[1]汎用ＷＳにおける仮想美術館の開発

4Q－4

[2]田辺香苗　堀内千尋　杉村一
[3]（株）東芝　東京システムセンター

1. はじめに

これまで、グラフィック市場では、専業システムベンダーによるハイエンドモデルのグラフィック専用マシンが主流であった。

今回の開発では既存の「仮想美術館システム」におけるＣＧエンジンを「グラフィック専用マシン」から「汎用ＷＳ」に移植した。

本稿では汎用ＷＳ（グラフィックアクセラレータを実装）を使用して開発した「仮想美術館」の描画性能についてデータ設計と処理の観点から述べる。

2. システム概要

移植対象のシステムは高速広帯域バックボーンネットワークを用いた美術情報サービス実験である。

本システムで提供する美術情報は

・高精細（ＨＤ）静止画情報。

・３次元ＣＧによる仮想美術館の動画情報。

美術館の鑑賞者は回線を通じてこれらの情報にアクセスすることができる。

今回はこのシステム構成中のＣＧエンジン部分の移植を行った。

3. 課題

ＣＧエンジンの移植においては、次の点が課題であった。

(1) 描画性能要求

汎用ＷＳ上で仮想美術館を実現する際の要件は「３ＤＣＧ空間を自在にウォークスルーできること」であった。そのためにはウォークスルーに最低限必要な描画速度(10frame/sec)を維持する必要がある。

(2) Ｈ／Ｗ性能の制限

		移植前	移植後
マシン		Onyx RE2（ＳＧＩ社）	Ultra（ＳＵＮ社）
ＣＰＵ		R440(200MHz)＊4	UltraSPARC(167MHz)＊1
ﾃｸｽﾁｬﾒﾓﾘ		１６ＭＢ	１５ＭＢ
ｸﾞﾗﾌｨｯｸ ﾗｲﾌﾞﾗﾘ	下層	IrisGL	OpenGL
	上層	Performer	OpenInventor

表１　ＣＧエンジン比較

移植前は、ｸﾞﾗﾌｨｯｸﾗｲﾌﾞﾗﾘのPerformerを使用している。これにより 4CPU をフルに活用し負荷の分散もできる。一方移植後のＣＧエンジンではこのPerformer はサポートされていなかった。よってＨ／Ｗ性能面においての制限を余儀なくされた。

4. ３次元空間データの分割

4.1.データ設計

移植後ＣＧエンジンでは Performer の代わりに OpenInventor を使用した。OpenInventor では３Ｄ空間のシーンデータを外部ファイル化することができる。描画性能を上げる為にはこのデータの設計が重要な工程となる。データに起因するパフォーマンス問題としては、

・頂点の比重が大きい。（ポリゴン数が多い）

・テクスチャの比重が大きい。（数、面積）

・画素の比重が大きい。（ウインドウサイズ）

といったものが典型的である。

データ設計段階ではこれらの問題を加味しながら可能な限り「シンプルな設計」を行った。

・ポリゴン数が極端に多い「木」を排除。

・丸みのあるオブジェクトの代わりに直線的なオブジェクトを配置。

・空間を屋外／屋内に分割し死角部分をつくる。

4.2.描画性能とデータ量（ポリゴン数）

汎用ＷＳの３Ｄ描画速度の素データを収集した。

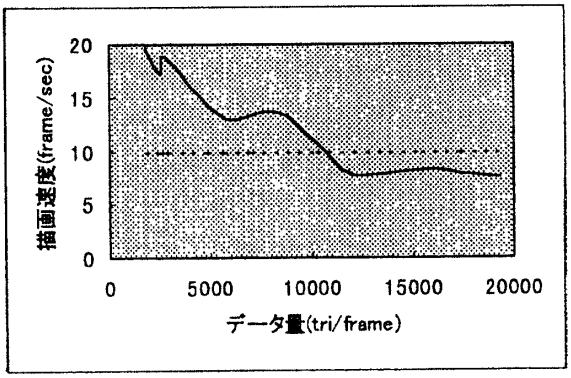

図１　描画速度とデータ量

違和感なくウォークスルーを行うためには、１秒間に最低でも 10frame の描画が必要となる。この要求性能と上のグラフから仮想美術館の３次元空間は１万ポリゴン以下で構築する必要があるという

[1] Development of Cyber Museum System on General WS

[2] Kanae TABE, Chihiro HORIUCHI, Hajime SUGIMURA

[3] TOSHIBA Corporation

ことが分かる。

4.3.データ切り替え

OpenInventorにおける、３次元シーン情報（シーングラフ）は下のようなツリー構造になっている。シーングラフを構成する基本要素をノードといい基本図形と材質ノード、座標変換ノードなどがある。

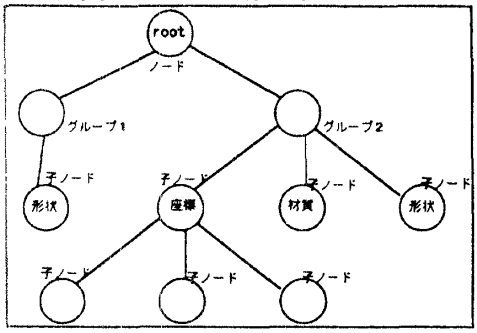

図２　OpenInventorのシーングラフ（ツリー）

ウォークスループログラムでは、性能面を考慮してシーングラフを動的に切り替えることにした。ウィンドウを複数起動し、データを切り替える。

空間全体を屋外、屋内に分割し、それぞれのシーングラフをroot（根本）から分けた。

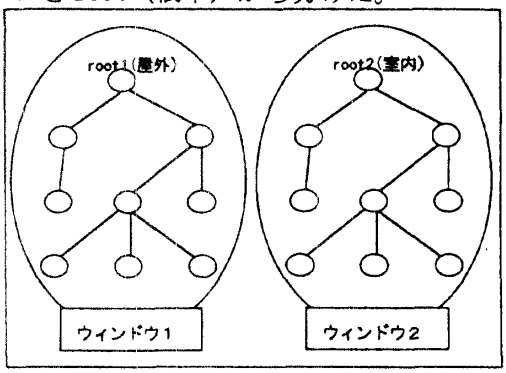

図３　シーングラフの分割

室内に居る時には屋外の景色は視野に入らない（死角となる）ような空間設計を行い屋外データと屋内データは起動時に別ウィンドウにロードする。ウォークスルースタート時は屋内ウィンドウは屋外ウィンドウの背面に隠れている。屋内エリアに入った瞬間に屋内ウィンドウがアクティブになり前面に移動する。これによってデータ量が削減され描画速度が向上する。

5. グループノードによるカリング

性能を改善するための２つめの手法として、シーングラフの構成要素（ノードのグループ）を動的に追加、削除（カリング）する方法がある。

この処理にはOpenInventorの「グループノード」の一つである「Separator」を使用する。

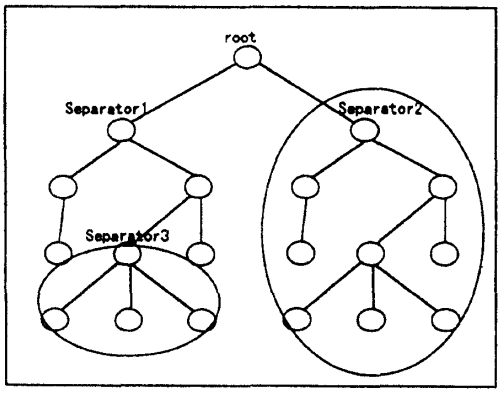

図４　Separator単位のカリング

Separatorは複数のノードをグループ化する。

Separatorノードのメソッドに含まれる

・addChild（子ノードの追加）

・removeChild（子ノードの削除）

を使用して随時死角データをカリングする。この手法はSeparatorノード単位に適用できるため、比較的詳細部分（例 Separator3 以下のノード）の操作が可能である。

6. 結果と考察

移植後の仮想美術館システムのデータ量と描画速度を次に示す。

データ量 (tri/frame)	最大時 1万1千	最小時 2千
描画速度	10frame/sec	14frame/sec

表２　移植後性能

ウォークスルーに要求される性能はクリアしている。３Dデータグラフのrootを分割する手法(4.3)ではウインドウを切り替える際に多少画面がちらつく。よって３D空間の細部の操作はSeparatorノードによるカリング(5)が効果的である。ただし、これらの処理を有効にするにはデータ設計段階での死角を使用したデータ分割が不可欠である。

7. おわりに

本稿ではグラフィックアクセラレータを実装した汎用ＷＳ上での仮想美術館の開発について述べた。Ｈ／Ｗの性能面における限界をデータ設計、ウォークスルー処理によってカバーし、最低限必要とされた描画性能を実現できた。

本結果を今後のＣＧ、ＶＲ分野及びマルチメディアシステムの開発に生かし、業界標準を考慮したより汎用的なシステムを提供したい。