ﾌﾟﾁｺﾝ
1ｶﾞﾒﾝﾌﾟﾛｸﾞﾗﾑ
ﾉｽｽﾒ

ｺﾚ1ﾂﾃﾞ
1ｶﾞﾒﾝﾌﾟﾛｸﾞﾗﾑﾉ
ｺﾞｸｲｶﾞﾜｶﾙ！

「プチコン」は皆さんご存じですか？

この冊子を手に取っている人で知らない人は居ないと思うのですが、プチコンとはスマイルブームから発売されているDSiウェアでBASIC言語によってさまざまなプログラムを作ることができるアプリです。自分でプログラムを作ることができるためRPGツクールなどとは異なりあらゆるジャンルのゲームやツールなどを作ることができます。

80年代に8bitパソコンでBASICを使用していた人ならば「BASICだと速度面で辛そう」というイメージがあるかもしれません。プチコンはBASICのみでマシン語は使えないのですが、BASICといっても当時のパソコンと比べて数10倍～100倍の速度が得られます。そのためプチコンゲームをWeb検索するとファミコンやメガドラの市販ゲームに匹敵するクオリティのゲームさえも見つかります。

ファミコンレベルのゲームでもちゃんと作り込もうとするならば個人で作るにはハードルが高いです。幸いプチコンにはプリセットデータがたくさんあるので普通にゲームを作るだけならばキャラや音楽はそれを使えばいいので簡単です。とはいえ、せっかく作るならば他の人を「あっ」と言わせるようなものを作ってみたいと考えている人も多いことでしょう。

そこで、オススメなのが**「1画面プログラム」**です。1画面プログラムとは何かというと編集画面の1画面分に収まるサイズのプログラムです。プチコンの編集画面は29文字×24行で構成されているためその範囲で収まるプログラムが1画面プログラムとなります。（改行マークに関しては画面からはみ出ても良い）

1画面プログラムはサイズが小さいので簡単に作れるためシンプルなゲームやツールを作るのに向いています。このように書くと「単なる初心者向けのプログラム」と思うかもしれません。**1画面プログラムというのは作るのは簡単なのですが、実際に作り込もうとするならばパズル的な要素が要求されるため非常に奥が深い**のです。したがって、BASIC初心者だけではなく中上級者の腕試しにも良いと思います。中上級者が限界に挑もうとしたら上記のような昔の市販ゲームレベルのクオリティが要求されてきますが、1画面プログラムであればお手軽に限界に挑戦ができます。（変な日本語だ）

では、実際に何かプログラムを作ってみましょう。

ルールが単純で誰もが楽しめるゲームということで「100m走」のゲームにしましょう。Aボタンと Bボタンを交互に押してスタートラインからゴールラインまでキャラを走らせるというシンプルなゲームです。非常にシンプルであるためBASIC初心者でも作るのは容易でしょう。1行29文字に収まる範囲内でマルチステートメントで記述すれば1画面化もそれほど難しいものではありません。

ということで、1画面に収まるサイズの「100m走」を試しに作ってみました。（サンプルA）

```
01 W=99.99↵
02 @START↵
03 ACLS:GCLS 10:X=0:F=0:T=0↵
04 GFILL 0,120,255,160,7↵
05 GLINE 20,120,20,160,15↵
06 GLINE 240,120,240,160,15↵
07 SPSET 0,64,3,0,0,0↵
08 SPANIM 0,4,4:SPOFS 0,4,130↵
09 LOCATE 9,4:?"WR ";W↵
10 LOCATE 9,9?"READY ":WAIT 60↵
11 ?"GO !":BGMPLAY 0↵
12 @MAIN↵
13 B=BUTTON():T=T+1/60:P=0↵
14 LOCATE 9,5:?"TIME";T↵
15 IF B==16 AND F==0THEN F=1:P=2↵
```

```
IF B==32 AND F==1THEN F=0:P=2
X=X+P:SPOFS 0,X+4,130:VSYNC 1
IF X<224 THEN @MAIN
BGMPLAY 4
LOCATE 9,9:?"GOAL IN !"
WAIT 60:IF W>T THEN W=T
@LOOP
IF BUTTON()==16THEN @START
GOTO @LOOP
```

PRG:OCHA100A

やっていることは単純なのでリストを見たら理解できるのではないでしょうか。簡単に内容を説明すると最初に初期設定とコースの描画を行いメインルーチン内ではボタンがちゃんと交互に押されているかの判定を行いタイムを1/60秒ずつ加算し、ゴールラインに達するまでそれを繰り返しているだけです。ゴールをしたら最高タイム（WR）と比較してそれより速ければ最高タイムの更新を行い、Ⓡボタンを押すことでリトライを可能にしています。

一応ゲームの形になっているもののプレイをしてみると不満に思う人がいるかもしれません。リストの長さの制限があるため画面があまりに寂しいというのが真っ先に感じるでしょうし1/60秒単位なのにタイムも無駄に小数第3位まで表示しなくても良いと感じるかもしれません。

あとゲームバランスも良いかどうかも微妙です。ゲームを作る場合には**ゲームバランスは非常に重要**です。ゲームバランスを調整するためにはテストプレイを十分に行い自分がベストと思うラインを見つけることが必要になってきます。これはパラメータが多いRPGなどでは複雑なのですが、100m走ならば誰でも簡単にバランス調整が可能になります。

それは現実の100m走において10秒を切ること（9秒台を出すこと）がトップレベルのラインになっているように100m走タイプのゲームだと連射の速い人で9秒台になるくらいがちょうど良くなるためです。現状では速い人だと7~8秒というタイムになってしまっているため連射が速い人だと超人的な記録を連発してしまいます。

これをちょうど良い感じにするには1回押した時の移動量を調整すれば簡単に調節が可能ですが、今度は10秒台くらいのタイムが出にくくなるという問題があります。それを改善するには現状のように連射速度が2倍なら2倍良いタイムが出るのではなく1.5倍とかになるようにしなくてはなりません。

いろいろ不満点があっても現状でぴったり24行なのでこれが限界だと思うかもしれません。

本当にそうでしょうか？　実は、1画面プログラムは「24行がすべて埋まったら完成」ではなくて、それでようやく**1画面プログラムを作り込むためののスタートラインに立った**だけなのです。

プチコンにおいては1画面プログラムなどで使えるプログラムリストを短縮するテクニックが多数あるためそれを知っているのと知らないのとでは大きな違いが生じます。
主なものを羅列していけば次のようなものがあります。

リスト短縮テクニック（主要なもの）

スペースと コロンの省略	プチコンでは多くの場合省略が可能であり、順番を並べ替えることで省略可能になる場合もある。
カッコの省略	マニュアルに記述されている演算優先順位を把握してやればカッコを外しても問題ない場合が分かる。 A=(B>2)　→　A=B>2　　これはOK A=4*(B>2)　→　A=4*B>2　　これはNG

論理式	論理式は条件を満たす時は「1」、満たさない時は「0」の値をとります。
	IF (B AND 4)=4 THEN X=X-1
	IF (B AND 8)=8 THEN X=X+1　→　X=X-((B AND 4)=4)+((B AND 8)=8)

！（論理否定）	論理否定は、「1（厳密に言えば0以外）を0」に「0を1」にできます。 A=(A=0)　→　A=!A SGN (A)　→　!!A　（A≧0の場合）
＊ （文字列の掛け算）	?"AAAAAAAAA"　→　?"A"*9　（5文字以上の時に短縮できる） IF A=2THEN ?"HELLO"　→　?"HELLO"*!(A-2)　（A=2以外のときは0文字表示している）
ＡＮＤ	A=B%16　→　A=B AND 15　（B≧0の場合）　※16に限らず2の累乗の剰余が可能 X=X-((B AND 4)=4)+((B AND 8)=8)　→　X=X-(4AND B)/4+(8AND B)/8 （論理式を使わずにビット演算ANDで得た値をそのまま使うことで短縮可能になる）
ＢＥＥＰ	BEEP 0　→　BEEP　（引数を省略したら0番の音が鳴る） BEEP ,,Z*127　→　BEEP,,-Z　（第3引数が負の数の時は127と同じ扱いになる）
ＢＧＰＵＴ	BGPUT 0,X,Y,10,0,0,0　→　BGPUT 0,X,Y,10 （パレット0で横反転、縦反転をしない場合は最後の3つの引数を省略可能） 通常の書き方　　BGPUT 0,X,Y,C,P,H,V スクリーンデータを使用　BGPUT 0,X,Y,C+H*1024+V*2048+P*4096 ↓（キャラ番号をC、パレット番号をP、横反転の有無をH 縦反転の有無をVとした場合） BGPUT 0,X,Y,345,12,1,1　→　BGPUT 0,X,Y,52569
ＤＩＭ	要素数が10以下の配列変数は宣言無しに使用できる テーブル化することでIF文を大幅に減らせる場合がある
ＦＬＯＯＲ	変数Aの値を整数化する場合 0 OR A　（ビット演算子ORを使う）※ビット演算では小数部分は無視される A-A%1　（剰余を使う） A/Z*Z　（乗除算を使う）※あらかじめZ=4096としておく（プチコンの演算精度を活用）

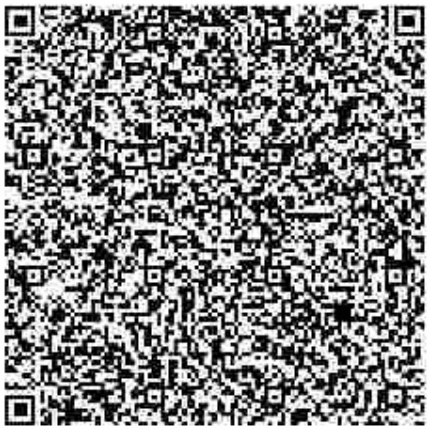
<table>
<tr><td>FOR～NEXT</td><td>
<table>
<tr><td>FOR I=0TO 1
（処理内容）
I=X<255
NEXT
X≧255で終了</td><td>FOR I=1TO X<255
（処理内容）
I=0
NEXT
X≧255で終了</td><td>FOR I=1TO X<255
（処理内容）
NEXT
X<255の時1回実行</td><td>FOR X=0TO 255
（処理内容）
NEXT
X≧255で終了</td></tr>
</table>
※GOTOを使用した場合にはラベルが必要になるため1画面プログラムでは不利になる
</td></tr>
<tr><td>IF</td><td>IF文は条件式の値が「0」か「0以外」かでTHEN以下を実行するかが決まる。
IF A!=0 THEN ～ → IF A THEN ～
IF A>0 AND B>0 THEN ～ → IF A*B THEN ～ （A,Bともに負数でない場合）
IF A>0 AND B=3 THEN ～ → IF A*(B=3)THEN ～
IF A!=1 THEN ～ → IF A-1THEN ～
IF A=0 THEN ～ → IF!A THEN ～</td></tr>
<tr><td>PRINT</td><td>PRINT A → ?A （超基本の省略形）
?A, ※表示の後にカンマを付けることで表示桁数が減った時の残像を防げる</td></tr>
<tr><td>％（剰余）</td><td>A-A%1 変数Aを整数化（負数の場合はFLOORとは異なる）※OOR A は FLOOR(A) と同等
(Y-X+3)%3 じゃんけんの勝敗判定（0＝あいこ、1＝勝ち、2＝負け）
※X:自分 Y:相手 グー＝1、チョキ＝2、パー＝3の場合
X=96%85 じゃんけんでグーチョキパーをボタンで選択
※BにBUTTON関数の値が入っておりグーチョキパーが[B][Y][X]ボタンの時
B AND 15 → B%16 B AND 12 → B%16-B%4 ※Bは0以上の整数の時
X=X-((B AND 4)=4)+((B AND 8)=8) → X=X-(B%16-B%4+1)%3+1
Y=Y-((B AND 1)=1)+((B AND 2)=2) → Y=Y-(B%4+1)%3+1
※BにBUTTON関数の値が入っているとき、X、Yを組み合わせて8方向移動が可能
X=X-(B+1)%3+1 左右移動のみできればいいのならばこれでもOK
・・・他多数
剰余を使うことで条件のふるい分けを行い幅広い用途でリスト短縮が可能になります。
その場合は差分を取るなどをしていかに効率よくパターン化できるかが重要となります。</td></tr>
<tr><td></td><td>これらはあくまで一例です。まだ様々なリスト短縮方法があります。
詳しくはWebで http://ww5.tiki.ne.jp/~ochane/petitcom/tips/list.htm</td></tr>
</table>

これらのテクニックを用いてサンプルAのプログラムリストを短縮してみました。（サンプルB）

```
01 W=99.99FOR I=0TO 1ACLS:T=0↵
02 GCLS 10GFILL 0,120,255,160,7↵
03 GLINE 20,120,20,160,15A=16↵
04 GLINE 240,120,240,160,15↵
05 SPSET 0,64,3,0,0,0↵
06 SPANIM 0,4,4:SPOFS 0,4,130↵
07 LOCATE 9,4?"WR "W:LOCATE 9,9↵
08 ?"READY ";:WAIT 60?"GO !"↵
09 BGMPLAY 0FOR X=4TO 226STEP 0↵
10 B=BUTTON()T=T+1/60LOCATE 9,5↵
11 ?"TIME";T↵
12 IF A==B THEN X=X+2:A=48-A↵
```

```
13 SPOFS 0,X,130:VSYNC 1NEXT
14 LOCATE 9,9?"GOAL IN !"
15 W=W+(T-W)*(W>T):I=0
16 BGMPLAY 4WAIT 60LINPUT A$NEXT
```

PRG:OCHA100B



24行だったプログラムがリスト短縮テクニックを用いることで同等の動作をするプログラムにも関わらず16行まで短縮されていることが分かると思います。

1画面プログラムを作る場合にはGOTO、GOSUBはできるだけ減らす必要があります。それはプチコンの場合にはラベルジャンプのみが可能でラベルのある行には他に何も記述できないためです。Aボタンを押した場合に最初の行から実行するという処理をする場合にもFOR～NEXTで無限ループを用意しておけば無駄な行を減らすことが可能になります。

サンプルAではIF文で記述していた新記録が出た場合の処理も論理式を用いてW=W+(T-W)*(W>T)とすることでT<Wの時のみW=Tを行うことができます。

このサンプルBでは前述のリスト短縮テクニックに含まないものとして**ボタン交互押しの短縮**や**LINPUTの空入力によるAボタン入力待ち**のリスト短縮テクニックが使用されています。

交互押しの判定は正しく入力されたかどうかを判定するBUTTON関数の値が16（Aボタン）と32（Bボタン）を交互に繰り返すため**A=48-Aが8と等しいかどうか**を判定すればわざわざフラグ管理を行いIF文を2つ使って判定する必要はなくIF文1つで判定可能になるため覚えておくと良いでしょう。

厳密にはこれでもリスト短縮の限界ではないのですが、**普通に作っただけのプログラムはリスト短縮テクニックによって大幅に短縮可能**ということさえ分かれば良いのでひとまずこれで置いておきます。1画面プログラムを作る場合にはサンプルA程度で完成と見なしサンプルBのようにどうすればもっとリスト短縮が可能になるのかを考えないことも多いので「普通に作ったプログラムはリスト短縮テクニックを駆使することでそれよりもずっと短縮可能である」ということを知っておくことは重要なことなのです。（当然ながら短縮しようと考えなければ短くなることはないため）

リスト短縮が出来るということは**従来ならば1画面プログラムでは諦めていたようなことが実現可能になる**という大きなメリットがあります。

まずは見た目の問題ですが、ここではトラック（走行しているコース）をスクロールさせようと思います。走るキャラが1キャラということでコースをスクロールさせるだけでは見た目を大幅に改善はできないため余っている画面の上部にスタジアムを描いてそれもスクロールさせることで2重スクロールにしようと思います。

無駄に小数第3位まで表示されているという問題は小数第3位を切り捨て表示することで簡単に実現できます。

ゲームバランスにおいては「連射が速い人で10秒を切るくらいにする」というのはボタンを1回押したときに移動量を変えれば良いだけなのですが、それでは先ほど書いたように連射速度に比例して良いタイムが出るようになってまうという問題は解決できません。連射は速い人と遅い人では2倍くらいの差があるため速い人が9秒台になるような設定だと遅い人は20秒近いタイムになってしまいます。これでも良いのですが、遅い人でも15秒前後、そこそこ速い人だと10秒台くらいのタイムが出せれば実際の100m走のタイムっぽくなりそうです。

すでに基本的なゲームシステムは出来ているため案は取捨選択が可能なので様々な案を用意しておくのが良いでしょう。すべての案を実装する必要はありませんが、自分の中での優先順位が高いものは確実に実装しましょう。リスト短縮を行ってもどうしても1画面に収まらない場合には優先順位が低いものから削っていくという決断が要求されます。(詳しくは後編のギタープログラムの制作を参照)

実際にどのように実装するかを考えていきます。
2重スクロールにおいてはBG画面を用いるのが最もポピュラーな方法です。プチコンではBG0、BG1の2つの画面をそれぞれスクロール可能であるためここではBG0をトラックに使用しBG1をスタジアムの表示に使用したいと思います。

プチコンのBG画面は縦横64キャラ(512ドット)で構成されているため初期設定の単純化ができるように256ドット単位でループするようなコースにします。(1ループ20mと考えて5ループでゴールイン)
スタジアムに必要なのはフェンスと観客です。観客はスプライトの男の子キャラの色違いを並べようと思いますがそのためにはスプライトキャラをBG画面用のキャラにコピーする必要があります。フェンスは適当なBGで埋めることにします。

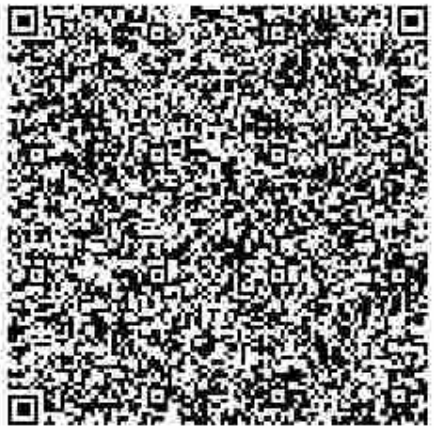
ゲームバランスの調整は速度のパラメータを用意することで実現します。
現状ではボタンを交互に押している場合はキャラが一定の距離進みそうでない場合はキャラが停止するため連射速度に反比例したタイムになっているのですが、速度のパラメータがあればボタンをちゃんと交互に押していない場合もある程度は進むため連射速度だけではなくちゃんと交互に押せるかどうかが要求されてきます。
この速度においても加速度が一定ならばその効果は薄いため速度が遅いときは加速度を大きくし、理論上の最高速度を設定してそれに近づくにつれて加速度を小さくしようと思います。こうすることで連射速度が名人を遥かに超えるくらい速い人でもある程度の値でタイムの向上は収束していくため極端にバランスが崩れることもありません。

それらを実装したのが次のプログラムです。(サンプルC)

```
01 W=99.99FOR I=0TO 1ACLS:T=0P=0↵
02 S=0A=16E=100FOR J=1TO 4↵
03 CHRREAD("SPU1",J+71),C$↵
04 CHRSET "BGU0",J,C$NEXT↵
05 S$="0001000200030004"↵
06 BGFILL 0,0,15,63,23,"9005"↵
07 BGFILL 0,7,15,7,23,"000F"↵
08 BGFILL 0,39,15,39,23,"000F"↵
09 BGFILL 1,0,0,61,5,S$↵
10 BGFILL 1,0,6,63,7,"700A"↵
11 GCLS 10SPSET 0,64,3,0,0,0↵
12 SPSCALE 0,200SPOFS 0,30,130↵
13 SPANIM 0,4,4LOCATE 2,9↵
14 ?"WR "(0OR W*E)/E↵
```

```
15 LOCATE 12,12?"READY ";↵
16 WAIT 60?"GO !":BGMPLAY 0↵
17 FOR X=0TO 1280B=BUTTON()X=X+S↵
18 IF A==B THEN P=(4-S)/5A=48-A↵
19 S=P+S*0.96Z=X%256/2T=T+1/60↵
20 Y=X%256BGOFS 0,Y,0BGOFS 1,Z,0↵
21 LOCATE 2,10?"TIME"(0OR T*E)/E↵
22 VSYNC 1X=X-1P=0NEXT↵
23 LOCATE 12,12?"GOAL IN !"↵
24 W=W+(T-W)*(W>T)I=0↵
25 BGMPLAY 4WAIT 60LINPUT A$NEXT↵
```

PRG：OCHA100C



このサンプルCでは見た目やゲーム性が大きく変わっていることが分かると思います。

2~4行でSPU1のキャラ番号72~75（スプライト番号82）の男の子のキャラをBGU0の1~4にコピーしています。それをBGFILLのスクリーンデータ文字列を使うことでスタジアム一杯に規則正しい配列で並べることが可能になります。（スクリーンデータを使用するとこういった一定パターンで配置する場合にFOR~NEXTを使う必要が無くなるため非常に便利）

正しいボタンを押した場合には速度Sに応じて一定の加速度Pを得ることができそしてそのSを座標Xに加算していきます。速度Sは常時一定割合で減速しているため自動的に速度に上限が発生します。1画面分（256ドット）進んだら20mでそれが5つ分で100mとなるためゴール時のXの値は1280以上となります。これはFOR~NEXTを使用することでIF文が不要になるだけではなくGOTOも不要になるためラベルを使用する無駄な行を減らすことができます。

ただし、残念ながら25行になってしまい1画面には収まりませんでした。もう少しは縮まる要素はありますがここから大幅なリスト短縮は難しいため何かを削らないと1画面に収まらないでしょう。

また、せっかく行った2重スクロールですが、現状ではそれは十分活かされているとは言えません。これは、リストを短縮する関係でスタジアムに配置しているキャラは一定のパターンで並べているだけであるためスクロールが分かりにくくなっているというのが理由です。これを改善するには、スタジアムにコース上の白線のような目印となるようなものが必要でしょう。

ただ、目印としてラインを引くというのもつまらないので現状では全く活用されてないフェンスに文字を入れることでスクロールを明確にしたいと思います。そこでフェンスにスポンサー（？）の「OCHAME」という文字を書くことにしましょう。これはフォントキャラ（BGF0）をBGUにコピーすればいいだけなのですが、ただでさえ1画面を大幅にオーバーしている上にこのような長くなる処理（SPUをBGUコピーする処理と同じくそれだけで4行分も必要になる）を加えたら別の要素を大幅に削らなければならないでしょう。

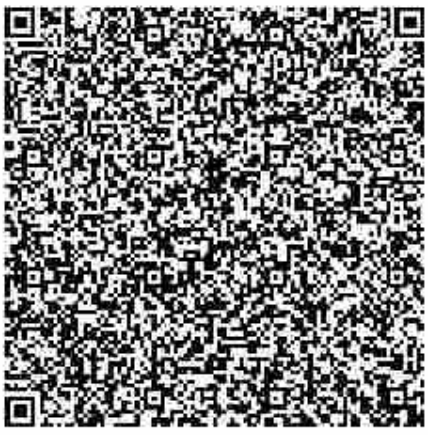
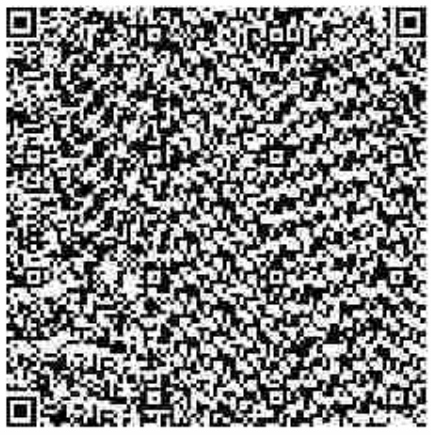
これが実現可能かどうかを確かめるためにはもう一度プログラムを一から見直す必要があります。
スクロールはBG画面を使う必要はあったのでしょうか？
プチコンはmk IIから加わったGCOPY命令を使うことでGRP画面をスクロールさせることも可能となっています。GRPならばGPUTCHRで表示するだけで良いのでスプライトキャラやフォントキャラをスクロール用に使用したい場合にさらなるリスト短縮ができます。（ただし、全画面をGCOPYするとそれだけで1.6フレームの時間がかかってしまうため60fpsを維持するにはスクロールさせる範囲をよく考える必要がある）

それを考慮して限界までリスト短縮をしたのがこの完成版の「100m走」です。

「プチコン100m走mk II」プログラムリスト

```
01 W=999N=200ACLS:GPAGE 1GCLS 7
02 U=72GLINE U,0,U,U,15S$="SPU1
03 H=256R=109P=0GFILL 0,96,H,H,4
04 FOR I=0TO H*6J=I%60=I%3*8V=50
05 Q=0-H+I%H*8GPSET RND(H),I%U,8
06 M=28912%(J+13)+!J*14G$="GOAL!
07 GPUTCHR Q+0,U+0,S$,U+I%4,0,1
08 GPUTCHR J*8,99,"BGF",M+65,4,1
09 NEXT:FOR E=0TO 1CLS:T=0A=32
10 SPSET 0,64,3,0,0,0SPSCALE 0,N
11 SPOFS 0,48,140SPANIM 0,4,4S=0
12 GPAGE 0GCLS 10?" "*H"WR "W/V
13 FOR X=0TO H*5B=BUTTON()X=X+S
14 IF!(A-B)THEN P=(4-S)/5A=48-A
15 S=P+S*0.97T=T+1Z=X%512/2Y=X%H
16 GCOPY 1,0,0,Y-1,55,H-Y,136,1
17 GCOPY 1,Y,0,255,55,0,136,1P=0
18 GCOPY 1,0,U,Z-1,R,H-Z,0,1
19 GCOPY 1,Z,U,255,R,0,0,1WAIT 1
20 FOR I=T TO 1?" "*75"READY ";
21 WAIT 99BGMPLAY 0E=0?"GO!
22 NEXT:X=X-1LOCATE 1,9?"  "T/V,
23 NEXT:W=W+(T-W)*(W>T)?" "*99G$
24 BGMPLAY 4WAIT 99LINPUT A$NEXT
```

1/2 PRG:OCHAL100 2/2

BG画面を使わずGRP画面のみでスクロールさせているのですが、60fpsでなめらかに動作しているし、見た目も大きく改善されたのが分かると思います。

1画面で最初からこの完成形を作るというのは難しいですが、リスト短縮テクニックを覚えたり、1行の文字数をうまく29文字になるように調整して空きを作りそこに実装したい処理を随時入れていくという方法で徐々にいろいろな要素を入れていき完成度を高めることが可能になります。

「100m走」では特にリスト短縮をせずに普通に作っても1画面に収まるようなシンプルなプログラムを元に徐々に肉付けしていき完成度を高めましたが、次はその逆をやってみます。つまり、普通に作ったら1画面に収まらないようなプログラムをリスト短縮および余分なものを削って1画面にすることで完成度を高めるというものです。

そのためにギタープログラム「プチコンギター」を「1画面」という制約は無しで作ってみました。1画面化は後から行いますが、これはプチコン用としては他に例のない1弦ずつ独立した音が鳴らすことによって実際のギター演奏のようなコード演奏を可能にするプログラムです。

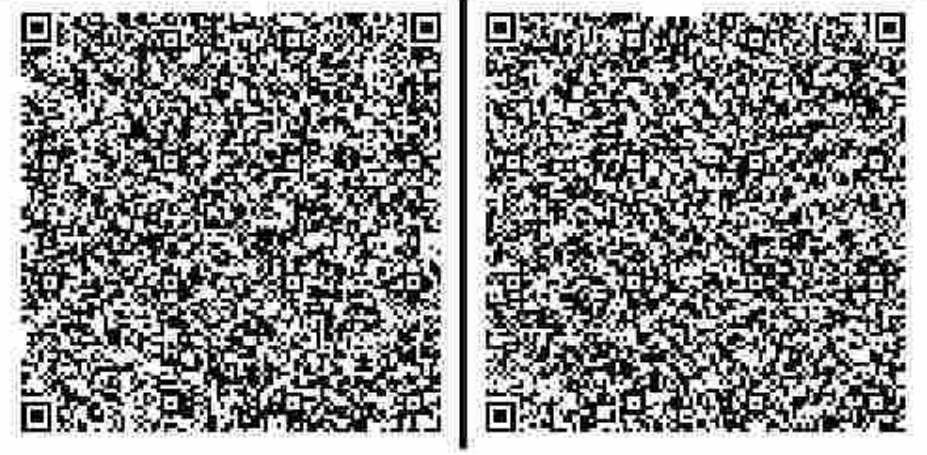
「プチコンギター」　プログラムリスト

```
01 ACLS:CLEAR↵
02 DIM C(1152),C$(16)↵
03 ↵
04 @CODE↵
05 READ D$↵
06 IF D$=="END"THEN @INIT↵
07 N$(A)=D$↵
08 FOR I=0TO 11↵
09   READ D$↵
10   FOR J=0TO 5↵
11     C(A*72+I*6+J)=VAL(MID$(D$,J,1))↵
12   NEXT↵
13 NEXT↵
14 A=A+1:GOTO @CODE↵
15 ↵
16 @INIT↵
17 BGMSET 128,"T1@$0N$1"↵
18 GPAGE 1:PNLTYPE "OFF"↵
19 GFILL 0,0,255,64,255↵
20 FOR I=0TO 5↵
21   GFILL I*30+60,0,I*29.5+63,191,7↵
22   A(I)=I*5+44-(I>3)↵
23   G(I)=27↵
24 NEXT↵
25 GPAGE 0↵
26 ↵
```

```
20 @MAIN↵
21 GCLS :X=TCHX:Y=TCHY↵
22 U=T:T=TCHST↵
30 W=V:V=0OR(X-60)/30↵
31 S=V+(V<W)↵
32 B=BUTTON()↵
33 C=(B%128-B%16)/16↵
34 E=!!(128AND B)↵
35 D=C*2-(C>2)-(C>5)+E-2↵
36 IF 512AND B THEN P=C*(A>C)↵
37 C$=CHR$(64+C)+"#"*E+N$(P)↵
38 FOR I=0TO LEN(C$)-1↵
39  GPUTCHR I*32+70,90,"BGF",ASC(MID$(C$,I,1))%1
5,4↵
40 NEXT↵
41 IF T*U*(W-V)*(S>=0)*(S<=5) THEN GOSUB @PLAY
↵
42 WAIT 1↵
43 GOTO @MAIN↵
44 ↵
45 @PLAY↵
46 GOSUB @CHANGE↵
47 N=(Y-65)*(Y>65)↵
48 Q=C(!!C*(D*6+P*72)+S)*!!C↵
49 BGMPLAY S,128↵
50 BGMSETV S,0,G(S)↵
51 BGMSETV S,1,A(S)+Q-(G(S)>31)*12↵
52 BGMVOL S,N↵
53 LOCATE 0,S:?S"=",G(S)↵
54 RETURN↵
55 ↵
56 @CHANGE↵
57 K=INSTR("8A264519",HEX$(B%16))↵
58 IF K+1THEN G(S)=K+24+(B AND 256)/32↵
59 RETURN↵
60 ↵
61 ' ﾒｼﾞｬｰ↵
62 DATA "  "↵
63 DATA 002220,113331,224442↵
64 DATA 032010,446664,000232↵
65 DATA 668886,022100,133211↵
66 DATA 244322,320003,466544↵
67 ↵
68 ' ﾏｲﾅｰ↵
69 DATA "m"↵
70 DATA 002210,113321,224432↵
71 DATA 335543,446654,000231↵
72 DATA 668876,022000,133111↵
73 DATA 244222,355333,466444↵
74 ↵
```

```
75 '  セブン↵
76 DATA  "7"↵
77 DATA  002020,113131,021201↵
78 DATA  032310,446464,000212↵
79 DATA  668686,020100,131211↵
80 DATA  242322,320001,454544↵
81 ↵
82 '  マイナーセブン↵
83 DATA  "m7"↵
84 DATA  002010,113121,224232↵
85 DATA  335343,446454,000211↵
86 DATA  668676,020000,131111↵
87 DATA  242222,353333,454444↵
88 ↵
89 DATA  "END"↵
```

1/2 PRG: OCHAGTAR 2/2



【 プチコンギター 操作説明 】

このプチコンギターは実際のギターのように太い弦が上になるような縦持ちで演奏します。小指が手前になるように左手の手のひらを上に向けてその上に普通に本体を置き親指でRボタンが操作できる位置にずらします。そうすれば自然に縦持ちになります。左手の人差し指、中指、薬指をうまく使ってABXYボタンを操作します。右手の親指でギターを演奏するように弦を弾いてください。1本1本独立制御しているため上から弾いても下から弾いても特定の1本のみを弾いてもOKです。

弦の弾く位置で音量が変わります。画面の下の方が大きく、上の方に行くに従って小さくなります。上手く弾けばビブラートっぽい音も出せます。画面の上1/3は消音となります。

上画面の数字は各弦の音色番号を示しています。(最も太い弦は正しくは「第6弦」ですがこのプログラム内では「0番弦」になっていて太い順に0～5の数字で示されている)

音色はプチコンmkIIで使えるギター系のものを8種類ほど十字ボタンの各方向に割り当てているため十字ボタンを押しながら弦を弾けばその弦の音色が変わります。難しく考えず「ジャーン」と6本全部弾けば6本全部の音色が変わります。デフォルトの27番の音色は十字ボタンの左下に割り当てられています。使用できる音色はギター系である24～31の8種ですが、Lボタンを同時に押すことでベース系の32～39の音色も使用可能です。ベース系の音色は1オクターブ下げていますが、高い2本の弦は実際の4弦ベースでは出ない音域であり、一般的な6弦ベースとも異なります。あくまでギタープログラムであり、ベースではないためバリトンギターのようなものと考えてください。

音階の選択	
A	Aボタン
B	Bボタン
C	A+Bボタン
D	Xボタン
E	A+Xボタン
F	B+Xボタン
G	A+B+Xボタン
#	Yボタン

コードの種別の選択	
メジャー	何も押してない状態でRボタンを押す
マイナー	Aボタンを押しながらRボタンを押す
セブン	Bボタンを押しながらRボタンを押す
マイナーセブン	A+Bボタンを押しながらRボタンを押す

※コードは自分でデータを追加することで16種まで追加が可能
データ形式は太い弦から順に押すフレット番号を6桁の数字で記述していく（0は開放弦）
A、A#(B♭)、B、C、C#(D♭)、D、D#(E♭)、E、F、F#(G♭)、G、G#(A♭)の12音階分羅列する。
コードデータの先頭には表示用の略号のデータを記述すればコード1種分のデータになる。（略号＋6桁の数字12個分が1セット）
※画面表示における「@」はすべての弦が開放弦になっていることを示す

このプログラムの簡単な仕組みを書いておきます。
まず、1本1本の弦の独立制御ですが、プチコンmkIIでは8トラックの同時演奏が可能であることを利用して1つの弦に1トラックを割り当てています。MMLは曲番号128に**T1@$0N$1**というものをセットしています。テンポT1でできるだけ長い音を鳴らし、MML内変数**$0**で音色番号、MML内変数**$1**で音程をBGMSETV命令を用いてプログラム内で書き換えることでMMLの制御がプログラムで可能になっています。（音程NをMML内変数で制御する場合には半音4つ分ずれることに注意）

タッチした座標を元にどの弦を鳴らすのかという判定は単純にGSPOITを使って「弦に触れているかどうか」で判定したら取りこぼしが多くなって実用になりません。座標を元に見た目よりも広く当たり判定を取った場合には触れてない弦で音が鳴る場合もあるためそれも不適切です。ゲームだと当たり判定は非常に重要ですが、このような演奏ツールでもゲームと同じくらい重要なのです。
そこでこのプログラムでは弦を境目にした7つのブロックに分けてタッチした状態で隣のブロックに移動した時にその境目にある弦の音を鳴らしています。1フレームで2ブロック以上進んだ場合にはこれでも正常な判定はできませんが通常はあり得ない速度（6本の弦を弾くのに0.1秒未満の時間になるような速度）であるため問題はありません。

音色変更処理は8種の音色を8つのIF文で判断すると長くなってしまうのでこのプログラムではINSTR関数を使いIF文1つで済むようにしています。十字ボタンの各方向を押したときのBUTTON関数の値を16進数化してその中の何番目かで変更する音色が変わるようになっています。十字ボタンを押していない場合はINSTR関数の値は-1になるためその場合は音色変更はしないようにしています。

このプログラムは論理式や論理否定を多用しているためラベル以外はリストにはそれほど無駄はないのですが、それでも89行というサイズとなっています。これを1画面、24行以内に短縮してみます。
全く同一内容で1画面に収まるならばそれがベストですが、現実的にはリスト短縮には限界があり、せいぜい3分の2程度にしかなりません。すでに無駄な部分がかなり減っている現状では3分の2も厳しいでしょう。

完成度を高めるのと効率よく1画面化を行うために各処理に優先順位を付けていきます。
弦の表示処理は必須ですが、リスト短縮のため太さを一定にするというのはどの弦の音が高いか（低いか）が見ただけでは分かりにくくなるためできるだけ避けたいところです。コードや音階の表示はGPUTCHRを用いずにもっとシンプルなものでも問題ないですが、表示無しだとさすがに困るのでリスト短縮との兼ね合いでどの程度のシンプルさにするのかは変わるでしょう。音色変更処理はリストを書き換えれば簡単にできるため優先度はかなり低めです。音量変更や消音機能は必須ではないですが、あると演奏の幅が広がるため優先度は中程度（エレキ系は音の減衰がほとんど無く鳴り続けるのでそれらを使う場合は消音機能に関しては必須）でしょう。コードデータは当然必須でしょう。4種×12音階分のコードは必要最低限であるためこれ以上は削りたくないところです。
上記はかなり大ざっぱですが、実際はもっと細かく見ていき優先度を設定することで必須部分が1画面に収まりそうならば1画面化はすぐにできます。ただし、このプログラムでは厄介なことに必須であるコードのデータだけでほぼ1画面埋まっています。注釈を削っても20行となるため1画面プログラムを作るならば残り4行しか空きがないため不可能です。

そこでコードデータを圧縮する必要があります。コードデータを圧縮することで1画面化も可能になるだけではなくどれだけの機能を実装できるかも決まってきます。そのためできるだけデータは圧縮したいところですが展開ルーチンもコンパクトサイズにできなければ意味がないため圧縮アルゴリズムはシンプルで高圧縮なものが必要になります。
データを見る限りは0～8の9種の数字が使われているためそれを4bitのデータと見なせば1文字（8bit）では2つ分のデータを入れることができます。つまり、1/2のサイズに圧縮できるということです。
また、「8」という数字はほとんど使われてないため0～7を3bitで表し、「8」のデータのみ例外処理を行えばさらに圧縮率が高まりそうな気もします。3bitならば6桁で18bitであるため8bit単位にするためには4つのデータをまとめて圧縮展開しなくてはなりません。そうなるとFOR～NEXTが1組余分に必要になるし、「8」の例外処理を考えるといくら圧縮率が高くても展開処理が長くなってしまうためトータルでは微妙なところです。

差分圧縮ならばそれは改善できます。例えば**D#m**のデータは**668876**ですが、6桁全部から5を引いて**5,113321**というデータにします。コードのデータは開放弦を除いて数字の上下差は3以下に収まっている（これは実際に指で押さえる場合に遠いフレットを押さえることができないのが理由）ため差分を取れば各弦のデータは2bitで収まります。また、Gのデータは**320003**ですが、これを**2,100001**としてしまうと復元不能になるため**1,210001**とする必要があります。（開放弦以外は0にならないようにする）
いくつ差分を取ったかというデータに4bit割り当てても6桁で合計16bit（＝2文字分）に収まり、最初のデータと比べて1/3のサイズに圧縮できるだけではなく展開ルーチンもシンプルに収まります。
データの8bit化の際には改行コードなどのリストで表記できないされない文字にならないような対策も必要ですが、今回用意したコードでは8bit化したあとのデータに1プラスするだけで改善できるため合計では20行あったコードデータが4行に圧縮でき展開ルーチンも3行に収まりました。

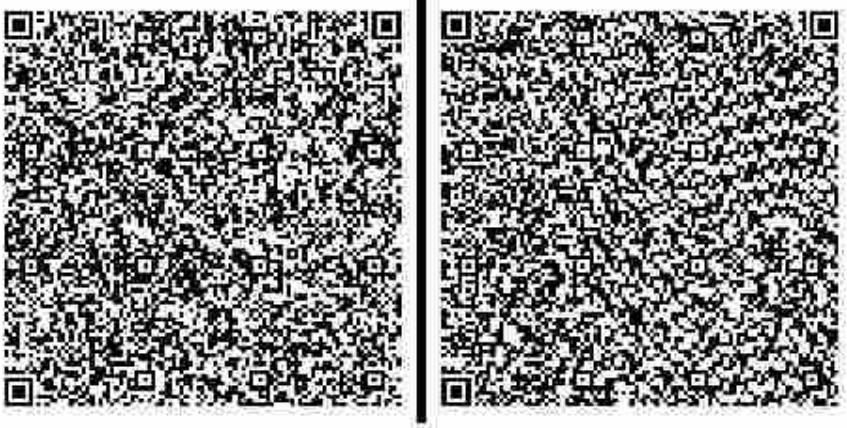
ということで出来たのがこの1画面プログラム「PETIT GUITAR」です。

「PETIT GUITAR」プログラムリスト

```
CLEAR:ACLS:GPAGE 1PNLTYPE"OFF⏎
W$="🅑⋯Q◇R◇ﾁ✚T◇🅑ⅇV◇" 回ㅿ¥ﾒ¥ｂ" ﾔ¥⏎
X$="🅐ﾐQpRpSpTp🅐ｭVpB🅑ㅿXﾒXﾓXﾔX⏎
Y$="🅑🅡Q×" Jﾁ↓T×🅐┓V×" ♣ㅿZﾒZｱAﾔZ⏎
Z$="🅐✚QhRhShTh🅐YVhB🅐ㅿVﾒVﾓVﾔV⏎
DIM C(999)Z=256A$=W$+X$+Y$+Z$⏎
FOR I=0TO 47B$=MID$(A$,I*2,2)⏎
X=ASC(RIGHT$(B$,1))*Z+ASC(B$)⏎
X=X-257Y=X%16W=16FOR J=2TO 7⏎
X=X-Y*(J<3)V=X%(W*4)/W:U=!!V⏎
```

```
11 X=X-V*W:C(J+I*6)=V+Y*U:W=W*4
12 GFILL J*30,0,J*29.5+4,Z,7H=64
13 A(J)=J*5+118-(J>5)NEXT:L=I+40
14 BGMSET I+128,"T1@27N"+STR$(L)
15 NEXT:FOR X=0TO 9B=BUTTON()CLS
16 X=0OR TCHX/30R=Y>X?CHR$(C+H);
17 E=!!(128AND B)C=(112AND B)/16
18 S=X+R:D=0OR(C*5-4)/3+E?"#"*E
19 M=TCHY:IF 512AND B THEN P=C
20 O=(S>1)*(S<8)N=(M-H)*(M>H)?P
21 Q=C(!!C*(D*6+P*72)+S*O)*!!C
22 F=!!(X-Y)*T*U*O:G=Q+A(S*F)U=T
23 T=TCHST:IF F THEN BGMPLAY S,G
24 BGMVOL S*F,N:Y=X:WAIT 1NEXT
```

1/2 PRG:OCHAPGTR 2/2



このプログラムでは音色変更機能を省き画面表示はややシンプルになりましたが、それ以外は89行あった元のプログラムと同等のものを24行で実現しています。（コードについては画面表示で「0」がメジャー、「1」がマイナー、「2」がセブン、「3」がマイナーセブンを示す）

コードデータの圧縮以外のリスト短縮としては、次のようなことも行っています。

元のプログラムでは1つのMMLにおいてMML内変数をリアルタイムに書き換えることで様々な音程の音を出していましたがこのPETIT GUITARではプログラム内で使う音程をあらかじめすべて別のMMLとしてBGMSETで定義するという力業により**BGMSETV**が不要になる分だけリスト短縮が可能になっています。この場合はFOR～NEXTが1組余分に必要であるため単純にリスト短縮に繋がるとは言えないのですが後述のようにFOR～NEXTを極限まで減らしているため全く問題はなくなっています。

A～Gの音階（半音無し）を示す0～7という値を使い半音ありの12音階に変換する場合はごく単純に考えれば**D=C*2-(C>2)-(C>5)-E-2**という式で表すことができます。しかし、これは**D=0OR(C*5-4)/3+E**と短縮することが可能です。

1画面プログラムにおいては1組のFOR～NEXTでさえ長くなるため100m走ではそれを可能な限り減らすことでリスト短縮していますが、このPETIT GUITARも同様にFOR～NEXTは極限まで減らしています。BGMSETとコードデータの展開処理と弦の描画は同じループを兼用しているし、メインルーチンもループ内で常に変化している変数Xの値をFOR～NEXTのカウンタ用の変数に使うことで無限ループを最小限の文字数で実現しています。（100m走では終了値にゴール時の座標をセットしていたけどこのPETIT GUITARではXが絶対に達しない値を終了値に設定している）

あとはよく使う定数は固定変数に入れたり、依存性のない処理は並べ替えて行末の無駄を無くす1画面プログラムでは定番的なテクニックを使用することで何とか1画面に収めることができました。

1画面プログラムの制作には「1画面に余裕で収まるシンプルなものを考えてそれに肉付けをする」という方法と「短め（概ね100行未満）のプログラムを作って優先順位を付けた取捨選択を行い1画面に収める」という方法の2つのアプローチがあります。

どちらが正解でどちらが間違いというわけではないのですが、より完成度を高めるならば1画面に収まらないサイズのものを1画面に収めるようにするという方法がベターです。それは削る場合には優先度の低いものを削るのことになるのに対して追加する場合にはそれが優先度が高いものであるとは限らないためです。それといかにして削らないように済む方法はないかを考えることでよりリスト短縮ができる場合もあるからです。

最後にまとめると1画面プログラムの完成度を高めるには妥協をしない（優先度の高い処理は無理矢理でも実装する）ということが必要になります。1画面プログラムはサイズが小さいため作ること自体は簡単にできますが、作り込もうとすると奥が深いことが分かってもらえたのではないでしょうか。

現実の100m走は「距離が100mくらい」ではなく厳密に100m同士で争われているためその限界に挑戦することには意味があります。それと同じく1画面プログラムもレギュレーションとして確立しているためその限界に挑戦することが可能になります。限界に挑戦する楽しさを味わって見てください。

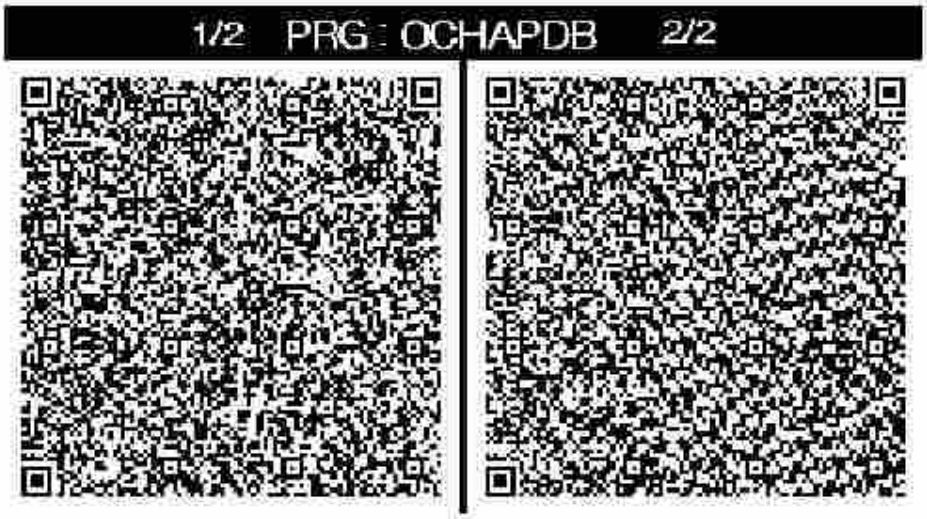
おまけ

ギターだけでは物足りないのでドラムとベースの両方ができるプログラムを1画面で作ってみました。

[START]ボタンでベース、[START]＋何か別のボタンでドラムとなります。ベースは🅐🅑🅧🅡ボタンで押さえるフレット数を選択して（順に1、2、4、8となっている）弦を引いて演奏します。ドラムは各ボタンにバンブラのフリー演奏のロックドラムと同じ楽器が割り当てられています。

「PETIT DRAM & BASS」プログラムリスト

```
01 CLEAR:ACLS:GPAGE 1PNLTYPE"OFF↵
02 FOR X=0TO 9L=!(1023AND B-1)↵
03 M$=" A#A B C#C D#D E F#F G#G↵
04 Z=64GCLS:FOR I=28TO 59J=I%10↵
05 D=ASC(MID$("*-/2&$6%.1",J,1))↵
06 BGMSET 160+J,"@128N"+STR$(D)↵
07 BGMSET I+100,"L1@34N"+STR$(I)↵
08 NEXT:FOR B=0TO 1023B=BUTTON()↵
09 X=0OR TCHX/40-1R=Y>X:T=TCHST↵
10 C=(112AND B)/16+(512AND B)/Z↵
11 S=X+R:M=TCHY:O=(S>0)*(S<5)CLS↵
12 N=(M-Z)*(M>Z)F=!!(X-Y)*T*U*O↵
13 G=C+S*5+123FOR J=2+!L*6TO 5↵
14 N$=MID$(M$,(C+9+J*5)%12*2,2)↵
15 PNLSTR J*5,0,N$LOCATE 0,0?P$;↵
16 E=BGMCHK(J-1)*RND(99)?"T BASS↵
17 GFILL J*40,-1,J*39+6,191,E+7↵
18 U=T:Y=X:IF F THEN BGMPLAY S,G↵
19 BGMVOL S*F,N:BGMSTOP!N*S*F?C↵
20 NEXT:FOR J=L*10TO 9H=POW(2,J)↵
21 H=H AND BTRIG()D$=P$+"T DRUM↵
22 PNLSTR 11,11,D$S=J-2+(J<3)+!J↵
23 IF H THEN BGMPLAY S,160+J↵
24 NEXT:P$="PETI"WAIT 1NEXT:NEXT↵
```



それでは皆さん、1画面プログラムに

Let's charange!

2013年12月31日発行
制作・発行　おちゃめくらぶ
発行人　　御茶目菜子（おちゃめ）

おちゃめくらぶ　Webサイト
http://ochameclub.web.fc2.com/

おちゃめ　twitter
https://twitter.com/ochame_nako

おちゃめくらぶ　2013