卒業論文

# 第一部

# WWW上の3次元空間表現
# A three-dimesional space expression on WWW

**江森　文**
**Aya EMORI**

**あらまし** コンピュータの処理能力とともに，コンピュータ上で仮想現実を構築することに関心が集まっている．VRMLは，これまで，2次元のデータしか見ることのできなかったインターネットの世界で，その制約を取り払い3次元の仮想空間を表示することを可能とする．本研究では，VRMLについて解説したあと，新しく建設された日本大学文理学部8号館の見やすいガイドホームページをVRMLを用いて作成する．また，第二部で3次元の表現法の一つである「Claymore」についての論文を解説する．

**キーワード** VRML(Virtual Reality Modeling Language),WWW(World Wide Web)

## 1　はじめに

現在，コンピュータの処理能力とともに，コンピュータ上で仮想現実を構築することに関する関心が高まっている．例えば，ナビゲーションシステム等において，3次元で道路地図を表現でき，視覚ですぐに交通状況を認識出来るようになっている．

本研究では，仮想空間を表現するVRMLを用い，新しく建設された日本大学文理学部8号館の構造を表現することを目的とする．VRMLとHTMLの応用により，8号館のガイドホームページを構築することで，来校者等は，インターネットを利用して事前に調べることが出来る．更に，アニメーション機能を利用し，夜久研究室までの動的ガイドを実現することも目的とする．このページは，Web上で内部を見ることが出来る．

### 1.1　VRMLの背景 [3][4]

1994年World Wide Web国際会議で初めてVRMLが発表．春のジュネーブで開催された会議のBOF(Birds of Feather;特定のトピックスについて興味を持つ人の非公式のミーティングの事)と呼ばれる正式の会議以外の課外ミーティングで，3次元データを統一的にやりとりできる仕組みを作ろうをいう事が話し合われたところから誕生．VRML Version1.0の最初の草案(First Draft)は1994年11月2日付けで発表され，1995年5月26日付で最初の仕様書(First 1.0 spec)として制定．その後1996年8月4日に大幅に機能拡張されたVersion2.0の仕様書が発表．VRML2.0では，センサー機能などにより，よりインタラクティブな，動きのある3次元空間を構築できるようになる．1997年VRML2.0は国際標準規格になり，VRML97に呼称を変える．

VRML1.0が単にシーンの静的な記述にとどまっていたのに対して，VRML2.0は動きの記述が可能になり，マウスのクリックによる形状や色の変更といったユーザとのインターラクション，2つのオブジェクトが衝突して跳ね返るようなオブジェクトの動きとそれらの間のインターラクションが記述できるようになっている．

さらに，VRML以外にもWeb上の3 D表現法の研究がされている．これについては，「Claymore」解説を

二部で紹介する.

### 1.2 VRML の目的 [3][4]

これまで，2 次元のデータしか見ることのできなかったインターネットの世界で，その制約を取り払い 3 次元の仮想空間を表示．自分の意志により，3 次元仮想空間内を自由に歩き回ることができる.

### 1.3 VRML のファイル構成 [2]

一行目にはこのファイルが VRML で書かれていることを示すヘッダ”♯VRML V2.0 utf8” を書く．この行によって，ブラウザはこのファイルが VRML のバージョン 2.0 で書かれていることを理解する．また，この行を除いて，”♯”を用いれば，シャープ記号より後ろの文字列は，コメントとみなされブラウザが無視する．VRML で書かれていることを示す拡張子は，.wrl（world の略）とする．また，ブラウザは，Windows95/NT で走るものとして，SGI 社で開発された Cosmo Player が無償で入手可能である.

## 2 VRML2.0 [1][2][3]

### 2.1 VRML のノードとフィールド

VRML のノードとフィールドについて解説する.

#### 2.1.1 基本形状（Shape ノード）

VRML で基本形状を定義するには Shape ノードを用いる.

⟨1⟩geometry フィールド

(i) 基本の図形

形状を示すノードを指定出来るフィールド．基本形状を示すノードとしては，Box（直方体），Cone（円錐），Cylinder（円柱），Sphere（球）が挙げられる.

| 形状ノード | フィールド | 意味 |
|---|---|---|
| Box | Size | 幅，高さ，奥行きのスペースでくぎって指定 |
| Cone | BottomRadius | 底面の半径 |
| | Height | 高さ |
| | Side | 側面の表示の有無 |
| | Bottom | 底の表示の有無 |
| Cylinder | Radius | 半径 |
| | Height | 高さ |
| | Bottom | 底の表示の有無 |
| | Side | 側面の有無 |
| | Top | 上面の有無 |
| Sphere | Radius | 半径 |

図 1 基本形状ノードと設定値

例 円柱

```
♯VRML V2.0 utf8
Shape{
  geometry Cylinder{
  bottom TRUE
  height 2
  radius 1
  side TRUE
  top FALSE
  }}
```

(ii) ユーザー指定の図形（IndexedFaceSet ノード）

IndexedFaceSet ノードを使うと，ユーザーが定義する任意の形状の立体モデルを作成することができる．すなわち，好きな図形を作れる． 例えば，六角柱を定義するには 0 ～ 11 の 12 点の座標データと，面を構成する頂点の組合せデータを使う.

IndexedFaceSet ノードの coord フィールドの Coordinate ノードの point フィールドに 0 ～ 11 の各頂点の座標データを指定し，coordIndex フィールドに面を構成する頂点番号の組合せデータを指定する．指定するときには，各頂点番号の終わりに区切りとして-1 を置くようになっている.

例 赤い六角柱

```
♯VRML V2.0 utf8
Shape{
  appearance Appearance{
```

```
    material Material{
      diffuseColor 1 0 0
    }}
  geometry IndexedFaceSet{
    coord Coordinate{
      point[-1 0 1,-2 0 0,-1 0 -1,1 0 -1,2 0 0,1 0 1,
        -1 -2 1,-2 -2 0,-1 -2 -1,1 -2 -1,2 -2 0,1 -2 1]
    }
    coordIndex[0 5 4 3 2 1 -1 6 7 8 9 10 11 -1
      0 1 7 6 -1 1 2 8 7 -1 2 3 9 8 -1 3 4 10 9 -1
      4 5 11 10 -1 5 0 6 11]
  } }
```

(iii) 文字を表示

VRML によって文字を表示するには Shape ノードの geometry フィールドに Text ノードを指定することで可能になる．Text ノードには string フィールドと fontStyle ノードを持つ．

文字列として日本語を使用することはできず，また文字列に厚みをつけることもできない．

フォント名には，SERIF（デフォルト），SANS, TYPEWRITER が指定でき，文字スタイルには，PLAIN（文字飾りなし:デフォルト），BOLD, ITALIC, BOLDITALIC が指定できる．

例 文字

```
♯VRML V2.0 utf8
Shape {
  appearance Appearance {
    material Material {}
  }
  geometry Text {
    string "MOJI"
    fontStyle FontStyle {
      family "SANS"
      style "BOLD"
      size 1
    }
  } }
```

⟨2⟩ 色や模様（appearance フィールド）

(i) 物質の材質（Material ノード）

Shape ノードの中で，色や模様などの外観を指定することができる．appearance フィールドは Material ノードを持っており，物質の材質（表面の色や透明度）を定義することができる．Material ノードは 6 つのフィールドを持ち，デフォルト値は以下の通りである．

| フィールド名 | 意味 | デフォルト値 |
|---|---|---|
| DiffuseColor | 拡散光の色 (基本色) | 0.8 0.8 0.8 |
| EmissiveColor | 放射光の色 | 0 0 0 |
| SpecularColor | 鏡面光の色 | 0 0 0 |
| ambientIntensity | 周囲光強度 | 0.2 |
| Shininess | 輝度 | 0.2 |
| Transparency | 透明度 | 0 |

図 2　Material ノードと設定値

| r(赤) | g(緑) | b(青) | 合成色 |
|---|---|---|---|
| 0 | 0 | 0 | 黒 |
| 0 | 0 | 1 | 青 |
| 0 | 1 | 0 | 緑 |
| 0 | 1 | 1 | 水色 |
| 1 | 0 | 0 | 赤 |
| 1 | 0 | 1 | 紫 |
| 1 | 1 | 0 | 黄 |
| 1 | 1 | 1 | 白 |
| 0 | 0 | 0.5 | 暗い青 |
| 0.2 | 0.2 | 0.2 | 暗い灰色 |

図 3　色指定の例

例 黄色の円柱

```
♯VRML V2.0 utf8
Shape{
  appearance Appearance{
    material Material{
      diffuseColor 1 1 0
      transparency 0.8
  }}
  geometry Cylinder{
    bottom TRUE
    height 2
    radius 1
    side TRUE
    top FALSE
}}
```

(ii) 画像を貼り付ける（ImageTexture ノード，TextureTransform ノード）

ImageTexture ノードの url フィールドにイメージファイルを指定することで，物体の表面に画像を貼りつけることができる．貼りつけられる画像をテクスチャと呼ぶ．

イメージファイルとしては，GIF 形式，JPEG 形式，PNG（Portable Network Graphics）形式のものが使用できる．ImageTexture ノードは appearance フィールドの texture フィールドに記述する．

TextureTransform ノードの scale フィールドで縦方向と横方向に貼る画像の枚数を指定することができる．TextureTransform ノードは，appearance ノードの textureTransform フィールドに記述する．

例 くまの画像を直方体に貼り付け

```
♯VRML V2.0 utf8
Shape {
  appearance Appearance {
    texture ImageTexture {
      url "./kuma.bmp"
    }
    textureTransform TextureTransform {
      scale 2 1
  }}
  geometry Box {size 8 3 0.1 }
}
```

### 2.1.2 位置の指定（Transform ノード）

VRML で表示される物体は，その物体が定義されている座標の原点（0,0,0）に中心を合わせて配置される．2 つ以上の物体を配置する場合には Transform ノードを使って物体を配置する位置を指定することができる．座標の平行移動は translation フィールドで行う．

例 赤の球と地面

```
♯VRML V2.0 utf8
Transform{
  translation -2 0 2
  children Shape{
    appearance Appearance{
      material Material{
        diffuseColor 1 0 0
        shininess 0.2
    }}
    geometry Sphere{
      radius 1
}}}
Transform
  translation 0 -1 0
  children Shape{
    appearance Appearance{
      material Material{
        diffuseColor 0.3 0.5 0
    }}
    geometry Box{
      size 50 0.1 50
}}}
```

### 2.1.3 視点の設定（Viewpoint ノード）

Viewpoint ノードはビューア（どのように VRML シーンを見るか）を定義する．ビューアを複数個定義することはできるが，それらを一度に使うことは出来ない．(画面が複数個に分割されてそれらを別の視点から見るということは出来ない)．そこで，ブラウザは Viewpoint ノード専用のスタックを持っている．スタックの一番上に積まれている（バインドされている）Viewpoint が現在使用中のビューアになる．ブラウザが専用のスタックを持つのは Fog や Background,NavigationInfo の各ノードと同じ．

視点の位置と視線方向は position と orientation に設定する．位置はローカル座標系における視点の位置を指定する．視線方向を SFRotation で設定するのは不自然に感じられるかもしれないが，orientation フィールドに書かれるこの回転移動はデフォルトのビューアをどのように回転させるかを記述する．デフォルトのビューアは z 軸の負方向を向き，x 軸の正方向を右，y 軸の正方向を上にしてシーンを眺める．ビューアを位置と注視点

で定めたのでは視線回りの回転の自由度が残ってしまうので「ビューアップベクトル」を指定する代わりに回転を指定する．

FieldOfView フィールドは視野の広さを角度 (ラジアン) で指定する．この角度を小さくすると望遠レンズを使ったように見え，その反対に大きくすると広角レンズを使ったようにみえる．Description にはこの Viewpoint を識別するための情報を書き込んでおく．

jump フィールドには Viewpoint が変更された際に，視点の変更を実際に行うかを制御する．jump が TRUE であればバインド時にジャンプ（視点変更に伴いアニメーション）する．

例 回転角度　1 (右に約 60°)

```
♯VRML V2.0 utf8
Viewpoint {
  position 0 0 8
  orientation 0 0 1 1
}
Shape {
  appearance Appearance {
    texture ImageTexture {
      url "./kouji.bmp"
    }
    textureTransform TextureTransform {
      scale 1 1
  }}
  geometry Box {}
}
```

デフォルト値 0 と回転角度　-3.14（左に約 180°）については，付録参照．

### 2.1.4　リンクの張り付け（Anchor ノード）

Anchor ノードを用いれば，物体にリンクを張り付けることができる．

Children フィールドにリンクを張る物体を指定し，url フィールドにリンク先の URL を指定する．Description ノードには，リンク位置でカーソルがハンドマークになったときにステータスバーに表示される文を指定する．

例 リンク

```
♯VRML V2.0 utf8
Background {
  skyColor 0 0.8 0.8
}
Anchor{
  description "Go to rouka!"
  url "kouji.html"
  children[
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 0.5 0.5 0.5
      }}
      geometry Box {
        size 0.2 0.2 0.5
}}]}
```

## 3　光源 [2]

VRML2.0 では，物体から反射される光は「拡散光」「鏡面光」「環境光」の 3 つに分類される．「拡散光」は，照明が面を照らしている時，すべての方向に均一に散乱する光で，どの方向から面を見ても同じ明るさに見える．これに対し，鏡に光をあてると光は特定の方向に反射し，その方向から鏡を眺めるととても明るく見える．多くの物質はこの鏡の性質を多少とも持ち，このように，照明に対して特定の方向に強く反射する光を「鏡面光」と呼んでいる．「環境光」は，照明からの光が何度となく反射を繰り返して，方向を特定できない光である．この光によってすべての面がその向きに無関係に照らされる．

照明はその種類によって様々な色の光を放射する．また，その光による拡散光，鏡面光，環境光の色も様々に変化する．このようなことを考慮して，VRML では照明に対して赤緑，青の光の成分を別々に設定することができるようになっている．

ここでは，拡散光，鏡面光，環境光がどのように計算されるかの概要について述べる．

VRML2.0 では Fog ノードによってフォグ (霧) を定義することができ，全体を霧で霞ませる (視点からの距離に応じてきりの色を混ぜ合わせる) ことも出来ますが，ここでは Fog ノードの影響を無視する．また，平行光源 (DirectionalLight)，点光源 (PointLight)，スポットライト (SpotLight) の 3 種類の光源を定義出来るが，ここでは点光源の場合について考える．

一つの点光源が設定された場合，ライティング式は (1) 式で与えられる．複数の光源が設定されている場合はそれらの影響を加えて重ね合わせる．

$$I_{rgb} = O_{ergb} + atten \times I_{prgb} \times (ambient + diffuse + apecular) \qquad (1)$$

ここで，

・ =ベクトルの内積，ただし，内積が負であれば 0

$I_{prgb}$ =光源の色

$I_a$ =光源の環境光強度

$L$ =輝度を計算すべき点から光源への単位ベクトル

$N$ =単位法線ベクトル

$O_a$ =材質の環境光反射率

$O_{drgb}$ =拡散光の色，Material，Color，Texture ノードに依存

$O_{ergb}$ =材質の放射光の色

$O_{srgb}$=材質の鏡面光の色

$V$ =点から視点への単位ベクトル

$atten = min\{\frac{1}{(c_1 + c_2 \times d_L + c_3 \times d_L^2)}, 1\}$

$ambient = I_{ia} \times I_{prgb} \times O_{drgb} \times O_a$

$c_1, c_2, c_3$ =光源の減衰係数

$d_L$ =光源と点との距離

$diffuse = k_d \times O_{drgb} \times (N \cdot L)$

$k_d = k_a$ =光源 i の強度

$shininess$ =材質の shininess

$speccular = k_s \times O_{srgb} \times (\frac{N \cdot (L+V)}{\mid L+V \mid})^{shininess \times 128}$

この式を用いて $I_{rgb}$ の RGB のそれぞれの値が計算される．物体の放射光成分 $O_{ergb}$ が加えられているとともに，環境光成分は ambient，拡散光成分は diffuse，鏡面光成分は specular を元にして与えられている．例えば，環境光成分は $atten \times I_{prgb} \times ambient$ として計算され，ambient に減衰係数（atten）と光源の色（$I_{prgb}$）をかけた値となる．

点光源から発せられる光は距離の 2 乗の逆数に比例して減衰するのが光の物理特性であり，減衰係数はこの特性を近似するため利用される．3 DCG では距離の 2 乗の逆数に比例させるだけではなく，より一般性を持たせるために，

$$atten = \frac{1}{(c_1 + c_2 \times d_L + c_3 \times d_L^2)}$$

として減衰係数を与えることが多く，VRML2.0 でもこの式の値と 1 の最小値を減衰係数として用いる．

環境光，拡散光，鏡面光の色合いは，光源の光の色に影響され $I_{prgb}$ が ambient，diffuse，specular にかけ合わされている．

## 3.1 ambient 成分 (環境光)

ambient 成分は，

$$ambient = I_a \times I_{prgb} \times O_{drgb} \times O_a$$

で計算され，光源の光（$I_{prgb}$）にその ambientIntensity $I_a$ をかけ，さらに物体の材質やテクスチャで与えられる色（$O_{drgb}$）に材質の ambienIntensity $O_a$ をかけている．この式からわかるように，環境光は立体の法線方向には依存せず，どの方向を向いた面でも同じである．

### 3.2 diffuse 成分 (拡散光)

diffuse 成分は，面の法線ベクトル $N$ と輝度を計算すべき点から光源への方向ベクトル $L$ の相対的な関係に依存し，

$$diffuse = k_d \times O_{drgb} \times (N \cdot L)$$

として計算される．光源の強度（$k_d$）と材質やテクスチャで与えられる色 ($O_{drgb}$) をかけ合わせるとともに，$N$ と $L$ の内積をかけている．$N$ と $L$ が一致している場合最大となり，2 つのベクトルのなす角度が大きくなるにつれて角度の cos に比例して小さな値となる．角度が 90° を超えた場合には 0 となる．

### 3.3 specular 成分 (鏡面光)

specular 成分の式は少し複雑ですが，その意味するところを考えてみると，specular 成分は，

$$specular = k_a \times O_{argb} \times (N \cdot \frac{(L+V)}{|L+V|})^{shininess \times 128} \quad (3.3.1)$$

と計算される．光源の強度（$k_a$ : VRML2.0 では $k_d$ と一致）と材質やテクスチャで与えられる鏡面光の色（$O_{srgb}$ : $O_{drgb}$ とは独立に指定できる）をかけ合わせるとともに，$L$ と $V$ との和を単位ベクトルとして得られるハーフウエイ（halfway）ベクトル $H$ と法線ベクトルの内積を掛け合わせる．$H$ を使って（3.3.1）の式を書き換え，さらに角度 $\beta$ を使うと，

$$\begin{aligned} specular &= k_s \times O_{srgb} \times (N \cdot H)^{shininess \times 128} \\ &= k_s \times O_{srgb} \times (cos\beta)^n \end{aligned} \quad (3.3.2)$$

となる．ここで，n=shininess × 128.

鏡面光の性質は，光が反射される方向 $R$ と視点への方向ベクトル $V$ とのなす角の cos の n 乗は 0 から 1 の値に正規化されており，n は通常 1 から 100 程度に比例すると考えるのが一般的である．したがって，

$$specular = k_s \times O_{srgb} \times cos^n \alpha$$

となる．$cos\alpha$ をベクトルを使って表すと，

$$\begin{aligned} cos\alpha &= R \cdot V \\ &= (2N\ (N \cdot L)\ -\ L) \cdot V \end{aligned}$$

となる．ただ，この式は光源と視点が無限大にある場合効率が $H$ を使った場合に比べて悪く（3.3.2）式が採用されている．$H$ は鏡面光が最大となる法線方向であり $\beta = 0$ のとき $\alpha$ も 0 となる．ただし，厳密には $\beta$ と $\alpha$ は一致しない．

## 4 VRML による 3D ガイド

新築された 8 号館のガイドホームページを完成させた．全体を一つのソースで作るのは困難であるので，各階を分けて作った．そして，各階の行き来は，階段，エレベータを使いリンクを張るようにした．それに伴いクリックをした場所に対応した位置が求められる．解決するために，その場その場に応じて視点を変えたソースが必要となる．例えば，2 階の場合，階段が 5 箇所，エレベータ，外からの出入りがそれぞれ 1 箇所あり，合計 7 つのソースが必要となる．他の階も同様である．

また，壁などはデジタルカメラで撮り，本物の写真を入れる事により，描くよりもユーザーに分り易くした．

さらに，2 次元ではなく，3 次元の 8 号館を作ろうと思ったのも，ユーザーが，8 号館に行く前に見るものなので，より実際のものに近づけるためである．

ソース

| 場所 | 外観 | 地下 1 階 | 1 階 |
|---|---|---|---|
| 行数 | 約 400 行 | 約 500 行 | 約 1000 行 |

| 2 階 | 3 階 | 4 階 | 5 階 |
|---|---|---|---|
| 約 1000 行 | 約 500 行 | 約 500 行 | 約 350 行 |

## 5 結論

本研究では新築の 8 号館のガイドホームページを作成した．8 号館を作成するにあたり、平面図、断面図、立面図を用いて設計図を完成させた。[5] こ

の VRML は Web 上の，http://www.am.chs.nihon-u.ac.jp/~yaku/archive/thesis/a5497031-vrml で見ることが出来る．これにより，来校者がインターネットで 8 号館のつくりを調べてから，来校出来るようになった．今後は，スムーズな操作を可能とする事，視点を幾つも設定出来て選択可能とする事が課題である．

## 参考文献

[1] 山本精一,『VRML2.0　パーフェクトガイド』, 技術評論社,（1996）,287

[2] 三浦憲次郎,『VRML2.0　3D サイバースペース構築言語』, 朝倉書店,( 1996）,272

[3] 河西朝雄, 河西雄一,『VRML2.0』, ナツメ社,(1998) ,198

[4] 高橋嘉一,WWW 上での 3 次元空間表現, 日本大学文理学部 応用数学科 卒業研究報告書,（1999）

[5] 入江三宅設計事務所, 日本大学文理学部 8 号館 1 階平面図，断面図，立面図