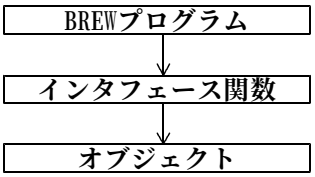
BREW & EZアプリ
ゲームプログラミング

○　第５回　グラフィックの描画　○

インタフェース

BREW APIは、BREW実行環境の機能を呼び出すための関数群ですが、その実体は、「オブジェクト」に作用する「インタフェース関数」です。
オブジェクトやインタフェースといった用語は、オブジェクト指向で用いるものですが、BREWのそれらは、若干意味が異なります。
BREWのオブジェクトとは、BREW APIが内部で使うデータのことをいいます。そして、そのデータを操作するのがインタフェース関数です。インタフェース関数の１つめの引数は、必ず操作対象となるオブジェクトを渡すようになっています。BREWでは、オブジェクトとインタフェース関数をまとめたものをインタフェースと呼びます。



たとえば、前回の課題で使用したISHELL_SetTimer関数は、IShellインタフェースのSetTimerという機能を呼び出しています。
IShellインタフェースは、BREW実行環境を操作するためのインタフェースです。このインタフェースには、BREW実行環境を管理するためのデータと、その機能を呼び出すための多数のインタフェース関数が備えられています。ISHELL_SetTimer関数は、１度だけ起動するタイマを設定するという動作をとります。つまり、この関数はBREW実行環境のタイマを設定している、ということになります。

```
ISHELL_SetTimer(pMe->pIShell, 引数…);
インタフェース関数  オブジェクト
```

IDisplayインタフェースとIGraphicsインタフェース

画面に描画を行うためのインタフェースとしてIDisplayインタフェースとIGraphicsインタフェースが用意されています。２つのインタフェースは、以下のように機能が異なります。

ｲﾝﾀﾌｪｰｽ	画面のクリア	ﾌﾚｰﾑの更新	ﾋﾞｯﾄﾏｯﾌﾟの転送	塗りつぶし	文字描画	点の描画	線の描画	矩形描画	三角形の描画	多角形の描画	円の描画	色の反転	ﾊﾞｯｸﾗｲﾄの制御
IDisplay	◎	◎	○(等倍)	○	○	×	△	○	×	×	×	○	○
IGraphics	○	○	○(拡縮)	○	×	○	○	◎	○	○	○	×	×

また、同じような動作をするにもかかわらず、関数名や性能が異なったり、制限のある場合もあるので、ドキュメントやシミュレータ(実機)で動作を確認してからどちらを使うか決定する必要があります。

インタフェースの生成と解放

IShellインタフェースとIDisplayインタフェースは、どのようなアプリケーションでもほぼ確実に使用するため、アプリケーションの起動時に、AEEClsCreateInstance関数で自動的に生成されます。
そのほかのインタフェースは、IShellインタフェースのISHELL_CreateInstance関数で生成する必要があります。引数にインタフェースのクラスIDを渡すと、そのインタフェースがメモリに生成され、オブジェクトが返されます。たとえば、IGraphicsインタフェースを生成するには、クラスID「AEECLSID_GRAPHICS」を渡します。

```
// IGrapicsインタフェースの生成
IGraphics  *pIGraphics;
ISHELL_CreateInstance(pMe->pIShell, AEECLSID_GRAPHICS, &pIGraphics);
```

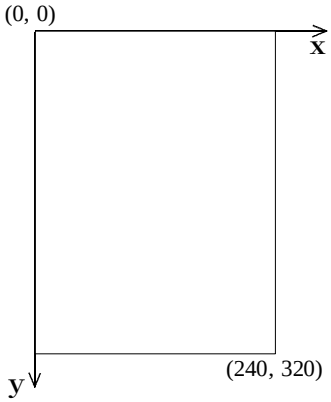
関数が成功すると、メモリにBREWの描画機能にアクセスするためのIGraphicsインタフェースが生成され、そのオブジェクトがpIGraphics変数に格納されます。以後、IGRAPHICS_で始まるインタフェース関数を呼び出せば、描画に関する機能を使用できます。

ISHELL_CreateInstance関数で生成したインタフェースが不用になったら、そのインタフェース関数のRelease関数を呼び出し、メモリから解放しなければなりません。

```
// IGraphicsインタフェースの解放
IGRAPHICS_Release(pIGraphics);
```

スクリーン座標

2Dでグラフィックを描画する場合、座標の指定はスクリーン座標で行います。スクリーン座標とは、画面のピクセルに１対１に対応する座標系のことです。スクリーン座標の原点は、左上にあります。また、ｙ軸の向きが逆になっていることも大きな特徴です。



QVGA対応端末では、２４０×３２０ドットの解像度を持ちますが、上部にバッテリー残量やアンテナ感度を表示するため、実際には、A5501Tで２４０×２９８ドット、A5503SAで２４０×２９１ドットとなります。

課　題

画面に図形を描画してみましょう。

(１)図形は、IGraphicsインタフェースで描画します。このインタフェースを用いるには、専用のヘッダファイルが必要になります。以下のプログラムを適切な場所に追加しましょう。

```
#include "AEEGraphics.h"    // Graphics interface definitions
```

(２)IGraphicsインタフェースを格納する変数をアプレット構造体に追加します。以下のプログラムを適切な場所に追加しましょう。

```
IGraphics   *pIGraphics; // give a standard way to access the Graphics interface
```

ヒント：// add your own variables here...

(３)プログラム開始時に、IGraphicsインタフェースを取得します。以下のプログラムを適切な場所に追加しましょう。なお、「？」は各自考えてください。

```
// IGraphicsインタフェース生成
ISHELL_?????????????(pMe->pIShell, AEECLSID_????????, &pMe->pIGraphics);
 if(pMe->pIGraphics == NULL)
      return FALSE;
```

IGraphicsインタフェースを生成し、そのオブジェクトをアプレット構造体のpIGraphicsメンバに格納します。以後、「pMe->pIGraphics」と記述することでIGraphicsオブジェクトを得ることができます。

ヒント：追加する場所は、第３回のどこかに記述されています

(４)プログラム終了時に、IGraphicsインタフェースを解放します。以下のプログラムを適切な場所に追加しましょう。なお、「？」は各自考えてください。

```
// IGraphicsインタフェース解放
if(pMe->pIGraphics != NULL) {
    IGRAPHICS_???????(pMe->pIGraphics);
    pMe->pIGraphics = NULL;
}
```

ヒント：// insert your code here for freeing any resources you have allocated...

(５)画面を更新する処理を追加します。以下のプログラムを*プロジェクト名*_Draw関数の最後に追加しましょう。

```
// 画面更新
IDISPLAY_Update(pMe->pIDisplay);
```

IDisplayインタフェースまたは IGraphicsインタフェースのUpdate関数を実行すると、前回のUpdate関数呼び出し以後に行ったすべての描画命令の結果が、画面に表示されます。よって、この関数は、フレームのすべての描画が終わったあとに呼び出します。

(６)点を描画してみます。以下のプログラムを適切な場所に追加しましょう。なお、「？」は各自考えてください。

```
AEEPoint   pt;    // 点の情報を格納する構造体変数

// 点の描画
pt.x = 120;       // ｘ座標
pt.y = 160;       // ｙ座標
IGRAPHICS_?????????(pMe->pIGraphics, &pt);
```

(７)描画される図形の色を設定しましょう。以下のプログラムを適切な場所に追加しましょう。なお、「？」は各自考えてください。

```
// 色の設定
IGRAPHICS_????????(pMe->pIGraphics, 255, 0, 0, 255);  // 左からRGBAの順
```

(８)画面に直線を描画してみます。以下のプログラムを適切な場所に追加しましょう。なお、「？」は各自考えてください。

－追加１－

```
AEELine   line;   // 線の情報を格納する構造体変数
```

－追加２－

```
// 線の描画
line.sx = 0;      // 開始点ｘ座標
line.sy = 0;      // 開始点ｙ座標
line.ex = 240;    // 終了点ｘ座標
line.ey = 298;    // 終了点ｙ座標
IGRAPHICS_DrawLine(pMe->pIGraphics, &line);
```

(９)画面に矩形を描画しましょう。

(10)画面に三角形を描画しましょう。

(11)画面に円を描画しましょう。

(12)矩形を塗りつぶしましょう。

(13)円を塗りつぶしましょう。

(14)塗りつぶす色を変更しましょう。

応用問題　画面に適当な図形を描画し、画面を往復するように動かしてみましょう。