ゲームプログラミング

▶　DirectDraw－第１７回　ビットマップの復元　◀

DirectDrawでは、サーフェイスメモリの復元時に、もともと保持していたビットマップを復元する方法を提供していません。保持していたビットマップは、再構築しなければなりません。

ビットマップの復元

サーフェイスメモリがロストした場合、Restoreメソッドで復元します。このとき、DirectDrawはサーフェイスメモリの復元はしますが、もともと保持していたビットマップは復元しません。しかし、ビットマップを復元しないと処理が再開できない場合がほとんどです。
保持していたビットマップを復元するには、残念ながら再構築するしかありません。ビットマップを再度読み込み、ビットマップを加工していた場合は再加工して元の状態まで戻します。こうすることで復元します。ロストさせたくないサーフェイスは、システムメモリに生成するようにします。

○ 課題 ○

サーフェイスの復元時に、もともと保持していたビットマップを復元する処理(ビットマップファイルの再読み込み)を追加しましょう。

(１)サーフェイスに読み込まれているビットマップファイル名を保存しておく領域を設けます。サーフェイスの復元時に、このファイル名を使って再読み込みします。以下のプログラムを適切な場所に追加しましょう。

－追加１－
```
#include <string>
using namespace std;
```

－追加２－
```
static string   g_strLoadFile[DDS_MAX];       // 読み込みファイル名(サーフェイス復元時に使用)
```

※Ｃ言語での文字列はchar型配列を使用しますが、基本的に固定配列なので文字数によっては無駄が生じたり、配列の要素数を越えてしまうことがあります。この問題を解決するために、C++のstring型を使用することにします。string型は標準ヘッダファイルstring.hをインクルードすることにより使用できるようになります。C++標準ヘッダファイルは".h"を付けないでインクルードします。また、C++の標準ライブラリで提供される機能を使用する場合は、"using namespace std"(標準名前空間を使用)を宣言する必要があります。
string型は文字列に特化されており、以下のように多くの利点があります。
①つねに文字列の長さに合わせてメモリが確保されるので、配列の大きさを気にする必要がない
②文字列に対し、演算子(+, =, +=, ==, !=, <, >, <=, >=)が使用できる
③従来のchar型配列の文字列のように、"[]"で特定の文字を取り出すことができる
④従来のchar型配列に変換することも、char型配列からstring型に変換することもできる
⑤文字列を操作するためにいくつかのメンバ関数が提供されている
⑥文字数を超える領域を操作しようとしてもエラー処理が行えるので、従来のchar型配列のようなメモリ破壊がおきない。また、終端文字'\0'を気にしながら処理する必要もまったくない

(２)ビットマップファイルの読み込み時にファイル名を保存しておくようにします。DDLoadFromFile関数を以下のように変更します。

```
bool DDLoadFromFile(const DDSRFC ddsDest, string strFileName)
{
#ifdef _DEBUG
    if(DDS_SPRITE1 <= ddsDest) {
        OutputDebugString("*** Error - サーフェイス指定エラー(DDLoadFromFile)\n");
        return false;
```

```
    }
#endif
    g_strLoadFile[ddsDest] = "";    // 読み込みファイル名初期化

    // ビットマップファイル読み込み
    HBITMAP   hBMP = (HBITMAP)LoadImage(NULL, strFileName.c_str(), IMAGE_BITMAP,
                                        0, 0, LR_LOADFROMFILE | LR_CREATEDIBSECTION);
    if(NULL == hBMP) {
        OutputDebugString("*** Error - ファイル読み込み失敗(DDLoadFromFile)\n");
        return false;
    }

    // メモリデバイスコンテキスト生成
    (省略)

    // ビットマップ持ち替え
    (省略)

    // ビットマップ情報取得
    (省略)

    // サーフェイスが未初期化の場合、サーフェイスを生成する
    (省略)

    // DirectDrawSurfaceデバイスコンテキスト取得
    (省略)

    // 描画
    (省略)

    // DirectDrawSurfaceデバイスコンテキスト解放
    (省略)

    // ビットマップ解放
    (省略)

    // 読み込みファイル名保存
    if(DDS_OFFSCRN1 <= ddsDest)
        g_strLoadFile[ddsDest] = strFileName;

    return true;
}
```

DDLoadFromFile関数の引数が変更されたので、プロトタイプ宣言も変更する必要があります。また、LoadImage関数の２つ目の引数は読み込むファイル名を指定しますが、string型の文字列は受け取れないので、c_strメソッドでchar型配列の文字列に変換します。

(３)サーフェイスの復元時に、ビットマップファイルを再読み込みします。DDRestoreSurface関数を以下のように改良します。

```
bool DDRestoreSurface(const DDSRFC dds)
{
    if(NULL == g_lpDDSurface7[dds])
        return false;
    if(DDERR_SURFACELOST != g_lpDDSurface7[dds]->IsLost())
        return false;
    if(DD_OK == g_lpDDSurface7[dds]->Restore()) {
        if("" != g_strLoadFile[dds])
            DDLoadFromFile(dds, g_strLoadFile[dds]);
    } else {
        OutputDebugString("*** Error - サーフェイス復元失敗(DDRestoreSurface)\n");
        return false;
    }

    return true;
}
```