



第4部

*Kamui2*ライブラリ編

この章では、最新のバージョンから公開された
Kamui ドライバについて、1 から 2 への変更の
経緯を含めて解説します。

目次

1	Kamui2 ライブラリについて	4
1.1	Kamui ライブラリの概要	4
1.2	Kamui ライブラリの位置付け	4
1.3	Kamui ライブラリの機能	4
1.3.1	ユーザープログラムが行う処理	5
1.4	基本ポリゴン	5
1.4.1	Shading カラーについて	6
1.5	Kamui ライブラリの座標系	6
1.6	Kamui ライブラリのシーケンス	7
1.7	ポリゴンリスト	7
1.8	ポリゴンパラメータ	8
1.9	VertexParameter	8
1.10	カラーフォーマット	9
1.11	UV フォーマット	9
1.12	タイプ一覧	9
2	基本的な処理の流れ	12
2.2	頂点データバッファと各種内部バッファ	15
2.2.1	ダイレクトモードとバッファモードについて	16
2.3	レイテンシモデル	17
2.3.1	3 Vレイテンシモデル	18
2.3.2	2 Vレイテンシモデル	20
2.3.3	処理オーバーした場合の対応	21
2.4	V-Sync 同期 / 非同期モード	22
2.4.1	(3 Vレイテンシモデルの) 同期モードの場合	22
2.4.2	(3 Vレイテンシモデルの) 非同期モードの場合	22
3	マルチパス	23
3.1	Auto-Sort / Pre-Sort の切り替え	23
3.2	半透明グルーピング	23
3.3	Zクリッピングプレーン	24
4	Kamui ライブラリの基本的な使用法	25
4.1	初期化	25
4.2	テクスチャの登録	25
4.3	レンダリングフロー	26
4.4	頂点の登録	28
5	PAL 拡張ディスプレイモード	30
5.1	PAL 拡張ディスプレイモードについて	30
5.2	PAL 拡張設定情報	30
5.3	PAL 拡張モードコールバック関数	31
5.4	PAL 拡張モードコールバック関数ポインタ登録関数	31
5.5	PAL 拡張モード用構造体	31
5.6	画面設定手順	31
5.7	注意、制限事項	32
5.8	PAL 拡張モード設定情報一覧	33
6	テクスチャフォーマット	34

6.1 Kamui のテクスチャフォーマット	34
6.2 Twiddled 及び Twiddled ミップマップフォーマット	36
6.3 VQ 及び VQ ミップマップフォーマット	38
6.4 スモールVQ及スモールVQミップマップフォーマット	41
6.4.1 <i>small VQ</i> テクスチャ	41
6.5 Palettized 4bpp/8bpp フォーマット	44
6.6 Rectangle フォーマット	47
6.7 Stride フォーマット	47
6.8 BUMP-Mapping フォーマット	49
7 Kamui2 ライブラリで廃止された関数	51
7.1 廃止関数の一覧	51
7.1.1 ダイレクトモード用関数について	51
7.2 (廃止関数の) 対応方法	52
8 Kamui2 ライブラリの関数一覧	54

1 Kamui2 ライブラリについて

この章では、最新のバージョンから公開された Kamui ドライバについて、1 から 2 への変更の経緯を含めて解説します。

1.1 Kamui ライブラリの概要

Dreamcast は、第二世代グラフィックチップ PowerVR2DC を搭載したシステムです。Kamui とは、この PowerVR2DC のドライバ及びハードウェアコントロールのためのライブラリです。

1.2 Kamui ライブラリの位置付け

Dreamcast には、グラフィックライブラリとして Ninja2 ライブラリがありますが、この Kamui ライブラリは Ninja ライブラリに対して下位に位置するライブラリです。

Ninja ライブラリに比べ、より細かな描画をコントロールすることが可能になります。

Kamui ライブラリには現在 2 つのバージョンがあり、Kamui2 ライブラリは SEGA Library 1 では非公開だった Kamui1 ライブラリの上位に位置するバージョンのライブラリです。

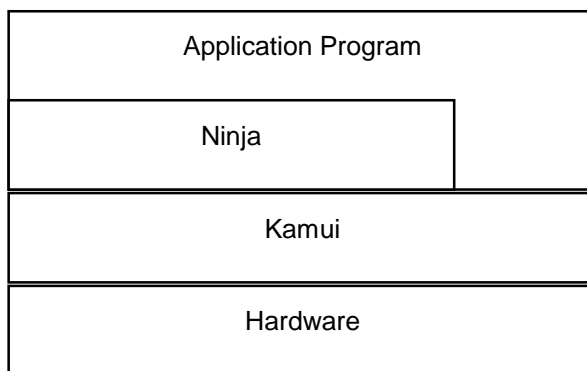


図 1 - 1 Kamui ライブラリのイメージ

1.3 Kamui ライブラリの機能

Kamui ライブラリは、以下の機能を持ちます。

- 三角形ストリップポリゴン、四角形ポリゴンをシーン¹へ登録します
- ポリゴン（ストリップ）ごとのレンダリング条件（コンテキスト）の設定します
- フレームバッファのハンドリングします
- テクスチャのハンドリングします
- 各種特殊効果（フォグ、モデファイアボリュームなど）の設定します
- その他、サービスルーチンなど

Kamui ライブラリ内部では、最小限のエラーチェックしか行いません。

注 意 Kamui ライブラリの各関数に、仕様以外の不正なパラメータが入力された場合の動作については保証致しません。

¹ ここでいう「シーン」とは、描画を行った 1 つの画面のことを指します。つまり秒間 60 フレームで描画している場合、その 60 分の 1 秒の間に行う描画を指します。

1.3.1 ユーザープログラムが行う処理

Ninja ライブラリにおける次のような処理は、ユーザープログラムから実行しなければなりません。

- モデル（オブジェクト）のハンドリング
- モーションの実行
- カメラモーション・ライトモーションの実行
- 3D空間ハンドリング
- 数学関数計算
- マトリクス計算
- スプライト描画

従って、Ninja ライブラリから利用する目的で作成された 3D グラフィックのデータは、Kamui ライブラリから簡易的に利用することはできません。利用するには、なんらかのサービスルーチンを作成する必要があります。

1.4 基本ポリゴン

Dreamcast に搭載している PowerVR2DC が、サポートしているポリゴンの種類は、以下の 3 通りです。

- 単独三角形ポリゴン（Single Triangle polygon）
- 単独四角形ポリゴン（Single Quad polygon）
- ストリップ三角形ポリゴン（Stripped Triangle polygon）

四角形ポリゴンの 4 頂点目の Z, U, V の各座標値及び Shading のカラー値は、ハードウェア内部で計算されたポリゴンの面の方程式から自動的に求められます。そのため、3D 座標系上に任意に置かれた単独四角形ポリゴンは正しく表示されない場合があります。

各ポリゴンの頂点の順番と繋がり方は、以下のようになります。

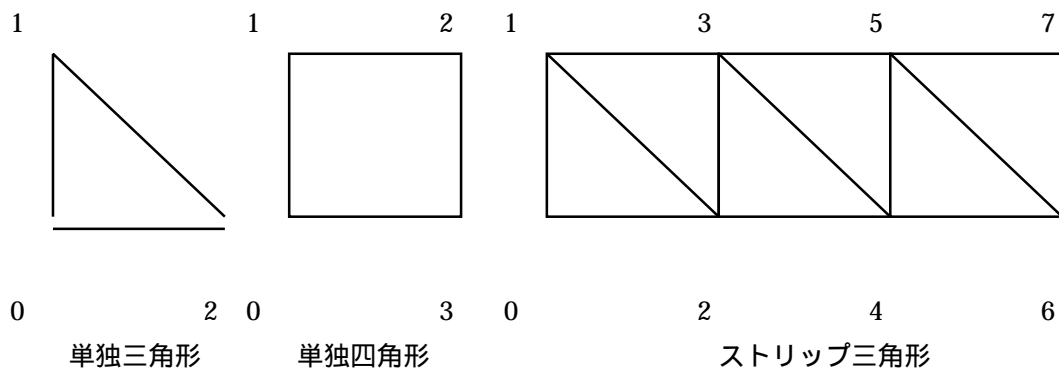


図 1 - 2 ポリゴンの種類

また、対応しているポリゴンタイプは次の6種類です。

形 式	テクスチャ	シェーディング	オフセットカラー
Non-Textured Flat Shaded	なし	フラット	なし
Non-Textured Gouraud Shaded	なし	ゲーロー	なし
Textured Flat Shaded	あり	フラット	なし
Textured Gouraud Shaded	あり	ゲーロー	なし
Textured Flat Shaded with Offset Color	あり	フラット	あり
Textured Gouraud Shaded with Offset Color	あり	ゲーロー	あり

表 1 - 1 ポリゴンタイプ

1.4.1 Shading カラーについて

Shading カラーには、「Base Color」と「Offset Color」と呼ばれる2つのデータがあります。

これらのShading カラーによる演算式は、ポリゴンパラメータ中のコントロール・ビットによって指定されます。基本的に「Base Color」は各頂点のShading 値を、Offset Color は各頂点のSpecular 値を指定します。

1.5 Kamui ライブラリの座標系

Kamui ライブラリに指定する座標系²は、スクリーン座標系で指定します。

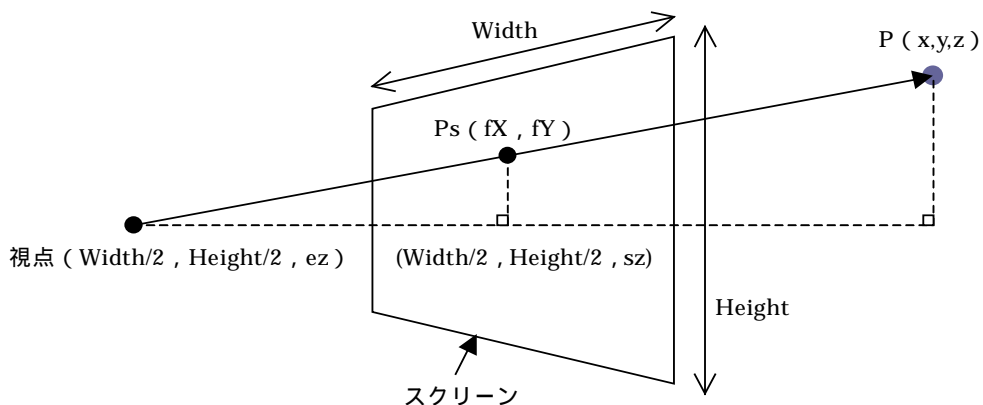


図 1 - 3 座標系

この図において、点 $P(x, y, z)$ の座標を指定する場合、ライブラリに指定する座標値 ($fX, fY, fInvW$) は、以下の計算で求めます。

$$\begin{aligned} fInvW &= (ez - sz) / (ez - z) \\ fX &= x \times fInvW \\ fY &= y \times fInvW \end{aligned}$$

ただし、テクスチャのUV 座標値に、 $fInvW$ を乗算する必要はありません。

² Ninja ライブラリでは、この座標系のことを「ハード座標系」と呼んでいます。

1.6 Kamui ライブラリのシーケンス

PowerVR2DC の描画部分には、ディスプレイリストの生成を補助するタイルアクセラレータと、描画を行う CORE と呼ばれる 2 つのブロックがあります。

描画用のポリゴンリストは、CPU からタイルアクセラレータに入力するパラメータと、CORE 用のディスプレイリストに変換され、テクスチャメモリ内の指定された領域に自動的に格納されます。

CORE ブロックは、テクスチャメモリ内の CORE 用ディスプレイリストとテクスチャデータを使ってポリゴンの描画を行い、テクスチャメモリ内のフレームバッファに描画データを格納します。

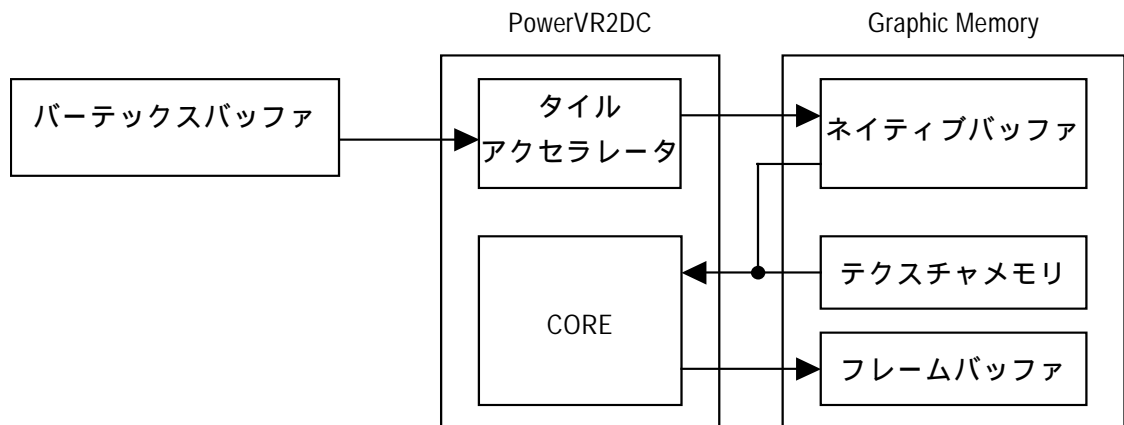


図 1 - 4 Kamui ライブラリのシーケンス

1.7 ポリゴンリスト

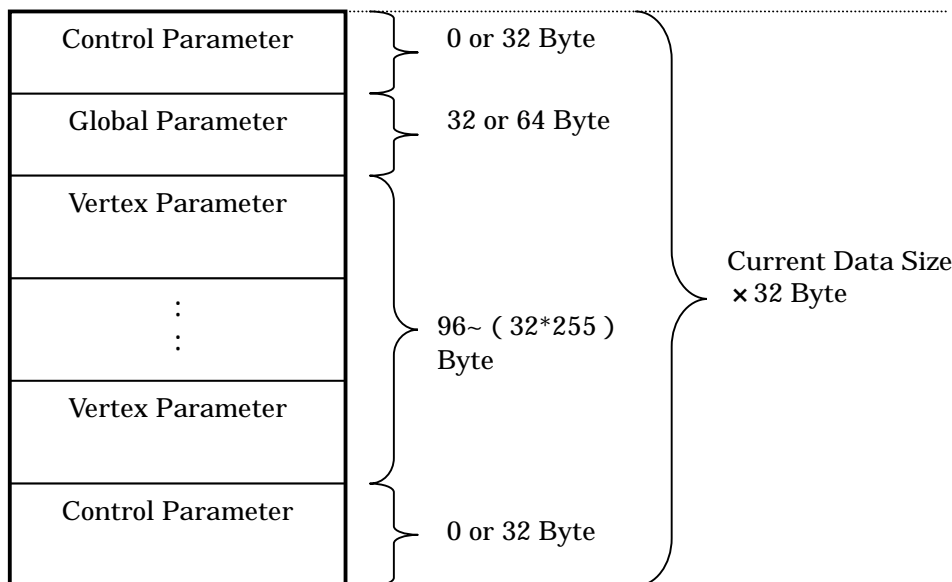
PVR 2 には、「トランスルーセント」「トランスルーセントモディファイア」「オペーク」「オペークモディファイア」「パンチスルー」の 5 種類のポリゴンリストがあります。

注 意 PowerVR2DC はポリゴンリストが混在した状態では、描画することはできません。必ず、各ポリゴンリスト単位で送る必要があります。

1.8 ポリゴンパラメータ

Kamui のバーテックスバッファにポリゴンパラメータを登録することで、フレームバッファに描画できるようになります。

ポリゴンパラメータは、「Control Parameter」「Global Parameter」「Vertex Parameter」の3種類から構成されています。



1.9 VertexParameter

バーテックスのフォーマットは、「カラーフォーマット」「UV フォーマット」の組み合わせ（15種類）、及びクアッド形式の2種類の設定が可能です。

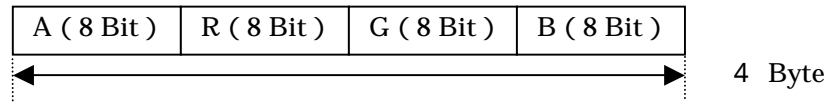
カラーフォーマットには「パック」「フロート」「インテンシティ」の3種類があり、UV フォーマットには「フロート」「16Bit フロート」の2種類があります。

	表示方式	カラーフォーマット	UV フォーマット
PolygonType0	ポリゴン	パック	
PolygonType1	ポリゴン	フロート	
PolygonType2	ポリゴン	インテンシティ	
PolygonType3	テクスチャ	パック	フロート
PolygonType4	テクスチャ	パック	16 Bit フロート
PolygonType5	テクスチャ	フロート	フロート
PolygonType6	テクスチャ	フロート	16 Bit フロート
PolygonType7	テクスチャ	インテンシティ	フロート
PolygonType8	テクスチャ	インテンシティ	16 Bit フロート
PolygonType9	ポリゴン（2P 用）	パック	
PolygonType10	ポリゴン（2P 用）	インテンシティ	
PolygonType11	テクスチャ（2P 用）	パック	フロート
PolygonType12	テクスチャ（2P 用）	パック	16 Bit フロート
PolygonType13	テクスチャ（2P 用）	インテンシティ	フロート
PolygonType14	テクスチャ（2P 用）	インテンシティ	16 Bit フロート
SpriteType0	クアッドポリゴン		
SpriteType1	クアッドテクスチャ		16 Bit フロート

1.10 カラーフォーマット

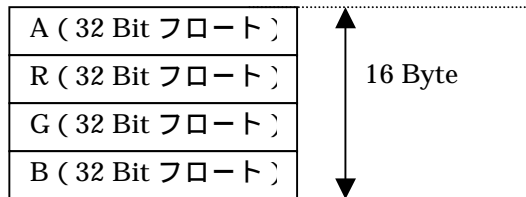
- パック カラー

ARGB を各 8 Bit で格納しています



- フLOAT カラー

ARGB を各 32 Bit フLOAT で格納しています



- インテンシティ カラー

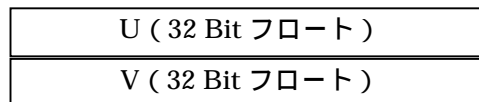
色の指定は「Global Parameter」を参照し、色の強さ情報だけです。

32 Bit フLOAT

1.11 UV フォーマット

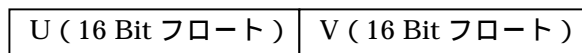
- フLOAT

各 UV 値を 32 Bit フLOAT で格納します



- 16 Bit フLOAT

各 UV 値を 16 Bit フLOAT で格納します



1.12 タイプ一覧

PolygonType0	PolygonType1	PolygonType2
X	X	X
Y	Y	Y
Z	Z	Z
	BaseColorAlpha	
	BaseColorR	
BaseColor	BaseColorG	BaseIntensity
	BaseColorB	

32 Byte

32 Byte

32 Byte

PolygonType3	
	32 Byte
X	
Y	
Z	
U	
V	
BaseColor	
OffsetColor	

PolygonType4	
	32 Byte
X	
Y	
Z	
U/V	
BaseColor	
OffsetColor	

PolygonType5	
	64 Byte
X	
Y	
Z	
U	
V	
BaseColorAlpha	
BaseColorR	
BaseColorG	
BaseColorB	
OffsetColorAlpha	
OffsetColorR	
OffsetColorG	
OffsetColorB	

PolygonType6	
	64 Byte
X	
Y	
Z	
U/V	
BaseColorAlpha	
BaseColorR	
BaseColorG	
BaseColorB	
OffsetColorAlpha	
OffsetColorR	
OffsetColorG	
OffsetColorB	

PolygonType7	
	32 Byte
X	
Y	
Z	
U	
V	
BaseIntensity	
OffsetIntensity	

PolygonType8	
	32 Byte
X	
Y	
Z	
U/V	
BaseIntensity	
OffsetIntensity	

PolygonType9	
	32 Byte
X	
Y	
Z	
BaseColor0	
BaseColor1	

PolygonType10	
	32 Byte
X	
Y	
Z	
BaseIntensity0	
BaseIntensity1	

PolygonType11	
	64 Byte
X	
Y	
Z	
U0	
V0	
BaseColor0	
OffsetColor0	
U1	
V1	
BaseColor1	
OffsetColor1	

PolygonType12	
	64 Byte
X	
Y	
Z	
U0/V0	
BaseColor0	
OffsetColor0	
U1/V1	
BaseColor1	
OffsetColor1	

PolygonType13	
	64 Byte
X	
Y	
Z	
U0	
V0	
BaseIntensity0	
OffsetIntensity0	
U1	
V1	
BaseIntensity1	
OffsetIntensity1	

PolygonType14	
	64 Byte
X	
Y	
Z	
U0/V0	
BaseIntensity0	
OffsetIntensity0	
U1/V1	
BaseIntensity1	
OffsetIntensity1	

SpriteType0	
	64 Byte
AX	
AY	
AZ	
BX	
BY	
BZ	
CX	
CY	
CZ	
DX	
DY	

SpriteType1	
	64 Byte
AX	
AY	
AZ	
BX	
BY	
BZ	
CX	
CY	
CZ	
DX	
DY	
AU/AV	
BU/BV	
CU/CV	

2 基本的な処理の流れ

Kamui ライブラリを使ったプログラムの、基本的な処理の流れの例を以下に示します。

(1) 初期化

リセット後、アプリケーションはまずフレームバッファ及び表示コントローラの設定を行う必要があります。

フレームバッファには通常、レンダリング完了時に二つのバッファを入れ替えるダブルバッファ方式を利用するため、表示用の Primary Surface と非表示のレンダリング用の Off Screen Surface の、2つの領域を確保します（最大4つのバッファを初期化時に指定が可能）。

また、フレームバッファの初期化設定以外にレイテンシモデル（3Vレイテンシモデルと2Vレイテンシモデル）、同期モード（V-Sync 信号に同期 / 非同期）、パス、頂点バッファなどの、Kamui が管理する動作環境情報の初期化を行います。

(2) シーンごとの設定

初期化が終了すると、シーン全体に関わるパラメータ（たとえばフォグテーブルのデータやバックグラウンドカラー）の設定を行います。

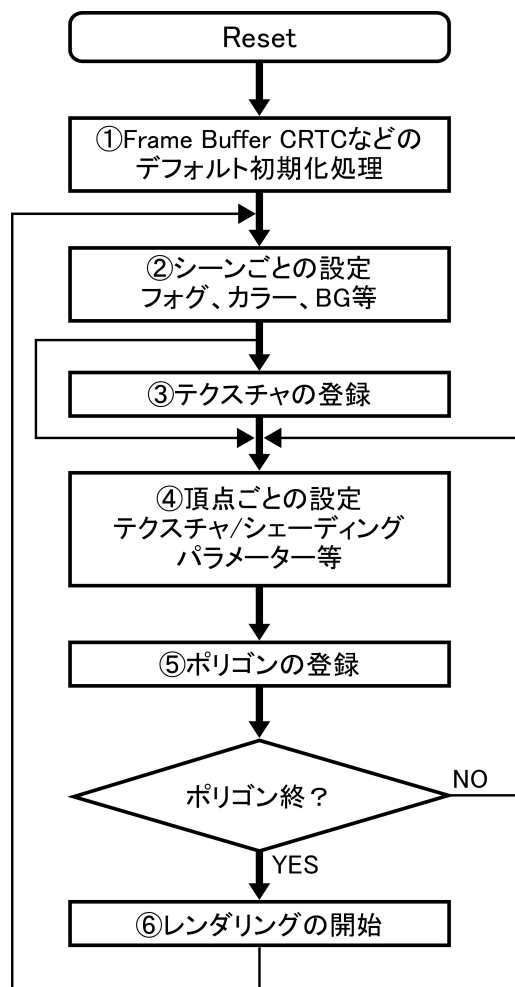


図 2 - 1 基本的な処理の流れ

(3) テクスチャの登録

次にテクスチャの登録を行います。テクスチャの登録は、単純にメインメモリ上に準備されたテクスチャデータをグラフィックメモリに転送することで行います。「図2-1 基本的な処理の流れ」の例では、シーン全体に関わるパラメータを登録した後にテクスチャを登録していますが、テクスチャの登録は、初期化処理の後であればフローの任意の位置で行うことができます。

KmBeginScene から KmEndScene 内でテクスチャ転送すると、DMA 転送することはできません。

注意 ただし、レンダリング途中のテクスチャデータを書き換えてはいけません。

(4) レンダリング条件の設定

次に登録するポリゴンごとに、データフォーマット、テクスチャ、テクスチャフィルタなどのレンダリング条件（コンテキスト）の設定を行います。毎回すべてのレンダリング条件を登録するのは非常に非効率な作業であるため、前回の設定から変化した分だけを設定します。

(5) ポリゴンの頂点データの登録

設定が終了した後に、ポリゴンの頂点データを登録します。頂点データはストリップ形式のほかに、Quad 形式もあります。

(6) レンダリングの開始

ポリゴンの登録が終わると、レンダリング開始コマンドを発行します。レンダリングの終了時に何らかの処理を追加したい場合は、レンダリング終了時に実行されるコールバック関数を用いて行うことができます。

レンダリング開始コマンドを発行する際に、フレームバッファの切り替え（Flip）を指定することができます。Flip を指定することにより、レンダリングが終了した次の V-Sync 信号のタイミングで自動的に、Primary Surface と Off Screen Surface が入れ替わります。これによりレンダリング途中で画面が切り替わってしまう状態、いわゆる「胴切れ」を回避することができます。

頂点データの登録から表示まで一連の処理は、状態管理情報で管理することができます。状態管理情報領域は、常に複数用意されています。レンダリング開始コマンドが発行されると、その時点の状態管理情報がキューに積みこまれるだけで、V-Sync 信号を検出するために待機する必要はありません。

Kamui を利用するプログラムは、レンダリングコマンドを発行した後で頂点データが登録できる状態であれば、別の状態管理情報領域を使用し、すぐに次のシーンの頂点データ登録を開始することができます。

ノート ここで言う頂点データ登録ができる状態とは、初期化の動作環境設定が V-Sync 信号に非同期で、現在レンダリングに使用されている領域への登録でないということです。

a. ダブルバッファリングについて

Kamui 内部では、レンダリングの頂点データもフレームバッファと同様にダブルバッファリングされます（初期化時に最大 8 バッファの指定が可能）。

ダブルバッファリングが実行されるのは、現在レンダリング中の頂点データと登録中のパラメータが、別々に必要になるためです。

ただし、プログラマ自身はそれを意識する必要はほとんどありません。

以下に、頂点データ登録及びレンダリング開始コマンド発行の処理の流れを示します。

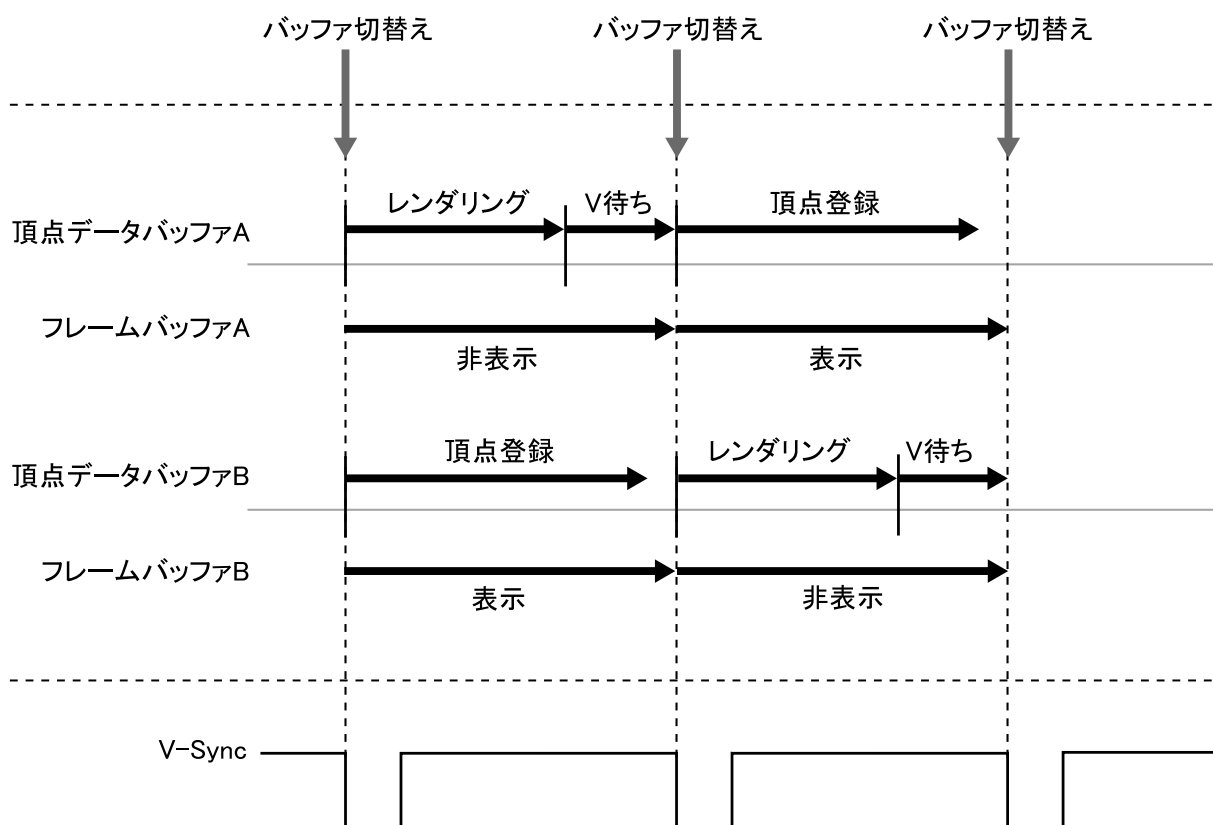


図 2 - 2 頂点登録とレンダリングのタイミング (同期モード)

2.2 頂点データバッファと各種内部バッファ

Kamui では、アプリケーションによってメインメモリ内に頂点データ登録用の頂点データバッファを確保し、そこに任意の数の頂点データを登録し、レンダリングコマンドを発行することによって描画します。

頂点データバッファはポリゴンの種類により、下記に示す5つに分割して使用します。

- 不透明ポリゴン用バッファ (Opaque Polygon)
- 不透明モデファイアポリューム用バッファ (Opaque Modifier)
- 半透明 / 透明ポリゴン用バッファ (Translucent Polygon)
- 半透明 / 透明モデファイアポリューム用バッファ (Translucent Modifier)
- パンチスルーポリゴン用バッファ (Punchthrough Polygon)

すべての頂点データは、上記5種のいずれかに登録されます。

また、タイルアクセラレータによって生成された PowerVR2DC のネイティブ形式のポリゴンデータを一時保管する、ネイティブデータバッファの確保も必要になります。ネイティブデータバッファは、フレームバッファメモリ内に確保します。

これらのバッファとポリゴンデータの流れを図に示すと、下記のようにになります。

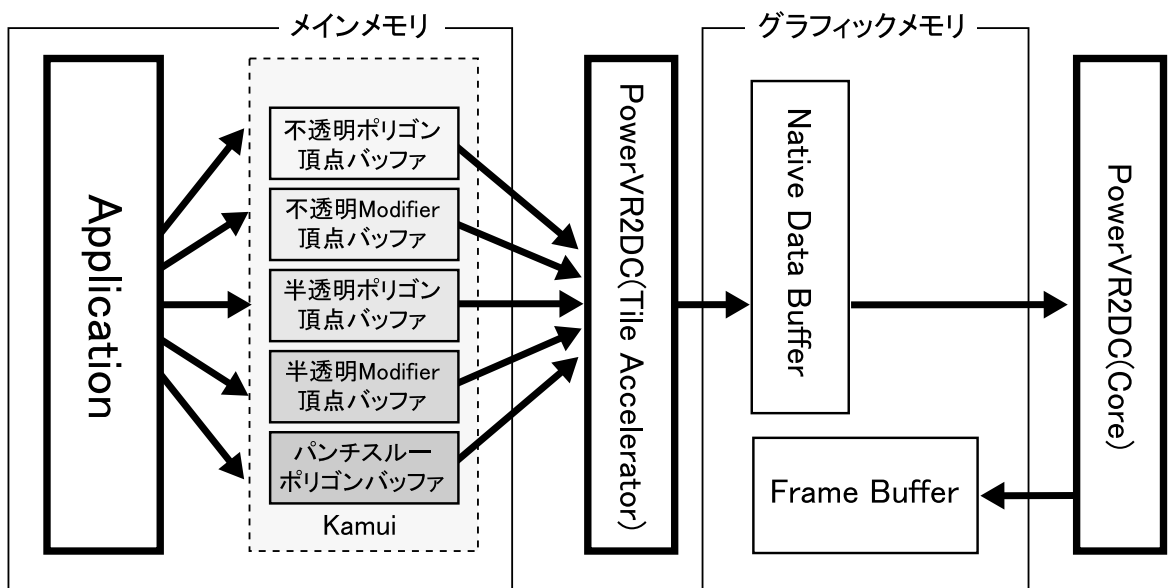


図 2-3 バッファードモードのみの描画

2.2.1 ダイレクトモードとバッファモードについて

メインメモリに5つの頂点データ登録用バッファを確保することなしに、直接ハードウェア（タイラアクセラレータ）に頂点データを送り込む方法をダイレクトモード（Direct Mode）と呼びます。

これに対して、メインメモリに頂点データバッファを置く前述の方法を、バッファードモード（Buffered Mode）と呼びます。

Kamui では、2 V レイテンシモデルの設定時に、5つの頂点データのうち1つだけをダイレクトモードに指定することができます。ダイレクトモードに指定された頂点データのメインメモリ上の登録用バッファは不要になります。特定の頂点データが非常に多い場合、その頂点データをダイレクトモードに指定することにより、必要となるメモリ容量は少なくて済みます。

この場合のバッファとポリゴンデータの流れを図に示すと、下記のようにになります。

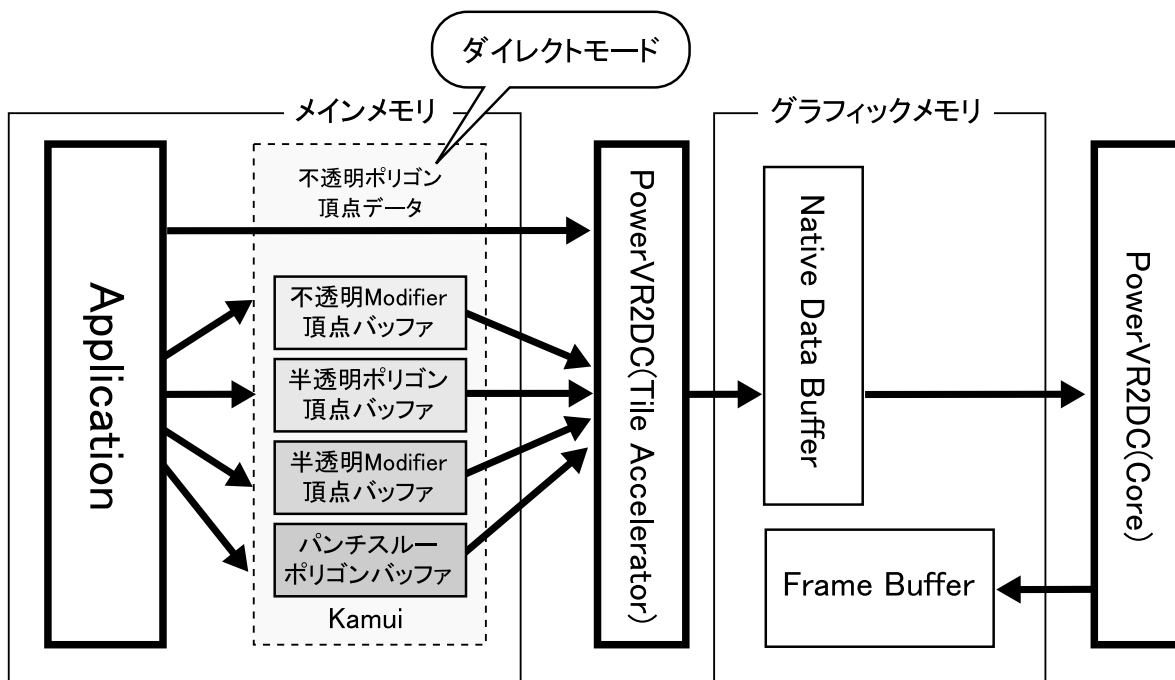


図 2 - 4 不透明ポリゴン頂点をダイレクトモードに設定した描画

2.3 レイテンシモデル

Kamui では、状況に応じた描画のレイテンシ³モデルを設定することで、レンダリング条件に適した描画方法を選択できます。

Kamui で選択できるレイテンシモデルには、3 V レイテンシモデルと2 V レイテンシモデルがあり、それぞれについて、特徴や利点が変わります。3 V ・ 2 V レイテンシモデルのそれぞれの利点を次にあげます。

- 3 V レイテンシモデル

頂点登録（描画関数）を実行したフレームから、3 V 後にディスプレイ表示されるモデルです。頂点登録、頂点転送、レンダリングが1 V 毎に行われるため2 V レイテンシモードより、多くのポリゴンを処理することができます。

頂点登録と頂点転送が同じフレーム内で実行されないため、頂点バッファはダブルバッファになります。

- 2 V レイテンシモデル

頂点登録（描画関数）を実行したフレームから、2 V 後にディスプレイに表示されるモデルです。頂点登録と同じフレームで頂点転送を行い、次のフレームでレンダリングが実行されます。

頂点登録と頂点転送が同じフレームで行われるため、頂点バッファはシングルバッファでよいので、メモリが節約できます。

以下、それぞれのモデルについての説明を行います。

³ 本来はデータのリクエストを行ってから、実際にデータが転送されるまでにかかる遅延時間のことを指します。ここでは描画を行うリクエストを行ってから、実際に表示されるまでの時間を指します。

2.3.1 3 V レイテンシモデル

Dreamcast システムにおいて最大パフォーマンスを得ることができ、なおかつ定率のレイテンシを確保するためのプログラミングモデルです。3 V の各期間にそれぞれ頂点データ登録、DMA 転送(タイルアクセラレータへ転送)、レンダリングを配置するやり方です。処理の流れを以下に示します。

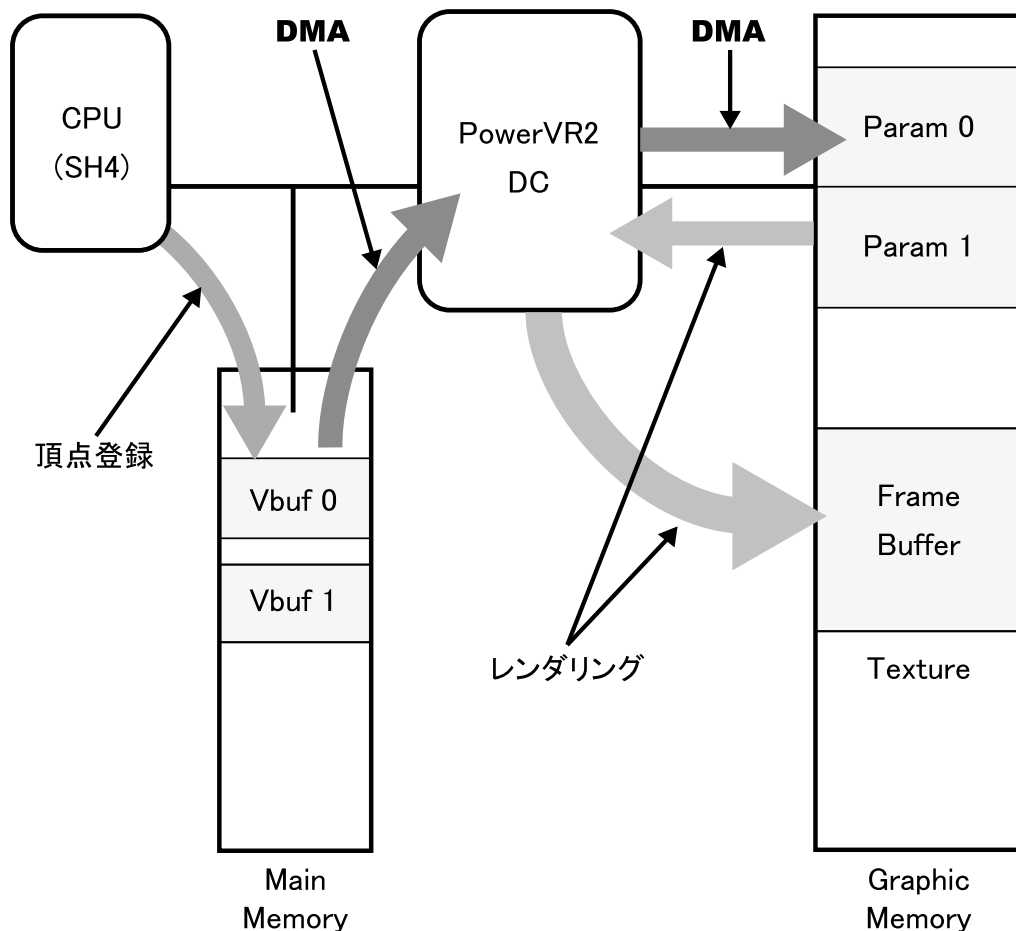


図 2 - 5 3 V レイテンシモデルの処理の流れ

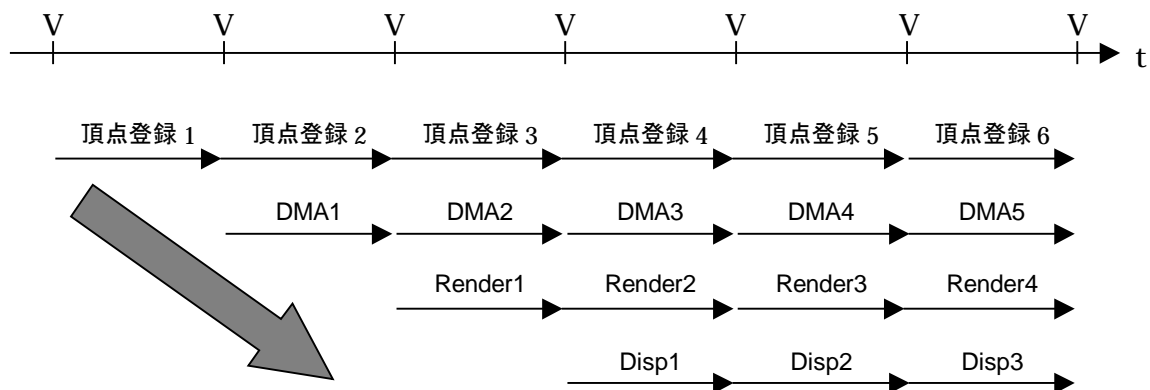


図 2 - 6 3 V レイテンシモデルのプログラミングパイプライン

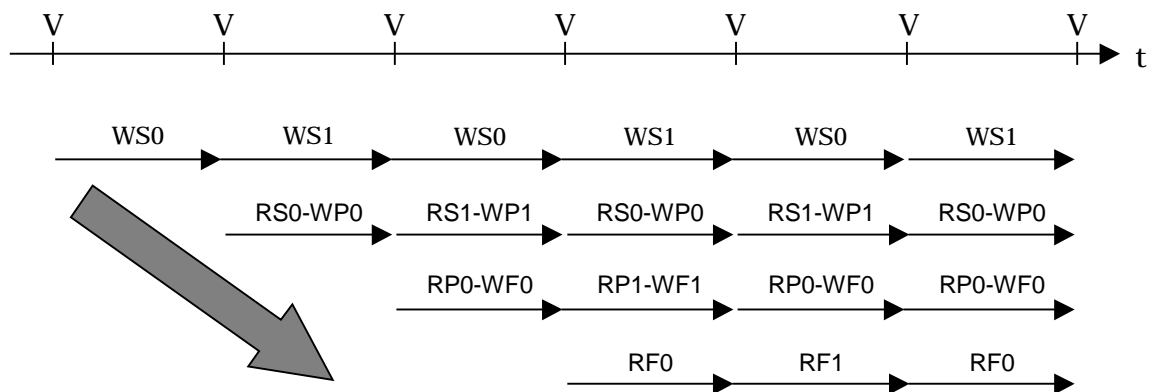
頂点登録：	シーンを構成する頂点データの登録を表します。ゲームシーケンス、ジオメトリ、タイルアクセラレータ転送用データ作成を含みます。
DMA：	DMA を使用して作成したタイルアクセラレータ転送用データをタイルアクセラレータへ送り、出力された描画パラメータをグラフィックメモリへ書き込みます。
Render：	タイルアクセラレータによって展開された描画パラメータを使用して、レンダリングを行います。
Disp：	レンダリング結果を表示します。

この場合（フレームバッファダブルバッファリング）、以下のメモリエリアが必要になります。

メモリの種類	内 容
メインメモリ	頂点データバッファ（1シーン全体分）× 2
グラフィックメモリ	PowerVR2DC 内部パラメータ形式のネイティブデータバッファ（1シーン全体分）× 2 表示用フレームバッファ × 2 テクスチャメモリ

表 2 - 1 必要とするメモリ

上記パイプラインでのメモリの使用状況を表すると、次のようになります。



WS0 / WS1	メインメモリへのライト（CPU による Write）
RS0 / RS1	メインメモリからリード（DMA による Read）
WP0 / WP1	ネイティブデータバッファへのライト（DMA による Write）
RP0 / RP1	ネイティブデータバッファからリード（レンダラによる Read）
WF0 / WF1	フレームバッファへのライト（レンダラによる Write）
RF0 / RF1	フレームバッファからリード（レンダラによる表示）

数値は、ダブルバッファの番号を示します。

このモデルを利用すると、ハードウェアリソースの競合はないため 60fps の場合の調停を最小限に押さえることが可能になります。

2.3.2 2 V レイテンシモデル

キー入力の反応時間を向上させる場合や、頂点バッファの容量を低減させる場合、2 V レイテンシモデルを使用します。

2 V レイテンシモデルの場合のパイプラインは、下図に示すように、3 V レイテンシモデルに比べて一つの V 期間内に頂点データ登録と DMA が圧縮されている構造になります。

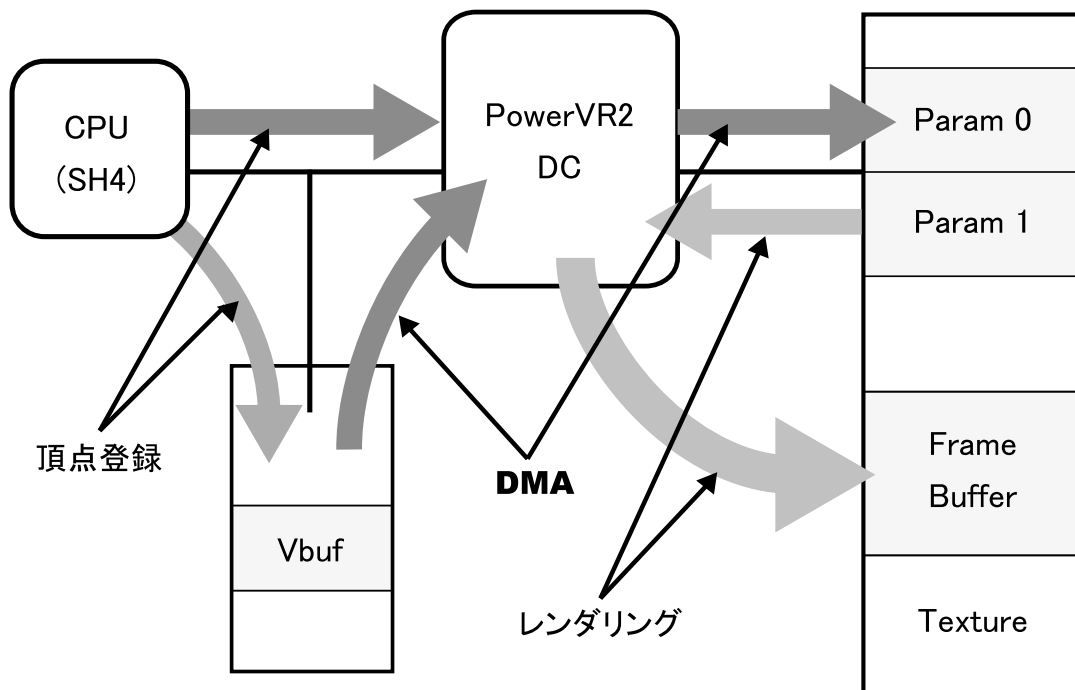


図 2 - 7 2 V レイテンシモデル処理の流れ

2 V レイテンシのプログラミングモデルの場合、頂点データの登録は特定の頂点タイプのリストを、直接タイルアクセラレータ上に送り込みます。

例えば、シーン全体で Opaque のポリゴンデータが最も多いとすると、Opaque のデータを直接タイルアクセラレータに送り込み、それ以外 (Opaque Modifier , Translucent , Translucent Modifier , Punchthrough) のリストに対して、登録されたデータをいったんメモリにストアしておきます。

KmRender() が実行されたとき、残りのリストタイプのデータを DMA 転送します。この場合は、DMA 転送を開始したフレーム内で DMA 転送が終了しないと、処理落ちすることになります。

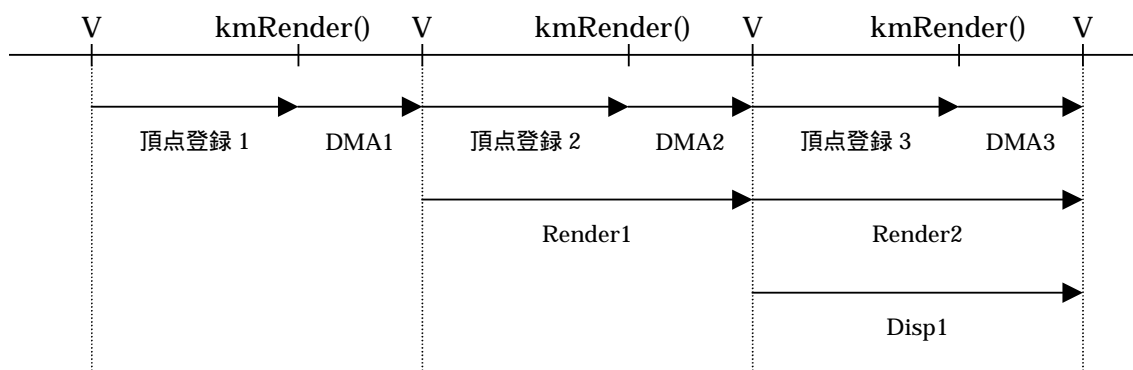


図 2 - 8 2 V レイテンシモデルのプログラミングパイプライン

直接転送に設定されている頂点データは、頂点データ登録関数で直接タイルアクセラレータ上へ転送されます。

それ以外の頂点データは、kmRender 発行時に DMA 転送を開始し、V 期間内に終了していなければなりません。

2.3.3 処理オーバーした場合の対応

各パイプライン処理が 1 Int 以内に収まらずに処理オーバーした場合の対応は、処理落ちの状態を進めることを基本としています。

レンダリングが 1 Int 以内に終了しない場合には、レンダラに強制リセットをかけて次のシーンのレンダリングを開始することも可能です。

注意 ただし、レンダリングを途中で強制的に停止させた場合は、シーンの途中でレンダリングが打ち切られることとなり、場合によっては不正な画面になってしまいますので注意して下さい。

処理落ちが発生し、待ち合わせ制御する場合の例を下記に示します。

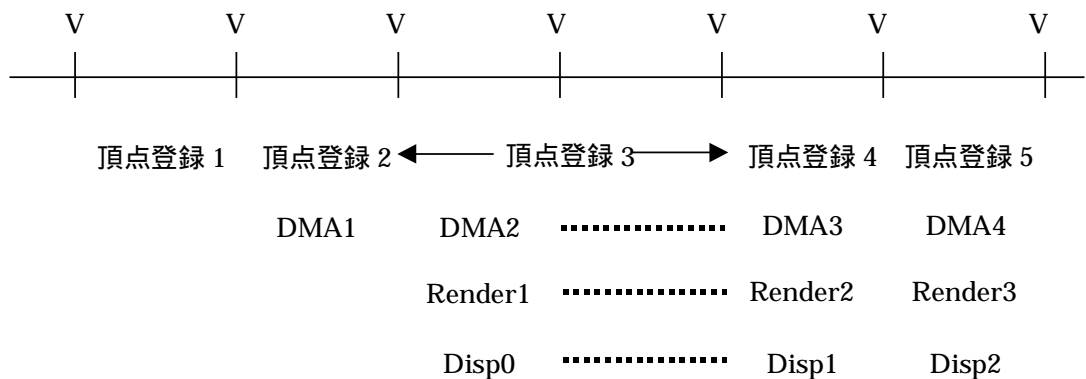


図 2 - 9 頂点登録で処理落ち

この場合、頂点登録3の処理が長引いたため、DMA3、Render2、Disp1の処理が待ち合わせされている状態です。

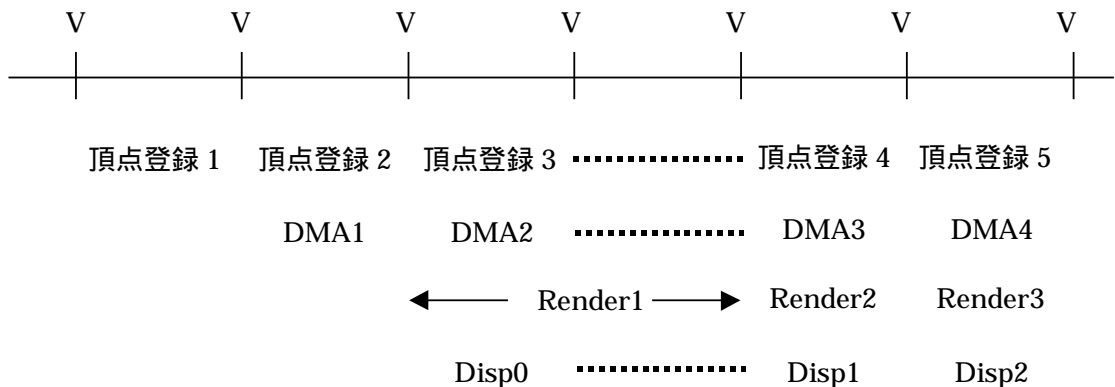


図 2 - 10 レンダリングで処理落ち

上図は、同様に Render1 に時間が掛かってしまったため、待ち合わせが生じる場合を示しています。

2.4 V-Sync 同期 / 非同期モード

レンダリング開始コマンド発行後の動作状態遷移を V-Sync 信号に同期させるか、非同期に状態遷移させるかの設定ができます。

2.4.1 (3 V レイテンシモデルの) 同期モードの場合

3 V レイテンシモデルの同期モードの場合、Kamui はレンダリング開始コマンドが発行されると頂点登録状態から DMA 転送開始待ち状態になります。(頂点バッファに登録された頂点情報は DMA を使用してタイルアクセラレータへ転送されます。) 次の V-Sync 信号のタイミングで DMA 転送が開始され、タイルアクセラレータにより、ネイティブバッファへレンダリングパラメータが作成されます。DMA 転送が終了するとレンダリング開始待ち状態になり、さらに次の信号のタイミングでレンダリングを開始します。レンダリングが終了すると表示待ち状態になり、また次の V-Sync 信号のタイミングでレンダリング結果を表示します。

このように同期モードの場合、すべての状態遷移が V-Sync 信号のタイミングで正規化されます。

注 意 レンダリング開始コマンドでは、状態遷移管理情報がキュー - に積み込まれるだけで、一連の処理が終了するまで呼び出し元へ戻らないという意味ではありません。

2.4.2 (3 V レイテンシモデルの) 非同期モードの場合

非同期モードの場合、レンダリング開始コマンド発行後に DMA 転送開始待ちにはならず、すぐに DMA 転送が開始され DMA 転送終了後すぐにレンダリングが開始されます。ただし、表示切り替えだけは必ず V-Sync 信号のタイミングで行う必要があるため、レンダリング終了では表示待ち状態になり、レンダリング終了後の V-Sync 信号のタイミングで表示が行われます。

頂点情報が多く、DMA 転送またはレンダリング時間が V-Sync 信号を超えてしまうような場合、同期モードでは表示の待ち合せが発生する可能性があります。

このような場合、非同期モードにすることによって V-Sync 信号の待ち時間が省かれ、本来の表示タイミングまでにレンダリング処理が終了できる可能性があります。

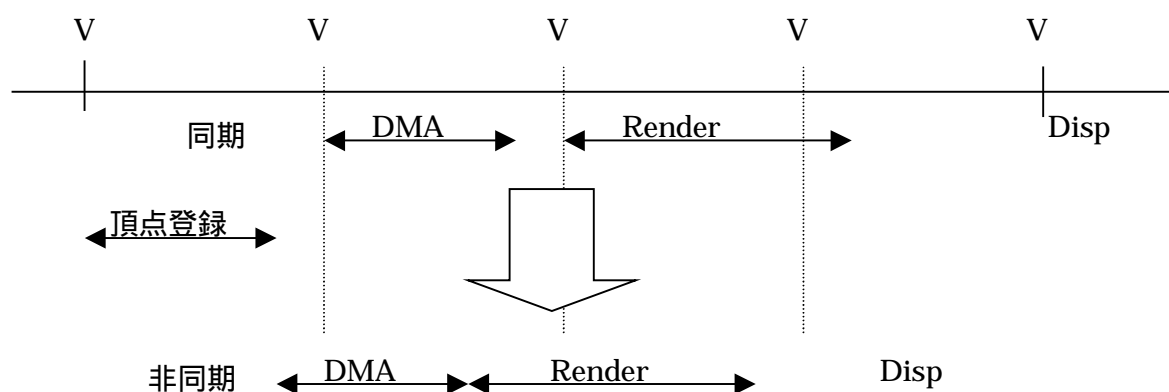


図 2 - 1 1 同期モードと非同期モード

3 マルチパス

Kamui の描画で言うパスとは、一度に送られるディスプレイリストのことを示します。パスを数回に分けて送るマルチパスを使用すると、パスごとの Z 値管理が可能になります。

マルチパスには以下のような、代表的な使い方があります。

- Auto / Pre-Sort の切り替え
- 半透明グルーピング
- Zクリッピングプレーン

3.1 Auto-Sort / Pre-Sort の切り替え

Z 値に関係なく、半透明ポリゴンを使用して情報を表示したい場合などに周囲のモデルに重ならないように Z 値を調整することは、非常に難しい場合があります。これは、すべての半透明が AutoSort で描画されているためであり、AutoSort を指定した場合、半透明の ZCompareMode は GE (Greater Equal) に固定されてしまうためです。

この場合、情報表示用のポリゴンを別パスで描画し、別パスを Pre-Sort モードにすることにより、半透明ポリゴンを Always などの ZCompareMode で描画することができます。

3.2 半透明グルーピング

半透明ポリゴンを AutoSort で大量に描画した場合、Fill Rate ではなく Sort 時間がレンダリングのボトルネック現象となる場合があります。通常、小さ目のポリゴンが大量に視点から Z 方向に対し重なった場合に、起こりやすい現象です。

このような場合、半透明をグルーピングすることで、このボトルネックを解消できる場合があります。

半透明グルーピングとは、半透明オブジェクトをいくつかのグループに分け、これらのパスを分けて描画するやり方です。視点から遠くのグループから描画することで、全ての半透明をソートし描画した場合とほぼ同じ描画結果を、得ることができます。n パス目と n+1 パス目の合成は、単なる重ねあわせのアルファブレンディングで行われるため、パス間のオブジェクトはクロスセクション（交差する領域）を持つ場合、描画結果が不正となる場合があります。

3.3 Zクリッピングプレーン

コックピットビューなど、遠景（背景やその他オブジェクト）と近景（コックピット）の3Dモデルを完全に分離したい場合があります。この場合にZのクリッピング値を設定することが可能です。

この場合、以下のような設定を行って下さい。

1 パス目	遠景のモデルを描画します。
2 パス目	完全透明（Always）のZクリッピングポリゴンのZ値を、クリップしたい値に合わせて描画します。
3 パス目	近景のモデルを描画します。

こうすることで、2パス目でいったんZ値がリセットされ、その後の近景のZ値は、このクリッピングプレーンのZ値をもとに描画されることになります。

注 意 この場合、Fill Rate が犠牲になります。

4 Kamui ライブラリの基本的な使用法

ここでは、Kamui ライブラリの具体的な使用法について説明します。

4.1 初期化

Kamui の初期化関数は、下記の順序で呼び出して下さい。

- (1) `kmInitDevice ()`
Kamui ライブラリの初期化を行います。Kamui 内部で使用する領域の初期化を行います。
- (2) `kmSetDisplayMode ()`
引数で指定されるディスプレイモードを設定し、フレームバッファの表示モードが設定されます。
- (3) `kmSetSystemConfiguration ()`
動作環境の設定を行います。
引数で指定されるレイテンシモデル、V-Sync 同期モード、バスの設定、頂点バッファ数、表示用フレームバッファ数に従い、Kamui 内部で管理するバッファの初期設定を行います。

4.2 テクスチャの登録

以下の手順でテクスチャの登録を行います。

- (1) `kmCreateTextureSurface ()`
テクスチャを管理するためのサーフェスを作成します。
管理領域であるサーフェスは、アプリケーションで用意されている領域で、初期化時の動作環境設定関数 `kmSetSystemConfiguration ()` の呼び出し時に Kamui へ通知されている領域です。作成されるサーフェスは、この領域から獲得されます。
サーフェスを作成するとともに、テクスチャメモリをフレームバッファ上に獲得します。
- (2) `kmLoadTexture ()`
`kmCreateTextureSurface ()`で確保されたテクスチャメモリへテクスチャデータを読み込みます。
`kmCreateTextureSurface ()`を呼び出す際には、読み込むテクスチャフォーマットを指定して下さい。

4.3 レンダリングフロー

レンダリングは、通常以下の手順で行います（１シーン分）。

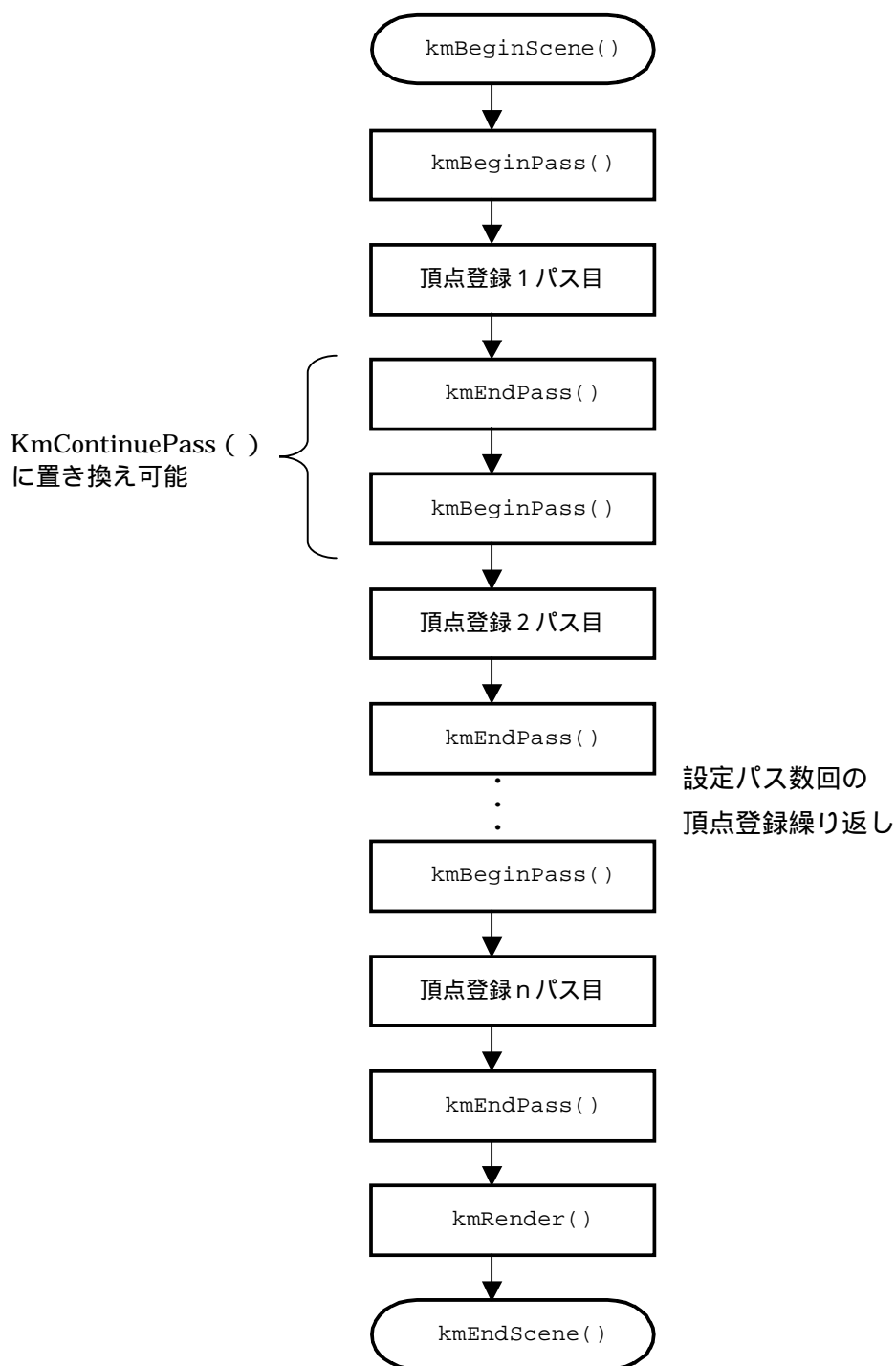


図 4 - 1 レンダリングフロー

kmBeginPass () と kmEndPass () は、パス数分だけ繰り返します。

その場合、初期化関数 `kmSetSystemConfiguration ()` で設定したパス数と同じ回数のパスを実行する必要があります。設定されたパス数と呼び出し回数が異なるとプログラムは正常に動作しません。

頂点データを登録しないパスについても、`kmBeginPass ()` と `kmEndPass ()` だけは呼び出して下さい。この場合、頂点登録関数の呼び出しは不要です。

(1) `kmBeginScene ()`

1 シーンのレンダリング準備を行います。

Kamui 内部で管理しているパス情報を先頭パスで初期化し、頂点登録が可能な状態にします。

(2) `kmBeginPass ()`

パスごとの頂点登録の準備を行います。

パス内で使用されるリスト別の頂点バッファ (Opaque Polygon , Opaque Modifier , Translucent Polygon , Translucent Modifier , Punchthrough Polygon) を準備します。

(3) `kmEndPass ()`

パスごとの頂点登録を終了します。

パス内で登録された頂点情報の登録の状態を検査します。アプリケーションで Fatal Error コールバック関数が登録されていると、頂点バッファがオーバーフローしたときにその登録された関数を呼び出します。

最終パスでなければ、Kamui 内部で管理しているパス情報を次のパスに進めます。

(4) `kmRender ()`

レンダリングを開始します。

引数に `KM_RENDER_FLIP` を指定することによりタイルアクセラレータへの DMA 転送、レンダリング、表示までの一連の処理を行います。

この関数は、状態遷移の管理情報をキューに積み込むだけですぐに復帰します。一連の処理が終了しなくても呼び出し元へ戻ります。

(5) `kmEndScene ()`

シーンの終了処理を行い、次のシーンの開始待ち状態にします。

`PKMSYSTEMCONFIGSTRUCT` 構造体による動作設定で V-Sync 信号に同期するように設定されている場合は V-Sync 信号の待ち合わせを行います。

このとき、待ち合わせを行うため呼び出し元へすぐには復帰しませんが、V-Sync 信号の待ち合わせのコールバック関数を登録しておくことで、V-Sync 信号に非同期なプログラムの作成が可能です。

ただし、あまり大きなコールバック関数を指定しないで下さい。また、コールバック関数内での無限ループなどは行わないで下さい。

V-Sync 信号に同期するように設定されていれば、シーンを V-Sync 信号の割り込みごとに管理できます。

V-Sync 信号に同期しないように設定されている場合は、V-Sync 信号の待ち合わせなしですぐに呼び出し元へ戻りますので続けて次のシーンの処理が行えます。

4.4 頂点の登録

最初に、ストリップ形式頂点データ登録の開始宣言コマンドを発行します。次に頂点登録コマンドを頂点数分だけ発行、最後に頂点データ終了宣言コマンドを発行して頂点登録を終了します。

頂点のレンダリング条件は、頂点データ登録の開始宣言コマンドで指定します。頂点登録の前にレンダリング条件情報を用意しておき、引数として頂点データ登録の開始宣言コマンドを発行します。レンダリング条件情報はタイルアクセラレータへ渡すパラメータ情報であり、このコマンドで Kamui 内部に確保されます。

タイルアクセラレータへ渡すパラメータ情報は、ストリップ形式データパラメータ構築関数 `kmGenerateStripHead()` を使用して作成します。レンダリング条件設定情報と作成先のパラメータ情報領域をアプリケーションで用意します。レンダリング条件設定情報が格納されている領域のポインタとパラメータ作成先領域のポインタを引数にしてストリップ形式データパラメータ構築関数を呼び出すことにより、タイルアクセラレータへ渡すパラメータ情報が作成されます。パラメータ構築関数は、設定情報をタイルアクセラレータ用の制御情報に変換する関数といえます。

<code>KMSTRIPHEAD StripHead</code>	作成先領域
<code>KMSTRIPCONTEXT StripContext</code>	設定情報領域
<code>StripContext</code>	領域へ設定情報を格納
<code>kmGenerateStripHead (</code>	
<code>&StripHead,</code>	作成先ポインタ
<code>&StripContext,</code>	設定情報ポインタ
<code>nVertexType,</code>	頂点タイプ
<code>);</code>	

このパラメータは頂点登録のたびに構築する必要はありません。初期化処理中に構築しておき、頂点データ登録の開始宣言コマンド発行時に引数として使用します。

レンダリング条件を一部変更するには、ストリップ形式頂点データのパラメータ変更関数 `kmChangeStripHead ~ ()` で行います。この変更の後に頂点データ登録の開始宣言コマンドを発行します。

また、ストリップ形式頂点データのパラメータを複数用意して管理する方法もあります。アプリケーションのメモリに余裕がある場合は、使用するすべてのレンダリング条件用のストリップ形式や頂点データのパラメータを構築しておき、頂点登録時に必要なレンダリングパラメータを頂点データ登録の開始宣言コマンドの引数として使用すれば、レンダリング条件を変更して頂点登録を行うことと同じ結果になります。

変更関数の呼び出しが必要ない分だけ処理に余裕ができます。変更関数は項目別に用意されており、変更項目が多い場合には有効です。

頂点の登録は、以下の手順で行います。

(1) `kmStartStrip()`

ストリップ形式の頂点データ登録の開始を Kamui へ知らせます。

このとき、同時にレンダリング条件の設定を行います。引数のストリップ形式頂点データのパラメータに従い、レンダリング条件を決定し、頂点登録状態にします。

(2) kmSetVertex ()

頂点登録を行います。3 V レイテンシモードであれば、リスト別の頂点バッファへ頂点データを格納します。2 V レイテンシモードの場合、直接転送リスト (ダイレクトモード) の頂点データであればタイルアクセラレータへ送り込みます。直接転送リストでなければ (バッファードモードの場合は) 3 V レイテンシモードと同じく、リスト別の頂点バッファへ頂点データを格納します。

kmStartStrip () で知らせた頂点タイプの頂点データを登録します。ストリップ形式の頂点データの登録が終了し、次に続く頂点データが同一のレンダリング条件であれば続けて登録できます。

(3) kmEndStrip ()

ストリップ形式頂点データ登録の終了を、Kamui へ知らせます。

5 PAL 拡張ディスプレイモード

ディスプレイモードには特殊なモードとして、PAL 拡張ディスプレイモードがあります。ディスプレイモードは、`kmSetDisplayMode()` 関数で設定します。

5.1 PAL 拡張ディスプレイモードについて

このモードは、PAL モニタ画面の縦方向が 480 ライン以上あるため、画面表示ライン数を追加して画面上下の非表示部を減らし、なるべく画面いっぱいに表示させるために用意されたモードです。

PAL 拡張ディスプレイモードとは、画面を 640×240 または 640×480 の PAL に設定するとき、縦方向に拡大して表示するモードのことです。元となる画像サイズ 640×240 または 640×480 に対し縦方向に指定された比率で拡大表示されます。

アプリケーションが管理する画面サイズは、640×480 または 640×240 になりますが、表示が縦方向に拡大されます。従って、アプリケーション側での頂点登録などで Kamui へ通知する座標情報は、640×480 または 640×240 内に指定することになります。

5.2 PAL 拡張設定情報

設定値	画面サイズ
KM_DSPMODE_PALNI640x240EXT	640 × (240 × 縦方向比率)
KM_DSPMODE_PALNI640x480FFEXT	640 × (480 × 縦方向比率)
KM_DSPMODE_PALNI640x480EXT	640 × (480 × 縦方向比率)
KM_DSPMODE_PALI640x480EXT	640 × (480 × 縦方向比率)

`kmSetDisplayMode()` 関数の呼び出し時に、上記いずれかのディスプレイモードが指定されると、Kamui は PAL 拡張モードと判断し、縦方向拡大比率を獲得するためのコールバック関数を呼び出します。

この縦方向拡大比率情報を Kamui へ知らせるコールバック関数は、アプリケーション制作者が用意しなければなりません。また、Kamui からコールバック関数を呼び出すために、`kmSetDisplayMode()` 関数を呼び出す前に PAL 拡張モード縦方向拡大比率設定コールバック関数を登録しておく必要があります。

コールバック関数には、以下の比率情報を設定します。

設定値	縦方向サイズ	
	元画像 480 ライン	元画像 240 ライン
KM_PALEXT_HEIGHT_RATIO_1_033	496 ライン表示	248 ライン表示
KM_PALEXT_HEIGHT_RATIO_1_066	512 ライン表示	256 ライン表示
KM_PALEXT_HEIGHT_RATIO_1_100	528 ライン表示	264 ライン表示
KM_PALEXT_HEIGHT_RATIO_1_133	544 ライン表示	272 ライン表示
KM_PALEXT_HEIGHT_RATIO_1_166	560 ライン表示	280 ライン表示

5.3 PAL 拡張モードコ - ルバック関数

縦の表示比率を決定するための定数を返すためのコールバック関数です。

```
VOID pPalExtCallback ( PVOID pData );
```

引数 pData には、Kamui 内部の KMPALEXTINFO 構造体へのポインタが渡されます。この構造体の nPALExtMode メンバに、縦方向拡大比率情報設定値を代入します。

5.4 PAL 拡張モードコ - ルバック関数ポインタ登録関数

縦の表示比率を決定する定数を返す、コールバック関数ポインタの登録を行います。

```
KMSTATUS KMAPI kmSetPALEXTCallback (
IN PKMCALLBACKFUNC pPalExtCallback,
IN PVOID pCallbackArguments
);
```

引数 pPalExtCallback に、アプリケーション制作者が用意した PAL 拡張モ - ドコ - ルバック関数ポインタを渡します。

引数 pCallbackArguments には、NULL を指定します。

5.5 PAL 拡張モード用構造体

KMPALEXTINFO は、PAL 拡張コ - ルバック関数で縦方向拡大比率情報を設定する際に使用する構造体です。

構造体メンバ	内 容
KMDISPLAYMODE nDisplayMode	kmSetDisplayMode () で設定されたディスプレイモードのこと
KMINT32 nPALExtMode	縦方向拡大比率を示す値のこと

5.6 画面設定手順

PAL 拡張モ - ド縦方向拡大比率設定コ - ルバック関数を用意します。

- デバイス初期化 kmInitDevice () を設定します。
- PAL 拡張モ - ド縦方向拡大比率設定コ - ルバック関数ポインタを登録します。
kmSetPALEXTCallback ()
- ディスプレイモ - ド設定 kmSetDisplayMode () を設定します。

上記手順で、PAL 拡張表示が可能になります。

5.7 注意、制限事項

(1) レンダリング

ストリップバッファは使用できません。レンダリング後に拡大しフレームバッファに書き込むためフレームバッファを増加したライン数分だけ余計に使用します。

(2) スキャンライン

描画する座標と表示する座標が合わなくなります。

描画座標とスキャンラインが合わなくなります。

(3) ちらつき

PAL 拡張モード時には通常のオブジェクトデータを元に縦 480 でレンダリングした後、フレームバッファに書き込む段階で設定した比率に従って拡大します。拡大方向へのレンダリングなのでフリッカーフィルタが有効にならないため、フリッカが発生する場合があります。フリッカを防止したい場合で レンダリングが次の V-Sync 信号がくるまでに収まる場合には、
KM_DSPMODE_PALNI640x480FFEXT モードを使用して下さい。

また、KM_DSPMODE_PALNI640x480FFEXT は V-Sync 同期モードで使用下さい。非同期モードでは動作しません。

フレームダブルバッファ方式で画像を作成しているため、
KM_DSPMODE_PALNI640x480FFEXT を使用した場合、次の V-Sync 信号までにレンダリング用のパラメータの用意が間に合わないときは、表示がおかしくなります。次の V-Sync 信号までにレンダリング用のパラメータの用意ができない可能性がある場合には、直前のレンダリングパラメータを使用して再度別のフレームバッファにレンダリングを行わせるよう、フレームバッファの数を 4 バッファ用意するようにして下さい。

(4) デフォルトの比率

PAL 拡張モードでは、コールバック関数が登録されていない場合にはデフォルトで縦拡大比率 = 1.066 を使用します。

5.8 PAL 拡張モード設定情報一覧

ディスプレイモード	縦方向拡大比率	ライン数
KM_DSPMODE_PALNI640x240EXT	KM_PAEXT_HEIGHT_RATIO_1_033	248
	KM_PAEXT_HEIGHT_RATIO_1_066	256
	KM_PAEXT_HEIGHT_RATIO_1_100	264
	KM_PAEXT_HEIGHT_RATIO_1_133	272
	KM_PAEXT_HEIGHT_RATIO_1_166	280
KM_DSPMODE_PALNI640x480FFEXT	KM_PAEXT_HEIGHT_RATIO_1_033	496
	KM_PAEXT_HEIGHT_RATIO_1_066	512
	KM_PAEXT_HEIGHT_RATIO_1_100	528
	KM_PAEXT_HEIGHT_RATIO_1_133	544
	KM_PAEXT_HEIGHT_RATIO_1_166	560
KM_DSPMODE_PALNI640x480EXT	KM_PAEXT_HEIGHT_RATIO_1_033	496
	KM_PAEXT_HEIGHT_RATIO_1_066	512
	KM_PAEXT_HEIGHT_RATIO_1_100	528
	KM_PAEXT_HEIGHT_RATIO_1_133	544
	KM_PAEXT_HEIGHT_RATIO_1_166	560
KM_DSPMODE_PALI640x480EXT	KM_PAEXT_HEIGHT_RATIO_1_033	496
	KM_PAEXT_HEIGHT_RATIO_1_066	512
	KM_PAEXT_HEIGHT_RATIO_1_100	528
	KM_PAEXT_HEIGHT_RATIO_1_133	544
	KM_PAEXT_HEIGHT_RATIO_1_166	560

ディスプレイモード設定関数 `kmSetDisplayMode()` でディスプレイモードを指定し、PAL 拡張モードコールバック関数で縦方向拡大比率情報の設定を行うことで、拡張表示されるようになります。

6 テクスチャフォーマット

ここでは、Kamui 上で使用するテクスチャフォーマットについて説明します。

6.1 Kamui のテクスチャフォーマット

Kamui のテクスチャフォーマットは、テクスチャのピクセルデータにサイズやタイプを示すヘッダ部(4×32 bit)が付いたものです。しかし Kamui ライブラリではヘッダ部分を一切参照しません。

そのかわり Kamui ライブラリは、テクスチャ読み込み関数などで指定された読み込み先のテクスチャサーフェスディスクリプタの情報から、テクスチャの形式などを判断します。ヘッダ部分は Ninja ライブラリなどの上位アプリケーションへの付加データです。

Kamui ライブラリのテクスチャ読み込み関数などでテクスチャのアドレスを指定する場合は、ヘッダ部分をスキップしてピクセルデータ部分の先頭アドレスを指定します。

なお、このピクセルデータ部分の先頭アドレスが 32 バイト境界にあり、かつピクセルデータ部分のサイズが 32 バイトの倍数であれば、テクスチャメモリへの転送に DMA を使用して高速な転送を行います。Kamui ライブラリでは、ピクセルデータ部分のサイズを 32 バイトの倍数に合わせるために、適宜 Dummy バイトを定義しています。

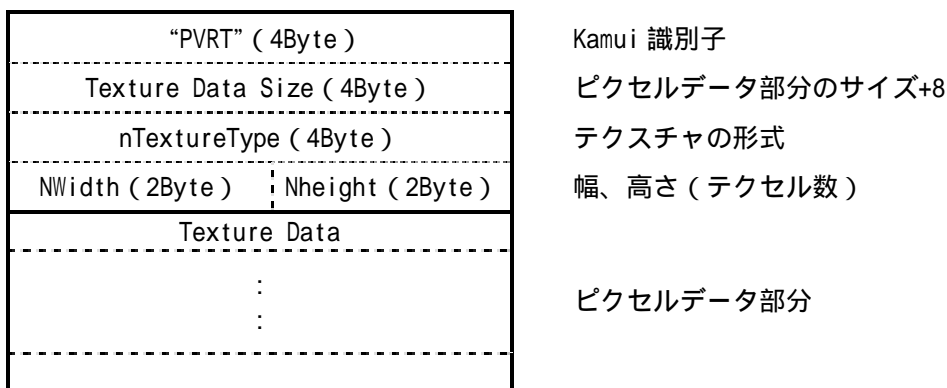


図 6-1 テクスチャフォーマット

- "PVRT"

Kamui のテクスチャの最初は、半角の"PVRT"4文字から始まります。これは Kamui のテクスチャであることを示す識別子を意味します。

- Texture Data Size

テクスチャのピクセルデータ部分のバイト数+8を保存します。複数の Kamui テクスチャを連続して1つのファイルに合成した場合、この値を使って次のテクスチャの先頭位置を知ることができます。

- nTextureType

カテゴリコードとピクセルフォーマットコードでテクスチャの形式を指定します。

enum 値を論理和して指定します。なお、このフィールドの上位 16 ビットは Ninja ライブラリのために予約されています。

● カテゴリコード

定 義	意 味
KM_TEXTURE_TWIDDLED, KM_TEXTURE_TWIDDLED_MM KM_TEXTURE_TWIDDLED_MM_ DMA	Twiddled-NonVQ-Square 形式です。
KM_TEXTURE_TWIDDLED_RECT ANGLE	Twiddled-NonVQ-Rectangle 形式です。 長方形テクスチャを意味します。
KM_TEXTURE_VQ, KM_TEXTURE_VQ_MM	Twiddled-VQ-Square 形式です。 VQ (Vector Quantization) 圧縮したテクスチャのことで、 テクスチャデータの前半は、コードブック・テーブルを意味 します。
KM_TEXTURE_SMALLVQ	Twiddled-small-VQ-Square 形式です。VQ (Vector Quantization) 圧縮したテクスチャのことで、テクスチャ データの前半はコードブック・テーブルを意味します。
KM_TEXTURE_SMALLVQ_MM	Twiddled-small-VQ-Square 形式です。 ミップマップを持つテクスチャのことです。
KM_TEXTURE_PALETTIZE4, KM_TEXTURE_PALETTIZE4_MM	Twiddled-NonVQ-Palettized 形式です。 インデックスが 4bpp (16 色) のパレット化テクスチャのこ とで、パレットはシステムに 1 つしかもてないため、テクス チャデータには、パレットの情報は含まれません。
KM_TEXTURE_PALETTIZE8, KM_TEXTURE_PALETTIZE8_MM	Twiddled-NonVQ-Palettized 形式です。 インデックスが 8bpp (256 色) のパレット化テクスチャの ことです。
KM_TEXTURE_RECTANGLE	Scan Order-Rectangle 形式の長方形テクスチャです。 ピクセルデータの配置は、Windows BMP 形式に準じます。
KM_TEXTURE_STRIDE	Scan Order-Stride 形式の長方形テクスチャです。 ピクセルデータの配置は、Windows BMP 形式に準じます。

表中で、_MM 付きは、ミップマップを持つテクスチャです。

● ピクセルフォーマットコード

定 義	意 味
KM_TEXTURE_RGB565	アルファ値がなく、R 値と B 値がそれぞれ 5 ビット、G 値 のみ 6 ビットからなる形式です。
KM_TEXTURE_ARGB4444	アルファ値、RGB 値それぞれ 4 ビットからなる形式です。 アルファ値は 0x0 で完全透明、0xF で完全不透明です。
KM_TEXTURE_YUV422	YUV422 形式です。
KM_TEXTURE_BUMP	バンプマッピング用テクスチャの指定します。
KM_TEXTURE_ARGB1555	アルファ値が 1 ビット、RGB 値がそれぞれ 5 ビットからな る形式です。 アルファ値は 0x0 で完全透明、0xF で完全不透明です。

ARGB8888 カラーは、Palettized 形式のみに指定できます。この場合は、nTextureType ではなく
kmPaletteMode () で、パレットに対して設定します。

● nWidth , nHeight

テクスチャの横サイズ及び縦サイズを指定します。この場合のテクスチャの縦・横サイズは、8 ,
16 , 32 , 64 , 128 , 256 , 512 , 1024 のいずれかでなければなりません。

● Texture Data

テクスチャのピクセルデータです。テクスチャメモリ上のイメージに準じます。

Kamui ライブラリのテクスチャ関連関数で、パラメータ pTexture で指定するテクスチャのアドレスは、この部分の先頭アドレスです。その際に、ヘッダ部分はスキップすることになりますので注意して下さい。

なお、このピクセルデータの先頭アドレスは、32 バイト境界に配置した値が適切です。

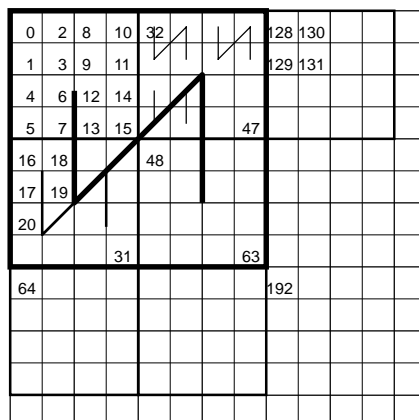
6.2 Twiddled 及び Twiddled ミップマップフォーマット

Twiddled-NonVQ-Square / Twiddled-NonVQ-Square ミップマップ形式のフォーマットを、通称 Twiddled / Twiddled ミップマップフォーマットと呼びます。

Twiddled フォーマットは PowerVR2DC のテクスチャの基本フォーマットです。

このフォーマットは、Bilinear フィルタなどのテクスチャフィルタリングを高速で行うために、アドレスリングをハードウェアに最適化しています。このため、Twiddled フォーマットのピクセルは、ラスタオーダで並んでいません。

Twiddled フォーマットのピクセルオーダは次の図のようになります。なお、Twiddled フォーマットは 16bpp です、実バイトアドレスを得るためには 2 倍する必要があります。



PCI/SH4 から見たテクスチャメモリのバイトアドレスでのイメージは、下記ようになります。

+00h	Dummy Zero (6h Bytes)
+06h	1x1 map (2h Bytes)
+08h	2x2 map (8h Bytes)
+10h	4x4 map (20h Bytes)
+30h	8x8 map (80h Bytes) :

SGL との互換性のため、ヘッダの後ろに 2 バイトの Dummy バイトを持っています。

Kamui テクスチャフォーマットでの Twiddled ミップマップフォーマットは、下記ようになります。その際に、Twiddled ミップマップ形式は、常に正方形でなくてはなりません。

+00h	KAMUI Texture Header (10h Bytes)
+10h	Dummy Zero (2h Bytes)
+12h	1x1 mipmap texture (2h Bytes)
+14h	2x2 mipmap texture (8h Bytes)
+1Ch	4x4 mipmap texture (20h Bytes)
+3Ch	8x8 mipmap texture (80h Bytes)
+BCh	16x16 mipmap texture (200h Bytes)
+2BCh	32x32 mipmap texture (800h Bytes)
+ABCh	64x64 mipmap texture (2000h Bytes)
+2ABCh	128x128 mipmap texture (8000h Bytes)
+AABCh	256x256 mipmap texture (20000h Bytes)
+2AABCh	512x512 mipmap texture (80000h Bytes)
+AAABCh	1024x1024 mipmap texture (200000h Bytes)

6.3 VQ 及び VQ ミップマップフォーマット

Twiddled-VQ-Square / Twiddled-VQ-Square-ミップマップ形式のフォーマットを、通称 VQ / VQ Mipamp フォーマットと呼びます。

VQ (Vector Quantization : ベクトル量子化) フォーマットは、高圧縮率の圧縮テクスチャフォーマットです。

VQ フォーマットは、大まかに 2 つの領域から構成されています。一つはコードブックと呼ばれるカラーテーブルで、もう一つはこのコードブックの位置を指す Index データです。

VQ フォーマットの構造は、Index データをコードブックより伸長し画像を作成するという点で、パレットテクスチャと非常に似ています。ただし、このコードブックはテクスチャごとに設定することが可能なものです。

VQ フォーマットもまた、各ピクセルインデックスは Twiddled 形式に並んでいます。

VQ フォーマットは、コードブック部分とインデックスデータ部分の 2 つから成っています。コードブックのサイズは常に 256 エントリで、1 エントリあたり 4 テクセルのデータを保持しています。

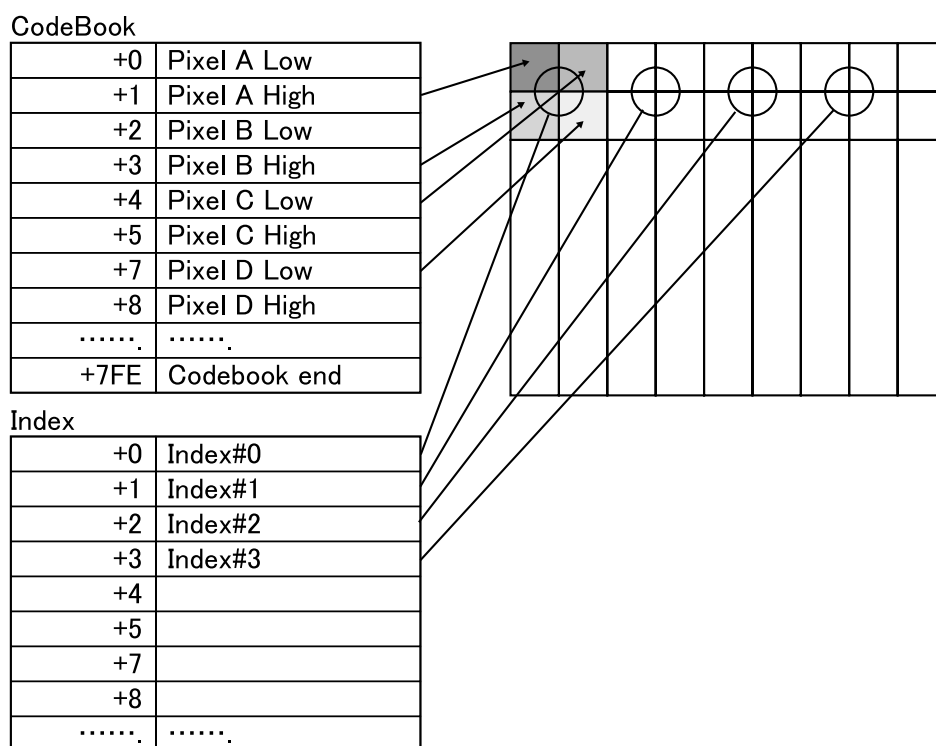


図 6 - 2 VQ のイメージ

例えば、256 × 256 のテクスチャを VQ 圧縮すると、以下のように計算できます。

$$\frac{\text{元画像}}{\text{インデックス+コードページ}} = \frac{256 \times 256 \times 16}{(128 \times 128 \times 8) + (256 \times 16 \times 4)} = \frac{7.1}{1}$$

このように、サイズが約 7 分の 1 となります。

PCI/SH4 から見たテクスチャメモリのバイトアドレスでのイメージは、下記ようになります。

- [VQ]

+00h	CodeBook (800h Bytes)
+800h	Index

- [VQ ミップマップ]

+00h	CodeBook (800h Bytes)
+800h	Dummy Zero (1h Byte)
+801h	2x2 mipmap Index (1h Byte)
+802h	4x4 mipmap Index (4h Bytes)
+806h	8x8 mipmap Index (10h Bytes)
+816h	16x16 mipmap Index (40h Bytes)
	:
+???h	Dummy Zero (Ah Byte)

図 6 - 3 テクスチャメモリのバイトアドレスでのイメージ

VQ では、1x1 ミップマップの Index は存在しません。
 Kamui での VQ フォーマットは、下記ようになります。
 (VQ フォーマットは、常に正方形でなくてはなりません)

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (800h Bytes)
+810h	Index

テクスチャのインデックス部のサイズは、以下のようになります。

縦・横サイズ	インデックス部 サイズ
8 × 8	10h Bytes ⁴
16 × 16	40h Bytes
32 × 32	100h Bytes
64 × 64	400h Bytes
128 × 128	1000h Bytes
256 × 256	4000h Bytes
512 × 512	10000h Bytes
1024 × 1024	40000h Bytes

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (800h Bytes)
+810h	Dummy Zero (1h Byte)
+811h	2x2 mipmap Index (1h Bytes)
+812h	4x4 mipmap Index (4h Bytes)
+816h	8x8 mipmap Index (10h Bytes)
+826h	16x16 mipmap Index (40h Bytes)
+866h	32x32 mipmap Index (100h Bytes)
+966h	64x64 mipmap Index (400h Bytes)
+D66h	128x128 mipmap Index (1000h Bytes)
+1D66h	256x256 mipmap Index (4000h Bytes)
+5D66h	512x512 mipmap Index (10000h Bytes)
+15D66h	1024x1024 mipmap Index (40000h Bytes)
+nnnh	Dummy Zero (Ah Byte)

VQ ミップマップテクスチャには、コードブックとインデックスデータの間に 1 バイト、もしくはインデックスデータの後ろに 10 バイトのダミーデータが必要になります。

⁴ テクスチャサーフェスは 32 バイト単位で確保するので、8 × 8 の VQ テクスチャのインデックスデータは、テクスチャメモリ上では 32 (20h) バイト消費します。

6.4 スモールVQ及スモールVQミップマップフォーマット

Twiddled-small-VQ-Square / Twiddled-small-VQ-Square-Mipmap 形式のフォーマットを、通称スモール VQ / スモール VQ ミップマップフォーマットと呼びます。

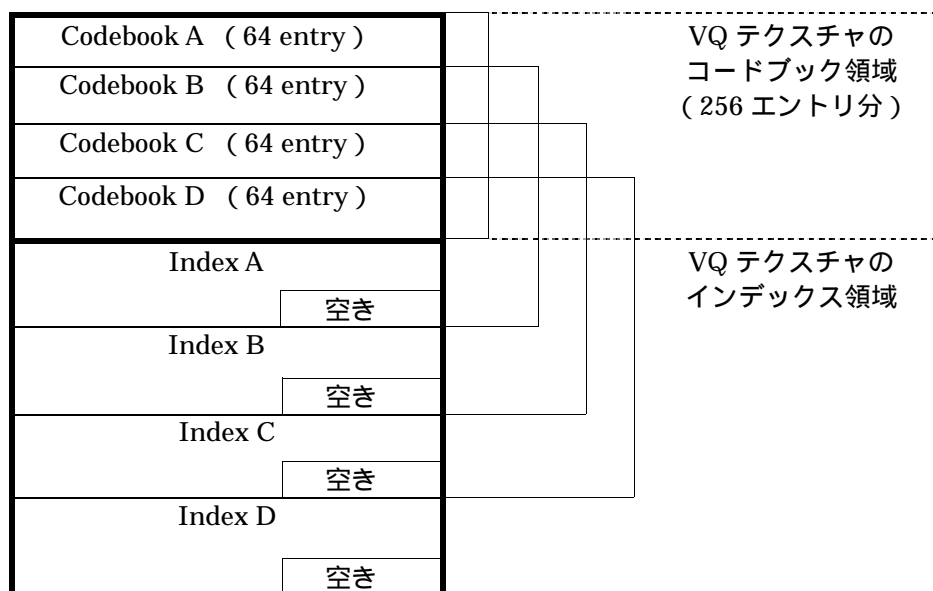
前述の VQ / VQ ミップマップフォーマットでは、コードブックのサイズが 256 エントリ (2048 バイト) 固定であるため、テクスチャの縦・横サイズが 64×64 テクセルより小さい場合には圧縮効率が悪化し、Twiddled フォーマットより大きいテクスチャメモリが必要になってしまいます。

6.4.1 small VQ テクスチャ

small VQ テクスチャとは、テクスチャの縦・横サイズにあわせてコードブックのサイズを小さくした VQ 圧縮形式のテクスチャのことを呼びます。

Small VQ テクスチャの基本的な構造は、前述の VQ / VQ ミップマップフォーマットと同じです。small VQ は VQ とハード的に同じものです。

また、コードブック部の先頭アドレスからインデックス部の先頭アドレスまでの相対距離は、常に 2048 バイトでなければなりません。そこで、下図のように小さなコードブックをもつ複数の VQ テクスチャをグループ化 (組み合わせ) することにより、小さいサイズの VQ テクスチャを定義することができます。これが small VQ テクスチャです。



この例では、コードブックを 64 エントリずつの 4 つに分割し、それぞれに 4 つの small VQ テクスチャをロードしています。

それぞれのテクスチャを参照するには、それぞれのコードブックの先頭アドレスを指定します。

Small VQ テクスチャで、メモリの使用効率を上げるには、上図のインデックス部分にできる「空き」エリアを極力小さくしなければなりません。そのためには、各 small VQ のコードブックのサイズとインデックスのサイズが等しいか、またはインデックスの方が小さくしなければなりません。以上より、各 small VQ テクスチャにおけるコードブックのサイズは下記の組み合わせが最もメモリ使用効率がよいことになります。

Size	ミップマップ	Codebook size		Index size (Bytes)	「空き」領域の サイズ	グループ化 できる数
		Entry	Bytes			
8x8	No	16	128 (80h)	16 (10h)	112 (70h)	16
	Yes	16	128 (80h)	32 (20h)	96 (60h)	16
16x16	No	16	128 (80h)	64 (40h)	64 (40h)	16
	Yes	16	128 (80h)	96 (60h)	32 (20h)	16
32x32	No	32	256 (100h)	256 (100h)	0	8
	Yes	64	512 (200h)	352 (160h)	160 (A0h)	4
64x64	No	128	1024 (400h)	1024 (400h)	0	2
	Yes	256	2048 (800h)	1376 (560h)	672 (2A0h)	1

この図から判別すると、32×32、64×64 でミップマップなしが最も効率良く（「空き」領域のサイズがゼロ）なり、逆に 64×64 でミップマップありでは、通常の VQ テクスチャよりサイズが大きくなり、最も効率が悪くなることがわかります。

また、8×8 のテクスチャの場合、テクスチャの全ピクセル数は 64 ピクセルです。そのため、16 エントリ未満の小さなコードブックを使用しないと圧縮したことになりません。テクスチャデータのアライメントが 32 byte なので 16 エントリ以下はないため、8×8 のテクスチャは small VQ を利用しても圧縮効果は得られません。同様に 8×8 ミップマップにおいても、Twiddled 形式の方が、サイズが小さくなります。

注 意 表に網掛けで示した部分は、Small VQ 形式より、Twiddled あるいは通常の VQ 形式の方がメモリ使用効率がよくなるため Kamui ライブラリではサポートしていません。注意して下さい。

small VQ テクスチャは、Kamui ライブラリではメモリ上で同じサイズのものどうしをグループ化して保存します。そのため、各サイズの small VQ テクスチャを同時にいくつ使うかによっても、メモリの使用効率が変化します。

たとえば、32×32 No-ミップマップテクスチャを 1 個しか使用しない場合も、8 個使用する場合も、テクスチャメモリ上では同じ容量（（256 + 256）× 8 = 4096 バイト）の領域がこのテクスチャのために予約されます（ただし kmGetFreeTextureMem（）では、どちらの場合も正しい空き容量を得ることが可能です）。

従って、small VQ テクスチャを使用する場合は、同時に使用するテクスチャの数にも注意する必要があります。

上の表で、各サイズに関して「グループ化できる数」の倍数を使用するのが最も効率良くなります。なお、グループ化は Kamui がロード時に自動的に行います。

アプリケーションが準備する small VQ のフォーマットは、下記ようになります。

- 16x16 small VQ ファイルフォーマット

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (80h Bytes)
+90h	Index (40h Bytes)

- 32x32 small VQ ファイルフォーマット

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (100h Bytes)
+110h	Index (100h Bytes)

- 64x64 small VQ ファイルフォーマット

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (400h Bytes)
+410h	Index (400h Bytes)

また、small VQ ミップマップの形式は、下記ようになります。

- 16x16 small VQ ミップマップ

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (80h Bytes)
+90h	Dummy Zero (1h Byte)
+91h	2x2 mipmap Index (1h Bytes)
+92h	4x4 mipmap Index (4h Bytes)
+96h	8x8 mipmap Index (10h Bytes)
+A6h	16x16 mipmap Index (40h Bytes)
+E6h	Dummy Zero (Ah Byte)

- 32x32 small VQ ミップマップ

+00h	KAMUI Texture Header (10h Bytes)
+10h	Code Book (200h Bytes)
+210h	Dummy Zero (1h Byte)
+211h	2x2 mipmap Index (1h Bytes)
+212h	4x4 mipmap Index (4h Bytes)
+216h	8x8 mipmap Index (10h Bytes)
+226h	16x16 mipmap Index (40h Bytes)
+266h	32x32 mipmap Index (100h Bytes)
+366h	Dummy Zero (Ah Byte)

6.5 Palettized 4bpp/8bpp フォーマット

Twiddled-NonVQ-Palettized 形式のフォーマットを、通称 Palettized フォーマットと呼びます。Palettized フォーマットには 4bpp と 8bpp の 2 種類が存在します。パレットはシステムに 1 つしかなく、そのエントリは全部で 1024 個あります。

- Palettized-4bpp の場合
1024 個のエントリを 64 のバンクに分けて使用します。(1024 エントリ / 16 色 = 64 バンク)
- Palettized-8bpp の場合
1024 個のエントリを 4 つのバンクに分けて使用します。(1024 エントリ / 256 色 = 4 バンク)

各バンクは、物理的に分離しているわけではなく、1024 のエントリへのポインタを計算で求めることにより作り出しています。1 つのシーンの中で、4bpp のテクスチャと 8bpp のテクスチャは混在可能ですが、1024 個のエントリの中で重なった部分については共用されます。その結果、パレットの内容の変更は、4bpp と 8bpp テクスチャの両方に影響が出ます。

パレットのバンクは、ストリップ単位で指定できます。

バンク番号は、`kmGenerateStripHead()` のストリップ形式で、頂点データパラメータ構築時の引数で指定します。

変更する場合には、`kmChangeStripPaletteBank()` を使用します。実際に使用されるエントリは、パレットバンク番号 (PaletteBank) とテクスチャの各テクセルのインデックス値 (index_data) より、下記のように選択されます。

```
if (PixelFormat == 8BPP)
{
    palette_entry = (PaletteBank << 4) & 0x300 + index_data;
}

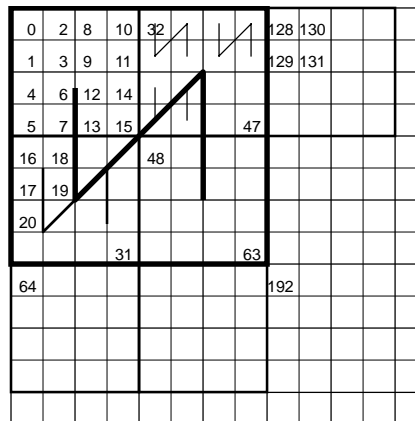
if (PixelFormat == 4BPP)
{
    palette_entry = (PaletteBank << 4) + index_data;
}
```

PaletteBank に指定できる値は、4bpp の時は 0 ~ 63 のいずれかになります。

また、8bpp の時も 0 ~ 63 ですが、この場合有効なのは 6 bit 中上位 2 bit のみなので、使用できる値は 0 (0 ~ 15), 16 (16 ~ 31), 32 (32 ~ 47), 48 (48 ~ 63) の 4 種の中からいずれかになります。

メモリ上のフォーマットは、twiddled フォーマットとほぼ同じ形式です。ただし、4bpp の場合は 4 ドットが 1 (16bit) ワードに、8bpp の場合は 2 ドットが 1 ワード (16bit) にまとめられています。

テクスチャのフォーマットは、以下のとおりになります。



アドレス順序は Twiddled と変わりません。これを (U=0, V=0), (U=0, V=1), (U=1, V=0), (U=1, V=1) の順にリトルエンディアンでパッキングします。従って 4bpp、及び 8bpp の場合は次のようになります。(Palettized / Palettized ミップマップ形式は、常に正方形でなくてはなりません)

● 4bpp の場合のテクセル並び

Bits 15-12	Bits 11-8	Bits 7-4	Bits 3-0
Texel U=1 V=1	Texel U=1 V=0	Texel U=0 V=1	Texel U=0 V=0

● 8bpp の場合のテクセル並び

Bits 15-8	Bits 7-0
Texel U=0 V=1	Texel U=0 V=0

Kamui テクスチャフォーマットでの Palettized フォーマットは、下記のようにになります。

+00h	KAMUI Texture Header (10h Bytes)
+10h	Texture Pixel Data

テクスチャのピクセルデータのサイズは、以下のようになります。

縦・横サイズ	ピクセルデータ サイズ	
	4bpp	8bpp
8x8	20h Bytes	40h
16x16	80h Bytes	100h
32x32	200h Bytes	400h
64x64	800h Bytes	1000h
128x128	2000h Bytes	4000h
256x256	8000h Bytes	10000h
512x512	20000h Bytes	40000h
1024x1024	80000h Bytes	100000h

Kamui テクスチャフォーマットでの Palettized ミップマップの形式は、下記ようになります。

4bpp			8bpp	
Offset	Length	Data	Length	Offset
+00h	10h	KAMUI Texture Header	10h	+00h
+10h	1h	Dummy Zero	3h	+10h
+11h	1h	1x1 mipmap Index	1h	+13h
+12h	2h	2x2 mipmap Index	4h	+14h
+14h	8h	4x4 mipmap Index	10h	+18h
+1Ch	20h	8x8 mipmap Index	40h	+28h
+3Ch	80h	16x16 mipmap Index	100h	+68h
+BCh	200h	32x32 mipmap Index	400h	+168h
+2BCh	800h	64x64 mipmap Index	1000h	+568h
+ABCh	2000h	128x128 mipmap Index	4000h	+1568h
+2ABCh	8000h	256x256 mipmap Index	10000h	+5568h
+AABCh	20000h	512x512 mipmap Index	40000h	+15568h
+2AABCh	80000h	1024x1024 mipmap Index	100000h	+55568h
+????h	14h	Dummy Zero	8h	+????

Palettized-4bpp のミップマップでは、インデックスデータの前に 1 バイト、後ろに 20 バイトのダミーデータが必要になります。

Palettized-8bpp のミップマップでは、インデックスデータの前に 3 バイト、後ろに 8 バイトのダミーデータが必要になります。

6.6 Rectangle フォーマット

Scan Order-Rectangle 形式のフォーマットを、通称 Rectangle フォーマットと呼びます。

注意 その他に、Twiddled-NonVQ-Rectangle 形式のテクスチャも存在するので、注意して下さい。

Rectangle フォーマットは、U、V サイズに違う値を設定できるテクスチャです。このフォーマットを使用した場合は、ミップマップすることはできませんし、パフォーマンスも Twiddled フォーマットに比べて低下します。

ただし、Rectangle フォーマットはラスタオーダーでピクセルデータが並ぶため、非常にアクセスが簡単です。(Twiddled フォーマットの様な複雑なアドレッシングを行う必要はありません) また、Rectangle フォーマットはレンダリングの対象とすることができます⁵ので、環境マッピングを行う場合などに使用することができます。

なお、Rectangle フォーマットで指定できるテクスチャの縦・横サイズは、8、16、32、64、128、256、512、1024 のいずれかになります。

Kamui テクスチャフォーマットでの Rectangle フォーマットは、下記のようになります。

+00h	KAMUI Texture Header (10h Bytes)
+10h	Texture Pixel Data

ピクセルデータの配置は、Windows の BMP (24bpp) 形式にならい、最初に (U=0,V=0) (テクスチャの左下) のピクセルデータ 2 バイトが置かれ、以降 (U=1,V=0) (U=2,V=0), ..., (U=Umax,V=0), (U=0,V=1), ..., の順に配置されます。

テクスチャのピクセルデータのサイズは、以下のように求められます。

サイズ = 横のテクセル数 (U-size) × 縦のテクセル数 (V-size) × 2 (Byte per texel)

6.7 Stride フォーマット

Scan Order-Stride 形式のフォーマットを、通称 Stride フォーマットと呼びます。Stride フォーマットは Rectangle フォーマットの特異な形です。Stride 値はレジスタへ設定するため、レンダリングのときに設定されている値が使用されます。その値に、以下のアドレッシングを用いてテクセルを決定します。

$$\text{Addr} = u + v \times \text{stride}$$

すなわち、テクスチャの横幅テクセル数が Stride 値によって自由に設定できます。ただし、Stride 値は 32 の倍数しか指定できません (32 - 992)。

たとえば、環境マッピングなどを使用する場合に 1024 × 1024 のテクスチャ中に 640 × 480 のエリアを作成する場合は、Stride 値として 640 を指定します。

KmRenderTexture () を使用してこのテクスチャサーフェスに対してレンダリングを行うと、画面の最初の 1 ライン ((x,y) = (0,0) - (639,0)) の部分が、テクスチャサーフェスの (U,V) = (0,0) - (639,0) に書き込まれ、2 ライン目 ((x,y) = (0,1) - (639,1)) の部分が、テクスチャサーフェスの (U,V) = (640,0) - (1023,0) と (0,1) - (255,1) に書き込まれます。

⁵ ただし、フレームバッファが 16bpp のモードでテクスチャに指定しているカラーモード (1555, 4444, 565) と同一である必要があります。

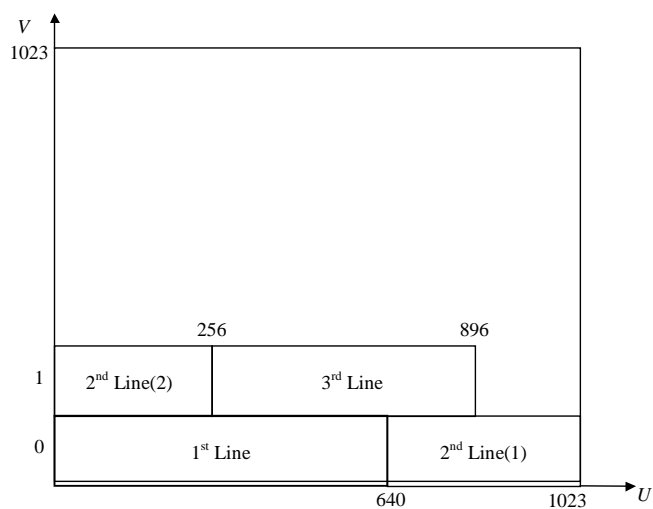


図 6 - 4 Stride フォーマット

そして、このテクスチャを使用してテクスチャマッピングを行う場合は、 $(U, V) = (0.0f, 0.0f) - (0.625f, 0.46875f)$ とすれば、画面全体をテクスチャとして張り付けることが可能です。

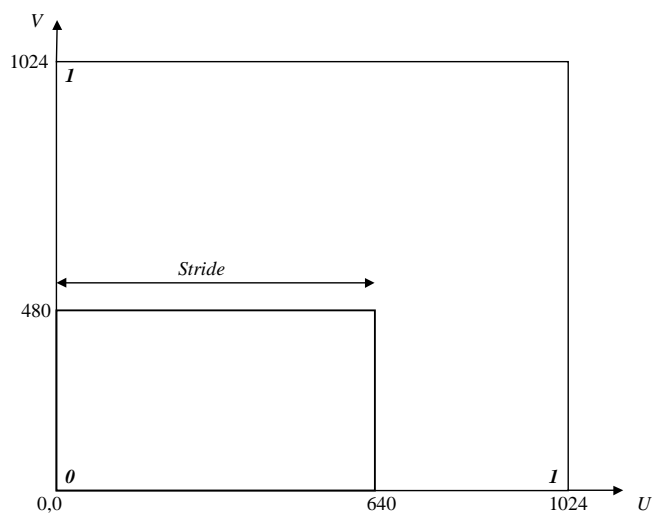


図 6 - 5 Stride フォーマットのテクスチャマッピング

ノート Kamui の Stride フォーマットは、Rectangle フォーマットと全く同じです。違いは、ヘッダの `nTextureType` に指定するテクスチャタイプのフラグのみです。

6.8 BUMP-Mapping フォーマット

PowerVR2DC での BUMP-Mapping は以下の手法によって実現します。

BUMP-Mapping 情報は、通常テクスチャのピクセルデータの代わりに、法線ベクトルをテクスチャデータに格納することによって、表面の凹凸を表現します。加えて、光源データをオフセットカラーのビットに設定し、この光源データに基づいてドットごとの内積を計算し表示します。

注意 BUMP-Mapping は、テクスチャ・オフセット付きの FLAT ポリゴンのみでしか使用することはできませんので、注意して下さい。

PowerVR2DC 内部では BUMP-Mapping の計算に極座標系を使用します。

まず、テクスチャ内に格納される法線ベクトルは、次の式で表されます。

$$\begin{aligned} x_s &= \cos(s')\cos(r') & s' &= \pi/2 \frac{s}{256} \\ y_s &= \sin(s') & \text{ここで} & \\ z_s &= \cos(s')\sin(r') & r' &= 2\pi \frac{r}{256} \end{aligned}$$

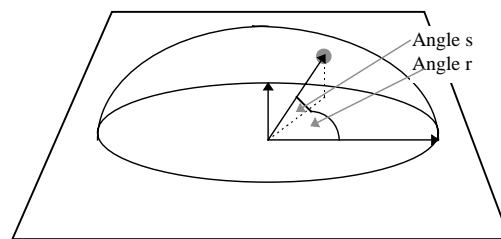


図 6 - 6 法線ベクトル

s 及び r がテクスチャ内に格納されます。

この場合の s 及び r は、8 bit のデータで上記の式で表わされるテクスチャ内 BUMP データの仰角及び方位角を表わします。通常、PowerVR2DC のテクスチャは 16bit でするので、1 テクスセルに付き 1 セットの s 及び r があることとなります。

次にこのテクスチャを貼り付けたポリゴンに対する光源ベクトルですが、これも同様に極座標を使用して表現されます。

$$\begin{aligned} x_l &= \cos(t')\cos(q') \\ y_l &= \sin(t') \\ z_l &= \cos(t')\sin(q') \end{aligned}$$

この場合の t 及び q は、テクスチャ内の法線ベクトルと同じく、

$$\begin{aligned} t' &= \pi/2 \frac{t}{256} \\ q' &= 2\pi \frac{q}{256} \end{aligned}$$

のように表されます。

この、光源ベクトルをポリゴンに導入するに当たって、Scale Factor を導入します。これは光源の強さを表わすことから、以下はこの Scale Factor を Strength と呼称します。

この Strength を用いて、以下の K_1 , K_2 , K_3 を求めます。

$$\begin{aligned}k_1 &= 1 - strength \\k_2 &= strength \cdot \sin(t') \\k_3 &= strength \cdot \cos(t')\end{aligned}$$

Strength 及び K 値は、1 に正規化した値で 8 bit の値となります。

ポリゴンに対してこの光源情報を与えるのは、以下のようにオフセットカラーの 32 bit データを使用します。

Base Colour: xRGB			
K_1	K_2	K_3	q'

VertexType5 などの Floating Color VertexType を使用する場合には、上記のデータを 1 に正規化して入力し、VertexType4 などの「Packed Color」、VertexType を使用する場合には、8 ビットデータ変換しパックする必要があります。

最終的なテクセルごとの明るさは、上記のテクスチャ法線ベクトル、光源法線ベクトルから、内積によって求められます。

$$\text{DotProduct} = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} \cdot \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = x_s x_l + y_s y_l + z_s z_l$$

注 意 また、BUMP-Mapping されたポリゴンに対して、さらにテクスチャを貼ることはできませんので注意して下さい。

さらにテクスチャを貼りたい場合には、PowerVR2DC では半透明のオートソートは GreaterEqual であることを利用し、BUMP-Mapping されたポリゴンと同一座標の半透明ポリゴンを重ねて登録することで、BUMP-Mapping ポリゴンにテクスチャを貼り付けることが可能です。

7 Kamui2 ライブラリで廃止された関数

ここでは、Kamui1 ライブラリで作成されたアプリケーションを、Kamui2 ライブラリで作成し直す場合についての対処方法を説明します。

7.1 廃止関数の一覧

Kamui1 でサポートされていた以下の関数は、Kamui2 では廃止されました。

関 数	機 能
kmStopDisplayFrameBuffer ()	フレームバッファの表示停止
kmChangeLatencyModel ()	レイテンシモードの切り替え
kmCreateFrameBufferSurface ()	フレームバッファサーフェスの生成
kmSetAlphaThreshold ()	アルファスレッシュホールド値の設定
kmActiveFrameBuffer ()	表示画面の決定
kmFlipFrameBuffer ()	表示画面の Flip
kmSetAutoSortMode ()	オートソートモードの設定
kmSetWaitVsyncCount ()	Vsync Wait 数の指定
kmSetUserClipLevelAdjust ()	画面の 2 分割モードの設定
kmSetStripLength ()	分割ストリップ形式データ長の設定
kmCreateVertexBuffer ()	頂点データ登録用バッファのアロケート
kmDiscardVertexBuffer ()	頂点データ登録用バッファの開放
kmUseAnotherModifier ()	モディファイアボリュームリストの指定

7.1.1 ダイレクトモード用関数について

以下の関数は、ダイレクトモード用の関数です。Kamui2 ライブラリでは、ダイレクトモードは廃止されました。

2 V レイテンシモデルの場合、1 つのリストに対して直接転送となりますが、この場合も Kamui の管理下での直接転送で、アプリケーションから直接タイルアクセラレータを制御することはできません。

関 数	機 能
kmGetCurrentVertexOffset ()	Vertex Buffer の現在書き込み位置の取得
kmChangeVertexPCW ()	Vertex コントロールワードの直接書き換え
kmFlushVertexBuffer ()	Vertex Buffer の Flush
kmCreateTABuffer ()	ネイティブデータバッファの確保
kmStartVertexStripDirect ()	頂点データストリップ登録の開始
kmSetVertexDirect ()	頂点データの直接登録
kmSetUserClippingDirect ()	ユーザークリッピング領域の設定
kmSetEndOfListDirect ()	頂点登録の終了通知
kmRenderDirect ()	頂点データ登録の終了通知
kmRenderTextureDirect ()	テクスチャメモリへのレンダリング

7.2（廃止関数の）対応方法

- `kmStopDisplayFrameBuffer ()` / フレームバッファの表示停止
`kmBlankScreen (KM_TRUE)` で、表示が OFF の状態になります。
- `kmChangeLatencyModel ()` / レイテンシモードの切り替え
`kmSetSystemConfiguration` 呼び出し時の `KMSYSTEMCONFIGSTRUCT` 構造体の `flag` 情報へ `KM_CONFIGFLAG_ENABLE_2V_LATENCY` を設定すると、2 V レイテンシモードで動作します。これが指定されない場合は、3 V レイテンシモードとなります。
- `kmCreateFrameBufferSurface ()` / フレームバッファサーフェスの生成
`kmSetSystemConfiguration ()` で作成されます。`kmSetDisplayMode ()` で設定されたディスプレイモードと `KMSYSTEMCONFIGSTRUCT` 構造体の `nNumOfFrameBuffer` で指定されたフレームバッファ数に従い、フレームバッファを取得します。
- `kmSetAlphaThreshold ()` / アルファスレッシュホールド値の設定
`PunchThrough` ポリゴンに対してのみ設定可能です。
`KmSetPunchThroughThreshold (KMDWORD dwThreshold)` で設定します。
`DwThreshold` には 0 ~ 2 5 5 の値を設定します。
- `kmActiveFrameBuffer ()` / 表示画面の決定 と `kmFlipFrameBuffer ()` / 表示画面の Flip
`kmRender (KM_RENDER_FLIP)` を呼び出すことにより、レンダリング終了した次の V タイミングで表示されます。表示用フレームバッファ及び描画用フレームバッファ切り替えは、Kamui が管理します。
- `kmSetAutoSortMode ()` / オートソートモードの設定
`kmSetSystemConfiguration ()` 呼び出し時の `KMSYSTEMCONFIGSTRUCT` 構造体の `Pass[pass]`, `dwRegionArrayFlag` へ以下の設定を行います。

定 義	意 味
<code>KM_PASSINFO_AUTOSORT</code>	現在の描画処理における Translucent ポリゴンのソートモードを AutoSort にする場合に、設定します。
<code>KM_PASSINFO_PRESORT</code>	現在の描画処理における Translucent ポリゴンのソートモードを PreSort にする場合に、設定します。

- `kmSetWaitVsyncCount ()` / Vsync Wait 数の指定
`kmSetVsyncCount` に
- `kmSetUserClipLevelAdjust ()` / 画面の 2 分割モードの設定
廃止された機能です。
- `kmSetStripLength ()` / 分割ストリップ形式データ長の設定
頂点タイプ別レンダリングパラメータ構築時に、タイプ別に設定されます。
- `kmCreateVertexBuffer ()` / 頂点データ登録用バッファのアロケート と `kmDiscardVertexBuffer ()` / 頂点データ登録用バッファの開放
`kmSetSystemConfiguration ()` 呼び出し時の `KMSYSTEMCONFIGSTRUCT` 構造体の `Pass[pass].FBufferSize` により、頂点バッファ配分を行われ、Kamui で管理されます。バッファ開放の機能はありません。

- kmUseAnotherModifier () / モディファイアポリュームリストの指定
 kmSetSystemConfiguration() 呼び出し時の KMSYSTEMCONFIGSTRUCT 構造体の Pass[pass]. dwRegionArrayFlag へ以下の設定を行います。

定 義	意 味
KM_PASSINFO_UA_TRMOD_AS_OP MOD	TransModifier は、OpaqueModifier と同じものを使用します。KM_PASSINFO_USE_ANOTHERLIST が設定されていない場合は、無効です。
KM_PASSINFO_UA_OPMOD_AS_TR MOD0	paqueModifier は、TransModifier と同じものを使用します。KM_PASSINFO_USE_ANOTHERLIST が設定されていない場合は、無効です。
KM_PASSINFO_UA_DISCADING_TR ANSPOLY	TransPolygon を PunchThroughPolygon として使用します。KM_PASSINFO_USE_ANOTHERLIST が設定されていない場合は、無効です。

8 Kamui2 ライブラリの関数一覧

Kamui2 ライブラリでは、以下の関数がサポートされています。
詳しい内容については、関数リファレンスをご参照下さい。

関数名	機 能
kmSetSystemConfiguration	システム構成を設定
kmSetEndOfVertexCallback	シーンのデータ転送終了時に呼ばれるコールバック関数の登録
kmSetEndOfYUVCallback	YUV 終了割り込みコールバック関数の登録
kmSetEORCallback	レンダリング終了時のコールバック関数の登録
kmSetFatalErrorCallback	Fatal エラー発生時のコールバック関数の登録
kmSetHSyncCallback	HSync 割り込みコールバック関数の登録
kmSetStripOverRunCallback	StripBuffer の縦サイズの表示期間中に次の Strip のレンダリングが終しなかった場合に呼ばれるコールバック関数の登録
kmSetTexOverflowCallback	テクスチャメモリの取得に失敗した時のコールバック関数の登録
kmSetVSyncCallback	VSync 割り込みコールバック関数の登録
kmSetWaitVsyncCallback	VSync 待ちのコールバック関数の登録
kmAdjustDisplayCenter	フレームバッファの表示位置の設定
kmBlankScreen	フレームバッファの画面への表示 / 非表示
kmChangeDisplayAntiAliasMode	アンチエイリアシングフィルタの設定
kmChangeDisplayDitherMode	ディザ処理の設定
kmGetCurrentDisplaySurface	現在表示中のサーフェスの取得
kmGetCurrentScanline	現在の HSync ラインを読み出し
kmGetDisplayColorMode	カラーモードの取得
kmGetDisplayFilterMode	フィルターモードの取得
kmGetDisplayInfo	ディスプレイの情報の取得
kmGetDisplaySize	画面サイズの取得
kmGetGunTriggerPos	ガンデバイスのトリガが引かれた位置の取得
kmSetDisplayMode	フレームバッファの表示モード設定
kmSetHSyncLine	表示何ライン目で割り込みを起こすかを指定
kmSetPALEXTCallback	PAL 拡張画面モード設定用コールバック関数の登録
kmWaitVBlank	VBlank 信号の待ち合わせ
kmConvertFogDensity	フォグ係数の変換
kmGenerateFogTable	フォグテーブルの自動生成
kmGetSystemMetrics	Kamui2 内部情報の取得
kmResetRenderer	レンダラの強制リセット
kmSetAutoSortMode	半透明ポリゴンのオートソートモードの ON/OFF
kmSetBackGround	背景表示の指定
kmSetBackGroundPlane	バックグラウンド平面の設定
kmSetBackGroundRenderState	Background Plane のレンダリングパラメータの設定
kmSetBorderColor	ボーダーカラーの設定
kmSetCheapShadowMode	チープシャドウモードの設定
kmSetColorClampMax	カラーのクランプの最大値の指定
kmSetColorClampMin	カラーのクランプの最小値の指定

関数名	機 能
kmSetColorClampValue	カラークランプ値の設定
kmSetCullingRegister	カルリングパラメータの設定
kmSetFogDensity	テーブルフォグの係数指定
kmSetFogTable	フォグテーブルの設定
kmSetFogTableColor	テーブルフォグカラーの指定
kmSetFogVertexColor	頂点フォグカラーの指定
kmSetGlobalClipping	グローバルクリッピングの設定
kmSetPaletteBank	オンチップパレットデータの書き換え(バンク単位)
kmSetPaletteBankData	オンチップパレットデータの書き換え
kmSetPaletteData	オンチップパレットデータの設定
kmSetPaletteMode	オンチップパレットモードの設定
kmSetPixelClipping	ピクセル単位のクリッピングの指定
kmSetPunchThroughThreshold	パンチスルーポリゴンのアルファ閾値の設定
kmSetStrideWidth	ストライドサイズの指定
kmUseAnotherModifier	モデファイアボリュームリストの指定
kmInitDevice	ハードウェアデバイスの初期化
kmUnloadDevice	ハードウェアおよびドライバの終了処理
kmGetRenderStatus	実行したレンダリングの状況を取得
kmQueryFinishLastTextureDMA	前回の DMA 転送によるテクスチャロードの終了チェック
kmRender	レンダリング開始
kmRenderTexture	テクスチャメモリへのレンダリング開始
kmCreateCombinedTextureSurface	複合テクスチャサーフェスの確保
kmCreateContiguousTextureSurface	連続したアドレス領域へのテクスチャサーフェスの作成
kmCreateFixedTextureArea	固定テクスチャ領域の確保
kmCreateTextureSurface	テクスチャ用サーフェスの確保
kmFreeFixedTextureArea	固定テクスチャ領域の解放
kmFreeTexture	テクスチャデータの解放
kmGarbageCollectTexture	テクスチャメモリのガーベジコレクション
kmGetCurrentTextureStatus	グラフィックメモリ管理情報の取得
kmGetFreeTextureMem	テクスチャメモリの空き容量の取得
kmGetTexture	テクスチャメモリ上のテクスチャの読み出し
kmLoadRectangleTexturePart	テクスチャデータの矩形領域転送
kmLoadTexture	テクスチャデータの読み込み
kmLoadTextureBlock	テクスチャデータの分割読み込み
kmLoadTexturePart	テクスチャデータの部分読み込み
kmLoadVQCodebook	VQ テクスチャのコードブック部分の再読み込み
kmLoadYUVTexture	YUV 形式テクスチャデータの読み込み
kmReloadMipmap	特定のミップマップテクスチャの再読み込み
kmuCalculateKamuiWorkareaSize	システムワークエリアのサイズ取得
kmuCheckPassTable	VERTEXCONTEXT のチェック
kmuConvertFBtoBMP	Rectangle 形式から Windows BMP 形式への変換
kmuConvertStripContext	KMSTRIPCONTEXT 構造体への変換
kmuCreateTwiddledTexture	Kamui ビットマップ形式から Twiddled 形式への変換
kmuCreateTwiddledTextureEx	Kamui ビットマップ形式から Twiddled 形式への変換
kmuGeneratePassTable	マルチパス用 VERTEXCONTEXT の自動生成

関数名	機 能
kmGetVersionInfo	ライブラリのバージョン情報を取得
kmChangeStripBlendingMode	BlendingMode の変更
kmChangeStripClampUV	ClampUV の変更
kmChangeStripColorClamp	ColorClamp の変更
kmChangeStripCullingMode	CullingMode の変更
kmChangeStripDCalcControl	DCalcControl の変更
kmChangeStripDepthCompareMode	DepthCompareMode の変更
kmChangeStripDSTSelect	DSTSelect の変更
kmChangeStripFaceColor	FaceColor の変更
kmChangeStripFaceOffsetColor	FaceOffsetColor の変更
kmChangeStripFilterMode	FilterMode の変更
kmChangeStripFlipUV	FlipUV の変更
kmChangeStripFogMode	FogMode の変更
kmChangeStripGouraud	Gouraud の変更
kmChangeStripIgnoreTextureAlpha	IgnoreTextureAlpha の変更
kmChangeStripIntensityMode	IntensityMode の変更
kmChangeStripListType	ListType の変更
kmChangeStripMipmapAdjust	MipmapAdjust の変更
kmChangeStripModifierInstruction	ModifierInstruction の変更
kmChangeStripOffset	Offset の変更
kmChangeStripPaletteBank	PaletteBank の変更
kmChangeStripShadowMode	ShadowMode の変更
kmChangeStripSpriteBaseColor	Sprite の BaseColor の変更
kmChangeStripSpriteOffsetColor	Sprite の OffsetColor の変更
kmChangeStripSRCSelect	SRCSelect の変更
kmChangeStripSuperSampleMode	SuperSampleMode の変更
kmChangeStripTextureAddress	TextureAddress の変更
kmChangeStripTextureShadingMode	TextureShadingMode の変更
kmChangeStripTextureSurface	TextureSurface の変更
kmChangeStripUseAlpha	UseAlpha の変更
kmChangeStripUserClipMode	UserClipMode の変更
kmChangeStripZWriteDisable	ZWriteDisable の変更
kmGenerateStripHead	レンダリングパラメータ (KMSTRIPHEAD) の構築
kmGenerateStripHead00	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType00 用)
kmGenerateStripHead01	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType01 用)
kmGenerateStripHead02	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType02 用)
kmGenerateStripHead03	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType03 用)
kmGenerateStripHead04	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType04 用)
kmGenerateStripHead05	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType05 用)
kmGenerateStripHead06	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType06 用)
kmGenerateStripHead07	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType07 用)

関数名	機 能
kmGenerateStripHead08	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType08 用)
kmGenerateStripHead09	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType09 用)
kmGenerateStripHead10	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType10 用)
kmGenerateStripHead11	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType11 用)
kmGenerateStripHead12	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType12 用)
kmGenerateStripHead13	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType13 用)
kmGenerateStripHead14	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType14 用)
kmGenerateStripHead15	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType15 用)
kmGenerateStripHead16	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType16 用)
kmGenerateStripHead17	レンダリングパラメータ (KMSTRIPHEAD) の構築 (VertexType17 用)
kmInitStripContext	ストリップコンテキストの初期化
kmProcessVertexRenderState	レンダリングパラメータ (KMVERTEXCONTEXT) の構築
kmRegisterStripContext	ストリップコンテキストをシステムに登録 (ユーザー定義)
kmSetModifierRenderState	レンダリングパラメータ (KMVERTEXCONTEXT) の登録 (Param2 使用時)
kmSetStripHead	レンダリングパラメータ (KMSTRIPHEAD) の登録
kmSetVertexRenderState	レンダリングパラメータ (KMVERTEXCONTEXT) の登録 (通常時)
kmChangeContextBlendingMode	BlendingMode の変更
kmChangeContextClampUV	ClampUV の変更
kmChangeContextColorClamp	ColorClamp の変更
kmChangeContextCullingMode	CullingMode の変更
kmChangeContextDCalcControl	DCalcControl の変更
kmChangeContextDepthCompareMode	DepthCompareMode の変更
kmChangeContextDSTSelect	DSTSelect の変更
kmChangeContextFaceColor	FaceColor の変更
kmChangeContextFaceOffsetColor	FaceOffsetColor の変更
kmChangeContextFilterMode	FilterMode の変更
kmChangeContextFlipUV	FlipUV の変更
kmChangeContextFogMode	FogMode の変更
kmChangeContextGouraud	Gouraud の変更
kmChangeContextIgnoreTextureAlpha	IgnoreTextureAlpha の変更
kmChangeContextIntensityMode	IntensityMode の変更
kmChangeContextListType	ListType の変更
kmChangeContextMipmapAdjust	MipmapAdjust の変更
kmChangeContextModifierInstruction	ModifierInstruction の変更
kmChangeContextOffset	Offset の変更

関数名	機 能
kmChangeContextPaletteBank	PaletteBank の変更
kmChangeContextShadowMode	ShadowMode の変更
kmChangeContextSpriteBaseColor	Sprite の BaseColor の変更
kmChangeContextSpriteOffsetColor	Sprite の OffsetColor の変更
kmChangeContextSRCSelect	SRCSelect の変更
kmChangeContextSuperSampleMode	SuperSampleMode の変更
kmChangeContextTextureAddress	TextureAddress の変更
kmChangeContextTextureShadingMode	TextureShadingMode の変更
kmChangeContextTextureSurface	TextureSurface の変更
kmChangeContextUseAlpha	UseAlpha の変更
kmChangeContextUserClipMode	UserClipMode の変更
kmChangeContextZWriteDisable	ZWriteDisable の変更
kmBeginPass	パスの開始
kmBeginScene	シーンの開始
kmContinuePass	パスの継続
kmEndPass	パスの終了
kmEndScene	シーンの終了
kmEndStrip	頂点データストリップの終了
kmEndVertexStrip	頂点データストリップの終了
kmSetUserClipping	ユーザークリッピング領域の設定
kmSetVertex	バッファへの頂点データの登録
kmStartStrip	頂点データストリップの開始（直接転送）
kmStartVertexStrip	頂点データストリップの開始
KMCURRENTLISTSTATE	カレント頂点バッファリスト構造体
KMFBSTATUS	グラフィックメモリ管理情報
KMFLOATCOLOR	FLOAT 型のカラー
KMIMAGECONTROL	イメージコントロール構造体
KMOBJECTCONTROL	オブジェクトコントロール構造体
KMPACKEDARGB	パック形式カラー
KMPALETTEDATA	オンチップパレットデータテーブル
KMPALEXTINFO	PAL 拡張モード設定用構造体
KMPASSINFO	パス情報構造体
KMSTRIPCONTEXT	ストリップコンテキスト
KMSTRIPCONTROL	ストリップコントロール構造体
KMSURFACEDESC	フレームバッファ、テクスチャのサーフェスディスクリプタ
KMSYSTEMCONFIGSTRUCT	システム設定情報構造体
KMSYSTEMMETRICS	システムメトリクス構造体
KMTWOVOLUMESTRIPCONTEXT	ストリップコンテキスト（2 パラメータポリゴン用）
KMVERSIONINFO	Kamui バージョン情報
KMVERTEX0	頂点構造体（タイプ 0）
KMVERTEX1	頂点構造体（タイプ 1）
KMVERTEX2	頂点構造体（タイプ 2）
KMVERTEX3	頂点構造体（タイプ 3）
KMVERTEX4	頂点構造体（タイプ 4）
KMVERTEX5	頂点構造体（タイプ 5）
KMVERTEX6	頂点構造体（タイプ 6）
KMVERTEX7	頂点構造体（タイプ 7）
KMVERTEX8	頂点構造体（タイプ 8）

関数名	機 能
KMVERTEX9	頂点構造体 (タイプ 9)
KMVERTEX10	頂点構造体 (タイプ 10)
KMVERTEX11	頂点構造体 (タイプ 11)
KMVERTEX12	頂点構造体 (タイプ 12)
KMVERTEX13	頂点構造体 (タイプ 13)
KMVERTEX14	頂点構造体 (タイプ 14)
KMVERTEX15	頂点構造体 (タイプ 15)
KMVERTEX16	頂点構造体 (タイプ 16)
KMVERTEX17	頂点構造体 (タイプ 17)
KMVERTEXBUFFDESC	頂点バッファディスクリプタ
KMVERTEXBUFFERPOINTER	頂点バッファポインタ構造体
KMVERTEXCONTEXT	頂点コンテキスト構造体
KMWORKAREASIZE	ワークエリアサイズ構造体
TYPES	Kamui2 の基本の型定義

用語集

Kamui2 ライブラリ編

日本語	英 語	内 容
アルファブレンディング	alpha blending	ポリゴンをブレンドさせて半透明を掛けること。 テクスチャが tga フォーマットであれば、標準で 値を格納できる。 bmp フォーマット使用時には、2 枚の bmp を使用して半透明をかけることができる
インテンシティ	intensity	強度
ダイレクト (モード)	direct mode	2 V レイテンシモードの場合、1 つの頂点バッファに対して頂点情報を登録せずに、直接タイルアクセラレータに DMA 転送すること
頂点バッファ	vertex buffer	頂点登録用のバッファ。ポリゴンの種類により 5 つに分割し使用する
テクスチャサーフェス	texture surface	テクスチャ管理用の情報。テクスチャ 1 枚について 1 つのテクスチャサーフェスを用意し、テクスチャを使用する
同期 (モード)		V-Sync 信号のタイミングで次の処理へ遷移していくモード
非同期 (モード)		V-Sync 信号とは関係せず、前の処理が終わった時点で次の処理へ遷移していくモード
ポリゴンパラメータ	polygon parameter	コントロールパラメータ、グローバルパラメータ、パーテックスパラメータから構成され頂点登録のはじめに登録し、ポリゴンの各種状態を定義する
マルチパス	multi pass	頂点バッファを複数に分けて転送すること。これを使うことにより半透明ポリゴンのソートは 1 つのパス毎に行われるため、ソート時間を短縮することができる。 しかし、1 つのパスでしかソートしないため、ユーザーが各パスの前後関係に注意し使用しなければならない
ミップマップ	mip mapping	3 次元図形処理で、テクスチャをあらかじめ複数用意 (計算) しておき、オブジェクトの大小に応じてそれを貼り替える手法。アンチエイリアシング処理を簡略化できるため、高速な処理が可能
PAL 拡張 (モード)		画面が 640×240, 640×480 の PAL モードの時のみラインを拡大して表示するモードです。1.003, 1.066, 1.100, 1.133, 1.166 倍に拡大が可能
VQ	vector quantization	ベクトル量子化
2 V レイテンシモデル	2v latency model	頂点登録 (描画関数) を実行したフレームから、2 V 後にディスプレイ表示されるモデル。頂点登録と同じフレームで頂点転送を行い、次のフレームでレンダリングが実行される。頂点登録と頂点転送が同じフレームで行われるため、頂点バッファはシングルバッファでよいので、メモリが節約できる
3 V レイテンシモデル	3v latency model	頂点登録 (描画関数) を実行したフレームから、3 V 後にディスプレイ表示されるモデル。頂点登録、頂点転送、レンダリングが 1 V 毎に行われるため 2 V レイテンシモードより、多くのポリゴンを処理することができる。頂点登録と頂点転送が同じフレーム内で実行されないため、頂点バッファはダブルバッファになる