J3W入門 for Linux講座

水谷純
http://www.nk.rim.or.jp/~jun/index.html

第1回　J3Wの文法

■J3Wとは？

「J3W」は、3Dアニメーション専用のプログラムを作成する環境です。実際の作成にあたっては、「J3C」というJava風の文法を持つプログラミング言語を使います。OpenGLやDirectXで思い通りの3DCGアニメーションを作成できる人には必要のない環境ですが、「あまりプログラミングの経験はないが、3Dのプログラムを作ってみたい」という3D初心者や、「プログラミングの経験は十分、しかし3Dは面倒そう」といった人が対象になっています。

J3C言語の守備範囲はリアルタイムに操作可能な3Dアニメーションの作成です。しかも汎用言語であるC、C++、Javaのような言語と文法的に近い言語です。少しでもプログラミングの経験があれば、比較的容易に使いこなせると思います。まずは、J3C言語の基本部分から、J3W付属のマニュアルを補う形で説明していこうと思います。

●プログラム

J3C言語のプログラムは、図1のようにクラス、ライブラリ、import宣言の繰り返しで構成されます。クラスが1つのプログラムと考えてください。実行を最初に始めるクラスとして名前がmainであるクラスが1つ必要です。

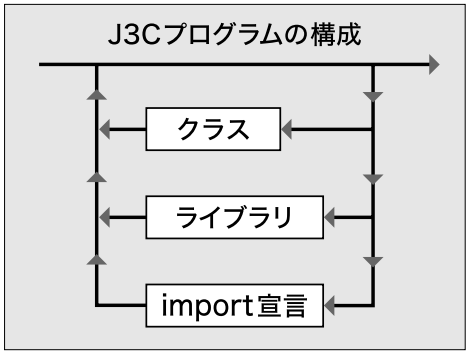
まずmainクラスだけの簡単な例でJ3C言語のプログラムミングの雰囲気に慣れましょう。すぐに理解できなくても、この記事を読み終わるころには理解できるようになっていると思います。リスト1のプログラムは1から100までの合計を計算します。変数sumを0にしておいて、変数iを1から順に100まで変えながらsumに加えるという方法をとっています。while文で条件式の値(iが100以下)が真(0でない)の間繰り返されることを利用しています。

リスト1を見て「i=i+1って間違ってない?」というアナタは見込みあるプログラミング初心者です。右側から先に読んで「i+1をiに代入」と解釈します。もしwhile文の条件式の値が最初から真でない場合は「{}」の部分は1度も実行されません。

リスト1の動作を実際に確認するには、実行例1のように3つのコマンドを実行してください。または、「j3cc -r list_1.j3c」のように1つのコマンドで実行することができます(実行例2)。j3ccは上記の3つのコマンドを順に実行するスクリプトになっています。

どうですか？ 普通の言語とあまり違いはありませんね。ここ

【図1】J3C言語のプログラムの流れ



【リスト1】list_1.j3c

```
/* list_1.j3c */
class main {
  volatile int i, sum;  // iとsumの使用を宣言
  int INIT(){           // ここから実行を始める
    i = 1;              // 変数iに1を代入
    sum = 0;            // 変数sum
    while (i <= 100) {  // iが100以下なら繰り返し
      sum = sum + i;    // sumにiの値を加算
      i = i + 1;        // iに1を加える
    }
    Number(sum);        // sumの値を表示
    Stop();             // プログラム終了
  }
  int RUN(){}           // 1度も実行されない
  int EVENT(){}
}
```

【実行例1】

```
j3c list_1.j3c
j3dasm list_1.j3m
j3w list_1.j3d
```

【実行例2】

```
jm:~$ j3cc -r list_1.j3c
 J3C ver. 1.04 2001 Jun Mizutani
     Source File : list_1.j3c
     Assembly File : list_1.j3m

 J3DASM  ver. 1.41  2001 Jun Mizutani
     Source File : list_1.j3m
     Object File : list_1.j3d
       PASS 1
       PASS 2
     Assemble 60 steps
 J3W ver.6.43 2001 Jun Mizutani
 Execute file : list_1.j3d
5050
 [finished]
```

＊[finished]の前の行に合計の5050が表示されています。

【リスト2】クラスの基本形

```
class クラス名 {変数宣言;
  定数宣言;
  int メソッド名1{…}
  int メソッド名2{…}
  int INIT() {…}
  int RUN() {…}
  int EVENT() {…}
}
```

【リスト3】

```
volatile int i, j, k;          // 変数
volatile int a[10];            // 配列変数
final int const = -9999999;    // 定数
final int id[5] = {1, 2, 3};   // 定数配列の初期化
```

からは、CやJavaといった言語との違いを中心にJ3C言語の解説を進めます。リスト1を参照しながら読むと理解しやすいと思います。

●クラス

1つのクラスは他のプログラミング言語の1つのプログラムのように独立して動作します。アニメーションする物体の1つ1つの形状や動きを指定することになります。クラスの基本的な形はリスト2のようになります。{}の中には文が並びます。

メソッドは、Cの関数に相当します。クラス内の別のメソッドから呼び出して使用します。クラスは名前が決められている3個のメソッドを必ず持っている必要があります。INITメソッドはクラスの中で最初に実行されるCのmain関数に相当するものです。INITメソッドが終了するとRUNメソッドが何度も繰り返し実行され、ほかのクラスから送られたメッセージを受信した場合に限ってEVENTメソッドが実行されます。

通常INITメソッドを初期化に使って、RUNメソッドがプログラムとしての動作を決定する部分となります。EVENTメソッドはクラス間の通信に使いますが、詳細はまたの機会に改めて解説します。INIT、RUN、EVENTメソッド以外のメソッドはクラス内のメソッドから呼び出されます。J3Cでは他のオブジェクト指向言語と異なってデータメンバ(変数)もメソッドもクラス外(派生クラスを除く)から呼び出すことはできません。

プログラムに1つ必要なmainクラスは自動的に実行されますが、main以外のクラスは他のクラスからnew()組み込み関数で起動する必要があります。起動後は別のプログラムのように独立して動作します。

クラスはINIT()、RUN()、RUN()というように実行されることを覚えておいてください。

●コメント

コメントはプログラムに付ける注釈で、プログラムの動作には全く影響しません。実際には、コンパイラが単に読み飛ばします。Cの形式の「/*」と「*/」に囲まれた部分(改行を含んでも良い)と、C++で使われる「//」以降行末までがコメントとみなされコンパイラに無視されます。/*と*/に囲まれた部分に、入れ子で「/* */」形式のコメントを含むことはできませんが、//形式のコメントを含むことはできます。

●変数、定数

変数とは、値を入れておく容器のことです。容器に入れる内容物の種類をデータ型といいますが、J3Cでは最も基本的な整数である32ビットの符号付整数(約±20億の範囲)のint型と、int型の配列だけと単純です。配列とは、数の決まったデータの並びに名前を付けて、その中の1つを番号(インデックス)で指定できる形式です。例えば、「weight[5] = 60;」が「出席番号5番の生徒の体重を60kgに設定」というように、1群のデータをまとめて取り扱う場合に使います。

定数とは、数値に名前を付けたものです。例えば100という数値が高さの最大値となっているとします。100という数値にMAXという名前を付けておいて、プログラム中で数値ではなくMAXという名前を使います。すると、例えば、後からMAXを200と変更するだけで、すべての100を200に変更しなくても、自動的に最大値を200に変更できることになります。「定数とは数値に分かりやすい名前を付ける機能」と考えてください。定数は多くの言語でconst宣言で指定しますが、J3CではJavaと同じくfinalで指定します。値が変化しない変数という扱いですが、J3Cではコンパイル時に値を決定してしまいます。final宣言した変数(定数)はメモリを消費しませんが、必ず初期化する必要があります。

変数は、宣言にvolatileを指定します。従って、「int i;」という宣言はできません。変数の場合は必ず「volatile int i;」とする必要あります(リスト3)。

配列は、宣言時にサイズを指定できないJavaと異なって、サイズを指定する必要があります。添え字(インデックス)の範囲は0から「サイズ指定した値-1」となることに注意してください。

また配列名のみを参照した場合はメモリの位置(ポインタ)が返りますが、J3Cでは一部の組み込み関数で使用するだけで、ポインタを操作することありません。

変数、定数には大きく分けてそれぞれ2種類あります。データメンバと局所変数(定数)です。データメンバはクラスの変数宣言と定数宣言部で宣言された変数や定数です。局所変数はメソッド内の「{}」で囲まれたブロックで宣言された変数や定数を示します。Java、C++と異なってブロック中の任意の位置で宣言することはできません。Cと同じようにブロックの先頭部分でまとめて宣言する必要があります。また局所変数では配列を宣言することはできません。

変数名、定数名、メソッド名、クラス名などの識別子は最大63文字までです。

●数値

数値は32ビットの符号付整数(約±20億の範囲)が使用できますが、10進数以外にも、16進数、キャラクタの各形式で記述可能です。例えばvalという変数に代入する数値の形式をリスト4のように指定できます。キャラクタを代入すると文字コードが整数として格納されます。

●文

文はプログラムを実行する場合の命令の最小単位です。「プログラムの実行」とは、文を順に実行することにほかなりません。文は処理の内容を実際に記述する単位で、代入文、条件文(if、switch)、繰り返し文(while、for、do)また処理の流れを制御するbreak、continue、return文、アセンブラに直接渡すasm文、別に定義した処理を実行するメソッド呼び出しとライブラリ呼び出し、J3Wの機能を直接実行する組み込み関数呼び出しがあります。文と文の区切りにはセミコロンを使います。

文は値を返さないため式として使用できません。Cのように代入文の右辺や条件式で文を置くことはできません。

```
[誤]a = b = 0;
[正]a = 0; b = 0;
```

```
[誤] if (a = b) 文;
[正] a = b;
     if (a != 0) 文;
```

条件文、繰り返し文などはC、C++、Javaとほとんど同じです。C、C++、Javaと特に異なる文はif文とreturn文です。if文ではelseの前にセミコロンを置くことはできません(PascalまたはDelphiと同じです)。

```
[正] if ( 式 ) 文 else 文;
[誤] if ( 式 ) 文; else 文;
```

【リスト4】

```
val = 123456789;            // 10進数
val = 0xABCDEF0;            // 16進数
val = 'A';                  // キャラクタ(文字コード)
```

【リスト5】

```
for (int i = 0; i < 100; i++) {...}
```

return文は式を評価するために使用し、関数の途中からの脱出の目的に使用することはできません。

また局所変数宣言と演算子の制限から、for文ではC++のように局所変数を最初に使う場所で宣言する、リスト5の形式は使用できません。ブロックの最初で局所変数宣言する必要があります。もちろん 変数iが宣言済みの場合は、

```
for (i=0; i<100; i=i+1) 文;
```

だけで構いません。for文を普通に使用する場合は気にする必要はありませんが、文を式として使用できないため、i<100の部分には必ず式を指定します(C上級者の人は要注意です)。

●式

式には、普通の算術演算と比較演算があります。式の結果はどちらもint型が返ります。算術演算子は+、−、*、/、|、&の6種類です。|はビットOR、&はビットANDです。

比較演算子は==、!=、>、>=、<、<=の6種類です。大きさの比較は見たままですが、==と!=は分かりにくいかもしれません。==は等しい場合に真、!=は等しくない場合に真となる演算子です。比較演算は偽(False)の場合は0、真(True)の場合は1となります。if、while、for文で使う式は0以外は真、0ならば偽と判断されます。優先順位は*、/、&が最も高く、+、−、|がそれに続き、比較演算子は最後に評価されます。

●組み込み関数

3Dアニメーション専用の組み込み関数が多く用意されていることが、J3C言語の特徴になっています。J3Cの組み込み関数は、内部的に仮想マシンであるJ3Wの命令1つ1つに対応しています。組み込み関数は機能面から以下のように分類できます。

・インスタンスに関する操作

J3Wの中心的な存在であるインスタンスの生成と削除、インスタンス間の通信に関する組み込み関数で10種類あります。

・「物体」に対する操作

1つのインスタンスは1つの「物体」を持つことができます。「物体」の形状を設定したり、位置や角度の設定と取得を行う組み込み関数があります。

・「物体」の運動に関する操作

「物体」を持っているインスタンスを実行すると、3次元空間内で「物体」が運動する組み込み関数です。

・演算

三角関数、平方根、乱数を求める組み込み関数が用意されています。

・レジスタに関する操作

レジスタ変数の設定、メモリへの格納、スタックへの退避とスタックからの復帰などに関する組み込み関数です。

・共有メモリ

各インスタンスからアクセスできる共有メモリに対する入出力用の組み込み関数です。共有メモリはメッセージの送受信とともにクラス間の情報交換の手段に使用します。

・入出力

テキスト画面とグラフィック画面への文字出力、線分の描画、音の出力とキーボード、マウスからの入力に関する組み込み関数です。これらの組み込み関数はインスタンスごとに独立していません。どのインスタンスも同じデバイスに対して操作することになります。

・システム関連組み込み関数

J3Wシステム(インスタンスに依存しない)に関する状態の取得と設定、共有メモリのファイル入出力に関する組み込み関数です。

組み込み関数の詳細はJ3Wに付属するj3c_func.htmlを参照してください。重要なものはサンプルコードで使う度に解説します。

●レジスタ変数

組み込み関数の中には引数をレジスタ変数を使って渡し、結果をレジスタ変数に返す関数があります。RX、RY、RZ、RH、RP、RBレジスタ変数は、組み込み関数のうち30種類程度の関数の呼び出し時の設定、または、返り値の取得に用いられます。また、クラスインスタンス生成時のRX、RY、RZ、RH、RP、RBレジスタ変数の値は生成されたインスタンスに引き継がれます。「コンストラクタの引数」的な働きをさせることができます。この「コンストラクタの引数」を利用すると、同じクラスを使って位置、色、大きさなどの異なる複数のインスタンスを生成できます。便利な機能なので詳細は改めて解説します。

RQレジスタ変数はクラスインスタンスの識別(プロセスIDと呼ぶ)に使用します。メッセージの送受信というインスタンス間の情報交換のために重要なレジスタ変数です。RLレジスタ変数は受信されたメッセージを保持しています。RLレジスタに書き込むことは可能ですが、クラスのEVENTメソッド内でメッセージの参照に使用するだけにしましょう。レジスタ変数はインスタンス(後述)内でグローバルな変数です。

・プログラム例

リスト1と同じように、1から100までの和を計算するプログラムを2つ紹介します。リスト6ではfor文を使って少し簡潔にしてみます。変数iの初期化と終了条件、繰り返しごとのiの変化の仕方をfor文1つで指定できます。J3Wのfor文は、CやJavaと異なっており、増分を「i=i+1」のように指定する必要があることに注意してください。いくつかの組み込み関数を使っています。Number(sum)は、変数sumの値を10進数として表示する組み込み関数です。Stop()はリスト6のプログラムを終了させます。

次のリスト7も同様に1から順に加えていきますが、ループ構造がありません。J3Cのクラスは、INITが最初に1回実行された後、RUNメソッドが繰り返して実行されるため、結果として

【リスト6】list_2.j3c

```
/* list_2.j3c */
class main {
  volatile int i, sum;
  int INIT(){
    sum = 0;              // 合計を0に初期化
    // 1から100まで繰り返す
    for (i=1; i<=100; i=i+1) {
      sum = sum + i;      //  sumの値をsum+iに変更
    }
  }
  int RUN(){
    Number(sum);          //  sumの値を表示
    Stop();               //  最初のRUNで終了
  }
  int EVENT(){}
}
```

【リスト7】list_3.j3c

```
/* list_3.j3c */
class main {
  volatile int i, sum;
  int INIT(){
    i = 1;                // 初期化だけ
    sum = 0;              // 合計を0に初期化
  }
  int RUN(){
    if (i <= 100) {       // iが100以下の場合
      sum = sum + i;      // 総和を求める
      i = i + 1;          // iの値に1を加える
    } else {              // iが101の時
      Number(sum);        // sumの値を表示
      Stop();             // 終了
    }
  }
  int EVENT(){}
}
```

ループとなります。どんなクラスもINITで初期化、RUNで繰り返しというJ3C言語の重要な特徴を示しています。INITが1回、RUNが101回実行されてプログラムが終了します。他の言語に習熟している人もこの部分だけは注意してください。

物体の回転や移動という動作は、RUNが繰り返されるこの形式を多用します。INITメソッドで物体の形状や初期位置を設定して、繰り返し実行されるRUNメソッド中で回転や移動の組み込み関数を実行することでクラスで定義される物体を自立的に運動するようにプログラムを組むことができます。

リスト1とリスト6は他の言語とあまり変わりませんが、リスト7はJ3C言語特有のプログラミング手法です。リスト7の動きを理解すると、J3Wに付属するサンプルプログラムが読みやすくなります。

■終わりに

今回は3Dアニメーションと直接関係ない文法の説明でした。プログラミング言語を習得する近道は、「プログラムに実行させたいことを具体的に考えて、日本語で書いてみる」ことだと思います。その文章にあいまいな部分がなくなり、文章として完成したと思ったら、対象となるプログラミング言語に翻訳します。その時点ではプログラムに実行させたい内容がはっきりしているため、マニュアルやサンプルの情報のうち必要な部分簡単に見付け出せるものです。特に初心者はプログラミング言語に振り回されず、まず日本語で実際に書いてみることです。私自身は、「プログラムが書けない原因は言語のせいではなく、単に自分の考えがまとまらないためである」と悟っているつもりです。

次回も、もう少しJ3Cの文法にお付き合いください。

Column J3W for Linux 最新版ソースコードについて

J3W for Linuxの最新版ソースコード「j3w643.tar.gz」を、本誌付録CD-ROMに収録しています。インストールを行うには、まず圧縮ファイルを展開後、j3w-643ディレクトリに移動します。その後ルートユーザーになり、「make && make install」とすると、作業は終了します。コマンドの検索パスに/usr/local/binが含まれているかどうかの確認も、忘れずに行ってください。インストールに関するさらに詳しい情報を知りたい場合には、作業を行う前にJ3W添付のREADME.eucファイルを確認してください。（編集部）

Column J3Wの内部構造の解説

J3Wを使う上ではJ3Wの内部構造を知る必要はありませんが、どのように動作しているか「仕組み」を知りたい人もいるかもしれませんので、これから数回に分けてJ3Wで使用しているデータ構造を中心に解説します。

3次元コンピュータグラフィックスのプログラムを3Dエンジンから作る場合には、三角関数、ベクトル、行列、4元数、座標変換といった数学の知識が必要となります。それらの代数幾何学の基本的なアルゴリズムに関する情報は、今では高額な書籍を購入しなくてもインターネットを利用すれば比較的容易に入手できるようになってきました（英語を読む覚悟があればさらに容易です）。コンピュータが高速になったおかげで、描画や計算の高速化のために固定小数点演算を使用したり、移動や物体の形状に制限を与える必要がないため、J3Wでは3Dに関連するアルゴリズムはほとんど教科書通りのものしか使っていません。しかしプログラムを構成する要素はアルゴリズムだけでなく、データ構造も同じぐらい重要な位置にあります。

●J3Wの3Dエンジン

3Dアニメーションは3次元のコンピュータグラフィックスを連続して高速に描画することで実現します。つまり、3Dアニメーションといっても普通の3次元グラフィックスと原理は同じです。

J3Wの3Dエンジンと呼べる部分の概念図を図に示します。J3WはC++で書かれていて、ほとんどのデータ構造はクラスを使って実装されています。機能としては、

1. 物体の形状を決める頂点や多角形を登録し、
2. 物体の移動や回転に伴って物体の形状を定義する最も基本的なデータとなる頂点座標の変化を管理し、
3. それらの頂点座標を1つだけ存在するワールド座標系にまとめ、
4. ワールド座標系の物体の座標を観察者の視点から見た風景に変換して、
5. 2次元のスクリーン座標系に変換して表示する。

のように集約することができます。

誌面の関係で細かい部分を省略することになりますが、J3Wのソースも参照してみてください。クラス宣言は拡張子がhのヘッダファイル中にあります。また、ファイル名はクラス名と似た名前になっているので迷うことはないと思います。

THobj3Dクラスは1つの物体の形状を表現するために頂点座標や面を管理しています。また、「ジオメトリックエンジンの機能」として、回転、移動、拡大縮小にともなう座標変換の計算をすべて担当しています。さらに、多関節オブジェクト（例えば人型ロボット）を実現するために必要な物体の階層構造を別のTHobj3Dオブジェクトへのポインタを持つことによって管理しています。

表示を受け持つ「レンダリングエンジン」の機能は、Spaceh3Dとそれが描画に使うTScreenのX用の派生クラスであるScrnXが担当しています。Spaceh3Dクラスはクリッピング、2次元への変換、光源の種類と照光処理、3Dに関係のない文字表示や直線描画に関する機能を持っています。TScreenはOSと表示用のライブラリに依存する部分（この場合Xlib）のラッパークラスとなっています。（水谷純）

【図】3Dエンジン部分の概念図