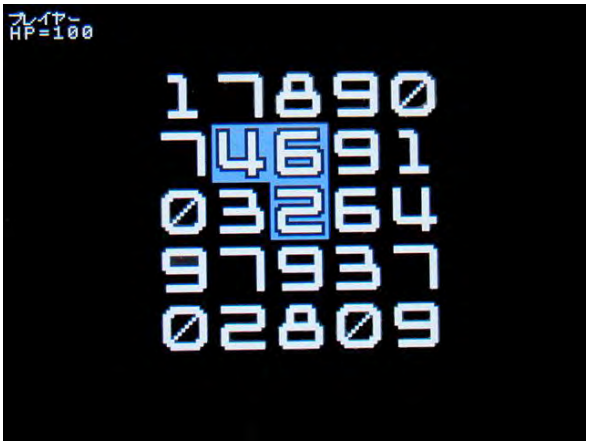
プチコンでパズルバトルゲームを作る（３）

●モンスターを攻撃する



前回までで、数字パネルをタッチしてパネルの色を変えるプログラムを作りました。
選んだパネルの数字によって、敵のモンスターを攻撃するプログラムにしましょう。

ルールは、「**選んだパネルの数を足して、10の倍数（10、20、30…）になったら攻撃**」とします。

```
1 '*** PUZZLE7 ***
2
3 ACLS
```

タイトルを「PUZZLE7」にする

メインループで、タッチペンを画面からはなした時に、合計を判定するようにします。

```
21 '=== メイン ループ ===
22 @MAINLOOP

（中略）

37   TOUCH OUT STTM,TX,TY
38 WEND
39
40 'ごうけい はんてい
41 GP=0
42 GS=0
43 AT=0
44
45 FOR Y=0 TO 4
46   FOR X=0 TO 4
47     IF T[X,Y]==1 THEN
48       GP=GP+1
49       GS=GS+P[X,Y]
50     ENDIF
51   NEXT
52 NEXT
```

パネル合計 GP、パネル数合計 GS を 0 にリセット

攻撃フラグ AT を 0 にリセット

パネル全体でループ

パネルがタッチ中だったら
パネル合計 GP を 1 枚足す
パネル数合計 GS にパネルの数を加算

```
53
54  'こうげき
55  IF GS MOD 10==0 THEN AT=1:
GOSUB @ATTACK
56
57  'タッチ クリア
58  FOR Y=0 TO 4
```

パネル数合計 GS が 10 で割り切れたら、
攻撃フラグ AT を1にして、
攻撃サブルーチン@ATTACK を呼ぶ

合計判定では、パネル全体（5×5＝25 枚）をチェックします。パネルがタッチされていれば（T[X,Y]が 1 ならば）、パネルの枚数をカウントする変数 GP を 1 増やし、パネルの合計数を入れる変数 GS へパネルの数字 P[X,Y]を加算します。

計算が終わった後に、パネルの合計数 GS が 10 の倍数かどうかを判定します。
判定では、余りを計算する MOD（モッド）（演算子）を使います。

```
GS  MOD  10
```

GS を 10 で割った余りを計算します。もし割り切れれば、答えは 0 になります。
IF 命令で余りが 0 かどうかを判断して、もし 0 なら、攻撃フラグ変数 AT を 1 にして、GOSUB 命令で攻撃サブルーチン@ATTACK を呼びます。

そして、プログラムの最後に、プレイヤーの攻撃サブルーチンを追加します。

```
122  '=== プレイヤーのこうげき ===
123  @ATTACK
124
125  DISPLAY 0
126  LOCATE 0,25
127  COLOR 13
128  PRINT "◆◆◆ プレイヤーのこうげき！ ◆◆◆"
129  FOR I=1 TO 3
130   GFILL 168,68,231,131,RGB(25
5,0,0)
131   VSYNC 5
132   GFILL 168,68,231,131,RGB(0,
0,0)
133   VSYNC 5
134  NEXT
```

「こうげき！」文字表示

モンスターが攻撃を受けた様子
グラフィックで赤の四角を 3 回点滅表示

```
135
136 AP=GS+(GP-1)*10
137 MHP=MHP-AP
138 LOCATE 0,26
139 COLOR 15
140 PRINT "ダメージ:";AP;"ポイント"
141 LOCATE 0,1
142 PRINT "HP=";MHP;"     "
143 VSYNC 120
144 LOCATE 0,25
145 PRINT "  "*48
146 PRINT "  "*48
147
148 RETURN
```

プレイヤーの攻撃ポイント AP を計算
パネル合計数＋（パネル枚数－1）×10

モンスターの
ダメージ表示

スペース 48 文字×2 行を表示して
画面の文字を消す

攻撃サブルーチンでは、まずモンスターが攻撃を受けた様子を表示します。
モンスターと同じ場所に、GFILL 命令で赤い四角を描いて、すぐ消して…を、FOR～NEXT 命令で 3 回くり返します。

次に、プレイヤーの攻撃ポイント AP を計算します。
計算ルールは、「**パネルの合計数 GS＋（パネルの枚数 GP－1）×10**」とします。
例えば、「1」と「9」の 2 枚のパネルを選んだとしたら、パネルの合計数＝10、パネルの枚数＝2 になるので、ポイントは 10＋（2－1）×10＝20 ポイントになります。
パネルの合計数、パネルの枚数が多いほど、ポイントが上がります。

※実は「0」のパネル 1 枚だけをタッチしても攻撃できます。
（0 を 10 で割ると余りは 0 なので、攻撃になる）
ただし、パネル合計数が 0、パネル枚数が 1 枚なので、攻撃ポイントは 0＋（1－1）×10＝0 ポイントとなり、モンスターは全くダメージを受けません。
（逆に言えば、この時に 0 ポイントになるように、攻撃ポイントの計算ルールを決めています）

計算した攻撃ポイントを画面に表示して、モンスターの体力 MHP をポイント分減らします。

プログラムを実行してみましょう。
パネルをタッチすると、モンスターに攻撃できます。
パネルの合計が 10 の倍数になると攻撃できること、攻撃ポイントが上の計算のとおりになることを確かめましょう。

●プレイヤーの勝利判定

このままだと、モンスターの HP がマイナスになってもそのままゲームが続いてしまうので、HP が 0 になったらプレイヤーの勝利になり、ゲームの最初にもどるようにしましょう。

プログラムの最初の部分を書きかえて、ラベルをつけて、もどってこられるようにします。

```
1 '*** PUZZLE7 ***
2
3 DIM P[5,5],T[5,5]
4
5 '=== バトルかいし ===
6 @BATTLESTART
7
8 ACLS
9 XSCREEN 2
```

行を入れ替えて、DIM 命令を最初に持ってくる

コメントとラベルを追加する

メインループの中で、モンスターへ攻撃した後に HP の値を見て、0 以下になっていたら「プレイヤーが勝った」処理をして、最初に戻るようにします。

```
57 'こうげき
58  IF GS MOD 10==0 THEN AT=1:GOSUB @ATTACK
59  IF MHP<=0 THEN GOSUB @PLAYERWIN:GOTO @BATTLESTART
60
61 'タッチ クリア
62 FOR Y=0 TO 4
```

MHP が 0 以下だったら、プレイヤー勝利サブルーチンを呼んだ後、最初にもどる

プログラムの最後に、プレイヤー勝利のサブルーチンを追加します。

```
154 '=== プレイヤーしょうり ===
155 @PLAYERWIN
156
157 DISPLAY 0
158 LOCATE 0,25
159 COLOR 13
160 PRINT "●●● ";MNAME$;"をたおした！ ●●●"
161 SPHIDE 100
162 VSYNC 180
163 RETURN
```

「モンスターをたおした」表示

モンスターを隠す

プログラムを実行してみましょう。
プレイヤーが攻撃して、モンスターの HP が 0 以下になると、「スライムをたおした！」と表示されて、最初にもどります。

ただ、このままだと、2 回目のバトルの時に、1 回目で数字パネルを選択していた表示が残ってしまいます。
数字パネルをセットするサブルーチンで、タッチ配列変数 T[X,Y]を 0 にリセットするようにしましょう。

```
71 '=== パネル セット ===
72 @SETPANEL
73
74 FOR Y=0 TO 4
75   FOR X=0 TO 4
76     P[X,Y]=RND(10)
77     T[X,Y]=0
78   NEXT
79 NEXT
80 RETURN
```

タッチ配列変数をクリア

プログラムを「PUZZLE7」の名前で保存（SAVE）しましょう。

●モンスターの攻撃

プレイヤーが攻撃した後、モンスターのターンになって、モンスターがプレイヤーに攻撃するようにしましょう。
メインループでタッチ操作が終わった後、モンスターの攻撃サブルーチンを呼びます。

```
 1 '*** PUZZLE8 ***
 （中略）
61 'タッチ クリア
62 FOR Y=0 TO 4
63   FOR X=0 TO 4
64     T[X,Y]=0
65   NEXT
66 NEXT
67 GOSUB @PRTPANEL
68
69 'モンスターのこうげき
70 GOSUB @MONATTACK
71
72 GOTO @MAINLOOP
```

プログラムタイトルを「PUZZLE8」に変更

モンスターの攻撃サブルーチンを呼ぶ

プログラムの最後に、モンスターの攻撃サブルーチンを追加します。

```
169 '=== モンスターのこうげき ===
170 @MONATTACK
171
172 DISPLAY 0
173 LOCATE 0,25
174 COLOR 11
175 PRINT "××× ";MNAME$;"のこうげき！ ×××"
176 AP=RND(21)
177 PRINT "ダメージ:";AP
178 HP=HP-AP
179 DISPLAY 1
180 LOCATE 0,1
181 COLOR 15
182 PRINT "HP=";HP;"  "
183
```

モンスターの攻撃を表示

攻撃ポイントを 0～20 の乱数で設定

プレイヤーの HP を攻撃ポイント分へらす

プレイヤーの HP を表示

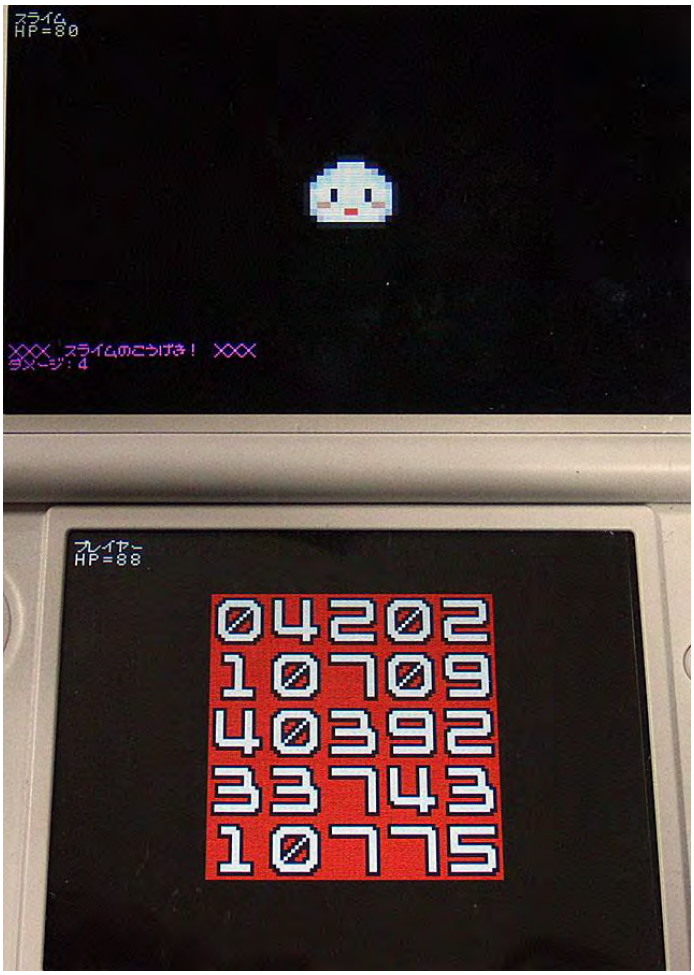
（↓次ページに続く）

パズルに赤い四角を 3 回表示
（攻撃を受けたイメージ）

```
184 FOR I=1 TO 3
185  GFILL 80,32,239,191,RGB(255,0,0)
186  VSYNC 5
187  GFILL 80,32,239,191,RGB(0,0,0)
188  VSYNC 5
189 NEXT
190 VSYNC 120
191
192 DISPLAY 0
193 LOCATE 0,25
194 PRINT " "*48
195 PRINT " "*48
196 RETURN
```

攻撃の文字をスペースで消す

プログラムを実行してみましょう。
プレイヤーが攻撃した後、モンスターが攻撃するようになります。



●ゲームオーバー

今のままだとプレイヤーの HP がマイナスになってもゲームが続いてしまいます。
プレイヤーの HP が 0 以下になったら、ゲームオーバーになるようにしましょう。
メインループの最後を改造して、ゲームオーバーのプログラムを追加します。

```
69 ’モンスターのこうげき
70 GOSUB @MONATTACK
71
72 IF HP>0 THEN @MAINLOOP
73
74 ’=== ゲーム オーバー ===
75
76 DISPLAY 0
77 LOCATE 0,25
78 COLOR 3
79 PRINT ”xxx ゲーム オーバー xxx”
80 VSYNC 120
81 COLOR 15
82 END
83
84 ’=== パネル セット ===
```

プレイヤーの HP が 0 より大きければ戻る。0 以下ならゲームオーバーへ。

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。
プレイヤーが攻撃を受けて、HP が 0 以下になると、ゲームオーバーになります。

プログラムを「PUZZLE8」の名前で保存（SAVE）しましょう。

●タッチしたパネルを消す

今のままだと、タッチした数字パネルがそのまま表示されっぱなしなので、何回も同じ攻撃ができてしまいます。
まずはタッチしたパネルが消えるようにしてみましょう。

これまで、配列変数 T[X,Y]を使って、数字パネルの状態を2つ表しました。

- T[X,Y]=0　通常の状態
- T[X,Y]=1　タッチで選択された状態

ここにもう 1 つ、「消えている状態」を追加します。

- T[X,Y]=0　通常の状態
- T[X,Y]=1　タッチで選択された状態
- T[X,Y]=2　消えている状態（空白）

まずメインループで、タッチ操作終了後のパネルをクリアするプログラムを改造します。
これまでは全部のパネルで「T[X,Y]=0」としてクリアしていましたが、攻撃が発動した時はタッチされているパネルで「T[X,Y]=2」にします。

```
1 '*** PUZZLE9 ***
2
```

プログラムタイトルを「PUZZLE9」に変更

```
61 'タッチ クリア
62 FOR Y=0 TO 4
63  FOR X=0 TO 4
64   IF T[X,Y]==1 AND AT==1 THEN T[X,Y]=2 ELSE T[X,Y]=0
65  NEXT
66 NEXT
67 GOSUB @PRTPANEL
```

パネルがタッチ中で、攻撃が発動している時は、パネルを空白に、そうでなければ通常にもどす

これで、配列変数 T[X,Y]の値の設定は OK です。

次に、パネル表示サブルーチンで、T[X,Y]が 2 だったら空白を表示するように改造します。

```
 95 '=== パネル ひょうじ ===
 96 @PRTPANEL
 97
 98 DISPLAY 1
 99 FOR Y=0 TO 4
100  FOR X=0 TO 4
101   N=P[X,Y]
102   S=X+Y*5
103   GX=80+32*X
104   GY=32+32*Y
105   SPSET S,48+N
106   SPOFS S,GX,GY
107   SPSCALE S,2,2
108    IF T[X,Y]==2 THEN SPHIDE S EL
SE SPSHOW S
109    IF T[X,Y]==1 THEN
110     GFILL GX,GY,GX+31,GY+31,RGB(1
28,128,255)
111    ELSE
112     GFILL GX,GY,GX+31,GY+31,RGB(0,
0,0)
113    ENDIF
114  NEXT
115 NEXT
116 RETURN
```

パネルが空白の時は、数字スプライトを隠す、そうでなければ表示する

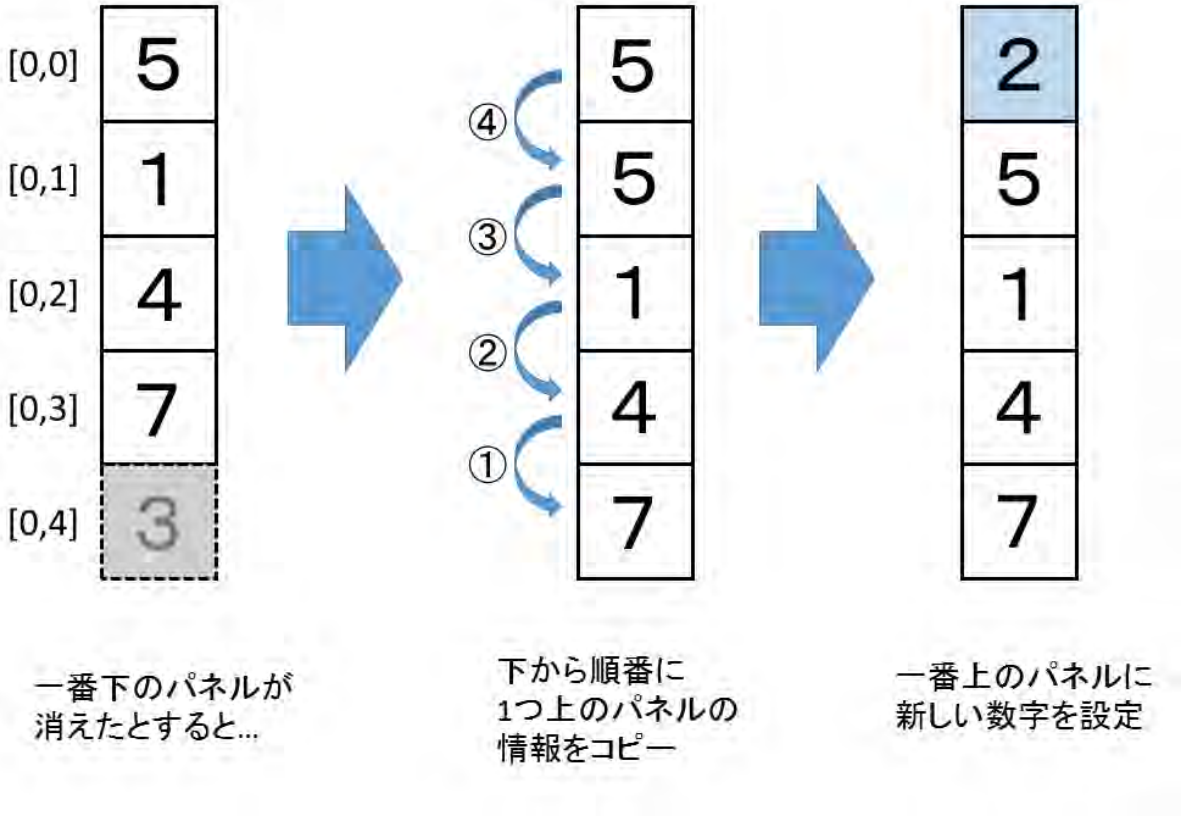
パネルがタッチ中の時は、背景に水色の四角を表示、そうでなければ黒い四角を表示する

プログラムを実行してみましょう。
合計が 10 の倍数になるようにパネルにタッチすると、パネルが消えます。

ただし、次に別のパネルにタッチすると、また表示されてしまいます。
後でパネルを動かして空白をうめるので、そこはそのままにしておきます。

●パネルを下へ落とす

パネルが消えて空白になったら、上のパネルを下へ落としてうめるようにします。
いわゆる「落ち物」パズルゲームの動きです。
例として、一番左側の縦 1 列を取り出して、パネルの動かし方を考えてみましょう。



プログラムの手順としては、以下のようになります。

(1)パネルが空白かどうか(T[X,Y]が 2 かどうか)をチェック。
(2－1)もし空白だったら、そこから 1 つ上の段のパネルの情報を順番にコピーして、一番上の段に新しいパネルを設定。
(2－2)もし空白でなかったら、落とす必要が無いので次へ。
(3)以上の(1)～(2)を、一番下の段から一番上の段まで 5 段分くり返す。
(4)以上の(1)～(3)を、一番左の列から一番右の列まで 5 列分くり返す。

自分でもパネルを紙に描いて、動かし方を考えてみてください。
いきなりプログラムを書き出すとわからなくなるので、最初に紙の上で動かし方を考えてから、プログラムを書くといいです。

メインループで、パネルのタッチ状態をクリア後、パネルを落とすサブルーチンを呼びます。

```
61 'タッチ クリア
62 FOR Y=0 TO 4
63  FOR X=0 TO 4
64   IF T[X,Y]==1 AND AT==1 THEN T[X,
Y]=2 ELSE T[X,Y]=0
65  NEXT
66 NEXT
67 GOSUB @PRTPANEL
68
69 'パネルをおとす
70 IF AT==1 THEN GOSUB @FALLPANEL
71
72 'モンスターのこうげき
73 GOSUB @MONATTACK
```

攻撃フラグ AT が 1 なら、
パネルを落とすサブルーチンを呼ぶ

プログラムの最後に、パネルを落とすサブルーチンを追加します。

```
211 '=== パネル おとす ===
212 @FALLPANEL
213
214 FOR FX=0 TO 4
215  FOR FY=4 TO 0 STEP -1
216
217   IF T[FX,FY]==2 THEN
218
219    '1だん おとす
220    FOR PY=FY TO 1 STEP -1
221     P[FX,PY]=P[FX,PY-1]
222     T[FX,PY]=T[FX,PY-1]
223     T[FX,PY-1]=2
224     GOSUB @PRTPANEL
225     VSYNC 10
226    NEXT
227
228    'いちばんうえに しんパネル
229    P[FX,0]=RND(10)
230    T[FX,0]=0
231    GOSUB @PRTPANEL
232    VSYNC 10
```

5列分くり返し

5段分くり返し

もしパネルが空白だったら、
1段落とす処理をする

1つ上のパネルの
情報をコピー

1つ上のパネルは空白にして、
画面に表示

一番上に、新しいパネルを設定

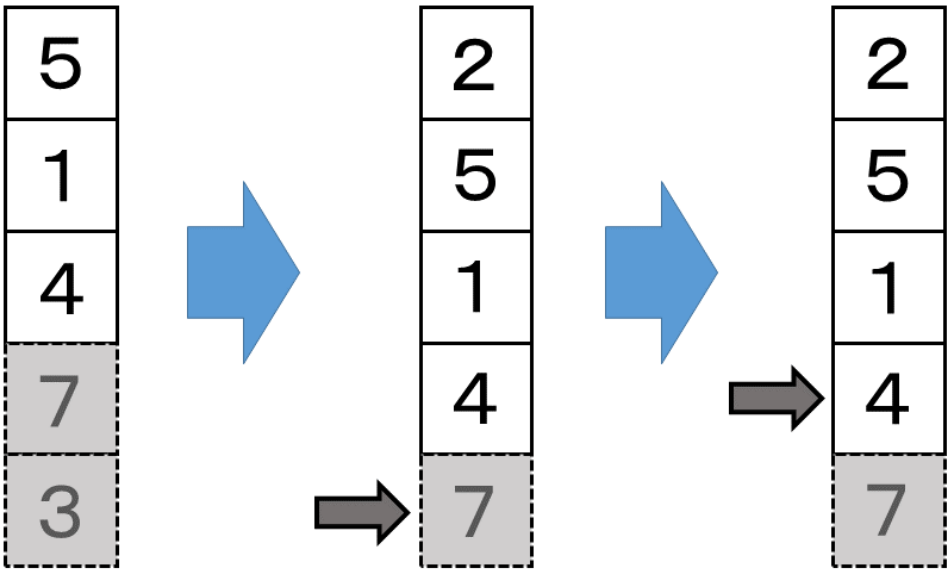
```
233
234    ENDIF
235
236  NEXT
237 NEXT
238
239 RETURN
```

最後に、パネルを落とすプログラムをテストするために、メインループの合計判定の所を改造して、パネルの合計数が10の倍数でなくても攻撃するようにします。（後でもどします）

```
57  'こうげき
58  IF GS MOD 10>=0 THEN AT=1:
GOSUB @ATTACK
```

「==」を「>=」に変えて、合計数 GS を 10 で割った余りがいくつでも、攻撃するようにする

プログラムを実行して、パネルをいろいろタッチして試してみましょう。
パネルを1個だけタッチすると、うまい具合にパネルが落ちます。
しかし、縦に2個以上続けてタッチすると、1段分しか落ちてくれません。
これは、パネルを落とすプログラムが、縦に2個以上空白が続くケースを想定していないからです。（空白かどうかを、IF 命令で1回しかチェックしていない）



下の2枚のパネルが
連続で消えたとすると…

一番下のパネルが
空白なので、上の
パネルを1段おとす

そのままチェック位置を上へ
移してしまうと、一番下の
空白パネルが取り残される

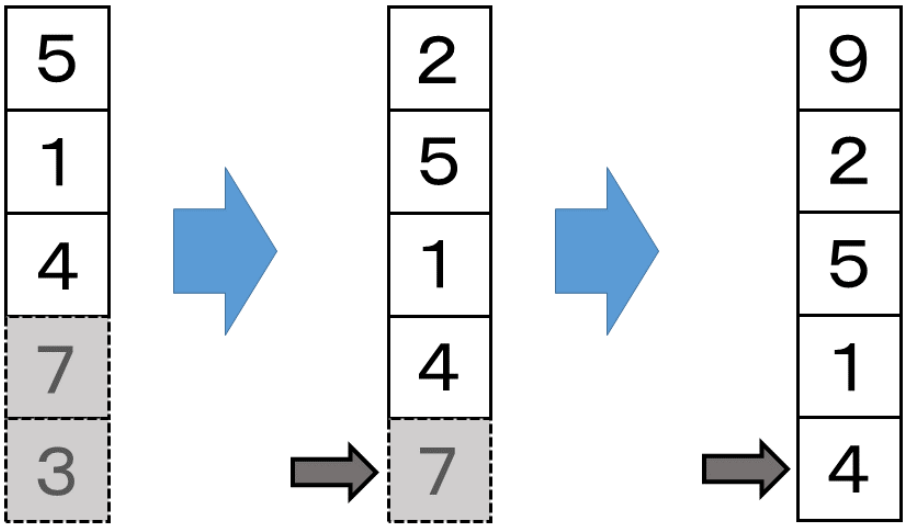
IF 命令の代わりに WHILE 命令を使って、パネルが空白である限りは続けてパネルを落とすようにしましょう。

```
211 '=== パネル おとす ===
212 @FALLPANEL
213
214 FOR FX=0 TO 4
215   FOR FY=4 TO 0 STEP -1
216
217     WHILE T[FX,FY]==2
    （中略）
231       GOSUB @PRTPANEL
232       VSYNC 10
233
234     WEND
235
236   NEXT
237 NEXT
```

IF を WHILE に変えて、パネルが空白である限り、落とし続ける

ENDIF を WEND に変える

これで、縦に空白が並んでいても、パネルが下まで落ちるようになります。



下の2枚のパネルが
連続で消えたとすると…

一番下のパネルが
空白なので、上の
パネルを1段おとす

チェック位置のパネルが空白である限り
そのままその位置をチェックし続けて
パネル落としを続ける

ちゃんとパネルが落ちることが確認できたら、パネル合計数の判定部分を元に戻します。

```
57 'こうげき
58 IF GS MOD 10==0 THEN AT=1:
GOSUB @ATTACK
```

「>=」を「==」にもどす

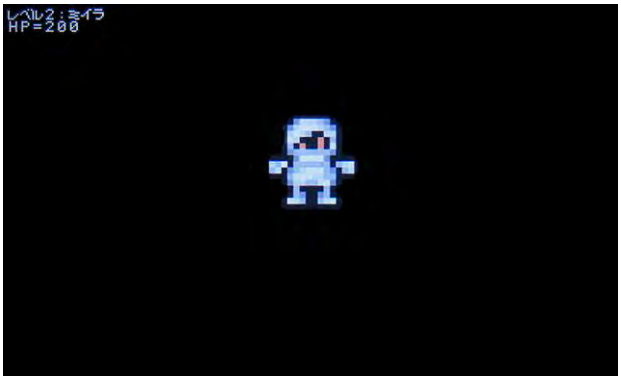
それから、パネルを落とすサブルーチンでも、動きをわかりやすくするためにパネルをゆっくり落としていたので、遅くしている VSYNC 命令をコメントアウトします。

```
211 '=== パネル おとす ===
212 @FALLPANEL
213
214 FOR FX=0 TO 4
215  FOR FY=4 TO 0 STEP -1
216
217   WHILE T[FX,FY]==2
218
219    '1ぜん おとす
220    FOR PY=FY TO 1 STEP -1
221     P[FX,PY]=P[FX,PY-1]
222     T[FX,PY]=T[FX,PY-1]
223     T[FX,PY-1]=2
224     GOSUB @PRTPANEL
225     'VSYNC 10
226    NEXT
227
228    'いちばんうえに しんパネル
229    P[FX,0]=RND(10)
230    T[FX,0]=0
231    GOSUB @PRTPANEL
232    VSYNC 10
```

先頭に「'」を付けてコメントにする

プログラムを「PUZZLE9」の名前で保存（SAVE）しましょう。

●モンスターの種類を増やす



これでパズルバトルのプログラムはだいたいできました。
今のままだとずっとスライムと戦い続けてしまって面白くないので、モンスターの種類を増やしましょう。

モンスターの名前、スプライト番号、レベルなどを、最初に配列変数を使って設定します。

```
 1 '*** PUZZLE10 ***
 2
 3 '=== しょき せってい ===
 4
 5 DIM P[5,5],T[5,5]
 6
 7 'モンスター
 8 DIM MNA$[5],MSP[5]
 9 FOR I=0 TO 4
10  READ MNA$[I],MSP[I]
11 NEXT
12
13 ML=0
14
15 '=== バトルがめん ===
```

プログラムタイトルを「PUZZLE10」に変更

「初期設定」のコメントを追加

モンスター名、スプライト番号の配列宣言

モンスター名、スプライト番号の配列にデータを読み込み

モンスターのレベル設定。最初は 0。

プログラムの最後に、モンスターのデータを並べます。

```
251 RETURN
252
253 '=== モンスター データ ===
254
255 DATA "スライム",1060
256 DATA "ミイラ" ,980
257 DATA "ガイコツ",1000
258 DATA "キメラ" ,1040
259 DATA "ゴースト",1020
```

新しく、READ（リード）命令、DATA（データ）命令が出てきました。

```
READ    MNA$(I)      ,MSP(I)      …
        変数名 1     変数名 2     …
```

DATA 命令で指定されたデータを変数に読み込みます。

```
DATA    ”スライム”      ,1060      …
          データ 1        データ 2     …
```

READ 命令で読み込むデータを指定します。

DATA 命令で指定したデータを、READ 命令で順番に読み込みます。多数のデータを多数の変数に一度に設定できます。
実際のプログラムでは、今回のように FOR～NEXT と組み合わせて、配列変数へ順番にデータを読み込ませることが多いです。
DATA 命令の位置は、READ 命令より前でも後でも構いませんが、プログラムの最後に書くことが多いです。

モンスターのデータは、5 種類設定しています。
スプライト番号とキャラクターは、スマイルツール（キーボードの「SMILE」ボタン）→「SPDEF」で確認してください。

●登場モンスターを切り替える

このモンスターのデータを使って、レベルが上がると登場するモンスターが切り替わるようにします。
まず、バトルを開始する時に、モンスターのレベルを設定します。

```
13 ML=0
14
15 '=== バトルかいし ===
16 @BATTLESTART
17
18 ACLS
19 XSCREEN 2
20
21 ML=ML+1
22 HP=80+20*ML
23
24 DISPLAY 1
```

モンスターレベルを1上げる

モンスターレベルからプレイヤーの HP を計算

モンスターレベル ML は、プログラムを実行した最初は、13 行目「ML=0」で 0 にしているので、21 行目で 1 増やすと 1 になります。
22 行目ではプレイヤーの HP を計算しています。最初は ML=1 なので、HP=80+20×1=100 になります。以後、ML が 1 上がると HP が 20 増えて、プレイヤーが少しだけ強くなります。

ここで設定した ML を使って、登場させるモンスターの設定を変えます。

```
132 '=== モンスター セット ===
133 @SETMONSTER
134
135 M=(ML-1) MOD 5
136 MNAME$=MNA$[M]
137 MS=MSP[M]
138 MHP=100*ML
139 RETURN
140
141 '=== モンスター ひょうじ ===
```

モンスター切り替え用変数 M（0～4）を計算

モンスター名、モンスターのスプライト番号、モンスターの HP を設定

最初にモンスターの切り替え用変数 M を設定します。MOD 演算子を使って、ML-1 を 5 で割った余り（0～4）を計算しています。その結果、M の値は 0～4 をくり返します。

ML	ML-1	(ML-1) MOD 5 (→M)
1	0	0
2	1	1
3	2	2
4	3	3
5	4	4
6	5	0
7	6	1
8	7	2
9	8	3
10	9	4
11	10	0
12	11	1
:	:	:

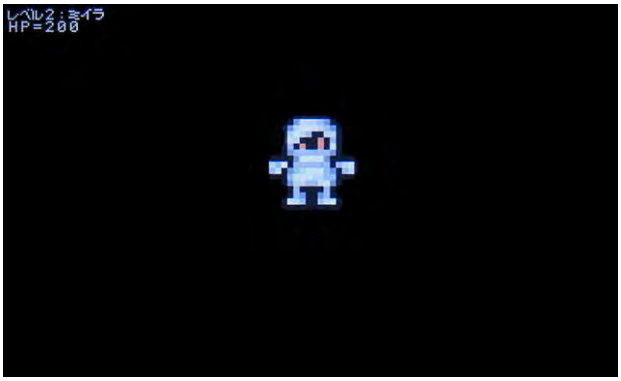
この M の値を使って、モンスター名 MNAME$、モンスターのスプライト番号 MSP、モンスターの体力 MHP を設定しています。

モンスターのレベルに合わせて、モンスターの攻撃も激しくなるようにしましょう。
モンスターの攻撃サブルーチンを改造します。

```
194 '=== モンスターのこうげき ===
195 @MONATTACK
196
197 DISPLAY 0
198 LOCATE 0,25
199 COLOR 11
200 PRINT "xxx ";MNAME$;"のこうげき！ xxx"
201 AP=RND(21)*ML        攻撃ポイントをモンスターレベルから計算
202 PRINT "ダメージ:";AP
203 HP=HP-AP
```

攻撃ポイント AP を乱数（0～20）×ML として、モンスターレベルが上がるほど AP が大きくなるようにします。

プログラムを実行してみましょう。
プレイヤーがモンスターを倒す度に、レベルアップした次のモンスターが出てきます。
モンスターの HP が増えて、攻撃も激しくなるので、倒すのがだんだん難しくなります。
レベル 6 まで行くと、またスライムに戻りますが、1 周目よりも強くなっています。



プログラムを「PUZZLE10」の名前で保存（SAVE）しましょう。

【できる人は】

- プログラム最後の DATA 命令で書かれているモンスター名やスプライト番号を変えると、出てくるモンスターを変えられます。
- 今は 5 種類のモンスターを設定していますが、種類を増やすこともできます。種類を増やした場合は、「モンスターセット」のサブルーチンで、切り替え変数 M の計算式を変えましょう。例えば 10 種類に増やしたら、「MOD 10」として 10 で割った余りを計算します。
- モンスターの体力 MHP の設定や、モンスターの攻撃ポイント AP の計算式を変えると、モンスターの強さが変わります。