卒業論文

Javaによる3Dアニメーション

氏名：植野 実直
学籍番号：2210940046-0
指導教員：山崎 勝弘 教授
提出日：1998年2月20日

立命館大学 理工学部 情報学科

内容梗概

図形はコミュニケーションの手段として、大変便利なものであるが、しかし、コンピュータ上で扱うには文字で行っていたときとは比較にならないぐらい大量のデータとより複雑な情報処理技術を必要とする。最近、コンピュータの性能向上に伴い、コンピュータ上でグラフィクスが多く利用されるようになってきている。こうした中で立体映像の研究が盛んになってきている。

本論文では、Javaを用いた3Dアニメーションシステムの作成について述べる。立体表示のための方法には多くの方法があるが、本研究ではアナグリフ法を用いた。

目 次

1 はじめに 1

2 3D アニメーション 3

2.1 立体視の原理 3

2.2 3D 表示の方式 4

2.2.1 アナグリフ方式 4

2.2.2 偏光方式 4

2.2.3 時分割方式 5

2.2.4 濃度差方式 6

3 Java 言語 7

3.1 抽象ウィンドウツールキット (AWT) 8

3.2 画像表示方法 9

4 3D アニメーション表示システム 10

4.1 入力部 10

4.2 処理部 11

4.3 画像表示部 14

5 実験と評価 15

5.1 実験方法 15

5.2 実験結果 16

5.3 評価 17

6 おわりに 18

参考文献 20

1 はじめに

図形は文字に比べ人が理解しやすく、万国共通の非常に便利な表現方法である。しかしながら、図形をコンピュータで利用するためには、文字とは比較にならないぐらい大量のデータと高速の処理が必要とされる。このため、これまでのコンピュータでは、主に文字が利用されてきた。しかし、最近の半導体技術、コンピュータ技術の発達、各種の画像処理の手法の確立により、コンピュータでの画像の利用が一般化してきた。これまでの画像の利用のほとんどは、2次元画像であったが、近年、よりリアルな画像の実現や、バーチャルリアリティ(仮想現実感)の要素の一つとして、3次元(3D)画像が利用されるようになってきた。宇宙空間や極限状況にある機械を遠隔操作する、架空の世界を作り、これをコンピュータグラフィックスによって表示しアニメーションさせる、あるいは、CADにより設計された物を3次元画像として表示させることによって、現実にその物体を作らずにその形状を把握する、など多くの分野で利用されている[2][3][8]。

JavaとはSun Micrsystemsが1995年に発表したオブジェクト指向言語であり、特徴としては、

- 中間言語であるバイトコードに変換してインタプリンタで実行することも、コンパイルしてバイナリコードにして実行することも可能。
- 他のプログラミング言語と違いソースレベル、バイトコードレベル共にプラットフォーム独立すなわち、多くの機種のコンピュータで共通のプログラムを利用できる。
- 最初からネットワーク、インターネットで使用されることを考慮されており、そのためのオブジェクトクラスが用意されている。
- 単体で実行できるアプリケーションとWWW上で使用されプラウザ上で実行されるアプレットの2つの形態を持つ。

等があげられる。こような特徴を持つため、最近活発に利用されており、バイトコードをハードウェアで実行できるJavaマシンも各社から発表されている[5][6]。

3次元映像の研究は、平画面を利用した擬似的な物から考えると二千年以上の歴史がある。このうち、両眼に視差のある画像を与えることにより、立体感のある画像を表示することができる。視差のある画像を与える方法としては、2枚の画像を光学系を用いて両眼に供給するステレオスコープや、1枚の画像に2色で画像を書き、それを色フィルタで分離して両眼に与えるアナグリフ方式など多くの方式が考案されてきた[2][3][8]。

本研究では両眼視差を利用した3D画像表示法の一つであるアナグリフ方式(2色眼鏡方式)を利用した3Dアニメーションシステムを Java 言語により実現する。

すなわち、左目用画像、右目用画像を別々に計算しておき、これを一枚の画像に合成したものを画面に表示し、色フィルタを付けた眼鏡で分離し各々の目に供給することにより実現される。

また、このシステムでは物体、視点のデータのからリアルタイムで画像を生成することにより、視点の移動、物体の回転をアニメーションする。

本論文では, まず第2章で3Dアニメーションについての考察を述べる。第3章では,Java言語に関して述べる。第4章では, 作成した3Dアニメーション表示システムのシステム構成について述べる。第5章では, 作成した3Dアニメーション表示システムを実際にテストし, その性能評価について述べる。第6章では, 本論文の結言と今後の課題を提示する.

2　3D アニメーション

2.1　立体視の原理

人は、「立体感」言い替えれば、奥行を知覚する能力を持つが、これは、大きく分けて視覚的な要因と視覚外の要因にわけられる。視覚的な要因としては、生理的なものと心理、記憶的なものが存在し、視覚外の要因としては、聴覚、嗅覚、触覚などが関係する。

3 次元映像技術は大きく分けると、平画面を利用した擬似的な物、両眼視差を利用した物、ホログラフィのように空間像を形成する物、の三種類に分類される。平画面を利用した疑似的なものは、遠近法、物の大小、コントラスト等により、目の錯覚、あるいは記憶による思い込みを利用して、立体感を出す。両眼視差を利用したものは、図 1 の様に両眼に視差のある画像を供給することにより、立体感を出す。しかし、この方式では焦点調整を伴わないため、画像が不自然に見えることがある。ホログラフィは光の干渉縞で物体の情報を記録、再生するために、画像の振幅、すなわち明るさの情報だけでなく、位相すなわち奥行の情報も記録、再生することが出来る。

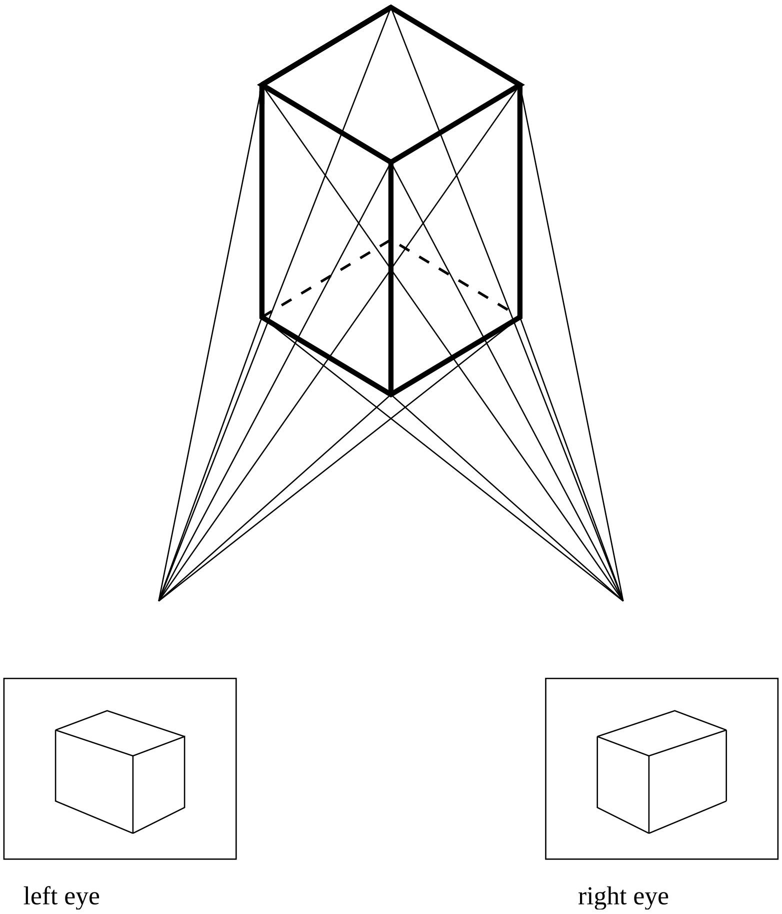


図 1　両眼視差

2.2 3D 表示の方式

2.2.1 アナグリフ方式

アナグリフ方式とは2色眼鏡方式とも呼ばれ、色フィルタを用いて、左右の目に両眼視差のある2枚の画像を供給することによって、奥行を認識させることが出来る。
アナグリフ方式では、左目用に青で書かれた画像と右目用に赤で書かれた画像の2種類の画像を用意し、それを1枚に合成し、その画像を左目は赤色フィルタ、右目は青色フィルタを通して見ることによって左右の目に違う画像が供給され奥行が認識される。

- 長所
  - 特殊な装置等を必要とせずに立体画像が得られる。
  - (コンピュータで生成する場合)2色なので計算量が少なくて済む。
- 短所
  - フルカラーが実現できない。
  - 青色フィルタは光量損失が大きく明るさやコントラストが十分でない。

2.2.2 偏光方式

偏向方式による立体映像の表示法は、両眼視差を持った左眼用、右眼用映像を2台のディスプレイに直交(直角)する偏光板を装着して画像を投影する。この画像を偏光メガネを通して見ると、ディスプレイの偏光板と一致した方の目にのみその像が見えるので両眼視差が発生する。現在最も広く用いられている方式である。図2に偏向式立体ディスプレイの構成図を示す。また、長所、短所としては以下のようなものがある。

- 長所
  - 複雑な回路、装置を必要としない。
  - カラー画像、動画も表示可能。
- 短所
  - ディスプレイの幾何学的な歪みが十分小さくなければならない。
  - 偏光板の角度をうまく合わせないと、片方の目で両方の画像が見えてしまう。
  - 頭を傾けると2枚の視差像の分離が悪くなる

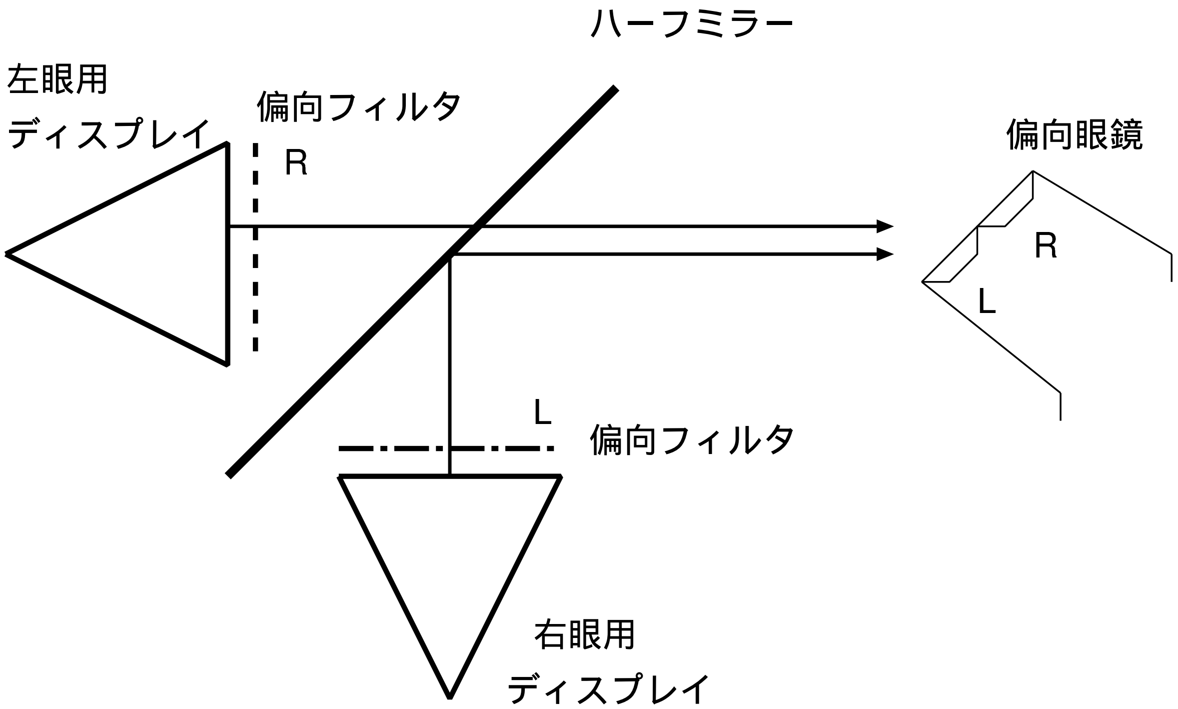


図 2 偏向式立体ディスプレイ

2.2.3 時分割方式

時分割方式とは、時間的に交互に左右の眼に対応する画像を入力する方式である。この方式では、1 台のディスプレイに左眼用、右眼用画像を交互に切り替えて表示する。この画像を眼に付けた液晶シャッタにより分離し左右の眼に入力する。切り替え周期は通常フィールド周期の 1/60 秒である。

- 長所
  - 左眼用、右眼用画像を 1 台のディスプレイ上に表示するため、幾何学的歪みが発生しない。
  - 眼の疲労が少ない。
  - 表示が 1 台のディスプレイですむので、経済的で省スペース。
- 短所
  - シャッタの開閉周期によっては、眼にちらつきが感じられる。
  - 眼鏡のシャッタに要求される条件が厳しい。

2.2.4 濃度差方式

濃度差方式とは、網膜から脳への信号の伝達は、画像が暗い時は遅くなる (プルフリッヒ効果)。そこで、横方向に移動する物体を含む画像を片目のみ暗くなるようなメガネを通して見ると左右の目の時間差によって両眼視差が発生する、このことにより立体画像を表示する。

- 長所
    - 通常の2次元画像とディスプレイがそのまま使える。
- 短所
    - 動画像でないと立体的に見えない。

3 Java言語

Java言語とはSun Micrsystemsが1995年に発表したオブジェクト指向言語であり、特徴としては、

(1) クラスに基づくオブジェクト指向言語

Java言語はクラスに基づくオブジェクト指向言語であり、クラスの概念を素直に表現することが出来る。またグローバル変数や独立したルーチンを持たず、クラスとインターフェースの集まりだけで表される。

(2) バイトコード方式

Javaプログラムを実行するには、ソースコードをコンパイルすることで得られるバイトコードをインタプリンタで解釈し処理する。バイトコードはポータブルであり、特定のプラットフォームに依存しない。

(3) 静的な型を持つ

コンパイル時に可能な限りエラーを見つけることが出来る。また、コンパイラの最適化に利用できる。

(4) ガーベジコレクション

動的に確保したメモリ領域は、参照されなくなると自動的に回収される。メモリ管理を自動化することによって、バグが発生する可能性が減少する。

(5) 頑健性

Java言語にはCやC++にあるようなポインタ演算がなく破壊的なプログラムになりにくく、ポインタ操作に関するバグの危険性がなくなる。

(6) 可搬性

Javaには、ソースコードレベルとバイトコードレベルの2つの可搬性があり、ソースコードレベルの可搬性は、特定のプラットフォームに依存しない言語仕様や、プラットフォームの違いを吸収するオブジェクト指向技術により実現されている。バイトコードレベルの可搬性は特定のプラットフォームに依存しない仮想機械で実現されている。

(7) マルチスレッド

マルチスレッドに基づく並行プログラミングをサポートしている。

等があげられる。こような特徴を持つため、最近活発に利用されている。

3.1 抽象ウィンドウツールキット (AWT)

AWT(抽象ウィンドウツールキット) は、Java 言語の特徴である可搬性を実現する為に、それぞれ異なるネイティブなウィンドウシステムを持つ様々なシステムで単一のウィンドウインターフェースを実現する。AWT の動作を図 3 に示す。

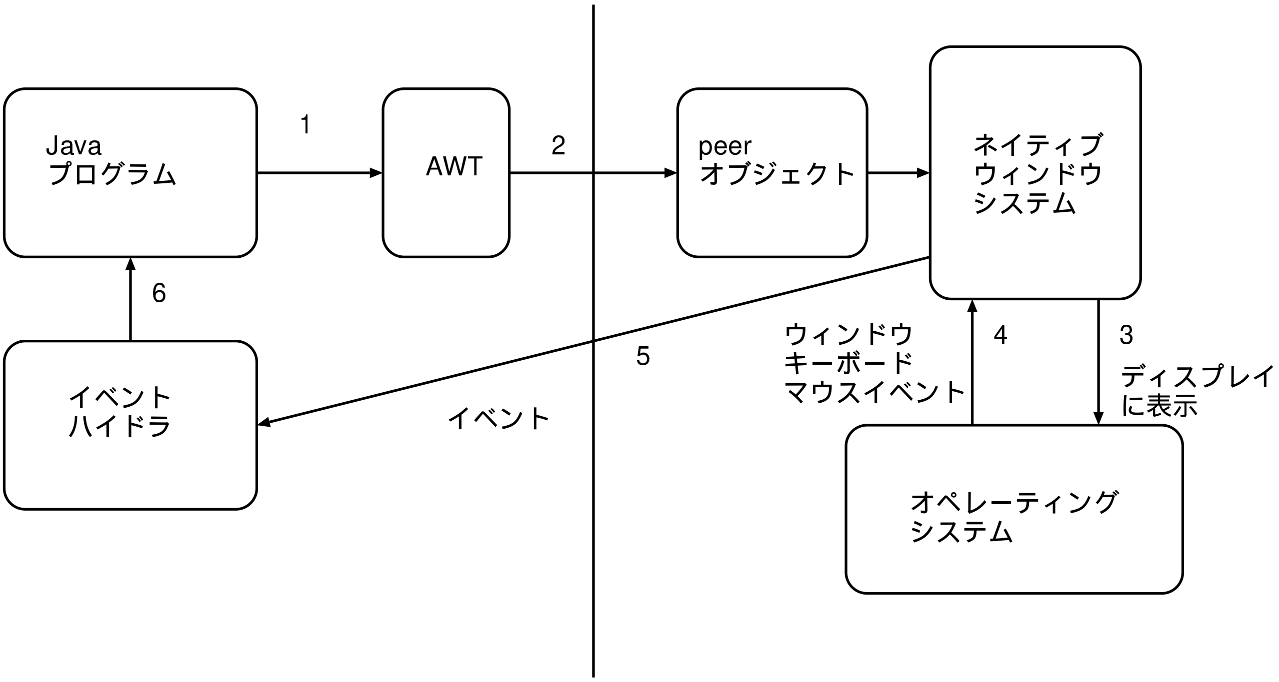


図 3 AWT の動作

1、Java プログラムがボタンをスクリーン上のウィンドウに追加するコールを行う。

2、AWT の実行ルーチンがネイティブウィンドウシステムにリクエストを発行する。

3、このリクエストはボタンをスクリーンに表示するリクエスト。

4、ボタンがマウスによって押される。

5、ネイティブウィンドウシステムが Java のイベントハイドラシステムに渡す。

6、イベントハイドラから Java のプログラムシステムに渡す。

3.2 画像表示方法

Javaには、Graphicsオブジェクトに文字、線、図形を描画するための多くのメソッドが用意されている。通常はこれらのメソッドを用いて描画を行い、repaintメソッドを用いて画面の書き換えを行う。

4　3D アニメーション表示システム

本研究において作成したシステムの構成を図?に示す。

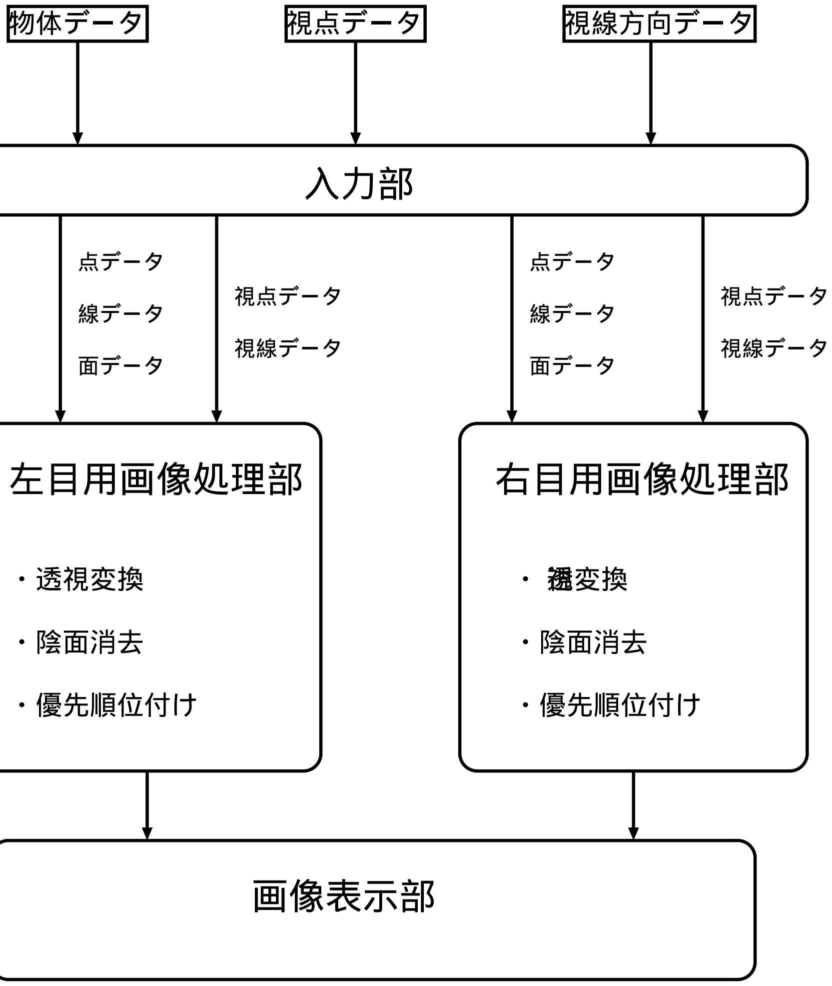


図 4 システム構成

本システムは図 4 の様に 4 つの部分よりなり、入力部ではデータの読み込みを行い、左目用画像処理部、右目用画像処理部で画像を作成し、それを画像表示部で画像表示部で表示する。

4.1　入力部

入力部では、物体データ、視点データ、視線方向データをファイルから読み込み、それらを点データ、線データ、面データ、視点データ、視線方向データに分離して、左右の画像処理部に渡す。

4.2 処理部

処理部では、入力部から渡されてきた3つのデータから画像の生成を行う。処理部では、投影変換、陰面処理を行う。

(1) 透視投影

渡されてくるデータはxyzの3次元座標で記述されているが、コンピュータの画面に表示するためには、xyの2次元座標でなければならない。このために、透視変換を行う。図5では、ディスプレイ面はxyz座標系のZ=0の面としている。また、右眼の座標をER(D2,0,D1)、左眼の座標をEL(-D2,0,D1)としている。このとき、三次元空間中の任意の点P(px,py,pz)を右眼で見たときのディスプレイへの投影点Rのx,y座標は、1点透視法によってそれぞれ次式で表される。

$$\left.\begin{aligned} x &= (D1/(D1-pz))*(px-D2)+D2 \\ y &= (D1/(D1-pz))*px \end{aligned}\right\} \tag{1}$$

これらの座標を、ディスプレイ面上での座標系であるxy座標(rx,ry) ($0\leq rx\leq 639, 0\leq ry\leq 399$)で表すと、次式のようになる。(ここでは縦400ドット横640ドットでディスプレイに表示するものする。)

$$\left.\begin{aligned} rx &= (D1/(D1-pz))*(px-D2)+D2*KX+320 \\ ry &= 200-(D1/(D1-pz))*px*KY \end{aligned}\right\} \tag{2}$$

ここで、KX、KYは単位長さ辺りのドット数、また、眼とディスプレイの距離D1=50 cmであり、人の両眼の間隔を6.4cmすなわちD2=3.2とすると(2)式は次のようになる。

$$\left.\begin{aligned} rx &= (50/(50-pz))*(px-3.2)+3.2*KX+320 \\ ry &= (50/(50-pz))*px*KY \end{aligned}\right\} \tag{3}$$

点P(px,py,pz)を左眼で見たときの投影点Lの座標(lx,ly)も、右眼の場合とまったく同様に1点透視法によって、次のように求めることが出来る。
式(2)の右辺でD2=-D2とおくと、次式が求まる。

$$\left.\begin{aligned} lx &= (D1/(D1-pz))*(px+D2)-D2*KX+320 \\ ly &= 200-(D1/(D1-pz))*px*KY \end{aligned}\right\} \tag{4}$$

また、式(4)の右辺の定数D1.D2,KX,KYを右眼の場合と同じようにすると次式が求まる。

$$\left.\begin{aligned} lx &= (50/(50-pz))*(px+3.2)-3.2*KX+320 \\ ly &= (50/(50-pz))*px*KY \end{aligned}\right\} \tag{5}$$

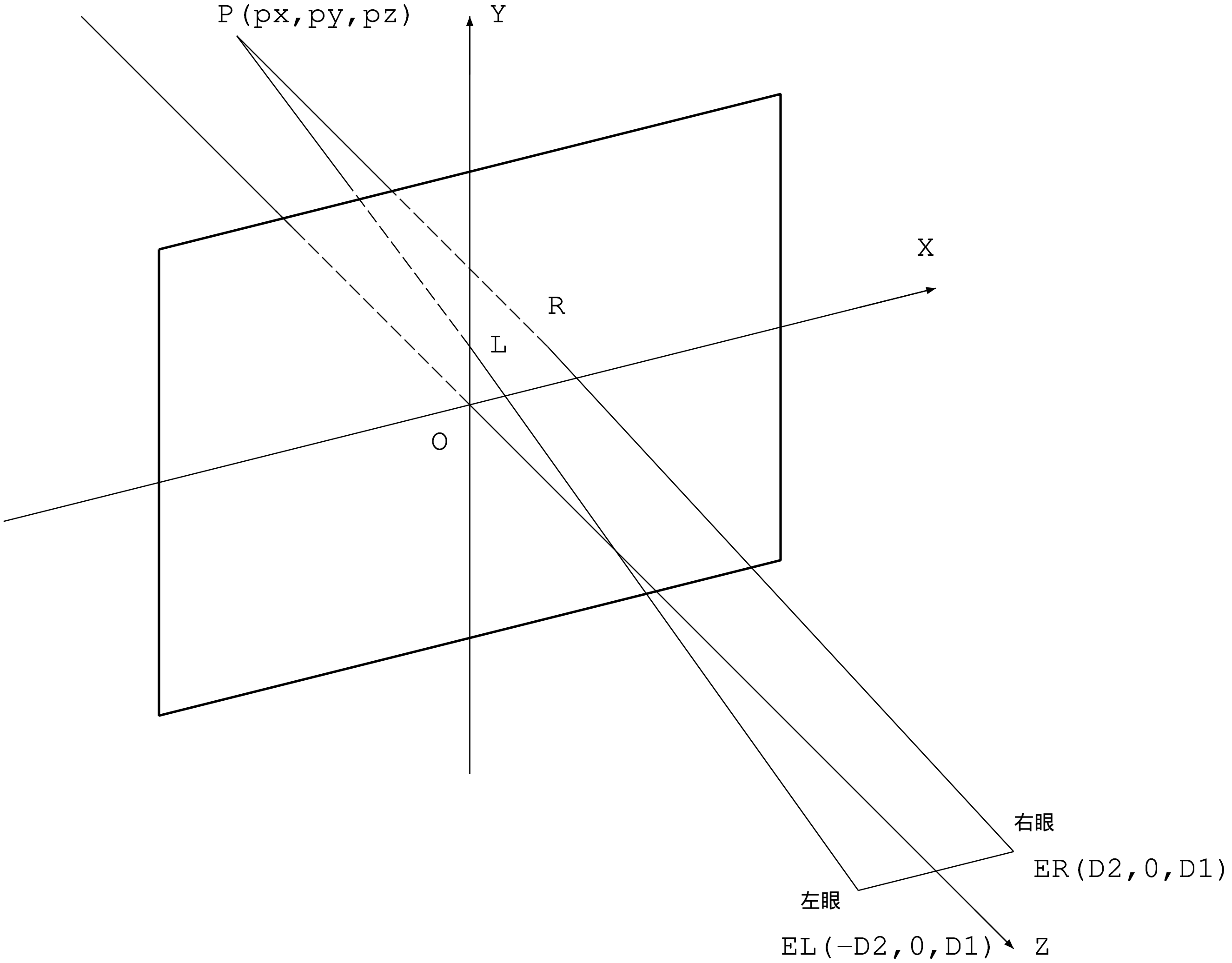
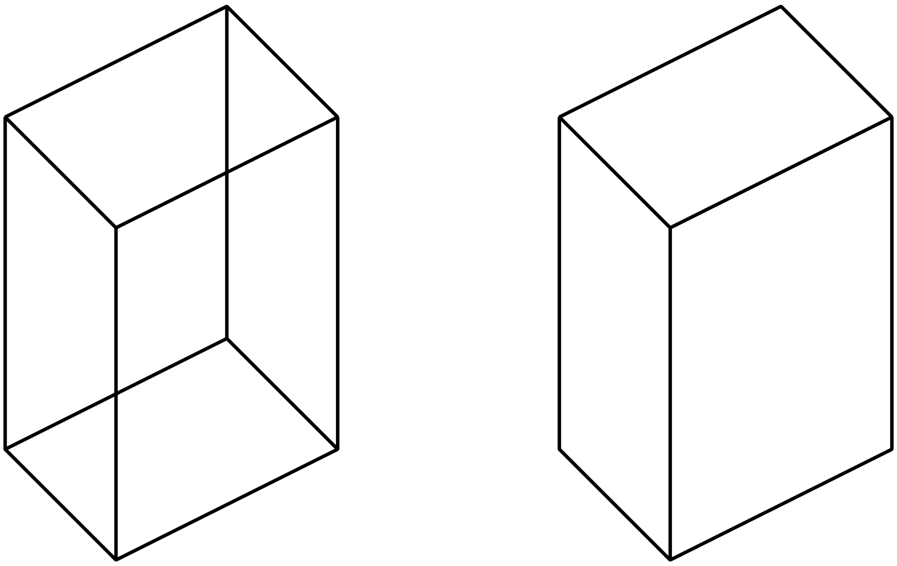


図 5 座標系と両眼の位置関係

(2) 陰面消去

たとえば、立方体のように、多数の面から構成されている立体の全ての稜線を透視変換して表示すると、図6の(1)に示す様に、本来隠れて見えない線も書かれてしまう。図6の(2)の様に視点側にある面の稜線だけを表示する方法が陰面消去である。



(1) 透視変換後の立体　　(2) 陰線処理後の立体

図6 立体表示

本研究では、多角形で近似された凸立体に対する陰面消去のアルゴリズムを用いた。このアルゴリズムは、平面を構成する点列の順序が時計回りか、反時計回りかのチェクだけで行われるという特徴を持っている。

はじめに、凸立体を多角形群で近似し、多角形を構成する各頂点を、物体の外側から見て時計方向の順序でコンピュータに入力しておく、たとえば図7に示す様な立体は、6個の四角形から構成され、もっとも手前の面は、時計回りのABCDの順による頂点列、また左側の面も、時計回りのADEFの点列として構成される。このように構成された物体は、ディスプレイ平面に透視変換された頂点列が、そのまま時計回りであれば”見える面（可視面）”、また逆に、反時計回りであれば,”隠れる面（不可視面）”となる。この処理を立体を構成する全ての面に対して行う。

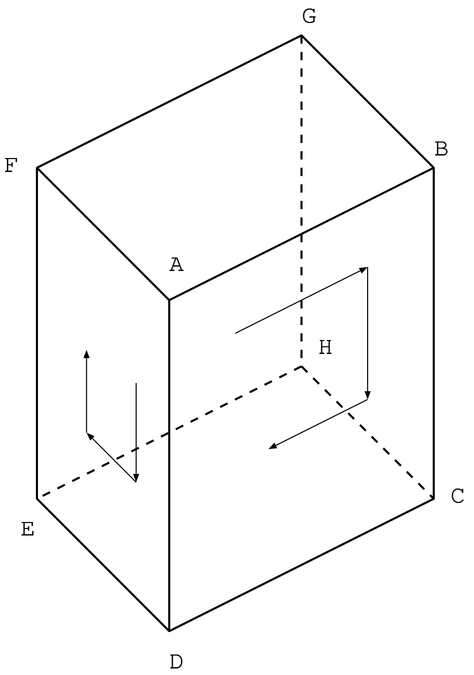


図 7 頂点列の作成

(3) 複数の物体の優先関係の判定

空間内に複数の物体が存在する場合、それぞれ個別に陰面消去された立体に対して、さらに、他の立体によって隠される部分の陰面消去の処理が必要になる。このために、視点と立体の距離を求め、より視点に近い立体ほど他の立体によって隠される可能性が少ないので、視点に近い順に優先順位をつけ、優先順位の低い順に重ね書きすることによって、最終的に陰面消去された画像が求まる。

4.3 画像表示部

画像表示部では、処理部において、陰面消去され優先順位を付けられた物体のデータを受け取り、優先順位の低いものから順に描画していく。そして、次の画面のデータが渡された時点で次の画面を描画する。

5　実験と評価

5.1　実験方法

2つのデータ(図8、図9)を用いて実験を行った。テストデータ1は立方体、テストデータ2は2つの立方体の間に直方体がある図形である。これらのデータ中の点と直線の数は、テストデータ1では点の数が8個、直線の数は12本、テストデータ2では点の数が24個、直線の数は36本である。
テストデータ1、テストデータ2、の各々について以下の項目のテストを行い、その結果を評価した。

1. 視点の移動時間 。
2. 一秒当たりの描画枚数。
3. 抜けている画像がないか ?

このテストでは、Z座標を1づつ10000まで増加させた。

なお、テスト結果は全て、富士通FMV DESKPOWER TP20(Intel Pentium Pro 200Mz,Mem 64MB)上で測定した値である。

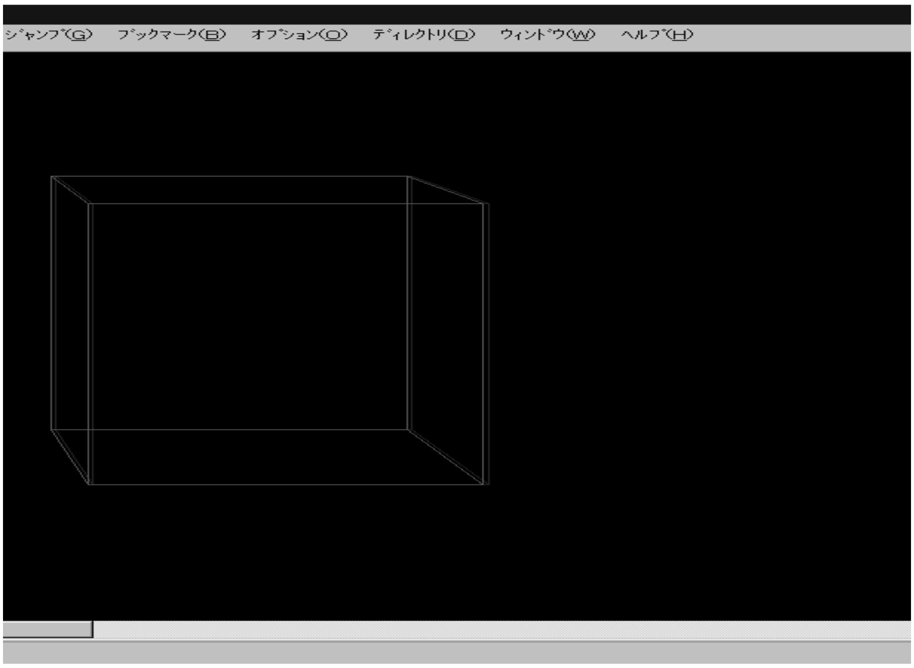


図8 テストデータ1

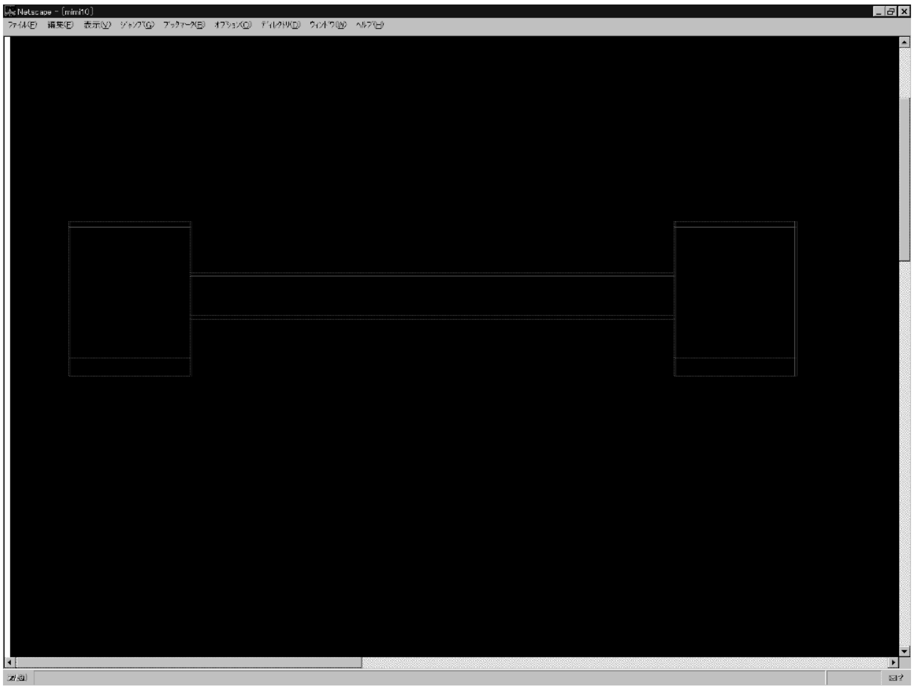


図 9 テストデータ 2

5.2 実験結果

1. 視点の移動時間

表 2 視点の移動時間

	時間 (秒)
データ 1	62,180
データ 2	74,700

2. 一秒当たりの描画枚数

表 3 一秒当たりの描画枚数

	枚数 (枚/秒)
データ 1	25,23
データ 2	21,51

3. 抜けている画像がないか ?

表4 抜けている画像がないか ?

	抜けた画像の枚数
データ1	8427
データ2	8393

5.3 評価

(1) 視点の移動時間

十分満足できる処理速度が実現できたと考えられる。

(2) 一秒当たりの描画枚数

通常のテレビ画像には及ばないもののアニメーションとして、満足できる水準に達していると考えられる。

(3) 抜けている画像がないか ?

画像の抜けはきわめて多く、全体の80割以上が抜けている。これは、Java言語のrepaintメソッドがCPUの余力があるときに画面の書き換えを行っているためと考えられる。確実に画面書換えを行う手法の確立が今後の課題だと考えられる。

6　おわりに

本研究では、3D アニメーションシステムを Java 言語により作成した。作成したシステムの問題点をとしては以下の物が挙げられる。

- 表示される画像よりか表示されない画像の方が多い。
- 画面の書換えによるちらつきが生じる。
- 使用するマシン、ブラウザソフトにより動作速度に大きな違いがあり、場合によっては動作しない。

今後の課題としては、

- 上に挙げた問題点の修正。
- 物体の入力形式、入力方法の改良。
- 陰面処理、優先度判定のアルゴリズムの改良。

が挙げられる。

謝辞

本研究の機会を与えてくださり、御指導いただきました山崎勝弘教授に心より感謝いたします。

また、数々の貴重な助言をいただきました大学院生の皆様方に感謝いたします。

参考文献

[1] 臼井 和敏, 片山 透: 詳解 JAVA プログラミング、CQ 出版社、1997

[2] 増田 千尋: 3 次元ディスプレイ、産業図書株式会社、1990

[3] Andrew S. Glassner: 最新 3 次元コンピュータグラフィックス、アスキー、1991

[4] 日向 俊二:入門 Visual J++、翔泳社、1997

[5] Laura Lemay,Charles L. Perkins:Java 言語入門 アプレット,AWT, 先進的機構、プレンティスホール出版、1996

[6] 戸松 豊和:Java プログラムデザイン、ソフトバンク株式会社、1997

[7] Stephen R. Davis:Microsoft Visual J++ セルフマスターブック、Micrsoft、1996

[8] 長谷川 伸:改訂 画像工学、コロナ社、1991

[9] 稲田 修一:三次元画像、昭晃堂、1991