|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 仕様書 | | 背景エフェクト | | |
| 種類 | 追加と編集 | | 作成者 | 古泉やっちん |
| 日付 | 2014/9/02 | |

**1　背景エフェクト実装についての報告**

　・Game\_BGEffect.h

名前空間BackGroundEffectの追加

BackGroundEffect::Objectクラスの追加

回転やUVアニメーションをさせるためのポリゴン管理クラス

Game\_BGEffectクラス以外で使用されることは想定していない

マクロ定数追加

GAME\_BG\_TYPE\_LOOP\_WAVE

GAME\_BG\_TYPE\_HUE

GAME\_BG\_TYPE\_PC

Game\_BGEffectクラスの変更。

主にパラメータやエフェクトで使用する変数・メソッドの追加。

　・Game\_BGEffect.cpp

BackGround::Objectの実装。

GameBGEffectクラスで各エフェクトの実装。

**2　各エフェクトについて**

　・TYPE\_LOOP\_WAVE

画像ループの平行移動とsin波・cos波によるUVの変更を行う。

sin波はVに、cos波はUに加算される。

使用するパラメータ

posX , posY

中心座標の位置

facterX , facterY

平行移動量（ピクセル単位）

radX , radY

sin波・cos波の変化角

param

sin波・cos波の最大値（ピクセル単位）

alpha

アルファ値 0~1.0f

inScreen

エフェクトで使用する画像ハンドル

　・TYPE\_HUE

時間経過による色相変化を行う。

使用するパラメータ

posX , posY

中心座標の位置

param

色相変化の速度

time / param \* 360 – 180で色相を計算している。

alpha

アルファ値 0~1.0f

inScreen

エフェクトで使用する画像ハンドル

　・TYPE\_PC

極座標変換もどきを行う。（θの範囲が0~πのためちゃんとした変換ではない）

使用するパラメータ

posX , posY

中心座標の位置

param

極座標変換の影響度の減衰率

！減衰率の計算について

Rate = r^2 \* param

極座標変換後のUV = 通常のuv \* rate + 極座標変換のuv \* (1.0f – rate)

facterX , facterY

極座標変換の円の中心位置を指定する。

alpha

アルファ値 0~1.0f

inScreen

エフェクトで使用する画像ハンドル

**3　使用方法についての補足**

　実装にあたり、パラメータの追加を行った。　それらのセッターは以下の通りである。

// 使用する場合はSetTypeIDを呼び出す前に呼ぶこと。

void SetInputScreen(int handle){ inScreen = handle; }

// 変形基準の要素を代入する。

void SetFacter(float \_x, float \_y){ facterX = \_x; facterY = \_y; }

// 変形角度の要素を代入する。

void SetRadian(float x, float y){ radX = x; radY = y; }

// 縮小拡大の要素を代入する。

void SetScale(float x, float y){ scaleX = x; scaleY = y; }

// アルファ値を代入する

void SetAlpha(float a){ alpha = a; }

　また、エフェクトの出力先をhDrawWindowではなく、指定したグラフィックハンドルにすることも可能。

// 出力先スクリーンを設定する

void SetOutputScreen(int handle){ outScreen = handle; }

// 出力先スクリーンを作成する。

int CreateOutputScreen(int w,int h){

if (isCreatedOutScreen) DeleteGraph(outScreen);

outScreen = MakeScreen(w, h, true);

isCreatedOutScreen = true;

return outScreen;

}

// 出力先スクリーンの取得

int GetOutputScreen(){ return outScreen; }

上記関数を使って、BGEffectクラスの外部の画像ハンドルを設定したり、BGEffectに出力先画像ハンドルを持たせたりということができる。

つまり、

Bg1.CreateOutputScreen(800,600);

Bg2.SetInputScreen(Bg1.GetOutputScreen());

のように行えばエフェクトを掛けた画像にさらにエフェクトを掛けることが可能。

アルファ値を付けて重ね掛けしたい場合は複数のBGEffectのインスタンスを実行してから出力された画像を描画するなどが考えられる。

**4　DxLibの設定についての注釈**

　一部DxLibの設定関数を使い設定を変更している。

　描画の際のアルファ値変更＞アルファ値を255に戻す。や、

　（出力先が変更されている場合）描画サーフェス変更＞描画サーフェスを戻す、や

　BackGroundEffect::Object内でテクスチャアドレッシングモードをWRAPに設定していたりする。

　問題があればGame\_BGEffect::Drawのあとに設定を元に戻す処理を追加されたし。