

VRAM マネージャ

VRAM からの動的なメモリの確保と解放

2008-04-08

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、厳重な取り扱い、管理を行ってください。

目次

1	はじめに	5
2	VRAMマネージャの概要	5
2.1	VRAMマネージャの共通関数	5
2.2	テクスチャVRAMマネージャとパレットVRAMマネージャ	6
2.2.1	テクスチャVRAMマネージャでのテクスチャ用メモリの確保と解放	6
2.2.1.1	4×4テクセル圧縮テクスチャ用メモリの確保	6
2.2.2	NNSGfdTexKey	6
2.2.2.1	NNSGfdTexKeyへの操作	6
2.2.3	パレットVRAMマネージャでのパレット用メモリの確保と解放	7
2.2.3.1	4色パレット用メモリの確保	7
2.2.4	NNSGfdPlttKey	7
2.2.4.1	NNSGfdPlttKeyへの操作	7
2.3	NITRO-Systemで提供されるVRAMマネージャ	7
3	フレームVRAMマネージャ	8
3.1	フレームテクスチャVRAMマネージャ	8
3.1.1	フレームテクスチャVRAMマネージャの初期化	8
3.1.2	テクスチャ用メモリの確保	9
3.1.2.1	標準テクスチャ用メモリの確保	9
3.1.2.2	4×4テクセル圧縮テクスチャの確保要求時の動作	9
3.1.3	テクスチャ用メモリの解放	10
3.1.4	フレームテクスチャVRAMマネージャの状態の保存と復帰	11
3.1.4.1	テクスチャ用メモリの使用状況の保存	11
3.1.4.2	テクスチャ用メモリの使用状況の復帰	11
3.1.4.3	テクスチャ用メモリの使用状況を初期状態に戻す	11
3.2	フレームパレットVRAMマネージャ	11
3.2.1	フレームパレットVRAMマネージャの初期化	12
3.2.2	パレット用メモリの確保	12
3.2.2.1	パレット用メモリの確保方向	12
3.2.3	パレット用メモリの解放	13
3.2.4	フレームパレットVRAMマネージャの状態の保存と復帰	13
3.2.4.1	パレット用メモリの使用状況の保存	13
3.2.4.2	パレット用メモリの使用状況の復帰	13
3.2.4.3	パレット用メモリの使用状況を初期状態に戻す	13
4	リンクドリストVRAMマネージャ	14
4.1	リンクドリスト テクスチャVRAMマネージャ	14
4.1.1	マネージャの初期化	14
4.1.2	テクスチャ用メモリの確保	15
4.1.3	テクスチャ用メモリの開放	16
4.2	リンクドリスト パレットVRAMマネージャ	17
4.2.1	初期化	17

4.2.2	パレット用メモリの確保、開放	17
-------	----------------------	----

表

表 2-1	VRAMマネージャの共通関数.....	5
表 2-2	共通関数で参照される関数ポインタ.....	5

図

図 3-1	フレームテクスチャVRAMマネージャの概念図	8
図 3-2	標準テクスチャ用メモリの確保.....	9
図 3-3	4×4テクセル圧縮テクスチャ用メモリの確保	10
図 3-4	リージョン3からの4×4テクセル圧縮テクスチャ用メモリの確保	10
図 3-5	フレームパレットVRAMマネージャの概念図	11
図 3-6	パレット用メモリの確保.....	12
図 4-1	リンクドリストVRAMマネージャ	14
図 4-2	テクスチャ用メモリの確保.....	15
図 4-3	テクスチャ用メモリの開放.....	16
図 4-4	空きリスト結合処理に失敗する例	16

改訂履歴

改訂日	改訂内容
2008-04-08	<ul style="list-style-type: none">・改訂履歴の書式を変更。・ドキュメントの副題の記載・改ページ位置の調整。
2005-01-05	NITRO という表記をニンテンドーDS に統一。
2004-10-03	リンクドリスト VRAM マネージャの追加。
2004-07-16	初版。

1 はじめに

ニンテンドーDS の3Dグラフィックスエンジンを用いてテクスチャが貼られたポリゴンを描画するためには、VRAM にテクスチャを格納する必要があります。このとき、プログラマが VRAM へのテクスチャの配置を細かく管理する必要がないように、NITRO-System では VRAM マネージャを用意しています。VRAM マネージャは、VRAM からの動的なメモリの確保と解放を行います。

2 VRAMマネージャの概要

2.1 VRAMマネージャの共通関数

VRAM の中を効率良く扱う方法は、アプリケーションによって大きく異なることが考えられます。これにより、アプリケーションによっては独自に VRAM マネージャを用意されることも考えられたため、NITRO-System のグラフィックスライブラリが利用する VRAM マネージャを取り替える事ができる仕組みを用意しています。

NITRO-System では使用する VRAM マネージャの種類に関わらず、同じ関数を使ってメモリの確保と解放ができるように、以下の表に示す4つの関数を定義しています。

表 2-1 VRAM マネージャの共通関数

関数名	関数の処理
NNS_GfdAllocTexVram()	VRAM からテクスチャ用メモリを確保します
NNS_GfdFreeTexVram()	VRAM から確保したテクスチャ用メモリを解放します
NNS_GfdAllocPlttVram()	パレット RAM からパレット用メモリを確保します
NNS_GfdFreePlttVram()	パレット RAM から確保したパレット用メモリを解放します

これらの関数の内部は、関数ポインタに登録されている実際の処理を行う関数を呼び出すようになっています。デフォルトでは、何も処理を行わずにエラーを返す関数が登録されています。これらの関数が内部で参照している関数ポインタは、以下のような大域変数として定義されています。

表 2-2 共通関数で参照される関数ポインタ

ポインタを参照する関数	関数ポインタの型名	関数ポインタの変数名
NNS_GfdAllocTexVram()	NNSGfdFuncAllocTexVram	NNS_GfdDefaultFuncAllocTexVram
NNS_GfdFreeTexVram()	NNSGfdFuncFreeTexVram	NNS_GfdDefaultFuncFreeTexVram
NNS_GfdAllocPlttVram()	NNSGfdFuncAllocPlttVram	NNS_GfdDefaultFuncAllocPlttVram
NNS_GfdFreePlttVram()	NNSGfdFuncFreePlttVram	NNS_GfdDefaultFuncFreePlttVram

VRAM マネージャを利用する NITRO-System のグラフィックスライブラリは、これらの4つの関数を用いて、VRAM からのメモリの確保と解放を行います。ユーザが使用したい VRAM マネージャが持つメモリ確保と解放を行うための関数を上記の関数ポインタに登録することにより、ライブラリが使用する VRAM マネージャを変更する事ができます。

2.2 テクスチャVRAMマネージャとパレットVRAMマネージャ

VRAM マネージャは、大きく2つに分けられます。1つはテクスチャ用メモリの確保と解放を行うテクスチャ VRAM マネージャであり、もう一つはパレット用メモリの確保と解放を行うパレット VRAM マネージャです。

2.2.1 テクスチャVRAMマネージャでのテクスチャ用メモリの確保と解放

テクスチャ VRAM マネージャでテクスチャ用メモリの確保と解放を行う場合は、以下に示す関数を使用します。

```
// テクスチャVRAMからのメモリ確保。
NNSGfdTexKey NNS_GfdAllocTexVram(u32 szByte, BOOL is4x4comp, u32 opt);

// テクスチャVRAMから確保したメモリの解放。
int NNS_GfdFreeTexVram(NNSGfdTexKey key);
```

NNS_GfdAllocTexVram()関数は、szSize で指定された大きさのテクスチャ用メモリを VRAM から確保し、その確保したテクスチャ用メモリを示す NNSGfdTexKey 型の値を返します。もしテクスチャ用メモリの確保に失敗した場合には、定数 NNS_GFD_ALLOC_ERROR_TEXKEY を返します。

NNS_GfdFreeTexVram()関数は、NNSGfdTexKey 型の値で指定されるテクスチャ用メモリを解放します。メモリの解放に成功した場合は、0を返します。

2.2.1.1 4×4テクセル圧縮テクスチャ用メモリの確保

テクスチャ VRAM マネージャでは、4×4テクセル圧縮テクスチャ用の領域を確保できる機能を持っています。テクスチャ用メモリ確保関数である NNS_GfdAllocTexVram()関数の2番目の引数 is4x4comp が TRUE の場合は、テクスチャイメージデータ用の領域の確保と同時に、テクスチャパレットインデックスデータ用の領域も確保します。この2つの領域の片方が確保できなかった場合には、NNS_GfdAllocTexVram()関数の呼び出しは失敗します。

2.2.2 NNSGfdTexKey

NNSGfdTexKey は、テクスチャ VRAM マネージャから確保されたテクスチャ用メモリを識別するためのキーであり、32 ビット長の整数値となっています。NNSGfdTexKey は、テクスチャ用メモリのアドレスとサイズ、及び4×4テクセル圧縮テクスチャ用メモリを示すフラグから生成されます。

2.2.2.1 NNSGfdTexKey への操作

確保したテクスチャ用メモリにテクスチャデータを転送する為には、実際の VRAM のアドレスを取得する必要があります。テクスチャ VRAM マネージャでは、NNSGfdTexKey から VRAM のアドレスやサイズを求める為の関数が提供されています。

```
// NNSGfdTexKeyからアドレスを得ます。
u32 NNS_GfdGetTexKeyAddr( NNSGfdTexKey memKey );

// NNSGfdTexKeyからサイズを得ます。
u32 NNS_GfdGetTexKeySize( NNSGfdTexKey memKey );

// NNSGfdTexKeyが圧縮テクスチャ用かどうかを調べます。
BOOL NNS_GfdGetTexKey4x4Flag( NNSGfdTexKey memKey );
```

2.2.3 パレットVRAMマネージャでのパレット用メモリの確保と解放

パレット VRAM マネージャでパレット用メモリの確保と解放を行う場合は、以下に示す関数を使用します。

```
// パレットRAMからのメモリ確保。
NNSGfdPlttKey NNS_GfdAllocPlttVram(u32 szByte, BOOL is4pltt, u32 opt);

// パレットRAMから確保したメモリの解放。
int NNS_GfdFreePlttVram(NNSGfdPlttKey key);
```

NNS_GfdAllocPlttVram は、szSize で指定された大きさのパレット用メモリをパレット RAM から確保し、その確保したパレット用メモリを示す NNSGfdPlttKey 型の値を返します。もしパレット用メモリの確保に失敗した場合には、定数 NNS_GFD_ALLOC_ERROR_PLTTKEY を返します。

NNS_GfdFreePlttVram は、NNSGfdPlttKey で指定されるパレット用メモリを解放します。メモリの解放に成功した場合は、0を返します。

2.2.3.1 4色パレット用メモリの確保

パレット VRAM マネージャを用いて4色カラーパレット用メモリを確保する場合には、パレット用メモリ確保関数である NNS_GfdAllocPlttVram関数の2番目の引数 is4pltt に TRUE を指定します。この場合、4色パレットを置く事が可能な位置にメモリを確保できたかどうかの判定が行われます。(4色パレットは、0x10000 以降に置く事が出来ません。)

2.2.4 NNSGfdPlttKey

NNSGfdPlttKey は、パレット VRAM マネージャから確保されたパレット用メモリを識別するためのキーであり、32 ビット長の整数値となっています。NNSGfdPlttKey は、確保したパレット用メモリのアドレスとサイズから生成されます。

2.2.4.1 NNSGfdPlttKey への操作

確保したパレット用メモリにパレットデータを転送する為には、実際のパレット RAM のアドレスを取得する必要があります。パレット VRAM マネージャでは、NNSGfdPlttKey からパレット RAM のアドレスやサイズを求める為の関数が提供されています。

```
// NNSGfdPlttKeyからアドレスを得ます。
u32 NNS_GfdGetPlttKeyAddr( NNSGfdPlttKey memKey )

// NNSGfdTexKeyからサイズを得ます。
u32 NNS_GfdGetPlttKeySize( NNSGfdPlttKey memKey )
```

2.3 NITRO-Systemで提供されるVRAMマネージャ

2.1で説明しました通り、NITRO-SystemのVRAMマネージャで提供されるメモリ確保と解放の関数は、初期状態では何も処理を行わないようになっています。NITRO-Systemでは、このメモリ確保と解放の関数に処理を与える幾つかのメモリ管理を持ったVRAMマネージャを提供する予定です。現時点ではフレームVRAMマネージャ(フレームテキストチャVRAMマネージャとフレームパレットVRAMマネージャ)、リンクドリストVRAMマネージャ(リンクドリストテキストチャVRAMマネージャとリンクドリストパレットVRAMマネージャ)のみの提供となります。

3 フレームVRAMマネージャ

フレーム VRAM マネージャは、Foundation ライブラリのフレームヒープマネージャの様に、指定された大きさのメモリブロックを確保することと、確保されたメモリブロックの全てを同時に解放することしかできません。その代わりに、メモリブロックに管理情報を一切持たないため、メモリ効率が良いものとなっています。以下にフレーム VRAM マネージャの特徴を示します。

- メモリ管理領域が不要です。
- 各 VRAM スロットの前後から、間を空けずにメモリブロックを確保します。
- 確保したメモリブロックを個別に解放することは出来ません。
- 確保されている全メモリブロックの一括解放が可能です。
- メモリブロックの確保状態の保存と復帰が可能です。

3.1 フレームテクスチャVRAMマネージャ

図 3-1 にフレームテクスチャVRAMマネージャの概念図を示します。フレームテクスチャVRAMマネージャは、VRAMを5つのリージョン(region)に分割して管理します。それぞれのリージョンには、上位側、下位側の2つのポインタが用意されています。これらの2つのポインタが、使用領域と未使用領域の境界を指し示しており、2つのポインタに挟まれた部分が未使用領域となります。テクスチャ用メモリは、リージョンの先頭と後尾の両方向から確保され、確保される度に2つのポインタがリージョンの中央に向かって移動していきます。

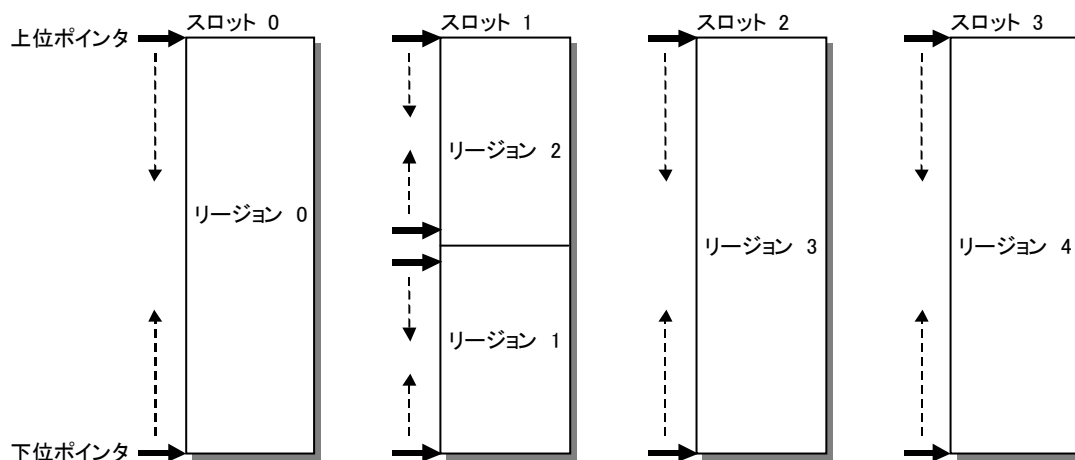


図 3-1 フレームテクスチャ VRAM マネージャの概念図

3.1.1 フレームテクスチャVRAMマネージャの初期化

フレームテクスチャ VRAM マネージャを使用する前には、フレームテクスチャ VRAM マネージャを初期化する必要があります。フレームテクスチャ VRAM マネージャの初期化には、以下の関数を用います。

```
void NNS_GfdInitFrmTexVramManager(ul6 numSlot, BOOL useAsDefault);
```

1番目の引数 numSlot は、フレームテクスチャ VRAM マネージャが管理するスロットの数を指定します。フレームテクスチャ VRAM マネージャは、VRAM スロット0から numSlot で指定された個数分の VRAM スロットを管理するように初期化されます。numSlot に指定できる最大値は、4となります。

2番目の引数 `useAsDefault` に `TRUE` が指定されると、テクスチャ VRAM マネージャの共通関数である `NNS_GfdAllocTexVram()`、または `NNS_GfdFreeTexVram()` が呼び出された時に、フレームテクスチャ VRAM マネージャのメモリ確保と解放の関数が使用されるように、関数ポインタを初期化します。後でテクスチャ VRAM マネージャの処理を入れ替えて使用するなどの特別な場合を除き、通常は `TRUE` を指定するようにします。

3.1.2 テクスチャ用メモリの確保

テクスチャ用メモリを確保する場合は、通常、テクスチャ VRAM マネージャの共通関数の `NNS_GfdAllocTexVram()` を使用します。

```
NNSGfdTexKey NNS_GfdAllocTexVram(u32 szByte, BOOL is4x4comp, u32 opt);
```

なお、`NNS_GfdAllocTexVram()` 関数の3番目の引数 `opt` は、フレームテクスチャ VRAM マネージャでは使用されていません。

以降では、マネージャの具体的な動作を解説していきます。解説の例は初期化時のパラメータ `numSlot` (マネージャが管理するスロットの数)として4を与えた場合として参照ください。(なお、マネージャは `numSlot` の値によって空き領域検索順位を若干変化させます。)

3.1.2.1 標準テクスチャ用メモリの確保

標準テクスチャ (4×4テクセル圧縮テクスチャ以外のテクスチャ) 用のメモリを確保する場合は、2番目の引数である `is4x4comp` に `FALSE` を指定します。

標準テクスチャ用メモリを確保する場合には、各リージョンの上位ポインタを使用してメモリを確保します。フレームテクスチャ VRAM マネージャは、始めにリージョン4からのメモリ確保を試み、もしリージョン4の空き容量が少なくメモリが確保できなければ、リージョン3、リージョン0、リージョン2、リージョン1の順に次々とメモリの確保を試みていきます。

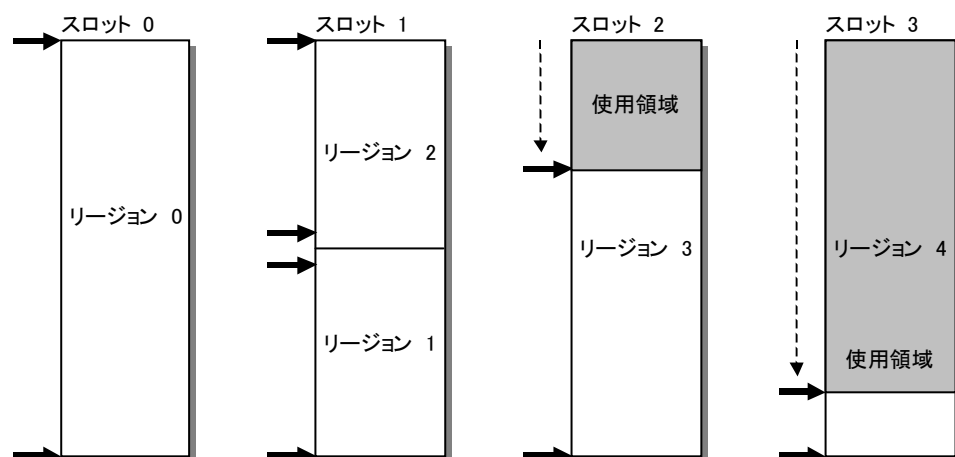


図 3-2 標準テクスチャ用メモリの確保

3.1.2.2 4×4テクセル圧縮テクスチャの確保要求時の動作

4×4テクセル圧縮テクスチャ用のメモリを確保する場合は、2番目の引数である `is4x4comp` に `TRUE` を指定します。

4×4テクセル圧縮テクスチャ用メモリを確保する場合は、まず始めにリージョン0の下位ポインタを使用してメモリの確保を試みます。4×4テクセル圧縮テクスチャを VRAM に格納する場合には、同時にテクスチャパレットインデックスデータもテクスチャイメージデータが置かれているアドレスに対応した位置に格納する必要があります。その為、リージョン

1の下位ポインタを使用して、テクスチャパレットインデックスデータ用の領域も同時に確保します。

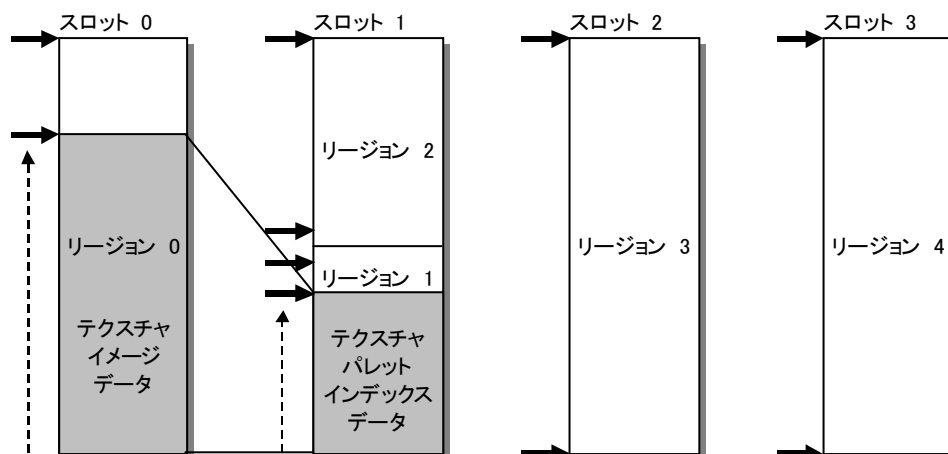


図 3-3 4×4テクセル圧縮テクスチャ用メモリの確保

もしリージョン0から要求された大きさの4×4テクセル圧縮テクスチャ用メモリの確保ができなかった場合には、リージョン3の下位ポインタを使用してメモリの確保を試みます。同時に、リージョン2の下位ポインタを使用してテクスチャパレットインデックスデータ用の領域も確保します。

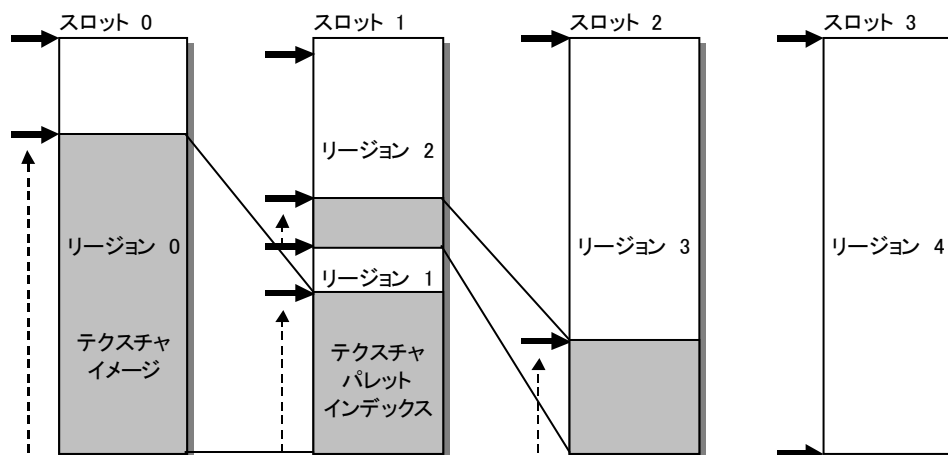


図 3-4 リージョン3からの4×4テクセル圧縮テクスチャ用メモリの確保

3.1.3 テクスチャ用メモリの解放

フレームテクスチャ VRAM マネージャでは、そのメモリ管理のアルゴリズムの特性により、確保したテクスチャ用メモリを個別に開放する事が出来ません。

フレームテクスチャ VRAM マネージャには、メモリ解放用関数として、NNS_GfdFreeFrmTexVram0関数が用意されていますが、この関数は何も処理をせずに戻ります。この関数は、テクスチャ VRAM マネージャの共通関数であるNNS_GfdFreeTexVram0の関数ポインタに登録することを目的として準備されています。

3.1.4 フレームテクスチャVRAMマネージャの状態の保存と復帰

フレームテクスチャ VRAM マネージャは、テクスチャ用メモリの使用状況を保存する機能と、テクスチャ用メモリの使用状況を保存されている状態に戻す機能を持っています。

3.1.4.1 テクスチャ用メモリの使用状況の保存

テクスチャ用メモリの使用状況を保存する為には、下記の関数を使用します。

```
NNS_GfdGetFrmTexVramState(NNSGfdFrmTexVramState* pState);
```

NNS_GfdGetFrmTexVramState()関数が呼ばれると、引数で指定された NNSGfdFrmTexVramState 構造体に現在のテクスチャメモリの使用状態を書き込みます。

3.1.4.2 テクスチャ用メモリの使用状況の復帰

フレームテクスチャ VRAM マネージャを、NNS_GfdGetFrmTexVramState()関数を使って保存したテクスチャ用メモリ確保状態に戻す為には、以下の関数を使用します。

```
void NNS_GfdSetFrmTexVramState(const NNSGfdFrmTexVramState* pState);
```

この操作は、pState で指定されるテクスチャ用メモリ確保状態が保存された時より後で確保したテクスチャ用メモリを解放する事になります。

3.1.4.3 テクスチャ用メモリの使用状況を初期状態に戻す

フレームテクスチャ VRAM マネージャを初期状態に戻す為には、下記の関数を使用します。

```
void NNS_GfdResetFrmTexVramState(void);
```

この操作は、フレームテクスチャ VRAM マネージャから確保したテクスチャ用メモリの全てを解放する事になります。

3.2 フレームパレットVRAMマネージャ

図 3-5 にフレームパレットVRAMマネージャの概念図を示します。フレームパレットVRAMマネージャは、パレットRAMの先頭および後尾からパレット用メモリを確保することができます。その為に、上位側、下位側の2つのポイントが用意されています。これら2つのポイントが使用領域と未使用領域の境界を指し示しており、2つのポイントに挟まれた部分が未使用領域となります。

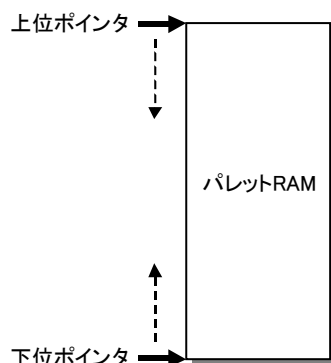


図 3-5 フレームパレット VRAM マネージャの概念図

3.2.1 フレームパレットVRAMマネージャの初期化

フレームパレット VRAM マネージャを使用する前には、フレームパレット VRAM マネージャを初期化する必要があります。フレームパレット VRAM マネージャの初期化には、以下の関数を用います。

```
void NNS_GfdInitFrmPlttVramManager(u32 szByte, BOOL useAsDefault);
```

1番目の引数 `szSize` には、フレームパレット VRAM マネージャが管理するパレット RAM の大きさを指定します。フレームパレット VRAM マネージャは、パレット RAM の先頭から `szSize` で指定されたバイト数のパレット RAM を管理するように初期化されます。

2番目の引数 `useAsDefault` に `TRUE` が指定されると、パレット VRAM マネージャの共通関数である `NNS_GfdAllocPlttVram()`、または `NNS_GfdFreePlttVram()` が呼び出された時に、フレームパレット VRAM マネージャのメモリ確保と解放の関数を使用されるように、関数ポインタを初期化します。後でパレット VRAM マネージャの処理を入れ替えて使用するなどの特別な場合を除き、通常は `TRUE` を指定するようにします。

3.2.2 パレット用メモリの確保

パレット用メモリを確保する場合は、通常、パレット VRAM マネージャの共通関数の `NNS_GfdAllocPlttVram()` を使用します。

```
NNSGfdPlttKey NNS_GfdAllocPlttVram(u32 szByte, BOOL is4Pltt, u32 opt);
```

3.2.2.1 パレット用メモリの確保方向

フレームパレット VRAM マネージャでは、3番目の引数 `opt` にパレット用メモリを確保する方向を指定する事ができます。`opt` に `NNS_GFD_ALLOC_FROM_LOW` が指定されている場合、パレット用メモリはパレット RAM の下位から確保されます。`opt` に `NNS_GFD_ALLOC_FROM_HIGH` が指定されている場合、パレット用メモリはパレット RAM の上位から確保されます。

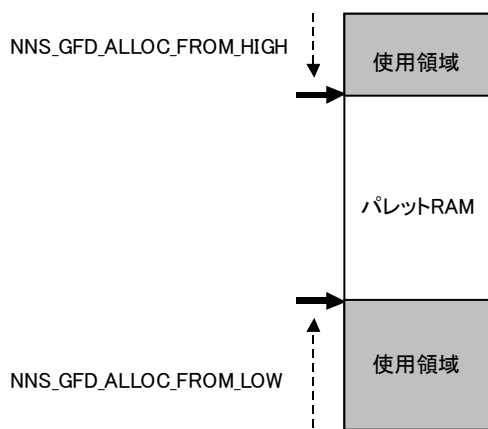


図 3-6 パレット用メモリの確保

3.2.3 パレット用メモリの解放

フレームパレット VRAM マネージャでは、そのメモリ管理のアルゴリズムの特性により、確保したパレット用メモリを個別に開放する事が出来ません。

フレームパレット VRAM マネージャには、メモリ解放用関数として、`NNS_GfdFreeFrmPlttVram()`関数が用意されていますが、この関数は何も処理をせずに戻ります。テクスチャ VRAM マネージャの共通関数である `NNS_GfdFreePlttVram()`の関数ポインタに登録することを目的として準備されています。

3.2.4 フレームパレットVRAMマネージャの状態の保存と復帰

フレームパレット VRAM マネージャでは、パレット用メモリの使用状況を保存する機能と、パレット用メモリの使用状況を保存されている状態に戻す機能を持っています。

3.2.4.1 パレット用メモリの使用状況の保存

パレット用メモリの使用状況を保存する為には、下記の関数を使用します。

```
NNS_GfdGetFrmPlttVramState(NNSGfdFrmPlttVramState* pState);
```

`NNS_GfdGetFrmPlttVramState()`関数を呼ぶと、引数で指定された `NNSGfdFrmPlttVramState` 構造体に現在のパレット用メモリの使用状態を書き込みます。

3.2.4.2 パレット用メモリの使用状況の復帰

フレームパレット VRAM マネージャを、`NNS_GfdGetFrmPlttVramState()`関数を使って保存したパレット用メモリ確保状態に戻す為には、以下の関数を使用します。

```
void NNS_GfdSetFrmPlttVramState(const NNSGfdFrmPlttVramState* pState);
```

この操作は、`pState` で指定されるパレット用メモリ確保状態が保存された時より後で確保したパレット用メモリを解放する事になります。

3.2.4.3 パレット用メモリの使用状況を初期状態に戻す

フレームパレット VRAM マネージャを初期状態に戻す為には、下記の関数を使用します。

```
void NNS_GfdResetFrmPlttVramState(void);
```

この操作は、フレームパレット VRAM マネージャから確保したパレット用メモリの全てを解放する事になります。

4 リンクドリストVRAMマネージャ

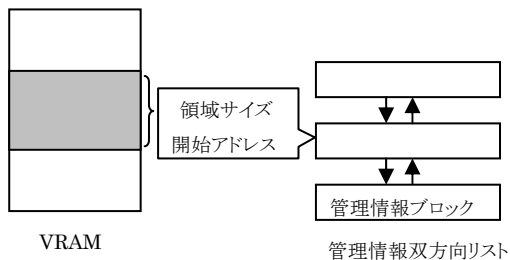


図 4-1 リンクドリスト VRAM マネージャ

リンクドリスト VRAM マネージャは、VRAM 領域をメインメモリ上の VRAM 管理情報を利用して管理します。管理情報は管理対象 VRAM 領域の開始アドレスと、バイトサイズ情報を持ちます。

また、管理情報は次の管理情報への参照、前の管理情報への参照情報を持ち、双方向リスト構造をとるようになっています。VRAM マネージャは管理情報の双方向リストによって、VRAM 中に点在する空き領域を管理しています。

リンクドリスト VRAM マネージャの特徴を以下に箇条書きにします。

- VRAM 領域の部分的な開放が可能です。
- 管理情報をメインメモリ上に確保する必要があります。
- 確保、開放の際に空き領域リストを検索するための処理負荷があります。

4.1 リンクドリスト テクスチャVRAMマネージャ

テクスチャ VRAM マネージャは、通常のテクスチャ用の管理領域と、4x4 圧縮テクスチャ用の管理領域を別々に管理しています。領域の確保請求があると、空き領域情報リストから要求を満たす空き領域を検索します。

4.1.1 マネージャの初期化

```
void NNS_GfdInitLnkTexVramManager
(
    u32      szByte,
    u32      szByteFor4x4,
    void*    pManagementWork,
    u32      szByteManagementWork,
    BOOL     useAsDefault
)
```

初期化の際には、テクスチャ管理領域サイズと、そのうちの 4x4 圧縮テクスチャに使用する領域サイズを別々に指定します。テクスチャ管理領域サイズ > 4x4 圧縮テクスチャに使用する領域サイズ を満たす必要があります。マネージャは指定されたサイズでフリーブロックを初期化します。管理情報として使用するメモリ領域を pManagementWork 引数として渡します。管理情報領域のサイズを算出するには、

```
u32 NNS_GfdGetLnkTexVramManagerWorkSize( u32 numMemBlk )
```

を使用します。ここで指定する numMemBlk が、空きメモリ領域が細分化可能な最大数となります。

テクスチャ VRAM マネージャは、4x4 圧縮テクスチャのパレットインデックス用領域の管理を行いません。よって、パレットインデックス用領域はつねに未使用となり、ユーザは必ず使用可能であると想定できます。また、パレットインデックス用領域以外の領域は通常テクスチャ用空き領域として初期化します。

4.1.2 テクスチャ用メモリの確保

マネージャは新規確保請求があると、確保すべき領域の種類(通常 Or 4x4 圧縮)に応じて空き領域リストを検索します。条件を満たす空き領域が発見された場合は、空き領域情報を使用した領域部分を差し引いたものに更新し、確保した領域をテクスチャキーとして返します。

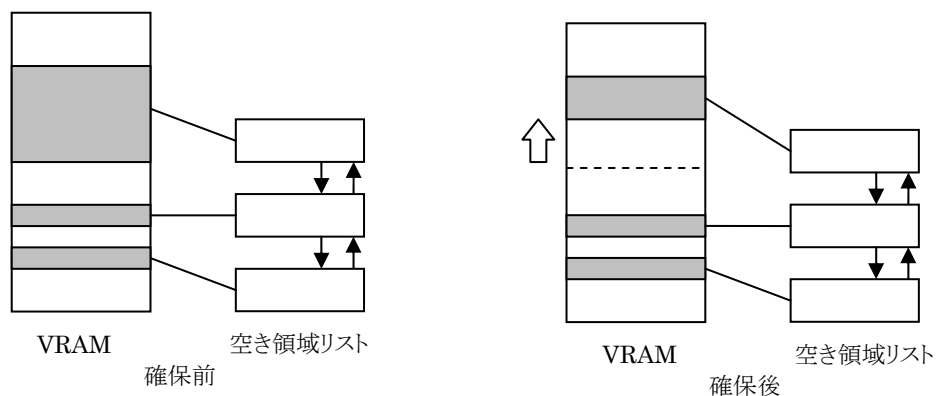


図 4-2 テクスチャ用メモリの確保

4.1.3 テクスチャ用メモリの開放

開放すべきテクスチャキーからメモリ領域を計算します。計算された領域について、領域の上位側、下位側に隣接する空き領域が存在しないか空き領域リストを線形検索します。隣接領域ブロックが発見された場合は、接する領域ブロックを結合しひとつの領域ブロックとします。この処理によってメモリの細分化を起こりにくくしています。

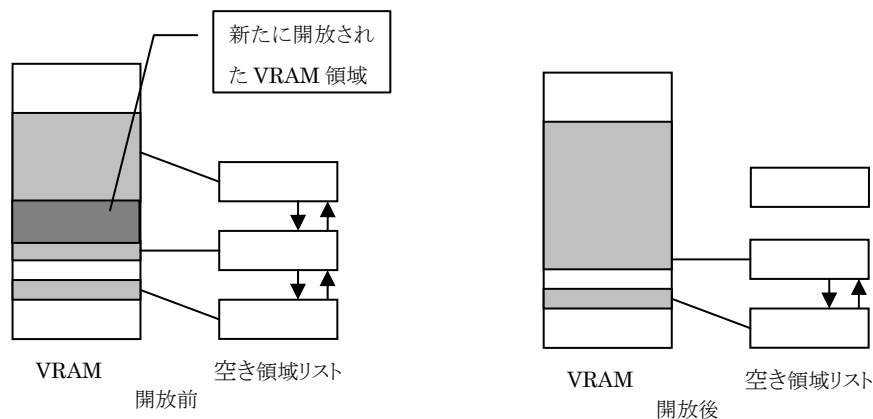


図 4-3 テクスチャ用メモリの開放

なお、領域ブロック結合に失敗した場合は、新たな空き領域ブロックを生成し、空き領域リストに追加登録を行います。このような場合にマネージャの管理情報が不足し、新たな空き領域ブロックが取得できない場合、メモリ領域の開放は失敗となりますので注意が必要です。

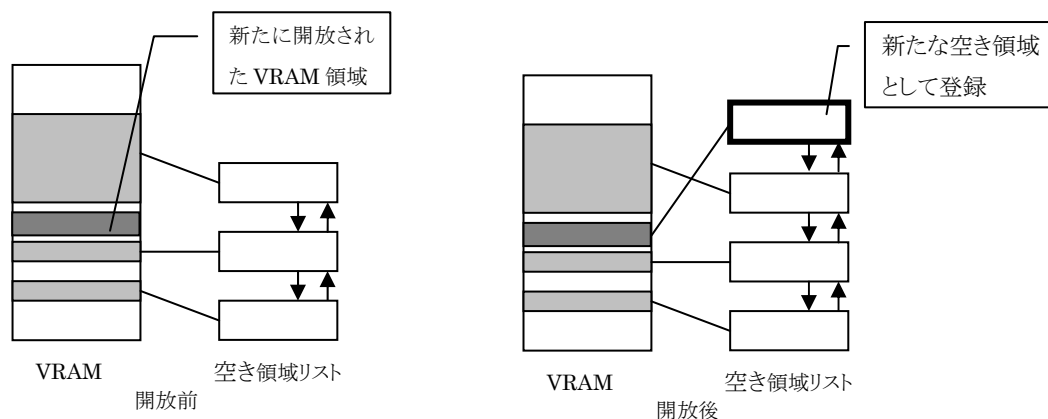


図 4-4 空きリスト結合処理に失敗する例

4.2 リンクドリスト パレットVRAMマネージャ

テクスチャ VRAM マネージャと同様に、パレット VRAM マネージャは管理情報の双方向リストによって、VRAM 中に点在する空き領域を管理しています。

4.2.1 初期化

```
void NNS_GfdInitLnkPlttVramManager
(
    u32      szByte,
    void*    pManagementWork,
    u32      szByteManagementWork,
    BOOL     useAsDefault
);
```

テクスチャ VRAM マネージャと同様に、管理領域に使用するメモリ領域とそのサイズを引数に渡して、パレット VRAM マネージャを初期化します。

4.2.2 パレット用メモリの確保、開放

メモリの確保、解放は基本的にテクスチャ VRAM マネージャと同様に動作しています。

4 色パレットの確保の際には、8 バイトアラインメントを施す必要があるため、アラインメントされた領域を確保します。アラインメントの際に発生した空きブロックはフリーブロックとして登録されます。不必要な管理領域の細分化を防ぐため、4 色パレットとそれ以外のフォーマットのパレットの領域確保をある程度まとめて行うことをお勧めします。

© 2004-2008 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複製・転写・頒布・貸与することを禁じます。