愛知教育大学研究報告，57（芸術・保健体育・家政・技術科学・創作編），pp. 31～36，March，2008

２サイクルエンジンと基本４サイクル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*　梶田祐希**
Kenji OHNISHI　Yuki KAJITA
*技術教育講座
**江南市立北部中学校

１　緒　　言

本研究は，中学校技術のエネルギー変換の教育内容の充実を図るため，リアル４サイクルエンジン，ロータリーエンジンに引き続き，２サイクルエンジンと基本４サイクルエンジンのシミュレーションプログラムを作成することとした。これら一連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教えることができるものと考える。

２　２サイクルエンジンのシミュレーション

作成する２サイクルエンジンのシミュレーションは，潤滑油が必要であることを示すため，ガソリンと潤滑油の分離タイプとした。また，吸気ガス，掃気中の吸気ガスと排気ガスの混合時及び排気ガスの流れをおのおの定めた色の矢印で描画し，その大きさと向きとで視覚的に理解しやすいように考慮した。ガスの温度と圧力の変化における細かな色の設定を行った。

本シミュレーションの特徴とその使い方や２サイクルエンジンについての基本事項を理解してもらうため説明画面などを作成した。

２－１　吸気，排気と混合ガスなどの色の設定

吸気，排気と混合ガスの色塗りは，温度と圧力の高さによって，吸気ガスで４種類（LowLow, Low, Middle, High），排気ガスで４種類（LowLow, Low, Middle, High），混合気ガスで３種類（Low, Middle, High）の色分けし，適切な描画回数とポイントに行った。混合気ガスの色は吸気ガス（青）と排気ガス（赤）の中間の色に近づくように設定した。

図１　説明画面

２－２　説明画面の作成

今まで作成したプログラムは,「開始」をクリックすると，シミュレーションが始まり，本シミュレーションの使い方や２サイクルエンジンなどについての基本事項を知っていないと理解が難しかったが，プログラムに係わる説明画面を作成し，本プログラムの概要とその特徴, Commandボタンによるプログラムの使い方等の説明,「吸気」,「排気」と「吸気と排気の混合気」ガスの種類とガスの圧力と温度により色分けの仕方などをその説明とともに記載した。また，吸気と排気の流れを矢印で示していることを説明し学習者にとっても分かりやすいものとした。説明内容もできるだけ簡単に理解できるように考慮した。

２－３　「始動時」と「連続運転」の選択起動ボタン

エンジンの作動中だけでなく，エンジン始動直後の吸気ガスと排気ガスなどの動きを細かく表現するため, 燃焼室に何もない状態（始動）からシミュレーションを開始できるように，新たに「始動時」運転を設け,「開始」前に「始動時」と「連続運転」の選択を可能とした。選択後の「始動時」画面を図2,「連続運転」画面を図３に示す。

エンジンの始動開始時点から，連続時につながるようにガスの動きの描写を分かりやすくするため，If 文を用い，描画２周目の描画９までを「始動時」と「連続運転」時の場合分けを行った。

「始動時」のシミュレーション１周目の初めは，混合ガス，吸気ガスが入っていないためシリンダ内のガスの描画はない。よって，燃焼室における矢印の描画，ガスの色塗りも行われない。「開始」すると，図４に示すように描画１で吸気孔より吸気ガス，図５に示すように描画13で掃気孔より吸気ガスが入ってくる。燃焼室に吸気ガスが入ってきた２周目の燃焼後は混合ガスが表示され，描画９で排気孔から出て行く様子が矢印とともに描画される。

描画10以降においては，排気ガス矢印（赤矢印），吸気ガス矢印（青矢印）が吸気と排気ガスの動きとともに描画され，ガスの流れを示すとともに排気ガスと吸気ガスの入れ替わりを示す。このように，エンジン内部の排気ガスと吸気ガスの動きを細かく表示するため，If 文を用いて場合分けを行った。

「始動時」が,「連続運転」と同じ動きになるのは３周目のシミュレーションからである。その一画面を図６に示す。

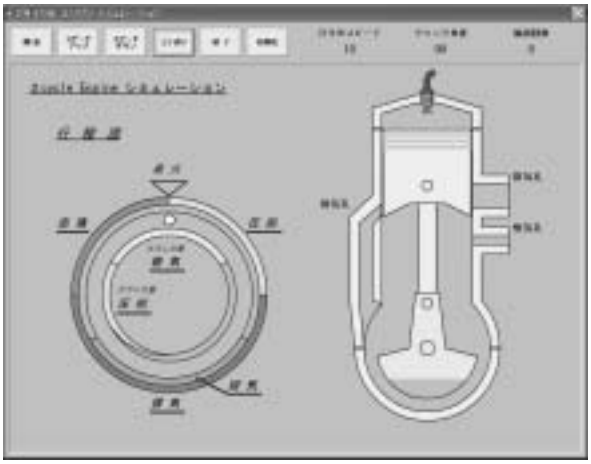

図２　「始動時」の選択画面

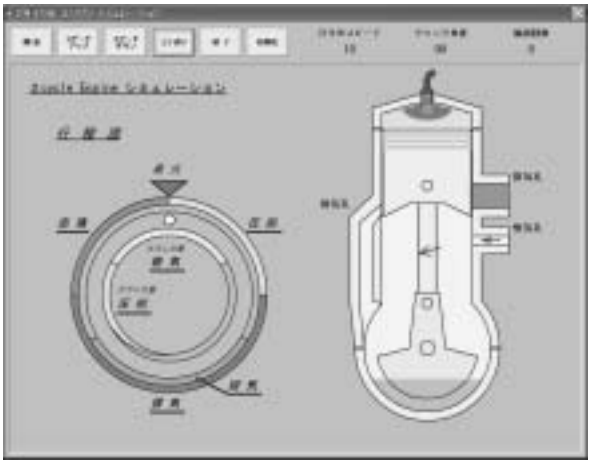

図３　「連続運転」の選択画面

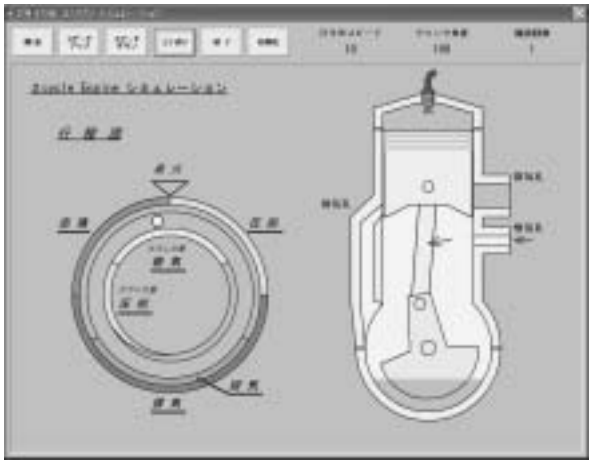

図４　吸気口よりの吸気

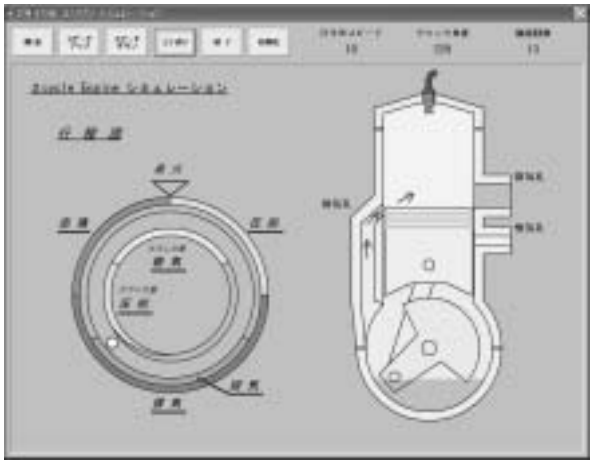

図５　掃気孔よりの吸気

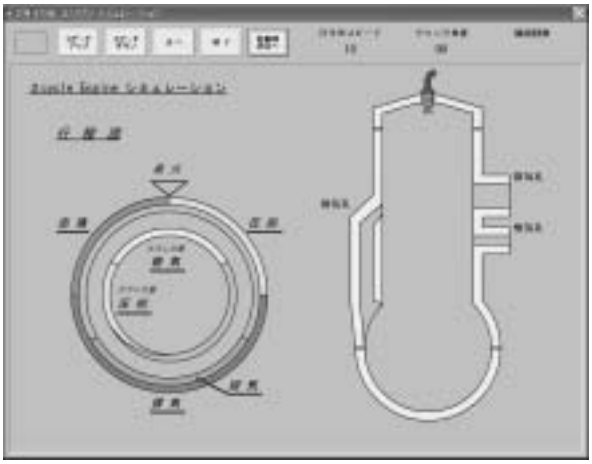

図７　初期化後の画面

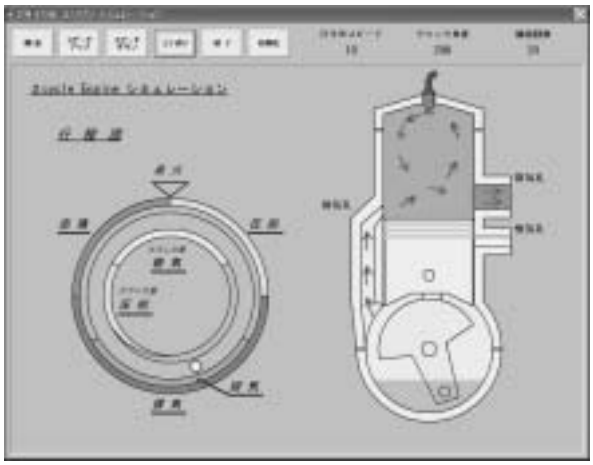

図６「掃気」時の一画面

２－４　「初期化」ボタンの作成

従来のプログラムは，シミュレーションが始まってから途中で初めの画面に戻りたい時は，コマ送りを押し続けるか，一度プログラムを終了させてから再起動させなくてはならなかった。しかし，Command オブジェクトで「初期化」ボタンを作ることによって，終了させなくても起動後に初期画面に戻ることができるようにした。その画面を図７に示す。

エンジンの勉強をするためにシミュレーションを何度も繰り返し見たい人にとっては，便利な機能である。初期化後は，「始動時」と「連続運転」を選択クリックすると，右端に「開始」コマンドボタンが表れるので，クリックしてシミュレーションを開始する。

３　基本４サイクルエンジンのシミュレーション

エンジンの特徴や作動理論などの説明と各エンジンパーツの形とその役割などを学習できる説明画面を作るのに合わせ，タイトル画面，終了画面も作ることとした。

３－１　プログラムの基本構成

従来は１つの Form モジュールと２つの標準モジュールでプロジェクトが構成されていたが，本研究では，新たにタイトル，説明，終了の３つの Form モジュールを付け加え，４つの Form モジュールと２つの標準モジュールで構成することとした。４つの Form モジュールの最初の画面（図８，９，10，11）とその関係図（図12）を以下に示す。

図８　タイトル画面

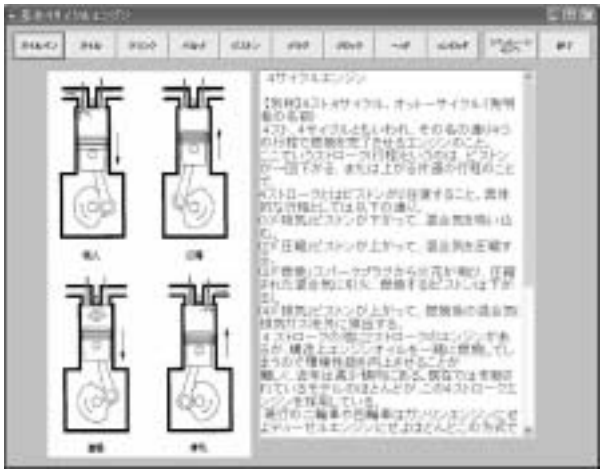

図９　説明画面の最初の画面

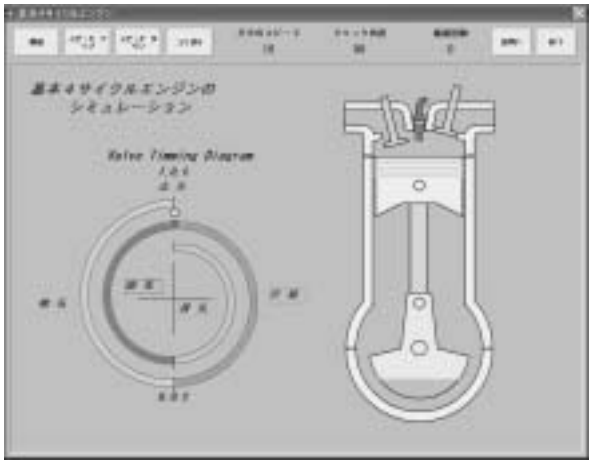

図10　シミュレーション画面

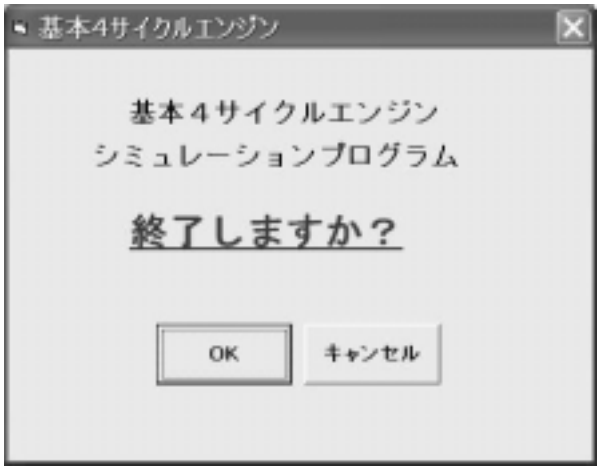

図11　終了確認の画面

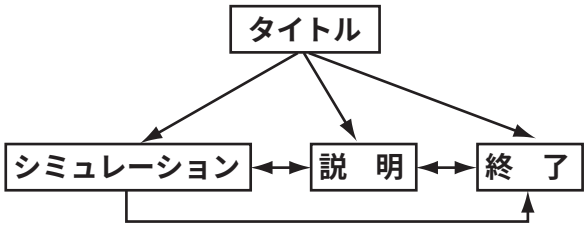

図12　４つの Form モジュールの関係図

３－２　説明画面

説明画面の Form モジュールは Command，Image，TextBox，Timer の４種類のオブジェクトなどで構成されている。Command ボタンをクリックするとパーツ画像と説明文が表示される。TextBox 内の文字と TextBox の大きさを変更出来なくするため，プロパティで Locked を True にすることで文字を，Form オブジェクトで数値を設定し大きさをそれぞれを固定した。

説明画面の初期画面は，図９に示すように基本４サイクルエンジンの行程，特徴などの説明文が出るようにした。

図13に示すように，どのパーツの説明をしているのかを分かりやすくするために Timer オブジェクトを使用し，選択されたパーツを赤色に変えて３回点滅させた。さらに，点滅後に図14と図15に示すように，選択パーツの小さい図と大きめ図をエンジン図の中央に表示させた。また，選択したパーツの説明文を TextBox に表示した。

また，説明画面の終了し，シミュレーション画面へ移る Command ボタンと，プログラム終了の Form モジュールに移動するための Command ボタンを付け加えた。

説明画面で使用する画像ファイルを置く場所をドライブ名，フォルダ名まで含め，Form オブジェクトで指定する必要がある。しかし，多くの人が本プログラム

図13　選択パーツの点滅

図14　小さな選択パーツの表示

図15　大きめの選択パーツの表示

を使用する際に常に一定の場所に画像ファイルを置くとは限らないため，画像ファイルを置くパスを取得する必要があった。画像ファイルのフォルダ “zu” をプロジェクトのあるフォルダの直下に作ることを前提にすると，「App.Path」関数でプロジェクトのあるフォルダのパスが取得できるので，画像ファイルのあるフォルダのパスは「FigPath = App.Path + “\zu\”」のよう記載することで対応した。

４　考　　察

２サイクルエンジンのシミュレーションにおいては，より便利な機能を持たせるためにまだまだ改良できることはあるものの，当初から目指してきた分かりやすく理解しやすいシミュレーションプログラムを作成できた。改良の内容として，Label オブジェクトを複数つくるときにプロパティで Index を設定するなどして，Form オブジェクトをスリム化させたり，BackColor の設定をオブジェクトごとに行いバージョンが変化しても色合いが変わらないようにしたり，上記で述べたように変数の宣言を統一して様々な数値の一括設定を可能にしたり，Command オブジェクトで「初期化」を作り，一度の起動で何回も初期画面に戻ることができるようにした。そして今回，始動時運転及び連続運転の選択を可能としたことにより，エンジンの勉強のために重要なことをポイントごとに理解することができる内容となった。

基本４サイクルエンジンのシミュレーションにおいては，従来のプログラムは１つの Form モジュールと２つの標準モジュールでプロジェクトが構成されていたが，本研究ではタイトル，説明，終了の３つの Form モジュールを付け加えてプロジェクトを構成した。特に，説明画面の Form モジュールを新たに付け加えることに重点を置いた。シミュレーションの説明画面を加えることにより，エンジンの動きだけではなく各パーツの名前や役割が理解できるようになり，内燃機関の学習をさらに深めることができるプログラムとなった。

プログラムの構成としては，従来のプログラムをさらに理解しやすいものとするため，変数，プロシージャ等で不要なステートメントを削除したり，Global で宣言する必要のない変数を Local 変数にしたり，逆に複数のプロシージャで同じように使われている Local 変数を Global で宣言するなどプログラムの整理を行った。そうすることで，Global の変数は General で変数名を宣言，Form オブジェクトでその値の設定を行うようにしたため，色や座標の数値を変更したい場合にも一カ所の数値変更で一括設定できるので便利になった。さらに，手続きの再確認を行ったことによって，必要のないコードや変数，プログラム自体の改良点をその都度見つけ解決していくことができた。Visual Basic について多少の知識がある人であればプログラム自体を簡単に理解でき，メンテナンスが行いやすくなった。また，プログラミングについてあまり知識のない人でもコードの１行上ないし，右横に書かれているレム文を読むことでそのコードはどんな役割を果たしているのかが分かるようになった。

一部修正した標準モジュール FuncSetFormSizeH のリストを掲載した。標準モジュール FuncPaintObject は，変更はない。

最後に，この一連のエンジンのシミュレーションプログラムを開発する機会を作った鈴木忠之氏に深く感謝の意を表する。

（平成19年９月13日受理）

List　標準モジュール FuncSetFormSizeH のリスト

'Wide ディスプレイが増えたため，form 画面の大きさを，ディスプレイ画面の高さに対する比で算出するように修正し，名前の最後に H を付けた。

```
Option Explicit

'指定されたフォームのクライアント領域のサイズを
'　画面のサイズ比より TwipsPerPixel の倍数で指定する
'　　FuncSetFormSize 関数の定義
'第１パラメータ：設定を行うフォーム名
'第２パラメータ：画面設定の比率（%）
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 Intege     '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 IntObjWidth 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）
    IntNonClientHeight = ObjFormName.Height - ObjFormName.ScaleHeight

    'クライアント領域を求める（Twip）
    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 = IntClientHeight Mod Screen.TwipsPerPixelY
    IntClientHeight = IntClientHeight - IntRemain

    'フォームサイズを設定する
    ObjFormName.Height = IntNonClientHeight + IntClientHeight
    IntObjWidth = ObjFormName.Height * 1.285
    IntRemain = IntObjWidth Mod Screen.TwipsPerPixelY
    ObjFormName.Width = IntObjWidth - IntRemain

End Function
```