

Dreamcast

ライブラリマニュアル

CRI MPEG SofdecF/X

1999年 12月22日
2000年 3月16日

Ver . 2.24
Ver . 2.28

変 更 履 歴

年月日	バージョン	変 更 内 容
1999.12.22	2.24	新規作成。
2000.03.16	2.28	mwPlyExecServer 関数の記述を削除。 記述ミスを修正。

目 次

1. 概 要	1
1.1 SofdecF/X とは	1
2. CRI MPEG SofdecF/X	2
2.1 システム構成	2
2.2 モジュール構成	2
2.3 コントロールフロー	3
2.4 サウンドリソース	4
2.5 ビデオリソース	4
3. 単純再生	5
3.1 Shinobi システムの設定	5
3.2 ライブラリの初期化・終了処理	5
3.3 ハンドルの生成・消去	5
3.4 再生開始・停止	6
4. メモリ再生	7
5. ストリームジョイント再生	8
5.1 サーバ関数	9
5.2 ハンドルの動作状態	9
6. 動画再生時の注意点	10
6.1 ハンドルの生成・消去	10
6.2 再生の開始・停止	11
6.3 ムービーの最終フレームの表示	11
6.4 キャッシュ	12
6.5 CPU 負荷の軽減	13
7. 表示タイプ	14
7.1 自動表示型	15
7.2 ウィンドウ表示型	16
7.3 サーフェス表示型	17
8. マルチウィンドウ再生	19
9. 合 成	20
9.1 特 長	20
9.2 合成方式	21
9.3 アルファ合成の仕組み	22
9.4 合成モード	23
9.5 加算合成	25
9.6 ルミナンスキー合成	25
9.7 3 値アルファ合成	26

9.8 5 値アルファ合成	26
9.9 256 値アルファ合成	26
9.10 合成モードの切り替え	27
9.11 フェードイン・アウト	28
10. テクスチャムービー	29
11. マルチストリーム再生	31
11.1 仕組み	31
11.2 バッファの設定	32
12. シームレス連続再生	33
12.1 複数ファイルのシームレス連続再生	33
12.2 シームレスループ再生	33
13. データ仕様	34
13.1 定数	35
13.2 データ型	37
14. 関数仕様	39
14.1 初期化と終了処理	42
14.2 サーバ関数	46
14.3 基本動作処理	48
14.4 音声出力制御	54
14.5 ウィンドウ制御	56
14.6 サーフェス制御	60
14.7 エフェクト	66
14.8 テクスチャムービー	68
14.9 シームレス連続再生	69
14.10 再生モードの設定	72
14.11 内部の状態取得	74
14.12 エラー処理	78

1. 概 要

1.1 SofdecF/X とは

SofdecF/X は、ビジュアルエフェクトなどの特殊再生機能を持っています。以下の2つのライブラリから構成されます。

(1) CRI MPEG SofdecF/X ライブラリ

MPEG 動画をポリゴンと合成し再生することができます。

(2) SAN ライブラリ

非圧縮のフルカラー静止画をピクセル単位で合成することができます。

本書は、CRI MPEG SofdecF/X ライブラリについて説明します。SAN ライブラリについては、「簡易アニメーションライブラリ 組み込みマニュアル」を参照してください。

本ライブラリの特長は以下の通りです。

(1) MPEG Sofdec の上位互換

本ライブラリは、CRI MPEG Sofdec ライブラリの上位互換です。基本的な API は、ミドルウェア API に準拠しています。従って、ソースプログラムを修正することなく、リコンパイルするだけで MPEG SofdecF/X に移行できます。また、従来の MPEG Sofdec データも再生できます。

(2) ビジュアルエフェクト機能

アルファ合成や加算合成の機能を持ち、ピクセル単位でポリゴンと合成することができます。

(3) マルチウィンドウ表示機能

動画イメージの任意位置を切り出して、ディスプレイ上の任意の位置に変形させて表示することができます。複数のウィンドウで表示することができます。

(4) テクスチャ表示機能

動画をポリゴンのテクスチャとして表示できます。アルファ合成することできるので任意の形に抜くことができます。

(5) マルチストリーミング機能

複数の動画ファイルを同時に並行再生できます。4 ファイルまで並行再生できます。

(6) シームレス連続再生機能

別々の動画ファイルをシームレスに接続し再生することができます。また、同じファイルをシームレスにループ再生することもできます。現在、映像のみのファイルをサポートしています。

2. CRI MPEG SofdecF/X

2.1 システム構成

MPEG SofdecF/X のシステム構成は以下の通りです。

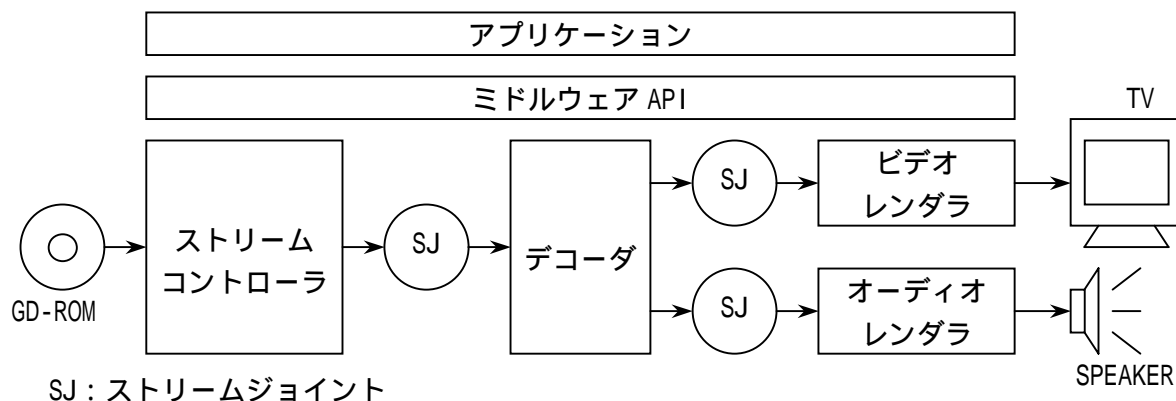


図 2-1 MPEG SofdecF/X ライブラリのシステム構成

2.2 モジュール構成

MPEG SofdecF/X は、以下のモジュールから構成されています。

表 2-1 内部モジュール

モジュール	略称	説 明
ミドルウェアプレイヤー	MWPLY	データの読み込みから出力までを管理し、映像や音声の再生を行います。
モジュールマネージャ	MWMNG	デコーダの CPU タイムを割り当てます。
ストリームコントローラ	MWGSC	映像や音声のデータを読み込みます。
デコーダ	CRI_SFDF	圧縮されたデータを展開します。
ビデオレンダラ	MWRNV	展開された映像データを出力します。
オーディオレンダラ	MWRNA	展開された音声データを出力します。

以下に、MPEG SofdecF/X の内部構成を示します。

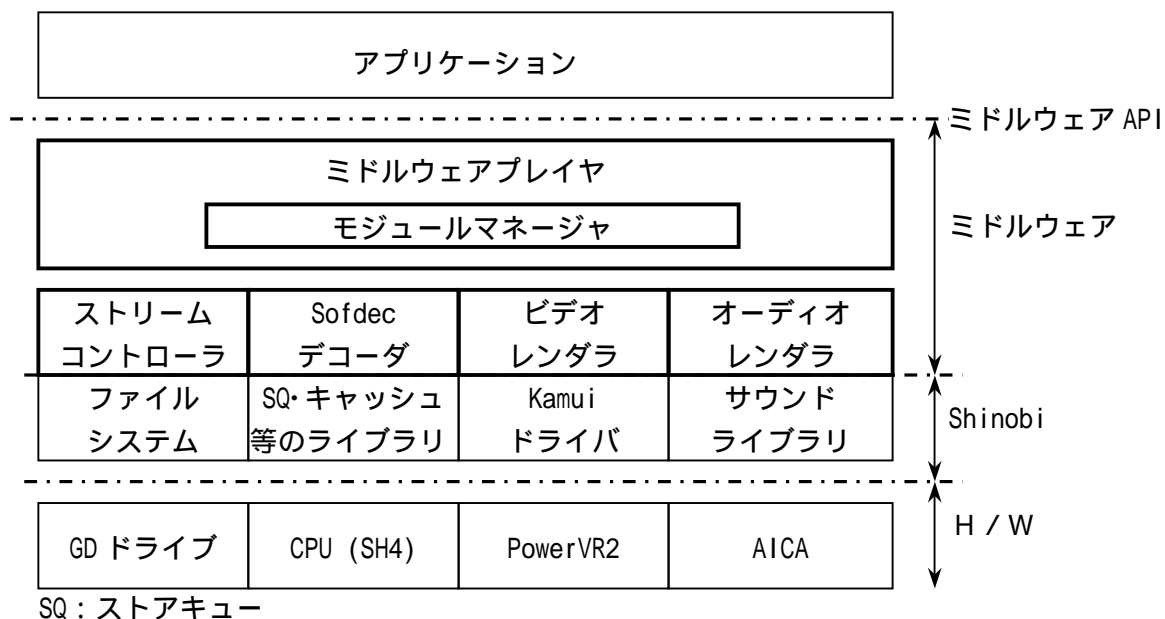


図 2-2 MPEG SofdecF/X の内部構成

2.3 コントロールフロー

MPEG SofdecF/X は、以下の 3 つのタイミングで処理が実行されます。

表 2 - 2 サーバ処理の種類

サーバ処理	説 明
V-Sync 割込サーバ処理	V-Sync 割込みによって実行されます。 音声データのデコードや音声の出力など、必ず定期的に行う処理を実行します。
メインサーバ処理	メイン処理内で、mwExecMainServer 関数が呼び出された時に実行されます。 映像データの転送や描画を行います。
アイドルサーバ処理	フレームフリップ関数内で、次のゲームフレームまでの待ち時間に実行されます。Ninja ライブラリでは、njWaitVSync 関数内で実行されます。 映像データのデコードを行います。

以下に、コントロールフローを示します。

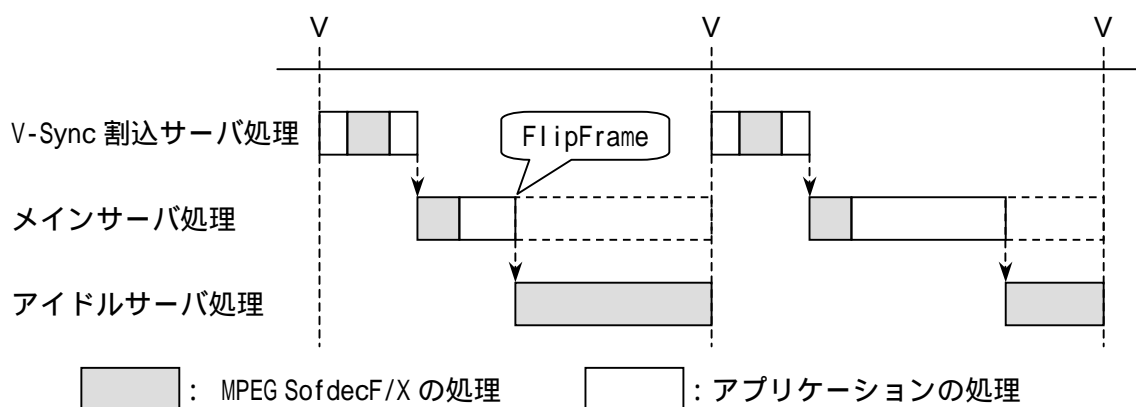


図 2 - 3 コントロールフロー

MPEG SofdecF/X における各サーバ処理の内訳は以下の通りです。

- : 音声データのデコードと音声出力処理
- : 映像データのテクスチャメモリへの転送と描画処理
- : 映像データのデコード

2.4 サウンドリソース

MPEG SofdecF/X における音声出力には、サウンドドライバとサウンドライブラリを使用しています。MPEG SofdecF/X 用にサウンドリソースを下記の通り確保して下さい。メモリブロックハンドルの最大値は 64 個なので、MPEG SofdecF/X を使用する事により、アプリケーション側で利用できるメモリブロックハンドルは減少します。

表 2-3 サウンドリソース (1 ハンドル)

サウンドリソース	必要量
PCM ストリームポート	1
メモリブロックハンドル	4
サウンド RAM	4040H(16448)byte × チャンネル数

Ver.2.24 ではチャンネル数は 2 に固定されています。

動画データを再生中に、サウンド RAM 内の効果音や MIDI シーケンスを同時に再生できます。セガ・ライブラリ環境において MPEG SofdecF/X は、サウンドライブラリの PCM ストリーム再生の機能を用いています。MPEG SofdecF/X 用 PCM ストリームバッファは、サウンド RAM の最後から 4040H (16448) byte 単位で割り当てられます。従って、MPEG SofdecF/X の使用するチャンネル数から PCM ストリームバッファのサイズを求め、その領域が重ならないようにマルチユニットファイルを作成する必要があります。

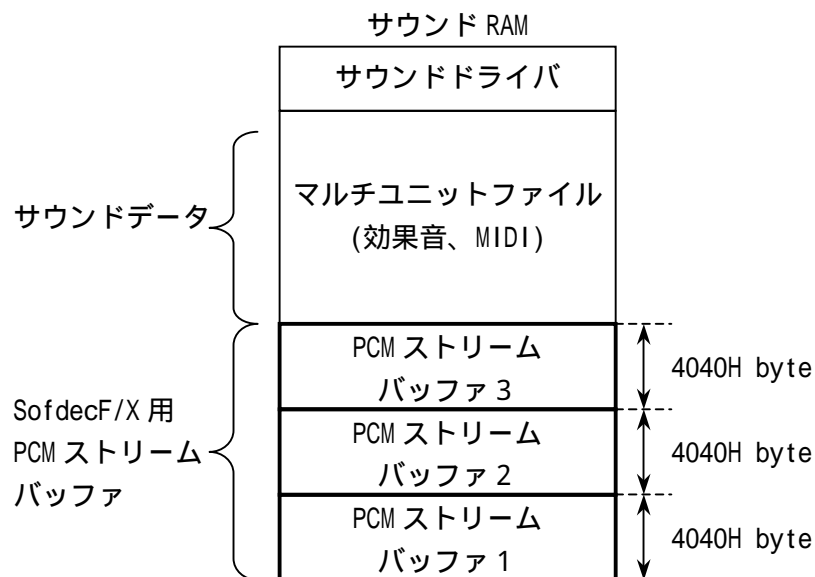


図 2-4 MPEG SofdecF/X 用 PCM ストリームバッファの位置

2.5 ビデオリソース

MPEG SofdecF/X は、Kamui ドライバ上に実装されています。初期化時に、Ninja ライブラリにアクセスし、頂点バッファなどの情報を取得しています。初期化時以外では、Ninja ライブラリと独立に動作します。以下に、グラフィックライブラリとの関係を示します。

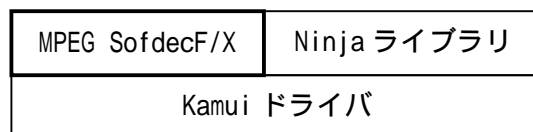


図 2-5 MPEG SofdecF/X とグラフィックライブラリとの関係

動画の再生には、YUV422 テクスチャを使用します。MPEG SofdecF/X は、ハンドル作成時に Kamui ドライバを使用して確保します。

3. 単純再生

3.1 Shinobi システムの設定

MPEG SofdecF/X は、mwPlyPreInitSofdec 関数をシステムの初期化(sbInitSystem)の前に実行する必要があります。

```
mwPlyPreInitSofdec();  
sbInitSystem(SYS_MODE, SYS_FRAME, SYS_COUNT); // Shinobi システムの初期化
```

3.2 ライブラリの初期化・終了処理

(1) Ninja ライブラリを使用する場合

mwPlyInitSofdec 関数を使用して初期化すると、MPEG SofdecF/X の内部で Ninja ライブラリの頂点バッファへのポインタを取得し、動画の描画を行います。

```
mwPlyInitSofdec(&iprm);  
/* 動画の再生 */  
mwPlyFinish();
```

(2) Kamui ドライバのみを使用する場合

mwPlyInitSfdFx 関数を使用して初期化します。頂点バッファは、mwPlySetVertexBuffer 関数によって、別途設定して下さい。

```
mwPlyInitSfdFx(&iprm);  
mwPlySetVertexBuffer(vbuf); // 頂点バッファの設定  
/* 動画の再生 */  
mwPlyFinishSfdFx();
```

3.3 ハンドルの生成・消去

MPEG SofdecF/X データを再生するためには、生成パラメータを設定しハンドルを生成します。

```
MWPLY          ply;  
MWS_PLY_CPRM_SFD cprm;  
  
memset(&cprm, 0, sizeof(cprm)); // 予約メンバのゼロ設定のため */  
cprm.compo_mode = MWD_PLY_COMPO_OPEQ; // 合成モード */  
cprm.ftype= MWD_PLY_FTYPE_SFD; // 再生ファイルタイプ */  
cprm.dtype= MWD_PLY_DTYPE_AUTO; // 画質優先 */  
cprm.max_bps = 450*1024*8; // ビットレート : 450 Kbyte/sec */  
cprm.max_width = 320; // 画像サイズ : 320x480 */  
cprm.max_height = 480;  
cprm.nfrm_pool_wk = 3; // フレームバッファの枚数 */  
cprm.wksize = mwPlyCalcWorkCprmSfd(cprm); // 作業領域サイズの計算 */  
cprm.work = syMalloc(cprm.wksize); // 作業領域の確保 */  
ply = mwPlyCreateSofdec(&cprm); // ハンドルの生成  
  
/* 動画の再生 */  
mwPlyDestroy(ply); // ハンドルの消去
```

3.4 再生開始・停止

GD-ROM 上のデータを再生する場合、mwPlyStartFname 関数により再生開始を制御します。再生方法に関わらず、mwPlyStop 関数により再生停止を制御します。

```
mwPlyStartFname(ply, "SAMPLE.SFD");          // 再生開始
/* 動画の再生 */
mwPlyStop(ply);                               // 再生停止
```

< サンプルプログラム >

```
void main(void)
{
    MWS_PLY_INIT_SFD  iprm;                    /* ライブラリ初期化パラメータ */
    MWS_PLY_CPRM_SFD  cprm;                    /* ハンドル生成パラメータ */
    MWPLY              ply;                    /* 再生ハンドル */

    mwPlyPreInitSofdec();                      /* Shinobi システム設定 */
    /*
     * 画面やサウンドの初期化
     */
    memset(&iprm, 0, sizeof(iprm));            /* 予約メンバのゼロ設定のため */
    iprm.mode= NJD_RESOLUTION_640x480_NTSCNI; /* 表示モード */
    iprm.frame= NJD_FRAMEBUFFER_MODE_ARGB8888; /* フレームバッファモード */
    iprm.count= 1;                             /* システムカウント */
    iprm.latency= 2;                            /* 表示レイテンシ */
    mwPlyInitSofdec(&iprm);                    /* ライブラリの初期化 */

    memset(&cprm, 0, sizeof(cprm));            /* 予約メンバのゼロ設定のため */
    cprm.compo_mode = MWD_PLY_COMPO_OPEQ;      /* 合成モード */
    cprm.ftype= MWD_PLY_FTYPE_SFD;             /* 再生ファイルタイプ */
    cprm.dtype= MWD_PLY_DTYPE_AUTO;            /* 画質優先 */
    cprm.max_bps = 450*1024*8;                 /* ビットレート : 450 Kbyte/sec */
    cprm.max_width = 320;                      /* 画像サイズ : 320x480 */
    cprm.max_height = 480;
    cprm.nfrm_pool_wk = 3;                     /* フレームバッファの枚数 */
    cprm.wksize = mwPlyCalcWorkCprmSfd(cprm); /* 作業領域サイズの計算 */
    cprm.work = syMalloc(cprm.wksize);         /* 作業領域の確保 */
    ply = mwPlyCreateSofdec(&cprm);            /* ハンドルの生成 */

    mwPlyStartFname(ply, "SAMPLE.SFD");        /* 再生開始 */
}
```

4. メモリ再生

mwPlyStartMem 関数によってメモリ上にある Sofdec データを再生できます。

< サンプルプログラム >

```
/* アプリケーションメイン関数 */
void main(void)
{
    void          *data;          /* Sofdec データへのポインタ */
    Sint32        size;          /* データサイズ */

    mwPlyPreInitSofdec();         /* Shinobi システム設定 */
    /*
     * 画面やサウンドの初期化
     */
    mwPlyInitSofdec(&iprm);
    ply = mwPlyCreateSofdec(&cprm); /* ハンドルの生成 */

    mwPlyStartMem(ply, (void *)data, size); /* 再生開始 */
    for (;;) {
        mwExecMainServer();         /* ミドルウェアライブラリの実行 */
        stat = mwPlyGetStat(ply);    /* ハンドルの状態の取得 */
        if ( stat == MWE_PLY_STAT_PLAYEND )
            break;
        njWaitVSync();              /* V-Sync 待 ち */
    }
    mwPlyDestroy(ply);             /* ハンドルの消去 */
    mwPlyFinishSofdec();           /* ライブラリの終了処理 */
}
```

5. ストリームジョイント再生

mwPlyStartSj 関数は、ユーザがストリームジョイントに供給した Sofdec データを再生します。ユーザが作成したストリームジョイントでも再生できますが、再生ハンドル内部のストリームジョイントを取得して再生することもできます。

< サンプルプログラム >

```
/* アプリケーションメイン関数 */
void main(void)
{
    SJ                sj;                /* ストリームジョイントハンドル */
    SJCK              ck, ck2;           /* チャンク */

    mwPlyPreInitSofdec();                /* Shinobi システム設定 */
    /*
     * 画面やサウンドの初期化
     */
    mwPlyInitSofdec(&iprm);
    ply = mwPlyCreateSofdec(&cprm);      /* ハンドルの生成 */
    sj = mwPlyGetInputSj(ply);           /* 入力ストリームジョイントの取得 */
    mwPlyStartSj(ply, sj);               /* 再生開始 */

    for (;;) {
        nroom = SJ_GetNumData(sj, SJ_LIN_FREE);
        SJ_GetChunk(sj, SJ_LIN_FREE, nroom, &ck); /* 空きチャンクを取得 */
        nbyte = user_supply_data(ck.data, ck.len); /* nbyte のデータを供給 */
        SJ_SplitChunk(&ck, len, &ck, &ck2); /* チャンクの分割 */
        SJ_PutChunk(sj, SJ_LIN_DATA, &ck); /* データチャンクを渡す */
        SJ_UngetChunk(sj, SJ_LIN_FREE, &ck2); /* 空きチャンクを戻す */

        mwExecMainServer();              /* メインサーバ処理の実行 */
        stat = mwPlyGetStat(ply);         /* ハンドルの状態の取得 */
        if ( stat == MWE_PLY_STAT_PLAYEND )
            break;
        njWaitVSync();                   /* V-Sync 待 ち */
    }

    mwPlyDestroy(ply);                   /* ハンドルの消去 */
    mwPlyFinishSofdec();                  /* ライブラリの終了処理 */
}
```

5.1 サーバ関数

MPEG SofdecF/X の内部状態は、メイン処理用サーバ関数(mwExecMainServer)を njWaitVSync 関数の前に呼び出すことで更新されます。

```
while (1) {  
    /* ペリフェラル情報の取得処理 */  
    :  
    mwExecMainServer();           // 動画データの描画処理など  
    /* 描画処理 */  
    :  
    njWaitVSync();  
}
```

5.2 ハンドルの動作状態

ハンドルの動作状態を下記に示します。生成した直後は、STOP 状態となります。再生開始後、状態は PREP -> PLAYING -> PLAYEND と遷移します。GD リードエラー等が発生すると、ERROR に遷移します。

表 5 - 1 ハンドルの状態

状 態	説 明
STOP	再生が停止している状態
PREP	再生の準備をしている状態
PLAYING	再生中の状態
PLAYEND	再生が終了した状態
ERROR	エラーが発生した状態

状態遷移図を以下に示します。

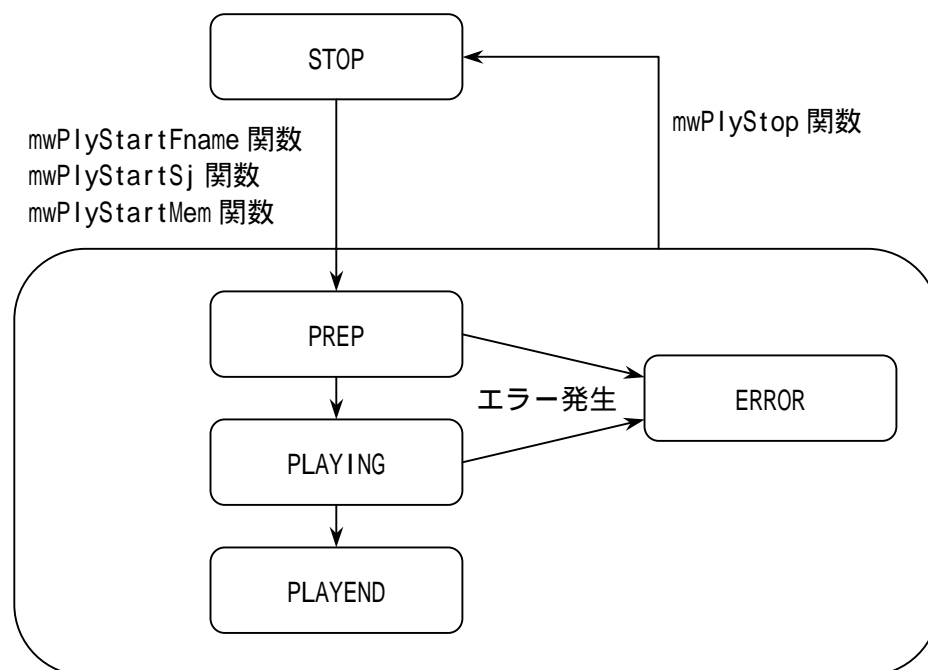


図 5 - 1 状態遷移図

6. 動画再生時の注意点

6.1 ハンドルの生成・消去

ハンドル生成時に、以下のビデオリソースを確保します。

(1) YUV422 テクスチャサーフェス

サーフェスのサイズは、動画の画像サイズ以上の2のべき乗になります。

<例> 動画サイズ サーフェスサイズ
 320 × 240 → 512 × 256

(2) 8ビットパレットテクスチャサーフェス

サーフェスのサイズは、動画の画像サイズ以上の2のべき乗の正方形になります。

<例> 動画サイズ サーフェスサイズ
 320 × 240 → 512 × 512

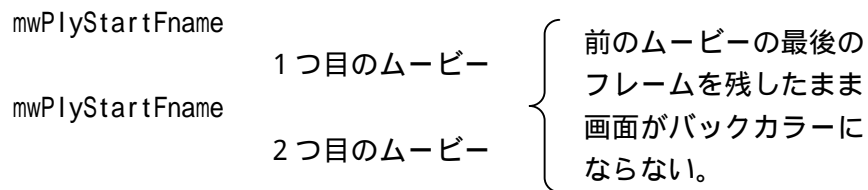
また、ハンドル消去時にテクスチャサーフェスを解放します。従って、再生中にハンドル消去する場合は、再生を停止した後、1V または 2V 分待ってからハンドルを消去してください。また、mwExecMainServer 関数は映像の描画を行っています。mwPlyDestroy 関数よりも後で、この関数を実行してください。

<サンプルプログラム>

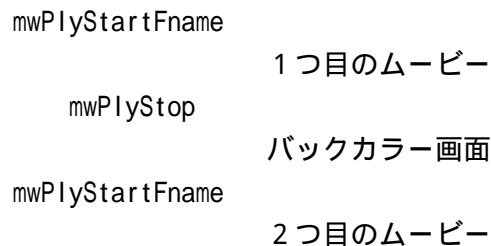
```
for (;;) {  
    if ( /* Bボタンがおされた */ ) {  
        mwPlyStop(ply);  
        njWaitVSync();      // レイテンシが 2V の時は 1 回  
        njWaitVSync();      // レイテンシが 3V の時は 2 回  
        mwPlyDestroy(ply);  
    }  
    mwExecMainServer();     // mwPlyDestroy 関数よりもあとに実行  
    njWaitVsync();  
}
```

6.2 再生の開始・停止

2つのムービーを続けて再生する場合、前のムービーの表示を残したまま、次のムービーに移りたい場合は以下の手順で、次のムービーを再生します。



mwPlyStop 関数は、動画の出力（描画）を停止します。ムービー再生中に mwPlyStop 関数を実行するとバックグラウンドカラーが表示されます。



説明の簡略化のため、mwPlyStartFname 関数を実行後、すぐにムービーが画面に出力されるように記述しました。しかし、実際には、mwPlyStartFname 関数を実行してもすぐにムービーが画面に出力されません。ミドルウェア再生ハンドルの状態が再生中(MWE_PLY_STAT_PLAYING)になって、初めて画面に出力されます。

6.3 ムービーの最終フレームの表示

ムービーの再生が終了すると、基本的には動画の最終フレームが描画され続けます。mwPlyStop 関数を実行すると描画が停止します。

また、MPEG の特性として、最後のフレームが表示されず 2 ～ 3 枚の画像がデコーダ内部に残ることがあります。従って、ムービーの最後を静止画として表示し残す場合は、ムービー作成時に 3 枚程度同じ画像を挿入して下さい。

MPEG SofdecF/X の動画は、ビットレートが高いため、I ピクチャの画質が悪くなります。P ピクチャまたは B ピクチャで静止するように調整してください。この画質の特性は、通常の MPEG データとは異なりますので注意が必要です。

< MPEG SofdecF/X のピクチャタイプによる画質 >

I ピクチャ < P ピクチャ = B ピクチャ

6.4 キャッシュ

MPEG SofdecF/X は、オペランドキャッシュがインデックスキャッシュモードのときに最高のパフォーマンスで動作します。インデックスキャッシュモードでなくても動作しますが、最大で約 20% 程度のパフォーマンス低下が生じます。

mwPlyInitSofdec 関数、mwPlyInitSfdFx 関数によって、ライブラリを初期化する時に、キャッシュモードをインデックスキャッシュモードに切り換えます。また、mwPlyFinishSofdec 関数、mwPlyFinishSfdFx 関数は、Shinobi のデフォルトのキャッシュモードに戻します。アプリケーションを効率良く動作させるためには、初期化と終了処理を頻繁に行うか、キャッシュ操作関数によってキャッシュモードを切り換えてください。

また、ポリゴンとムービーが共存する場合は、ムービーの再生負荷よりキャッシュモードを決めて下さい。目安としては、ムービーの解像度が 160×120 以下の場合は、通常のキャッシュモード、それ以上の場合はインデックスキャッシュモードに設定して下さい。

<キャッシュのコントロール例 1>

```
#define CACHE_SHINOBI%           // 通常のキャッシュモード
(SYD_CACHE_FORM_IC_ENABLE | SYD_CACHE_FORM_OC_ENABLE | SYD_CACHE_FORM_P1_CB )

#define CACHE_SOFDEC%           // MPEG SofdecF/X 用キャッシュモード
(SYD_CACHE_FORM_OC_INDEX | CACHE_SHINOBI)

mwPlyInitSofdec(...);           // インデックスキャッシュモード
syCacheInit(CACHE_SHINOBI);     // 通常モード
/* ポリゴンの描画 */
syCacheInit(CACHE_SOFDEC);      // インデックスキャッシュモード
mwPlyStartFname("sample.sfd");  // ムービー再生
```

<キャッシュのコントロール例 2>

```
mwPlyInitSofdec(...);          // インデックスキャッシュモード
mwPlyFinishSofdec();           // 通常モード
/* ポリゴンの描画 */
mwPlyInitSofdec(...);          // インデックスキャッシュモード
mwPlyStartFname("sample.sfd");  // ムービー再生
```

syCacheInit 関数は、Shinobi2 では関数名が変更される可能性があります。

6.5 CPU 負荷の軽減

以下の方法によって、CPU 負荷を軽減し、コマ落ちを軽減できます。

(1) ハーフペル処理の簡略化

B ピクチャのハーフペル処理を簡略化し、3 ~ 4 % ぐらい高速化することができます。画質はわずかに劣化します。

```
mwPlySetFastHalfpel(ply, ON);
```

(2) B ピクチャの除去

エンコード時に B ピクチャを含まないストリームとしてエンコードします。例えば、N=6, M=1 のパラメータでデコードすることにより B ピクチャの含まないストリームを作成できます。B ピクチャのデコードは、他のピクチャよりも負荷が大きいため、B ピクチャが無くなることで 6 ~ 9 % 程度 CPU 負荷を軽減できます。

```
c:\>sfvencd -in=sample.avi -out=sample.m1v -gop_n=6 -gop_m=1
```

(3) ビットレートの削減

CPU 負荷は、SFD データのビットレートに比例して大きくなります。データレートを落とすことにより CPU 負荷を軽減できます。

(4) 解像度の調整

解像度を低くすることにより CPU 負荷を軽減させることができます。例えば、NTSC の TV では 640 × 480 の解像度の上下 16 ドットは表示されません。従って、320 × 480 の動画を 320 × 448 の動画にクロップすることによって 6 % 程度 CPU 負荷を減少させることができます。

(5) フレームプール数の増加

フレームプール数を増加させることにより、コマ落ちを減少させることができます。この方法では、CPU 負荷は変わりませんが、CPU 負荷の変動を吸収できます。MPEG では、ピクチャの種別、動画の複雑さにより CPU 負荷が大きく変動します。

```
cprm. nfrm_pool_wk = 6;           // この値を増やすことによりコマ落ちを軽減  
ply = mwPlyCreateSofdec(&cprm);
```

7. 表示タイプ

ハンドル生成パラメータの表示タイプによって表示方法を指定できます。表示タイプには、以下のような型があります。指定された表示タイプによって使用できる関数が異なります。

- (1) 自動表示型 ... MWD_PLY_DTYPE_AUTO
- (2) ウィンドウ表示型 ... MWD_PLY_DTYPE_WND
- (3) サーフェス表示型 ... MWD_PLY_DTYPE_SRF

MWD_PLY_DTYPE_WND は従来の MWD_PLY_DTYPE_FULL と同じです。MPEG SofdecF/X から名前が変更されました。

< サンプルプログラム >

```
cprm.dtype = MWD_PLY_DTYPE_AUTO;  
ply = mwPlyCreateSofdec(&cprm);
```

表 7-1 表示タイプ

表示タイプ	表示方法	使用可能な関数
自 動 AUTO	動画の解像度から自動的に表示領域の大きさを決定します。バイリニアフィルタによるボケが少なくなるように、UV 値を調節します。	mwPlySetDispPos
ウィンドウ WND	矩形のウィンドウ表示を行います。ウィンドウ全体に対しパラメータを設定します。扱えるウィンドウは1つです。	mwPlySetDispPos mwPlySetDispSize mwPlySetBright mwPlySetBrightOfst
サーフェス SRF	ウィンドウの4頂点を独立に制御できます。マルチウィンドウ表示を行うためのモードです。	mwPlySetSrfPos mwPlyGetSrfPos mwPlySetSrfBright mwPlyGetSrfBright mwPlySetSrfBrightOfst mwPlyGetSrfBrightOfst mwPlySetImgPos mwPlyGetImgPos

7.1 自動表示型

自動表示型は、バイリニアフィルタによるボケを軽減し、シャープな映像が得られるように自動的に表示サイズを調整します。例えば、320×240 のムービーは、639×479 ドットで表示されます。従って、バックグラウンドカラーが設定されていると、画面の右端と下に 1 ドット分にバックグラウンドカラーが表示されてしまいます。以下に動画の解像度と表示サイズの関係を示します。

表 7-2 X 方向の動画サイズと表示サイズの関係

動画 X サイズ	320	352	480	640	左記以外
表示 X サイズ	639	640	640	640	640

左右が 16 ドットずつ表示されない

表 7-3 Y 方向の動画サイズと表示サイズの関係

動画 Y サイズ	224	240	320	336	448	480	左記以外
表示 Y サイズ	447	479	320	448	448	480	480

< バイリニアフィルタ処理について >

バイリニアフィルタ処理は、UV 値を 0.0～1.0 として設定すると、下図のような演算が行われ全体的にぼかした映像になります。

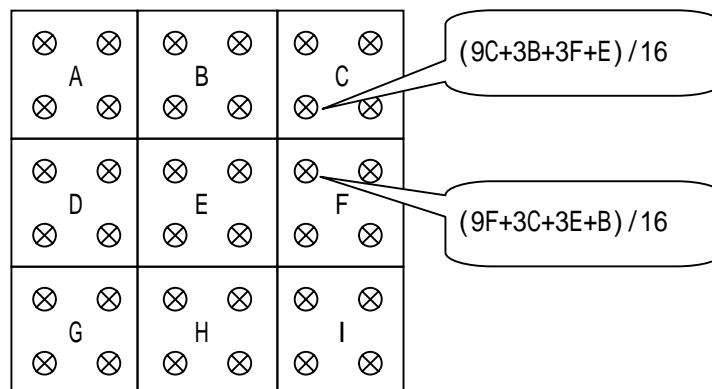


図 7-1 通常のバイリニアフィルタ処理

MPEG SofdecF/X では、2 倍に拡大するときは UV 値を下図のように設定し、ボケのすくない映像で再生できます。この方法では (2 倍 - 1) ドットの点しか生成できないため、320×240 の画像は 639×479 として表示されます。

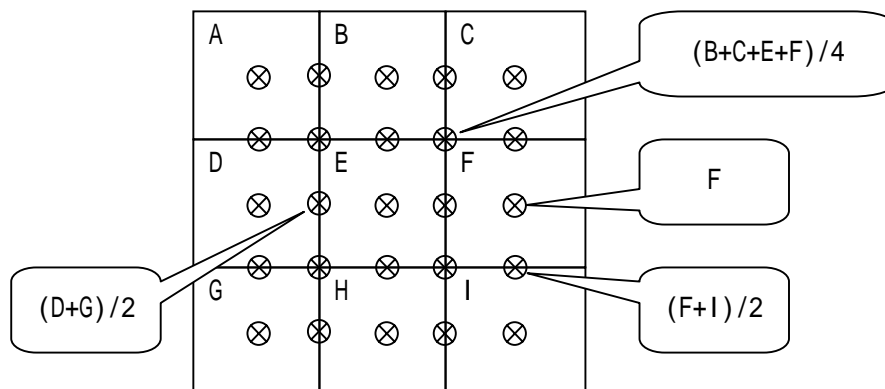


図 7-2 MPEG SofdecF/X におけるバイリニア処理

7.2 ウィンドウ表示型

ウィンドウ表示型は、矩形のウィンドウ単位で制御できます。

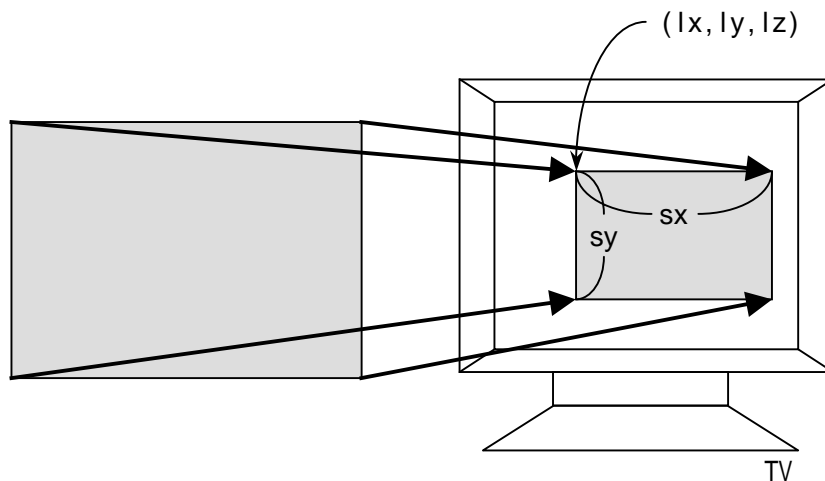


図 7-3 ウィンドウの制御

< サンプルプログラム >

```
ply = mwPlyCreateSofdec(&cprm);
mwPlySetDispPos(ply, lx, ly);           // 表示位置の設定
mwPlySetDispSize(ply, sx, sy);         // 表示サイズの設定
mwPlySetDispZ(ply, lz);                // 奥行きの設定
mwPlySetBright(ply, bright);           // 輝度の設定
mwPlySetBrightOfst(ply, bofst);        // 輝度オフセットの設定
mwPlyStartFname(ply, "sample.sfd");
for (;;) {
    if ( ウィンドウの移動 ) {
        mwPlySetDispPos(ply, lx++, ly++);
        mwPlySetDispZ(ply, lz++);
    }
    if ( ウィンドウサイズの変更 ) {
        mwPlySetDispSize(ply, sx++, sy++);
    }
    if ( 輝度の変更 ) {
        mwPlySetBright(ply, bright++);
    }
    if ( 輝度オフセットの変更 ) {
        mwPlySetBrightOfst(ply, bofst++);
    }
    mwExecMainServer();
    /* ポリゴンの描画 */
    njWaitVSync();
}
```

7.3 サーフェス表示型

サーフェス表示型は、ウィンドウの各4頂点を独立に以下のパラメータを設定できます。この表示タイプは映像を複数のウィンドウに表示できます。

(1) 表示位置

X座標、Y座標、Z座標を設定できます。X座標、Y座標はフレームバッファの座標値です。左上が(0.0, 0.0)、右下が(639.0, 479.0)になります。Z座標を指定しますが、透視変換は行いません。

(2) 頂点輝度

アルファ、赤、緑、青の強さを指定します。1.0 から 0.0 までの値を設定します。映像データに対して、この設定値が乗じられます。これらの値を 0.0 から 1.0 まで変化させると黒からのフェードイン、1.0 から 0.0 まで変化させるとフェードアウトできます。また、アルファ値は、背後の映像との混合比率になります。これを変化させることによりディゾルブ効果が得られます。

(3) 頂点輝度オフセット

アルファ、赤、緑、青の加算成分を指定します。1.0 から 0.0 までの値を設定します。映像データに対しこの設定値と 255 を乗じた値が加算されます。これらの値を 0.0 から 1.0 まで変化させると白へのフェードアウト、1.0 から 0.0 まで変化させると白からフェードインできます。

(4) 映像切り出し位置

各頂点对応する映像データの位置を指定します。UV 座標に相当しますが、指定する値は、映像データに対するピクセル単位での位置です。

また、各頂点のパラメータは 'Z' を描くような順序で設定します。

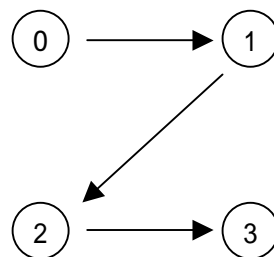


図 7-4 サーフェス型の各頂点の設定順序

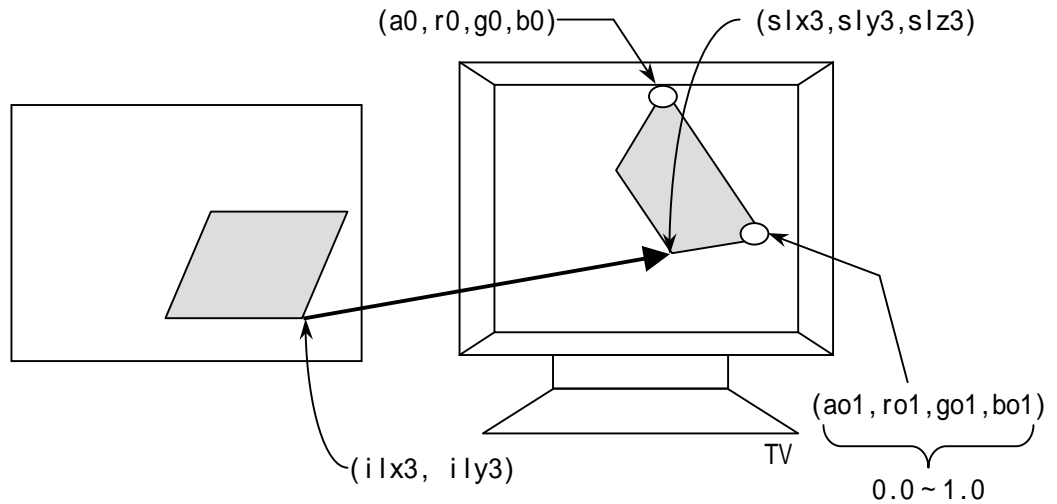


図 7-4 サーフェスの制御

< サンプルプログラム >

```
ply = mwPlyCreateSofdec(&cprm);

// イメージ位置の設定
mwPlySetImgPos(ply, 0, ilx0, ily0);
mwPlySetImgPos(ply, 1, ilx1, ily1);
mwPlySetImgPos(ply, 2, ilx2, ily2);
mwPlySetImgPos(ply, 3, ilx3, ily3);

// サーフェス位置の設定
mwPlySetSrfPos(ply, 0, slx0, sly0, slz0);
mwPlySetSrfPos(ply, 1, slx1, sly1, slz1);
mwPlySetSrfPos(ply, 2, slx2, sly2, slz2);
mwPlySetSrfPos(ply, 3, slx3, sly3, slz3);

mwPlySetSrfBright(ply, 0, a0, r0, g0, b0);           // 輝度の設定
mwPlySetSrfBrightOfst(ply, 1, ao1, ro1, go1, bo1);  // 輝度オフセットの設定

mwPlyStartFname(ply, "sample.sfd");
for (;;) {
    mwExecMainServer();
    /* ポリゴンの描画 */
    njWaitVSync();
}
```

8. マルチウィンドウ再生

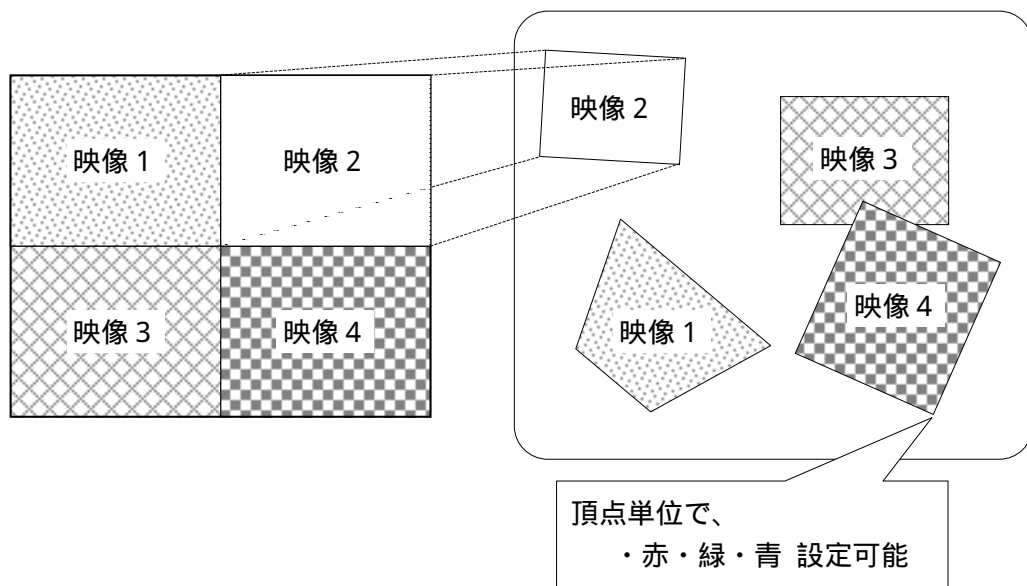


図 8 - 1 マルチウィンドウ表示

サーフェス表示型によりマルチウィンドウ表示を行うことができます。以下の手順でマルチウィンドウ表示を行います。

(1) サーフェスモードの設定

ハンドル生成時に表示モードをサーフェスモードにします。

```
cprm.dtype = MWD_PLY_DTYPE_SRF  
ply = mwPlyCreateSofdec(&cprm);
```

(2) サーフェスポイント用バッファの確保と設定

サーフェスポイントとは、ウィンドウの各頂点を示し、上記のパラメータを持ちます。ウィンドウは、4つのサーフェスポイントから構成されます。例えば、25個のウィンドウを表示するためには、100個のサーフェスのポイントが必要になります。

```
size = mwPlyCalcSrfBufSize(ply, 4*25);           // ポイントバッファサイズの計算  
buf = syMalloc(size);  
mwPlySetSrfPntBuf(ply, 4*25, buf, bsize);       // ポイントバッファの設定  
.  
/* 10 番目のウィンドウの設定 */  
mwPlySetSrfPos(ply, 10*4 + 0, lx0, ly0, lz0);   // ポイント番号に注意  
mwPlySetSrfPos(ply, 10*4 + 1, lx0, ly0, lz0);  
mwPlySetSrfPos(ply, 10*4 + 2, lx0, ly0, lz0);  
mwPlySetSrfPos(ply, 10*4 + 3, lx0, ly0, lz0);
```

9. 合 成

9.1 特 長

MPEG SofdecF/X は、ビジュアルエフェクトのための合成機能を持ちます。

(1) アルファ付きムービーをポリゴンと合成

アルファ付きのムービーをポリゴンの前景に表示し、ピクセル単位で合成することができます。

(2) 様々な合成方式をサポート (半透明合成 / 加算合成 / ルミナンスキー合成 / アルファ合成)

状況に応じた複数の合成方式をサポートしています。方式の違いによって C P U 負荷や機能が異なります。

(3) 合成モードの途中切り替え可能

ムービーの途中で合成モードを切り換えることができます。カットごとに適切な合成方式を選択し切り換えることができます。

(4) フェードイン・アウト可能

映像全体にアルファ値を持つことができます。C P U 負荷を増加させることなくフェードイン・フェードアウトすることができます。この機能を用いてスムーズにビジュアルエフェクトを開始することができます。

(5) 専用ツールにより AVI ファイルを変換

アルファ付き AVI ファイルから CRI MPEG CRAFT を用いて合成用のムービーデータを作成します。

9.2 合成方式

MPEG SofdecF/X の合成方法として以下の方法があります。

(1) 不透明表示

映像を合成せずに描画します。

(2) 半透明表示 (ウィンドウ全体でのアルファ合成)

ウィンドウ単位でアルファブレンディングします。

(3) 加算合成

ポリゴンの RGB 値とムービーの RGB 値を単純に加算します。

(4) アルファ合成 (ピクセル単位のアルファ合成)

ポリゴンの RGB 値とムービーの RGB 値をピクセル単位でアルファブレンディングします。

表 9 - 1 SofdecF/X の使用リソースと負荷

	使用リソース	レンダリング負荷	CPU 負荷の増加
半透明・加算合成	なし	半透明 1 枚	なし
アルファ合成	256 個のパレット (ARGB8888)	半透明 2 枚	5 ~ 10%

<アルファブレンディングとは>

アルファブレンディングは、混合値 により以下のように表現されます。

$$\text{output} = \quad \cdot s + (1 - \quad) \cdot d$$

output : 合成後の RGB 値

 : 混合比率

s : ムービーの RGB 値

d : ポリゴンの RGB 値

ポリゴンの RGB 値とは、ムービーよりも奥にあるオブジェクトが描画されたフレームバッファの RGB です。

9.3 アルファ合成の仕組み

ピクセル単位のアルファ合成は、以下のように2枚の半透明ポリゴンを描画することによって実現されています。マスクサーフェスを描画（乗算処理）後、映像サーフェスを加算しています。

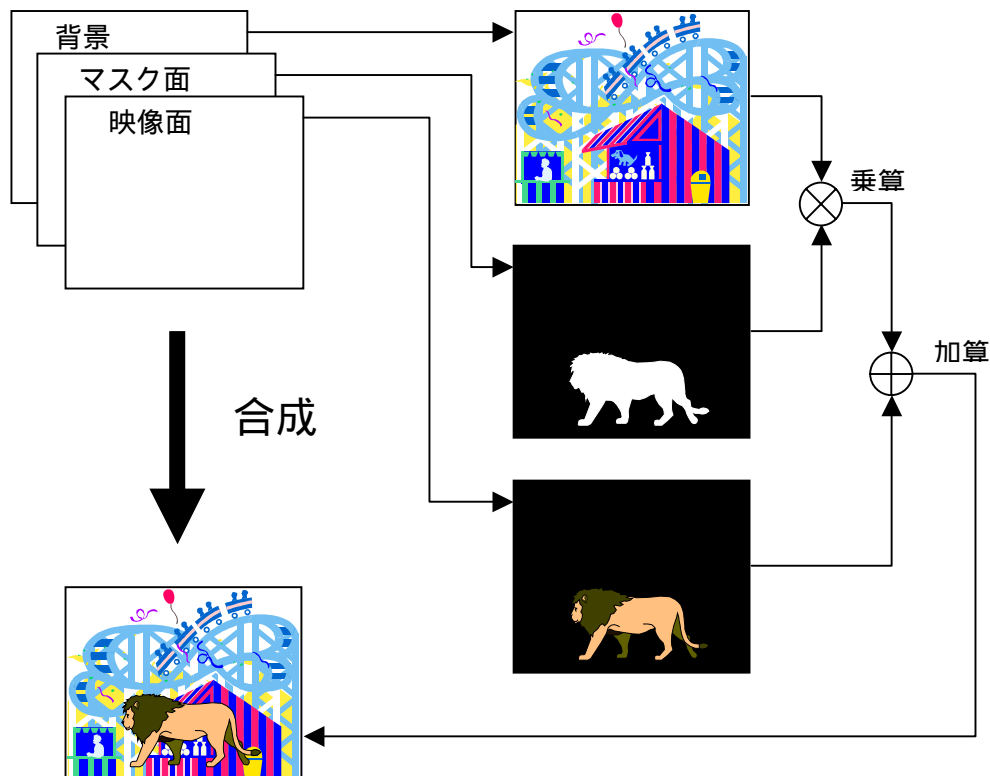


図 9 - 1 合成の仕組み

SofdecF/X では、ハンドルから以下の2枚のテクスチャサーフェスを取得できます。

(1) 映像サーフェス

矩形形式の YUV422 テクスチャ。デコードされた映像を格納。

(2) マスクサーフェス

Twiddle 形式の 8bit/pixel のパレットテクスチャ。マスクイメージを格納。

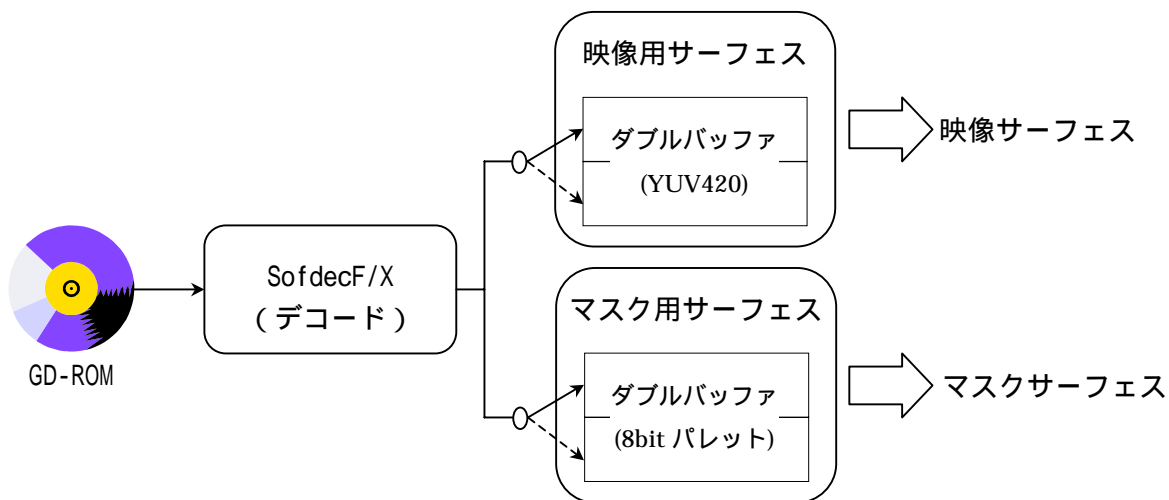


図 9 - 2 SofdecF/X のテクスチャサーフェス

9.4 合成モード

MPEG SofdecF/X は以下の合成モード（コンポジションモード）を持ちます。ハンドル生成パラメータの `cprm.compo_mode` メンバーによって設定します。

```
cprm.compo_mode = MWD_PLY_COMPO_~; （合成モード）  
ply = mwPlyCreateSofdec(&cprm);
```

（１）不透明表示モード(MWD_PLY_COMPO_OPEQ)

不透明ポリゴンとしてムービーを表示します。デフォルトのモードです。

（２）半透明合成モード(MWD_PLY_COMPO_TRNSP)

半透明ポリゴンとしてムービーを表示します。ピクセル単位の合成はできませんが、ウィンドウ全体としてのアルファ合成はできます。

（３）加算合成モード(MWD_PLY_COMPO_ADD)

映像全体を加算します。従って、黒い部分が完全に抜けて見えます。光のエフェクトなどに向いています。

（４）ルミナンスキー合成モード(MWD_PLY_COMPO_LUMI)

輝度に応じてアルファ値が変化します。暗い部分ほど透過し、明るい部分是不透明になります。デフォルトでは、下記の設定になっています。mwPlySetAlphTbl 関数によって、輝度からアルファ値への変換テーブルを設定できます。

0 - 15	完全に透明
16 - 100	アルファ値が 0.0（透明）から 1.0（不透明）まで徐々に増加
100 - 255	完全に不透明

（５）３値アルファ合成モード(MWD_PLY_COMPO_ALPH3)

不透明・半透明（アルファ値は 50%）・透明の 3 段階の合成ができます。量子化ビット数が約 7 ビット程度の画質になります。

（６）５値アルファ合成モード(MWD_PLY_COMPO_ALPH5)

透明(0%)・25%・50%・75%・不透明(100%)の 5 段階の合成ができます。しかしながら、映像の品質は劣化します。

（７）フルアルファ合成モード(MWD_PLY_COMPO_ALPH256)

256 段階のアルファ値によって合成できます。最高の品質で合成できますが、作業領域がほぼ 2 倍近く必要になります。小さなムービーに向いています。

（８）混合合成モード(MWD_PLY_COMPO_MIX)

フルアルファ合成モード以外の合成モードを、mwPlySetCompoMode 関数によって再生途中で切り換えることができます。

各合成モードにおける使用リソースとCPU 負荷とレンダリング負荷を示します。これは、あくまで目安であり、ムービーデータにより変動します。

< 素材条件 >

解像度 : 256 × 256
 データレート : 300Kbyte/sec
 GOP パターン : N=6, M=1

表 9 - 2 各合成モードの使用リソースと負荷

合成モード	CPU 負荷	レンダリング 負荷	テクスチャ	パレット
不透明表示	40%	不透明 1 枚	YUV422	未使用
半透明	40%	半透明 1 枚	YUV422	未使用
加算合成	40%	半透明 1 枚	YUV422	未使用
ルミナンス合成	50%	半透明 2 枚	YUV422 PLT8	使用
3 値アルファ合成	50%	半透明 2 枚	YUV422 PLT8	使用
5 値アルファ合成	50%	半透明 2 枚	YUV422 PLT8	使用
フルアルファ合成	65%	半透明 2 枚	YUV422 PLT8	使用
混合合成			YUV422 PLT8	使用

< 使用リソース >

YUV422 テクスチャ : 256Kbyte (256 × 256, 2 枚)
 8 ビットパレットテクスチャ : 128Kbyte (256 × 256, 2 枚)
 パレット : ARGB8888, 768 番目から 256 エントリ

9.5 加算合成

加算合成は、表示されている映像に対して、ムービーのイメージデータを単純に加算しています。光を利用したエフェクトに向いています。

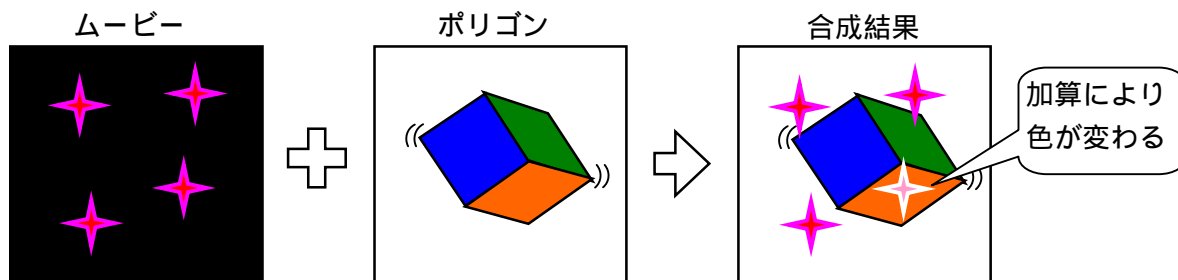


図 9 - 3 加算合成

< サンプルプログラム >

```
cprm.compo_mode = MWD_PLY_COMPO_ADD;           // 加算合成モード
:
ply = mwPlyCreateSofdec(&cprm);
mwPlyStartFname(ply, "effect.sfd");
for (;;) {
    mwExecMainServer();
    (ポリゴン描画)
    njWaitVSync();
}
```

9.6 ルミナンスキー合成

ルミナンスキー合成は、輝度に応じた混合比率により合成しています。黒い部分を抜く事ができます。炎・煙・稲妻などの黒い部分がないエフェクトに効果的です。

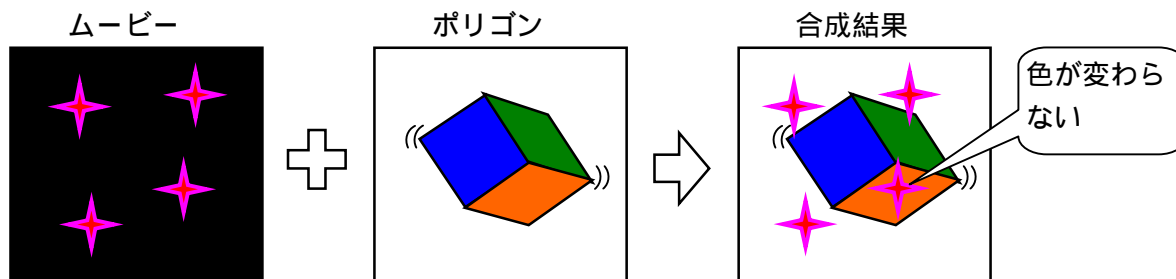


図 9 - 4 ルミナンスキー合成

< サンプルプログラム >

```
cprm.compo_mode = MWD_PLY_COMPO_LUMI;         // ルミナンスキー合成モード
:
ply = mwPlyCreateSofdec(&cprm);
mwPlyStartFname(ply, "effect.sfd");
:
```

9.7 3 値アルファ合成

3 値アルファ合成は、不透明・半透明・透明の 3 段階のアルファ値を扱うことができます。アルファ情報を映像データ内に埋め込むため、映像の品質は 7 ビット相当の画質になります。セル画的な合成に向きます。また、ルミナンス合成では合成できない黒い映像を合成することができます。

< サンプルプログラム >

```
cprm.compo_mode = MWD_PLY_COMPO_ALPH3;          // 3 値アルファ合成モード
:
ply = mwPlyCreateSofdec(&cprm);
mwPlyStartFname(ply, "effect.sfd");
:
```

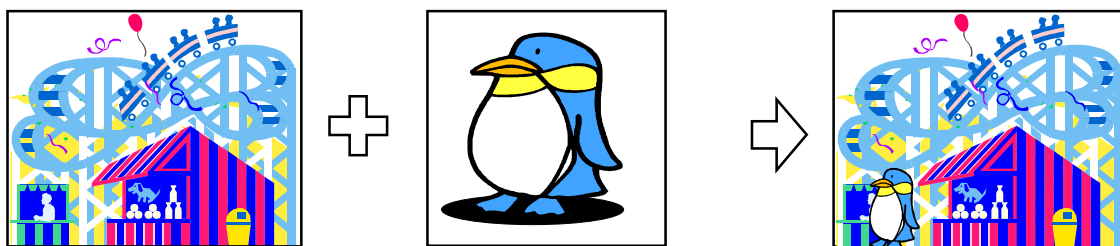


図 9 - 5 3 値アルファ合成

9.8 5 値アルファ合成

5 値アルファ合成は、0% (透明)、25%、50%、75%、100% (不透明) の 5 段階のアルファ値によって合成することができます。半透明を必要とするビジュアルエフェクト的な合成ができますが、アルファ情報を映像データ内に埋め込むため、映像の品質は 5 ビット相当の画質となってしまいます。

< サンプルプログラム >

```
cprm.compo_mode = MWD_PLY_COMPO_ALPH5;          // 5 値アルファ合成モード
:
ply = mwPlyCreateSofdec(&cprm);
mwPlyStartFname(ply, "effect.sfd");
```

9.9 256 値アルファ合成

256 段階のアルファ合成を行うことができます。最高の品質の合成を行うことができますが、CPU 負荷が増加します。また、メモリなどのリソースも他の合成方式よりも必要です。

9.10 合成モードの切り替え

ムービー中のカット毎に合成モードを切り換えることができます。256 値アルファ合成以外の合成モードを切り換えることができます。ハンドル生成時に合成モードを MW_PLY_COMPO_MIX に設定してください。

mwPlySetCompoMode 関数によって合成方式を変更できますが、エフェクトコールバック関数内で実行してください。このコールバック関数は、マスクを生成する直前に呼び出されます。引数としてフレーム番号を渡しますので、この番号に従って合成モードを変更して下さい。

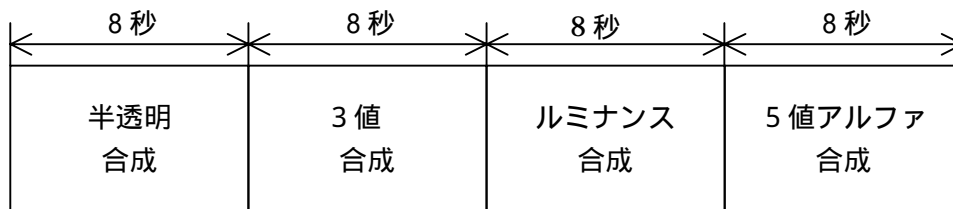


図 9-6 合成モードの切り替え

< 合成モードの切り替えサンプル >

```
void user_chg_compo(void *obj, Sint32 fno, Sint32 time, Sint32 tunit)
{
    MWPLY ply=(MWPLY)obj;

    if ( fno == 0 ) {
        mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_TRNSP);
    } else if ( fno == 1*8*30 ) {
        mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_LUMI);
    } else if ( fno == 2*8*30 ) {
        mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_ALPH3);
    } else if ( fno == 3*8*30 ) {
        mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_ALPH5);
    }
}

void main(void)
{
    cprm.compo_mode = MWD_PLY_COMPO_MIX;
    ply = mwPlyCreateSofdec(&cprm);
    mwPlyEntryFxCb(ply, user_chg_compo, (void *)ply);
    mwPlyStartFname(ply, "vfx01.m1v");
}
```

9.1.1 フェードイン・アウト

表示ウィンドウの各頂点のアルファ値を操作することにより、動画のフェードイン・アウトを行うことができます。合成方式は不透明(MWD_PLY_COMPO_OPEQ)以外を指定して下さい。また、表示タイプはサーフェス表示型でなければなりません。

< サンプルプログラム >

```
cprm.dtype = MWD_PLY_COMPO_TRNSP;    // 半透明合成モード
cprm.dtype = MWD_PLY_DTYPE_SRF;      // サーフェス表示型
ply = mwPlyCreateSofdec(&cprm);
mwPlyStartFname(ply, "vfx01.m1v");
for (;;) {
    njWaitVsync();
    if ( per->on & PDD_DGT_KU ) {
        a += 1.0f/60.0f;    // 1秒でフェードイン
        a = ( a < 1.0f ) ? a : 1.0f;
        for (i=0; i<4; i++)
            mwPlySetSrfBright(ply, i, a, 1.0f, 1.0f, 1.0f); // アルファ値の設定
    }
    if ( per->on & PDD_DGT_KD ) {
        a -= 1.0f/60.0f;    // 1秒でフェードアウト
        a = ( a > 0.0f ) ? a : 0.0f;
        for (i=0; i<4; i++)
            mwPlySetSrfBright(ply, i, a, 1.0f, 1.0f, 1.0f); // アルファ値の設定
    }
    mwExecMainServer();
}
```


10. テクスチャムービー

SofdecF/X(MPEG, SAN)は、Kamui ドライバを使用して映像用テクスチャサーフェス、マスク用テクスチャサーフェスを確保しています。アプリケーションは、各ライブラリのハンドルからサーフェスを取得し、アプリケーションのオブジェクトサーフェスに映像とマスクを貼ることができます。

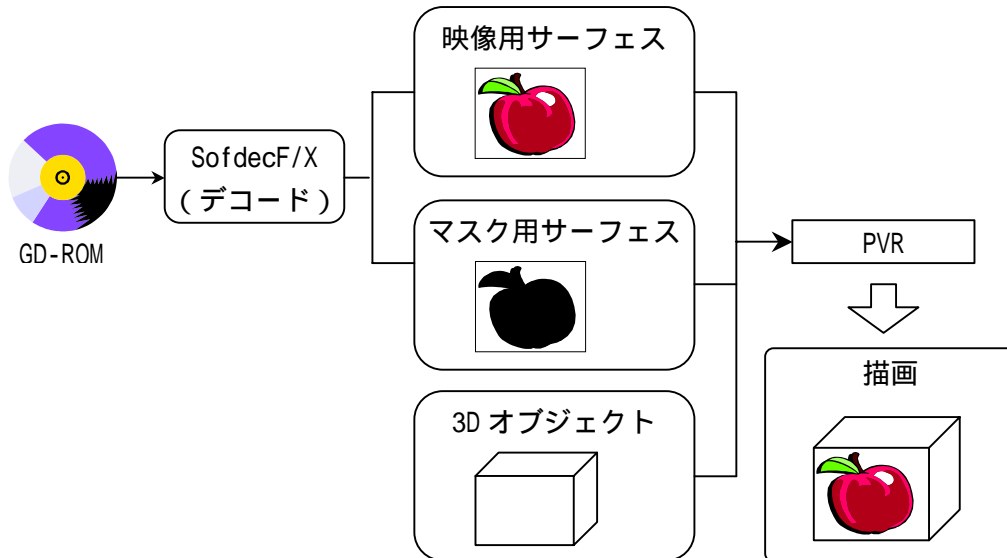


図 10-1 テクスチャムービーの仕組み

マスクの付いていない通常のムービーは、ポリゴンに映像サーフェスを貼り付けるだけで、テクスチャムービーとして再生できます。マスク付きムービーは、あらかじめポリゴンに2枚のテクスチャを貼っておく必要があります。

内側（マスク面）： アルファ付き (FBS_SA|FBD_ISA)

SRCBlendingMode=SRCALPHA, DSTBlendingMode=INVSRALPHA

外側（映像面）： 加算 (FBS_SA|FBD_ONE)

SRCBlendingMode=SRCALPHA, DSTBlendingMode=ONE

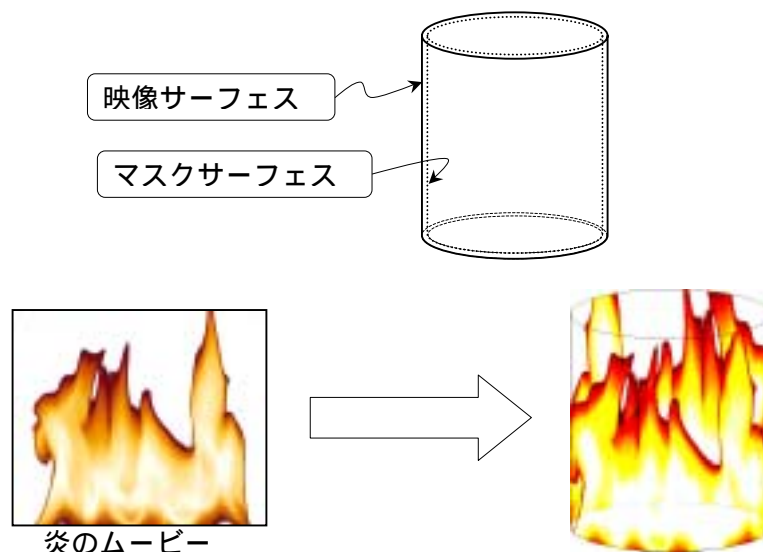


図 10-2 マスク付きテクスチャムービー

< MPEG SofdecF/X によるサーフェスの取得 >

```
mwExecMainServer();           // テクスチャ RAM への転送
mwPlyGetMvFrm(ply, &frm);      // 映像サーフェスの取得
user_set_video_surface(frm.srf); // 映像サーフェスの貼り付け
mwPlyGetMskFrm(ply, &frm);      // マスクサーフェスの取得
user_set_msk_surface(frm.srf);  // マスクサーフェスの貼り付け
/* ポリゴンの描画 */
```

frm.srf は Kamui ドライバの割り当てたサーフェスです。

< Ninja ライブラリによるテクスチャムービー >

njSetMvSurface 関数によって、ムービーサーフェスをテクスチャリスト内の指定された番号のテクスチャに登録できます。njSetMvSurface 関数は、ミドルウェアのサンプルプログラム内で定義しています。

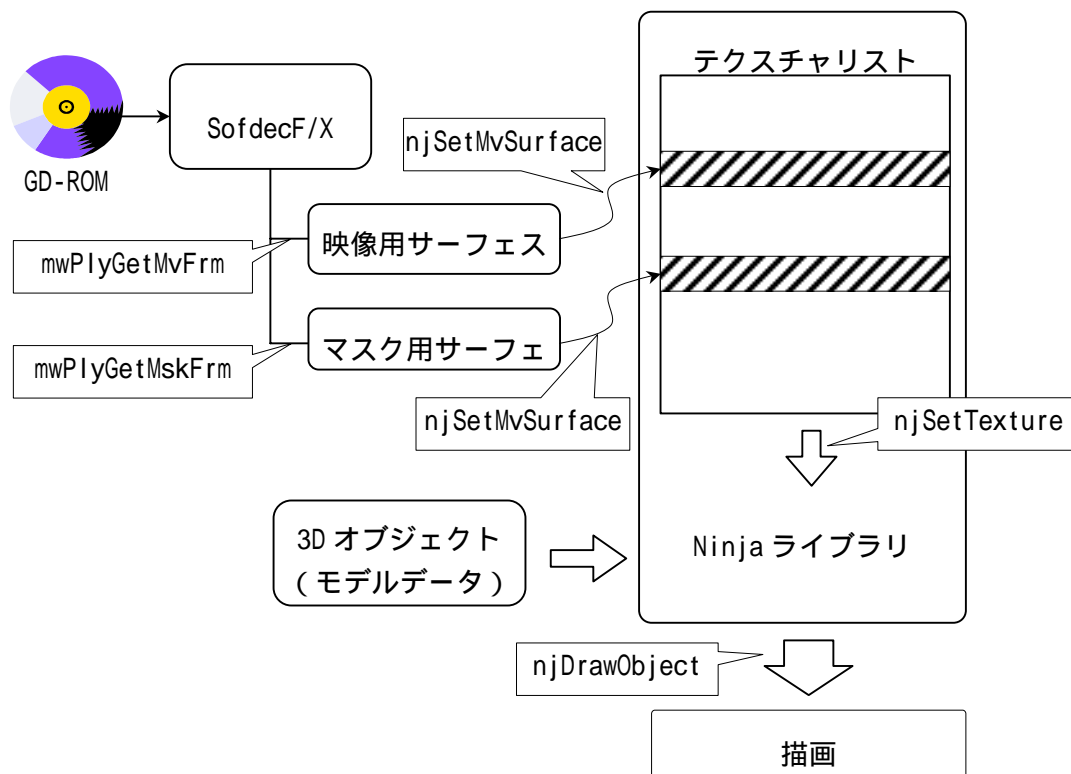


図 10-3 Ninja ライブラリによるテクスチャムービー

1 1 . マルチストリーム再生

1 1 . 1 仕組み

MPEG SofdecF/X では、GD-ROM 上に別々に格納された複数のムービーファイルを同時に再生できます。この機能は、ADX のマルチストリーミング機能を用いていますので、ADX をリンクする必要があります。以下のように複数のハンドルを生成し、各ハンドルに対して、非同期に再生開始することができます。

```
ply1 = mwPlyCreateSofdec(&cprm1);  
ply2 = mwPlyCreateSofdec(&cprm2);  
for (;;) {  
    if ( A ボタンが押された )  
        mwPlyStartFname(ply1, "movie1.sfd");  
    if ( B ボタンが押された )  
        mwPlyStartFname(ply2, "movie2.sfd");  
}
```

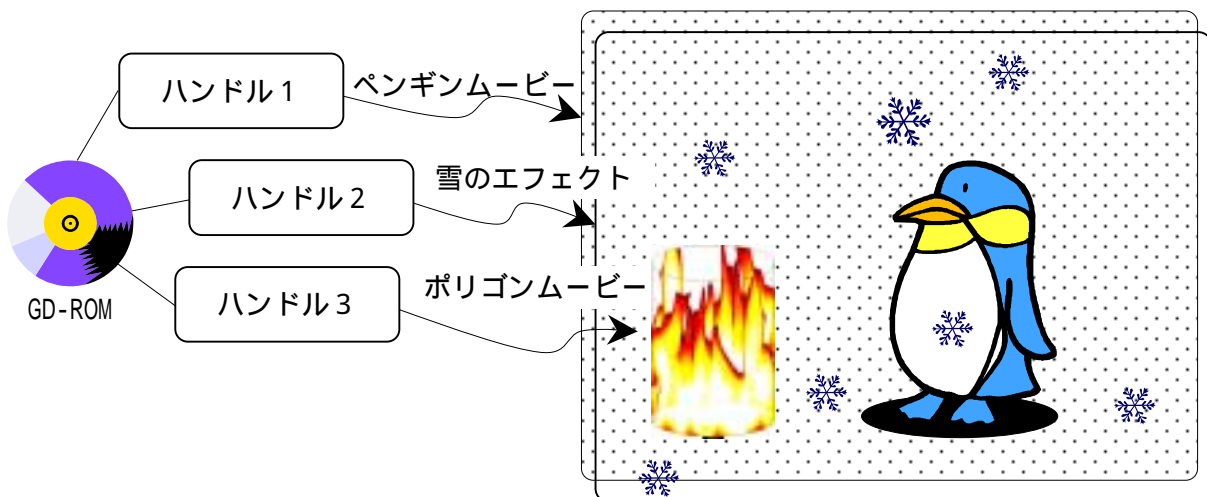


図 1 1 - 1 マルチストリーム再生

1 1.2 バッファの設定

マルチストリーミングを行う場合は、入力バッファを大きくする必要があります。入力バッファの大きさは、生成パラメータの max_bps によって決まります。例えば、2つのファイルを同時に再生する場合は、max_bps を2倍にしなければなりません。

```
#define NSTM(3)                // 同時再生ストリーム数
#define MAX_BPS1(150*1024*8)   // 最大ビットレート 1 (150Kbyte/sec)
#define MAX_BPS2(100*1024*8)   // 最大ビットレート 2 (100Kbyte/sec)
#define MAX_BPS3(75*1024*8)    // 最大ビットレート 3 (75Kbyte/sec)

// ハンドルの生成 1 (150Kbyte/sec 用)
cprm1.max_bps = MAX_BPS1*NSTM;
cprm1.wksize = mwPlyCalcWorkCprmSfd(&cprm);
cprm1.work = syMalloc(cprm.wksize);
ply1 = mwPlyCreateSofdec(&cprm1);

// ハンドルの生成 2 (100Kbyte/sec 用)
cprm2.max_bps = MAX_BPS2*NSTM;
cprm2.wksize = mwPlyCalcWorkCprmSfd(&cprm2);
cprm2.work = syMalloc(cprm2.wksize);
ply2 = mwPlyCreateSofdec(&cprm2);

// ハンドルの生成 3 (75Kbyte/sec 用)
cprm3.max_bps = MAX_BPS3*NSTM;
cprm3.wksize = mwPlyCalcWorkCprmSfd(&cprm3);
cprm3.work = syMalloc(cprm3.wksize);
ply3 = mwPlyCreateSofdec(&cprm3);
```

1 2 . シームレス連続再生

1 2 . 1 複数ファイルのシームレス連続再生

MPEG SofdecF/X は、分割された複数のムービーファイルをシームレスに接続し再生できます。同じファイルを繰り返し再生することもできます。最大 9 時間まで連続して再生できます。現バージョンでは、映像の連続再生のみをサポートしています。CRI ADX ライブラリが必要です。

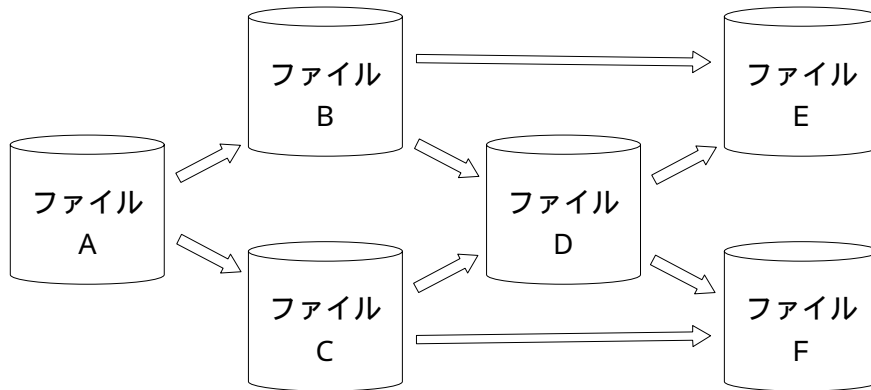


図 1 2 - 1 シームレス連続再生

下記の手順により、シームレス連続再生を行うことができます。

- (1) 再生するファイルの登録 (mwPlyEntryFname 関数)
- (2) シームレス連続再生の開始 (mwPlyStartSeamless 関数)
- (3) 再生中に、更に接続するファイルを登録する事も可能
- (4) シームレス連続再生モードの解除 (mwPlyReleaseSeamless 関数)
- (5) 再生終了状態へ

最大 8 ファイルまで登録しておくことができます。最後のムービーファイルの再生が終了する 1 秒前までに、次のファイルを登録してください。

```
ply = mwPlyCreateSofdec(&cprm);
mwPlyEntryFname(ply, "movie1.m1v");
mwPlyEntryFname(ply, "movie2.m1v");
mwPlyStartSeamless(ply);
for (;;) {
    if ( A ボタンが押された )
        mwPlyEntryFname(ply, "movie3.m1v");
    if ( B ボタンが押された )
        mwPlyReleaseSeamless(ply);
    if ( mwPlyGetStat(ply) == MWE_PLY_STAT_PLAYEND )
        break;
}
```

1 2 . 2 シームレスループ再生

mwPlyStartFnameLp 関数によって単一ファイルを簡単にシームレスループ再生することができます。

```
ply = mwPlyCreateSofdec(&cprm);
mwPlyStartFnameLp(ply, "movie1.m1v");
```

13. データ仕様

ライブラリのデータ一覧を以下に示します。

表 13-1 データ一覧

データ名		機 能	番号
定 数			
MWE_PLY_STAT_~		ハンドルの状態	1.1
MWE_PLY_FTYPE~		再生するファイルのタイプ	1.2
MWE_PLY_DTYPE~		動画の表示タイプ	1.3
MWD_PLY_COMPO~		合成モード	1.4
データ型			
MWPLY		ミドルウェア再生ハンドル	2.1
MWS_PLY_INIT_SFD		初期化パラメータの構造体	2.2
MWS_PLY_CPRM_SFD		ハンドル生成のパラメータ構造体	2.3
MWS_PLY_MVFRM		映像サーフェス情報構造体	2.4

1 3.1 定 数

Title データ	Data Name MWE_PLY_STAT_~	Data ハンドルの状態	No 1.1
--------------	-----------------------------	-----------------	-----------

以下の定数は、ハンドルの状態を表します。

定数名	説 明
MWE_PLY_STAT_STOP	停止中
MWE_PLY_STAT_PREP	準備中
MWE_PLY_STAT_PLAYING	再生中
MWE_PLY_STAT_PLAYEND	再生終了
MWE_PLY_STAT_ERROR	エラー状態

Title データ	Data Name MWE_PLY_FTYPE~	Data 再生するファイルのタイプ	No 1.2
--------------	-----------------------------	----------------------	-----------

以下の定数は、MPEG SofdecF/X で再生するファイルのタイプです。

定数名	説 明
MWE_PLY_FTYPE_SFD	MPEG Sofdec (音声 + 映像)
MWE_PLY_FTYPE_MPV	MPEG/Video (映像のみ)

Title データ	Data Name MWE_PLY_DTYPE ~	Data 動画の表示タイプ	No 1.3
--------------	------------------------------	------------------	-----------

以下の定数は、動画の表示タイプを表します。

定数名	説 明
MWE_PLY_DTYPE_AUTO	画面の表示サイズを自動的に調整 画質を優先させる時はこのモードを使用します。 テクスチャの UV 座標を調整しているため、バイ リニアフィルタを使用してもボケの少ない映像 が再生されます。 このモードを使用した場合は、mwPlySetDispPos, mwPlySetDispSize 関数を使用してはいけない。
MWE_PLY_DTYPE_FULL	全画面に表示されるように調整 Ver.1 との整合性のために残しています。
MWE_PLY_DTYPE_WND	表示位置とサイズをウィンドウレベルで調整 mwPlySetDispPos,mwPlySetDispSize 関数により 設定します。
MWE_PLY_DTYPE_SRF	表示位置とサイズをサーフェスレベルで調整 mwPlySetSrf ~ 関数を使用し、頂点後とに表示位 置や輝度を設定する事が可能。

Title データ	Data Name MWD_PLY_COMPO ~	Data 合成モード	No 1.4
--------------	------------------------------	---------------	-----------

以下の定数は、合成モードを表します。

定数名	説 明
MWD_PLY_COMPO_OPEQ	不透明
MWD_PLY_COMPO_TRNSP	透明
MWD_PLY_COMPO_ADD	加算合成
MWD_PLY_COMPO_LUMI	ルミナンスキー合成
MWD_PLY_COMPO_ALPH3	3 値アルファ合成
MWD_PLY_COMPO_ALPH5	5 値アルファ合成
MWD_PLY_COMPO_ALPH256	フルアルファ合成
MWD_PLY_COMPO_MIX	混合合成モード 1 つのストリームに複数の合成モードが混在。

1 3.2 データ型

Title	Data Name	Data	No
データ	MWPLY	ミドルウェア再生ハンドル	2.1

MPEG SofdecF/X の再生を制御するためのハンドル。

Title	Data Name	Data	No
データ	MWS_PLY_INIT_SFD	初期化パラメータの構造体	2.2

MPEG SofdecF/X 用の初期化関数に設定するパラメータ構造体。Ninja の初期化で、実際に設定した表示パラメータと同じ値を設定してください。メンバーは以下の通りです。

メンバー	型 名	説 明
mode	Sint32	画面モード
frame	Sint32	フレームバッファのカラーモード
count	Sint32	フレームカウント数
latency	Sint32	表示レイテンシ (2V または 3V)

Title データ	Data Name MWS_PLY_CPRM_SFD	Data ハンドル生成のパラメータ構造体	No 2.3
--------------	-------------------------------	-------------------------	-----------

MPEG SofdecF/X 用ミドルウェア再生ハンドルを生成する時に、設定するパラメータの構造体。
メンバーは以下の通りです。

メンバー	型 名	説 明
ftype	Sint32	再生するファイルのタイプ MWE_PLY_FTYPE_~ から選択する。
max_bps	Sint32	最大のビットストリーム量（単位：bit/sec） 実際よりも少なくすることも可能。 再生が途切れる時は、この値を増やしてください。
max_width	Sint32	再生画像サイズの最大幅（単位：ピクセル）
max_height	Sint32	再生画像サイズの最大高さ（単位：ピクセル）
nfrm_pool_wk	Sint32	システム領域のフレームプール数（通常：3） 処理負荷の変動によりフレーム落ちが発生した場合は、この値を増やしてください。
work	Sint8*	使用するバッファへのポインタ
wksize	Sint32	確保したバッファサイズ（単位：バイト）
dtype	Sint32	動画の表示タイプ MWE_PLY_DTYPE_~ から選択する。
compo_mode	Sint32	合成モード MWD_PLY_COMP_~ から選択する。

Title データ	Data Name MWS_PLY_MVFRM	Data 映像サーフェス情報構造体	No 2.3
--------------	----------------------------	----------------------	-----------

テクスチャムービーとして再生する時に得られる動画のサーフェス情報です。

メンバー	型 名	説 明
srf	void*	テクスチャサーフェス
width	Sint32	有効サーフェスの幅（単位：ピクセル）
height	Sint32	有効サーフェスの高さ（単位：ピクセル）

14. 関数仕様

ライブラリの関数一覧を以下に示します。

表 14-1 関数一覧

関数名	機 能	番号
初期化と終了処理		
mwPlyPreInitSofdec	システム初期化の設定	1.1
mwPlyInitSofdec	ライブラリの初期化(Ninja 用)	1.2
mwPlyFinishSofdec	ライブラリの終了処理(Ninja 用)	1.3
mwPlyInitSfdFx	ライブラリの初期化(Kamui、Kaumui2 用)	1.4
mwPlyFinishSfdFx	ライブラリの終了処理(Kamui、Kaumui2 用)	1.5
mwPlySetDispMode	表示モードの設定	1.8
mwPlySetVertexBuffer	頂点バッファの設定	1.9
サーバ関数		
mwExecMainServer	サーバ関数	2.1
mwPlyExecTexSvr	テクスチャ転送用サーバ関数(Kamui2 用)	2.2
mwPlyExecDrawSvr	映像描画用サーバ関数(Kamui2 用)	2.3
基本動作処理		
mwPlyCreateSofdec	ハンドルの生成	3.1
mwPlyCalcWorkCprmSfd	作業領域サイズの計算	3.2
mwPlyDestroy	ハンドルの消去	3.3
mwPlyStartFname	再生開始(GD-ROM からの再生)	3.4
mwPlyStartSj	再生開始(ストリームジョイントからの再生)	3.5
mwPlyStartMem	再生開始(メモリからの再生)	3.6
mwPlyStop	再生停止	3.7
mwPlyGetStat	ハンドルの状態の取得	3.8
mwPlyGetTime	再生サンプル数の取得	3.9
mwPlyGetInputSj	入力ストリームジョイントの取得	3.10
mwPlyPause	ポーズの設定	3.11
音声出力制御		
mwPlySetOutVol	ボリュームの設定	4.1
mwPlyGetOutVol	ボリュームの取得	4.2
mwPlySetOutPan	パンポットの設定	4.3
mwPlyGetOutPan	パンポットの取得	4.4

表 14-2 関数一覧

関数名	機 能	番号
ウィンドウ制御		
mwPlySetDispPos	表示位置の設定	5.1
mwPlySetDispSize	表示サイズの設定	5.2
mwPlySetBright	輝度の設定	5.3
mwPlyGetBright	輝度の取得	5.4
mwPlySetBrightOfst	輝度オフセットの設定	5.5
mwPlyGetBrightOfst	輝度オフセットの取得	5.6
mwPlySetDispZ	表示スクリーンの奥行き値の設定	5.7
mwPlyGetDispZ	表示スクリーンの奥行き値の取得	5.8
サーフェス制御		
mwPlyCalcSrfBufSize	サーフェス [*] インタ用バッファサイズ [*] の計算	6.1
mwPlySetSrfPntBuf	サーフェスポイント用バッファの設定	6.2
mwPlySetSrfPos	表示位置の設定	6.3
mwPlyGetSrfPos	表示位置の取得	6.4
mwPlySetSrfBright	輝度の設定	6.5
mwPlyGetSrfBright	輝度の取得	6.6
mwPlySetSrfBrightOfst	輝度オフセットの設定	6.7
mwPlyGetSrfBrightOfst	輝度オフセットの取得	6.8
mwPlySetImgPos	イメージ位置の設定	6.9
mwPlyGetImgPos	イメージ位置の取得	6.10
mwPlyGetImgSize	イメージサイズの取得	6.11
エフェクト		
mwPlySetCompoMode	合成モードの設定	7.1
mwPlyEntryFxCb	エフェクトコールバック関数の登録	7.2
mwPlySetAlphTbl	輝度から 値への変換テーブルの設定	7.3
テクスチャムービー		
mwPlyGetMvFrm	ムービーフレームの取得	8.1
mwPlyGetMskFrm	マスクフレームの取得	8.2
シームレス連続再生		
mwPlyEntryFname	シームレス連続再生ファイルの登録	9.1
mwPlyStartSeamless	シームレス連続再生の開始	9.2
mwPlyReleaseSeamless	シームレス連続再生の解除	9.3
mwPlySetLpFlg	シームレスループ再生の設定	9.4
mwPlyStartFnameLp	シームレスループ再生の開始	9.5

表 14-3 関数一覧

関数名	機 能	番号
再生モードの設定		
mwPlySetFastHalfpel	高速ハーフペル処理の設定	10.1
mwPlySetAudioSw	音声出力スイッチの設定	10.2
mwPlySetVideoSw	映像表示スイッチの設定	10.3
内部の状態取得		
mwPlyGetNumDropFrm	コマ落ちしたフレーム数の取得	11.1
mwPlyGetNumDecPool	デコード済みフレーム数の取得	11.2
mwPlyGetNumTotalDec	デコードした全ピクチャ数の取得	11.3
mwPlyGetNumTotalSkip	スキップした全ピクチャ数の取得	11.4
mwPlyGetNumIbufRoom	入力バッファの空き容量の取得	11.5
mwPlyGetNumDispWait	表示待ちフレーム数の取得	11.6
mwPlyGetNumRend	レンダリング中のフレーム数の取得	11.7
mwPlyGetNumLoadFrm	テクスチャメモリへ転送したフレーム数の取得	11.8
エラー処理		
mwPlyEntryErrFunc	エラー発生時に呼ばれる関数の登録	12.1

1 4.1 初期化と終了処理

ライブラリの初期化と終了処理を行います。

Title 関 数	Function Name	Function	No
	mwPlyPreInitSofdec	システム初期化の設定	1.1

[書 式] void mwPlyPreInitSofdec(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] 割込みスタックを設定します。割込みスタックサイズは 16Kbyte となります。
この関数を実行すると、割込み処理を実行する時にスタックを切り替えます。
必ず sbInitSystem 関数の前で実行してください。

Title 関 数	Function Name	Function	No
	mwPlyInitSofdec	ライブラリの初期化(Ninja 用)	1.2

[書 式] void mwPlyInitSofdec(MWS_PLY_INIT_SFD *iprm);

[入 力] iprm: 初期化パラメータ

[出 力] なし

[関数値] なし

[機 能] ライブラリを初期化します。オペランドキャッシュモードをインデックスキャッシュモードに切り換えます。Ninja を使用する場合に mwPlyFinishSofdec 関数と対で使用して下さい。各種サーバ関数を設定します。サーバ関数には以下の種類のものがあります。

(1) メインサーバ

メイン処理内で、mwExecMainServer が呼び出された時に実行されます。

(2) V-Sync 割込みサーバ

V-Sync 割り込み時に実行されます。

(3) アイドルサーバ

njWaitVSync 内で、次のゲームフレームまでの待ち時間に実行されます。

このサーバ内で動画のデコードを実行します。

[備 考] Ninja に、実際に設定した表示モードと同じ値を初期化パラメータに設定します。

[例] 使用例を以下に示します。

```
sbInitSystem(SYS_MODE, SYS_FRAME, SYS_COUNT);  
njInitVertexBuffer(VB_OP, VB_OM, VB_TP, VB_TM, VB_PT);  
iprm.mode= SYS_MODE;  
iprm.frame= SYS_FRAME;  
iprm.count= SYS_COUNT;  
iprm.latency= MWD_PLY_LATENCY(VB_OP, VB_OM, VB_TP, VB_TM, VB_PT);  
mwPlyInitSofdec(&iprm);
```

Title 関 数	Function Name	Function	No
	mwPlyFinishSofdec	ライブラリの終了処理(Ninaj 用)	1.3

[書 式] void mwPlyFinishSofdec(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] ライブラリの終了処理を行います。mwPlyInitSofdec 関数で切り換えたキャッシュモードを Shinobi のデフォルトに戻します。

Ninja を使用する場合は、mwPlyInitSofdec 関数と対で使用して下さい。

Title 関 数	Function Name	Function	No
	mwPlyInitSfdFx	ライブラリの初期化(Kamui、Kamui2 用)	1.4

[書 式] void mwPlyInitSfdFx(MWS_PLY_INIT_SFD *iprm);

[入 力] iprm: 初期化パラメータ

[出 力] なし

[関数値] なし

[機 能] ライブラリを初期化します。オペランドキャッシュモードをインデックスキャッシュモードに切り換えます。Kamui を使用する場合に mwPlyFinishSfdFx 関数と対で使用して下さい。また、必ず mwPlySetVertexBuffer 関数によって描画用の頂点バッファを設定して下さい。各種サーバ関数を設定します。サーバ関数には以下の種類のものがあります。

(1) メインサーバ

メイン処理内で、mwExecMainServer が呼び出された時に実行されます。

(2) V-Sync 割り込みサーバ

V-Sync 割り込み時に実行されます。

(3) アイドルサーバ

njWaitVSync 内で、次のゲームフレームまでの待ち時間に実行されます。

MPEG Sofdec ライブラリは、このサーバ内で動画のデコードを実行します。

[備 考] Kamui に、実際に設定した表示モードと同じ値を初期化パラメータに設定します。

[例] 使用例を以下に示します。

```

sbInitSystem(SYS_MODE, SYS_FRAME, SYS_COUNT);
iprm.mode    = SYS_MODE;
iprm.frame   = SYS_FRAME;
iprm.count   = SYS_COUNT;
iprm.latency = 2;
mwPlyInitSofdec(&iprm);
mwPlySetVertexBuffer(vbuf);

```

Title 関 数	Function Name	Function	No
	mwPlyFinishSfdFx	ライブラリの終了処理(Kamui、Kamui2 用)	1.5

[書 式] void mwPlyFinishSfdFx(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] ライブラリの終了処理を行います。mwPlyInitSfdFx 関数で切り換えたキャッシュモードを Shinobi のデフォルトに戻します。

Kamui を使用している場合は mwPlyInitSfdFx 関数と対で使用して下さい。

Title 関 数	Function Name	Function	No
	mwPlySetDispMode	表示モードの設定	1.6

[書 式] void mwPlySetDispMode(Sint32 mode, Sint32 frame, Sint32 count, Sint32 latency);

[入 力] mode : 画面モード

frame : フレームバッファのカラーモード

count : フレームカウント数

latency : 表示レイテンシ

[出 力] なし

[関数値] なし

[機 能] 表示モードを設定します。

[注 意] Ninja に設定した表示モードと同じ値を設定してください。

初期化時に MWS_PLY_INIT_SFD で設定した場合は、この関数は必要ありません。

表示モードを変更する場合は、この関数で再設定する必要があります。

[備 考] (1) フレームカウント数について

インタレースの場合、表示更新 VSYNC 数は $2 \times \text{count}$ となります。例えば、インタレースでフレームカウント数 2 なら、4V 表示となります。

1V 表示は、**24fps** など、任意のフレームレートの動画をスムーズに表示します。

1V 表示は、29.97fps の NTSC 用ムービーを **PAL で再生する時に**、スムーズに表示します。

2V 表示は、より多くの CPU 時間を MPEG SofdecF/X に与え、コマ落ちの危険を軽減しますが、スムーズに再生できるフレームレートが限定されます。

(NTSC:29.97fps, VGA:30fps, PAL:25fps)

3V 以上の表示は、動画再生には推奨しません。

(2) レイテンシについて

320×240 を再生する場合、2V レイテンシの時で 500Kbyte、3V の時で 750Kbyte のテクスチャ領域が使用されます。

Title 関 数	Function Name mwPlySetVertexBuffer	Function 頂点バッファの設定	No 1.7
--------------	---------------------------------------	-----------------------	-----------

[書 式] void mwPlySetVertexBuffer (void *vbuf);

[入 力] vbuf : 頂点バッファ

[出 力] なし

[関数値] なし

[機 能] 頂点バッファを設定します。Kamui を使用する場合は、この関数によって描画に使用する頂点バッファを設定しなければなりません。

Ninja を使用している場合は、この関数は必要ありません。

1 4.2 サーバ関数

メインルーチン内で、定期的呼びます。

Title	Function Name	Function	No
関 数	mwExecMainServer	サーバ関数	2.1

[書 式] void mwExecMainServer(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] テクスチャ転送や映像の描画を行います。この関数は、再生ミドルウェアだけでなく、録音ミドルウェアでも使用します。内部モジュールの多重呼び出しを避けるために、共通化してあります。アプリケーションからはゲームフレームに1回だけ呼ぶようにしてください。

Title	Function Name	Function	No
関 数	mwPlyExecTexSvr	テクスチャ転送用サーバ関数(Kamui2 用)	2.2

[書 式] void mwPlyExecTexSvr(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] テクスチャデータをビデオメモリへ転送します。Kamui2 用の関数です。

Title 関 数	Function Name mwPlyExecDrawSvr	Function 映像描画用サーバ関数(Kamui2 用)	No 2.3
--------------	-----------------------------------	----------------------------------	-----------

[書 式] void mwPlyExecDrawSvr(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] 映像の描画を行います。Kamui2 用の関数です。

1 4.3 基本動作処理

動画を単純に再生する上で最低限必要な関数です。

Title 関 数	Function Name	Function	No
	mwPlyCreateSofdec	ハンドルの生成	3.1

[書 式] MWPLY mwPlyCreateSofdec(MWS_PLY_CPRM_SFD *cprm);

[入 力] cprm : ハンドルの生成パラメータ

[出 力] なし

[関数値] ミドルウェア再生ハンドル

[機 能] ハンドルを生成します。テクスチャ領域などのビデオリソースを確保します。

[例] 使用例を以下に示します。

```
memset(&cprm, 0, sizeof(cprm));          /* 予約メンバのゼロ設定のため */
cprm.compo_mode = MWD_PLY_COMPO_OPEQ;    /* 合成モード */
cprm.ftype= MWD_PLY_FTYPE_SFD;           /* 再生ファイルタイプ */
cprm.dtype= MWD_PLY_DTYPE_AUTO;          /* 画質優先 */
cprm.max_bps = 450*1024*8;                /* ビットレート：450 Kbyte/sec */
cprm.max_width = 320;                     /* 画像サイズ：320x480 */
cprm.max_height = 480;
cprm.nfrm_pool_wk = 3;                    /* フレームバッファの枚数 */
cprm.wksize = mwPlyCalcWorkCprmSfd(cprm); /* 作業領域サイズの計算 */
cprm.work = syMalloc(cprm.wksize);        /* 作業領域の確保 */
ply = mwPlyCreateSofdec(&cprm);
```

Title 関 数	Function Name	Function	No
	mwPlyCalcWorkCprmSfd	作業領域サイズの計算	3.2

[書 式] Sint32 mwPlyCalcWorkCprmSfd(MWS_PLY_CPRM_SFD *cprm);
[入 力] cprm : ハンドルの生成パラメータ
[出 力] なし
[関数値] 作業領域サイズ (単位 : バイト)
[機 能] MPEG SofdecF/X の再生に使用する作業領域サイズを計算します。

Title 関 数	Function Name	Function	No
	mwPlyDestroy	ハンドルの消去	3.3

[書 式] void mwPlyDestroy(MWPLY ply);
[入 力] Ply : ミドルウェア再生ハンドル
[出 力] なし
[関数値] なし
[機 能] ハンドルを消去します。

Title	Function Name	Function	No
関 数	mwPlyStartFname	再生開始 (GD-ROM からの再生)	3.4

[書 式] void mwPlyStartFname(MWPLY ply, Sint8 *fname);

[入 力] ply : ミドルウェア再生ハンドル

fname : 再生する動画のファイル名

[出 力] なし

[関数値] なし

[機 能] GD-ROM にある動画の再生を開始します。

Title	Function Name	Function	No
関 数	mwPlyStartSj	再生開始 (ストリームジョイントからの再生)	3.5

[書 式] void mwPlyStartSj(MWPLY ply, SJ sj);

[入 力] ply : ミドルウェア再生ハンドル

sj : ストリームジョイント

[出 力] なし

[関数値] なし

[機 能] ストリームジョイントに供給される動画の再生を開始します。

Title 関 数	Function Name	Function	No
	mwPlyStartMem	再生開始(メモリからの再生)	3.6

[書 式] void mwPlyStartMem(MWPLY ply, void *addr, Sint32 len);

[入 力] ply : ミドルウェア再生ハンドル
addr : 動画データへのポインタ
len : データの長さ

[出 力] なし

[関数値] なし

[機 能] メモリ上の動画の再生を開始します。

Title 関 数	Function Name	Function	No
	mwPlyStop	再生停止	3.7

[書 式] void mwPlyStop(MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] なし

[機 能] 動画の再生を停止します。

Title 関 数	Function Name mwPlyGetStat	Function ハンドルの状態の取得	No 3.8
--------------	-------------------------------	------------------------	-----------

[書 式] MWE_PLY_STAT mwPlyGetStat(MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] ハンドルの内部状態

[機 能] ミドルウェア再生ハンドルの内部状態を取得します。通常、STOP->PREP->PLAYING->PLAYEND に遷移しますが、GD リードエラー等が発生すると、ERROR に遷移します。

定数名	説 明
MWE_PLY_STAT_STOP	停止中
MWE_PLY_STAT_PREP	準備中
MWE_PLY_STAT_PLAYING	再生中
MWE_PLY_STAT_PLAYEND	再生終了
MWE_PLY_STAT_ERROR	エラー状態

Title 関 数	Function Name mwPlyGetTime	Function 再生サンプル数の取得	No 3.9
--------------	-------------------------------	------------------------	-----------

[書 式] void mwPlyGetTime(MWPLY ply, Sint32 *ncount, Sint32 *tscale);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] ncount : 時刻

tscale : 時刻の単位 (単位: ヘルツ)

[関数値] なし

[機 能] 再生時刻を取得します。

[備 考] 実時間(rtime)は、以下の式で計算されます。

$$rtime = ncount / tscale;$$

Title 関 数	Function Name	Function	No
	mwPlyGetInputSj	入力ストリームジョイントの取得	3.10

[書 式] SJ mwPlyGetInputSj (MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] ストリームジョイント

[機 能] デコーダに入力されるデータをバッファリングしているストリームジョイントを取得します。取得したストリームジョイントを使用して mwPlyStartSj 関数で再生することができます。

Title 関 数	Function Name	Function	No
	mwPlyPause	ポーズの設定	3.11

[書 式] void mwPlyPause (MWPLY ply, Sint32 sw);

[入 力] ply : ミドルウェア再生ハンドル

sw : ポーズスイッチ (0 : ポーズ解除、1 : ポーズ)

[出 力] なし

[関数値] なし

[機 能] ポーズの切り換えを行います。

ポーズ中に、もう一度ポーズする (1 を設定する) とコマ送りします。

1 4.4 音声出力制御

音声出力を制御します。

Title	Function Name	Function	No
関 数	mwPlySetOutVol	ボリュームの設定	4.1

[書 式] void mwPlySetOutVol (MWPLY ply, Sint32 vol);

[入 力] ply : ミドルウェア再生ハンドル

vol : ボリューム (-999 ~ 0) (単位 : 1/10dB)

[出 力] なし

[関数値] なし

[機 能] ボリュームを設定します。デフォルトでは0が設定されています。

この値を-999~0まで増加させることにより、フェードインすることができます。

また、反対に0~-999まで減少させることにより、フェードアウトすることができます。

Title	Function Name	Function	No
関 数	mwPlyGetOutVol	ボリュームの取得	4.2

[書 式] Sint32 mwPlyGetOutVol (MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] ボリューム (単位 : 1/10dB)

[機 能] 現在設定されているボリューム値を取得します。

Title 関 数	Function Name	Function	No
	mwPlySetOutPan	パンポットの設定	4.3

[書 式] void mwPlySetOutPan (MWPLY ply, Sint32 chno, Sint32 pan);

[入 力] ply : ミドルウェア再生ハンドル
chno : 設定するチャンネル
モノラル/ステレオ: 左 = MWD_CH_L(0)
ステレオ: 右 = MWD_CH_R(1)
pan : パンポット (-15 ~ 15, -128)
MWD_PAN_LEFT=-15, MWD_PAN_RIGHT=15
MWD_PAN_CENTER=0, MWD_PAN_AUTO=-128

[出 力] なし

[関数値] なし

[機 能] 出力チャンネル毎にパンポットを設定します。
デフォルトでは MWD_PAN_AUTO となり、モノラルの場合は MWD_PAN_CENTER 、
ステレオの場合は左が MWD_PAN_LEFT 、右が MWD_PAN_RIGHT に設定されます。

Title 関 数	Function Name	Function	No
	mwPlyGetOutPan	パンポットの取得	4.4

[書 式] Sint32 mwPlyGetOutPan (MWPLY ply, Sint32 chno);

[入 力] ply : ミドルウェア再生ハンドル
chno : 取得するチャンネル
モノラル/ステレオ: 左 = MWD_CH_L(0)
ステレオ: 右 = MWD_CH_R(1)

[出 力] なし

[関数値] チャンネルに対応したパンポット値 (-15 ~ 15)

[機 能] 現在設定されているパンポット値を取得します。

1 4.5 ウィンドウ制御

映像のウィンドウ表示を制御します。

Title	Function Name	Function	No
関 数	mwPlySetDispPos	表示位置の設定	5.1

[書 式] void mwPlySetDispPos(MWPLY ply, float lx, float ly);

[入 力] ply : ミドルウェア再生ハンドル

lx : X座標

ly : Y座標

[出 力] なし

[関数値] なし

[機 能] 表示位置を設定します。

Title	Function Name	Function	No
関 数	mwPlySetDispSize	表示サイズの設定	5.2

[書 式] void mwPlySetDispSize(MWPLY ply, float sx, float sy);

[入 力] ply : ミドルウェア再生ハンドル

sx : X座標

sy : Y座標

[出 力] なし

[関数値] なし

[機 能] 表示サイズを設定します。

Title 関 数	Function Name mwPlySetBright	Function 輝度の設定	No 5.3
--------------	---------------------------------	-------------------	-----------

[書 式] void mwPlySetBright(MWPLY ply, Sint32 val);

[入 力] ply: ミドルウェア再生ハンドル

val: 輝度 (0 ~ 255)

[出 力] なし

[関数値] なし

[機 能] 輝度を設定します。デフォルトでは、224 が設定されています。

Dreamcast の映像出力は、255 で 110IRE の輝度となるため、デフォルトで 224 を設定しています。素材によってはこの値を調整する必要があります。

Title 関 数	Function Name mwPlyGetBright	Function 輝度の取得	No 5.4
--------------	---------------------------------	-------------------	-----------

[書 式] Sint32 mwPlyGetBright(MWPLY ply);

[入 力] ply: ミドルウェア再生ハンドル

[出 力] なし

[関数値] 輝度

[機 能] 輝度を取得します。

Title 関 数	Function Name mwPlySetBrightOfst	Function 輝度オフセットの設定	No 5.5
--------------	-------------------------------------	------------------------	-----------

[書 式] void mwPlySetBrightOfst(MWPLY ply, Sint32 val);

[入 力] ply: ミドルウェア再生ハンドル
val: 輝度オフセット (0~255)

[出 力] なし

[関数値] なし

[機 能] 輝度オフセットを設定します。デフォルトでは、6 が設定されています。
CG ムービーなどでは、黒がつぶれる傾向があるため、デフォルトで 6 が設定されています。
素材によっては、この値を調整する必要があります。
この値を 255 から徐々に減少させることにより、フェードインすることができます。
また、反対に 255 まで徐々に増加させることにより、フェードアウトすることができます。

Title 関 数	Function Name mwPlyGetBrightOfst	Function 輝度オフセットの取得	No 5.6
--------------	-------------------------------------	------------------------	-----------

[書 式] Sint32 mwPlyGetBrightOfst(MWPLY ply);

[入 力] ply: ミドルウェア再生ハンドル

[出 力] なし

[関数値] 輝度オフセット

[機 能] 輝度オフセットを取得します。

Title 関 数	Function Name	Function	No
	mwPlySetDispZ	表示スクリーンの奥行き値の設定	5.7

[書 式] void mwPlySetDispZ(MWPLY ply, float z);

[入 力] ply: ミドルウェア再生ハンドル

z : 奥行き値 1.0(最手前) ~ 65536.0(最奥)

[出 力] なし

[関数値] なし

[機 能] 表示スクリーンの奥行き値を設定します。

Title 関 数	Function Name	Function	No
	mwPlyGetDispZ	表示スクリーンの奥行き値の取得	5.8

[書 式] float mwPlyGetDispZ(MWPLY ply);

[入 力] ply: ミドルウェア再生ハンドル

[出 力] なし

[関数値] 奥行き値 1.0(最手前) ~ 65536.0(最奥)

[機 能] 設定されている表示スクリーンの奥行き値を取得します。

1 4.6 サーフエス制御

映像のサーフェス表示を制御します。

Title 関 数	Function Name	Function	No
	mwPlyCalcSrfBufSize	サーフェスポイント用バッファサイズの計算	6.1

[書 式] Sint32 mwPlyCalcSrfBufSize(MWPLY ply, Sint32 npnt);

[入 力] ply : ミドルウェア再生ハンドル
npnt : サーフエスポイントの数

[出 力] なし

[関数値] サーフエスポイント用バッファサイズ(単位 : バイト)

[機 能] サーフエスポイント用のバッファサイズを取得します。

Title 関 数	Function Name	Function	No
	mwPlySetSrfPntBuf	サーフェスポイント用バッファの設定	6.2

[書 式] void mwPlySetSrfPntBuf(MWPLY ply, Sint32 npnt, void *buf, Sint32 bsize);

[入 力] ply : ミドルウェア再生ハンドル
npnt : サーフエスポイントの数
buf : バッファ
bsize : バッファサイズ(単位 : バイト)

[出 力] なし

[関数値] なし

[機 能] 指定されたバッファをサーフェスポイント用バッファに設定します。マルチウィンドウ表示を行う場合は必ずこのバッファを割り当てて下さい。

[例] 25 個のウィンドウを表示する場合(100 個のサーフェスポイント)

```
size = mwPlyCalcSrfBufSize(ply, 4*25);  
buf = syMalloc(size);  
mwPlySetSrfPntBuf(ply, 4*25, buf, size);
```


Title 関 数	Function Name mwPlySetSrfPos	Function 表示位置の設定	No 6.3
--------------	---------------------------------	---------------------	-----------

[書 式] void mwPlySetSrfPos(MWPLY ply, Uint32 no, float lx, float ly, float lz);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

lx : X 座標

ly : Y 座標

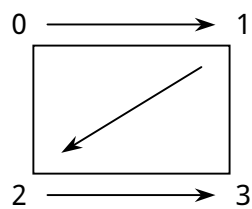
lz : Z 座標(1.0(最手前) ~ 65536.0(最奥))

[出 力] なし

[関数値] なし

[機 能] 表示するサーフェスの位置を各頂点ごとに設定します。

[備 考] 頂点番号の”Z”順とは以下の通りです。



Title 関 数	Function Name mwPlyGetSrfPos	Function 表示位置の取得	No 6.4
--------------	---------------------------------	---------------------	-----------

[書 式] void mwPlyGetSrfPos(MWPLY ply, Uint32 no, float *lx, float *ly, float *lz);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

[出 力] lx : X 座標

ly : Y 座標

lz : Z 座標(1.0(最手前) ~ 65536.0(最奥))

[関数値] なし

[機 能] サーフェスの位置を各頂点ごとに取得します。

Title 関 数	Function Name	Function	No
	mwPlySetSrfBright	輝度の設定	6.5

[書 式] void mwPlySetSrfBright(MWPLY ply, Uint32 no, float a, float r, float g, float b);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

a : アルファ値(0.0 ~ 1.0)

r : 赤の成分値(0.0 ~ 1.0)

g : 緑の成分値(0.0 ~ 1.0)

b : 青の成分値(0.0 ~ 1.0)

[出 力] なし

[関数値] なし

[機 能] 各頂点ごとに輝度を設定します。

Title 関 数	Function Name	Function	No
	mwPlyGetSrfBright	輝度の取得	6.6

[書 式] void mwPlyGetSrfBright(MWPLY ply, Uint32 no, float *a, float *r, float *g, float *b);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

[出 力] a : アルファ値(0.0 ~ 1.0)

r : 赤の成分値(0.0 ~ 1.0)

g : 緑の成分値(0.0 ~ 1.0)

b : 青の成分値(0.0 ~ 1.0)

[関数値] なし

[機 能] 各頂点ごとに輝度値を取得します。

Title 関 数	Function Name	Function	No
	mwPlySetSrfBrightOfst	輝度オフセットの設定	6.7

[書 式] void mwPlySetSrfBrightOfst(MWPLY ply, Uint32 no, float a, float r, float g, float b);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

a : アルファ値(0.0 ~ 1.0)

r : 赤の成分値(0.0 ~ 1.0)

g : 緑の成分値(0.0 ~ 1.0)

b : 青の成分値(0.0 ~ 1.0)

[出 力] なし

[関数値] なし

[機 能] 各頂点ごとに輝度オフセットを設定します。

Title 関 数	Function Name	Function	No
	mwPlyGetSrfBrightOfst	輝度オフセットの取得	6.8

[書 式] void mwPlyGetSrfBrightOfst(MWPLY ply, Uint32 no, float *a, float *r, float *g, float *b);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

[出 力] a : アルファ値(0.0 ~ 1.0)

r : 赤の成分値(0.0 ~ 1.0)

g : 緑の成分値(0.0 ~ 1.0)

b : 青の成分値(0.0 ~ 1.0)

[関数値] なし

[機 能] 各頂点ごとの輝度オフセットを取得します。

Title 関 数	Function Name mwPlySetImgPos	Function イメージ位置の設定	No 6.9
--------------	---------------------------------	-----------------------	-----------

[書 式] void mwPlySetImgPos(MWPLY ply, Uint32 no, float lx, float ly);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

lx : X 座標

ly : Y 座標

[出 力] なし

[関数値] なし

[機 能] イメージの位置を各頂点ごとに設定します。

Title 関 数	Function Name mwPlyGetImgPos	Function イメージ位置の取得	No 6.10
--------------	---------------------------------	-----------------------	------------

[書 式] void mwPlyGetImgPos(MWPLY ply, Uint32 no, float *lx, float *ly);

[入 力] ply : ミドルウェア再生ハンドル

no : 頂点番号(“Z”順に 0 ~ 3)

[出 力] lx : X 座標

ly : Y 座標

[関数値] なし

[機 能] 各頂点ごとのイメージ位置を取得します。

Title 関 数	Function Name mwPlyGetImgSize	Function イメージサイズの取得	No 6.11
--------------	----------------------------------	------------------------	------------

[書 式] void mwPlyGetImgSize(MWPLY ply, Sint32 *isx, Sint32 *isy);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] isx : X座標

isy : Y座標

[関数値] なし

[機 能] イメージのサイズを取得します。

14.7 エフェクト

エフェクトムービー再生時に使用する関数です。

Title 関 数	Function Name	Function	No
	mwPlySetCompoMode	合成モードの設定	7.1

- [書 式] void mwPlySetCompoMode(MWPLY ply, Sint32 mode);
- [入 力] ply : ミドルウェア再生ハンドル
mode : 合成モード (MWD_PLY_COMPO ~)
- [出 力] なし
- [関数値] なし
- [機 能] 合成モードを設定します。合成モードを再生の途中で切り換える時に使用します。
合成モードの切り換えは、mwPlyEntryFxCb 関数で登録したコールバック関数が実行される
タイミングで行います。

Title 関 数	Function Name	Function	No
	mwPlyEntryFxCb	エフェクトコールバック関数の登録	7.2

- [書 式] void mwPlyEntryFxCb(MWPLY ply,
void (*fn)(void *obj, Sint32 fno, Sint32 time, Sint32 tunit), void *obj);
- [入 力] ply : ミドルウェア再生ハンドル
fn : コールバック関数
obj : コールバック関数の第 1 引数
- [出 力] なし
- [関数値] なし
- [機 能] 合成処理を行う前に実行する関数を登録します。このタイミングで、合成モードを
変更してください。 コールバック関数の引数は以下の通りです。
fno : ムービーデータのフレーム通し番号 (0 が先頭)
time : 再生時刻 (time/tunit で実時間になります)
tunit: 再生時刻の単位 (Hz)
- [例] 使用例を以下に示します。
- ```
void user_chg_compo(void *obj, Sint32 fno, Sint32 time, Sint32 tunit)
{
 MWPLY ply=(MWPLY)obj;

 if (fno == 0) {
 mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_TRNSP);
 } else if (fno == 1*8*30) {
 mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_LUMI);
 } else if (fno == 2*8*30) {
 mwPlySetCompoMode(ply, compo_mode=MWD_PLY_COMPO_ALPH3);
 }
}

mwPlyEntryFxCb(ply, user_chg_compo, (void *)ply);
```

|              |                                  |                               |           |
|--------------|----------------------------------|-------------------------------|-----------|
| Title<br>関 数 | Function Name<br>mwPlySetAlphTbl | Function<br>輝度から 値への変換テーブルの設定 | No<br>7.5 |
|--------------|----------------------------------|-------------------------------|-----------|

[ 書 式 ] void mwPlySetAlphTbl(MWPLY ply, long bno, float alph[256]);

[ 入 力 ] ply : ミドルウェア再生ハンドル

bno : バンク番号

alph : アルファテーブル

[ 出 力 ] なし

[ 関数値 ] なし

[ 機 能 ] 輝度から 値への変換テーブルを設定します。

[ 例 ] 輝度 Y < 128 : = 0.0

輝度 Y 128 : = 1.0 の場合

```

for (y=0; y<128; y++) {
 alph[y] = 0.0;
}
for (; y<256; y++) {
 alph[y] = 1.0;
}
mwPlySetAlphTbl(ply, 0, alph);

```

#### 14.8 テクスチャムービー

テクスチャムービーに使用する関数です。

| Title | Function Name | Function    | No  |
|-------|---------------|-------------|-----|
| 関 数   | mwPlyGetMvFrm | ムービーフレームの取得 | 8.1 |

[書 式] Sint32 mwPlyGetMvFrm(MWPLY ply, MWS\_PLY\_MVFRM \*frm);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] frm : 映像サーフェス情報

[関数値] ムービーフレームを取得できたか否か (1: 取得成功、0: 取得失敗)

[機 能] ムービーフレームを取得します。

テクスチャムービーを行う時に、この関数によって得られたムービーフレームのサーフェスを使用して、ポリゴンを描画して下さい。Ninja を使用している場合は、このサーフェスをテクスチャリストに設定して下さい。

| Title | Function Name  | Function   | No  |
|-------|----------------|------------|-----|
| 関 数   | mwPlyGetMskFrm | マスクフレームの取得 | 8.2 |

[書 式] Sint32 mwPlyGetMskFrm(MWPLY ply, MWS\_PLY\_MVFRM \*frm);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] frm : 映像サーフェス情報

[関数値] マスクフレームを取得できたか否か (1: 取得成功、0: 取得失敗)

[機 能] マスクフレームを取得します。

テクスチャムービーを行う時に、この関数によって得られたマスクフレームのサーフェスを使用して、ポリゴンを描画して下さい。Ninja を使用している場合は、このサーフェスをテクスチャリストに設定して下さい。



#### 1 4.9 シームレス連続再生

シームレス連続再生用の関数です。

| Title | Function Name   | Function         | No  |
|-------|-----------------|------------------|-----|
| 関 数   | mwPlyEntryFname | シームレス連続再生ファイルの登録 | 9.1 |

[書 式] void mwPlyEntryFname(MWPLY ply, char \*fname);

[入 力] ply : ミドルウェア再生ハンドル

fname : ファイル名

[出 力] なし

[関数値] なし

[機 能] シームレス連続再生するファイルを登録します。

シームレス連続再生は、MPV(映像)形式のみ可能です。

| Title | Function Name      | Function     | No  |
|-------|--------------------|--------------|-----|
| 関 数   | mwPlyStartSeamless | シームレス連続再生の開始 | 9.2 |

[書 式] void mwPlyStartSeamless(MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] なし

[機 能] 登録されているファイルのシームレス連続再生を開始します。

シームレス連続再生は、MPV(映像)形式のみ可能です。

| Title<br>関 数 | Function Name        | Function     | No  |
|--------------|----------------------|--------------|-----|
|              | mwPlyReleaseSeamless | シームレス連続再生の解除 | 9.3 |

[書 式] void mwPlyReleaseSeamless(MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] なし

[機 能] シームレス連続再生を解除します。現在登録されているファイルを再生後、再生終了状態になります。

| Title<br>関 数 | Function Name | Function      | No  |
|--------------|---------------|---------------|-----|
|              | mwPlySetLpFlg | シームレスループ再生の設定 | 9.4 |

[書 式] void mwPlySetLpFlg(MWPLY ply, long flg);

[入 力] ply : ミドルウェア再生ハンドル

flg : ループ再生フラグ (1: ループ、0: ループ解除)

[出 力] なし

[関数値] なし

[機 能] ループ再生フラグを ON(1)にすることにより、登録されているすべてのファイルをシームレスにループ再生します。

|              |                                    |                           |           |
|--------------|------------------------------------|---------------------------|-----------|
| Title<br>関 数 | Function Name<br>mwPlyStartFnameLp | Function<br>シームレスループ再生の開始 | No<br>9.5 |
|--------------|------------------------------------|---------------------------|-----------|

[書 式] void mwPlyStartFnameLp(MWPLY ply, char \*fname);

[入 力] ply : ミドルウェア再生ハンドル

fname : ファイル名

[出 力] なし

[関数値] なし

[機 能] 指定されたファイルをシームレスループ再生します。  
シームレスループ再生は、MPV(映像)形式のみ可能です。

#### 14.10 再生モードの設定

再生モードを設定します。

| Title | Function Name       | Function     | No   |
|-------|---------------------|--------------|------|
| 関 数   | mwPlySetFastHalfpel | 高速ハーフペル処理の設定 | 10.1 |

[書 式] void mwPlySetFastHalfpel(MWPLY ply, Sint32 sw);

[入 力] ply : ミドルウェア再生ハンドル

sw : 高速ハーフペルスイッチ (0 : 高速処理しない、1 : 高速処理する)

[出 力] なし

[関数値] なし

[機 能] 高速ハーフペルの処理を設定します。

[備 考] 高速ハーフペルに設定した場合、CPU 負荷は軽減しますが、多少画質が劣化します。

| Title | Function Name   | Function    | No   |
|-------|-----------------|-------------|------|
| 関 数   | mwPlySetAudioSw | 音声出力スイッチの設定 | 10.2 |

[書 式] void mwPlySetAudioSw(MWPLY ply, Sint32 sw);

[入 力] ply : ミドルウェア再生ハンドル

sw : 音声出力スイッチ (0 : 音声出力しない、1 : 音声出力する)

[出 力] なし

[関数値] なし

[機 能] 音声出力スイッチを設定します。

[備 考] デフォルトでは、音声を出力します。

音声付き動画ファイルでも映像のみを再生することができます。

| Title | Function Name   | Function    | No   |
|-------|-----------------|-------------|------|
| 関 数   | mwPlySetVideoSw | 映像表示スイッチの設定 | 10.3 |

[書 式] void mwPlySetVideoSw(MWPLY ply, Sint32 sw);

[入 力] ply : ミドルウェア再生ハンドル

sw : 映像表示スイッチ (0: 映像を表示しない、1: 映像を表示する)

[出 力] なし

[関数値] なし

[機 能] 映像表示のスイッチを設定します。テクスチャムービーとして表示する場合は、映像スイッチに 0 を指定して下さい。

#### 1 4.1 1 内部の状態取得

ミドルウェア内部の状態を取得するデバッグ用の関数です。

| Title | Function Name      | Function       | No   |
|-------|--------------------|----------------|------|
| 関 数   | mwPlyGetNumDropFrm | コマ落ちしたフレーム数の取得 | 11.1 |

[ 書 式 ] Sint32 mwPlyGetNumDropFrm(MWPLY ply);

[ 入 力 ] ply : ミドルウェア再生ハンドル

[ 出 力 ] なし

[ 関数値 ] フレーム数

[ 機 能 ] コマ落ちしたフレーム数を取得します。

表示モードがインターレースで、29.97、30fps の動画の時のみ有効です。

| Title | Function Name      | Function       | No   |
|-------|--------------------|----------------|------|
| 関 数   | mwPlyGetNumDecPool | デコード済みフレーム数の取得 | 11.2 |

[ 書 式 ] Sint32 mwPlyGetNumDecPool(MWPLY ply);

[ 入 力 ] ply : ミドルウェア再生ハンドル

[ 出 力 ] なし

[ 関数値 ] デコード済みフレーム数

[ 機 能 ] デコード済みフレーム数を取得します。コマ落ちが発生する場合は、必ずこの値が0になります。

| Title<br>関 数 | Function Name       | Function        | No   |
|--------------|---------------------|-----------------|------|
|              | mwPlyGetNumTotalDec | デコードした全ピクチャ数の取得 | 11.3 |

[書 式] Sint32 mwPlyGetNumTotalDec(MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] デコードした全ピクチャ数

[機 能] デコードした全ピクチャ数を取得します。

| Title<br>関 数 | Function Name        | Function        | No   |
|--------------|----------------------|-----------------|------|
|              | mwPlyGetNumTotalSkip | スキップした全ピクチャ数の取得 | 11.4 |

[書 式] Sint32 mwPlyGetNumTotalSkip(MWPLY ply);

[入 力] ply : ミドルウェア再生ハンドル

[出 力] なし

[関数値] スキップした全ピクチャ数

[機 能] スキップした全ピクチャ数を取得します。

| Title<br>関 数 | Function Name       | Function       | No   |
|--------------|---------------------|----------------|------|
|              | mwPlyGetNumIbufRoom | 入力バッファの空き容量の取得 | 11.5 |

[ 書 式 ] Sint32 mwPlyGetNumIbufRoom(MWPLY ply);

[ 入 力 ] ply : ミドルウェア再生ハンドル

[ 出 力 ] なし

[ 関数値 ] 入力バッファの空き容量 (単位: バイト)

[ 機 能 ] 入力バッファの空き容量を取得します。

| Title<br>関 数 | Function Name       | Function     | No   |
|--------------|---------------------|--------------|------|
|              | mwPlyGetNumDispWait | 表示待ちフレーム数の取得 | 11.6 |

[ 書 式 ] Sint32 mwPlyGetNumDispWait(MWPLY ply);

[ 入 力 ] ply : ミドルウェア再生ハンドル

[ 出 力 ] なし

[ 関数値 ] 表示待ちフレーム数

[ 機 能 ] 表示待ちフレーム数を取得します。



|              |                                  |                              |            |
|--------------|----------------------------------|------------------------------|------------|
| Title<br>関 数 | Function Name<br>mwPlyGetNumRend | Function<br>レンダリング中のフレーム数の取得 | No<br>11.7 |
|--------------|----------------------------------|------------------------------|------------|

[書 式] Sint32 mwPlyGetNumRend(MWPLY ply);  
 [入 力] ply : ミドルウェア再生ハンドル  
 [出 力] なし  
 [関数値] レンダリング中のフレーム数  
 [機 能] レンダリング中のフレーム数を取得します。

|              |                                     |                                   |            |
|--------------|-------------------------------------|-----------------------------------|------------|
| Title<br>関 数 | Function Name<br>mwPlyGetNumLoadFrm | Function<br>テクスチャメモリへ転送したフレーム数の取得 | No<br>11.8 |
|--------------|-------------------------------------|-----------------------------------|------------|

[書 式] Sint32 mwPlyGetNumLoadFrm(MWPLY ply);  
 [入 力] ply : ミドルウェア再生ハンドル  
 [出 力] なし  
 [関数値] テクスチャメモリへ転送したフレーム数  
 [機 能] テクスチャメモリへ転送したフレーム数を取得します。

#### 1 4.1 2 エラー処理

エラー処理を行います。

| Title | Function Name     | Function         | No   |
|-------|-------------------|------------------|------|
| 関 数   | mwPlyEntryErrFunc | エラー発生時に呼ばれる関数の登録 | 12.1 |

[ 書 式 ] void mwPlyEntryErrFunc(MW\_PLY\_ERRFN errfn, void \*obj);

[ 入 力 ] errfn :エラー関数

obj :エラーオブジェクト

[ 出 カ ]

[ 関数値 ] なし

[ 機 能 ] エラー発生時に呼ばれる関数を登録します。

[ 備 考 ] エラー関数を登録すると、エラーが発生と共に、登録した関数が呼び出されます。

エラー関数は、第 2 引数に文字列が渡される。この文字列によりエラーの内容が確認できます。

[ 例 ] 使用例を以下に示します。

```
/* ユーザエラー関数 */
void user_error(void *user_obj, char *msg)
{
 for (;;) {
 njPrintC(NJM_LOCATION(3, 3), msg);
 }
}

void main(void)
{
 mwPlyEntryErrFunc(user_error, user_obj);
}
```