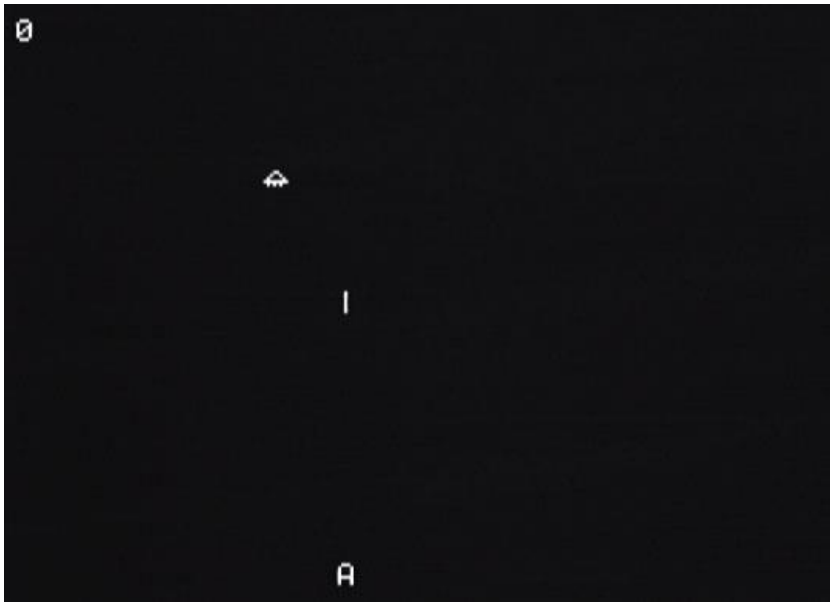
IchigoJam でシューティングゲームを作る

●今回の目標

IchigoJam で動くシューティングゲームを作ります。
自機を左右に動かして、ビームを発射して、上空の UFO を打ち落とします。



●シューティングゲームを作る手順

以下の順番で、シューティングゲームを作っていきます。

●ゲームの画面を表示する

まず、プログラムの初期設定をした後、画面にスコア（点数）を表示します。

```
10 '*SHOOTING*
20 CLS:CLV
30 LOCATE 0,0
40 PRINT S
```

コメントで、プログラムのタイトルを入れる

画面をクリア、変数をクリア

カーソルを画面左上へ移動

スコアを表示

入力できたら、「RUN」で実行してみましょう。
画面がクリアされて、左上に「0」が表示されます。

プログラムの内容を説明します。

10 行：コメントで、プログラムのタイトルを入れています。
先頭に「'」（アポストロフィ、キーボードでは Shift キーを押しながら「7」を押す）を付けると、その行はコメントとなり、何も実行されません。ですから、「'」のあとは好きな文字を書くことができます。
プログラムを後で見た時にわかりやすくするために、プログラムにいろいろコメントを入れるといいでしょう。

20 行：CLS 命令で、画面をクリアします。
その後の **CLV**（シーエルブイ）命令は、**変数**（へんすう）をクリアする命令です。
変数とは、数字を入れる入れ物（箱）と思ってください。
小学校の算数でやる「□」（四角）、中学校の数学でやる「x」と考えればいいです。
今回は、ゲームの最初に、全ての変数をクリアしています。

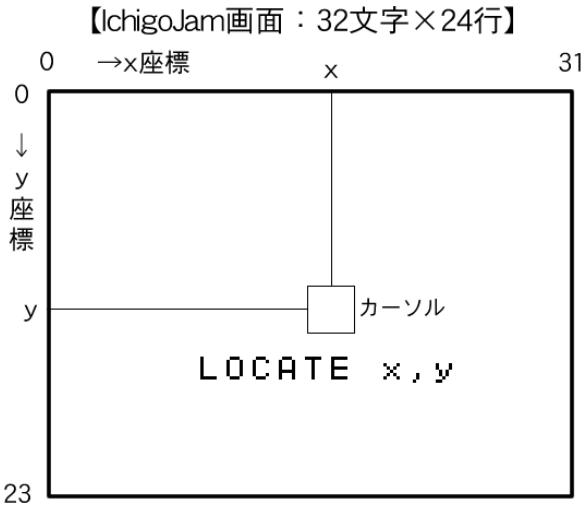
この行では、2 つの命令を「:」（コロン）で続けて書いています。「:」を使うと、複数の命令を続けて書くことができます。

30 行:LOCATE(ローケート)命令は、画面に文字を表示する位置(カーソル位置)を設定する命令です。

```
LOCATE  0  ,0
        x 座標   y 座標
```

x 座標	画面の x 座標(0～31)。
y 座標	画面の y 座標(0～23)。

IchigoJam の画面サイズは、横 32 文字×縦 24 行になっています。
横(x 方向)の座標は 0～31、縦(y 方向)の座標は 0～23 になっています。
今回は「LOCATE 0,0」として、画面左上の座標を指定しています。



40 行:「PRINT S」で、画面にスコア変数 S の値を表示しています。
最初に CLV 命令で変数をクリアしていて、S の値も 0 なので、「0」と表示されます。
「PRINT “S”」と S をダブルクォーテーションで囲むと、文字「S」を表示しますが、「PRINT S」だと、「変数 S の値を表示する」という意味になります。

LOCATE 命令の座標の数字を変えると、スコア「0」の表示位置が変わります。
いろいろ変えて試してみましょう。

●自機を表示する

スコアと同じように、自機を表示してみましょう。
以下のプログラムを追加します。

```
10 '*SHOOTING*
20 CLS:CLV
30 LOCATE 0,0
40 PRINT S
50 X=15:Y=22
60 LOCATE X,Y
70 PRINT "A";
```

50 自機の x 座標を 15、y 座標を 22 にする
60 カーソルを座標(X,Y)へ移動
70 自機を表示

プログラムを実行してみましょう。
画面中央下に「A」が表示されます。

追加したプログラムの内容を説明します。

50 行:自機の横座標の変数 X、縦座標の変数 Y に値を入れています。
「X=15」は、「X と 15 が等しい」という意味ではなく、「X に 15 を入れる」(代入する)という意味です。

60 行:LOCATE 命令で、カーソルを(X,Y)の座標へ移動します。
このように、座標を変数で指定することもできます。

70 行:PRINT 命令で、自機「A」を表示しています。
ここでは「**PRINT "A";**」と最後に「;」(セミコロン)を付けています。
PRINT 命令で文字を表示した後は、通常は自動的に改行してカーソルが次の行へ移るのですが、「;」を付けると、改行せずに次の文字位置にカーソルがとどまります。
(※改行するとあとで表示がおかしくなるので、こうしています)

今の所は、「A」を表示した後にプログラムが終了して、続いて「OK」を表示してしまいます。
これから続きのプログラムを作っていくので、今回は気にしないことにします。

50 行の変数 X,Y の値を変えれば、自機の表示位置が変わります。
いろいろ変えて試してみましょう。

●自機を動かす

自機をカーソルキー（矢印キー）で左右に動かせるようにしてみましょう。
以下のプログラムを追加します。

```
…（前略）…
60 LOCATE X,Y
70 PRINT "A";
80 '*GAMELOOP
90 LOCATE X,Y
100 PRINT " ";
110 IF BTN(LEFT)=1 THEN X=X-1
120 IF BTN(RIGHT)=1 THEN X=X+1
130 LOCATE X,Y
140 PRINT "A";
150 GOTO 80
```

コメント

自機を消す

左矢印キーが押されていたら
自機の x 座標を 1 減らす

右矢印キーが押されていたら
自機の x 座標を 1 増やす

自機を表示

80 行へもどる

プログラムを実行してみましょう。
カーソルキーの左と右を押すと、自機が左右に移動します。
このままだとプログラムがいつまでも終わらないので、ESC キーを押して止めてください。

追加したプログラムの内容を説明します。

80 行：ここからゲームの処理をするプログラムになるので、コメントを入れています。
ループしてくり返すので、「GAMELOOP」としています。

90～100 行：自機を一度消します。

110 行:自機を左へ動かす処理をします。

BTN(ボタン)関数を使って、左矢印キーが押されているかどうかを判断します。
BTN 関数の文法は以下のとおりです。

```
BTN( LEFT)
     キー指定
```

キー指定	LEFT…左矢印キー(←) RIGHT…右矢印キー(→) UP…上矢印キー(↑) DOWN…下矢印キー(↓) SPACE…スペースキー
返り値	キーが押されている＝1 キーが押されていない＝0

指定したキーが押されているとBTN 関数の値が 1、押されていないと 0 になります。
その BTN 関数の値を、**IF**(イフ)命令で判断します。

```
IF BTN(LEFT)=1 THEN X=X-1 ELSE ～
   条件式          条件が成り立つ時に実行   条件が成り立たない時に実行
```

条件式	条件を判断する式。
THEN(ゼン)の後	条件が成り立つ時に実行するプログラム。
ELSE(エルス)の後	条件が成り立たない時に実行するプログラム。ELSE 以下は省略可能。

この行は、「もし BTN(LEFT)が 1 だったら(＝左矢印キーが押されていたら)、変数 X を 1 減らす」という意味になります。
なお「X=X-1」は、「**X から 1 を引いて、それを X に入れる**」という意味です。「X と X-1 が等しい」という意味ではないので注意してください。

120 行:110 行と同じように、自機を右へ動かす(x 座標を 1 増やす)処理をします。

130～140 行:自機を新しい座標に表示します。

150 行:続けて自機を動かすため、GOTO(ゴートゥー)命令で前へ戻しています。

```
GOTO 80
     行番号
```

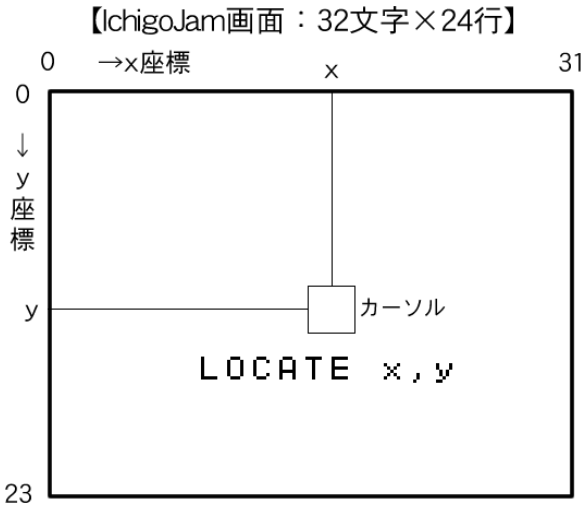
指定した行番号へ実行を移します。

このプログラムだと、以下の問題があります。

- 自機が画面左はじへ行っても左矢印キーを押し続けると、自機は動かないが、次に右へ動かそうとすると、左矢印キーを押していたのと同じ時間だけ右矢印キーを押し続けないと、右へ動かない。
- 右はじでも同様。

これは、自機の横座標 X の範囲を考えずに増やしたり減らしたりしてしまい、X がマイナスの値になってしまったりするからです。

画面サイズを考えると、自機を動かす範囲は、x 座標が 0～31 の間にしないといけません。



自機が画面からはみ出さないように、プログラムを改造します。

```
…(前略)…
80 '*GAMELOOP
90 LOCATE X,Y
100 PRINT " ";
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
130 LOCATE X,Y
140 PRINT "A";
150 GOTO 80
```

左矢印キーが押されている、
かつ、自機の x 座標が 0 より大きかったら、
x 座標を 1 減らす

右矢印キーが押されている、
かつ、自機の x 座標が 31 より小さかったら、
x 座標を 1 増やす

IF 命令の条件式に、「**AND**」(アンド)でつないで、2つの条件を入れます。

```
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
```

この条件式は、「BTN(LEFT)が 1 と等しい、**かつ**、X が 0 より大きい」という意味になります。
両方の条件が成り立った時だけ、「THEN」以下の命令が実行されます。
画面の左端に来ると、X が 0 になるので、条件が成り立たなくなり、それ以上左へ行きません。

同じように、画面の右はじでもはみ出さないように、AND 条件式を入れます。

```
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
```

「SAVE 0」でプログラムを保存しておきましょう。

●UFO を表示する

自機の次は、敵の UFO を表示しましょう。
まず、UFO の最初の設定をするプログラムを追加します。

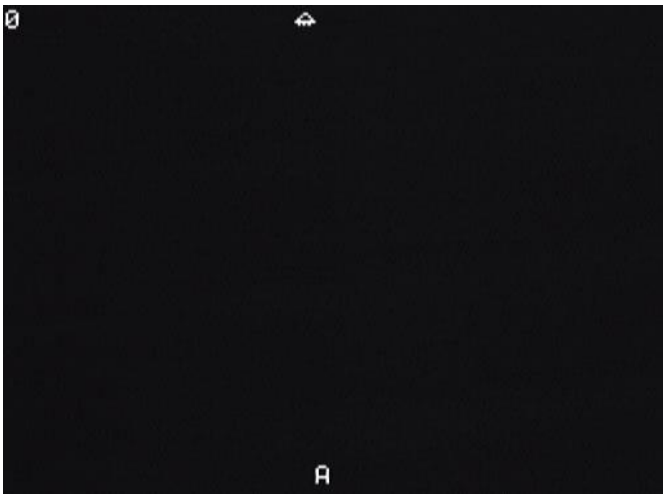
```
10 '*SHOOTING*
20 CLS:CLV
30 LOCATE 0,0
40 PRINT S
50 X=15:Y=22
60 LOCATE X,Y
70 PRINT "A";
72 U=RND(32):V=0
74 LOCATE U,V
76 PRINT CHR$(241);
80 '*GAMELOOP
90 LOCATE X,Y
```

UFO の横座標 U を乱数で指定
縦座標 V は 0 にする

UFO を画面に表示

70 行と 80 行の間に行を追加するので、行番号を 72～76 にしています。

プログラムを実行してみましょう。
UFO が画面に表示されます。
プログラムを実行するたびに、UFO の表示位置が変わります。



次ページから、プログラムの内容を説明します。

72 行:UFO の横座標 U・縦座標 V を指定します。毎回同じ場所に出てくると面白くないので、U は**乱数**(らんすう)で指定します。乱数とは、毎回違うでたらめな数字です。
乱数を取り出すには、**RND**(ランダム)関数を使います。

```
RND(  32)
```
乱数の最大値

乱数の最大値	0～最大値-1 の乱数が出てきます。

「RND(32)」と指定すると、0～31 の乱数が出てくるので、UFO の横座標 U が画面の左はじから右はじのどこかになります。
縦座標 V は、一番上(0)にします。

74～76 行:UFO を画面に表示します。ここでは UFO の文字を **CHR$**(シー・エッチ・アール・ドル、キャラクター)関数で指定しています。

```
CHR$(  241)
```
文字コード

文字コード	0～255 の数字で文字コードを指定

今回は文字コード 241 番を指定して、UFO を画面に表示しています。

IchigoJam の文字コードは、右のようになっています。

UFO(241 番)

CHR$関数の文字コードを変えると、表示されるキャラクターが変わります。試してみましょう。
自機の「A」を CHR$関数に変えて、自機のキャラクターをいろいろ変えるのも面白いでしょう。

例 `PRINT CHR$(240);`

●UFO を動かす

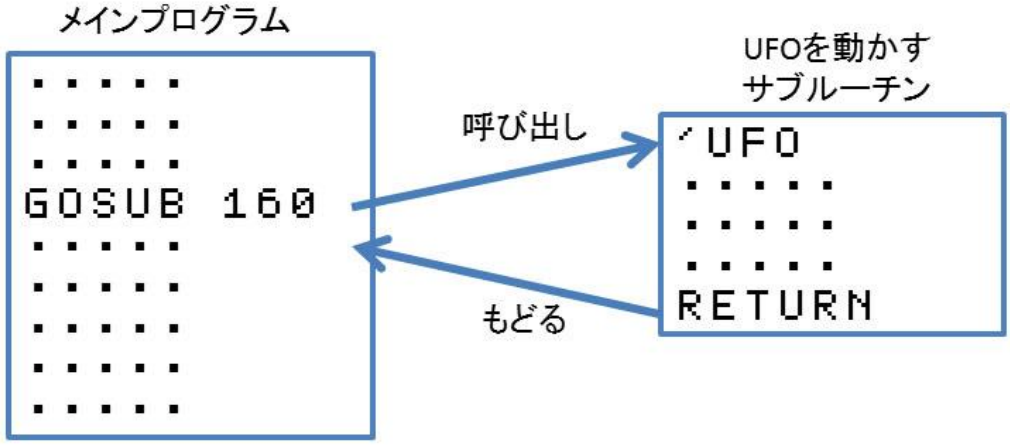
表示した UFO を動かしてみましょう。
自機と UFO を同時に動かさないとゲームにならないので、自機を動かしているループのプログラムから、UFO を動かすプログラムを**サブルーチン**として呼び出す形にします。

```
…(前略)…
80 '*GAMELOOP
90 LOCATE X,Y
100 PRINT " ";
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
130 LOCATE X,Y
140 PRINT "A";
145 GOSUB 160
150 GOTO 80
160 '*UFO
170 LOCATE U,V
180 PRINT " ";
190 U=U+RND(3)-1
200 V=V+RND(3)-1
210 LOCATE U,V
220 PRINT CHR$(241);
230 RETURN
```

UFO を動かすサブルーチンを呼び出す
UFO を動かすサブルーチン
UFO をいったん消す
UFO の横座標・縦座標を乱数で変化
UFO を表示
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。UFO がランダムに動き回ります。
それでは、プログラムの内容を説明します。

145 行:「GOSUB」(ゴーサブ)命令で、UFO を動かすサブルーチンを呼び出します。

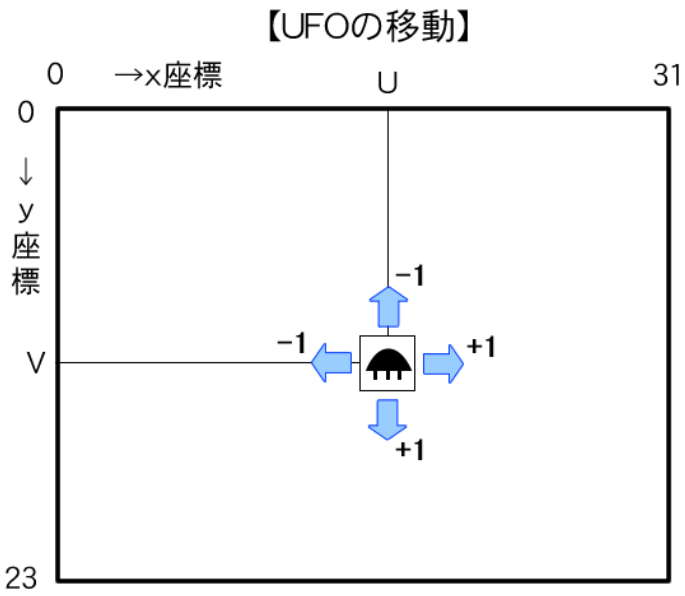


メインプログラムからは「GOSUB」命令でサブルーチンへジャンプし、サブルーチンからは「RETURN」(リターン)命令でもどります。もどった後は、GOSUB 命令の続きへプログラムの処理が移ります。
サブルーチンに分けると、プログラムがすっきりしてわかりやすくなります。また、何度も同じサブルーチンを呼び出して使うことができます。

160 行～:UFO を動かすサブルーチンです。

170～180 行:UFO をいったん消します。

190～200 行:UFO の横座標 U、縦座標 V を乱数で変化させています。「RND(3)」で「0,1,2」のどれかの数が出るので、「RND(3)-1」とすると「-1,0,1」のどれかになります。これで、UFO の座標が縦横に 1 ずつランダムに変化することになります。



210～220 行:UFO を表示します。

230 行:RETURN 命令でメインプログラムへもどります。

今のプログラムだと、UFO が画面の外へはみ出して消えてしまいます。
自機の時と同じように、はみ出さないようにプログラムを改造します。

```
190 U=U+RND(3)-1
192 IF U<0 THEN U=0
194 IF U>31 THEN U=31
200 V=V+RND(3)-1
202 IF V<0 THEN V=0
204 IF V>22 THEN V=22
210 LOCATE U,V
```

U が 0 より小さくなったら、0 にもどす
U が 31 より大きくなったら、31 にもどす
V が 0 より小さくなったら、0 にもどす
V が 22 より大きくなったら、22 にもどす

プログラムを実行してみましょう。今度は UFO が画面からはみださなくなります。

190 行・200 行の RND 関数の最大値や引き算する値を変えると、UFO の動きが変わります。

「SAVE 0」でプログラムを保存しておきましょう。

●ビームを発射する

自機からビームを発射してみましょう。
まず、ゲームのメインループのプログラムで、「スペースキーが押されていたら、ビームのサブルーチンを呼ぶ」ようにします。

```
80 '*GAMELOOP
90 LOCATE X,Y
100 PRINT " ";
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
130 LOCATE X,Y
140 PRINT "A";
142 IF BTN(SPACE)=1 THEN GOSUB 240
145 GOSUB 160
150 GOTO 80
```

スペースキーが押されていたら、ビームのサブルーチンを呼ぶ

次に、プログラムの最後に、ビームのサブルーチンを追加します。

```
240 '*BEAM
250 B=X
260 FOR C=Y-1 TO V STEP -1
270 LOCATE B,C
280 PRINT "|";
290 LOCATE B,C
300 PRINT " ";
310 NEXT
320 RETURN
```

ビームの横座標 B を X にする(自機の横座標と同じ)

ビームの縦座標 C を、自機の1つ上から UFO の高さまで変化させてくり返し

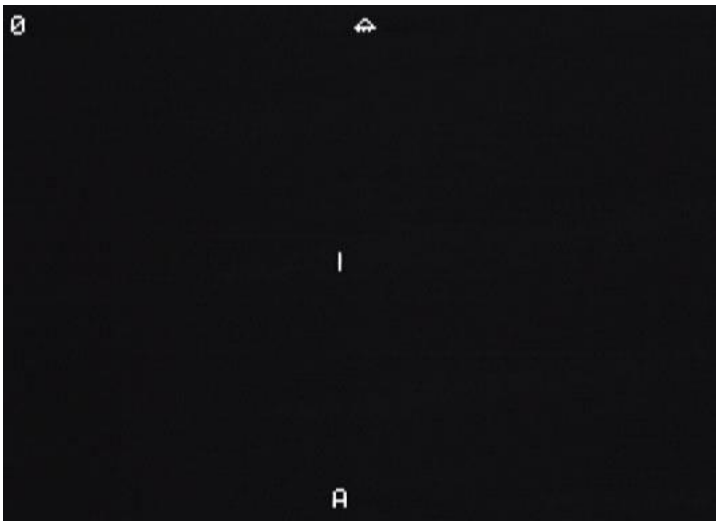
ビームを表示

ビームを消去

ここまでくり返し

メインプログラムにもどる

プログラムを実行してみましょう。
スペースキーを押すと、ビームが自機から UFO の高さまで飛びます。



プログラムの内容を説明します。

240 行:ここからビームのサブルーチンが始まります。

250 行:ビームの横座標 B を、自機の横座標Xにします。

260 行:ビームを自機から UFO まで飛ばすため、**FOR**(フォー)命令と **NEXT**(ネクスト)命令を使って、ビームの縦座標 C を変化させてくり返します。

```
FOR  C=Y-1  TO   V   STEP  -1
```

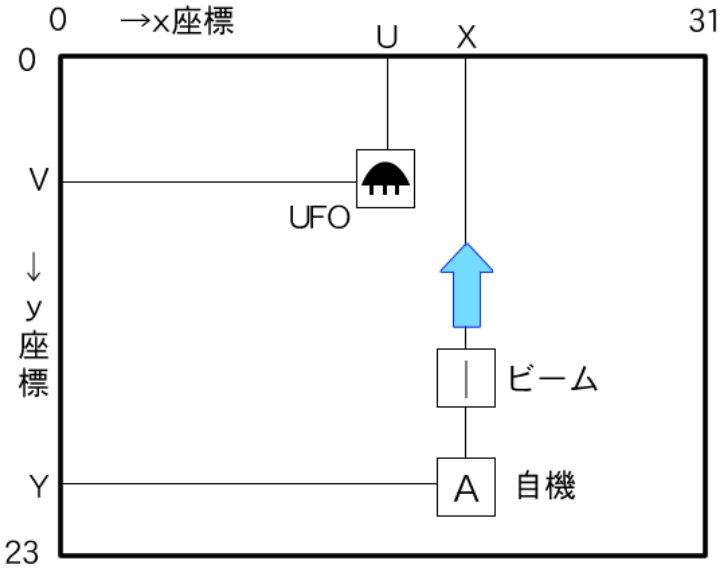
変数の初期値　変数の終値　変数の増分

変数の初期値	ループ変数の最初の値
変数の終値	ループ変数の最後の値
変数の増分	ループ変数をどのくらい変化させるかの値 STEP 以下を省略すると1ずつ増やす

ビームの動きを考えると、自機の1つ上で発射されて、UFO まで上向きに飛んでいきます。
ですから、ビームの縦座標 C を、自機の1つ上(Y-1)から、UFO の縦座標(V)まで、1 つずつ減らしていけばいいことになります。

この FOR 命令から、310 行の NEXT 命令までのプログラムを、C の値を Y-1 から V まで変化させてくりかえします。

【ビームの移動】



270～280 行:ビームを表示します。「|」の文字は、Shift キーを押しながら「¥」を押します。

290～300 行:ビームを消します。

310 行:くり返しの終わりを示す NEXT 命令です。

320 行:RETURN 命令で、メインプログラムへもどります。

このままだと、ビームの動きが速すぎてよく見えません。
時間待ちの命令を入れて、動きを遅くしてみましょう。

```
240 '*BEAM
250 B=X
260 FOR C=Y-1 TO V STEP -1
270 LOCATE B,C
280 PRINT "|";
285 WAIT 2
290 LOCATE B,C
300 PRINT " ";
310 NEXT
320 RETURN
```

時間待ちをする

プログラムを実行してみましょう。
今度はビームの動きが遅くなるので、UFO の高さまで飛ぶのがよくわかります。

時間待ちをするために、WAIT（ウェイト）命令を使います。

```
WAIT  2
```
待ち時間

待ち時間	60 分の 1 秒単位で指定する。「60」で 1 秒。

「SAVE 0」でプログラムを保存しておきましょう。

●ビームと UFO の当たり判定

今のプログラムだと、ビームが UFO に当たっても何も起きません。
ビームが UFO に当たったら、爆発するようにしましょう。

```
310 NEXT
320 IF B<>U THEN RETURN
330 '*HIT
340 BEEP
350 LOCATE U,V
360 PRINT "*";
370 WAIT 20
380 LOCATE U,V
390 PRINT " ";
400 RETURN
```

RETURN 命令の行を書きかえ。もしビームが外れていたらもどる
命中音を出す
爆発の表示
時間待ち
表示を消す
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。
ビームが UFO に当たると、命中音が出て爆発して、UFO が消えます。

320 行:ビームの横座標 B と、UFO の横座標 U が違っていれば、ビームが当たっていないので、RETURN 命令でメインプログラムへもどります。
「B<>U」は「B と U が等しくない」という意味の条件式です。

330 行:ビームが当たった時のプログラムが、ここから始まります。

340 行:**BEEP**(ビープ)命令で、命中音を出します。BEEP 命令の文法は以下のとおりです。

```
BEEP   30     ,30
       音の高さ   音の長さ
```

音の高さ	1～255 で指定する。省略可能。
音の長さ	60 分の 1 秒単位で指定する。「60」で 1 秒。省略可能。

数字を変えると、音の高さや長さが変わります。

350～360 行:UFO の座標に、爆発「*」を表示します。

370 行：爆発を表示した後に、しばらく時間待ちをします。

380～390 行：爆発表示を消します。

400 行：RETURN 命令で、メインプログラムへもどります。

このままだと、UFO をうち落としても点数が入らないので、スコアを加算します。

```
330 '*HIT
340 BEEP
350 LOCATE U,V
360 PRINT "*";
362 S=S+10
364 LOCATE 0,0
366 PRINT S
370 WAIT 20
380 LOCATE U,V
390 PRINT " ";
400 RETURN
```

スコア S を 10 点加算

スコアを画面に表示する

プログラムを実行してみましょう。
UFO をうち落とすと、スコアが増えていくようになります。

362 行：スコア変数 S を 10 点加算します。

364～366 行：画面左上にスコアを表示します。

362 行の加算する数字を変えると、UFO をうった時の点数が変わります。
いろいろ変えて試してみましょう。

UFO をうち落とした後、同じ場所にまた出てくると、次にうつのが簡単になってしまいます。
UFO が上にもどって、違う場所に表示されるようにしましょう。

```
330 '*HIT
340 BEEP
350 LOCATE U,V
360 PRINT "*";
362 S=S+10
364 LOCATE 0,0
366 PRINT S
370 WAIT 20
380 LOCATE U,V
390 PRINT " ";
395 U=RND(32):V=0
400 RETURN
```

UFO の位置を再設定する

最初の設定と同じように、UFO の位置を乱数で決めます。

プログラムを実行してみましょう。
UFO をうち落とすと、違う場所に出てくるので、次にうつのが難しくなります。

「SAVE 0」でプログラムを保存しておきましょう。

●UFO の侵略

今のままだと、UFO が一方的にやられるだけなので、ゲームとして面白くありません。
UFO が自機の段まで降りてきたら、ゲームオーバーになるようにしましょう。
まず、UFO を動かすプログラムを改造して、UFO がだんだん下へ降りてくるようにします。

```
160 '*UFO
170 LOCATE U,V
180 PRINT " ";
190 U=U+RND(3)-1
192 IF U<0 THEN U=0
194 IF U>31 THEN U=31
200 V=V+RND(4)-1
202 IF V<0 THEN V=0
204 IF V>22 THEN V=22
210 LOCATE U,V
220 PRINT CHR$(241);
230 RETURN
```

乱数の最大値を「4」に変える

今までは「RND(3)-1」だったので、UFO が縦方向に動く幅が「-1,0,1」のどれかだったのですが、「RND(4)-1」とすると、「-1,0,1,2」になるので、だんだん下へ降りてくるようになります。

そして UFO が一番下まで降りると、縦座標 V が 22 になるので、その場合はゲームオーバーになるようにします。メインループのプログラムを改造します。

```
80 '*GAMELOOP
90 LOCATE X,Y
100 PRINT " ";
110 IF BTN(LEFT)=1 AND X>0 THEN X=X-1
120 IF BTN(RIGHT)=1 AND X<31 THEN X=X+1
130 LOCATE X,Y
140 PRINT "A";
142 IF BTN(SPACE)=1 THEN GOSUB 240
145 GOSUB 160
150 IF V<22 THEN GOTO 80
152 BEEP 30,30
154 LOCATE 12,12
156 PRINT "GAME OVER"
158 END
160 '*UFO
```

V が 22 より小さかったら、侵略されていないのでもどる

ゲームオーバー音を出す

画面中央にゲームオーバー表示

プログラムを終了する

プログラムを実行してみましょう。
UFO が自機の段まで来ると、ゲームオーバーになります。

150 行:これまで「GOTO 80」と無条件に前へもどっていた行を、IF 命令を使って「V が 22 より小さかったらもどる」ようにします。

152 行:BEEP 命令で、ゲームオーバーの音を出します。

154～156 行:画面の中央に「GAME OVER」と表示します。

158 行:**END**(エンド)命令で、プログラムを終了します。

「SAVE 0」でプログラムを保存しておきましょう。

★改造点

今のままだと、UFO が下に降りてくる速度が速いので、すぐに侵略されてしまいます。
UFO の移動量を決める乱数の範囲を調整するといいでしょう。

例
```
200 V=V+RND(7)/2-1
```

RND(7) → 0,1,2,3,4,5,6
RND(7)/2 → 0,1,2,3　(IchigoJam では小数点以下は切り捨てられる。
3 になる確率は 0,1,2 の半分)
RND(7)/2-1 → -1,0,1,2　(2 になる確率は-1,0,1 の半分)