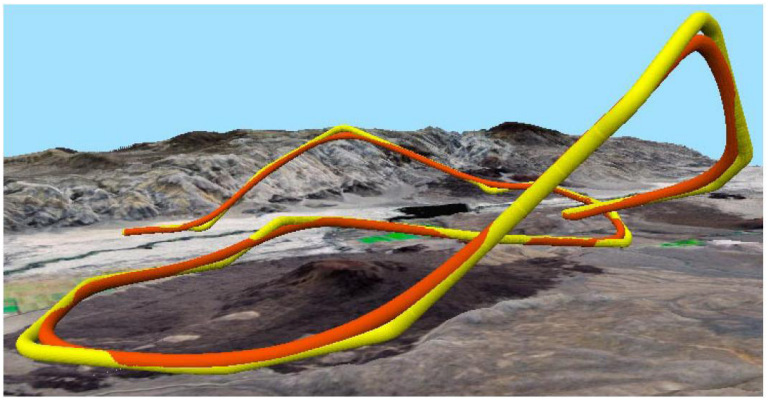
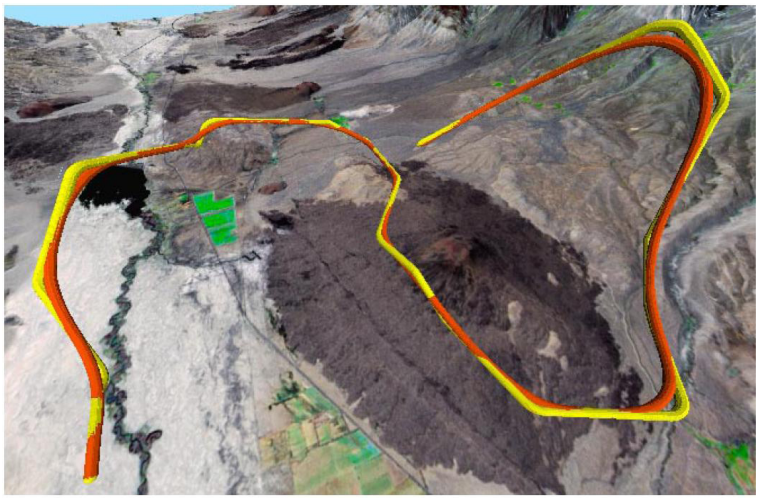
v73 新機能

飛行経路の平滑化

TNTsim3D の Advanced Flight Recorder スクリプトを使って録画した飛行経路には、急な旋回や上昇、視界方向の急激な変化を伴うことがあります。特にジョイスティックではなくキーボードを使って飛行をコントロールした場合、よく起こります。フライトレコーダーや、経路から作成した動画で飛行経路を再生する際、このような経路の形状は障害になるかもしれません (「TNTsim3D: 動画の作成 (TNTsim3D:Making Movies)」と題されたテクニカルガイドをご覧ください)。Advanced Flight Recorder スクリプトのダイアログウィンドウの中の [平滑化処理 (Smoothing Process)] パネルのコントロールを使って、これらの障害となる形状を減少させたり消去して飛行経路を修正することができます。

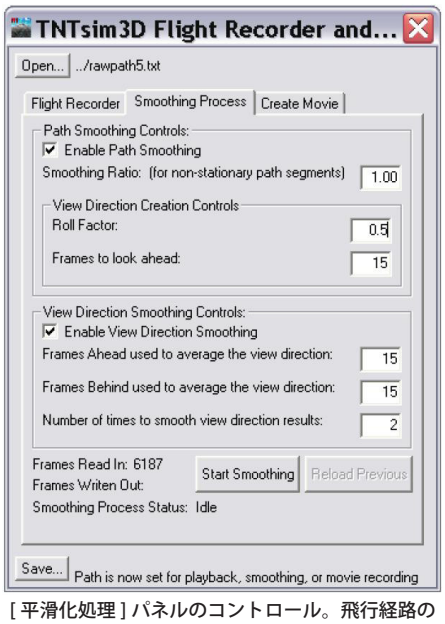




TNTsim3D で録画・平滑化した飛行経路を異なる方角から 3 次元表示しています。黄色の 3 次元ラインは加工前の飛行経路で、オレンジ色の 3 次元ラインは一度平滑化処理した後の経路を示しています。この図を作成するために、飛行経路はテキストファイルで保存され、3 次元のベクタラインとして TNTmips にインポートされ、3 次元グループに表示しました。

飛行経路には、経路中の各位置 (フレーム、コマ) に対して 2 つの値がセットされます。視点の位置 (x、y、z) と、視界の向き (ピッチ (伏角)、ロール (回転)、機首方位) です。Advanced Flight Recorder スクリプト (このページの裏側に抜粋を掲載) では、視点の位置や視界の向きを別々または連動させて平滑化できます。スクリプトによって作られた平滑化後の経路は、メモリに自動で保存されるので、経路に対して段階的に平滑化を適用できます。

「Path Smoothing(経路の平滑化)」処理は、ラインの間引きとスプラインの組み合わせによって、経路上の視点の位置を平滑化します。元の経路の位置を部分的に消したり、新たに作成します。[平滑化率 (Smoothing Ratio)] パラメータを使って、平滑化前後の経路位置の総数の比を設定できます。[平滑化率] が 1.0 より大きいと、フレーム (コマ) 数が多くなって経路が平滑化されるので、結果的に TNTsim3D での再生速度は遅くなります (フレームレートが同じ場合)。他方、比率が 1.0 より小さいと、元の経路よりフレーム (コマ) 数が少なくなりますので、再生速度が速くなります。



[平滑化処理] パネルのコントロール。飛行経路の平滑化に使用。

視点位置の平滑化の間、元の視界の向きは無視されますが、視点は飛行経路に沿って常に前方を見ているので、ピッチや機首方位の値はスクリプトによって正しく計算し直されます。「経路の平滑化」コントロールでは、[Frames to look ahead(前方を見るフレーム (コマ))] の数を設定して、視界の向きが再作成されます。録画中のフレームレート (秒あたりのフレーム (コマ) 数) と同じに設定した場合、通常は良い結果が得られます。このパラメータに大きな値を入れると、スクリプトは、経路に沿ってより遠くの前方を見て視界方向を再計算しますので、経路の曲がり角で視界の向きが早く変わるようになります。[回転係数 (Roll Factor)] パラメータでは、曲がり角でどれだけロールが変化するかを決めます。値が 0 のとき、経路を通してずっと回転がなく、他方、値を大きくすると曲がり角での回転角が大きくなり、飛行機のバンク (横転、ロール) の感覚を作り出します。

[視界方向の平滑化コントロール (View Direction Smoothing Controls)] では、視界方向に対して独自の平滑化 (平均化) 処理を適用できます。上で説明した視界の向きの再計算同様、この平滑化操作は前方と後方のフレーム (コマ) 数を指定して、平均の視界方向を計算します。

www.microimages.com/freestuf/scripts.htm にはダウンロード可能な多くのサンプルスクリプトがあり、スクリプトやクエリーで TNT 製品のスクリプト言語をどのように利用したらよいか解説しています。

Advanced Flight Recorder スクリプト (抜粋) (AdvRecorder.sml)

間引き (Thin) とスプライン (Splining) メソッドを使ったポリラインの平滑化

```
proc SmoothLine( ) {

    local class POINT3D p;
    local numeric points, i;
    local numeric PopulateIterations = 3;
    local numeric origLen, origPoints, distperframe;

    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(0) + "% complete");
```

元の経路の統計値を計算

```
    origPoints = P.GetNumPoints();
    origLen = P.ComputeLength();
    distperframe = origLen / origPoints;

    local numeric thinPercent = vcTRnum.GetValue() / 100.0;
    local numeric numPointsCut = (thinPercent * origLen) / distperframe;
```

ユーザの入力値に基づきポリラインを間引きます。“100%” ではライン全体を間引いて、端点間を直線にします。“0%” では、ラインは全く間引きされません。一般的な値は 0.01 から 10%の間です。

```
    P.Thin("Minimum", thinPercent * origLen);
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(10) + "% complete");
```

X、Y の値をスプラインして、経路を平滑化してラインの構成点を増やします。

```
    P.Spline("Cubic", numPointsCut, 1, "DontMoveEnds");
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(30) + "% complete");
    P.Spline("Cubic", 100, 1, "DontMoveEnds");
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(40) + "% complete");
```

ラインを再間引きして、X と Y のスプラインによる Z の値への影響を軽減します。

```
    P.Thin("Minimum", thinPercent * origLen * 0.01);
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(50) + "% complete");
```

Z の値をスプラインして、高さを平滑化してラインの点を増やします。

```
    P.SplineZ("Cubic", numPointsCut, 1, "DontMoveEnds");
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(60) + "% complete");
    P.SplineZ("Cubic", 100, 1, "DontMoveEnds");
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(70) + "% complete");

    local numeric thinFactor = distperframe * (1/smoothFactor);
    local numeric adjustThinning = 1;
    local class POLYLINE tempLine;
    local numeric errCount = 0;
    local numeric thinError = 0.98;
    local numeric currNum, wantNum, needNum;
```

計算の推測に基づき、試行錯誤によって、出力の頂点数が入力の頂点数とほぼ一致するように平滑化されたポリラインを間引きます。

```
    while ( adjustThinning ) {
        tempLine = P;
        tempLine.Thin("Minimum", thinFactor);

        currNum = tempLine.GetNumPoints();
        needNum = smoothFactor * origPoints;

        if (currNum > needNum) {
            if ( currNum < (needNum * (2 - thinError)) ) {
                adjustThinning = 0;
            } else {
```

thinFactor (間引き係数) を大きくします。

```
                thinFactor = thinFactor * 1.01;
                adjustThinning = 1;
            }

        } else if (currNum < needNum) {

            if ( currNum > (needNum * thinError) ) {
                adjustThinning = 0;
            } else {
```

thinFactor (間引き係数) を小さくします。

```
                thinFactor = thinFactor * 0.99;
                adjustThinning = 1;
            }

        } else {
            adjustThinning = 0;
        }
```

エラーの許容範囲に入らない場合、警告をレポート。

```
        errCount++;
        if (errCount > 100) {
            local string errMsg = "Warning: Unable to thin the line back to within \n";
            errMsg = errMsg + NumToStr(thinError) + "% of the specified Smoothing
                  Ratio.\n";
            errMsg = errMsg + "Current result is the closet value it has calculated\n.";
            errMsg = errMsg + "For more accurate results, adjust the 'thinError'
                  variable \n";
            errMsg = errMsg + "in script near this error message.";
            PopupMessage(errMsg);
            adjustThinning = 0;
        }
    }
     smStatus.SetLabel("Smoothing Process Status: "+"Smoothing Line "+
          NumToStr(80) + "% complete");

    P = tempLine;
    points = P.GetNumPoints();
```

配列のサイズ変更とデータの配列への保存

```
    ResizeArrayClear(tmpposX, points);
    ResizeArrayClear(tmpposY, points);
    ResizeArrayClear(tmpposZ, points);
    ResizeArrayClear(tmppitch, points);
    ResizeArrayClear(tmproll, points);
    ResizeArrayClear(tmpturn, points);

    local numeric n;
    for n = 1 to points {
        p = P.GetVertex3D(n-1);
        tmpposX[n] = p.x;
        tmpposY[n] = p.y;
        tmpposZ[n] = p.z;
    }
     smStatus.SetLabel("Smoothing Process Status: "+"Smoothing Line "+
          NumToStr(90) + "% complete");
```

平均化メソッドを使って Z 値を平滑化します。

```
    SmoothZValues(numPointsCut);
    smStatus.SetLabel("Smoothing Process Status: " + "Smoothing Line " +
          NumToStr(100) + "% complete");
}
```