(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3888477号
(P3888477)
(45)発行日　平成19年3月7日(2007.3.7)　(24)登録日　平成18年12月8日(2006.12.8)

(51)Int.Cl.　F I
G06T 15/00 (2006.01)　G06T 15/00 300

請求項の数 2 (全 87 頁)

(21)出願番号　特願平8-142702
(22)出願日　平成8年6月5日(1996.6.5)
(65)公開番号　特開平9-16785
(43)公開日　平成9年1月17日(1997.1.17)
審査請求日　平成15年5月12日(2003.5.12)
(31)優先権主張番号　486,447
(32)優先日　平成7年6月8日(1995.6.8)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　イーサン・ダブリュー・ガネット
アメリカ合衆国80525コロラド州フォート・コリンズ、シャドウ・マウンテン・ドライブ　2607

審査官　村松　貴士

最終頁に続く

(54)【発明の名称】テクスチャマッピング・コンピュータ・グラフィックス・システム

(57)【特許請求の範囲】

【請求項１】

テクスチャ・マッピング・コンピュータ・グラフィックス・システムであって、

プロセッサ（１９）と、少なくとも１つのテクスチャの一連のＭＩＰマップを含むテクスチャ・データを格納するシステム・メモリ（１７）と、を有するホスト・コンピュータ（１５）と、

任意の時点で前記システム・メモリに格納されているテクスチャ・データの少なくとも一部を、サイズの等しいブロックとして格納する、2個の別々のメモリ・バンクを含むローカル・メモリ（１５５）を有し、当該ホスト・コンピュータに接続されたグラフィック・ハードウェハ装置（１５０）であって、当該ローカル・メモリに格納されたテクスチャ・データを使用して、テクスチャ・マップ化された画像を描画するグラフィック・ハードウェハ装置とを備え、

前記プロセッサ上で実行されるソフトウェア・プロセス（１６０）は、画像の描画を行うのに、当該グラフィック・ハードウェハ装置が必要とする場合に、当該システム・メモリから当該ローカル・メモリへ、テクスチャ・データのブロックをダウンロードし、

当該ソフトウェア・プロセスは、少なくとも１つのテクスチャの少なくとも１つの一連のＭＩＰマップのうちのあるマップ（Ｔ３）が、所定のマップ・サイズ以下のサイズを有する場合に、当該マップの2個の別々の部分を、当該システム・メモリから、当該システム・メモリの2個の別々のブロック（８７２、８７３）へコピーし、その結果、当該マップの当該2個の別々の部分が、当該ローカル・メモリの別々のバンクへダウンロードされ

ようにし、
当該ソフトウェア・プロセスは、あるマップが、所定のマップ・サイズより大きいサイズを有する場合に、当該マップが位置する、当該システム・メモリの位置へのポインタ（８６０）を格納するだけである、テクスチャ・マッピング・コンピュータ・グラフィックス・システム。
【請求項２】
テクスチャ・マッピング・コンピュータ・グラフィックス・システムにおいて、テクスチャ・データを管理する方法であって、テクスチャ・マッピング・コンピュータ・グラフィックス・システムは、
プロセッサ（１９）と、少なくとも１つのテクスチャの一連のＭＩＰマップを含むテクスチャ・データを格納するシステム・メモリ（１７）と、を有するホスト・コンピュータ（１５）と、
任意の時点で前記システム・メモリに格納されているテクスチャ・データの少なくとも一部を、サイズの等しいブロックとして格納する、2個の別々のメモリ・バンクを含むローカル・メモリ（１５５）を有し、当該ホスト・コンピュータに接続されたグラフィック・ハードウェハ装置（１５０）であって、当該ローカル・メモリに格納されたテクスチャ・データを使用して、テクスチャ・マップ化された画像を描画するグラフィック・ハードウェハ装置とを含み、
画像の描画を行うのに、当該グラフィック・ハードウェハ装置が必要とする場合に、当該システム・メモリから当該ローカル・メモリへ、テクスチャ・データのブロックがダウンロードされ、当該方法は、
少なくとも１つのテクスチャの少なくとも１つの一連のＭＩＰマップのうちのあるマップ（Ｔ３）が、所定のマップ・サイズ以下のサイズを有する場合に、当該マップの2個の別々の部分を、当該システム・メモリから、当該システム・メモリの2個の別々のブロック（８７２、８７３）へコピーし、その結果、当該マップの当該2個の別々の部分が、当該ローカル・メモリの別々のバンクへダウンロードされようにし、
あるマップが、所定のマップ・サイズより大きいサイズを有する場合に、当該マップが位置する、当該システム・メモリの位置へのポインタ（８６０）を格納することを含む、テクスチャ・データを管理する方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
この発明は、一般にテクスチャマッピング・コンピュータ・グラフィックス・システムのテクスチャ・マップのソフトウェア・メモリ管理に関し、特に、テクスチャマッピング・データを格納するハードウェア装置ローカル・キャッシュ・メモリ・システムのデーモン(daemon)ベースの仮想メモリ管理に関する。
【０００２】
【従来の技術】
コンピュータ・グラフィックス・システムは、対象のグラフィック表現を２次元表示画面の上で表示するために一般に使用される。現在のコンピュータ・グラフィックス・システムは、非常に詳細な表現を提供することができて、いろいろな適用において使用される。
【０００３】
典型的なコンピュータ・グラフィックス・システムにおいて、表示画面（ディスプレイ・スクリーン）に表示するために選ばれる対象は、複数のグラフィックス・プリミティブに分解される。プリミティブは、グラフィックス・ピクチャの基本成分であって、点、線、ベクトルおよび三角形のような多角形を含むことがある。典型的に、ハードウェア/ソフトウェア方式が、スクリーンに表現されてつつある１つまたは複数の対象の投影図を表現するグラフィックス・プリミティブを２次元表示画面に描画するために実行される。
【０００４】
典型的に、描画されるべき３次元の対象を規定するプリミティブは、各プリミティブをプ

リミティブ・データで規定するホストコンピュータから提供される。たとえば、プリミティブが三角形であるとき、ホストコンピュータがその頂点のx,y,z座標、および各頂点のR,G,B色値でプリミティブを規定することがある。レンダリング・ハードウェアが、各プリミティブおよび各ピクセルのR,G,B値を表現するためにオンにされる表示画面ピクセルを計算するためにプリミティブ・データを補間する。

【０００５】

初期のグラフィックス・システムは、複雑な３次元の対象を表現またはモデル化するに十分に現実的な態様で、画像を表示することに失敗した。そのようなシステムによって表示される画像は、テクスチャがなく、こぶ、ひっかき傷、影およびモデル化される対象に存在する他の表面詳細のない非常にスムーズな表面を表示した。

【０００６】

この結果、画像を改善された微細な表面で表示する方法が発達した。テクスチャマッピングが、そのような方法であり、テクスチャと呼ばれるソースイメージを３次元の対象の表面にマップし、その後、テクスチャされた３次元の対象を２次元グラフィックスの表示画面にマップして、結果の画像を表示することを含む。一般にテクスチャマップされた表面の詳細属性は、色、鏡面反射、ベクトル動揺、反射、透明度、影、表面の不規則性および粒度を含む。

【０００７】

テクスチャマッピングは、テクスチャの１つまたは複数の点要素（テクセル）を、テクスチャがマップされつつある対象の表示された部分の各点要素（ピクセル：画素）に適用することを含む。テクスチャマッピング・ハードウェアは、従来より、テクスチャ・マップにおけるテクセルが対象を表現する表示画面上のピクセルに対応する態様を示す情報を備えている。テクスチャ・マップにおける各テクセルは、２次元テクスチャ・マップにおけるその位置を識別するSおよびT座標で規定される。各ピクセルについて、それにマップする対応する１つまたは複数のテクセルが、テクスチャ・マップからアクセスされ、ピクセルのために生成される最終的なR,G,B値に取り入れられて、表示画面上でテクスチャのある対象が表現される。

【０００８】

対象プリミティブにおける各ピクセルが、対象のあらゆる景観のためのテクスチャ・マップにおける一つのテクセルと１対１の対応でマップしないかもしれないことが理解されなければならない。例えば、対象が表示画面に表現されるビュー・ポートに近いほど、対象は大きく表現される。対象の表示画面上での表現が大きくなるにつれて、テクスチャの表示はより詳細になる。このように、対象が表示画面のかなり大きな部分を消費するとき、大きな数のピクセルが、対象を表示画面に表現するために使用され、そして、対象を表現する各ピクセルがテクスチャ・マップにおける一つのテクセルと１対１の対応でマップすることがあり、または一つのテクセルが、複数のピクセルにマップすることがある。しかし、対象が表示画面の比較的小さい部分をとるとき、より小さい数のピクセルが、対象を表現するために使用され、テクスチャが少し低い詳細度で表現されることになるので、各ピクセルが、複数のテクセルにマップすることになる。各ピクセルは、テクスチャが対象の小さい部分にマップされるとき複数のテクセルにマップすることになる。結果として生じるテクセル・データは、2以上のテクセルにマップする各ピクセルのために計算される。ピクセルが複数のテクセルにマップすることは一般的なので、１つのピクセルのための結果として生じるテクセル・データは、典型的にそのピクセルにマップするテクセルの平均を表現する。

【０００９】

テクスチャマッピング・ハードウェア・システムは、典型的には、描画されつつある対象に関連するテクスチャを表現するデータを格納する局部メモリを含む。上で述べたように、１つのピクセルが複数のテクセルにマップすることがある。テクスチャマッピング・ハードウェアが１つのピクセルにマップする多数のテクセルの平均値を生成するため局部メモリから読みとる必要があるならば、多数のメモリ読取り、および多くのテクセル値の平

均算出が必要となり、これには時間がかかり、システム性能の低下させる。

【００１０】

この問題を克服するために、各テクスチャのために一連のMIPマップを作り出すこと、およびテクスチャマッピング・ハードウェアの局部メモリにおいて描画されつつある対象と関連したテクスチャのMIPマップを格納することを含む方式が発達した。テクスチャのためのMIPマップは、一連のフィルタされたマップおよびテクスチャマップに直接対応するベース・マップを含み、2つのテクスチャ・マップ寸法のそれぞれにおいて、各連続したマップの大きさが2の係数だけ減少する。一組のMIPマップの例が図1に示される。MIP（multum in parvo:小さい場所にたくさんのもの）マップは、８掛ける８テクセルの大きさのベース・マップ100、およびそれぞれ、４掛ける４テクセル、２掛ける２テクセルおよび１テクセルの大きさの一連のマップ102、104、108を含む。

【００１１】

４掛ける４のマップ102は、マップ102の各テクセルがベース・マップ100における平均４つのテクセルに対応するように、ベース・マップ100をボックス・フィルタ（10分の１にする）して生成される。例えば、マップ102のテクセル110がマップ100のテクセル112-115の平均に等しく、そして、マップ102のテクセル118および120が、マップ100のテクセル121-124および125-128の平均にそれぞれ等しい。同様に、２掛ける２のマップ104は、マップ102をマップ104のテクセル130がマップ102のテクセル 110および118-120の平均に等しくなるように、ボックス・フィルタすることによって生成されるマップ108の一つのテクセルは、マップ104の4つのテクセルを平均することによって生成される。

【００１２】

従来のグラフィックス・システムは、概して、ホストコンピュータの主記憶装置からテクスチャマッピング・ハードウェアの局部メモリへ、表示画面に描画されるべきプリミティブとともに使用される任意のテクスチャのための完全なシリーズのMIPマップをダウンロードする。このようにテクスチャマッピング・ハードウェアは、MIPマップのシリーズのいずれからも、テクスチャデータをアクセスすることができる。任意の特定のピクセルのためのテクセル・データを提供するためにどのマップをアクセスするかの決定は、ピクセルがマップするテクセルの数に基づく。たとえば、ピクセル・マップがテクスチャの一つのテクセルと１対１の対応であるならば、ベース・マップ100がアクセスされる。しかし、ピクセルが４つ、１６、または２４のテクセルにマップするならば、マップ102、104および108が、それぞれアクセスされる。それらのマップがテクスチャ・マップにおける4つ、16、24のテクセルの平均を表すテクセルデータをそれぞれ格納しているからである。

【００１３】

さらに、あるテクスチャマッピング・システムは、いろいろな操作が同時に異なる対象プリミティブで実行されるようパイプラインされる。しかし、テクスチャのための一連のMIPが大きいことがある。ほとんどのシステムが、一度にそのような大きいシリーズのMIPマップを一つだけ格納することができる局部メモリを使う。このように、使用されるテクスチャの切換えがプリミティブの描画中にあるとき、システムはMIPマップの新しいシリーズをダウンロードしなければならない。典型的に、新しいテクスチャ・データをテクスチャマッピング・ハードウェアの局部メモリにロードするために使用されるデータ経路は、システムのプリミティブ描画パイプラインを通過することを含む。したがって、新しいテクスチャがマップされるべきとき、MIPマップの新しいシリーズがダウンロードされる前に、プリミティブ描画パイプラインが、からにされなければならない。MIPマップのシリーズがダウンロードされると、パイプラインは再び満たされなければならない。新しいテクスチャが必要な都度プリミティブ描画パイプラインをフラッシュする必要性は、システムのバンド幅（帯域幅）を減らす。

【００１４】

図2は、ホストコンピュータがテクスチャ・マップされた画像を描画するために通信する、システム・ホストコンピュータのプロセッサーの上およびハードウェア装置上で走る、従来のコンピュータ・グラフィックス・システムのソフトウェアの層を示す部分的にハー

ドウェアで部分的に階層的なソフトウェアのブロック図である。ホストコンピュータのシステム・メモリは、図2に示されるように、システム・プロセッサの上で走ることができる複数ユーザ対話型プロセスPR1、PR2．．．．PRNを格納することができる。各プロセスは、データベース、CAD/CAMアプリケーション、建築上の設計アプリケーション、土木工学アプリケーション、ワードプロセッシング・パッケージなどのような高水準コンピュータ図形処理のソフトウェア・プログラムまたはアプリケーションである。

【００１５】

特定のプロセス内で、ユーザは複数のコンテクスト(context)をつくることができる。プロセス内のつくられた各コンテクストは、同じ画像の異なる景観を含むことができる。たとえば、建築上の工学設計応用プロセスの、ユーザは、ビルディングのような同じ構造の異なる景観をつくることがありえる。各コンテクストは、テクスチャマッピングに画像を描画することを要求することがありえる。この例では、テクスチャ・マップは、床、壁、天井および建物のその他の表面属性を描画することを要求されることがある。このように、ユーザがそのコンテクストの画像を描画する１つまたは複数のテクスチャ・マップを提供する。同じプロセス内でつくられたコンテクストが典型的には同じ画像の異なる景観を含むので、プロセス内のコンテクストは、一般に１つまたは複数の同じテクスチャを必要とする。

【００１６】

コンテクストがユーザによってつくられるとき、ユーザがその画像を描くために高水準プロセス・プログラミング言語によって認識可能なコマンドを出す。画像がテクスチャ・マップを必要とするならば、そのことを示すコマンドが入れられ、テクスチャ・マップが、外部入力装置、フロッピーディスクなどからユーザによって提供される。次いで、プロセス、または高水準プログラミング言語がこのコマンドをより低いレベルのソフトウェアへ・コマンドに翻訳し、そして、ユーザによって入れられるテクスチャをコピーする。図２に示すような先行技術システムにおいては、一つのプロセス内でつくられた各コンテクストが、そのテクスチャのそれ自身のコピーを必要とする。各コンテクストは、テクスチャのそのコピーをホストコンピュータ上のシステム・メモリの専用領域内に格納する。このように、プロセス内の複数のコンテキストが同じテクスチャの使用を要求するとき、1つが各コンテクストに対応する、そのテクスチャの複数のコピーがシステム・メモリ内に格納される。

【００１７】

特定のテクスチャを使用して画像を描くためにユーザ入力され、プロセスによって一度翻訳されたコマンドは、プロセスから基礎をなすグラフィックス・アプリケーション・プログラマ・インターフェイス（API）に伝えられる。図2に示されるように、ユニークなグラフィックスAPIは、システムの各プロセスと通信する。各グラフィックスAPIは、ホストコンピュータのプロセッサ上で、対応するプロセスから受け取った各高水準グラフィックス・コマンドを、低レベルのソフトウェア・コードのコマンドに翻訳する働きをする下位レベルのソフトウェア言語である。グラフィックスAPIは、また画像を描画するのに必要な各テクスチャのコピーをシステム・メモリのそれ自身の専用の位置に格納する。グラフィックスAPIが、描画すべき画像を四辺形または多角形のようなグラフィックス成分に分ける。そのような情報は、次いで対応するハードウェア・ドライバに提供される。

【００１８】

ユニークなハードウェア・ドライバが各グラフィックスAPIと通信する。各ハードウェア・ドライバは、別の基礎をなす低レベルのグラフィックス・ソフトウェア言語であるAPIから受け取った各低レベルのグラフィックス・コマンドを、ホストコンピュータが接続されるグラフィックス・ハードウェア装置によって認識できるハードウェア・コマンドに翻訳する。次いで、これらのコマンドが、ハードウェア・ドライバからハードウェア装置に伝えられ、これに応答して、ハードウェア装置がテクスチャマップされた画像をハードウェア装置に接続した表示画面に描画する。図2に示されるように、ハードウェア・ドライバHWD1，HWD2...HWDNのユニークな例が、各グラフィックスAPIである API1，AP12...APIN

と通信する。

【００１９】

図2に示した例では、ハードウェア・ドライバのすべては、一つのグラフィックス・ハードウェア装置150に接続されている。しかし、当業者によって理解されるように、ホストコンピュータは、異なるハードウェア・ドライバが各ハードウェア装置と通信する複数の異なるグラフィックス・ハードウェア装置に接続されてもよい。

【００２０】

グラフィックスAPIが、提供された特定のテクスチャを使用してテクスチャマップした多角形を描画する低レベルのソフトウェア・コマンドを出す時、グラフィックス・ハードウェア・ドライバは、その低レベルのコマンドをハードウェア装置150が認識できるハードウェア・コマンドに翻訳する。次いで、グラフィックス・ハードウェア・ドライバが、ハードウェア装置内の適当なレジスタおよび入力バッファに書き込むことによって、ハードウェア装置150に多角形を描画しろというコマンドを出す。さらに、グラフィックス・ハードウェア・ドライバが、その特定の多角形のためのテクスチャ・データをハードウェア装置150の局部メモリ155に提供し、そこでテクスチャ・データは、その多角形のレンダリングの間、一時格納される。ここで記述したように、先行技術のコンピュータ・グラフィックス・システムは、テクスチャが、画像または画像の成分を描画することを要求される都度、一つのテクスチャのためのMIPマップの全シリーズを、グラフィックス・ハードウェア・ドライバからハードウェア装置の局部（ローカル）メモリにダウンロードする。

【００２１】

図2に示されたもののような、従来のシステムでは、一つのテクスチャの複数のコピーが、ホストコンピュータのシステム・ソフトウェア・メモリの異なる位置に格納される。たとえば、プロセスPRlのコンテクスト1Aとコンテクスト1Bが同じテクスチャに特定の画像の異なる景観を描画することを要求するならば、それが最初に確定されプロセスに導入される時、ユーザがそのテクスチャの第１のコピーを提供する。そのプロセスPR1内の各コンテクスト1Aおよび1Bもテクスチャのコピーを格納し、プロセスPR1によって格納される同じテクスチャのソフトウェア・コピーの小計を3にする。グラフィックスAPI1が、そのテクスチャの4番目のコピーを作り、そして、それをシステム・ソフトウェア・メモリのその割り当てられた位置に格納する。グラフィックス・ハードウェア・ドライバHWD1が、そのテクスチャの5番目のコピーを作り、そして、それをシステム・ソフトウェア・メモリのその割り当てられた位置に格納する。このように、その同じテクスチャを使用して同じ画像の2つの景観を描画するのに、同じテクスチャの5つのユニークなコピーが、システム・ソフトウェア・メモリの異なる位置の格納される。さらに先に述べたように、テクスチャの6番目のコピーが、ハードウェア装置に提供され、それの局部メモリに格納される。

【００２２】

一連のテクスチャMIPマップが、格納のために多量のシステム・ソフトウェア・メモリを必要とすることになる。たとえば、1024 x 1024テクセルのテクスチャ・ベース・マップを持つテクスチャのための一連のMIPマップは、MIPマップされたテクスチャの一つのコピーを格納するために、5メガバイト以上のシステム・ソフトウェア・メモリ必要とする。このように、MIPマップされたテクスチャの多重に格納されたコピーは、システム・ソフトウェア・メモリのかなりの量を使用する。

【００２３】

システム・ソフトウェア・メモリは、数ギガバイトのソフトウェア・データを格納することができるが、テクスチャ・ハードウェア装置の典型的な専用の局部ハードウェア・テクスチャ・メモリは、ずっと少ない量のデータしか格納することができない。そのような局部テクスチャ・ハードウェア・メモリは、4メガバイトから16メガバイトまでわたることができる。したがって多くのテクスチャマッピング適用において、テクスチャによって消費されるシステム・メモリの量は、局部ハードウェアのテクスチャ・メモリの量をはるかに越える。システム・メモリは、典型的には、プロセスの複数のコンテクストで使用する

ための複数のMIPマップされたテクスチャを格納するが、局部ハードウェア装置メモリは、限られたサイズの一つのテクスチャだけを任意の一つの時に格納することできるだけである。したがって、ハードウェア装置の局部テクスチャ・メモリを適切に管理することが、システムの最大性能を達成するために重大である。

【００２４】

【発明が解決しようとする課題】

テクスチャの多重のコピーの格納によるシステム・ソフトウェア・メモリの大きな消費に加えて、従来の局部ハードウェア・テクスチャ・メモリ管理方式は、システムの帯域幅でも一般に苦しむ。従来の局部ハードウェア・メモリ管理方式は、局部ハードウェア・メモリ内で繰り返してテクスチャMIPマップの全シリーズを置き換える。多角形を描画するためにテクスチャが必要とされる都度、テクスチャのための一連のMIPマップが、局部メモリで置き換えられる。ような方式は、そのテクスチャの使用のヒストリー（履歴）を考慮することにも、そしてそのテクスチャの将来の使用を予測することにも失敗する。繰り返してシステム・メモリから局部ハードウェア・テクスチャ・メモリに、同じテクスチャの全シリーズをダウンロードすることによって、システムの帯域幅は、消極的な影響を受ける。

【００２５】

従来のテクスチャ・ハードウェア・メモリ置き換えアルゴリズムが全テクスチャMIPマップ・シリーズをダウンロードするので、各テクスチャMIPマップ・シリーズは、局部ハードウェア・テクスチャ・メモリの物理的な記憶容量限度を越えることができない。したがって、特定のハードウェア装置に接続するグラフィックス・プロセスのユーザは、その容量より大きいテクスチャ容量を使わないように局部テクスチャ・メモリの容量を知っている必要がある。多くの適用（プロセス）がそれぞれユニークな局部テクスチャ・メモリの制約を持ついくつかの異なる基礎をなすハードウェア装置で動作することができるが、そのような制約に関する知識を持ち、装置に依存する態様で画像をつくる負担は、ユーザのものとなる。

【００２６】

【課題を解決するための手段】

この発明の一つの形態において、テクスチャマッピング・コンピュータ・グラフィックス・システムが提供される。システムは、テクスチャ・データを格納するシステム・メモリを持つホストコンピュータを含む。ホストコンピュータに結合するグラフィックス・ハードウェア装置が、テクスチャ・マップされた画像を描画する。ハードウェア装置は、システム・メモリに格納されたテクスチャ・データの少なくとも一部分を任意の一時点で格納する局部メモリを含む。ソフトウェア・デーモン(daemon)は、ホストコンピュータのプロセッサ上で走る。ソフトウェア・デーモンが、画像を描画するためにハードウェア装置によって必要とされるときテクスチャ・データをシステム・メモリから局部メモリに転送するのを管理する。

【００２７】

発明の別の形態において、テクスチャマッピング・コンピュータ・グラフィックス・システムにおいてテクスチャ・データを管理する方法が提供される。システムが、テクスチャ・データを格納するメイン・メモリを有するホストコンピュータを含む。ハードウェア装置は、メイン・メモリに格納したテクスチャ・データの少なくとも一部を任意の時点でブロックで格納する、ホストコンピュータに結合した局部メモリを持つ。ハードウェア装置は、局部メモリに格納されたテクスチャ・データを使用してテクスチャ・マップされた画像を描画する。この方法は、次のステップを含む：局部メモリの格納したテクスチャ・データブロックのハードウェア装置による使用を監視するステップ；メイン・メモリに格納したテクスチャの優先度を追跡するステップ；そして、局部メモリにおける最下位優先度および使用最近度の最も低いテクスチャ・データブロックを画像を描画するためにハードウェア装置によって必要とされるテクスチャ・データブロックで置き換えるステップ。

【００２８】

この発明の別の形態は、コンピュータ・グラフィックス・システムのプロセッサ上で実行されるソフトウェアシステムに関連する。プロセッサは、少なくとも一つのコンピュータ・グラフィックス・プロセス、各プロセスのための基礎をなすグラフィックスAPI、そして、各プロセスのための基礎ハードウェア・ドライバを走らせる。コンピュータ・グラフィックス・システムは、テクスチャマップした画像を描画する、プロセッサに接続されたハードウェア装置を含む。ソフトウェアシステムが、プロセッサからハードウェア装置へのテクスチャ・データのダウンロードを管理するデーモンを含む。ソフトウェアシステムは、デーモン、グラフィックスAPIおよびハードウェア・ドライバによってアクセス可能なテクスチャ・データを格納する共用メモリ領域をも含む。

【００２９】

発明の別の形態において、テクスチャマッピング・コンピュータ・グラフィックス・システムのテクスチャ・データを管理するソフトウェア・デーモンが提供される。システムは、その上でソフトウェア・デーモンおよび少なくとも一つのグラフィックス・プロセスが走るプロセッサを含む。ユーザは、各プロセス内に複数のコンテキストをつくることができる。システムは、この少なくとも一つのグラフィックス・プロセスでつくられたテクスチャマップされた画像を描画する、プロセッサに結合した少なくとも一つのグラフィックス・ハードウェア装置を含む。デーモンは、テクスチャの一つの格納されたコピーが同じプロセス内の複数のコンテキストの間で共有されるように、各プロセスのためのテクスチャ・データを別に管理する装置独立の部分を含む。デーモンは、グラフィックス・ハードウェア装置にテクスチャ・データを書くために装置に依存する部分を含む。

【００３０】

【発明の実施の形態】

Ｉ．システム概要

この発明は、コンピュータ・グラフィックス・システム・ホスト・コンピュータのプロセッサ上で走り、そしてホストコンピュータに接続したテクスチャマッピング・グラフィックスハードウェア装置の局部メモリ内で、テクスチャ・データの記憶を管理するソフトウェア・デーモンに関する。デーモンは、メモリが仮想に見えるような方法で局部メモリを管理する。ハードウェア装置の局部メモリにテクスチャMIPマップの全シリーズをダウンロードする先行技術システムと対照的に、この発明のソフトウェア・マネージャは、画像または画像の一部分を描画するために任意のある時に必要なテクスチャ MIPマップの部分だけをダウンロードする。マネージャは、画像またはその一部を描画するために新しいテクスチャ・データがハードウェア装置によって要求されるとき、局部的なハードウェア・メモリのどの部分を置き換えるか決定するのに、特定のテクスチャ・マップ部分の最近の過去の使用頻度、およびそのようなマップ部分の将来の使用予測（テクスチャの相対的優先度に基づく）を考慮する。

【００３１】

発明の一つの実施例において、マネージャは、マネージャ、ハードウェア・ドライバ、および対応するグラフィックスAPIの間でテクスチャがシステム・メモリ内に共有記憶されるようにする。マネージャは、一つのプロセスの複数のコンテキストの間でテクスチャが共有記憶されるようにする。この発明の管理方式は、バンド幅の大きな節約、および先行技術に対比してシステム・ソフトウェア・メモリ空間の節約を提供する。システムは、さらに、基礎をなすハードウェア装置の局部メモリの記憶容量限度の知識なしで、高水準グラフィックス・プロセスのユーザが装置から独立の態様で画像をつくることを可能にする。

【００３２】

発明の第１の実施例が、図3のブロック図に示される。そこでは同じ参照文字が図2のものと同じ要素を示すために使用されている。図2の先行技術の例のように、システムはシステム・プロセッサ上で走る複数のユーザ対話型プロセスPR1、PR2．．．PRNを含む。それぞれのプロセス内で、ユーザは、同じテクスチャ・マップを必要とする複数のコンテキストを作ることができる。システムは、それぞれのプロセスのために、基礎をなすグラフィ

ックスAPIおよびグラフィックス・ハードウェア・ドライバを含む。

【００３３】

この発明は、ユーザの知らないホストコンピュータのプロセッサ上で走る、独立の、スタンドアローンのソフトウェア・プロセスであるテクスチャ割込み管理デーモン（TIM）160に関連する。この発明のTIM 160は、別個のソケットを通してグラフィックス・ハードウェア・ドライバのそれぞれと通信する。当業者に理解されるように、ソケットは、それを通して2つのソフトウェア・プログラムが通信することができるソフトウェア通信パスである。TIMは、バス162を通してハードウェア装置150とも通信する。

【００３４】

この発明のTIMは、コンピュータ・グラフィックス・ワークステーションとして知られる、コンピュータ・グラフィックス・システム・ホストコンピュータの従来のプロセッサ上で走ることができる。そのようなワークステーションは、例えばヒューレット・パッカードTM 9000のシリーズJ200を含む。

【００３５】

TIM 160が、ハードウェア装置150の局部メモリ155内でのテクスチャ・データの格納を管理する。より詳細を後述するように、TIMは、ソフトウェア・テクスチャ・メモリ管理およびグラフィックス・ハードウェア・ドライバとのソケットを通しての通信を扱う、装置独立の部分と、バス162を通してハードウェア装置への書き込みおよびこれからの読み取りを行う装置に依存する部分と、を含む。

【００３６】

この発明の一つの実施例において、ハードウェア装置の局部メモリが、キャッシュ・メモリとして準備され、テクスチャの一部分が、任意のある時にハードウェア装置の局部メモリに格納される。TIMの装置独立の部分が、キャッシュに格納されたテクスチャ・データの一部分の使用を追跡し、そして、テクスチャの優先度を監視してそれらの部分の将来の使用を予測する。キャッシュに現在格納されていないテクスチャ・データが、画像を描画するためにハードウェア装置によって必要とされるとき、ハードウェアのキャッシュ・ミスが、起こる。キャッシュ・ミスが生じるとき、TIMは、キャッシュ内のテクスチャ・データのどのブロックまたは複数のブロックの使用最近度が最も低いか、およびどのテクスチャが、最下位優先度を持つかを考慮することによって、キャッシュ内のテクスチャ・データのどのブロックを置き換えるか決定する。

【００３７】

TIMは、ホストコンピュータに画像を描画するために接続した特定のハードウェア装置の局部メモリによって認識できるフォーマットにテクスチャ・データを割り当てる。発明の一つの実施例において、１つのテクスチャの一連のMIPマップのそれぞれのマップが、４つに分割され、そしてTIMによってハードウェア装置の局部的キャッシュ・メモリにダウンロードされるデータのブロック（複数）の間で割り当てられる。キャッシュ割込みが生じる時、TIMの装置依存部分が、実際にデータのダウンロードを実行する。

【００３８】

発明の一つの実施例において、TIMは、システム・メモリ内に共有記憶ロケーションを持ち、そこでは、大きいテクスチャまたはテクスチャ部分が格納され、そして特定のプロセスのために、グラフィックスAPI、グラフィックス・ハードウェア・ドライバ、およびTIMの間で共有される。より詳細に下で説明されるように、グラフィックスAPI、グラフィックス・ハードウェア・ドライバ（特定のプロセスのために）およびTIM 160のそれぞれが、格納されたテクスチャ・コピーをアクセスするためシステム共用メモリー領域内のロケーションに対するポインタを含む。さらに、TIMは、一つのプロセス内で作られたコンテキストのすべてがそれらのコンテキストのために必要とされるテクスチャの同じコピーを共有するようにするメモリ管理方式を提供する。ある適用について、この発明のTIMは、システム・ソフトウェア・メモリ空間の相当な節減を提供することができる。

【００３９】

プロセスPR1のコンテキスト1Aおよびコンテキスト1Bが特定の画像の異なる景観を描画す

るために同じテクスチャを必要とする、図2を参照して上に記述した例に関して、テクスチャが大きいならば、この発明のTIMは、相当なシステム・メモリ空間の節減を提供する。説明した先行技術システムと対照的に、コンテキスト1Aおよびコンテキスト1Bの間で共有されるテクスチャの一つのコピーを格納することができる。さらに、テクスチャが共用メモリ・ロケーションに格納するのを正当化するに十分大きいならば、テクスチャの一つの他のコピーだけが、システム共用メモリ・ロケーションに格納され、そして、TIM、グラフィックスAPI1、およびグラフィックス・ハードウェア・ドライバHWD1の間で共有される。このようにこの例では、同じテクスチャを使用する同じ画像の2つの景観を描画するために同じテクスチャの3つのユニークなコピーだけが、システム・ソフトウェア・メモリの異なるロケーションに格納される。一つのコピーが、ユーザ・コピーであり、一つのコピーが、プロセスPR1の2つのコンテキストの間で共有され、そして、一つのコピーがグラフィックスAPI、グラフィックス・ハードウェア・ドライバ、およびTIMの間で共有される。図2の先行技術システムにおいては、上述したように、その同じテクスチャの5つのコピーがシステム・ソフトウェア・メモリに格納されなければならなかった。したがって、この発明のTIMでは、テクスチャの2つ少ないコピーが、システム・ソフトウェア・メモリに格納されることを要求され、大きいテクスチャについては、大きなメモリ空間の節減になる。

【００４０】

ユーザが画像をつくろうとするとき、ユーザは、その画像を描くために高水準プロセス・プログラミング言語によって認識できるコマンドを出し、その画像をテクスチャ・マップするに必要なテクスチャを入力する。プロセスが、次いでそのコマンドをより低レベルのソフトウェア・コマンドに翻訳し、そしてそのテクスチャをコピーする。翻訳されたコマンドが、そのプロセスからそのプロセスに関連する基礎をなすグラフィックスAPIに伝えられる。グラフィックスAPIが、次いでプロセスから受け取られたグラフィックス・コマンドをより低レベルのソフトウェア・コード・コマンドに翻訳し、そして画像を四辺形または多角形のようなグラフィックス成分に分ける。その情報が次いでハードウェア・ドライバに提供される。

【００４１】

その情報がハードウェア・ドライバに知らせられるとき、それとTIMの間の対応するソケットが開かれる。より詳細に下で説明されるように、ハードウェア・ドライバは、ユーザが画像を特定のテクスチャで描画したいことを次いでTIMに知らせる。ハードウェア・ドライバが、TIMがすでにテクスチャの格納されたコピーを持っているかどうかを判断するためにTIMにソケットを通して通信する。持っていれば、そのテクスチャを使用してその多角形を描画するために、ハードウェア・ドライバがコマンドをそれが接続されているコンピュータ・グラフィックス・ハードウェア装置に発する。持っていなければ、TIMは、下に詳述するように、そのテクスチャをそれ自分の割り当てられたシステム・ソフトウェア記憶ロケーションか、またはその共有メモリ・ロケーションまたはそれらそれぞれの一部分にコピーする。

【００４２】

ハードウェア・ドライバが新しいテクスチャがユーザによって要求されたことをTIMに知らせるとき、テクスチャの優先度およびテクスチャのサイズも、ソケットを通してTIMに知らされる。利用可能であるならば、TIMはテクスチャに対するテクスチャ識別子を提供する。その時点で、テクスチャのサイズおよびそのテクスチャのMIPマップ・レベルの番号に依存して、そのテクスチャが、共用メモリ・ロケーションに格納されるかどうか、またはそのテクスチャがTIM自身の割当システム・ソフトウェア記憶ロケーションに格納されるかどうかの決定がなされる。この方式は、より詳細に下で説明される。

【００４３】

新しいテクスチャのコピーを格納しているTIMの一部としてTIMの装置独立の部分が、テクスチャ・データを、それが接続されたハードウェア装置によって認識可能なフォーマットに割り当てる。発明の一つの実施例において、テクスチャ・データは、ハードウェア装置

によって要求されるように、ブロックに割り当てられて、ハードウェア装置の局部的キャッシュ・メモリに一度に一つのブロックずつダウンロードされる。

【００４４】

ハードウェア・コマンドは、テクスチャ・マップを使用している画像を描画するために、グラフィックス・ハードウェア・ドライバによってハードウェア装置に出される。例えば、ハードウェア・コマンドは、図6の中のバス16に沿って伝えられる。対応するテクスチャ・マップが、対応するハードウェア装置の局部メモリにまだ格納されないならば、割込みが生じ、そしてTIMがその割込みを処理する。TIMが、画像を描画するために必要なテクスチャの部分を決定するためにハードウェア装置と通信し、必要とされたテクスチャ部分で上書きするためキャッシュ・メモリ内の適当なブロックを選び、それから必要とされたブロックを局部的キャッシュ・メモリの選ばれたロケーションに書く。割込みが生じるとき、ハードウェア装置の局部メモリのどのブロックのテクスチャ・データを置き換えるかの決定は、ハードウェア装置内でのそのブロックの使用の履歴およびブロックに格納されたテクスチャの優先度の両方に基づく。テクスチャの優先順位は、外部ユーザによって規定された優先度および内部デフォルト優先度に基づく。内部的には、新しくつくられた画像を描画するために必要とされる１つまたは複数のテクスチャに最も高い優先度が与えられ、詳細には下に記述されるように、最も最近使用されたテクスチャに次に高い優先度が与えられる。このように、ユーザがテクスチャを必要としている新しい画像をつくる時、そのテクスチャにその時、最も高い優先度が与えられる。

【００４５】

図4は、システムがA、BおよびCとも分類される3つの別々のグラフィックス・ハードウェア装置164、166および168と通信する一つのホストコンピュータを含む、この発明の代替実施例のブロック図である。これらのハードウェア装置のそれぞれは、異なる物理的パラメーターと要求事項を含むことがありえる。3つの別々のTIM 170、172と174（また、A,BとCとラベルを付けられる)は、グラフィックス・ハードウェア装置164、166と168と通信する。各TIMは、それが接続されたグラフィックス・ハードウェア装置の局部メモリ内でテクスチャ・データの記憶を管理することに対して責任がある。したがって、各TIMが対応するハードウェア装置の、局部メモリの記憶容量、テクスチャ・フォーマット要求事項、バス幅、バッファ・サイズ、レジスタ・サイズ、その他のような物理的なパラメーターおよび要求事項の知識を持つ。

【００４６】

各TIMが、各プロセスでつくられたテクスチャが格納され、そしてTIM、グラフィックスAPI、およびそのプロセスのハードウェア・ドライバの間で共有される共用メモリ・ソフトウェア・ロケーションを含む。より詳細に下で説明するように、TIM、グラフィックスAPIおよびハードウェア・ドライバのそれぞれは、テクスチャをアクセスするため共用メモリ・ロケーションに対するポインタを格納する。共用メモリ方式は、相当なシステム・メモリ空間を節約することができる。各TIMは、テクスチャまたはテクスチャの一部分のコピーを格納するそれ自身の専用システム記憶域を含む。これらの記憶域は、図4に示される。

【００４７】

図4のシステムにおいて、TIM A 170が、画像がハードウェア装置A 164で描画されるプロセスPR1とPR2が必要とするテクスチャを管理する。同様に、TIM B 172が、画像がハードウェア装置B 166によって描画されるプロセスPR3とPR4が必要とするテクスチャを管理する。最後に、TIM C 174が、画像がハードウェア装置168によって描画されるプロセスPR5のためにテクスチャを管理する。

【００４８】

一つのTIM 176が、複数の基礎をなすハードウェア装置164、166と168の局部メモリにおけるテクスチャ・データ記憶を管理する、この発明のシステムの別の代替実施例が、図5に示される。TIM 176は、装置独立の部分178と装置に依存する部分180を含む。装置独立のTIM 176の部分178は、テクスチャ・メモリを管理し、各ハードウェア・ドライバとのソケ

ットによる通信を制御し、そしてハードウェア装置の割り込みが生じる時、テクスチャ・データのどのブロックがハードウェア装置によって必要とされるか、およびハードウェア装置の局部メモリ内のテクスチャ・データのどのブロックが上書きされなければならないかを決定するルーチンを実行する（これが転じて、装置に依存するルーチンを呼ぶ）。TIM 176の装置に依存する部分182、184および188は、基礎をなすハードウェア装置164、166および168のそれぞれから読み出しこれに書き込む。特定のハードウェア装置内に割込みが生じるとき、TIMの対応する装置依存の部分が、ハードウェア装置からデータを読みとり、TIMの装置独立の部分と通信する。次いでルーチンが、ハードウェア装置によって必要とされるテクスチャ・データを決定するために実行される。次いで、ハードウェア装置の局部メモリ内のテクスチャ・データのどのブロックが上書きされねばならないか決定するために別のルーチンが実行される。次いで、TIMの対応する装置に依存する部分内のルーチンが、新しいテクスチャ・データをそれが接続されたハードウェア装置の局部メモリ内の適当なロケーションに書き込む。

【００４９】

この発明のデーモン（TIM）が動作することができる特定のテクスチャマッピング・コンピュータ・グラフィックス・システム・ハードウェア装置の例を、図6ないし27を参照して詳細に説明する。ハードウェア装置の説明に続いて、この発明のソフトウェア・デーモンの動作の詳細な説明を行う。下に説明する例のハードウェア装置に関する装置固有のパラメーターを含む発明のソフトウェアの動作の例が、提供される。そのような装置固有の説明は、例としてものにすぎず、この発明のソフトウェアは、画像を描画するために必要なテクスチャ・マップの部分を一時的に格納する局部的なハードウェア・メモリを有する多数の異なるハードウェア装置で動作することができる。。ハードウェア装置がその時局部的なハードウェア装置メモリに格納されていないテクスチャまたはその一部分を必要とするとき、割込みが生じ、必要とされるテクスチャまたはテクスチャのいち部分をダウンロードするために、発明のソフトウェア・デーモンによって扱われる。発明のソフトウェアは、ホストコンピュータのシステム・プロセッサ上で走り、ホストコンピュータのシステム・メモリに格納される。そのホストコンピュータは、グラフィックス・ハードウェア装置に接続されている。

【００５０】

II. ハードウェア装置の例

A.ハードウェア装置システム概要

図6は、発明のソフトウェアを使用することができるグラフィックス・システム・ハードウェア装置の一つの実施例のブロック図である。装置は、テクスチャ・データをローカルに格納するためのキャッシュ・メモリを持つテクスチャマッピング・ハードウェアを含む。ボードおよびチップの数、それらが分割される態様、バス幅、およびデータ転送速度に関して示された実施例は単に例としてのものであることが、理解されなければならない。多数の他の実施例を使うことができる。図に示すように、システムは、フロントエンド（前縁）ボード10、テクスチャマッピング・ボード12およびフレーム・バッファ・ボード14を有する。フロントエンド・ボードは、52ビットバス16を通してホストコンピュータ15と通信する。フロントエンド・ボードは、描画すべきプリミティブをホストコンピュータからバス16を通して受け取る。プリミティブは、プリミティブが三角形である時の頂点についてのような、プリミティブの部分についてのx,y,zベクトル座標データ、R,G,B色データおよびテクスチャS,T座標によって指定される。３次元でプリミティブを表現するデータが、次いでフロントエンド・ボード10によって、85ビットのバス18を通してテクスチャ・マッピング・ボード12およびフレーム・バッファ・ボード14に提供される。テクスチャマッピング・ボードが、プリミティブを表現するスクリーン・ディスプレイ・ピクセルを計算するために受け取られたプリミティブ・データを補間し、そして、各プリミティブ・ピクセルについて対応する結果のテクスチャ・データを判断する。結果のテクスチャ・データが、フレーム・バッファ・ボードに図6に一つの母線として示される5つの55ビット・バス28を通して提供される。

【００５１】
フレーム・バッファ・ボード14は、ディスプレイ・スクリーンの上で各プリミティブ(基本要素、primitive）を表現するピクセルを計算するためそして各ピクセルのための対象物の色の値を決定するために、フロントエンド・ボード10から受け取られたプリミティブ・データを補間する。フレーム・バッファ・ボードが、各ピクセルのために結果として生ずる画像のR,G,B値を生成するために、次いで対象物の色の値をテクスチャマッピング・ボードから提供された結果のテクスチャ・データと、ピクセルごとに結合する。テクスチャ・マップされたプリミティブを表す結果として生ずる画像を画面上で表示するよう、ディスプレイ・スクリーン（図示しない）のピクセルを制御するため、各ピクセルのためのR,G,B色調節信号が、R,G,B線29にそれぞれ提供される。

【００５２】
フロントエンド・ボード10、テクスチャマッピング・ボード12とフレーム・バッファ・ボード14がそれぞれパイプラインされ、複数のプリミティブについて同時に動作する。テクスチャマッピングおよびフレーム・バッファ・ボードがフロントエンド・ボードによって提供されるプリミティブについて動作する間、ントエンド・ボードが、ボード12と14のパイプラインが満杯になるまで、新しいプリミティブについて動作し提供し続ける。

【００５３】
フロントエンド・ボード10は、分配器チップ0、３次元（3-D）幾何アクセラレータ（加速器）チップ32A、32Bおよび32C、２次元(2-D)幾何アクセラレータ・チップ34、および集信装置（コンセントレータ）チップ36を含む。分配器チップ30は、X,Y,Z座標およびカラーのプリミティブ・データをバス16を通してホストコンピュータから受け取り、3ーDのプリミティブ・データを3ーD幾何アクセラレータ・チップ32A、32B、および32Cに均等に分配する。このように、3つのグループのプリミティブが同時に処理されるので、システムの帯域幅が増大する。データが、40ビットのバス38Aを通して三次元の幾何加速器チップ32Aおよび32Bに提供され、40ビットのバス38Bを通してチップ32Cに提供される。バス38Aおよび38Bの両方が、60MHZの速度でデータを転送し、そして、2つの三次元の幾何アクセラレータ・チップを支援する十分なバンド幅を提供する。2ーDのプリミティブ・データは、44ビットのバス40を通して、2ーD幾何アクセラレータ・チップ34に、40MHZの速度で提供される。

【００５４】
各3-D幾何アクセラレータ・チップが、対応するスクリーン空間座標に受け取られるプリミティブを規定するx,y,z座標を変形させ、スクリーン空間座標のための対象物のR,G.B値およびテクスチャのS、T値を判断し、プリミティブ四辺形を三角形に分解し、そして、各三角形を規定する三角形面方程式を計算する。スクリーン内に複数のウィンドウが表示されるか、または、プリミティブの一部がディスプレイ・スクリーン上で表現される景観ボリュームを越えて広がるときに、結果として生ずる画像の正確なスクリーン表示を確保するために、各三次元の幾何アクセラレータ・チップも、景観クリッピング動作を実行する。3-D幾何アクセラレータ・チップ32A、32Bおよび32Cからの出力は、それぞれ44ビットのバス42A,42Bおよび42Cを通して60MHZの速度で集信チップ36に提供される。２次元の幾何アクセラレータ・チップ34は、46ビットのバス44を通して45MHZの速度で集信チップ36に出力データを提供する。集信チップ36は、3-D幾何アクセラレータ・チップ32AーＣから受け取られた三次元のプリミティブ出力データを結合し、それらが分配器チップ30による分配に先立ち持っていたもとの順序にプリミティブを再順序付けし、そして、バス18を通して、結合されたプリミティブ出力データをテクスチャマッピングおよびフレーム・バッファ・ボードに提供する。

【００５５】
テクスチャマッピング・ボード12は、テクスチャマッピング・チップ46、およびキャッシュ・メモリとして準備をされるのが好ましい局部メモリ48を含む。好ましい実施例において、キャッシュ・メモリは、後述の理由で、複数のSDRAM（同期ダイナミックＲＡＭ）チップから形成される。より詳細に下で説明するように、キャッシュ・メモリ48は、フレー

ム・バッファ・ボードで描画されていたプリミティブと関連するテクスチャMIPマップ・データを格納する。テクスチャMIPマップ・データは、ホストコンピュータ15の中のメイン・メモリ17からバス40を通り、2-D幾何アクセラレータ・チップ34を通り、24ビットのバス24をダウンロードされる。

【００５６】

テクスチャマッピング・チップ46は、ディスプレイ・スクリーン上で描画されるべきプリミティブを表わすプリミティブ・データをバス18を通して続いて受け取る。上で示したように、三次元の幾何アクセラレータ・チップ32A-Cから提供された22のプリミティブは、点、線および三角形を含む。テクスチャマッピング・ボードは、点または線のテクスチャマッピングを実行しないで、三角形プリミティブだけに作用する。三角形プリミティブを表現するデータは、少くとも一つの頂点、少くとも一つの頂点の対象物のカラーR、G、Bの値のためのx、y、zの対象物ピクセル座標、および三角形の少くとも一つの頂点および面方程式に対応するテクスチャ・マップの部分のS、Tの座標を含む。テクスチャマッピング・チップ46が、対象物ピクセルのz座標および対象物のカラーR、G、Bの値を無視する。チップ46が、x,yピクセル座標を補間し、プリミティブを表現する各x,yスクリーン・ディスプレイ・ピクセルに対応するSおよびT座標を補間する。各ピクセルのために、テクスチャマッピング・チップは、テクスチャMIPマップのそれに対応する部分をキャッシュ・メモリからアクセスし、そしてピクセルのため複数のテクセルの加重平均を含むことがありえる結果のテクスチャ・データを計算する。

【００５７】

一つの典型的な実施例において、キャッシュは256x256テクセルの64のブロックを格納する。先の技術システムのテクスチャマッピング・ハードウェアに使用された局部メモリと違って、この発明のキャッシュ・メモリは、大きいテクスチャについてのように描画中のプリミティブにマップするテクスチャのMIPマップの全シリーズを格納することはない。さらに、先行技術と違ってシステム、キャッシュ・メモリが、描画中のプリミティブにマップする複数のテクスチャのためのMIPマップのシリーズを格納することはない。むしろ、キャッシュ・メモリは、任意の一つの時には、実際に、現在描画中のプリミティブを描画することの使用されるMIPマップのシリーズの特定の部分だけを格納する。したがって、ほとんどの適用のために、描画中の画像のための完全なテクスチャ・データの一部だけが、任意の一時点でキャッシュ・メモリに格納される。

【００５８】

各テクスチャのためのMIPマップの完全なシリーズが準備され、ホストコンピュータ15の中のメイン・メモリ17に格納される。この発明のTIMは、ホストコンピュータのプロセッサ上で走る。描画中のプリミティブの各ピクセルについて、テクスチャマッピング・チップ46が、テクスチャMIPマップの対応する１つまたは複数のテクセルが現にキャッシュに存在するかどうか判断するために、キャッシュ・メモリ48のディレクトリをアクセスする。アクセスの時点で対応するテクセルがキャッシュ・メモリに格納されるならば、キャッシュ・ヒットが生じ、テクセルがキャッシュから読みとられ、フレーム・バッファ・ボードに渡される結果のテクスチャ・データを計算するためにテクスチャマッピングにチップ46によって処理される。

【００５９】

しかし、テクスチャマッピング・チップ46によってアクセスされるとき、プリミティブ・ピクセルについて対応するテクセルがキャッシュ・メモリに格納されていないならば、キャッシュ・ミスが生じる。キャッシュ・ミスが生じるとき、プリミティブを描画するために必要とされたテクスチャMIPマップ・データの部分が、ホストコンピュータ15の中のメイン・メモリ17からTIMを使用してキャッシュ・メモリ48に、おそらくは前に格納されたいくつかのデータを置き換えてダウンロードされる。しかし、描画中のプリミティブのMIPマップの全シリーズをダウンロードする従来のテクスチャマッピング・システムと異なり、この発明のTIMは、プリミティブを現在描画するのに必要なMIPマップのシリーズの部分だけかまたはその現在描画された部分だけをダウンロードする。より詳細に下で説明さ

れるように、キャッシュ・ミスが生じるとき、ホストコンピュータ15のテクスチャ割込みマネージャを開始するためにテクスチャマッピング・チップ46によって割込み制御信号が生成される。割込み制御信号が、線94を通して分配器チップ30に提供され、そこから線95を通してホストコンピュータに割込み信号が提供される。

【００６０】

要求されたテクスチャ・データが、TIMによってそのメイン・メモリから読み出され、フロントエンド・ボードとテクスチャマッピング・チップを通って三次元プリミティブ描画パイプラインをバイパスして、バス24を通してテクスチャマッピング・ボード48にダウンロードされる。このように、キャッシュ・ミス割込みが生じるとき、フロントエンド・ボードが、三次元のプリミティブに動作し続けることができ、キャッシュ・ミスを引き起こしたプリミティブに関連するテクスチャ・データがメイン・メモリ17からダウンロードされている間に、バス18を通してテクスチャマッピング・チップおよびフレーム・バッファ・ボードに、出力プリミティブ・データを提供する。従来のテクスチャマッピング・システムと対照的に、テクスチャ・データのテクスチャマッピング・ハードウェアへのダウンロードは、三次元のプリミティブ・パイプラインのフラッシングを必要とせず、システムのバンド幅および性能を向上させる。

【００６１】

各ピクセルのための結果のテクスチャ・データは、テクスチャマッピング・チップ46によってフレーム・バッファ・ボードに５つのバス28を通して提供される。5つのバス28は、フレーム・バッファ・ボードに設けられた5つのフレーム・バッファ・コントローラ・チップ50A、50B、50C、50Dおよび50Eにそれぞれ結合され、フレーム・バッファ・コントローラ・チップに、並列に結果のテクスチャ・データを提供する。フレーム・バッファ・コントローラ・チップ50A-Eは、関連するVRAM（ビデオＲＡＭ）チップ51A-Eのグループにそれぞれ結合される。フレーム・バッファ・ボードは、さらに4つのビデオ・フォーマット・チップ、52A、52B、52Cおよび52DならびにRAMDAC（ランダム・アクセス・メモリ・ディジタル／アナログ変換器）54を含む。フレーム・バッファ・コントローラ・チップは、ディスプレイ・スクリーンの異なる、重ならないセグメントを制御する。各フレーム・バッファ・コントローラ・チップが、フロントエンド・ボードからのプリミティブ・データをバス18を通して受け取り、バス28を通してテクスチャマッピング・ボードからの結果のテクスチャマッピング・データを受け取る。フレーム・バッファ・コントローラ・チップが、そのそれぞれのセグメントにおけるプリミティブを表現するスクリーン・ディスプレイのピクセル座標、および各ピクセル座標のための対応する対象物R、G、Bのカラー値を計算するために、プリミティブ・データを補間する、結果のテクスチャデータが、テクスチャマッピング・ボードから提供されるプリミティブ（すなわち、三角形）について、ピクセルごとに、フレーム・バッファ・コントローラ・チップが、対象物のカラー値および結果のテクスチャ・データを結合し、ディスプレイ・スクリーン上で表示される各ピクセルのための最終的なR、G、B値を生成する。

【００６２】

対象物とテクスチャのカラー値が結合される態様は、いくらかの異なる方法で制御することができる。例えば、置換モードにおいて、対象物のカラー値が、単にテクスチャのカラー値で置き換えられることができ、テクスチャのカラー値が、ピクセルを描画するのに使用される。代わりに、変調モードにおいて、対象物とテクスチャのカラー値を一緒に掛けて、ピクセルのための最終的なR、G、B値を生成することができる。さらに、対応するテクスチャ・カラー値が対象物のカラー値と組み合わせられる方法を規定する比率を指定する色調節ワードを各テクセルについて格納することができる。結果の色調節ワードを、各ピクセルに対応する結果のテクセル・データのために決定し、コントローラ・チップが、各ピクセルの最終的なR、G、B値を決定するために、対応する結果の調整ワードによって指定された比率を使用することができるよう、バス28を通してフレーム・バッファ・コントローラ・チップに提供することができる。フレーム・バッファ・コントローラ・チップ50A-Eによって生成された、各ピクセルのためのR、G、B値を含む結果の画像ビデオ・デー

タは、対応するVRAMチップ51A－Eに格納される。VRAMチップ51A－Eの各グループが、8つのVRAMチップを含み、40のVRAIMチップが１つのフレーム・バッファ・ボード上に位置する。が接続されるビデオ・フォーマット・チップ52A-Dのそれぞれが異なる組の10のVRAMチップに接続され、データを受け取る。ビデオ・データは、VRAMチップから直列にシフト出力され、64ビットのバス58A、58B、58Cおよび58Dを通して4つのビデオ・フォーマット・チップ52A、52B、52Cおよび52Dに33MHZの速度でそれぞれ提供される。ビデオ・フォーマット・チップが、RAMDACによって扱われることができるようにビデオ・データをフォーマットし、フォーマットされたデータを３２ビット・バス60A、60B、60Cおよび60Dを通してRAMDAC 54に33MHZの速度で提供する。RAMDAC 54が、転じてディジタル色データをアナログR、G、B色調節信号に変換し、各ピクセルのためのR、G、B制御信号をR、G、B制御線29にそってスクリーン・ディスプレイに提供する。

【００６３】

一つの実施例において、テクスチャマッピング・ボード12およびフレーム・バッファ・ボード14上のハードウェアが、繰り返されるので、ある種のプリミティブ描画作業を複数のプリミティブ上で平行して実行することができ、システムのバンド幅を増やすことができる。そのような代替実施例のハードウェア装置の例が、この発明のTIMを使用することができるコンピュータ・グラフィックス・ハードウェア装置のブロック図である図7に示される。特定のハードウェア装置のある種のハードウェアが繰り返されている。図7のシステムは、4つの三次元の幾何アクセラレータ・チップ32A、32B、32Cおよび32D、それぞれキャッシュ・メモリ48Aおよび48Bと関連する2つのテクスチャマッピング・チップ46Aおよび46B、およびそれぞれ関連するグループのVRAMチップを有する10のフレーム・バッファ・チップ50A-50Jを含む。図7のシステムの動作は図６のシステムのものとにている。図7の実施例のハードウェアの多重化が、システムのバンド幅を増大させるある種のプリミティブ描画動作を複数のプリミティブについて平行して実行することができる。

【００６４】

B.テクスチャマッピング・チップの概要

テクスチャマッピング・チップ46のブロック図が図8に示される。チップ46が、対象物およびテクスチャ・プリミティブ・データをフロントエンド・ボードから64ビットのバス18を通して受け取るフロントエンド・パイプライン・インターフェース60を含む。テクスチャマッピング・チップで操作された三角形プリミティブが52個の３２ビット・ディジタル・ワードによって規定されるが、異なる長さのワードによって規定されてもよい。パイプライン・インターフェースは、一組のマスター・レジスタおよび一組の対応するスレーブ・レジスタを含む。レンダリングの間、マスター・レジスタは、順にプリミティブを規定するデータの52のディジタル・ワードで満たされる。次いで、適当なレンダリング・コマンドを受領すると、データがパイプライン・インターフェースのスレーブ・レジスタにてシフトされ、パイプライン形態で、マスター・レジスタが別のプリミティブを表すデータで満たされる。バス18を通して提供されたプリミティブ・データは、少なくとも一つの三角形頂点のためのx、y、zベクトル座標データ、S、Tテクスチャ座標およびR、G、B対象物色データ、ならびに三角形の面方程式を表現するデータを含む。上で示したように、テクスチャマッピング・チップは、対象物ピクセルz座標および対象物のカラーR、G、B値を無視し、そして、フロントエンド・パイプライン・インターフェース60における他のデータだけを格納する。

【００６５】

パイプライン・インターフェース60のスレーブ・レジスタは、バス62を通してパラメーター補間回路回路64にプリミティブ・データを転送する。パラメーター補間回路回路64が、三角形を表現する各ディスプレイ・スクリーン・ピクセル座標について、ピクセルにマップするテクスチャ・マップのためのS、Tテクスチャ・マップ座標、そしてSおよびTの勾配値（ΔS、ΔT）を決定するために各プリミティブ三角形を補間する。SおよびTの勾配は、隣接ピクセルの間のSおよびTの座標変化にそれぞれ等しく、以下に説明する方法で計算される。

【００６６】
図9に示すパラメーター補間回路回路64は、すべて直列に接続したエッジ・ステッパー66、FIFO（「先入れ先出し」）バッファ68、スパン・ステッパー70、および勾配および遠近法修正回路72を含む。エッジ・ステッパーが、三角形頂点の一つのx、yピクセル座標で始まり、三角形面(plane)方程式を利用して、三角形の輪郭をステップさせ三角形の輪郭を規定するピクセル座標を決定する。各ピクセル座標について、テクスチャ・マップのどのテクセルが各ディスプレイ・スクリーン・ピクセルの座標に対応するかを識別するために、テクスチャ・マップのSおよびT座標が、三角形頂点のS、T値に基づいて決定される。一時的に、ピクセルおよびテクセル座標がFIFOバッファに格納され、それからスパン・ステッパーに提供される。三角形の輪郭に沿った各x、yピクセル位置でスパン・ステッパーが、三角形の対応するスパンにわたって、スパンに沿っての各ピクセル位置のS、Tテクセル座標を決定する。

【００６７】
テクスチャのためのMIPマップのシリーズの一つにおける一つのテクセルにピクセルが、直接（1対1対応で）マップしなければ、ディスプレイ・スクリーン・ピクセルのそれぞれのSおよびT座標は、整数部分と端数部分を持つことがある。上で説明したように、テクスチャ・マップにマップされるとき、各ディスプレイ・スクリーン・ピクセルは、テクスチャのためのMIPマップのシリーズの一つにおける複数のテクセルの間にあることがありえ、さらに、そして、シリーズの隣接する（大きさにおいて）MIPマップの間にあることがありえる。

【００６８】
勾配および遠近法の修正回路72が、各ディスプレイ・スクリーン・ピクセルのための勾配値Sおよび（ΔS、ΔT）を決定する。一つの実施例において、勾配ΔSが、勾配ΔSx、および勾配ΔSyの大きいものに選ばれる。勾配ΔSxは、座標xがディスプレイ・スクリーン上の隣接ピクセルの間で変わるときのテクスチャマップにおけるS座標における変化であり、勾配ΔSyは、座標yが隣接のピクセルの間で変わるときの変化である。勾配ΔTも同様に計算される。ディスプレイ・スクリーン・ピクセルについての勾配ΔS、ΔTは、対応するS、T次元におけるディスプレイ・スクリーン上の一つのピクセルの変化について、テクスチャ・マップ内の座標位置における変化の割合を示し、どのMIPマップ（１つまたは複数）が、そのピクセルに対する結果のテクスチャ・データを提供するためにアクセスされなければならないかを決定するのに使用される。たとえば（ディスプレイ・スクリーンのピクセルについて2に等しい勾配は、ピクセルが4つの（すなわち、下に説明するように2^2）テクセルにマップされることを示すので、大きさにおいて2だけベース・マップ（例えば、図1のマップ102）から縮小されたMIPマップが、そのピクセルのための結果のテクスチャ・データを提供するために、アクセスされなければならない。このように、勾配が増加するにつれて、ピクセルのための結果のテクスチャ・データを提供するためにアクセスされるMIPマップのサイズは、減らされる。

【００６９】
一つの実施例において、勾配が、そのピクセルのΔSx、ΔSy、ΔTxおよびΔTyのうち最も大きいものに等しくなるように、ΔSおよびΔTの大きいものと等しい一つの勾配が、各ピクセルのために適当なMIPマップを選ぶために使用される。しかし、勾配は、それらの値の最も小さいもの、それらの値の平均その他の組み合わせを選ぶなど異なる形態でも選ぶことができることが理解されなければならない。S、T座標の一つだけにおける変化の割合を示す一つの勾配が選ばれるから、勾配の二乗が、対応するピクセルにマップするテクセルの数を表現する。

【００７０】
勾配から、パラメーター補間回路が、ピクセルがマップする最も近いマップおよびそのマップからの距離を判断する。その情報が、最も近いマップの全マップ数として、そしてマップ小数値において最も近いマップからのピクセルの距離を示す小数値として提供される。

【００７１】
再び図8のテクスチャマッピング・チップのブロック図を参照すると、パラメーター補間回路回路64から出力されたテクセル・データが、線70を通してタイラー（画面分割器）兼境界チェッカー72に提供され、そこで、テクセル・データによって指定されたテクスチャ・マップのそれぞれの位置に最も近い4つのテクセルのアドレスが判断され、そして、それぞれがテクスチャの境界内にあるかどうか決定するためにチェックされる。テクセル・データは、補間されたS、T座標（整数と小数値）、およびマップ番号およびマップ断片を含む。タイラーは、パラメーター補間回路64で計算したSおよびT座標の整数部分を使用し、それぞれの整数部分に１を加えて四つの最も近いテクセルのアドレスを生成する。境界チェッカーが、次いでこれらの4つのテクセルのいずれかのためのS、T座標がテクスチャ・マップの境界の外側に落ちるかどうか判断する。ディスプレイ・スクリーン・ピクセルがテクスチャ・マップの境界の外側に落ちるS、T座標位置にマップするならば、いくつかのテクスチャマッピング方式の一つが、結果のテクスチャ・データがそのピクセルのために生成されることになっているかどうか、およびどのようにそのデータが生成されるべきかを決定するために実行される。そのような方式の例は、ラッピング（テクスチャの繰り返し）、ミラーリング（テクスチャのミラー画像の繰り返し）、テクスチャマッピングを境界の外側にそらすこと、および同一調の色を境界の外側に表示することを含む。

【００７２】
その境界を越えたテクスチャ・マップのロケーションにピクセルをマップするのを可能にする能力は、テクスチャが、対象物プリミティブにマップされる態様に融通性を提供する。例えば、テクスチャが対象物の複数の部分にマップされるような繰り返し形態で対象物にテクスチャをマップするのが望ましいことがある。。たとえば、テクスチャが、[0、0]含むから（10，10）含まずにわたるS、T座標を持つと規定されるならば、ユーザが、S、T座標、[10，10]含むないし(20，20)含まずにマップするために対象物のある部分を指定してもよい。角括弧でくくられた内包座標の表記法は、それらの座標が対象物にマップされるテクスチャの部分に含まれることを示し、丸括弧の非内包座標の表記法は、対象物がそのS、T座標の前までだけにマップし、丸括弧の非内包座標は含まないことを示す。折り返し（wrapping：ラッピング）機能が、テクスチャの境界の外側に落ちるS、T座標のために選ばれるならば、S、T座標[10、10]ないし（20、20）は、S、T座標[0、0]ないし（10、10）のテクセルにそれぞれマップする。

【００７３】
上で説明したように、一つのピクセルのための２次元テクスチャ・マップからの結果のテクスチャ・データは、8つのテクセルの組合せの結果すなわち最も近い2つのMIPマップにおける最も近い4つのテクセルであることがある。結果のテクセル・データを生成するために8つのテクセルが、結合されるいくらかの方法がある。例えば、平均算出が必要でなくなるよう最も近いマップにおける最も近くの一つのテクセルを選ぶことができる。代わりに、最も近い二つのマップのそれぞれにおける最も近くの一つのテクセルを勾配の値に基づいて一緒に平均することができる。そのような方式は、最も近い八つのテクセルが平均されるときのように正確にテクスチャをマップしない。

【００７４】
ハードウェア装置の一つの実施例において、３線形の(trilinear,３点直線)補間が支援され、一つのピクセルのための結果のテクスチャ・データが、8つのテクセルの加重平均として計算されることがある。そこからテクスチャ・データをアクセスする最も近い二つのMIPマップを識別するために、S、Tの変化の割合を表わす勾配が使用され、そして各マップ内の最も近い4つのテクセルがアクセスされる。各マップ内の4つのテクセルの平均は、ディスプレイ・スクリーン・ピクセルがマップするMIPマップの位置のS、T座標にどのテクセルが最も近いかに基づいて重み付けられる。ピクセルのSおよびT座標の端数部分が、この重み付けを実行するために使用される。次いで最も近い二つのMIPマップのそれぞれから平均値が、勾配の値に基づいて重み付けられる。小数値は、この重み付け処理に使用する勾配から計算される。例えば、3の勾配は、それぞれ2と4の勾配に対応するMIPマップ

の間の中間である。

【００７５】

テクセル補間プロセスは、テクセル補間回路76によって実行される。各ディスプレイ・スクリーン・ピクセルのためのSおよびT座標の端数の部分が、パラメーター補間回路からタイラー/境界チェッカーを通り、線74によってテクセル補間回路76に提供される。端数の部分が、結果のテクセル・データを計算するとき、複数のテクセルの補間の間に各テクセルに与えられる重みを決定するためにテクセル補間回路によって使用される。

【００７６】

上で説明したように、描画中のプリミティブと関連するテクスチャMIPマップは、ローカルにキャッシュ・メモリ48（図11）に格納される。一つの実施例において、キャッシュは充分に連想記憶である。キャッシュは、4つのインタリーブに分割された8つのSDRAMチップを含み、各インタリーブに2つのSDRAMチップがある。各インタリーブ内のSDRAMチップを同時にアクセスすることができるように、各インタリーブに一つが対応する4つの別々のコントローラが備えられる。各SDRAMチップが、メモリの2つの別個のバンクを含み、ここでは、従来のDRAMにおいてデータを2つの異なるページから（すなわち２つの異なる行アドレスから）アクセスするときに一般的な再ページ割付のペナルティーを負うことなく、メモリの異なるページが連続的な読み出しサイクルでアクセスされることができる。

【００７７】

テクスチャ・データ（すなわち、MIPマップ）は、256x256のテクセル・ブロックのデータに分割される。キャッシュ・メモリは、64のブロックのデータを一度に格納することができる。各ブロックが、ブロックをユニークに識別する関連したブロック・タグを持つ。キャッシュは、現在キャッシュに格納されているデータのブロックに対応するブロック・タグを格納するキャッシュ・ディレクトリ78を含む。より詳細に下で説明するように、各ブロック・タグは、データのブロックが表現する特定のテクスチャを識別するテクスチャ識別子（テクスチャID）、データのブロックが表現するマップのテクスチャ・シリーズ内で特定のMIPマップを識別するマップ番号、および特定のマップ内でデータのブロックのロケーションを識別する高次のSおよびT座標を含む。キャッシュ・ディレクトリ内のブロック・タグの物理的なロケーションは、キャッシュ・メモリ内の対応するデータのブロックのロケーションを表現する。

【００７８】

２以上のテクスチャからのMIPマップが、異なるテクスチャを見分けるテクスチャ識別子とともにキャッシュ・メモリに同時に格納されることがある。あるMIPマップは、256x256より少ないテクセルしか含まず、したがってデータの全ブロックを消費しない。例えば、一連のMIPマップにおけるより小さいマップまたは小さいテクスチャのための大きいマップでさえ、256x256テクセルを越えることはない。メモリ空間を効率的に利用するために複数のマップの部分が、各マップ部分をブロック内のサブブロックに割り当てて、テクスチャ・データの一つのブロックの格納されることがある。一つのブロック内に格納された複数のマップのそれぞれが、マップのブロック内のロケーションを識別する関連するサブテクスチャ識別子（ID）を持つ。

【００７９】

レンダリングの間、タイラー/境界チェッカー72が、テクスチャ・データのブロックのために描画すべきピクセルにマップするキャッシュ読みとりタグを生成する。キャッシュ・ブロック・タグとキャッシュ読みとりタグが生成される態様をより詳細に下で説明する。タグは、テクスチャのテクスチャIDを表わす8ビットを含む23ビットのフィールドであり、1ビットがテクスチャ・データのマップ番号を決定するのに使用され、7ビットがテクスチャ・データの高次のSおよびT座標を決定するのに使用される。キャッシュ・ディレクトリ78が、タイラー/境界から提供されるキャッシュ読み取りタグをディレクトリに格納されたブロック・タグと比較して、レンダリングに使用されるテクスチャ・データのブロックがキャッシュ・メモリにあるかどうか判断する。描画すべきプリミティブにマップするテクスチャ・データのブロック・タグがキャッシュ・ディレクトリに格納されている（す

なわち、ヒット）ならば、キャッシュ・ディレクトリが、ヒット・タグに対応するキャッシュのテクスチャ・データのブロックの物理的なロケーションを示すブロック索引を生成する。ブロック索引の計算のより詳細を下で説明する。また、テクセル・アドレスが、タイラー/境界チェッカー72によってキャッシュから読みとられる各テクセルについて生成され、ブロック内のテクセルのロケーションを示す。テクセル・アドレスは、より大きいサイズのマップの補間されたS、T座標の下位アドレス・ビットを含み、より小さいサイズのマップのために後述するアルゴリズムに基づいて計算される。ブロック索引およびテクセル・アドレスは、キャッシュ内のテクセルのロケーションを示すキャッシュ・アドレスを含む。

【００８０】

より詳細に下で説明されるように、テクセルが4つのキャッシュ・インタリーブのどれに格納されるかを決定するために、SおよびT座標のLSBがデコードされ、キャッシュのアドレスされたロケーションに格納されたテクセル・データを読みとるために、キャッシュ・アドレスの残っているビットが、線84を通してコマンドと一緒にテクセル・キャッシュ・アクセス回路82に提供される。

【００８１】

キャッシュ読みとりタグが、キャッシュ・ディレクトリ78に格納されたブロック・タグのいずれにも一致しないならば、ミスが生じ、そして、キャッシュ・ディレクトリ78が、線94（図4）を通してフロントエンド・ボードの上の分配器チップ30に割込み制御信号を生成する。分配器チップ30が、線95を通してホストコンピュータ15に割込みを生成する。割込みに応答してTIMは、キャッシュ・ディレクトリからミスしたブロック・タグを読みとり、フロントエンド・ボード10およびテクスチャマッピング・チップ46における三次元のプリミティブ・パイプラインをバイパスする態様で、キャッシュ・メモリにテクスチャ・データの対応するブロックをダウンロードする。TIMからダウンロードされたテクスチャ・データが、バス24により、テクセル・ポート92（図8）を通ってテクセル・キャッシュ・アクセス回路82に提供される。

【００８２】

回路82は、キャッシュ・メモリを形づくるSDRAMに、データを書き込む。キャッシュ・ミスが生じるとき、テクスチャマッピング・チップは、ミスが生じたプリミティブの処理を続ける前に、新しいテクスチャ・データがダウンロードされるのを待つ。しかし、キャッシュ読取りに続くパイプラインのステージは、ミス・プリミティブに先立ち受け取られたプリミティブの処理を続ける。同様に、キャッシュ読取りに先行するパイプラインのステージも、パイプラインがキャッシュ読取り動作の後にいっぱいにならなければ、そしていっぱいになるまで、新しいテクスチャ・データのダウンロードを待ちながら、プリミティブの処理を続ける。

【００８３】

レンダリングの間、フレーム・バッファ・ボード14のパイプラインの後のステージは、プリミティブに対応するテクスチャ・データがテクスチャマッピング・ボードから受け取られるまで、プリミティブの処理を続けない。したがって、キャッシュ・ミスが生じ、そして、テクスチャマッピング・チップが新しいテクスチャ・データがダウンロードされるのを待つとき、同様に、フレーム・バッファ・ボード14は、結果のテクスチャ・データがテクスチャマッピング・チップから提供されるのを待つ。テクスチャマッピング・チップと同様に、テクスチャマッピング・データを受け取るステージに続くパイプラインのステージは、ミス・プリミティブに先立ち受け取ったプリミティブの処理を続け、そして、テクスチャマッピング・データを受け取るステージに先行するパイプラインのステージも、パイプラインがいっぱいでないなら、そしていっぱいになるまで、プリミティブの処理を続ける。

【００８４】

キャッシュ・ミスに応答して新しいテクスチャ・データを待っているときに、テクスチャマッピング・ボードまたはフレーム・バッファ・ボードのいずれかのパイプラインがバッ

クアップするとき、同様に、フロントエンド・ボード10のパイプラインがバックアップする。キャッシュ・ミスが生じ、その結果、そしてホストコンピュータ・メイン・メモリへのアクセスおよび完了するのにいくつかのサイクルをとるテクスチャ・データのダウンロードになるので、フレーム・バッファ・ボードのパイプラインがバックアップされることにより、テクスチャマッピング・チップのパイプラインが決して待たなくてもよいようにすることが望ましい。したがって、一つの実施例において、　フレーム・バッファ・ボードが、テクスチャマッピング・ボードより深いプリミティブ・パイプラインを備えているので、テクスチャマッピング・パイプラインは、フレーム・バッファ・パイプラインが利用可能になるのを待つことによって遅延してはならない。

【００８５】

一つの実施例において、テクスチャマッピングをオフにする能力が備えられている。これは、ホストコンピュータのプロセッサ19上で操作しているTIMが、テクスチャマッピング・ボード12およびフレーム・バッファ・ボード14の両方におけるレジスタをセットすることによって達成される。テクスチャマッピングをオフにするようセットされたとき、これらのレジスタは、それぞれテクスチャマッピング・チップ46がフレーム・バッファ・ボード14にテクスチャ・データを提供させないようにし、そしてフレーム・バッファ・ボードにテクスチャマッピング・ボードからのテクスチャ・データを待つことなくプリミティブの描画を続けるよう命令する。

【００８６】

先に述べたように、２次元テクスチャ・マップからのテクスチャ・データで描画される各ディスプレイ・スクリーン・ピクセルについて、一つのMIPマップからの4つのテクセル（２線形補間）または2つの隣接するMIPマップからの8つのテクセル（3線形補間）を、そのピクセルのための結果のテクスチャ・データを決定するためにキャッシュ・メモリからアクセスすることができる。キャッシュから読みとられたテクセルが、バス86（図8）を通してテクセル補間回路76に提供され、回路76は、各ピクセルのための結果のテクセル・データを計算するために複数のテクセルを補間する。補間は、システムのために確立されたモードに基づいて変化することができる。点サンプリング補間モードが確立されるとき、結果のテクセル・データは、テクスチャ・マップにおけるピクセルのS、T座標によって規定されるロケーションに最も近い一つのテクセルに等しい。代わりに2線形または3線形補間が使われるとき、結果のテクセル・データは、それぞれ一つまたは二つの最も近いマップにおける四つまたは八つの最も近いテクセルの重み付けされた平均である。複数のテクセルのそれぞれに与えられた重量が、勾配の値、およびタイラー/境界チェッカーからテクセル補間回路76に提供されるSおよびT座標の端数の成分に基づいて決定される。

【００８７】

ディスプレイ・スクリーン・ピクセルのための結果のテクセル・データは、バス88を通してフレーム・バッファ・インターフェースFIFOバッファ90に順に提供される。フレーム・バッファ・インターフェースFIFOバッファ90は、最高64の結果のテクセルを格納することができる。

【００８８】

各結果のテクセルは、R,G,Bおよびαのそれぞれを表わす8つのビットを含む３２ビット・ワードである。αバイトは、フレーム・バッファ・ボード14（図6）に、テクセルにマップする任意のピクセルのための最終的なディスプレイ・スクリーンR,G,B値の計算において、結果のテクスチャ・データのR,G,B値が、フレーム・バッファ・ボードによって生成された対象物データのR,G,B値と組み合わせられなければならない態様を示す。フレーム・バッファ・インターフェースFIFOバッファの出力T0-T4がフレーム・バッファ・ボード14（図6）にバス28を通して提供される。フレーム・バッファ・ボードが、各ディスプレイ・スクリーン・ピクセルのための最終的なR,G,B値を生成するため?a?によって指定された態様で、結果のテクセル・データのR,G,B値を対象物R,G,B値と組み合わせる。

【００８９】

C. キャッシュ・メモリ組織

図10は、テクセル・ポート92、テクスチャ補間回路76、キャッシュ・ディレクトリ78およびテクセル・キャッシュ・アクセス回路82を含むテクスチャマッピング・チップの部分に結合した一つ実施例のキャッシュ・メモリのブロック図である。この実施例において、キャッシュ・メモリ48は、4つのインタリーブ204A、204B、204Cおよび204Dを含む。各インタリーブが、同時にアクセスされることができ、読取りサイクルの間、それぞれがデータの8ビットを提供する2つのSDRAMチップ（図示しない）を含む。したがって、一つの読取りサイクルの間、各インタリーブがテクセル・データの16ビットを提供する。テクセル・データの各３２ビット・ワードが、8ビットを、インタリーブの各SDRAMにおける2つの連続ロケーションのそれぞれに格納して、一つのインタリーブのキャッシュに格納される。このようにキャッシュからテクセルを読み出すために、2つの読取りサイクルは、テクセル・データの32のビットを提供するために適当なインタリーブの連続的なロケーションで実行される。下で説明するように、各インタリーブ内で2つの連続サイクルでデータのバーストを生むために、一つのアドレス・ワードだけ（行および列のデータを含む）が、SDRAMに提供される必要がある。バーストは、所与のアドレスから第１のサイクルで提供された16のビット、および1だけ増加された同じ行および列を持つアドレスから第２のサイクルで提供された16のビットを含む。

【００９０】

テクセル・キャッシュ・アクセス回路82は、コントローラA (200A)、コントローラB (200B)、コントローラC (200C)とコントローラD (200D)と分類された4つの別々のコントローラを含む。4つのコントローラA、B、C、およびDは、並行バス202A、202B、202Cと202Dを介して4つのインタリーブ204A、204B、204Cおよび204Dから同時にデータをアクセスすることができる。コントローラは、コマンドに応答し、それぞれ、バス84A、84B、84Cおよび84Dを通して受け取られたアドレスでメモリ48からテクセル・データを読む。

【００９１】

先に述べたように、各ピクセルは、一つのMIPマップから4つのテクセルにまたは複数のMIPマップから8つのテクセルに潜在的にマップすることができる。より詳細に下で説明するように、キャッシュにダウンロードされたテクセル・データが、ホストコンピュータのメイン・メモリにTIMによって組織されるので、各MIPマップの任意の4つの隣接テクセルが別々のインタリーブで見つけられ、平行してアクセスされることができる。このように2線の補間により結果のテクセル・データを生成するために必要とされることがあるMIPマップにおける任意の4つの隣接テクセルは、一つの読取り動作で読取ることができる。3線の補間が使われるとき、隣接のMIPマップからの4つのテクセルの2つの組を、2つの読取り動作で読みとることができる。

【００９２】

図11が、キャッシュ・メモリの4つのインタリーブを利用してMIPマップの任意の4つの隣接テクセルが同時に読みとられるように、テクスチャ・データ（あるテクセルだけは示される）のブロックが組織される態様の例を図示する。それぞれのテクセルが、テクセルが格納されているキャッシュ・メモリのインタリーブを識別するためにA, B, CまたはDのラベルをつけられる。マップのロケーションがA、B、CおよびDと分類された4つのテクセルの間にくるようにA-Dラベルのパターンが、繰り返す。このように、マップ内のロケーションにマップするピクセルについて、4つの最も近いテクセルが別々のインタリーブA-Dにあるので、それらは、同時に4つの独立のコントローラ200A-Dによってアクセスされることができる。例えば、ピクセルP0が、A、B、CとDと分類された4つのテクセルの間のロケーションにマップし、そして、ピクセルP1は、B、A、DおよびCと分類された4つのテクセルの間のロケーションにマップする。

【００９３】

上記の説明されたキャッシュ実施例は、単なる例として示されており、別の例を使うことができることを理解しなければならない。例えば、キャッシュは、8つの別々のインタリーブで8つの別々のコントローラとともに実行されることができるので、3線形の補間が使われるとき、8つのテクセルは、一つの読取り動作で同時にキャッシュからアクセスする

ことができる。

【００９４】

キャッシュ・メモリのそれぞれのSDRAMチップが、同時に別々の能動ページ（すなわち共通の行アドレスを持つ記憶場所のグループ）を維持することができる2つの同じ大きさのバンクに内部的に分割される。このように、データは、従来のDRAMの異なるページからのデータの連続的読み取りに一般に関連する再ページ割付のペナルティーを負うことなく、連続的な読取りサイクルでSDRAMチップの2つのバンク内の異なるページからアクセスされることができる。

【００９５】

より詳細に下で説明されるように、3線形の補間が実行されるときページをクロスさせるペナルティーを最小にするため、テクスチャ・データが、キャッシュ・メモリにおいてこのSDRAMの特徴を利用するよう構成されている。3線形の補間のために必要な8つのテクセルは、2つのMIPマップからの4つのテクセルの組を含む。一つのマップの4つの隣接のテクセルの各組は、一つがインタリーブA、B、C、およびDのそれぞれに、4つのテクセルが同時にアクセスされうるように上で説明された態様で格納されるように準備をされる。

【００９６】

さらに、任意のテクスチャのためのマップのシリーズの隣接のMIPマップからの共通のデータは、キャッシュの異なるSDRAMバンクに格納される。3線形の補間が実行されるとき、一つのMIPマップからの4つのテクセルが、第１のバーストの2つの読取りサイクルの間にインタリーブA-DのSDRAMバンクの一つから同時に読取られ、そして、隣接のMIPマップからの4つのテクセルは、続いて起こるバーストの2つの読取りサイクルの間に他のSDRAMバンクから読取られる。SDRAMの両方のバンクが同時に行-能動でありうるから、4つのテクセルの2つの組は、再ページ割付のペナルティーを負うことなく背中合わせのバーストでアクセスされることができる。対象物のピクセルが描画されているとき、隣接のピクセルが、そのテクスチャの同じ2つのMIPマップに頻繁にマップし、キャッシュへの読取りが、2つのマップに共通のデータを格納するキャッシュ・ブロックの間で連続的に切り替わることを要求することを理解しなければならない。ディスプレイ・スクリーン・ピクセルの描画中に2つの隣接のMIPマップの間で切り換えるとき、サイクル毎に再ページ割付ペナルティーを負うことなく、3線形の補間が実行されることを許すので、2ページが各SDRAM内で能動のままであるのを許すこのキャッシュ構成が有利である。

【００９７】

図12は、キャッシュ・メモリの上記の説明された実施例のより詳細なブロック図である。キャッシュは、4つのインタリーブ204A-204Dの間で均一に分けられたSD1-SD8と分類された8つのSDRAMチップを含み、それぞれのインタリーブが2つのSDRAMチップを含む。それぞれのインタリーブの2つのSDRAMは、次の共通の線を共有する：アドレス線（ADD）、行および列アドレスストローブ（RASとCAS）、書き込みイネーブル(WE)、クロック・イネーブル(CKE)、およびデータ入出力マスク（DQM）。各インタリーブ内のSDRAMが、8つの別々のデータ線に結合され、これを通して8ビットのデータが、それぞれの読取りまたは書込みサイクルの間にそれぞれ読取られまたは書き込まれる。それぞれのSDRAMチップが、メモリの2つのバンクを含み、それぞれのバンクがテクスチャ・データの1,048,576もの８ビット・ワードを格納する。

【００９８】

各インタリーブの2つのSDRAMが、同時にアクセスされ、SDRAMの一方がデータビット[15:08]を提供し、他方がデータビット[07:00]を提供してデータの16ビットを一緒に提供することができる。上で示したように、一つのバーストの2つの連続的な読取りサイクルが、それぞれのインタリーブからデータの完全な３２ビットのテクセルを与え、別々の８ビット・ワードがテクセルのR、G、Bおよびα値のそれぞれを表わす。

【００９９】

SDRAMチップは、それぞれのバンク内の1,048,576の８ビット・ワードをデコードするために線ADD上で多重化された20のアドレス・ビットを受け取る。詳細に下で説明するように

、6ビットのブロックインデックスおよび6ビットのテクセル・アドレスは、キャッシュからアクセスされるそれぞれのテクセルのためにために計算される。ブロックインデックスは、テクセルが、データの64のブロックのどれに位置するかを示し、そして、テクセル・アドレスは、ブロック内のテクセルの正確なS、T座標アドレスを示す。8つのSビットと8つのTビットは、256x256テクセルの正方形のブロックを仮定するテクセル・アドレスを含む。キャッシュ・アドレスは、ブロックインデックス（6つのMSB）とテクセル・アドレス（16のLSBS）の組合せを含む22のビット・ワードである。キャッシュ・アドレスは、キャッシュ内のテクセルの正確なロケーションを示す。

【０１００】

レンダリングの間、タイラー/境界チェッカーが、テクセル・アドレスのLSB SビットとLSB Tビット（すなわち、LSB S座標とLSB T座標）をデコードし、テクセルがキャッシュの4つのインタリーブのどれに格納されているか判断する。キャッシュ・アドレスの残りの20のより大きいアドレス・ビットは、アドレス線ADDに沿って適当なインタリーブ内の2つのSDRAMチップに提供される。2つのSDRAMSに提供された20のアドレス・ビットのうち9つのビットが、テクセル・データをアクセスするため、SDRAM内の列を選ぶために使用されて、11ビットが行を選ぶために使用される。当業者によって理解してなければならない列および行のアドレス・ビットが、SDRAMに異なるサイクルでラッチされ、そして、RASとCASストローブは、従来通りにデータをアクセスするために使用される。

【０１０１】

2サイクルのバースト中、第１のサイクルの間に、同じインタリーブ内の2つのSDRAMのアドレスされたロケーションからの16のビットが提供され、次いで別のアドレスを提供することなく、第２のサイクルの間に2つのSDRAMの別のロケーションからの16ビットが提供される。他方のアドレスが、1だけ増加された同じ行アドレスおよび列アドレスを含む。また、ひとたび1ページ（特定の行アドレス）が起動されると、異なる行アドレスが提供されるまで、残りが起動されたままになることを理解しなければならない。したがって、同じインタリーブからアクセスされる連続的なテクセルが同じページ（同じ行アドレスを含む）にあるならば、次いで、行アドレスは、連続バーストの最初のものの間に一度だけ提供される必要がある。

【０１０２】

さらに、RAS、CASおよびWE線は、従来の態様でSDRAMチップをアドレスしデータを書くために使用される。クロック・イネーブル信号CKE信号が非アクティブにされるとき、内部クロックは、中断される。SDRAMは、この信号に応答してデータをそのまま保持し 両方のバンクをアイドル状態にする。データ入出力マスクDQM信号が、読取りサイクルの間、出力イネーブルとして、そして書込みサイクルの間、入力データ・マスクとしての役目をする。

【０１０３】

従来のSDRAM使用は、現在のデータ読取りサイクルが完了する前に現在のページから現在のデータをアクセスし、その将来のページを起動させながら、次のデータがどの将来のページからアクセスされるか決定することを含む。SDRAMが2つの異なるページを同時にアクティブにイネーブルするので、従来のSDRAM使用は、一般に、従来のDRAMの異なるページからデータをアクセスすることに関連する再ページ割り付けのペナルティーを避ける。従来のSDRAM使用は、しかし多くの連続的な読取りサイクルで読取られるべきデータが異なるページに位置するとき、先を見て将来のページを起動させるために2以上のサイクルが必要なので、そのような利点を提供しない。ここで説明されたテクスチャ・データ記憶方法は、ペナルティーを負うことなく異なるページから複数の連続的なSDRAM読取りサイクルが生じるのを可能にすることによって従来のSDRAMに対して利点を提供する。特に、テクスチャの隣接MIPマップから共通データをSDRAMの別々のバンクに格納する（それは3線形の補間の実行中連続的な読取りサイクルの間、アクセスすることを必要とする）ことによって、別々のバンクからのデータは、連続的な読取りサイクルでペナルティーなしでアクセスすることができる。SDRAM性能を改善するためのデータ記憶割当て方法を示し、テ

クスチャマッピング・データの記憶に関して説明したが、この方法がそう制限されないことを理解しなければならない。特に、この方法は、複数の連続的な読取りサイクルがデータを異なる記憶場所からアクセスしなければならない場合、任意のタイプのデータを割り当てるために適用できる。

【０１０４】

D. キャッシュ制御FIFO

図13は、境界チェッカー72、キャッシュ・ディレクトリ78、キャッシュ・アクセス回路82、キャッシュ・メモリ48およびテクセル補間回路76を含む部分的なテクスチャマッピング・チップのより詳細なブロック図である。テクセル・キャッシュ・アクセス・ユニット82は、4つのキャッシュ・アクセス・コマンドFIFO 206A、206B、206Cおよび206Dを含む。キャッシュ・アクセス・コマンドFIFO 206A-Dは、16ビットのバス84A、84B、84Cと84Dを通して境界チェッカーからそれぞれ受け取ったキャッシュ・アクセス・コマンドを格納する。キャッシュ・アクセス・コマンドFIFO 206A-Dは、それぞれ図10に示されたコントローラ200A-Dに対応する。たとえば、FIFO 206Aのコマンドは、インタリーブ204A内のSDRAMのキャッシュ・アクセスを起動する。この実施例において、それぞれのキャッシュ・アクセス・コマンドFIFOは、8つの16ビットのコマンドを一時的に格納することができる。このように、システムのパイプライン能力を高めるために、キャッシュ・アクセス・ユニットが行動する前に、8つのコマンドがキャッシュ・アクセス・コマンドFIFOのそれぞれに格納されることができる。

【０１０５】

上で示したように、レンダリングの間、境界チェッカー72が、作用中のピクセルにマップするテクスチャ・データの各ブロックについて読取りキャッシュ・タグをキャッシュ・ディレクトリ78に格納されたブロック・タグの各々と比較し、テクセルがキャッシュにあるかどうか判断する。ヒットが生じるならば、キャッシュ内のテクスチャ・データの対応するブロックのロケーションを表すブロックインデックスが生成される。タイラー/境界チェッカーが、特定のテクセルの補間されたS、T座標、テクスチャIDおよびサブ・テクスチャID、ならびにそこからテクセルがアクセスされるマップのマップ番号および詳細を後述するテクスチャのベース・マップのサイズから、テクセル・アドレスを決定するルーチンを同時に実行する。ブロックインデックスとテクセル・アドレス（一緒にキャッシュ・アドレスを構成する）から、オプティマイザー（最適化器）が、後述するようにテクセルが格納されるキャッシュの特定のインタリーブ、およびそのインタリーブのSDRAMチップの列および行アドレス・ビットを次いで判断する。アドレス情報は、対応するキャッシュ・アクセス・コマンドFIFOに（キャッシュを読むコマンドと一緒に）提供される。

【０１０６】

テクセル補間回路76は、214A0、214Al、214B0、214B 1、214C0、214C1、214D0および214D1と分類された8つのテクセル・データFIFOを含む。テクセル・データFIFO 214A0および214A1が、キャッシュ・メモリのインタリーブ204Aに対応し、FIFO 214B0および214Blがインタリーブ204Bに対応し、FIFO 214C0および214C1がインタリーブ204Cに対応し、FIFO 214D0および214D1がインタリーブ204Dに対応する。

【０１０７】

先に述べたように、キャッシュ・メモリの4つのインタリーブのそれぞれは、別々のキャッシュ・アクセス経路を通して同時にアクセスすることができる。レンダリング中にテクセル・キャッシュ・アクセス・ユニット82がテクセル・データをキャッシュ・メモリ48からアクセスするとき、テクセル・アクセス・コントロール・ワードが、バス208A、208B、208Cおよび208Dを通してキャッシュ・メモリ48に提供される。4つのテクセルは、2つの背中合わせの16ビットの読取りサイクル中に同時に4つのインタリーブからアクセスされる。4つのテクセルが、バス210A、210B、210Cおよび210Dを通してテクセル・データA FIFOの一つ（214A0または214A1）、テクセル・データB FIFOの一つ（214B0または214B1）、テクセル・データC FIFOの一つ（214C0または214C1）、およびテクセル・データD FIFOの一つ（214D0または214D1）にそれぞれ提供される。それぞれのインタリーブA-Dに対応する

テクセル・データFIFOの対（すなわち、ゼロおよび1）が交互の形態でロードされる。例えば、インタリーブAから読みとられた第１のテクセルが、テクセル・データFIFO 214A0に格納され、インタリーブAから読みとられた第２のテクセルが、FIFO 214A1に格納され、インタリーブAからの第３のテクセルがFIFO 214A0に格納される、等々。この交互方式は、下で説明される理由のために用いられる。

【０１０８】

テクセル・データFIFOのそれぞれは、幅32ビットと8段の深さである。組合せにおいて、8つのFIFO 214が8つのパイプラインされた段を格納し、8つのテクセルを含む各段が3つの線の補間の間に結果のテクセル・データを決定するために使用された。バス210A。210B、210C、および210Dは16ビット幅である。各インタリーブの各SDRAM対は、各読取りサイクルの間、16ビットのデータを提供する。各バーストの間、第１の16のビットが、各SDRAM対から第１の16ビットのレジスタ（示さない）に提供され、そして次の16ビットが、各SDRAM対から第２の16ビット・レジスタ（示さない）に提供される。バーストの第２のサイクルの終わりに両方のレジスタからのデータが、対応する３２ビットのバス212A、212B、212Cまたは212Dに提供される。結果のテクセル・データをピクセルのために決定するためにテクセル補間回路76が、8つのテクセルの次のステージを読取るためにFIFOをアクセスし、それらのテクセルを上で説明したた態様で補間する。結果のテクセル・データが、次いでバス28を通してフレーム・バッファ・ボード14（図6）に提供され、上で説明した態様でディスプレイ・スクリーン・ピクセルを描画するのに使用される。

【０１０９】

3線形の補間が実行されるとき、一つのMIPマップの4つのテクセルと隣接のMIPマップの4つのテクセルから任意のピクセルのための結果のテクセル・データが、補間される。隣接のディスプレイ・スクリーン・ピクセルは、概して連続して描画される。しばしば、隣接のディスプレイ・スクリーン・ピクセルが、テクスチャMIPマップの隣接するロケーションにマップする。この結果連続的に描画されたプリミティブのために結果のテクセル・データを補間するときに、ある共通のテクセル・データが、使用されることがよくある。一つの実施例において、いくらかの密に間隔をあけられた読取りサイクル内で共通のテクセル・データが複数回アクセスされる時、キャッシュは、第１の読取りのためにアクセスされるだけであり、 各連続した読取りのためにキャッシュ読取りサイクルを節約する。最も最近読取られたテクセルは、テクセル・データFIFO内に格納される。このように、それらのテクセルへの後続のアクセスは、キャッシュからではなくFIFOからされる。それによって、これは必要なキャッシュ・アクセスの数を減らし、 システムのバンド幅を増やす。

【０１１０】

テクセル・データ経路A、B、C、およびDのそれぞれについて、前のピクセルのためにテクセル・データFIFO 0または1の一つに最も最近書かれたテクセル・データが、キャッシュをアクセスするためパイプライン位置に現在あるピクセルのためのテクセル・データに一致するならば、キャッシュ・アクセス・コマンドは、対応するキャッシュ・アクセスFIFO 206A、B、CまたはDに提供されない。その代わりに、コマンドがテクセル補間回路に送られ、テクセル・データが、対応するテクセル・データFIFO 214A、B、CまたはDの最も最近書かれたロケーションに格納されることを示す。キャッシュをアクセスするためのパイプライン位置に現在あるピクセルに対応するテクセル・データが対応するテクセル・データFIFOの最も最近書かれたロケーションにおけるデータに一致しない経路A、B、CおよびDのいずれについても、テクセル・キャッシュ・アクセス・コマンドが、キャッシュ・メモリ48からそのテクセル・データを読むために、対応するテクセル・キャッシュ・アクセス・コマンドFIFOに提供される。

【０１１１】

キャッシュ・アクセスが考慮されなければならないパイプライン位置に現在あるピクセルについて、インタリーブA-Dのいくつかに異なる結果が生じることがありえることが理解されなければならない。例えば、連続的なピクセルのための共通のテクセル・データが、

インタリーブAのために存在するが、インタリーブB-Dのためには存在しないことがありえる。そのような状況において、テクセル・データをキャッシュからアクセスするためパイプライン位置にあるとき、テクセル・データが、連続ピクセルの第２のもののためにインタリーブB-Dからの読みとられるが、その第２のピクセルのためのインタリーブAからのテクセル・データは、テクセル・データFIFO 214A0または214Alの一つである同じロケーションから読まれる。キャッシュをアクセスすることなく複数のピクセルのためにテクセルがテクセル・データFIFOからの再読取りされるとき、この方式がバンド幅節減を提供する。

【０１１２】

テクセル補間回路76は、53ビット・コマンドを境界チェッカー72から53ビットのバス218を通して受け取るテクセル補間回路コマンドFIFO 216を含む。テクセル補間回路コマンドFIFOは、どのテクセル・データFIFOロケーションが、各サイクル中に結果のテクセル・データを補間するのに使用すべきテクセル・データを含むかを補間回路に示す最高16のコマンドを格納することができる。補間回路コマンドは、補間のモードをも示し（すなわち、点サンプリング、2線形(bilinear)か、3線形(trilinear)か）、各テクセルが補間において重み付けされるべき態様を指定するSおよびT座標の勾配および小数値を含む。コマンドは、テクセル・データFIFO 214A0、A1、B0、B1、C0、C1、D0、またはD1のどれから4つ（2線形）または8つ（3線形）のテクセルのそれぞれが読まれるか、およびテクセル・データが新しいか古いかを示すデータを含む。テクセル・データは、その経路のいずれかのテクセル・データFIFOの最も最近書かれたロケーションに格納されたテクセル・データと異なるとき、新しい。新しいとき、キャッシュ読取りが必要である。テクセル・データは、いずれのテクセル・データFIFOのも最も最近書かれたロケーションに格納されたものと同じであるとき、古い。古いとき、キャッシュ読取りは、必要でない。テクセル・データが新しいとき、FIFO読取りポインタは、FIFO内の次のロケーションの方へ動かされなければならないが、テクセル・データが古いときは、同じデータが同じFIFOロケーションから読まれ、読取りポインタは、動かされる必要がない。

【０１１３】

図14および15を参照して説明する次の例が、さらに図13のテクセル・アクセス回路の動作を図示する。図14は、上部のMIPマップの複数のテクセルおよび下部の（サイズがより小さい）MIPマップの複数のテクセルを示す。テクセルは、図11に関して説明したラベル方式に従ってAn、Bn、CnとDnに（nは整数を表す）分類される。描画される7つのピクセルは、P0、P1、．．．P6と名称をつけられる。示されるように、描画されるピクセルは、直接MIPマップのテクセルにマップにしない。この例では、上部のマップからの4つのテクセルと下部のマップからの4つのテクセルがアクセスされ、各ピクセルについて補間されるように、3線形の補間が、実行される。ストライド（stride：一またぎ）の方向は、レンダリングの方向であって、ピクセルの数字で表した番号付けに対応する。

【０１１４】

図15が、キャッシュ・アクセス・コマンドFIFO 206A、テクセル・データFIFO A0（214A0）、テクセル・データFIFO A1(214Al）およびテクセル補間回路コマンドFIFO 216を図示する。他のテクセル・データ経路B、CおよびDの各々のためのFIFOが同じ態様で作用するのでテクセル・データA経路と関連するFIFOだけが、便宜上示される。各FIFOバッファが、データを書き込み、および読取るFIFO内の一つのロケーションをそれぞれ指す書込みポインタおよび読取りポインタを含む。ポインタは、この実施例において一度に一つのロケーションを動くことができる。

【０１１５】

ピクセルP0が、上部のマップにおけるテクセルA0、B0、C0、およびD0、ならびに下部のマップのテクセルA0、B0、C0およびD0にマップし、これらの8つのテクセルが、ピクセルP0のための結果のテクセル・データを生成するために補間される。ピクセルP0について、上部のマップのテクセルA0のアドレス（すなわち、uA0）が、そのアドレスでキャッシュから読まれるテクセル・データをテクセル・データFIFO 214A0に書込まなければならないこ

とを示すアドレスとともに、キャッシュ・アクセス・コマンドFIFO 206Aの第１のロケーションに書き込まれる。次に、キャッシュ・アクセス・コマンドFIFO 206Aの書込みポインタが、一つのロケーションだけ動かされ、そして、下部のマップテクセルA0のアドレス（すなわち、lA0）が、、そのアドレスにおけるキャッシュから読み出されるテクセル・データをテクセル・データFIFO 214A1に書込まなければならないことを示すアドレスとともにそのFIFOの次のロケーションに書かれる。このように、テクセル・データFIFO 0および1は、上で説明された理由のために交替される。キャッシュ・アクセス・コマンドFIFO 206B-Dが上部および下部マップにおけるテクセルB0、C0、およびD0に関して同様の態様で更新される。

【０１１６】

ピクセルP1について、アドレスuA1およびlA1にそれぞれ格納された、上部および下部のマップにおけるテクセルA1は、補間される必要がない。上部および下部のマップのテクセルA1は新しいテクセルであり、前のピクセルP0からのテクセルに対応しないから、それらは、キャッシュからアクセスされる。このようにこれらのテクセルのためのテクセル・アドレスは、それらのアドレスから読まれたテクセル・データがテクセル・データFIFO 214A0および214A1に格納されることになることをそれぞれ示す対応するアドレスとともにキャッシュ・アクセス・コマンドFIFO206Aの次の2つのロケーションに加えられる。図15はこの情報でアップデートしたあとのキャッシュ・アクセス・コマンドFIFO 206Aを表す。

【０１１７】

最初の2つのピクセルP0およびP1について共通のアドレスされたテクセルがないので、キャッシュ・メモリは、両方のためのテクセル・データを読み出すためにアクセスされる。第１のコマンドが、キャッシュ・アクセス・コマンドFIFO 206Aから読まれ、アドレスuA0でのテクセル・データがキャッシュ・メモリから読まれ、テクセル・データFIFO 214A0の第１のロケーションに書かれる。次いで、次のコマンドが、キャッシュ・アクセス・コマンドFIFOからの読取られ、そして、アドレスlA0でのテクセル・データがキャッシュからアクセスされて、テクセル・データFIFO 214A1の第１のロケーションに書かれる。次のコマンドが、次いでキャッシュ・アクセス・コマンドFIFOからの読取られ、そして、アドレスuA1でのテクセル・データがキャッシュからアクセスされて、テクセル・データFIFO 214A0の次のロケーションに書かれる。最後に、4番目のコマンドが、キャッシュ・アクセス・コマンドFIFOから読まれ、アドレスlA1でのテクセル・データがキャッシュからアクセスされて、テクセル・データFIFO 214A1の次のロケーションに書かれる。

【０１１８】

描画すべき次のピクセルP2のために、アドレスuA1およびlA1でのテクセルが補間される必要がある。これらのテクセルは以前に描画されたピクセルP1のためにアクセスされたので、それらは、テクセル・データFIFO 214A0および214A1の最も最近書かれたエントリにそれぞれ格納されている。このように、それらのテクセルのための新しいキャッシュ・アクセス・コマンドは、キャッシュ・アクセス・コマンドFIFO 206Aに提供されない。むしろピクセルP1のための結果のテクセル・データが補間されたあと、アドレスuA1およびlA1に格納されたテクセル・データは、キャッシュをアクセスすることなく、テクセル補間回路によってテクセル・データFIFO 214A0および214A1の最も最近読まれたロケーションから、それぞれアクセスされることができる。直接FIFOバッファからデータを読むことは、データをキャッシュ・メモリからアクセスするよりも時間がかからない。したがって、キャッシュ・アクセスを減らすFIFOバッファがシステムのバンド幅を増やす。

【０１１９】

上で示したように、インタリーブA-Dの各々に対応するテクセル・データFIFO 214は、別に制御されたFIFOゼロおよび1を含む。FIFOは、このように3線形の補間を実行するために効率的に分割される。上記の実施例からわかるように、連続的な読取りのために同じエントリを指すように、その読み取りポインタを維持することによって、テクセル・データFIFO 214の各々がその最も最近読まれたエントリへのアクセスを提供する。このように、連続的な読み取りサイクルの間、各インタリーブが2つのマップの読み取りの間で交互にな

るが、別々のFIFOが、一つのマップ内の連続的な読み取りを実行することができ、読取りポインタがFIFOへの連続的アクセスにおいて同じテクセル・データを指すことを可能にする。

【０１２０】

各ピクセルが、タイラー/境界チェッカー72によって作用されており、コマンドがキャッシュ・アクセス・コマンドFIFOに提供されているので、コマンドは、テクセル補間回路コマンドFIFO 216にも書かれる。例えば、アドレスuA0でのテクセルをアクセスするコマンドが、ピクセルP0のためにキャッシュ・アクセス・コマンドFIFOに提供されるとき、コマンドNew0がテクセル補間回路コマンドFIFO 216の第１のロケーションに提供される。コマンドNew0は、インタリーブAからの次のテクセル・データがキャッシュからアクセスされ、テクセル・データFIFO 214A0に提供され、ることをテクセル補間回路に示し、FIFOからテクセル・データを読むために、テクセル補間回路が、最も最近読み出したロケーションからFIFO読み取りポインタを一つのロケーションだけ動かさなければならないことを示す。

【０１２１】

テクセル・アドレスlA0を対応するキャッシュ・アクセス・コマンドFIFOに提供された次のコマンドのために、コマンドNewlがテクセル補間回路コマンドFIFOの次のロケーションに提供される。コマンドNewlは、インタリーブAからの次のテクセル・データも新しく、テクセル・データ補間回路214A1から読取られねばならないことをテクセル補間回路に示す。同様に、ピクセルP1に対応するテクセル・アドレスuA1およびlA1に関連するコマンドのために、コマンドNew0およびNewlは、テクセル補間回路コマンドFIFO 216の次の2つのロケーションに、それぞれ書かれる。

【０１２２】

ピクセルP2のために、アドレスuA1およびlA1でのテクセル・データが前のピクセルP1のためにFIFOに書かれたデータと同じであるから、テクセル補間回路コマンドFIFO 216の次の2つのロケーションに書かれたコマンドは、0ld0および0ld1であり、それぞれ、次のテクセル・データが、テクセル・データFIFO 214A0と214A1の最も最近読み出されたロケーションから再読取りされねばならないことをテクセル補間回路に示す。0ld0および0ld1コマンドは、FIFOから次のテクセル・データを読み出すために、テクセル補間回路が、最も最近読み出されたロケーションからFIFO読み取りポインタを動かしてはならないことを示す。

【０１２３】

図14は、次の3つのテーブルをリストする。すなわち、ピクセルの各々のために補間される必要があるテクセルを示す第１のテーブル、テクセル・データFIFO A0、B0、C0およびD0に格納される必要がある別々のテクセル・データ値をリストしている第２のテーブル、そして、テクセル・データFIFO A1、B1、C1、およびD1に格納される必要がある別々のテクセル・データ値をリストしている第３のテーブルである。ブランクスペースが、以前にキャッシュから読み出されキャッシュから再び読み出される必要のなく、FIFOからその代わりにアクセスされることができる共有されたテクセル・データを示す。このチャートが示すように、複数のピクセルのための結果のテクセル・データが補間されるとき、多数のキャッシュ・アクセスが、FIFO方式によって節約されることができ、結果としてシステムのバンド幅を増やすことができる。

【０１２４】

図16は、各インタリーブにおいて、ピクセルのために読み出されるテクセル・データが、最も最近描画されたピクセルについて読み出されたかどうか決定するために、テクスチャマッピング・チップによって使用される回路のブロック図である。この回路は、新しいデータがキャッシュから読み出されるようにするため、またはテクセル・データが、古くて、テクセル・データFIFOの中で一つから読み出されなければならないことを示すコマンドをテクセル補間回路コマンドFIFOに書き込むために、キャッシュ・アクセス・コマンドFIFOの一つに新しいコマンドを書くかどうかを決定するために使用される。図l6は、インタ

リーブAに対応する一つの回路だけを示す。しかし、インタリーブB、CおよびDのためにも同様の回路が用意されている。回路は、タイラー/境界チェッカーのオプティマイザー要素内に位置する。補間される各テクセルについてタイラー/境界チェッカーによって受け取られた補間されたS、T値から、オプティマイザーは、バス220A上にテクセル・アドレス（ブロック・タグおよびテクセル・アドレスを含む）を提供する。テクセル・データFIFO 214A0および214A1に割り当てられた最も最近処理されたテクセルのアドレスは、それぞれアドレスレジスター222A0および222A1に格納される。現在のテクセル・アドレスが、コンパレーター224A1と224A1によって、レジスタ222A0および222A1に格納されたテクセル・アドレスとそれぞれ比較される。

【０１２５】

このテクセル・アドレスがレジスタ222A0と222A1に格納したアドレスのいずれとも一致しないとき、テクセル・アドレスに対応するテクセル・データがキャッシュ・メモリからアクセスされた必要があり、そして、適当なコマンドがキャッシュ・アクセス・コマンドFIFOに書かれる。しかし、テクセル・アドレスがアドレスレジスター222A0または222A1に格納されたアドレスにマッチする時、テクセル・データは、そのアドレスに対応するテクセル・データをアクセスする直前にテクセル補間回路によって読み出されるロケーションにおいてテクセル・データFIFO 212A0または212A1にそれぞれ格納される。したがって、キャッシュ・アクセス・コマンドは、キャッシュ・アクセス・コマンドFIFOに書かれず、テクセル・データが古く、読み取りポインタを動かすことなく最も最近読み出されたFIFOロケーションからアクセスすべきことを示すコマンドが対応するテクセル補間回路コマンドFIFOに書かれる。

【０１２６】

E.テクスチャ・データのブロックの組織

図1は、8x8テクセルの中のベース・マップ100を含む一連の正方形のテクスチャMIPマップを示す。ベース・マップから、各連続したマップが、大きさにおいて最も小さいサイズ・マップ108にろ過される（すなわち一つのテクセルだけを含む）。最も小さいマップ108にゼロのマップ番号が割り当てられ、連続的に大きいマップのマップ番号がが、一つだけ増加させられるので、ベース・マップ100は、この例では3のマップ番号を持つ。マップ番号は、下で説明される態様で、テクスチャ・データの各ブロックのためにブロック・タグを決定するのに使用される。このマップ命番方式によると、正方形のテクスチャ・ベース・マップを仮定すると、10のマップ番号が1024x 1024のテクセルのマップに対応し、9のマップ番号が、512x512テクセルのマップを表し、8のマップ番号が、256x256テクセルのマップを表すという具合になる。テクスチャ・ベース・マップが正方形でないならば、マップ番号10が、1024テクセルより大きい寸法を持つマップに対応する。この議論は正方形のテクスチャ・ベース・マップを仮定しているが、長方形のマップも可能である。長方形であるならば、マップ番号は、マップの長い方の寸法のテクセルの数によって決定される。例えば、10のマップ番号を持っている長方形のマップは、1024のテクセルでより長い寸法を持つ。また、他のマップ命番方式を代わりに使用することができることを理解しなければならない。

【０１２７】

10のマップ番号を持つ1024x 1024のテクセル・マップは、各テクセルのロケーションをマップ内でユニークに識別するために10ビットのS座標S[9:0]および10ビットのT座標T[9:0]を必要とする。同様に、9のマップ番号を持っているマップは、各テクセルのロケーションを識別するために9ビットのSおよびT座標を必要とし、8のマップ番号を持っているマップは、各テクセルのロケーションを識別するために8ビットのSおよびT座標を要求するといった具合である。任意のピクセルに対応するMIPマップにおけるテクセルのロケーションをユニークに識別するSおよびT座標は、上で説明された態様で補間される。

【０１２８】

より詳細に下で説明するように、テクスチャ・データは、ホストコンピュータ15（図6）の中のメイン・メモリ17に、256x256テクセルのブロックで格納される。キャッシュ・ミ

スが生じるとき、キャッシュにおいて失敗したテクスチャ・データのブロックを識別する読み取りキャッシュ・タグが、TIMで読まれ、テクスチャ・データのそのブロックが、テクスチャマッピング・ボードのキャッシュ・メモリ48に次いでダウンロードされる。テクスチャ・データの64のブロックがキャッシュ・メモリに任意の一時点に格納されることができる。テクスチャ・データのこれらの64のブロックは、１つまたは複数のテクスチャの複数のMIPマップからのデータを含むことができる。各ブロックは、それをユニークに識別する関連するブロック・タグを持つ。 9以上のマップ番号を持っているMIPマップは、256x256を超えるテクセルを含み、したがって、複数のブロックに格納される。複数のブロックに格納される任意のマップの高次のS、T座標は、マップを格納するデータのブロックのブロック・タグに含まれる。

【０１２９】

例えば、9のマップ番号をもつMIPマップは、512のテクセルに等しい寸法を持ち、正方形であれば、512x512テクセルである。マップは、256x256テクセル（正方形のテクスチャ・マップを仮定する）の4つのブロックに分割される。したがって、それらのブロックの各々のためのブロック・タグは、マップ内のブロックのロケーションを識別する一つの高次のS座標ビットおよび一つの高次のT座標ビット（すなわち、S[8]およびT[8]）を含む。同様に、10のマップ番号を持とMIPマップは、1024x 1024テクセルの大きさで、16のブロックデータに分割される。したがって、それらのブロックの各々のためのブロック・タグは、マップ内のブロックのロケーションを識別する2つの高次S座標ビットおよび2つの高次T座標ビット（すなわち、S[9：8]およびT[9:8]）を含む。

【０１３０】

下に説明するように、3線形の補間の間、システムの帯域幅を減らすために、テクスチャMIPマップが小分けされ、隣接のMIPマップの同じ部分が、反対のSDRAMバンクに格納されるようにメモリに格納される。さらに、キャッシュ・メモリ内のメモリ空間の効率的利用のために、256x256テクセルより小さい複数のマップは、キャッシュ・メモリの一つのブロックに格納されることができる。

【０１３１】

図17は、表面の画像を含む特定のテクスチャのための一組のテクスチャMIPマップを示す。図17に見られるように、一つのテクスチャのためのMIPマップのシリーズの各MIPマップが、正方形のテクスチャ・マップについては等しいサイズの4つの四分区間に分割される。図17に示した例において、ベース・マップは、9のマップ番号を持ち、四分区間9Q1（画像Lを含む）、9Q2（画像Aを含む）、9Q3（画像9を含む）、および9Q4（画像5を含む）に分割される。同様に、マップ番号8が、それぞれ画像L、A、9および5を含む四分区間SQI、8Q2、8Q3、8Q4、に分割される。同様に、マップ番号7が、それぞれ画像L、A、9および5を含む四分区間7Q1、7Q2、7Qに分割される。同様に、より小さいマップが四分区間に小分けされる。

【０１３２】

各MIPマップの2つの四分区間が、キャッシュを形づくるSDRAMの一方のバンクに格納され、他方の2つの四分区間が反対のバンクに格納される。テクスチャ・データ配分計画によると、8以上の番号（大きさにおいて256x256テクセル以上）のベース・マップを持つテクスチャのために、そのテクスチャのすべてのMIPマップのすべての四分区間のためのメモリ空間のブロック内の記憶場所が、事前に決められている。例えば、マップ番号9の四分区間9Qlおよび9Q4が、キャッシュ・バンク1内の別々のブロックに格納され、そして、図18に示されるように、四分区間9Q2および9Q3は、キャッシュ・バンク0の別々のブロック内に格納される。隣接のMIPマップの対応する四分区間は、反対側のバンク内のブロックに格納される。このように、この例では、それぞれ四分区間9Qlと9Q4のボックスフィルタされたテクスチャ・データを含む四分区間8Qlと8Q4が、キャッシュ・バンク0内の同じブロックに格納される。同様にそれぞれ9Q2と9Q3の四分区間のボックスフィルタされたテクスチャ・データを含む四分区間8Q2および8Q3が、キャッシュ・バンク1内の同じブロックに格納される。図18は、図17に関してスケールを合わせていない。図17のマップ四分区間が

、図18のものと同じサイズであることを理解しなければならない。

【０１３３】

マップのそれぞれのサイズのため、マップ番号9の各四分区間が256x256テクセルの完全なブロックを占めるが、マップ番号8の四分区間は、それぞれブロックの1/4だけを占める。したがって、四分区間8Q2と8Q3は、一緒に同じブロックの1/2を占め、そして、四分区間8Q1と8Q4は、反対側のバンク内の別のブロックの1/2を占める。キャッシュ・メモリ空間を効率的に割り当てるために、それらのブロックの各々内の空いているロケーションは、マップ番号7以下のマップの適当な四分区間によって占められる。したがって、ゼロから8までの番号を持つマップのすべてが、2つのブロックを占め、2つのブロックのそれぞれが別のバンクにある。

【０１３４】

8以下のマップ番号を持っているマップのための四分区間のケーション（8以上のマップ番号を持つベース・マップが与えられたとして）は、図18に示された態様であらかじめ規定されている。図に示されるように、右上四分区間8Q2と左下四分区間8Q3が、同じ物理的関係を維持し、それぞれ、第１のブロックの右上と左下の四分区間を占め、そして左上四分区間8Q1および右下四分区間8Q4も、同じ物理的関係を維持し、それぞれ第１のブロックと異なるバンクにある第2のブロックの左上および右下の四分区間を占める。また、四分区間7Q1と7Q4が、同じ物理的関係を維持して、それぞれ第１のブロックの左上の四分区間を占め、そして、四分区間7Q2と7Q3が、同じ物理的関係を維持し、それぞれ第２のブロックの右上四分区間を占める。

【０１３５】

3線形補間の間に、ピクセルが一つのMIPマップの4つのテクセルと隣接のMIPマップの4つのテクセルの間にあるテクスチャ・マップの位置にマップするならば、すべての8つのテクセルがキャッシュからアクセスされる。両方のMIPマップからアクセスされたテクセルは、共通のテクスチャ・データを含み、小さいマップからのデータは大きいマップからのデータをフィルタしたバージョンでる。上で説明したように、対象物のピクセルが描画されているとき、隣接のピクセルがそのテクスチャのための同じ2つのMIPマップに頻繁にマップし、キャッシュへの読み取りは、2つのマップを格納するキャッシュ・ブロックの間で連続的に切り換えられることを要求する。共通データを隣接のMIPマップからキャッシュSDRAMチップの異なるバンクに格納することによって、キャッシュ読み取りが、連続的な読み取りサイクルの間2つのMIPマップの間で切り換わるとき、再ページ割り付けのペナルティーが生じない。これは、3線形の補間の効率的な実施の役にたつ。

【０１３６】

以上のことからわかるように、テクスチャが、8以上のマップ番号を持つベース・マップを含むとき、そのテクスチャのためのブロックの間でのMIPマップの割当ては、説明された実施例に従って事前に決められている。上で説明し、図18に示すようにマップ番号8を持っているマップの2つの四分区間が一方のバンクの第１のブロックの事前に決められたロケーションを占め、そして、他の2つの四分区間が、反対側のバンクの別のブロック内の事前に決められたロケーションを占めるので、このようになる。しかし、7以下のマップ番号でベース・マップを持っているテクスチャについては、メモリの2つのブロック内の複数のロケーション（各バンクに一つのブロック）がマップを格納するために利用可能であり、TIMによって選ばれる。複数のマップの部分がデータの一つのブロックを共有するとき、サブテクスチャ識別子（ID）が、下で説明された態様の共有されたブロック内の各マップのロケーションを識別するために割り当てられる。

【０１３７】

図17のMIPマップのシリーズの組織に加えて、図18も、異なるテクスチャ（すなわち、チェス盤パターン）からのMIPマップの第2のシリーズが、メモリ・ブロックの間で割り当てられる態様を示す。この第２のテクスチャのMIPマップは、小分けされて、データの別々のブロックに第１のテクスチャと同じ態様で格納される。図18の組織が別々のブロックに組織されつつあるものとして異なるテクスチャのMIPマップを示すけれども、2つの異なる

テクスチャからのテクスチャ・データが、同じブロック内に格納されることができることが理解されなければならない。上で示したように、一つの実施例において、キャッシュ・メモリが、テクスチャマッピング・データの64のブロックまで格納することができ、各ブロックが256x256テクセルを含む。キャッシュ・メモリが、2つのバンクに分割され、ブロック0-31がバンクゼロにあり、ブロック32-63がバンク1にある。キャッシュ・ディレクトリは、キャッシュのブロックに対応する最高64のブロック・タグ・エントリを含む。キャッシュ・ディレクトリ内の各ブロック・タグの物理的なロケーションは、キャッシュ・メモリ内のテクスチャ・データの対応するブロックの物理的なロケーションを識別する。ブロックインデックスは、ブロックのロケーションを示すブロック・タグから生成される。キャッシュの任意のテクセルのためのキャッシュ・アドレスは、そのブロックのブロックインデックスおよびブロック内のテクセル・アドレスから形成される。テクセル・アドレスは、テクセルのために下位の補間されたS、T座標を含み、そして、また下に説明するサブテクスチャ IDからのビットを含みうる。

【０１３８】

図19は、四分区間に小分けされる9のマップ番号を持っているテクスチャMIPマップの例を示す。MIPマップは、512x512テクセルの大きさであり、したがって各四分区間は256x256テクセルの大きさであり、メモリの一つのブロックに対応する。この発明の一つの実施例に従って、TIMによって単純な方式が、MIPマップの各四分区間が、割り当てられるべきキャッシュのバンクを決定するために実行される。下で説明するように、各MIPマップ四分区間について、四分区間のためのSおよびT座標の最上位ビットの値に対する論理的排他的OR演算の結果が、その四分区間が割り当てられるキャッシュのSDRAMバンクを決める。

【０１３９】

512x5l2テクセルのマップについて、9つのS座標ビットS[8：0]および9つのT座標ビットT[8:0]が、マップ内の各テクセルのロケーションを指定する。四分区間境界は、SおよびT次元の両方においてマップの中途の点で確立され、最上位のSおよびT座標ビットS[8]およびT[8]で表わされる。したがって、9のマップ番号を持っているMIPマップの4つの四分区間の各々についてキャッシュ・バンクを決定するために、排他的ＯＲ演算が、各四分区間についてその対応する最上位のSおよびT座標ビットS[8]のおよびT[8]の値に対して実行される。同様に、10のマップ番号を持っているMIPマップのために、各四分区間のためのキャッシュ・バンクが、その最上位のSおよびT座標ビットS[9]およびT[9]の対応する値に対する排他的ＯＲ演算によって決定される。奇数のマップ番号を持っているMIPマップのために、隣接のマップからの共通のデータが異なるバンクに格納されるよう、排他的ＯＲ演算の結果が反転される。

【０１４０】

図19の例において、ブロック1ないしブロック4がそれぞれ512x5l2テクセル・マップの左上四分区間、右上四分区間、左下四分区間および右下四分区間に対応する。ブロックlないしブロック4についてビットS[8]、T[8]がそれぞれ[0,0]、 [1,0]、 [0,1]、および[1,1]に等しい。したがって、ブロックlについてXOR演算S[8] XOR T[8]の結果は、ゼロに等しい。マップが奇数のマップ番号（すなわち、9）を持つので、この結果の反転（1に等しい）が、ブロック1が、キャッシュのバンク1に格納されることを示すのに使用される。ブロック2については、XOR演算動作S[8]XOR T[8]の結果の反転がゼロに等しく、ブロック2がキャッシュのバンク0に格納されることを示す。ブロック3およびブロック4のそれぞれについては、XOR演算S[8] XOR T[8]の結果の反転がそれぞれ１およびゼロに等しく、ブロック3が、バンク1に格納され、ブロック4がバンク0の格納されることを示す。

【０１４１】

図19の例に示すように、同じテクスチャについて10のマップ番号を持っているマップについては、マップが1024xl024テクセルの大きさであるから、マップはそれぞれ256×256のテクセルの16のブロックに分割される。各ブロックのために、S [9] XOR T[9]の結果が、その特定のブロックのバンク番号を示す。10のマップ番号を持つマップの各ブロックに対するXOR演算の結果は、それらが、マップ番号9を持っているマップに隣接するものに対す

るものであるから、反転されず、2つのマップの中の対応する四分区間が異なるキャッシュ・バンクに格納される。

【０１４２】

マップのサイズに依存して、マップを表すテクスチャ・データのブロックのためのブロック・タグは、特定のMIPマップ内のブロックのロケーションを示す少なくとも一つの高次S座標ビットおよび一つの高次T座標ビットを含むことができる。9のマップ番号を持っている512x512テクセルMIPマップについては、MIPマップ内の各ブロックのロケーションを示すために、一つのS座標ビットと一つのT座標ビットだけがブロック・タグの中で必要である。10のマップ番号を持ちデータの16のブロックを含む1024x 1024のテクセルMIPマップについては、MIPマップ内の各ブロックのロケーションを示すために、2つのS座標ビットと2つのT座標ビットがブロック・タグの中で必要である。8以下のマップ番号を持っているマップについては、ブロック・タグの中でSおよびTビットは、必要ない。ホストコンピュータのメイン・メモリからキャッシュ・メモリにテクスチャMIPマップ・データをダウンロードしているとき、TIMが、データの各ブロックを書かなければならない特定のバンクを決定するために上に説明された排他的ＯＲ方式を使用して、ブロック・タグの上位レベルのSおよびT座標ビットをデコードする。

【０１４３】

不使用のメモリ空間が最小になるようにテクスチャ・データを割り当てるために、データの各ブロックを、64x64テクセルの16のサブ・ブロックに、さらに小分けすることができる。テクスチャ・データの各サブブロックは、ブロック内の特定のサブブロックのロケーションを識別するサブ・テクスチャIDを含む。サブ・テクスチャIDは、2つのSビット、S[1:0] および2つのTビット、T[1: 0]を含む。１つまたは複数のテクスチャの１つまたは複数のMIPマップからの複数のサブ・テクスチャは、一つのブロックに格納されることができる。

【０１４４】

図20は、それぞれバンク0およびキャッシュの一つに割り当てられ、それぞれ64x64テクセルの大きさの16のサブ・テクスチャに小分けされたブロック1とブロック2を図示する。各ブロックのサブ・テクスチャが、ST0-ST15の名称をつけられ、2つのS座標ビットと2つのT座標ビットを含むサブ・テクスチャIDによって識別される。サブ・テクスチャは、上述のメモリ配分計画と一致するよう、2つのキャッシュ・バンク内のミラー・ロケーション以外に一貫した名称を持つ。64x64テクセルのサブ・テクスチャのサイズが、典型的に選ばれて、変えられることができる。例えば、より小さいサブ・テクスチャは、より多くのテクスチャを同じブロック内にパックするのを可能にする。サブ・テクスチャのサイズが小さくなるにつれて、サブ・テクスチャIDがより多くのビットを含む必要があることが理解されなければならない。

【０１４５】

レンダリングの間、補間すべきテクセルの各ストリームについて、テクスチャID、サブ・テクスチャIDおよびそれらのテクセルと関連するテクスチャのベース・マップのサイズを表す８ビットのワードが、三次元のパイプラインを通して、一時的にそのデータを20ビットのレジスタ（図示しない）に格納するタイラー/境界チェッカーに提供される。補間されるテクセルが異なるサブ・テクスチャIDまたはテクスチャIDを持つとき、新しいデータが、タイラー/境界チェッカーに提供され、レジスタに格納される。サブ・テクスチャIDは、下に説明するように、テクセル・アドレスの一部として使用されることがある。

【０１４６】

テクセル・アドレスがサブ・テクスチャIDのS、T座標ビットを含むかどうかは、アドレスされつつあるマップのサイズおよびそのテクスチャのベース・マップのサイズに依存する。アドレスされつつあるマップが7以下のマップ・サイズを持ち、また、その対応するベース・マップのサイズが7以下であるならば、テクセル・アドレスのある上位のアドレス・ビットが、下で説明されるように、ブロック内のサブ・テクスチャのロケーションをアドレスするためにサブ・テクスチャIDからのビットを含む。上で説明したように、ベース

・マップが8以上のマップ番号を持つとき、データのそれらのそれぞれのブロック内のテクスチャのMIPマップ四分区間のすべてロケーションが、事前に決められる。したがって、8以上のマップ番号を持つテクスチャのマップの一つからテクセルがアクセスされるとき、それらの事前に決められたロケーションが、知られていて、サブ・テクスチャIDを使用することなく各四分区間のテクセル・アドレスの上位ビットを生成するために使用される。しかし、テクスチャのベース・マップが7以下のマップ番号を持つとき、MIPマップ四分区間のロケーションが、事前に決められておらず、サブ・テクスチャIDビットが、サブ・テクスチャのロケーションを決定するためにテクセル・アドレスの上位ビットとして使用される。

【０１４７】

上で述べたように、そのテクスチャからのベース・マップが十分に小さい限り、異なるテクスチャからの複数のマップが、データの一つのブロックの異なるサブ・テクスチャ内に格納されることができる。これが生じるとき、各マップのためのテクスチャ・アドレスがサブ・テクスチャIDビットを含む。例えば4つの異なるテクスチャからの7のマップ番号を持つ4つの異なるマップが、ブロック内の異なるサブ・テクスチャの間で割り当てられ、そして各テクスチャのベース・マップのマップ番号が7であるならば、サブ・テクスチャIDからの一つのS座標ビットと一つのT座標ビットが、テクスチャを見分けるためのテクセル・アドレスの一部となる。タイラー/境界チェッカーがテクセル・アドレスを計算するルーチンを図22を参照して説明する。

【０１４８】

図の実施例で、テクスチャMIPマップ・データは、一度に一つのブロックずつダウンロードされる。しかし、代わりに、サブ・テクスチャがメイン・メモリからダウンロードされるようにサブ・テクスチャIDをブロック・タグに含めることができる。また、この実施例の中で説明されたブロックとサブ・テクスチャのサイズは、単に例としてのものであり、任意のアプリケーションに合せて変えることができる。発明の一つの実施例の中で、TIMは、図35を参照して下に説明するように、テクスチャ・データを割り当てることができる。

【０１４９】

F.キャッシュ・ブロック・タグとブロックインデックス

キャッシュ・ディレクトリが、その64のエントリの各々のためのブロック・タグを含み、各エントリのための対応するブロックインデックスを識別する。ブロックインデックスが、テクスチャ・データの対応するブロックの始めが格納されているキャッシュにおける物理的なロケーションを識別する。ブロック・タグは、テクスチャ・データの各ブロックを図21に示された態様でユニークに識別する23ビットの識別子である。

【０１５０】

テクスチャ・データの任意のテクセルをユニークに識別するために、それが対応するテクスチャが識別されなければならない。一つの実施例において、テクスチャマッピング・ハードウェアが、テクスチャをユニークに識別する８ビット・テクスチャIDを支援する。さらに、同じブロック内に格納された異なるテクスチャからのテクスチャ・データのために、付加的な4ビットのサブ・テクスチャIDが、テクスチャを識別するためにハードウェアによって支援される。このように、テクスチャマッピング・ハードウェアが、任意の一つの時にアクティブであることができる2^{12}すなわち4,096のユニークなテクスチャを支援する。

【０１５１】

上で示したように、各テクスチャは、一連のMIPマップによって表される。上で示したように、MIPマップの各々は、MIPマップのシリーズにおけるその位置を示すマップ番号を備えている。このように、データの任意のテクセルは、テクスチャID、サブ・テクスチャIDおよびそのテクスチャのベース・マップのサイズだけによって識別されるのではなく、それが対応するMIPマップのマップ番号によっても識別される。最後に、テクセルはMIPマップ内でそのSおよびT座標（すなわち、その補間されたS,T値）によってユニークに識別さ

れる。

【０１５２】

サブ・テクスチャIDおよびテクスチャ・マップ・ベース・サイズ以外に、テクセルをユニークに識別する上に説明したパラメーターが、23ビットのブロック・タグを生成するために使用される。マップ番号ならびにSおよびT座標に関し、SおよびT座標を生成するために使用されたハードウェアは、15ビットに限られている。したがって、この実施例について、ハードウェアによって支援された最も大きいテクスチャ・マップが、15ビットの Sフィールド[14：0]および15ビットの Tフィールド[14：0]を持ち、大きさにおいて32Kx 32Kテクセルである最大テクスチャ・マップになる。上で示したように、テクセル・データの各ブロックは、256 x 256のテクセルを含む。このように、下位のSおよびTビット（すなわち、T[7:0]とS[7:0]）が、テクセル・データのブロック内の特定のテクセルを識別するために使用される。高次のSおよびTビットだけ（T[14:8]およびS[14:8]）が、テクセル・データの特定のブロックを識別するためにブロック・タグにおいて使用される。

【０１５３】

上で述べたように、各MIPマップが、その対応するテクスチャのためのマップのシリーズ内でそれをユニークに識別するマップ番号を割り当てられる。一つのテクスチャのためのマップのシリーズにおけるMIPマップの数と関係なく、シリーズの中で最も小さいMIPマップ（すなわち、一テクセルの大きさ）にマップ番号0が割り当てられる。32KX32Kテクスチャのための MIPマップの最も大きいシリーズが16のMIPマップを含むから、支援される最も大きいマップ番号は、15である。

【０１５４】

ブロック・タグが扇形に広がる態様が図21の表に示される。ブロック・タグ[22：15]の高次の8つのビットは、テクスチャ・データのブロックによって表されたテクスチャのテクスチャIDに対応する。ブロック・タグ[13：00]の下位ビットは、高次のTおよびS座標、T[14：08]およびS[14：08]に対応する。ブロック・タグ[14]は、高次T座標フィールドの中の値と関連し、マップ番号の識別を可能にするマップ・ビットに対応する。最大32Kx32Kより小さいマップは、全面的なSおよびTアドレス・フィールドを使わないので、マップが小さいほど、不使用の高次のSおよびTアドレス・ビットが多くなることを理解しなければならない。図21に示すように、8より大きいマップ番号を持っているマップについて、最下位の不使用のT座標ビットに対応するブロック・タグ・ビットが論理的「0」にセットされ、そして、残りの高次T座標ビットに対応するブロック・タグ・ビットおよびマップ・ビットが、論理的「1」にセットされる。T座標ビットのすべてを使用するマップ番号15について、マップ・ビットが、論理的「0」にセットされる。マップ・ビットおよび高次T座標ビット[14:8]に対応するブロック・タグ・ビット[14:07]を読みとることによって、左から右に読みとって遭遇した第１の論理的な「0」の位置が、ブロック・タグによって表されるマップ番号を示す。論理的1が、ブロック・タグ・ビット[14：08]のすべてに含まれるならば、マップ番号8以下が表される。

【０１５５】

先に述べたように8以下のマップ番号を持っている特定のテクスチャのすべてのマップが、データの2つのブロック内に格納され、各ブロックがキャッシュの異なるバンク内に位置する。8以下のマップ番号を持つマップの各々の2つの四分区間、または半分が2つのブロックの各々内に格納される。ブロック・タグ・ビット[07]は、2つのブロックのどちらに8以下のマップ番号を持っているマップの各半分部分が格納されるかを表す。このように8のマップ番号を持っているマップの各々のためにブロック・タグ・ビット[07]は、そのマップ（バンク0のブロックに格納されている）の半分（2つの四分区間）のために「0」の値を持ち、そのマップ（バンク1のブロックに格納されている）の他の半分（2つの四分区間）のために「1」の値を持つ。8以下のマップ番号を持つ特定のテクスチャからのマップのすべてが2つのブロック内に格納されるので、それらの2つのブロックを識別するために一つのブロック・タグ・ビットだけが使用されることが理解されねばならない。番号8以下を持っているマップの各々のための特定のマップ番号は、したがって、ブロック・

タグフィールドの一部として格納されない。

【０１５６】

8以下のマップ番号を持つマップのそれぞれの各四分区間についてのブロック・タグ・ビット[07]の値が、四分区間が格納されるべきバンクを決定するための方式に基づいて計算される。この方式は、偶数番号のマップの各四分区間のMSBの値の論理的排他的ＯＲ演算および奇数番号の各四分区間についての演算の反転を含む。

【０１５７】

図21に示すように、高次Sアドレス・ビットに対応するブロック・タグ・ビット[6：0]が、Sアドレス・ビットが不使用のとき小さいマップのための論理的「0」にセットされるので、これらのビットのいずれかが、マップ番号に関連して論理的「1」として検出され、それらが論理的「0」に等しいべきであることが示されるならば、キャッシュ・ディレクトリ・エントリに有効なデータがないことを示すのに使用することができる。

【０１５８】

上で示したように、各MIPマップ四分区間について、四分区間の最上位のSおよびT座標の値の論理的な排他的論理和（XOR）演算の結果が四分区間が割り当てられるキャッシュのSDRAMバンクを示す。偶数マップ番号を持っているマップについて、バンク番号が、このXOR演算と等しく、奇数のマップ番号を持っているマップについてバンク番号は、、XOR演算の論理的反転に等しい。これが、図21の表の右の列に示され、記号「Λ」がXOR演算を、記号「！」が論理的反転を示す。9以上のマップ番号を持っているマップについて、各四分区間がデータの少なくとも一つの全ブロックを消費し、各ブロックが、図21の表の最後の列に示されるXOR演算によって示されるバンクに格納される。

【０１５９】

8以下のマップ番号を持っているマップについては、それらのマップのすべてが、各バンクに一つのブロックで、データの2つのブロックを占める。図21の表の最後の2つの行が、8以下のマップ番号を持つ任意のマップの異なる半分（2つの四分区間）に対応する。ブロック・タグ・ビット[07]は、バンク・ゼロ・ブロックまたはバンク１ブロックのどちらに２分の１マップが格納されるかを表す。そのビット[07]の値が、説明したXOR演算に基づいて計算される。たとえば、マップ番号8のマップでは、マップの各四分区間について、ブロック・タグ・ビット[07]は、S[7] XOR T[7]に等しい。マップ番号7を持っているマップの各四分区間では、ブロック・タグ・ビット[07]は、S[6] XOR T[6]の反転に等しい。同様に、ブロック・タグ・ビット[07]が、より小さいマップの各四分区間について計算され、奇数の番号をつけられたマップについてだけXOR演算の結果を反転する。以上のことからわかる

各マップ（8以下のマップ番号持つ）の2つの四分区間が同じブロックに格納されるので、各マップのそれらの2つの四分区間は、同じブロック・タグ・ビット[07]を持つ。ヒットが、補間されたS、T座標（アクセスすべきテクセルをアドレスする）およびキャッシュ・ディレクトリにおける23ビットのブロック・タグの一つの間に生じるとき、キャッシュ・ディレクトリが、そのテクセルが格納されるキャッシュ・ブロックを含むキャッシュ・メモリ内の物理的なロケーションを識別するブロックインデックスを生成する。キャッシュは、任意の一時点でテクセル・データの64のブロックを格納する。したがって、キャッシュ・メモリのブロック・アドレスを識別するためには、2^6=64なので、6ビットのブロックインデックスが用意され、先に述べたようにキャッシュへの高次アドレス・ビットの役目を果たす。

【０１６０】

テクセル・アドレスは、256x256テクセルのブロック内でアクセスされるべきテクセルのロケーションを示す、ビットS[7:0]およびT[7:0]を含む16ビットのワードである。テクセル・アドレスは、補間されたS、T座標、アクセスされるべきマップのマップ番号、テクスチャおよびサブ・テクスチャのID、およびテクスチャのベース・マップ・サイズ、から図22を参照して下で説明するルーチンに従って計算される。上で示したように、テクセル・アドレスのLSB SビットおよびLSB Tビットが、テクセルが格納される適当なインタリーブ

を決定するためにデコードされる。テクセル・アドレスの残りの14ビットが、ブロックインデックスの6ビットと関連して、キャッシュのデコードされたインタリーブ内のSDRAM対に提供されるキャッシュ・アドレスとして作用する（ブロックインデックスの6ビットがキャッシュ・アドレスの6つのMSBである）。

【０１６１】

G.テクセル・アドレス計算

レンダリングの間、タイラー/境界チェッカー要素72が、パラメーター補間回路64から、アクセスされるべきテクセルの補間されたS、T値およびテクセルがアクセスされるべきマップのマップ番号を表す4ビットのワードを受け取る。パラメーター補間回路から受け取られた補間されたSおよびT座標の値の各々は、16の整数ビットおよび8つの小数ビットを含む。マップ番号を表わす4ビットのワードは、マップ番号0（一テクセルの大きさ）からマップ番号15（32kx32kテクセルの大きさ）にわたるマップを含み、先に述べたように、勾配から計算される。補間されたS、T値とキャッシュ・ディレクトリのブロック・タグ・エントリとの比較が、次いで実行される。ブロック・タグの一つでヒットが生じるならば、ブロックインデックスが生成される。キャッシュ・ディレクトリ・サーチが実行されているのと同じ時に、テクセル・アドレスが、図22のフローチャートを参照して下で説明されるルーチンに従って計算される。

【０１６２】

テクセル・アドレスは、テクスチャID、サブ・テクスチャID、マップ番号、ベース・マップ番号およびテクセルの補間されたS、T座標に基づいてタイラー/境界チェッカーによって計算される。タイラー境界チェッカーは、この情報のすべてを持つ。アクセスされる各ユニークなテクセルについて、タイラー/境界チェッカーが、補間されたS、T座標（SおよびTのそれぞれについて16の整数ビットおよび8つの小数ビットを含む)、およびテクセルがアクセスされることになっているマップ番号を表わす4ビット・ワードを、パラメーター補間回路から受け取る。さらに、三次元のパイプライン（パラメータ補間器を通ってくる）を通して、８ビットのテクスチャID、4ビットのサブ・テクスチャID、およびそのテクスチャに対するベース・マップのサイズを表す８ビットのワードを含むコマンドが受け取られる。ベース・マップのサイズを表す８ビットのワードは、マップ命番方式に対応し、それぞれベース・マップのS方向とT方向のサイズを規定する4つのSビットおよび4つのTビットを含む。例えば、4ビットのSおよびTワードのそれぞれは、ゼロ（一つのテクセルの大きさに対応する）から15（32 kのテクセルの大きさに対応する）までの値の範囲を持つことができる。テクスチャIDを含むデータの20ビット、サブ・テクスチャIDおよびベース・マップ番号が、キャッシュからアクセスされる後続のテクセルのための新しい異なるデータで置き換えられるまで、タイラー/境界チェッカー内の20ビットのレジスタ（図示しない）に一時的に格納される。この情報で、タイラー/境界チェッカーは、各テクセルのためのテクセル・アドレスを計算する。

【０１６３】

上で説明したように、8以上のマップ番号のベース・マップ（256x256テクセル以上のベース・マップに対応する）を持つテクスチャについては、そのテクスチャ内の各マップの四分区間が、テクスチャ・データおよびキャッシュ・メモリ・バンクのブロック内で事前に決められたロケーションを持つ。このように、そのようなテクスチャの任意のテクセルのためのテクセル・アドレスの各ビットが、既知の事前に決められた配分計画に従って計算される。7以下のマップ番号のベース・マップ（128x 128テクセル以下のベース・マップに対応する）を持っているテクスチャについては、しかし、そのテクスチャのマップの各四分区間のためにいくつかの別個の記憶場所が利用可能であり、したがって、テクセル・アドレスのある上位レベルのビットが、サブ・テクスチャIDのいくつかまたはすべてのビット（またはそれらのビットの反転）を含む。

【０１６４】

タイラー/境界チェッカーによってテクセル・アドレスを計算するために実行されたルーチンは、図22のフローチャートによって示される。ルーチンは、完了するのに一つのサイ

クルを必要とする。ルーチンは、テクスチャマッピング・チップの境界チェッカー部分を形づくる一組の論理ゲート（図示しない）によって実行される。図22のフローチャートによって示されるルーチンを実行するためにどのように論理ゲートを実行するかは、当業者に自明である。たとえば、ルーチンが、ベリログ（Verilog）のようなソフトウェア・シミュレーション言語で書くことができ、それからシノプシス（Synopsys、商標）のような合成ツールによってロジック・ゲート回路に変換される。そのような合成ツールは、汎用プロセッサ上で作用する。ルーチンは、代わりにソフトウェアで書かれ、コンピュータ・プロセッサによって実行されることができる。

【０１６５】

ルーチンが、ステップ250で始まり、テクセル・アドレス・ビットS[7:0] T[7:0]が、補間されたS、T座標ビットS[7:0]、T[7:0]に等しく予めセットされる。テクセル・アドレスのビットの各々は、後にルーチンでリセットされなければ、このステップで予めセットされる（対応するSまたはT座標に等しく）値のまま残る。次いで、ルーチンがステップ252へさらに進み、補間中のテクセルが格納される特定のマップが8以上のマップ番号を持つかどうか判断する。そうならば、ルーチンはそのようなテクセルについて終わり、テクセル・アドレスのビット値が補間されたS、T座標に等しくプリセットされたまま残る。

【０１６６】

マップ番号が8以上でないならば、ルーチンがステップ254へ進み、テクセルがバンク番号１バンク番号0に格納されているかどうかが決定される。先に述べたように、テクセルがバンク番号１またはバンク番号ゼロに格納されているかどうかが、ブロック・タグ・ビット[07]の値を調べることによって知られる。

テクセルがバンク番号１に格納されているならば、ルーチンがステップ256に進み、そこではあるテクセル・アドレス・ビットがそれらのプリセット値からリセットされる。マップ番号1ないし4のマップについて、テクセル・アドレス・ビットS[4]=1で、マップ番号1および2のマップについては、テクセル・アドレス・ビットS[2]=1である。テクセルが、バンク0に格納されるならば、ルーチンがステップ258に進み、そこではマップ番号0ないし5のマップについて、テクセル・アドレス・ビットS[5]=1で、マップ番号0ないし3のマップについて、テクセル・アドレス・ビットS[3]=1、マップ番号0および1のマップについては、テクセル・アドレス・ビットS[1]=1である。

【０１６７】

ステップ256および258のいずれからも、ルーチンがステップ260に進み、そこでベース・マップが8以上のマップ番号を持つかどうかが判断される。そうであるならば、ルーチンがステップ262に進み、そこで、テクセルがバンク0またはバンク1内に格納されているかどうかが判断される。テクセルがバンク1に格納されているならば、ルーチンがステップ264に進み、そこでは、マップ番号7を持っているマップについて、テクセル・アドレス・ビットS[7]=0であり、0ないし6のマップ番号を持っているマップについて、テクセル・アドレス・ビットS[7:6]=0:1である。ルーチンは、そのようなテクセルについて終えられる。

【０１６８】

バンク0に格納されたテクセルについて、ルーチンがステップ266に進み、そこでは、7のマップ番号を持っているマップについて、テクセル・アドレス・ビットS[7]=1で、0ないし6のマップ番号を持っているマップについて、テクセル・アドレス・ビットS[7:6]=1:0である。ルーチンは、そのようなテクセルについて次いで終了する。

【０１６９】

ベース・マップが8以上のマップ番号を持たないならば、ルーチンがステップ268に進み、そこで、ベース・マップが7に等しいマップ番号を持つかどうかが判断される。そうであるならば、ルーチンがステップ270へ進み、そこで、テクセルがバンク0または1に格納されているどうかが判断される。テクセルがバンク1に格納されていれば、ルーチンは、次いでステップ272に進み、そこでは、マップ番号7について、テクセル・アドレス・ビットS[7]がサブテクスチャIDビットS[1]の反転に等しく、テクセル・アドレス・ビットT[7]が

サブテクスチャIDビットT[1]に等しく、そしてマップ番号0ないし6を持っているマップについて、テクセル・アドレス・ビットS[7:6]がサブテクスチャIDビットS[1]および1の反転に等しく、テクセル・アドレス・ビットT[7]が、サブテクスチャIDビットT[1]に等しい。ルーチンは、そのようなテクセルについて終了する。テクセルがバンク0に格納されているならば、ルーチンがステップ274に進み、そこでは、マップ番号7を持っているマップについて、テクセル・アドレス・ビットS[7]が、サブテクスチャIDビットS [1]に等しく、テクセル・アドレス・ビットT[7]がサブテクスチャIDビットT[1]に等しく、マップ番号0ないし6を持っているマップについて、テクセル・アドレス・ビットS [7:6]がサブテクスチャIDビット[1]および0にそれぞれ等しく、そして、テクセル・アドレス・ビットT[7]がサブテクスチャIDビットT[1]に等しい。ルーチンは、そのようなテクセルについて終了する。

【０１７０】

テクスチャのベース・マップが8以上のマップ番号を持たず（ステップ260で決定する）、7に等しくもない（ステップ268で決定する）ならば、もちろんテクスチャのベース・マップが、6以下マップ番号を持つことが知られ、ルーチンがステップ276に進み、そこで、テクセルがバンク0またはバンク1に格納されているかどうかが判断される。テクセルがバンク1に格納されているならば、ルーチンがステップ278に進み、そこでは、テクセル・アドレス・ビットS[7:6] がサブテクスチャIDビットS[1:0]の反転に等しく、そして、テクセル・アドレス・ビットT[7:6]がサブテクスチャIDビットT[1:0]に等しい。ルーチンは、そのようなテクセルについて完了する。テクセルがバンク0に格納されているならば、ルーチンがステップ280に進み、そこでは、テクセル・アドレス・ビットS[7:6]がサブテクスチャIDビットS[1:0]に等しく、テクセル・アドレス・ビットT[7:6]が、サブテクスチャIDビットT[1:0]に等しい。ルーチンは、そのようなテクセルについて完了する。

【０１７１】

H. テクスチャ・データ組織の例

次の例が、TIMが発明の上に説明した実施例に従って、テクスチャ・データを組織する手順を説明する。特定のアプリケーションについて、描画されるべきプリミティブAがテクスチャAにマップし、プリミティブBがテクスチャBにマップすることがある。一つの可能性は、TIMがテクスチャAをテクスチャ・データの複数のブロックに組織し、それから、テクスチャAと同じブロック内で、テクスチャBを異なるサブ・テクスチャに組織することである。TIMが、プリミティブAおよびBを描画する前に、テクスチャAおよびBを含むテクスチャ・データのブロックをキャッシュ・メモリにダウンロードする。

【０１７２】

代わりに、TIMは、テクスチャAをテクスチャ・データの複数のブロックに組織し、それからキャッシュ・メモリにテクスチャAを含むブロックをダウンロードすることができる。TIMは、次いでテクスチャBを、メイン・メモリにおいてテクスチャAと同じブロック内で異なるサブ・テクスチャに組織することができる。この状況において、TIMは、テクスチャマッピング・チップ46（図6）の演算を停止させるコマンドを出し、テクスチャ・データ（同じブロックのテクスチャAおよびBを含む）の新しく組織されたブロックをテクスチャマッピング・システムのキャッシュ・メモリにダウンロードする。停止状態が実行されず、メイン・メモリからの新しく組織されたデータがテクスチャマッピング・システムのキャッシュ・メモリにダウンロードされなかったならば、間違ったテクスチャマッピング・データが、プリミティブBのレンダリング中にアクセスされることがありうることを理解しなければならない。テクスチャBを含むデータのブロックのための読み取りキャッシュ・タグが、テクスチャAを格納するキャッシュにおけるデータのブロックに対応するブロック・タグとマッチするので、プリミティブBを描画しているとき、ヒットがキャッシュ・ディレクトリの中で生じることがあるから、このようになる。しかし、キャッシュの中のデータのブロックは、テクスチャAに関連するテクスチャ・データだけを格納しテクスチャBのものを格納しない。

【０１７３】

Ｉ．　３次元のプリミティブ・パイプラインのバイパスとテクスチャ・マップをダウンロードするための割込み方式
上で示したように、ハードウェア装置の一つの機能が、新しいテクスチャのためのMIPマップが、三次元のプリミティブ・データを扱うためのパイプラインとは別のデータ経路を通してテクスチャマッピング・ハードウェアの局部メモリにダウンロードされるのを可能にする。図に表された実施例を参照すると、テクスチャマッピング・ボード12（図6）およびテクスチャマッピング・チップ46（図8）の各々は、三次元のプリミティブ・データおよびテクスチャ・データをそれぞれ受け取るために別々のポートを持つ。三次元のプリミティブ・データが、集信装置チップ36からバス18を通して受け取られ、一方、テクスチャ・データは、２次元の幾何アクセラレータ・チップ34からバス24を通して受け取られる。したがって、新しいテクスチャ・データがホストコンピュータ15からTIMによってテクスチャマッピング・チップ46にダウンロードされるとき、フロントエンド・ボード10およびテクスチャマッピング・チップ46を通る三次元プリミティブ・パイプラインは、フラッシュされる必要がなく、それによって、従来のテクスチャマッピング・システムと比較されるとき、増大したバンド幅を提供する。

【０１７４】
従来のテクスチャマッピング・システムは、新しいテクスチャ・データがテクスチャマッピング・ハードウェアの局部メモリにダウンロードされるときはいつでも、三次元のプリミティブ・パイプラインのフラッシングを必要とした。テクスチャ・データをダウンロードするための、三次元のプリミティブ・パイプラインをバイパスする別のデータ経路は、特に上で説明した、テクスチャマッピング・ボード12の局部メモリがキャッシュとして実施される実施例と関連して有益である。上で示したように新しいテクスチャ・データがキャッシュにダウンロードされるとき、テクスチャのMIPマップの全シリーズではなく、必要なMIPマップの部分だけがダウンロードされる。このように、三次元のパイプラインのバイパスは、パイプラインをフラッシュすることなくキャッシュ・ミスを扱うことを可能にする。

【０１７５】
上で示したように、図7に示した一つの実施例において、グラフィックス・システムのいくつかの部分は、システムのバンド幅を増やすために２重にされている。テクスチャマッピング・ボード12には2つのテクスチャマッピング・チップ46Aおよび46B、ならびに2つのキャッシュ・メモリ48Aおよび48Bが備えられている。この実施例において、両方のキャッシュ・メモリ48は、両方の2つのテクスチャマッピング・チップが同時に同じテクスチャ・データを使用してプリミティブに典型的に作用するので、いつでも同じテクスチャ・データを維持する。したがって、一つによってミスが受け取られるときにいつでも両方のキャッシュを更新することにより、この実施例は、同じテクスチャ・データが、別々の演算の中で2つのキャッシュにダウンロードされなくてもよいようにすることによって、システムのバンド幅を保全する。

【０１７６】
図7の二部分からなるテクスチャマッピング・チップ実施例において、各キャッシュ・メモリが、ホストコンピュータからダウンロードされたテクスチャ・データだけで更新され、テクスチャマッピング・ハードウェアからローカルに書かれない。したがって、キャッシュの一つからのミスに答えてテクスチャ・データがホスト・コンピュータからダウンロードされるときはいつでも、確実に両方のキャッシュが新しいテクスチャ・データで更新されるようにすることによって、2つのキャッシュ・メモリの間の整合性が、維持される。このように、テクスチャマッピング・チップ46の一つからキャッシュ・ミスが生じ、割込みが生成されるとき、両方のキャッシュ・メモリがダウンロードされたテクスチャ・データで更新されることができるように両方のテクスチャマッピング・チップ46が停止させられる。さらに、ハードウェアが2つのテクスチャマッピング・チップ46から異なるキャッシュ・ブロックへの同時キャッシュ・ミスを支援し、ミスに応答して両方のキャッシュにテクスチャ・データの両方の新しいブロックをダウンロードする。

【０１７７】
図6の実施例において、テクスチャ・データをダウンロードするために2ーD（二次元）幾何アクセラレータ・チップ34を通る2ーDプリミティブ・パイプラインを使うことによって、3ーD（三次元）プリミティブ・パイプラインのバイパスが、達成される。理解しなければならないテクスチャマッピング・チップ46にテクスチャ・データをダウンロードするためのデータ経路は、3-Dプリミティブ・パイプラインをバイパスしながら、いくつかの別な方法で実施することができる。例えば、専用のデータ経路がホストコンピュータからテクスチャマッピング・ボードに提供されることができる。
【０１７８】
ホストコンピュータは、同時に複数のプロセスを作用させることができ、あるシステム資源をロックするためにプロセスを割り当てるためのなんらかの手法を提供するLJNIXのようなオペレーティング・システムを使うことができる。この場合、プロセスは、ロックされたとき割り込まれない。このロッキング方式を使用することによって、あるハードウェア資源を使用するプロセスは、それらの資源のロックを解除するまでプロセスが交換（スワップ・アウト）されてしまわないようにすることができる。
【０１７９】
一つの実施例において、プロセスによる使用、すなわち、速いロックおよび遅いロックのために、2種類のロックが提供される。速いロックが使われるとき、交換（スワップ・イン）されつつあるプロセスが、適当なハードウェア資源をチェックしてそれらの資源を使用する最後のプロセスであったかどうか判断する。そうであった場合、プロセスは単にハードウェア資源の状態を復元することなく続く。プロセスが資源を使用する最後のものでないならば、遅いロックが要求され、プロセスが最後に交換（スワップ・アウト）されたときにそれらがあった状態に、ハードウェア資源を回復させる。いくつかの代替技法が、同じ結果を達成するために使われうることを理解しなければならない。
【０１８０】
3-Dプリミティブを描画しながらテクスチャ・データをダウンロードするために2ーDプリミティブ・パイプラインが使用される実施例において、2-Dおよび3-Dのプロセスは、同時には動作されない。この制限は、ホストコンピュータのオペレーティング・システムによって提供されるロッキング方式を使用して、3-Dパイプラインが空でなければ、2ーDプロセスが開始しないようにし、2-Dパイプラインが空でなければ、3-Dプロセスが開始しないようにすることによって、満足することができる。。3-Dプロセスが開始するとき、ロックを主張し、以前のプロセスが2ーDプロセスであったとき、2-Dパイプラインが空になるまで開始を待つ。同様に2ーDプロセスが開始するとき、ロックを主張し、以前のプロセスが3-Dプロセスであったとき、3-Dパイプラインが空になるまで開始を待つ。
【０１８１】
あるプロセスは、3-Dおよび2-D演算の両方を実行し、そして、遅いロックをあきらめることなく3-Dプリミティブと2-Dプリミティブとの間で切り換えることができる。そのようなプロセスは、ハードウェアに2-Dプリミティブ・データをダウンロードする前に3-Dパイプラインが空であることを確実にし、同様に、3-Dプリミティブ・データをダウンロードする前に2ーDパイプラインが空であることを確実にする方式を実行する。この結果を達成するために、2-Dおよび3-Dプリミティブ・パイプラインのそれぞれが空であるかどうかを示すレジスタ・ステータス・ビットが、提供されてもよい。2-Dおよび3-Dプリミティブ・データを使う任意のプロセスは、2ーDおよび3-Dプリミティブ・データの間の切り換え前にパイプラインが空であることを確実にするために、このステータスレジスタを読み出す。
【０１８２】
図の中で表された実施例がキャッシュとして実行されるテクスチャマッピング・ボード上に局部メモリを含むけれども、実施例は、それほど制限されないことが理解されなければならない。テクスチャマッピング・システムは、テクスチャマッピング・ボード上の局部メモリがキャッシュでないようにし、他の技法を使って、プリミティブを描画するために必要とされるテクスチャマッピング・データの各ブロックを、プリミティブが描画される

前に、3-Dプリミティブ・パイプラインと別の経路を通ってダウンロードし、プリミティブが描画される時テクスチャマッピング・データが局部メモリから得られるようにして、実施することができる。

【０１８３】

さらに、局部メモリのデータのブロックを更新するためにホストコンピュータに割込みを生成するための方式は、多くの他のアプリケーションで使用することができ、テクスチャマッピング・ハードウェアシステムにおける使用に制限されないことが理解されなければならない。この方式は、処理すべきデータのブロックを格納するメイン・メモリを持つホストコンピュータ、および処理中のデータのブロックを格納する局部メモリを持つデータ処理ハードウェアを含む任意のデータ処理システムで有用である。

【０１８４】

J. キャッシュ・ブロック置き換え方式

上で示したようにキャッシュにないテクセル・データのブロックについてミスが生じるとき、TIMは、キャッシュ48（図6）にテクスチャ・データの要求されたブロックをダウンロードする。ミスが生じたときキャッシュが満杯であるならば、キャッシュ・ブロックの一つが、テクスチャ・データの新しくダウンロードされたブロックによって置き換えられる。発明の一つの実施例において、ソフトウェア・デーモンが、どのブロックの使用の最近度が低いか、およびどのブロックが、低優先度のテクスチャ・データを持つかを考慮することによって、どのブロックを置き換えるかを判断する。決定を作るためのルーチンは、下で説明される。テクスチャマッピング・チップ46が、どのキャッシュ・ブロックを置き換えるかを判断する際にソフトウェアを支援する2組のレジスタを含む。キャッシュ・ミスが生じるとき、これらのレジスタが3-Dバイパス・データ経路を通してTIMによって読まれ、どのキャッシュ・ブロックを置き換えるかを判断するのに使用される。

【０１８５】

レジスタの第１の組は、それぞれキャッシュ48のバンク0および1に対応する2つの３２ビットの最も最近使用されたレジスタMRU0とMRU1（集合的にMRU）を含む。これらのレジスタの中の各ビットが、その対応するキャッシュ・バンクに含まれた32のキャッシュ・ブロックの一つに対応する。ヒットがキャッシュのブロックに生じる都度、MRU0およびMRULの対応するビットがセットされるので、最も最近使用されたレジスタがそのキャッシュについてのヒットを累算する。

【０１８６】

レジスタの第２の組は、それぞれキャッシュのバンク0および1に対応する2つの３２ビットの現在使用中のレジスタCU0およびCUI（集合的にCU）を含む。ビットがCU0またはCUIのいずれかにセットされるとき、それは、対応するキャッシュ・ブロックが、現在キャッシュのミニのディレクトリにあって、置き換えられてはならないことを示す。キャッシュ・ミニのディレクトリは、詳細に下で説明される。

【０１８７】

K. キャッシュ演算の停止

一つの実施例において、プリミティブのレンダリング中に必要とされる前に3-Dプリミティブのためのテクスチャ・データがメモリ48にダウンロードされるように、禁止キャッシュ・ミスによってテクスチャマッピング・ボード上の局部メモリ48のキャッシュ演算を停止にする能力が備えられる。各テクスチャマッピング・チップ46が、キャッシュとしてのその局部メモリの動作が使用可能であることを示すステータスビットを含む。このステータスビットがアサートされる（アクティブにされる）とき、キャッシュ・ミスは、ホストコンピュータの割込みとテクスチャマッピング・チップの中断になる。ステータスビットがアサートされないとき、テクスチャマッピング・ボード上の局部メモリ48は、キャッシュとして作用せず、プリミティブによって必要とされる前に任意のプリミティブのためのテクスチャ・データがメモリ48にダウンロードされるので、メモリへのミスが生じない。発明の一つの実施例において、キャッシュとしての局部メモリの動作が使用不可能にされるとき、テクスチャ・データと対応する3-Dプリミティブ・データとの同期を容易にする

ために、テクスチャ・データが、TIMによってテクスチャマッピング・ボード上の局部メモリに3-Dプリミティブ・パイプラインを通してダウンロードされる。

【０１８８】

L. キャッシュ・ミスに応答してテクスチャ・データをダウンロードするための方式を支援するテクセル・ポート・レジスタ

上で示したように、テクスチャマッピング・チップ46（図6）は、TIMによってダウンロードされたテクスチャ・データを受け取るために使用されるテクセル・ポート92（図8）を含む。テクセル・ポートは、テクスチャ・データのダウンロードを支援するいくつかのレジスタを含む。これらのレジスタのいくつかは、レジスタMRUおよびCUを含めて上で説明した。他のテクセル・ポート・レジスタは、コマンド・レジスタ、ステータスレジスタ、テクセル・データレジスタ、ディレクトリ・タグ・レジスタ、キャッシュ・アドレスレジスタおよびパイプ・タグ・レジスタを含み、それぞれの作用を下に説明する。

【０１８９】

アクセスが、それらが3-Dプリミティブ・パイプラインを通して書かれるのを可能にするようテクセル・ポート・レジスタに提供される。テクセル・ポート・レジスタは、3-Dパイプラインが忙しいときでも、書かれることができ、レジスタを書くためのデータは、単にパイプラインに置かれる。さらに、テクセル・ポート・レジスタは、24ビットのバス24（図6）を通して提供された3-Dパイプライン・バイパスを通してアクセスすることができる。テクセル・ポート・レジスタをアクセスするとき、バス24の8つのビットが、どのテクセル・ポート・レジスタが読み出されまたは書き込まれるべきかを指定するレジスタ・アドレスとして使用され、データがテクセル・ポート・レジスタに書かれているとき、バスの他の16のビットがデータを提供する。テクセル・ポート・レジスタの組織は、図23に示される。一つの実施例において、たとえいくつかのレジスタのいくつかのビットが不使用であるとしてもテクセル・ポート・レジスタの各々が、32ビットを含む。

【０１９０】

1. テクセル・コマンド・レジスタ

テクセル・コマンド・レジスタは、下で説明されるようにキャッシュ・ミスを処理するTIMによって使用されるいくつかのビットを含む。停止ビット350は、TIMによってセットされて、テクスチャマッピング・チップにその動作を停止するように命令する。上で述べたように、2つのテクスチャマッピング・チップが備えられる実施例において、両方のテクスチャマッピング・チップが、いずれかからのキャッシュ・ミスに応答して同じテクスチャ・データで更新されるので、キャッシュが一貫したままに保たれる。このように、ミスがいずれかのテクスチャマッピング・チップから受け取られるとき、両方は、それらのそれぞれのテクセル・コマンド・レジスタに停止ビット350をセットすることによって停止させられる。停止ビットは、キャッシュ・ミスに応答してTIMによって新しいテクスチャ・データがホストコンピュータからダウンロードされたあと、ビットをクリアするためにコマンド・レジスタに書込むことによって、キャッシュ・ミスを扱うソフトウェア・ルーチンにより消去される。

【０１９１】

割込みイネーブル・ビット352は、アサートされると、キャッシュ・ミスが生じるとき、テクセル・ポートからの割込みをイネーブルする。このビットは、テクスチャマッピング・ボード12（図6）上の局部メモリ48をキャッシュとして作用しなくする上で説明した能力を提供するためにアサート解除される。

【０１９２】

書込みLoki0および書込みLoki1ビット354および356は、テクセル・ポート・レジスタのための書込みイネーブルである。Lokiは、テクスチャマッピング・チップ46を識別するために使用される短縮名である。2つのそのようなチップが使用される実施例において、チップは、それぞれLoki0およびLoki1と呼ばれる。一つのテクスチャマッピング・チップだけが使われているとき、それはLoki0として識別される。テクセル・ポート・レジスタのいずれかに書き込むためにコマンドがテクセル・ポート・バス24を通して受け取られるとき

、その書込みビットがイネーブルされているかどうか判断するために、各テクスチャマッピング・チップ（すなわち、Loki0およびLoki1）がそのコマンド・レジスタをチェックし、もしイネーブルされているならば、受け取られた書込みコマンドに従ってそのテクセル・ポート・レジスタをアップデートする。このように、書込みLoki0および書込みLoki1ビット354および356の値を制御することによって、TIMが、2つのテクスチャマッピング・チップのテクセル・ポート・レジスタに別々にまたは組合せて書き込むことができる。

【０１９３】

Loki読み取りビット358が、テクスチャマッピング・チップの一つのテクセル・ポート・レジスタの読み取りをイネーブルする。テクセル・ポート・レジスタを読み出すためにコマンドがテクセル・バス24を通して受け取られるとき、そのテクセル・ポート・レジスタの中身をバスに提供するために、テクスチャマッピング・チップの一つだけが一度に応答する。2つのテクスチャマッピング・チップが備えられる実施例において、各々が、チップがLoki0であるかまたはLoki1であるかを示すよう配線されたピンを備えていてもよい。Loki読み取りビットは、ソフトウェアによってセットされるとき、読み取りがLoki1からイネーブルされていることを示し、読み取りビットがアサート解除されているとき、それは、読み取りがLoki0のためにイネーブルされていることを示す。以上のことからわかるように、テクセル・コマンド・レジスタのフォーマットは、それを同時に同じデータで両方のテクスチャマッピング・チップ（Loki0とLoki1）に書き込むのを可能にし、それによって、両方のコマンド・レジスタを書き込むのに一つの書込みサイクルだけが要求される。

【０１９４】

2. テクセル・ステータス・レジスタ

テクセル・ポート・ステータスレジスタは、アサートされるとき、システムが、2つのテクスチャマッピング・チップを含むことを示すデュアルLokiビット360を含む。ビット352がコマンド・レジスタにおいてアサートされるときはいつでも、割込みイネーブル・ビット362がアサートされ、テクスチャマッピング・チップの局部メモリがキャッシュとして作用していることを示す。このキャッシュは、キャッシュにないテクスチャ・データが必要とされるとき、ホストコンピュータに割り込むキャッシュ・ミスを生成する。このビットが、コマンド・レジスタおよびステータスレジスタに含められるので、テクセル・ポートのステータスは、単にステータスレジスタを読み出すことによって読み出すことができる。

【０１９５】

割込みがテクスチャマッピング・チップから生じ、そのチップが新しいテクスチャ・データがダウンロードされるのを待っているとき、割込み有効ビット364がアサートされる。このビットは、キャッシュ・ディレクトリ・タグ・レジスタ（下で説明する）に、キャッシュにおいてミスしたタグであるパイプ・タグ・レジスタ（下で説明する）に格納されたキャッシュ読み取りタグにマッチするキャッシュ・タグが書かれるとき、クリアされる。ステータスレジスタは、キャッシュ・ミスが生じる時テクスチャマッピング・チップの停止を支援する2ビットを含む。停止イネーブル・ビット368は、停止ビット350がコマンド・レジスタにおいてセットされクリアされるときはいつでもTIMによってセットされクリアされ、そのビットがアサートされる時テクスチャマッピング・チップにそれ自身を停止するよう命令する。このビットは、ステータスレジスタおよびコマンド・レジスタに提供されるので、テクスチャマッピング・チップのステータスが一つのレジスタに格納される。キャッシュ・ミスが生じたキャッシュ・ディレクトリがデータがダウンロードされるのを待っているとき、割込み有効368がハードウェアによってテクスチャマッピング・チップにセットされる。このビットは、キャッシュ・ディレクトリ・タグ・レジスタ（下で説明する）にキャッシュでミスしたブロックアドレスにマッチするキャッシュ・タグが書かれる時クリアされる。

【０１９６】

3. パイプ・タグ・レジスタ

パイプ・タグ・レジスタは、テクスチャマッピング・チップにおけるパイプラインによってインデックスを付けられた最後のブロック・タグを格納する。キャッシュ・ミスが生じるとき、パイプ・タグ・レジスタが、キャッシュでミスしたブロック・タグ370を格納する。このように、テクセル・ポート・バス24を通してパイプ・タグ・レジスタを読むことによって、キャッシュ・ミス割込みに応答するソフトウェアが、ミス応答してキャッシュにダウンロードされなければならないキャッシュ・ブロックのためのタグを決定することができる。

【０１９７】

4. テクセル・データレジスタ

キャッシュ・ミスが生じるとき、テクセル・データレジスタは、キャッシュ48にテクスチャ・データをダウンロードするために使用される。上で述べたように、各テクセルが、32ビットのデータによって表され、一つのバイト372がアルファを表し、一つのバイト374が赤の値を表し、一つのバイト376が緑の値を表し、一つのバイト378が青の値を表す。

【０１９８】

5. テクセル・キャッシュ・アドレス・レジスタ

テクセル・キャッシュ・アドレス・レジスタは、キャッシュにデータを書き込みキャッシュ・ディレクトリにブロック・タグを書き込むのに使用される。上で示したようにキャッシュが、各ブロックが256x256テクセルのアレイを含むテクスチャ・データの64のブロックを格納する。テクセル・キャッシュ・アドレス・レジスタが、キャッシュの64のブロックの中で読み出されまたは書き込まれる特定のものを識別する6ビットのブロック・インデックス・フィールド380を含む。さらに、レジスタは、ブロック・インデックス・フィールドにおいて識別されたブロック内で読み出しまたは書き込み中の特定のテクセル・アドレスを識別する16ビットのブロックアドレス・フィールド382を含む。データが、キャッシュ・ミス応答してテクスチャ・メモリにダウンロードされるとき、ブロック・インデックスが、下で説明される置き換え方式を使用するソフトウェア・ルーチンによってセットされ、ブロックアドレス・フィールド382が、ブロックの最初のテクセルを書き込むためにゼロに初期設定される。キャッシュ・アドレス・レジスタは、テクセル・データレジスタがアクセスされるときはいつでも、ブロックアドレス・フィールド382をインクリメントさせる。このように、ブロックアドレス・フィールドが、テクセル・データの新しいブロックをキャッシュに書き込むためにキャッシュ・ブロック内のブロックアドレスのすべてにわたって増加されることができる。

【０１９９】

6. テクセル・ディレクトリ・タグ・レジスタ

テクセル・ディレクトリ・タグ・レジスタは、キャッシュ・ブロック・タグを表す23ビットのブロック・タグフィールド384を含み、キャッシュ・アドレス・レジスタのブロックインデックスフィールド380で規定されるキャッシュ・ディレクトリ・エントリを書き込むために使用される。上で示したようにキャッシュ・ブロック・タグの23のビットは、テクスチャIDの8ビット、S座標の7ビット、T座標の7ビット、およびブロック・タグに対応するテクスチャ・データのブロックによって表されるマップのMIPマップ・シリーズにおけるマップ番号を識別する付加的な1ビットを表す。テクスチャ・データの新しいブロックが、キャッシュ・ミスに応答してTIMによってダウンロードされるとき、そのブロック・タグがディレクトリ・タグ・レジスタにテクセル・バス24を通してロードされる。ディレクトリ・タグ・レジスタからブロック・タグが、キャッシュ・アドレス・レジスタのブロックインデックスフィールド380によって識別されるエントリにおいてキャッシュ・ディレクトリに書かれる。上で述べたように、パイプ・タグ・レジスタのタグにマッチする（その読み取りがキャッシュ・ミスを生じたもの）ブロック・タグがディレクトリ・タグ・レジスタに書かれるとき、キャッシュ・ミス割込みがクリアされる。

【０２００】

M. キャッシュ・ミス割り込みを処理するためのソフトウェア・ルーチン

以上のことからわかるように、テクセル・ポート・レジスタが、ホストコンピュータ15の

上で作用し、キャッシュ・ミス割込みを処理して必要なテクスチャ・データをダウンロードするTIMによって使用される。このTIM動作のフローチャートは、図24に示される。ステップ400において、Loki0とLoki1のためのテクセル・コマンド・レジスタは、停止ビット350を両方にセットするために書かれる。この方法は、テクセル・ステータスレジスタの停止させられたビット368を読み出すために次いでステップ402に進み、両方のLokiが停止させられたかどうか判断する。この方法は、両方が停止されたことが判断されるまで連続的にLoki0およびLoki1のステータスレジスタを読み出し、それからステップ404に進む。システムが、一つのテクスチャマッピング・チップ46だけ（すなわち、Loki0）を含むとき、Loki0は、そのレジスタの中身をテクセル・バス24に提供するによってLoki1のテクセル・ポート・レジスタを読み出す要求にも応答する。このように、TIMがステップ402において、両方のLokiが停止したかどうか判断するためにチェックするとき、Loki0がLoki1の読み取りに応答して、Loki0が停止しているとき、方法はステップ404さらに進む。

【０２０１】

ステップ404において、Loki0のテクセル・ステータスレジスタの割り込み有効ビット364を読み、Loki0がキャッシュ・ミスを引き起こすように割り込んだかどうか判断し、割り込んでいれば、この方法は、ステップ406に進み、キャッシュにおいてミスしたテクスチャ・データのブロックのブロック・タグを識別するためにLoki0のパイプ・タグ・レジスタが、読み出される。TIMが、このブロック・タグを使用してメモリ17（図6）におけるテクスチャ・データに対応するブロックをアクセスし、ステップ408に進んで、キャッシュの中のどのブロックが、ダウンロードされる新しいキャッシュ・ブロックで置き換えられなければならないかを判断する。この判断は、発明の最下位優先度および最低使用最近度(least recently used)方式を使用して、下で説明する態様で行われる。

【０２０２】

上で述べたように、システムが2つのテクスチャマッピング・チップを含むとき、各々におけるキャッシュは、全く同じエントリを持つように維持される。したがって、テクスチャマッピング・チップの一つからのキャッシュ・ミスに応答してホストコンピュータからダウンロードされたテクスチャ・データが、両方のチップのキャッシュに書かれる。このように、置き換えられるべきキャッシュ・ブロックが識別されたならば、この方法は、ステップ410に進み、Loki0とLoki1（Loki1が存在するならば）のキャッシュ・アドレスレジスタにステップ408の間に決定されたブロック・インデックスが書きこまれる。ステップ412において、ディレクトリ・タグ・レジスタに、キャッシュ・ミスに応答してテクスチャ・キャッシュにダウンロードされるテクスチャ・データのブロックのブロック・タグが書かれ、ステップ414において、テクスチャ・データがテクセル・データレジスタに書かれる。このようにして、この方法は、キャッシュ・ミスに応答して、キャッシュにおいてミスされたテクスチャ・データのブロックをダウンロードし、キャッシュにこのブロックのデータを書く。

【０２０３】

テクスチャ・データのブロックがLoki0とLoki1にステップ406-414でダウンロードされたあと、またはLoki0が割込みしなかったことがステップ404で判断されるならば、この方法は、ステップ416に進み、Loki1ステータスレジスタにおける割込み有効ビット364がセットされているどうかに関して判断がなされる。セットされていれば、キャッシュ・ミスが、Loki1の中で生じたことを意味する。上で示したように、システムが一つのテクスチャマッピング・チップだけを含むならば、Loki0が、Loki1テクセル・ポート・レジスタの読み取りに応答する。Loki0は、Loki1のステータスレジスタの読み取りに応答するとき、その割込み有効なビット364をマスクするのでTIMがステップ416で、Loki1が割込みしなかったことを判断する。このマスキングは、TIMがLoki0からの割り込みを再度処理してステップ406-414でダウンロードされたテクスチャ・データのブロックを再びダウンロードすることがないようになされる。したがって、一つのテクスチャマッピング・チップだけがあるシステムにおいて、この方法は、Loki1が割込みしなかったことをステップ416で判断し、ステップ418に進み、Loki0のコマンド・レジスタが停止ビット350をアサート解除する

よう書き込まれ、テクスチャマッピング・チップがプリミティブをそのパイプラインで処理することを可能にする。

【０２０４】

システムが2つのテクスチャマッピング・チップを含むとき、この方法は、Lokilが割り込んだかどうかをステップ416で判断し、割り込んでいなければ、直接ステップ418に進み、停止ビットが両方のテクスチャマッピング・チップでアサート解除され、それらがプリミティブの処理に進むことを可能にする。しかし、キャッシュ・ミス応答してLoki1が割り込んだことがステップ416で判断されるとき、この方法は、ステップ406-414に関連してLoki0からの割込みの扱いを説明したのと同じ態様で割込みを処理するため、ステップ420-424ないしプロセスに進む。この方法は、次いでステップ418に進み、両方のテクスチャマッピング・チップの停止ビットがアサート解除される。

【０２０５】

2つのテクスチャマッピング・チップがあるシステムにおいては、両方のチップが、同時に同じブロック・タグまたは異なるブロック・タグのためにキャッシュ・ミス割込みを生成することができることが理解されねばならない。両方のテクスチャマッピング・チップが同じブロック・タグのためにキャッシュ・ミス割り込みを生成する時、割込みは、ステップ400-414で処理される。したがって、ステップ416において、この方法はLoki1からの割込みを発見しない。ミスされたブロック・タグをステップ412で両方のLokiのディレクトリ・タグ・レジスタに書込むことによってLoki1からの割込みがクリアされるからである。このように、図24に示された方法が、テクスチャマッピング・チップの一つまたは両方から同時の割込みに応答することができる。

【０２０６】

N. キャッシュ・ミニのディレクトリおよびメイン・ディレクトリ

上で述べたように、一つの実施例において、キャッシュは、データの256×256のテクセルの64のブロック、および23ビットのブロック・タグの64のエントリを含む完全に連想的なキャッシュ・ディレクトリを含む。装置が3線形の補間モードで作用しているとき、8つのテクセル読み取りが、ピクセルについての結果のテクセル・データを決定するために実行される。その際、一つのマップの4つのテクセルが一つの読み取り動作で同時に読み出され、そして、他のマップの4つのテクセルが第２の読み取り動作で同時に読み出される。処理中のピクセルが、キャッシュ・ブロック境界の付近にあるマップのロケーションにマップするならば、一つのマップ内で結果のテクセル・データを生成するためにキャッシュから読み出された4つのテクセルは、それぞれ異なるキャッシュ・ブロックにあることができる。このように各ピクセルについてキャッシュから同時に4つのテクセルを読み取るには、キャッシュ・ディレクトリにおいて64のブロック・タグ・エントリと4つの別々の比較をすることを必要とする。

【０２０７】

従来の完全に連想式 のキャッシュが2つの方法のうちの一つで動作する。第１に、読み出されたタグが一つのサイクルであらゆるキャッシュ・タグ・エントリと比較されることができるよう、各キャッシュ・タグ・エントリについて別のハードウェア・コンパレータを用意する。そのような技法は、ハードウェア・コストが大きく、そこでは、4つの読み取りが、同時に行われ、そして、256（すなわち、4×64）の23ビットのコンパレーターを必要とする。従来の完全に連想式 のキャッシュによって使われる第２の技法は、一つのキャッシュ・タグ・コンパレーターを使用し、そして、各キャッシュ・エントリは、読み出されたタグと直列に比較される。そのような技法は、システムのバンド幅に消極的な影響を与え、そこでは、一つの読み出された動作中に読み出された4つのテクセルの各々がキャッシュにあったどうか判断するために、潜在的に256の読み出しサイクルがキャッシュ・ディレクトリに必要である。

【０２０８】

これらの問題を克服するために、キャッシュ・システムは、ミニ・ディレクトリ（図25）およびメイン・ディレクトリ（図26）を含む。ミニのディレクトリは、完全に連想式 で

、5つの最も最近読み出されたキャッシュ・ブロック・タグおよびそれぞれについての対応するブロックインデックスを含む。図25に示すように、ミニ・ディレクトリ 500は、ミニのディレクトリから出力501-505通してそれぞれ出力される5つのエントリを含み、それぞれが4つのグループのタグ・コンパレータ507-510に結合する。各グループの中のタグ・コンパレータ507-510が、5つの23ビットのコンパレータ（示さない）を含み、そして、2線または３線形の補間を実行するとき、一つの読み取り動作で実行された4つのキャッシュ読み取りタグの一つに対応する。このように、ミニ・ディレクトリの完全に連想式 の性質が、同時に読み出されるタグの数にミニ・ディレクトリのエントリの数を掛けた数に等しい20の23ビット・コンパレータで実行される。

【０２０９】

一つのピクセルについて同時に読み出される4つのキャッシュ読み取りタグが、ピクセルがマップするマップにおけるロケーションに最も近い4つのテクセルを含むキャッシュ・ブロックを識別し、これらのタグは、左上（UL）タグ、右上（UR）タグ、左下（LL）タグおよび右下（LR）タグと呼ばれる。左上、右上、左下および右下テクセルのためのキャッシュ読みとりタグは、それぞれ左上、右上、左下と右下タグ・コンパレータ507-510のグループに接続されている。各グループのタグ・コンパレータ507-510が、その対応するキャッシュ読み取りタグをミニ・ディレクトリに格納した5つのブロック・タグに対して比較し、タグがミニ・ディレクトリ・エントリの一つと一致するかどうかを示すヒット出力を生成し、そしてその場合、テクセル・データの対応するブロックが格納されているキャッシュのロケーションを示すブロックインデックスを出力する。

【０２１０】

以上のことからわかるように、4つのキャッシュ読み取りタグ(UL, UR, LL, LR)のそれぞれが、ミニ・ディレクトリにあるならば、テクセル・データの対応する4つのブロックが格納されているキャッシュのロケーションを識別するブロックインデックスを判断するために、一つのディレクトリ・アクセスだけが必要である。１つまたは複数の読み取りタグがミニ・ディレクトリにない場合だけ、アクセスはメイン・キャッシュ・ディレクトリにされる。キャッシュ読み取りタグがミニ・ディレクトでミスする都度、ミニ・ディレクトリ500が更新されるので、ミニ・ディレクトリ500は、テクスチャ・データの5つの最も最近アクセスされたブロックのブロック・タグを常に含む。

【０２１１】

4つのキャッシュ読み取りタグの１つまたは複数がミニ・ディレクトリでヒットしないならば、アクセスは、メイン・キャッシュ・ディレクトリ520（図26）になされる。上で述べるメイン・ディレクトリが、それぞれがブロック・タグを含む64のエントリを含む。メイン・ディレクトリが、64の23ビット・コンパレータ522を備えているので、キャッシュ読み取りタグは、一つのサイクルで全メイン・ディレクトリと比較されることができる。コンパレーター522が、キャッシュ読み取りタグは、メイン・ディレクトリのエントリの一つをヒットしたかどうかを示す信号を提供し、ヒットしていれば、読み取りタグに一致したコンパレーターのロケーションも、テクセル・データの対応するブロックがキャッシュのどこにあるかを識別するブロックインデックスを生成するために使用される。読み取りタグがメイン・キャッシュ・ディレクトリのエントリのいずれとも一致しないならば、キャッシュ・ミスが生成され、上で説明された態様で、ホストコンピュータが割り込まれテクスチャ・データの要求されたブロックをダウンロードする。

【０２１２】

上で述べたように、4つのキャッシュ読み取りタグ(UL, UR, LL, LR)の１つまたは複数がミニ・ディレクトリをヒットしないとき、メイン・キャッシュ・ディレクトリ520が、アクセスされるだけである。2以上のキャッシュ読み取りタグがミニ・ディレクトリをミスするならば、メイン・ディレクトリが各キャッシュ読み取りタグについて別々のサイクルでアクセスされる必要があったならば被られる性能ペナルティを減らすことが望ましい。この結果を達成するために、図27に示す実施例では、一群の6つの付加コンパレータ526-530が備えられる。6つのコンパレータは、同時にアクセスされる4つのキャッシュ読みとり

タグを同一のものがあるかどうか判断するために他のものと比較する。コンパレータは、URタグに対してULタグを比較するコンパレータ526、ULとLLタグを比較するコンパレータ527U、ULとLRタグを比較するコンパレータ528、URとLLタグを比較するコンパレータ529、URとLRタグを比較するコンパレータ530、およびLLとLRタグを比較するコンパレータ532を含む。

【０２１３】

コンパレータ526-532によって実行される比較は、性能ペナルティを被らないよう他の比較と平行して実行することができる。例えば、これらの比較は、キャッシュ読み取りタグがミニ・ディレクトリと比較されるサイクル中、またはミニ・ディレクトリでミスした第１キャッシュ読み取りタグがメイン・メモリと比較されるサイクル中に実行することができる。少くとも2つのキャッシュ読み取りタグが、メイン・ディレクトリをヒットせず、等しいことが判断されるとき、コンパレータ526-532の出力は、メイン・ディレクトリが、これらの少くとも2つのキャッシュ読み取りタグについては一度だけアクセスされればよいことを示すのに使われる。このように同一のタグのために、メイン・ディレクトリをアクセスする際に複数のサイクルを用いる必要はなく、それによって2以上のキャッシュ読み取りタグがミニ・ディレクトリをミスするときのシステムのバンド幅への影響を最小にする。

【０２１４】

以上のことからわかるように、キャッシュ・ミニ・ディレクトリを利用する実施例は、高いシステムのバンド幅を達成しながらキャッシュ・ディレクトリを実行するために比較的小さい量のハードウェアを使うという競合するゴールを効果的にバランスさせる。2以上のキャッシュ読み取りタグがミニ・ディレクトリをミスするときに被る性能ペナルティは、アプリケーションに依存する。4つのキャッシュ読み取りタグの2つのユニークな組が、2サイクル毎にミニ・ディレクトリを通して処理されることが可能であるけれども、4つのキャッシュ読み取りタグの各セットにおいて典型的には一つまたは2つだけのユニークなブロック・タグが現れると考えられる。上で示したように、対象物のピクセルが、描画されつつあり、3線形の補間が使われているとき、隣接のピクセルが頻繁にMIPマップの同じ2つのマップにマップし、キャッシュへの読み取りが、2つのマップを格納するキャッシュ・ブロックの間で連続的に切り換えられることを要求する。図25の中で示された実施例において、たとえ現在処理される組の読み取りタグのための4つのユニークなキャッシュ・タグがミニ・キャッシュに常駐していても、前の組の読み取りタグにおいてアクセスされた少なくとも一つのタグが、ミニ・ディレクトリの中に残るようにするために、ミニ・ディレクトリが、5つのブロック・タグを格納する。このように、3線形の補間中に、4つのユニークなキャッシュ・タグの2つの組の間で切り換えているときでも、各セットについて読み取りキャッシュ・タグの少なくとも一つがミニ・ディレクトリに残るので、4つのキャッシュ・タグは、メイン・ディレクトリに対して直列の形態で比較される必要がない。

【０２１５】

テクセルのレンダリングの間、3線形の補間が使われているとき、キャッシュへの連続的な読み取りが、一つのマップにおいて第１の組の4つのテクセルを読み出し、そして、他のものにおいて第２の組の4つのテクセルを読み出す。プリミティブが、描画されるにつれて、2つのマップのそれぞれ内の隣接のテクセルがそれぞれ１サイクルおきにアクセスされ、2以上のテクセルが、一つのキャッシュ・ブロック内に一般的に配置される。したがって、一つまたは2つのユニークなタグが、各組の4つのキャッシュ読み取りタグに現れ、各キャッシュ読み取りタグがミニ・ディレクトリ500をヒットしながら多数のピクセルを描画することができる。4つの各組において一つのキャッシュ読み取りタグだけがミニ・ディレクトリをミスするならば、次の組の4つの読み取りタグがミニ・ディレクトリと比較される一方で、そのタグは、メイン・ディレクトリに対して比較されることができるののので、性能ペナルティは被らない。

【０２１６】

メイン・ディレクトリおよびより小さいミニ・ディレクトリを含むキャッシュ・ディレクトリは、多くの他のアプリケーションで使用することができ、テクスチャマッピング・ハードウェアシステムでの使用に制限されないことを理解する必要がある。ミニ・キャッシュ・ディレクトリ方式は、複数のキャッシュ・タグ読み取りが同時に処理されるとき、および連続的にアクセスされた以前に使用されたタグにキャッシュ読み取りタグが相関しているときに、完全に連想式 のキャッシュを実行し、またディレクトリ・タグ比較のコストを減らすのに際に特に有用である。たとえば、Xタグを任意の一時点で格納するキャッシュ・メモリについて、Nキャッシュ読み取りタグがディレクトリ・ブロック・タグに対して同時に比較される場合に、MがN以上の値として、M 個のタグを含むミニ・ディレクトリを維持すれば十分である。Mミニ・ディレクトリ・タグの各々が、一つの読み取り動作でNキャッシュ読み取りタグに対して比較される。メイン・ディレクトリが、ミニ・ディレクトリでヒットしないいかなるキャッシュ読み取りタグについても順にアクセスされる。そのような読み取りタグは、メイン・ディレクトリ・タグに対して一つのサイクルで比較される。メイン・ディレクトリの中のXタグの各々が、一つの読み取り動作でN読み取りタグに対して比較されるシステムからのコンパレータの観点からのハードウェア節減は、(X + M*N)/(X*N)の比率に依存する。

【０２１７】

このハードウェア節減を達成する性能ペナルティは、連続的な読み取り動作でアクセスされるタグのシーケンスのふるまいに基づき、アプリケーション依存である。各読み取りの組において一つのタグしかミニ・ディレクトリをミスしないならば、次の組の読み取りタグがミニ・ディレクトリに対して比較されるのと平行して、ミスされたタグがメイン・ディレクトリに対して比較されることができるので、ペナルティは被られない。性能ペナルティを減らすために使用される上で説明したコンパレータ526-530に関して、2以上のキャッシュ読み取りタグが、ミニ・ディレクトリでミスするとき、4つの読み取りタグが同時にアクセスされるので、6つが使用される。他方に対して各キャッシュ読み取りタグを比較するのに使われているコンパレータの数は、同時にアクセスされる読み取りタグの数Nに依存し、（N-1）階乗に等しい。

【０２１８】

図25ないし26のミニ・ディレクトリおよびメイン・ディレクトリを含むキャッシュ・ディレクトリの一つのここで説明する実施例が、図28に示される。図25で示された実施例は、単に例としてのもので、他の実施例を使うことができることを理解しなければならない。

【０２１９】

ミニ・ディレクトリ・エントリ501-505（図25）は、タグ・レジスタ501T-505Tに格納されるタグ成分、およびインデックスレジスタ501I-505Iに格納されるインデックス成分にスプリットされる。上で説明したように、キャッシュ・ディレクトリは、処理中のピクセルがマップするMIPマップ中のロケーションに最も近い4つのテクセル（すなわち、UL、UR、LLとLR）に対応する一組の4つの読み取りキャッシュ・タグを受け取る。4つの読み取りタグの各々は、六つタグ・コンパレータ541-546に提供される。コンパレータの5つ（すなわち、542-546）は、それぞれ5つのミニ・ディレクトリ・タグ・レジスタ50IT-505Tの一つに結合している。例えば、コンパレータ542が、ミニ・ディレクトリ・エントリ1のためのタグ・レジスタ501Tに結合し、そのミニ・ディレクトリのそのエントリのタグが、読み取りキャッシュ・タグUL、UR、LLまたはLRのいずれかのタグにマッチするかどうかを示す出力を生成する。コンパレータ543-546が、同様の態様の中で作用し、それぞれ、ミニ・ディレクトリ・エントリ2-エントリ5のためのタグをそれぞれ格納するタグ・レジスタ502T-505Tに対して読み取りキャッシュ・タグUL、UR、LLおよびLRを比較する。4つの読み取りキャッシュ・タグの各新しい組は、ミニ・ディレクトリに対して一つのサイクルの中で比較される。そのサイクルの終わりに、4つのタグUL、UR、LLおよびLRが、レジスタ550-553にそれぞれ格納される。図28に示すように、レジスタ550-553のそれぞれもミニ・ディレクトリ・タグ・コンパレータ542-546の出力を受け取る制御回路559に結合される。4つの読み取りタグの新しい組がミニ・ディレクトリ・タグと比較されるサイクルの終わりに、

また、レジスタ550-553の各々にその対応するタグ（すなわち、UL、UR、LL、LR）がミニ・ディレクトリ・エントリの一つと一致したかどうかおよび一致した場合、どのエントリが一致したかを識別するデータがロードされる。

【０２２０】

上で示したように、ミニ・ディレクトリにおいて一つのキャッシュ読み取りタグだけがミスするならば、次の組の4つのテクセル読み取りタグがミニ・ディレクトリに対して比較される間に、そのタグがメイン・ディレクトリに対して比較される。ミスがミニ・ディレクトリで生じるとき、ミニ・ディレクトリが、ミスしたタグを含むよう更新されるので、ミニ・ディレクトリは、五つの最も最近アクセスされたキャッシュ・タグを常に反映する。ミニ・ディレクトリでミスした読み取りキャッシュ・タグがメイン・ディレクトリに対して比較されるサイクル中、次の組の4つの読み取りタグがミニ・ディレクトリに対して比較される間に、ミニ・ディレクトリ・タグ・レジスタ50IT-505Tが、ミニ・ディレクトリを前のサイクルでミスしたキャッシュ・タグを含むようにまだ更新されていない。したがって、読み取りキャッシュ・タグの次の組がミニ・ディレクトリに対して比較されるとき、6番目のコンパレータ541が、4つの読み取りタグ（UL、UR、LLとLR）を、前のサイクルでミニ・ディレクトリでミスし、そしてメイン・ディレクトリに対して比較されているタグと比較するために使用される。

【０２２１】

4つのキャッシュ読み取りタグ（UL、UR、LLとLR）の組の中の2以上のユニークなタグがミニ・ディレクトリをミスするならば、複数の比較がメイン・ディレクトリで生じるのでキャッシュ・ディレクトリを通るパイプラインが停止させられる。しかし、一つのユニークなタグだけががミニ・ディレクトリをミスすｒならば、パイプラインは、キャッシュ・ディレクトリが、サイクルごとに4つのキャッシュ読み取りタグの新しい組を受け取るような態様で継続する。

【０２２２】

上で述べたように、ミニ・ディレクトリに対して前のサイクルで比較された読み取りタグが、レジスタ550-553に格納される。これらのレジスタの出力が、メイン・ディレクトリに対して比較されるべきそれらのレジスタを一度に一つ選ぶ4対1マルチプレクサ555に結合され、サイクルの終わりにミニ・ディレクトリにロードされるので、ミニ・ディレクトリは、最も最近受け取られた読み取りキャッシュ・タグで更新される。また、マルチプレクサ555の出力が、6番目のコンパレータ結合されるので、ミニ・ディレクトリを前のサイクルでミスしたキャッシュ読み取りタグが読み取りタグUL、UR、LLとLRの新しい組の各々に対して比較される。コンパレータ542-546と協力してコンパレータ541は、ミニ・ディレクトリが、キャッシュ・ディレクトリによって受け取られた4つのキャッシュ読み取りタグの各組を5つの最も最近受け取られた読み取りタグに対して比較するようにする。

【０２２３】

上で述べたように、マルチプレクサ555からのキャッシュ読み取りタグ出力が、それがメイン・ディレクトリに対して比較されるサイクルの終わりに、ミニ・ディレクトリ・タグ・レジスタ50IT-505Tの一つにロードされる。このように、ミニ・ディレクトリは最も最近なアクセスされたキャッシュ・タグを含むよう更新される。どのエントリにマルチプレクサ555からの新しいキャッシュ・タグが書かれるかの決定は、下で説明される置き換え方式によってなされる。

【０２２４】

図27に関連して上で説明した六つのコンパレータ526-532の組が、図28に便宜上一つのコンパレータ・ブロックとして示される。これらのコンパレータの出力およびコンパレータ541-546の出力が、いくつかの機能を果たす制御回路559にそれぞれ提供される。ミニ・ディレクトリへのミスが生じるとき、ミニ・ディレクトリのどのエントリが新しい読み取りキャッシュ・タグで置き換えられるかを制御回路559が決定する。制御回路559は、ミニ・ディレクトリに対して比較されてつつある4つの新しく受け取られた読み取りキャッシュ・タグの一つ、またはメイン・ディレクトリに対して比較された最後の読み取りキャッシ

ュ・タグによってヒットされたどのエントリも置き換えないで、これらのエントリにミニ・ディレクトリで維持されるための最も高い優先度を割り当てる。さらに、制御回路559は、どのミニ・ディレクトリ・エントリが前の組の4つの読み取りタグによってヒットされたかに関する状態情報を格納し、そして、それらにミニ・ディレクトリの中で維持されるために次に最高の優先度を割り当てる。残りのエントリは、低い優先度を割り当てられる。

【０２２５】

制御回路559が、少くとも一つのエントリを含む最下位優先度グループにあるエントリを置き換えのために選ぶ。このように、ミニ・ディレクトリに対して比較されつつある、4つの新しく受け取られた読み取りキャッシュ・タグの一つによってヒットされず、メイン・ディレクトリに対して比較される最後の読み取りキャッシュ・タグでなく、前の組の4つの読み取りタグにない少くとも一つのエントリが低い優先度グループにあるならば、低い優先度グループの中のエントリの一つが置き換え用に選ばれる。しかし、低い優先度グループにエントリがないならば、最も高い優先度のエントリ（すなわち、4つの新しく受け取られた読み取りキャッシュ・タグの一つによっってヒットされたもの、およびメイン・ディレクトリに対して比較された最後の読み取りキャッシュ・タグ）だけを除外する、より大きいグループのエントリが選ばれ、そしてそのグループからのエントリが置き換えのために選ばれる。

【０２２６】

最下位優先度の利用可能なミニ・ディレクトリ・エントリのグループが識別されると、一つが置き換えられるごとに5つのミニ・ディレクトリ・エントリのそれぞれを通って循環する置き換え方式に従って、グループの中のどのエントリが置き換えられなければならないかが決定される。これは、いくらかの方法でされることができる。一つの実施例では、5つのミニ・ディレクトリ・エントリは、1ないし5と名称をつけられる。置き換えられるエントリが、最下位優先度グループから、まずグループにない最も高い番号のエントリを第１に識別し、次いで置き換えのためにグループにある次に高い番号のエントリを選ぶことによって、選ばれる。エントリ5がグループにないとき、この方式がくるまるので、エントリ1が次の最高の番号のエントリとみなされる。この置き換え方式を通して、制御回路559は、一つが置き換えらねばならない時ごとにミニ・ディレクトリのエントリを循環し、選ばれたミニ・ディレクトリ・タグ・レジスタ501T-505Tのローディングを制御する。制御回路559は、4つの読み取りタグ（UL、UR、LLとLR）の各々のためのデータを生成するためにコンパレータ541-546の出力をデコードし、読み取りタグがエントリをミニ・ディレクトリの中で一致したかどうかおよび一致した場合、どのエントリが一致したかを示す。このデータは、読み取りタグ（UL、UR、LLおよびLR）の各々について対応するレジスタ550-553の中で格納される。

【０２２７】

例えば、読み取りタグULがミニ・ディレクトリ・エントリ3に一致するならば、制御回路559でデコードされたデータが、ULレジスタ550に格納され、読み取りタグが、ミニ・ディレクトリ・エントリ3と一致したことを示す。下で説明されるように、そのデータが、キャッシュ・ディレクトリ・パイプラインを通され、ULテクセルのためのブロックインデックスが、ミニ・ディレクトリ・エントリ3のためのブロックインデックスを保持するレジスタ5031に格納されていることを示す。読み取りタグUL、UR、LLとLRの組のための一つのユニークなタグだけがミニ・ディレクトリをミスするとき、その読み取りタグを格納するレジスタ550-553の各々にデータがロードされ、対応するテクスチャ・データのためのブロックインデックスが、ミニ・ディレクトリにないことを示す。次のサイクルの間、ミスされたタグを格納するレジスタ550-553の一つからの出力がメイン・ディレクトリ520に対して比較され、読み取りタグのためのブロックインデックスがメイン・ディレクトリから、メイン・ディレクトリのブロックインデックスを格納するレジスタ561にロードされる。ブロックインデックスが、ミニ・ディレクトリのいずれのエントリにも対応しないことを示すデータが、マルチプレクサ555の出力から提供される入力562からレジスタ561にも

格納される。

【０２２８】

先に述べたように、キャッシュ・メモリが4つのインタリーブA－Dを含むので、4つのテクセルを同時にアクセスすることができる。4つのテクセル読み取りタグUL、UR、LLとLRの組は、インタリーブADに任意の態様において対応することができる。どのミニ・ディレクトリ・エントリがテクセルUL、UR、LLとLRの各々に対応するブロックインデックスを格納するかを識別するレジスタ550-553に格納されたデータが、テクセルUL、UR、LLとLRの各々をその対応するインタリーブA-Dに関連させるよう制御されるバレル・シフト装置563を通される。バレル・シフト装置の出力が、それぞれインタリーブA－Dに対応し、各々がミニ・ディレクトリ・エントリを識別し、もしあるならばインタリーブのためのブロックインデックスを格納するインタリーブ・インデックス・コントロール・レジスタ565-568にロードされる。一つのユニークな読み取りキャッシュ・タグだけがミニ・ディレクトリをミスするとき、レジスタ550-553からの出力のシフティング、およびメイン・ディレクトリ520へのアクセスと平行してレジスタ565-568のローディングが生じる。

【０２２９】

上で述べたように、レジスタ565-568にロードされたデータが、あるとしたらどのミニ・ディレクトリ・エントリが対応するインタリーブのためのブロックインデックスを格納するかを識別する。このデータが、複数のインタリーブ・インデックス・マルチプレクサを制御するために使用される。このマルチプレクサは、571で識別され、各インタリーブのための対応するブロックインデックスを、ミニ・ディレクトリ・インデックスレジスター5011-5051の一つおよびメイン・ディレクトリ・ブロックインデックスレジスタ561から選ぶ。複数のインタリーブ・インデックス・マルチプレクサ571は、4つの独立の6対1マルチプレクサを表す。一つのマルチプレクサが各インタリーブに対応し、5つのミニ・ディレクトリ・インデックスレジスタ501I-505Iおよびメイン・ディレクトリ・ブロックインデックスレジスタ561を選ぶ。各インタリーブ・インデックス・マルチプレクサが、同じインタリーブに対応し、どのミニ・ディレクトリ・エントリがそのインタリーブのためのブロックインデックスを格納するかを識別するレジスタ565-568の一つによって制御される。

【０２３０】

インタリーブのためのブロック・インデックスが、いずれのミニ・ディレクトリ・エントリにも見つけられないことを、このデータが示すとき、対応するマルチプレクサが、メイン・ディレクトリ・ブロック・インデックスレジスタ561から提供されるインデックスを選ぶ。このレジスタ561は、ミニ・ディレクトリへのミスに続いてメイン・ディレクトリからのブロック・インデックス読み取りを格納する。インタリーブA－Dの各々のためのブロック・インデックスが、線580-583を通して提供され、上で説明された態様の中でキャッシュSDRAMをアドレスするために使用される。上で説明されたように、読み取りキャッシュ・タグUL、UR、LLおよびLRの組の２つ以上がミニ・ディレクトリをミスするが、ユニークなキャッシュ・タグを一つだけ含むとき、その読み取りタグのためのブロック・インデックスを提供するために、メイン・ディレクトリ520が一度アクセスされるだけである。このプロセスは、4つの読み取りタグの中の任意の2つが一致するかどうか識別するためにコンパレータ526-532の出力を使用する制御回路559でも制御される。4つの読み取りタグの2以上の組が、ミニ・ディレクトリをミスするが、同じキャッシュ・タグを含むならば、対応するレジスタ550-553の各々が、ブロック・インデックスが、どのミニ・ディレクトリ・エントリにも含まれないことを示すために制御回路559によりセットされる。このように、それらの読み取りタグに対応するデータがインタリーブ・インデックスレジスタ565-568を通るとき、各々が、その対応するインタリーブ・インデックス・マルチプレクサ571へ通じるメイン・ディレクトリ・ブロック・インデックス制御レジスタ561を選ぶ。

【０２３１】

制御回路559は、どの読み取りタグ・レジスタ550-553がメイン・ディレクトリに対して比

較されるかを制御するディレクトリ制御レジスタ573をもセットする。レジスタ573は、メイン・ディレクトリに対して比較されるレジスタ550-553のうちの一つを一時に選択するためにマルチプレクサ555を制御する。読み取りタグUL、UR、LL、LRの2以上がミニ・ディレクトリをミスするが、共通のタグを共有するならば、制御レジスタ573が、レジスタの一つであるだけがメイン・ディレクトリに対して比較されなければならないことを示すためにセットされる。このように、4つの読み取りキャッシュ・タグの組がミニ・ディレクトリをミスする一つのユニークなタグだけを含むとき、メイン・ディレクトリが一度だけアクセスされる。

【０２３２】

4つの読み取りキャッシュ・タグ(UL，UR，LL，LR)の組がミニ・ディレクトリをミスする2以上のユニークなタグを含むならば、キャッシュ・ディレクトリ・パイプラインを通上に説明された流れが変えられ、キャッシュ・ディレクトリがビジーになり、読み取りタグの新しい組を次のサイクルで受け取らない。ディレクトリがビジー状態を示すので、ミニ・ディレクトリをミスした読み取りタグを含むレジスタ550-553の各々がメイン・ディレクトリに対して比較されることができ、新しい読み取りタグで上書きされることはない。さらに、ディレクトリ・パイプラインを通る流れが変えられるので、メイン・ディレクトリが、ミニ・ディレクトリをミスした各読み取りタグのためにアクセスされることができ、それらに対応するブロック・インデックスがメイン・ディレクトリからレジスタ501I-505Iまたは561の一つにロードされることができる。パイプラインは、読み取りタグ(UL，UR，LL，LR)の組のためのブロック・インデックスのすべてが、メイン・ディレクトリからの読み取られているか、またはミニ・ディレクトリにすでに存在するようになるまで、レジスタ550-553のいずれかにおけるデータがバレルシフト装置563を通るのを禁止するようにされている。このように、テクセルUL、UR、LLおよびLRの組は、グループとしてそれらの対応するインタリーブと相関している。

【０２３３】

一組の読み取りタグの2以上のユニークなタグがミニ・ディレクトリをミスするとき、ミスされたタグは、直列に処理される。第１サイクル中（すなわち、タグの組がミニ・ディレクトリに対して比較されるとき）、制御回路559が、ミニ・ディレクトリの中のどのエントリを、第１のミスした読み取りタグで置き換えるべきかを判断し、対応するレジスタ550-553にデータがロードされ、そのブロック・インデックスが、そのミニ・ディレクトリ・エントリに格納されることが示される。第１の処理したミス・タグを格納するレジスタ550-553の出力が第２のサイクル中にメイン・ディレクトリ520に対して比較され、メイン・ディレクトリ・ブロック・インデックスレジスタ561が、どのミニ・ディレクトリ・インデックスレジスタ501I-505Iが置き換えられるかを示すデータで更新される。第３のサイクル中に、対応するブロック・インデックスが、レジスタ561から置き換えのために選ばれたミニ・ディレクトリ・エントリに対応するレジスタ501I-505Iにロードされる。

【０２３４】

その後処理されるミニ・ディレクトリをミスしたユニークなタグの各々が、最後のミス・タグまで同じ態様で扱われる。最後のミス・タグは、2つのユニークなタグだけがミニ・ディレクトリをミスしたときは第２のミス・タグであり、第３または第４のミス・タグでありうる。キャッシュ・ディレクトリを通して処理された最後のミス・タグは、まるでミニ・ディレクトリをミスする読み取りタグの組における唯一のユニークなタグであるかのように扱われる。最後のミス・タグの処理が開始するとき、ディレクトリは、ビジー状態を示す信号をアサート解除するので、読み取りタグの新しい組を受け取ることができる。

【０２３５】

最後に処理されたミス・タグについて、制御回路559が、その対応するレジスタ550-553に、そのタグのためのブロック・インデックスがいずれのミニ・ディレクトリ・エントリにも格納されていないことを示すデータをロードする。これは、すべての読み取りタグがミニ・ディレクトリに対して比較される第１のサイクルの間に行うことができ、または他のミス・タグの処理と平行して行うことができる。最後のミス・タグがメイン・ディレクト

リに対して比較されるサイクル中に、レジスタ550-553の中のデータが、バレルシフト装置563をとおり、インタリーブ制御レジスタ565-568にロードされ、そして、ミス・タグのブロック・インデックスがメイン・ディレクトリからメイン・ディレクトリ・ブロック・インデックスレジスタ561にロードされる。最後にディレクトリの最後のパイプライン段で、インタリーブ・インデックス制御レジスタ565-568の出力が、それらの対応するインタリーブ・インデックス・マルチプレクサ571を制御するために使用されるので、最後に処理されたミス・タグのためのインデックスが、メイン・ディレクトリ・ブロック・インデックスレジスタ561から提供され、組の他の読み取りタグの各々のためのブロック・インデックスがその対応するミニ・ディレクトリ・インデックスレジスタ5011-5051から提供される、。最後に処理されたミス・タグのためのブロック・インデックスをメイン・ディレクトリ・ブロック・インデックスレジスタ561からアクセスすることによって、このタグのためのブロック・インデックスがそのミニ・ディレクトリ・インデックスレジスタにロードされるのを待たないことにより、１サイクルが節約されることを理解する必要がある。

【０２３６】

III．この発明のTIMソフトウェア・デーモン

A．総体的な動作

図29は、この発明のTIMソフトウェア・デーモンの動作を示すフローチャートである。流れ図の上部の部分に示されるのは、TIMの装置独立の動作のステップあり、流れ図の下部の部分の中で示されるのは、TIMの装置依存の動作のステップである。また、TIMの装置依存のルーチンに接続するのは、高水準プロセス（上で説明した）のユーザによって要求されたテクスチャ・マップされた画像を描画するグラフィックス・ハードウェア装置660である。グラフィックス・ハードウェア装置660は、任意の数のグラフィックス・ハードウェア装置のであってよい。イラストの簡単のために、グラフィックス・ハードウェア装置が、テクスチャ・データのブロックを格納し、割込みが処理される局部的なキャッシュ・メモリを含むと仮定する。ハードウェア装置は、上で説明したものと同一または同様であってよい。

【０２３７】

動作は、ステップ600でTIMの装置独立の部分から始まる。動作は、関連したプロセス内のユーザ・コマンドに応答してハードウェア・ドライバの一つで最初にコンテキストをつくる第１のコマンドが生成されるそのときに開始する。その時点で、ルーチンがステップ602に進みTIM内のすべてのルーチンおよびデータ構造が初期状態にされる。初期化テップ602は、ステップ604を含み、ステップ604において、TIMがそれを呼び出した特定のプロセスから分離するので、デーモンとしてそのプロセスから独立に走ることができる。言い換えると、ユーザが描画すべき画像をつくったプロセスが対応するグラフィックスAPIおよびグラフィックス・ハードウェア・ドライバを通してTIMを呼び出し、次いで、初期化中にTIMがその発呼プロセスから分離する。

【０２３８】

次いで、ステップ604においてTIMはそのプロセスとは独立に動作を開始する。動作が、ステップ606に進み、TIMがそのデータ構造および装置依存のルーチンを両方とも初期化するので、それらは、作用するために装置独立のルーチンから後に呼ばれることができる。TIMは、動作中にTIMがアクセスするデータを格納するいくつかのデータ構造を含む。これらのデータ構造は、ステップ606の間に初期化にされる。言い換えると、そのようなデータ構造のために割り当てられたシステム・ソフトウェア・メモリの中の記憶ロケーションが、二進ゼロのようなある初期値を備えている。

【０２３９】

データ構造は、TIMの装置独立のデータ構造であるテクスチャ・メモリ・データ構造を含む。このデータ構造は、特定のプロセス内で各コンテキストのために利用可能なテクスチャのすべてのコピー（共用メモリ・データ構造の中で格納されたそれらを除いて）を維持し、それらのテクスチャの対応する優先度およびソフトウェア・テクスチャID（下で説明

する）を格納する。より大きい詳細には下で説明するように、このテクスチャ・メモリ・データ構造は、テクスチャのTIM自身のソフトウェア・コピー、またはグラフィックスAPIおよびグラフィックス・ハードウェア・ドライバで共有されるテクスチャのコピーのための共用メモリ・ロケーションへのポインタを含む。

【０２４０】

この初期化は、TIMに接続された最初のプロセス内の最初のコンテキストが開かれるときに生じる。その時、接続されたハードウェア・ドライバは、TIMをCRT IDおよびデバイスファイルを呼出しのパラメータとして起動する。CRT IDは、そのプロセスに接続され、そのコンテキストの画像を描画する特定のハードウェア装置を表す３２ビット・ワード（二進数）である。CRT IDを使用してTIMが、装置依存のルーチン（その特定のハードウェア装置に関連する）を必要とする装置独立のルーチンを初期化するので、装置独立のルーチンは、装置依存のルーチンを呼ぶよう準備される。特に、TIM変数が、適当な装置依存のルーチン（特定のハードウェア装置に関連する）にポイントするため装置独立のルーチン内の機能ポインタを呼んだので、装置独立のルーチンは、正しい装置依存のルーチンを呼ぶ。この装置独立のルーチン初期化も、ステップ606の中で生じる。

【０２４１】

最初のプロセス内の最初のコンテキストが開かれるとき、ハードウェア・ドライバがTIMに送るデバイスファイルは、そのハードウェア・ドライバに対応する特定のハードウェア装置を表す文字列である。特定のハードウェア装置は、デバイスファイルを通してシステム・ソフトウェア・メモリ内のメモリの連続ブロックにマップされる。ファイルは、グラフィックス装置をマップするために開かれることができる。デバイスファイルが開かれるとき、システムはハードウェア装置を説明するよう求められる。これに応答して、システムはハードウェア装置IDを提供する。ハードウェア装置IDが、TIMにハードウェア・ドライバによって提供されたCRT IDと比較される。装置IDがCRT IDと同一であるならば、ハードウェア装置は、正しくマップされる。同一であるならば、そのハードウェア装置をマップするのに使われたデバイスファイルがCRT IDによって識別した期待されたグラフィックス・ハードウェア装置に対応する。マップされるとき、ハードウェア装置内のレジスタは、ホストコンピュータ・システムのあるアドレス記憶ロケーションに対応する。このように、データが装置依存のルーチンを通してシステム・メモリのアドレス・ロケーションに書かれるとき、ハードウェア装置内の対応するレジスタが書き込まれる。マッピングは、ステップ608で生じる。

【０２４２】

初期化動作ステップ602が、次いでステップ610に進み、示された特定のハードウェア装置（CRT IDによって表す）に関連する適当な装置依存のルーチンが、初期化される。また、ステップ610で、装置依存のルーチンに関連しているデータ構造も初期化される。一つの装置依存のデータ構造が、キャッシュ・ミラー・データ構造を含む。

【０２４３】

このデータ構造に、その割り当てられたシステム・ソフトウェア・メモリロケーションにおいて、基礎をなすハードウェア装置の局部メモリ（キャッシュ・メモリのような）に現在格納されているテクスチャ・データのブロックのすべての記録およびそれらのテクスチャの対応する優先度が、格納される。初期化ルーチンは、次いでステップ612へ進み、テクスチャ転送バッファが作られる。より詳細に下で説明するように、テクスチャ転送バッファは、テクスチャをハードウェア・ドライバからTIMに提供する一つの手段であり、TIMは、それ自身の割り当てられたテクスチャ・メモリ・データ構造にそのテクスチャのコピーを格納することができる。テクスチャ転送バッファの作成は、この分野の技術に熟練した人にとってあきらかなように従来どおりになされる。

【０２４４】

初期化ルーチンは、次いで通信ソケットがつくられるステップ614に進む。上で説明したように、各ハードウェア・ドライバとTIMの間にこのソケットのための仮想回線接続が存在する。当業者に明らかなように、通信ソケットは、2つのソフトウェア・ルーチンが通

信することができる経路である。ソケットの作成は、通信経路から読み取り通信経路へ書込むバッファをつくることを含む。ソケット通して通信を確立する際に、一つのルーチンが、仮想回線関連を通してソケットに通信することを要求する。最初にソケットがつくられ、名前に連結し、次いで、ソケットのクリエイターが、仮想回線接続を求めて聞き始める。その後は、通信している装置が、仮想回線を通してソケットへの接続を要求することができ、その新しい接続を介して通信を始めることができる。

【０２４５】

ソケットがステップ614でつくられたあと、ステップ602の初期化ルーチンが完了し、TIMがステップ616でプロセッサ上を走る独立のデーモンとしてアイドルになる。TIMは、ステップ616の中でアイドルにある間、そのアイドル状態から目ざめる前に3つの事象の一つが生じるのを待つ。それらの事象は、ハードウェア割込み、仮想回線接続、またはソケット通信を含む。ルーチンは、アイドル状態616からステップ618、620および622に進み、それからステップ616でアイドル状態へループしてもどる。

【０２４６】

ステップ618で、TIMは、ハードウェア割込みが生じたかどうかを判断する。上で特定のハードウェア装置の例と関連して説明したように、ハードウェア装置内でテクスチャされマップされた多角形のレンダリング中に、キャッシュ・ミスが生じたとき、割込みが生じる。キャッシュ・ミスが生じるとき、割込み信号が、ハードウェア装置からホストコンピュータに提供される。割込みが生じるならば、ルーチンがステップ624まで進み、割込みが装置独立のルーチンで処理される。割込みを処理するためのTIMの装置独立のルーチンは、図28のフローチャートを参照して下に詳細に説明される。

【０２４７】

割込みを処理するための装置独立のルーチンに加えて、2つの装置依存のルーチンが、割込みを処理するために実行される。それらの装置依存のルーチンは、ハードウェア装置によって必要とされるテクスチャ・ブロックを判断するため、およびキャッシュ・メモリで置き換えるべきブロックを選ぶための装置依存のルーチン642と、ハードウェア装置660の局部的なキャッシュ・メモリ内で選ばれたロケーションにテクスチャ・データの必要とされたブロックを書くための装置依存のルーチン648とを含む。

【０２４８】

ハードウェア割込みが生じないならば、ステップ618で判断されるように、または割込みが、ステップ624の装置独立ルーチンによって処理されたあとに、装置独立ルーチンが、ステップ620に進み、TIMが新しいソケット接続要求が受け取られたかどうか判断する。ハードウェア・ドライバがTIMに情報を通信するために新しいソケットを開けたいとき生じる、新しいソケット接続のための要求が受け取られているならば、ルーチンは、新しいソケットが接続されるステップ626に進む。

【０２４９】

ステップ620中に決定が生じたあと新しいソケット接続のための要求が受け取られていないか、または新しいソケットがステップ626で接続されているならば、ルーチンがステップ622に進み、TIMがソケット通信が生じているかどうか判断する。そうでなければ、ついでルーチンは、TIMがアイドルになるステップ616に戻る。ソケット通信が生じているならば、ルーチンは、ステップ628に進み、TIMがソケット通信を処理する。言い換えると、TIMはソケットに耳を傾け、ソケットをとおって伝えられる情報を受け取る。情報は、データのパケットまたはデータストリームを含んでよい。

【０２５０】

ステップ628から、ルーチンがステップ630に進み、そこで、TIMは、通信が起こったプロセスがまだ存在するかどうか判断する。そのプロセス内であるコンテキストをつくったのち、プロセスがユーザによって終了されているこかもしれない。プロセスがユーザによって終了されているならば、ルーチンがステップ632に進み、終了されたプロセスに関連するテクスチャ（孤児になるテクスチャとして知られる）が処理される。このステップで、装置独立ルーチンが、TIMのテクスチャ・コピー・データ構造から孤児になるテクスチャ

のTIMのコピーを消去する。さらに、ステップ632で、TIMは、通信が送られたソケットを聞くのをやめる。ステップ632の装置独立ルーチンは、プロセスが終了されたとき、装置依存のルーチン654も呼ぶ。より詳細を下で説明するように、装置依存のルーチン654が、現在、ハードウェア装置のキャッシュ・メモリに格納されている、終了されたテクスチャからのテクスチャ・データを含むブロックを無効にする。装置独立ルーチンは、次いでステップ634（上で説明した）にもどる。

【０２５１】

図29の流れ図からよくわかるように、装置に依存するルーチンが初期化され、あるステップまたはTIMの総体的な装置独立の部分のルーチンによって作用のために呼ばれる。たとえば、装置に依存するルーチンが初期化される、装置独立ルーチンのステップ608の間、初期化の装置依存ルーチンのステップ636が呼ばれる。この装置依存ルーチンは、テクセル・キャッシュ・データ構造が初期化されるステップ638の装置依存ルーチンを含む。上で説明したように、この構造が、テクスチャ・データのどのブロックが、ハードウェア装置660のキャッシュ・メモリに現在格納されているかを記録する。装置に依存するルーチン636は、完了するまでハードウェア装置の既存の割込みが処理される装置依存ルーチンのステップ640を含む。

【０２５２】

TIMの装置依存ルーチンは、複数のハードウェア装置からのデータを格納する共有ライブラリを持つ。このように、基礎をなすグラフィックス・ハードウェア装置がシステムに加えられるならば、そのハードウェア装置のための装置依存データが、共有ライブラリに格納される。割込みを処理するためのステップ624の装置独立ルーチンが、割込みを処理するためにステップ642の装置に依存するルーチンを呼ぶ。ステップ642の割込みを処理するためのこの装置に依存するルーチンは、現在のプリミティブを描画するのにハードウェア装置660で必要とされるテクスチャ・ブロックを判断するためにステップ644のルーチンを含む。このルーチンは、ハードウェア装置に関して上で詳細に説明した。上で説明されるように、このルーチンはミスされたブロックのパイプ・タグの読み出し、必要なテクスチャ・データの特定のブロックを判断するためにそれをデコードすることを含んでいる。割込みを処理するためのステップ642の装置依存ルーチンは、ハードウェア装置660の局部的なキャッシュ・メモリ内に置き換えるテクスチャ・ブロックを選ぶためにステップ646の装置に依存するルーチンも含む。ステップ646のルーチンは、図32を参照して説明される。

【０２５３】

割込みを処理するためのステップ624の装置独立ルーチンは、ハードウェア装置660のキャッシュ・メモリの選ばれたブロック・ロケーションににテクスチャ・ブロックを書くためにステップ648の装置に依存するルーチンも呼ぶ。詳細で上で説明したように、テクスチャ・データのブロックが、ハードウェア・メモリの3-Dレンダリング・パイプラインを避けるバスを通してキャッシュ・メモリに書かれる。

【０２５４】

TIMの装置独立の部分が、ソケット通信をステップ628で処理しているとき、その装置独立ルーチンが、装置依存ルーチン650、652および654の任意の一つを呼ぶ。停止状態が生じたとき、装置依存ルーチン650が呼ばれる。上で詳細に説明したように、第１のテクスチャを含むブロックがキャッシュにダウンロードされ、次いで、TIMが第２のテクスチャを同じブロックに詰めるときに、停止状態が生じる。その時点で、装置依存ルーチン650が始められ、ハードウェア装置の動作が停止させられ、修正されたテクスチャ・ブロックがキャッシュ・メモリ内の前のブロックの既存のロケーションに書かれる。

【０２５５】

テクスチャ・データのブロックがハードウェア装置660のキャッシュ・メモリにダウンロードされたとき、ステップ652の装置依存ルーチンが、呼ばれる。ステップ652の装置依存ルーチンが、ダウンロードされたキャッシュ・データ構造の中でテクスチャ・ブロックの優先度を更新する。テクスチャ優先度は、下で詳細に説明される。

【０２５６】
図30は、二つの間の通信の間のハードウェア・ドライバとTIMの動作を図示する部分的にフローチャートになった機能的ブロック図である。TIMと複数のハードウェア・ドライバの間の操作関係は、図3のブロック図に示される。ハードウェア・ドライバの各々が、図30のルーチンに従ってTIMと通信する。図30のルーチンが、三つの次の機能を行うためにハードウェア・ドライバとTIMによって実行される：(1)ハードウェア・ドライバからTIMへテクスチャの優先度を通信する；(2)TIMにハードウェア・テクスチャIDを各新しいテクスチャに割り当てさせるか、またはそのテクスチャのすでにあるハードウェア・テクスチャIDが正しいことを検証する；(3)TIMにまだ持っていないそれ自身のシステム記憶域にテクスチャをコピーさせるグラフィックス・ハードウェア・ドライバの動作を示すフローチャートが、図30の左側に示され、ハードウェア・ドライバの動作のあるものに応答するTIMの機能が図30の右側に示される。
【０２５７】
上で示したように、ユーザが高水準プロセス内でテクスチャマッピングを必要とする、コンテキストをつくるとき、コマンド、およびその特定のプロセスのためにグラフィックスAPI/ハードウェア・ドライバのシステム・ソフトウェア・メモリに格納されているテクスチャのコピーが、プロセスからグラフィックスAPIに、次いでハードウェア・ドライバに渡される。さらにそのコンテキストの画像が、グラフィックス・ハードウェア・ドライバに提供される多角形のような画像成分に分けられる。情報が、グラフィックス・ハードウェア・ドライバに提供されるとき、図30のルーチンが、ステップ700から始まり、グラフィックス・ハードウェア・ドライバは、プロセスが接続される基礎をなすグラフィックス・ハードウェア装置が、選ばれたテクスチャを扱うことができるかどうかを、判断する。グラフィックス・ハードウェア・ドライバが、テクスチャのサイズのような属性を調べ、そのような属性が基礎をなすグラフィックス・ハードウェア装置によって支援されうるかどうか判断することによってこの判断をする。基礎をなすグラフィックス・ハードウェア装置が、レジスタ・サイズ、限られた局部的なテクスチャ記憶容量、バッファ・サイズ、その他のような物理的な制限を持つことがあり、この物理的な制限は、テクスチャの属性またはそれを多角形に適用する手段によって越えられることがありえる。先に述べたように、グラフィックス・ハードウェア・ドライバが、基礎をなすグラフィックス・ハードウェア装置の物理的な制限についての知識を持ち、その知識でステップ700の判断を行う。さらに当業者が理解するように、基礎をなすグラフィックス・ハードウェア装置が、ユーザの要求する機能を実行できないことがある。ステップ700で、ハードウェア・ドライバが、基礎をなすハードウェア装置が、要求されたテクスチャマッピング機能を物理的に扱うことができないと判断するならば、ルーチンがステップ705に進み、要求されたテクスチャマッピング機能が、ハードウェア装置を除外してソフトウェアによって実行される。ソフトウェア・テクスチャマッピング（ハードウェア装置の使用のないテクスチャマッピング）は、当業者によってよく理解されているものであるから、ここで説明しない。
【０２５８】
ステップ700中に、グラフィックス・ハードウェア・ドライバが、基礎をなすハードウェア装置が、要求されたテクスチャマッピング機能を扱うことができると判断するならば、ルーチンはステップ720に進み、グラフィックス・ハードウェア・ドライバがそれぞれの新しいテクスチャ優先度をTIMに知らせる。テクスチャ優先度リストが、優先度の順に対応して処理するために現在利用可能なテクスチャのソフトウェア・テクスチャIDを含む。そのリストは、新しくつくられたテクスチャについて一つのテクスチャのように小さいことがありえる。この優先度リストは、ソフトウェアシステム・メモリの割当領域に格納される。優先度は、ユーザ規定された優先度および内部優先度を含むことができる。内部優先度より優位をとるユーザ規定の優先度は、図34を参照して下に説明するように、テクスチャの使用予測についてのユーザ提供の知識に基づく優先順位である。内部的に、システムは最も最近使用されたテクスチャに、最も高い優先度を与える。新しくつくられたまたは現在アクティブなテクスチャが、下で説明されるようにユーザ規定の優先度、および最

も最近使用された内部優先度の両方に対して最も高い優先度を与えられる。

【０２５９】

ステップ720において、それぞれの新しいテクスチャ優先度がグラフィックス・ハードウェア・ドライバからTIMに知らされる。新しいテクスチャが内部優先度リストに加えられる都度、そのリスト内のテクスチャの優先度のすべてが変わる。こうして新しい優先度がハードウェア・ドライバからTIMに伝えられる。優先度は、ソケット728を通して情報のパケットの形で伝えられ、各パケットが、そのグラフィックス・ハードウェア・ドライバに対応するプロセスのプロセスID、優先度リストの各テクスチャのソフトウェア・テクスチャID、およびこれらのテクスチャの対応する優先度を含む。上で説明したように、プロセスIDおよびソフトウェア・テクスチャIDは、３２ビットのディジタル・ワードである。優先度は、整数として伝えられる。TIMは、この特定のプロセスのために以前に送り込まれたテクスチャを記録している。したがって、TIMはこの優先度リストにテクスチャのいくつかをすでに格納していることがある。TIMは、次いでそのプロセスのために既に持っているテクスチャの優先度を修正する。

【０２６０】

TIMがシステム・メモリのそれ自身の割当領域にパケットで送られたIDによって表されるテクスチャの一つを格納していないならば、TIMは、そのテクスチャを無視する。そのテクスチャIDによって表されるテクスチャは、後にハードウェア・ドライバから提供されるか、または必要とされない。ユーザが特定のテクスチャの使用を終わる特別な場合では、そのテクスチャの優先度が、優先度0としてTIMに知らされ、TIMは、そのテクスチャを自身の割り当てられたシステム・ソフトウェア・メモリロケーション、または共用メモリ・ロケーション（下で説明する）から取り除き、ハードウェア装置の局部的なテクスチャ・メモリからそのテクスチャを取り除くために装置依存ルーチンを起動する。その装置依存ルーチンは、図29のルーチン654である。上で説明された特定のハードウェア装置について、例えば、終了されたテクスチャを含む局部的なキャッシュ・メモリに格納されたデータのブロックは、その装置依存ルーチンによって取り除かれる。TIMの動作は、ステップ725で示される。

【０２６１】

ルーチンが、次いでステップ730に進み、グラフィックス・ハードウェア・ドライバが、新しく要求されたテクスチャのためのハードウェア・テクスチャIDが存在するかどうか判断する。システム・ソフトウェア・メモリの割当記憶域に格納したデータ構造が、ソフトウェア・テクスチャIDおよび対応するハードウェア・テクスチャID（割り当てられるとき）を格納する。ハードウェア・テクスチャIDおよびサブテクスチャ IDは、（ここで説明する）基礎をなすハードウェア装置によって認識できるIDである。上で説明された例の中で、ハードウェア・テクスチャIDは、テクスチャをお互いから見分ける８ビット・ワードである。ハードウェア・ドライバは、データ構造内でソフトウェア・テクスチャIDのアドレスされたロケーションを見、そして、対応するハードウェア・テクスチャIDがリストされるかどうか見るることによって、ハードウェア・テクスチャIDが割り当てられているかどうか判断する。

【０２６２】

ハードウェア・テクスチャIDが割り当てられないならば、ルーチンがステップ740に進み、ハードウェア・ドライバがTIMからハードウェア・テクスチャIDの割り当てを要求する。この要求は、TIMにソケット728を通して情報のパケットで送られる。パケットは、対応するプロセスのためのプロセスＩＤ、ソフトウェア・テクスチャID、およびハードウェア・テクスチャIDが割り当てられないことをTIMに示すハードウェア・テクスチャIDのための値、ならびにベース・マップのS、T座標値によるテクスチャのサイズを含む。ステップ745において、TIMがハードウェア・テクスチャIDをそのテクスチャに割り当て、そのハードウェア・テクスチャIDをソケット728を通してグラフィックス・ハードウェア・ドライバに戻す。

【０２６３】

ハードウェア・テクスチャIDを割り当てる際に、TIMは、グラフィックス・ハードウェア・ドライバによって提供されるS、T座標値を見ることによって、ベース・テクスチャ・マップのサイズをチェックする。テクスチャ・ベース・マップが128x128テクセルより小さいならば、TIMが、新しいテクスチャで占められうるTIM自身の割当られたシステム記憶域に格納された対応するプロセスのために、テクスチャ・データのブロックのいずれかに空の空間があるかどうか、判断する。空の空間が存在するならば、その新しいテクスチャに割り当てられたハードウェア・テクスチャIDが、空の空間を持つブロックに格納されたテクスチャのものと同じであり、新しいテクスチャに別個のサブ・テクスチャIDが割り当てられる。これが、ハードウェア装置についての上の説明で説明され、また図35を参照して下で説明される。TIMの専用のテクスチャ・メモリ・データ構造内の前からあるテクスチャとともに新しいテクスチャが同じブロックに詰められる時点で、停止が生じ、そして、現在キャッシュに格納されているブロックに上書きするよう一つまたは複数のブロックをダウンロードする対応する装置依存ルーチンが生じる。停止(HALT) 状況の発生が、ハードウェア装置とともに上で説明された。HALTを処理するための装置依存ルーチンが図29のステップ650で示される。

【０２６４】

テクスチャIDとサブ・テクスチャIDを割り当てるこの方式は、上で典型的なハードウェア装置に関して説明した特定のテクスチャ配分計画に基づく。しかし、この特定の方式は、典型的な例としてだけのもので、制限的なものではない。上で説明したように、この発明のTIMは、上で説明された実施例とは異なるテクスチャ記憶割当て要求を持つ異なるハードウェア装置で作用することができるからである。

【０２６５】

新しいテクスチャが128x128テクセルを越える大きさであるならば、TIMは新しいハードウェア・テクスチャIDをテクスチャに割り当てる。この時、ステップ745において、TIMは、新しいテクスチャのコピーを格納するために自身のシステム記憶域内に空間を空ける。新しく割り当てられたハードウェア・テクスチャIDが、ソケット728通して復路のパケットによりグラフィックス・ハードウェア・ドライバに戻される。次いでハードウェア・ドライバが、新しく割り当てられたハードウェア・テクスチャIDをテクスチャIDデータ構造リストの中の適当なロケーションに格納する。

【０２６６】

ルーチンは、ステップ740からステップ750に進み、グラフィックス・ハードウェア・ドライバがハードウェア・テクスチャIDが割り当てられているかどうか判断する。ハードウェア・テクスチャIDは８ビット・ワードであるので、256の別個のハードウェア・テクスチャIDだけが利用可能である。したがって、すべての256の別個のハードウェア・テクスチャIDが割り当てられてしまう可能性がある。その場合、TIMは、256のテクスチャのいずれかが他のものより低い優先度を持つかどうか判断する。もしあれば、TIMが、以前に割り当てられた低い優先度のテクスチャのためのハードウェア・テクスチャIDの一つを新しいテクスチャに再割り当てすることができる。すべての256のテクスチャが最も高い優先度テクスチャである場合、新しいハードウェア・テクスチャIDは割り当てられないで、ルーチンがステップ755に進み、テクスチャマッピングがソフトウェアでなされる。

【０２６７】

ハードウェア・テクスチャIDが割り当てられたならば、ルーチンがステップ750からステップ760に進み、グラフィックス・ハードウェア・ドライバがテクスチャがTIMにコピーされる必要があるかどうか判断する。言い換えるとグラフィックス・ハードウェア・ドライバは、TIMが、割当られたシステム・ソフトウェア・テクスチャ・データ構造にすでにテクスチャのコピーを持つか、またはそのテクスチャまたはその一部分が、共用メモリ・ロケーションに格納されているかどうか判断する。TIMがまだそのテクスチャのコピーを格納していないならば、そのテクスチャが、ステップ770でグラフィックス・ハードウェア・ドライバによってTIMにコピーされる。TIMにテクスチャをコピーするための3つの別個の方法がある。これを図33を参照して詳細に説明する。テクスチャのサイズおよび他のパ

ラメータの詳細がTIMにソケット728を通してパケットで送られる。ステップ775で、TIMが次いでテクスチャをコピーし、TIM自身のデータ構造に格納されるべきテクスチャおよびその部分のために、テクスチャは、ハードウェア装置によって認識できるブロックにパックされる。ブロックは、TIMの割当られたデータ構造に次いで格納される。より詳細に下で説明するように、大きいテクスチャおよび大きいテクスチャの部分が、最初にハードウェアによって認識できるブロックにパックされることなく共用メモリ領域に格納され、TIMがテクスチャへのポインタをTIM自身のメモリ・データ構造で共用メモリ領域に格納する。ステップ770の後、ルーチンは終わる。

【０２６８】

ステップ730で、ハードウェア・テクスチャIDが、要求されたテクスチャにすでに割り当てられていると判断されるならば、ルーチンがステップ790に進み、グラフィックス・ハードウェア・ドライバが、前に割り当てられたハードウェア・テクスチャIDが変わっていないことを検証する。グラフィックス・ハードウェア・ドライバは、ソフトウェア・テクスチャIDおよび前に割り当てられたハードウェア・テクスチャIDを、ソケットを通してTIMに知らせることによって、この検証を実行する。TIMは、これに応答して、ステップ795で、ソフトウェア・テクスチャIDおよびハードウェア・ドライバによって送られる対応するハードウェア・テクスチャIDの両方を見て、それをTIMの格納されたIDリストと比較する。ハードウェア・テクスチャIDが変えられている（TIMが、前に割り当てられたハードウェア・テクスチャIDを別のテクスチャに割り当てた場合にありうる）ならば、TIMは、使えるならば、新しいハードウェア・テクスチャIDを割り当てる。したがって、TIMは、ハードウェアIDが変わっていないことを検証するか、または、新しいIDが割り当てられるならば、ソケット728を通して新しいIDをハードウェア・ドライバに知らせる。

【０２６９】

ステップ790から、ルーチンがステップ750に進み、IDが割り当てられたかどうかが判断される。ルーチンは、上で概説されるように進む。

【０２７０】

ステップ760で、TIMがすでにそのテクスチャの格納されたコピーを持つのでテクスチャがコピーされる必要がないならば、ルーチンは完了する。

【０２７１】

図31は、割込みを処理するための装置独立ルーチンのステップを示すフローチャートである。このルーチンは、図29のステップ624で生じる。キャッシュ・ミスが生じた後、ハードウェア装置で割り込みが生じるとき、割込み信号が、ハードウェアからホストコンピュータのプロセッサに提供され、割込みフラグをセットする。そのフラグがプロセッサによって読み取られ、TIMが図29のステップ618で起こされる。

【０２７２】

割込みを処理するための装置独立ルーチンがステップ800で開始し、ステップ802に進み、NOT-HALTEDと呼ばれるソフトウェア変数が偽(false)にセットされる。ルーチンは、次いでステップ804に進み、TIMが変数NOT-HALTEDが真かどうか判断する。ルーチンがちょうど開始し、変数NOT-HALTEDがステップ802で偽に等しくセットされているので、ステップ804の決定への回答は、否定(no)であり、ルーチンはステップ806に進む。変数NOT-HALTEDが偽であるとき、割込みは、まだ処理されていない。

【０２７３】

ステップ806で、MAP-NUMBERと称する変数が-1に等しくセットされる。変数MAP-NUMBERは、実際のマップ番号を表す。このように、変数MAP-NUMBERが-1に等しくないとき、有効テクスチャがその割り込みに関連する。

【０２７４】

ルーチンは、次いでステップ808に進み、割込みを処理するための装置依存ルーチンが呼び出される。この装置に依存するルーチンは、図29のルーチン644である。このルーチンは、ハードウェア装置を停止させること、および特定の多角形を描画するためにハードウェア装置によって必要とされるテクスチャ・ブロックを決定すること、を含む。必要とさ

れるブロックを判断する際に、パイプ・タグが、読み出され、パイプ・タグから、テクスチャID、MAP-NUMBERおよびS,T MSB座標が決定される。この方式は、ハードウェア装置の説明の中で詳細に上で説明されている。装置依存ルーチンは、次いで変数MAP-NUMBERを必要とされたブロックの適当なMIPマップ番号にリセットする。

【０２７５】

ルーチンが、次いでステップ810に進み、変数MAP-NUMBERが-1に等しくないかどうか判断される。ステップ808の装置依存割込みルーチンが変数MAP-INUMBERをハードウェア装置によって必要なブロックのマップ番号にセットするので、ステップ810の決定への回答は、肯定(yes)である。

【０２７６】

ルーチンは、次いでステップ812に進み、ここでTIMは、ステップ808の装置依存ルーチンによって規定されるハードウェア・テクスチャIDによって表わされるテクスチャが、このプロセスのために存在するかどうか判断する。TIMは、そのテクスチャIDに関連するテクスチャのデータ構造を探索することによって、そのテクスチャが存在するかどうか判断する。テクスチャが存在しないならば、ルーチンはステップ814に進む。ステップ814において、装置独立ルーチンが図29のステップ648で示される装置依存ダウンロード・ルーチンを呼ぶ。装置依存ルーチンは、割込みをクリアするよう指示され、変数NOT-HALTEDを真にセットする。ルーチンが、ステップ804に進み、TIMが変数NOT-HALTEDが真かどうか判断する。変数NOT-HALTEDがちょうど真と等しくセットされたので、その判断への回答はイエスであり、ルーチンが完結する。この状況において、テクスチャが終了されていることがありえ、またはテクスチャがつくられたプロセスがユーザによって終了されていることがありえる。

【０２７７】

代わりに、ステップ808の装置依存の割込みルーチンによって見つけられたテクスチャIDによって表されるテクスチャが存在するならば、ルーチンはステップ816に進む。ステップ816で、必要とされたテクスチャが共用メモリ・ロケーションに格納されるかどうか、TIMが判断する。テクスチャが共用メモリ・ロケーションに格納されないならば、ルーチンはステップ818に進む。ステップ818で、TIM自身の割当られたシステム・メモリにおけるデータ構造のブロック・ポインタが、MAP-NUMBER変数によって表わされるテクスチャの特定のMIPマップのはじめのアドレスに進められる。データ構造のブロック・ポインタは、TIM自身のシステム・メモリ・データ構造内に格納された任意のブロックを指すことができる。上でハードウェア装置に関して説明したように、8より大きいマップ番号を持っているマップについては、256x256テクセルより大きいマップのそれぞれがデータの少なくとも一つのブロックをとる。例えば、マップ番号9を持っているマップについては、サイズが512x5l2テクセルであり、そのマップのデータを格納するのにデータの4つのブロック（各ブロックが256x 256テクセル）が必要である。同様に、マップ番号10を持つ1024x 1024テクセルの大きさの一つのマップのデータのすべてを格納するために、16のブロックが、必要である。しかし、マップ番号8以下のマップについては、マップのすべてが、データの2つのブロックにパックされ、この2つのブロックの一つがキャッシュ・メモリの一つのバンクに格納される。そして、この2つのブロックの他方がキャッシュ・メモリの他方のバンクに格納される。このメモリ配分計画は、図17-19に関連して上で説明した。

【０２７８】

ルーチンがステップ820に進み、TIMがマップ番号が8を超えるかどうか判断する。TIMは、変数MAP-NUMBER（マップ番号と等しい）を見ることによってマップ番号が8を超えるかどうか判断する。マップ番号が8を超えるならば、ルーチンはステップ822に進み、メモリ・データ構造のブロック・ポインタが、S,T MSB座標を使用してハードウェア装置によって必要とされる特定のMIPマップのデータ部分を格納する特定のブロックのはじめに進められる。図17-19に関して上で説明したように、必要な特定のMIPマップがマップ番号9であるならば、4つの異なるブロックが、そのマップからのデータを格納する。このように、一つのS MSB座標と一つのT MSB座標が、4つのブロックのどれが、ハードウェア装置によ

って要求されるマップ9のテクスチャ・データ部分を格納するか判断するのに必要である。それゆえに、一つのS MSB座標と一つのT MSB座標を使用して、ブロック・ポインタが、マップの4つのブロックの適当なものに進められる。

【０２７９】

必要とされたMIPマップがマップ番号10を持つならば、16の異なるブロックが、そのマップのデータを格納するために存在する。このように、16のブロックのどれがハードウェア装置が必要とするマップのテクスチャ・データ部分を含むかをデコードするのために、2つのMSB S座標と2つのMSB T座標が必要である。SおよびT MSB座標ビット使用して、適当なブロックが決定され、ブロック・ポインタがそのブロックに進ませられる。

【０２８０】

ステップ820で、マップ番号が8以下であることが判断されると、ルーチンはステップ824に進む。ステップ824で、ブロック・ポインタは、もしもT MSBビットがセットされているならば8以下のマップ番号を持つマップのテクスチャ・データを格納する2つのブロックの一つから他方のブロックに動かされる。上で説明したように、8以下のマップ番号を持つマップのテクスチャ・データのすべてが2つのブロック内に格納される。このように、2つのブロックのどちらが、ダウンロードするに適当なブロックであるかをデコ―ドするために一つのビットだけが必要である。このように、8以下のマップ番号を持つマップの任意のものについて2つのブロックのどちらが、適当なブロックであるかデコ―ドするために、T MSBビットが使用される。

【０２８１】

ステップ816で、必要なテクスチャが、システムの共用メモリ・ロケーションに格納されていることがはんだんされるならば、ルーチンはステップ826に進み、マップ番号が8以下かどうかが判断される。マップ番号が8以下であるならば、ルーチンがステップ828に進み、共用メモリ・データ構造のブロック・ポインタが必要なマップを格納する2つのブロックのスタートに進められる。次いで、ルーチンがステップ824に進み、上で説明したように、T MSBビットがセットされているならば、ブロック・ポインタが、ブロックの一方から他方に進められる。

【０２８２】

代わりにステップ826で、マップ番号が8を超えることが判断されるならば、ルーチンがステップ830に進み、ブロック・ポインタが必要な特定のMIPマップのテクスチャ部分を格納する複数のブロックの最初のブロックに進められる。先に説明したように、必要なマップが9のマップ番号を持つならば、4つの異なるブロックの一つが必要なブロックである。必要なマップがマップ番号10を持つならば、16の異なるブロックの一つが必要な特定のブロックである。

【０２８３】

ルーチンが次いでステップ832に進み、ブロック・ポインタが、ステップ822で行われたのと同様にS、T MSB座標ビットを使用して共用メモリ・データ構造内の必要な特定のブロックに進められる。ダウンロードされることを要求される特定のブロックを指すために適当なブロック・ポインタが進められるステップ822、824および832の各々から、ルーチンは、ステップ834に進む。ステップ834で、装置独立ルーチンが、必要なブロックをダウンロードし、これ以上割込みがないならば割込みをクリアするために装置依存ダウンロード・ルーチンを呼ぶ。ブロックをダウンロードするための装置依存ルーチンは、図29のステップ648で示されるダウンロード・ルーチンである。

【０２８４】

ブロックをダウンロードするためのこの装置依存ルーチンが、ステップ822、824、または832で実行されるように、ポインタによって指されたブロックをダウンロードする。上で詳細に説明したように、この例のハードウェア装置のために、ハードウェア装置の3-Dパイプライン・レンダリング経路を避け、2-D経路をとおしてブロックが、ダウンロードされる。その特定のブロックは、キャッシュ・メモリの選ばれたロケーションにダウンロードされる。この選ばれたロケーションは、そのときキャッシュ・メモリに格納されていた

64のブロックの上書きする一つを含む。バンド幅節減のために将来の割込みを避けるため上書きする最適のブロックは、図29のステップ646で示され図32のフローチャートを参照して説明した装置依存ルーチンで判断される。

【０２８５】

図31のルーチンは、ステップ834からステップ804に進み、変数NOT-HALTEDが真かどうか判断される。そうであるならば、ルーチンは完了する。ブロックがダウンロードされこれ以上割込みが存在しないことが判断されたあと、ステップ834の装置依存ダウンロード・ルーチンが、変数NOT-HALTEDを真にセットする。これが、割込みをクリアして、ルーチンを終わる。

【０２８６】

ステップ810で、マップ番号がー１に等しくないと判断されるならば、上で説明したようにルーチンがステップ814に進む。

【０２８７】

図32は、割込みが生じるときハードウェア装置の局部キャッシュ・メモリのどのブロックを置き換えるか判断するためにTIMによって実行される装置依存ルーチンのブロック図である。この装置依存ルーチンは、図29のステップ646として示される。上で図26を参照して説明したように、割込みが処理されるとき装置依存ルーチンが、ステップ624の装置独立ルーチンによって呼ばれる。キャッシュ・メモリの中で置き換えるブロックを選ぶ方式は、そのときキャッシュ・メモリにあるすべてのブロックの過去の使用頻度、およびブロックに格納されたテクスチャの相対的優先度に基づくそのようなブロックの将来の使用予測の両方を考慮する。この方式は、システムのバンド幅を減らすため将来の割込みの番号を減らすことを狙う。

【０２８８】

図32を参照して説明されたルーチンは、上で説明された特定のハードウェア装置の使用を仮定する。このように、下で詳細に説明されるように、ルーチンは、ハードウェア装置の最も最近使用された（MRLT）および現在使用される（CU）レジスタを更新する。しかし、図32のルーチンは、この発明のシステムが使用される任意の基礎をなすハードウェア装置に適合するよう変えられてもよいことを理解する必要がある。例えば、装置は、説明した例のものと全く同じ物理的構造を持つ必要はないが、システム・メモリに格納されたテクスチャ・データの一部を格納することができる必要があり、画像を描画するためにそのとき必要とされるテクスチャ・データが局部メモリにないとき、割込みを出す必要がある。

【０２８９】

図32のルーチンは、割込みがハードウェア装置に生じ、信号がハードウェア装置からプロセッサに送られ、この信号が読み出されてTIMが起こされ、TIMが図29のステップ624の装置独立ルーチンからこのルーチンを呼ぶとき、ステップ900で始まる。図32のルーチンは、ステップ902に進み、そこでTIMがハードウェア装置からの各SDRAMバンクに対応するMRUおよびCUレジスタを読みとる。これらのレジスタが、キャッシュにおけるテクスチャ・データのどのブロックが最も最近使用された（MRU: most recently used）か、または現在用いられている(CU: currently in use)かに関する情報を得続ける。上で説明されたように、レジスタの各々が、３２ビット・レジスタであり、対応するバンクにおいて一つのビットが32のブロックの各々に対応する。対応するブロックが使用された（すなわち、テクセルがキャッシュのそのブロックからアクセスされた）とき、MRUレジスタの中の１ビットが、アサートされ、対応するブロックがミニ・ディレクトリにあるとき（上で説明した）、CUレジスタの中の１ビットがアサートされる。

【０２９０】

以上のハードウェアの説明で述べたように、連続ピクセルがしばしば同じ一つまたは複数のブロックに格納されたテクセルにマップし、プリミティブ（すなわち、三角形のような多角形の部分）のピクセルを描画するときに、ブロックの同じ組に繰り返して使用することが要求され。このように繰り返される割込みを避けるために、最近使用されたブロックは、続くピクセルを描画するときにそのブロックをアクセスする必要があるかもしれない

ので、置き換えないことが望ましい。次のピクセルを描画する際にハードウェア装置によって必要とされる見込みが小さいまたは最小のブロックを置き換え用に選ぶことによって、システムのバンド幅が最小にされる。

【０２９１】

MRUおよびCUレジスタを読み出したあと、ルーチンがステップ904に進み、書き込むべきキャッシュ・メモリのSDRAMのバンク0またはバンク1の一つが選ばれる。どのバンクに書き込むかの決定は、ハードウェア装置と関連して上で説明した方法に従ってなされる。ルーチンが、次いでステップ906に進み、変数BLOCK NUMBERがマイナス１に等しくセットされ、下で説明されるように繰返しが完結し、ブロックが置き換えのために選ばれたとき、指示を出すために使用される。変数BLOCK NUMBERは、置き換えのために選ばれた実際のブロック番号を表す。

【０２９２】

ルーチンが、次いでステップ908に進み、変数BLOCK NUMBERがマイナス１に等しいかどうか判断される。第１の繰返しでは、ちょうどセットされたばかりなので回答は、イエスで、ルーチンがステップ910に進み、どのバンクがステップ904での書き込みに選ばれたかが判断される。書き込まれるバンクがバンク0であるならば、ルーチンはステップ912および914に進む。代わりに、書かれるバンクが、バンク１であるならば、ルーチンはステップ916および918に進む。各バンクのための2対のステップが同様の操作を実行する。操作は、最も最近使用されたかまたは現在使用されているブロックを利用不能にするよう設計されている。これらのブロックは、そのときMRUおよびCUレジスタでアサートされているビットに対応する。

【０２９３】

テクスチャ・ハードウェア・メモリの説明の中で述べたように、キャッシュは、最高64のブロックのテクスチャ・データを保持することができる。各ブロックは、ブロック・タグによって識別される。すべての64のブロックは、置き換えのための可用性についてチェックされる。先に述べたようにキャッシュ・メモリのバンク0が、ブロック0ないし31を保持し、バンク１がブロック32ないし64を保持する。ステップ912で、ブロック0が、置き換え可用性について調べる際の開始ブロックに指名され、ブロック31が、バンク0のブロックの置き換えのための可用性について調べる際の終わりのブロックに指名される。これが、変数START BLOCKをゼロにセットし、変数END BLOCKを31にセットすることによって達成される。同様に、ステップ916で、バンク１について、変数START BLOCKが、32に等しくセットされ、変数END BLOCKが63に等しくセットされ、バンク１のブロックの置き換えのための可用性チェックにおいて、開始ブロックが、ブロック32で、終了ブロックがブロック63であることが示される。

【０２９４】

バンク0およびバンク１の各々について、変数BLOCKS TO USE（使用するブロック）は、３２ビット・ワードであり、１ビットがそのバンクの各ブロックに対応し、ブロックのいずれかが置き換えのため使用可能であるかどうかを表す。BLOCKS TO USE変数のビットの一つがアサートされるならば、そのバンク内の対応するブロックが置き換えに使用可能である。

【０２９５】

バンク0が選ばれるならば、ルーチンがステップ912からステップ914に進み、変数BLOCKS TO USEが変数BLOCKS TO USE BANKS ZERO（バンク0を使うブロック）に等しくセットされ、次いで、バンク0のMRUまたはCUレジスタのいずれかでアサートされたビットに対応するBLOCKS TO USE変数内のビットがアサート解除される。ここで、またはは、論理和演算である。同様に、一つのバンクが置き換えのために選ばれるならば、ルーチンがステップ916からステップ918に進み、変数BLOCKS TO USEが変数BLOCKS TO USE BANK ONE（バンク１を使うブロック）に等しくセットされ、バンク１のMRUまたはCUレジスタでアサートされたBLOCKS TO USE変数内のビットがアサート解除される。

【０２９６】

ステップ914または918のいずれかから、ルーチンがステップ920に進む。ステップ920で、変数AVAILABLE BLOCKSと変数BLOCK PRIORITYが両方ともゼロにセットされる。このステップ920は、そのバンクに使えるブロックが存在するかどうか判断する際、各ブロックの繰り返しごとに一度訪問される。変数AVAILABLE BLOCKS（使用可能ブロック）は、置き換えに使えるバンク内のブロック数えるために使用される。そのバンク内の使えるブロックの数が、下で説明されるように、ルーチンがどのように流れるかに影響する。置き換えに使えるブロックの最下位優先度を選ぶために変数BLOCK PRIORITY（ブロック優先度）が、そのバンク内の使えるブロックの優先度をモニタするのに使用される。

【０２９７】

ステップ922-936は、一連のステップである使えるブロックの数および使えるブロックの最下位優先度を判断するために各バンク内の各ブロックのために繰り返される。ステップ920から、ルーチンがステップ923に進み、変数START BLOCK（ブロック開始）が変数END BLOCK（ブロック終了）以下かどうかは判断される。変数START BLOCKが変数END BLOCK以下であるならば、そのバンク内のブロックのすべてが、置き換え可用性のためにチェックされてはいない。ステップ922-936のシリーズの各繰返しの間、変数START BLOCKが、ステップ934で１ずつ増加させられ、現在チェックされているブロックを表す。START BLOCKの方がEND BLOCKより大きいとき、ルーチンが下で説明するステップ938に進む。

【０２９８】

変数START BLOCKが変数END BLOCK以下であるならば、そのバンクのブロックは、置き換え可用性のためにチェックする必要がなく、ルーチンはステップ924に進む。ステップ924で、BLOCKS TO USE（使用するブロック）変数においてチェックされているそのブロックに対応するビットが１にひとしいかどうか判断される。現在チェックされているブロックに対応するBLOCKS TO USE変数の中のビットが１に等しいならば、そのブロックが置き換えのために使え、ルーチンは、ステップ926に進み、変数AVAILABLE BLOCKS（使用可能ブロック）が、１だけ増加される。

【０２９９】

ステップ926から、ルーチンはステップ928に進み、変数START BLOCKに対応するブロックの優先度である、現在使えるブロックの優先度が、これまでステップ922-936の繰返しによって使用可能として見つけられたブロックの最下位優先度より大きいか小さいか判断される。使えるブロックの最下位優先度は、変数BLOCK PRIORITYに保持される。これまで使用可能なブロックの最下位優先度より現ブロックの優先度が小さいならば、ルーチンはステップ928からステップ930に進み、変数BLOCK NUMBER（ブロック番号）が変数START BLOCKに等しくセットされるので、変数BLOCK NUMBERは、これまでの使えるブロックの最下位優先度のブロック番号を格納する。

【０３００】

ルーチンは、次いでステップ932に進み、変数BLOCK PRIORITYが変数START BLOCKによって表される現ブロックの優先度にセットされる。ステップ928で、現在使えるブロックの優先度（変数START BLOCKによって表わされる）が、これまでの使えるブロックの最も低い優先度（変数BLOCK PRIORITYによって表わされる）より低いことが判断されるならば、ステップ930および932がバイパスされ、ルーチンはステップ934に進む。そのような場合、変数BLOCK NUMBERも変数BLOCK PRIORITYも更新されない。

【０３０１】

ステップ924で、可用性のためにチェックされている現ブロックに対応する変数BLOCKS TO USEのビットが、１に等しくないならば、そのブロックは、置き換えのために使えない。そして、ルーチンはステップ934に進む。ステップ934で、変数START BLOCKが、１だけ増加されるので、そのバンク内の次のブロックが可用性について調べられる。ルーチンは、次いでステップ936に進む。ステップ936において、変数BLOCKS TO USEが、可用性について調べるべき次のブロックに対応する次のビットに進められる。次いで、ルーチンは別の繰返しのためにステップ922に進む。

【０３０２】

そのバンク内のすべてのブロックが、可用性のためにチェックされたとき、変数START BLOCKが変数END BLOCKより大きく、ルーチンがステップ922から「1」に進む。これが図32に示される。「1」から、ルーチンはステップ938に進み、変数AVAILABLE BLOCKSがゼロに等しいかどうか判断される。変数AVAILABLE BLOCKSがゼロに等しくないならば、ルーチンがステップ940に進み、変数AVAILABLE BLOCKSが10以下かどうか判断される。変数AVAILABLE BLOCKSが10を超えるならば、ルーチンは「2」に進み、次いでステップ908に進む。ステップ908で、変数BLOCK NUMBERが－1に等しいかどうか判断される。10以上のブロックが使えるとわかったので、そのうちの一つが最下位優先度を持ち、変数BLOCK NUMBERが、使えるブロックの最下位優先度のブロック番号と等しく、その判断への回答は、ノーである。

【０３０３】

したがって、ルーチンはステップ942に進み、置き換えられるブロックがバンク0またはバンク１にあるかどうか判断される。ブロックがバンク0にあるならば、ルーチンがステップ944に進み、置き換えのために選ばれたブロック（変数BLOCK NUMBERによって表わされたブロック）に対応するそのバンクの変数BLOCKS TO USE内のビットがアサート解除される。置き換えられるブロックがバンク0にあるならば、ルーチンがステップ946に進み、ステップ944で行われたのと同様の機能が行われる。次いで、使えるすべてのブロックの中で最下位の優先度を持つ変数BLOCK NUMBERによって表されるブロックが置き換えのために選ばれるので、ルーチンはステップ980で完了する。以上のことから、変数AVAILABLE BLOCKSによって表わされる使えるブロックのすべては、最近使用されなかったまたは現在使用されていないもの、および置き換えのために選ばれたブロックが、最小優先度のテクスチャを格納するブロックであるものであることを理解する必要がある。

【０３０４】

ステップ940で、使えるブロックの数が10以下であるならば、ルーチンがステップ948に進み、使えるブロックがバンク0またはバンク１にあるかどうか判断される。バンク0にあるならば、ルーチンがステップ950に進む。バンク１にあるならば、ルーチンはステップ952に進む。ステップ950および952で、ハードウェア・レジスタMRU（バンク0および１についてそれぞれ一つ）がゼロにリセットされる。950および952のステップによって、そのバンクについて使えるブロックの数がほんの少しの状況においてそのバンクのMRUレジスタがリセットされる。このようにされるので、MRUレジスタのビットはそのすべてが、アサートされるわけではなく、将来の割込みのために、ランダムに置き換えるブロックを選ばなければならない。ステップ950または952のいずれかから、ルーチンが上で説明した再びステップ908から始まる「2」に進む。

【０３０５】

ステップ938で、そのバンクについて使えるブロックの数がゼロに等しいことが判断されるならば、ルーチンがステップ954に進み、そのバンクにとってルーチンがこのステップに来る最初であるかどうか判断される。そうであれば、ルーチンがステップ956に進み、バンク0またはバンク1のブロックをチェックしているか判断される。バンク0であれば、ルーチンはステップ958に進む。バンク1であれば、ルーチンはステップ960に進む。ステップ958または960のいずれかにおいて、MRUレジスタまたはCUレジスタ（そのバンクの）のいずれにおいてもアサートされていないビットに対応する変数BLOCKS TO USE（対応するバンクのための）の中のビットがアサートされる。言い換えると、ステップ958および960で、そのバンクのためのMRUまたはCUレジスタにおいてアサートされたビットによってそのとき表されていないブロックに対応する変数BLOCKS TO USEのビットが1に等しくセットされる。この操作は、対応するバンクのMRUまたはCUレジスタにおいてそのときアサートされているビットによって表されるすべてのブロックについてBLOCKS TO USE変数の中のビットをアサート解除する、ステップ914と918の中で行ったものと異なる。ステップ958と960の操作は、そのバンクでより多くのブロックを使えるようにするための試みである。ステップ958または960のいずれかから、ルーチンが「2」に進み、ルーチンが再びステップ908から始まり、使えるブロックが存在するかどうかチェックするための別の繰返しが行われる。

【０３０６】
繰返しが再び生じ、そのバンクについて使用可能なブロックがないことが判断されるならば、ステップ954において、そのバンクについてルーチンがそこに来たことがあるかどうかの判断に対する回答は、イエスであり、ルーチンはステップ962に進む。ステップ962において、ブロックがバンク0のまたはバンク1でチェックされているかどうか判断される。バンク0であれば、ルーチンがステップ964および966に進む。バンク1であれば、ルーチンはステップ968および970に進む。ステップ964および968において、対応するバンクのためのMRUレジスタのビットがゼロにリセットされる。次いで、ステップ966および970において、対応しているバンクのためにそのバンクのCUレジスタのアサート解除されたビットに対応する変数BLOCKS TO USEのビットがアサートされる。次いで、ルーチンが「2」に進み、再びステップ908から始まる。CUレジスタでアサート解除されるビットをBLOCKS TO USEレジスタにアサートすることによって、ステップ966および970において、ルーチンは、次の繰返しのため十分な数の使用可能なブロックを提供しようとする。ステップ966または970のいずれかから、ルーチンが「2」に進み次の繰返しが始まる。

【０３０７】
図34は、ソフトウェア・テクスチャ・データを、特定のプロセスのグラフィックスAPI/グラフィックス・ハードウェア・ドライバからTIMに移すことができる異なる方法を示すグラフィック・イラストである。図の左上部分に示されるのが、特定のプロセスのグラフィックスAPI/ハードウェア・ドライバ850である。グラフィックスAPl/ハードウェア・ドライバ850は、そのプロセス内でつくられたテクスチャのコピーを格納するのためにそれ自身の割当られたシステム・ソフトウェア記憶域851を持つ。領域851内に、図34の例は、格納されつつあるテクスチャT1およびT2、ならびに共用メモリ・テクスチャT3へのポインタ860を示す。この例では、テクスチャT1の大きさが256x256テクセル、テクスチャT2の大きさが16xl6テクセル、そして、テクスチャT3の大きさが1024xl024テクセルである。

【０３０８】
図表の右上部分に示すのが、この発明のTIM 160である。TIM 160は、それ自身のデータ記憶領域854を含み、ここに、あるテクスチャのコピー、またはその一部分が格納される。図に示すように、データ記憶領域854内に格納されるのは、ブロック870、871、872、873、およびポインタ860である。ブロック870および871内にあるのは、テクスチャT1およびT2の部分である。テクスチャTIの大きさが256x256テクセルであり、テクスチャT2の大きさが128xl28テクセル未満であるので、テクスチャT1が、2つのブロック870および871の中に割り当てられ、そして、それらのブロックの中にテクスチャT2を格納する空の空間が存在する。こうして、TIMがテクスチャT2をテクスチャT1と同じブロックにパックし、ハードウェア装置に関連して上で説明し図36に関して下で説明するように、テクスチャT2にサブ・テクスチャ識別子を割り当てる。

【０３０９】
ブロック872と873は、テクスチャT3の部分を含む。それらの部分が、テクスチャT3のMIPマップ・レベル8およびそれ以下を含む。先に述べたように、特定のテクスチャの8以下のマップ番号を持っているマップが、２つのブロックの間で割り当てられる。図表の左下の部分に示されるのが、共用メモリ領域852であり、図34の例では、テクスチャT3を格納する。グラフィックスAPI/ハードウェア・ドライバのデータ記憶領域851が、共用メモリ領域852内のそのテクスチャへのポインタ860を含み、TIMのデータ記憶領域854は、テクスチャT3格納されている共用メモリ852内のそのロケーションへの別のポインタ860も格納する。

【０３１０】
この発明のソフトウェア方式は、テクスチャ・データをグラフィックスAPI/ハードウェア・ドライバからTIMに転送することができる3つの異なる方法を提供する。第１の方法は、共用メモリ転送バッファ856を通してデータのブロックを転送するものである。この方法において、ハードウェア・ドライバがマップをハードウェア装置によって認識できるブロックに分ける。この特定の例では、それらのブロックは、それぞれ256x256テクセルの大

きさである。ひとたび分けられると、ハードウェア・ドライバは、パケットをソケット858を通してTIMに送る。パケットは、テクスチャ・データの一つまたは複数のブロックが共用メモリ転送バッファ856を通してやってくることのインジケータを含む。パケットは、ソフトウェア・テクスチャID、テクスチャが関連させられるプロセスＩＤ、S、Tテクスチャ・ベース・マップ・サイズ、そのテクスチャのMIPマップ・レベルの数、および通信されるブロックの数をも含む。発明の一つの実施例において、ソフトウェア・テクスチャIDが３２ビット・ワードであり、そしてプロセスＩＤも３２ビット・ワードである。ソフトウェア・テクスチャIDが、特定のプロセス内のテクスチャをお互いから識別し、そして、プロセスＩＤがTIMによって管理されるプロセスをお互いから識別する。TIMが、この情報を次いで受け取り、共用メモリ転送バッファからのテクスチャ・ブロックの読み出しに備える。

【０３１１】

8より大きいマップ番号を持つマップについて、各ブロックが、特定のマップからだけのデータを含め共用メモリ転送バッファを通して移される。これらのブロックのそれぞれについて、TIMが、次いでブロックをコピーし、そのブロックをそれ自身のデータ記憶領域854に格納する。8のマップ番号を持っているマップについては、マップ全体が一つの256x256テクセル・ブロック内で送られる。TIMが、次いでそのブロックをコピーし、マップの部分を2つのブロックの間でハードウェア装置のために最適のフォーマットに割り当て、そして、それらのブロックをそのデータ記憶記憶域854に格納する。マップ番号7以下のマップについては、マップのすべてが、ハードウェア・ドライバによって一つのブロックにパックされ、そして共用メモリ転送バッファ856を渡って送り出される。TIMが、ブロックを転送バッファから受け取り、先に述べたようにキャッシュ・メモリのSDRAMの2つの別のバンクに続いて記憶させるために、テクスチャ・データを2つのブロックの間で割り当てる。上の例で示した以外の別のハードウェア装置を使用するならば、TIMがマップ番号8以下のマップをハードウェア装置の特定の局部メモリによって認識できるパターンにパックする。

【０３１２】

図34の例で示すように、256x256テクセルの大きさのテクスチャTlが、共用メモリ転送バッファを通して一つのブロックとして伝えられる。次いでTIMが、そのブロックをコピーし、テクスチャT1をハードウェア装置によって認識できる2つのブロック870および871の間で割り当てるので、それらのブロックは、キャッシュ・メモリのSDRAMの別のバンクにダウンロードされることができる。次いで、それらのブロック870と871は、TIM 160のデータ記憶メモリロケーション854内に格納される。

【０３１３】

グラフィックスAPI/ハードウェア・ドライバからTIMにテクスチャ・データを転送することができる第２の方法は、テクスチャをソケット858を通して送るものである。この発明の一つの実施例において、テクスチャは、それらが64x64テクセル以下の大きさのときソケットを通して送られる。テクスチャ・データがソケットを通して送られる前に、ハードウェア・ドライバが、パケットをソケット858を通してTIMに送り、TIMに、テクスチャ・データがソケットを通して転送されることを示す。パケットは、そのテクスチャのベース・サイズを表すS、T座標、テクスチャのソフトウェア・テクスチャID、そのテクスチャに対応するプロセスＩＤ、そのテクスチャ内にいくらのマップ・レベルがあるかの指示、およびバイトの数の観点からのテクスチャ・データの物理的なサイズ、を含む。TIMは、この情報をソケットからパケットを読み出すことによって受け取り、テクスチャ・データの読み出しに備える。次いで、テクスチャ・データが、データストリームの形でハードウェア・ドライバからソケットを通して伝えられる。TIMは、次いでソケットからテクスチャ・データを読み取り、テクスチャ・データをハードウェア装置によって認識できるブロックにパックする。

【０３１４】

図34で示した例において、16x 16テクセルの大きさのテクスチャT2がソケットを通して送

られ、TIMがテクスチャT2を読み取り、そのテクスチャT2を2つのブロック870および871の間でパックする。これらのブロックは、テクスチャT1の一部分をも格納する。ハードウェア装置例と関連して先に述べたように、テクスチャT2は、次いでテクスチャT1と同じテクスチャIDを割り当てられるが、テクスチャT1と区別するために異なるサブ・テクスチャIDが用いられる。サブ・テクスチャIDは、ブロック内のサブ・テクスチャのロケーションを識別する2つのS座標ビットと2 T座標ビットを含む4ビットのワードである。テクスチャ・データをグラフィックスAPI/ハードウェア・ドライバからTIMに伝えることができる第３の方法は、テクスチャ・データを、グラフィックスAPI/ハードウェア・ドライバおよびTIMによってアクセスされることができる共用メモリ・ロケーションに格納するものである。ハードウェア・ドライバおよびTIMは、共用メモリ領域852に格納されたテクスチャのアドレスされたロケーションへのポインタをそれぞれ格納する。テクスチャが共用メモリ領域852に格納されることになっているとき、ハードウェア・ドライバが、テクスチャが、共用メモリ領域852に格納されることをTIMに示すパケットをソケット858を通してTIMに送る。また、パケットが共用メモリIDと等しいソフトウェア・テクスチャID、プロセスＩＤ、そのテクスチャのS、Tベース・マップ・サイズ、およびそのテクスチャのMIPマップ・レベル、を含む。TIMが、次いでパケットを読み出し、共用メモリIDをポインタに変換し、そのポインタをそれ自身のデータ記憶領域854に格納する。ポインタは、共用メモリ領域に格納された特定のテクスチャの開始アドレスである共用メモリ領域内のアドレス・ロケーションを指す。

【０３１５】

TIMは、また、その特定のテクスチャのためのMIPマップ・レベルが、マップ8以下を含むかどうか判断する。テクスチャが、MIPマップ・レベル8以下を含むならば、次いで、TIMがその特定のテクスチャのMIPマップ・レベル8以下だけをコピーし、上で概説された配分計画に従って2つのブロックにそれらをパックする。次いで、そのテクスチャのためのMIPマップ・レベル8以下を含むテクスチャ・データのパックされたブロックが、TIM自身の記憶域854に格納され、一方、マップ・レベル9以上は、共用メモリ領域852に格納される。TIMは、それ自身の領域854内においてポインタを、そのテクスチャのマップ・レベル9以上の開始アドレスに格納する。

【０３１６】

図34で示された例が、この方式をよく表している。テクスチャT3が1024xI024テクセルのベース・テクスチャ・マップ・サイズを持つと仮定する。そのサイズのため、領域851と854に格納されつつあるテクスチャの2つの別のコピーを持つのではなく、テクスチャが、共用メモリ領域852に格納されなければならないと決定される。テクスチャT3がMIPマップ・レベル8ないしゼロを含むならば、TIMがテクスチャT3のマップ8ないしゼロをコピーし、それらを2つの別のブロック872および873にパックする。ブロック872および873は、TIMの割当られたシステム記憶域854に、次いで格納される。先に述べたように、TIMは、テクスチャT3が格納されている共用メモリ852内の最初のアドレスを指すポインタ860をも格納する。このように、マップ番号9および10を持つマップがテクスチャT3のために必要であるとき、TIMは、共用メモリ・ロケーション852へのポインタ860を使用してそのデータをアクセスすることができる。

【０３１７】

TIMがそれ自身の記憶域854に格納されたテクスチャT3のマップ9および10のコピーを持たないので、テクスチャT3のマップ9および10は、割込みが生じるまでハードウェア装置によって認識できるブロックに分割されない。このように、テクスチャまたはその一部分が共用メモリ領域852に格納されるとき、システムのバンド幅の観点からのわずかな犠牲が、生じる。しかし、相当なシステム・メモリ空間節減が、テクスチャまたはその一部を共用メモリ領域に格納することによって生じる。このように、テクスチャまたはその一部を共用メモリ領域に格納するかどうか決定する際、バランスがメモリ節減とシステム性能との間にとられる。この発明の一つの実施例において、1024x 1024テクセル以上の大きさのベース・マップを持っているテクスチャがシステムの共用メモリ領域852に格納される。

システムのユーザが、しかしテクスチャを共用メモリ領域に格納するためのカットオフ・サイズとして、任意のサイズのテクスチャを選ぶことができる。

【０３１８】

図35は、テクスチャ優先度が特定のプロセスのためにどのように規定されるかを示す機能ブロック図である。図表の先頭の行に示されるのが、ユーザ・アプリケーションまたはプロセス1030である。その下にグラフィックスAPI 1032およびグラフィックス・ハードウェア・ドライバ1034が示されるハードウェア・ドライバの下にTIM 1036が示される。プロセス1030において、ユーザは、規定されたテクスチャのためにテクスチャ優先度リスト1040を供給することができる。

この例ではユーザが、5つの以前につくられたテクスチャに、テクスチャ優先度リスト1040に示される優先度を割り当てている。各テクスチャは、異なるテクスチャIDを割り当てられている。この特定の例においてユーザによって供給される優先度リスト1040は、優先度順にID 3、ID 1、ID 5、ID 2、およびID 6のIDを持つテクスチャを含む。さらに、ID 8を持つテクスチャがユーザによって新しく規定され、グラフィックスAPIまたはハードウェア・ドライバによって処理されようとしている。そのようなテクスチャ（ID 8）が、例えば、APIまたはハードウェア・ドライバによって作用されようとしている新しくつくられたコンテキストと関連させられることができる。

【０３１９】

グラフィックスAPI 1032が、プロセス1030を含む各プロセスのテクスチャの最近の使用を追跡する。図31に示された例において、ID 7を持つテクスチャ、ID 4を持つテクスチャおよびID 1を持つテクスチャが、グラフィックスAPIによってプロセス1030のために最近処理されており、ID 7を持つテクスチャが最も最近処理されている。最近処理されたテクスチャ優先度リスト1044が、APIによってプロセス1030のために維持され、システム・メモリに格納される。最近処理されたテクスチャ（リスト1044に維持されたもの）の中で、最も最近処理したテクスチャ（この例ではID 7を持つ）に最も高い優先度が与えられ、新しいテクスチャが処理されたとき、各テクスチャが優先度において１シフトダウンする。このように、ID 7を持つテクスチャに最も高い優先度が与えられ、ID 1を持つテクスチャに、グラフィックスAPI 1032で維持される、最も最近処理されたテクスチャ優先度リスト1044にリストされた、最も最近処理されたテクスチャのなかで、最下位の優先度が与えられる。

【０３２０】

次いで、グラフィックスAPI 1032は、ユーザ-供給されたテクスチャ優先度リスト1040を、最も最近処理されたテクスチャ優先度リスト1044と合体して、システムの総体的なテクスチャ優先度リスト1042を計算する。グラフィックスAPIは、最も最近処理された優先度に対してユーザ-供給された優先度に優位を与える。このようにID 3、ID 1、ID 5、ID 2およびID 6のIDを持つテクスチャは、ID7およびID 4のIDを持つテクスチャに対して優位を与えられる。ID 8を持つテクスチャは、それがグラフィックスAPIによって処理されようとしている新たに規定されたテクスチャであるので、最も高い優先度を与えられる。優先度リスト1042は、この方式に従って絶えず更新され、システム・ソフトウェア・メモリに格納される。ハードウェア・ドライバは、グラフィックスAPIによって生成したテクスチャ優先度リスト1042を使用してテクスチャ優先度をTIMに知らせる。

【０３２１】

上で図30を参照して説明したように、テクスチャ優先度リスト1042がTIMに知らせられ、TIMは、テクスチャの優先度をリスト1048にも格納する。TIMのテクスチャ優先度リスト1048も、図35に示され、グラフィックスAPIおよびグラフィックス・ハードウェア・ドライバのテクスチャ優先度リストと一致する。ユーザは、ユーザ供給の優先度リスト1040をいつでも更新することができ、その場合にはグラフィックスAPIが、それ自身の優先度リスト1042を更新し、上で説明したように、最も最近処理されたテクスチャ優先度に対しユーザ-供給されたテクスチャ優先度に優位が与えられる。ユーザ-供給された優先度リストが備えられないならば、グラフィックスAPI総体的優先度リスト1042は、最も高い優先度を与

えられる新しくつくられた（まだ処理しない）テクスチャを除いて、最も最近処理されたテクスチャ優先度リスト1044のそれである。

【０３２２】

図36は、上で説明された例の特定のハードウェア装置のメモリ配分計画に従って、同じブロック内に複数のテクスチャが割り当てられることができ異なるサブ・テクスチャIDが割り当てられる3つの異なる方法を示すグラフィック・イラストである。各ブロックの大きさは256x256テクセルである。256x256テクセルの大きさのマップ番号8のベース・マップを持つ第１のテクスチャT1については、そのマップは、2つのブロックの中で割り当てられ、一つのブロックがキャッシュ・メモリ内のSDRAMのバンク0に格納され、一つブロックがバンク1に格納される。このように、各ブロック内の空の空間が、テクスチャT1と同じプロセス内でつくられる、128x128テクセルの大きさの別のテクスチャT2の記憶を可能にする。これらのブロックの一つ1000が図36に示される。テクスチャT1とT2が、同じテクスチャIDを含むが、上でハードウェア装置の説明の中で説明されたように異なるサブ・テクスチャIDを持っている。

【０３２３】

別の例として、それぞれ128x 128テクセルの大きさで同じプロセス内でつくられた（おそらくそのプロセス内の複数のコンテキストで共有される）4つのテクスチャT3、T4、T5およびT6を、別のブロック1010の4つの四分区間にパックすることができる。テクスチャT3、T4、T5およびT6の各々が、異なるサブ・テクスチャIDを割り当てられるが、上で説明したように同じテクスチャIDを含む。

【０３２４】

第３の例は、各テクスチャの大きさが64x64テクセル以下で同じプロセス内でつくられた、16のユニークなテクスチャT7-T22を含む。すべての16のテクスチャT7-T22を、同じブロック1020内にパックすることができ、各テクスチャに同じテクスチャIDおよび異なるサブ・テクスチャIDが割り当てられる。上で示したように、サブ・テクスチャIDは、ブロック内の16の異なるサブテクスチャを見分けるために2つのS座標ビットと2 つのT座標ビットを含む。このように発明の少なくとも一つ実施例を説明したが、いろいろな変更、修正および改善が、技術に熟練した者には容易に生じるだろう。そのような変更、修正と改善は、この発明の精神と範囲内のある。それゆえに、以上の説明は例としてだけのもので、限定的なものではない。この発明は、特許請求の範囲の規定およびその均等の範囲によってのみ制限される。

【０３２５】

この発明は、例として次の実施態様を含む。

【０３２６】

1.テクスチャ・データを格納するシステム・メモリを有するホストコンピュータと、
任意の時点でシステム・メモリに格納されるテクスチャ・データの少なくとも一部を格納する局部メモリを含み、ホストコンピュータに結合し、テクスチャ・マップされた画像を描画する、グラフィックス・ハードウェア装置と、
ホストコンピュータのプロセッサ上で走り、ハードウェア装置が画像の描画を必要とするとき、システム・メモリから局部メモリへのテクスチャ・データの転送を管理するソフトウェア・デーモンと、
を備えるテクスチャマッピング・コンピュータ・グラフィックス・システム。

【０３２７】

2.システムは、テクスチャ・データを格納するメイン・メモリをもつホストコンピュータと、ホストコンピュータに結合し、任意の時点でメイン・メモリに格納されたテクスチャ・データの少なくとも一部をブロックを格納する局部メモリを持つハードウェア装置と、
を備え、ハードウェア装置が、局部メモリに格納したテクスチャ・データを使用して、テクスチャマップされた画像を描画するテクスチャマッピング・コンピュータ・グラフィックス・システムでテクスチャ・データを管理する方法であって、
局部メモリに格納したテクスチャ・データブロックのハードウェア装置による使用をモニ

タするステップと、
メイン・メモリに格納されたテクスチャの優先度を追跡するステップと、
局部メモリの中で最下位優先度および使用最近度が最低のテクスチャ・データブロックを、画像を描画するためにハードウェア装置によって必要とされるテクスチャ・データブロックで置き換えるステップと、
を含むテクスチャ・データの管理方法。
【０３２８】
3．コンピュータ・グラフィックス・システムのプロセッサの上で走るソフトウェアシステムであって、
プロセッサは、少なくとも一つのコンピュータ・グラフィックス・プロセス、各プロセスのために基礎をなすグラフィックスAPI、および各プロセスのための基礎をなすハードウェア・ドライバを走らせ、コンピュータ・グラフィックス・システムが、プロセッサに結合した、テクスチャ・マップされた画像を描画するハードウェア装置を含み、
ソフトウェアシステムは、
プロセッサからハードウェア装置へのテクスチャ・データのダウンロードを管理するデーモンと、
デーモン、グラフィックスAPIおよびハードウェア・ドライバによってアクセス可能なテクスチャ・データを格納する共用メモリ領域と、を備える。
4．コンピュータ・グラフィックス・システムは、その上でソフトウェア・デーモンおよび少なくとも一つのグラフィックス・プロセスが走るプロセッサを含み、ユーザが各プロセス内で複数のコンテキストをつくることができ、プロセッサに結合され、少なくとも一つのグラフィックス・プロセスの中でつくられたテクスチャ・マップされた画像を描画する少なくとも一つのグラフィックス・ハードウェア装置、テクスチャマッピング・コンピュータ・グラフィックス・システムでテクスチャ・データを管理するソフトウェア・デーモンであって、
テクスチャの一つの格納されたコピーが同じプロセス内の複数のコンテキストの間で共有されるように、各プロセスのために別々にテクスチャ・データを管理する、装置独立の部分と、
テクスチャ・データをグラフィックス・ハードウェア装置に書き込むための装置依存部分と、
を備える上記ソフトウェア・デーモン。
【０３２９】
5．装置に依存する部分が共有ライブラリー・ロケーションを含む上記4に記載のソフトウェア・デーモン。
【０３３０】
6．テクスチャマッピング・コンピュータ・グラフィックス・システムでソフトウェア・テクスチャ・データを割り当てる方法であって、
システムは、ホストコンピュータおよびホストコンピュータにとって結合され、テクスチャ・マップされた画像を描画するグラフィックス・ハードウェア装置を含み、データが少なくとも一つのシリーズのテクスチャMIPマップを含み、
ソフトウェア方法は、ホストコンピュータのプロセッサ上で作用し、
少なくとも一つのシリーズのマップの各マップを少くとも2つのマップ部分に分割するステップと、
マップ部分を複数のブロックのデータに割り当てるステップと、あるブロックに格納されたテクスチャ・データがハードウェア装置によって必要とされるとき、ホストコンピュータからハードウェア装置にあるブロックをダウンロードするステップと、を含む。
【０３３１】
【発明の効果】
この発明によると、ハードウェア装置が画像の描画を必要とするとき、システム・メモリから局部メモリへのテクスチャ・データの転送をソフトウェア・デーモンで管理するので

、メモリの有効利用が図られる。

【図面の簡単な説明】

【図１】一組のテクスチャMIPマップを図形化した図。

【図２】先行技術コンピュータ・グラフィックス・システムの一部ハードウェア、一部階層的なソフトウェアのブロック図。

【図３】この発明に従うコンピュータ・グラフィックス・システムの一部ハードウェア、一部ソフトウェアのブロック図。

【図４】この発明に従うもう一つの実施例のコンピュータ・グラフィックス・システムの一部ハードウェア、一部ソフトウェアのブロック図。

【図５】この発明に従うさらにもう一つの実施例のコンピュータ・グラフィックス・システムの一部ハードウェア、一部ソフトウェアのブロック図。

【図６】この発明のソフトウェア・デーモンを使用することができるコンピュータ・グラフィックス・ハードウェア装置の一つの実施例のブロック図。

【図７】この発明のソフトウェア・デーモンを使用することができるコンピュータ・グラフィックス・ハードウェア装置の別の実施例のブロック図。

【図８】図6のハードウェア装置のテクスチャマッピング・ハードウェア・チップのブロック図。

【図９】テクスチャマッピング・ハードウェアのパラメーター補間回路要素のより詳細なブロック図。

【図１０】キャッシュ・メモリとテクスチャマッピング・ハードウェアの一部のブロック図。

【図１１】局部的ハードウェア装置メモリを利用する配置のテクセルをもつテクスチャMIPマップの図表。

【図１２】キャッシュ・メモリを形づくるメモリーチップの組織の詳細なブロック図。

【図１３】テクスチャマッピング・ハードウェアの一部分の詳細なブロック図。

【図１４】テクスチャマッピング方式に従うピクセルのストリームのそれぞれのについて、隣接のMIPマップからアクセスしたテクセルの例を示す図表。

【図１５】図14の例に従うテクスチャマッピング・ハードウェア・バッファおよび関連するデータエントリーの図表。

【図１６】テクスチャマッピング・ハードウェアで採用される回路のブロック図。

【図１７】一組のテクスチャMIPマップの例の図表。

【図１８】図15の例のMIPマップがメモリ格納方式に従ってメモリに格納される方法を図示している図表。

【図１９】MIPマップがメモリ格納方式に従って分割される方法を図示しているMIPマップの図表。

【図２０】マップがメモリ格納方式に従ってさらに分割される方法を図示している図19に示したマップの部分のより詳細な図表。

【図２１】キャッシュ・ブロック・タグが生成される方法を図示する図表。

【図２２】提供された補間されたテクセル・データから対応するテクスチャ・データブロック内で、テクセル・アドレスを判断するための方法を図示するフローチャート。

【図２３】テクスチャマッピング・チップに設けられたテクセル・ポート・レジスタを図示する図表。

【図２４】割込み処理して、データの中でどのブロックをダウンロードするか決定するために、ホストコンピュータのプロセッサによって実行されるソフトウェア・ルーチンを図示するフローチャート。

【図２５】キャッシュのミニ・ディレクトリのブロック図。

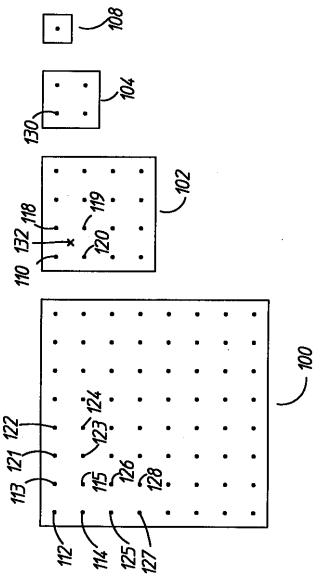
【図２６】キャッシュの主ディレクトリのブロック図。

【図２７】キャッシュ読み取りタグがミニ・ディレクトリをミスするとき、性能ペナルティを減らすために設けられた一連のコンパレータのブロック図。

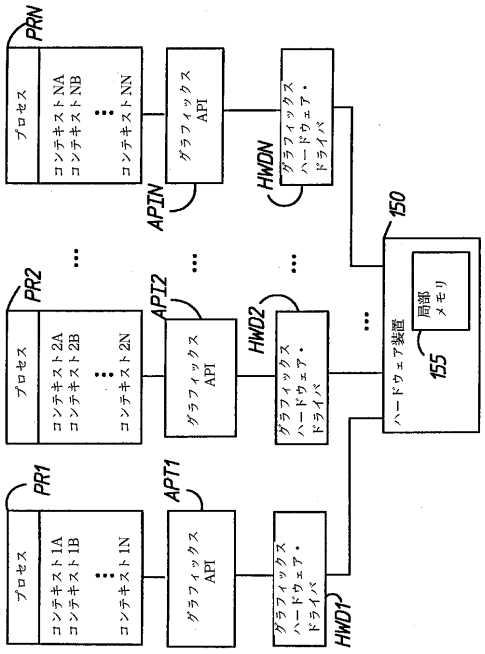
【図２８】キャッシュ・ディレクトリの実施例のブロック図。

【図２９】この発明のテクスチャ割込みマネージャー（TIM）ソフトウェア・デーモンの全体的な操作の流れ図。
【図３０】２つの関係するテクスチャの間の通信の間のハードウェア・ドライバおよびTIMの操作の一部流れ図、一部機能的ブロック図。
【図３１】割込みを処理するためのTIMの装置独立のルーチンの流れ図。
【図３２】割込み中にハードウェア装置の局部メモリ内でどのブロックを置き換えるかを決定するためのTIMの装置に依存するルーチンの流れ図。
【図３３】割込み中にハードウェア装置の局部メモリ内でどのブロックを置き換えるかを決定するためのTIMの装置に依存するルーチンの流れ図。
【図３４】テクスチャ・データをハードウェア・ドライバからTIMに伝える方法、およびテクスチャのコピーを格納するためのTIMの手段を示す機能的なブロック図。
【図３５】システム内のソフトウェア層によって維持されるテクスチャ優先度リストを示し、最終的なシステム・テクスチャ優先度リストを決定するための方式を示す機能ブロック図。
【図３６】比較的小さいテクスチャのテクスチャ・データをブロックにパックするための異なる方式を示す機能的なブロック図。
【符号の説明】
１０　　フロントエンド・ボード
１２　　テクスチャ・マッピング・ボード
１４　　フレーム・バッファ・ボード
１７　　メイン・メモリ
４６　　テクスチャ・マッピング・チップ
４８　　局部（キャッシュ）メモリ

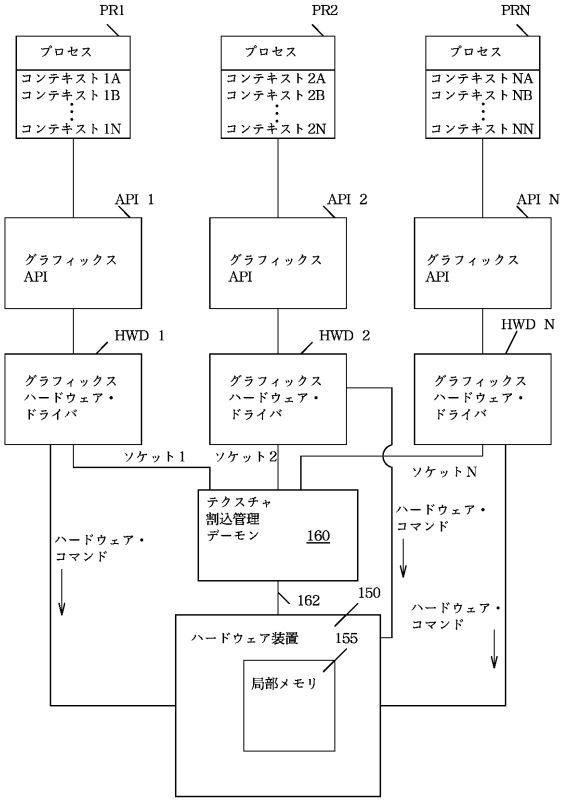
【図１】



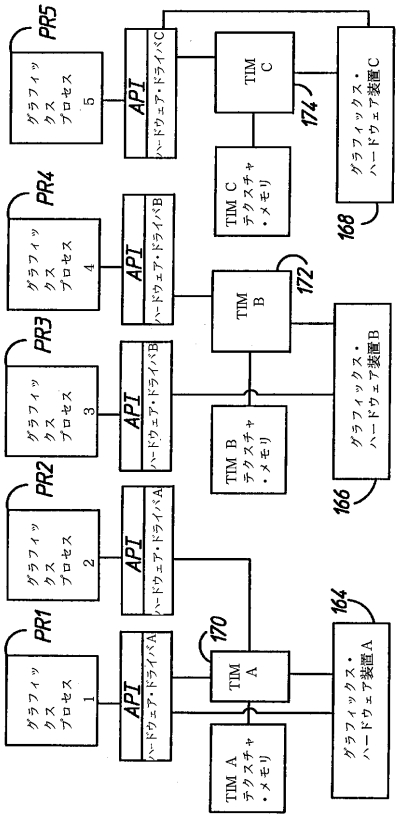
【図２】



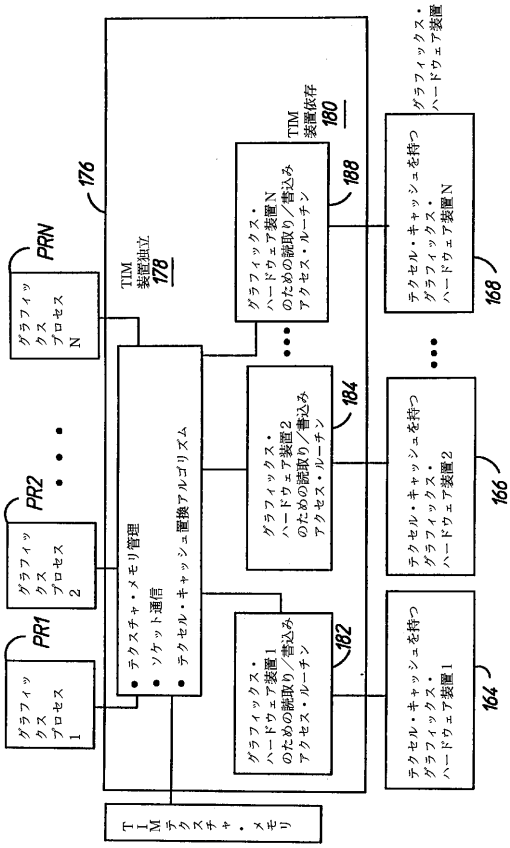
【図３】


PR1
PR2
PRN
プロセス
コンテキスト1A
コンテキスト1B
コンテキスト1N
コンテキスト2A
コンテキスト2B
コンテキスト2N
コンテキストNA
コンテキストNB
コンテキストNN
API 1
API 2
API N
グラフィックス API
HWD 1
HWD 2
HWD N
グラフィックス ハードウェア・ドライバ
ソケット1
ソケット2
ソケットN
テクスチャ 割込管理 デーモン 160
ハードウェア・コマンド
162
150
ハードウェア装置 155
局部メモリ

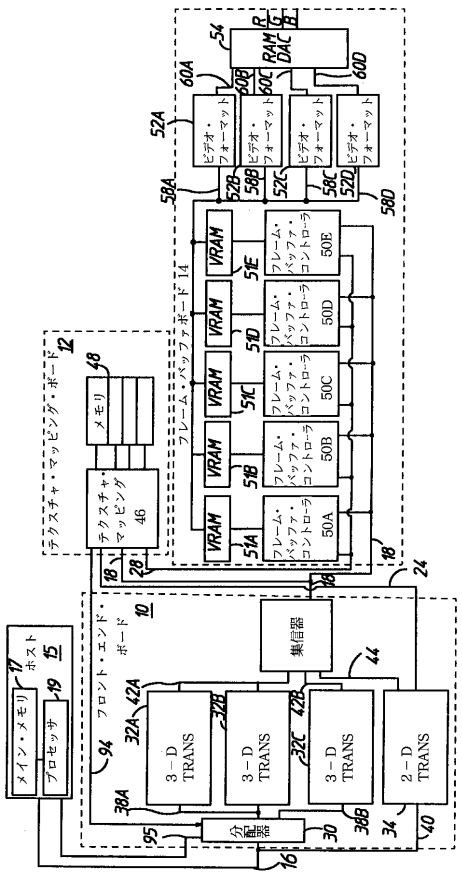
【図４】


PR5
PR4
PR3
PR2
PR1
グラフィックス プロセス 5
グラフィックス プロセス 4
グラフィックス プロセス 3
グラフィックス プロセス 2
グラフィックス プロセス 1
API
ハードウェア・ドライバC
ハードウェア・ドライバB
ハードウェア・ドライバA
TIM C
TIM B
TIM A
TIM C テクスチャ・メモリ
TIM B テクスチャ・メモリ
TIM A テクスチャ・メモリ
グラフィックス・ハードウェア装置C
グラフィックス・ハードウェア装置B
グラフィックス・ハードウェア装置A
168
172
166
174
170
164

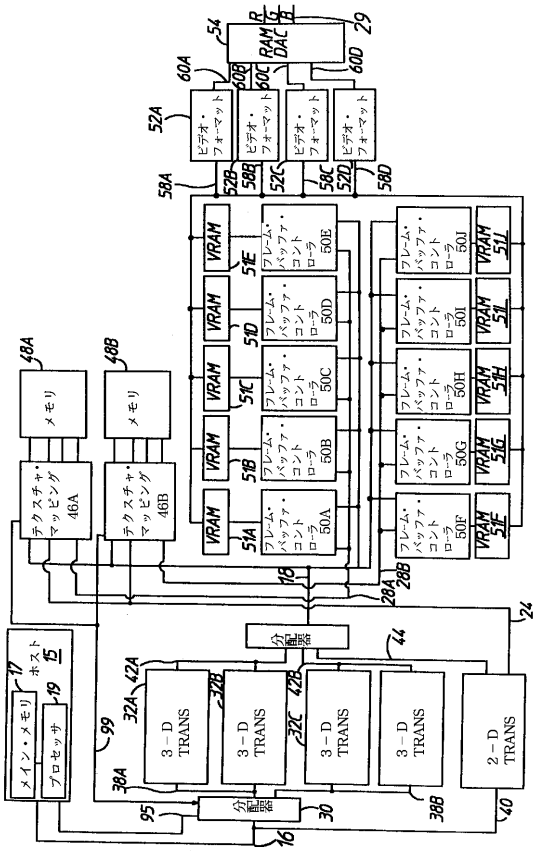
【図５】


PRN
PR2
PR1
グラフィックス プロセス N
グラフィックス プロセス 2
グラフィックス プロセス 1
TIM 装置独立 178
176
• テクスチャ・メモリ管理
• ソケット通信
• テクセル・キャッシュ置換アルゴリズム
TIM 装置依存 180
グラフィックス・ハードウェア装置N のための読取り／書込み アクセス・ルーチン
188
グラフィックス・ハードウェア装置2 のための読取り／書込み アクセス・ルーチン
184
グラフィックス・ハードウェア装置1 のための読取り／書込み アクセス・ルーチン
182
テクセル・キャッシュを持つ グラフィックス・ハードウェア装置N
168
テクセル・キャッシュを持つ グラフィックス・ハードウェア装置2
166
テクセル・キャッシュを持つ グラフィックス・ハードウェア装置1
164
TIM テクスチャ・メモリ

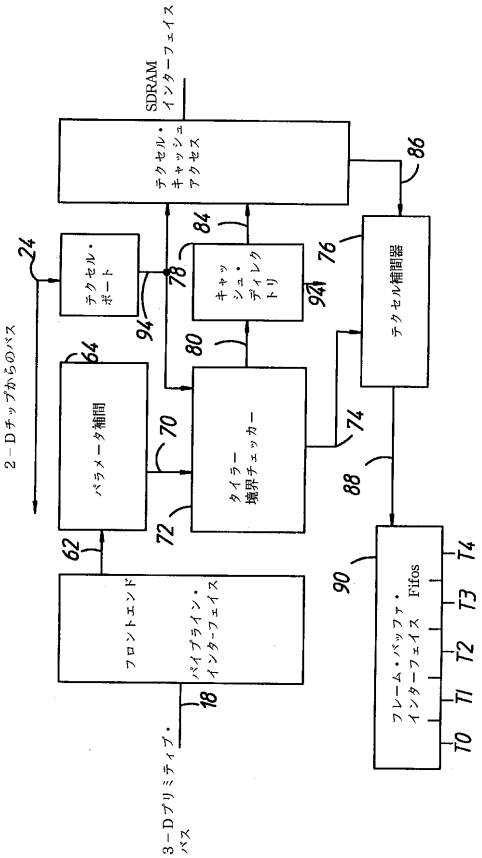
【図６】


メイン・メモリ
ホスト
プロセッサ
フロント・エンド・ボード
テクスチャ・マッピング・ボード
メモリ
テクスチャ・マッピング
フレーム・バッファ・ボード 14
VRAM
フレーム・バッファ・コントローラ
ビデオ・フォーマット
RAM DAC
集信器
3－D TRANS
2－D TRANS
分配器

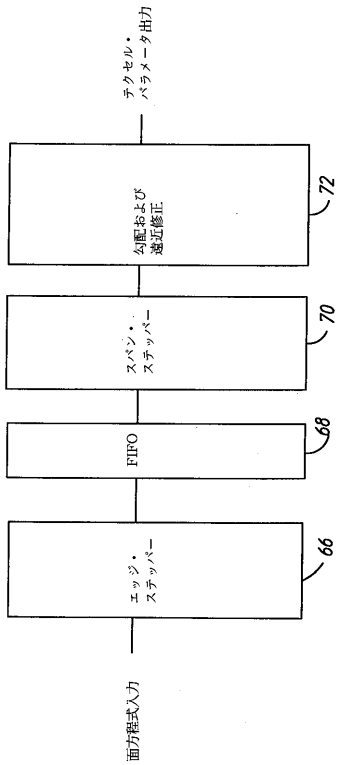
【図７】


メモリ
テクスチャ・マッピング
ホスト
プロセッサ
メイン・メモリ
3－D TRANS
2－D TRANS
分配器
VRAM
フレーム・バッファ・コントローラ
ビデオ・フォーマット
RAM DAC

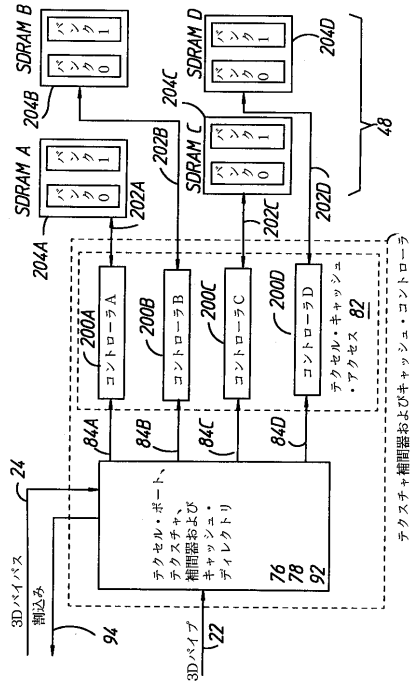
【図８】


SDRAM インターフェイス
テクセル・キャッシュ・アクセス
テクセル・ポート
キャッシュ・ディレクトリ
テクセル補間器
パラメータ補間
タイラー境界チェッカー
フロントエンド
パイプライン・インターフェイス
フレーム・バッファ・インターフェイス Fifos
2－Dチップからのバス
3－Dプリミティブ・バス

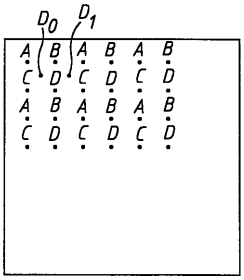
【図９】


テクセル・パラメータ出力
勾配および遠近修正
スパン・ステッパー
FIFO
エッジ・ステッパー
面方程式入力

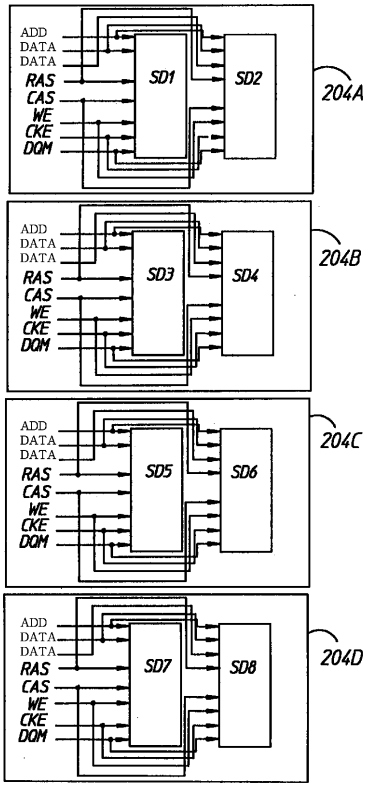
【図１０】


SDRAM A
SDRAM B
SDRAM C
SDRAM D
バンク0
バンク1
コントローラA
コントローラB
コントローラC
コントローラD
テクセル・キャッシュ・アクセス
テクセル・ポート、テクスチャ、補間器およびキャッシュ・ディレクトリ
テクスチャ＋補間器およびキャッシュ・コントローラ
3Dバイパス
割込み
3Dバイプ

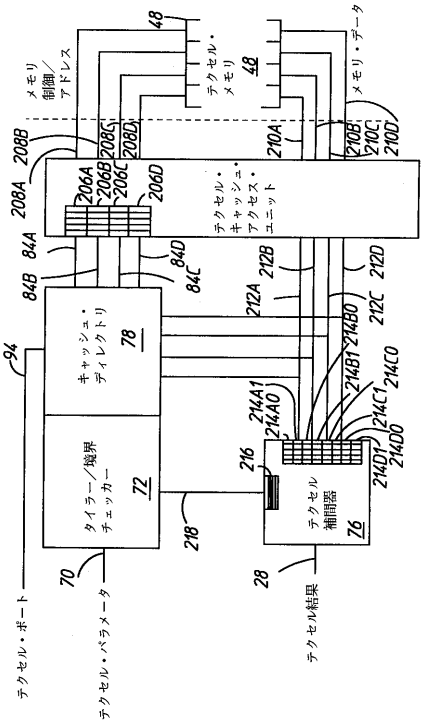
【図１１】



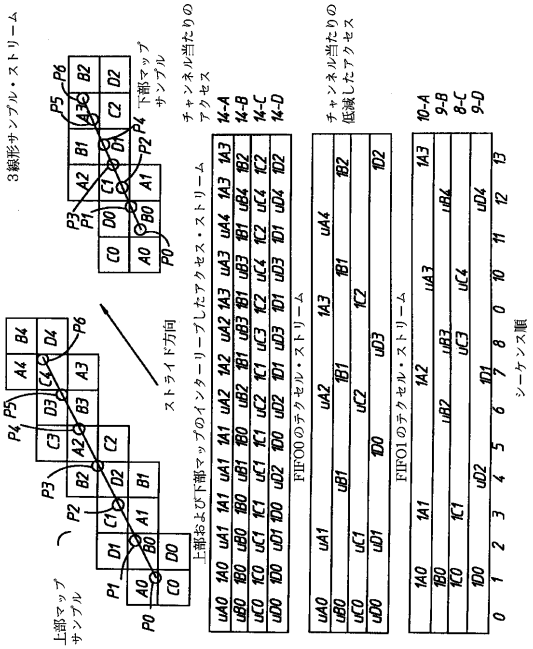
【図１２】



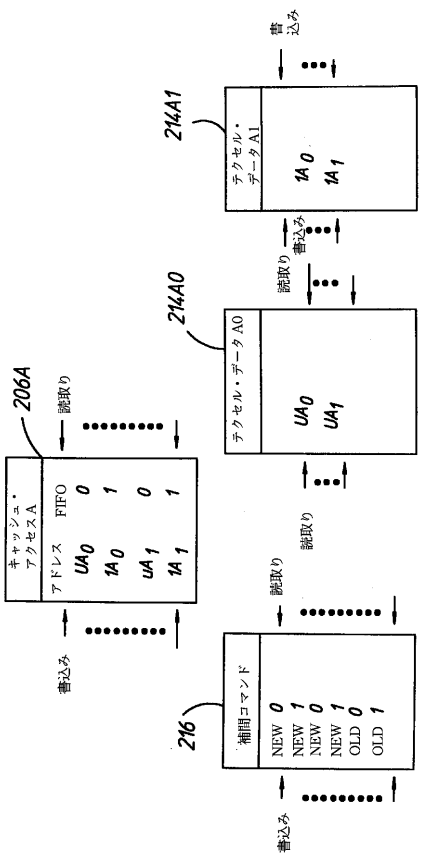
【図１３】



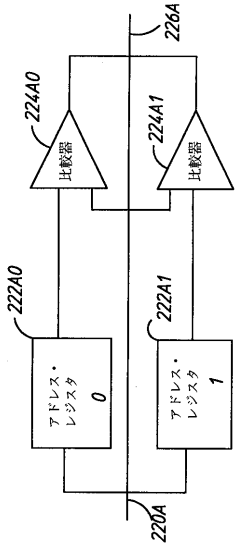
【図１４】



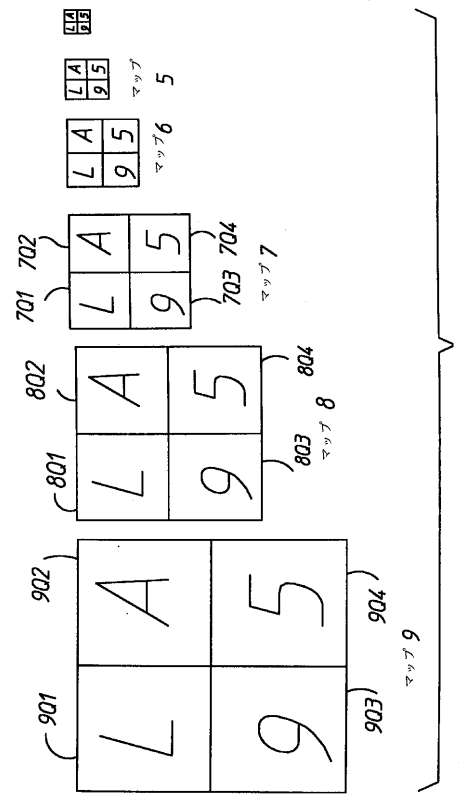
【図１５】



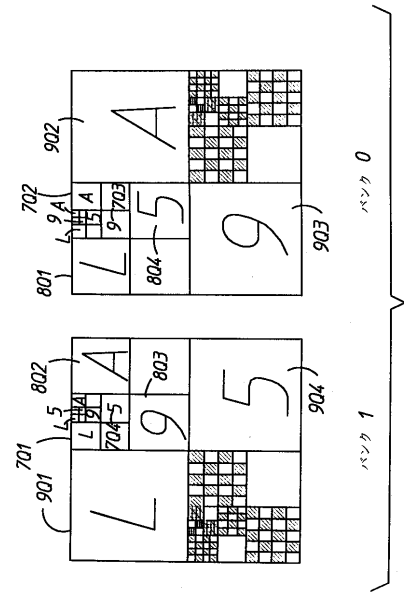
【図１６】



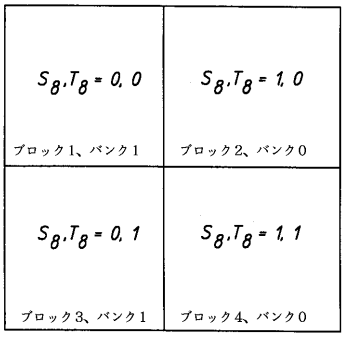
【図１７】



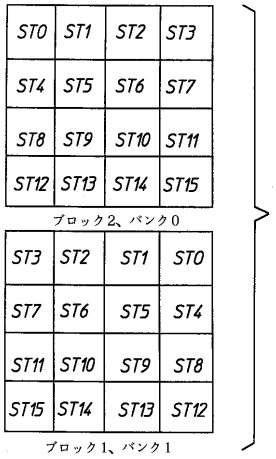
【図１８】



【図１９】



【図２０】

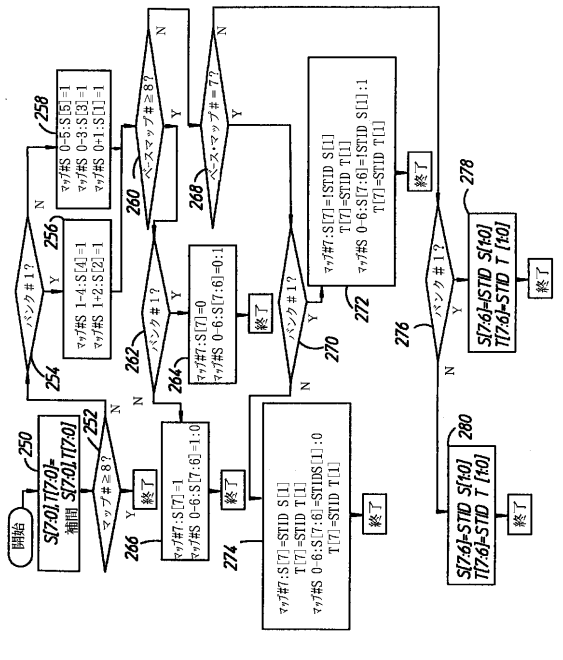


【図２１】

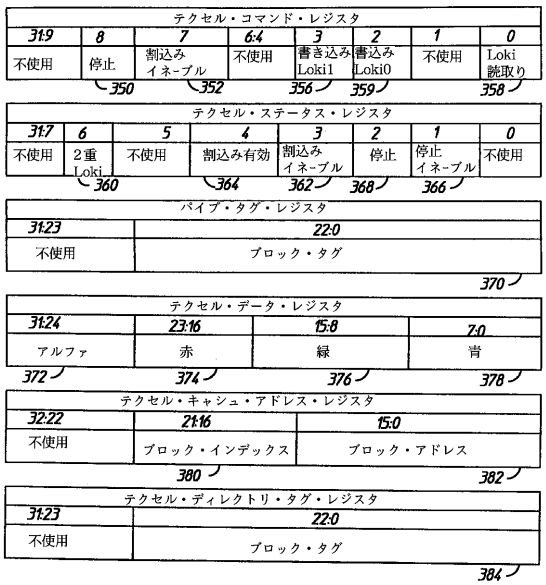
ブロック・タグ [22:0] = {テクスチャID [1:0],マップ・ビット,T[14:8],S[14:8]}

マップ番号	マップビット	Tパラメータ軸MSB							Sパラメータ軸MSB							バンク
15	0	t14	t13	t12	t11	t10	t09	t08	s14	s13	s12	s11	s10	s09	s08	t14^s14
14	1	0	t13	t12	t11	t10	t09	t08	0	s13	s12	s11	s10	s09	s08	t13^s13
13	1	1	0	t12	t11	t10	t09	t08	0	0	s12	s11	s10	s09	s08	t12^s12
12	1	1	1	0	t11	t10	t09	t08	0	0	0	s11	s10	s09	s08	t11^s11
11	1	1	1	1	0	t10	t09	t08	0	0	0	0	s10	s09	s08	t10^s10
10	1	1	1	1	1	0	t09	t08	0	0	0	0	0	s09	s08	t9^s9
9	1	1	1	1	1	1	0	t08	0	0	0	0	0	0	s08	t8^s8
≦8	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	タグ [7]
≦8	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	タグ [7]

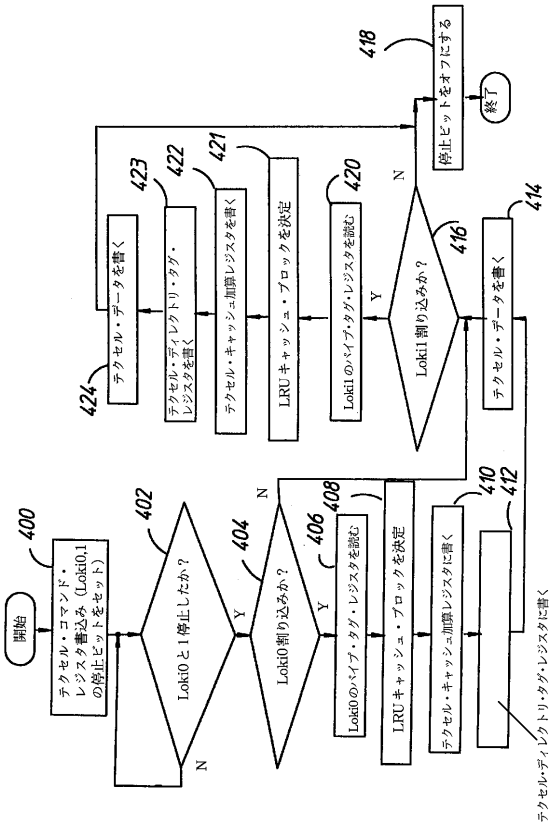
【図２２】



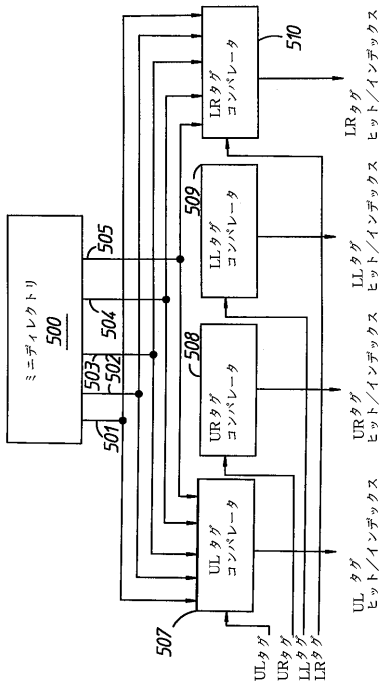
【図２３】



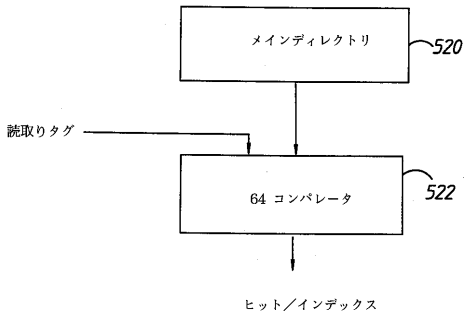
【図２４】



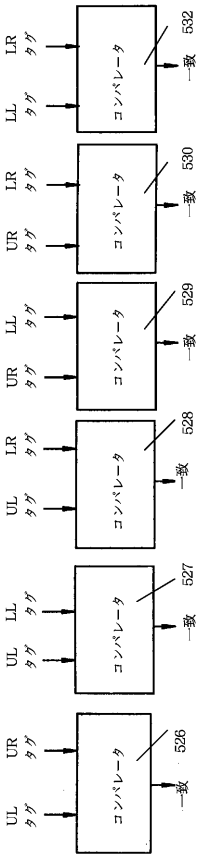
【図２５】



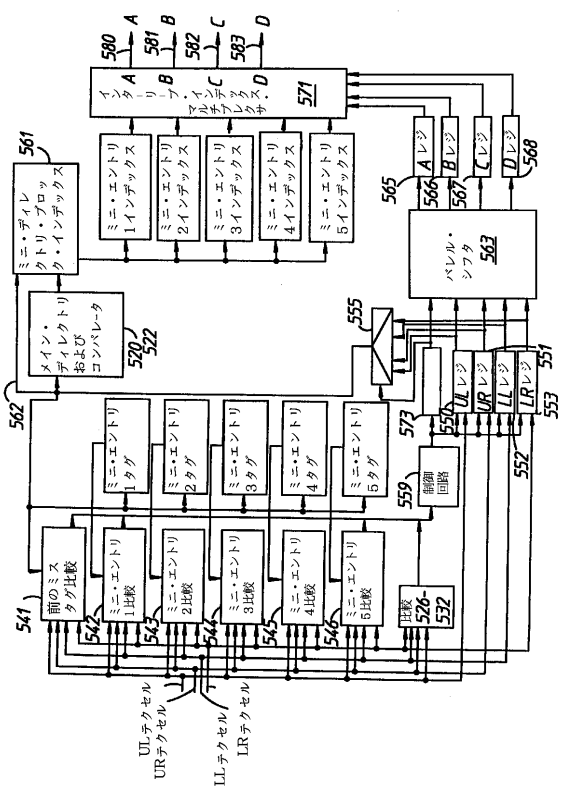
【図２６】



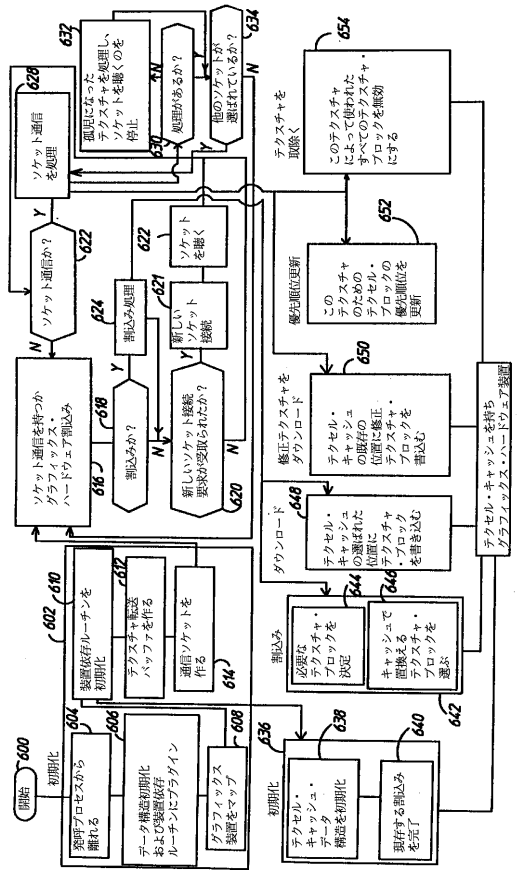
【図２７】



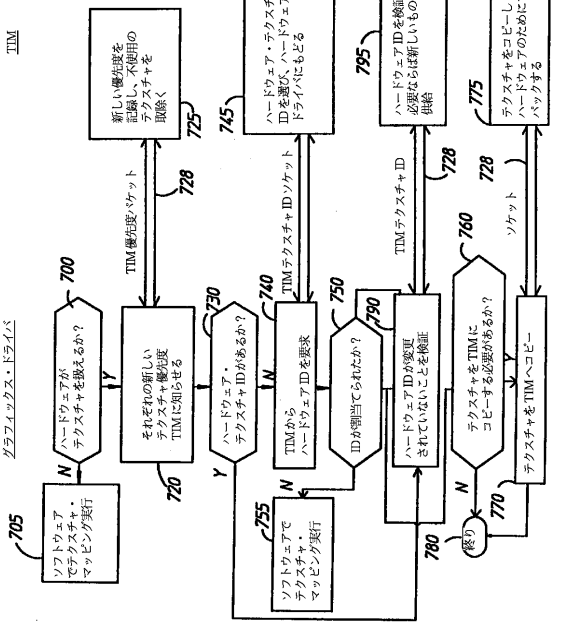
【図２８】



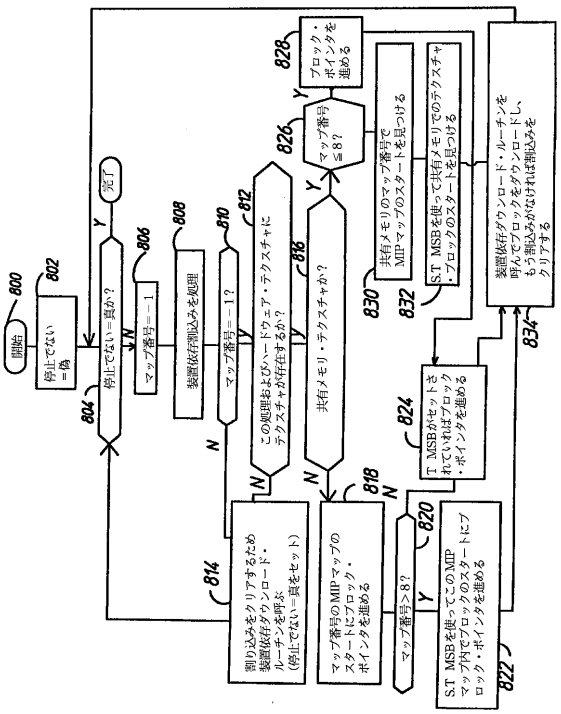
【図２９】



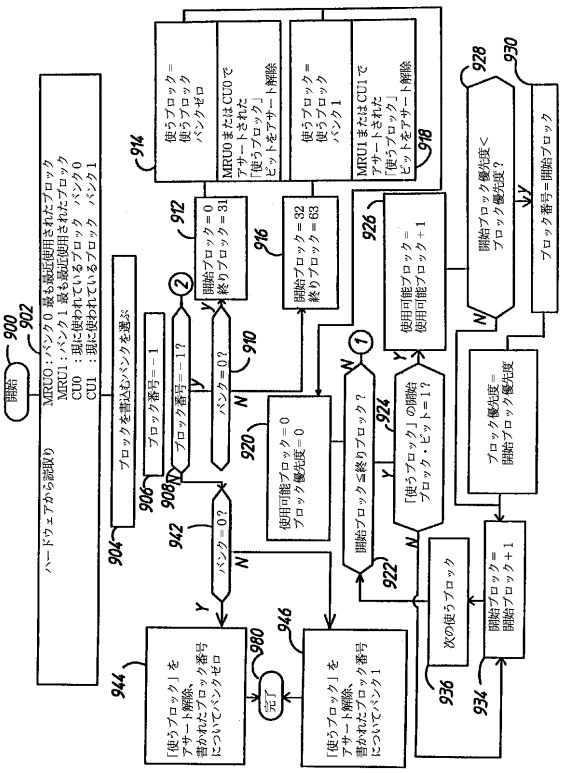
【図３０】



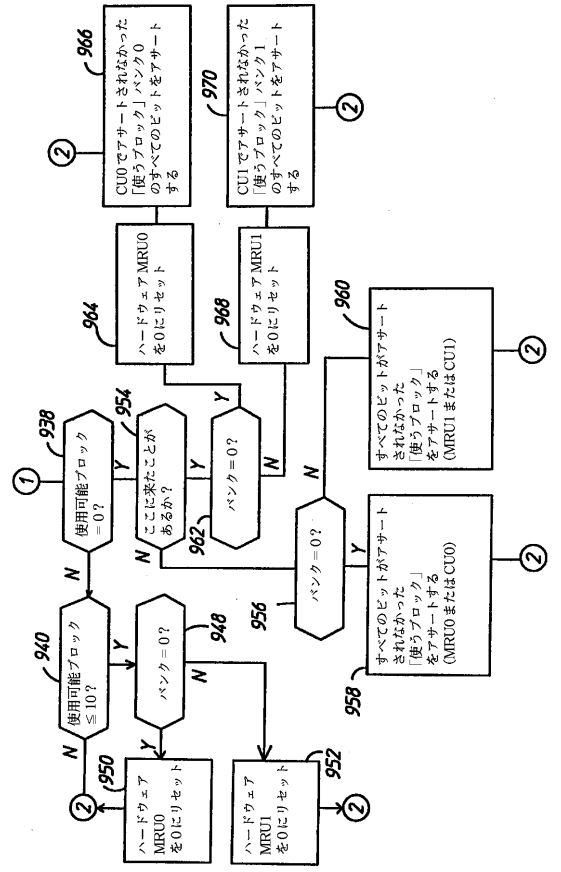
【図３１】



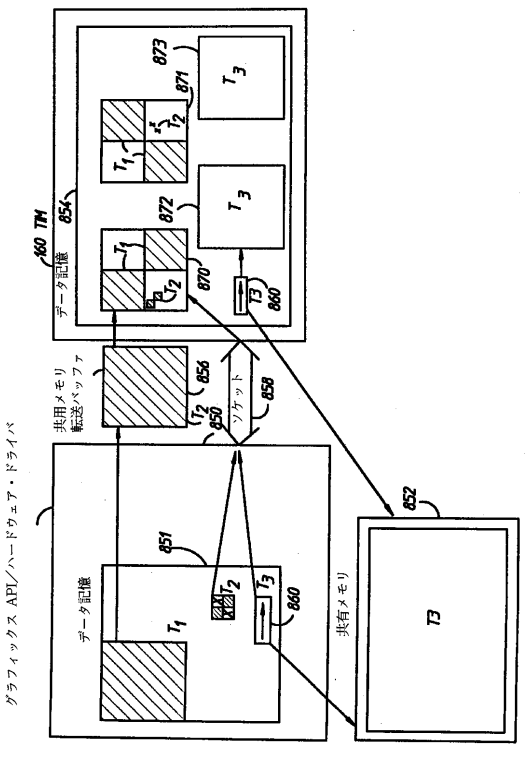
【図３２】



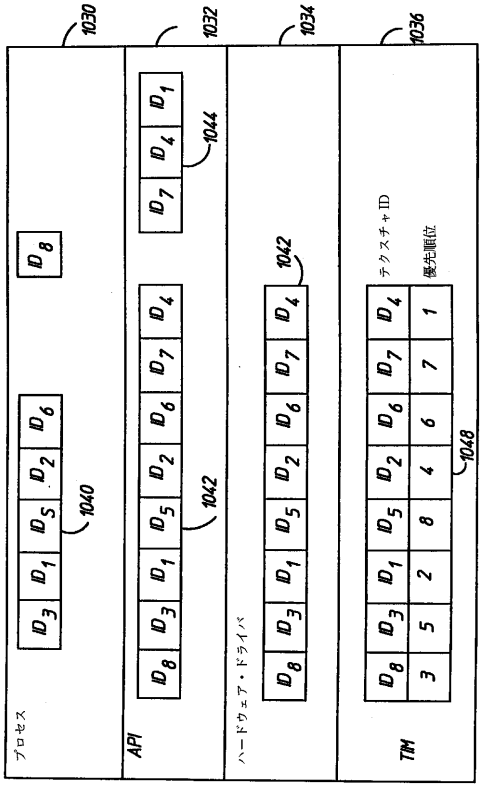
【図３３】



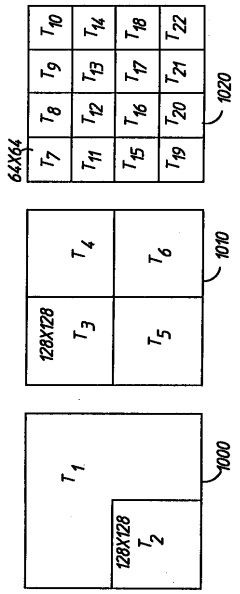
【図３４】



【図３５】


1030
1032
1034
1036
プロセス
API
ハードウェア・ドライバ
TM
テクスチャID
優先順位
1040
1042
1044
1048

【図３６】


64X64
128X128
128X128
1000
1010
1020

フロントページの続き

(56)参考文献　特開平０４－２１１８７８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06T 15/00 - 15/60