ゲームプログラミング

▶　DirectDraw－第１５回　スプライトの描画　◀

Direct3DXには、スプライトを描画するための関数がいくつか用意されています。これらを使用することにより、スプライトを簡単に描画することができます。

シーンの開始と終了

スプライトはDirect3Dを使った描画なので、描画前にDirect3D7オブジェクトのBeginSceneメソッドを呼び出してシーンの開始を宣言し、システムにレンダリングの開始を通知する必要があります。

```
// シーン開始
LPDIRECT3DDEVICE7   lpD3DDevice7;   // Direct3DDevice7オブジェクト(初期化済み)
lpD3DDevice7->BeginScene();
```

Direct3Dを使った描画がすべて終了した場合は、Direct3D7オブジェクトのEndSceneメソッドを呼び出してシーンの終了を宣言し、システムにレンダリングの終了を通知します。

```
// シーン終了
lpD3DDevice7->EndScene();
```

スプライトの描画

Direct3DXが提供するスプライト描画関数には、D3DXDrawSpriteSimple関数、D3DXDrawSpriteTransform関数、D3DXDrawSprite3D関数があり、いずれもアルファブレンド、スケーリング、回転が行えます。

D3DXDrawSpriteSimple関数
－説明－
D3DXDrawSpriteSimple関数は、スプライトをDirect3Dデバイスにレンダリングします。

－パラメータ－
１つ目の引数は、スプライトの転送元として使用するDirectDrawSurface7オブジェクトのインタフェースです。

２つ目の引数は、スプライトの描画先Direct3DDevice7オブジェクトのインタフェースです。

３つ目の引数は、スプライトの描画先座標を格納したD3DXVECTOR3構造体のアドレスです。この座標にスプライトの中心点が描画されます。

４つ目の引数は、スプライトの不透明度です。1.0が完全に不透明で、0.0に近づくほど透明になります。1.0がデフォルト値です。最終的な不透明度はスプライトのアルファ情報と組み合わされます。

５つ目の引数は、スプライトのスケーリング値です。デフォルト値は1.0で、ソースと同じサイズでレンダリングします。

６つ目の引数は、スプライトの中心点を軸とした回転角度です。角度は反時計回りにラジアン単位で指定します。デフォルト値は0.0です。度数からラジアンへ変換するには、D3DXToRadianマクロを使用します。

７つ目の引数は、スプライトの中心点を格納したD3DVECTOR2構造体のアドレスです。この値はスプライトの中心からのオフセット値を設定します。デフォルト値はNULLで、スプライトの中心が中心点になります。

８つ目の引数は、転送元の領域を格納したRECT構造体のアドレスです。デフォルト値はNULLで、転送元領域全体が使用されます。

－戻り値－
成功した場合はS_OK、それ以外の場合はエラーコードを返します。

```
LPDIRECTDRAWSURFACE7   lpDDSTexture;              // テクスチャサーフェイスオブジェクト(初期化済み)
D3DXVECTOR3   dxv3Pos(320.0f, 240.0f, 0.0f);      // レンダリング位置
D3DXDrawSpriteSimple(lpDDSTexture, lpD3DDevice7, &dxv3Pos,
                     1.0f, 1.0f, D3DXToRadian(45.0f), NULL, NULL);
```

D3DXDrawSpriteTransform関数
－説明－
D3DXDrawSpriteTransform関数は、変換行列により変換されたスプライトをDirect3Dデバイスにレンダリングします。

－パラメータ－
１つ目の引数は、スプライトの転送元として使用するDirectDrawSurface7オブジェクトのインタフェースです。

２つ目の引数は、スプライトの描画先Direct3DDevice7オブジェクトのインタフェースです。

３つ目の引数は、スプライトの変換行列を格納したD3DXMATRIX構造体のアドレスです。

４つ目の引数は、スプライトの不透明度です。1.0が完全に不透明で、0.0に近づくほど透明になります。1.0がデフォルト値です。最終的な不透明度はスプライトのアルファ情報と組み合わされます。

５つ目の引数は、転送元の領域を格納したRECT構造体のアドレスです。デフォルト値はNULLで、転送元領域全体が使用されます。

－戻り値－
成功した場合はS_OK、それ以外の場合はエラーコードを返します。

```
D3DXMATRIX   dxmtTrans;          // 変換行列(設定済みとする)
D3DXDrawSpriteTransform(lpDDSTexture, lpD3DDevice7, &dxmtTrans, 1.0f, NULL);
```

D3DXDrawSprite3D関数は、D3DVECTOR4構造体で描画座標４点を指定することにより、これらの点を内包するようにスプライトを描画します。

透過処理

スプライトでも、カラーキーを用いた透過処理を行うことができます。DirectDrawサーフェイスと同じように、SetColorKeyメソッドで透過色を設定し、Direct3DDevice7オブジェクトのレンダリングステート「D3DRENDERSTATE_COLORKEYENABLE」を有効(TRUE)に設定します。
しかし、環境によっては正しく設定したにもかかわらず透過できない場合があります。どのような環境でも透過したい場合は、アルファ情報付きの画像を読み込みます。こうすると、カラーキーを設定しなくても透過することができます。アルファ情報をサポートしている画像形式は「DDS」と「TGA」です。DDS形式は、DirectX SDK付属のテクスチャエディタ(DXTEX.EXE)で作成することができます。TGA形式はPhotoshopなど市販のグラフィックエディタで作成することができます。

```
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_COLORKEYENABLE, TRUE);
```

背面カリング

スプライトの描画時に、ｙ軸を中心に180°以上360°未満回転させると、スプライトが描画されません。Direct3Dでは、デフォルトでそのような動作をするようになっています。この動作を背面カリングと呼びます。背面カリングによって、レンダリング時に裏向きのポリゴンの描画が省かれるので、レンダリング速度が向上するという利点があります。
しかし、スプライトの描画など、背面カリングをしないで描画したい場合があります。背面カリングの方法は、Direct3DDevice7オブジェクトのレンダリングステート「D3DRENDERSTATE_CULLMODE」で設定することができます。背面カリングをしないようにする場合は、「D3DCULL_NONE」に設定します。

```
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_CULLMODE, D3DCULL_NONE);
```

ブレンドモード

転送先の色と転送元の色の合成方法をブレンドモードといいます。ブレンドモードはアルファブレンディングを用いてスプライトを描画したときに使われます。一般的なブレンドは、転送元のアルファ値をαとしたとき、

$$\text{合成される色} = \text{転送元の色} \times \alpha + \text{転送先の色} \times (1 - \alpha)$$

という式で計算されます。これは、乗算合成と呼ばれる合成です。D3DXPrepareDeviceForSprite関数が

設定するのもこの方法です。また、転送先の色に転送元の色を加算する加算合成という方法もあり、以下の式で計算されます。

合成される色 ＝ 転送元の色 × α ＋ 転送先の色

加算合成は、爆発などのテクスチャを合成するときに使われています。そのほかに減算合成などもあります。ちなみに、デフォルトの合成(合成なし)は以下の式で計算されています。

合成される色 ＝ 転送元の色 × 1 ＋ 転送先の色 × 0

ブレンドモードは、Direct3D7オブジェクトのSetRenderStateメソッドで設定します。このとき、転送元と転送先の色をどのように扱うかを設定します。

```
// 乗算合成
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_SRCBLEND,  D3DBLEND_SRCALPHA);     // 転送元×α
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_DESTBLEND, D3DBLEND_INVSRCALPHA);  // 転送先×(1－α)

// 加算合成
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_SRCBLEND,  D3DBLEND_SRCALPHA);    // 転送元×α
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_DESTBLEND, D3DBLEND_DESTCOLOR);   // 転送先をそのまま使用

// デフォルト(合成なし)
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_SRCBLEND,  D3DBLEND_ONE);          // 転送元×1
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_DESTBLEND, D3DBLEND_ZERO);         // 転送先×0
```

SetRenderStateメソッドで設定した方法により求められた転送先と転送元の色値が加算され、描画される色となります。Direct3D7では、加算以外の方法は設定できないため、減算合成を行うことはできません。

テクスチャフィルタ

テクスチャフィルタとは、テクスチャのピクセル(テクセル)をポリゴン上のピクセルに貼りつけるときに行われる処理のことです。つまり、テクスチャが拡大または縮小されるときに行われる処理のことです。テクスチャフィルタには、テクスチャが拡大されるときに使用される拡大フィルタ、テクスチャが縮小されるときに使用される縮小フィルタ、ミップマップを生成するときに使用されるミップマップフィルタの３種類があります。

３つのフィルタには、フィルタモードを設定することができます。フィルタモードには、点フィルタ、バイリニア補間フィルタ、拡大フィルタのみ異方性テクスチャフィルタがあります。点フィルタは単純に拡大または縮小されるフィルタです。バイリニア補間フィルタは周辺のピクセルと補間処理されて拡大または縮小されるフィルタです。異方性テクスチャフィルタはテクスチャとスクリーン間の角度の相違により起こる歪みを補正するフィルタです。

テクスチャフィルタは、Direct3D7オブジェクトのSetTextureStageStateメソッドで設定します。スプライトの描画では、ステージ０のステートを設定します。

```
// 点フィルタ
lpD3DDevice7->SetTextureStageState(0, D3DTSS_MAGFILTER, D3DTFG_POINT);   // 拡大フィルタ
lpD3DDevice7->SetTextureStageState(0, D3DTSS_MINFILTER, D3DTFN_POINT);   // 縮小フィルタ

// バイリニア補間フィルタ
lpD3DDevice7->SetTextureStageState(0, D3DTSS_MAGFILTER, D3DTFG_LINEAR);  // 拡大フィルタ
lpD3DDevice7->SetTextureStageState(0, D3DTSS_MINFILTER, D3DTFN_LINEAR);  // 縮小フィルタ
```

D3DXPrepareDeviceForSprite関数を呼び出すと、拡大、縮小フィルタともにバイリニア補間フィルタに設定されます。

パースペクティブ補正

テクスチャは、ポリゴンの頂点間を線形補間して貼り付けられます。画面に対して平行に広がるポリゴンには正しく貼り付けられますが、画面に対して奥行き方向に広がるポリゴンには、補間が線形ではなく曲線で増加するのが正しいため、ゆがんだ状態で貼り付けられます。

これを補正するのが「パースペクティブ(遠近)補正」です。デフォルトでは無効になっており、有効にするにはDirect3DDevice7オブジェクトのSetRenderStateメソッドの１つ目の引数に「D3DRENDERSTATE_TEXTUREPERSPECTIVE」、２つ目の引数に「TRUE」を指定します。

```
// テクスチャのパースペクティブ補正を有効にする
lpD3DDevice7->SetRenderState(D3DRENDERSTATE_TEXTUREPERSPECTIVE, TRUE);
```

○ 課題 ○

(１)以下のプログラムは、シーンを開始するDDBeginScene関数です。関数の仕様をよく読んで完成させ、適切な場所に追加しましょう。

DDBeginScene関数
－説明－
DDBeginScene関数は、シーンを開始します。

－パラメーター－
なし

－戻り値－
なし

```
/*****************************************************************************************************/
/*                                          シーン開始                                                 */
/*****************************************************************************************************/
void DDBeginScene()
{
#ifdef _DEBUG
    if(NULL == g_lpD3DDevice7) {
        OutputDebugString("*** Error - Direct3DDevice7オブジェクト未初期化(DDBeginScene)\n");
        return;
    }
#endif
    ※ここは各自考えましょう;
}
```

(２)以下のプログラムは、シーンを終了するDDEndScene関数の仕様です。DDBeginScene関数を参考に、この関数を完成させ、適切な場所に追加しましょう。

DDEndScene関数
－説明－
DDEndScene関数は、シーンを終了します。

－パラメーター－
なし

－戻り値－
なし

(３)以下のプログラムは、D3DXDrawSpriteSimple関数でスプライトを描画するDDDrawSprite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。

DDDrawSprite関数
－説明－
DDDrawSprite関数は、D3DXDrawSpriteSimple関数でスプライトを描画します。

－プロトタイプ－
```
void DDDrawSprite(const DDSRFC dds, const D3DXVECTOR3& dv3Translation, const float fAlpha = 1.0f,
                  const float fScale = 1.0f, const float fAngle = 0.0f,
                  const D3DXVECTOR2* pdv2RotationCenter = NULL, LPCRECT lprcSrc = NULL);
```

－パラメーター－
const DDSRFC dds…転送元テクスチャサーフェイス。DDSRFC列挙体型を指定
const D3DXVECTOR3& dv3Translation…描画先座標
const float fAlpha…不透明度。1.0が完全に不透明で、0.0に近づくほど透明になる
const float fScale…スケール値。1.0が元のサイズと同じ
const float fAngle…回転角。角度は反時計回りで度数単位
const D3DXVECTOR2* pdv2RotationCenter…回転を行う場合の中心点。テクスチャの中心からの距離(オフセット値)を格納したD3DXVECTOR2構造体のアドレス。NULLはテクスチャの中心
LPCRECT lprcSrc…描画元領域。NULLを指定すると領域全体

－戻り値－
　なし

```
/*****************************************************************************************************/
/*                                          スプライト描画                                           */
/*****************************************************************************************************/
void DDDrawSprite(const DDSRFC dds, const D3DXVECTOR3& dv3Translation,
                  const float fAlpha, const float fScale, const float fAngle,
                  const D3DXVECTOR2* pdv2RotationCenter, LPCRECT lprcSrc)
{
#ifdef _DEBUG
    if(DDS_TEXTURE1 > dds || DDS_TEXTURE4 < dds) {
        OutputDebugString("*** Error - テクスチャ以外のサーフェイスを指定(DDDrawSprite)\n");
        return;
    }
    if(NULL == g_lpDDSurface7[dds]) {
        OutputDebugString("*** Error - テクスチャ未初期化(DDDrawSprite)\n");
        return;
    }
    if(NULL == g_lpD3DDevice7) {
        OutputDebugString("*** Error - Direct3DDevice7オブジェクト未初期化(DDDrawSprite)\n");
        return;
    }
#endif

    ??????????????????????(g_lpDDSurface7[dds], g_lpD3DDevice7, &dv3Translation,
                           fAlpha, fScale, ???????????(-fAngle), pdv2RotationCenter, lprcSrc);
}
```

－使用例－

```
    DDDrawSprite(DDS_TEXTURE1, D3DVECTOR3(320.0f, 240.0f, 0.0f), 0.8f, 0.5f, 45.0f);
```

(４)以下のプログラムは、D3DXDrawSpriteTransform関数でスプライトを描画するDDDrawSpriteTransfo rm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。

DDDrawSpriteTransform関数

－説明－

DDDrawSpriteTransform関数は、D3DXDrawSpriteTransform関数でスプライトを描画します。

－プロトタイプ－

```
void DDDrawSpriteTransform(const DDSRFC dds, const D3DXVECTOR3& dv3Translation,
                           const D3DXVECTOR3& dv3Size, const float fAlpha = 1.0f,
                           const D3DXVECTOR3* pdv3Rotation = NULL,
                           const D3DXVECTOR3* pdv3RotationCenter = NULL, LPCRECT lprcSrc = NULL);
```

－パラメーター－

const DDSRFC dds…転送元スプライトサーフェイス。DDSRFC列挙体型を指定
const D3DXVECTOR3& dv3Translation…描画先座標
const D3DXVECTOR3 dv3Size…描画サイズを格納したD3DXVECTOR3構造体。この構造体のメンバｘに幅、メンバｙに高さを指定する
const float fAlpha…不透明度。1.0が完全に不透明で、0.0に近づくほど透明になる
const D3DXVECTOR3* pdv3Rotation…回転角を格納したD3DVECTOR3構造体のアドレス。この構造体のメンバｘにｘ軸の回転角、メンバｙにｙ軸の回転角、メンバｚにｚ軸の回転角を指定。角度は反時計回りで度数単位。NULLを指定すると回転しない
const D3DXVECTOR3* pdv3RotationCenter…回転を行う場合の中心点。テクスチャの中心からの距離(オフセット値)を格納したD3DXVECTOR3構造体のアドレス。NULLはテクスチャの中心
LPCRECT lprcSrc…描画元領域。NULLを指定すると領域全体

－戻り値－

　なし

```
/*****************************************************************************************************/
/*                                          スプライト描画                                           */
/*****************************************************************************************************/
void DDDrawSpriteTransform(const DDSRFC dds, const D3DXVECTOR3& dv3Translation,
                           const D3DXVECTOR3& dv3Size, const float fAlpha,
                           const D3DXVECTOR3* pdv3Rotation,
```

```
                                const D3DXVECTOR3* pdv3RotationCenter, LPCRECT lprcSrc)
{
#ifdef _DEBUG
    if(DDS_TEXTURE1 > dds || DDS_TEXTURE4 < dds) {
        OutputDebugString("*** Error - テクスチャ以外のサーフェイスを指定(DDDrawSpriteTransform)\n");
        return;
    }
    if(NULL == g_lpDDSurface7[dds]) {
        OutputDebugString("*** Error - テクスチャ未初期化(DDDrawSpriteTransform)\n");
        return;
    }
    if(NULL == g_lpD3DDevice7) {
        OutputDebugString("*** Error - Direct3DDevice7オブジェクト未初期化(DDDrawSpriteTransform)\n");
        return;
    }
#endif

    // クォータニオン生成
    D3DXQUATERNION    dqRotation;
    if(NULL == pdv3Rotation)
        D3DXQuaternionRotationYawPitchRoll(&dqRotation, 0.0f, 0.0f, 0.0f);
    else
        D3DXQuaternionRotationYawPitchRoll(&dqRotation, D3DXToRadian(pdv3Rotation->y),
                                           D3DXToRadian(pdv3Rotation->x), D3DXToRadian(-pdv3Rotation->z));

    // 変換行列生成
    D3DXMATRIX   dxmTransform;
    D3DXMatrixTransformation(&dxmTransform, NULL, NULL, &dv3Size,
                             pdv3RotationCenter, &dqRotation, &dv3Translation);

    // 描画
    ※ここは各自考えましょう;
}
```

－使用例－

```
    D3DXVECTOR3   dv3Pos  (320.0f, 240.0f, 0.0f);     // 位置
    D3DXVECTOR3   dv3Size (120.0f, 240.0f, 1.0f);     // サイズ
    D3DXVECTOR3   dv3Angle(45.0f, 30.0f, 45.0f);      // 回転角
    RECT          rcSrc = {0, 0, 240, 340};           // 転送元領域

    DDDrawSpriteTransform(DDS_TEXTURE1, dv3Pos, dv3Size, 0.8f, &dxv3Angle, NULL, &rcSrc);
```

(５)以下のプログラムは、ブレンドモードを設定するDDSetBlendMode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。

DDSetBlendMode関数

－説明－

DDSetBlendMode関数は、ブレンドモードを設定します。

－パラメータ－

const DDBLENDMODE ddBlendMode…ブレンドモード列挙体。以下の値を指定

DDBLEND_DEFAULT	デフォルト(合成なし)
DDBLEND_MODULATE	乗算合成
DDBLEND_ADD	加算合成

－戻り値－

なし

```
/*****************************************************************************************************/
/*                                        ブレンドモード設定                                           */
/*****************************************************************************************************/
void DDSetBlendMode(const DDBLENDMODE ddBlendMode)
{
#ifdef _DEBUG
    if(NULL == g_lpD3DDevice7) {
        OutputDebugString("*** Error - Direct3DDevice7オブジェクト未初期化(DDSetBlendMode)\n");
        return;
    }
#endif
```

```
    switch(ddBlendMode) {
      // デフォルト
      case DDBLEND_DEFAULT:
        ※ここは各自考えましょう
        ※ここは各自考えましょう
        break;

      // 乗算合成
      case DDBLEND_MODULATE:
        ※ここは各自考えましょう
        ※ここは各自考えましょう
        break;

      // 加算合成
      case DDBLEND_ADD:
        ※ここは各自考えましょう
        ※ここは各自考えましょう
        break;
    }
}
```

－追加－

```
/*****************************************************************************************************/
/*                                          ブレンドタイプ定数                                         */
/*****************************************************************************************************/
enum DDBLENDMODE {
    DDBLEND_DEFAULT,    // デフォルト(合成なし)
    DDBLEND_MODULATE,   // 乗算合成
    DDBLEND_ADD         // 加算合成
};
```

(６)以下のプログラムは、テクスチャフィルタを設定するDDSetTextureFilter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。

DDSetTextureFilter関数

－説明－

DDSetTextureFilter関数は、スプライト描画で使用するテクスチャステージ０に対して、拡大および縮小テクスチャフィルタのフィルタモードを設定します。

－パラメータ－

const DDTEXTUREFILTER ddTextureFilter…テクスチャフィルタ列挙体。以下の値を指定

DDTF_POINT	点フィルタ
DDTF_LINEAR	バイリニア補間フィルタ
DDTF_ANISOTROPIC	異方性テクスチャフィルタ(拡大フィルタのみ)

－戻り値－

なし

```
/*****************************************************************************************************/
/*                                        テクスチャフィルタ設定                                       */
/*****************************************************************************************************/
void DDSetTextureFilter(const DDTEXTUREFILTER ddTextureFilter)
{
#ifdef _DEBUG
    if(NULL == g_lpD3DDevice7) {
        OutputDebugString("*** Error - Direct3DDevice7オブジェクト未初期化(DDSetTextureFilter)\n");
        return;
    }
#endif

    switch(ddTextureFilter) {
      // 点フィルタ
      case DDTF_POINT:
        ※ここは各自考えましょう
        ※ここは各自考えましょう
        break;

      // バイリニア補間フィルタ
```

```
      case DDTF_LINEAR:
        ※ここは各自考えましょう
        ※ここは各自考えましょう
        break;

      // 異方性テクスチャフィルタ
      case DDTF_ANISOTROPIC:
        g_lpD3DDevice7->SetTextureStageState(0, D3DTSS_MAGFILTER, D3DTFG_ANISOTROPIC);
        break;
    }
}
```

－追加－

```
/*****************************************************************************************************/
/*                                        テクスチャフィルタ定数                                       */
/*****************************************************************************************************/
enum DDTEXTUREFILTER {
    DDTF_POINT,         // 点フィルタ
    DDTF_LINEAR,        // バイリニア補間フィルタ
    DDTF_ANISOTROPIC    // 異方性テクスチャフィルタ
};
```