

マイコン部 2024

講習会資料

第3回

目次

第 2 章 基本の命令 ver2

2-5 キー・マウスの読み取り

2-5.1 getkey

2-5.2 stick

第 3 章 プログラム作成

3-1 画面の初期化

3-1.1 boxf

3-1.2 cls

3-1.3 redraw

3-2 時計の作成

3-2.1 プログラムの作成

3-2.2 時計の装飾

3-3 ボールを動かす

3-3.1 ボールの移動

3-3.2 ボールが跳ね返るようにする

3-3.3 操作できるボールの追加

3-3.4 ボールの数を増やす

3-4 じゃんけんゲーム

3-4.1 素材の用意

3-4.2 プログラムの作成

3-4.2 勝敗の表示

3-5 プログラム紹介

3-5.1 合成音声プログラム

3-5.2 ブロック崩し

目次

3-5.3 タイピングゲーム

3-5.4 RPG 風ゲーム

第2章 基本の命令 ver2

2-5 キー・マウスの読み取り

ゲームを作成する際、キーボードやマウスの入力を読み取る必要があります。ここでは、簡単なキーボード入力とマウス入力の検出方法を紹介します。

2-5.1 getkey

キーボード、マウス入力を読み取るために、最もよく使われる方法が `getkey` 命令を使用することです。この命令は、キーボードの入力、マウスのクリックを取得することができます。

```
getkey p1,p2 //キー入力チェック
```

p1=変数 : 読み込むための変数

p2=1～(1): キーコード

設定したキーコードのキーボードのキーが押されているまたはマウスがクリックされている場合は、変数に 1、されていない場合は変数に 0 が入ります。設定できるキーコードは以下の通りです。

第2章 基本の命令 vol2

キーコード : 実際のキー			
1	: マウスの左ボタン	32	: スペースキー
2	: マウスの右ボタン	33	: [PAGEUP]
3	: キャンセル	34	: [PAGEDOWN]
4	: ホイールクリック	35	: [END]
8	: [BACKSPACE]	36	: [HOME]
9	: [TAB]	37	: カーソル[←]
13	: [ENTER]	38	: カーソル[↑]
16	: [SHIFT]	39	: カーソル[→]
17	: [CTRL]	40	: カーソル[↓]
18	: [ALT]	48~57	: [0]~[9]
20	: [CAPSLOCK]	65~90	: [A]~[Z]
27	: [ESC]	96~105	: [0]~[9] (テンキー)
		112~121	: [F1]~[F10]

2-5.2 stick

HSP には、キー/クリック検出のために、ほかの命令も用意されています。stick 命令は getkey 命令と同じく、キーやクリックの検出を行います。getkey と違うのは、一つの命令で同時検出ができる。(getkey 命令でも変数を複数用意することで同時検出は可能) getkey 命令は入力をしている間はずっと変数に数値が代入されるが、stick 命令はキーやクリックを押した瞬間のみ数値が代入される、検出できるキー/クリックの種類が getkey に比べて少ないなどの点です。

`stick p1,p2,p3` //キー入力情報取得

p1=変数 : 読み込むための変数

p2=0~(0) : 非トリガータイプキー指定

p3=0~1(1) : ウィンドウアクティブチェック ON/OFF

stick 命令で検出出来るボタン情報は以下の通りです。

1 : カーソルキー左(←)

2 : カーソルキー上(↑)

4 : カーソルキー右(→)

8 : カーソルキー下(↓)

16 : スペースキー

32 : Enter キー

64 : Ctrl キー

128 : ESC キー

256 : マウスの左ボタン

512 : マウスの右ボタン

1024 : TAB キー

何もボタンが押されていない場合には0が代入されます。また、もし複数のボタンが同時に押されていた場合には、それらの数値がすべて加算されて変数に代入されます。

第3章 プログラム作成

ここまでは、HSP 言語における基本的な命令やオブジェクトについて解説してきました。ここからは実際にこれまで学んだことを利用してプログラムを作成していきます。

3-1 画面の初期化

HSP でゲームなどの動きのあるプログラムを作成する際、画面の初期化は必須になってきます。画面の初期化を行わないと、ウィンドウに1つ前に表示されたコンテンツが残り続け、どんどん重なって行ってしまいます。そのような事態を防ぐために、HSP では主に2つの方法を使ってウィンドウの初期化を行います。

3-1.1 boxf

1つ目は、boxf 命令を使用する方法です。boxf 命令はもともと、ウィンドウの指定範囲を指定された色で塗りつぶすという意味を持った命令になります。これを利用して、ウィンドウ全体を好きな色で塗りつぶしてやれば、まっさらな画面に初期化することができますというわけです。

```
boxf p1,p2,p3,p4 //矩形を塗りつぶす
```

p1=0～(0) : 矩形の左上 X 座標

p2=0～(0) : 矩形の左上 Y 座標

p3=0～ : 矩形の右下 X 座標

p4=0～ : 矩形の右下 Y 座標

boxf 命令のすべてのパラメータを省略すると、画面全体を指定した色で塗りつぶすことができます。

この命令は基本的に color 命令と組み合わせて使用します。先に color 命令で塗りつぶしたい色を指定し、そのあとに boxf 命令を実行すれば、color 命令で指定した色に塗りつぶしてくれます。例えば、画面全体を真っ白に初期化したい場合は、color 255,255,255 を先に指定しておき、次の行で boxf 命令を実行すればよいわけです。

第3章 プログラム作成

注意点として、boxf 命令ではウィンドウ上にあるオブジェクトを消去することができません。オブジェクトを消去したい場合は clrobj 命令か後述する cls 命令を利用してください。

3-1.2 cls

2 つ目が cls(clear screen)命令を使用する方法です。この方法では、オブジェクトを含むすべてのウィンドウ内の情報をクリアすることができます。

```
cls p1 //画面クリア
```

p1=0~4(0) : クリアする時の色

p1 でクリアする 5 種類の色を指定することができます。

色の指定値 : (0=白 / 1=明るい灰色 / 2=灰色 / 3=暗い灰色 / 4=黒)

また cls 命令は color+boxf の方法よりも処理速度が遅いため、アクションゲームなどの動きが激しいコンテンツの使用には向かないことに注意する必要があります。

3-1.3 redraw

redraw 命令は boxf 命令や cls 命令のように画面を初期化するものではありませんが、動きのあるプログラムを作成する際、同じく重要になる命令です。たとえば、繰り返し処理のあるプログラムを実行している際に画面がちらつく場合があります。そのような現象を予防するのが redraw 命令です。

```
redraw p1,p2,p3,p4,p5 //再描画の設定
```

p1=0~3(1) : 描画モードの設定

p2=0~(0) : 再描画する左上 X 座標

p3=0~(0) : 再描画する左上 Y 座標

p4=0~(0) : 再描画する大きさ X (ドット単位)

p5=0~(0) : 再描画する大きさ Y (ドット単位)

redraw 命令は、ちょうど前の画面をパソコンに表示している間に裏に次のこまを描画して、描画が完了した後に裏返してそれを見せ、また裏に次の描画をするというような命令です。描画モード 0 で仮想画面 (裏画面) モードになり、描画モード 1 で画面反映

第3章 プログラム作成

(表画面) モードになります。したがって、redraw 0 を実行したのちにプログラムの処理を行い、redraw1 にして処理を画面に反映させ、もう一度 redraw0 に戻るといったようなプログラムを作成することができます。

また注意点として、cls 命令を使用している場合は redraw を利用しても画面がちらつく場合があります。その場合は boxf 命令を利用してください。

3-2 時計の作成

ここからはいよいよプログラムの作成に入ります。まずは簡単な時計を作ってみましょう。

3-2.1 プログラムの作成

まずは、時計本体のプログラムを作成しましょう。HSP で時刻を取得するには gettimeofday 命令を利用します。

```
val = gettimeofday(p1) //時間・日付を取得する
```

p1=0~7(0): 取得するタイプ

gettimeofday 命令は設定するパラメータの値に応じて日付や時刻を返してくれます。設定できるパラメータは以下の通りです。

0: 年(Year)	4: 時(Hour)
1: 月(Month)	5: 分(Minute)
2: 曜日(DayOfWeek)	6: 秒(Second)
3: 日(Day)	7: ミリ秒(Milliseconds)

これを利用して、まずは日時が表示されるプログラムを作成してみましょう。

まずは時刻を表示するプログラムを記述します。

第3章 プログラム作成

```
10
110
120
130
140
150
160
170
4 DayOfWeek = 0
5 Day = 0
6 Hour = 0
7 Minute = 0
8 Second = 0
9 Youbi = ""
10
11 //gettimeの処理
12 Year = gettime(0)
13 Month = gettime(1)
14 DayOfWeek = gettime(2)
15 Day = gettime(3)
16 Hour = gettime(4)
17 Minute = gettime(5)
18 Second = gettime(6)
19
20 //数字になっている曜日を文字に変更
21 if (DayOfWeek = 0){ //0番なら
22     Youbi = "Sun" //Sunday
23 }else : if (DayOfWeek = 1){
24     Youbi = "Mon"
25 }else : if (DayOfWeek = 2){
26     Youbi = "Tue"
27 }else : if (DayOfWeek = 3){
28     Youbi = "Wed"
29 }else : if (DayOfWeek = 4){
30     Youbi = "Thu"
31 }else : if (DayOfWeek = 5){
32     Youbi = "Fri"
33 }else : if (DayOfWeek = 6){
34     Youbi = "Sat"
35 }
36
37 //文字の表示
38 mes str(Year) + "/" + str(Month) + "/" + str(Day) + "(" + Youbi + ")"
39 mes str(Hour) + ":" + str(Minute) + ":" + str(Second)
40 await 1000
```

Hot Soup Processor ver.3.4
2022/5/23 (Mon)
7:4:5

うまく表示することができました。しかし、これでは時刻が1度しか表示されません。そこで、repeat-loop 命令を利用して何回も文字を表示させるようにします。

第3章 プログラム作成

```
4 DayOfWeek = 0
5 Day = 0
6 Hour = 0
7 Minute = 0
8 Second = 0
9 Youbi = ""
10 repeat
11     //gettimeの処理
12     Year = gettime(0)
13     Month = gettime(1)
14     DayOfWeek = gettime(2)
15     Day = gettime(3)
16     Hour = gettime(4)
17     Minute = gettime(5)
18     Second = gettime(6)
19
20     //数字になっている曜日を文字に変更
21     if (DayOfWeek = 0){ //0番なら
22         Youbi = "Sun" //Sunday
23     }else : if (DayOfWeek = 1){
24         Youbi = "Mon"
25     }else : if (DayOfWeek = 2){
26         Youbi = "Tue"
27     }else : if (DayOfWeek = 3){
28         Youbi = "Wed"
29     }else : if (DayOfWeek = 4){
30         Youbi = "Thu"
31     }else : if (DayOfWeek = 5){
32         Youbi = "Fri"
33     }else : if (DayOfWeek = 6){
34         Youbi = "Sat"
35     }
36
37     //文字の表示
38     mes str(Year) + "/" + str(Month) + "/" + str(Day) + "(" + Youbi + ")"
39     mes str(Hour) + ":" + str(Minute) + ":" + str(Second)
40     await 1000
41 loop
```

Hot Soup Processor ver.3.4

```
2022/5/23(Mon)
7:5:55
2022/5/23(Mon)
7:5:56
2022/5/23(Mon)
7:5:58
2022/5/23(Mon)
7:5:59
2022/5/23(Mon)
7:6:0
2022/5/23(Mon)
7:6:1
2022/5/23(Mon)
7:6:2
2022/5/23(Mon)
7:6:3
2022/5/23(Mon)
7:6:4
2022/5/23(Mon)
7:6:5
```

時間は表示されましたが、前の時間の表示が消えずどんどん下に重なって行ってしまいます。そこで、先ほど学んだ表示の初期化を利用してみましょう。今回は cls 命令を使います。

第3章 プログラム作成

```
4 DayOfWeek = 0
5 Day = 0
6 Hour = 0
7 Minute = 0
8 Second = 0
9 Youbi = ""
10 repeat
11     //gettimeの処理
12     Year = gettime(0)
13     Month = gettime(1)
14     DayOfWeek = gettime(2)
15     Day = gettime(3)
16     Hour = gettime(4)
17     Minute = gettime(5)
18     Second = gettime(6)
19
20     //数字になっている曜日を文字に変更
21     if (DayOfWeek = 0){ //0番なら
22         Youbi = "Sun" //Sunday
23     }else : if (DayOfWeek = 1){
24         Youbi = "Mon"
25     }else : if (DayOfWeek = 2){
26         Youbi = "Tue"
27     }else : if (DayOfWeek = 3){
28         Youbi = "Wed"
29     }else : if (DayOfWeek = 4){
30         Youbi = "Thu"
31     }else : if (DayOfWeek = 5){
32         Youbi = "Fri"
33     }else : if (DayOfWeek = 6){
34         Youbi = "Sat"
35     }
36
37     //文字の表示
38     mes str(Year) + "/" + str(Month) + "/" + str(Day) + "(" + Youbi + ")"
39     mes str(Hour) + ":" + str(Minute) + ":" + str(Second)
40     await 1000
41     //表示の初期化
42     cls
43 loop
```

Hot Soup Processor ver.3.4

2022/5/23(Mon)


7:7:52

これで、きれいに表示することができました。

3-2.2 時計の装飾

日時を表示することができたら、次は時計を装飾しましょう。装飾にはこれまでに学んだ pos,color,font,picload,mmpplay などが利用できます。

```
1 //使う変数の宣言 (無くてもOK)
2 Year = 0
3 Month = 0
4 DayOfWeek = 0
5 Day = 0
6 Hour = 0
7 Minute = 0
8 Second = 0
9 Youbi = ""
10 repeat
11     //gettimeの処理
12     Year = gettime(0)
13     Month = gettime(1)
14     DayOfWeek = gettime(2)
15     Day = gettime(3)
16     Hour = gettime(4)
17     Minute = gettime(5)
18     Second = gettime(6)
19
20     //数字になっている曜日を文字に変更
21     if (DayOfWeek = 0){ //0番なら
22         Youbi = "Sun" //Sunday
23     }else : if (DayOfWeek = 1){
24         Youbi = "Mon"
25     }else : if (DayOfWeek = 2){
26         Youbi = "Tue"
27     }else : if (DayOfWeek = 3){
28         Youbi = "Wed"
29     }else : if (DayOfWeek = 4){
30         Youbi = "Thu"
31     }else : if (DayOfWeek = 5){
32         Youbi = "Fri"
33     }else : if (DayOfWeek = 6){
34         Youbi = "Sat"
35     }
36
37     //文字の表示
38     pos 150,150
39     font "游明朝",20,2
40     color 35, 59, 108
41     mes "NowTime(JST)"
42     mes str(Year) + "/" + str(Month) + "/" + str(Day) + "(" + Youbi + ")"
43     font "游明朝",50,1
44     mes str(Hour) + ":" + str(Minute) + ":" + str(Second)
45     await 1000
46     //表示の初期化
47     cls
48 loop
```



第3章 プログラム作成

3-3 ボールを動かす

次は、アクションゲームなど動きのあるコンテンツを作成するために必須の、物体の移動を行ってみましょう。

3-3.1 ボールの移動

まずは、ウィンドウにボールを表示してみましょう。ボールは circle 命令で描画することができます。

```
circle p1,p2,p3,p4,p5 //円を描画する
```

p1=0~(0) : 矩形の左上 X 座標

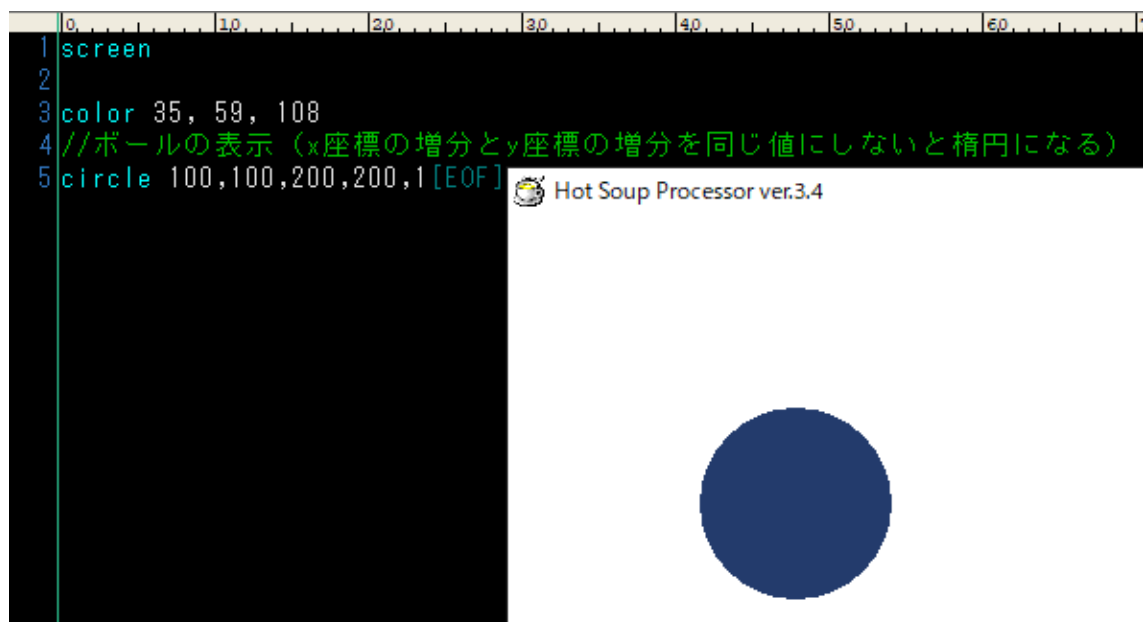
p2=0~(0) : 矩形の左上 Y 座標

p3=0~ : 矩形の右下 X 座標

p4=0~ : 矩形の右下 Y 座標

p5=0~1(1) : 描画モード(0=線,1=塗りつぶし)

早速、color で好きな色を指定して、ボールを描画してみましょう。




うまく描画されましたか？うまくできた場合、今度はその表示されたボールを移動させてみましょう。ボールの初期位置を変数に入力して circle 命令をループさせ、await 命

第3章 プログラム作成

令を追加して矩形の X,Y 座標に一度のループで移動する距離を代入演算子（第2章—2 数値計算参照）で入力するとボールが移動するはずです。今回は横方向に動かしたいので x に 10 程度の増分で試してみましょう。


```
1 screen
2
3 BallUpperLeftX = 100
4 BallUpperLeftY = 100
5 BallBottomRightX = 200
6 BallBottomRightY = 200
7
8 color 35, 59, 108
9 repeat
10 BallUpperLeftX += 10
11 BallBottomRightX += 10
12 //ボールの表示 (x座標の増分とy座標の増分を同じ値にしないと楕円になる)
13 circle BallUpperLeftX, BallUpperLeftY, BallBottomRightX, BallBottomRightY, 1
14 wait 10
15 loop[EOF]
```



実行結果を見てみるとボールの軌跡が表示されており、帯のようになってしまいました。これは、先ほど説明した画面の初期化処理が書かれてないために起こることです。boxf 命令と redraw 命令を利用してちらつきを防ぎつつ、ループするごとに画面を初期化させてみましょう。

第3章 プログラム作成

```
1 screen
2
3 BallUpperLeftX = 100
4 BallUpperLeftY = 100
5 BallBottomRightX = 200
6 BallBottomRightY = 200
7
8 repeat
9     redraw 0
10    color 35, 59, 108
11    BallUpperLeftX += 10
12    BallBottomRightX += 10
13    redraw 1
14    //ボールの表示 (x座標の増分とy座標の増分を同じ値にしないと楕円になる)
15    circle BallUpperLeftX, BallUpperLeftY, BallBottomRightX, BallBottomRightY, 1
16    wait 10
17    color 255,255,255
18    boxf
19 loop [EOF]
```



うまくボールが移動しましたか？うまくボールが移動できれば最初のステップはクリアしたことになります。

3-3.2 ボールが跳ね返るようにする

前のステップで作ったプログラムを実行して、少し気になる箇所があります。それは、ボールが画面の端を貫通してどこまでも進んでいってしまうことです。そこで、今回はボールが画面端で反射するようにしてみましょう。

ボールを壁に反射させるためには、ボールや壁に当たり判定を作る必要があります。したがって、外側の壁はサイズを固定するために screen 命令でスクリーンの大きさを調整し、ボールは円形の当たり判定は非常に面倒くさいため四角形の当たり判定で代用することとします。

プログラムの趣旨としては、ボールを移動させ、ボールの当たり判定のラインが画面端のラインを越えれば、ボールの移動方向を反対にすればよいわけです。例えば、ボールが右側に移動していて、ボールの右側のライン（ボール右の当たり判定）が、右の画面端のラインよりも外側へ行けば、ボールの移動方向を－に書き換えてやればよいということになります。実際にプログラムを組んで確認してみましょう。

第3章 プログラム作成

```
1 screen 0,800,600
2 //ボール左上の座標
3 BallUpperLeftX = 100
4 BallUpperLeftY = 100
5 //ボール右下の座標
6 BallBottomRightX = 200
7 BallBottomRightY = 200
8 //ボールの移動スピード
9 SpeedX = 20
10 SpeedY = 10
11
12 repeat
13     //仮想画面に飛ぶ
14     redraw 0
15     //画面の初期化
16     color 255,255,255
17     boxf
18     //ボールの左側が左の画面端の外へ行くと、xの移動スピードを反転
19     if (BallUpperLeftX < 0) : SpeedX *= -1
20     //ボールの上側が上の画面端の外へ行くと、yの移動スピードを反転
21     if (BallUpperLeftY < 0) : SpeedY *= -1
22     //ボールの右側が右の画面端の外へ行くと、xの移動スピードを反転
23     if (BallBottomRightX > 800) : SpeedX *= -1
24     //ボールの下側が下の画面端の外へ行くと、yの移動スピードを反転
25     if (BallBottomRightY > 600) : SpeedY *= -1
26     //ボールの移動の計算
27     BallUpperLeftX += SpeedX
28     BallBottomRightX += SpeedX
29     BallUpperLeftY += SpeedY
30     BallBottomRightY += SpeedY
31     //ボールの表示 (x座標の増分とy座標の増分を同じ値にしないと楕円になる)
32     color 35, 59, 108
33     circle BallUpperLeftX, BallUpperLeftY, BallBottomRightX, BallBottomRightY, 1
34     //裏画面ですべて表示が終わったら表画面に反映
35     redraw 1
36     //wait命令だとガクガクになるのでawait命令使用
37     await 16
38 loop [EOF]
```

3-3.3 操作できるボールの追加

勝手に画面上を跳ね回ってくれるボールが完成したところで、今度は自分で操作できるボールを追加してみましょう。操作できるボールの追加は、getkey/stick 命令でカーソルキーの入力を取得し、入力するキーの種類に応じてボールを移動することによって実装できます。しかし、ボールの数が増えるにしたがって変数の数も増えるため、登録する変数が被らないように注意しましょう。今回は、変数の整理のためにマクロを利用してみました。

```
#define マクロ名 マクロ定義 //新規マクロを登録する
```


第3章 プログラム作成

マクロは、マクロ名で指定されたキーワードを指定された定義に置き換えられるようにプリプロセッサに登録をします。例えば今回のように、変数名を後ろに置くことで変数を別名に単純化したり、固有の数値や文字（ディスプレイサイズやウィンドウサイズ、ファイルのパスなど）の意味が分かりやすくなるようにしたり、よく使う命令をマクロとして登録することもできます。

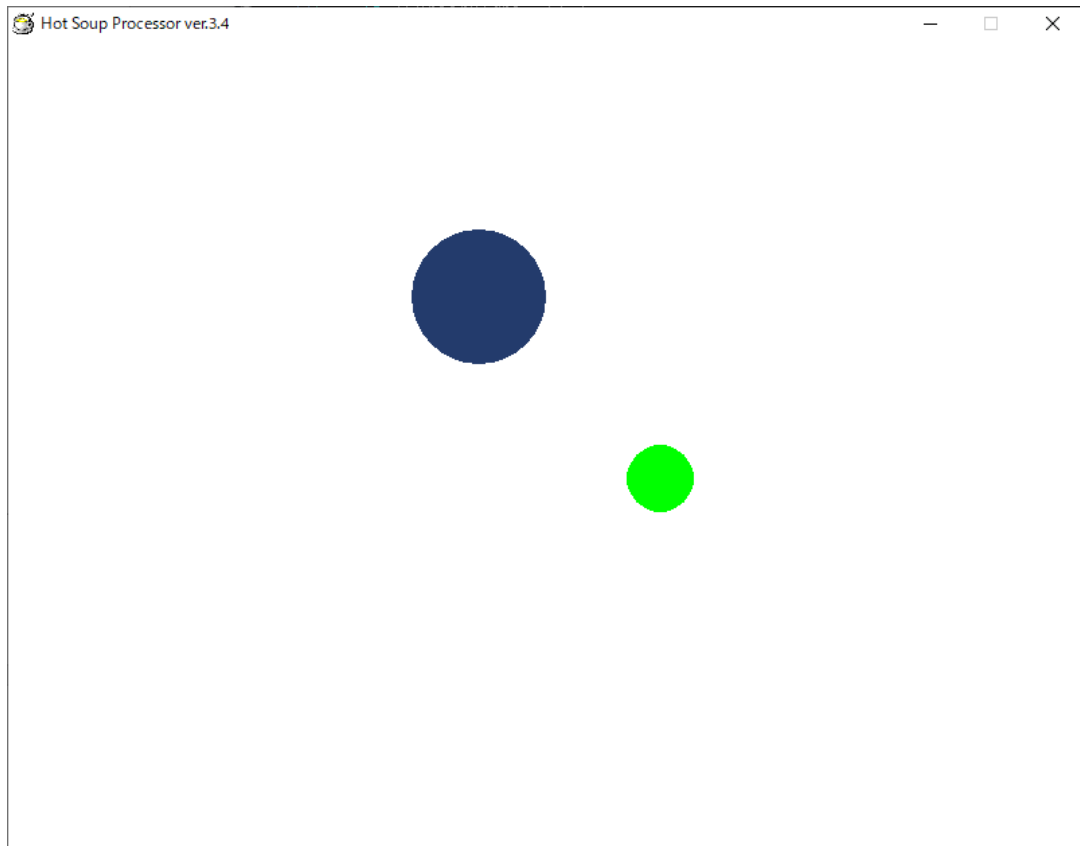
```
1 screen 0,800,600
2 //ボール左上の初期座標
3 BallUpperLeftX = 100
4 BallUpperLeftY = 100
5 //ボール右下の初期座標
6 BallBottomRightX = 200
7 BallBottomRightY = 200
8 //ボールの移動スピード
9 SpeedX = 20
10 SpeedY = 10
11 //動かせるボール左上の初期座標
12 MoveableBallUpperLeftX = 500
13 MoveableBallUpperLeftY = 500
14 //動かせるボール右下の初期座標
15 MoveableBallBottomRightX = 550
16 MoveableBallBottomRightY = 550
17 //動かせるボールの移動スピード
18 MoveableBallSpeedX = 20
19 MoveableBallSpeedY = 20
20
21 //マクロを設定（長すぎる変数名を省略するため）
22 //1つ目のボールのマクロを設定
23 #define BALL_1_UPPER BallUpperLeftY
24 #define BALL_1_BOTTOM BallBottomRightY
25 #define BALL_1_LEFT BallUpperLeftX
26 #define BALL_1_RIGHT BallBottomRightX
27 //2つ目のボールのマクロを設定
28 #define BALL_2_UPPER MoveableBallUpperLeftY
29 #define BALL_2_BOTTOM MoveableBallBottomRightY
30 #define BALL_2_LEFT MoveableBallUpperLeftX
31 #define BALL_2_RIGHT MoveableBallBottomRightX
32
33 repeat
34     //仮想画面に飛ぶ
35     redraw 0
36     //画面の初期化
37     color 255,255,255
38     boxf
39     //1つ目のボールの処理=====
40     //ボールの反射
41     //ボールの左側が左の画面端の外へ行くと、xの移動スピードを反転
42     if (BALL_1_LEFT < 0) : SpeedX *= -1
43     //ボールの上側が上の画面端の外へ行くと、yの移動スピードを反転
44     if (BALL_1_UPPER < 0) : SpeedY *= -1
45     //ボールの右側が右の画面端の外へ行くと、xの移動スピードを反転
46     if (BALL_1_RIGHT > 800) : SpeedX *= -1
47     //ボールの下側が下の画面端の外へ行くと、yの移動スピードを反転
48     if (BALL_1_BOTTOM > 600) : SpeedY *= -1
49     //ボールの移動の計算
50     BALL_1_LEFT += SpeedX
51     BALL_1_RIGHT += SpeedX
```

第3章 プログラム作成

```
49 //ボールの移動の計算
50 BALL_1_LEFT += SpeedX
51 BALL_1_RIGHT += SpeedX
52 BALL_1_UPPER += SpeedY
53 BALL_1_BOTTOM += SpeedY
54 //ボールの表示 (x座標の増分とy座標の増分を同じ値にしないと楕円になる)
55 color 35, 59, 108
56 circle BALL_1_LEFT, BALL_1_UPPER, BALL_1_RIGHT, BALL_1_BOTTOM, 1
57
58 //2つ目のボールの処理 (自分で動かせるほう) =====
59 //カーソルキー取得
60 getkey CursorUp,38
61 getkey CursorDown,40
62 getkey CursorLeft,37
63 getkey CursorRight,39
64 //ボールの移動
65 if (CursorUp = 1) {
66     BALL_2_UPPER -= MoveableBallSpeedY
67     BALL_2_BOTTOM -= MoveableBallSpeedY
68 }
69 if (CursorDown = 1) {
70     BALL_2_UPPER += MoveableBallSpeedY
71     BALL_2_BOTTOM += MoveableBallSpeedY
72 }
73 if (CursorLeft = 1) {
74     BALL_2_LEFT -= MoveableBallSpeedX
75     BALL_2_RIGHT -= MoveableBallSpeedX
76 }
77 if (CursorRight = 1) {
78     BALL_2_LEFT += MoveableBallSpeedX
79     BALL_2_RIGHT += MoveableBallSpeedX
80 }
81 //ボールが境界より外に行かないようにするための処理
82 //ボールが境界を越えたら反対方向へ移動させる
83 if (BALL_2_UPPER < 0) {
84     BALL_2_UPPER += MoveableBallSpeedY
85     BALL_2_BOTTOM += MoveableBallSpeedY
86 }
87 if (BALL_2_BOTTOM > 600) {
88     BALL_2_UPPER -= MoveableBallSpeedY
89     BALL_2_BOTTOM -= MoveableBallSpeedY
90 }
91 if (BALL_2_RIGHT > 800) {
92     BALL_2_LEFT -= MoveableBallSpeedX
93     BALL_2_RIGHT -= MoveableBallSpeedX
94 }
95 if (BALL_2_LEFT < 0) {
96     BALL_2_LEFT += MoveableBallSpeedX
97     BALL_2_RIGHT += MoveableBallSpeedX
98 }
```

```
99 //ボールの表示
100 color 0, 255, 0
101 circle BALL_2_LEFT, BALL_2_UPPER, BALL_2_RIGHT, BALL_2_BOTTOM, 1
102 |
103 //裏画面ですべて表示が終わったら表画面に反映
104 redraw 1
105 //wait命令だとガクガクになるのでawait命令使用
106 await 16
107 loop[EOF]
```

第3章 プログラム作成



3-3.4 ボールの数を増やす

ここまでできれば、基本的なアクションゲームの要素はクリアできたと思います。最後に思い思いにボールやオブジェクトを追加して遊んでみましょう。ボール同士が激突するようにしたり、ボールの座標を端の方に表示してみたり、ボールの速度を変えられるようにしたり、跳ね返るたびにボールの色が変わったりしても面白いかもしれません。

第3章 プログラム作成

3-4 じゃんけんゲーム

続いてはじゃんけんゲームを作成してみましょう。じゃんけんゲームの作成を通して画像の表示や効果音の再生などを学びます。

3-4.1 素材の用意

じゃんけんゲームを作成するために。まずはグー/チョキ/パーの画像を用意しましょう。用意出来たら、まずはその画像たちを並べて配置するプログラムを作成していきます。今回はカスタムボタンというものを利用してみます。

```
objimage id,x1,y1,x2,y2,x3,y3 //カスタムボタンの設定
```

id : カスタムボタンの参照バッファ ID

x1,y1 : カスタムボタンの参照座標 1(通常時)

x2,y2 : カスタムボタンの参照座標 2(押し下げ時)

x3,y3 : カスタムボタンの参照座標 3(マウスオーバー時)

カスタムボタンの設定では、ボタンに張り付ける画像の設定を行うことができ、これ以降の button 命令にはこの objimage 命令で設定された画像が張り付けられます。

objimage 命令は、何もしていないとき/マウスオーバー時/マウスを押したときのそれぞれの場合に読み込んだ画像のどの部分を参照するかを決めることができます。(左上座標)

また、onjimage 命令は、直接画像を読み込むわけではなく、参照バッファに保存されている画像を読み込みます。(参照バッファとは、プログラムが画像を利用するために一時的に画像や音声を保存する場所のこと) そのため、カスタムボタンを作成する前にバッファに画像を読み込む必要があります。

```
celload "filename",p1,p2 //画像ファイルをバッファにロード
```

"filename" : ロードするファイル名

p1=1~(-1) : 読み込み先ウィンドウ ID

p2=0~1(0) : 初期化する画面モード

第3章 プログラム作成

バッファの読み込みはこのようなイメージです。

それ以外にも `mousex,mousey` という命令を利用してマウスの座標を取得してどの画像をクリックしているのか知るという方法もあります。

```
mousex //マウスカーソルの X 座標
```

```
mousey //マウスカーソルの Y 座標
```

これらの命令と `getkey` 命令を組み合わせることで、クリックされた瞬間のマウスの座標を取得し、どの画像の上でクリックしたのかを知ることができます。

今回は、カスタムボタンを使用してクリックできる画像と、画像のクリックによってメッセージが表示できることを確認しましょう。

```
1 screen 0,900,900
2 celload "pic¥¥gu.png",1
3 celload "pic¥¥choki.png",2
4 celload "pic¥¥pa.png",3
5
6 objsize 300,300
7 objimage 1
8 button goto "",*main
9 objimage 2
10 button goto "",*main
11 objimage 3
12 button goto "",*main
13
14
15 *main
16 mes "a"
17 stop
18 return
19 [EOF]
```



メッセージの表示が確認出来たら、まず、スクリーンのサイズを 900×900 、画像のサイズを 300×300 として、グー、チョキ、パーのそれぞれのカスタムボタンを $x=0,y=600$ の位置、 $x=300,y=600$ の位置、 $x=600,y=600$ の位置に表示してください。また、カスタムボタンには `goto` を使用し、ジャンプ先のラベルは `*hand1`、`*hand2`、`*hand3` としてください。次に、相手の手として、 300×300 のグーの画像を $x=300,y=0$

第3章 プログラム作成

に表示してください。カスタムボタンでなく、単に画像を表示したい場合は `cellput` 命令を使用します。

`cellput id, no, zoomx, zoomy, angle`

`id=0~(1)` : 画像素材を持つウィンドウ ID

`no=0~(0)` : 分割画像 No.

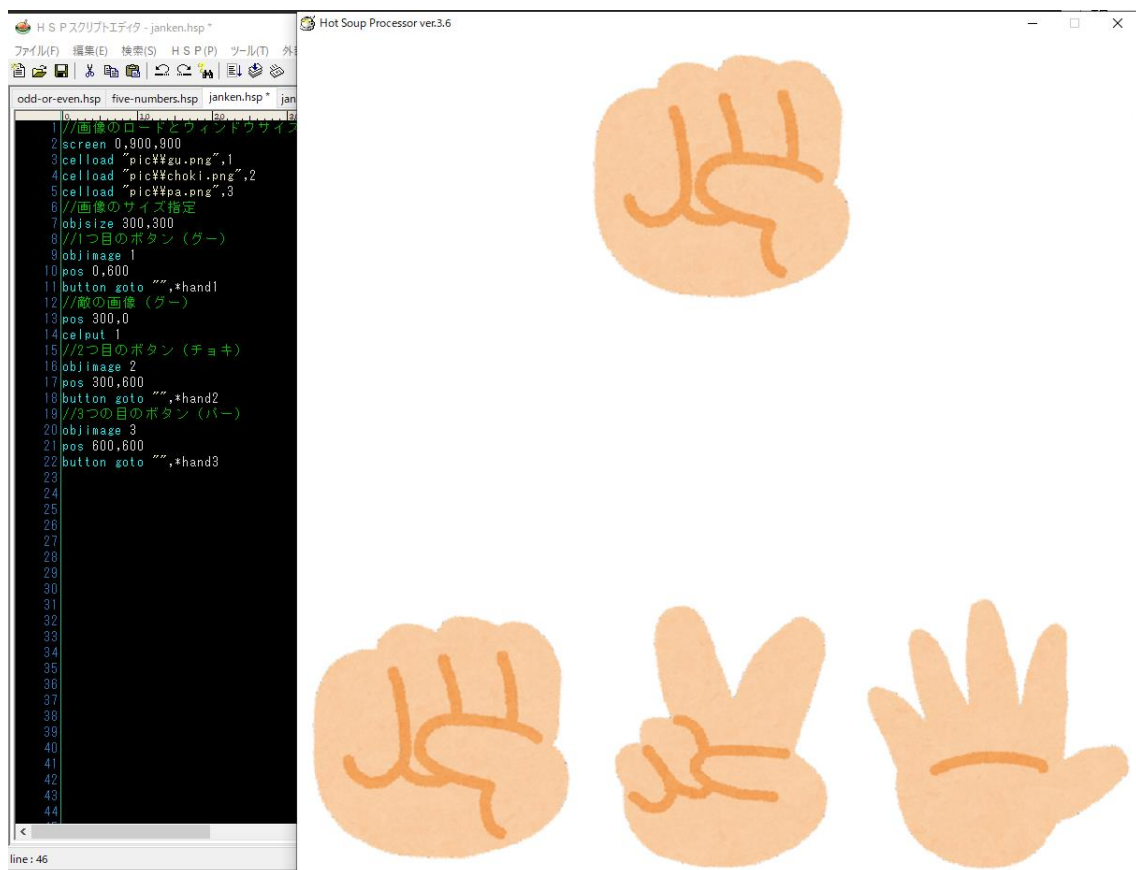
`zoomx=0.0~(1.0)` : 横方向の表示倍率(実数)

`zoomy=0.0~(1.0)` : 縦方向の表示倍率(実数)

`angle=0.0~(0.0)` : 回転角度(単位はラジアン)

`cellput` 命令で、`cellload` 命令で読み込んだ画像の ID を指定することで、画像を表示することができます。

自分の手と相手の手を表示させるプログラムは以下のようになります。



第3章 プログラム作成

3-4.2 文字の表示

前のステップで、カスタムボタンの作成と動作を行いました。ここではじゃんけんゲームのプログラムを作成しましょう。

まずはウィンドウのレイアウトです。下側に自分の手札（カスタムボタン）を並べ、上側に相手の手札、中央にテキストを表示するようにしましょう。その際、画面が窮屈になるのでウィンドウサイズも変更しておきましょう。

じゃんけんをする際、最初に「さいしょはグー じゃんけん」と中央に表示してみましょう。文字をゆっくり表示したいため、ここでは text 命令+emes 命令を利用します。

しかし、text 命令や emes 命令は HSP の標準命令ではサポートされていません。したがってプログラムの最初に拡張パッケージをインクルードする文を書かなければいけません。（詳しくは第4章で解説）

```
#include "hsp3util.as" //hsp3util.ac プラグインのインクルード
```

HSP で標準にサポートされていない命令を使用する際はこのように記述してください。

```
text p1 //修飾文字表示の待ち時間を設定する
```

p1(0) : 表示待ち時間(ms)

```
emes "strings" //修飾文字を表示
```

"strings" : 表示するメッセージまたは変数

text 命令で、何ミリ秒でテキストを表示するか設定し、その後の emes 命令で、text 命令で設定した時間をかけて文字を表示します。

実際に文字を表示させるプログラムは以下のようになります。

```
27 //じゃんけんの掛け声
28 pos 0,450
29 font "游ゴシック",50,1
30 text 250
31 emes "さいしょはグー"
32 wait 100
33 pos 550,450
34 text 300
35 emes "じゃんけん"
36 stop
```

3-4.3 自分と相手の手の表示

ゆっくり文字が表示されるようになりましたか？この処理がうまくいけば、次はボタンをクリックした際に自分の押した手が表示されるようにしましょう。そのために、自分の出した手に番号を付けます。ここでは、グーを1、チョキを2、パーを3として、ラベル*hand1 / 2 / 3 に飛んだ際に変数 MHand に値を代入しています。その後、goto でラベル*main にジャンプし、そこで自分の手を表示しています。

また、*main の部分で相手の手をランダムに決めて表示しています。その際はプログラムの最初に randomize 命令を記述してください。

この処理を実際に記述すると以下ようになります。

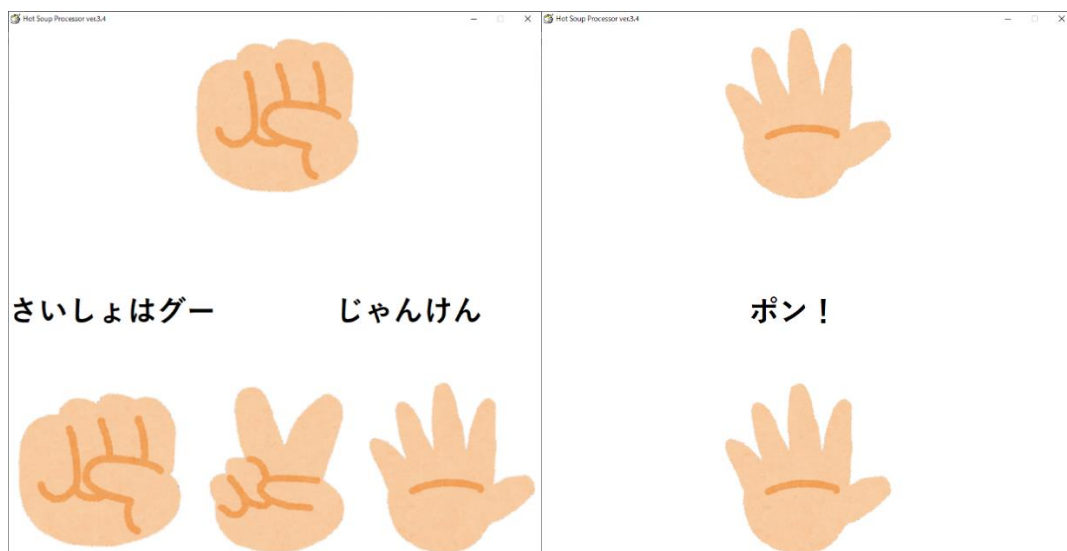
```
37 |
38 | *hand1//グーを押したときに飛ぶ処理
39 |     MHand = 1//MHandは自分の出した手
40 |     goto *main
41 | return
42 |
43 | *hand2//チョキを押したときに飛ぶ処理
44 |     MHand = 2
45 |     goto *main
46 | return
47 |
48 | *hand3//パーを押したときに飛ぶ処理
49 |     MHand = 3
50 |     goto *main
51 | return
52 |
53 | *main//じゃんけんの勝敗を決める処理
54 | cls
55 | //相手の手を決める
56 | EHand = rnd(3) + 1
57 | //ボン！の字を表示
58 | font "游ゴシック",50,1
59 | pos 350,450
60 | mes "ボン！"
61 | //相手の手を表示
62 | pos 300,0
63 | celput EHand
64 | //自分の手を表示
65 | pos 300,800
66 | celput MHand
67 |
68 |
69 | stop
70 | return
71 |
```


第3章 プログラム作成

ここまでの内容をまとめると次のようになります。

```
1 //インクルード文
2 #include "hsp3util.as"
3 //画像のロードとウィンドウサイズの調節
4 screen 0,900,900
5 celload "pic%$gu.png",1
6 celload "pic%$choki.png",2
7 celload "pic%$pa.png",3
8 //乱数の初期化
9 randomize
10 //画像のサイズ指定
11 objsize 300,300
12 //1つ目のボタン(グー)
13 objimage 1
14 pos 0,600
15 button goto "",*hand1
16 //敵の画像(グー)
17 pos 300,0
18 celput 1
19 //2つ目のボタン(チョキ)
20 objimage 2
21 pos 300,600
22 button goto "",*hand2
23 //3つ目のボタン(パー)
24 objimage 3
25 pos 600,600
26 button goto "",*hand3
27 //じゃんけんの掛け声
28 pos 0,450
29 font "游ゴシック",50,1
30 text 250
31 emes "さいしょはグー"
32 wait 100
33 pos 550,450
34 text 300
35 emes "じゃんけん"
36 stop
37

38 *hand1//グーを押したときに飛ぶ処理
39 MHand = 1//MHandは自分の出した手
40 goto *main
41 return
42
43 *hand2//チョキを押したときに飛ぶ処理
44 MHand = 2
45 goto *main
46 return
47
48 *hand3//パーを押したときに飛ぶ処理
49 MHand = 3
50 goto *main
51 return
52
53 *main//じゃんけんの勝敗を決める処理
54 cls
55 //相手の手を決める
56 EHand = rnd(3) + 1
57 //ボン!の字を表示
58 font "游ゴシック",50,1
59 pos 350,450
60 mes "ボン!"
61 //相手の手を表示
62 pos 300,0
63 celput EHand
64 //自分の手を表示
65 pos 300,600
66 celput MHand
67
68
69 stop
70 return
71
```



3-4.5 勝敗の表示

これで、じゃんけんプログラムの大きな作成はできました。最後にじゃんけんの勝敗を表示できるようにしてみましょう。

勝敗を表示するためには、自分が出した手（グー、チョキ、パー）に応じて、相手の出した手との勝敗判定をする必要があります。したがって、じゃんけんの文字を表示した直後に相手の手をランダムに決めておき、カスタムボタンからジャンプした先の*hand1/2/3 内で勝敗の処理を行うと良いでしょう。その後、*main 内で勝ち負けを表示してください。今回のプログラムでは、勝敗を WorL という変数に格納しており、勝ちの場合は1、引き分けの場合は2、負けの場合は3となります。

```
31 emes "さいしょはグー"
32 wait 100
33 pos 550,450
34 text 300
35 emes "じゃんけん"
36 //相手の手を決める 1 : グー 2 : チョキ 3 : パー
37 EHand = rnd(3) + 1
38 stop
39
40 *hand1 //グーを押したときに飛ぶ処理
41 MHand = 1 //MHandは自分の出した手 1 : グー 2 : チョキ 3 : パー
42 if Ehand = 1 {
43     WorL = 2 //WorLは勝敗 勝ち : 1 引き分け : 2 負け : 3
44 } else : if Ehand = 2 {
45     WorL = 1
46 } else : if Ehand = 3 {
47     WorL = 3
48 }
49 goto *main
50 return
51
52 *hand2 //チョキを押したときに飛ぶ処理
53 MHand = 2
54 if Ehand = 1 {
55     WorL = 3
56 } else : if Ehand = 2 {
57     WorL = 2
58 } else : if Ehand = 3 {
59     WorL = 1
60 }
61 goto *main
62 return
63
64 *hand3 //パーを押したときに飛ぶ処理
65 MHand = 3
66 if Ehand = 1 {
67     WorL = 1
68 } else : if Ehand = 2 {
69     WorL = 3
70 } else : if Ehand = 3 {
71     WorL = 2
72 }
73 goto *main
74 return
75
```

第3章 プログラム作成

これまでのプログラムをまとめると以下のようになります。

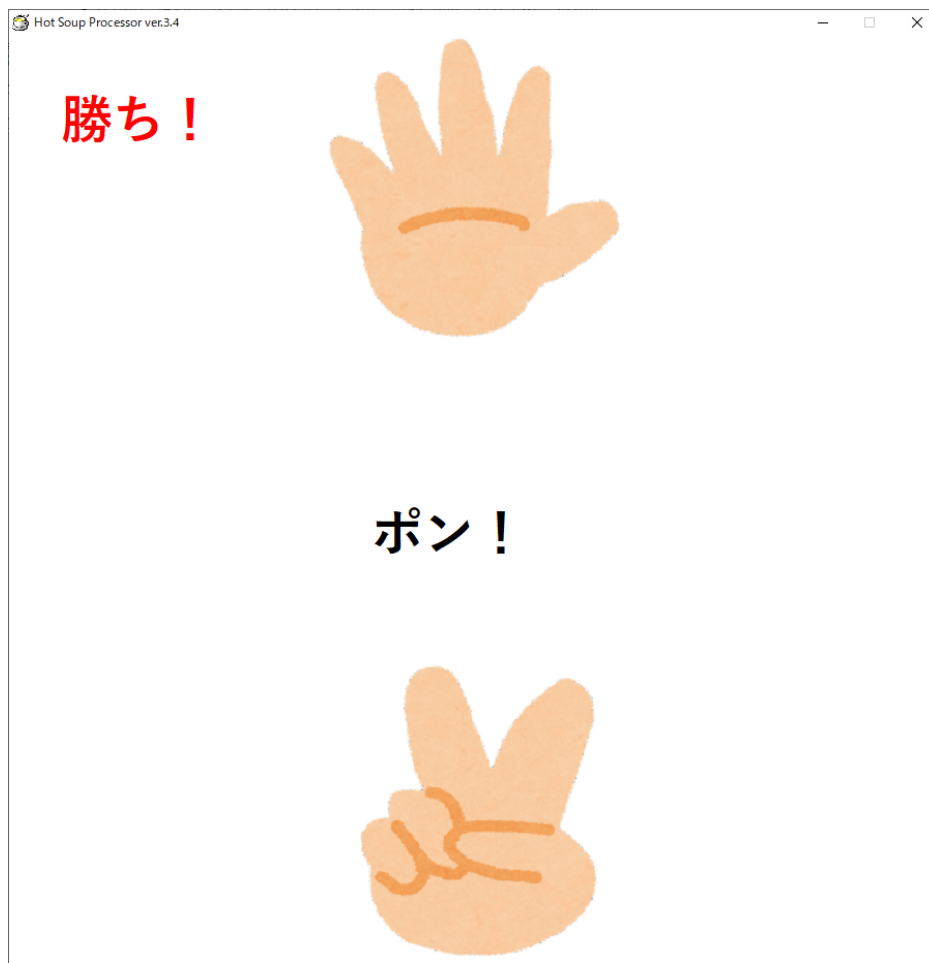
```
1 //インクルード文
2 #include "hsp3util.as"
3 //画像のロードとウィンドウサイズの調節
4 screen 0,900,900
5 celload "pic%$gu.png",1
6 celload "pic%$choki.png",2
7 celload "pic%$pa.png",3
8 //乱数の初期化
9 randomize
10 //画像のサイズ指定
11 objsize 300,300
12 //1つ目のボタン (グー)
13 objimage 1
14 pos 0,600
15 button goto "",*hand1
16 //敵の画像 (グー)
17 pos 300,0
18 celput 1
19 //2つ目のボタン (チョキ)
20 objimage 2
21 pos 300,600
22 button goto "",*hand2
23 //3つ目のボタン (パー)
24 objimage 3
25 pos 600,600
26 button goto "",*hand3
27 //じゃんけんの掛け声
28 pos 0,450
29 font "游ゴシック",50,1
30 text 250
31 emes "さいしょはグー"
32 wait 100
33 pos 550,450
34 text 300
35 emes "じゃんけん"
36 //相手の手を決める 1 : グー 2 : チョキ 3 : パー
37 EHand = rnd(3) + 1
38 stop
39
40 *hand1 //グーを押したときに飛ぶ処理
41 MHand = 1 //MHandは自分の出した手 1 : グー 2 : チョキ 3 : パー
42 if Ehand = 1 {
43     WorL = 2 //WorLは勝敗 勝ち : 1 引き分け : 2 負け : 3
44 }else : if Ehand = 2 {
45     WorL = 1
46 }else : if Ehand = 3 {
47     WorL = 3
48 }
49 goto *main|
50 return
```

```

52 *hand2//チョキを押したときに飛ぶ処理
53     MHand = 2
54     if Ehand = 1{
55         WorL = 3
56     }else : if Ehand = 2{
57         WorL = 2
58     }else : if Ehand = 3{
59         WorL = 1
60     }
61     goto *main
62 return
63
64 *hand3//パーを押したときに飛ぶ処理
65     MHand = 3
66     if Ehand = 1{
67         WorL = 1
68     }else : if Ehand = 2{
69         WorL = 3
70     }else : if Ehand = 3{
71         WorL = 2
72     }
73     goto *main
74 return
75
76 *main//じゃんけんの勝敗を決める処理
77 cls
78 //ボン!の字を表示
79 font "游ゴシック",50,1
80 pos 350,450
81 mes "ボン!"
82 //相手の手を表示
83 pos 300,0
84 celput EHand
85 //自分の手を表示
86 pos 300,600
87 celput MHand
88 //勝敗処理
89 pos 50,50
90 if WorL = 1{
91     color 255,0,0
92     mes "勝ち!"
93 }else : if WorL = 2{
94     color 255,0,255
95     mes "引き分け"
96 }else : if WorL = 3{
97     color 0,0,255
98     mes "負け..."
99 }
100 stop
101 return
102 [EOF]

```

第3章 プログラム作成



ここまで出来たらじゃんけんプログラムの完成です。ここからのアレンジとして、何回も繰り返しプレイできるようにする、○勝○負○分と勝率の表示、効果音の追加などがあります。作成したプログラムを書き換えて楽しんでみてください。