ゲームプログラミング

▶　DirectDraw－第４回　サーフェイス　◀

DirectDrawでは、画像を保持しておく領域をサーフェイスと呼びます。サーフェイスはいくつか種類があり、特性や性質をよく考えて使用しないと動作が遅くなってしまいます。

サーフェイス

サーフェイスとは、DirectDrawが扱う画像を保持しておくメモリ領域です。ディスクに保存されている画像は、DirectDrawが直接扱うことができません。また、画像が必要になるたびにディスクから読み込んでいたのでは、動作がかなり遅くなってしまいます。DirectDrawが扱う画像は、サーフェイスに格納しておく必要があります。
サーフェイスは、グラフィックカード上のメモリ(VRAM)か、システムメモリに作成することができます。どこに作成しても同じメソッドを使用することができますが、性能が若干異なります。

VRAMに作成したサーフェイス
VRAMに作成したサーフェイスは、以下のような特徴があります。

- 基本的にHALで動作するため高速
- Bltメソッドの拡大および縮小でアンチエイリアスがかかる場合がある
- VRAMに作成したほかのサーフェイスへの転送はもっとも高速
- 同一サーフェイスに対する連続描画は遅くなることが多い
- 非同期転送が可能(画像転送時、転送完了を待たずに次の命令を実行することが可能)。ただし、同一サーフェイス間では非同期転送できない場合がある
- ポインタを取得しての直接的なサーフェイスへのアクセスはかなり遅い
- アプリケーションが非アクティブ状態になるとロストする

システムメモリに作成したサーフェイス
システムメモリに作成したサーフェイスは、以下のような特徴があります。

- HELで動作する
- HALの動作が完了しないとHELは動作できないため、VRAM同士とシステムメモリの転送を交互に実行するような動作はかなり遅くなる
- ポインタを取得しての直接的なサーフェイスへのアクセスは、かなり速い
- アプリケーションが非アクティブ状態になってもロストしない
- PageLockメソッドでページロックすることができる。ページロックされているシステムメモリ同士の転送は、ハードウェア(DMA)転送されるようになり高速化され、非同期転送も可能になる

サーフェイスには、目的別に「プライマリサーフェイス」「バックバッファ」「オフスクリーンサーフェイス」があります。

プライマリサーフェイス
プライマリサーフェイスは、ディスプレイに表示される画像を保持するサーフェイスです。このサーフェイスは必ずVRAMに作成されます。VRAMの容量を超えるサーフェイスや、ディスプレイモードと異なるフォーマットのサーフェイスは生成できません。
プライマリサーフェイスのピクセルを変更すると、ディスプレイに表示されているピクセルも直ちに変更されます。

バックバッファ
バックバッファは、プライマリサーフェイスと関連づけられている(アタッチされている)サーフェイスで、プライマリサーフェイスと同一の性能を持っています。バックバッファは、画面に表示されることのない、プライマリサーフェイスの裏側です。ディスプレイと同期してプライマリサーフェイスと瞬時に入れ換えることができるので、ちらつきのない描画を実現するために使用されます。
このサーフェイスは、VRAMに作成できないときはシステムメモリに作成されます。

オフスクリーンサーフェイス
オフスクリーンサーフェイスは、背景やキャラクターなどの画像を保持しておく領域です。このサーフェイスも画面に表示されません。バックバッファと違い、プライマリサーフェイスから完全に独立しており、サイズやピクセルフォーマットを自由に指定して作成することができます。単純に画像を保持したり、画像データを加工するためのサーフェイスです。

サーフェイスの生成

サーフェイスの生成は、DDSURFACEDESC2構造体に生成するサーフェイスの情報を設定し、この構造体をDirectDrawオブジェクトのCreateSurfaceメソッドに渡すことで行います。DDSURFACEDESC2構造体はたくさんのメンバがありますが、サーフェイスを生成するときに設定するのは、以下のメンバです。

メンバ	データ型	機　能	対応するフラグ
dwSize	DWORD	この構造体のサイズ(バイト数)	－
dwFlags	DWORD	有効な(値を設定した)メンバを示すフラグ	－
dwWidth	DWORD	サーフェイスの幅	DDSD_WIDHT
dwHeight	DWORD	サーフェイスの高さ	DDSD_HEIGHT
dwBackBufferCount	DWORD	バックバッファの数	DDSD_BACKBUFFERCOUNT
ddsCaps	DDSCAPS2	サーフェイスの機能	DDSD_CAPS

CreateSurfaceメソッド
－説明－
CreateSurfaceメソッドは、サーフェイスを管理するDirectDrawSurfaceオブジェクトを作成し、そのインタフェースを取得します。

－パラメータ－
１つ目の引数は、生成するサーフェイスの情報が格納されたDDSURFACEDESC2構造体変数のアドレスです。

２つ目の引数は、生成されるオブジェクトのインタフェースを受け取る変数のアドレスです。LPDIRECTDRAWSURFACE7型の変数のアドレスを指定します。

３つ目の引数は、将来の互換性のためにあり、現時点ではNULLにしないとエラーになります。

－戻り値－
成功した場合はDD_OK、それ以外はエラーの原因をエラーコードで返します。

プライマリサーフェイスとバックバッファの生成

プライマリサーフェイスとバックバッファは一緒に作成されます(フルスクリーンモードの場合)。これらを作成するときは、DDSURFACEDESC2構造体のdwFlagメンバにDDSD_CAPSフラグとDDSD_BACKBUFFERCOUNTフラグを指定します。dwBackBufferCountメンバは生成するバックバッファの数で、１(ダブルバッファ)または２(トリプルバッファ)が一般的です。ほとんどの場合は１で十分です。ddsCapsメンバはDDSCAPS2構造体の変数で、サーフェイスの機能を定義します。ほとんどの場合は、この構造体のdwCapsメンバだけを使用します。プライマリサーフェイスとバックバッファを生成する場合は、以下の３つのフラグを指定します。

DDSCAPS_PRIMARYSURFACE…プライマリサーフェイスを表します。プライマリサーフェイスを生成する場合は必ず指定します
DDSCAPS_FLIP…プライマリサーフェイスとバックバッファがフリップ可能であることを表します。ウィンドウモードでは使用できません
DDSCAPS_COMPLEX…DDSCAPS_FLIPと一緒に使うフラグで、プライマリサーフェイスとバックバッファが複合サーフェイス(関連づけられているサーフェイス)であることを表します

プライマリサーフェイスは必ずディスプレイモードと同じサイズになるため、dwWidthメンバとdwHeightメンバを指定する必要はありません。
以上のメンバの設定が終わったら、CreateSurfaceメソッドを呼び出します。

```
DDSURFACEDESC2    ddsd2;
ZeroMemory(&ddsd2, sizeof(ddsd2));
ddsd2.dwSize            = sizeof(ddsd2);
ddsd2.dwFlags           = DDSD_CAPS | DDSD_BACKBUFFERCOUNT;
ddsd2.ddsCaps.dwCaps    = DDSCAPS_PRIMARYSURFACE | DDSCAPS_FLIP | DDSCAPS_COMPLEX;
ddsd2.dwBackBufferCount = 1;

// プライマリサーフェイス生成
LPDIRECTDRAWSURFACE7    lpDDSPrimary;
lpDDraw7->CreateSurface(&ddsd2, &lpDDSPrimary, NULL); // lpDDraw7はDirectDraw7オブジェクト
```

これで、プライマリサーフェイスとバックバッファが生成されます。しかし、バックバッファオブジェクトのインタフェースを取得していないので、バックバッファへの操作が行えません。このインタフェースを取得するには、プライマリサーフェイスオブジェクトのGetAttachedSurfaceメソッドを呼び出します。DDSCAPS2構造体のdwCapsメンバにDDSCAPS_BACKBUFFERフラグを指定し、この構造体をGetAttac

hedSurfaceメソッドに渡します。このメソッドは、１つ目の引数にDDSCPAS2構造体変数のアドレス、２つ目の引数にバックバッファオブジェクトのインタフェースを格納する変数のアドレスを指定します。

```
// バックバッファ取得
LPDIRECTDRAWSURFACE7   lpDDSBackBuffer;
DDSCAPS2    ddsc2;
ZeroMemory(&ddsc2, sizeof(ddsc2));
ddsc2.dwCaps = DDSCAPS_BACKBUFFER;
lpDDSPrimary->GetAttachedSurface(&ddsc2, &lpDDSBackbuffer);
```

ウィンドウモードの場合はフリップできないため、バックバッファをプライマリサーフェイスと同時に生成することはできません。バックバッファはオフスクリーンサーフェイスで作成します。

オフスクリーンサーフェイスの生成

オフスクリーンサーフェイスを生成する手順は、プライマリサーフェイスとほとんど同じです。
DDSURFACEDESC2構造体のメンバddsCpas構造体のdwCapsメンバに、DDSCAPS_OFFSCREENPLAINフラグを指定します。このとき、VRAMとシステムメモリのどちらに生成するかを指定できます。VRAMに作成したい場合はDDSCAPS_VIDEOMEMORYフラグ、システムメモリに作成したい場合はDDSCAPS_SYSTEMMEMORYフラグを指定します。どちらのフラグも指定しない場合は、まずVRAMに作成され、空きがない場合はシステムメモリに生成されます。
また、必ず幅と高さを指定しなければならないので、dwWidthメンバとdwHeightメンバの設定も行います。ここには自由な値を設定できますが、生成できるサイズはグラフィックカードにより異なります。
dwFlagsメンバは、ddsCpasメンバ、dwWidthメンバ、dwHeightメンバの設定をしているので、DDSD_CAPSフラグ，DDSD_WIDTHフラグ，DDSD_HEIGHTフラグを指定します。
以上のメンバの設定が終わったら、CreateSurfaceメソッドを呼び出します。

```
LPDIRECTDRAWSURFACE7   lpDDSOffscreen;
DDSURFACEDESC2    ddsd2;
ZeroMemory(&ddsd2, sizeof(ddsd2));
ddsd2.dwSize         = sizeof(ddsd2);
ddsd2.dwFlags        = DDSD_CAPS | DDSD_WIDTH | DDSD_HEIGHT;
ddsd2.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN | DDSCAPS_VIDEOMEMORY;    // ビデオメモリに作成
ddsd2.dwWidth        = 320;
ddsd2.dwHeight       = 240;
lpDDraw7->CreateSurface(&ddsd2, &lpDDSOffscreen, NULL);
```

サーフェイスの解放

生成したサーフェイスは、DirectDrawオブジェクトを解放する前に解放しておく必要があります。解放はDirectDrawSurfaceオブジェクトのReleaseメソッドで行いますが、サーフェイスの種類により解放する手順が異なります。
プライマリサーフェイスとバックバッファは複合サーフェイスなので、プライマリサーフェイスを解放するとバックバッファも解放されます。ただし、GetAttachedSurfaceメソッドでバックバッファを取得している場合は解放する必要があります。DirectDraw7では、バックバッファを解放する場合、プライマリサーフェイスより先に解放しないと一般保護エラーが発生する場合があります。
オフスクリーンサーフェイスはReleaseメソッドを呼び出すだけで解放されます。

```
// オフスクリーンサーフェイス解放
if(NULL != lpDDSOffscreen) {
    lpDDSOffscreen->Release();
    lpDDSOffscreen = NULL;
}

// バックバッファ解放
if(NULL != lpDDSBackBuffer) {
    lpDDSBackBuffer->Release();
    lpDDSBackBuffer = NULL;
}

// プライマリサーフェイス解放
if(NULL != lpDDSPrimary) {
    lpDDSPrimary->Release();
    lpDDSPrimary = NULL;
}
```

解放処理は、生成した順番と逆にするともっとも安全に行うことができます。

○ 課題 ○

プライマリサーフェイス、バックバッファ、オフスクリーンサーフェイスを生成しましょう。

(１)DDUtilsでは、サーフェイスオブジェクトのインタフェースを配列にまとめて扱います。配列で管理するのは、サーフェイスごとに変数を与える方法、たとえば以下のように

```
LPDIRECTDRAWSURFACE7   lpDDSPrimary;     // プライマリサーフェイス
LPDIRECTDRAWSURFACE7   lpDDSBackbuf;     // バックバッファ
LPDIRECTDRAWSURFACE7   lpDDSOffscreen1;  // オフスクリーンサーフェイス１
LPDIRECTDRAWSURFACE7   lpDDSOffscreen2;  // オフスクリーンサーフェイス２
```

とするよりも利点が多いためです。たとえば、すべてのサーフェイス対して同じ操作を行いたい場合、配列ならループを回すだけで処理することができます。しかし、たとえば

```
LPDIRECTDRAWSURFACE7   lpDDSurface[6];
```

と宣言した場合、lpDDSurface[0]はプライマリなのかバックバッファなのか、それともオフスクリーンサーフェイスなのか、ソースをよく解析しないとわかりにくくなります。そこで列挙体を利用します。
DDUtils.hにサーフェイスを指定するための列挙体を定義します。これを添字として使用します。

－追加１－

```
enum DDSRFC {
    DDS_PRIMARY,          // プライマリサーフェイス
    DDS_BACKBUF,          // バックバッファ
    DDS_DIRECTSHOW,       // DirectShowレンダリングサーフェイス
    DDS_OFFSCRN1,         // オフスクリーンサーフェイス１
    DDS_OFFSCRN2,         // オフスクリーンサーフェイス２
    DDS_OFFSCRN3,         // オフスクリーンサーフェイス３
    DDS_OFFSCRN4,         // オフスクリーンサーフェイス４
    DDS_TEXTURE1,         // テクスチャ１
    DDS_TEXTURE2,         // テクスチャ２
    DDS_TEXTURE3,         // テクスチャ３
    DDS_TEXTURE4,         // テクスチャ４
    DDS_MAX               // サーフェイス最大数
};
```

上記の定義に過不足がある場合は、各自調整してください。次に、DDUtils.cppのグローバル変数として、サーフェイスオブジェクトのインタフェースを配列で定義します。

－追加２－

```
static LPDIRECTDRAWSURFACE7   g_lpDDSurface7[DDS_MAX] = {NULL};
```

このように定義した場合、たとえばプライマリサーフェイスを指定するには

```
g_lpDDSurface7[DDS_PRIMARY]
```

と記述します。

```
g_lpDDSurface7[0]
```

よりタイプする文字数は増えますが、ソースを見たときのわかりやすさには雲泥の差があります。

(２)プライマリサーフェイスとバックバッファは必須の機能なので、DirectDraw初期化時に生成するようにします。以下のプログラムを完成させ、適切な場所に追加しましょう。

－追加３－

```
    // プライマリサーフェイス生成
    DDSURFACEDESC2    ddsd2;
    ZeroMemory(&ddsd2, sizeof(ddsd2));
    ddsd2.dwSize            = ※ここは各自考えましょう
    ddsd2.dwFlags           = ※ここは各自考えましょう
    ddsd2.ddsCaps.dwCaps    = ※ここは各自考えましょう
    ddsd2.dwBackBufferCount = ※ここは各自考えましょう
    if(※ここは各自考えましょう){
        OutputDebugString("*** Error - プライマリサーフェイス生成失敗(DDInit)\n");
        DDRelease();
        return false;
    }

    // バックバッファ取得
    DDSCAPS2    ddsc2;
    ZeroMemory(&ddsc2, sizeof(ddsc2));
    ddsc2.dwCaps = ※ここは各自考えましょう
    if(※ここは各自考えましょう) {
```

```
        OutputDebugString("*** Error - バックバッファ取得失敗(DDInit)\n");
        DDRelease();
        return false;
    }
```

(３)プライマリサーフェイスとバックバッファを解放する処理を適切な場所に追加しましょう。

(４)以下のプログラムは、オフスクリーンサーフェイスを生成するDDCreateSurface関数です。関数の仕様をよく読み、プログラムを完成させましょう。なお、この関数は(５)で作成する関数も必要です。

DDCreateSurface関数
－説明－
DDCreateSurface関数は、オフスクリーンサーフェイスを生成します。指定されたサーフェイスがすでに生成されている場合は、エラーとせずに解放したあとに生成します。

－パラメータ－
const DDSRFC dds…生成するサーフェイスの指定。DDSRFC列挙体型を指定
const DWORD dwWidht…生成するサーフェイスの幅
const DWORD dwHeight…生成するサーフェイスの高さ
const DWORD dwCaps…サーフェイスを生成する場所。以下のフラグを１つ指定します
0　VRAMに生成、空きがなければシステムメモリに生成
DDSCAPS_VIDEOMEMORY　VRAMに生成
DDSCAPS_SYSTEMMEMORY　システムメモリに生成

－戻り値－
生成に成功した場合はtrue，生成に失敗したり、オフスクリーンサーフェイス以外のサーフェイスを指定した場合はfalseを返します。

－追加４－
```
/*****************************************************************************************************/
/*                                          サーフェイス生成                                         */
/*****************************************************************************************************/
bool DDCreateSurface(const DDSRFC dds, const DWORD dwWidth, const DWORD dwHeight, const DWORD dwCaps)
{
#ifdef _DEBUG
    if(NULL == g_lpDDraw7) {
        OutputDebugString("*** Error - DirectDraw未初期化(DDCreateSurface)\n");
        return false;
    }
    if(DDS_BACKBUF >= dds || DDS_TEXTURE1 <= dds) {
        OutputDebugString("*** Error - オフスクリーン以外のサーフェイスを指定(DDCreateSurface)\n");
        return false;
    }
#endif
    DDReleaseSurface(dds);  // サーフェイス初期化

    DDSURFACEDESC2   ddsd2;
    ZeroMemory(&ddsd2, sizeof(ddsd2));
    ddsd2.dwSize         = sizeof(ddsd2);
    ddsd2.dwFlags        = ※ここは各自考えましょう
    ddsd2.dwWidth        = ※ここは各自考えましょう
    ddsd2.dwHeight       = ※ここは各自考えましょう
    ddsd2.ddsCaps.dwCaps = ※ここは各自考えましょう。ただし、必ずdwCaps変数も使用すること
    if(※ここは各自考えましょう) {
        OutputDebugString("*** Error - サーフェイス生成失敗(DDCreateSurface)\n");
        return false;
    }

    return true;
}
```

(５)サーフェイスを解放するDDReleaseSurface関数の仕様を参考に、関数を完成させましょう。

DDReleaseSurface関数
－説明－
DDReleaseSurface関数は、プライマリとバックバッファ以外の指定されたサーフェイスを解放します。

－パラメータ－
const DDSRFC dds…解放するサーフェイスの指定。DDSRFC列挙体型を指定

－戻り値－
なし

－追加５－
```
/*********************************************************************************************************/
/*                                          サーフェイス解放                                             */
/*********************************************************************************************************/
void DDReleaseSurface(const DDSRFC dds)
{
#ifdef _DEBUG
    if(DDS_BACKBUF >= dds) {
        OutputDebugString("*** Error - プライマリ, バックバッファを指定(DDReleaseSurface)\n");
        return;
    }
#endif
    if(NULL != g_lpDDSurface7[dds]) {
        ※ここは各自考えましょう
        ※ここは各自考えましょう
    }
}
```

(６)以下のDDReleaseAllSurfaces関数は、プライマリサーフェイスとバックバッファ以外のすべてのサーフェイスを解放する関数です。不足部分を補って関数を完成させましょう。

－追加６－
```
/*********************************************************************************************************/
/*                                          全サーフェイス解放                                           */
/*********************************************************************************************************/
void DDReleaseAllSurfaces()
{
    for(int dds = DDS_OFFSCRN1; dds < DDS_MAX; dds++)
        ????????????????((DDSRFC)dds);
}
```

この関数は、シーンチェンジのようなすべてのオフスクリーンサーフェイスを解放したいときに使用すると便利です。

(７)－追加７－を適切な場所に追加し、DirectDraw解放時にすべてのサーフェイスが正しく解放されるようにしましょう。

－追加７－
```
    // すべてのオフスクリーンサーフェイスを解放
    DDReleaseAllSurfaces();
```