グラフィクスとCAD 94－8
（1999．2．26）

Multiresolution による適応型 3D CG 表示システム

新井正敏　宮田亮介　村上公一
富士通研究所ヒューマンインタフェース研究部　〒674-8555 明石市大久保町西脇64番地

概要

Multiresolution 技術を用いて，ポリゴン数を任意に変化させながら変形アニメーションをリアルタイムに表示するシステムを開発し，機械性能から画像品質を設定することにより，状況に適応した表示を可能とした．このシステムは，低性能から高性能な機械まで同じアニメーションデータを使用して機械性能にあった画像品質で表示することが可能であり，キャラクタとカメラの距離関係に応じシームレスにテクスチャ付で表示を行うことで，画像品質を下げることなくレンダコストの削減を実現した．なお，シンプルなデータ構造を有しているため任意のポリゴン数を構築する計算コストを低く押さえ，かつアニメーションのデータ量の削減を行っている．

An Adaptive 3D CG Display System using Multiresolution

Masatoshi Arai, Ryosuke Miyata, and Koichi Murakami
Human Interface Laboratory, Fujitsu Laboratories LTD.

Abstract

Using a multiresolution technique, we have developed a real-time three-dimensional CG display system that plays animated scenes with a texture map by dynamically changing the number of polygons. Setting the graphics quality with machine specifications, this system can obtain an optimum number of polygons according to the distance from the camera to the character position and can seamlessly display the polygon animation. Thus, it is possible to reduce the rendering time while maintaining graphics quality. The amount of actual animation data is small, and it takes only a short time for the system to construct the polygon data with any number of polygons

1. はじめに

近年３Ｄグラフィックアクセラレータの普及やめざましいＣＰＵ性能の向上に伴いグラフィックスワークステーション（ＧＷＳ）はもちろん，パーソナルコンピュータ（ＰＣ）においても高速なリアルタイム描画が可能となってきている．このように急速に発展をとげる中でＰＣのソフトウエア（ゲームなど）を開発する場合，グラフィックアクセラレータの有無やＣＰＵの性能をもとに最低スペックマシンを確定し，最低スペックに適応したソフトウエアの設計（キャラクタのポリゴン数やフレームレートの決定）を行っているのが現状である．このため，最低スペックマシンとして低性能機械を選択した場合，ほとんどのマシンで動作する反面，高機能機械で動作させても，画像品質は低性能機械で動作させた場合とかわらない．逆に最低スペックマシンとして高機能機械を選択した場合リアルタイムに動作しなくなる機械が多くなってしまう．このため同じアニメーションを幅広い機械でリアルタイムに動作させるには機械性能に応じてレンダリング時間を下げる必要がある．レンダリング時間を下げるためにポリゴン数を下げる方法が考えられるが，現状では機械毎に任意なポリゴン数でアニメーションをリアルタイムに生成する手法は用いられていない．このため我々は機械性能に合わせた最適なポリゴン数をもつアニメーションをリアルタイムに作成するシステムを作成した．

2. 従来の手法

従来のアニメーション再生方法はモデルをポリゴンで作成し，そのモデルに変形をかけてアニメーションを作成し，再生時にキャラクタの位置を変えて表示をする方法をとっていた．ポリゴン数は一定であるため，カメラからキャラクタが遠く離れていてもレンダリング時間は削減することができない．レンダリング効率を上げるため，ポリゴン数を削減したモデルをあらかじめ作成しておき（通常３種類：カメラからの距離に応じ近距離，中距離，遠距離）カメラからの距離に応じてモデルを切り替える方法がとられる．例えば，MultiGen Inc. の GemeGenII [6] などを使用し，距離とポリゴン数を制御する付加データを作成する手法がある．しかしながら，これには大きく２つの問題が発生する．

(i) アニメーションデータをポリゴン数の種類（通常３種類）用意する必要がある．

(ii) ポリゴン数が変化する際，急激に変化する過程が目立ってしまう．

一方，近年，Multiresoution, Level Of Detail, Progressive Mesh の分野で，ポリゴンモデルに対し元の形状を維持してポリゴン数を連続的に減らす研究が盛んに行われている[1][2][3][4][5]．この技術は品質の高いポリゴンリダクションを実現しおり，機械性能別のキャラクタモデルを作り出す分野に使われている．しかし，この連続的に変化するポリゴンを使ってアニメーションデータを作成し再生する研究はほとんどなされていない．

このため，今回我々は，Multiresolution を使用してシームレスにポリゴン数を動的に変化させ，かつ１つのアニメーションデータを任意のポリゴン数で使用できるようなシステムを開発した．

3. 適応型 3D CG 表示システム

3.1. 機能

表示パラメータを基に表示するキャラクタのポリゴン数をダイナミックに変化させ，状況に適応した表示を行うことを目的とする．なお，実用化に向け，以下の点について保証する．

(i) ポリゴン数が変化してもアニメーションデータは共通である．

(ii) アニメーションのデータ量を極力少なくする．

(iii) 任意のポリゴン数に変化し，テクスチャを有することができる．

また，ここで述べている表示パラメータとは

(iv) グラフィックアクセラレータの有無やＣＰＵ性能による表示品質の度合い．

(v) カメラとキャラクタの距離による表示品質の度合い．

これらを実現することにより，高性能機械ではより詳細なポリゴンで表示することができ，低性能機械でもそれなりの表示を可能とする．また，同一機械内でも，カメラからの距離に応じて表示キャラクタのポリゴン数をダイナミックに変化させることにより画像品質を下げることなくレンダリング時間を削減することが可能となる．

3.2. 概要

適応型３ＤＣＧの全体図を図１に示す．本システムは３つの段階を持っている．

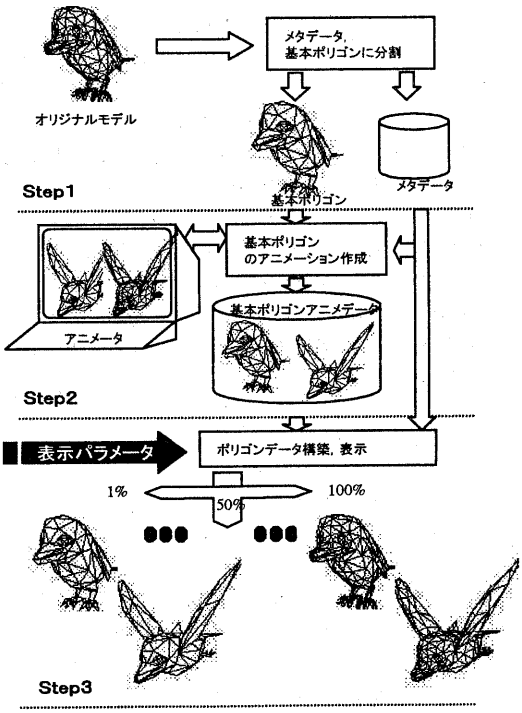


図1. 適応型3DCGシステム全体図

(i) Step1: モデリング段階

Multiresolution 技術を使用して，オリジナルキャラクタポリゴンモデルを基本ポリゴンデータとメタデータに分割する．これは

GWS などの機械を使用してモデル1体に対し1度行う.

(ii) Step2: アニメーションデータ作成段階

Step1 で作成した基本ポリゴンに対し変形を加え，アニメーションデータを作成する．これはGWS などを使用してアニメーションの数だけ実行する．我々は今回 Silicon Graphics Inc. (SGI) 上に SOFTIMAGE の plug in を作成しアニメーション作業を行うシステムを構築した．この plug in は基本ポリゴンモデルを変形させると，同時に任意のターゲトポリゴン（通常は 100%のポリゴンモデル）も変形し，アニメータにターゲットポリゴン変形イメージを伝えながらアニメーション作成作業が行えるように plug in を構築している．

ここで，アニメーションデータは基本ポリゴンで格納されるため，データ量が非常に小さくなる．

(iii) Step3: 実行段階

Step1 で作成したメタデータと Step2 で作成したアニメーションデータ，さらにマシン性能等のパラメータを基にポリゴン数を算出し，表示を行う．なお，メタデータはモデルに対して1種類，アニメーションデータは複数種類必要となる．実行はGWS，PCなどさまざまなマシンで実行が可能である．

なおここでメタデータと呼んでいるものはポリゴンを再構成するための頂点，面の情報を含んだデータであり，基本ポリゴンデータから任意のポリゴンデータを高速に再構成することができる．このメタデータについての詳細は次節で詳しく述べる．

4. データ構造

データは基本ポリゴンデータとメタデータに分けて管理しており，基本ポリゴンが変形した場合でも任意のポリゴンが再構築できるデータ構造になっている．

$$M^0 \xrightarrow{vsplit0} M^1$$

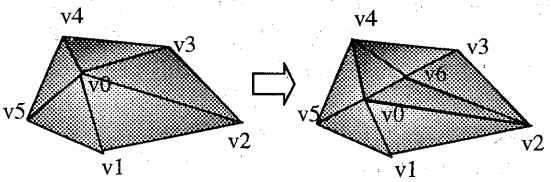


図2. Vertex split の例

なお，ポリゴンを再構築する際，高速に演算を行う必要があるためシンプルなデータ構造にしている．図2に vertex split する例を示す．図2では，v0 が分離し新しい頂点 v6 と位置を変更した v0 になる．

次にこの v6 データをどのように管理するかについて図3を使って説明する．

$$\mathbf{V20} = \overrightarrow{v2 - v0}$$
$$\mathbf{V30} = \overrightarrow{v3 - v0}$$
$$\mathbf{Vp} = \overrightarrow{v2 - v0} \otimes \overrightarrow{v3 - v0} \qquad \mathbf{V60} = r\mathbf{V20} + s\mathbf{V30} + t\mathbf{Vp}$$

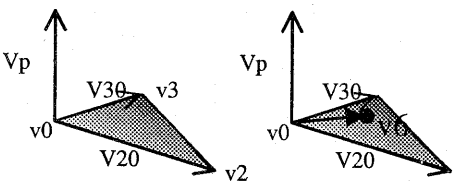
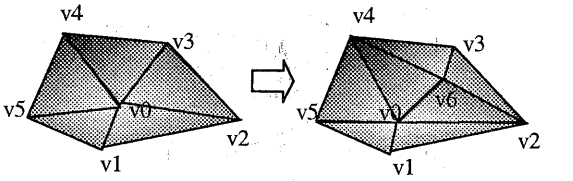


図3. 頂点 v6 の再生方法

頂点 v6 を新らたな頂点として挿入する際，v6 は v0 から分離し，新らたな面 {v6, v2, v3}, {v6, v3, v4}, {v6, v4, v0}, {v6, v0, v2} を造りだす．この面の中で，面 {v6, v2, v3}, {v6, v3, v4} はそれぞれ面{v0, v2, v3}, {v0, v3, v4} が変形したものである．このため v6 は v2 と v0 のベクトル V20 と v3 と v0 のベクトル V30，および V20 と V30 の外積をとったベクトル Vp で表表現できる．すなわち，それぞれベクトル V20, V30, Vp の r, s, t 倍の場所に新たな頂点 v6 を形成するようにデータとして r, s, t を保持することでポリゴンを再構成できる．このとき，{V20, V30, Vp} の代わりに {V30, V40, Vq} (Vq は V30 と V40 の外積) を用いることも可能である．Vp < Vq の場合は後者を選択し，逆の場合は前者を選択する．これは，外積値が大きいものを用いることにより r, s, t に誤差を少なくするためである．

$$M^0_{alt} \xrightarrow{vsplit0} M^1_{alt}$$



$$\mathbf{V20}_{alt} = \overrightarrow{v2 - v0}_{alt} \qquad \mathbf{V60}_{alt} = r\mathbf{V20}_{alt} + s\mathbf{V30}_{alt} + t\mathbf{Vp}_{alt}$$
$$\mathbf{V30}^{alt} = \overrightarrow{v3 - v0}^{alt}$$
$$\mathbf{Vp}_{alt} = \overrightarrow{v2 - v0}^{alt} \otimes \overrightarrow{v3 - v0}^{alt}$$

図4. 変形後のVertex split

ここで，今回のデータ構造を用いて，オブジェクトが変形した場合の例を見てみる．図2の頂点v0が移動した場合の例を図4に示す．頂点 v0 の位置が v0alt に更新された場合でも， 新たなベクトル V20alt, V30alt, そして Vpalt を求めそれぞれに対し r, s, t の値をかけることにより変形後 V6alt, V0alt を算出することが可能となる．

このように基本ポリゴンの頂点データの変形情報（頂点の移動）のみを格納しておき，必要に応じて再生すれば，変形後のポリゴンデータを任意のポリゴン数で得ることが可能となる．

次に，r, s, t の算出方法について述べる．V20, V30, Vp に r, s, t を掛けることにより，新たな頂点 v6 の座標 x-y-z 値が求められる．これを式に書くと

$$\begin{bmatrix} V20_x & V30_x & Vp_x \\ V20_y & V30_y & Vp_y \\ V20_z & V30_z & Vp_z \end{bmatrix} \begin{bmatrix} r \\ s \\ t \end{bmatrix} = \begin{bmatrix} V6_x \\ V6_y \\ V6_z \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} r \\ s \\ t \end{bmatrix} = \begin{bmatrix} V20_x & V30_x & Vp_x \\ V20_y & V30_y & Vp_y \\ V20_z & V30_z & Vp_z \end{bmatrix}^{-1} \begin{bmatrix} V6_x \\ V6_y \\ V6_z \end{bmatrix} \quad (2)$$

（1）式になる．ここで，（2）のように V20, V30, Vp の逆行列を v6 に掛けることにより，r, s, t を求めることができる．

5. 距離におけるポリゴン数の検討

ここでは，画像品質を1ポリゴンあたりのピクセル数で見積もり，画像品質が一定になるようなキャラクタのポリゴン数を算出する．

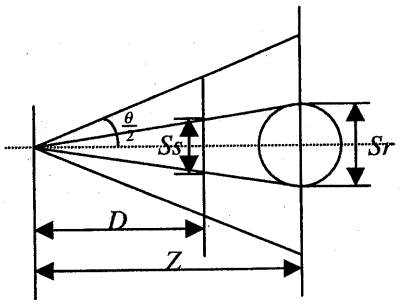


図5. キャラクタのディスプレイ面への投影

いま図5のようにカメラから距離 Z の位置にキャラクタが存在し，距離 D に Display 面があり投影されている場合を考える．なお，カメラから Display 面の画角は $\theta/2$ である．キャラクタを前面に投影した面積を Sr とし，これが Display 面上では Ss の面積になるとする．

$$Z^2 : Sr = D^2 : Ss \text{ より，} \quad Ss = \left(\frac{D}{Z}\right)^2 Sr \quad (3)$$

ここで，モデル単位系（ここでは cm）からピクセル単位系(pixel)に単位系の変換を行い，ピクセル単位系での面積 s を求める．

$$2D\tan\frac{\theta}{2}[cm] = w[pixels] \text{とし} \quad \alpha = \frac{w}{2D\tan\frac{\theta}{2}}[pixels/cm] \quad (4)$$

(3), (4) より，s を求めると，

$$s = Ss \times \alpha^2 = \left(\frac{D}{Z}\right)^2 Sr \times \left(\frac{w}{2D\tan\frac{\theta}{2}}\right)^2 = \left(\frac{w}{2Z\tan\frac{\theta}{2}}\right)^2 Sr \quad (5)$$

になる．ここで，ポリゴンはキャラクタに対し均一に分布していることを前提とし，全体のポリゴン数を Nap とした場合，投影されるポリゴンはほぼ，全体の半分と見積もれるため，投影されるポリゴンは Nap/2 である．ここで，1ポリゴンあたりの平均面積 Sp（平均ピクセル数）を求める．

$$Sp = \left(\frac{w}{2Z\tan\frac{\theta}{2}}\right)^2 Sr / \frac{Nap}{2} = \frac{w^2 Sr}{2Nap\,Z^2\tan^2\frac{\theta}{2}} \quad (6)$$

$$Np = \frac{w^2}{2Z^2\tan^2\frac{\theta}{2}} \frac{Sr}{Sp} \quad (7)$$

(6)において，Sp は画像品質とみなすことができる．この Sp を一定にし，ポリゴン数（Np）を求める式は（7）になる．（7）から，キャラクタのポリゴン数 Np はキャラクタとカメラの距離 Z の2乗に反比例することになることを示している．なお，画像品質（Sp）はマシン性能によるパラメータから設定することで，各マシンに適応した表示を行うことが可能となる．

我々が作成したキャラクタをアニメーションさせて，640x480 でレンダリングを行った結果を図6から図8に示す．使用したパラメータは，

$w^2 = 640 \times 480$

$\theta = 65\deg.$

$Sr = 75cm \times 59cm$

$Sp = 74.43 \quad [Z = 150cm, Np = 1000 poly から]$

であり，これから最適なポリゴン数 Np は

$$N_p = \frac{2250 \times 10^4}{Z^2} polygons$$

となる．図6から8の結果は上段から，1000 ポリゴン一定，Multiresolution を適用した場合（1000 ポリゴン一定と画像品質が同じ），Multireesolution を適用したモデルを横からワイヤーフレームで表示した結果である．最下段は距離 Z と 1000 ポリゴン一定と画像品質が等しいポリゴン数（緑）ならびに 1000 ポリゴン一定の半分の画像品質をもつポリゴン数（青）の関係を示している．これらの図から，Z に応じた最適なポリゴン数を算出しレンダリングを行うことで，ほとんど画像品質を劣化することなくレンダリングできることがわかる．

次に，具体的なレンダリング時間を計算してみる．キャラクタは Z=150cm から 275cm の距離を往復する．レンダリング時間は主にポリゴン数に比例する．（比例係数を k とする）．1000 ポリゴン一定の場合と Multiresolution を使用した場合で往路にかかるレンダリング時間を計算すると，

$$Tconst = k\int_{150}^{275} Np(z)dz = k\int_{150}^{275} 1000dz = 125 \times 10^3 k$$

$$Tmulti = k\int_{150}^{275} \frac{2250 \times 10^4}{z^2} dz = 68.2 \times 10^3 k$$

となり，Multiresolution を用いた場合，1.8 倍もレンダリングコストを押さえることができる．また最下段に示したグラフで青色の曲線のように遅い機械の場合画像品質を 1/2 にすることで，レンダリング時間をさらに倍増することが可能になる．

6. むすび

オリジナルポリゴンモデルを基本ポリゴンとメタデータに分けて管理することにより，データ量の削減と任意のポリゴン数をもつモデルの再構築を実現することが可能であることを示した．また，この手法を使用することにより，機械性能，カメラとキャラクタ（オブジェクト）との関係から最適なポリゴン数を算出することで画像品質を一定に保ちながらポリゴンアニメーション表示が行えることを示した．この手法を使うことにより，マシンスペックを上げることなく，レンダリング時間を短縮することが可能になり，今まで以上に，同時にリアルタイム表示できるキャラクタの数を増やすことが可能になると考えられる．

謝辞

本稿を執筆するにあたり（株）ビッグタウンズの吸原栄二氏にデータ作成等行って頂いた．心から御礼申し上げたい．

参考文献

[1] Michael Garland and Paul S. Heckbert. "Surface Simplification Using Quadric Error Metrics." *SIGGRAPH97* p.p. 209-216

[2] Hugues Hoppe. "View-Dependent Refinement of Progressive Meshes." *SIGGRAPH97* p.p. 189-198

[3] Hugeus Hoppe. "Progressive meshes." *SIGGRAPH96* p.p. 99-108

[4] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Wemer Stuetzle. "Multiresolution Analysis of Arbitrary Meshes.", *SIGGRAPH95* p.p. 173-182

[5] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. "Simplification Envelopes." *SIGGRAPH96* p.p. 119-126

[6] "GameGein II USER'S GUIDE", MultiGen-Paradigm, Inc.

[7] Will Schroeder, Ken Martin, and Bill Lorensen. "The Visualization Toolkit An Object-Oriented Approach to 3D Graphics." Prentice Hall.

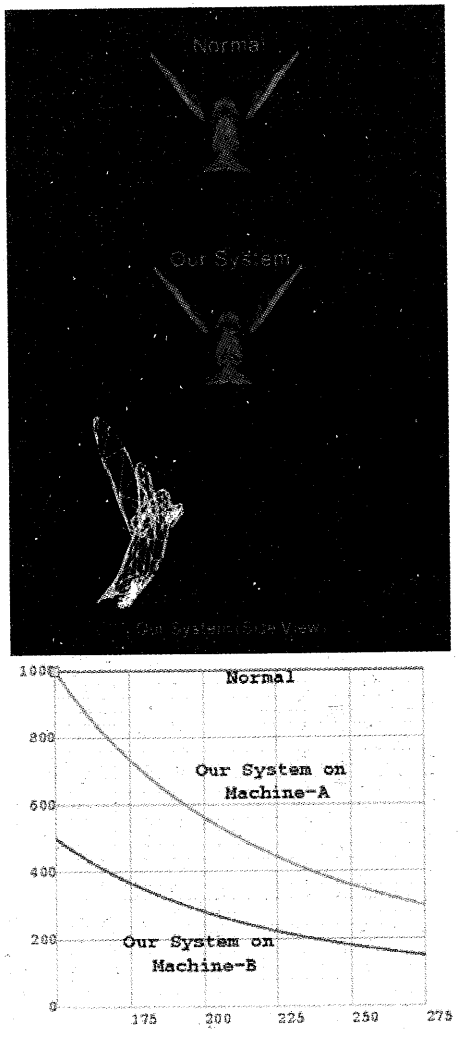


Figure 6. Z = 150 cm,
1000 polygons v.s. 1000 polygons

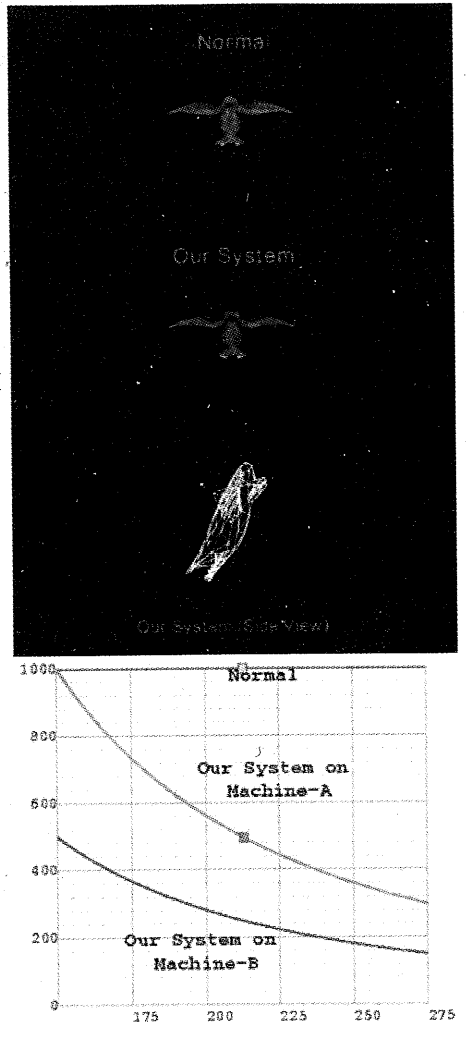


Figure 7. Z = 213 cm,
1000 polygons v.s. 496 polygons

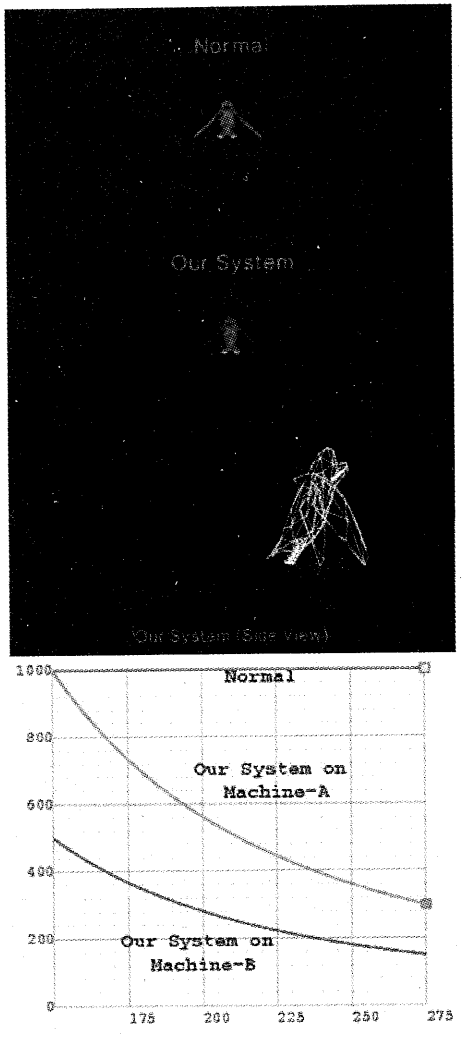


Figure 8. Z = 275 cm,
1000 polygons v.s. 298 polygons