ゲームプログラミング

▶　DirectDraw－第１２回　ガンマコントロール　◀

DirectDrawには、色の輝度を自由に設定することができるガンマコントロールという機能があります。赤、緑、青の要素ひとつひとつの輝度を自由に変更することができ、画面全体の赤みを強くしたり暗くしたりするといったことが簡単に行えます。

ガンマコントロール

色の輝度のことを「ガンマ値」と呼びます。DirectDrawでは、DirectDrawGammaControlオブジェクトでガンマ値を制御することができます。Windowsには赤、緑、青のそれぞれに0～255の256段階の輝度値がありますが、このすべての輝度値についてのガンマ値を自由に設定することができます。
ガンマ値は、一度設定すると解除するまで有効です。この機能によって、画面を明るくしたり、暗くしたり、青写真やセピア色のような効果を出すことができます。なお、ガンマ値はフルスクリーンモードモードでなければ設定できません。

DirectDrawGammaControlオブジェクトの生成

ガンマコントロールオブジェクトは、プライマリサーフェイスオブジェクトのQueryInterfaceメソッドで取得します。このメソッドは、１つ目の引数にガンマコントロールオブジェクトのインタフェース参照識別子"IID_IDirectDrawGammaControl"、２つ目の引数に生成されるガンマコントロールオブジェクトのインタフェースを受け取る変数(LPDIRECTDRAWGAMMACONTROL型)のアドレスをLPVOID*型にキャストして指定します。

```
LPDIRECTDRAWSURFACE7        lpDDSPrimary;          // プライマリサーフェイス(初期化済みとする)
LPDIRECTDRAWGAMMACONTROL    lpDDGammaCtrl = NULL;  // ガンマコントロールオブジェクト
lpDDSPrimary->QueryInterface(IID_IDirectDrawGammaControl, (LPVOID*)&lpDDGammaCtrl);
```

このオブジェクトの解放は、ほかのオブジェクトと同じようにReleaseメソッドで行います。

ガンマ値の設定

ガンマ値の設定は、ガンマコントロールオブジェクトのSetGammaRampメソッドで行います。このメソッドの１つ目の引数はキャリブレーションフラグで、通常は０にします。２つ目の引数は、ガンマ値を設定したDDGAMMARAMP構造体変数のアドレスです。
DDGAMMARAMP構造体は、赤、緑、青のすべての輝度値(0～255)についてのガンマ値を設定します。ガンマ値は0～65,535の範囲で設定します。０に近づくほど暗く、65,535に近づくほど明るくなります。デフォルトのガンマ値は輝度値×256です。たとえば、赤の輝度値128のデフォルト値は128×256です。デフォルト値に設定すると元の輝度に戻り、デフォルト値より大きくすると明るく、小さくすると暗くなります。

○ 課題 ○

ガンマ値を自由に設定する機能を追加しましょう。

(１)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－

```
static LPDIRECTDRAWGAMMACONTROL    g_lpDDGammaCtrl = NULL;
```

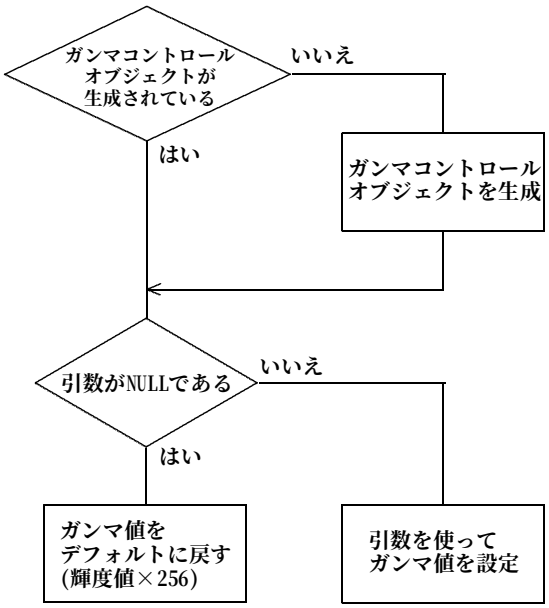
(２)以下のプログラムは、ガンマの設定を行うDDSetGammaRamp関数です。関数の仕様とフローチャートをよく読んで完成させ、適切な場所に追加しましょう。

DDSetGammaRamp関数
－説明－
　DDSetGammaRamp関数は、ガンマ値の設定を行います。

－パラメータ－
　LPDDGAMMARAMP lpDDGRamp…ガンマ値が設定されたDDGAMMARAMP構造体変数のアドレス。NULLはデフォルトのガンマ値に戻します。

－戻り値－
　関数が成功した場合はtrue，それ以外はfalseを返す。

－フローチャート－



－追加２－
```
/*****************************************************************************************************/
/*                                          ガンマ設定                                               */
/*****************************************************************************************************/
bool DDSetGammaRamp(LPDDGAMMARAMP lpDDGRamp)
{
#ifdef _DEBUG
    if(NULL == g_lpDDSurface7[DDS_PRIMARY]) {
        OutputDebugString("*** Error - プライマリサーフェイス未生成(DDSetGammaRamp)\n");
        return false;
    }
#endif

    // DirectDrawGammaControlオブジェクト生成
    if(NULL == g_lpDDGammaCtrl) {
        if(DD_OK != ※ここは各自考えましょう) {
            OutputDebugString("*** Error - GammaControlオブジェクト生成失敗(DDSetGammaRamp)\n");
            return false;
        }
    }

    if(NULL != lpDDGRamp) {
        // ガンマ設定
        if(DD_OK != ※ここは各自考えましょう) {
            OutputDebugString("*** Error - ガンマ値設定失敗(DDSetGammaRamp)\n");
            return false;
        }
    } else {
        // ガンマ値をデフォルト値に戻す
        DDGAMMARAMP   ramp;
        for(int i = 0; i < 256; i++) {
            ramp.red  [i] = i * ???;
            ramp.green[i] = i * ???;
```

```
            ramp.blue [i] = i * ???;
        }
        if(DD_OK != ??????????????->???????????(0, &ramp)) {
            OutputDebugString("*** Error - ガンマ値設定失敗(DDSetGammaRamp)\n");
            return false;
        }
    }

    return true;
}
```

(３)ガンマコントロールオブジェクトの解放を行う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－

```
    // ガンマコントロールオブジェクト解放
    if(NULL != g_lpDDGammaCtrl) {
        g_lpDDGammaCtrl->???????();
        g_lpDDGammaCtrl = NULL;
    }
```

(４)Test関数に以下のプログラムを追加し、ガンマが変更されることを確認しましょう。

```
    DDGAMMARAMP   ddgr;
    for(int i = 0; i < 256; i++) {
        ddgr.red  [i] = i * 128;
        ddgr.green[i] = i *   0;
        ddgr.blue [i] = i * 128;
    }
    DDSetGammaRamp(&ddgr);
```

(５)以下のプログラムは、簡単なフェードインを行うFadeIn関数です。(４)で追加したプログラムを削除し、この関数をGameFunc.cppに追加しましょう。

```
/*****************************************************************************************************/
/*                                          フェードイン処理                                         */
/*****************************************************************************************************/
void FadeIn()
{
    static int      nFadeCnt  =   0;                            // フェードカウンタ
    const  int      FADE_TIME = 300;                            // フェード時間
    const  double   RATIO     = (double)nFadeCnt / FADE_TIME;   // フェード比率

    // ガンマ設定
    DDGAMMARAMP   ddgr;
    for(int i = 0; i < 256; i++) {
        const WORD   wRamp = (WORD)(i * 256 * RATIO);
        ddgr.red  [i] = wRamp;
        ddgr.green[i] = wRamp;
        ddgr.blue [i] = wRamp;
    }
    DDSetGammaRamp(&ddgr);

    nFadeCnt++;
    if(FADE_TIME < nFadeCnt)
        nFadeCnt = 0;
}
```

(６)TestProc関数に以下のプログラムを追加し、フェードインが実行されることを確認しましょう。

```
    FadeIn();     // フェードイン
```

(７)FadeIn関数を参考に、フェードアウトを行うFadeOut関数を作成しましょう。