ゲームを作りたいプログラミング初心者におすすめ！

# HSPでシューティングゲームを作ろう！

# 第一章　HSPのインストール
## ｼｭｰﾃｨﾝｸﾞ：自機の移動

# HSPのダウンロード＆インストール

## ▶http://hsp.tv/

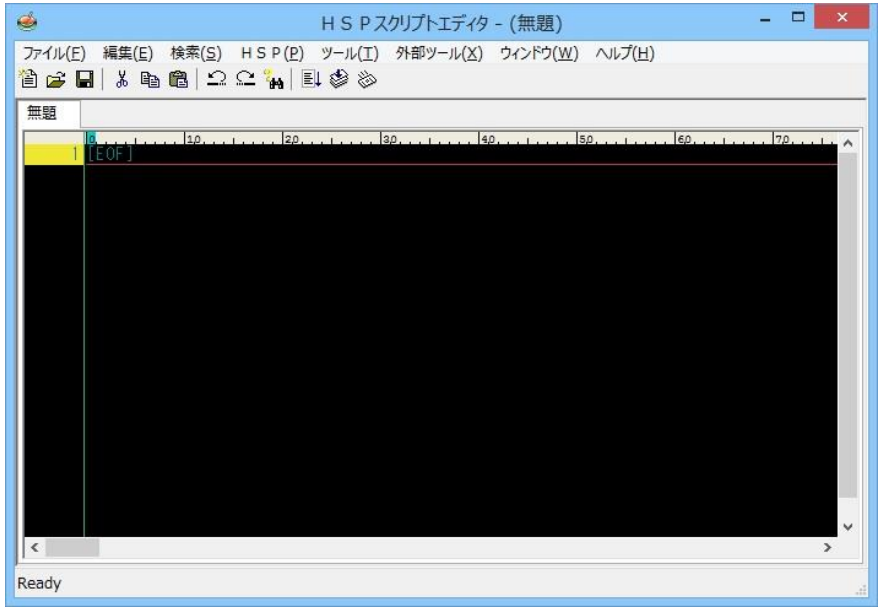
↑のURLからHSPをダウンロード

1. 


2.メアドを入力せずに[Free Download HSP]

# 起動してみよう


デスクトップに出来た左図のアイコンをダブルクリックして起動



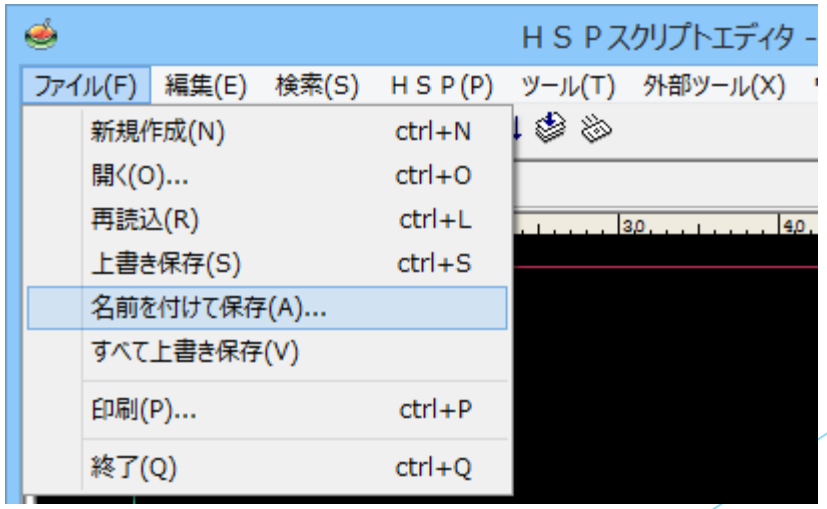

←の画面が出たら
起動成功

# 作業フォルダの作成


- まずはデスクトップに適当な名前のフォルダを作ろう

- HSPスクリプトエディタのメニューバーから
  「ファイル」→「名前を付けて保存」

  デスクトップに作ったフォルダを選んで適当な
  名前を付けて保存！



# 命令とパラメータ

- HSPのスクリプトは主に**命令**と**パラメータ**からなる

命　令　：コンピュータに何をさせるのかを
知らせるための単語

パラメータ：命令を使うときにより細かな設定をする
ための情報

例：　mes “Hello, HSP World!“

命令　　パラメータ

Hello, HSP Worldというテキストを表示

# 変数

- 変数とは・・・

  **文字列**や**数字**を入れるための**箱**(入れ物)

  自由に好きな名前を付けることが出来る

  例：　　a = 100　　aという変数に100を代入

  　　str = “hsp”　　strという変数にhspという文字列を代入

# さっそくプログラムを書いてみよう

```
＊main
    screen 0, 400, 600
    font msgothic, 32
    repeat
        color 255, 255, 255
        boxf
        color 127, 191, 255
        pos x, y
        mes “凸”
        await (1000/60)
    loop
```

書き終わったら**F5**

もしくはメニューバーの
ＨＳＰ → コンパイル＋実行

# プログラムの解説

p1 ：パラメータ1の意味
p2 ：パラメータ2の意味
p3 ：パラメータ3の意味
末尾の…… ：まだ後ろにパラメータが続く

- screen p1, p2, p3, ……

「指定したウィンドウIDを初期化して使用できるようにします。」
p1：ウィンドウID
p2：初期化する画面サイズX（1ドット単位）
p3：初期化する画面サイズY（1ドット単位）

- font “fontname”, p1, p2

「テキスト書体の設定をします。」
“fontname”：フォント名　今回はMSゴシックを使用
p1：フォントの大きさ
p2：フォントのスタイル　今回は省略したので0 (通常のスタイル)

- repeat p1, ……
- loop

「repeat～loopの間をくり返し実行します。」
p1：ループ回数　省略すると無限ループ

- boxf p1, p2, p3, p4

「現在の描画色で矩形 (四角形)を塗りつぶします。」
p1：矩形の左上X座標
p2：矩形の左上Y座標
p3：矩形の右上X座標
p4：矩形の右上Y座標
※全パラメータを省略すると全体塗りつぶし

- color p1, p2, p3

「メッセージ表示、描画などの色を指定した値に設定します。」
p1：R（赤）の輝度
p2：G（緑）の輝度
p3：B（青）の輝度
※0, 0, 0で黒　255, 255, 255で白

- pos p1, p2

「メッセージ表示、オブジェクトの表示などの基本座標となる
カレントポジションの座標を指定します。」
p1：カレントポジションのX座標
p2：カレントポジションのY座標

- mes “strings”

「ウィンドウ内に、指定されたメッセージを表示します。 」
“strings”：表示するメッセージ、または変数

“”（ダブルクォーテーション）で囲んであるものは
文字列として認識されます

例：　”string”の場合はstringという文字列を出力
stringの場合はstringという変数の内容を出力

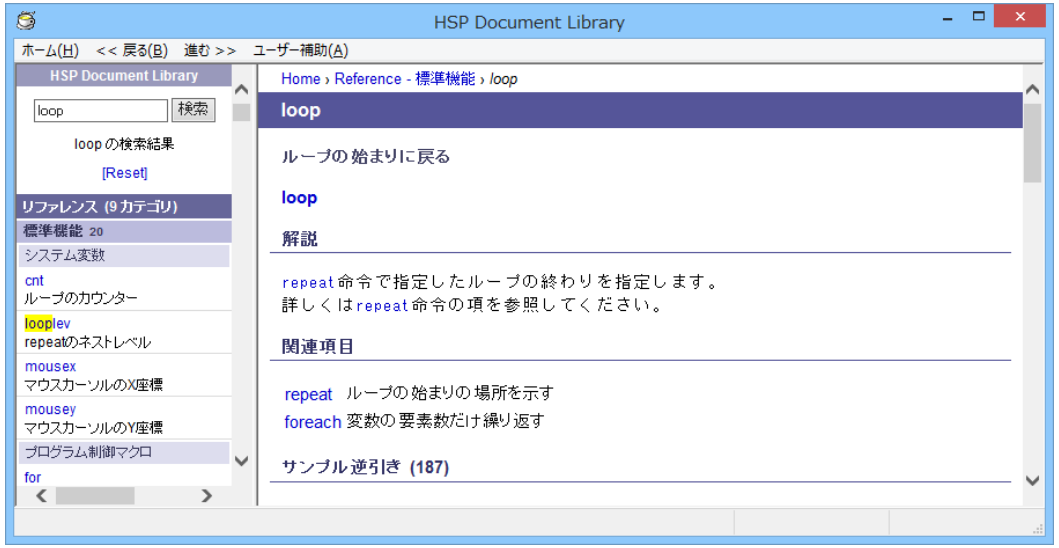
- await p1

「プログラムの実行を一定時間だけ中断します。 」
p1：待ち時間(1ms単位)
※今回は**(1000/60)** msなので1秒間に60回ゲーム画面が更新される

# ヘルプを活用しよう

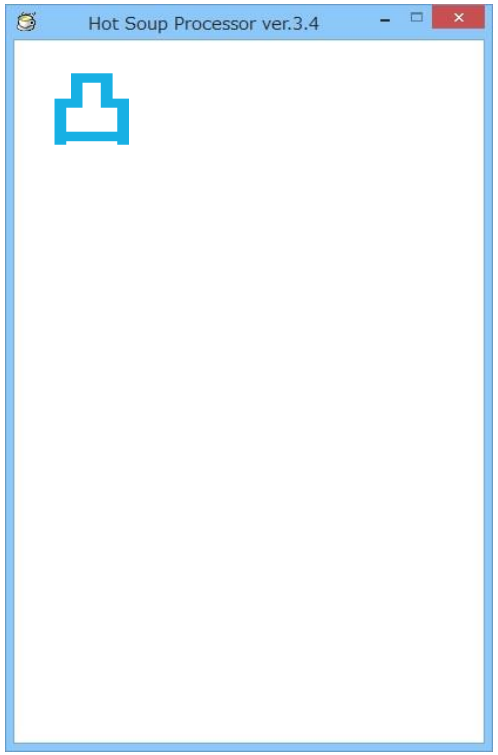
- 命令の説明を毎回するのも大変なので**ヘルプ**で確認

命令の後ろにカーソルを置いて**F1**を押すと




↑のウィンドウが開き命令の内容が確認できる！

# どういう動きをしてるの？




1. screen命令でウィンドウを作成
2. boxf命令で画面を白く塗りつぶす
3. mes命令で”凸”という文字列を表示
4. await命令で1/60秒待機
5. loop命令でrepeatへ（2まで戻る）

# 自機を動かしてみよう

- 自機を動かすには・・・

  自機を表示する座標を少しづつ変化させる

  このプログラムでの自機の座標を示す変数はxとy
  キーボードのカーソルキーで動かしたい

  キーボードのカーソルキーでxとyを変化させる

# 押されたキーを検出する

- stick  p1, p2, ……

  p1：読み込むための変数
  p2：非トリガータイプキー指定

  押されているキーの情報がp1に格納される

  押された瞬間に一度だけ情報を格納

  キーが押されている間はずっと格納されていてほしい！

  p2で押されている間ずっと検出されるキーを指定

# stick命令

緑の文字は入力済みのスクリプト

```
    ⋮
repeat
    color 255, 255, 255
    boxf
    gosub *key_check
    color 127, 191, 255
    ⋮
loop

*key_check
```

```
stick key, 5
if (key & 1) : x = x - 1
if (key & 4) : x = x + 1

return
```

書き終わったら**F5**

**カーソルキー左右で動かしてみよう**

# プログラムの解説

- gosub *label

「指定したラベルにサブルーチンジャンプをします。」
＊label：ラベル名

ラベルとは・・・
スクリプトの中のある位置を表す目印のようなもの

ラベルを使って動作ごとにスクリプトを分けると見やすくなる！

このスクリプトの中には＊main と ＊key_check の２つがある
（自由に作成可能）

- return

「サブルーチンを終了し、呼びだされた行の次の行へ戻ります。」

- if p1

「p1の条件式が満たされていれば、それ以降の命令を実行し、
満たされていない場合は、次の行に移ります。」
p1：条件式

```
if (key & 1) : x = x - 1
```

条件式を括弧でくくると
見やすい

：（コロン）の後に
実行する命令を記述

# 変数に値を入れてみよう

- 現在は自機が画面上部に表示されている

  **画面下部**に表示させたい

自機の縦方向の位置を表す

変数 y に値を入れる

```
＊main
    screen 0, 400, 600
    font msgothic, 32
    y = 470
    repeat
        color 255, 255, 255
        boxf
     ⋮
```

# 画面のチラつきをなくそう

- 実行すると画面がチラつく・・・

「画面に描画していく過程が見えている」状態

画面クリア　→　自機表示　の間も見えている

・仮の画面に自機を配置してから実際の画面にコピーする
・自機などの描画中は画面を更新しないようにする

# redraw命令

```
*main
    screen 0, 400, 600
    font msgothic, 32
    y = 470
    repeat
        redraw 0
        color 255, 255, 255
        boxf
        ⋮
        ⋮
        pos x, y
        mes"凸"
        redraw 1
        await (1000/60)
    loop
    ⋮
```

書き終わったら**F5**

**画面のチラつきを確認してみよう**

# redraw命令

- redraw p1, …

「画面の描画モードを指定します。」
p1：描画モード

描画モードについて

描画モード0：画面制御命令(mesなど)が実行されても
仮想画面を書き換えるだけで
実際の画面には反映されません。

描画モード1：画面制御命令が実行されると、実際の画面にも反映されます。

# 第一章　HSPのインストール
## ｼｭｰﾃｨﾝｸﾞ：自機の移動

次回は「弾丸の発射」についてやるかも

おわり