Java３Ｄを用いた３次元表示空間情報システム

島根大学　総合理工学部
数理・情報システム学科　計算機科学講座　卒業論文

田中研究室
S99444-H 垰田千帆
2002 年 2 月 12 日

目次

第１章　序論

近年インターネットの普及に伴って Web3D 技術の需要が高まっている。Web3D 技術とはその名の通り WEB 上で３D 画像を使用する技術全般である。インターネット上の３次元技術で広く一般に知られているのは、３次元グラフィックスフォーマットである VRML（Virtual Reality Modeling Language　1994 年に VRML1.0）である。VRML は、1997 年に ISO／IEC に VRML97 ISO/IEC 14772 という国際標準仕様として承認され、その後 VRML Consortium は VRML だけでなくインターネット上の 3 次元技術すべてにおいて業界の中心的役割を果たすという意味で 1999 年に「Web3D Consortium」と改名し今日に至っている（[1]より抜粋）。

上記したように、Web3D は特定技術の事を指すわけではなく、インターネット上の３次元技術全般を指すもので、中心にあるのが VRML である。VRML の特徴として[2]

・ネットワークを利用してアクセス可能な３次元空間の構築が可能。
・Web ページとの連動、組み込みが可能
・独自のブラウザ・一般的 Web ブラウザを用いて動作させることが可能
・Java 等を用いて双方向（インタラクティブ）な動作が可能。

等が挙げられる。しかし、３D の複雑な操作や高度な処理には限界がある。その機能に長けている Web３D のもう一つの中心が Java３D である。

Java３D とは Java アプリケーション内で３次元グラフィックスを取り扱うための API(Application Program Interface)を提供する。Java の特性に従って特定のハードウェアやシステムに依存しない。Java3D を用いることでインタラクティブな３次元グラフィックの製作が可能となる。ブラウザ上でも実行が可能なため、Web ページに３次元グラフィックスを掲載することも可能である[3]。

上記の Java3D の特徴を生かし本研究では３次元空間情報システムを構築する。空間情報システムとは地理情報データと統計データを融合したものであり、例として四方面上の自然や建物などの人工の地物、境界、領域などの地理的な位置が明示された情報等で、自然社会、経済、文化的な特徴を表す情報である。

現在では地理情報に関する高精度のデータが多種にわたり採取されている。しかし、一般的に膨大な量のデータを前にしても、数値だけでは利用は困難であり利用目的に合わせてデータを解析する必要がある。本研究では国土地理院　国土地理データを用い、地理の位置情報における様々なデータを平面・立体両方のデータを編集・加工高精度化し、Java3D を用いて立体的に可視化必要な数値データをより分かりやすく視覚的に表示、高度な分析・判断をより易くし問題解決・意思決定支援に役立てるためのものである[4]。

第２章　Java3D

２．１　Java3D について

Java 3D は、Java の標準的な 3 次元グラフィック API（正式名称は Java 3D API）1998 年に Sun (JavaSoft)によって開発された。Java 3D は、シーン・グラフをベースにした 3 次元グラフィック API であり、実際の描画処理は、OpenGL や Direct X などハードウエアに密着した既存のグラフィック API に担当させている。

Java では、"Write once, run anywhere"というスローガンがよく使われ、これは一度コンパイルすれば、Java をサポートするどんな環境でも実行可能になることをうたっているが、Java 3D では、これによく似た"Write once, view anywhere"というスローガンが掲げられ、一度コンパイルされれば Java 2 をサポートするどんな環境でも表示可能になることを目指している。このため、ハードウエア側の事情にに左右されないように、Java 3D のプログラムモデルは抽象化されている。具体的には、プログラマーは個々の描画 API を直接使用するのではなく、シーン・グラフと呼ばれるオブジェクト・モデルを構築していくことでグラフィック・プログラミングを行う。

この「ハードウエアに左右されない」という利点は、特定ハードウエアでの処理効率(パフォーマンス)を追求する場合には欠点にもなる。しかし、アプレットや分散オブジェクトでネットワーク･プログラミングを一新させた Java の優位性は Java 3D にも当てはまる。Java 3D を使ったアプレットは、通常のアプレットと同様にネットワークを経由して動的にダウンロードされ、ブラウザ上で実行できる。Java 同様無償で配布されており、簡単に質の高い３D グラフィックスを作成できる。レンダリングの処理は静的なものではなくリアルタイムで実行されるため、 CG の製作や CAD などの分野だけでなく、ゲームなどのアミューズメント分野への応用が可能である[4]。

２．２　座標系

Java３D の座標系は下記の様に（通常-1.0～1.0 範囲の）ｘ、ｙ軸に加えて奥行きを表すｚ軸で構成されている。

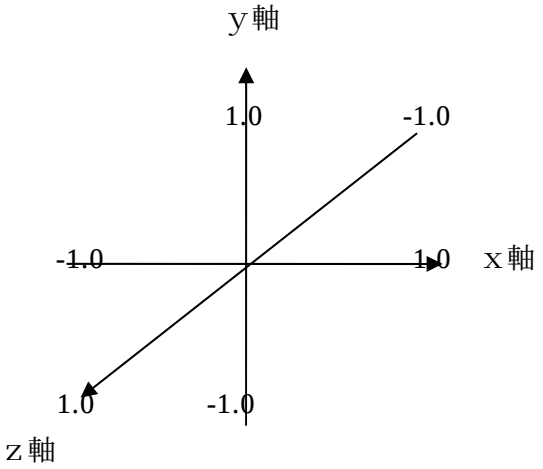


図　2.1

２．３　　シーングラフ

Java3D の特徴の一つとして、シーングラフを使用してグラフィックスを作成できるという点がある。シーングラフとは３次元仮想空間グラフィックスに必要なデータを木構造概念的に表現した``データベース"で、樹形図を用いて表現される。

Java3D では、物体を書く場合に　輪郭等の座標逐一指定→線画していくのではなく、あらかじめ用意されているオブジェクトを利用して３D グラフィックスを一気に描写する方式である。

例として「地面に箱が置かれている」３次元グラフィックスを作成するとする。Java3D ではこの際、仮想的な宇宙空間を作成しその中に物体や世界を作成する　というスタイルで行う。基本的に

・宇宙空間を表すオブジェクト　・箱（描写の対象）を表すオブジェクト
・大地を表すオブジェクト　　・カメラ（視点）を表すオブジェクト

このシーングラフを図示すると以下の様になる

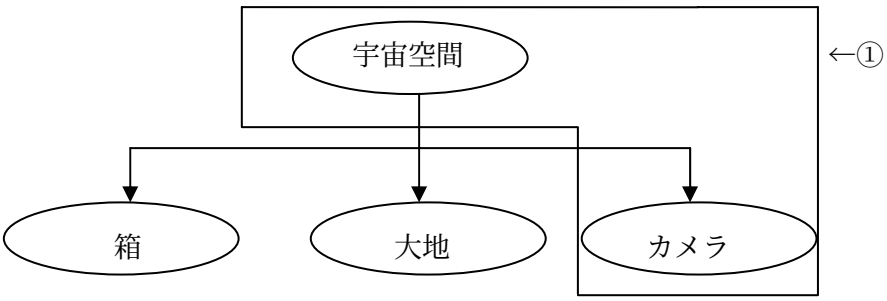


図 2.2

Java3D の全てのプログラムは上記のようなツリー状のシーングラフという概念に基づいて作成される。表示する物体、又は使用する機能が増えれば新たにオブジェクトを作成し、シーングラフに追加すると表示できるようになる。

図２．２の①の様に自分で表示するオブジェクトだけでなく、宇宙空間・カメラを表すオブジェクト等他のオブジェクトも作成する必要がある。そこで①の通常必要な宇宙・カメラのオブジェクトを一つにまとめてくれる SimpleUniverse オブジェクトが存在する。SimpleUniverse オブジェクトはシーングラフのルートの役割を果たすオブジェクトで、この下に全てのオブジェクトを接続することになる。

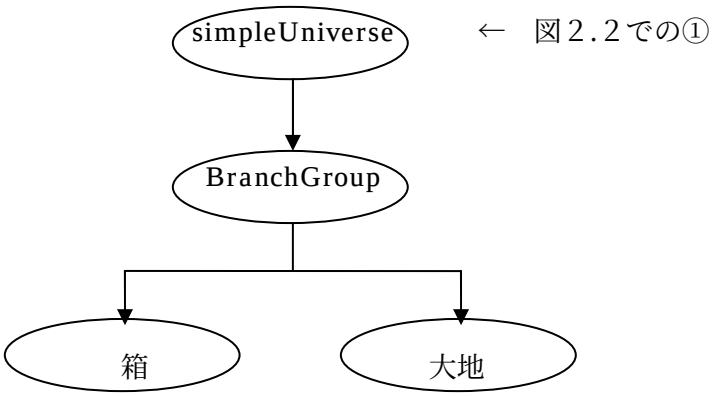


図 2.3

SimpleUniverse オブジェクトの下にシーングラフを接続する場合に必要なのが BranchGroup オブジェクトである。図 2.3 の形が Java3D の接続の基本形となる[8]。

第３章　数値地図情報

３．１　概要

全国的な規模で数値地図情報を整備する場合、“標準地域メッシュ”方式が広く採用されている。本研究に用いたデータも全てその形式に基づいたデータである。一定間隔で正方形に土地を分割　１次＞２次＞３次メッシュ　階層的に一定単位で全国を分割する[5]。

３．１．１　第１次地域区画

全国の地域を１度毎の経線と３分の２度（４０°）毎の緯線によって縦横に分断して第１次地域区画（国土地理院薄幸の縮尺１/２００，０００地勢図の通常の区画に相当する範囲）が作られている。

第一次区画の地域メッシュ・コードは、区画南端の経度を１．５倍した２桁の数字と、西端経度から、１００を引いた２桁の数字とを１．緯度　２．緯度の順に組み合わせた４桁の数字として定義されている[5]。（図 3.2.1-1 3.2.1-2 参照）

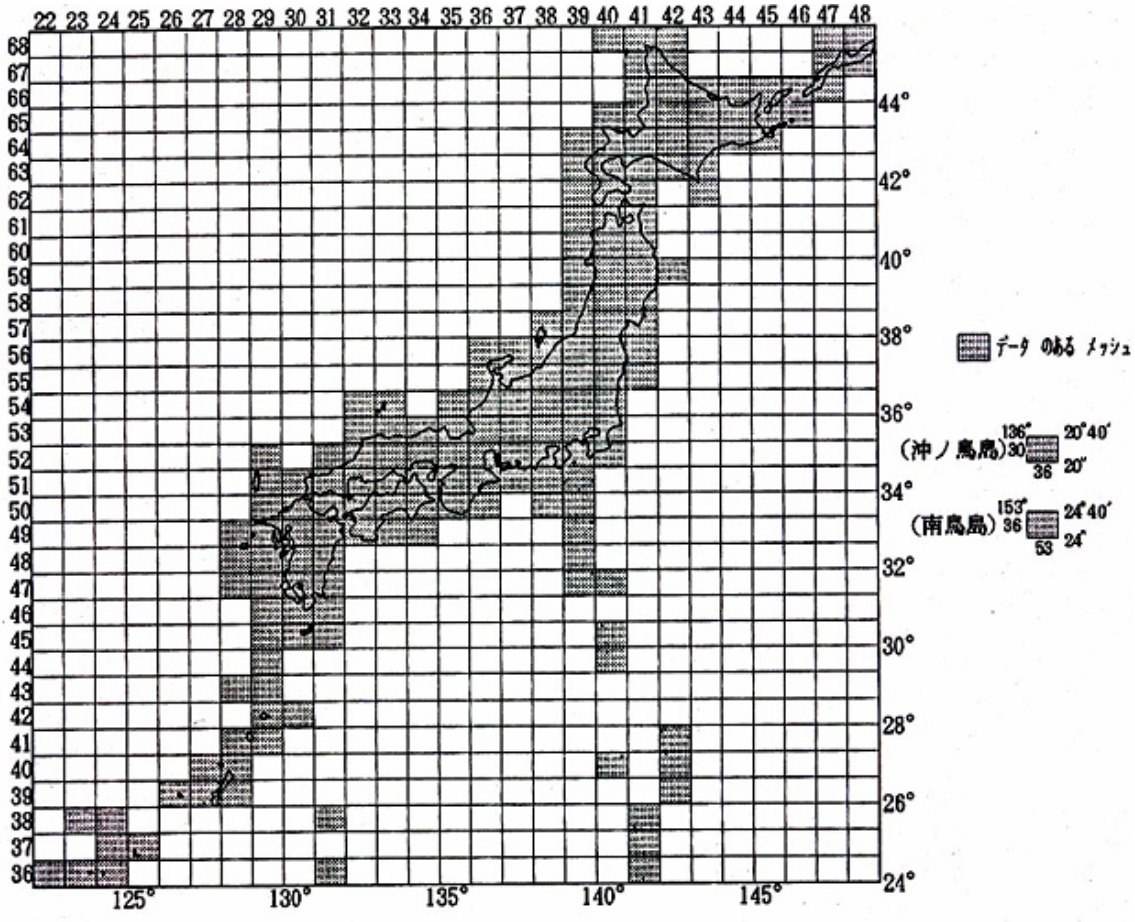
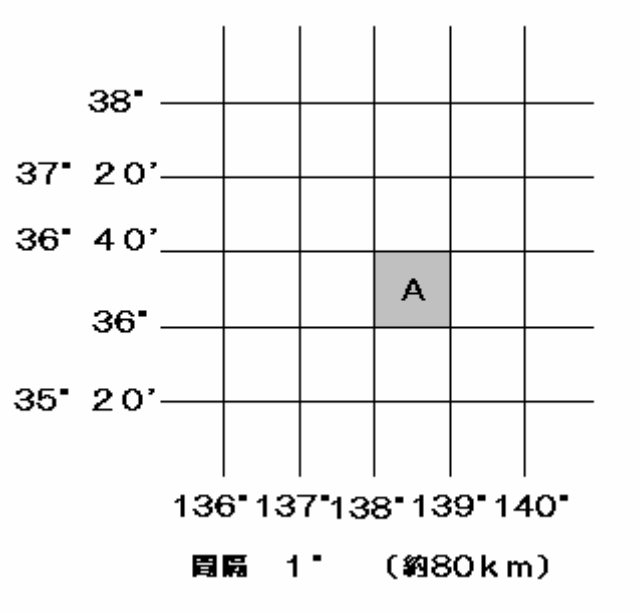


図 3.2.1-1　標準地域メッシュの第一次地域区画



例えばAのメッシュ・コードは

５　４　３　８

36*1.5＝54

（南端緯度の 1.5 倍）

138-100＝38

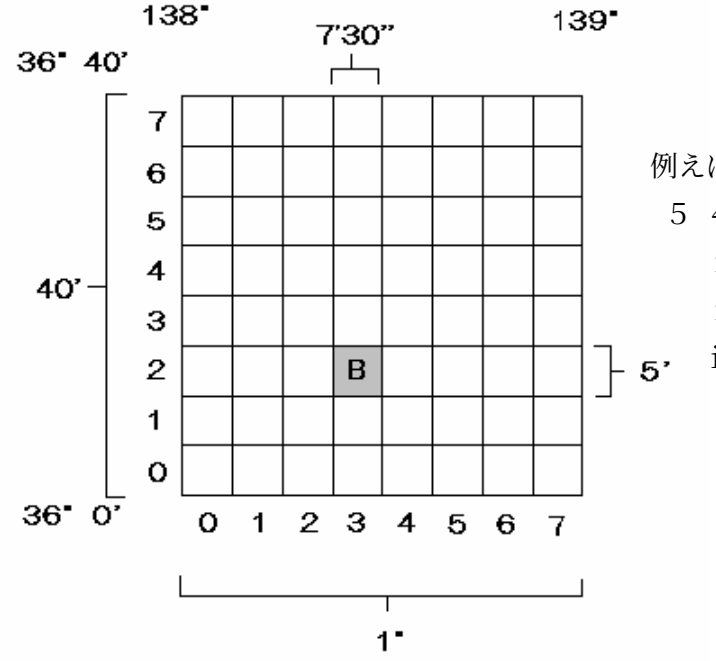
（西端経度の下２桁）

間隔 2°*1/3＝40'

図 3.2.1-2　1 次メッシュコードの付け方

３．１．２　第 2 次地域区画

第一次地域区画の縦横をそれぞれ８等分して第 2 次地域区画（国土地理院薄幸の縮尺 1/25.000 地形図の通常の区画に相当する範囲）が作られている。第 2 次地域区画の地域メッシュ・コードは、第 1 次を８等分した区画に、経線方向については南から、緯線方向については西から、それぞれ０から７までの数値を付け、これを１．経線方向　２．緯線方向の順に組み合わせた２桁の数字として定義されている[5]。（図 3.2.2-1）



例えばBのメッシュ・コードは

５　４　３　８　－　２　３

ｉ　　ii

ｉ　第 1 次地域区画を示す数字

ii　第 2 次地域区画を示す数字

図 3.2.2-１　第 2 次地域区画及びコードの付け方

３．１．３　第 3 次地域区画

第 2 次地域区画の縦横それぞれを１０等分して、第 3 次地域区画（ほぼ1平方キロメートル）が作られている。第 3 次地域区画の地域メッシュコードは、１０等分した区画に、経線方向からについては南から、緯線方向については西から０から９までの数値を付け、これを１. 経線方向　２. 緯線方向の順に組み合わせた 2 桁の数字として定義されている。（図　3.2.3-1）また、第 3 次地域区画は基準地域メッシュとも呼ばれており、国勢調査等全国的なメッシュ情報の多くはこの区画ごとの情報として記録されている[5]。

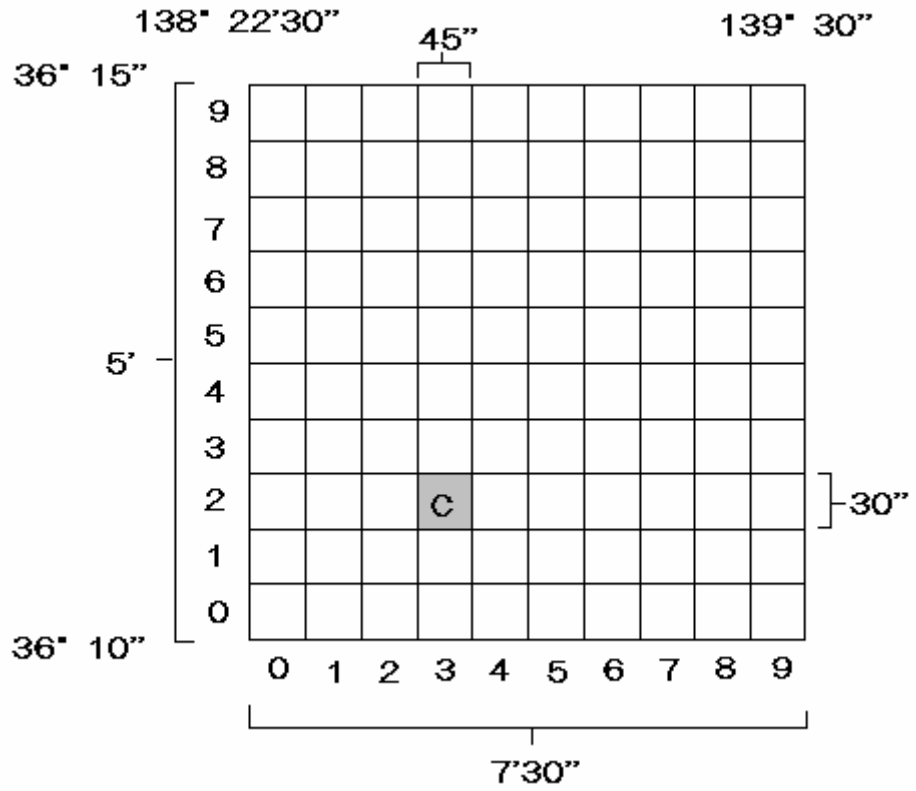


図　3.2.3-1　　第 3 次地域区画およびコードの付け方

例えばCのメッシュ・コードは

５　４　３　８　－　２　３　－　２　３

　　i　　ii　　iii

i　第 1 次地域区画を示す数字

ii　第 2 次地域区画を示す数字

iii　第 3 次地域区画を示す数字

第４章　オリジナルデータのフォーマットについて

４．１ 数値地図 50m メッシュ（標高）データ

４．１．１　概要

国土地理院が刊行している 2 万 5 千分 1 地形図に描かれている等高線から求めた数値標高モデル（DEM: Digital Elevation Model）データ。2 次メッシュを経度方向および緯度方向に 200 等分して得られる各区画（1/20 細分メッシュ、2 万 5 千分 1 地形図上で約 2 ㎜）の中心点の標高値が記録されいる。標高点間隔は緯度（南北）方向で 1.5 秒、経度（東西）方向で 2.25 秒となり、実距離では約 50m となる[6]。

４．１．２　フォーマット

・ファイルフォーマット　　２次メッシュ単位に１つのファイルになっている。

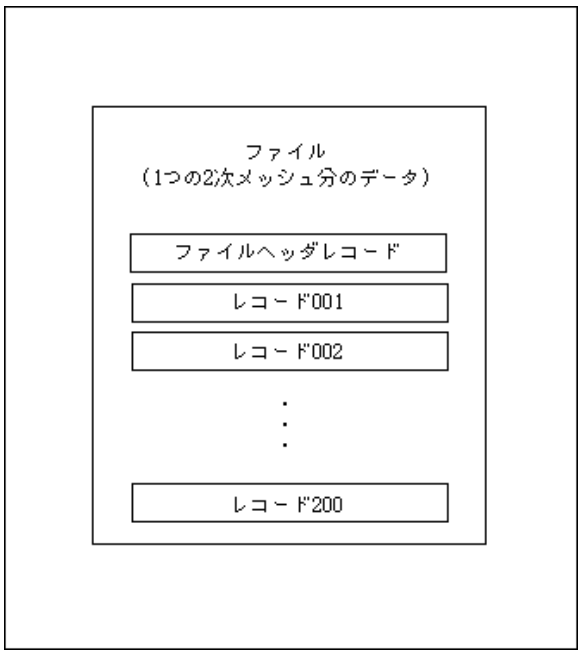


図　4.1.2-1　ファイルフォーマット

・ファイルヘッダーレコード

項　目	開始	終了	仕様(*1)	内　容					
2次メッシュコード	1	6	I6	標準2次メッシュコード	原図の縮尺	7	11	I5	縮尺の分母の数値を記録
原図測量年紀	12	15	I4	西暦年号					
原図修正年紀	16	19	I4	西暦年号					
数値化年紀	20	23	I4	西暦年号					
東西方向の点数	24	26	I3	東西方向のデータ点数					
南北方向の点数	27	29	I3	南北方向のデータ点数					
2次メッシュ左下の緯度	30	36	I7	度3桁、分2桁、秒2桁					
2次メッシュ左下の経度	37	43	I7	度3桁、分2桁、秒2桁					
2次メッシュ右上の緯度	44	50	I7	度3桁、分2桁、秒2桁					
2次メッシュ右上の経度	51	57	I7	度3桁、分2桁、秒2桁					
図葉面数	58	58	I1	当該メッシュにかかる図葉の面数(*2)					
1番目の図名	59	78	N10						
位置フラグ	79	79	I1	1番目の図葉の位置を示すフラグ(*3)					
2番目の図名	80	99	N10						
位置フラグ	100	100	I1	2番目の図葉の位置を示すフラグ					
3番目の図名	101	120	N10						
位置フラグ	121	121	I1	3番目の図葉の位置を示すフラグ					

4番目の図名	122	141	N10	
位置フラグ	142	142	I1	4番目の図葉の位置を示すフラグ
記録レコード数	143	145	I3	記録されているレコードの数
コメント	146	225	N40	利用の際に参考となる情報を記録
レコード1のフラグ	226	226	I1	(*4)
・ ・	・ ・	・ ・	・ ・	レコード2～レコード199のフラグ
レコード200のフラグ	425	425	I1	
余　白	426	1009	584X	
復帰･改行	1010	1011		(*5)

図　4.1.2-2　ファイルフォーマット

・２次メッシュ内のレコード構成

メッシュコード	レコード番号	標　高　値
??????	001	1　2　3　4　･･･　199　200
??????	002	1　2　3　4　･･･　199　200
??????	003	1　2　3　4　･･･　199　200
・	・	・
・	・	・
・	・	・
??????	199	1　2　3　4　･･･　199　200
??????	200	1　2　3　4　･･･　199　200

各標高値は、メッシュの中心の標高値

図　4.1.2-3　２次メッシュ内のレコード構成

レコードは、北端から南端への順序で並んでいる。

各レコードには、2 次メッシュコード、レコード番号、200 個の標高値が記録されている。但し、レコードに含まれる全ての区画が海の場合にはレコード数は変化する事がある。ファイルヘッダレコードの記録レコード数欄に記録された個数となる。･各レコードは、復帰・改行コードで区切られている。文字コードは、シフト JIS を使用している。

項　目	開始	終了	仕様	内　容
メッシュコード	1	6	I6	標準2次メッシュコード
レコード番号	7	9	I3	北→南の順
標高値　1	10	14	I5	西→東の順。
標高値　2	15	19	I5	
標高値　3	20	24	I5	
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	
標高値　２００	1005	1009	I5	
復帰･改行	1010	1011		ＣＲ･ＬＦコード

図　4.1.2-3　レコードフォーマット

"開始"と"終了"は、レコードの各項目の開始と終了の位置。先頭からのバイト数で示している。つまり 1bit 目から 6bit 目までの６桁が、そのデータの２次メッシュコード。
標高値は、０.１m単位で表現されている。（１００．０mは０１０００と表現されている）ただし０．１mの位はすべて０となっている。海部には、－９９９９（－９９９．９m）が入っている[6]。

４．１．３　データの加工

５０mメッシュ標高データに含まれているのは、２００*２００の標高地のデータである。ファイルヘッダーレコード部分については、データの欠損など特別な理由がない場合は、本研究に特に必要はない。データファイルの編集の際に不必要なデータを全て除いている。実際必要なデータはデータ自体のメッシュ番号　標高データとその位置

作成したファイル名でメッシュ番号、データの並びで標高値の相対的位置を確認できるため、ヘッダの部分、各コード・レコード番号を取り除き、標高地だけを取り出すことで標高値の地図を作成するためのデータは十分である。

４．２　JMC マップデータ　　（道路データ）

４．２．１　概要

ＪＭＣマップは、20 万分 1 相当のベクトル形式の地図データ（平面)。データ項目は、行政界･海岸線、道路、鉄道、河川･湖沼、市区町村名等の記号･注記。

データは、１次メッシュ（約 80km 四方：20 万分 1 地勢図）単位に１つのファイルにまとめられている。本研究では各種道路データを、５０ｍメッシュ標高のデータと重ねて表示させるように抜粋、加工している[7]。

４．２．２　フォーマット

・１次メッシュ単位にファイルになっている。
・ファイル内のデータは、２次メッシュ番号順に並んでいる。
・２次メッシュ内のデータは、メッシュ･ヘッダー･レコードに続き、必要に応じてノーレコード、ラインレコード、エリアレコード、ポイントレコード、注記テキストレコードが続きます。なお、ラインレコードには座標値レコード、エリアレコードにはのエリア構成レコードが続く。
・２次メッシュ内の各座標値は、左下を(0,0)、右上を(10000,10000)とする正規化座標で記録されています。なお、ｘ座標は右（東）方向、ｙ座標は上（北）方向となります。
・各レコードは、復帰･改行コードで区切られている。
・文字コードは、シフト JIS コードを使用している。

データファイルは１次メッシュ単位になっており、以下のような構造になっている。

・データファイル（１次メッシュ単位）

メッシュヘッダーレコード（２次メッシュ単位）

レイヤーヘッダーレコード

ノードレコード
ラインレコード
　　座標値レコード
エリアレコード
　　エリア構成ライン番号レコード

レイヤーヘッダーレコード（道路）

ラインレコード
　座標地レコード

レイヤーヘッダーレコード（鉄道）

ラインレコード
　座標地レコード

レイヤーヘッダーレコード（河川・湖沼）

ラインレコード
　座標地レコード

レイヤーヘッダーレコード（記号・注記）

ポイントコード
　座標地レコード
　注記テキストレコード

レイヤーヘッダーレコード（鉄道）

ラインレコード
　座標地レコード

メッシュヘッダーレコード（２次メッシュ単位）

・
・

図 4.2.2-1　データファイルフォーマット

ファイルフォーマットに関する説明は大量にあるが、本研究で使用した道路に関するデータフォーマットについてのみを記述する。

・メッシュ･ヘッダー･レコード

項目	開始	終了	形式	内容
レコードタイプ	1	2	A2	“ M”を記録
２次メッシュコード	3	8	I6	２次メッシュコード
図名	9	２８	N10	当該２万５千分１地形図
レイヤー総数	２９	３１	I3	当２次メッシュに含まれるレイヤー総数
・ 中略 ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・
ライン総数	３７	４１	I５	当２次メッシュに含まれるライン総数
・ 中略 ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・
空白	５７	７２	１６X	

図 4.2.2-2　メッシュ・ヘッダー・レコードフォーマット

・レイヤー･ヘッダー･レコード

項目	開始	終了	形式	内容
レコードタイプ	1	2	A2	"H1":構造化が行われていないレイヤー "H2":構造化が行われているレイヤー
レイヤーコード	3	4	I2	レイヤーの内容を表す 1:行政界・海岸線 2:道路 3:鉄道 5:河川・湖沼 7:記号･注記
ノード総数	5	9	I5	当レイヤーに含まれるノード総数
ライン総数	１０	１４	I5	当レイヤーに含まれるライン総数
エリア総数	１５	１９	I5	当レイヤーに含まれるエリア総数
ポイント総数	２０	２４	I5	当レイヤーに含まれるポイント総数

レコード総数	２５	２９	I5	当ヘッダーレコードを除いた当レイヤー内のレコード総数
空白	３０	３０	1X	
最初の作成年月	３１	３４	I４	当レイヤーの最初の作成年月 西暦年の下 2 桁と月 2 桁
空白	３５	３５	１X	
最終の更新年月	３６	３９	I４	当レイヤーのデータ更新年月 西暦年の下 2 桁と月 2 桁
空白	４０	７２	３３X	

図 4.2.2-3　レイヤー･ヘッダー･レコードフォーマット

・ラインレコード

項目	開始	終了	内容
レコードタイプ	**1**	**2**	**L を記録**
レイヤーコード	**3**	**4**	**１、　行政海岸線** **２、　道路** **３、　鉄道** **４、　河川湖沼**
データ項目コード	**5**	**6**	**①ライン項目コード表参照**
ライン一連番号	**7**	**11**	**レイヤー内でラインが何番目に位置するかを示す一連番号**
ライン種別コード	**12**	**17**	**②ライン項目コード表参照**
始点ノード番号	**18**	**22**	
始点接続情報	**23**	**23**	**０、　図葉内ノード** **１、　隣接図葉に接続する図郭線上のノード** **２、　隣接図葉に接続しない図郭線上のノード**
終点ノード番号	**24**	**28**	
終点接続情報	**29**	**29**	**始点接続情報と同じコード**
左側行政コード	**30**	**34**	**ラインの向きに対しての左側の行政コード**
右側行政コード	**35**	**39**	**ラインの向きに関しての右側の行政コード**
座標点の数	**40**	**45**	**当ラインを構成する XY 座標点の数**
空白	**46**	**72**	

図 4.2.2-4 ラインレコードフォーマット

① ライン項目レコード

レイヤー	コード	データ項目
	1	高速道路および自動車専用道
	2	一般国道
道路 2	3	主要地方道
	4	一般都道府県道

図 4.2.2-5 ライン項目レコード

② ライン一連番号

レイヤー内でラインが何番目に位置するかを示す一連番号。（通し番号のようなもの）

③ ライン種別コード

レイヤー	コード	データ項目
道路 2	0	地上
	1	地下・トンネル

図 4.2.2-6 ライン種別コード

上記のヘッダーレコードに続く数字が実際の道路情報X，Y座標点と思われる。上記では道路情報を扱うことを主に解説をしているが、JMC マップには行政界、鉄道、河川湖沼のデータが含まれている。

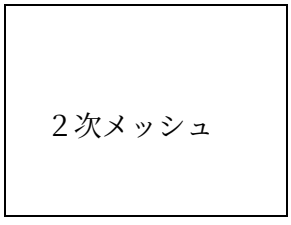
例　　L 2 3　 8　 0　 00　 00　　10 0 0

6458 4228 6392 4061 6423 3606 6396 3047 6378 2500 6140 2337 5946 2421

5374 2446 5104 2299 4673 2060

上記のように次の L （又は M、等のヘッダーレコードの先頭の値が）が出るまでの間の数値が道路を形成する座標の集まりだと思われる。この座標値は下記のように２次メッシュの左下を原点とした相対座標である[7]。

(0 , 10000)　　(10000 , 10000)



(0 , 0)　　(10000, 0)

図 4.2.2-6　統合ファイルの座標値

４．２．３　データの加工

目的のデータは

M　（目的の２次メッシュ番号 ）　で開始される２次メッシュファイル

↓

H１　２　　もしくは

H２　２　　で始まる　レイヤーヘッダーレコード

↓

L　２　　で開始するライン・座標値レコードのデータ

上記の様に必要なデータのファイルを検索し、２次メッシュ単位でファイルに分割格納する。

２次メッシュ単位でメッシュヘッダーレコード（必要な部分のみ使用)、各道路座標データだけを取り出せばよい。

ただし上記のように２次メッシュ範囲内で、ラインを繋ぐための点を１０００*１０００の座標値で表しているので、５０ｍメッシュ標高データと重ねるためには、座標の計算を行わなければならない。　（後述）

４．３　国土数値情報データ（土地利用データ）

４．３．１　概要

KS-200-1（土地利用面積）
KS-202-1（１／１０細分区画土地利用データ）
KS-271（河川単位流域台帳）
KS-273（流域界，非集水界線位置）
KS-602（３次メッシュ流域・非集水域面）
KS-617（１／１０細分方眼流域・非集水域）
上記のデータが各フォーマットで収められている。今回使用したデータは
KS-202-1（１／１０細分区画土地利用データ）　その土地が何に使用されているか
例、田畑、森林、建設用地、海　などが収められている[5][8]。

４．３．２　フォーマット

国土数値情報（KS－２０２－１）1/１０細分区画土地利用データフォーマット

項目	単位	行数	累積行数	データ形式	備考
３次メッシュコード		８	８	9（8）	注１
土地利用コード		2*100	208	９９	
（注１）土地利用コード　01：田　02：畑　03：果樹園　04：その他の樹木畑　05：森林、06：荒地　07：建設用地　09：幹線交通用地　10：その他の用地　11：内水池　14：海浜、15：海水域					

図 4.3.2-1　データフォーマット

３次メッシュ内のデータの順番

91	92	・　・　・　・	99	100
・	・		・	・
・	・		・	・
・	・		・	・
・	・		・	・
・	・		・	・
・	・		・	・
11	12	・　・　・　・	19	20
01	02	・　・　・　・	09	10

図 4.3.2-2　３次メッシュデータの並び

《レコード・フォーマット》（ＫＳ－２０２－１）

３次メッシュコード	土地利用コード				……			
	1	2	3	4	……	98	99	100
9(8)	99	99	99	99		99	99	99
1	9	11	13	15	……	203	205	207

図 4.3.2-3 レコードフォーマット　[5][8]

４．３．３　データの加工

ファイルは１次メッシュ単位に収められているため、他の空間データ（JMC マップデータ、国土数値情報データ等）をあわせるために２次メッシュ単位で切り取る必要がある。各データには１行毎に３次メッシュ番号が記録され、データの並びでその属性（例、田畑、森林等）の位置が認識できる。特にヘッダーなども無い。ただし土地利用データは３次メッシュ毎に昇順でデータが格納されている。他のデータと合わせて２次メッシュ分の範囲でデータ」を扱う場合には、５０ｍメッシュ標高データと同じ様にデータを並び替える必要がある。（後述）

第 5 章　Java3D を用いた描写

本研究では 2 次メッシュの範囲を一つの範囲として地図を作成している（実距離　５０ｍ＊２００）。Java３D において標準で画面内に写る範囲の座標は(-1<x,y,z<1)である。縮小拡大を含めば設定した描写範囲（光源の届く範囲まで）までは作成可能となる。Java3D を実際に用いて正確に描写するには、前章のデータの前処理と Java３D のクラス及びメソッドに合わせて行う必要がある。

５．１　　５０ｍメッシュ標高

Java3D を用いて、５０ｍメッシュ標高データに基づいた３次元立体地図を作成、２次メッシュのサイズを単位として扱う。５０ｍメッシュ標高データにおいて、２次メッシュ単位の範囲内には、２００*２００の相対的位置とその地点での標高値が格納されている。隣接する４点で４角形の面を作り、それを張り合わせて地図を描写する。

●QuadArray

Shape3D クラスで新しい物体を定義、Geometry クラスで実際の形状を引数として指定する。形状　QuedArray　＝　四角形（形状）を描くクラス。引き数に点の数、点のフォーマット、色、点の座標を指定する。視点より左回りに４つの点を指定する。引数となる点の数は必ず４の倍数でなければならない。

視点の方向から反時計回りに点を指定することで、四角形を作成する。

（■０　→　点 0,200,201,1　■１　→　点 1,201,202,2 ・・・・）

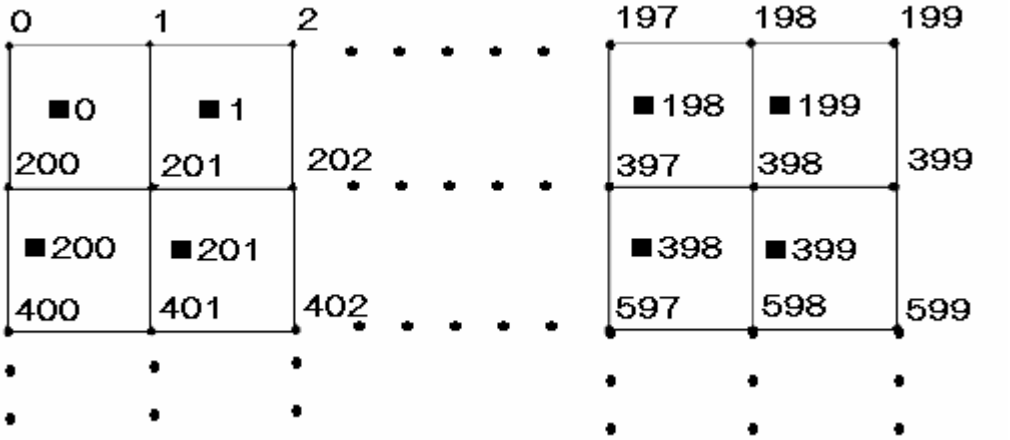


図 5.1-1 QuadArray の描写

上記の様に作成することで 199*199 の四角形に

ｘ、ｙ座標値　　隣接する点との距離（実５０ｍ）を一定に縮尺したもの

ｚ座標値　　ｘ、ｙ座標と同じ縮尺で格納されている標高値を縮小したもの

を設定、3 次元地図を作成する。５０ｍメッシュ標高だけを用いたなら上記の方法で良い。

しかし、国土数値情報　土地区画利用データと重ねて描写する場合はデータを拡張する必要がある。上記の方法では 2 次メッシュ区画内に 199*199 の四角形が作られる。国土数値情報に格納されているのは 200*200 の土地利用データである。描写の最小の単位が点同士を結んだ 4 角形であるため、全てのデータを重ねることができない。それを解決するため、隣接する 2 次メッシュデータから縦横 1 点ずつデータを抜き出す。200*200 の四角形を作りそのまま要素を重ねることにするため、５０ｍメッシュ標高のデータを 201*201 のデータの修正する。

例として　次の並びの　523216 のデータを編集する

523216	523217
523206	523207

図 5.1-2　メッシュの並び

523216 メッシュの標高データを作る際に　元々523216 データには 200*200 の標高データが含まれている。横一列２００のデータが２００行並んでいるわけだが、横一列の最後尾に右隣の 523217 のデータを 1 点だけ２００行分全てに付け加える。加えて最下 1 行を、（523206 の最上 1 行（200）＋523207 の左上 1 点データ）付け加える。

523216 メッシュ↓　　↓523217 メッシュ

0　，　1　・・・・・・・198、199	0
200,201・・・・・・・398,399	200
・　　・	・
・　　・	・
・　　・	・
39600,39601・・・39798,39799	39600
39800,39801・・・39998,39999	39800
0　，　1　・・・・・・・198、199	0

523206 メッシュ↑　　↑523207 メッシュ

図 5.1-3　２次メッシュ内のデータ（番号）の並び

523216↓

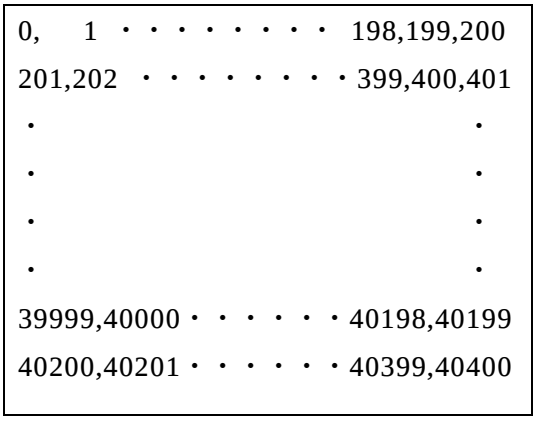


図 5.1-4 編集後の２次メッシュの並び

上記の手順で編集したデータ　201*201 点　200*200 四角形が作成できる。

５．２　JMC マップデータの投影

上記したとおり JMC マップ（道路データ）には、高速道路および自動車専用道、一般国道、主要地方道、一般都道府県道　の種類別に区別された道路データと、その座標（ｘ、ｙのみの 2 次元要素）が格納されている。

前述した様に **Jmc** マップ道路データは **2** 次メッシュ **1** つ分が　**0**＜ｘ、ｙ＜**10000**　の座標で収められている。５０ｍメッシュ標高も **2** 次メッシュ一つ分が　**-1**＜ｘ、ｙ＜**1**　内の座標で収められているため、**Jmc** マップの座標データを５０ｍメッシュ標高の座標データを合わせるため（毎回）プログラム中で変換を行う。

※データファイルはＪＭＣマップの座標そのままの値　（**5680,1288**）等の値で取り込み随時プログラム中で計算式に当てはめ、**Java3D** の座標値　（**0.136,-0.7424**）等に変換するプログラム中でＪＭＣマップの座標系から５０ｍメッシュ標高の座標系に、座標値を変換（例（5680,1288）→（0.136,-0.7424））しているのであるが、一見元から変換したデータを用意しそれを取り込むデータとすれば、計算時間が短縮できるように見える。しかし、ヘッダから必要な情報を取り込む必要がある事、座標データは５桁の範囲内に右詰でデータが格納されている事により、プログラム中でファイルからデータを呼び出す際に１byte づつ文字コードで呼び出し、再び演算で５桁（以下）の数字に戻す必要がある。そのために取り込むデータの桁、種類（マイナス記号が付く、少数が含まれるようになる）が増えると、それを取り込むための演算でむしろ計算量が増えることになる。

例として、オリジナルでの座標（5680,1288）　変換後の座標（0.136,-0.7424）とする
・オリジナルの座標を取り込みプログラム中で演算を行ったとする。取り込んだ文字コー

ドを５桁づつ元の数字に変換する。文字コードは
（文字コード→元の数字）48→0、49　　→1、・・・・57→9、32→空白
という変換になる。さらにそれを５桁で一まとめに考えて、
1桁目×10000 ＋ ２桁目×1000 ＋ ３桁目×100 ＋ ２桁目×10 ＋ １桁目　という手続きで元の５桁（もしくはそれ以下）の数字に戻す。それから座標変換の演算
〔　元の座標 － 5000 × 0.0002 〕を行う事になる。

・一方変換後のデータをファイルに格納しプログラムで取り込む場合は、最後の
〔　元の座標 － 5000 × 0.0002 〕　　という演算だけは省くことが出来るが、小数点のコードの判別、－等の符号の判別、桁がバラバラになる事にたいする条件選択などの演算が増えることになる。また、今回の演算では当てはまらないが、演算の結果数字の桁が増えてしまった場合、桁落ち等でデータを正確に記録又は呼び出せず値が変わってしまう恐れもある。

以上の理由から元のＪＭＣマップの座標系の値を取り込みプログラム中で５０ｍメッシュ標高の座標値への変換を行っている。

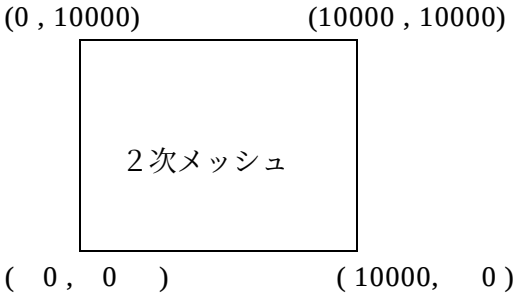


図 5.2-１　JMC マップ道路データの座標値

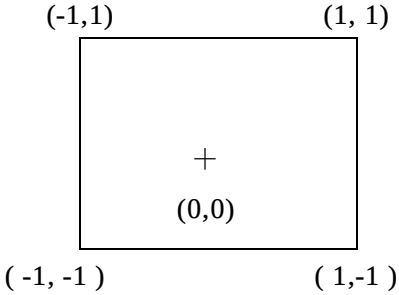


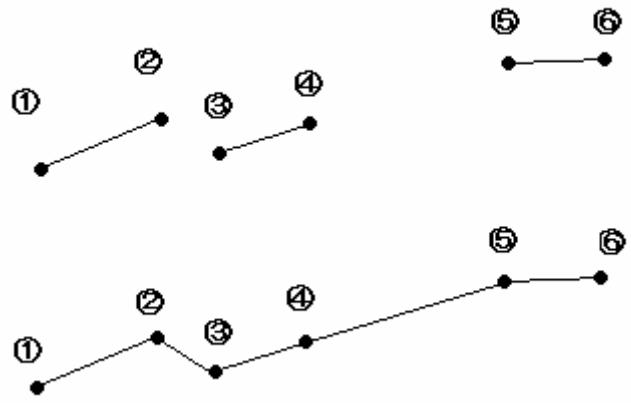
図 5.2-2　Java3D で実際に使っている座標値

● LineArray　クラス

QuadArray 同様 Shape3D—Geometry で LineArray を指定し、直線を描写する。引数

として頂点の数、頂点の形式を指定。このクラスは４つの頂点を指定したら順に線が結ばれていく訳ではなく、１番目と２番目の間で線を引き、３番目と４番目の間で線を引くのであって、２番目と３番目の間の線は引かれない。このことを配慮して配列にデータを格納しなければならない。必ず２の倍数の頂点を常に用意する必要がある。

実行時に毎回プログラム中で
①、Java３D での座標値に変換する
Java３D 上での座標値　＝（ファイルの座標値　－　５０００）　×　０．０００２
②、点の座標を重複して配列に指定する
LineArray クラスで直線を引くために配列に点の座標を格納するが、このとき
ファイル中では　　（※　点１の座標＝点１　として　）
（１）（　点１、点２、点３、点４　・・・・・　点 N-1、点 N）
と格納されているが描写の際には
（２）（　点１、点２、点２、点３、点３、点４、点４　・・・・・点 N-1、点 N-1、点 N）
始点と終点の点以外を重複して配列に格納している。



←（１）の場合

←（２）の場合

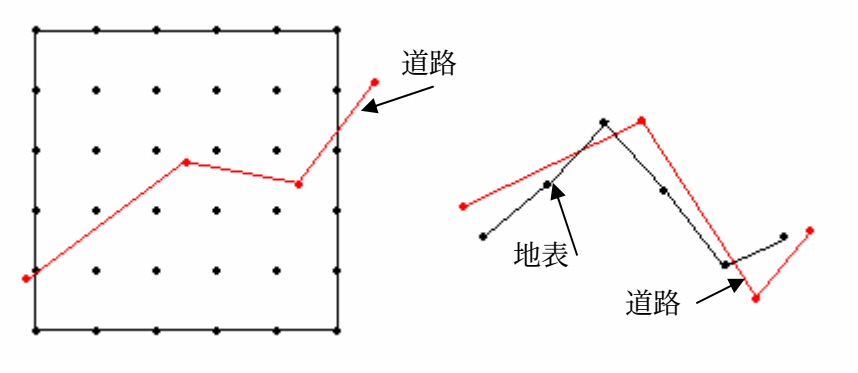
図 5.2-3　LineArray

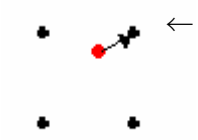
ここまでの段階のデータで３次元標高＋道路の描写を行った結果、道路にあたるラインが一部地図の表面に埋没してしまった。原因は以下のように考えられる。

道路データのｚ座標はｘ、ｙ座標から見て最も近い５０ｍメッシュ標高の点のｚ座標を取っている。５０ｍメッシュ標高のデータは一定間隔（２次メッシュ一中に２００*２００の点）で標高値（ｚ座標）が取られている。それに対し道路データの方は、不均一な間隔で取られる折れ線の点の座標の部分のみで高さの値を与えられるためと思われる。

真上から見た場合　　真横から見た場合

↓ 地表平面





← 道路データの点のｚ座標は道路データの点のｘ,ｙ座標に近い５０ｍメッシュ標高のｚ座標を使用している。

図 5.2-4 道路と地表の関係

上記のような座標のずれと、データの精度の差が原因と思われる。
解決策として
① 道路データの点の間隔をより細かく取り、データの誤差を少なくする。
（データファイルの編集）
②全体の道路データ座標の底上げを行う
この二つを行った。

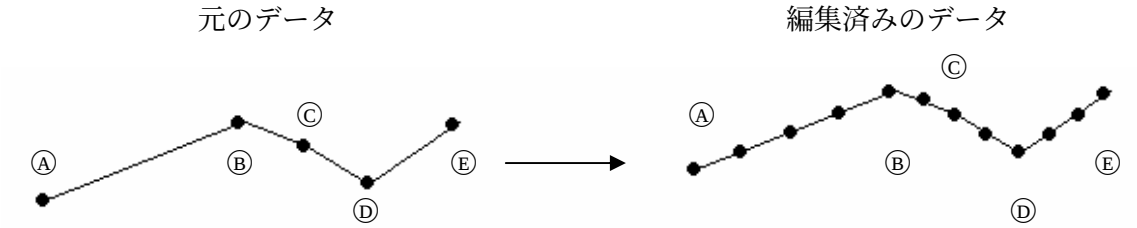


図 5.2-5 ライン細分割

①の作業 （１）隣接する点と点の間の距離を測る
（２）基準とする距離で割る
（※今回の場合およそ５０ｍメッシュ標高の隣接する点間の距離）
（３）（２）の結果で隣接する点の間を分割、その点の座標（ｘ、ｙ）と、記録し、増えた分のデータ数をカウントする
（４）同様に（１）～（３）の作業を全ての点において行う
Ⓐ～Ⓔ ラインが全て終了したら増分のデータの点数をファイルに記録する。

以上の作業でデータを細かく取ることで、元々線分 ⒶⒷでは Ⓐ Ⓑ ２点でしか高さの値をとっていなかったが、図の通り分割した分の点数分（上図では３点）だけ細かく

高さの値を取る事が出来、誤差を（線の埋没等）防ぐことが出来る。

以下に実例を挙げる

ｉ）抜粋しただけのデータ　　↓元の点の数

L　2 2　　7　　0　　00　　00　　2 0　0

8683 6033 8674 6204

L　2 3　　8　　0　　00　　00　　10 0　0

6458 4228 6392 4061 6423 3606 6396 3047 6378 2500 6140 2337 5946 2421

5374 2446 5104 2299 4673 2060

ii）編集を行ったデータ　　↓点の数が増加

L 2 2　　7　　0　　00　　00　　4 0　0

8683 6033 8680 6090 8677 6147 8674 6204

L 2 3　　8　　0　　00　　00　　69 0　0

6458 4228 6436 4172 6414 4116 6392 4061 6395 4010 6398 3959 6402 3909

6405 3858 6409 3808 6412 3757 6416 3707 6419 3656 6423 3606 6420 3555

6418 3504 6415 3453 6413 3402 6410 3351 6408 3301 6405 3250 6403 3199

6400 3148 6398 3097 6396 3047 6394 2992 6392 2937 6390 2882 6388 2828

6387 2773 6385 2718 6383 2664 6381 2609 6379 2554 6378 2500 6330 2467

6282 2434 6235 2402 6187 2369 6140 2337 6091 2358 6043 2379 5994 2400

5946 2421 5894 2423 5842 2425 5790 2427 5738 2430 5686 2432 5634 2434

5582 2436 5530 2439 5478 2441 5426 2443 5374 2446 5329 2421 5284 2397

5239 2372 5194 2348 5149 2323 5104 2299 5056 2272 5008 2245 4960 2219

4912 2192 4864 2166 4816 2139 4768 2113 4720 2086 4673 2060

アンダーラインの部分が元のデータと共通。元の点と点の中間をある程度の距離を持って、新しいデータを採っている

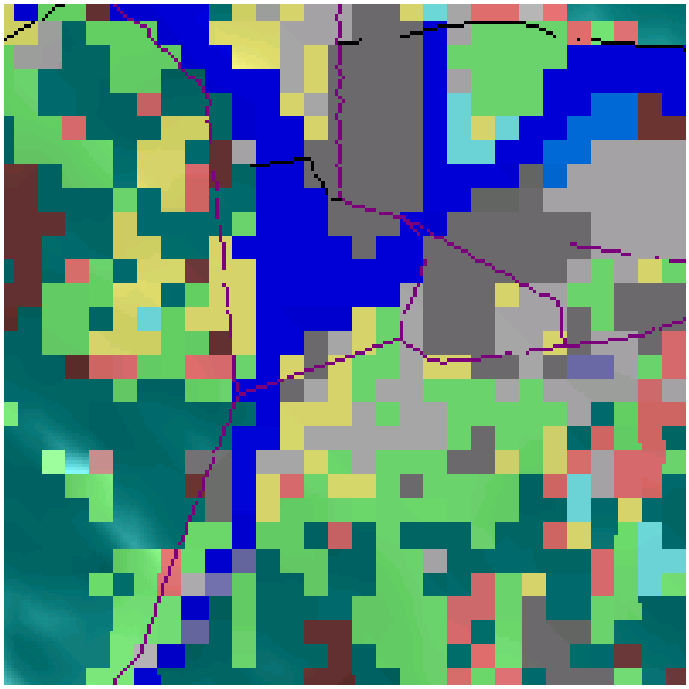


図 5.2-7　取り出しただけのデータでの描写　埋没している部分が見られる

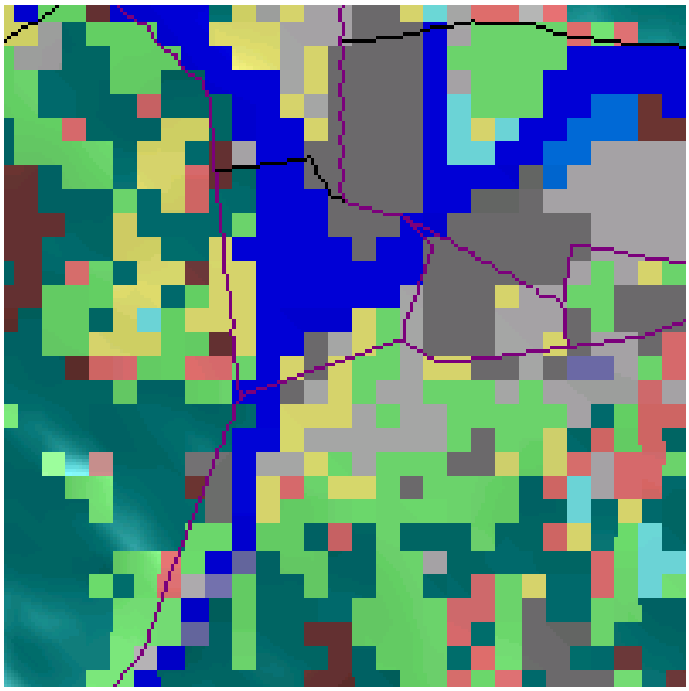


図 5.2-8　補完したデータでの描写　埋没部分が無くなっている

5.3　国土数値情報土地利用データの投影

オリジナルのファイルでは、ファイルは１次メッシュ毎に編集され、ファイルの中は３次メッシュ単位でレコードが記録されている。

上から

52320000, 52320001, 52320002,・・・・・・・・・・・・・・・・52320098, 52320099

52320100, 52320101, 52320102,・・・・・・・・・・・・・・・・52320198, 52320199

・　・　・　・　・

・　・　・　・　・

52327600, 52327601, 52327602,・・・・・・・・・・・・・・・・52327698, 52327699

52327700, 52327701, 52327702,・・・・・・・・・・・・・・・・52327798, 52327799

上記の３次メッシュ番号の順に、レコードが並んでいる。

さらに

（１）

90	**91**	・	・	・	・	・	・	**98**	**99**
80	**81**	・	・	・	・	・	・	**88**	**89**
・	・							・	・
・	・							・	・
・	・							・	・
・	・							・	・
・	・							・	・
・	・							・	・
10	**11**	**12**	・	・	・	・	・	**18**	**19**
00	**01**	**02**	・	・	・	・	・	**08**	**09**

図 5.2-1　２次メッシュ内の３次メッシュの並び

（２）

91	92	93	・	・	・	・	98	99	100
81	82	83	・	・	・	・	88	89	90
71	72	73	・	・	・	・	78	79	80
・	・	・					・	・	・
・	・	・					・	・	・
・	・	・					・	・	・
・	・	・					・	・	・
21	22	23	・	・	・	・	28	29	30
11	12	13	・	・	・	・	18	19	20
01	02	03	・	・	・	・	08	09	10

図 5.2-2　3 次メッシュ内でのデータの並び

上記のようにファイル内には 3 次メッシュも、その内容のデータも昇順にデータが格納されている。一方、５０ｍメッシュ地図標高データは左上を０、右下を３９９９９番目の要素とした、データである。プログラム上データを並び替える必要がある。

計算量を減らすため、あらかじめ 523200** 、523201**、・・・523277**（※**は 00～99)の２次メッシュ単位でファイルを分割しておく。

データを５０ｍメッシュ標高のデータにあわせると、２次メッシュ地図上の左上から開始、右下で終了なので具体的には

例、５２３２１６ファイル
レコード番号９０　　データの順番　９１　～１００　までを read write
　　９１　　９１　～１００　まで
　　・
　　・
レコード番号９９　　データ　　９１　～１００　までを read write
※ここまでで５０ｍメッシュの左から右への 1 行分のデータになる
レコード番号９０　　データ　　８１　～　９０
レコード番号91　　データ　　８１　～　９０

・
・
レコード番号９９　　データ　　８１　～　９０
※　　同様に
・
・
レコード番号９０　　データ　　０１　～　１０
レコード番号91　　データ　　０１　～　１０
・
レコード番号９９　　データ　　０１　～　１０
※ここまでで3次メッシュの９０～９９までのデータは完了。以下同様に
レコード番号８０　　データ　　９１　～　１００
レコード番号８１　　データ　　９１　～　１００
・
レコード番号８９　　データ　　９１　～　１００
・
・
レコード番号８０　　データ　　０１　～　１０
レコード番号８１　　データ　　０１　～　１０
・
レコード番号８９　　データ　　０１　～　１０
・
・
・同様に続けていき
レコード番号００　　データ　　９１　～　１００
レコード番号０１　　データ　　９１　～　１００
・
レコード番号０９　　データ　　９１　～　１００
・
・
レコード番号００　　データ　　０１　～　１０
レコード番号０１　　データ　　０１　～　１０
・
レコード番号０９　　データ　　０１　～　１０
この順番にデータをつなぎ変える必要が在る。
　加えて上記の方法でデータの編集を行った場合、3次メッシュ自体は１０×１０の点

2 次メッシュは　　１０×１０の 3 次メッシュで表される。国土数値情報データから引き出されるのは　１００×１００のデータである。５．１で５０ｍメッシュ標高データは１次メッシュ分に 201*201 の点のデータが格納されるように編集した。結果 200*200 の四角形の面で１次メッシュ分の地図が表されることになる。

次に５０ｍメッシュ標高に、国土数値情報データの数を合わせる

国土数値情報データを視覚的に下記の様に分割する

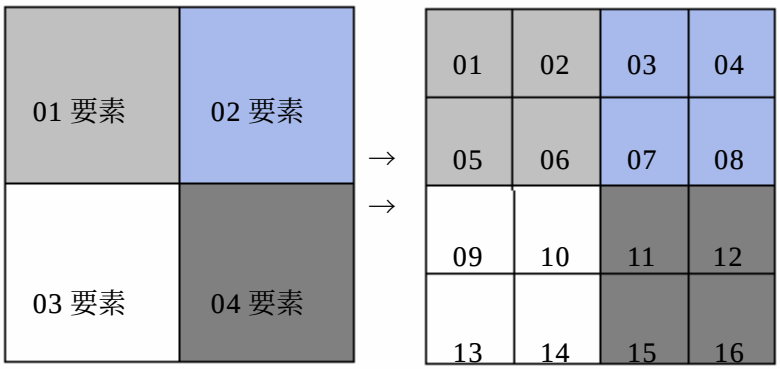


図 5.2-3 データの分割

例

実際のデータ要素が　　01　02

03　04

だった場合　　↓

01　01　02　02

01　01　02　02

03　03　04　04

03　03　04　04

という並びにデータを再編集する。これで国土数値情報データが 200*200 のデータになる。

５．４　データ処理のまとめ

・データの前処理

○50m メッシュ標高データ

データを２次メッシュ単位に編集して一つのファイルにする。

200×200 の点のデータを隣接する２次メッシュから 201×201 のデータに再編集する（土地利用データ KS-202-1　と重ねるため)。

○JMC マップ（道路）データ

道路のデータだけを抜き出し、２次メッシュ毎に編集して一つのファイルにする。

一定距離毎に（地図の縮尺・描写範囲による）演算で新たな点を取る。

○土地利用データ（KS-202-1）

データを 2 次メッシュ単位に編集して一つのファイルにする。

３次メッシュ毎の並びを、50m メッシュ標高データと同様に２次メッシュ毎の並びに編集する。

50m メッシュ標高データと重ねられるようにデータを 100×100　→　200×200　に編集する。

↓↓

・プログラム中での処理

○50m メッシュ標高データ

データの順から画面上での x,y 座標を、それと同じ縮尺でｚ座標も計算する。

［例、画面内は (-1<= x,y <= 1)で点が 201×201 であるから、点と点の距離は約 2/201 ］

［　　一番左上の点の座標は(-1 , 1)その右隣が(-1+2/201 , 1)となる。　　］

QuadArray に引数として渡される為、図 5.1-1 の様に四角形毎に左回りの順に配列に格納される。

○JMC マップ（道路）データ

(0 <= x,y <= 10000)座標のデータを取り込んで(-1 <= x,y <= 1)の座標に変換し、

対応する位置にあるｚ座標を 50m メッシュ標高より引き出し重ねる。

LineArray に引数をして渡される為、図 5.1-3 の要領で点を重複して配列に格納する。

○土地利用データ（KS-202-1）

編集は済んでいるので、そのままの順序で対応する位置の 50m メッシュ標高データに色の要素として重ねる。

５．５　プログラムとそのシーングラフ概略

・使用クラスについて　[8]

SimpleUniverse	シーングラフのルートの役割を果たす。
BranchGroup(objRoot)	SimpleUniverse の下に実際に描写するオブジェクトを接続する役割を持つ。これ以下が Canvas に実際に反映される。
Canvas3D	グラフィックスが描写されるウィンドウのオブジェクトを現す
BorderLayout	表示画面のレイアウトの設定を行う
BoundingSphere	描写に使う３次元空間は無限に広がっているが、無限の空間を演算することは不可能。このオブジェクトよって作られた球体で一定範囲を指定する。
Color3f	red,blue,green (1.0f,1.0f,1.0f)　色の指定を行う
Vecter3f	引数で指定された座標－原点を結んだ線に、方向を指定する。
DirectionalLight	光源の設定を行う
PickRotateBehavior	ピッキング（マウス操作）　選択した対象の回転を行う
PickZoomBehavior	ピッキング（マウス操作）　選択した対象のズームを行う
MainFrame	描写画面の設定（画面の大きさ等）を行い画面に表示
TransformGroup	既存のワールド座標系（0.0,.0.0,0.0）以外に新しく座標系（ローカル座標系）を設置する
Transform3D	ローカル座標系について座標変換を行う
Vector3d	(x, y, z)の各座標方向に関する情報を一括して表記する
Shape3D	プリミティブ（Java3D で事前に用意されている物体）とは別に自分で物体を定義するために使用
LineArray	指定した二つの座標間に線を描写。座標は必ず２の倍数
Point3D	３次元での座標の指定に用いる
Appearance	外観の設定を物体に反映させる
LineAttributes	線を描画する際の線の太さ、破線などの情報を指定
GeometryInfo	物体の形状を定義する
NormalGenerator	法線の設定（自動的に計算をさせ）を行う

・本研究は描写の際

プログラム①　　土台となるキャンバス（描写範囲、光源、拡大縮小等）の描写を行うプログラム

↓（呼び出し）

プログラム②　　最小単位（２次メッシュ１つ分）の画像を組み合わせて座標変換・縮尺を決め一つの画像にまとめるプログラム

↓（呼び出し）

プログラム③　　最小単位の２次メッシュ１つ分のプログラム（場合に合わせて複数）データ直接の

この順で最低でも３つのプログラムを呼び出して画像を作っている。

※実際の演算・データの呼び出しは③でのみ行われ、２次メッシュ毎に地表の図が作られる。①、②はその２次メッシュを部品として扱い、最終的な結果を作る作業を行っている。

以下対応するシーングラフ

・プログラム①

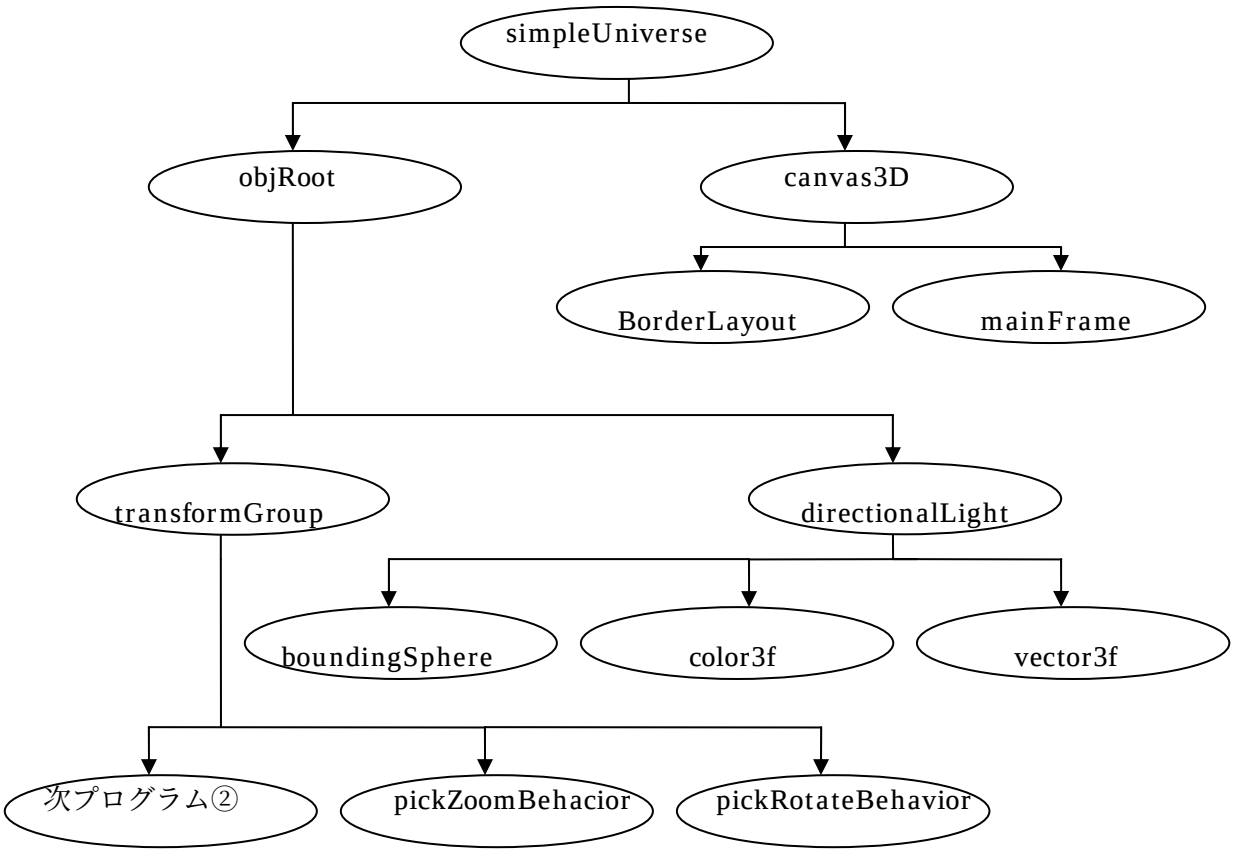


図 6.3-1 プログラム①のシーングラフ

・プログラム②

２次メッシュ一つを呼び出す部分

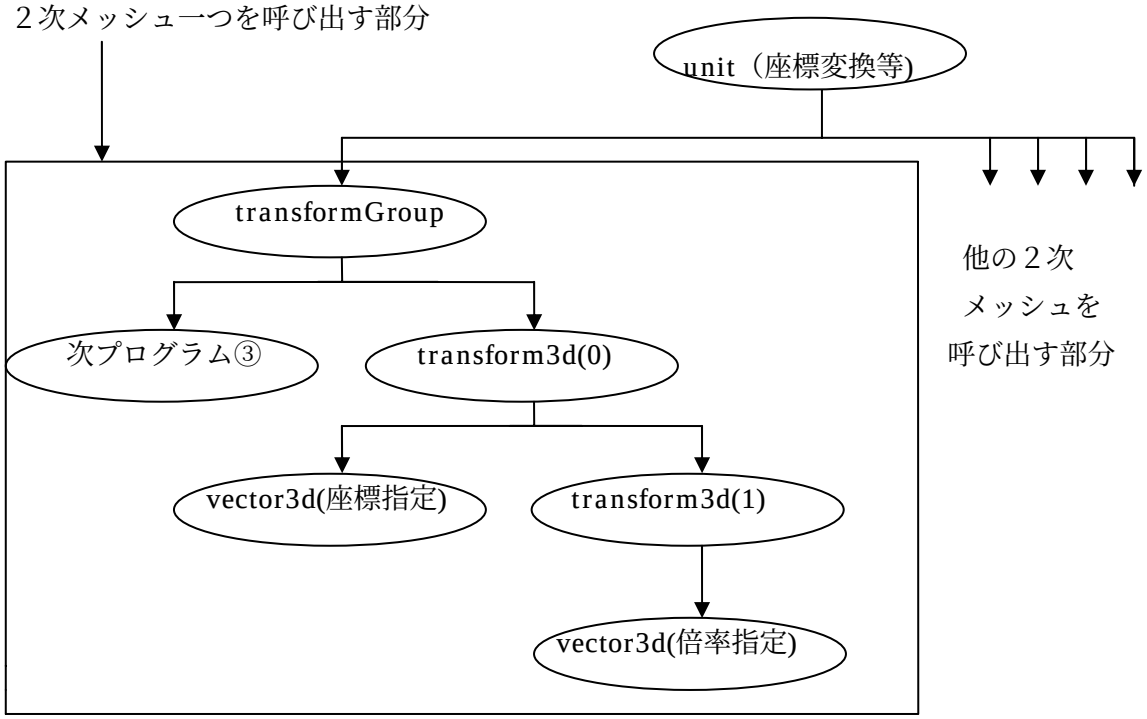


図 6.3-2 プログラム②のシーングラフ

四角の部分で最低限一つの２次メッシュを呼び出す。座標と倍率は呼び出すメッシュの数に応じて入力しておく必要がある。例として２次メッシュ４つ分（２×２）描写を行いたい場合は　倍率を 1/2（４つ共通）に指定し、２次メッシュの座標系をそれぞれ
左上のオブジェクト　(-0.5　0.5)　右上(0.5　0.5)　左下(-0.5 -0.5)　右下(0.5 -0.5)
という風に指定する。

・プログラム③

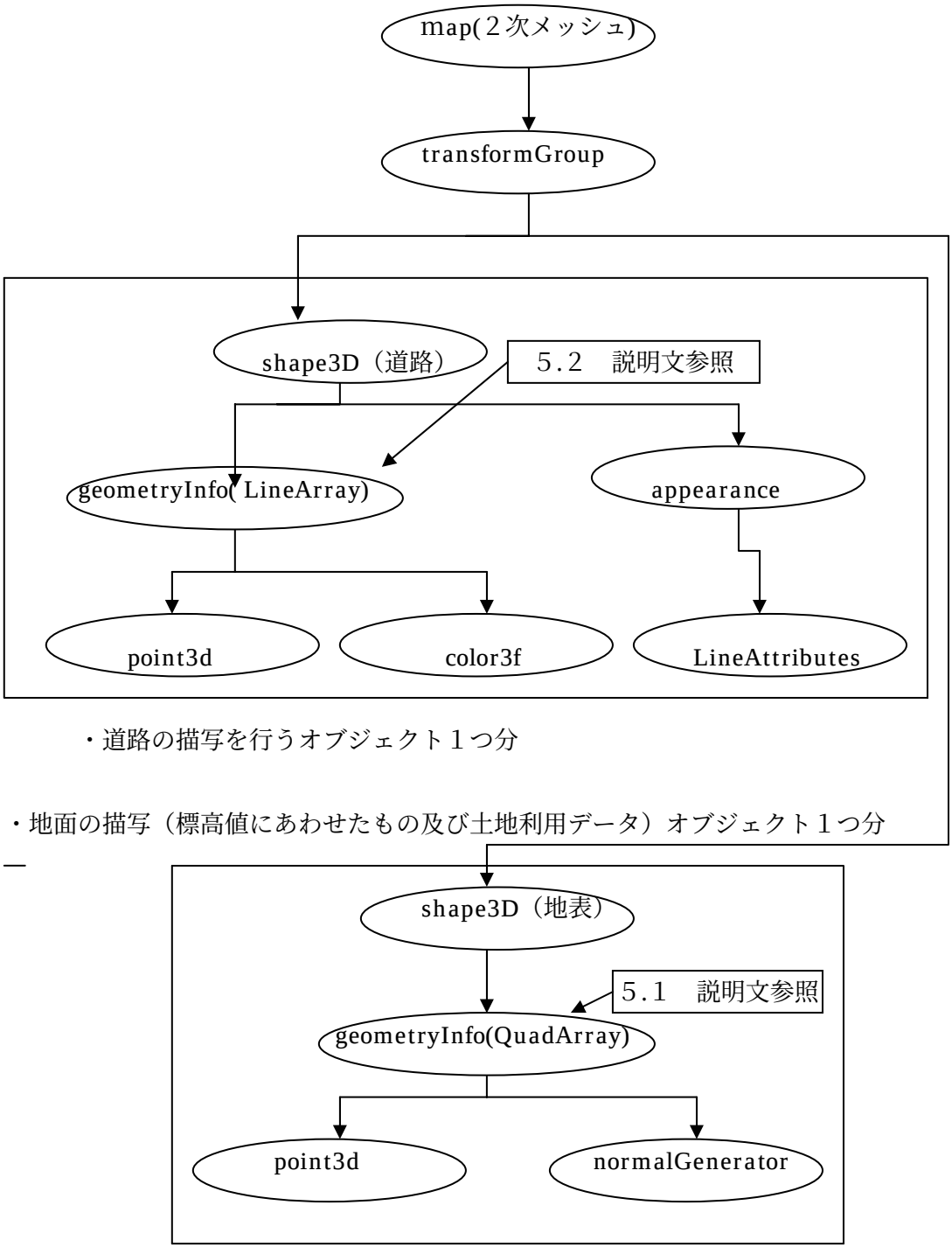


図 6.3-3 プログラム③のシーングラフ

四角で囲われた部分がそれぞれ最小のオブジェクトとなる。色毎に別のオブジェクトとして認識している。色の種類分それぞれのオブジェクトを作成し、transformGroup の下につなげる必要がある。

第６章　描写範囲の拡大

６．１　低解像度化

画面上で地図を描写する範囲を広げる。プログラム中では取り込むデータファイルの関係で２次メッシュ１つ分を最小の単位として表示するようになっている。

下記の様に１次メッシュの中には　８＊８　６４の２次メッシュが含まれることになる。加えて、国土数値情報データの土地利用データの描写のために、下、右端のデータを取ってこなければならないので５２３４、５１３４、５０３２、５０３３、５０３４のデータからも任意の部分だけ取り出す必要がある。

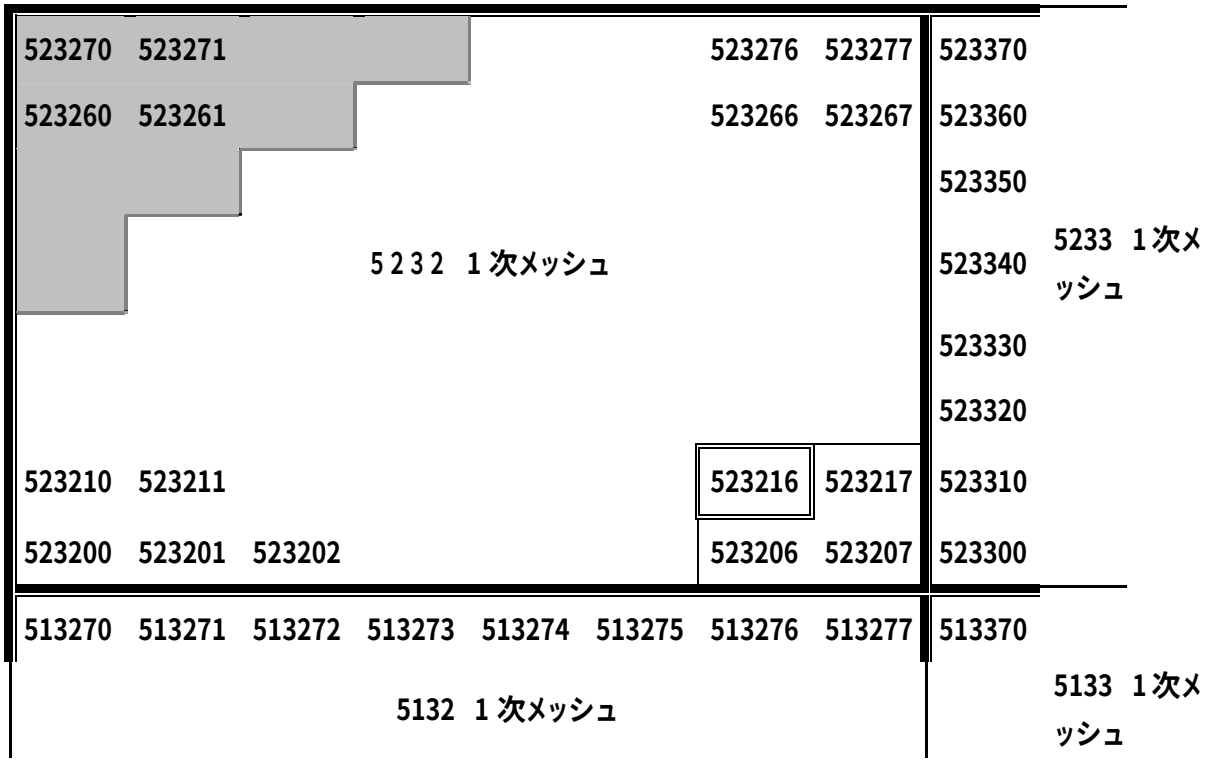
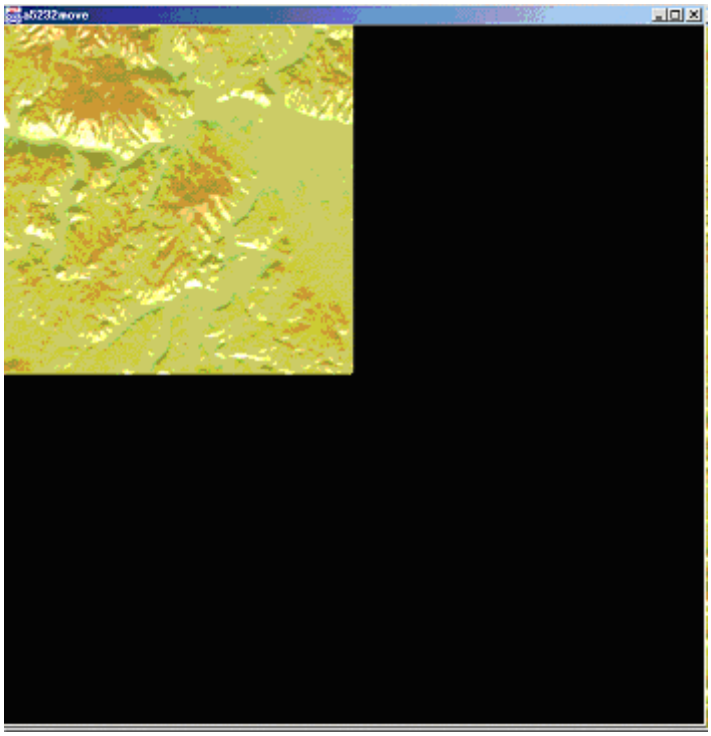


図 6.1-1 隣接する１次メッシュ・２次メッシュの位置関係

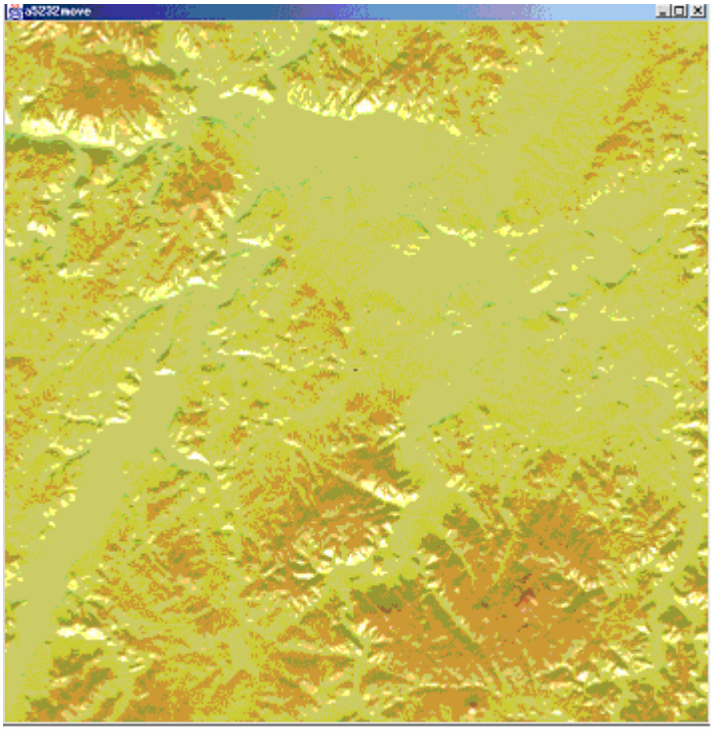
太線で囲まれている部分が 5232　の１次メッシュ、右下の 2 重線で囲ってある部分が今まで描写してきた 523216（三次市）の 2 次メッシュにあたる。

上記したように 2 次メッシュ 1 つ分を単位をして、それを拡大・縮小・座標指定を行い組み合わせて任意の範囲の描写を行う。

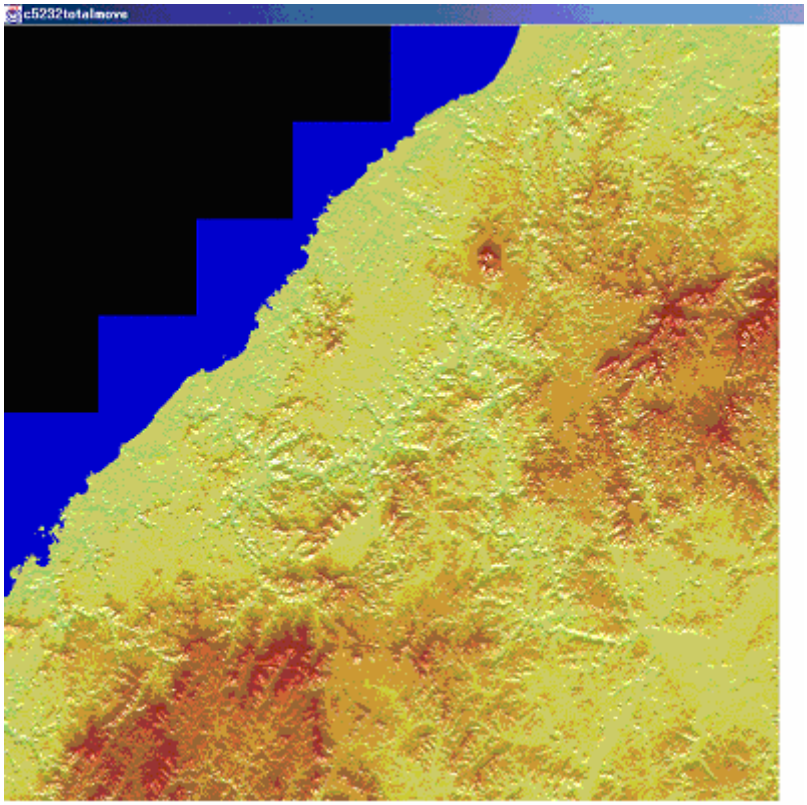
（1）↓表示の単位になる２次メッシュの１つ分の表示



（2）↓＊４つ分を合成したもの



プログラム１から　それぞれの２次メッシュ（4つ分）を表示するプログラム
４つ呼び出して合成を行っている。
　呼び出す側で座標変換、倍率の調整を行って最終的な表示を行っている。
　複数の２次メッシュ（部品）を一つの物体として認識させるために
当初
２次メッシュの地図作成　→　統合、ピッキング、照明の調整、位置．倍率の設定、表示
上記の二つに分割していたプログラムを
２次メッシュの地図作成　→　統合、位置．倍率の設定
　　→　ピッキング、照明の設定、表示
３つのプログラムに分割して、表示の地図を一つのクラスにまとめて、表示用のプログラムに渡している。
上記（１）（２）と同じ要領で（縮小、座標変換を行い　８＊８　１次メッシュ分のデータを表示する。
（３）１次メッシュ分の範囲の表示



ここまでの流れで上記の図が表示できた。
しかし表示の際処理が非常に重く、実行から結果が出るまでに１０～１５分程度かかる、加えて描写後に拡大、縮小等の処理操作を行うとエラーで処理が中断してしまう場合もある。

６．２　低解像度化への設計

広範囲の地図を作成する場合に前章の問題点を解決する為プログラムの設計を変更する。

現在使用しているモニターは　SVGA　1024*768　ドット　で画面を描写している

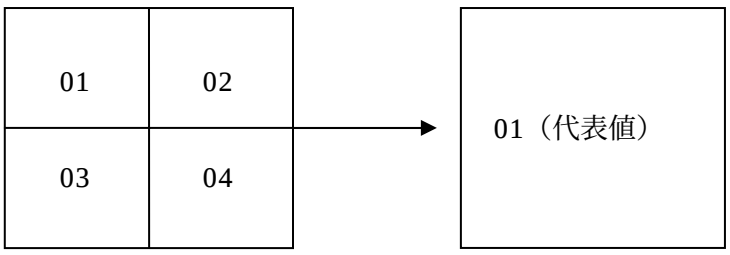
786432

2 次メッシュ 1 つ分を画面に表示した場合、200*200 の四角形を描写している。
1 次メッシュ全体を描写しようとすると、2 次メッシュを 8*8 で並べるので、
画面内に 1600*1600 の四角形を描写していることになる。

2560000

実際プログラムでは画面内に正方形の Window を作りその中に描写しているので
おおよそ 700*700 以下で描写を行っているものと思われる。

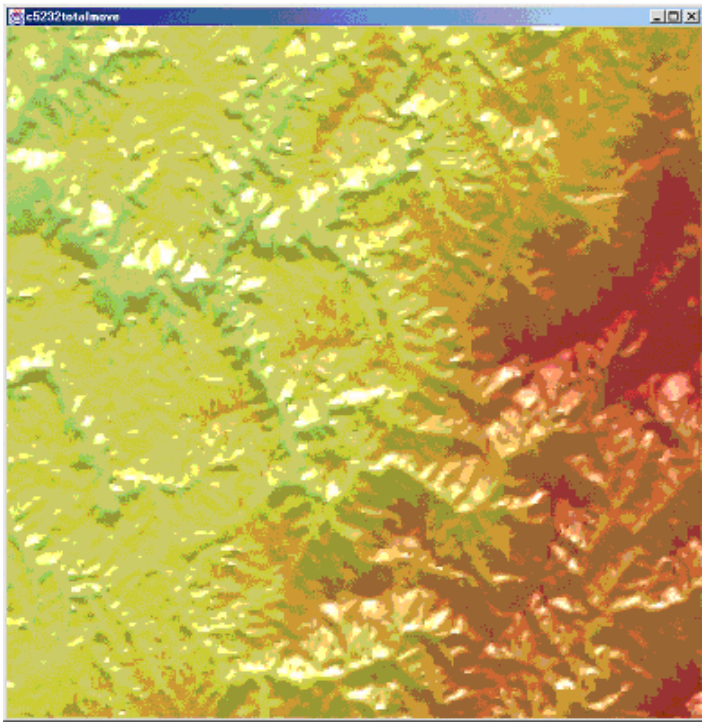
実際に描写できる 5 倍以上の処理を行っていることになる。つまり、画面に反映されない不必要なデータと処理が多すぎ、処理速度を遅らせている。
よって広範囲の地図を表示する場合、表示できない分のデータを取り込む各データと、プログラム中での処理を省略して解像度下げ実際の描写に合わせる事にする。
描写できない余分なデータを一つにまとめて省略することで、データの量を減らし、
プログラムの負担を軽くする。国土数値情報データの場合は分割する前の元のデータを使用すれば良い。
５０ｍメッシュ標高データの場合



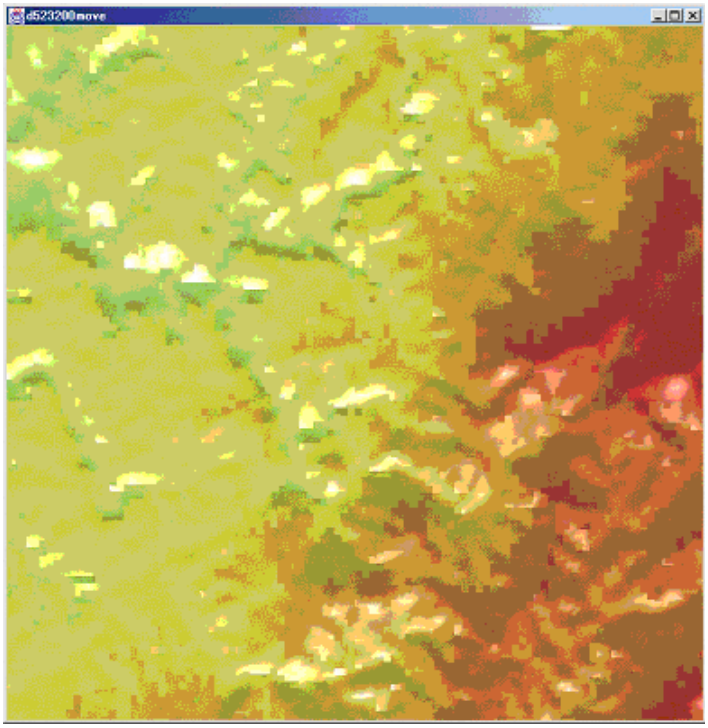
左の描写では見た目は 9 点で 4 つ四角形を描写しているように見えるが、プログラム中では、個々の四角形の頂点を切り離して考えているので１６点の（ｘ、ｙ、ｚ）座標を計算していることになる。
右図への省略で、4 点の座標を計算することになるので、計算量は最低でも半分以下になる。
※基本的に取り込むデータファイルを組み替えるだけでよい。
ただし、プログラムに使うデータファイルをあらかじめ作成しておく必要がある。
解像度を下げると広い範囲を見た場合は特に問題は無いが、狭い範囲を描写したとき画像が荒くなる。描写する範囲の大きさに合わせて元のと省略版と 2 種類データファイルを用

意しておく必要がある。

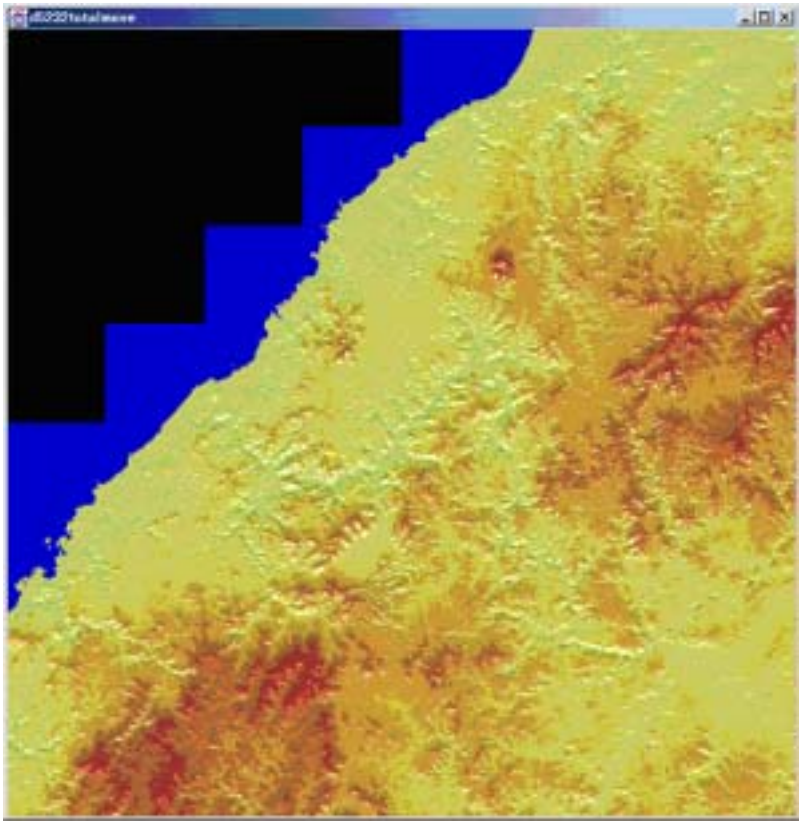
（1）狭い範囲用　省略無しデータを使用した図



（2）広範囲用　省略した（解像度を下げた）データを使用した図



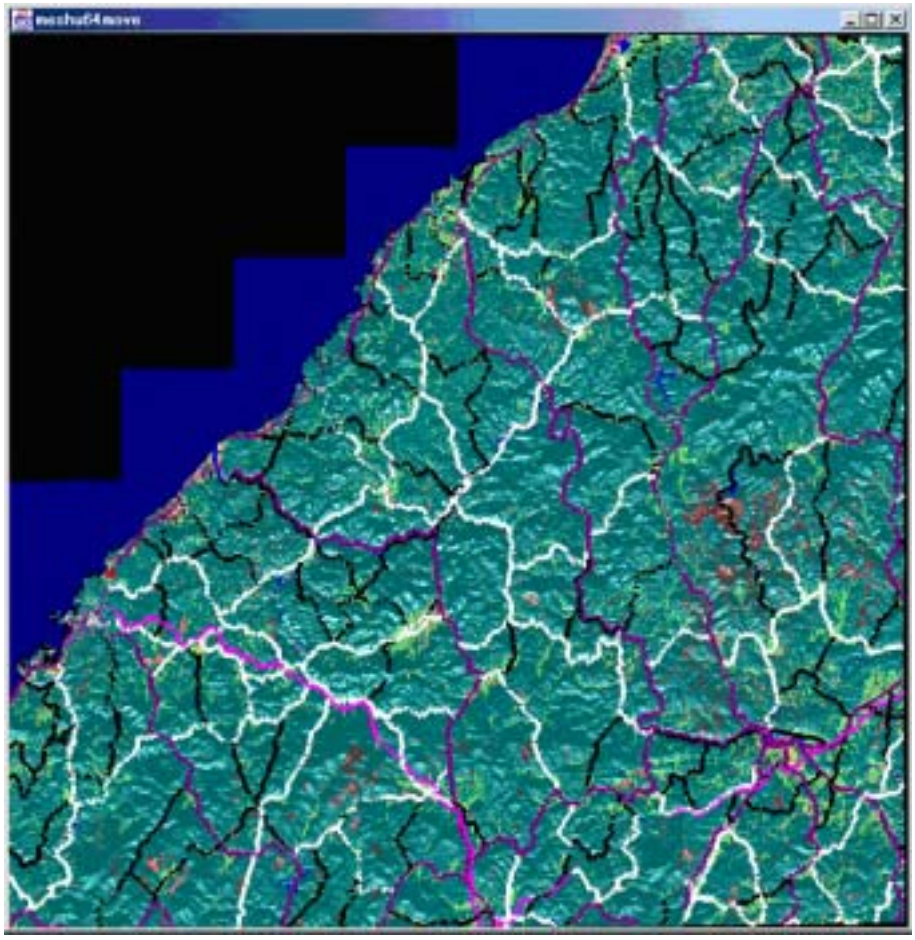
（３）　（２）と同じデータを使用して１次メッシュの描写を行った図



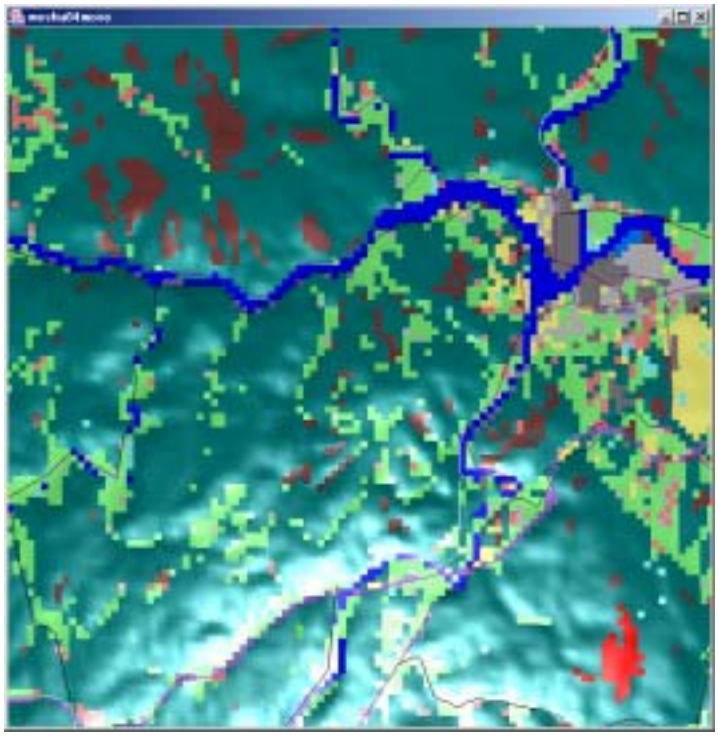
拡大縮小等の処理も行うことが出来る。
実行から描写まで３分程度　前回の１/5 程の実行時間で行うことが出来る。見る限りは以前（データを省略する前の）の実行結果とほぼ変わらない外見になる。

前回述べたとおり地図の表示される画面は約７００＊７００程度のドットが入る。
前回までの地図は　1600*1600 ＝ 2560000
今回の場合は　　800 * 800 ＝ 640000
かなり無駄な部分が省略されていると思われる。最適に近いデータ数を思われる。

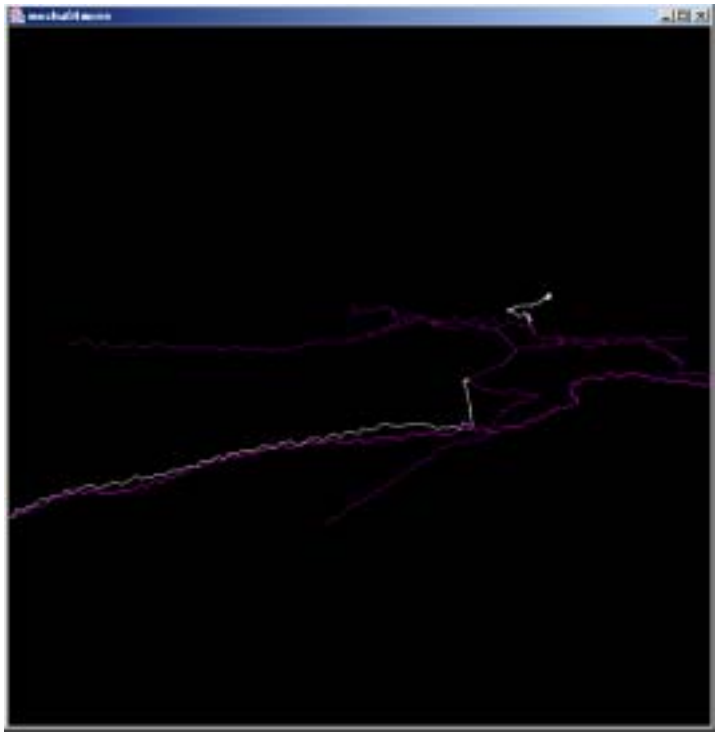
低解像度化した後 50m メッシュ、Jmc 道路データ、国土数値情報データを重ねたもの



２次メッシュ分の立体土地利用＋道路データを正面から見たもの



上と同じものを回転させ水平に近い角度から見たもの



※光に対して平行に傾いているので法線設定のある地表の部分は殆んど見えない

第７章　終論

終論として本研究の今後の課題や展望等を述べる

・画像のＳ/Ｃ上での実行

本研究で使用したデータは、５０ｍメッシュ標高データ、JMC マップ（道路）データ、国土数値情報（土地利用）データの３種類である。何れもオリジナルはもちろん編集後も大量で、個々のクライアントで扱うには負担が大きいデータである。その為、サーバー・クライアント環境での使用が有効である。本研究は Java3D　つまり Java を使用して行っているためハードウェアに依存しない。Java であるからサーバにアプレットの組み込まれた html 文章を設置し、クライアントからそのブラウザ上でＵＲＬを指定する事で、プログラムのデータをクライアントに落とし実行、結果を見ることが出来る。上記した通り html 文章にアプレットを組み込むことでも、サーバ側だけにデータ・プログラムを置いてクライアントで使用、実行することが可能である。しかし、この場合 WWW ブラウザでページを指定した時点で Java アプレットが自動的にクライアントにダウンロード→実行、つまりクライアント側に実行の負荷がかかっている事になる[9]。最も理想的なのはサーブレットを使った実行である。サーブレットでの実行ならば、サーバ側でプログラムの処理を行う、つまり実行の負荷はサーバ側にかかることとなる。調べた限り Java3D に結果を他のマシンに描写する機能は無いと思われる。別の形（JPEG 等）に実行結果を変換→クライアントに転送→クライアント上で表示となる。

・負荷のかかるマシン

クライアント　→　HTML に埋め込んだアプレットを参照でプログラムを実行
HTML ブラウザ上で結果が表示される。

サーバ　→　サーバ上で実行したプログラムの結果（３D 画像を）
JPEG 形式などに変換　データとしてクライアントに
送信してクライアント側で表示させる

こうする事でサーバに置いたプログラム及び付随のデータを複数のクライアントから、利用することができると思われる。さらに本研究の様にマシンに負担の大きいプログラムの場合、複数のクライアントから要求があり、サーバ側に過度の負担が掛かる様な場合、サーブレットの実行からアプレットの実行に切り替え、負担を分散するのが望ましいと思われる。

謝　辞

本研究にあたり、最後まで熱心な御指導をいただきました田中先生には、心より御礼申し上げます。また、田中研究室の辰巳さん、貫目さん、高木さん、田辺さんには、本研究に関して数々の御協力と御助言をいただきました。厚く御礼申し上げます。なお、本論文，本研究で作成したプログラム及びデータ，並びに関連する発表資料等のすべての知的財産権を，本研究の指導教官である田中教授に譲渡致します。

引用文献

[1]　http://www.atmarkit.co.jp/fjava/special/web3d01/web3d01.html

[2]　http://www.sys.cs.meiji.ac.jp/~masao/Board/link/4-3/sld001.htm

[3]　独習 Java ジョゼフ･オニール著　（1999 年）

[4]　http://www.javaopen.org/jfriends/index.html

[5]　数値地図ユーザーズガイド　監修　建設省国土地理院　（平成４年７月１日）

[6]　数値地図５０ｍメッシュ（標高）日本-Ⅲ 国土地理院 CD-ROM Index.htm

[7]　JMC マップ（日本）（財）日本地図センターCD-ROM　Readme.txt

[8]　Java3D グラフィックス入門　田中成典編集　（2002 年６月）

[9]　http://home.catv.ne.jp/dd/chiba/ken/Java/Applet.html