プチコンでシューティングゲームを作る（２）

●ビームを発射する

いよいよシューティングゲームらしく、自機からビームを発射させてみましょう。
まず、ボタンを押して自機からビームを発射するプログラムを作ります。

★ビームの初期設定

ビームのキャラクターは、スプライトの 223 番を使います。
自機の設定部分の続きに、ビームの初期設定のプログラムを追加します。

```
23
24 '--- ﾋﾞｰﾑ ---
25
26 SPSET 1,223,2,0,0,2
27 BX=-20
28 BY=-20
29 SPOFS 1,BX,BY
30 BEAM=0
31
```

1 番のスプライトに
ビームのキャラを設定

ビームのx座標を BX、y 座標を BY に設定
座標を-20 にして、画面の外におく

ビームの状態を表す変数BEAMを用意して、ビームがある＝1、ビームが無い＝0 と設定。あとでビームを動かす時に、ビームがあるのか無いのかを区別する。

★ビームの発射

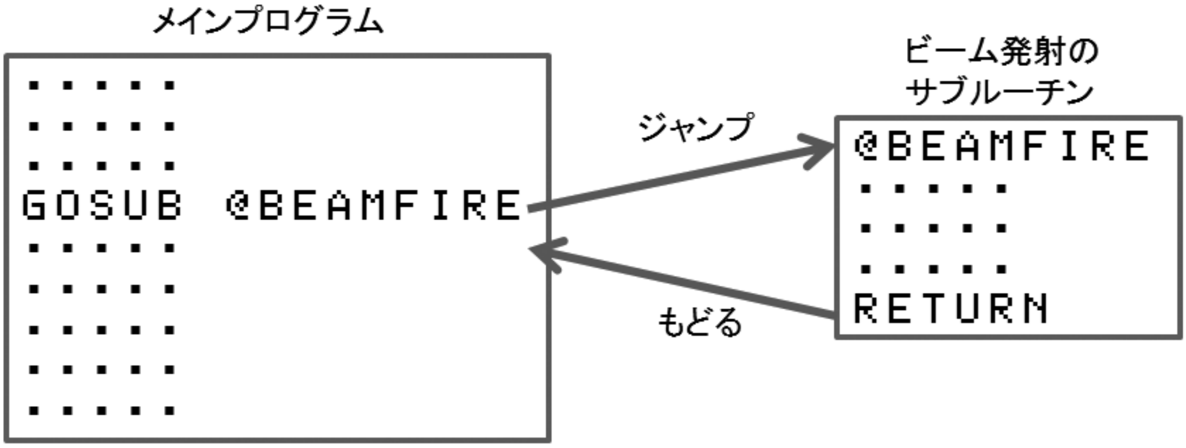
「A ボタンが押されたら、ビームを発射する」ようにしてみましょう。
BUTTON 関数で十字キーを読み取って自機を左右に動かしている所で、Aボタンを読み取るプログラムを追加します。

```
36 '--- ﾙｰﾌﾟ ---
37 @LOOP
38
39 B=BUTTON()
40 IF B==4 AND PX>8 THEN PX=PX-2
41 IF B==8 AND PX<248 THEN PX=PX+2
42 IF B==16 AND BEAM==0 THEN GOSUB
@BEAMFIRE
```

キー入力の変数 B が 16 と等しい（A ボタンが押された）、かつ、状態変数 BEAM が 0 と等しい（ビームが画面上に無い）時に、ビーム発射の処理をします。

★サブルーチンを使う

新しく「GOSUB」（ゴーサブ）命令が出てきました。これはサブルーチンを呼び出す命令です。ビームを発射するプログラムを、メインのプログラムとは別の「**サブルーチン**」にします。



メインプログラムからは「GOSUB」命令でサブルーチンへジャンプし、サブルーチンからは「RETURN」（リターン）命令でもどります。もどった後は、GOSUB 命令の続きへプログラムの処理が移ります。
サブルーチンに分けると、プログラムがすっきりしてわかりやすくなります。また、何度も同じサブルーチンを呼び出して使うことができます。

ビームを発射するプログラム（サブルーチン）を、プログラムの最後に追加します。

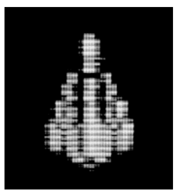
```

'--- ﾋﾞｰﾑ ﾊｯｼｬ ---
@BEAMFIRE

BEAM=1
BX=PX-7
BY=PY-16
SPOFS 1,BX,BY
RETURN
```

- BEAM=1：「ビームあり」の状態にする
- BX=PX-7／BY=PY-16：ビームのx座標、y座標を、自機の座標から設定
- SPOFS 1,BX,BY：ビームのスプライトを移動
- RETURN：メインプログラムへもどる

プログラムを実行してみましょう。
A ボタンを押すと、自機のすぐ上にビームが表示されます。
まだ発射するプログラムしか作っていないので、
それ以上は動きません。



●たくさんの物を同時に動かす

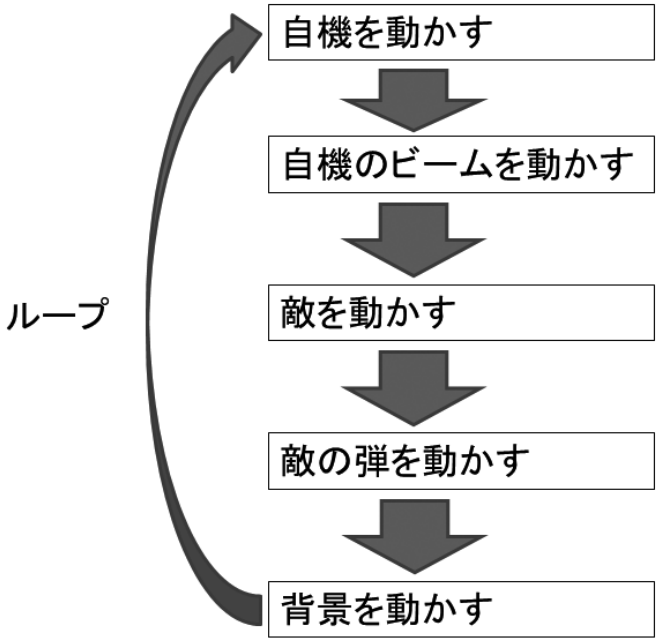
自機から発射したビームを、画面の上の方へ動かさないといけません。
その前に、プログラム全体の構成を考えてみます。

ここまで作ったプログラムは、自機と背景だけを動かしていました。
これから作っていくシューティングゲームは、自機の他に、

- 自分が打ったビーム
- 敵
- 敵の弾

などの「たくさんの物を」「同時に動かす」プログラムにしないといけません。

そのためのプログラムの流れを図で書くと、右のようになります。
（このような流れ図を「フローチャート」と言います）
それぞれのキャラを順番に少しずつ動かして、全体をループしてくり返すようにします。



今は、一番上の「自機を動かす」と、一番下の「背景を動かす」だけができている状態です。
これから、他のキャラを動かすプログラムを追加していきます。

ループの中にキャラを動かすプログラムを直接書いてしまうと、ループが長くなって見づらくなるので、それぞれのプログラムをサブルーチンにして、GOSUB 命令で呼び出すようにするといいでしょう。

●ビームを動かす

先ほど発射したビームを、上へ動かすプログラムを作ります。
ビームを動かすプログラムをサブルーチンにして、メインのループから呼び出すようにします。

```
36 '--- ループ ---
37 @LOOP
   （中略）
43 SPOFS 0,PX,PY
44 IF BEAM==1 THEN GOSUB @BEAMMOVE
45 BGY=BGY-1
```

IF命令を使って、状態変数 BEAM が1だったら（＝ビームが存在したら）ビームを動かすサブルーチン「@BEAMMOVE」を呼ぶようにしています。

次に、ビームを動かすサブルーチンを、プログラムの最後に追加します。
ビームは上へ飛んでいくので、y座標（BY）を減らして表示します。

```
60 '--- ビーム イドウ ---
61 @BEAMMOVE
62
63 BY=BY-2
64 IF BY<0 THEN BEAM=0:BY=-20
65 SPOFS 1,BX,BY
66 RETURN
```

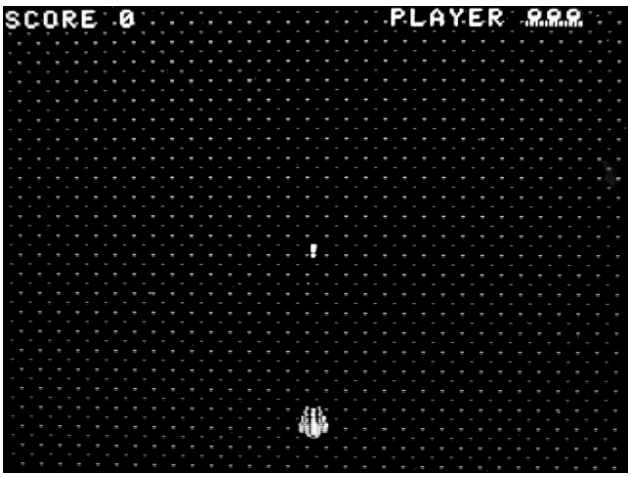
ビームのy座標を2減らす

もしビームが上に達したら、ビーム無し状態にして、画面の外へ移動

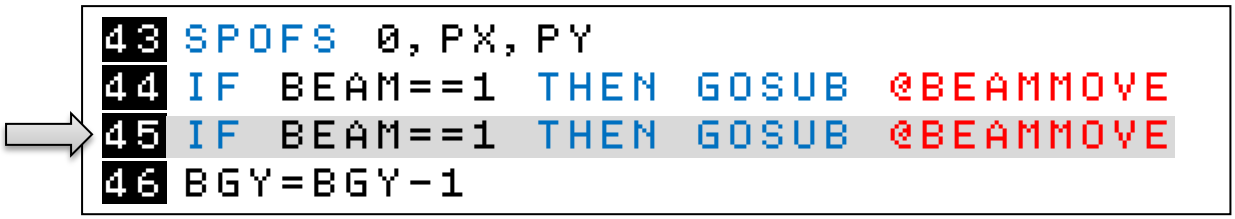
ビームを表示

メインへもどる

プログラムを実行してみましょう。
Aボタンを押すと、ビームが発射されて上へ飛んでいきます。



このままだと、ビームの移動速度がちょっと遅いです。
例えば、63 行目の「BY=BY-2」を「BY=BY-4」に変えれば、速度は2倍になりますが、あとで敵を動かしたり、敵に当たったかどうかのチェックをするので、ここはそのままにします。
その代わり、メインのループでビーム移動サブルーチンを呼び出す部分を、2回呼び出すようにします。



```
43 SPOFS 0,PX,PY
44 IF BEAM==1 THEN GOSUB @BEAMMOVE
45 IF BEAM==1 THEN GOSUB @BEAMMOVE
46 BGY=BGY-1
```

このように、サブルーチンを使うと簡単に同じ処理を2回呼ぶことができるので、便利です。

●キー入力部分の改良

今のプログラムだと、自機を左右に動かしながらAボタンを押しても、ビームを発射できません。これは「2つのキーが同時に押された時」を考えていないからです。
キー入力をチェックするプログラムを、もう一度見てみます。

```
39 B=BUTTON()
40 IF B==4 AND PX>8 THEN PX=PX-2
41 IF B==8 AND PX<248 THEN PX=PX+2
42 IF B==16 AND BEAM==0 THEN GOSUB
@BEAMFIRE
```

例えば、40 行目「IF B==4 THEN～」は、十字キーの左が押されているかをチェックしています。しかし、十字キーの左とAボタンが同時に押されると、B の値は 4＋16＝20 になるので、自機の移動（40 行目）もビームの発射（42 行目）も、何もせずに通り過ぎてしまいます。

これを防ぐには、AND を使って、IF命令の 3 行を以下のように変更します。

```
40 IF (B AND 4)==4 AND PX>8 THEN PX
=PX-2
41 IF (B AND 8)==8 AND PX<248 THEN
PX=PX+2
42 IF (B AND 16)==16 AND BEAM==0 TH
EN GOSUB @BEAMFIRE
```

こうすると、他のキーが押されているかいないかに関わらず、十字キー左、右、A ボタンをちゃんとチェックできます。
（AND で「論理演算」〈ろんりえんざん〉という計算をしていますが、難しいので説明しません）

●敵を表示する

ビームが発射できるようになったので、いよいよ敵を登場させてみましょう。
敵のキャラクターは、スプライトの 137 番を使います。

まず、敵のキャラクターの初期設定を、「ビーム」と「ハイケイ」（背景）の間に追加します。
内容はビームの設定とほぼ同じなので、コピー＆ペーストで作っても構いません。

```
31
32 '--- テキ ---
33
34 SPSET 11,137,2,0,0,2
35 TX=-20
36 TY=-20
37 SPOFS 11,TX,TY
38 TEKI=0
39
```

管理番号 11 番に、137 番のスプライトを設定

敵のx座標 TX、y 座標 TY を画面の外に設定

スプライトの座標を TX、TY に

敵の存在を表す変数 TEKI を 0 に

ビームと同様に、「敵を登場させるサブルーチン」と、「敵を動かすサブルーチン」の2つを作り、メインのループで、それぞれのサブルーチンを呼びます。

```
44 '--- ﾙｰﾌﾟ ---
45 @LOOP
  （中略）
52    （わかりやすくするために、1 行空ける）
53 IF BEAM==1 THEN GOSUB @BEAMMOVE
54 IF RND(100)==0 AND TEKI==0 THEN
GOSUB @TEKIDERU
55 IF TEKI==1 THEN GOSUB @TEKIMOVE
56 IF BEAM==1 THEN GOSUB @BEAMMOVE
57    （わかりやすくするために、1 行空ける）
```

100 分の 1 の確率で敵が出る

敵が存在するなら、動かす

54 行目では、乱数（らんすう）の RND（ランダム）関数を使って、ある一定の確率で敵が登場するようにしています。乱数は、実行する度にランダムに違う値が出てくる数です。
RND 関数の文法は、以下のとおりです。

RND(　　100)
　　最大値

最大値	乱数の最大値。0～最大値－1までの乱数が発生する。

54 行目では、「RND(100)==0」という条件で、100 分の 1 の確率で敵が登場する設定にしています。この値を変えると、敵が登場する確率が変わります。
また、AND 条件で「TEKI==0」として、敵が画面にいない時だけ、敵を登場させるサブルーチン「@TEKIDERU」を呼んでいます。

55 行目は、画面に敵がいる時だけ（TEKI==1）、敵を動かすサブルーチン「@TEKIMOVE」を呼んでいます。

そして、プログラムの最後に、敵を登場させるルーチンを追加します。
先に作った、ビームを発射するサブルーチンと似ています。

```
80
81 '--- テキ トウジョウ ---
82 @TEKIDERU
83
84 TEKI=1
85 TX=RND(240)
86 TY=0
87 SPOFS 11,TX,TY
88 RETURN
```

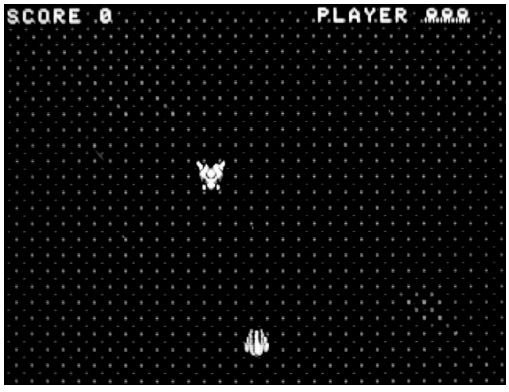
敵の状態変数を 1 に
敵のx座標を乱数で指定
敵のy座標は画面一番上に
敵を TX,TY に表示

続いて、敵を動かすルーチンを追加します。
先に作った、ビームを動かすサブルーチンと似ています。

```
89
90 '--- テキ イドウ ---
91 @TEKIMOVE
92
93 TY=TY+1
94 IF TY>191 THEN TEKI=0:TY=-20
95 SPOFS 11,TX,TY
96 RETURN
```

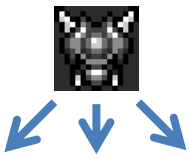
敵のｙ座標を 1 増やす
敵が画面の下に着いたら、状態変数を 0 にして、画面の外へ
敵を表示

プログラムを実行してみましょう。
敵が上からときどき飛んできます。



●敵の動きの工夫

今のままだと、敵がいつも真っ直ぐ下へ飛んでくるので、おもしろくありません。
敵の動きをいろいろ変えてみましょう。



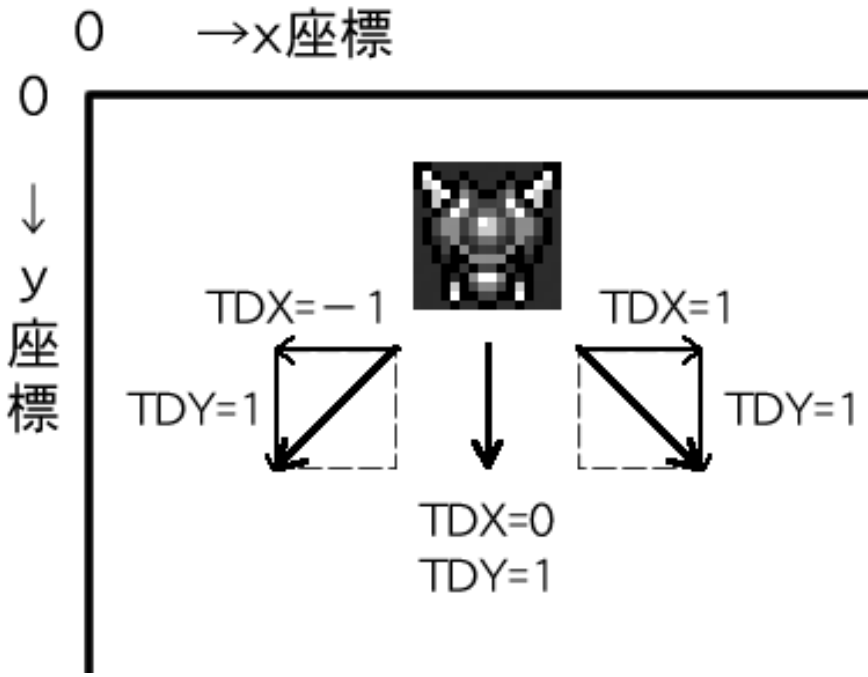
敵を真っ直ぐ下だけではなく、斜め左、斜め右にも飛ばしてみましょう。
敵の動きの変化量を表す変数 TDX、TDY を新しく使います。
まず、敵を登場させるサブルーチンで、TDX、TDY を乱数で決めます。

```
81 '--- テキ トウジョウ ---
82 @TEKIDERU
83
84 TEKI=1
85 TX=RND(240)
86 TY=0
87 TDX=RND(3)-1
88 TDY=1
89 SPOFS 11,TX,TY
90 RETURN
```

x 方向の変化量を乱数で
−1、0、1 のどれかにする

y 方向の変化量は 1

87 行目「TDX=RND(3)-1」は、まず「RND(3)」で「0,1,2」の乱数が作られます。そこから 1 を引くので、TDX は「-1,0,1」のどれかになります
88 行目では、TDY を 1 にしています。
敵の移動する方向と、移動量 TDX、TDY との関係は、以下の図を見てみてください。



そして、敵を動かすサブルーチンで、TDX、TDY の分だけ敵を移動します。

```
 91
 92 '--- テキ イドウ ---
 93 @TEKIMOVE
 94
 95 TX=TX+TDX
 96 TY=TY+TDY
 97 IF TY>191 THEN TEKI=0:TY=-20
 98 IF TX<-16 THEN TX=255
 99 IF TX>255 THEN TX=-16
100 SPOFS 11,TX,TY
101 RETURN
```

x 座標を TDX だけ、
y 座標を TDY だけ変化させる

敵が画面の左はじへ行ったら右はじから出てくる。
画面の右はじへ行ったら、左はじから出てくる。

プログラムを実行してみましょう。
敵が左・真下・右の 3 方向へ移動します。

さらに、x 座標 TX を変化させる行を、以下のように変えてみてください。

```
 92 '--- テキ イドウ ---
 93 @TEKIMOVE
 94
 95 TX=TX+TDX*SIN(TY/32)*1.5
```

プログラムを実行してみましょう。
敵が左右に振れながら飛んでくるようになります。

ここでは三角関数 SIN(サイン)を使っています。
三角関数は高校の数学で習いますが、難しいのでここでは説明しません。
「/32」や「*1.5」の数字を増やしたり減らしたりすると、動きが変わります。いろいろ試してみてください。

●ビームと敵の当たり判定

敵の動きができたので、いよいよビームが敵に当たった時の処理を作ります。
ビームを動かすサブルーチン「@BEAMMOVE」の中で、敵と当たったかどうか判定します。

```
73 '--- ﾋﾞｰﾑ ｲﾄﾞｳ ---
74 @BEAMMOVE
75
76 BY=BY-2
77 IF BY<0 THEN BEAM=0:BY=-20
78 SPOFS 1,BX,BY
79 IF SPHITSP(1,11) THEN GOSUB @BEAMHIT
80 RETURN
```

キャラクター（スプライト）同士が当たったかどうかの「当たり判定」には、SPHITSP（エスピーヒットエスピー）関数を使います。

```
SPHITSP(      1          ,11)
          管理番号    相手管理番号
```

2つのスプライトが重なると、SPHITSP 関数の値が「真」（TRUE）になるので、IF 命令の THEN 以降の文が実行されます。
ここでは、ビームが当たった処理をするサブルーチン「@BEAMHIT」を呼び出しています。
この続きに、サブルーチン「@BEAMHIT」を追加します。

```
81
82 '--- ﾋﾞｰﾑ ｱﾀﾘ ---
83 @BEAMHIT
84
85 SC=SC+100
86 LOCATE 6,0
87 PRINT SC
88 BEAM=0
89 SPOFS 1,-20,-20
90 TEKI=0
91 SPOFS 11,-20,-20
92 RETURN
93
```

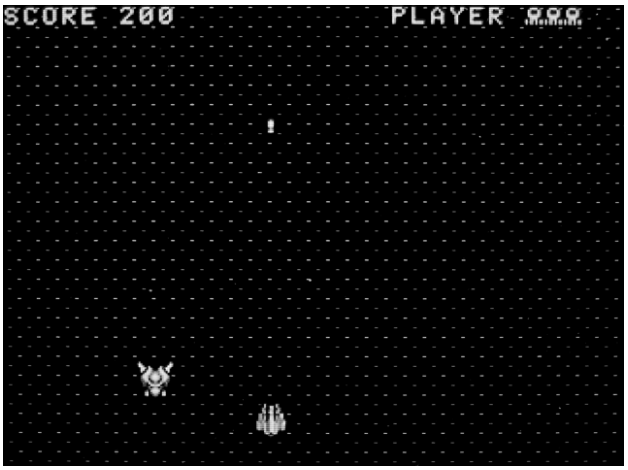
得点を 100 点増やす

画面に得点を表示

ビーム状態変数を 0 にして、
画面の外へ移動

敵の状態変数を 0 にして、
画面の外へ移動

プログラムを実行してみましょう。
ビームが敵に当たると、ビームと敵が消えて、
得点が 100 点入ります。



●敵と自機の当たり判定

敵が自機に当たった場合は、ミスになるようにしましょう。
敵を動かすサブルーチン「@TEKIMOVE」の中で、自機と当たったかどうか判定します。
ビームの時と同じく、SPHITSP 関数を使います。

```
105 '--- テキ イドウ ---
106 @TEKIMOVE
 …(中略)…
113 SPOFS 11,TX,TY
114 IF SPHITSP(0,11) THEN MISS=1
115 RETURN
```

自機と敵の当たり判定なので、管理番号 0 と 11 の当たり判定になります。
当たった場合は、ミス状態を表す変数 MISS を 1 にしてもどります。

そして、プログラムの最初から書き換えます。
まず、ミスした後にプレイを再開できるように、画面を表示する所で、ラベル「@REPLAY」を追加します。
また、先ほど作った、ミス状態を表す変数 MISS を 0 に設定します。

```
  6 '--- ガメン ---
  7 @REPLAY
  8
  9 MISS=0
 10 ACLS
 11 BGPAGE 0
```

メインループの最後の部分を書き換えて、ミスした時の処理を追加します。
ミスした時は、変数 MISS が 1 になるはずなので、IF 命令で判断します。

```
64 IF MISS==0 THEN GOTO @LOOP
65
66 '--- ミス ---
67
68 SPOFS 0,-20,-20
69 PL=PL-1
70 LOCATE 27,0
71 PRINT "R"*PL;" "
72 WAIT 120
73 GOTO @REPLAY
```

MISS が 0 だったらミスしていないのでループ
自機の表示を消す
自機の残り数を1減らす
残り数の表示を書き換え
2 秒待つ
再プレイにもどる

新しい命令「WAIT」（ウェイト）が出てきました。これは時間待ちをする命令です。

```
WAIT        120
```

待ち時間　　→60 分の 1 秒単位で指定します。60＝1 秒です。

プログラムを実行してみましょう。
飛んできた敵が自機に当たると、ミスになって、自機が1機減ります。

このままだと自機は減っていきますが、いつまでもゲームが続いてしまいます。
自機が0になったらゲームオーバーになるようにしてみましょう。

PL が 0 より大きかったら、まだ自機があるので再プレイ

```
73 IF PL>0 THEN GOTO @REPLAY
74
75 '--- ｹﾞｰﾑ ｵｰﾊﾞｰ ---
76
77 LOCATE 7,11
78 PRINT "*** GAME OVER ***"
79 END
80
```

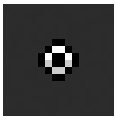
ゲームオーバーを表示

プログラムを終了

プログラムを実行してみましょう。
自機が残り 3 機でスタートして、敵に当たると自機が減っていき、残りが 0 になるとゲームオーバーになります。

●敵がミサイルを発射する

今のままだと敵が無抵抗に飛んでくるので、
敵がミサイルを発射するようにしてみましょう。
ミサイルのキャラクターは、スプライトの 190 番を使います。



ミサイルの初期設定を、プログラム前半の「テキ」と「ハイケイ」の間に追加します。
内容は敵の設定とほぼ同じなので、コピー＆ペーストで作っても構いません。

```
41
42 '--- ミサイル ---
43
44 SPSET 15,190,2,0,0,2
45 MX=-20
46 MY=-20
47 SPOFS 15,MX,MY
48 MISL=0
49
```

管理番号 15 番に、190 番のスプライトを設定

ミサイルのx座標 MX、y 座標 MY を画面の外に設定

スプライトの座標を MX、MY に

ミサイルの存在を表す変数 MISL を 0 に

敵と同様に、「ミサイルを発射するサブルーチン」と、「ミサイルを動かすサブルーチン」の2つを作り、メインのループから、それぞれのサブルーチンを呼びます。

```
54 '--- ﾙｰﾌﾟ ---
55 @LOOP
  （中略）
63 IF BEAM==1 THEN GOSUB @BEAMMOVE
64 IF RND(100)==0 AND TEKI==0 THEN
GOSUB @TEKIDERU
65 IF TEKI==1 THEN GOSUB @TEKIMOVE
66 IF BEAM==1 THEN GOSUB @BEAMMOVE
67 IF RND(100)==0 AND TEKI==1 AND M
ISL==0 THEN GOSUB @MISLFIRE
68 IF MISL==1 THEN GOSUB @MISLMOVE
69
```

もしミサイルが存在すれば、
ミサイルを動かす

100 分の 1 の確率
＆敵が存在する
＆ミサイルが存在しない
だったら、ミサイルを発射

そして、プログラムの最後に、ミサイルを発射するルーチンを追加します。
先に作った、敵を登場させるサブルーチンと似ています。

```
143
144 '--- ミサイル ハッシャ ---
145 @MISLFIRE
146
147 MISL=1
148 MX=TX
149 MY=TY
150 MDX=RND(3)-1
151 MDY=1
152 SPOFS 15,MX,MY
153 RETURN
```

ミサイルの状態変数を 1 に

ミサイルのx座標・y 座標を敵の座標から設定

ミサイルの方向を、左下・下・右下の 3 方向に、乱数で決める

ミサイルを MX,MY に表示

これに続けて、ミサイルを動かすルーチンを追加します。
先に作った、敵を動かすサブルーチンと似ています。

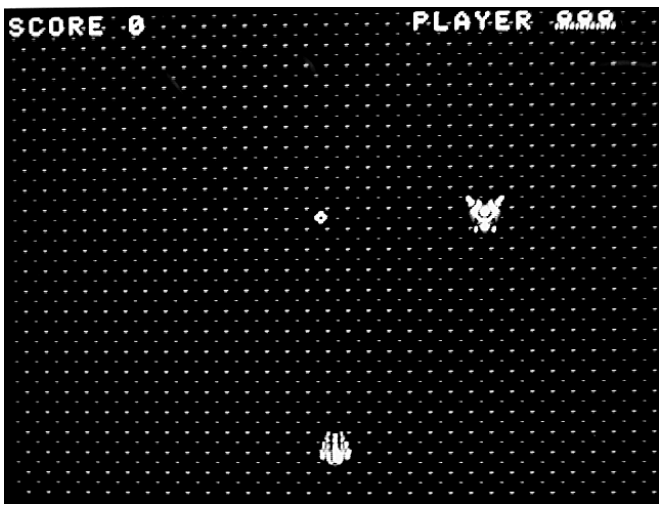
```
154
155 '--- ミサイル イドウ ---
156 @MISLMOVE
157
158 MX=MX+MDX
159 MY=MY+MDY
160 IF MY>191 THEN MISL=0:MY=-20
161 SPOFS 15,MX,MY
162 RETURN
```

ミサイルのx座標・y座標を変化させる

ミサイルが画面の下に着いたら、状態変数を 0 にして、画面の外へ

ミサイルを表示

プログラムを実行してみましょう。
敵がときどきミサイルを発射します。



●ミサイルと自機の当たり判定

ミサイルの動きができたので、ミサイルが自機に当たった時の当たり判定を作ります。
ミサイルを動かすサブルーチン「@MISLMOVE」で、自機と当たったかどうか判定します。
敵の当たり判定と同じく、SPHITSP 関数を使います。

```
154
155 '--- ミサイル イドウ ---
156 @MISLMOVE
157
158 MX=MX+MDX
159 MY=MY+MDY
160 IF MY>191 THEN MISL=0:MY=-20
161 SPOFS 15,MX,MY
162 IF SPHITSP(0,15) THEN MISS=1
163 RETURN
```

自機とミサイルの当たり判定なので、管理番号 0 と 15 の当たり判定になります。
当たった場合は、ミス状態を表す変数 MISS を 1 にしてもどります。

プログラムを実行してみましょう。
ミサイルが自機に当たると、ミスになります。

●基本は完成！

これで、シューティングゲームの基本はできました。

- 自機を動かして、ビームを打つ。
- 敵にビームが当たると、得点が入る。
- 敵またはミサイルが自機に当たるとミスになる。3 回ミスするとゲームオーバー。

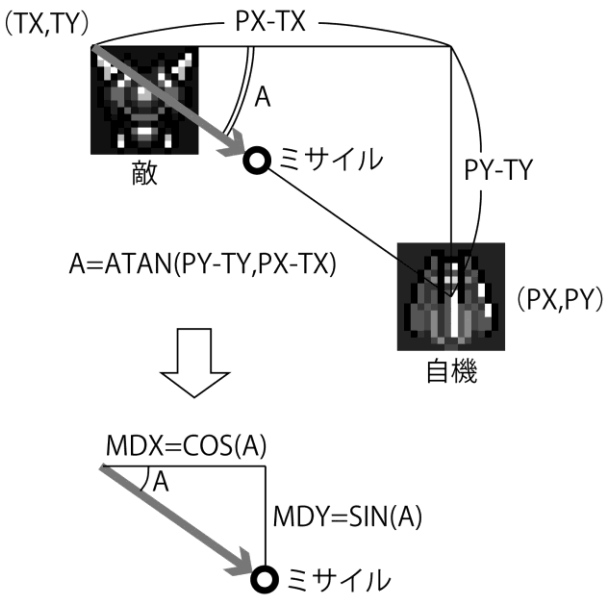
あとはゲームをさらにおもしろくするために、いろいろなルールや効果を追加していきます。

★できる人は…改造編

今のプログラムでは、ミサイルを適当な方向に
発射しているので、あまり怖くありません。
必ず自機を狙って発射するようにしてみましょう。
ミサイル発射のサブルーチンで、移動量
MDX、MDY を設定する所を変更します。

```
143
144 '---  ミサイル  ハッシャ  ---
145 @MISLFIRE
146
147 MISL=1
148 MX=TX
149 MY=TY
150 A=ATAN(PY-TY,PX-TX)
151 MDX=COS(A)
152 MDY=SIN(A)
153 SPOFS  15,MX,MY
154 RETURN
```

ここでは逆三角関数「ATAN」（アークタンジェント）、三角関数「COS」（コサイン）と「SIN」（サイン）を使っています。
三角関数は高校の数学で、逆三角関数は大学の数学で習います。
図に描くと→のようになりますが、難しいのでここでは説明しません。



プログラムを実行してみてください。
ミサイルが自機に向かって飛んできます。

このままだと、敵が真横にいる時にミサイルを発射されると、自機は横にしか動けないので逃げられません。
角度 A が真横に近い時は、方向をずらすようにしましょう。

```
150 A=ATAN(PY-TY,PX-TX)
151 IF  A<0.5  THEN  A=0.5
152 IF  A>2.6  THEN  A=2.6
153 MDX=COS(A)
```