM(BM)==1 THEN GOSUB @BEAMMOVE
78 NEXT BM
79 (改行を入れる)
80 IF RND(100)==0 AND TEKI==1 AND MISL
==0 THEN GOSUB @MISLFIRE
81 IF MISL==1 THEN GOSUB @TEKIMOVE
82 (改行を入れる)
```

ビーム発射ルーチンを呼ぶ所を
BM=0～2 でくり返す

ビーム移動ルーチンを呼ぶ所を、BM=0～2 でくり返す

ビーム移動ルーチンを呼ぶ所を、BM=0～2 でくり返す

最後に、ビーム発射サブルーチン、ビーム移動サブルーチン、ビーム当たり処理のサブルーチンを、配列変数を使ったプログラムに書き換えます。

```
104 '--- ﾋﾞｰﾑ ﾊｯｼｬ ---
105 @BEAMFIRE
106
107 BEAM(BM)=1
108 BX(BM)=PX-7
109 BY(BM)=PY-16
110 SPOFS 1+BM,BX(BM),BY(BM)
111 RETURN
112
113 '--- ﾋﾞｰﾑ ｲﾄﾞｳ ---
114 @BEAMMOVE
115
116 BY(BM)=BY(BM)-2
117 IF BY(BM)<0 THEN BEAM(BM)=0:B
Y(BM)=-20
118 SPOFS 1+BM,BX(BM),BY(BM)
119 IF SPHITSP(1+BM,11) THEN GOSU
B @BEAMHIT
120 RETURN
121
122 '--- ﾋﾞｰﾑ ｱﾀﾘ ---
123 @BEAMHIT
124
125 SC=SC+100
126 LOCATE 6,0
127 PRINT SC
128 BEAM(BM)=0
129 SPOFS 1+BM,-20,-20
130 TEKI=0
131 SPOFS 11,-20,-20
132 RETURN
```

BM で指定した配列変数を使うように書き換え

BM で指定した配列変数を使うように書き換え

BM で指定した配列変数を使うように書き換え

さて、これでプログラムを実行すると……ビームが 1 個しか表示されません。
これは、メインループのプログラムで、A ボタンを押した時にビーム発射ルーチンを高速でくり返し呼び出しているので、3 個のビームが同じタイミングで 3 連発されて、重なって 3 個表示されてしまうからです。
これを防ぐには、A ボタンを「押した瞬間」だけビームを発射するようにします。

メインループで、A ボタンを押した時にビームを発射する部分を改造します。

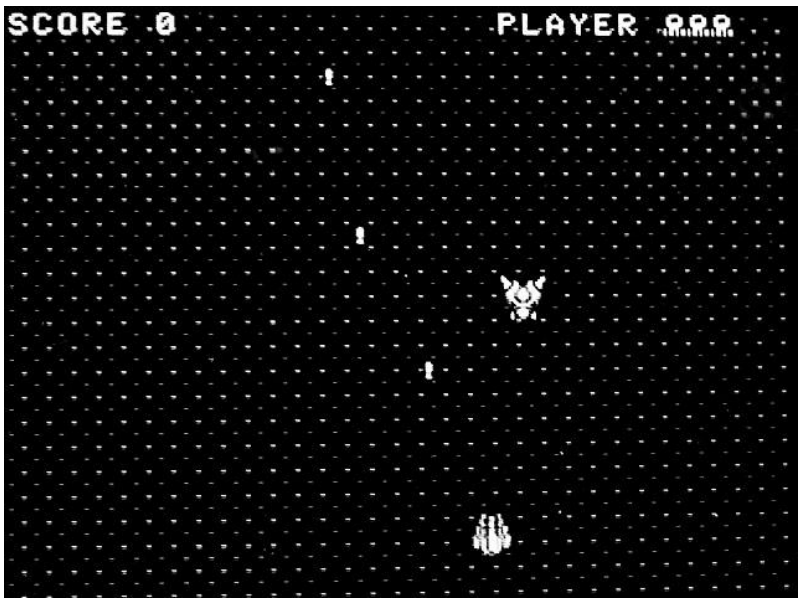
```
58 '--- ﾙｰﾌﾟ ---
59 @LOOP
（中略）
64 FOR BM=0 TO 2
65 IF (BTRIG() AND 16)==16 AND BEAM(B
M)==0 THEN GOSUB @BEAMFIRE:BM=2
66 NEXT BM
```

BUTTON 関数の代わりに
BTRIG 関数を使う

BM=2 を追加して、一発発射したら
FOR～NEXT ループを終了する

新しく「BTRIG」（ビートリガー）という関数を使います。
BTRIG 関数は BUTTON 関数と似ていますが、押した瞬間だけ（押した後の短い時間だけ）押されたボタンやキーの値を持ちます。その後はボタンを押しっぱなしでも「0」になります。
また、それだけだと、高速でくり返してやはり 3 連発してしまうので、最後に「:BM=2」を追加して、ビームを一発発射したら FOR～NEXT のループを終了するようにします。

これでプログラムを実行してみましょう。
A ボタンをくり返して押すと、ビームが 3 発出るようになります。



●敵を 3 体出す

ビームの数を増やしたので、敵の数も増やしてみましょう。
ビームと同じように、配列変数を使って、くり返しで 3 回同じ処理をします。

まず、プログラムの最初の初期設定の部分で、敵の配列変数の DIM 命令を追加します。

```
1 '*** SHOOT7 ***
2
3 CLEAR
4 DIM BX(3), BY(3), BEAM(3)
5 DIM TX(3), TY(3), TEKI(3), TDX(3), TDY(3)
6 SC=0
```

DIM 命令で、敵に関する配列変数を
3個分指定する

敵の初期設定の部分も、配列変数を使って 3 体分を設定します。

```
39 '--- ﾃｷ ---
40
41 FOR TK=0 TO 2
42 SPSET 11+TK,137,2,0,0,2
43 TX(TK)=-20
44 TY(TK)=-20
45 SPOFS 11+TK,TX(TK),TY(TK)
46 TEKI(TK)=0
47 NEXT TK
```

メインループのプログラムでは、敵を登場させるサブルーチン、敵を動かすサブルーチンを呼ぶ部分を、FOR～NEXT で 3 回くり返すようにします。

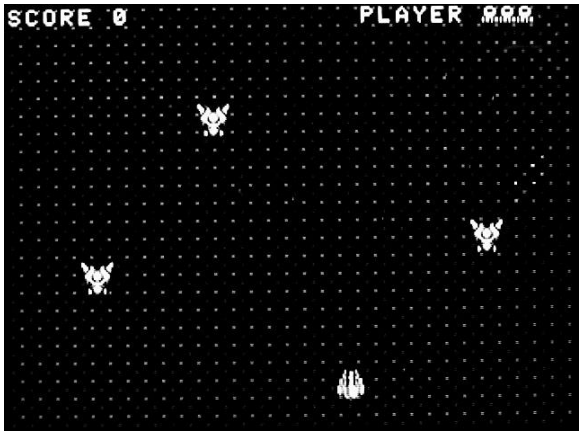
```
61 '--- ﾙｰﾌﾟ ---
62 @LOOP
  （中略）
76 FOR TK=0 TO 2
77 IF RND(100)==0 AND TEKI(TK)==0 THEN GOSUB @TEKIDERU
78 IF TEKI(TK)==1 THEN GOSUB @TEKIMOVE
79 NEXT TK
80
```

最後に、敵を登場させるサブルーチン、敵を動かすサブルーチンを、配列変数を使ったプログラムに書き換えます。

```
139 '--- テキ トウジョウ ---
140 @TEKIDERU
141
142 TEKI(TK)=1
143 TX(TK)=RND(240)
144 TY(TK)=0
145 TDX(TK)=RND(3)-1
146 TDY(TK)=1
147 SPOFS 11+TK,TX(TK),TY(TK)
148 RETURN
149
150 '--- テキ イドウ ---
151 @TEKIMOVE
152
153 TX(TK)=TX(TK)+TDX(TK)*SIN(T
Y(TK)/32)*1.5
154 TY(TK)=TY(TK)+TDY(TK)
155 IF TY(TK)>191 THEN TEKI(TK)
=0:TY(TK)=-20
156 IF TX(TK)<-16 THEN TX(TK)=
255
157 IF TX(TK)>255 THEN TX(TK)=
-16
158 SPOFS 11+TK,TX(TK),TY(TK)
159 IF SPHITSP(0,11+TK) THEN M
ISS=1
160 RETURN
```

敵に関する変数を、TK で指定した配列変数に書き換える

プログラムを実行してみましょう。
敵が3体登場します。
ただ、3体分の当たり判定を
まだ作っていないので、
ビームを当てても
すり抜けてしまいます。



●敵 3 体分の当たり判定

さらに、ビームと敵の当たり判定を、配列変数を使ったプログラムに書き換えます。

ビーム移動サブルーチンで、ビームと敵の当たり判定を3体分くり返します。

```
118 '--- ﾋﾞｰﾑ ｲﾄﾞｳ ---
119 @BEAMMOVE
120
121 BY(BM)=BY(BM)-2
122 IF BY(BM)<0 THEN BEAM(BM)=0:BY(TM)=-20
123 SPOFS 1+BM,BX(BM),BY(BM)
124 FOR TK=0 TO 2
125 IF SPHITSP(1+BM,11+TK) THEN GOSUB @BEAMHIT
126 NEXT TK
127 RETURN
```

ビームと敵の当たり判定を
ループ変数 TK で3体分くり返す

ビームが当たった時のサブルーチンも、配列変数を使って書き換えます。

```
128
129 '--- ﾋﾞｰﾑ ｱﾀﾘ ---
130 @BEAMHIT
131
132 SC=SC+100
133 LOCATE 6,0
134 PRINT SC
135 BEAM(BM)=0
136 SPOFS 1+BM,-20,-20
137 TEKI(TK)=0
138 SPOFS 11+TK,-20,-20
139 RETURN
```

プログラムを実行してみましょう。
敵3体とも、ビームが当たれば、消えて得点になります。

●ミサイルを 3 発出す

敵が3体に増えたので、敵が打つミサイルの数も3発に増やしてやりましょう。

まず、プログラムの最初に、ミサイルの分の DIM 命令を追加します。

```
1 '*** SHOOT7 ***
2
3 CLEAR
4 DIM BX(3),BY(3),BEAM(3)
5 DIM TX(3),TY(3),TEKI(3),TDX(3),
TDY(3)
6 DIM MX(3),MY(3),MISL(3),MDX(3),
MDY(3)
7 SC=0
```

ミサイルに関する変数を、配列変数で3発分指定

ミサイルの初期設定部分を、配列変数を使うように書き換えます。

```
50 '--- ミサイル ---
51
52 FOR ML=0 TO 2
53 SPSET 15+ML,190,2,0,0,2
54 MX(ML)=-20
55 MY(ML)=-20
56 SPOFS 15+ML,MX(ML),MY(ML)
57 MISL(ML)=0
58 NEXT ML
59
```

ループ変数 ML で、ミサイルに関する変数設定を3発分くり返し

メインループの、ミサイル発射サブルーチン、ミサイル移動サブルーチンを呼ぶ部分を、3発分くり返すようにします。

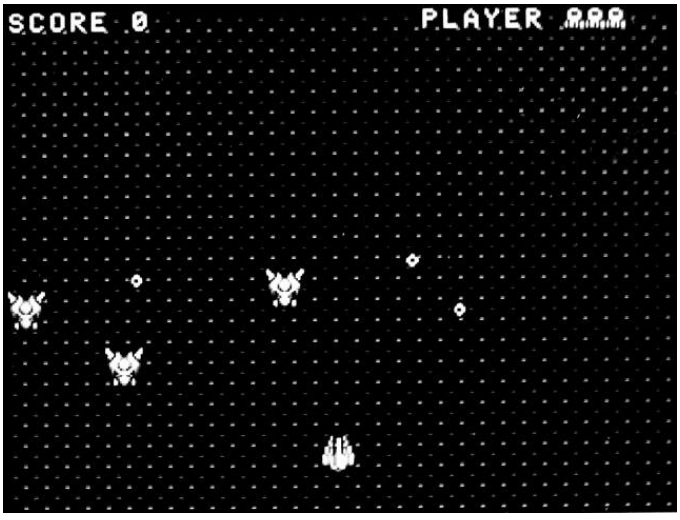
```
64 '--- ﾙｰﾌﾟ ---
65 @LOOP
 （中略）
88 FOR ML=0 TO 2
89 IF RND(100)==0 AND TEKI(ML)==1 AND M
 ISL(ML)==0 THEN GOSUB @MISLFIRE
90 IF MISL(ML)==1 THEN GOSUB @MISLMOVE
91 NEXT ML
92
```

ミサイル発射サブルーチン、ミサイル移動サブルーチンを、配列変数を使って3つのミサイルを動作させるプログラムにします。

```
169 '--- ミサイル ハッシャ ---
170 @MISLFIRE
171
172 MISL(ML)=1
173 MX(ML)=TX(ML)
174 MY(ML)=TY(ML)
175 A=ATAN(PY-TY(ML),PX-TX(ML))
176 IF A<0.5 THEN A=0.5
177 IF A>2.6 THEN A=2.6
178 MDX(ML)=COS(A)
179 MDY(ML)=SIN(A)
180 SPOFS 15+ML,MX(ML),MY(ML)
181 RETURN
182
183 '--- ミサイル イドウ ---
184 @MISLMOVE
185
186 MX(ML)=MX(ML)+MDX(ML)
187 MY(ML)=MY(ML)+MDY(ML)
188 IF MY(ML)>191 THEN MISL(ML)=
0:MY(ML)=-20
189 SPOFS 15+ML,MX(ML),MY(ML)
190 IF SPHITSP(0,15+ML) THEN MIS
S=1
191 RETURN
```

ML で指定したミサイルに関する処理をするように書き換え

プログラムを実行してみましょう。
敵とともに、ミサイルも3発まで動くようになります。



●BGM を付ける

これまで全く音が無い状態だったので、まずは BGM（音楽）を付けてみましょう。
ゲームがスタートしたら、音楽が流れるようにします。

リプレイが始まる所で、「BGMPLAY」（ビージーエムプレイ）命令を入れます。

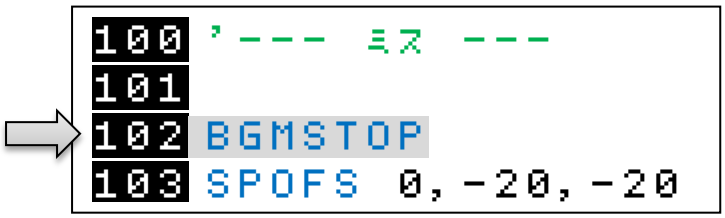
プログラムを実行すると、ゲーム開始から曲が流れます。

BGMPLAY 命令の文法は、以下のとおりです。

```
BGMPLAY     23
            曲番号      曲番号は 0～29。
```

曲は 30 曲あります。一覧リストは→を見てください。
曲番号を変えると、違う曲になります。

このままだと、曲がずっと流れっぱなしなので、ミスした時は止めるようにしましょう。



BGM を止めるには、「BGMSTOP」（ビージーエムストップ）命令を使います。

番号	説明
0	軽快な曲
1	湿った暗い感じの曲
2	緊張感高まる曲
3	激しくアップテンポな曲
4	スタートジングル
5	クリアジングル
6	ゲームオーバー
7	メニューセレクト
8	結果発表
9	スタッフロール
10	スタッフロール　その 2
11	時代劇ゲーム風
12	軽快なマーチバンド風
13	激しいロック調
14	軽快な曲　その 2
15	WOND
16	考え中
17	WOND2
18	未来系
19	BAL
20	BAL_2
21	スパイ系
22	SCI
23	シューティング
24	パッド
25	SEN
26	ピュア
27	ROA
28	CUR
29	FIG

さらに、ゲームオーバーになった時の BGM も付けてみましょう。

●効果音を付ける

ビームを発射する時、敵に当たった時などの効果音を付けてみましょう。
効果音を出すには、「BEEP」（ビープ）命令を使います。
文法は以下のとおりです。

```
BEEP    0      ,0     ,127    ,64
        波形   ピッチ  音量    パン
        番号                   ポット
```

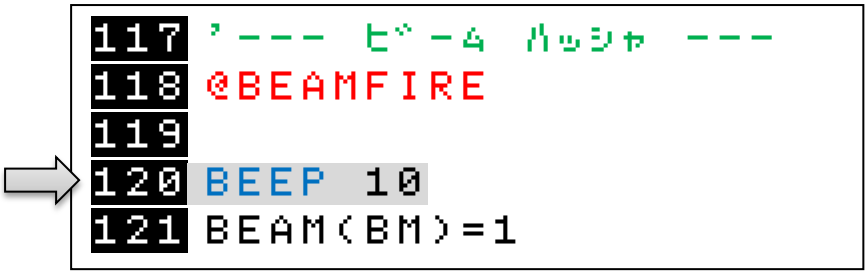
波形番号	0～69。詳しくは下の表を見てください。省略時は 0。
ピッチ	音の高さ。0＝標準。-8192＝2 オクターブ下、8192＝2 オクターブ上。
音量	0（無音）～127（最大）。
パンポット	ステレオスピーカーを使って、音を左右にふる。 0＝左、64＝中央、127＝右。

※それぞれの値は省略可能。

波形番号（0～69）は以下のとおりです。いろいろな効果音の他、楽器の音もあります。

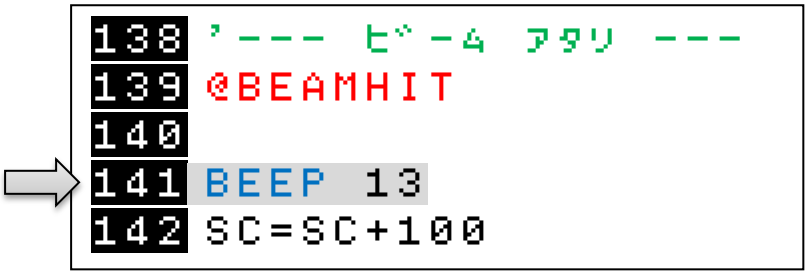
0	BEEP	18	シンセブラス	36	AUTO	54	ダンス HH
1	ノイズ	19	シンセベース	37	キラ	55	ヒット
2	カーソル移動	20	ギター	38	ESC	56	ティンバレス
3	決定	21	オルガン	39	バンジョー2	57	チャイニーズシンバル
4	キャンセル	22	ピアノ	40	スクラッチ	58	ミニシンバル
5	上昇	23	カウベル	41	ギター2	59	シェーカー
6	下降	24	タム	42	オルガン 2	60	鈴
7	コイン	25	シンバル	43	ピアノ 2	61	太鼓
8	ジャンプ	26	オープンハイハット	44	PASS	62	シンセ
9	着地	27	クローズハイハット	45	UP2	63	かっこう
10	発射	28	ハンドクラップ	46	録音	64	パフ！
11	ダメージ	29	リムショット	47	シンセタム	65	nohkan
12	金属	30	スネアドラム	48	カウベル 2	66	humandr1
13	爆発	31	バスドラム	49	metro	67	humandr2
14	叫び声	32	OK2	50	tri	68	犬
15	ブレーキ	33	BALL	51	コンガ	69	猫
16	バンジョー	34	和風	52	ダンス BD		
17	シンセストリングス	35	VOLT	53	ダンス SD		

まず、ビームを発射する時の効果音を付けてみましょう。

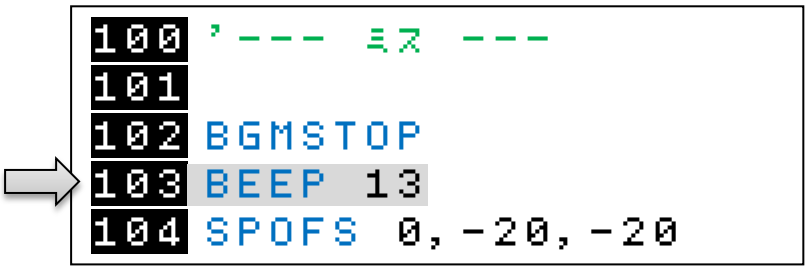


プログラムを実行してみましょう。ビームを発射する時に発射音が出ます。

次に、ビームが敵に当たった時に、爆発音が出るようにしましょう。



そして、敵と自機が当たってミスになった時も、爆発音が出るようにしましょう。



プログラムを実行してみましょう。ビームと敵、敵と自機が当たった時に爆発音が出ます。
効果音が付くと、ゲームが盛り上がりますね。

BEEP 命令の波形番号を変えると、音が変わります。
いろいろ試してみてください。

●自機の爆発のアニメーション

今は自機と敵が当たった場合、ビームが敵に当たった場合は、そのまま消えていますが、爆発のアニメーションをつけてみましょう。

キャラクターを表示するスプライトに、爆発のアニメーションの素材があります。
スプライト番号 248 番から 8 コマあります。

まず、自機に敵が当たった時の爆発アニメーションです。
ミスの処理をするプログラムで、SPOFS 命令で自機を消していた部分を書き換えます。

```
100 '--- ミス ---
101
102 BGMSTOP
103 BEEP 13
104 SPCHR 0,248
105 SPANIM 0,8,10,1
106 PL=PL-1
```

自機を爆発のキャラに変える

8 コマ分アニメを表示する

新しく、スプライトのキャラクターを変更する「SPCHR」（エスピーキャラクター）命令と、スプライトをアニメーションさせる「SPANIM」（エスピーアニメ）命令が出てきました。
文法は以下のとおりです。

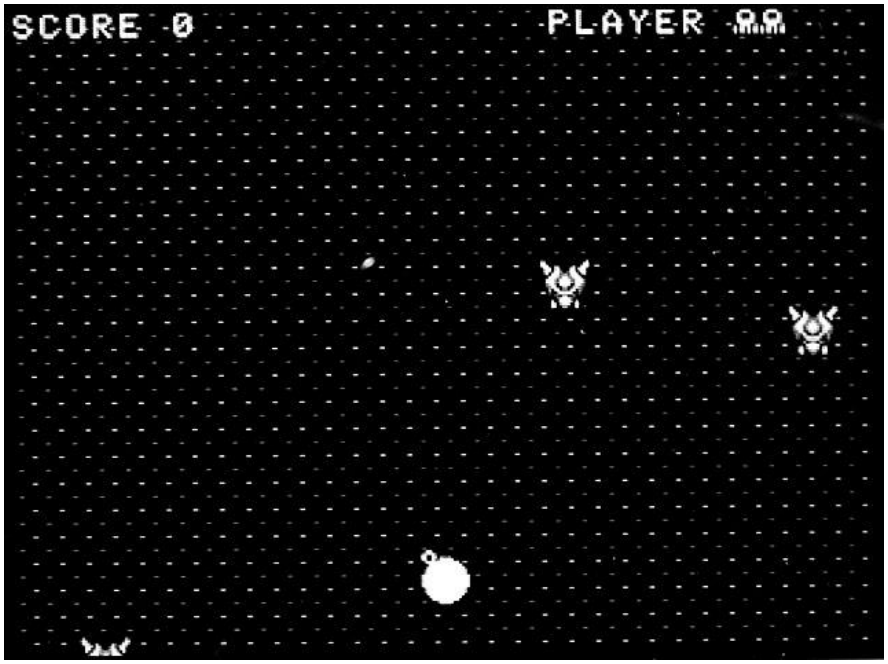
```
SPCHR  0      ,248     ,2       ,0       ,0       ,0
       管理   スプライト パレット 横反転   縦反転   優先順位
       番号   番号     番号
```

管理番号	キャラクターを変えたい管理番号。0～99。
スプライト番号	新しく設定するスプライト番号。
パレット番号（省略可）	色のパレット番号を指定します。0～15。
横反転（省略可）	0＝なし、1＝横反転（左右逆に表示）
縦反転（省略可）	0＝なし、1＝縦反転（上下逆に表示）
優先順位（省略可）	スプライトを表示する優先順位 0＝コンソール（文字表示）の前 1＝手前の BG 面（背景表示面）より前 2＝2 枚の BG 面の間 3＝奥の BG 面の後ろ

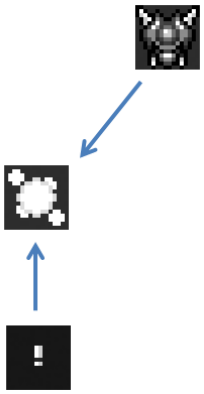
```
SPANIM    0        ,8        ,10       ,1
          管理     アニメ    表示      ループ
          番号     枚数      時間      回数
```

管理番号	アニメーションさせたい管理番号。0～99。
アニメ枚数	アニメに使うコマの枚数。
表示時間	1 コマ当たりの表示時間。1/60 秒単位。
ループ回数（省略可）	0＝無限ループ、1～＝アニメのくり返し回数 （省略時は無限ループ）

プログラムを実行してみましょう。
敵やミサイルが自機に当たると、爆発アニメーションが表示されます。



●敵の爆発のアニメーション



自機の次は、敵にビームが当たった時に爆発するアニメーションをつけてみましょう。
自機の爆発と違って、敵の爆発の場合はいろいろ工夫が必要です。
なぜなら、自機がミスになって爆発する時は、

ゲームの進行を止めて、自機の爆発アニメだけ表示すればいい

のですが、敵にビームが当たって爆発する時は、

敵の爆発アニメを表示している間も、ゲームを進行させて、
他の物はこれまでどおり動かさないといけない

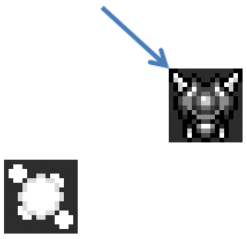
からです。

① どこで爆発アニメを表示するか？
敵を動かす移動サブルーチンの中で、爆発アニメを表示する処理を少しずつします。

② 爆発中の動きをどうするか？
敵は空中を飛行（移動）している訳ですが、ビームが当たって爆発した時は、移動を止めてその場で爆発アニメを表示しないといけません。

爆発中は、移動を止めて
その場でアニメを表示する

③ 敵の「存在」の扱いをどうするか？
これまでは敵にビームが当たって消える時に、敵の状態変数「TEKI()」を 0 にして消えたことにして、次の登場に備えていました。
爆発アニメを入れると、爆発が終わったかどうかをいちいちチェックして、終わるまでは敵が存在していることにしないといけません。そうしないと、爆発が終わらないうちに次の敵が出てしまって、敵の数がオーバーして処理できなくなります。



爆発中は、まだ存在していることにしないと、次の敵が登場してしまう。
3機より増えると、処理できなくなる。

④ 敵が爆発中にさらにビームが当たったらどうするか？
敵が爆発中は、ビームの当たり判定を無しにしないといけません。そうしないと、連射したビームが爆発中の敵に当たると、その分得点が入ってしまいます。

敵が爆発中にまたビームが当たると、得点が入ってしまっては困る。
爆発中は当たり判定を無しにする。

これらの問題を解決するために、まず敵の状態変数「TEKI()」の処理を変えます。
これまでは

TEKI()=0　敵がいない（画面に登場していない）
TEKI()=1　敵が存在する

の 2 種類だったのですが、これを以下の 3 種類に変えます。

TEKI()=0　敵がいない
TEKI()=1　敵が存在する
TEKI()=2　敵が爆発中

まず、メインループのサブルーチン呼び出しの部分を、上の状態変数 TEKI()の変更に合わせて書き換えます。

```
65 '--- ﾙｰﾌﾟ ---
66 @LOOP
 （中略）
80 FOR TK=0 TO 2
81 IF RND(100)==0 AND TEKI(T
K)==0 THEN GOSUB @TEKIDERU
82 IF TEKI(TK)>=1 THEN GOSU
B @TEKIMOVE
83 NEXT TK
```

「TEKI(TK)>=1」と「1 以上だったら」の条件に変えて、1 か 2 の時に敵の移動サブルーチン @TEKIMOVE を呼びます。

次に、敵とビームの当たり判定の所で、爆発アニメの表示を開始して、状態変数 TEKI()を 2（爆発中）に、敵の移動量を 0 に変えます。

```
140 '--- ﾋﾞｰﾑ ｱﾀﾘ ---
141 @BEAMHIT
 （中略）
148 SPOFS 1+BM,-20,-20
149 TEKI(TK)=2
150 SPCHR 11+TK,248
151 SPANIM 11+TK,8,10,1
152 TDX(TK)=0
153 TDY(TK)=0
154 RETURN
```

状態変数を 2 にする
スプライトを爆発キャラに
アニメ 8 コマ分を設定
移動量を 0 に

次に、敵の移動サブルーチンで、「爆発アニメが終わったら、状態変数 TEKI()を0にして、敵を画面から消す」部分を追加します。

```
167 '--- テキ イドウ ---
168 @TEKIMOVE
170
171 IF TEKI(TK)==2 AND SPCHK(11+TK)==0 THEN TEKI(TK)=0:SPCHR 11+TK,137:SPOFS 11+TK,-20,-20:RETURN
172 TX(TK)=TX(TK)+TDX(TK)*SIN(TY(TK)/32)*1.5
```

新しく「SPCHK」（エスピーチェック）という関数が出てきました。
これは、スプライトがアニメなどの補完動作をしているかをチェックします。

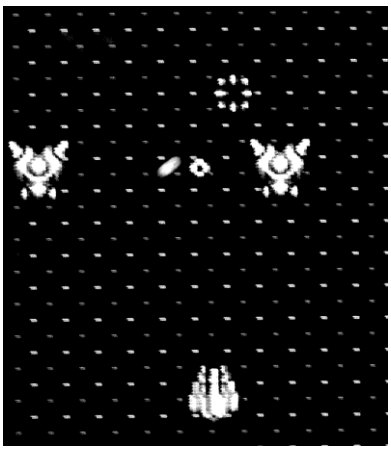
SPCHK(0)
管理番号

管理番号	チェックするスプライトの管理番号。0～99。	
返ってくる値	位置の補間	補間中=1、補間していない=0
	角度の補間	補間中=2、補間していない=0
	大きさの補間	補間中=4、補間していない=0
	アニメの補間	補間中=8、補間していない=0

※2種類以上の補間動作をしている時は、それぞれの値の足し算になる。

爆発中はアニメ以外の動作をしていないので、SPCHK()=0なら爆発のアニメが終わっているとみなせます。
これを使って、「TEKI()が2（爆発中）」かつ「爆発のアニメが終わっている」時は、「敵の状態変数 TEKI()を0にする」「SPCHR 命令でスプライトを敵のキャラに戻す」「SPOFS 命令でスプライトの位置を画面の外にする」処理をして、RETURN でメインループに戻っています。

これでプログラムを実行してみましょう。
敵にビームが当たると、爆発のアニメが表示されます。



このままだと、敵が爆発中にさらにビームが当たると得点になってしまうので、ビームと敵の当たり判定の部分を変更しましょう。

```
129 '--- ﾋﾞｰﾑ ｲﾄﾞｳ ---
130 @BEAMMOVE
  （中略）
135 FOR TK=0 TO 2
136 IF SPHITSP(1+BM,11+TK) AND TE
KI(TK)==1 THEN GOSUB @BEAMHIT
137 NEXT TK
138 RETURN
```

「ビームと敵のスプライト同士が当たった時」かつ「敵が登場している」（TEKI()=1）の時に、ビームが当たった処理をするサブルーチンを呼ぶようにします。
こうすると、敵が爆発中（TEKI()=2）の時は条件が成り立たないので、ビームが当たらなくなります。

プログラムを実行してみましょう。
敵が爆発中にもう一発ビームが来ても、当たらずに素通りするようになります。

●いろいろな工夫

これでシューティングゲームが一通り完成しました。
あとは自分のオリジナルのルールやキャラクターを追加して、改造するといいでしょう。

- 自機が左右だけでなく、上下にも動けるようにする。
- ビームを真上だけでなく、斜め方向にも発射できるようにする。
- 得点がある点数を超えると、自機が1機追加される。（自機ボーナス）
- 出てくる敵の種類を増やす。動き方を変える。
- ある程度敵を倒したら、ボス敵が出る。何発かビームを当てないと倒せないようにする。
- ボス敵を倒したら、次のステージへ行く。（背景が変わる、敵の数が増えるなど）
- ボーナスカプセルを登場させて、それを取るとボーナス得点が入る。あるいはビーム砲が強化される（連射数が増える、斜めにも打てるようになるなど）。
- 壁、岩などの障害物を登場させて、自機が当たるとミスになる。ビームが当たるとそこで止まるようにする。