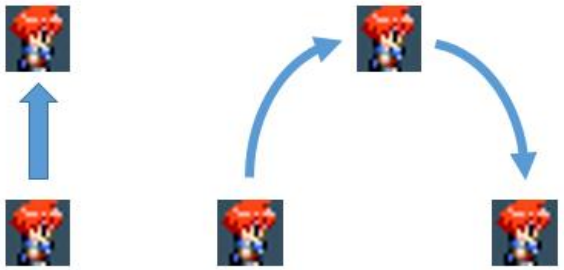
プチコンでアクションゲームを作る（３）

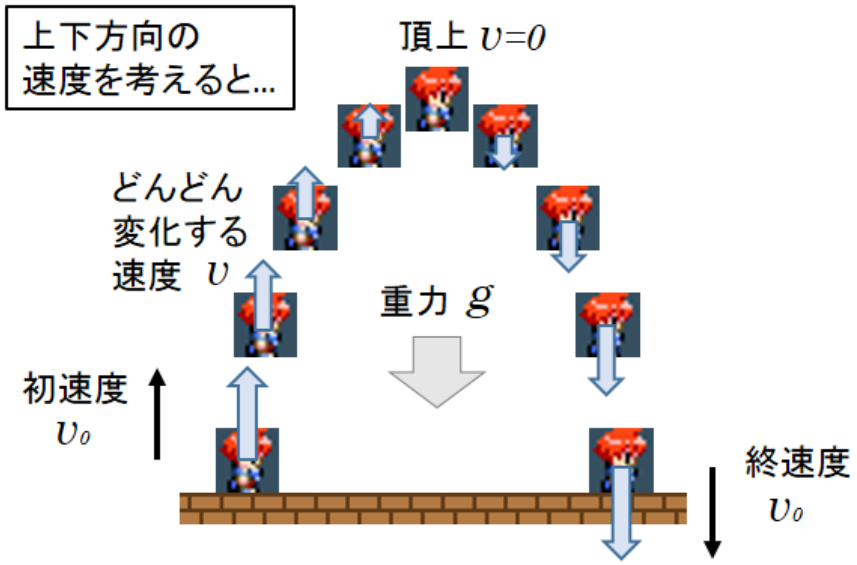
●キャラクターをジャンプさせる

ここまでで、背景に沿ってキャラクターを動かすことができました。
今度は、アクションゲームでよくある、キャラクターをジャンプさせる動きを作ってみましょう。
キャラクターをジャンプさせると、アクションの動きが広がっておもしろくなります。



★そもそもジャンプの動きって？

プログラムからちょっと離れて、上へジャンプする時の動きがどんなものか考えてみましょう。



ここでは上下方向の速度だけ考えます。
最初に地面をけってジャンプすると、ある速度で上へ飛び出します。
その後は、下向きに重力がずっとかかるので、だんだん速度が遅くなり、頂上では速度が 0 になります。
今度は下向きに速度が増えていって、地面についた時には最初の速度と同じ（ただし下向き）になります。

> 高校の物理で習いますが、このように一定方向に力（重力）がかかる時の動きは「等加速度運動」と言います。
> 初速度を v_0、重力加速度を g とすると、t 秒後の速度 v は、
> $$v = v_0 + gt$$

★プログラムではどうする？

この上下方向のキャラクターの動きを、どうやってプログラムにするか考えてみましょう。
これまでのプログラムでは、キャラクターを一定の速度で動かしていました。

```
58 '--- ウエ ---
59 @UE
   （※中略）
69 Y=Y-2
70 S=76
71 RETURN
```

上へ 2 ドット動かす

ジャンプする時は、上下方向の速度がどんどん変わっていくので、この動かす量を変数 DY で表すことにしましょう。
また、前のページで出てきた、初速度 V0、重力 G などの変数も新しく作ります。

まず、プログラムの最初で、新しく変数を設定します。

```
 1 '*** ACT5 ***
 2
 3 ACLS
 4
 5 'サイショ
 6 X=0
 7 Y=168
 8 SPSET 0,64,2,0,0,2
 9 SPANIM 0,4,10
10 SPOFS 0,X,Y
11
12 DY=0
13 V0=-8
14 G=1
15 JP=0
16
17 '--- ステージ ---
```

タイトルを「ACT5」に変更

DY　y 方向（上下）の移動量
V0　ジャンプの初速度
G　 重力加速度
JP　ジャンプ状態を表す
　JP=0　ジャンプしていない
　JP=1　ジャンプ中

ジャンプの初速度 V0 をマイナスの値にしているのは、プチコン画面で上へ移動する時は、y 座標を減らす方向だからです。
後のプログラムで使うために、ジャンプの状態を表す変数 JP も設定しています。

次に、キャラクターを歩かせるプログラムで、ジャンプするボタンをチェックするようにします。
ジャンプは A ボタンにしましょう。

```
42 '--- ｱﾙｸ ---
43 @ARUKU
44
45 'ｼﾞｭｳｼﾞ ｷｰ
46 B=BUTTON()
47 IF B==1 AND Y>0 THEN GOSUB @UE
48 IF B==2 AND Y<175 THEN GOSUB @SHITA
49 IF B==4 AND X>0 THEN GOSUB @HIDARI
50 IF B==8 AND X<239 THEN GOSUB @MIGI
51 'Aｼﾞｬﾝﾌﾟ
52 IF B==16 AND JP==0 THEN GOSUB @JUMP
53
54 'ﾑｷｶﾞ ｶﾜｯﾀﾗ ｽﾌﾟﾗｲﾄ ｻｲｾｯﾃｲ
55 IF B!=B0 THEN SPSET 0,S,2,0,0,2:SPAN
IM 0,4,10
56
57 IF JP==1 THEN Y=Y+DY:DY=DY+G
58 SPOFS 0,X,Y
```

A ボタンを押した時は、BUTTON 関数の値は 16 になるので、それをチェックします。
また、ジャンプの途中で A ボタンが押されたらまたジャンプしてしまうのは困るので、ジャンプ状態変数 JP が 0 の時だけ、ジャンプのサブルーチンを呼ぶようにします。

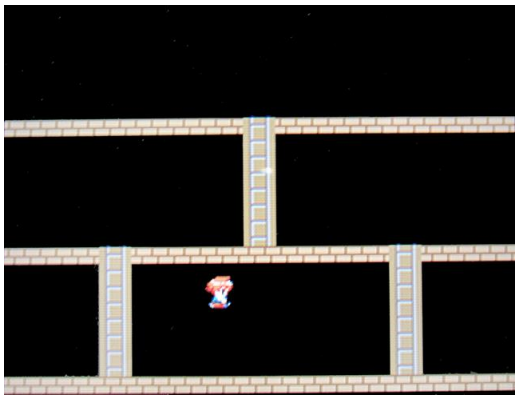
ジャンプ中の時は、キャラクターの y 座標を DY だけ移動します。
また、重力が働くので、DY の値を G だけ変化させます。

最後に、ジャンプの設定をするサブルーチンを、プログラムの最後に追加します。

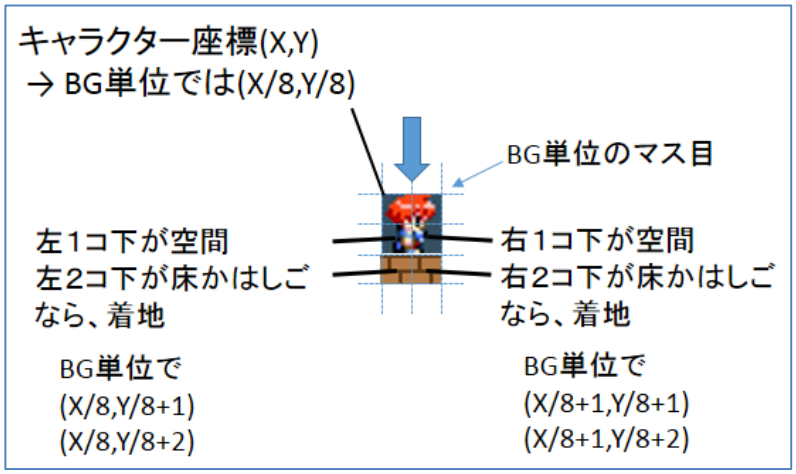
```
115 '--- ｼﾞｬﾝﾌﾟ ---
116 @JUMP
117
118 JP=1
119 DY=V0
120 RETURN
```

ジャンプ状態変数 JP を 1（ジャンプ中）に、
DY を初速度 V0 にする

これでプログラムを動かしてみましょう。
A ボタンを押すと、キャラクターがジャンプします。
ただし、着地するプログラムを作っていないので、
そのまま下へ落ちていってしまいます。



これではゲームにならないので、足元に床やはしごがある時は、着地して止まるようにしましょう。
キャラを上下左右に動かした時と同じように、背景（BG）との位置関係で、着地する条件を考えてみます。



ちょうど床やはしごに着地した時は、キャラクター座標の 1 個下は空間、2 個下が床・はしごになるはずです。
その条件が成り立った時は、ジャンプを止めましょう。

キャラクターの座標を変化させて表示した後に、着地のチェックをします。

```
61 'コンカイノ キーヨ ホゾン
62 B0=B
63
64 'チャクチ?
65 BX=X/8
66 BY=Y/8
67 BGREAD(1,BX,BY+1),C1,P,H,V
68 BGREAD(1,BX,BY+2),C2,P,H,V
69 BGREAD(1,BX+1,BY+1),C3,P,H,V
70 BGREAD(1,BX+1,BY+2),C4,P,H,V
71 IF C1==0 AND C2!=0 THEN JP=0:DY=0
72 IF C3==0 AND C4!=0 THEN JP=0:DY=0
73
74 GOTO @ARUKU
```

左 1 個下、2 個下の BG を読み取り

右 1 個下、2 個下の BG を読み取り

○左 1 個下が空間＆2 個下が床・はしご
○右 1 個下が空間＆2 個下が床・はしご
→その時はジャンプを止める

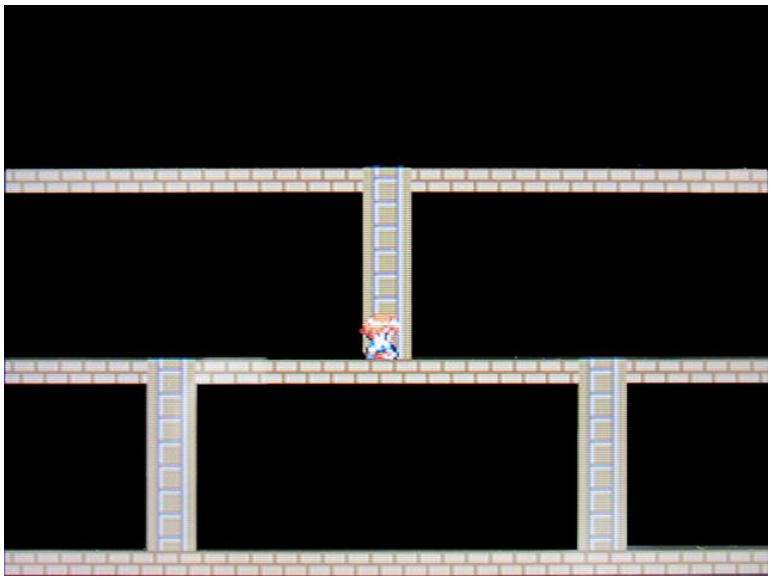
これでプログラムを動かしてみましょう。
ジャンプした後、ちゃんと床に着地して止まるようになります。

…が、何度もやっていると、ちょっと変な動きをする時があるのがわかります。
ちょうどはしごがある所でジャンプすると、床で着地しないで下へ落ちてしまいます。
これは前のページの図を見ればわかるのですが、背景にはしごがある時は、背景が「空間」ではないので、「1 個下が空間＆2 個下が床・はしご」という条件が成り立たないのです。
はしごの所でジャンプした時も、2 個下が床だったらそこで着地して止まるようにしましょう。
着地のプログラムに 1 行追加します。

```
64 'チャクチ?
65 BX=X/8
66 BY=Y/8
67 BGREAD(1,BX,BY+1),C1,P,H,V
68 BGREAD(1,BX,BY+2),C2,P,H,V
69 BGREAD(1,BX+1,BY+1),C3,P,H,V
70 BGREAD(1,BX+1,BY+2),C4,P,H,V
71 IF C1==0 AND C2!=0 THEN JP=0:DY=0
72 IF C3==0 AND C4!=0 THEN JP=0:DY=0
73 IF C2==677 OR C4=677 THEN JP=0:DY=0
74
75 GOTO @ARUKU
```

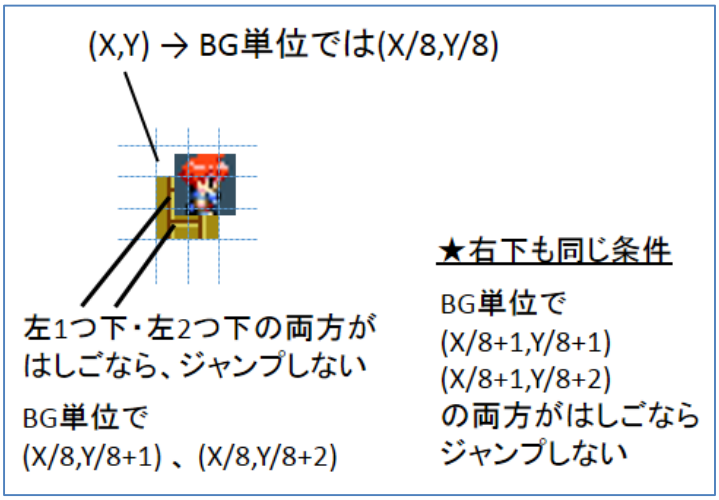
左 2 個下が床　または　右 1 個下が床の時はジャンプを止める

これで、はしごの所でジャンプしても、床で着地して止まるようになります。



また、他にもおかしな動きをする時があります。
はしごの途中にいる時に A ボタンを押すと、そこでジャンプして変な着地をしてしまいます。
これを防ぐには、ジャンプの設定をするサブルーチンで、「はしごの途中にいる時はジャンプしない」と判断する処理を入れればいいでしょう。

キャラクターと背景の組み合わせを考えると、はしごの途中にいる時は、キャラクター座標の 1 個下も 2 個下も両方はしごになるはずです。
その条件が成り立った時は、ジャンプしないようにしましょう。



ジャンプの設定サブルーチンを、以下のように改造します。

```
126 '--- ｼﾞｬﾝﾌﾟ ---
127 @JUMP
128
129 BX=X/8
130 BY=Y/8
131 BGREAD(1,BX,BY+1),C1,P,H,V
132 BGREAD(1,BX,BY+2),C2,P,H,V
133 BGREAD(1,BX+1,BY+1),C3,P,H,V
134 BGREAD(1,BX+1,BY+2),C4,P,H,V
135 IF C1==28 AND C2==28 THEN RETURN
136 IF C1==29 AND C2==29 THEN RETURN
137 IF C3==28 AND C4==28 THEN RETURN
138 IF C3==29 AND C4==29 THEN RETURN
139
140 JP=1
141 DY=V0
142 RETURN
```

左 1 個下、2 個下の BG を読み取り

右 1 個下、2 個下の BG を読み取り

1 個下・2 個下ともはしごだったらジャンプしないでリターン

プログラムを実行してみましょう。
はしごの途中でジャンプできなくなります。

さて、ジャンプの基本の動きはこれで OK です。
ただ、今のままだと、十字キーと A ボタンを同時に読み取れるようになっていないので、「横に動きながらジャンプする」ことができません。
BUTTON 関数の読み取りの所を改造しましょう。

```
42 '--- アルク ---
43 @ARUKU
44
45 'ｼﾞｭｳｼﾞ キー
46 B=BUTTON()
47 IF (B AND 1)==1 AND Y>0   THEN GOS
UB @UE
48 IF (B AND 2)==2 AND Y<175 THEN GOS
UB @SHITA
49 IF (B AND 4)==4 AND X>0   THEN GOS
UB @HIDARI
50 IF (B AND 8)==8 AND X<239 THEN GOS
UB @MIGI
51 'Aｼﾞｬﾝﾌﾟ
52 IF (B AND 16)==16 AND JP==0 THEN G
OSUB @JUMP
```

BUTTON 関数の値が入った変数 B に対して、AND（論理演算）を使って処理をしています。
これは 2 進数の知識が必要なので、難しいのでここでは説明しません。

これで、横に動きながらジャンプできるようになります。
自分のキャラクターの基本的な動きは完成です。