



# Dreamcast™

## 描画バッファの仕組み

SEGA Library 環境においてアプリケーションを設計するとき、SH4 および Holly を用いた描画モデルを理解しておかなければなりません。このドキュメントでは、どのような手順とタイミングで描画が行われているを説明します。

株式会社セガ・エンタープライゼス



## 1 描画に関するデータの流れ

SEGA Library ではモデルを描画する時、バッファを使用して次のような手順で描画します。

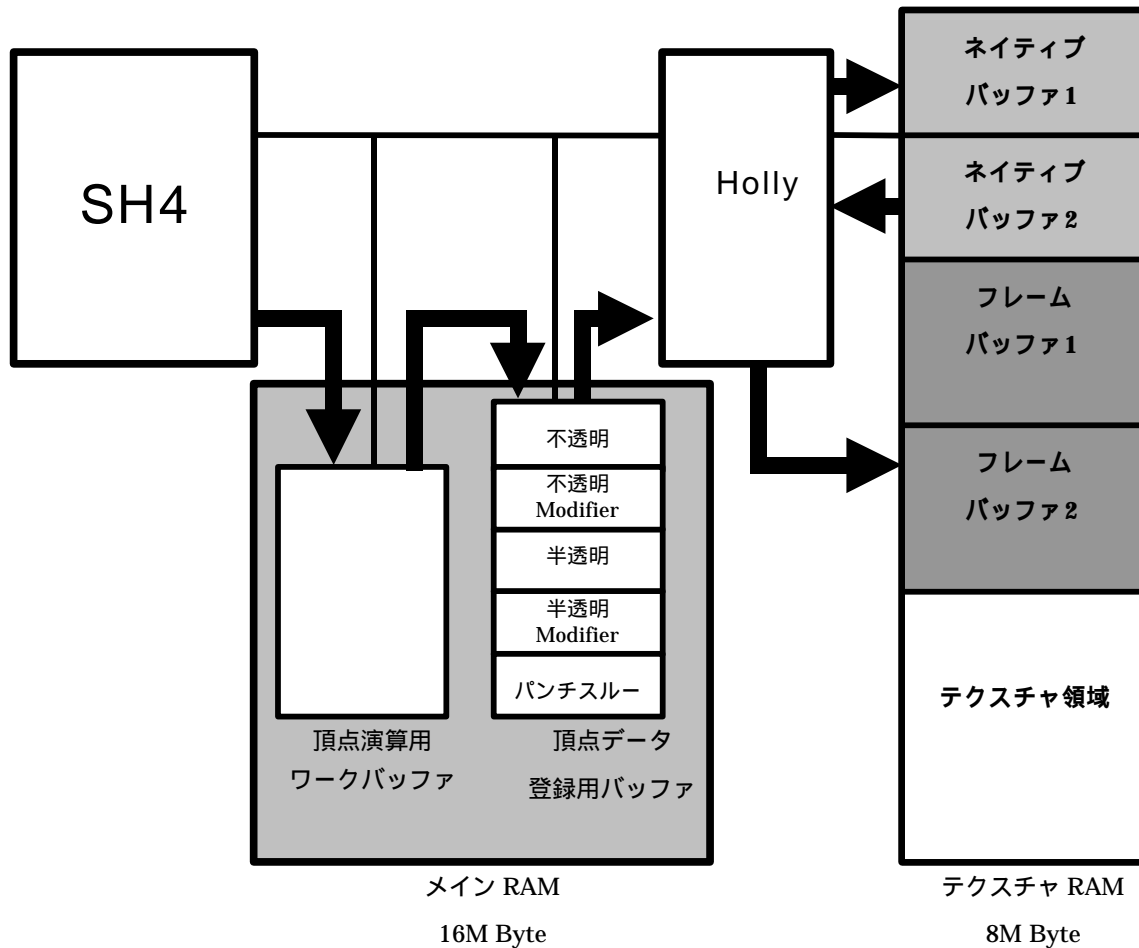


図1.描画モデル

図1 でわかるように 3 段階のステップで画面に描画されます。

頂点登録

DMA 転送

レンダリング

頂点登録の終了は njWaitVSync()関数で区切りとします。

## 2 メイン RAM の使用状況

描画を行うにはメイン RAM 上に「頂点演算用ワークバッファ」「頂点データ登録用バッファ」の 2 つのバッファが必要になります。

### 頂点演算用ワークバッファ

一時的なバッファです。njInit3D()関数でバッファの設定を行います。

### 頂点データ登録用バッファ

頂点データ登録用バッファは次の 5 つのバッファから構成されています。

- ・不透明ポリゴン頂点バッファ
- ・不透明 Modifier 頂点バッファ
- ・半透明ポリゴン頂点バッファ
- ・半透明 Modifier 頂点バッファ
- ・パンチスルー頂点バッファ

パンチスルーの Modifier は不透明 Modifier に登録されます。

バッファのサイズは njInitVertexBuffer()関数で設定を行います。

### 3 テクスチャ RAM

テクスチャ RAM の総容量は 8MByte で

- ・「フレームバッファ」
- ・「テクスチャバッファ」
- ・「ネイティブバッファ」

の 3 つのバッファから構成されています。「フレームバッファ」「ネイティブバッファ」はダブルバッファなのでそれぞれ 2 枚ずつ必要になります。

njCalcTexture()関数を使用することでテクスチャ RAM の残り容量を取得することができます。

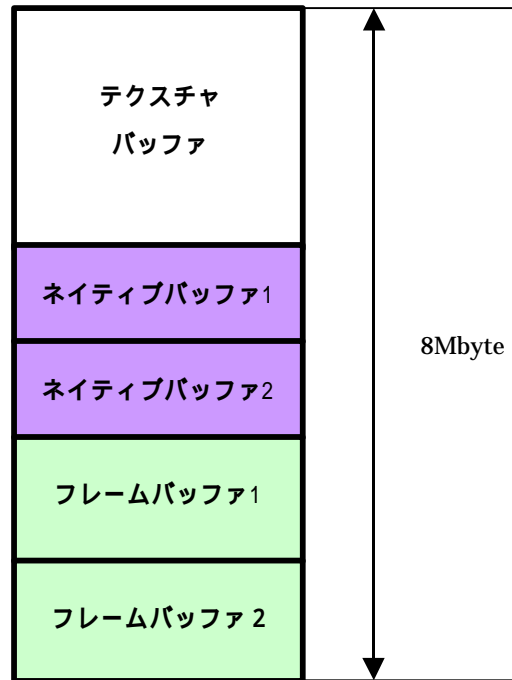


図2. テクスチャ RAM

#### フレームバッファ

フレームバッファがどれだけテクスチャメモリを占有するかは画面モードとフレームバッファモードによって決定されます。

sbInitSystem()関数を実行して画面モードとフレームバッファモードの設定を行うと、以下のサイズをテクスチャ RAM に確保します。

Mode \ Frame	RGB565, RGB555 ARGB1555 (16Bit)	RGB888 (24Bit)	ARGB8888 (32Bit)
VGA	0x12C000	0x1C2000	0x258000
320X240_NTSCNI	0x04B000	0x070800	0x096000
320X240_NTSCI	0x04B000	0x070800	0x096000
640X240_NTSCNI	0x096000	0x0E1000	0x12C000
640X240_NTSCI	0x096000	0x0E1000	0x12C000
320X480_NTSCNI	0x096000	0x0E1000	0x12C000
320X480_NTSCI	0x096000	0x0E1000	0x12C000
640X480_NTSCNI	0x12C000	0x1C2000	0x258000
640X480_NTSCI	0x12C000	0x1C2000	0x258000
640X480_NTSCNI_FF	0x096000	0x0E1000	0x12C000

数値はフレームバッファ 2 枚分の容量です。

表1. 画面モードとフレームバッファモードの設定で定義される容量

## テクスチャバッファ

テクスチャ RAM の中でテクスチャとして使用できる容量は `njSetTextureMemorySize()`関数で設定できます。

## ネイティブバッファ

ネイティブバッファはフレームバッファに描画する為のコマンドを登録しているバッファです。

「ネイティブバッファ」の容量は 8Mbyte から「テクスチャバッファ」と「フレームバッファ」の容量を引いた値が割り振られますが、「ネイティブバッファ」は最大で 4Mbyte 以上確保しません。

## 4 レイテンシー(Latency)

SEGA Library 環境では描画方法として「3V Latency」と「2V Latency」の2つの描画モードをサポートしています。これらの設定は `njInitVertexBuffer()` 関数で行います。

### 3V Latency

3V Latency は「頂点登録」「DMA転送」「レンダリング」「表示」という順番で画面の描画を行うシンプルな手法です。

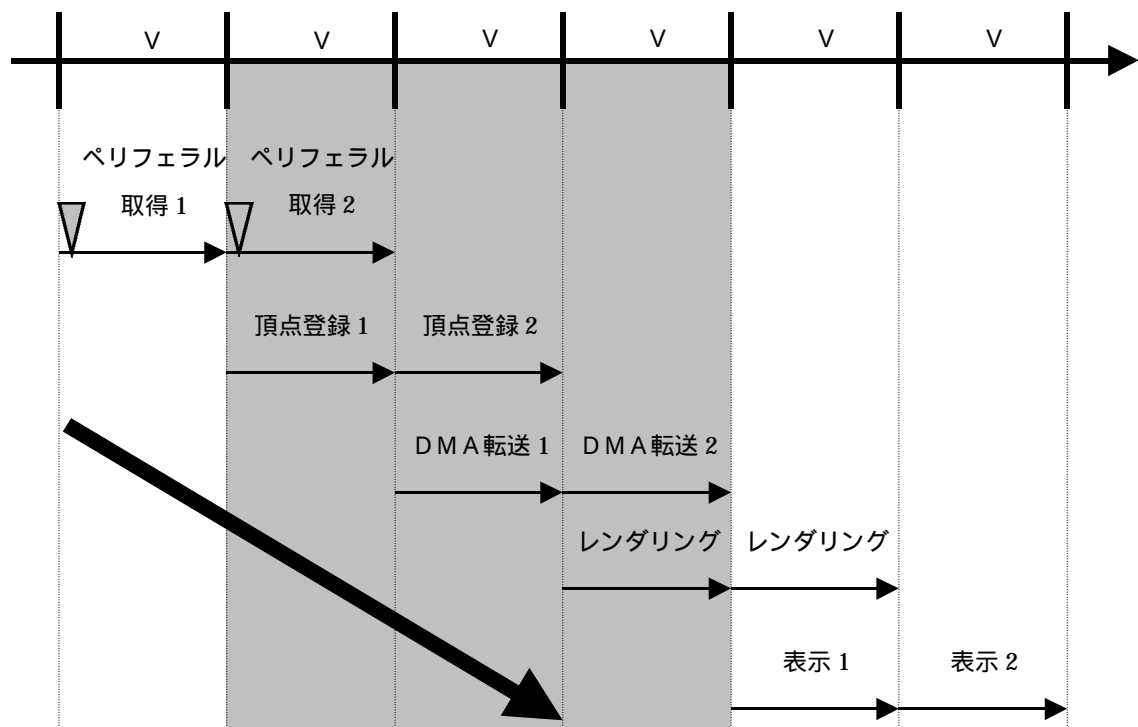


図3. 3V Latency パイプライン

このような方法をとるため、頂点データ登録用にダブルバッファが必要になってしまいます。

3V Latency は図4のようになります。

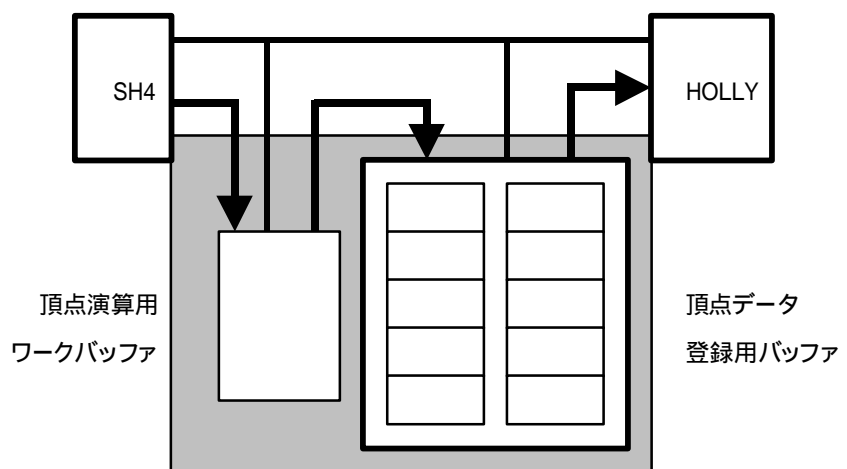


図4. 3V Latency モデル

## 2V Latency

2V Latency は「頂点登録+DMA転送」「レンダリング」「表示」という順番で画面の表示を行います。

全てのデータを「頂点データ登録用バッファ」に貯めてから DMA 転送を行う 3V Latency とは異なり、5 つのバッファのうちユーザーが指定した使用頻度の一番高い 1 つは、データを貯めるのではなく随時 Holly に転送します。それ以外のバッファは頂点登録が終了次第転送します。

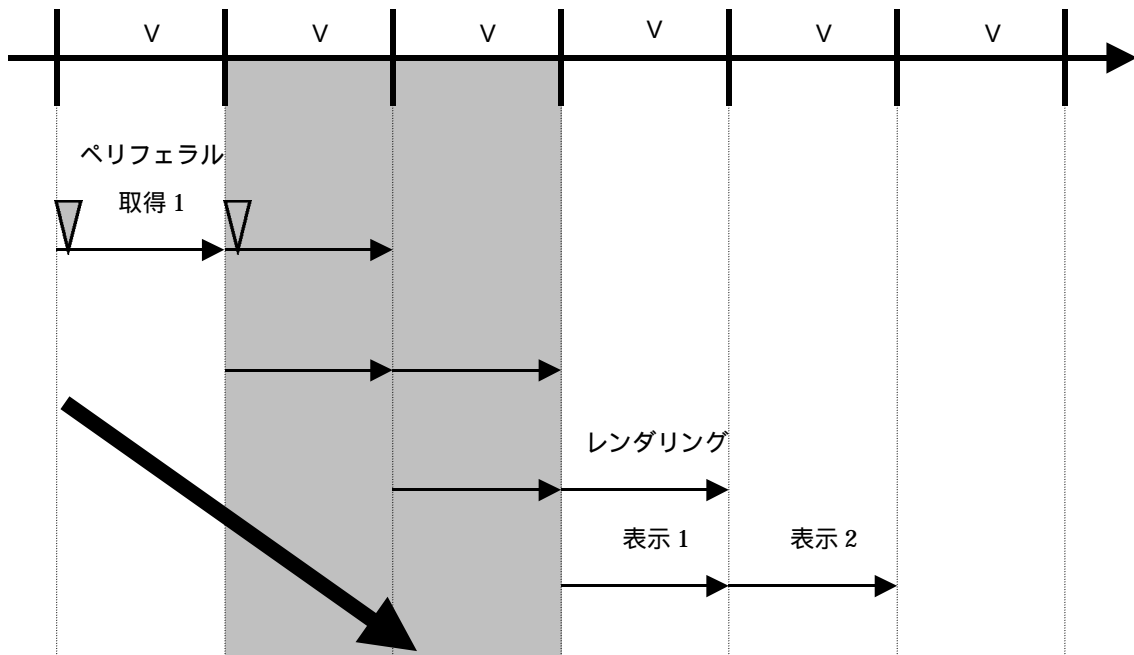


図5. 2V Latency パイプライン

2V Latency は 3V Latency に比べ頂点データ登録用バッファがシングルバッファであるため、必要とする容量は半分以下ですむだけでなく、ペリフェラルからの信号の対応を速くすることができます。

一般的には不透明バッファの使用頻度が高くなるので不透明バッファをスルーさせるようユーザーが指定する場合があります。

2V Latency は図6のようになります。

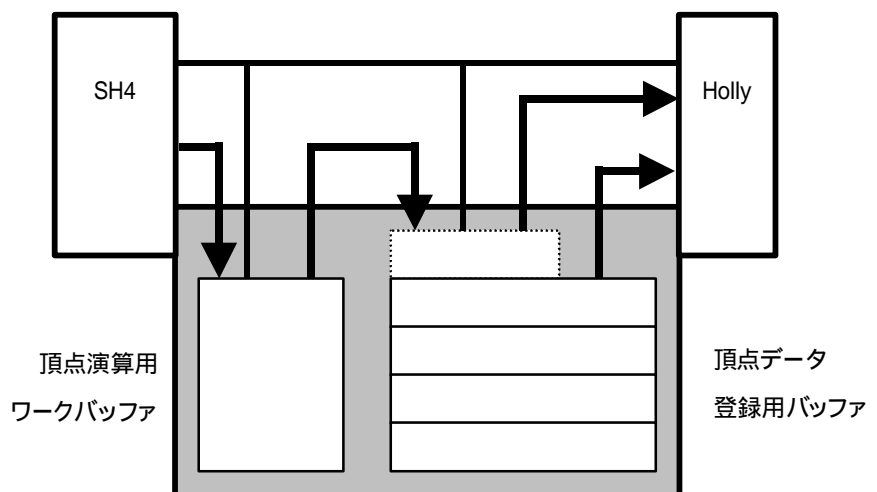


図6. 2V Latency モデル