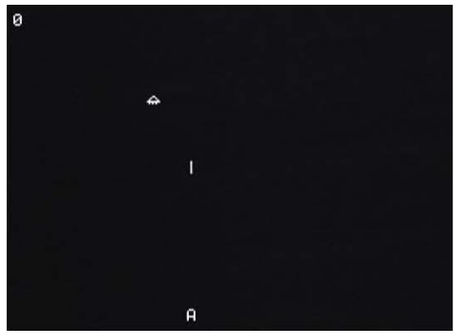
IchigoJam（いちごじゃむ）でプログラミング

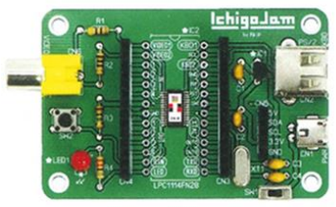
ミッションカード Mission POSSIBLE! CARD

君だけの
小っちゃな
コンピュータ。



ミッション（指令）

UFOを撃ち落とせ



IchigoJam

Ichigojamでshootingゲームをプログラムしてみよう！

2024..07.14 Ver.3.02

・本資料はNPO法人NEXTDAYの協力のもとNPO法人小樽青少年の科学の芽を育てる会が作成しました。
・資料の作成にあたり以下の資料を参照しました。
＞親子でベーシック入門 IchigoJamではじめてのプログラミング （出版社: ジャムハウス）
＞IchigoJamでプログラミング （発売：プログラミングクラブネットワーク）
・原稿についてはICHIGOJAM開発者**福野 泰介様**のブログを参照させて頂いてます。
・上田市マルチメディア情報センター 斎藤　史郎 様の講習会テキストを参照させて頂いています。

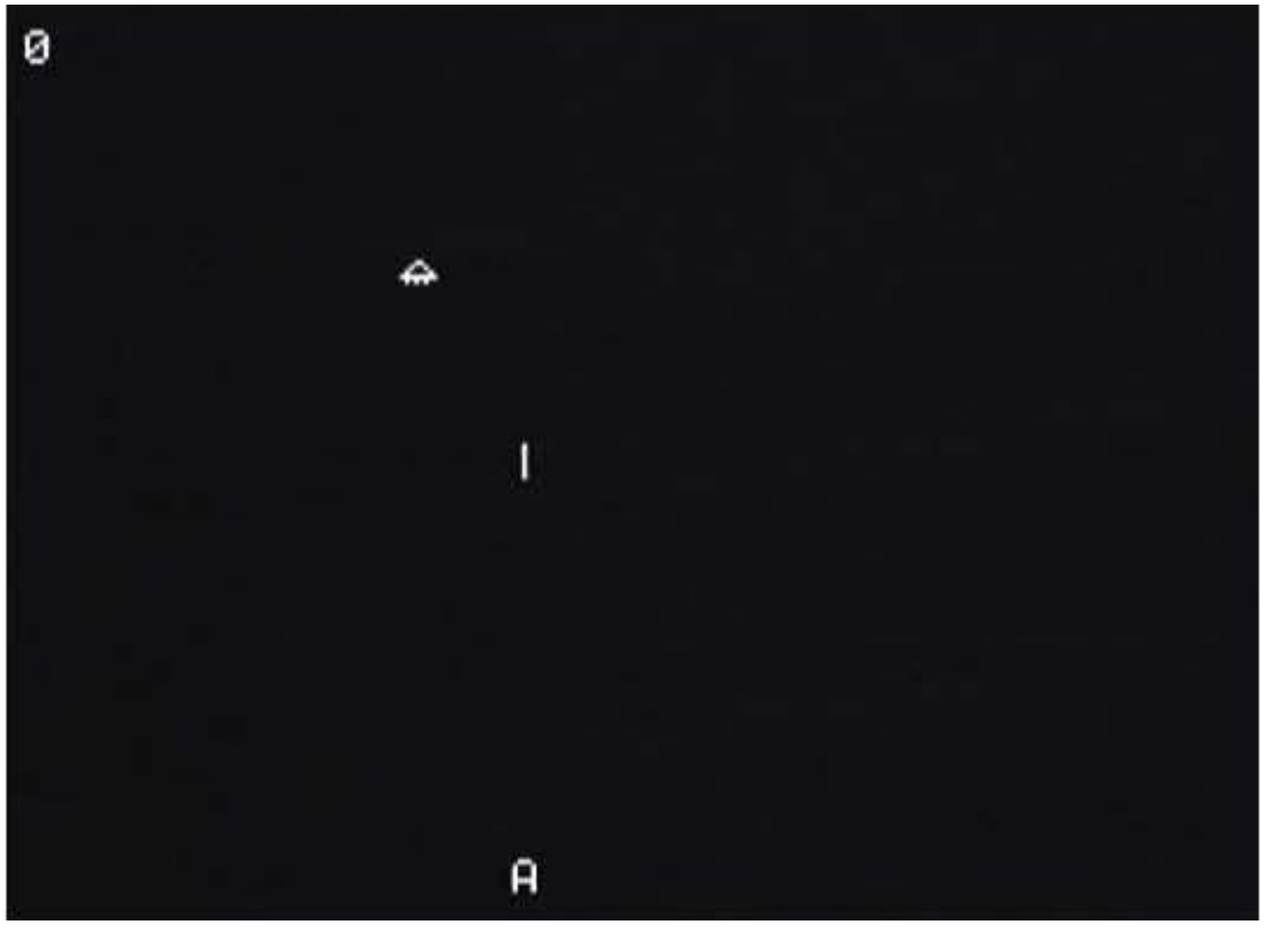
小樽別院　寺子屋教室
主催　　小樽青少年の科学の芽を育てる会
協力　　浄土真宗本願寺派 本願寺小樽別院
協力　　NPO法人NEXTDAY

shootingゲームをプログラムしてみよう！！

●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を撃ち落とします。



●シューティングゲームを作る手順

項目	内容
ゲーム画面を表示	点数を画面に表示
自機を表示	自機を画面に表示
自機を動かす	矢印キーで自機を左右に動かす
UFOを表示	敵のUFOを画面に表示する
UFOを動かす	UFOをランダムに動かす
ビームを発射する	自機から上へビームを発射する
ビームとUFOの当たり判定	ビームがUFOに当たった時の処理
UFOの侵略	UFOが最下段まで来たらゲームオーバー

●ゲームの画面を表示する

プログラムの初期設定をした後、画面にスコア(得点)を表示します

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

10 : コメントでプログラムのタイトルを入れてます

20 : CLS命令で画面をクリアします。 CLV命令で変数の 値を
クリアします

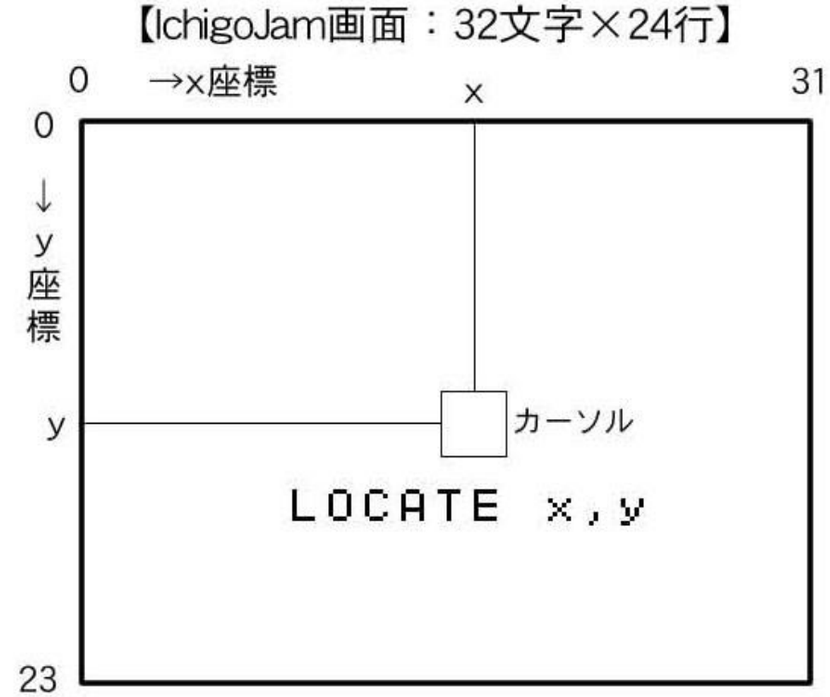
30 : 画面に文字を表示する位置(カーソル位置)を設定します

Ichigojamの画面サイズは、横32文字×縦24行になっています。
LOCATE 0,0 として、画面左上の座標を指定しています

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)。

40 : 画面にスコア変数 S の値を表示します



SAVE 0 ↵ (プログラムの退避をしよう)

RUN ↵ (プログラムを実行してみよう)

●自機を表示する

スコアと同じように、自機を表示します
プログラムを追加していきます

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

50 : 自機の位置を、変数Xと変数Yに値を設定します

60 : カーソルをの位置を指定した変数の位置に移動します

70 : 指定した位置に自機　A を表示します。
　　最後に　;をつけます　これは表示後に自動的に改行してしまいますが、改行しないで次の文字が表示されるようになります

変数のXとYを変えることで、自機の位置がかわります。
試してみましょう

```
SAVE  0
```

プログラムを打ち込んだら、必ず退避してから実行してね

●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ます
プログラムを追加して行きます

```
80  '＋GAME LOOP
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
```

80 : ここからゲームの処理をするプログラムなので、
　　目印のコメントをいれます

90 : 自機を画面から一度消します

プログラムを実行してみましょう
カーソルキーを押すと自機が左右に移動します

SAVE　0

110 : BTN(ボタン)関数を使って、左矢印キーが押されたかを判断します

120 : :BTN(ボタン)関数を使って、右矢印キーが押されたかを判断します

```
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関数が1となり、押されていないと0となります

そのBTN関数の値を、IF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。

BTN(LEFT)が1だったら変数Xから-1した値をXにいれます
BTN(RIGHT)が1だったら変数Xから+1した値をXにいれます

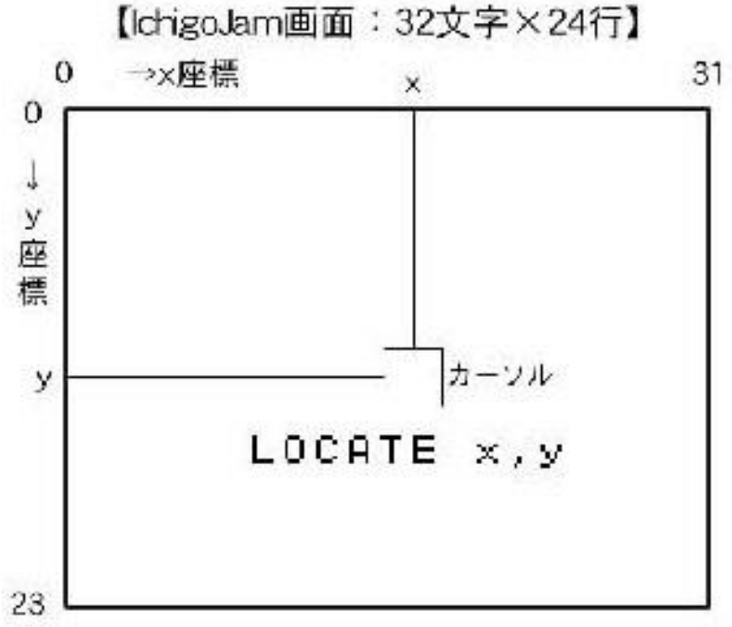
130 : 140 : 自機を、新しい座標の位置に表示します

150 : 続けて自機を動かすために、GOTO命令で80行目に戻ります

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

このプログラムは、問題があるのです
　自機が画面左端に行っても左矢印キーを押し続けても動かなくなってしまい、次に右矢印キーを押しても左矢印キーを押した回数だけ右矢印キーを押さないと右へ移動しません。右端でも同様です

```
80  '＋GAME LOOP
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
```



自機のX座標の範囲を考えないで、増やしたり減らしたりしてしまいXの値がマイナスになったり31を超えた値になってしまうからです

画面サイズからXの値は0～31の範囲の座標にしないといけません
プログラムの110行目と120行目を改造します

プログラムを実行して確認してみましょう

IF命令の条件式に「AND」でつないで2つの条件式とします
110 : この条件式はBTN(LEFT)が1と等しく、かつ、Xが0より大きいと言う意味になり両方の条件がそろった時だけ命令が実行されます
120: この条件式はBTN(RIGHT)が1と等しく、かつ、Xが31よりちいさいと言う意味になり両方の条件がそろった時だけ命令が実行されます

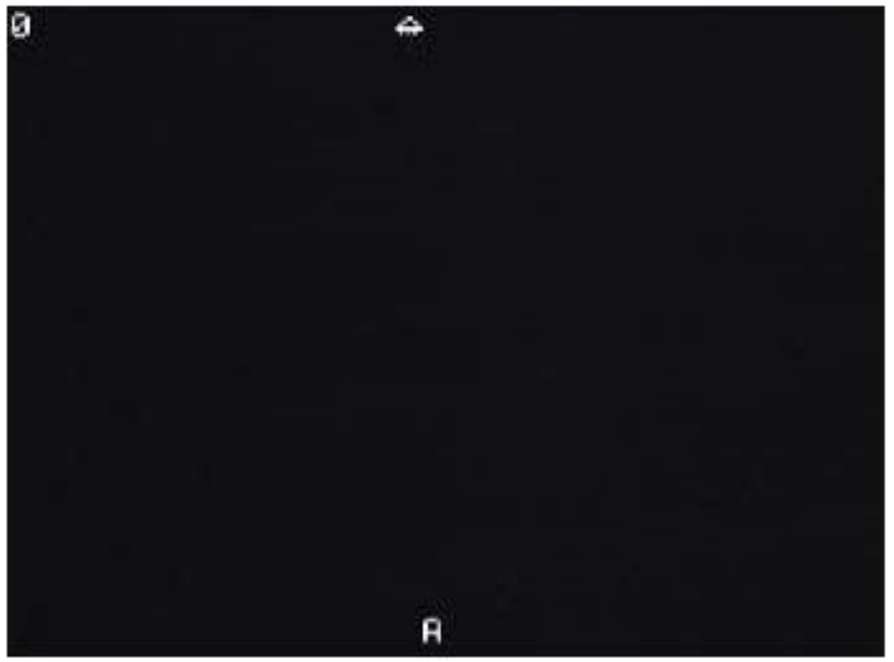
```
SAVE  0
```

●UFO を表示する

自機の次はUFOを表示します

UFOを表示するプログラムを追加します
70行と80行の間に追加するので、行番号を72～76にしています

```
60  LOCATE X,Y
70  PRINT " A ";
72  U=RND(32):V=0
74  LOCATE U,V
76  PRINT CHR$(241);
80  '＋GAME LOOP
```



プログラムを実行してみましょう
UFOが表示されます
実行するたびに、UFOの位置が変わります

```
SAVE  0
```

72 : UFOの横座礁U、縦座礁Vの位置を設定します
同じ位置に表示されないようにRND関数で乱数を使って横座礁Uを決定します

```
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)と指定することで0～31の乱数がでてくるので、UFOは画面の左端から右端までのどこかに表示されます
縦座礁はVは、画面の一番上に設定しています

74～76 : UFOを画面に表示します
ここでは、CHR$関数でUFOを指定しています

UFO（241番）

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を表示しています
Ichigojamの文字コードは左の一覧表のようになっています
参考にしてください

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

例　PRINT CHR$(241)

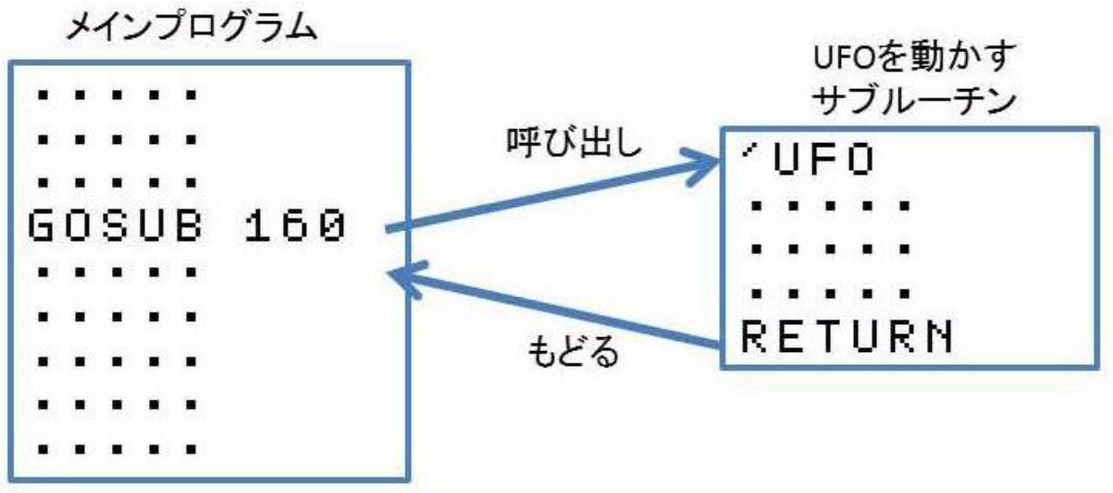
自機の A をCHR$関数に変えて、自機のキャラクターをいろいろ変えてみる事もできます

●UFOを動かす

表示したUFOを動かします
自機とUFOを同時に動かないとゲームにならないので、自機を動かしているプログラムから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

```
130  LOCATE X,Y
140  PRINT "A";
145  GOSUB 160
150  GOTO 80
160  '＊UFO＊  SUB
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
```

メインプログラムからGOSUB命令でサブルーチンへジャンプして、サブルーチンからはRETURN命令で戻ります
サブルーチンに分けると、プログラムがすっきりして解りやすくなり、何度も同じ処理を呼び出して使う事ができます



プログラムを実行してみましょう
UFOがランダムに飛び回ります

```
SAVE  0
```

145 : GOSUB命令で、UFOを動かすサブルーチンを呼び出します

160 : サブルーチンを示すコメントです

170-180 : UFOを一旦消します

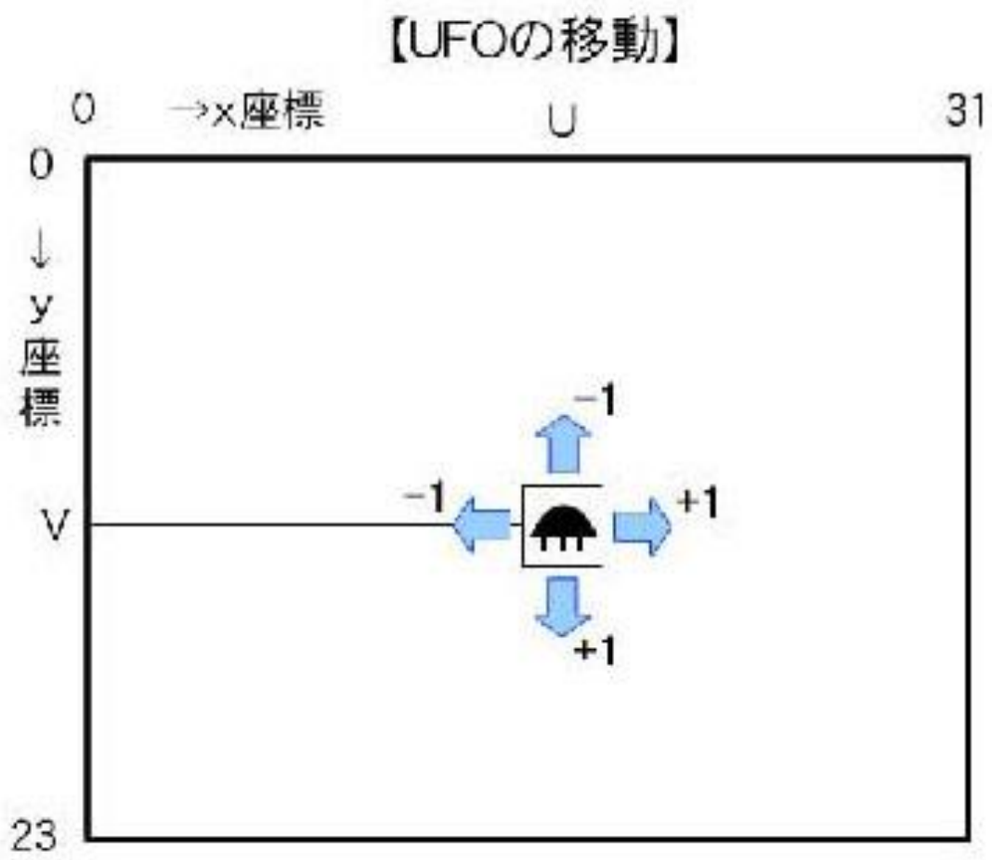
190-200 : UFOの横座礁U、縦座礁Vを乱数で変化させています

210-220 : UFOを表示します

230 : RETURN 命令でメインプログラムへ戻ります

このプログラムだと、UFOが画面の外にはみ出して消えてしまいます
自機の時とむ同じように、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
```



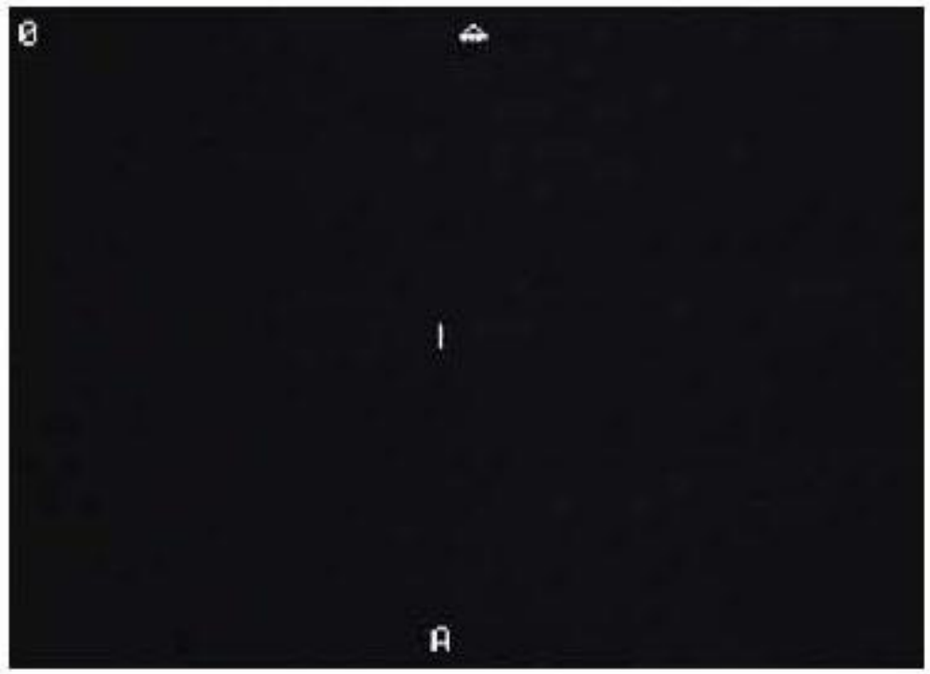
変数の変化はRND(3)で0-,1,2のどれかの数値となり、RND(3)-1することで-1,0,1となるのでUFOの座標が縦横に1ずつランダムに変化します

改造したらプログラムを実行してみましょう
UFOが画面からはみ出さないで飛び回ります

●ビームを発射する

自機からビームを発射します
ゲームのメイングループのプログラムで「スペースキー」が押されたら、ビームサブルーチンを呼びます

```
140  PRINT "A";
142  IF BTN(SPACE) = 1 THEN GOSUB 240
145  GOSUB 160
```



ビームのサブルーチンを追加します

```
240  '＊BEAM＊  SUB
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
```

ビームの縦座標Cを、自機の1つ上からUFOの高さまで変化させて繰り返します

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

240 : コメントでプログラムのタイトルを入れてます

250 : ビームの横座標Bを自機の横座標Xにします

260 : ビームを自機からUFOまで、飛ばす為FOR命令とNEXT命令を使って、ビームの座標Cを変化させて繰り返します

```
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

270 -280 : ビームを表示します　(SHIFT+¥)

290-300 : ビームを消します

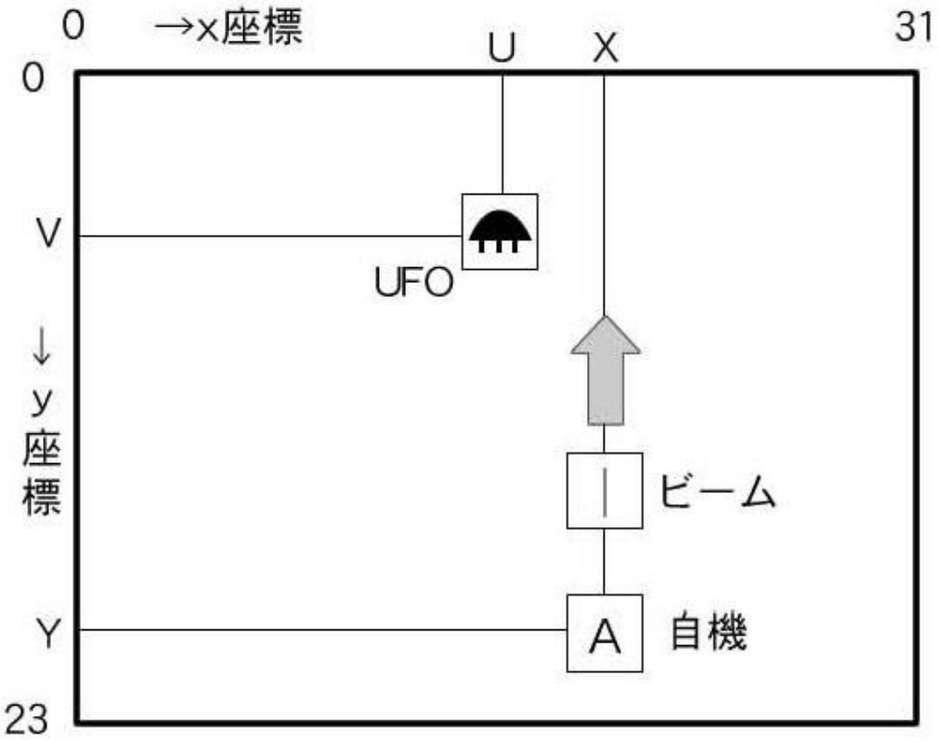
310 : NEXT命令で繰り返しの終わりです

320 : RETURN命令デメインのプログラムに戻ります

ビームのスピードが速すぎるのでWAIT命令を追加して調整します

```
280  PRINT "I";
285  WAIT 2
290  LOCATE B,C
```

【ビームの移動】



ビームの動きを考えると、自機の1つ上で発射されUFOまで上向きに進んで行きます　ビームの縦座礁Cを自機の一つ上Y-1からUFO座礁Vまで1つずつ減らしています

ビームの動きが遅くなるので、UFOの高さまで飛ぶのが解りやすくなります

プログラムを実行してみましょう

```
SAVE  0
```

●ビームとUFOの当たり判定

ビームがUFOに当たると爆発するようにします
ビームが当たった時のザブルーチンHITを追加します

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

320 : ビームが当たったかの判定に変更します

330 : コメントでプログラムのタイトルを入れてます

ビームの横座礁Bと、UFOの横座礁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に戻ります。
B<>Uは BとUが等しくないと言う条件式です

実行してみましょう
ビームがUFOに当たると命中音がでて爆発してUFOが消えます

```
SAVE  0
```

340 : BEEP命令で、命中音をだします
　　数字を変えると、音の高さや長さを変える事ができます

```
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。

350-360 : UFOの座礁に、爆発「＊」を表示します

370 : 爆発後しばらく時間待ちをします

380-390 : 爆発表示を消します

400 : RETURN命令でメインプログラムに戻ります

UFOを撃ち落としたら得点をスコアに追加して表示します

```
360 PRINT " ＊ ";
362  S=S+10
364  LOCATE 0,0
366  PRINT S
370  WAIT 20
```

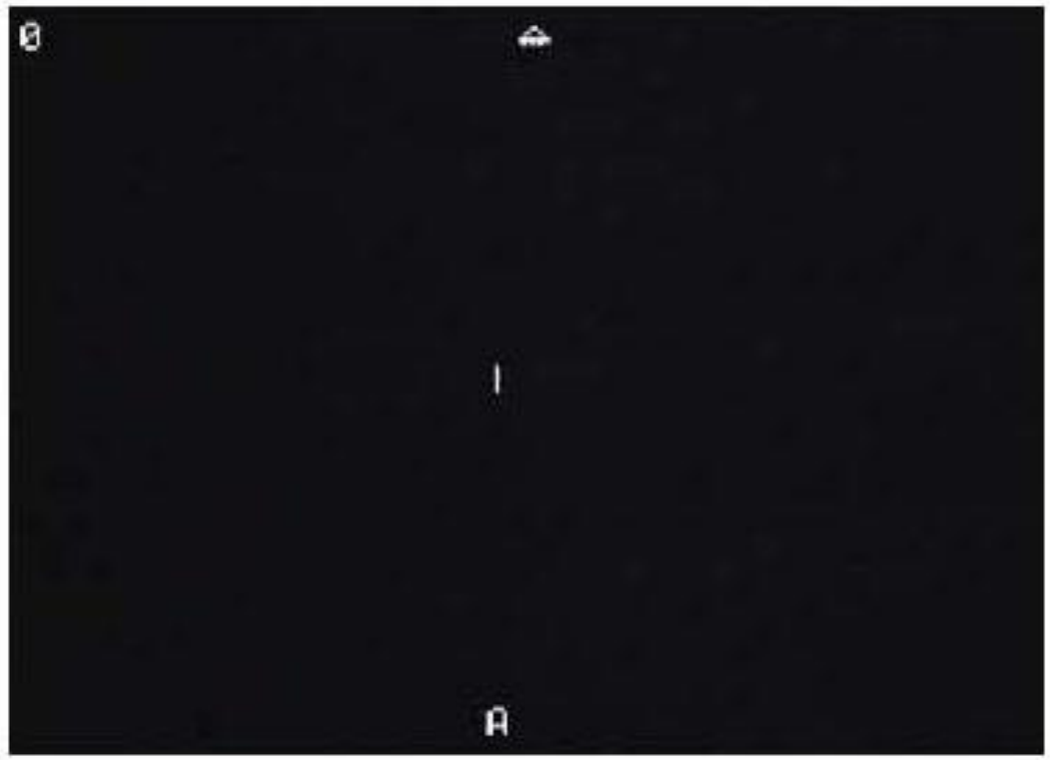
改造したらプログラムを実行してみましょう
UFOを撃ち落とすとスコアが増えて行きます
得点を変えると、UFOを撃ち落とした時の点数が変わります

362 : スコアに得点を追加します

364-366 : 画面上にスコアを表示します

UFOを撃ち落とした後、同じ場所に出てくると次に撃つのが簡単になってしまいます
UFOが上に戻って違う場所に表示されるようにします

```
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
```



最初の設定と同じようにUFO表示位置を乱数で決定する命令を追加します

改造したらプログラムを実行してみましょう
UFOを撃ち落とすと、違う場所に出てくるので、次に撃ち落とすのが難しくなります

```
SAVE  0
```

●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、ゲームとしては面白くありません。
UFOが自機の段までおりてきたら、ゲームオーバーになるようにします

```
194  IF U>31 THEN U=31
200  V=V+RND(4)-1
202  IF V<0 THEN V=0
```

```
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＊ SUB
```

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
今までは、変数の変化はRND(3)で0-,1,2のどれかの数値だったのですがRND(4)-1することで-1,0,1,2となるので段段下に降りてくるようになります

UFOが一番下まで降りると縦座礁Vが22となるので、その時はゲームオーバーとなるようにメインルーチンのプログラムを改造します

改造したらプログラムを実行してみましょう

```
SAVE  0
```

150 : 無条件で80行目に戻る命令をIF命令で縦座礁Vが22より
　　小さい時は80行目に戻るようにします

152 : BEEP命令でゲームオーバーの音を出します

154-156 : 画面に「GAME OVER」を表示します

158 : END命令でプログラムを終了します

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ので、UFOの移動量を決める乱数の範囲を調整します

```
194  IF U>31 THEN U=31
200  V=V+RND(7)/2-1
202  IF V<0 THEN V=0
```

乱数範囲の調整
RND(7)　　→　0,1,2,3,4,5,6
RND(7)/2　→　0,1,2,3
RND(7)2/-1 →　-1,0,1,2
となり2になる確率は1/2となります

チャレンジ
これでプログラムは完成ですが、これで終わりではありません。

あなたのオリジナル改造をしてみましょう
自機が上下にも移動できるようにしてみたり
UFOがミサイルで自機を攻撃してくるようにしてみたり　・・・などなど　いろいろ考えてみましょう

改造したらプログラムを実行してみましょう

SAVE　0

```
10  ' * SHOOTING *  MAIN
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  '+GAME LOOP
90  LOCATE X,Y
100  PRINT " ";
110  IF BTN(LEFT) = 1 THEN X=X-1 AND
X>0 THEN X=X-1
120  IF BTN(RIGHT) = 1 THEN X=X+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 *  SUB
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(7)/2-1
202  IF V<0 THEN V=0
204  IF V>22 THEN V=22
210  LOCATE U,V
220  PRINT CHR$(241);
230  RETURN
```

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

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
```

Ichigojamパソコンを持っていなくてもProgramを作ったり実行する事ができるよ

インターネットに接続しているパソコンのブラウザから下のアドレスをアクセスしてね

```
https://fukuno.jig.jp/app/IchigoJam/ ↵
```

コマンドをおぼえよう　！！

プログラムをリストするには

LIST ↵

範囲を指定してリスト
LIST　10,100

プログラムを実行するには

RUN ↵

プログラムをメモリーに記憶させるには

SAVE ↵

(0～3)の4個のメモリーがあります

プログラムをメモリーから読みだすには

LOAD ↵

メモリーの内容は電源を切っても記憶しています

メモリーの内容をみるには

FILES ↵

プログラムをすべて削除するには

NEW ↵

画面表示をすべて消します（クリア スクリーン）

CLS ↵

プログラムの行削除は行番号だけを打ちます

10 ↵

実行しているプログラムを停止するには

キーを押します

エスケープと読みます
(Escape：逃れる、抜け出す、脱出する)の省略型です

コマンドはキー入力だけでなくファンクションキーでも代用できるよ

F1	F2	F3	F4	F5	F6	F7	F8	F9
CLS	LOAD	SAVE	LIST	RUN	?FREE()	OUT0	VIDEO1	FILES

IchigoJam BASIC 1.1 cheatsheet

＊赤文字は省略可能を示す　＊青文字は説明文

ファンクションキー

F1	F2	F3	F4	F5	F6	F7	F8	F9
CLS	LOAD	SAVE	LIST	RUN	?FREE()	OUT0	VIDEO1	FILES

Dec	Hex	Bin
0	#00	0000
1	#01	0001
2	#02	0010
3	#03	0011
4	#04	0100
5	#05	0101
6	#06	0110
7	#07	0111
8	#08	1000
9	#09	1001
10	#0A	1010
11	#0B	1011
12	#0C	1100
13	#0D	1101
14	#0E	1110
15	#0F	1111

4bit：0～15
8bit：0～255
(-128～127)
16bit：0～65535
(-32768～32767)

＊扱える数値は 16 ビット範囲
小数値は扱えません

VIDEO1	KBD1
VIDEO2	EX1
IN1	KBD2
IN2	SOUND
IN3	ISP
IN4	RESET
VCC	GND
GND	VCC
OUT1	-
OUT2	-
OUT3	OUT5
OUT4	OUT6
BTN	TXD
LED	RXD

演算の優先順位

高い	()	- ～ ! NOT	* / % MOD << >> & ^	+ - |	= != < > <= >=	AND	OR	低い

●式／演算

【 加　　算 】数 + 数
【 減　　算 】数 - 数
【 乗　　算 】数 * 数
【 除　　算 】数 / 数
【 剰　　余 】数 % 数
【 否　　定 】NOT 式
【 論理積　　】数 & 数
【 論理和　　】数 | 数
【排他的論理和】数 ^ 数
【 右シフト 】数 >> 数
【 左シフト 】数 << 数
【 ビット反転】～数
【優先順位変更】(～)
式 AND 式　省略形：&&
式 OR 式　省略形：||

●代入／変数／配列変数

[数] 0～101 まで
LET 変数 , 数　省略形：変数 = 数
LET [数], 数 ,.. 数 n

●リセット／初期化

CLK　キーバッファ消去
CLP　パターン初期化
CLS　画面消去
CLT　時間をリセット
CLV　変数を消去し全て 0 に
SRND 数　乱数の種を設定

●キー入力／ボタン

BTN(0/UP/DOWN/RIGHT/LEFT/SPACE)
INKEY()　リアルタイムキー入力
INPUT 文字列 , 変数

●画面関係

LOCATE X 座標 ,Y 座標 省略形：LC
PRINT 数や文字列　省略形：?
SCR(X 座標 ,Y 座標)
SCROLL 数　0: 上、1: 右、2: 下、3: 左
VIDEO 数 1, 数 2

●関数

ABS(数)
ASC(" 文字 ")
BIN$(数 , 桁数)
CHR$(数 ,.. 数 n)
DEC$(数 , 桁数)
HEX$(数 , 桁数)
RND(数)

●数値表記

123　10 進数
(-32768 ～ 32767)
#E9　16 進数 (0～#FFFF)
`1001　2 進数

●定数

LEFT　左 : 28
RIGHT　右 : 29
UP　上 : 30
DOWN　下 : 31
SPACE　空白 : 32

●条件判断／条件式

IF 数 THEN 次 ELSE 次 2
【 等 し い 】数 1= 数 2
【 等しくない 】数 1<> 数 2
【 小 さ い 】数 1< 数 2
【小さいか等しい】数 1<= 数 2
【 大 き い 】数 1> 数 2
【大きいか等しい】数 1>= 数 2

●移動／繰り返し／サブルーチン

FOR 変数 = 数 1TO 数 2 STEP 数 3～NEXT
GOSUB 行番号　省略形：GSB
GOTO 行番号
RETURN　省略形：RTN

●ハードウェア

ANA(数)　0～1023
BPS 通信速度　省略時：115,200bps
I2CR(数 1, 数 2, 数 3, 数 4, 数 5)
I2CW(数 1, 数 2, 数 3, 数 4, 数 5)
IN(数)　IN1-9 から入力
LED 数　0: 消灯、1: 点灯
OUT 数 1, 数 2　OUT1-7 に出力
PWM 数 1, 数 2, 数 3
RESET　リセット
SLEEP　スリープ（ボタンを押すと復帰）
UARTシリアル出力設定 , シリアル受信設定
WAIT 数　60 で約 1 秒

●ファイル

FILE()
FILES 数 1, 数 2
LOAD 数
LRUN 数 , 行番号
RUN
SAVE 数

●プログラム

CONT　再度実行する
END　プログラムを終了
FREE()　プログラムの残りメモリ数
LINE()　現在実行中の行番号
LIST 行番号 1, 行番号 2
NEW　プログラムを消す
RENUM 数 1, 数 2
STOP　処理を中断する

●メモリ操作／マシン語

PEEK(アドレス)
POKE アドレス , 数 ,... 数 n
USR(アドレス , 数)

●その他

HELP　メモリマップを表示
REM 注釈　省略形：'
TICK()　tick 時間 (1/60) を返す
VER()　バージョン番号を返す

メモリマップ

#0000	文字パターン (#00～ #DF)
#0700	PCG パターン (#E0～ #FF)
#0800	配列変数・変数
#0900	画面 (32 文字 ×24 行)
#0C00	プログラムリスト
#1001	
#1002	キーが押されたビット
#1003	キーバッファ格納数 (最大 14)
#1004～F	キーバッファ

●音楽／サウンド

BEEP 周期 , 長さ　BEEP を鳴らす
PLAY【MML】　MML なしで演奏停止
SOUND()　再生中なら 1 を返す
TEMPO テンポ　テンポを指定

■MML (Music Macro Language)

[音]　音 (C D E F G A B R)
[音]n　音長（. を付けると 1.5 倍長）
[音]+　半音上げる
[音]-　半音下げる
Tn　テンポ (初期値 :120)
Ln　デフォルトの音長 (初期値 :4)
On　オクターブ指定（1～5）
>　1 オクターブ上げる
<　1 オクターブ下げる
$　以後の MML を繰り返す
Nn　音の高さを指定
(音長で指定可能な値：1,2,3,4,8,16,32)

●制御（コントロール）コード

08(#08)　バックスペース（後退）
13(#0D)　リターン
14(#0E)　インサート（挿入）
127(#7F)　デリート（削除）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	●	○	■	□	▒
1	▬	▭	▮	▯	▪	▫	▬	▭	▮	▯	▪	▫	▬	▭	▮	▯
2	▪	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	7F
8	80	▙	▛	▀	▜	▟	▄	▐	▌	▗	▝	▘	▖	▞	▚	▁
9	▪	▬	▭	▮	▯	▪	▫	▬	▭	▮	▯	▪	▫	▬	▭	▮
A	゛	゜	▗	。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
B	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
C	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
D	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
E	▪	▬	▭	▮	▯	▪	▫	▬	▭	▮	▯	▪	▫	▬	▭	▮
F	▪	▬	▭	▮	▯	▪	▫	▬	▭	▮	▯	▪	▫	▬	▭	▮

1 文字は 8×8 ドットで構成