

Dream cast ハードウェア概要

1998/8/5 1

1. 更新履歴	4
2. ハードウェアの構成	5
3. 各モジュールの概要	6
3-1. グラフィック機能	6
3-1-1. ポリゴン	8
3-1-2. 頂点	9
3-1-3. タイル	9
3-1-4. Sort Mode	11
3-1-5. Modifier Volume	12
3-1-6. Mipmap	14
3-1-7. アルファブレンディング (混合処理)	14
3-1-8. テクスチャフィルタリング	15
3-1-9. フォグ	16
3-1-10. クリッピング	17
3-1-11. バッファ	18
3-1-12. Image Super Sampling & Y スケーラフィルタリング	19
3-1-13. テクスチャ	20
3-1-14. Bump Mapping	24
3-1-15. ストリップ	26
3-1-16. Sprite	26
3-1-17. カラークランプ	27
3-2. サウンド機能	27
3-2-1. サウンド機能の概要	27
3-2-2. ADPCM とは?	27
3-3. ペリフェラル	29
3-4. GD-ROM (仮称)	30
3-4-1. GD-ROM とは?	30
3-4-2. GD-ROM の構造	31
3-5. CPU	32
3-5-1. FPU	32
3-5-2. MMU	32

3-5-3. DMA	34
3-5-4. タイマ.....	35
3-5-5. キャッシュ.....	35
3-5-6. パイプライン・スーパースカラ	35

表 1 代表的な機能7

図 1 半透明ポリゴンと不透明ポリゴン	8
図 2 タイル	9
図 3 描画のしくみ	10
図 4 タイルアクセラレータの特徴	11
図 5 Modifier Volume の原理	12
表 2 Modifier Volume のモード	13
図 6 Modifier Volume の注意点	13
図 7 MipMap 効果	14
図 8 MipMap テクスチャ	14
図 9 アルファブレンディング	15
表 3 テクスチャフィルタの種類	15
図 10 テクスチャフィルタの効果	16
図 11 Lookup Table Mode	17
図 12 Global Tile Clip と User Tile Clip	17
図 13 クリッピング処理の流れ	18
図 14 フレームバッファのしくみ	18
図 15 Strip Buffer	19
表 4 テクスチャのピクセルフォーマット	21
表 5 Twiddled 形式	22
図 17 VQ 圧縮の原理	23
表 6 DREAMCAST の VQ 方式の特徴	24
図 18 バンプマップテクセルの意味	24
図 19 バンプマッピングの原理	25
図 17 テクスチャ + バンプマッピング	25
図 21 ストリップ系ポリゴンと独立系ポリゴンの違い	26
図 22 音の波形	28
図 23 PCM のデータとは？	28
図 24 ADPCM とは？	29
図 25 コントローラとビジュアルメモリ	30

図 26 GD-ROM の構造	32
図 27 MMU の働き	33
図 28 キャッシュの利点	35
図 29 初期のCPU	36
図 30 パイプライン処理 (図では3本のパイプライン)	36
図 31 効率のよいパイプラインの使い方	37
図 32 スーパースカラ	38

1-1-1-1. 更新履歴

< 1998 年 8月 5日 >

3-1-1 ポリゴン:

・図 1 の修正。

Holly 2 (SET5 2 搭載の最終チップ)による機能拡張 (パンチスルー)に関する情報公開。

3-1-7 アルファブレンディング (混合処理):

・パンチスルーポリゴンに置き換えられた処理の記述を削除。

3-1-16 スプライト:

・パンチスルーポリゴンに置き換えられたことにより、記述内容変更。

2 ハードウェアの構成

DREAMCAST のハードウェアは大きく分けて次のようなモジュールに分けることができます。

- ・ CPU (Hitachi SH7091 カスタム 200MHz)
- ・ グラフィック (Holly)
- ・ サウンド (AICA)
- ・ メモリ (システムメモリ 16MB、テクスチャRAM 8MB、サウンドRAM 2MB)
- ・ GD-ROM ドライブ (GigaByte Disk ROM ドライブ 仮称)

それぞれについて簡単に説明します。

1) CPU

CPU として、200MHz で動作する日立製作所製の SH4 をベースに、セガ仕様にカスタマイズした SH7091 マイクロンを使用しています。

2 グラフィック

グラフィック LSI として Holly を搭載していますが、この LSI はグラフィックコントロール部分 (PowerVR) とシステムコントローラを含んだ構造になっています。SATURN の VDP1 と SCU、SMPC の機能がひとつになったものと見ることもできます。

3 サウンド

サウンド用の LSI には、YAMAHA 製の AICA を搭載しています。

AICA には、サウンド処理を行うための部分以外にサウンドコントロール用として、Advanced RISC Machines (ARM) 社製の ARM7DI を内蔵しており、この LSI だけで、サウンドに関するすべての仕事ができます。

4 メモリ

メモリは、メイン RAM として 16MB、テクスチャデータや、フレームバッファ、ディスプレイリスト用に 8MB、サウンド用に 2MB 積んでいます。

いずれも、シンクロナス DRAM という高速なアクセスの可能な RAM を使っています。

5 GD-ROM ドライブ

ソフトの供給元として 1 GigaByte CD-ROM という DREAMCAST 専用メディアを読むことができる GD-ROM ドライブを搭載しています。

GD-ROM は、ディスクの回転速度が一定の方式であるため、ディスクの内周部と外周部では読み込み速度が異なり、4 倍速から 12 倍速の読み込み速度になります。

3 各モジュールの概要

次のモジュールについてより詳しく説明します。

- ・ グラフィック機能
- ・ サウンド機能
- ・ ペリフェラル
- ・ GD-ROM
- ・ CPU

3-1. グラフィック機能

DREAMCAST のグラフィック機能は、Holy チップ内の PowerVR という部分で取り扱われます。

PowerVR は、複雑な 3D モデルを効率よく扱うことができるように設計されており、さまざまな機能を持っています。以下の表 1 に PowerVR が持つ代表的な機能とその使用例を挙げましたので、以後の説明を見るための参考にしてください。

注釈 2) PowerVR アーキテクチャについては、NEC 社のホームページにて解説されていますのでそちらもご利用下さい。

機能名	代表的な使い方	関連する項目
Modifier Volume	スポットライトや、凹凸がついたモデルに影をつけたりできます。	Modifier Volume
半透明のソート	半透明の Z ソートを、ハードによってピクセル単位に自動的にソートするモードと、あらかじめソフトで Z ソートして入力順に描画するモードの 2 つがあります。ピクセル単位のソートは、ポリゴンの交差が多いほど処理に時間がかかりますが、その分精度は高くなります。	Auto Sort Pre Sort
テクスチャフィルタリング	遠くにあるモデルのテクスチャが崩れてしまうのを軽減するいくつかのテクスチャデータのフィルタリングを提供します。	Bi-Linear Filter Tri-Linear Filter Texture Super Sampling
画像スケーリング	レンダリングした結果を、X、Y 方向に拡大縮小し、かつフィルタリングもできます。この機能を使うと高精細の画面モード (インターレース) でもちらつきません。	Image Super Sampling Flicker Free Interlace
フォグ	ハードウェアで霧などの効果を出すことができます。霧の掛かり方をテーブルにすることで、さまざまな雰囲気 の霧を作り出すことができます。	Fog
MIP MAP	奥行き の深さにあわせていくつかの最適化されたテクスチャデータを用意しておくことで、遠くにあるモデルのテクスチャをきれいに見せることができます。	MipMap
クリッピング	ポリゴン単位でクリッピングを変える ことができます。画面上に複数の窓 を設けることもできます。	Clipping Global Clipping User Clipping
テクスチャ圧縮	テクスチャを圧縮できます。	VQ Twiddled
バンプマッピング	擬似的に表面の凹凸感を出す ことができます。	Bump Mapping
カラークランプ	フェードイン・アウトの効果を 出す事が出来ます。	Color Clamp

関連する項目」の太字は代表する項目名で、細字はその中の詳細な項目になります。

表 1 代表的な機能

3-1-1. ポリゴン

DREAMCAST には、次の 3種類のポリゴン形態があります。

- ・ 不透明ポリゴン
- ・ 半透明ポリゴン
- ・ パンチスルーポリゴン

不透明ポリゴンは、後ろの絵が透けて見えないポリゴンで、非常に高速に処理することが可能です。

一方半透明ポリゴンは、そのポリゴンを通して、後ろのポリゴンや背景が透けて見えます。

たとえば、色付きのガラスのような効果を出すこともできますが、ある部分を完全に抜いてしまうこともできます。

ただし、半透明ポリゴンは、不透明ポリゴンのように高速に処理できないため、多くの半透明ポリゴンが重なってしまうと極端に処理スピードが落ちてしまいます。

つまり極力不透明ポリゴンを使用し、半透明ポリゴンを減らすことで、非常に高い描画能力を発揮することができます。

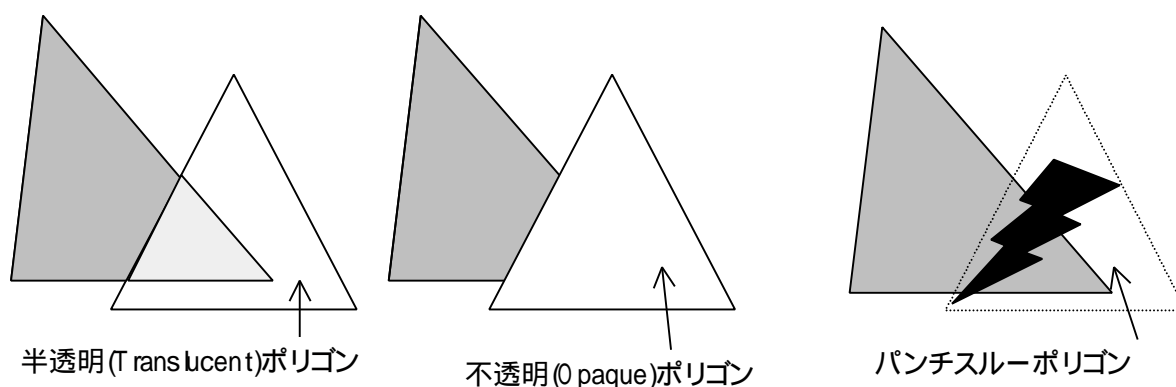


図 2 半透明ポリゴン , 不透明ポリゴン , パンチスルーポリゴン

パンチスルーポリゴン “通称 : 抜きテクスチャ” は、ポリゴンの中に完全な透明色が含まれるもので、主に木や文字、スプライト処理等に用いられます。このパンチスルー機能は Holly 1 までは半透明ポリゴンで扱われていましたが、Holly 2 より別ステータスとして処理を行います。

これによって、不透明ポリゴンに近い描画性能が発揮でき、描画性能が飛躍的に向上します。

3-1-2. 頂点

DREAMCAST のハードウェアは、頂点に対してさまざまな効果を与えることができます。

たとえば、各頂点ごとに別々の色を与えて、その間にあるポリゴンの色属性を変化させるグーローシェーディング (色補完シェーディング) や、光が当たって明るくなっている頂点の色の輝度を上げて、ハイライトがかかっているように見せるスペキュラー効果などです。

DREAMCAST には、各頂点が元々持っている色情報や、それに加えて与えることができる色情報 (カラーオフセット) を指定することもできるので、ポリゴンで作られたモデルが光の加減や位置によって変化するような、リアルな映像を作り出すことが可能です。

3-1-3. タイル

PowerVR は、画面を 32 ドット× 32 ドットのタイルという単位に分割して処理するという風変わりな特徴を備えています。(ARC1 では 32× 8ドット)

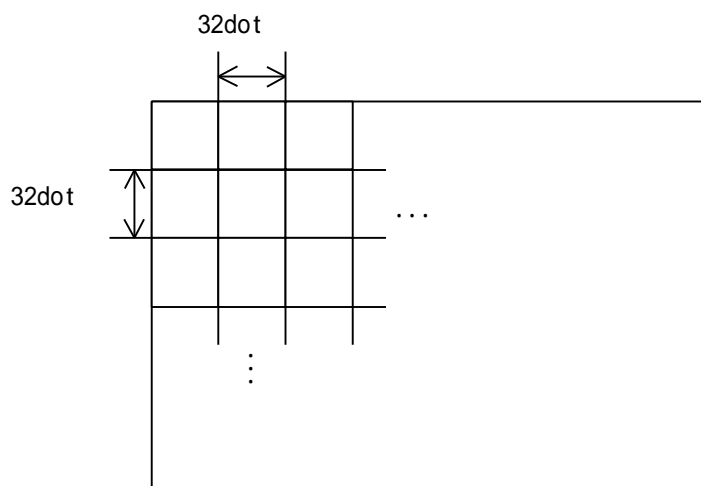


図 3 タイル

ポリゴンの描画はすべてタイル単位で行われます。

PowerVR コアの中には、1 つのタイルを非常に高速にレンダリングするためのモジュールが含まれているので、このチップは他のグラフィックチップに比べて大量のポリゴンを描画することができるのです。

まず、ユーザがポリゴンの表示を要求すると、どのポリゴンがどのタイルに属しているのかをハードウェア (Tile Accelerator) が計算し描画エンジンにデータを渡します。

描画エンジンは、タイルごとにポリゴンのソートを行い、陰面処理や透過処理を施し、該当するフレームバッファに書き込みます。

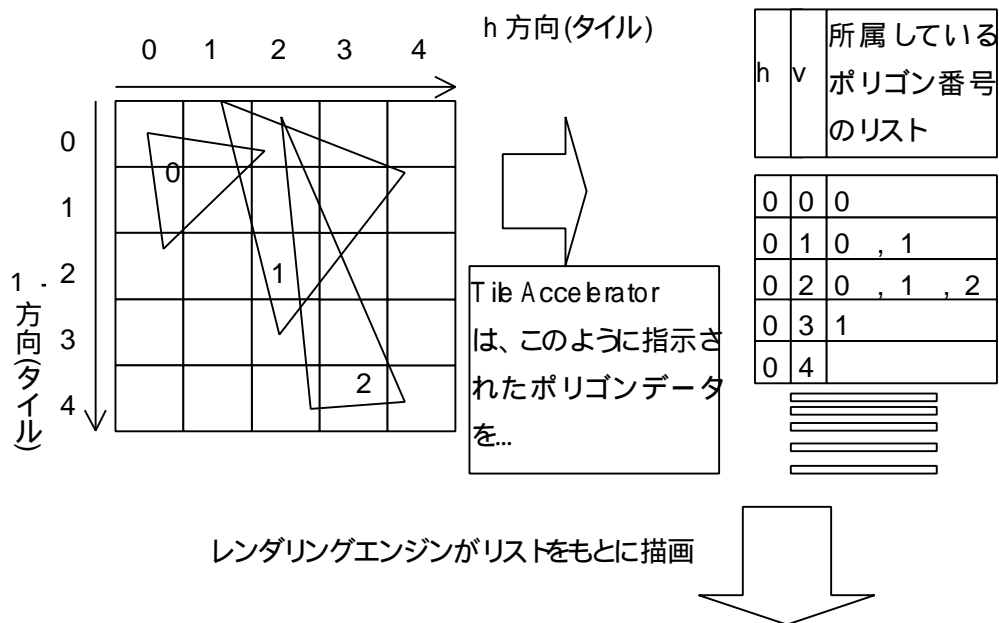


図 4 描画のしくみ

ただし、タイルアクセラレータは、そのポリゴンを含む矩形領域（長方形）のタイルすべてについてポリゴンが存在していると登録するため、極端な話だと斜めに長い三角形ポリゴンの場合、非常に大きな領域についてそのポリゴンが存在しているとみなします。(図 4)

登録されているタイルの数が多ければ、結果として何も表示されないとしても多くの処理時間を必要とするので、その分描画の性能が落ちてしまいます。このハードウェアでは斜めに線状に入るようなポリゴンは可能な限り避けたほうがより性能を引き出すことができます。もし、あらかじめこのようになることが分かっていたら、線分を小さく分けて表示させたほうが、ポリゴン数は多くなりますが、描画能力は向上します。

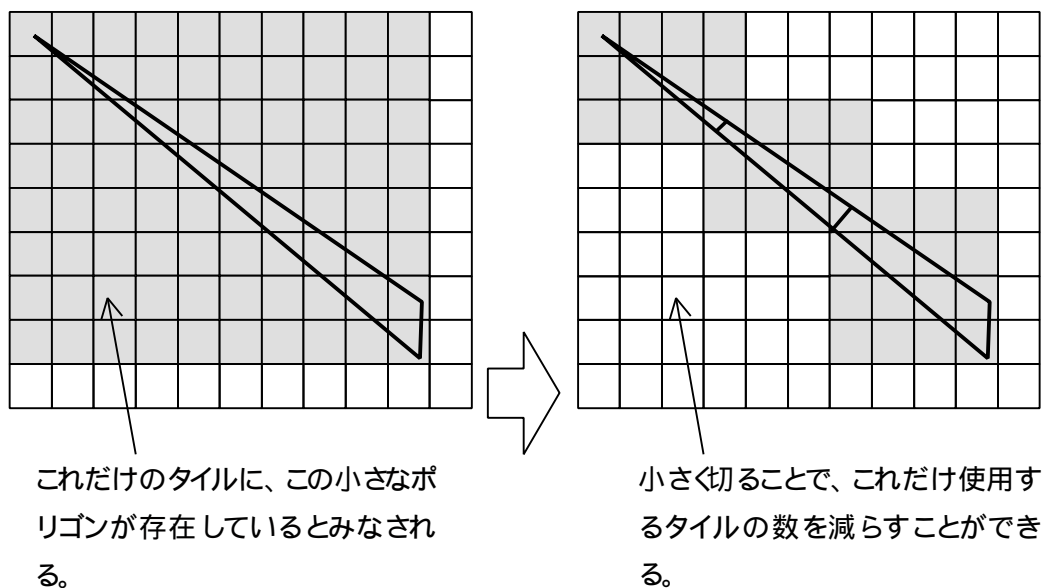


図 5 タイルアクセラレータの特徴

もし次の描画タイミング（例えば V-Blank）までに描画処理が終了しない場合、未処理のタイルについては、その前のフレームのデータ（ダブルフレームバッファであれば 2 つ前のフレーム）の絵が残ってしまいます。

SATURN を含む一般的なレンダリング方法では、ポリゴンごとに描画が落ちるため、ポリゴンがところどころ消えたようになります。

また、どのタイルから処理するのかをユーザ側で指定することができるため、極端な話をいえば、中央から螺旋上に描画するようなこともできます。

さらに、描画落ちをしたときに画面への影響を極力小さくしたり、普段は変化しない部分の描画を後に回すなどの技法を用いることもできます。

3-1-4. Sort Mode

半透明ポリゴンの場合、ポリゴンのソートを自動的に行う機能があります。

ポリゴンのソートを自動的に行うと、ポリゴン同士が重なった場合でも正しい描画結果を出力できますが、その分処理には相当の時間がかかります。

あらかじめソフトでソートしてからハードウェアにデータを渡す Pre Sort モードもサポートされていますが、自動的にソートを行ったのと同じ結果は得られません。

3-1-5. Modifier Volume

Modifier Volume とは？

Modifier Volume とは、あるモデルに対して別の仮想的なモデルを与え、仮想モデルがそのモデルと交わっている部分とそうでない部分とでテクスチャやマテリアルなどのデータを変えてしまう機能のことです。下の図 5 を見てください。

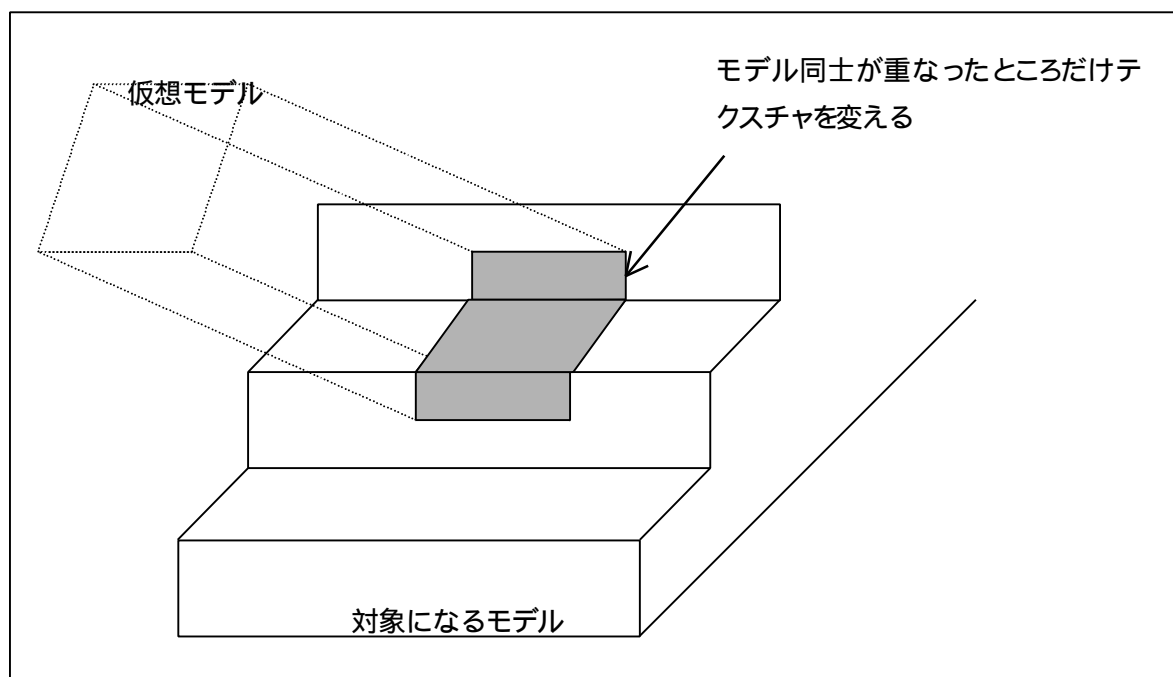


図 6 Modifier Volume の原理

図 5 では、モデルが重なったところのテクスチャデータには、仮想モデルが持っていたテクスチャデータが貼られますが、逆に重なったところだけ、元のモデルが持っていたテクスチャデータで、重なっていないところを仮想モデルのテクスチャデータにすることもできます。

前者を Inclusion Modifier Volume、後者を Exclusion Modifier Volume といいます。

また、差し替えるものはテクスチャである必要はなく、マテリアルのデータでも頂点の色でも構いません。

この Modifier Volume は図を見ても分かるように、凹凸のあるモデルに影をつけるときに有効な方法です。

ただ影の場合は、単純に影の部分を暗くするだけなので、別のテクスチャやマテリアルを設定するのは少々無駄が多いため、重なった部分または重なっていない部分だけを単純に暗くする Intensity モードも用意されています。

これらのモードの違いを表 2 に示します。

Parameter Section Mode

Intensity Mode

特徴	領域内には別のテクスチャを貼ったり、色を自由に変えたりできる。	領域を暗く変えるだけ。
長所	影の部分のテクスチャを変えるなどのさまざまな表現が可能。	影をつけるという決まりきった処理の場合、リソースの消費が少ない分効率がよい。影にはこれで十分。
短所	普通のモデルに比べて 2 倍のリソースを必要とする。	影以外の表現は無理

表 2 Modifier Volume のモード

・ 注意点

Modifier Volume を使う際には注意しなければならないことがあります。

それは、半透明モデルには半透明の仮想モデルを、不透明モデルには不透明の仮想モデルを使用しなくてはならないということです。

半透明モデルに、不透明の仮想モデルをあてた場合、たとえ物理的に交わっていたとしても、Modifier Volume の処理は起こりません。

簡単な図で説明しましょう。

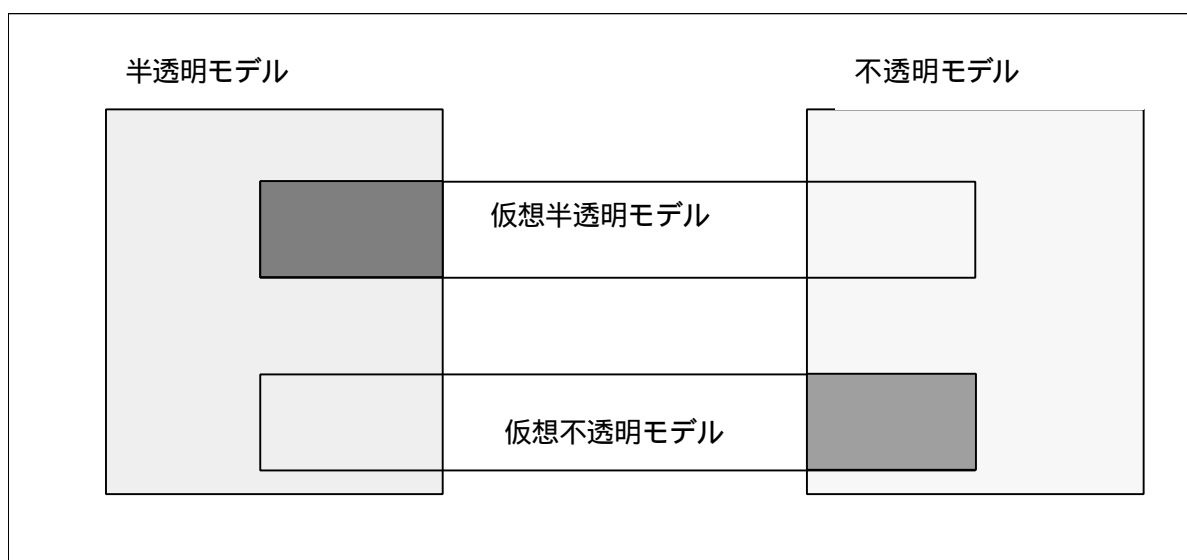


図 6 Modifier Volume の注意点

図 6 では、半透明モデルと仮想半透明モデルとの間では模様が変わっていますが、不透明モデルと仮想半透明モデルとの間は交わっているにもかかわらず模様 (テクスチャ) は変わっていません。同様に、半透明モデルと仮想不透明モデルとの間にも同じことが起こっています。

3-1-6. MipMap

Z 方向に動きのある画面では、テクスチャデータを拡大、縮小する必要があります。今までのゲーム機では、単に一枚のテクスチャを拡大縮小していたため、拡大時には絵が荒く、縮小時には絵がつぶれてジラジラしました (図 7)。この場合 Z 方向の距離に応じて、あらかじめフィルタリングしてあるテクスチャを用意し差し替えることで、視覚的に自然でなめらかな表現が可能になります。この方法を MipMap (ミップマップ) と呼びます。

MipMap 処理を行うためには、テクスチャのサイズは、指定サイズから 1×1 の最小サイズまでは 2 の累乗になるようにデータを作成します。 (図 8)

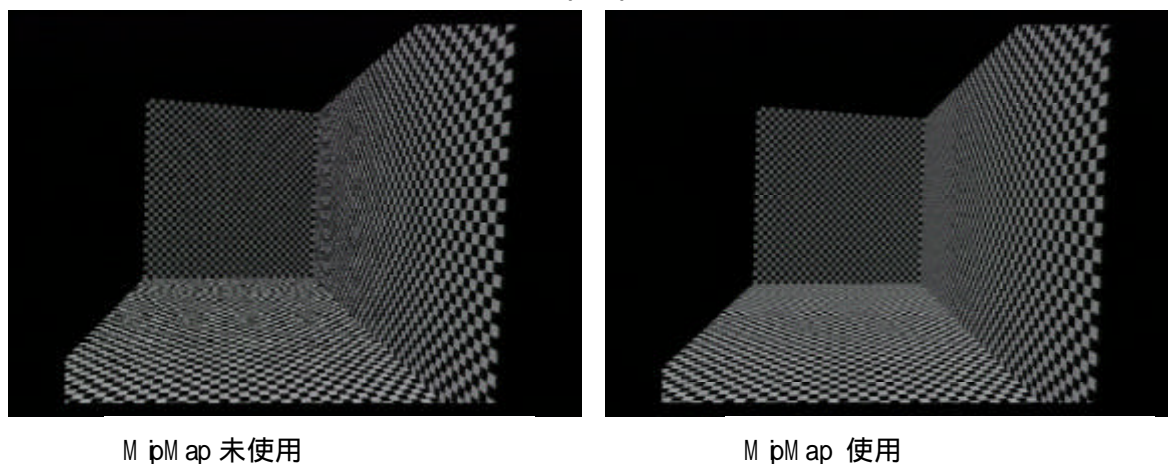


図 7 MipMap 効果

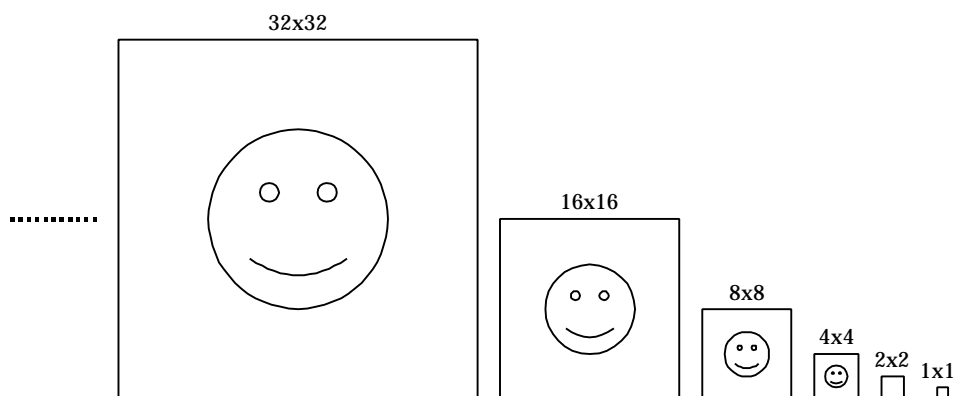


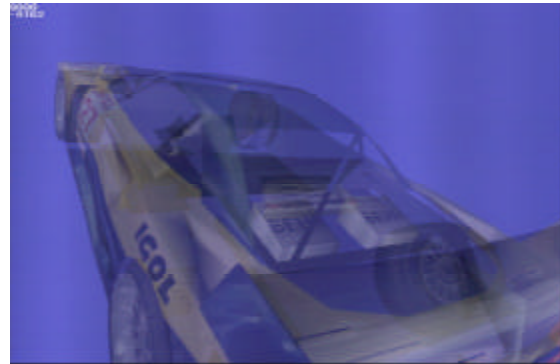
図 8 MipMap テクスチャ

3-1-7. アルファブレンディング (混合処理)

Direct3D の混合処理は、OpenGL、Direct3D と等価で、単純な半透明処理以外に、光学フィルタを通したような画像を作成したり、レンズフレア、煙、遠くにあるオブジェクトを背景に溶かし込むデプスキューイング処理を行うことも可能です。



アルファ値 1.0



アルファ値 0.5

図 9 アルファブレンディング

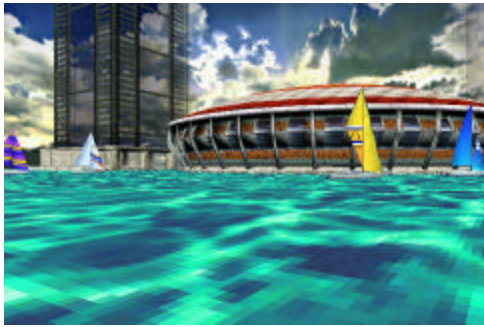
3-1-8. テクスチャフィルタリング

テクスチャフィルタリングには、以下の 4 つがあります。表 3 にそれぞれの特徴などを説明します。

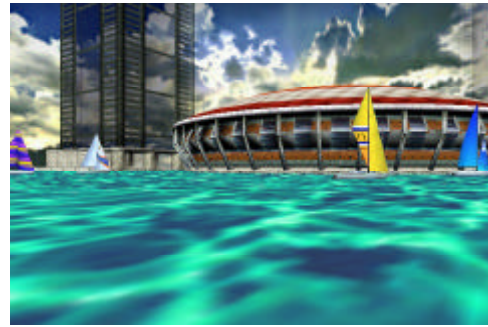
	Point Sampling	B i-L near Filter	Tri-L near Filter	Texture Super Sampling
特徴	テクスチャデータからピクセルデータへの変換が 1 対 1。通常のテクスチャマッピング処理。	対象になるテクセルデータの近傍 4 テクセルからピクセルデータを算出。	M ipm ap を使っているときだけ。前後の M ipm ap に B i-L near Filter をかけて加重平均をとる。	4 倍のサンプリングポイントを使い、より高精度なフィルタリングをする。
長所	最も処理が軽い。	拡大・縮小時のちらつきなどが軽減。	M ipm ap テクスチャの切り替わりがスムーズ。	他の 3 つのフィルタリングとあわせて使うことができる。非常に精度が高い。
短所	縮小したときにテクスチャの絵が崩れる。	絵がぼけたようになる。Non-Tw idded テクスチャの場合、Point Sample の 2 倍の処理時間がかかる。	非常に計算時間がかかる。	通常の 3~4 倍の時間を必要とする。

表 3 テクスチャフィルタの種類

各種テクスチャフィルタリングの効果については、図 10 を参照してください。



Point Sampling



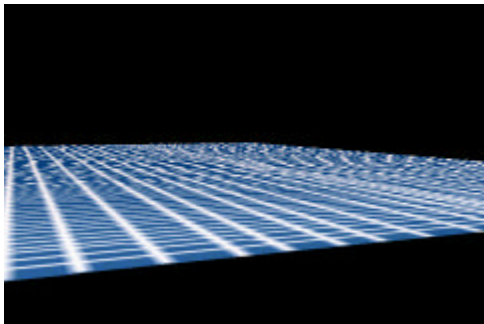
Bi-Linear Filter



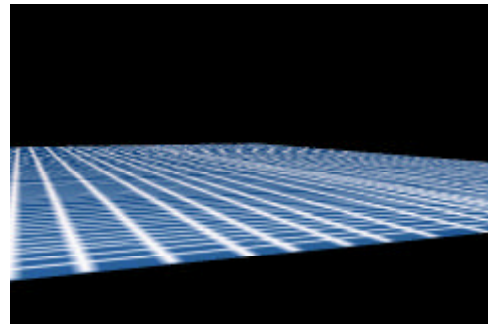
No Tri-Linear Filter



Tri-Linear Filter



No Texture Super Sampling



Texture Super Sampling

図 10 テクスチャフィルタの効果

3-1-9. フォグ

フォグ（霧）の効果を出します。ハードウェアにある程度の計算をさせてしまう方法と、自分で計算して頂点データの一部として設定する方法があります。

ハードウェアに計算させる場合、Z 方向つまり奥行き方向にのみフォグを掛けることができますが、自分で計算する場合は上下方向に霧の効果を出すことも可能です。

ハードウェアに計算させるモードは Lookup Table Mode と呼ばれ、Z 方向のある単位ごとに霧の掛かり方をテーブル化して、その間の Z 位置についてはハードウェアが補間して、霧の掛かり具合を割り出すという方法です。（図 11）

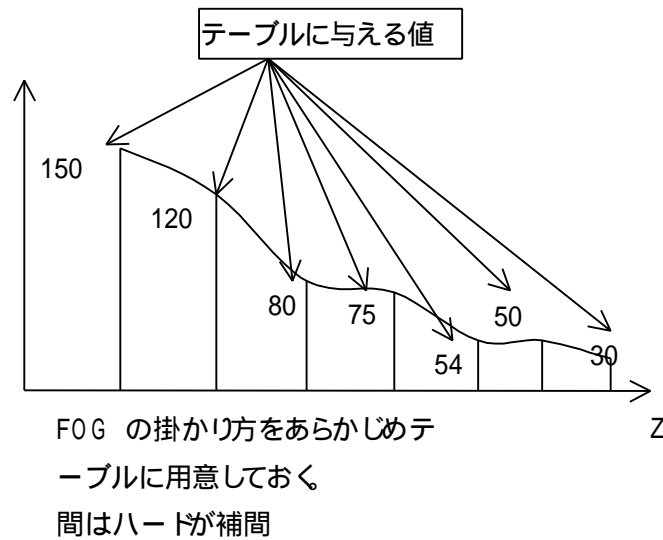


図 11 Lookup Table Mode

一方 Per Vertex Mode は、頂点ごとにすべて霧の掛かり具合を計算し、ハードウェアはレンダリングの処理のみを担当するというもので、Lookup Table Mode のように Z 方向にしか霧が掛からないということはありませんが、すべて自分で計算しなければならないので処理に時間がかかります。

3-1-10. クリッピング

クリッピングには 2 種類あります。

それらは、タイルクリッピングと、ピクセルクリッピングと呼ばれます。

タイルクリッピングにはさらに 2 種類あり、全画面を通して有効な Global Tile Clip とポリゴンごとに設定可能な User Tile Clip があります。

User Tile Clip はクリップエリアの内側を有効にするか外側を有効にするかを切り替えることができます。

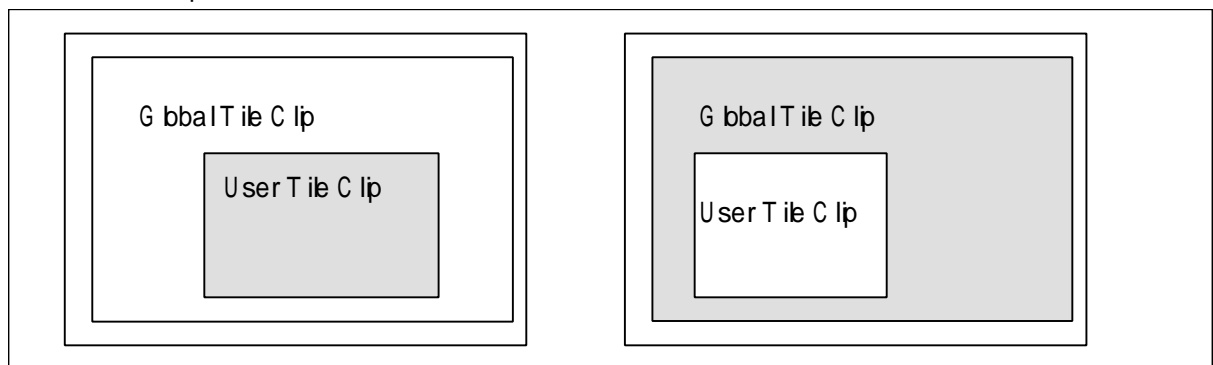


図 12 Global Tile Clip と User Tile Clip

Tile Clip は、32x32 ピクセルを 1 つのタイルとした、PowerVR のハードウェアにあわせたものです。

ピクセルクリッピングは、実際にレンダリングしたデータをフレームバッファに送るときに、今度はピクセル

単位でのクリッピングを行います。当然ながら、ピクセルクリッピングはすべてのモデルに対して有効です。ただし、1 画面に対し2 度のレンダリングが発生するため、描画性能は大幅に低下します。

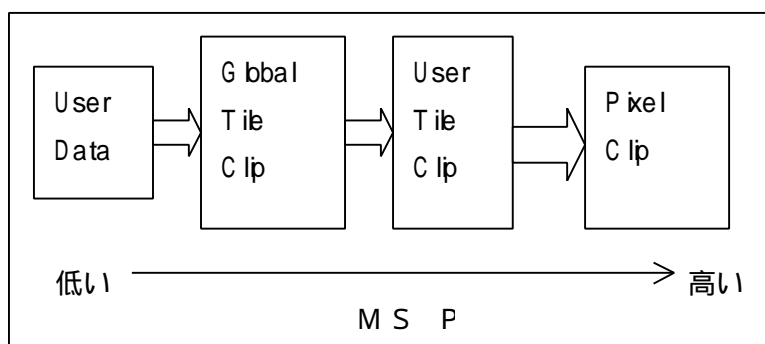


図 13 クリッピング処理の流れ

3-1-11. バッファ

画面用のバッファとして通常のフレームバッファ以外にストリップバッファを用意しています。

・ストリップバッファとは？

ストリップバッファを説明する前にフレームバッファについて簡単に説明し、その後両者の違いを解説します。

フレームバッファは、TV 画面 1 枚分の画像データを保存しておき、ハードウェアはそのイメージを参照して、1 枚の TV 画面を作ります。当然、画面が大きければそれだけ多くのメモリを使ってしまいます。(図 14)



図 14 フレームバッファのしくみ

一方、ストリップバッファでは、横方向は画面サイズ分で、縦に短い横長のバッファを 2 つ使って、走査線の位置にあわせて切り替えて使います。たとえば、走査線が 64 ライン目から 95 ライン目までを描画して

いる間はバッファ1を使い、そのときは、96 ライン目から 127 ライン目までのレンダリング結果をバッファ2 に入れます。96 ライン目から 127 ライン目まではバッファ2 を使って描画し、その間にバッファ1 に 128 ライン目から 160 ライン目までのレンダリング結果をいれます。

概念図を図 15 に示します。

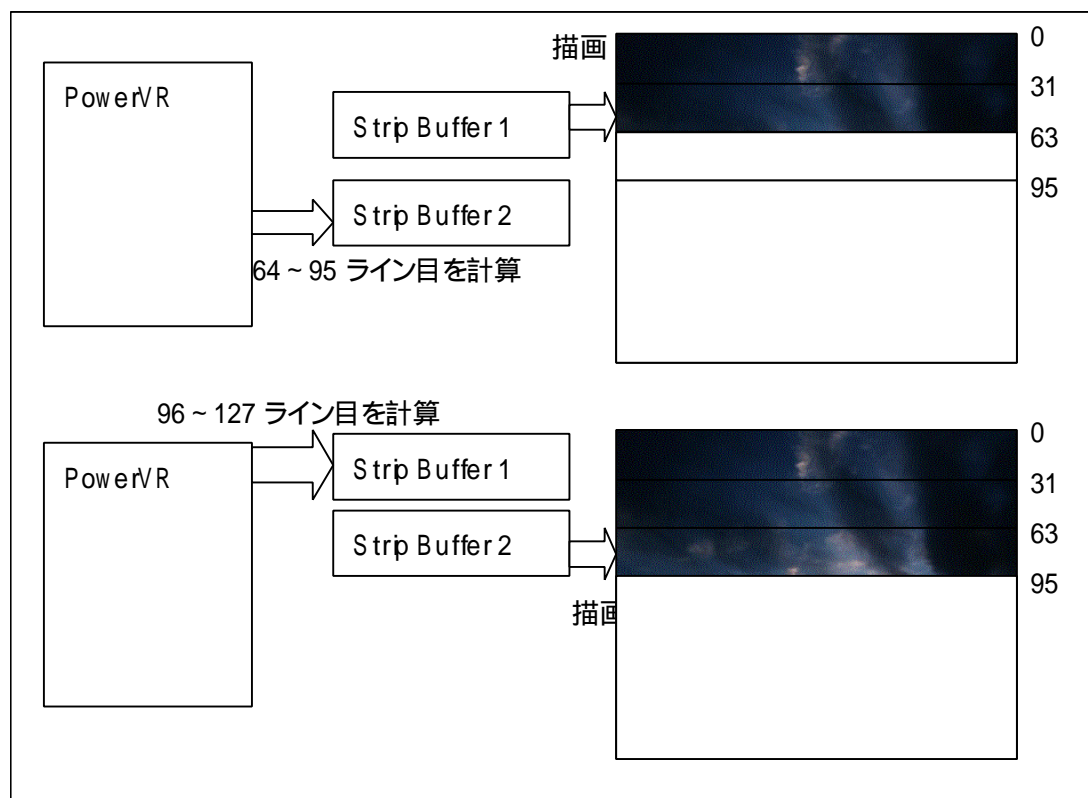


図 15 S trip Buffer

注意点

S-trip Buffer を使用する上で注意しなければならないのは、S-trip Buffer は、そのバッファが TV の描画で使用される時点ですでに準備ができていなければならないということです。

つまり、1 つのストリップバッファ分を描画する時間以内に各ストリップバッファ分のレンダリングをしなければならないのです。ポリゴンの量やテクスチャフィルタなど、レンダリング時間はそこに存在するデータの種類によって変化します。それらすべてが 1 つのストリップバッファ分を描画する時間以内にレンダリングを終了しなければいけません。

ストリップバッファは、1 個所にポリゴンが集中するようなアプリケーションには不向きです。

3-1-12. Image Super Sampling (X & Y スケーラフィルタリング)

PowerVR では、フレームバッファイメージに対してフィルタリングを行う機能があります。この機能を利用することで、画面全体に対しアンチエイリアスを行ったり、インターレース時のフリッカを押さえることが可能

になります。この機能は、X 方向 2ピクセル、Y 方向 3ピクセルを参照して 1ピクセルを描画します。通常は (図 16) の様に 1280×480 で作成された画像を 640×240 に縮小し、フィルタリングをした画面をフレームバッファに格納して表示します。また、X 座標、Y 座標 単独で指定が可能なため、たとえば、640×480 の VGA 画像を、Y 方向のみフィルタリングして縮小し、640×240 ノンインターレスとして表示することも可能になります。

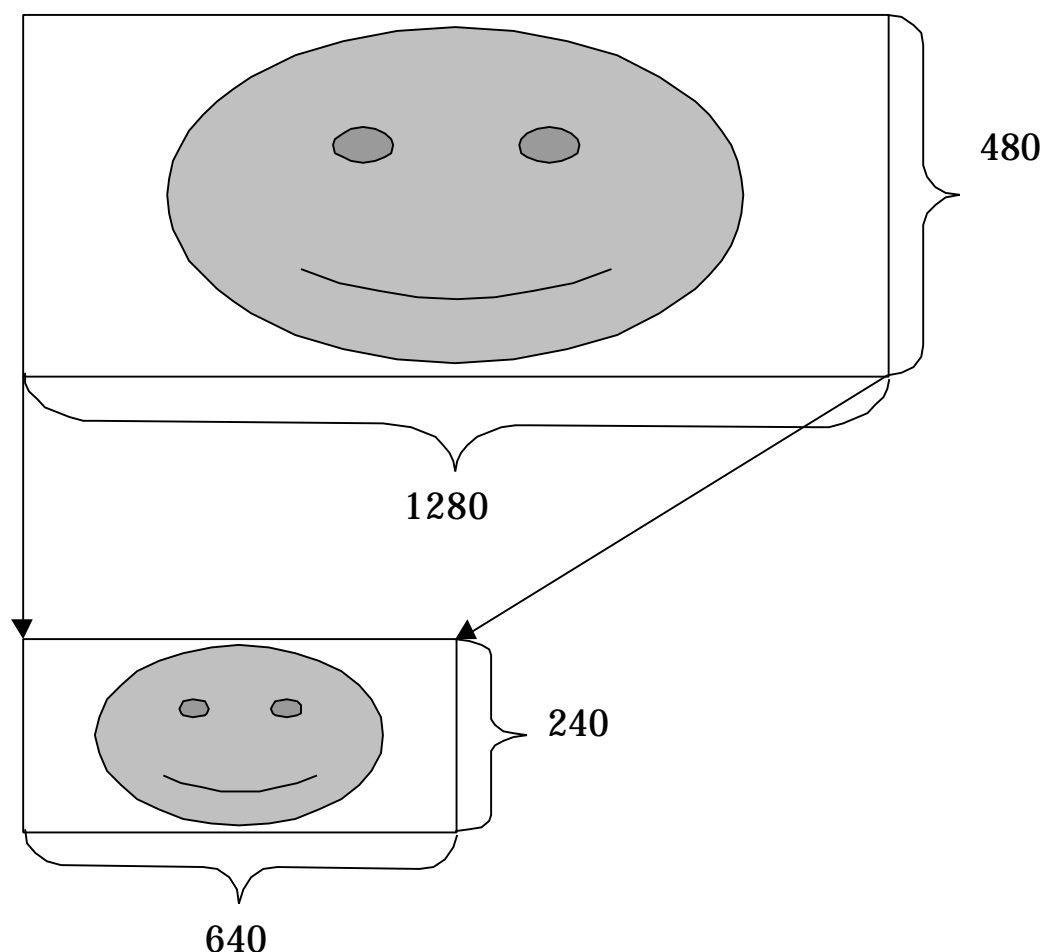


図 16 Image Super Sampling

3-1-13. テクスチャ

3-1-13-1. ピクセルデータ

DREAMCAST のテクスチャ用のピクセルデータ (テクセル) として使用できるフォーマットは次のとおりです。

形式		詳細
R G B	1555	1 ビット、RGB 各 5 ビット
	565	R5 ビット、G6 ビット、B5 ビット
	4444	RGB 各 4 ビット
YUV		YUV 各 8 ビット
Bump Mapping		SR 各 8 ビット
Palette	4bit	各パレット16 色
	8bit	各パレット256 色

表 4 テクスチャのピクセルフォーマット

パレットは 1024 色まで持てます。

3-1-13-2. テクスチャデータ

- ・ テクスチャデータのサイズ

テクスチャデータのサイズは非常に限定されており、基本的には、つまり、8、16、32、...、512、1024 ということになります。

$$2^n \quad (3 \leq n \leq 10)$$

縦方向と横方向で同じサイズにする必要はありません。

通常テクスチャは Twiddled 形式という形式を使いますが、この形式を使わない場合は、

横 (U) 方向 : $32 \cdot n$ $(1 \leq n \leq 16)$

縦 (V) 方向 : $32 \cdot m$ $(1 \leq m \leq 32)$

Stride というモードがあります、Stride モードのテクスチャサイズは、

横方向が 32、64、96、128、192、...、512

縦方向が 32、64、96、128、192、...、992、1024

になります。ただし、縦方向は全画面ですべて同じサイズになります。

(Twiddled 形式の場合はこのような制限はありません。)

- ・ データ形式

テクスチャのフォーマットは一般的に Twiddled と呼ばれる変則的な形式が取られます。

Twiddled 形式は、テクスチャデータへの高速なアクセスをするために作られたフォーマットです。それ以外に、一般的なスキャンラインフォーマットもサポートしています、下の表 5 に DREAMCAST が持つテクスチャデータの形式の比較を示します。

TW DDLED 形式	NON-TW DDLED 形式
-------------	-----------------

形式		<p>例 $l=16p$ $ke l/2=8p$ $ke l$ の場合</p>
	<p>サイズ</p> $2^n \times 2^m, 2^n \times 2^m$	<p>サイズ</p> $2^n \times 2^m, 2^n \times 2^m, 32 \cdot k \times 32 \cdot l$
	<p>圧縮</p> <p>可能</p>	<p>圧縮</p> <p>不可能</p>
	<p>特徴</p> <p>Bi-Linear、Tri-Linear フィルタなどが可能。</p>	<p>特徴</p> <p>フレームバッファへの出力をテクスチャ領域に描画した時に使用。 Bi-Linear フィルタは可能。</p>
	<p>速度</p> <p>速い</p>	<p>速度</p> <p>遅い</p>

表 5 Twiddled 形式

- ・ 圧縮

Twiddle形式を用いた場合、テクスチャを圧縮することができます。

テクスチャの圧縮は、VQ (Vector Quantization:ベクトル量子化) と呼ばれる技法を使用します。データの圧縮率やクオリティなどはデータの性質によって変わります。

圧縮の仕方を説明するために、例として図 17 のような 8×8 のモノクロのテクスチャを圧縮する場合を考えてみましょう。

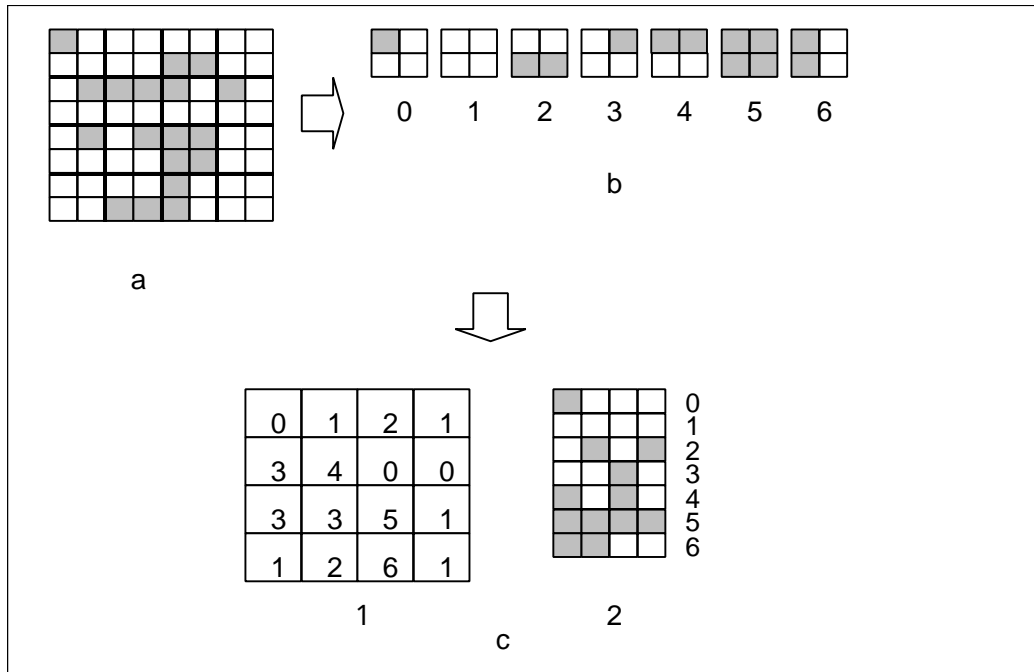


図 17 VQ 圧縮の原理

まず、図 17 の a は元絵です。この元絵をわかりやすくするために、2×2 の格子で切ってみると、この絵は b のように 7 つの 2×2 の格子のパターンでできているのが分かります。

つまり a の絵は、c のように、どの 2×2 の格子がどのような配列で並んでいるのかという情報に整理できます。これが VQ 圧縮の原理になります。

図 17 の c,1 を Index 情報、c,2 を CodeBook と呼びます。

DREAMCAST で使われる VQ は、1 つのテクスチャに必ず 256 個の CodeBook を必要とします。

CodeBook には 16 ビットのカラー形式で収められるので、VQ 圧縮された 1 枚のテクスチャデータには最低でも

$$4 \times 2 \times 256 = 2048$$

2048 バイト必要とします。一方 Index 情報は CodeBook が 1 つにつき 256 (= 2⁸) です。つまり、4x2 バイトを 1 バイトにする圧縮方法です。

ただし、前述のように、2048 バイト未満のテクスチャを圧縮すると、CodeBook の分だけ逆にデータサイズを必要としてしまいます。

$$2048 + \frac{n}{8} < n \times 2$$

$$n - \frac{n}{16} = \frac{15}{16}n < 2048$$

$$n < \frac{2048 \times 16}{15} \cong 2184 > 32 \times 32(\text{pixel}) \times 2(\text{byte})$$

つまり、32x32 以下のテクスチャデータでは逆にデータサイズが大きくなってしまいます。

さて、VQ 圧縮方式は、8x8 から 1024x1024 までのサイズに対応していますが、実際の使用に耐えられるサイズは、大体 64x64 から 256x256 程度までになるでしょう（それ以上のサイズはテクスチャ自体が大きくて使いづらいでしょう）。

VQ 圧縮を有効に使用するには、テクスチャのサイズに気を配る必要があります。

CodeBook のサイズ	256 個 (2048 バイト)
1 ピクセルのデータサイズ	16 ビット (2 バイト)
使用可能なサイズ	8x8 ~ 1024 x1024 32x32 以下はかえって効率が落ちる。
使用できるフォーマット	Twiddled

表 6 DREAMCAST の VQ 方式の特徴

3-1-14. Bump Mapping

バンプマッピングは、テクスチャデータに、ピクセルデータの代わりに表面の凹凸情報を持たせ、本来は凹凸がないのにもかかわらずあたかも凸凹しているかのように見せる技術です。

本来のポリゴンの上に仮想的な凸凹したポリゴン形状を仮定し、各テクセル上での仮想面の法線ベクトルを数値化して、テクスチャデータとして記述します。

したがって、バンプマッピング用テクスチャのテクセルデータは、カラーデータではなく、そのテクセルの法線ベクトルを 2 つの角度 S、R で表します。

この 2 つの角度、S と R を図 18 に説明します。

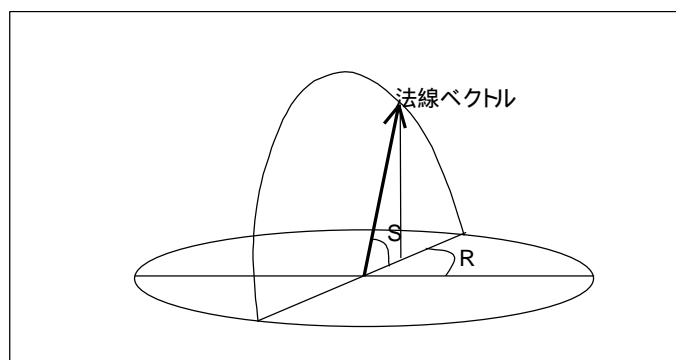


図 18 バンプマップテクセルの意味

また、単に凸凹の形状のみが表現できればいいので、この法線ベクトルは常に単位ベクトルとして輝度情報が計算されます。

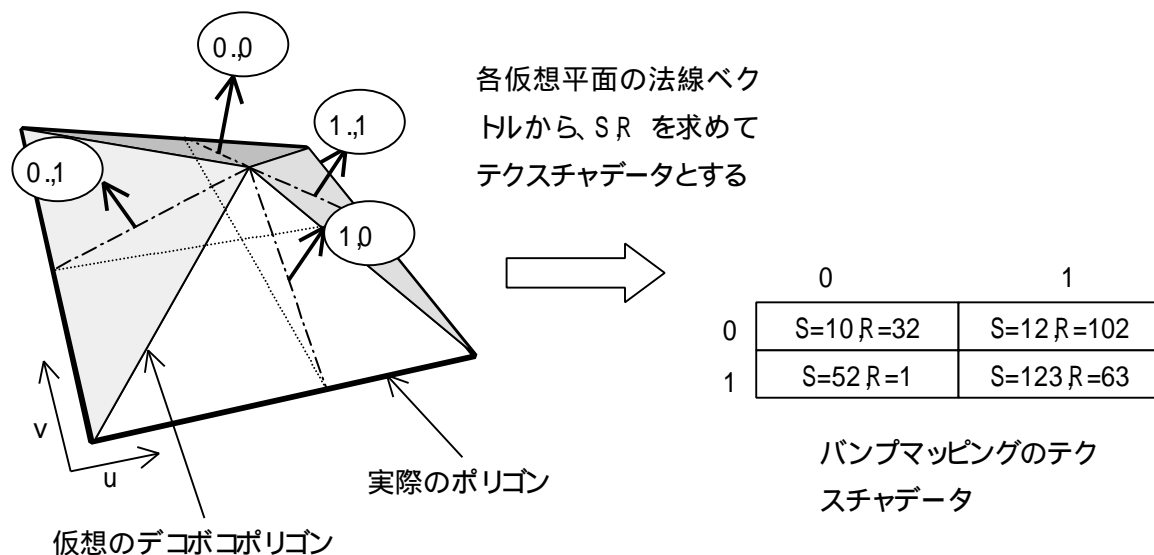
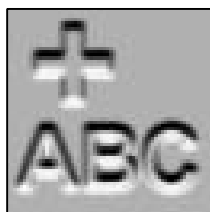


図 19 バンプマッピングの原理

バンプマップテクスチャはバンプマッピング処理されると 値のみが変化する単一色テクスチャとなります。このままではオブジェクトを凸凹に見せることはできませんので、他のテクスチャとあわせて使用します。

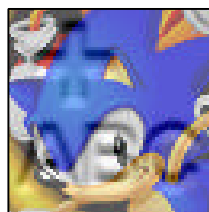
たとえば、テクスチャをデコボコに見せるには、バンプマップ処理されたテクスチャを貼ったポリゴンを描画し、その上に半透明テクスチャを描画します。そうすることで、テクスチャを凸凹に見せることができます。(図 20)

Bump Mapped ポリゴン

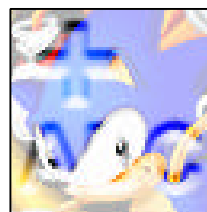


Decal Alpha
Base Color = 0x0000 0000

Textured ポリゴン + Bump Mapped ポリゴン



Decal Alpha
Base Color = 0x8000 0000



Modulate Alpha
Base Color = 0xFFFF FFFF

<Blend Function>

SRC = SRC Alpha, DST = Inverse SRC Alpha

図 17 テクスチャ + バンプマッピング

3-1-15. ストリップ

ポリゴン形状として、独立三角形ポリゴン、および独立四角形ポリゴンをサポートしています。

それ以外に、ポリゴン処理の際の頂点計算を高速化する手段として、OpenGLなどで知られた Triangle Strip という技法を使用できます。

Strip を使用すると、ポリゴンのデータ量を削減するだけでなく計算時間を短縮できます。

ストリップの長さとしては、無限長のストリップデータにも対応しています。

独立三角形ポリゴンと、ストリップポリゴンの違いについては、下の図 21 をご覧ください。

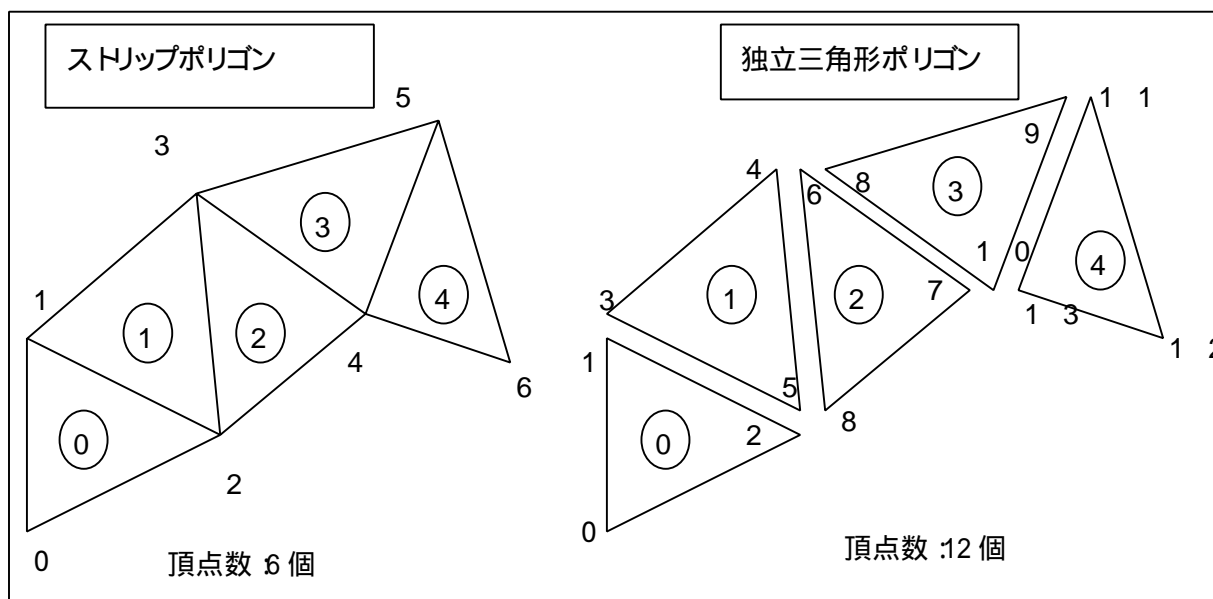


図 21 ストリップ系ポリゴンと独立系ポリゴンの違い

例えば、ポリゴン1は、ストリップポリゴンの場合は頂点番号 123 で構成されるのに対して、独立三角形ポリゴンの場合は頂点番号 345 で構成されます。ストリップポリゴンでは、前のポリゴンのデータを作成したときに使ったデータを流用するので、その分計算をする手間が省けます。しかしながら、もし前のポリゴンが隠れて見えない場合でも、前のポリゴンのデータを必要とするような構造なので、常にストリップ単位で計算しなければなりません。

ストリップを使用するときは、このような無駄がない構造にしなければなりません。

また、上図 21 のモデルでは、ストリップのモデルは全部で6頂点で構成されるのに対して、独立三角形のモデルでは倍の12頂点で構成されます。ストリップポリゴンは必要なデータ量が少ない構造でもあるのが分かります。

3-1-16. Sprite

四角形のスプライトをサポートしています。ただし、スプライトデータは内部的にパンチスルーポリゴンとして扱われます。パンチスルーポリゴンは描画に要する時間は不透明ポリゴンよりも若干多くなります。

3-1-17. カラークランプ

フレームバッファに描画する前に、クランプカラーと比較して描画することが可能で、比較結果はその条件によってクランプカラーに固定されます。これによりフェードイン / アウトの効果を出すことが可能になります。

3-2. サウンド機能

3-2-1. サウンド機能の概要

PCM 音源内蔵

128 ステップの DSP を搭載

デジタルミキサー搭載

外部 Digital Audio 入力を 1 系統サポート

PCM のデータフォーマットは 8,16bit Linear / 4bit ADPCM

(ADPCM の方式は YAMAHA のオリジナル方式)

スロット別の独立した LFO (Low Frequency Oscillator) を搭載

4 セグメントの EG (Envelope Generator) を 64ch 搭載

4 セグメントの EG によってカットオフ周波数を時変動する LPF を搭載

フォワードループ機能搭載

最大 64 音同時発音

最大 1 オクターブまでピッチチェンジ可能な ADPCM

3-2-2. ADPCM とは？

ADPCM の話をする前に音について簡単に説明し、ADPCM の理論的な背景となっている PCM の説明をします。その後 ADPCM の説明をします。

音というのは、空気中を伝わるいろいろな波長の波の合成であることが分かっています。

この波形を絵にすると

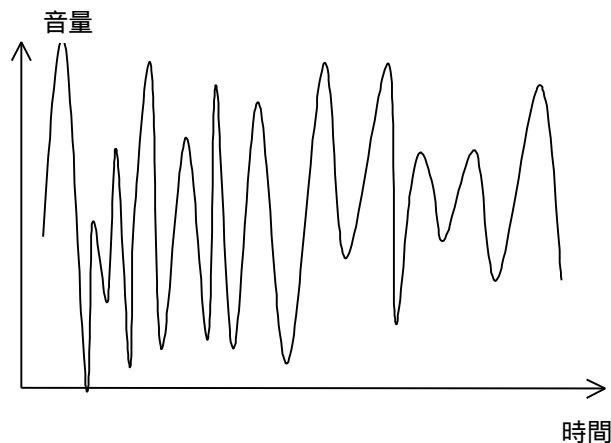


図 22 音の波形

というような、時間と音量の関係で表すことができます。

さて、PCM とは、ある特定の時間でどれだけの音量だったかを記録する方法で、たとえば 44.1kHz の PCM では、1 秒間を 44100 等分して、 $1/44100$ 秒のときの音量はいくつ、 $2/44100$ 秒のときの音量はいくつといった具合に記録されます。

当然、1 秒の音には 44100 個の測定点が存在し、それらすべての測定点での音量が記録されます。通常ここで測定された音量値はアナログな値で、たとえば 16db という感じです。これをデジタル化し、たとえば 0 から 255 までとか、0 から 65535 までというように範囲を決めてその範囲の値で記録します。この例の 44.1kHz という数字をサンプリング周波数、音量のデジタル数値の範囲を量子化ビット数（例えば 0 から 255 (= 2 の 8 乗) までならば 8 ビット）といいます。

詳しくは下図（図 23）をご覧ください

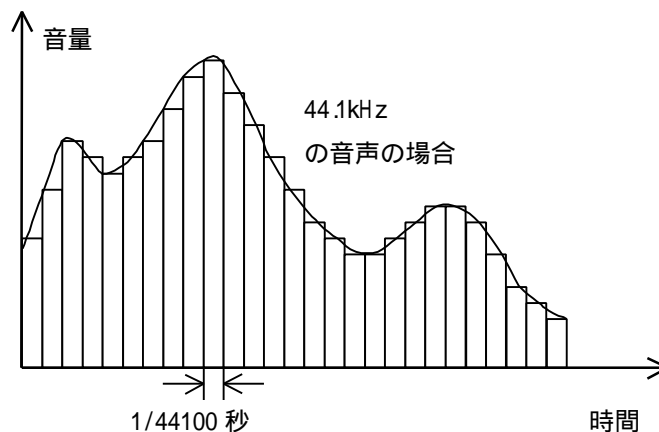


図 23 PCM のデータとは？

さて、ADPCM はどうなのでしょう？

ADPCM 方式とは、ある時間の音量を測定値で表すのではなく、前の結果から推測される音量値と実測値との差をその時間での音量の情報として記録する方法です。

測定した音量値よりも予測した音量値との差分のほうが変化する値の幅が小さいので、その分少ない記録量で PCM 方式と同じ波形を再現できます。

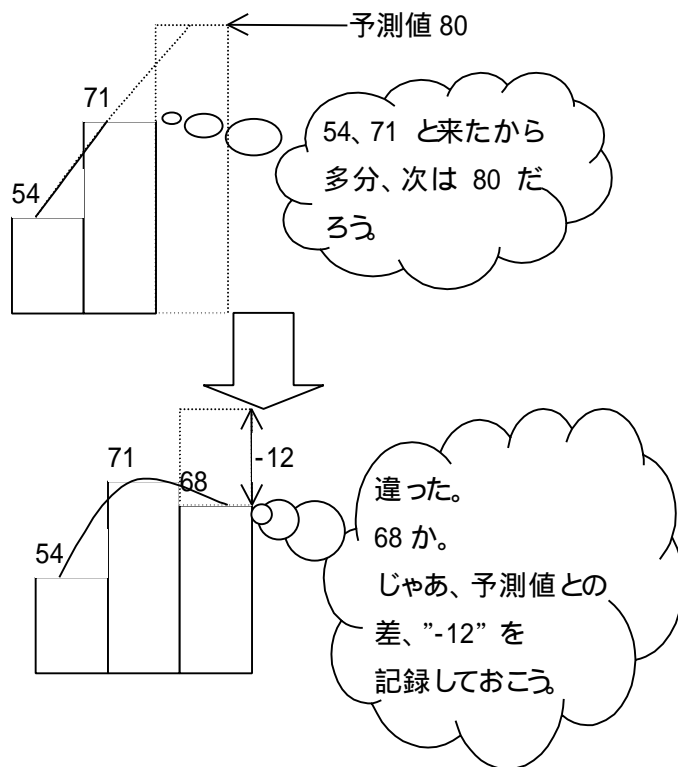


図 24 ADPCM とは？

上図 (図 24) を見て分かると思いますが、人によっては 54、71 という数字が来た後に、80 ではなく 78 という人もいれば 75 と予測する人もいるかも知れません。

そうです、この予測の方法は実際さまざまであり、例えば有名なものと Philips 社の予測方法は一般に XA 規格の一部として知られています。ただし圧縮したときと解凍したときで同じ予測の仕方をしなければ、元の音声の波形と圧縮解凍したときの波形が異なってしまいます。そのため ADPCM の圧縮ツールが手元にあるからといってそれを安易に使うことはできないのです。

DREAMCAST の予測の仕方は DREAMCAST で使用しているサウンドの LSI を開発した YAMAHA 社のオリジナルの予測方法を使用しています。

この YAMAHA 方式の予測方法を使った ADPCM の圧縮をするためのツールはセガから提供されます。

3-3. パリフェラル

DREAMCAST には 4 つのパリフェラルの差込口があり、それぞれの差込口にはアナログコントローラや GUN などの入力装置を接続できるほか、専用コントローラには、液晶の小さな画面を備え外部バックアップメモリとしての機能を持つビジュアルメモリという装置を装着することもできます。

ビジュアルメモリはそれ単体でも使うことができ、単体時にはたまごっちなどの携帯ゲーム機にもなります。

また、コントローラを接続した場合、それぞれのプレイヤーのコントローラについているビジュアルメモリにはそれぞれ別々のメッセージを表示したり、画面を表示できるので、複数プレイヤーが同時に遊んだときに個々のプレイヤーに違う情報を伝えることができます。



図 25 コントローラとビジュアルメモリ

3-4. GD-ROM (仮称)

3-4-1. GD-ROM とは？

DREAMCAST のCDROM はGD-ROM (G iga by te D isk ROM) と呼ばれ、独自のフォーマットを採用しています。

回転速度が一定の機構で、同じ読み取り時間でも、データの読み取り部分（レコードの針に相当する部分）が外側ほど、大量のデータを読み取ることができます。（通常の CD-ROM ドライブは線速度一定という形式を使用しているので、読み取り部分がどこにあっても同じ時間内に読みとれるデータの量は一定です。その代わり、読み取り部分の位置によって回転速度が変化します。ディスクドライブの部分に耳を当てれば、ディスクの回転音が違うのがわかるでしょう。）つまり、一度に大量のデータを読み取る必要があるときは、そのデータを GD-ROM の外側に置くほうが高速に読み取れます。

このような理由で、DREAMCAST で採用している CD-ROM は、4～12 倍速と速度が変化します。最近の CD-ROM ドライブでは、“12～24 倍速”といった表記のものがありますがこれらは同じ方式を採用した CD-ROM ドライブです。

3-4-2. GD-ROM の構造

ディスクは構造上 3 つのブロックに分かれています。

一番内側を単密エリアと呼び、ここには一般的な CD-ROM の規格に従った形式でデータや CDDA が収められます。

当然、CD プレイヤーやコンピュータでこのエリアを参照することが可能です。

このエリアは 4 分間程度しかないので、例えば

「このディスクはセガ DREAMCAST 用のゲームディスクです。ゲームのデータが入っていますので再生しないでください。」

といった、SATURN でも行っていたようなメッセージを入れる場合が想定されています。

また、DREAMCAST 用のソフトからこの領域にアクセスすることはできません。

次のエリアは高密エリアで、

ここにゲームのデータやプログラム、(ゲームで使用する) CD-DA のデータなどを収納することができます。

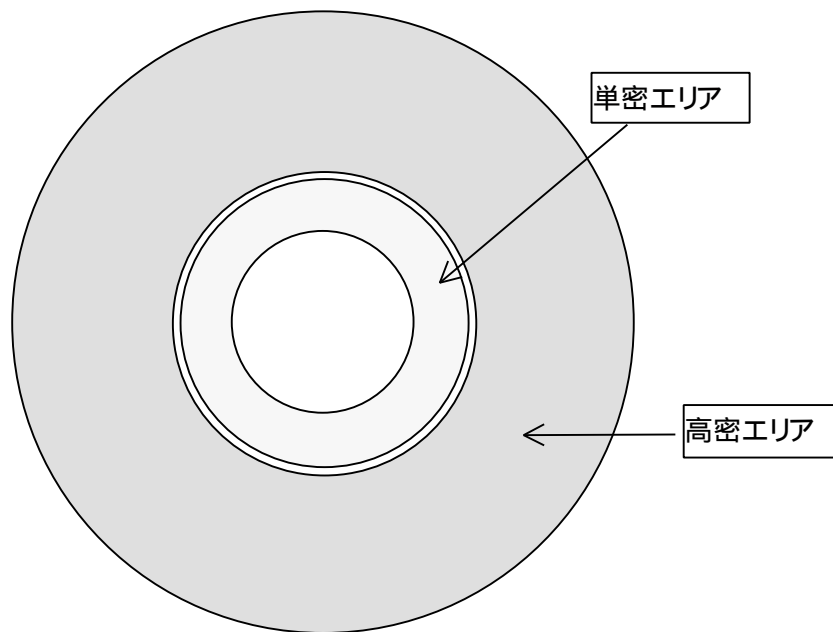


図 26 GD-ROM の構造

3-5. CPU

DREAMCAST には HITACHI 製の SH7091 という CPU が搭載されます。

SH7091 は外部クロック 100MHz 内部クロック 200MHz で動作し、

FPU (浮動小数点計算モジュール)

MMU (仮想メモリ管理モジュール)

DMA コントローラ

タイマ

命令 8KB データ 16KB のキャッシュ

5 段のパイプラインと 2 本のスーパースカラ

などの機能を持っています。

3-5-1. FPU

FPU は、小数点を含む数値の加減乗除を高速に処理するためのモジュールで、3D の座標計算など通常は時間がかかる数値処理を専門に行います。

3-5-2. MMU

MMU は実メモリの番地に対して仮想的な番地を割り当て、あたかも広大なメモリがあるように振る舞うことができます。

DragonOS はこの MMU モジュールを使用し、実メモリ空間をある程度隠蔽し、本来は分断されているメモリを仮想的に連続したメモリのように見せることができます。

使う側では、実際にはメモリが分断されているということを気にする必要はありません。
MMU 用のアドレス変換バッファ (TLB) として、
命令 (データ用としても利用できる) 4 エントリ
データ 64 エントリ
を備えています。

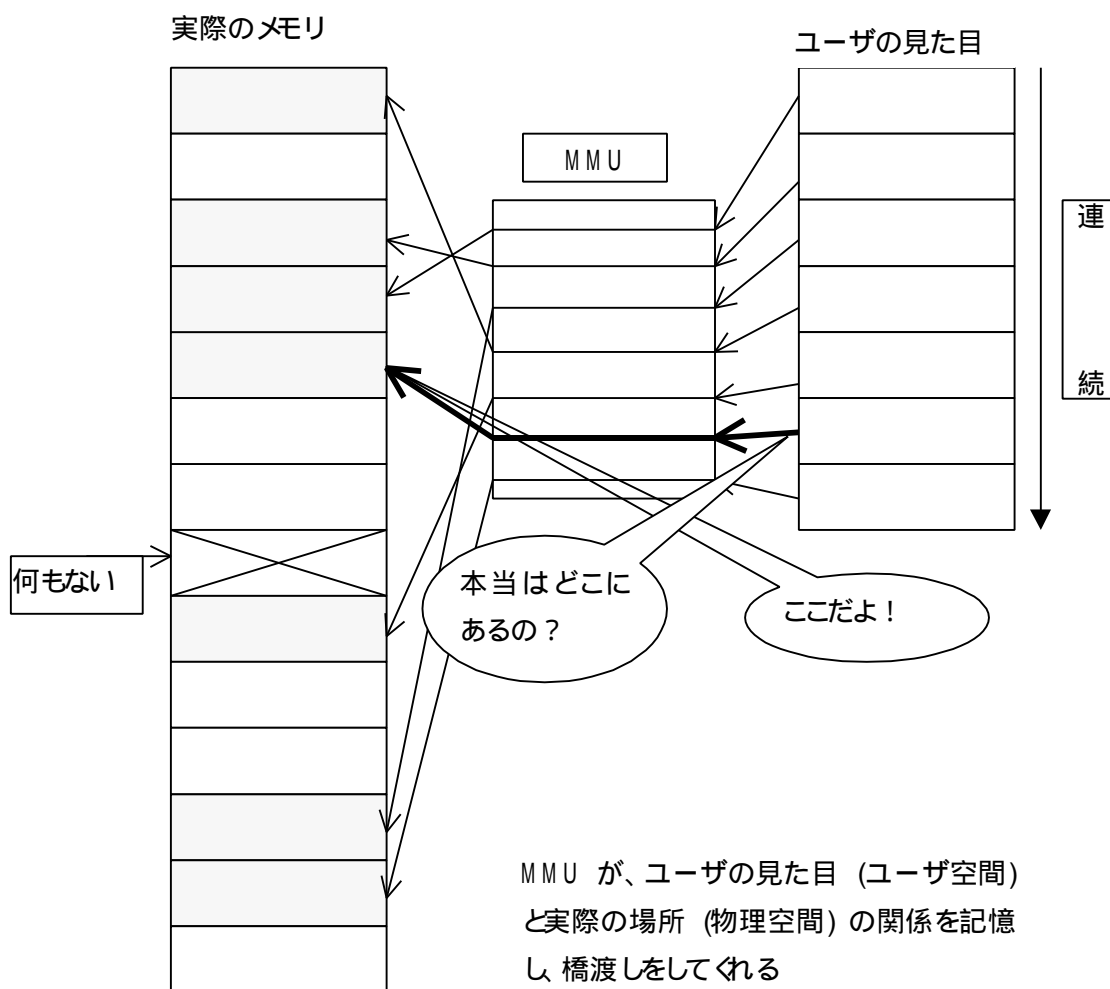


図 27 MMU の働き

また、SH7091 は他の MMU を積んだ CPU と異なり、ページサイズ (MMU が扱える最小のサイズ) が可変で、
1k、4k、64k、1M
単位でこれらのページサイズの混在を認めています。たとえば、
0x00000000 ~ 0x0FFFFFFF はページサイズ 1K
0x10000000 ~ 0x1FFFFFFF はページサイズ 4k

0x20000000 ~ 0x7FFFFFFF はページサイズ 64k

0x80000000 ~ 0xFFFFFFFF はページサイズ 1M

とすることができます。

単純にすべてのメモリ空間を 1k バイトのページサイズとした場合、DREAM CAST のすべてのメインメモリをカバーするには、

$$16 \times 1024 = 16384$$

の変換テーブルが必要になります。しかし、SH7091 には高々 68 個の変換テーブルしかありません。内部で持っているテーブルに自分の探したいアドレス変換情報が存在しないとき、MMU はメインメモリ上に確保してあるマスターの変換テーブルを探し、内部のバッファに取り込みます。

しかし、この取り込み作業は内部バッファを参照するのに比べて非常に多くの時間がかかります。この例の場合、頻繁にマスターテーブルを見に行くため、処理の時間としては多くの無駄が発生します。

では、ページサイズを思い切って 1M にするとどうなるでしょう？

ページサイズが 1M のとき、必要なテーブルのエントリは

16 個

十分に内部バッファに入る大きさです。

ところがこの場合別の問題が発生します。

MMU は、メモリのアロケートが指示されたとき、ページサイズでメモリを確保します。

つまり、たとえば 16 バイトのメモリを確保するために、1M バイトの実メモリを確保するのです。

それはそれで非常に無駄が多いことがお分かりになるでしょう。

つまり、大量のデータを必要とするところ、例えばムービーのバッファ領域などは大きいページサイズで、プログラムの領域やテンポラリのデータ領域（ヒープ領域）などは小さ目のページサイズにすることで、MMU の内部変換テーブルを有効に利用し、かつメモリも上手に使用することが可能なのです。

しかしながら、通常これらの仕事は、OS などの管理を担当するモジュールが行うため、それほど意識する必要はありません。

3-5-3. DMA

DMA は大量のデータのある場所から別の場所に一気に転送するもので、DMA がデータを転送している間、CPU の計算機能などは平行して動くことができるので、転送が終了するまで処理が中断することがありません。そのため時間を無駄なくしようすることが可能です。

SH7091 には DMA が 4 つありますが、それぞれのチャンネルは目的ごとに異なります。

4 つのうち 2 つは、システムが使用するため、ユーザレベルで使用できるのは 2 つ分です。システムで予約している DMA のうちひとつは、高速にテキストチャメモリにアクセスするための DMA で、メインメモリから、テキストチャ RAM への一方向の転送にしか使用できません。システムで予約しているもうひとつの DMA は、各外部デバイスとメモリとの間で転送を行うもので、これにはテキストチャ RAM からメインメモリへの転送も含まれます。

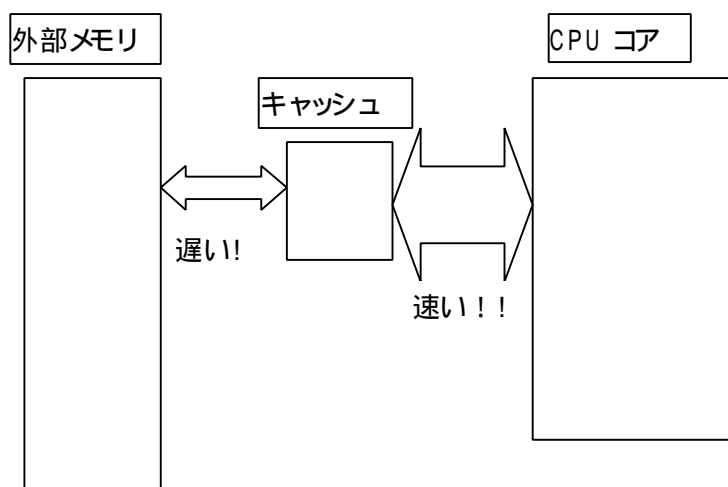
3-5-4. タイマ

SH7091 は内部に独立して動くタイマを内蔵しています。

3-5-5. キャッシュ

SH7091 は命令キャッシュ 8KB、データキャッシュ 16KB を持ち、時間がかかる外部メモリへのアクセスを減らすことができます。

DREAMCAST の場合、キャッシュにアクセスするのに比べ、外部バスにアクセスすると 10 倍も余計に時間がかかります。



キャッシュは、外部メモリの内容をアドレス（場所）の情報と一緒に保存している。（沢山ほどよい。）

最初は（時間がかかる）外部メモリにアクセスしても、次に同じところの情報を引き出すときは、キャッシュに溜まっている内容を読みに行くので速い。

図 28 キャッシュの利点

3-5-6. パイプライン・スーパースカラ

SH7091 にはスーパースカラと呼ばれる機能が備わっています。

これを簡単に説明すると、昔の CPU は、命令（受注）を受けてから、その処理が完了（納品）するまで、別の命令を受け付けることはできませんでした。それは、すべての工程が 1 つの機械によって処理されていたからと考えるとよいでしょう。

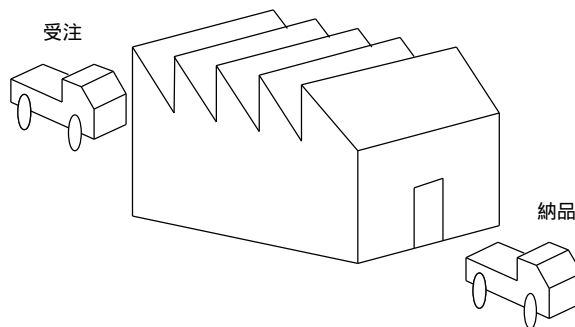
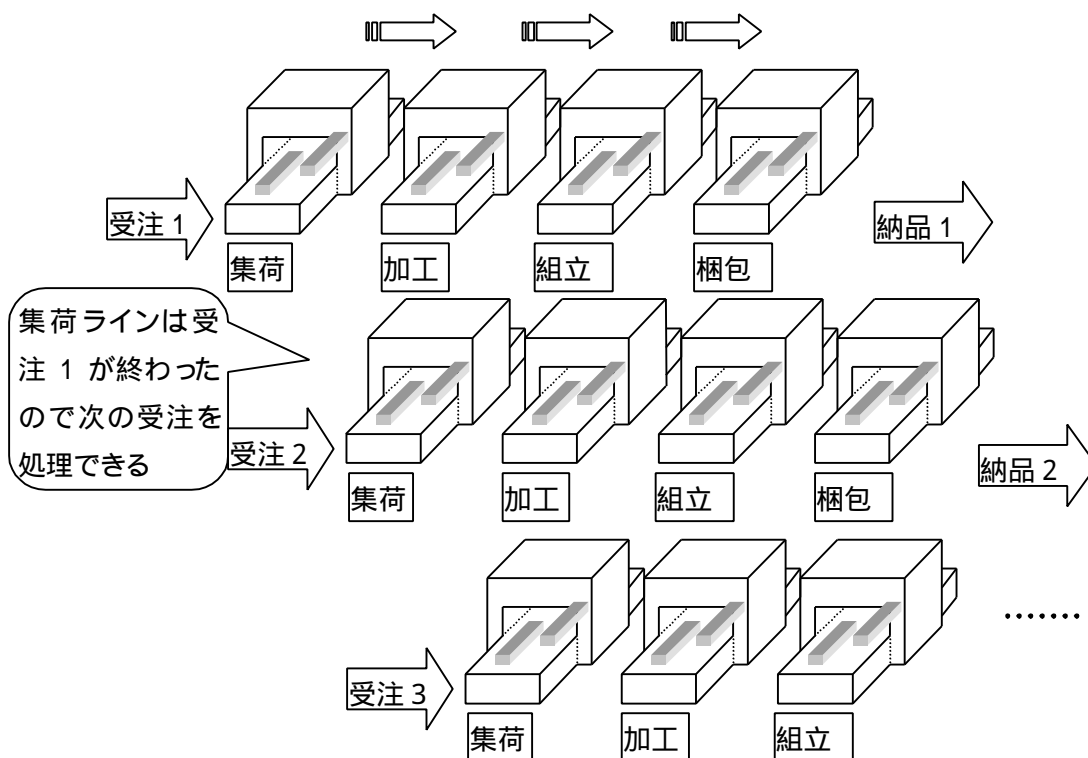


図 29 初期の CPU

さて、それでは効率が悪いので、作業工程を見直して、いくつかの独立した工程に分けました。
図 30 に従えば、各工程を専門で行う機械を導入し、機械ごとに仕事を振り分けたと考えてください。
これで、見かけ上複数の受注を捌いているように見えます。



たとえば、集荷のラインは、自分の作業が終われば暇になるので、次の受注を処理できる。加工や組立、梱包のラインもそれぞれ独立に動ける。

図 30 パイプライン処理 (図では 3 本のパイプライン)

SH7091 の 5 本のパイプラインとは、このように 5 つの命令を同時に処理する能力のことを指します。ただし、気を付けなければいけないのは、前の注文で作った品物を使用したものを次の注文の材料として使いたい場合で、この場合は、次の注文の材料は前の注文が終わるまでは、取り掛かることはできないので、作業効率が落ちます。(つまりそういう作り方を極力避けるのがよいパフォーマンスを出すためのよい方法なのです。)

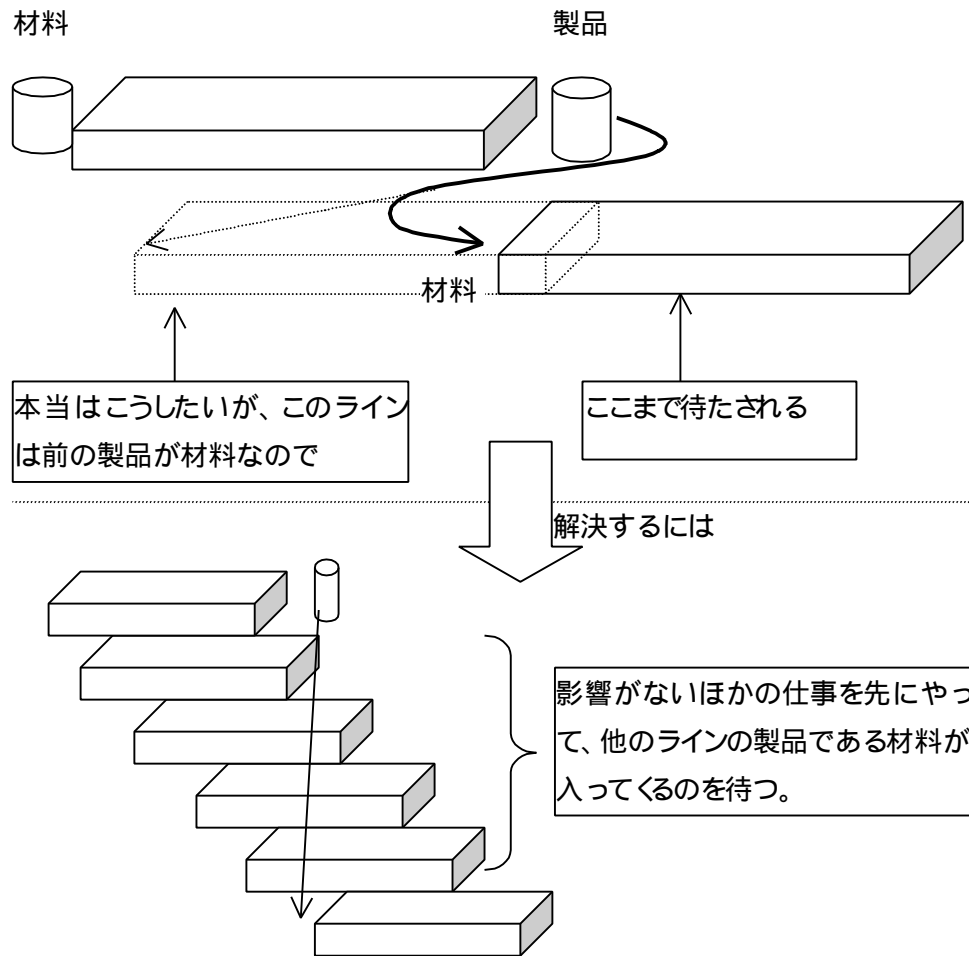


図 31 効率のよいパイプラインの使い方

さて、さらに効率を上げるにはどうしたらいいでしょう？

一番簡単な方法としては、工場をもう1つ建設してしまうことです。

SH7091 のスーパースカラとは、この工場を増やすという方法に当たるもので、2 つの命令を全く同時に捌くことができるのです。

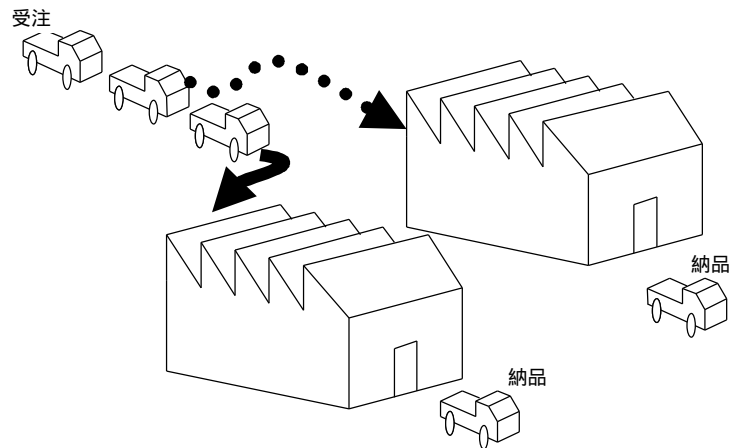


図 32 スーパースカラ

当然各工場は、工程が分離（パイプライン化）しており来た受注を効率よく捌くことができるようになっています。

ただし、この場合も先に挙げたような前の注文で作った商品を後で使うという場合には作業が止まってしまいます。

このことは、CPU を効率よく使うためのノウハウなのです。

もうひとつの SH7091 の高速化テクニックはだらだらやっていた仕事をてきぱきと終わらせる（クロックアップ）ことです。SH7091 のクロックは 200MHz、SATURN で使用していた SH2 が 28MHz で動いていたことを考えると、実に 7 倍もてきぱき動いていることになります。