愛知教育大学研究報告，56（芸術・保健体育・家政・技術科学編），pp. 35～41，March，2007

ロータリーエンジンのシミュレーション プログラムの開発

大　西　研　治*
Kenji Ohnishi
*技術教育講座

1．緒　　言

本研究は中学校技術の内熱機関に関する授業の必要性を考え，リアル４サイクルエンジンに引き続きロータリーエンジンのシミュレーションプログラムを製作することとした。これらのシミュレーションを利用することにより，内熱機関のしくみを短時間で効果的にかつ視覚的に学習できるものと考える。

2．プログラム開発にあたり考慮した事項

プログラムの作成に際して，実物のロータリーエンジンを分解し，ハウジングのカーブの具合からエピトロコイド曲線の係数，ローターの円弧部の曲率などを求め，プログラムの作成に反映した。

2.1　基本構成

・ディスプレイの解像度を上げるほど標準画面は小さくなるので，ディスプレイの解像度に関係なくシミュレーション描画ができるように，Form の大きさを画面の高さ比で設定する標準モジュールを用いる。本研究では画面の高さの80%とする。
・オブジェクトの指定領域内を指定した色で塗りつぶす標準モジュールを用いる。

2.2　プログラムの記述の書式

プログラムの記述の書式として，以下の点を考慮して記述した。

・グローバル変数は General ですべて変数型を宣言する。
・グローバル変数は変数名の頭に変数の型（Int，Lng，…，ただし，Single は除く）を付与する。
・本プログラムの定数は Form でその値を設定する。
・プログラムの手続きが理解しやすいよう手続きの説明をレム文で記述する。
・各プロシージャに，関係した Global の変数と，各プロシージャだけで使用する Local 変数をレム文で記述する。次にプロシージャで使用する Local 変数の型を定義する。
・Pictures と Command などは Index を用いて記述する。
・フォームモジュールのプロパティや，From 画面，Picture 画面，command および Label などの種々の設定は，From に記述し，プログラムを理解しやすくする。
・エンジンのパーツごとにサブルーチンプロシージャを作り，引数は描画オブジェクト名，描画座標と色などとし，パーツ表示などにも応用できるようにする。
・一周目と二周目以降では吸気ガス排気ガスの様子が異なるので，IntPaintFlag で場合分けをする。

2.3　付加する機能

・タイミングダイアグラムを表示する。
・シミュレーションの表示切り替え速度を制御する。
・シミュレーションのコマ送りができるようにする。

2.4　エンジンの描画

エンジン描画において実現する点を以下に述べる。

・混合気・燃焼ガスの色を行程別に変化をつける。
・点火時を解りやすくするため，点火時にプラグ先端部の周りを赤く塗る。
・ローターとハウジングの隙間による色抜けを防止するためにローターの頂点より放線上に短線を引く。
・ステーショナリーギアとインターナルギアを解りやすくするために色で分け，歯の部分を太い一点鎖線で描画する。
・吸気矢印は吸気口の広さに対応して，矢印の大きさや向きなどを設定する。
・燃焼室を分ける３つのアペックスシールを色の異なる小さな円で描画する。
・吸気口について，ローターと重なって塞がっているときは吸気口内をローターの色で塗りラインを破線で描画する。
・ローターの基準線とエキセントリックシャフト回転基準線を描画する。
・ローターの円弧部とハウジングの両プラグ部の間隙を，円弧部の曲率を大きくし実物から測定した間隙より大きくし見やすくする。さらに，ローターの円弧中央部にある３つの燃焼室部の描画を省略する。

2.5　タイミングダイアグラムの描画

タイミングダイアグラムの描画において実現する点を以下に述べる。

・行程の変化を示すため２重円を用い，行程ごとにその内部を各行程のエンジン内のガスの色に合わせて色を塗る。

・エンジンの現在の状態を示すため，マーカを内円に沿って移動させる。
・点火時期であることを知らせるため，ダイヤグラムの外円の外側３カ所に大小の三角形の点火目印を描き，プラグ点火時には赤く塗る。
・各行程名を表記し，行程の変化を解りやすくする。

2.6　表示画面の構成

図１に示すように，form 画面の最上部に制御コマンドを配置し，残りを Picture 画面とした。また，Picture 画面を３つの部分に分け，左上部にプログラム名を，左下部にエンジン・タイミングダイアグラムを，右側にエンジン･シミュレーションを描画することとした。

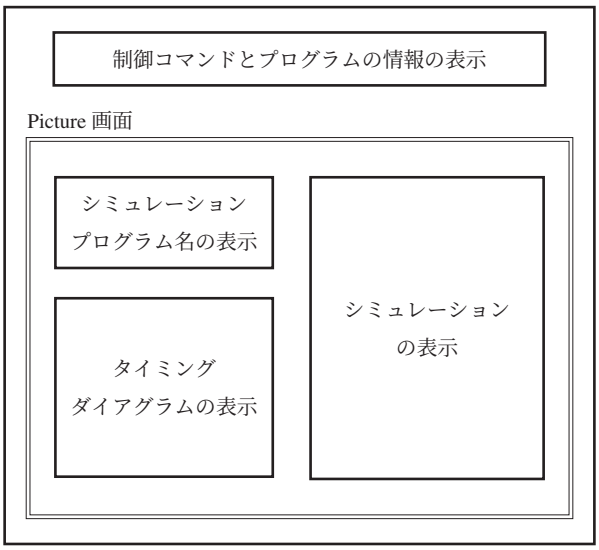

図１　Form 画面と Picture 画面の配置

３．プログラムの作成

本プログラムは，２つの標準モジュールと１つのフォームモジュールから成る。

3.1　標準モジュール

a．フォームのクライアント領域のサイズを画面の高さのサイズ比より設定するモジュール

ディスプレイの解像度により Form の大きさが変化しないように，ディスプレイ画面の高さ比で Form の大きさを設定する標準モジュールを用いた。本プログラムでは，解像度の違いによる円弧の描画のずれをなくすため，Form 画面の縦横比を1.283とし，多くのディスプレイで確認したところ色漏れの発生は無かった。

b．画面の指定領域内をブラシで塗りつぶす標準モジュール

サブプロシージャを作る方法もあり，多くの書籍にも掲載されているが，今後のプログラム開発のツールとするため標準モジュールにしたものを用いた。さらに，前述の標準モジュールを使用するため，色塗り時に生じる座標のずれを補正する ScaleX と ScaleY の行を組み入れている。

標準モジュールは，画面の領域を現在のブラシで塗りつぶす API 関数 FloodFill（又は ExtFloodFill）の宣言と指定されたオブジェクトの領域の指定された色の境界内を塗りつぶす FuncPaintObject 関数の定義がなされている。

3.2　フォームモジュールの基本構成

General，From，Command0，Label，Pictures，Timer の６つのオブジェクトと座標計算，パーツ描画用などの14のサブプロシージャからなる。

3.3　作成する主なサブプロシージャ

3.3.1　エピトロコイド曲線・ローター円弧の計算用プロシージャ

エピトロコイド曲線描画のための Sub Toro とローター円弧の描画に必要な座標を計算する Sub StEndRad からなる。

3.3.2　エンジン本体パーツ等の描画用プロシージャ

ハウジング，エンジン本体の両足，点火プラグ，吸気口と排気口を描いている。

3.3.3　タイミングダイアグラム描画用プロシージャ

Sub DaiyaGuramuDisp でダイヤグラムを描画して，Sub DaiyaGuramuPointDisp １で行程の目印を描画している。また，Sub DaiyaGuramuSparkPointPaint で点火目印の色塗りの座標を計算している。

3.3.4　矢印描画用プロシージャ

矢印の軸元の座標，矢印の太さ，長さ，傾きを指定し，任意の大きさ・傾きの矢印を描画する。

3.4　プログラムの手続きの流れ

２つの標準モジュール，General，Form オブジェクトと Timer オブジェクトプログラムの手続きの概略を以下に示す。

3.4.1　標準モジュール

a. Function FuncSetFormSize は，

1. ローカルな変数の宣言
2. 非クライアント領域の幅，高さを求める（Twip）
3. クライアント領域を求める（Twip）
4. クライアント領域を TwipsPerPic の倍数に設定する（Twip）
5. フォームサイズを設定する

以上の５つの手続きで構成されている。

b. FuncPaintObjec は，

0. API 関数 FloodFill の宣言
1. FuncPaintObject 関数の定義
2. ローカル変数の宣言
3. デバイスコンテキストのハンドルを取得
4. 塗りつぶしのスタイルを設定
5. 座標を Pixel 単位に変更

6. FloodFill 関数を呼び出して塗りつぶしを実行
7. 塗りつぶしのスタイルを設定解除
8. 再描画を実行

以上の８つの手続きと API 関数の宣言とで構成されている。

3.4.2　General の書式

Form モジュールの General には，変数名とそのデータ型の設定を記述する。

・エピトロコイド曲線やローター円弧センターなどの座標に関わる変数名と型の設定
・シミュレーションの表示スピードや描画回数の変数名と型の設定
・ペインティング用の色名の変数名と型の設定
・ロータリーエンジンの中心座標やローターを描くために必要な変数名と型の設定

3.4.3　フォームオブジェクトの構成

フォームオブジェクトに記述する手続きを以下に示す。

1. プログラムに用いる変数の値の設定
2. 標準モジュール FuncSetFormSize 関数の呼び出し，表示画面の大きさを設定
3. 各オブジェクトのバックカラーを同一色にするために LngBcolor を設定
4. Form に関わる設定
   Form の座標をユーザ定義で設定
   Form を Screen の中央に表示
   Form の BackColor を LngBcolor に設定
5. プログラムの多重起動の防止
6. Command ０オブジェクトの設定
   　オブジェクトの幅と高さ，配置する XY 座標，描画線の太さ
   コマンドに表記する文字の設定
7. Pictures オブジェクトの設定
   　Pictures オブジェクトの幅と高さと左上の XY 座標，描画線の太さ。Pictures オブジェクトが重なっても表示するため再描写の設定
8. Label オブジェクトの設定
9. 色の指定
10. ロータリーやダイヤグラムの大きさや中心座標などの値の設定
11. ハウジング等を Pictures（０），（１）に描画
12. ダイヤグラムを Pictures（０），（１）に描画
13. Pictures（０）を表示後，停止

3.4.4　タイマーオブジェクトの構成

タイマーオブジェクトに記述する手続きを以下に示す。

1. 描画回数の奇数と偶数により，Picture オブジェクトを切り替え
2. ローターの回転後の座標の計算
   ローター頂点を計算
   色抜け防止座標を計算
   混合気・燃焼ガスの Paint ポイント座標を計算
3. ローターの描画
   ローター外基準の計算
   ローター円弧のセンターを計算
   ローター円弧のセンターから各円弧の半径の設定
   各円弧の描画範囲の計算
   ローター頂点の描画
   ローターの曲線の描画
4. 混合気・燃焼ガスの色抜け防止のためラインを描く
5. Case 文を用い描画回数の進行に合わせ，吸気・排気ガスの色塗りの制御
6. Case 文を用い描画回数の進行に合わせ，2つのプラグの点火とダイヤグラム点火目印の描画の制御
7. ダイヤグラムの動く点のサブプロシージャを呼び出し
8. 回転角などの文字の表記を呼び出し
9. ローターの曲線と頂点を LngSitagaki で塗り直し
10. インターナルギアとステーショナリーギアの描画と色塗り
11. ローターの曲線と頂点を指定した描き直し
12. ローターの基準線，エキセントリックシャフト回転基準線，インターナルギヤの基準点を表示する円とローター中心の描画
13. 吸気口の描画
14. 吸気・排気の矢印の描画
15. 表示されていない Picture の制御
16. 描画回数の表示と制御
17. コマ送りフラッグによる Timer の制御

４．画面の説明

シミュレーションの学習効果等を考慮した結果，ローターの１回転を120等分して描画することとした。

4.1　初期画面

実行した初期画面を図２に示す。フォームの大きさは，ディスプレイの高さの約80%比とした。

4.2　点火の描画

図３に示すように点火時には燃焼室，プラグ１とプラグ２の先端部の周りを赤く塗った。同時に，図４に示すダイヤグラムにある点火を示す下部の大きい三角形の内部を赤く塗った。さらに，他の２回の点火時にも両プラグ先端部の周りとダイヤグラムの該当する小さい三角形の内部を赤く塗った。

4.3　吸気と排気の入れ替わりの描画

オーバーラップ時は図５から図７に示すように燃焼室に吸気ガスと排気ガスを描画をするため，吸気ガスと排気ガスの境界を設定し，エンジン本体の描画後，まず境界線を描き，次いで吸気と排気の色塗り，最後に図８に示す境界線を吸気の色で再度描いた。

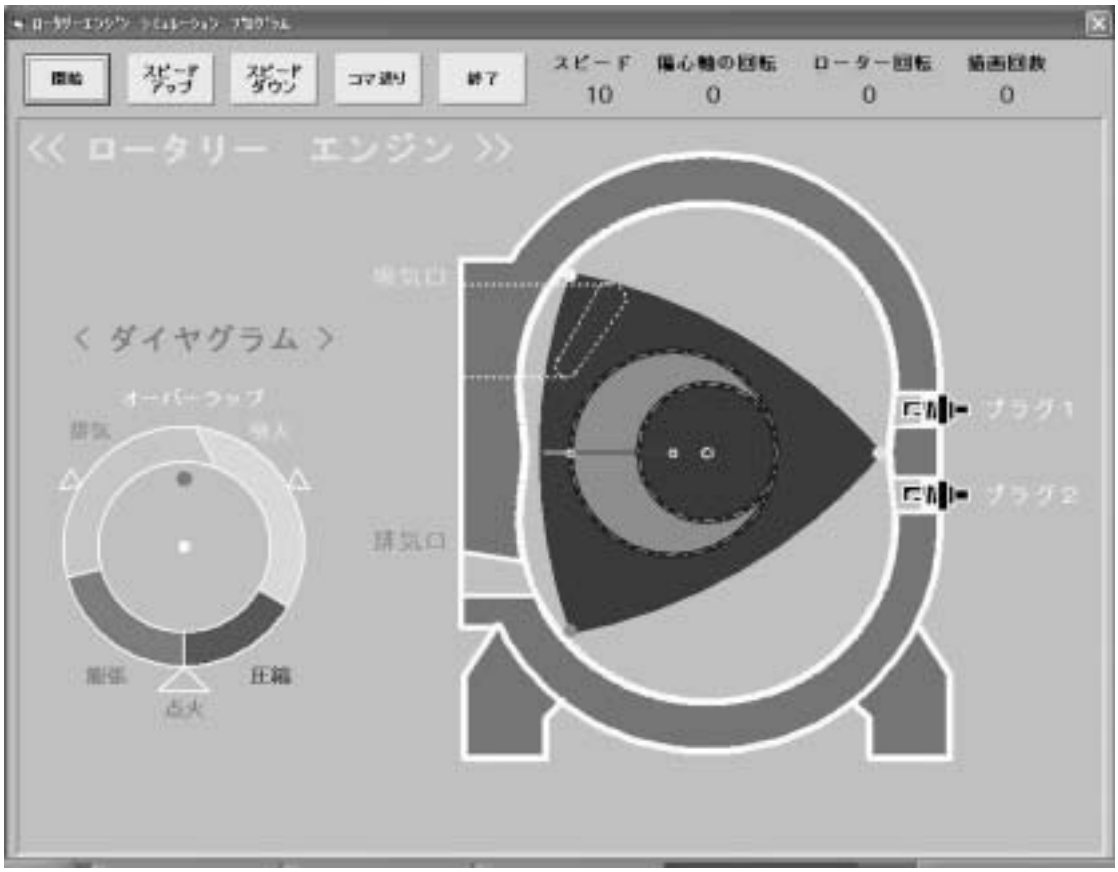

図２　シミュレーションの初期画面

５．考　　察

プログラムリストを構造化プログラミングの考え方に合わせて記述し，描く手続きの説明をレム文で書き，変数名の頭に変数名の型を付与することによりプログラムの理解に役に立つものと考える。

エンジンとダイヤグラムの描画について，2.4と2.5で述べたことは，ほぼプログラム上に実現できた。

今後の課題としては，プラグ点火時のプラグ先端部と燃焼室部を分けるハウジング部ラインを燃焼の色に変えること，用いた各部の色を再検討すること，ロータリーエンジンの説明画面の作成すること，本シミュレーションの取扱説明等を作成することなどがある。

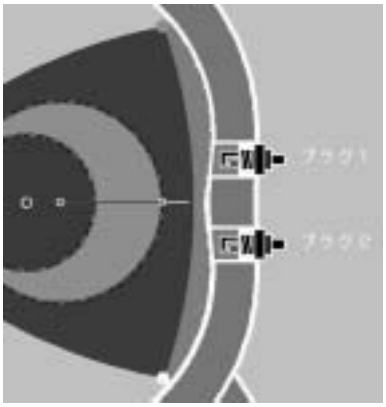

図３　点火の描画

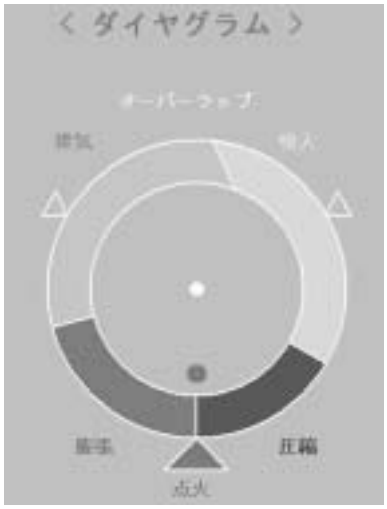

図４　点火時のダイヤグラム

パソコンの解像度の違いやワイドディスプレイ化のため色抜けを生じることがある。そのために，Circleステートメントと計算による円弧を描画し，円と円弧が重なるForm画面の縦横比を確認できるプログラムを作成し，本プログラムと共に公開する予定なので，色抜けを生じたときはこの修正用プログラムを使用して確認上，標準モジュールのForm画面の縦横比を変更して本プログラムを利用していただきたい。

（平成18年９月19日受理）

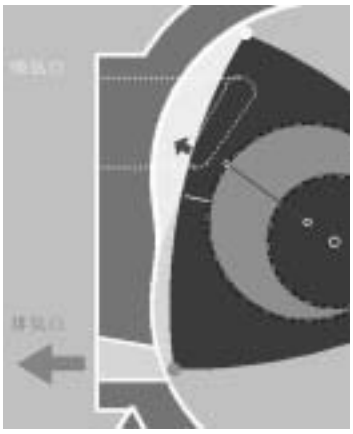

図５　オーバーラップ時１

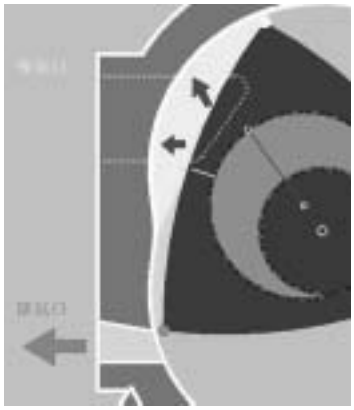

図７　オーバーラップ時３

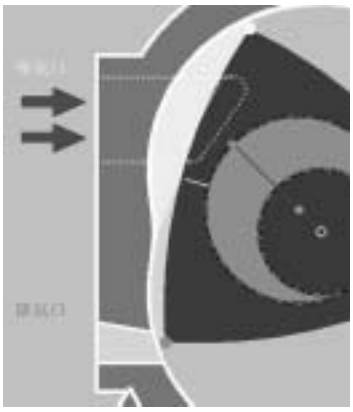

図６　オーバーラップ時２

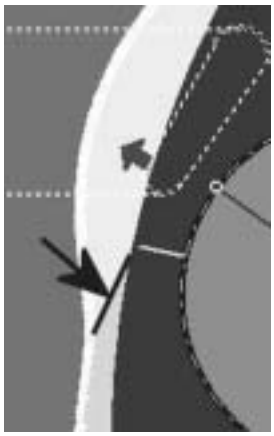

図８　描画用ライン

List １　標準モジュール FuncSetFormSize

```
Function FuncSetFormSize _
   (ByVal ObjFormName As Object, _
   ByVal LngRequestRate As Long)

   'ローカルな変数の宣言
   Dim IntNonClientWidth As Integer    '非クライアント領域の幅
   Dim IntNonClientHeight As Integer   '非クライアント領域の高さ
   Dim IntClientWidth As Integer       'クライアント領域の幅
   Dim IntClientHeight As Integer      'クライアント領域の高さ
   Dim IntRemain As Integer            '余りを格納
   Dim IntObjHeight As Integer         'form の幅を，高さに対する比で算出するための変数

   '非クライアント領域の幅，高さを求める（Twip）
   IntNonClientWidth = ObjFormName.Width - Form1.ScaleWidth
   IntNonClientHeight = ObjFormName.Height - Form1.ScaleHeight

   'クライアント領域を求める（Twip）
   IntClientWidth = Screen.Width ＊ LngRequestRate ／ 100
   IntClientHeight = Screen.Height ＊ LngRequestRate ／ 100
```

```
    'クライアント領域を TwipsPerPic の倍数に設定する（Twip）
    IntRemain = IntClientWidth Mod Screen.TwipsPerPixelX
    IntClientWidth = IntClientWidth + IntRemain
    'IntRemain = IntClientHeight Mod Screen.TwipsPerPixelY
    'IntClientHeight = IntClientHeight + IntRemain

    'フォームサイズを設定する
    ObjFormName.Width = IntNonClientWidth + IntClientWidth
    'ObjFormName.Height = IntNonClientHeight + IntClientHeight
    IntObjHeight = ObjFormName.Width ／ 1.283     'Form 画面の縦横比 （640 ／ 480） 283
    IntRemain = IntObjHeight Mod Screen.TwipsPerPixelY
    ObjFormName.Height = IntObjHeight + IntRemain
End Function
```

List 2　標準モジュール　FuncPaintObject

```
Option Explicit
'　　／／／／／　　API 関数の宣言　　／／／／／
'画面の領域を現在のブラシで塗りつぶす API 関数 FloodFill の宣言
'第１パラメータ：デバイスコンテキストのハンドル
'第２パラメータ：塗りつぶしの開始点 X 座標（Pixel）
'第３パラメータ：塗りつぶしの開始点 Y 座標（Pixel）
'第４パラメータ：塗りつぶす領域の境界の色（RGB）
'第５パラメータ：塗りつぶすの方法（0,1）
Private Declare Function ExtFloodFill Lib "gdi32" _
    （ByVal hdc As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal crColor As Long, _
    ByVal wFillType As Long） As Long

'　　／／／／／　　API 関数で用いる変数の宣言　　／／／／／
'API 関数で用いる定数の宣言
Private Const FLOODFILLBORDER = 0　　'指定された色の境界内の塗りつぶしを行うための定数
```

```
'指定されたオブジェクトの領域の指定された色の境界内を塗りつぶす
'　FuncPaintObject 関数の定義
'第１パラメータ：塗りつぶしを行うオブジェクト名
'第２パラメータ：塗りつぶしの色（RGB）
'第３パラメータ：塗りつぶしのスタイル（FillStyle）
'第４パラメータ：塗りつぶしの開始点の X 座標
'第５パラメータ：塗りつぶしの開始点の Y 座標
'第６パラメータ：塗りつぶす領域の境界の色（RGB）
Function FuncPaintObject _
    （ByVal ObjPaintObject As Object, _
    ByVal LngFillColor As Long, _
    ByVal IntFillStyle As Integer, _
    ByVal IntPointX As Integer, _
    ByVal IntPointY As Integer, _
    ByVal IntBorderColor As Long）
```

```
    'ローカル変数の宣言
    Dim LngPixelPointX As Long          'Pixel で表現した塗りつぶしの X 座標
    Dim LngPixelPointY As Long          'Pixel で表現した塗りつぶしの Y 座標
    Dim LngReturn As Long               'API 関数の戻り値
    Dim LngPaintObjectHdc As Long       '指定されたオブジェクトのデバイスコンテキストハンドル

    'デバイスコンテキストのハンドルを取得
    LngPaintObjectHdc = ObjPaintObject.hdc

    '塗りつぶしのスタイルを設定
    ObjPaintObject.FillColor = LngFillColor
    ObjPaintObject.FillStyle = IntFillStyle

    '座標を Pixel 単位に変更
    LngPixelPointX = ObjPaintObject.ScaleX（IntPointX, 0, 3）
    LngPixelPointY = ObjPaintObject.ScaleY（IntPointY, 0, 3）

    'FloodFill 関数を呼び出して塗りつぶしを実行
    LngReturn = ExtFloodFill（LngPaintObjectHdc, _
                    LngPixelPointX, _
                    LngPixelPointY, _
                    IntBorderColor, _
                    FLOODFILLBORDER）

    '塗りつぶしのスタイルを設定解除
    ObjPaintObject.FillStyle = 1      'IntFillStyle

    '再描画を実行
    ObjPaintObject.Refresh
End Function
```