

# D言語ゲームライブラリ「dHell2」

## これは何？

D言語で簡単にゲームが作れるようにライブラリ「dHell」の続編です。  
OpenGL使っているのに、2Dゲーム専用だったりします。

## 謝辞

- このアプリはD言語を使っています。( version 1.0.0 )
- SDL ( Simple DirectMedia Layer ) を使っています。
- OpenGLを使っています。
- shinichiro\_hさんのD - porting ( <http://shinh.skr.jp/d/porting.html> ) のD言語用SDL/OpenGLヘッダファイルを使っています。
- hell.bmpはtaroさんの著作物です ( <http://www5.atwiki.jp/yaruhara/pages/45.html> )
- おめがさんの「RectWinderNTGL」のコードをかなり使わせていただきました。
- フォントにはおめがさんの「font.bmp」を使わせていただきました

---

## 目次

---

- [D言語ゲームライブラリ「dHell2」](#)
  - [これは何？](#)
  - [謝辞](#)
  - [ダウンロード](#)
  - [更新履歴](#)
  - [既知の問題](#)
  - [dHellであった機能](#)
  - [dHell2の新機能](#)
  - [Dコンパイラのインストールなど](#)
  - [ビルド方法](#)
  - [使い方](#)
  - [特殊なキーの動作](#)
  - [「dHell2」リファレンス](#)
    - [Hell\\_init\(char\[\] caption, int width, int height, bool fullscreen=false\)](#)
    - [Hell\\_quit\(\)](#)
    - [Hell\\_setCaption\(...\)](#)
    - [Hell\\_write\(...\)](#)
    - [Hell\\_write\(Exception e\)](#)
    - [Hell\\_loadTexture\(char\[\] key, char\[\] filepath, int\[3\] mask=\[-1, 0, 0\]\)](#)
    - [Hell\\_drawTexture\(char\[\] key, float cx, float cy, int ox=0, int oy=0, int ow=0, int oh=0, float dx=1.0f, float dy=1.0f, float rot=0, int r=0xFF, int g=0xFF, int b=0xFF, int a=0xFF\)](#)
    - [Hell\\_drawTextureEx\(char\[\] key, float cx, float cy, int ox=0, int oy=0, int ow=0, int oh=0, float dx=1.0f, dy=1.0f, float rot=0, int r=0xFF, int g=0xFF, int b=0xFF, int a=0xFF\)](#)

- [Hell\\_hasTexture\(char\[\] key\)](#)
- [Hell\\_capture\(char\[\] key\)](#)
- [Hell\\_setAlpha\(char\[\] key, int alpha\)](#)
- [Hell\\_getMouseX](#)
- [Hell\\_getMouseY](#)
- [Hell\\_isPressMouse](#)
- [Hell\\_isPushMouse](#)
- [Hell\\_isPressKey](#)
- [Hell\\_isPushKey](#)
- [Hell\\_loadWAV\(char\[\] key, char\[\] path\)](#)
- [Hell\\_playWAV\(char\[\] key, int loops=0, int channel=-1\)](#)
- [Hell\\_stopWAV\(\)](#)
- [Hell\\_disposeWAV\(char\[\] key=null\)](#)
- [bool Hell\\_isPressJKey\(int id\)](#)
- [bool Hell\\_isPushJKey\(int id\)](#)
- [bool Hell\\_isPressJButton\(int id\)](#)
- [bool Hell\\_isPushJKey\(int id\)](#)
- [Hell\\_isPressEnter\(\)/Hell\\_isPushEnter\(\)](#)
- [Hell\\_isPressCancel\(\)/Hell\\_isPushCancel\(\)](#)
- [Hell\\_isPressMenu\(\)/Hell\\_isPushMenu\(\)](#)
- [Hell\\_isPressUp\(\)/Hell\\_isPushUp\(\)](#)
- [Hell\\_isPressLeft\(\)/Hell\\_isPushLeft\(\)](#)
- [Hell\\_isPressDown\(\)/Hell\\_isPushDown\(\)](#)
- [Hell\\_isPressRight\(\)/Hell\\_isPushRight\(\)](#)
- [Hell\\_playBgm\(char\[\] path, int loops=-1, int ms=0\)](#)
- [Hell\\_stopBgm\(int ms=0\)](#)
- [Hell\\_isPlayBgm\(\)](#)
- [Hell\\_drawRect\(float x=0, float y=0, float dx=0, float dy=0, float width=0, float rot=0, int r=0x00, int g=0x00, int b=0x00, int a=0xFF\)](#)
- [Hell\\_drawRectEx\(float cx=0, float cy=0, float dx=0, float dy=0, float width=0, float rot=0, int r=0x00, int g=0x00, int b=0x00, int a=0xFF\)](#)
- [void Hell\\_drawLine\(float x1, float y1, float x2, float y2, int width=1, int r=0x00, int g=0x00, int b=0x00, int a=0xFF\)](#)
- [Hell\\_drawTriangle\(float x, float y, float radius, float rot=0, int width=1, int r=0x00, int g=0x00, int b=0x00, int a=0xFF\)](#)
- [Hell\\_drawCircle\(float cx, float cy, float radius, int width=1, int r=0x00, int g=0x00, int b=0x00, int a=0xff\)](#)
- [Hell\\_drawFont\(char\[\] msg, float x, float y, float zoom=1.0f, int r=0xff, int g=0xff, int b=0xff, int a=0xff\)](#)
- [Hell\\_drawFontEx\(float x, float y, float zoom=1.0f, int r=0xff, int g=0xff, int b=0xff, int a=0xff, ...\)](#)
- [Hell\\_sin\(float rad\)](#)
- [Hell\\_cos\(float rad\)](#)
- [Hell\\_atan2\(float y, float x\)](#)
- [Hell\\_sinEx\(float deg\)](#)
- [Hell\\_cosEx\(float deg\)](#)
- [Hell\\_atan2Ex\(float y, float x\)](#)
- [Hell\\_deg2rad\(float deg\)](#)
- [Hell\\_rad2deg\(float rad\)](#)
- [Hell\\_randSeed\(uint seed, uint index\)](#)
- [Hell\\_rand\(uint range\)](#)
- [Hell\\_randf\(float range\)](#)
- [Hell\\_randInt\(int begin, int end\)](#)
- [Hell\\_choice\(int\[\] array\)](#)
- [Hell\\_shuffle\(int\[\] array\)](#)
- [Hell\\_MessageBox\(wchar\[\] message, wchar\[\] caption="", uint type=MB\\_OK\)](#)

## ダウンロード

[ここ](#)からライブラリー式をダウンロードします。

## 更新履歴

- 2010.01.21 -- 1.0.8 色々更新
  - OpenGLのportingを更新にして、最新のDコンパイラでもコンパイルを通るようにした
  - キーユーティリティを追加 ( Hell\_isPressEnter,Hell\_isPushEnter,Hell\_isPressCancelなど )
  - BGMが再生中かどうか調べる「Hell\_isPlayBgm」を追加
  - 線分の描画座標の指定がおかしかった不具合を修正
  - drawTexture()/drawTextureEx()で 値が指定できなかったのも、引数を追加した
  - 現在の描画内容をキャプチャーする「Hell\_capture()」を追加
  - 乱数周りのユーティリティ「Hell\_rand()」「Hell\_randf()」「Hell\_choice()」「Hell\_shuffle()」を追加
  - メッセージボックス表示「Hell\_MessageBox()」を追加
- 2007.07.25 -- 1.0.7 drawTexture()/drawTextureEx()でテクスチャを切り取ったときに描画がおかしくなる不具合を修正。
- 2007.03.07 -- 1.0.6 loadWAV()で既に存在するキーを指定した場合、既存のWAVを破棄して読み込むように修正。
- 2007.02.22 -- 1.0.5 キャプション文字列の変更関数Hell\_setCaption()を追加。数学・乱数関連の関数を追加。
- 2007.02.21 -- 1.0.4 Hell\_write()/Hell\_drawFontEx()でprintf()っぽい書式を使えるように
- 2007.02.20 -- 1.0.3 SDL\_FreeSurface()でSurfaceをちゃんと破棄するように修正。F9キーでウィンドウ・フルスクリーン切替。F12でスクリーンショットをBMPで保存。
- 2007.02.19 -- 1.0.2 loadTexture()で既に存在するキーを指定した場合、既存のテクスチャを破棄して読み込むように修正。「F4」キーで全てのテクスチャを読み込みなおす処理を追加。
- 2007.02.16 -- 1.0.1 円描画命令「Hell\_drawCircle()」の追加
- 2007.02.13 -- 1.0.0 公開開始

## 既知の問題

- 画像ファイルはBMPで、2^nのサイズのみとなっています。
- 60枚くらいテクスチャを読み込みウィンドウを最小化・元に戻すと、CPU使用率が100%近くになってしまう。

## dHellであった機能

dHellではこんな感じでした。

- 画像読込・描画 ( BMPのみ )
- BGM読込・再生 ( WAV/OGGのみ )
- 効果音読込・再生 ( WAVのみ )
- キーイベント ( マウス・キーボード・ジョイスティック ) のハンドリング
- フォントの描画 ( asciiのみ )

## dHell2の新機能

こんな機能が追加されました。

- 画像の拡大縮小・回転 ( Hell\_drawTexture/Hell\_drawTextureEx )
- 画像の加算合成 ( Hell\_setAlpha )
- 線分/三角形の描画・回転
- 円の描画
- それぞれの描画をRGBAを指定して描画できるように

( ようやく、普通のゲームライブラリっぽくなった……? )

## Dコンパイラのインストールなど

「D言語ゲーム開発入門」

<http://www5.atwiki.jp/yaruhara/pages/74.html>

などを参考にインストールします。

## ビルド方法

build.batをテキストファイルで開いて、  
D言語コンパイラ・リンクのパスを設定します。  
( パスを通してあるのであれば、必要ありません )

で、build.batを実行します。

## 使い方

「test.d」を編集していきます。  
なんか色々ありますが、最低限必要なコードはこれだけです。

```
import hell;

void main()
{
    try
    {
        // Hellシステム初期化 ( キャプション "Hell" 640x480 )
        Hell_init("Hell", 640, 480);
        // ゲームループ開始
        while(true)
        {
            // 描画エリアの消去
            Hell_begin();
            // 画面更新
            Hell_update();
            // 33ms待つ
            Hell_wait(33);
        }
    }
    catch(Exception e)
    {
```

```

        // 例外をログに書き込み
        Hell_write(e);
    }
    finally
    {
        // Hellシステム終了
        Hell_quit();
    }
}

```

注意が必要なのは、「try ~ catch ~ finally」です。  
 ウィンドウが閉じられたときには、Hell\_wait()が例外を返します。  
 その場合、catchして、ログの出力をし、Hell\_quit()で終了します。

画像ファイルが存在しなかった場合など、「run.log」にエラーが書き込まれるので、それを参照します。

## 特殊なキーの動作

- F4キー：登録しているテクスチャを全て読み込みし直す
- F5キー：ボスが来た！ ウィンドウ最小化
- F9キー：ウィンドウモード・フルスクリーン切替（たぶん使えます）
- F10キー：マウスカーソル表示切替
- F12キー：スクリーンショットをhell.bmpで保存（たぶん使えます）
- ESCキー：アプリケーションを終了します。

## 「dHell2」リファレンス

**Hell\_init(char[] caption, int width, int height, bool fullscreen=false)**

dHellを初期化します。

- パラメータ
  - caption ウィンドウタイトル文字列
  - width 画面縦サイズ
  - height 画面横サイズ
  - fullscreen フルスクリーンフラグ

**Hell\_quit()**

dHellの終了処理を行います。

**Hell\_setCaption(...)**

キャプション（タイトルバー）の文字列を変更します。  
 （ただし日本語は表示できません）  
 C言語のprintf()のような書式で書くことができます。

## Hell\_write(...)

ログファイル「run.log」にメッセージを書き込みます。  
C言語のprintf()のような書式で書くことができます。  
より詳しい記述方法は、std.format.doFormat  
[http://www.kmonos.net/along/d/phobos/std\\_format.html](http://www.kmonos.net/along/d/phobos/std_format.html)  
を参考に。

## Hell\_write(Exception e)

ログファイル「run.log」に例外情報の書き込みを行います。

## Hell\_loadTexture(char[] key, char[] filepath, int[3] mask=[-1, 0, 0])

BMPをテクスチャとして読みこみます。

### ・パラメータ

- key 画像のキー（Hell\_drawTextureで使います）
- filepath 画像ファイルのパス
- mask 抜き色（mask[0]に「-1」で抜き色なし。「-2」で座標指定）

## Hell\_drawTexture(char[] key, float cx, float cy, int ox=0, int oy=0, int ow=0, int oh=0, float dx=1.0f, float dy=1.0f, float rot=0, int r=0xFF, int g=0xFF, int b=0xFF, int a=0xFF)

BMPを描画します。

ox,oy,ow,ohにそれぞれ「0」を指定すると、BMP全体を描画します。

抜き色を有効にする場合、Hell\_setAlpha()に「HELL\_ALPHA\_NORMAL/HELL\_ALPHA\_ADD」を指定する必要があります。

### ・パラメータ

- key キー（Hell\_loadTextureで読み込み済みのもの）
- x X座標
- y Y座標
- ox 切り取り開始X座標
- oy 切り取り開始Y座標
- ow 切り取る幅
- oh 切り取る高さ
- dx 拡大サイズ（X）
- dy 拡大サイズ（Y）
- rot 回転角度（0～360, 左回り）
- r マスク色(赤)
- g マスク色(緑)
- b マスク色(青)
- a 値

## Hell\_drawTextureEx(char[] key, float cx, float cy, int ox=0, int oy=0, int ow=0, int oh=0, float dx=1.0f, float dy=1.0f, float rot=0, int r=0xFF, int g=0xFF, int b=0xFF, int a=0xFF)

Hell\_drawTexture()は左上座標を指定ですが、これは中心座標を指定して描画する関数です。

## Hell\_hasTexture(char[] key)

指定のキーのテクスチャが存在するかどうかチェックします

- パラメータ
  - key 画像のキー
- 戻り値
  - キーが存在すれば「true」を返します

## Hell\_capture(char[] key)

現在の画面の内容をキャプチャーします。例えば、前の描画内容を半透明で消しながら重ね合わせることでクロスフェードっぽいことができるようになります。

ただし、重たい処理なので毎フレームキャプチャーするのはオススメできません。

## Hell\_setAlpha(char[] key, int alpha)

アルファ値を設定します。

- パラメータ
  - alpha アルファ値 (HELL\_ALPHA\_DISABLE/HELL\_ALPHA\_NORMAL/HELL\_ALPHA\_ADD)
  - HELL\_ALPHA\_DISABLE : ブレンドしない
  - HELL\_ALPHA\_NORMAL : 半透明
  - HELL\_ALPHA\_ADD : 加算

## Hell\_getMouseX

マウスのX座標を取得します。

- 戻り値 --- マウス座標X

## Hell\_getMouseY

マウスのY座標を取得します。

- 戻り値 --- マウス座標Y

## Hell\_isPressMouse

マウスを押し続けているかどうかを判定するフラグを取得します。

- 戻り値 --- マウスを押し続けているかどうかのフラグ

### 使い方

```
if(Hell_isPressMouse() & HELL_BUTTON_LEFT)
{
    printf("press left");
}
if(Hell_isPressMouse() & HELL_BUTTON_MIDDLE)
{
    printf("press middle");
}
if(Hell_isPressMouse() & HELL_BUTTON_RIGHT)
{
    printf("press right");
}
```

## Hell\_isPushMouse

マウスをそのフレームに押したかどうかを判定するフラグを取得します。

- 戻り値 --- マウスをそのフレームに押したかどうかのフラグ

### 使い方

```
if(Hell_isPushMouse() & HELL_BUTTON_LEFT)
{
    printf("push left");
}
if(Hell_isPushMouse() & HELL_BUTTON_MIDDLE)
{
    printf("push middle");
}
if(Hell_isPushMouse() & HELL_BUTTON_RIGHT)
{
    printf("push right");
}
```

## Hell\_isPressKey

キーを押し続けているかどうかを判定するフラグを取得します。

- 戻り値 --- キーを押し続けているかどうかのフラグ

### 使い方

```
if(Hell_isPressMouse(HELL_DOWN))
{
    printf("press down");
}
if(Hell_isPressMouse(HELL_RIGHT))
{
```

```
        printf("press right");
    }
    if(Hell_isPressMouse(HELL_LEFT))
    {
        printf("press left");
    }
}
```

### **Hell\_isPushKey**

キーをそのフレームに押したかどうかを判定するフラグを取得します。

- 戻り値 --- キーをそのフレームに押したかどうかのフラグ

使い方

```
if(Hell_isPushKey(HELL_RETURN))
{
    printf("push return");
}
if(Hell_isPushKey(SDLK_z))
{
    printf("push z");
}
if(Hell_isPushKey(HELL_UP))
{
    printf("push up");
}
```

### **Hell\_loadWAV(char[] key, char[] path)**

SE ( WAVファイル ) を読み込みます

- パラメータ
  - key キー ( Hell\_playWAVで使います )
  - path サウンドファイルのパス

### **Hell\_playWAV(char[] key, int loops=0, int channel=-1)**

SE ( WAVファイル ) を再生します

- パラメータ
  - key キー
  - loops ループ回数。0で1回再生。(-1で無限ループ)
  - channel チャンネル番号(-1であいているチャンネルを自動で使う)

## **Hell\_stopWAV()**

SE (WAVファイル) の再生を全て停止します。

- パラメータ
  - channel 停止チャンネル番号(-1で全て停止)

## **Hell\_disposeWAV(char[] key=null)**

SE (WAVファイル) を破棄します。

- パラメータ
  - key キー (Hell\_loadWAVで指定したもの) nullで全てを破棄します

## **bool Hell\_isPressJKey(int id)**

ジョイスティックの十字キーを押しているかどうかを調べます。

- 戻り値---ジョイスティックの十字キーを押しているかどうか

使い方

```
if(Hell_isPressJKey(HELL_J_UP))
{
    printf("press up");
}
if(Hell_isPressJKey(HELL_J_DOWN))
{
    printf("press down");
}
if(Hell_isPressJKey(HELL_J_LEFT))
{
    printf("press left");
}
if(Hell_isPressJKey(HELL_J_RIGHT))
{
    printf("press right");
}
```

## **bool Hell\_isPushJKey(int id)**

ジョイスティックの十字キーを押したかどうかを調べます。

- 戻り値---ジョイスティックの十字キーを押したかどうか

使い方

```
if(Hell_isPushJKey(HELL_J_UP))
{
    printf("push up");
}
```

```
if(Hell_isPushJKey(HELL_J_DOWN))
{
    printf("push down");
}
if(Hell_isPushJKey(HELL_J_LEFT))
{
    printf("push left");
}
if(Hell_isPushJKey(HELL_J_RIGHT))
{
    printf("push right");
}
```

### **bool Hell\_isPressJButton(int id)**

ジョイスティックのボタンを押しているかどうかを調べます。

- 戻り値---ジョイスティックのボタンを押しているかどうか

#### 使い方

```
if(Hell_isPressJButton(0))
{
    printf("press 1button");
}
if(Hell_isPressJButton(1))
{
    printf("press 2button");
}
if(Hell_isPressJButton(2))
{
    printf("press 3button");
}
if(Hell_isPressJButton(3))
{
    printf("press 4button");
}
```

### **bool Hell\_isPushJKey(int id)**

ジョイスティックのボタンを押したかどうかを調べます。

- 戻り値---ジョイスティックのボタンを押したかどうか

#### 使い方

```
if(Hell_isPushJButton(0))
{
    printf("push 1button");
}
if(Hell_isPushJButton(1))
{
    printf("push 2button");
}
if(Hell_isPushJButton(2))
{
    printf("push 3button");
}
```

```
}  
if(Hell_isPushJButton(3))  
{  
    printf("push 4button");  
}
```

#### **Hell\_isPressEnter()/Hell\_isPushEnter()**

Zキー、Returnキー、Spaceキー、Joystickの1ボタン、マウスの左クリックのいずれかが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_isPressCancel()/Hell\_isPushCancel()**

Xキー、Joystickの2ボタン、マウスの右クリックのいずれかが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_isPressMenu()/Hell\_isPushMenu()**

Cキー、Joystickの3ボタン、マウスの真ん中クリックのいずれかが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_isPressUp()/Hell\_isPushUp()**

上キーが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_isPressLeft()/Hell\_isPushLeft()**

左キーが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_isPressDown()/Hell\_isPushDown()**

下キーが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_isPressRight()/Hell\_isPushRight()**

右キーが押されていれば/そのフレームに押したら「true」を返す

#### **Hell\_playBgm(char[] path, int loops=-1, int ms=0)**

BGMを再生します

- パラメータ
  - path ファイルパス
  - loops ループ回数 (-1で無限ループ)
  - フェードイン時間 (ms)

## **Hell\_stopBgm(int ms=0)**

BGMを停止します

- パラメータ
  - フェードアウト時間 (ms)

## **Hell\_isPlayBgm()**

BGMが再生中かどうか

- 戻り値
  - 再生中であれば「true」

## **Hell\_drawRect(float x=0, float y=0, float dx=0, float dy=0, float width=0, float rot=0, int r=0x00, int g=0x00, int b=0x00, int a=0xFF)**

矩形の描画

- パラメータ
  - x,y 左上座標
  - dx,dy 幅,高さ
  - width 線分の太さ(0で塗りつぶし)
  - r,g,b,a RGBA値

## **Hell\_drawRectEx(float cx=0, float cy=0, float dx=0, float dy=0, float width=0, float rot=0, int r=0x00, int g=0x00, int b=0x00, int a=0xFF)**

矩形の描画 (中心座標を指定)

- パラメータ
  - cx,cy 中心の座標
  - それ以外はHell\_drawRectと同じ

## **void Hell\_drawLine(float x1, float y1, float x2, float y2, int width=1, int r=0x00, int g=0x00, int b=0x00, int a=0xFF)**

線分の描画

- パラメータ
  - x1,y1 開始座標
  - x2,y2 終端座標
  - width 線の太さ

r,g,b,a RGBA値

。

`Hell_drawTriangle(float x, float y, float radius, float rot=0, int width=1, int r=0x00, int g=0x00, int b=0x00, int a=0xFF)`

三角形を描画します。

・ パラメータ

- x X座標 (中心)
- y Y座標 (中心)
- radius 半径
- rot 回転角度 (0 ~ 360)
- width 線の太さ (0で塗りつぶし)
- r R成分
- g G成分
- b B成分
- a A成分

`Hell_drawCircle(float cx, float cy, float radius, int width=1, int r=0x00, int g=0x00, int b=0x00, int a=0xff)`

円を描画します。

・ パラメータ

- cx 中心座標X
- cy 中心座標Y
- radius 半径
- width 線の太さ (0で塗りつぶし)
- r R成分
- g G成分
- b B成分
- a A成分

`Hell_drawFont(char[] msg, float x, float y, float zoom=1.0f, int r=0xff, int g=0xff, int b=0xff, int a=0xff)`

フォントの描画 (asciiのみです)

・ パラメータ

- msg ascii文字列
- x,y 描画開始位置
- zoom 拡大サイズ (1.0fで32x48)
- r R成分
- g G成分
- b B成分
- a A成分

**Hell\_drawFontEx(float x, float y, float zoom=1.0f, int r=0xff, int g=0xff, int b=0xff, int a=0xff, ...)**

Hell\_drawFont()をC言語のprintf()のような書式で書くことができます。  
より詳しい記述方法は、std.format.doFormat  
[http://www.kmonos.net/alang/d/phobos/std\\_format.html](http://www.kmonos.net/alang/d/phobos/std_format.html)  
を参考に。

**Hell\_sin(float rad)**

サインを求めます。パラメータはラジアンです。

**Hell\_cos(float rad)**

コサインを求めます。パラメータはラジアンです。

**Hell\_atan2(float y, float x)**

アークタンジェントを求めます。戻り値はラジアンです。

**Hell\_sinEx(float deg)**

サインを求めます。パラメータは度を指定します。

**Hell\_cosEx(float deg)**

サインを求めます。パラメータは度を指定します。

**Hell\_atan2Ex(float y, float x)**

アークタンジェントを求めます。戻り値は度になります。

**Hell\_deg2rad(float deg)**

度をラジアンに変換します。

**Hell\_rad2deg(float rad)**

ラジアンを度に変換します。

**Hell\_randSeed(uint seed, uint index)**

乱数を初期化します。  
わざわざ呼ばなくてもプログラム開始時に乱数は初期化されます。  
再現性のある乱数を生成したい場合にコールします。

- パラメータ
  - seed 種
  - index 次の乱数インデックスへの増分

### **Hell\_rand(uint range)**

0 ~ (range-1)の範囲で乱数を生成する

### **Hell\_randf(float range)**

0 ~ rangeの範囲で乱数を生成する (float版)

### **Hell\_randInt(int begin, int end)**

begin <= x < endの範囲で乱数を生成します。

- パラメータ
  - begin 開始値
  - end 終了値

### **Hell\_choice(int[] array)**

指定の配列からランダムで値を取り出す

- パラメータ
  - 配列
- 戻り値
  - 配列の中のランダムな値

### **Hell\_shuffle(int[] array)**

指定の配列の値をランダムに入れ替える

- パラメータ
  - 配列

### **Hell\_MessageBox(wchar[] message, wchar[] caption="", uint type=MB\_OK)**

Windowsダイアログを表示する

- パラメータ
  - message メッセージ文字列
  - caption キャプション文字列
  - type メッセージボックスの種類