SmoothGraphic 操作マニュアル

URL:http://tirauraarea.web.fc2.com/

目次

1. 起動方法

ダウンロードしたファイルを解凍した後、解凍時に生成されたフォルダ内の「SmoothGraphic.exe」をダブルクリックすれば起動します。SmoothGraphic を起動した時、初期ウィンドウ(図 1)をプレビューウィンドウ(図 2)が画面に表示されます。

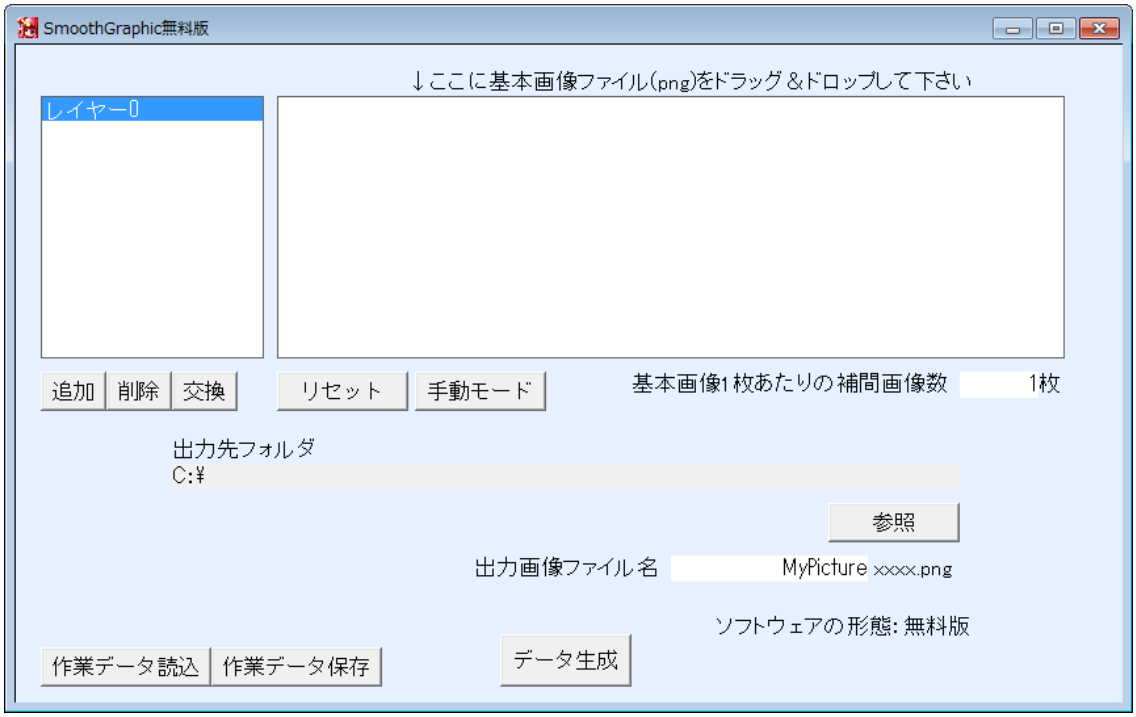

図 1:初期ウィンドウ

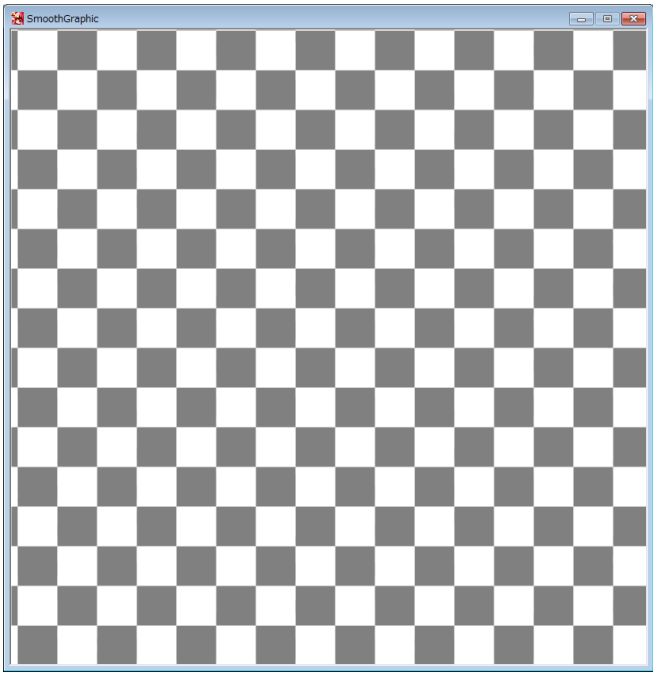

図 2:プレビューウィンドウ

2. 基本画像ファイルの読み込み

初期ウィンドウにフレーム補間の元となる基本画像のファイル(透明度付き png 形式画像)をドラッグ・アンド・ドロップします。すると、基本画像リストボックス(図 3)に基本画像ファイルの名前がリストアップされます。

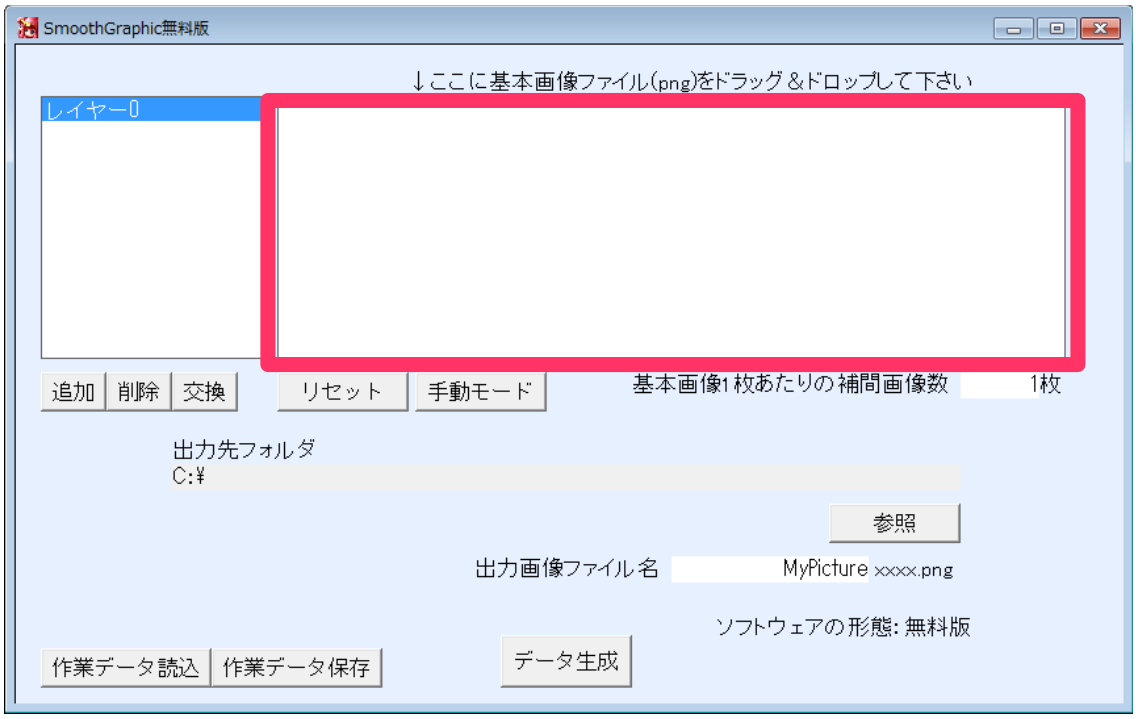

図 3:基本画像リストボックス(初期ウィンドウ右上)

本ソフトウェアではリストの上から順番にそれぞれ対応する補間画像を生成していきます。例えば、

- 基本画像 A
- 基本画像 B
- 基本画像 C
- 基本画像 D

とリストアップされた場合を考えます。この場合本ソフトウェアは、以下の順番で補間画像を生成します。

1. 基本画像 A と基本画像 B の中間に当たる補間画像を生成
2. 基本画像 B と基本画像 C の中間に当たる補間画像を生成
3. 基本画像 C と基本画像 D の中間に当たる補間画像を生成
4. 基本画像 D と基本画像 A の中間に当たる補間画像を生成

基本画像を読み込み直す場合、基本画像リストボックス左下にある「リセット」ボタンを一度クリックして下さい。「リセット」ボタンを押すと、基本画像リストがリセットされます。リセットされたら、もう一度好きな順番に基本画像ファイルをドラッグ・アンド・ドロップして下さい。

※注意

- 基本画像の画像サイズは全て統一して下さい。統一されていない場合、手動モード、補間画像の生成を行うことが出来ません。
- 本ソフトウェアは、透明度付き png 形式画像のみを扱います。

3. レイヤ操作

Ver1.1.0 以前のバージョンでは、一枚の基本画像を複数の基本画像に分割すること無く、そのまま歪める事でフレーム補間を行う仕様でした。しかし、一枚の絵をそのまま歪めるやり方には、以下の問題点があります。

- 2つ以上のオブジェクトがすれ違う部分を、正確にフレーム補間出来ない。

この問題点を解決する手段として、「あらかじめ分割された基本画像を扱えるようにする」という方法を採用しました。

本ソフトウェアでは、複数の画像を重ね合わせて1つのフレームにすることが出来ます。例えば、「のっぺらぼうの顔のみの画像」、「髪の毛のみの画像」、「目のみの画像」、「口のみの画像」など、複数の画像が用意されたとします。これら複数の画像を重ねあわせる事で、人物の顔の画像を作る事が出来ます。レイヤの操作を行うことにより、以上の例のように複数の画像を組み合わせる事が出来ます。

1. レイヤ操作

初期ウィンドウ画面左上のリストボックス(レイヤ操作リストボックス)があります。(図 4)

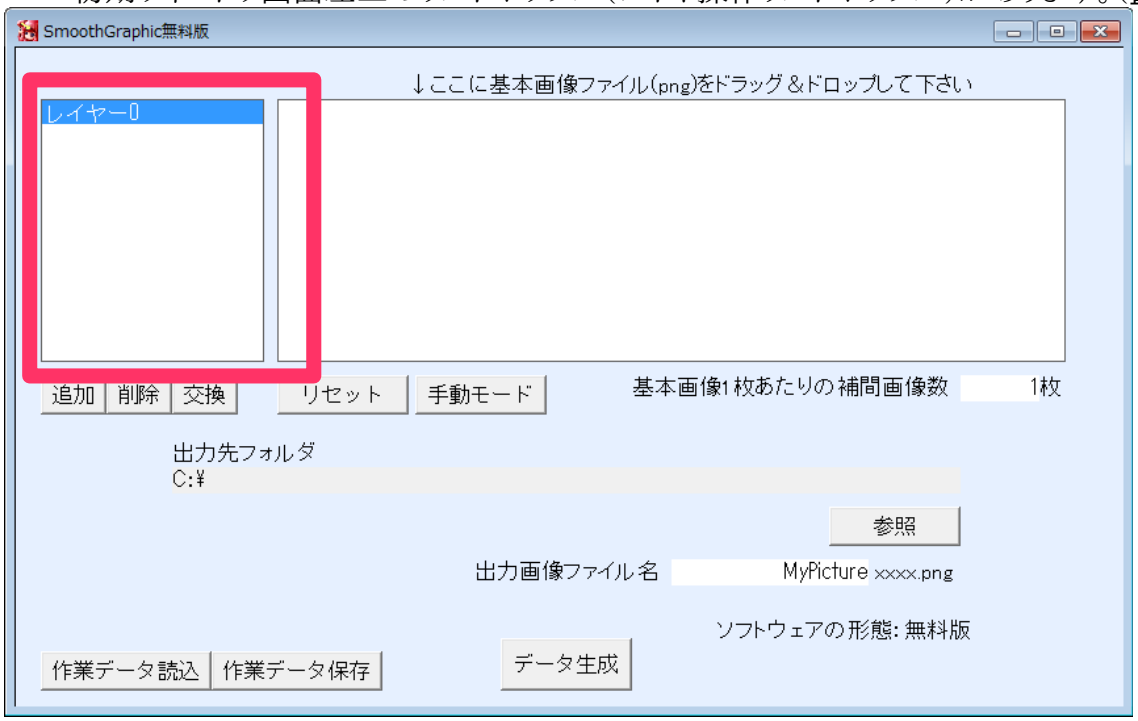

図 4:レイヤ操作リストボックス(初期ウィンドウ左上)

基本画像を読み込む時、現在選択中のレイヤ内に、基本画像データが記憶されます（表 1）。また、レイヤ操作リストボックス内の、上にあるレイヤ項目ほど、手前に表示されます。

表 1:レイヤとフレームの関係の例

	フレーム（時間）	基本フレーム0	基本フレーム1	基本フレーム2	基本フレーム3	基本フレーム4
手前に描画	レイヤ0	前髪0	前髪1	前髪2	前髪3	前髪4
↑	レイヤ1	目0	目1	目2	目3	目4
↓	レイヤ2	口0	口1	口2	口3	口4
奥に描画	レイヤ3	顔0	顔1	顔2	顔3	顔4

表 1の例では、基本フレーム０の画像を作成する場合、「顔０」→「口０」→「目０」→「前髪０」の順番に奥側から描画していきます。

レイヤを操作するためには、レイヤ操作リストボックス直下にある「追加」「削除」「交換」の3つのボタンを使います。

1. **「追加」ボタン**
   空のレイヤを追加します。その後、基本画像を読み込む時は、あらかじめ分割された基本画像群をドラッグ・アンド・ドロップします。
2. **「削除」ボタン**
   現在選択中のレイヤを削除します。削除するレイヤに登録された基本画像データもすべて開放されます。
3. **「交換」ボタン**
   「現在選択中のレイヤ A」と、「A に隣接するレイヤ」の順番を入れ替えます。これにより、画像描画の順番を変えることが出来ます。

※注意

全レイヤ内の全基本画像の画像サイズ、各レイヤごとの基本画像数はすべて統一して下さい。統一されていない場合、手動モードおよび補間画像生成を行うことが出来ません。

4. 補間画像数の指定

「基本画像一枚当たりの補間画像数」を指定します。「基本画像一枚当たりの補間画像数」とは、「N 番目の基本画像」と、「（N+1）番目の基本画像」の中間に当たる補間画像の枚数のことです。

5. 出力先フォルダ

補間画像ファイルを出力するためのフォルダを指定します。補間画像は、指定されたフォルダに出力されます。

6. 出力画像ファイル名

出力画像ファイル名を指定します。出力された画像は透明度付き png 形式の画像ファイルとして保存されます。ファイル名のあとの「xxxx」の部分には生成された順番を示す番号が入ります。

7. データ生成

「基本画像ファイルの読み込み」、「レイヤ操作」、「補間画像数の指定」、「出力先フォルダ」、「出力画像ファイル名」等の指定が済んだならば、「データ生成」ボタンをクリックします。次に、以下の 2 つの内、どちらのデータを生成するかを選択します。

- 補間画像の作成
- ゲーム開発用データの作成

1. 補間画像の作成

「補間画像を生成しています」と書かれたウィンドウが表示されます。ここで「キャンセル」ボタンをクリックすれば、処理を中止することが出来ます。

基本画像が後述の「手動モード」によって編集されていた場合、編集されたままの制御点情報が補間画像生成に適用されます。他方、全く制御点編集されていない基本画像の場合は、制御点を自動生成した上で、補間画像を生成します。

2. ゲーム開発用データの作成

「後述の 9.3.ゲーム開発用ファイル(.sgd)の作成」を参照。

8. 手動モード

手動モードでは、補間画像作成のために必要な制御点を手動で設定する事ができます。ここで、制御点とは、画像の動き(歪み)を制御するための点のことです。

本ソフトウェアでは、通常、制御点を自動的に生成し、その制御点を使って、画像の動きを推定しています。ですが、常に正しく画像の動きを推定できるわけではありません。特に、画像と画像の間の違いが大きすぎる場合、画像の動きを正確に推定することは難しいのです。このような問題点への対策として、この「手動モード」を実装しました。

1. 手動モードの起動

初期ウィンドウの「手動モード」ボタンをクリックすると、手動モードが起動します。手動モードが起動した時、3 つのウィンドウが新たに起動します。「手動モードウィンドウ(図 5)」と「制御点編集ウィンドウ左(図 6)」、「制御点編集ウィンドウ右(図 7)」です。手動モードウィンドウでは、手動モード用の各種機能の利用を行います。制御点編集ウィンドウ左では、現在の基本画像を歪める為の編集を、制御点編集ウィンドウ右では、現在から 1 つ次の基本画像を歪める為の編集を行います。

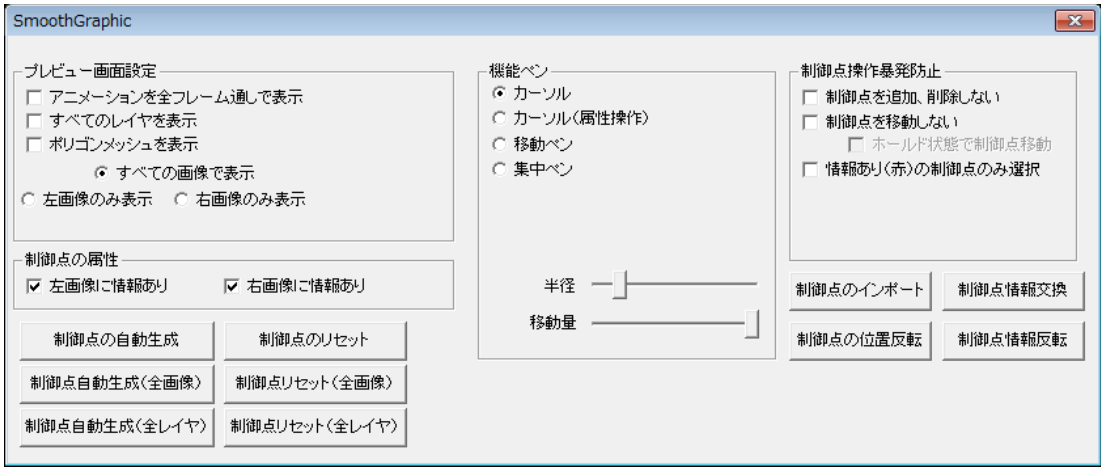

図 5:手動モードウィンドウ

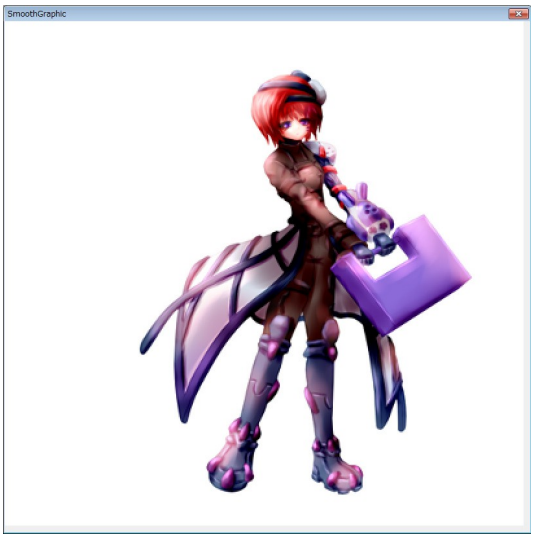

図 6:制御点編集ウィンドウ左

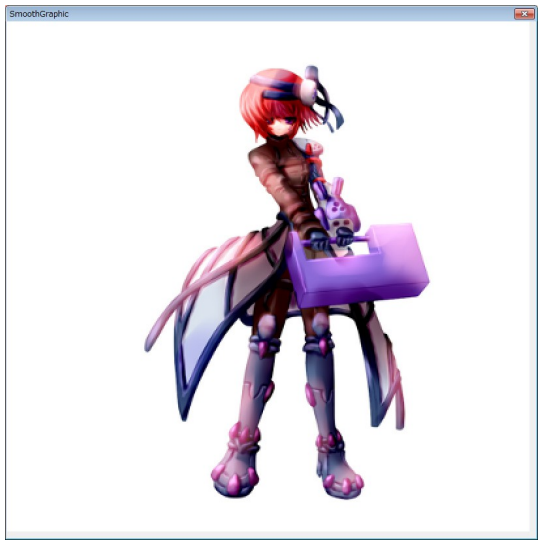

図 7:制御点編集ウィンドウ右

2. 制御点編集ウィンドウ基本操作

1. 制御点の追加

制御点編集ウィンドウ内、制御点のない場所で左クリックすれば、新たに制御点が追加されます。

2. 制御点の削除

制御点編集ウィンドウ内、既存の制御点にマウスカーソルを合わせた状態で、右クリックをすると、制御点を削除することが出来ます。

3. 制御点の移動

制御点の移動には以下の2つの方法があります。

- **制御点をドラッグ**

制御点編集ウィンドウ内、既存の制御点にマウスカーソルを合わせた状態で、マウス左ボタンを押し続けます。すると、制御点をドラッグすることが出来ます。

- **スペースキー+左クリック**

  スペースキーを押し続けている間、選択中の（マークされている）制御点を掴んだ状態になります。そのまま、制御点編集ウィンドウ内の任意の場所で左クリックすれば、制御点を移動できます。

4. 制御点編集ウィンドウのズーム

制御点編集ウィンドウ内、任意の場所にマウスカーソルを合わせ、マウスホイールを回すと、マウスカーソルの場所を基準にズームイン、ズームアウトすることが出来ます。

3. 制御点の属性操作

Ver1.1.0 以前では、「『片方の基本画像で見えていたものが、他方の基本画像で見えなくなっている』場合への対処が難しい」という問題点がありました。例えば、「前画像では見えていた左耳が、次画像では、顔の陰に隠れて見えなくなっている」ような場合への対処です。この例の場合、以下の 2 つの状況を満たしている事になります。

- 前画像では左耳が見えているので、左耳の部分を歪めたい。
- 次画像では左耳は見えていないので、有りもしない左耳のために、画像に不要の歪みを与えたくない。

Ver1.1.0 以前では 、1 つの制御点が、2 つの基本画像に同時に作用する仕様なので、上記の例の様な事が起こった場合、対処が難しくなります。

そこで、制御点には「情報有り（赤）」と「情報なし（青）」の 2 つの属性を実装しました。ユーザはこの 2 つの属性を利用することで、片方の画像のみを歪める制御点を利用する事ができます。例えば、上記の例の場合、

- 前画像では、左耳が見えているので、左耳にある制御点の属性を「情報あり（赤）」にする。
- 次画像では、左耳が見えていないので、制御点の属性を「情報なし（青）」にする。

というような使い方で、ユーザは 2 つの制御点を利用できます。

1. 制御点の属性の操作方法

制御点をの属性を操作する方法は以下の 2 つです。

- 手動モードウィンドウ内、「制御点の属性」枠にある 2 つのチェックボックスを使って制御点の属性を設定する。
- 制御点編集ウィンドウ内に存在している、任意の制御点にマウスカーソルを合わせ、「SHIFT+左クリック」を押すことで、制御点の属性を切り替える。

4. 制御点編集をする基本画像の切り替え

Ver2.0.0 以降では、手動モードウィンドウが起動している状態でも、初期ウィンドウは有効のままになっています。そのまま初期ウィンドウから、任意のレイヤー及び任意の基本画像を選択すれば、制御点編集する為の基本画像を切り替える事が来ます。

5. プレビュー画面の操作

プレビュー画面の表示設定は手動モードウィンドウ内、プレビュー画面設定枠で行います。

1. アニメーションを通しで確認

１つの基本画像間だけでなく、全体のアニメーションを確認したい時は、「アニメーションを通しで表示」チェックボックスにチェックをします。さらに、すべてのレイヤーを表示したい時は、「すべてのレイヤを表示」チェックボックスにチェックをします。

2. ポリゴンメッシュを表示

本ソフトウェアでは、画像を歪ませる為に2Ｄポリゴンを使用しています。画像が思うように歪められない場合、「ポリゴンメッシュを表示」チェックボックスにチェックをする事で、現在のポリゴンメッシュの状態を確認することが出来ます。

さらに、「左画像のみ表示」、「右画像のみ表示」、「すべての画像で表示」のいずれかを選択することで、それぞれの選択肢に対応したポリゴンメッシュを確認することが出来ます。

6. 制御点の自動生成、制御点のリセット

手動モードウィンドウ左下部分に制御点生成関係のボタンが6つあります。

- **制御点の自動生成、リセット**

  現在選択中のレイヤの、現在選択中のフレーム間にのみ制御点が自動生成されます。リセットについても、同様のフレーム間の制御点のみリセットされます。

- **制御点の自動生成、リセット（全画像）**

  現在選択中のレイヤ内の、すべてのフレーム間について制御点が自動生成されます。リセットについても、同様のフレーム間の制御点がリセットされます。

- **制御点の自動生成、リセット（全レイヤ）**

  すべてのレイヤ内のすべてのフレーム間について制御点が自動生成されます。リセットについても、同様のフレーム間の制御点がリセットされます。

7. 機能ペン

機能ペンとは、一度に複数の制御点を操作するために、試験的に実装された機能です。現時点では、必要になる機会はあまりないと思いますが、制御点が大量に存在してしまう場合などに機能ペンが重宝するはずです。

- **カーソル**
  通常のマウスカーソルの機能です。デフォルトではこのモードに設定されています。
- **カーソル(属性操作)**
  このモードを選択すると、制御点編集ウィンドウ内に、マウスカーソルを中心とした円が表示されます(範囲円)。その範囲円内の制御点の属性を一度に切り替える(揃える)事ができます。
  「情報あり(赤)」属性にする場合は左クリック、「情報なし(青)」にする場合は「Shift+左クリック」でそれぞれ属性操作が出来ます。
  また、機能ペン枠下部のスライドバーで、範囲円の大きさを調整できます。
- **移動ペン**
  「移動ペン」では、範囲円内の制御点を一度に移動することが出来ます。ただし、範囲円内の制御点は、マウスカーソルから離れているほど、移動量が小さくなる仕様です。また、機能ペン枠下部のスライドバーで範囲円の大きさ、移動量を調整出来ます。
- **集中ペン**
  「集中ペン」では、範囲円内の制御点を範囲円の中心に近づける(または遠ざける)事ができます。また、機能ペン枠下部のスライドバーで範囲円の大きさ、移動量を調整出来ます。移動量スライドバーを右に寄せれば「中心に近づける」、左に寄せれば「中心から遠ざける」ように移動量を調整出来ます。

8. 制御点操作暴発防止

「制御点を間違えて選んでしまった」、「うっかり制御点を動かしてしまった」などの操作ミスを未然に防ぐための機能です。手動モードウィンドウの左上、「制御点操作暴発防止」の枠にあるチェックボックス群を操作することで、制御点操作に制限をつける事ができます。必要に応じて、ユーザは各項目にチェックを入れたり、チェックを外したりすることが出来ます。

9. 制御点のインポート、位置反転、属性反転

手動モードウィンドウの右下に以下の3つのボタンが有ります。

- **制御点のインポート**
  選択されたレイヤ内の、一つ前のフレーム間、または次のフレーム間からすべての制御点をインポートします。
- **制御点の位置反転**
  現在の制御点編集ウィンドウ上にある制御点の位置を、上下反転、または左右反転します。

- **制御点情報反転**
  現在の制御点編集ウィンドウ上にある制御点の属性を反転します。これは、制御点を前後反転した状態に近い状態になります。
- **制御点情報交換**
  現在の制御点情報を 2 画像間で交換します。反復動作のアニメーションを作る時に役に立つはずです。

9. ゲーム開発用ライブラリ(SG ライブラリ)

本ソフトウェアでは、フレーム補間された画像を、複数の画像ファイル(補間画像)として生成します。ユーザは、生成された画像を、他の画像編集ソフト(または動画編集ソフト)を使って加工することができます。加工された画像は、各々の目的に合わせて使用する事ができます。例えば、gif アニメの作成、動画の作成、ゲームの開発などです。

ただし、ゲーム開発に補間画像を利用する場合、大量の画像を読み込むか、膨大な解像度の画像を読み込まなければなりません。その為、ビデオメモリを圧迫してしまうなどの問題点があります。

この問題点を開発するため、SG ライブラリを作成しました。ユーザ SG ライブラリを利用することで、リアルタイムレンダリングのフレーム補間を実現し、画面に補間画像をリアルタイムで描画することが出来ます。さらに、読み込まれる画像は、最小限のデータサイズで済みます。

1. 開発環境

以下の条件を満たす開発環境でのみ使用可能です。

- VC++による開発環境
- DX ライブラリ[1]を使用している

2. 下準備

SG ライブラリを、DX ライブラリの「DxLib.h」があるフォルダと同じ場所に以下の4つのファイルを移動して下さい。

- SG_Lib.h
- SGlib.h
- SGlib.lib
- SGlib_d.lib

あとは、DX ライブラリと同様の方法を取れば、下準備は完了します。

3. ゲーム開発用ファイル(.sgd)の作成

前述のデータ生成を行うとき、ゲーム開発用データの作成を選択すると、sgd 形式のファイルと、同名の png 形式画像が生成されます。SG ライブラリは、この sgd ファイルと同名の png 画像を読み込むことで、ゲーム開発用ファイルを扱います。

---

1 DX Library Copyright (C) 2001-2014 Takumi Yamada. URL:http://homepage2.nifty.com/natupaji/DxLib/

※sgd ファイルの名前を png ファイルの名前は常に同じでなければなりません。さらに、この2つのファイルは同一のフォルダに存在しなければ、正しく読み込むことは出来ません。

※sgd ファイル付随の png 画像には大きな空白部分が存在する場合があります。これは、sgd ファイル付随の png 画像をポリゴンで扱えるようにする 為、画像サイズを、$2^n \times 2^m$に調整した時に生まれた空白部分です。空白部分を減らすためには、基本画像の大きさを各自ユーザが調整していく必要があります。

4. SG ライブラリ専用構造体

SG ライブラリでは、SG_STATUS 型の構造体を定義しています。SG_STATUS 型構造体では、スクリーンに SG 専用画像を描画する為、必要になる情報を扱います。

SG_STATUS 型構造体は以下のメンバで構成されています。

- int SG_Handle　SG ライブラリ専用のグラフィックハンドル
- int x,y　SG 専用画像を回転描画する画面上の中心座標
- int sc,cy　SG 専用画像を回転描画する画像上の中心座標
- double ExtRateX　横方向の拡大率
- double ExtRateY　縦方向の拡大率
- double Angle　描画角度（ラジアン指定）
- int TurnFlag　左右反転フラグ
- float Frame　アニメーションの時点
  整数部分：基本フレーム番号
  小数部分：補間率

5. 関数の説明

本項では、SG ライブラリで定義されている関数を説明します。サンプルコードについては「9.6.サンプルコード」を参照して下さい。

SG ライブラリの初期化

- **int SG_Init();**
  引数：なし
  戻り値：0
  説明
  SG ライブラリの初期化をします。DxLib_Init()を使用した直後で呼び出します。

ゲーム開発用ファイル(.sgd)読み込む

- **int SG_FileRead(char *FileName);**
  引数：*FileName　ファイル（.sgd）のアドレスが格納されたポインタ。
  戻り値：　SG 専用画像のハンドル
  説明
  - ゲーム開発用ファイル(.sgd)を読み込みます。DX ライブラリの LoadGraph 関数と使い勝手はほぼ同じです。

- SG 専用画像のハンドルは、LoadGraph 関数で得られるグラフィックハンドルとは性質が全く異なります。故に、GraphBlend 関数のように、DX ライブラリ本来のグラフィックハンドルを引数にするような関数には対応していませんので注意して下さい。
- 「指定されたファイルが存在しない」などの原因で例外が起こった場合、SG_FileRead 関数は戻り値を返さずにエラーを出してしまうので注意して下さい。

読み込んだ SG 専用画像を描画

- **int SG_DrawGraph(SG_STATUS *status);**

  引数　**SG_STATUS　*status**　　SG_STATUS 型構造体（前述）のポインタ

  戻り値　0

  説明
  - 描画先スクリーンに SG 専用画像を描画します。DX ライブラリの DrawRotaGraph3 関数と使い勝手はほぼ同じです。ただし、SG_DrawGraph 関数では引数に SG_STATUS 型構造体のポインタを指定する点が異なります。
  - SG_DrawGraph 関数内部では、DX ライブラリで定義された関数を使ってスクリーンに描画しています。故に、SetDrawBlendMode 関数、SetDrawScreen 関数、SetDrawArea 関数などにも対応している事になります。
  - グラフィックハンドルは SG 専用画像のハンドルです。故に、GraphBlend 関数のように、DX ライブラリ本来のグラフィックハンドルを引数にするような関数には対応していませんので注意して下さい。
  - 引数（ SG_STATUS 型構造体へのポインタ、または実体）に不適切な値があったなどの原因で例外が起こった場合、SG_DrawGraph 関数は戻り値を返さずにエラーを出しますので注意して下さい。

読み込んだ SG 専用画像のサイズを取得

- **int　SG_GetGraphSize(int SG_Handle,int *x,int *y);**

  引数　int SG_Handle　　SG 専用画像ハンドル

  　　int *x ,*y　　サイズを格納する変数（横サイズ、縦サイズ）へのポインタ

  戻り値　0

  説明
  - 読み込んだ SG 専用画像のサイズを取得するための関数です。引数で指定するハンドルが SG 専用画像ハンドルであることを除けば DX ライブラリの GetGraphSize 関数と使い勝手はほぼ同じです。
  - 引数に不適切な値があったなどの原因で例外が起こった場合、

SG_GetGraphSize 関数は戻り値を返さずにエラーを出しますので注意して下さい。

読み込んだ SG 専用画像を開放する

- **int SG_DeleteGraph(int SG_Handle);**
  引数 int SG_Handle　　SG 専用画像ハンドル
  戻り値 0
  説明
  - 読み込んだ SG 専用画像を開放します。引数で指定するハンドルが SG 専用画像ハンドルであることを除いて、DX ライブラリの DeleteGraph 関数と使い勝手はほぼ同じです。
  - 「引数に不適切な値があった」、「引数が既に開放されたハンドルだった」などの原因で例外が起こった場合、SG_GetGraphSize 関数は戻り値を返さずにエラーを出しますので注意して下さい。

SG ライブラリの終了処理

- **int SG_End();**
  引数　なし
  戻り値　0
  説明
  - SG ライブラリの終了処理をします。DX ライブラリの DxLib_End 関数の直前に使用します。
  - 終了処理とはいっても、読み込んだ SG 専用画像を開放するわけではありません。読み込んだ SG 専用画像は必ず SG_DeleteGraph 関数で開放する必要があります。

6. サンプルコード

```
//DxLib.h は、SG_Lib.h の内部で呼び出すので宣言の必要なし
#include "SG_Lib.h"


int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR lpCmdLine,
int nCmdShow ){

	//ウィンドウモードで起動
	ChangeWindowMode(TRUE);

	//DXライブラリ初期化処理
	if( DxLib_Init() < 0  || SetDrawScreen( DX_SCREEN_BACK )!=0) return -1;
	SG_Init();//SGライブラリの初期化DxLib_Init()の直後に記述

	double ang = 0;
```

```
	SG_STATUS status;//SG_DrawGraph()の引数にする構造体

	//SG_STATUS構造体のメンバを設定----------------------------------------------------
	//SGライブラリ専用のグラフィックハンドル
	status.SG_Handle = SG_FileRead("sample0.sgd");//SG専用画像を読み込み

	//SG専用画像を回転描画する画面上の中心座標
	status.x = 320;
	status.y = 240;

	//SG専用画像を回転描画する画像上の中心座標
	SG_GetGraphSize(status.SG_Handle,&status.cx,&status.cy);//SG専用画像のサイズを取得
	status.cx/=2;//SG専用画像の中心を回転の中心に
	status.cy/=2;//SG専用画像の中心を回転の中心に

	//拡大率
	status.ExtRateX = 0.5;//横方向の拡大率（1.0で等倍）
	status.ExtRateY = 0.5;//縦方向の拡大率（1.0で等倍）

	//描画角度（ラジアン指定）
	status.Angle = 0;

	//左右反転のフラグ
	status.TurnFlag = FALSE;

	//描画するSG専用画像のアニメーションの時点　整数部分：基本フレーム、小数部分：補間率。
	status.Frame = 0;

	while(ProcessMessage() != -1){
		//SG専用画像をアニメーションしながら、回転
		status.Frame += 0.25;
		status.Angle += 0.001;

		//読み込んだSG専用画像を描画
		SG_DrawGraph(&status);


		ScreenFlip();
		ClearDrawScreen();

		if(CheckHitKey(KEY_INPUT_ESCAPE)) break;
	}

	//読み込んだSG専用画像を開放
	SG_DeleteGraph(status.SG_Handle);

	//SGライブラリの終了処理
	SG_End();
	//DXライブラリの終了処理
	DxLib_End();
```

```
	return 0;
}
```

10. sgd 形式ファイル閲覧ソフト「SGDViewer」

本ソフトウェアには、SGDViewer を付属しています。SGDViewer とは、前述の sgd 形式ファイルを閲覧するソフトウェアです。ファイル名は「SGDViewer.exe」です。

1. 下準備

「ファイルとプログラムの関連付け」によって、sgd 形式ファイルと「SGDViewer.exe」を関連付けてください。「ファイルとプログラムの関連付け」が済んだ後、sgd 形式ファイルを開けば、sgd 式ファイルを閲覧できます。
※「SGDViewer.exe」を直接起動しても、何も出来ません。sgd 形式ファイルから間接的に起動してください。

2. 基本操作

- ↑、↓キー：再生スピード調整
- ←、→キー：閲覧するファイルを切り替え
- フルキー 1：通常ループ再生
- フルキー 2：往復再生
- フルキー 3：逆ループ再生

11. 著作権表記のお願い

何らかの創作活動に本ソフトウェアを使用する場合、クレジットに著作権表記として、以下の一文を記載していただけると有難いです。

- SmoothGraphic Copyright(C)2013-2017　チラ裏エリア

また、何らかの創作活動に SG ライブラリを使用する場合は、クレジットに著作権表記として、以下の一文を記載してただけると有難いです。

- SG Library Copyright(C)2016 チラ裏エリア

SG ライブラリは、DX ライブラリと必要とするライブラリです。なので、SG ライブラリを使用する場合、DX ライブラリの著作権表記も忘れない様にお願いします。