卒業論文

3DCGソフトウェアMayaを用いた格子モデル視覚化システムの改善

関西学院大学理工学部
情報科学科 6611 正木佳宏

2010年3月

指導教員　　西谷滋人　教授

概 要

現実の物理現象において理論，数式を考案するには，モデル化が重要となるが，模式的なモデルだけではその理解が困難な場合がある．西谷研究室で取り扱っている物理現象として，析出現象の自由エネルギー計算，SiC の液相成長，変態などがある．これらを視覚化させることは，初学者が研究を理解するのに有効な手段となり得る．一昨年，その手段として西谷研究室の学生が 3DCG 作成ソフトである Maya を用いて，格子モデルを視覚化させるシステムを開発した．しかし，そのシステムの使用にはプログラミング言語 Ruby の学習と経験が必要になり，誰もが容易に使える仕様となっていない．実際，開発者以外の研究生は使用できていないというのが現状である．本研究では，そのような状況を打開するため，誰もが直感的に使用できるように視覚化システムの改善することを目的とした．

従来の視覚化システムでは，視覚化を行うインターフェースとして，Ruby で記述するスクリプトと，Maya とで構成されていた．本研究においても，この雛形を基に改善を行った．改善後の具体的なシステムの構成として，外装ではキーワードで構成された簡易なインプットファイルによるインターフェースを実装させ，それを読み込ませることでモデルの視覚化を行った．また Maya 上へ表示されたモデルを GUI を介してカスタマイズできるようにした．システムの内装では，ライブラリの中身が関数のみで複雑で構成が階層化されていないという欠点があり，それを緩和させるためにライブラリのクラス化を行った．

結果として，数行となったインプットファイルによるユーザインタフェースは，ユーザの操作量を大幅に減らした．さらに Maya 上に GUI を実装したことで，視覚化のサイクルを極端に短くすることができた．またライブラリのクラス化を行ったことで，乱雑であったシステム構成はオブジェクト指向の概念をもとに整理された．このことで機能の追加，変更が容易になった．視覚化システムは，より直感的な操作が可能となり，開発者以外の研究生でもシステムを使いこなせるようになった．その例として第 5 章で，私でない西谷研究室の学生が作成したモデルを紹介する．

目 次

第1章　序論

1.1　従来の視覚化システム

西谷研究室で取り扱っている物理現象として，析出現象の自由エネルギー計算や変態などがある．実際，Si に不純物 O を入れる前と後での構造の変化であったり，SiGe の組成比を変化させた場合の構造の変化などが挙げられる．しかし，析出現象を理解するためには，初期状態からどのような過程を経て，終状態となるのかといったことを動力学として考える必要がある．この入り口で初学者がこれらを視覚化し，イメージすることができれば研究を理解するのに有効な手段となり得る．そこで，理論や数式が自然界のどのような現象を解するのかを直感的に理解するために，西谷研究室の学生が Maya を用いて，物理現象を視覚化させるシステムを開発した．このシステムは析出現象の理解や変態などの物理現象を視覚的に比較，確認でき，研究を理解する上で，非常に役に立つ．しかし，開発者以外の学生は使いこなせないという欠点が存在する．この状況を打開するため，研究室の学生誰もが容易に使える柔軟なシステムを構築することを目的とする．

1.2　結晶構造

SiC

シリコンカーバイド（SiC）はその性質から次世代パワーでバイスとして注目されており，各所で積極的に研究が進められている．関西学院大学においても理工学部の金子教授らによって SiC の新奇な結晶成長プロセスである準安定溶媒エピキタシー（Metasteble solvent Epitaxy:）が発明されており，同大学の西谷研究室でも SiC の研究が行われている．MSE では，基盤と原料版に用いる SiC の結晶多形（ポリタイプ）の違いによって生じる化学ポテンシャルの差が動力となっている [1]．このように結晶構造の違いは物性を支配している 1 つの要素になっているため，その理解が材料研究の出発点になる．

SiO_2

主要なシリコン (Si) 単結晶育成法であるチョコラルスキー法 (Czochralski 法 : CZ 法) では，多結晶 Si を石英 (SiO_2) 坩堝で融液化させるため，融液中に酸素原

子が流れ込む．CZ 法でひきあげられた Si 単結晶には，その酸素原子が飽和中に取り込まれることになる．これは，その後のデバイス作成工程，特性に大きな影響を与えるため，酸素の挙動の制御が求められる．デバイス中のシリコンと酸素の関わりを解明するため，凝集の極限とも言える (SiO_2) 結晶についての理解が求められる．

SiGe

Si を歪ませるために主としてシリコンゲルマニウム（SiGe）が代表的な不規則性混合物として知られている．しかし，その局所的な規則性は未だ解明されていない．結晶性の良い歪み Si の生成には同様に結晶性の良い SiGe 基盤が必要となるため m，歪み Si の応用において，SiGe の規則性に関する研究が重要視されている．

格子モデル

結晶構造を視覚化するには，原子を ball，原子間の結合を stick，で表した格子モデルを作成するのが一般的である．その種類は表（1.1）のようになる．

表 1.1: 格子モデル表現法一覧

ball 表示	原子のみ表示する方法
stick 表示	結合のみ表示する方法
ball&stick 表示	原子と結合の両方を表示する方法

第2章　手法

物理モデルを視覚化させるシステムとして，[Ruby - MEL - Maya] を紹介する．

2.1　Maya

Maya は，オープンアーキテクチャを基板とした強力な統合型の 3D モデリング，アニメーション，レンダリングソリューションである．多くのフィルムやビデオアーティスト，ゲーム開発者，マルチメディアデザイナー，3DCG に関わる SOHO デザイナーなどが使用しているプロ仕様のハイエンドソフトである．グラフィックス性に優れ，非常に高度な機能を有しているため，自由度も高い．また，Maya はその GUI の全てを MEL というスクリプト言語で実行可能となっている．本研究では，実際に視覚化を行うインターフェースとして Maya を選定した．

MEL

MEL(Maya Embedded Lanquage) とは Maya で使用できるスクリプト言語であり，Maya の GUI の機能の全てをまかなうことが可能となっている．MEL のみで CG の作成，カスタマイズを行うことができるが，Maya 専用のスクリプトエディタ上でしか実行できない．また，線形計算などの数値計算に向いておらず，構文も複雑な構成となっている．[5]

2.2　Ruby

Ruby とはまつもとゆきひろ氏により開発されたオブジェクト指向スクリプト言語である．一般的に，Ruby は数値計算には向いていないと言われる．実際に，Ruby は C や Fortan などの他の言語と比べ，実行速度などの面で遅い．しかし，Ruby の言語は，単純な構文で書かれており，可読性に優れている．さらにコンパイルを必要としないインプリンタ方式を採用しているため，実行の手間が少ない．加えて，高機能なスクリーンエディタとして有名な Emacs と併用すれば，構文に応じてスクリプトが色分けされるので．開発環境が非常に良いものとなる．本研究では，以上のことからインターフェースとして Ruby を選定した．

2.3　従来の視覚化システムの構成

従来の視覚化システム Ruby - MEL - Maya の実行の流れを図 2.1 に示す．先に記したように，視覚化を実現させるツールとして，Ruby，Maya を使用した．まず，ユーザが数値計算を行うメインスクリプトを Ruby にて作成，実行する．そのデータを反映させた MEL スクリプトを出力させる．そして，Maya のインターフェースで生成した MEL スクリプトを読み込むことで視覚化を実現させる．また，この作業を効率良く行うため，視覚化の目的に応じてカスタマイズを行う関数のライブラリがある [6].

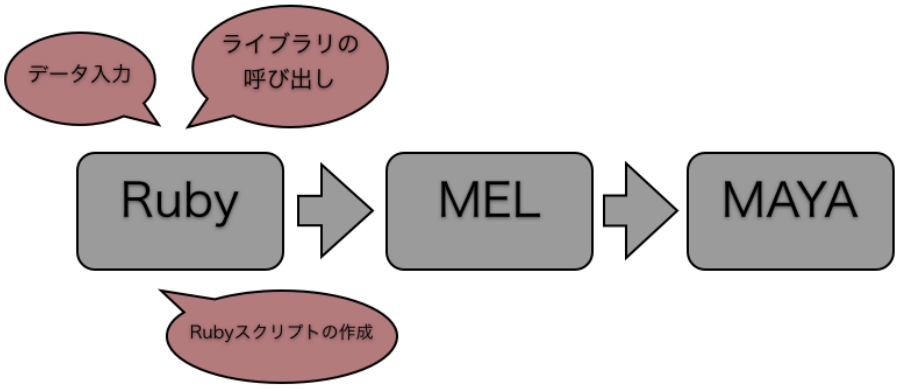


図 2.1: 従来のシステムの流れ

しかし，従来のシステムには多くのデメリットが存在する．

- ライブラリに用意されている描画関数が等レベルで階層化されていない．
- ユーザが作成するメインスクリプトが冗長になっている．
- モデルのカスタマイズをプログラミング言語で行うため，直感的なユーザインタフェースとなっていない．
- モデルの更新をする度，メインスクリプトを変更させて，図 2.1 の工程を行うので，手間が多い．

上記の状況を打開するため，本研究の手法を以下に示す．

- 現在のライブラリの複雑さを緩和させるためにライブラリのクラス分けによって適切な階層構造を作成する．
- 誰もが直感的にシステムを使用できるように，簡易なインプットファイルによるインタフェースを実装させる．
- Maya でのインタラクションを可能にする GUI を作成する．

2.4 デザインパターン

書籍「オブジェクト指向における再利用のためのデザインパターン」において，GoF と呼ばれる 4 人の共著者は，デザインパターンという用語を初めてソフトウェア開発に導入した．GoF はエーリヒ・ガンマ，リチャード・ヘルム，ラルフ・ジョンソン，ジョン・ブリディーズの 4 人である．

コンピュータのプログラミングで，素人と達人の間ではびっくりするほどの生産性の差があるが，その差はかなりの部分が経験違いからきている．達人は，さまざまな難局を，何度も何度も耐え忍んで乗り切ってきている．そのような達人達が同じ問題に取り組んだ場合，典型的にはみな同じパターンの解決策にたどり着くのだが，これがデザインパターンである．

それぞれのパターンは，プログラマの間で何度も繰り返し考えだされてきた．したがって，それは最善の解決策ではないかもしれないが，その種の問題に対するトレードオフも考慮した，典型的な解決策ではある [2]．

以下に示す 2 つのパターンは従来のライブラリのクラス分けを行う際の考え方として用いた．

Template Method パターン

Template Method パターンは GoF によって定義されたデザインパターンの 1 つであり，その目的は骨格となるメソッドを持った抽象基底クラスを構築することである．この骨格となるメソッドは抽象メソッドを呼ぶことによって変更に対応するような処理を扱う．抽象メソッドが呼び出されると，その実際の処理は具象サブクラスが提供するこになる．様々な具象クラスの中から 1 つを選ぶことで，必要とする処理のバリエーションを選択することができる．結果的に変わらないもの（テンプレートメソッドに記述された基本的なアルゴリズム）と変わるもの（サブクラスで提供される詳細な処理）を分離できる [3].

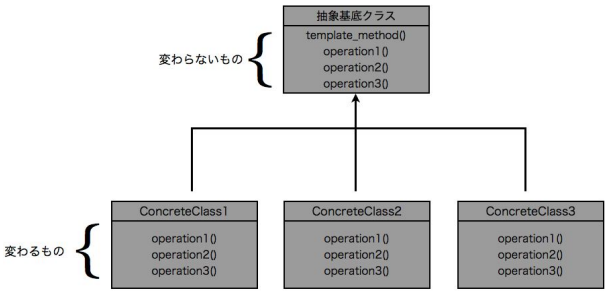


図 2.2: Template Method の考え方

Strategy パターン

Strategy パターンも Template Method パターンと同様，GoF によって定義されたデザインパターンの 1 つである．アルゴリズム中の変わる部分を抜き出してサブクラスへと押し込む代わりに，アルゴリズムのパターンごとにばらばらのオブジェクトとしてシンプルに実装する．異なるストラテジオブジェクトをコンテキスト（利用者）に対して提供することで，アルゴリズムに多様性をもたらすことができる [4].

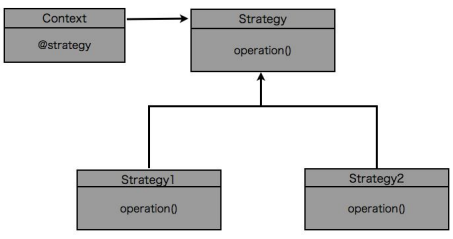


図 2.3: Strategy の考え方

第3章　現在の視覚化のシステム

本章では，[Ruby - MEL - Maya] のシステムの実装，使用法を解説する．

3.1　データディレクトリ

格子モデルを作成するのに必要な原子位置を示すデータを格納しておく．格子モデルを作成するパラメータとして，結晶の並進対称性を規格化する基本並進ベクトル（Primitive Vector）と，それを元に原子位置を示したベクトル（Basis Vector）が必要になる．Primitive Vector を式 (3.1)，Basis Vector を式 (3.2) とした場合，実空間における原子位置（X,Y,Z）は式 (3.3) で表される．

$$P_0 = \begin{bmatrix} p_{0,0} \\ p_{0,1} \\ p_{0,2} \end{bmatrix}, P_1 = \begin{bmatrix} p_{1,0} \\ p_{1,1} \\ p_{1,2} \end{bmatrix}, P_2 = \begin{bmatrix} p_{2,0} \\ p_{2,1} \\ p_{2,2} \end{bmatrix} \tag{3.1}$$

$$B = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.2}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = P_0 x + P_1 y + P_2 z \tag{3.3}$$

以下の小説にデータソースとその形式を紹介する．なお，データのサンプルはいずれも 4H-SiC としている．基本的にデータをコピーしたファイルを作成し，そのファイル名が入力ファイルの DataFile のタグ部分にあたる．

3.1.1 web データ

データソースは結晶構造のデータを公開している 'Crystal Lattice Structures' という web ページである [7].
また，構文解析時のキーワードは'Primitive'，'Basis' となっている.

web データ

```
4H-SiC

Primitive vectors
a(1) =   1.54025500 -2.66779992  0.00000000
a(2) =   1.54025500  2.66779992  0.00000000
a(3) =   0.00000000  0.00000000 10.08480000


Volume =  82.87874525


Reciprocal vectors
b(1) =   0.32462157 -0.18742035  0.00000000
b(2) =   0.32462157  0.18742035  0.00000000
b(3) =   0.00000000  0.00000000  0.09915913


Basis Vectors:
Atom    Lattice Coordinates               Cartesian Coordinates

Si  0.00000000  0.00000000  0.00000000     0.00000000  0.00000000  0.00000000
Si  0.00000000  0.00000000  0.50000000     0.00000000  0.00000000  5.04240000
C   0.00000000  0.00000000  0.18750000     0.00000000  0.00000000  1.89090000
C   0.00000000  0.00000000  0.68750000     0.00000000  0.00000000  6.93330000
Si  0.33333333  0.66666667  0.24982500     1.54025500  0.88926664  2.51943516
Si  0.66666667  0.33333333  0.74982500     1.54025500 -0.88926664  7.56183516
C   0.33333333  0.66666667  0.43732500     1.54025500  0.88926664  4.41033516
C   0.66666667  0.33333333  0.93732500     1.54025500 -0.88926664  9.45273516
```

3.1.2 POSCAR

データソースは第一原理計算ソフト VASP の原子位置入力ファイルとなる POSCAR である．POSCAR には初期構造の Primitive Vector と Basis Vector のデータが書かれている．これらのデータはモデルを視覚化する上で必須である．構文解析時のキーワードは'Direct' となっている．

POSCAR

```
4H-SiC_unitcell_fix_newdata_toga

   1.000000000000000
    3.0800000000000000    0.0000000000000000    0.0000000000000000
   -1.5399999300000000    2.6673582800000000    0.0000000000000000
    0.0000001800000000    0.0000003200000000   10.0810000000000000

Si C 4 4
Direct

 0.0000000000000000  0.0000000000000000  0.0000000000000000
 0.0000000000000000  0.0000000000000000  0.5000000000000000
 0.3333333300000021  0.6666666699999979  0.2500000000000000
 0.6666666699999979  0.3333333300000021  0.7500000000000000
 0.0000000000000000  0.0000000000000000  0.1875000000000000
 0.0000000000000000  0.0000000000000000  0.6875000000000000
 0.3333333300000021  0.6666666699999979  0.4375000000000000
 0.6666666699999979  0.3333333300000021  0.9375000000000000
```

3.1.3 OUTCAR

データソースは第一原理計算ソフト VASP の出力ファイルとなる OUTCAR の一部である．OUTCAR には Primitive Vector と 原子位置位置表のデータが書かれている．これらのデータはモデルの視覚化を行う上で必須である．構文解析時のキーワードは'direct'，'POSITION' となっている．

OUTCAR

```
VOLUME and BASIS-vectors are now :

-----------------------------------------------------------------------------
 energy-cutoff  :      600.00
 volume of cell :       82.82
     direct lattice vectors                 reciprocal lattice vectors
    3.080000000  0.000000000  0.000000000     0.324675325  0.187451375 -0.000000012
   -1.539999930  2.667358280  0.000000000     0.000000000  0.374902767 -0.000000012
    0.000000180  0.000000320 10.081000000     0.000000000  0.000000000  0.099196508
POSITION                                       TOTAL-FORCE (eV/Angst)
-----------------------------------------------------------------------------------
     0.00000      0.00000      0.00000          0.000000      0.000000     -0.055940
     0.00000      0.00000      5.04050          0.000000      0.000000     -0.055940
     0.00000      1.77824      2.52025          0.000000      0.000000     -0.032100
     1.54000      0.88912      7.56075          0.000000      0.000000     -0.032100
     0.00000      0.00000      1.89019          0.000000      0.000000     -0.052808
     0.00000      0.00000      6.93069          0.000000      0.000000     -0.052808
     0.00000      1.77824      4.41044          0.000000      0.000000      0.140848
     1.54000      0.88912      9.45094          0.000000      0.000000      0.140848
-----------------------------------------------------------------------------------
   total drift:                                 -0.000009      0.000014      0.000623
```

3.2　ユーザインタフェース

入力ファイル

ユーザの望むモデルの情報を入力するテキストファイルである．それを図 3.1 に示した．ユーザはファイル名（任意），表示形態（Stick,Ball,Stick&Ball），拡張サイズ（1*1*1），ワイヤーフレームの有無 (True，False)，モデルに含まれる原子そしてその数を入力する．表示形態は小節 1.1 に記したように表示される．拡張サイズについては小節 3.5.2 で詳しい説明をしており，そちらを参考にすると良い．モデルに含まれる原子は MEL で作られるオブジェクトに名前を付けるために必要なデータである．また AtomsNumber はデータファイルの構文解析時に必要なデータである．これらのデータを記した入力ファイルを Ruby スクリプトへ読み込ませることで視覚化を実現させる．
また，構文解析のキーワードは”DataFile”，”ViewFormatter”，”SuperCell”，”WireFrame”，”Atoms”，”AtomsNumber”となっている．

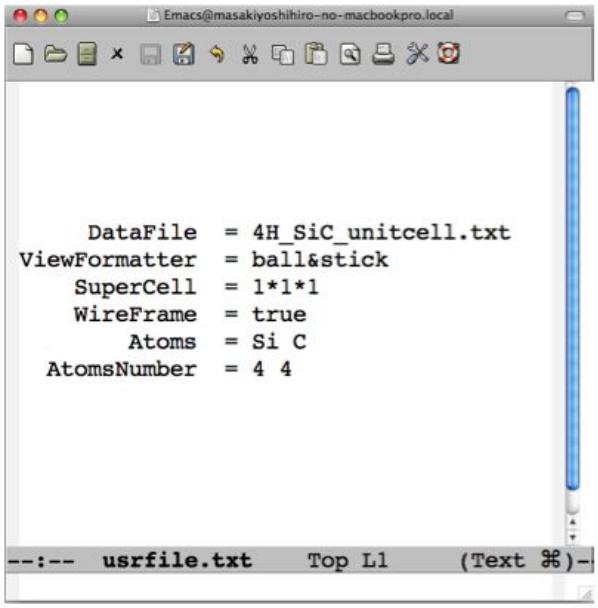


図 3.1: 入力ファイル

GUI

Maya へと表示されたモデルを GUI を介してカスタマイズすることができる．その GUI を図 3.2 に示す．図 3.3 のように原子の大きさをカスタマイズする Scale スライダーと，原子や結合部分の色，それらの透明度をカスタマイズする Color スライダーを用意した．それぞれのスライダー値を変更したいときは，スライダーバーをドラックまたは入力欄に直接入力し，OK ボタンを押すと Maya へ反映される．色の変更は R，G，B 値で調節を行う．GUI によるインタラクションを可能にすることによって，よりユーザが直感的に操作を行えるようになった．

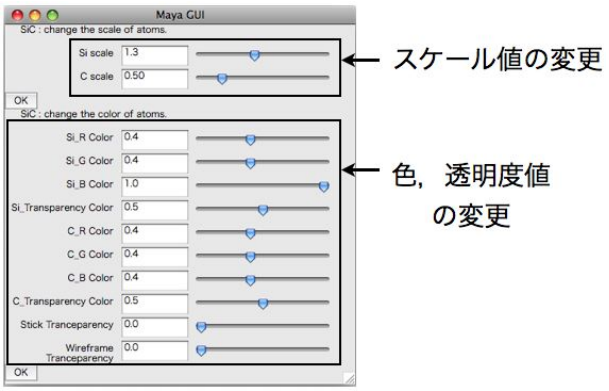


図 3.2: GUI

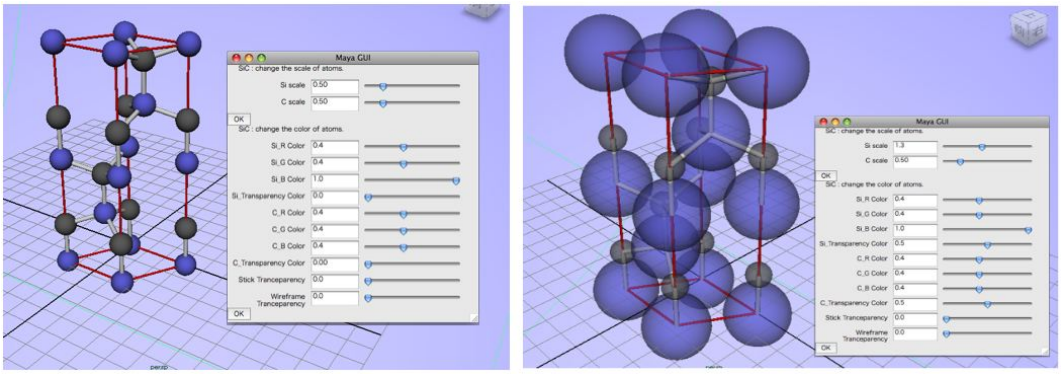


図 3.3: (a) スケール値 0.5 透明度 0.0 (b): Si スケール値 1.3 C スケール値 0.5 Si 透明度 0.5 C スケール値 0.5

3.3 クラスライブラリ

インプットクラス

インプットクラスは，Strategy の考え方を参考に POSCAR，OUTCAR の構文解析を行うコードブロックを作成した．アルゴリズム中の変わる部分を抜き出してサブクラスへと押し込む代わりに，アルゴリズムのパターンごとにばらばらのオブジェクト（POSCAR，OUTCAR）としてシンプルに実装します．異なるストラテジオブジェクトをコンテキスト（利用者）に対して提供することで，アルゴリズムに多様性をもたらすことができる．POSCAR，OUTCAR ファイルの中に書かれている Premitive Vector，Basis Vector，原子位置座標を返す．

Ruby の記述例

```
if inputfile.include?("Direct") then #POSCAR 判別
a.input(POSCAR)
elsif inputfile.include?("POSITION") then #OUTCAR 判別
a.input(OUTCAR)
end
```

`#inputfile` は，読み込んだデータファイルの中身を格納している変数である．

Ball，Stick，Stick&Ball クラス

Ball クラス，Stick クラス，Stick&Ball クラスを Template Method の考え方を参考に作成した．その階層構造を図 3.3 に示す．その目的は骨格となるメソッドを持った抽象基底クラスを構築することである．この骨格となるメソッドは抽象メソッドを呼ぶことによって変更に対応するような処理を扱う．抽象メソッドが呼び出されると，その実際の処理は具象サブクラス（Ball クラス，Stick クラス，Stick&Ball クラス）が提供するこになる．様々な具象サブクラスの中から 1 つを選ぶことで，必要とする処理のバリエーションを選択することができる．結果的に変わらないもの（テンプレートメソッドに記述された基本的なアルゴリズム）と変わるもの（サブクラスで提供される詳細な処理）を分離できる．

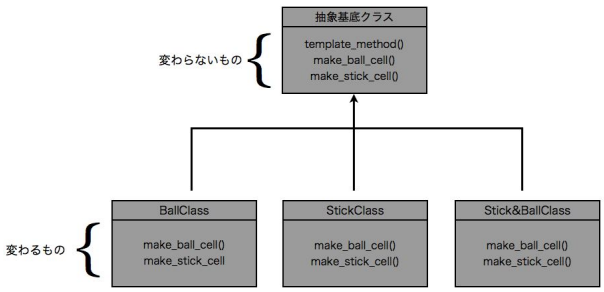


図 3.4: Template Method の考えたを利用した階層構造

3.4 関数ライブラリ

視覚化させる格子モデルをカスタマイズする上で，用途に応じた構文を関数として作成し，ライブラリとして格納した．その種類として，格子モデルの拡張や作成を行う関数，オブジェクトの作成，オブジェクトのスケールの変更，回転，色の作成を行う関数などがある．上記の 3 つのクラスに共通する関数はモジュールとして定義してある．

3.4.1 オブジェクト作成

make ball cell 関数

make_ball_cell 関数を作成した．これは，小節 1.1 で記した ball 表示法になる．実空間上の原子位置ベクトル配列，原子の種類名，色の変数と ball の scale 値を引数とし，ball の格子モデルを作成する MEL スクリプトを返す．

Ruby の記述例

```
make_ball_cell(@position[@name[0]],@name[0],@cellcolor[0],[0.5,0.5,0.5])
make_ball_cell(@position[@name[1]],@name[1],@cellcolor[1],[0.5,0.5,0.5])


#@postion[@name[0]], @postion[@name[0]] は原子の種類とその原子座標を格納しているハッシュである．
#@name[0], @name[1] は原子の種類を格納している配列である．
#@cellcolor[0], @ cellcolor[1] は原子の色を格納している配列である．
#[0.5,0.5,0.5] は原子のスケール値である．
```

make stick cell 関数

make_stick_cell 関数を作成した．これは，小節 1.1 で記した stick 表示法になる．結合させる 2 種類の原子位置ベクトル配列，ボンドの長さの上限下限，色の変数と stick の scale 値を引数とし，stick の格子モデルを作成する MEL スクリプトを返す．

Ruby の記述例

```
len=bond_length(@position[@name[0]],@position[@name[1]])
make_stick_cell(@position[@name[0]],@position[@name[0]],
[0,len+0.1],@stickcolor[count],[2.0,0.1,0.1])


#len は bond_length 関数によって原子と原子の最小の長さが返された値を格納している.
#[0, len+0.1] 0 <= 原子間距離 < len+0.1 のとき, stick を作成する.
#@stickcolor[count] は stick の色になる.
#[2.0, 0.1 0.1] は stick の scale である.
```

基本的な MEL スクリプト

sphere の作成

MEL の記述例

```
sphere;
scale $a $b $c;
move $x $y $z;


#ball オブジェクトを作成する際, MEL の sphere コマンドを使用する.
#同じく, scale コマンドでオブジェクトのスケールを変更する.
#move コマンドでオブジェクトの移動を行う.
#rotate コマンドでオブジェクトの角度を変更する.
#$a, $b, $c, $x, $y, $z には数値を入力する.
```

cylinder の作成

MEL の記述例

```
cylinder;
scale $a $b $c;
move $x $y $z;
rotate $α $β $γ;


#stick オブジェクトを作成する際, MEL の cylinder コマンドを使用する.
#同じく, scale コマンドでオブジェクトのスケールの変更,
#move コマンドでオブジェクトの移動を行う.
#$a, $b, $c, $x, $y, $z, $α, $β, $γ には数値を入力する.
```

表 3.1: MEL オブジェクト一覧

オブジェクトの種類の入力名	作成されるオブジェクト
sphere	球
cylinder	円柱
cone	円錐
nurbsCube	直方体

3.4.2 格子モデルの拡張

supercell

supercell には，格子モデルの拡張を行う supercell 関数がある．拡張サイズを引数とし，拡張させた原子位置ベクトルを返す．格子モデルの拡張の仕方を図 3.3 に示す．拡張サイズの変更は入力ファイルから行う．拡張サイズの引数の形式として，[x, y, z] の配列とする．この x, y, z はそれぞれ，インプットクラスで求められた p_vec[0], p_vec[1], p_vec[2] 方向の拡張サイズとなる．(a) は，拡張を行っていない，すなわち [x, y, z]=[1, 1, 1] の状態である．入力ファイルでは 1*1*1 と同じである．また，拡張サイズを 2*2*2 とすると，(b) のように拡張され，3*3*3 にすると (c) のように拡張される．図 3.4 は見やすさを考慮したため，2 次元方向のみ拡張させた．

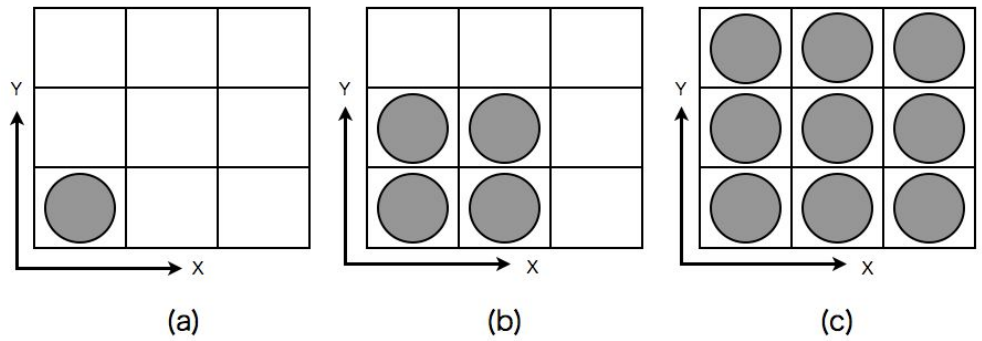


図 3.5: 格子モデルの拡張の模式図

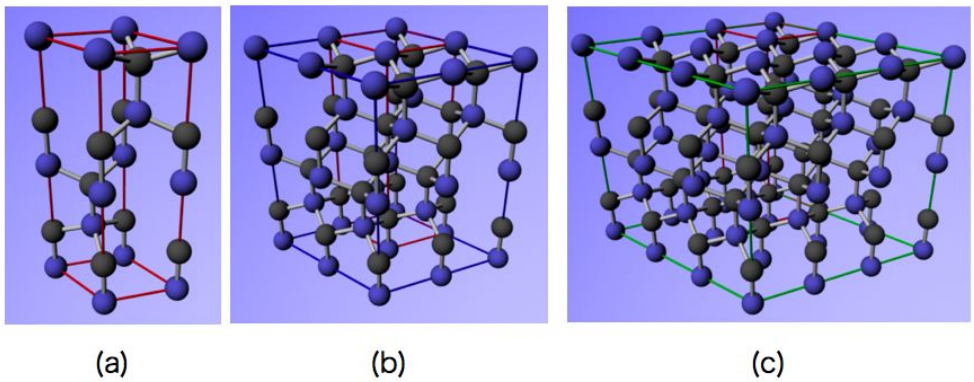


図 3.6: (a):3C-SiC,(b):4H-SiC,(c):6H-SiC の拡張後のモデル

color

color には，色を作成する color_lambert 関数，color_incandescence 関数がある．color_lambert 関数は色を指定する RGB 値を引数とし，色を指定する MEL スクリプトを返す．color_lambert 関数は色を作成するときに用いる関数なので，最も使用頻度が高い．color_incandescence 関数は色を光らせたいときに用いる関数である．

MEL の記述例

```
@cellcolor=[color_lambert([0.4,1.0,0.4]),color_lambert([1.0,1.0,0.4])]
@stickcolor=[color_lambert([1.0,1.0,1.0]),color_lambert([1.0,1.0,0.9])]
@wirecolor=[color_lambert([1.0,0.0,0.0]),color_lambert([0.0,0.0,1.0])]

#@cellcolor，@stickcolor，@wirecolor は色を格納している配列でこれらは，先に示した
make_ball_cell 関数，make_stick_cell 関数の引数となる．
```

wireframe

wireframe には格子モデルのユニットセルをワイヤーフレームで囲う wireframe_unitcell 関数がある．基本並進ベクトルと色の変数を引数とし，ユニットセルを囲うワイヤーフレームを作成する MEL スクリプトを返す．

MEL の記述例

```
wireframe_unitcell(@p_vec,@wirecolor[count])


#入力ファイルでキーワード ”WireFrame ”に対して，”true ”と入力した場合，呼び出すよ
うにしている．
```

unitcell

unitcell は，周期境界条件を考慮して格子モデルのユニットセルを表示させる関数である．周期境界条件を考慮した原子位置座標を返す．

MEL の記述例

```
unitcell


#入力ファイルでキーワード ”WireFrame ”に対して，”true ”と入力した場合，呼び出す関数である．
```

object

object には，オブジェクトの種類を決定する object_add 関数，オブジェクトを作成する make_object 関数，オブジェクトを移動させる move_object 関数，オブジェクトのスケールを変更する scale_object 関数，オブジェクトを回転させる rotate_rotate 関数がある．object_add 関数はオブジェクトの種類，オブジェクトにつける任意の名前と色の変数を引数とし，任意のオブジェクトを作成する MEL スクリプトを返す．それ以外の関数はオブジェクトの名前もしくは色の変数と変更値を引数として，オブジェクトを変更する MEL スクリプトを返す．なお，作成されたオブジェクトは位置 [0,0,0]，スケール [1,1,1]，回転 [0,0,0] がデフォルトとなっている．これらは，面の表示を行う際に用いられる関数である．

Ruby の記述例

```
name = 'add_atom'
object_add('nurbsCube',name,@surfacecolor[0])
move_object('name',surface[0])
rotate_object(name,[90,0,90])
scale_object(name,[10,0,10])
```

表 3.2: 関数ライブラリ一覧

make ball cell	sphere を作成する．
make stick cell	stick を作成する．
color	色の作成，カスタマイズを行う．
wireframe	ユニットセルの枠を作成する．
unitcell	周期境界条件を考慮したユニットセルを作成する．
supercell	格子モデルの拡張を行う．

3.5　MELの出力

Ruby スクリプトからの出力ファイルとなる．以下に Maya への出力方法を記述する．

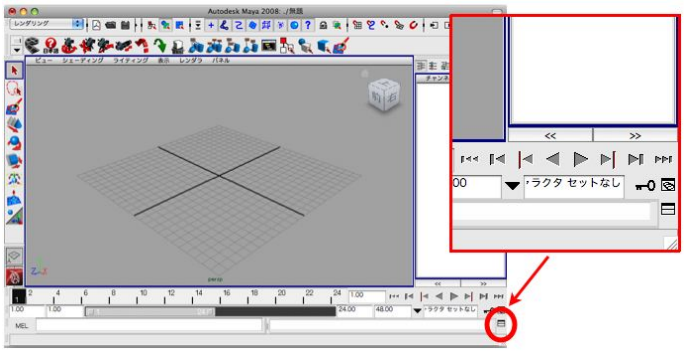
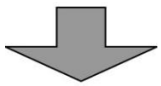


図 3.7: Maya インタフェース



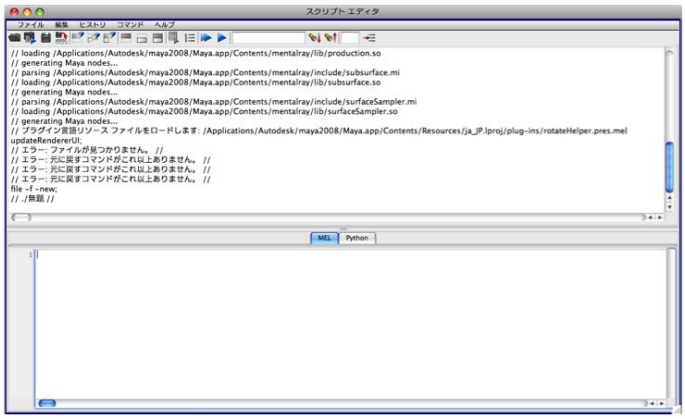


図 3.8: Maya スクリプトエディタ

MEL スクリプトの出力には，まず Maya のスクリプトエディタを開く必要がある．図 3.3 の円で囲んだ箇所をクリックすると図 3.4 にある Maya のスクリプトエディタが表示される．

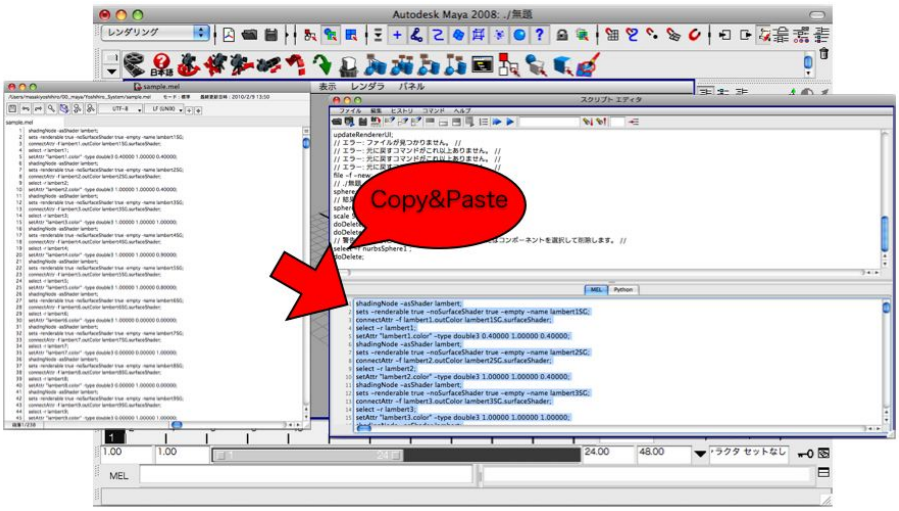


図 3.9: MEL の実行

図 3.4 のように ’mi ’のようなテキストディタで開いた MEL スクリプトを MEL スクリプトエディタに貼付ける．MEL スクリプトを図 3.4 のように全選択した状態で，キーボードより ’Control + enter ’ を押す．
最終的に図 3.6，図 3.7 の酔うな出力が得られる．オブジェクトの線表示とカラー表示はキーボードの ’5 ’のキーで変更できる．

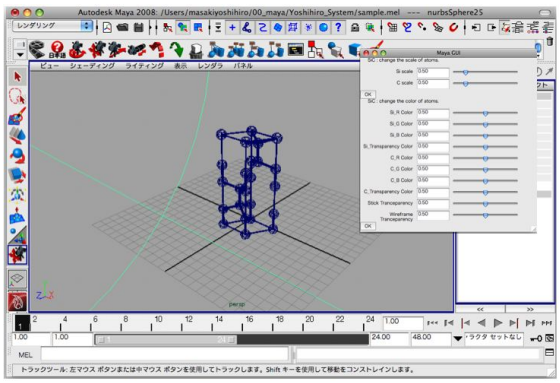


図 3.10: 出力結果（線表示）

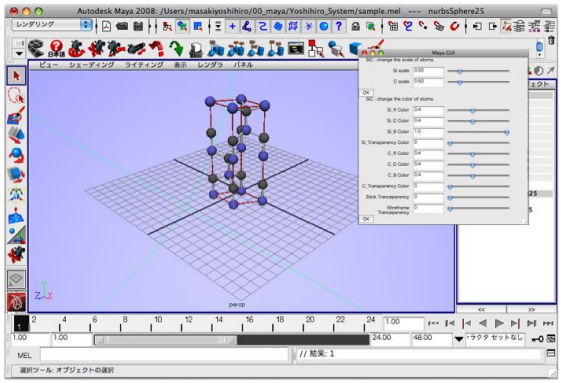


図 3.11: 出力結果（カラー表示）

3.6　現在の視覚化システムの構成

新しいシステムの流れを図 3.11 に示す．まず，ユーザは用意された入力ファイルに必要なデータを入力する．その後，ターミナル上で，入力ファイルの各々のデータを Ruby で作成したメインスクリプトへ読み込ませ，自動的にそれらのデータに見合った操作のクラスを呼び出し，それによって反映された MEL スクリプトを出力させる．その際，Ruby での工程はブラックボックス化した．そして，Maya のインターフェースで生成した MEL スクリプトを読み込むことで視覚化を実現させた．また，Maya 上で GUI を実装させ，容易にモデルのカスタマイズを行うことを可能とした．

現システムは従来のシステムと比べて多くのメリットが存在する．

- ライブラリのクラス化を行い，柔軟性とシステムの開発のし易さを向上させた．
- 容易にモデルのカスタマイズができるようにユーザインタフェースとして，入力ファイルを作成した．
- GUI の実装により，視認性と操作性が優れ，直感的な操作を可能とした．
- ユーザの操作量を極端に減らすことができた．

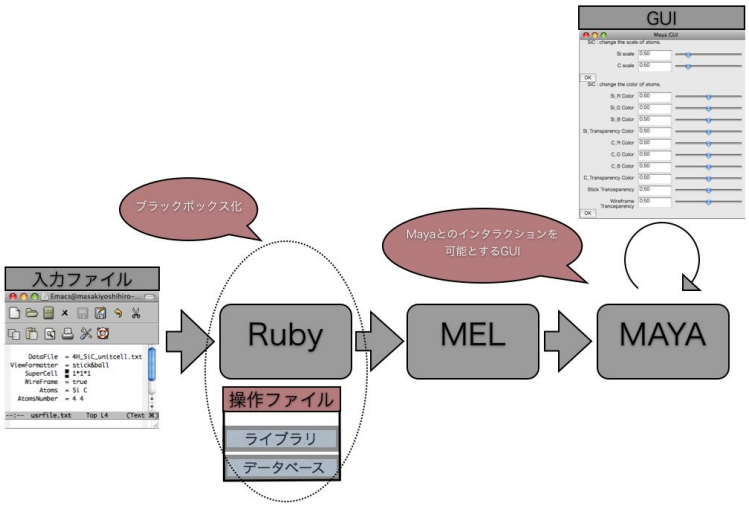


図 3.12: 現システムの流れ

第4章　従来の視覚化システムと現在の視覚化システムとの比較

4.1　システムの流れ

この章ではシステムの比較を行う．図 4.1 に旧システムの流れ，図 4.2 に現在のシステムの流れを示す．

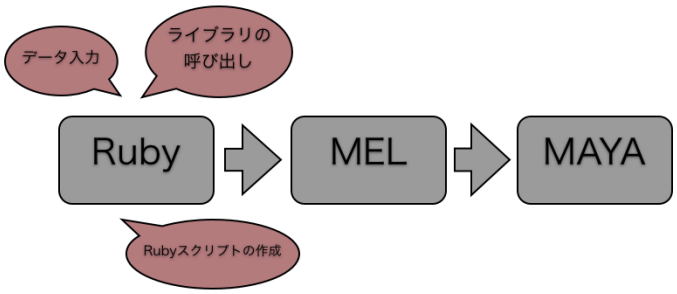


図 4.1: 従来のシステムの流れ

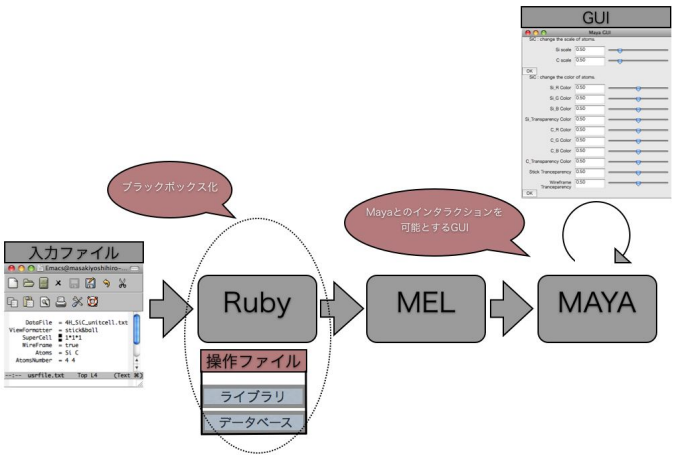


図 4.2: 現システムの流れ

4.2　メインスクリプト

旧視覚化システムと現在の視覚化システムのメインスクリプトを図3.10に示す．明らかに，(a) は冗長でモデルの更新の際にはわかりにくい．一方，(b) ではタグが用意され，直感的にわかりやすいインターフェースとなった．また，Maya へと表示されたモデルをカスタマイズする際，入力ファイルを書き替えれば視覚化を実現できる．

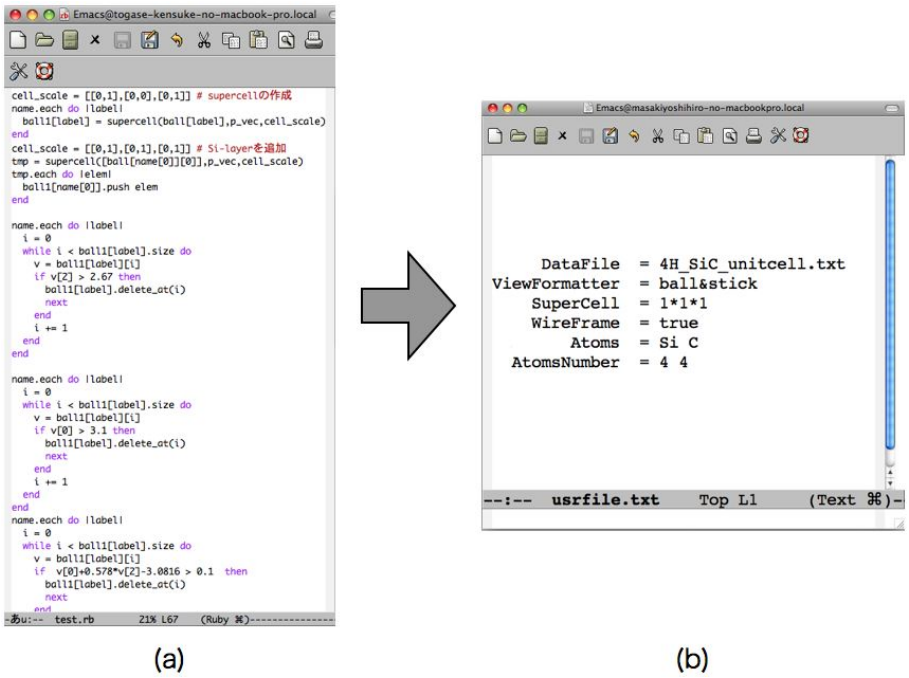


図 4.3: (a):旧視覚化システムのメインスクリプト，(b):現在の視覚化システムのメインスクリプト

4.3 GUI

また，Maya へと表示されたモデルを GUI を利用してカスタマイズもできる．GUI において，自由自在に原子の大きさ，色，その透明度を変えることができることによって柔軟な操作を実現した．また，色の透明度を変えれることは視界の妨げを防ぐことができる．これらのインタフェースの実装により，直感的な操作を可能とし，手間と時間を削減できた．

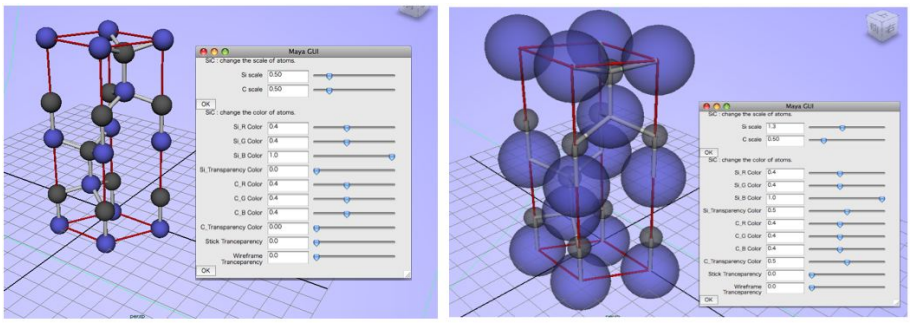


図 4.4: (a) スケール値 0.5 透明度 0.5 (b): Si スケール値 1.3 C スケール値 0.5 Si 透明度 0.5 C スケール値 0.5

第5章　格子モデルの視覚化

5.1　4H-SiCの視覚化

5.1.1　従来のシステム

SiCには，100種類以上の結晶多形が存在する．多くの結晶多形の中でも発生率が高く，応用上重要となるRamsdell notationで3C-, 4H-, 6H-SiCがある．これら結晶多形の表記法は，積層方向の単位格子に含まれるSi-C単位層の数と，結晶系の頭文字 (C：立方晶，H：六方晶) を組み合わせて表されている．代表例として，4H-SiCの視覚化を従来と現在のシステムとで比較する．
まず，図5.1のようにメインスクリプトにデータファイル名，出力ファイル名，ballやstickに関するパラメータを設定する．そして，図5.2に示したターミナルで ”ruby メインスクリプト名 ”を実行する． 実行後，図5.3のようにMayaに4H-SiCのモデルが表示される．

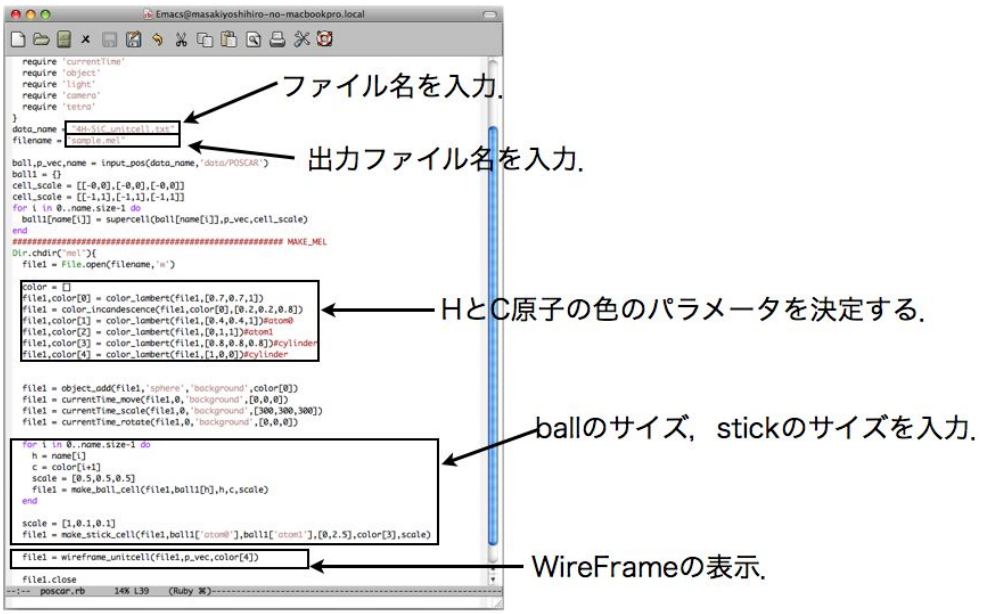


図 5.1: 4H-SiCの表示（従来のシステム）

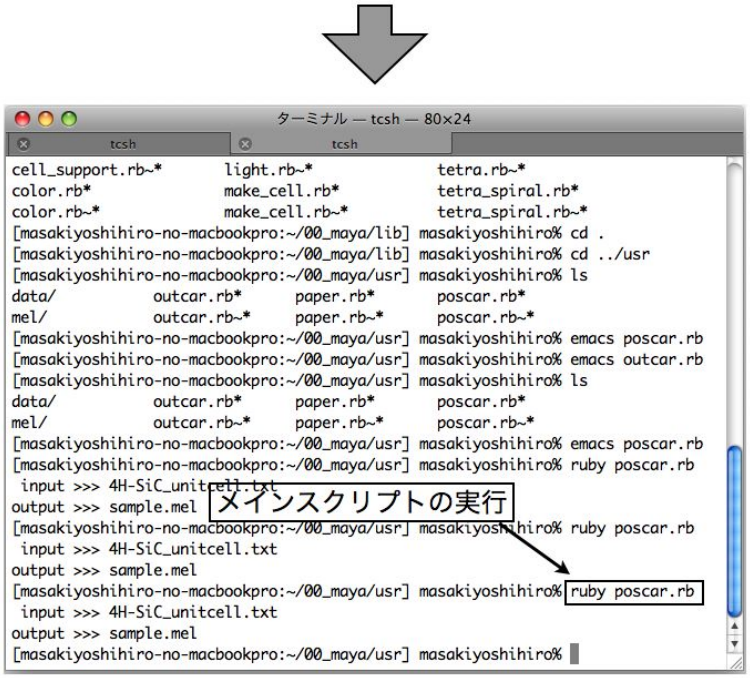


図 5.2: ターミナル

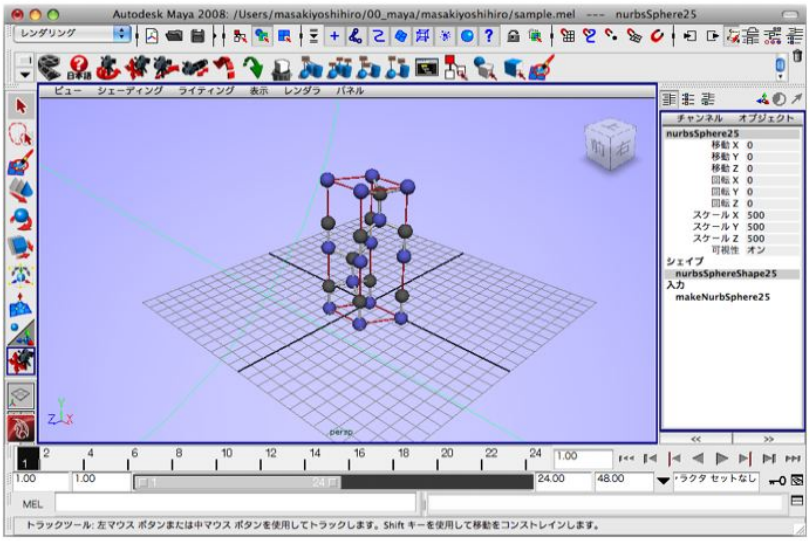


図 5.3: 4H-SiC-unitcell

5.1.2　現在のシステム

図 5.4 に示す入力ファイルに必要なデータを入力する．入力後，図 5.5 に示したターミナルで ”masaki 入力ファイル名 出力ファイル名 ”と入力し，実行する．実行後，出力ファイル名を開くと図 5.6 の用に Maya へ表示される．

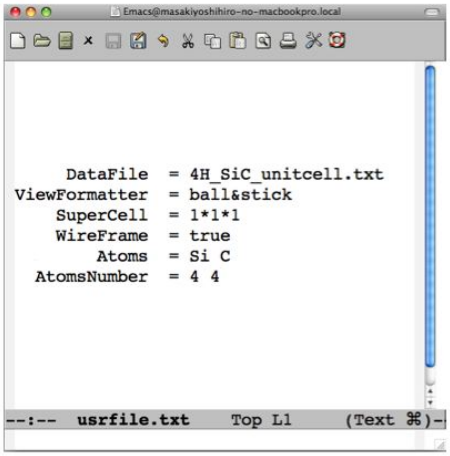


図 5.4: 入力ファイル

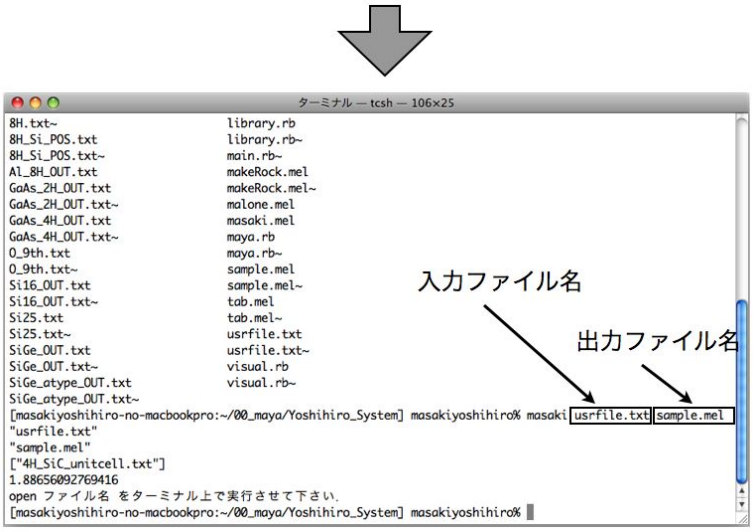
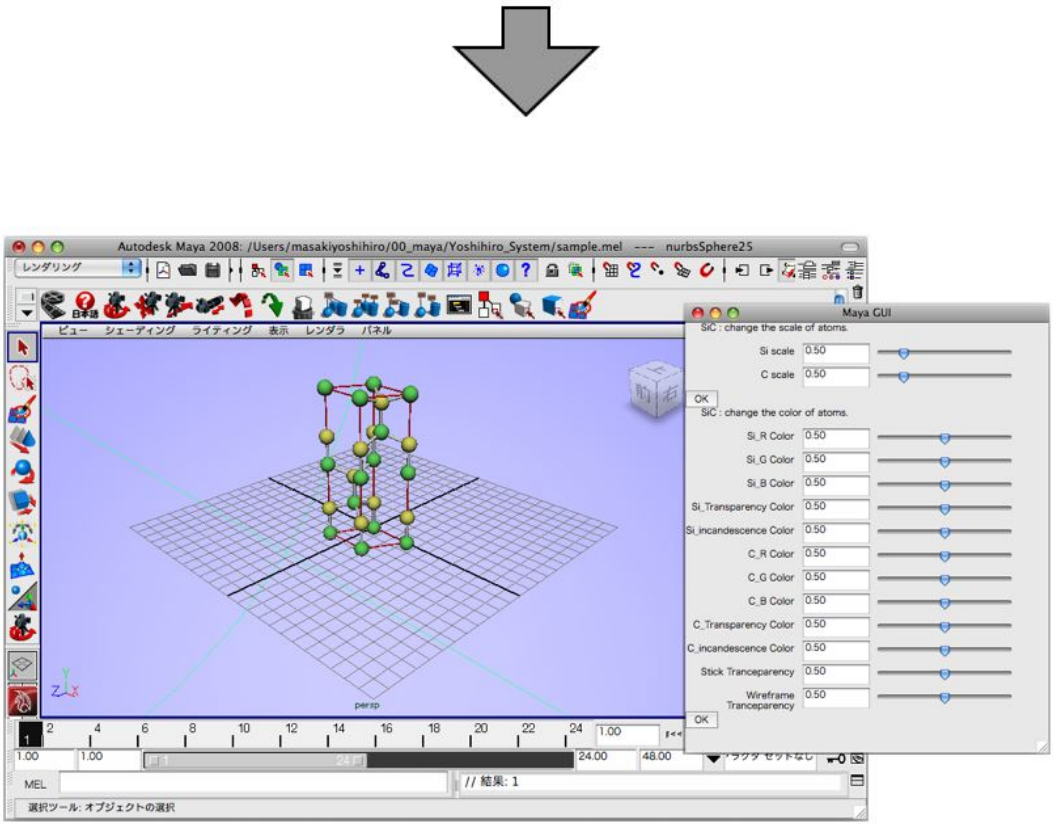


図 5.5: 入力ファイル


Autodesk Maya 2008: /Users/masakiyoshihiro/00_maya/Yoshihiro_System/sample.mel --- nurbsSphere25
Maya GUI
SiC : change the scale of atoms.
Si scale 0.50
C scale 0.50
OK
SiC : change the color of atoms.
Si_R Color 0.50
Si_G Color 0.50
Si_B Color 0.50
Si_Transparency Color 0.50
Si_incandescence Color 0.50
C_R Color 0.50
C_G Color 0.50
C_B Color 0.50
C_Transparency Color 0.50
C_incandescence Color 0.50
Stick Tranceparency 0.50
Wireframe Tranceparency 0.50
OK
persp
MEL
// 結果: 1
選択ツール: オブジェクトの選択

5.2 SiGeの視覚化

西谷研究室の学生がSiGeの規則性に関する研究を行った．応用として，実際にその学生に視覚化を行ってもらい，SiGeの4タイプの格子モデルを紹介する．

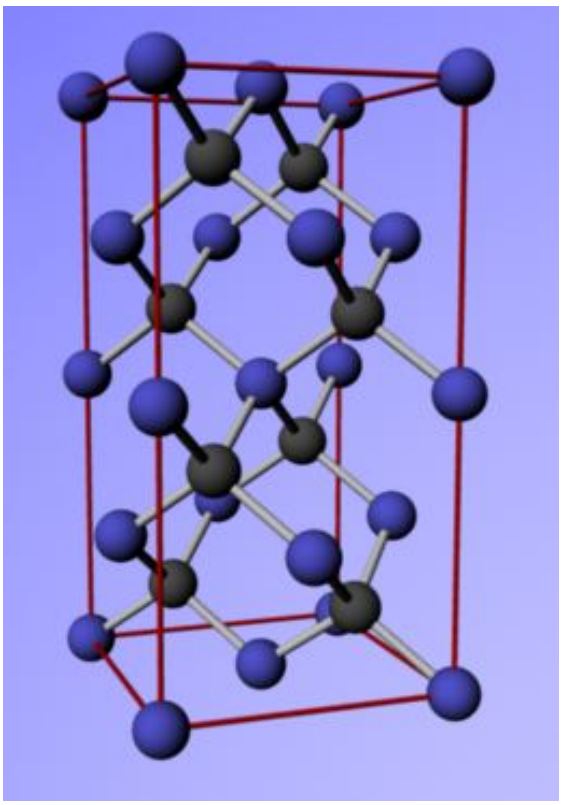


図 5.6: SiGe 組成比 8:8

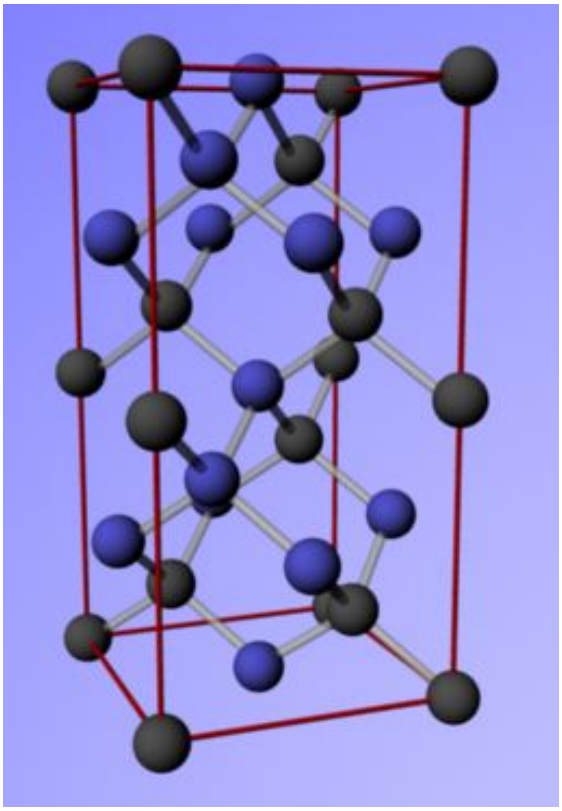


図 5.7: SiGe 組成比 8:8

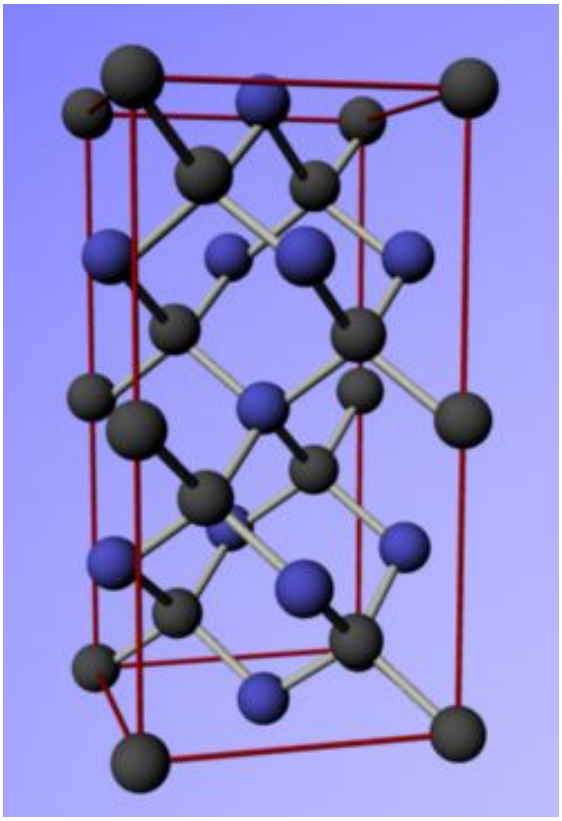


図 5.8: SiGe 組成比 6:10

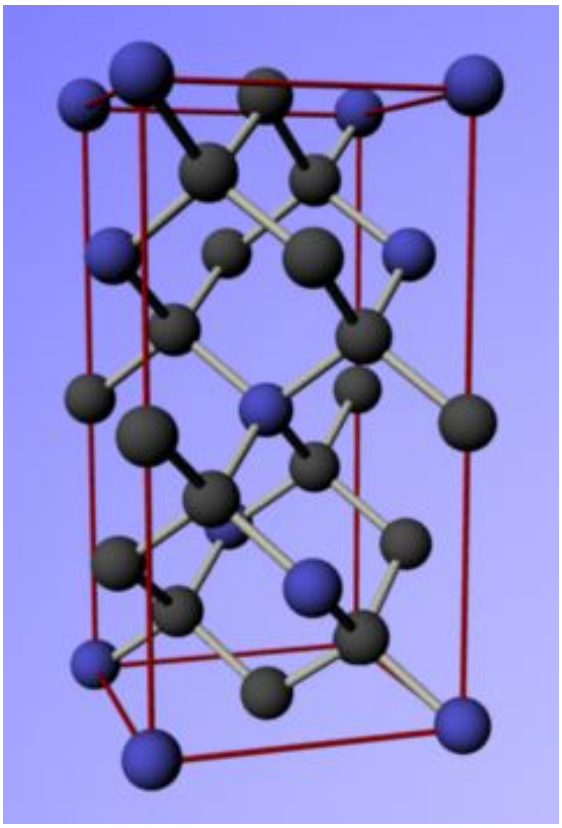


図 5.9: SiGe 組成比 4:12

第6章　総括

本研究の成果を以下に示す.

- キーワードで構成された簡易な入力ファイルの各々のデータをRubyで作成したスクリプトへ読み込ませ，自動的にそれらのデータに見合った操作のクラスを呼び出し，それによって反映されたMELスクリプトを出力させる．その際，Rubyでの工程はブラックボックス化させた．そして，Mayaのインターフェースで生成したMELスクリプトを読み込むことで視覚化を実現させた．また，Maya上でGUIを実装させ，容易にモデルのカスタマイズを行うことを可能とした.

- 従来のシステムではモデルの更新やカスタマイズを行う際，その度にRubyスクリプトを書き替えていたが，その必要がなくなり，入力ファイルの中身を変更するだけで視覚化を可能とした．また，GUIを用いることで，原子の大きさ，色，透明度を変更することができる．これらにより，視認性と操作性が優れた上，視覚化システムのサイクルを大幅に短くした.

- 乱雑であったライブラリの構成はそれぞれの機能を持ったクラスを作成することで，階層構造を実現させた．それによりインプットファイルの要求に応じてクラスの選択ができるようになった．また，クラス個々の独立性が高まり，全体の把握がし易くなった．これらは新たなクラスの追加，変更が容易となり開発のし易さを向上させた．さらにはRubyスクリプトの可読性を高めた.

- 視覚化システムはより直感的な操作が可能となり，開発者以外の研究生でもシステムを使いこなせるようになった．実際に研究生に視覚化してもらい，従来の視覚化システムと比較して使い易さを感じて頂いた.

参考文献

[1] 坂本憲　著,『SiC 単結晶成長の原子レベルシミュレーション』,（関西学院大学　理工学研究科　情報科学専攻　修士論文　2008）.

[2]　ラス・オルセン　著,『Ruby によるデザインパターン』,（ピアソン・エデュケーション　2009　xvii）.

[3]　ラス・オルセン　著,『Ruby によるデザインパターン』,（ピアソン・エデュケーション　2009　P60）.

[4] ラス・オルセン　著,『Ruby によるデザインパターン』,（ピアソン・エデュケーション　2009　P73）.

[5] 阿部智弘　著,『MEL 教科書ー Maya プログラミング入門ー』,（株式会社ボーンデジタル　2004）.

[6] 戸賀瀬健介　著,『視覚化システムの構築と利用』,（関西学院大学　理工学部　情報科学科　卒業論文　2008）.

[7] 『Crystal lattice Lattice Structure』`http://cst-www.nrl.navy.mil/lattice/`.

付 録A

本研究では，視覚化させるツールとしてRuby，MEL，Mayaを使用したが，Rubyを使用せずにMELのみで視覚化を行うことを試みた．MELのみを使用することで，計算時間の短縮やGUIの実装を簡単に行えるなどのメリットが挙げられる．しかし，データファイルの構文解析ができず今回は諦めた．途中ではあるが，そのMELスクリプトを紹介する．

MELスクリプト

```
string $filename;
string $usrfilename;


$filename = "/Users/masakiyoshihiro/00_maya/Maya_System/4H_SiC_unitcell_POS.txt";
$usrfilename = "/Users/masakiyoshihiro/00_maya/Maya_System/usrfile.txt";


$file1 = `fopen $usrfilename "r"`;


string $nextLine = `fgetline $file1`;
while ( size( $nextLine ) > 0 ){

     print($nextLine);
     if(`match "AtomsNumber" $nextLine`)
     print("masaki\n");
     $nextLine = `fgetline $file1`;

}


fclose $file1;


$file = `fopen $filename "r"`;


string $nextLine = `fgetline $file`;
while ( size( $nextLine ) > 0 ){
```

```
        print($nextLine);
        if($nextLine=="Direct\n")
        print("yes\n");
        $nextLine = `fgetline $file`;

}


fclose $file;
```

謝辞

本研究を遂行するにあたり，終始多大なる有益なご指導，及び丁寧な助言を頂いた西谷滋人教授に深い感謝の意を表します．

また，本研究の進行に伴い，西谷研究員の皆様にも様々な知識の提供，ご協力を頂き，本研究を大成することができました．最後になりましたが，この場を借りて心から深く御礼申し上げます．