WPF初音ミクのそだて方

主演：WPF 初音ミク

保護者：えムナウ（児玉宏之）

http://mnow.jp/

http://mnow.wankuma.com/

http://blogs.wankuma.com/mnow/

http://www.ailight.jp/blog/mnow/

つりがき

- なれそめ
- おいたち
- よそおい
- てならい
- まとめ

なれそめ

初音ミク3D計画というのがあるのは知っていました。
しかし私には関係ないものだと思っていました。

MikuMikuDance というソフトがあることを知りました。
「似たようなことはWPFのタイムラインでできるんじゃないかな」

この思いだけで長く遠いいばらの道を進むことが決定しました。

なれそめ

同じころちょうどC# MVPの宇宙仮面さんが初音ミクをXAML化したということを知りました。

こうして「おやじ二人の初音ミクXAML化計画」が始まったわけです。

宇宙仮面さん2008年03月04日

えムナウ2008年03月05日のことです。

おいたち

- おなかの中

データの研究

Masa Miku

LightWave から DXFファイル をエクスポート

ZAM3D で DXFファイル をインポート

口周りに特殊な処理をしているらしいので
　XAML で表示できないことが判明

キオ式

(1) 初音ミク　簡易モデルVer.1.1S4用.6kt　約 12MB 六角
(2) 初音ミク　簡易モデルVer.1.1S5用.6KT 約 3MB 六角
(3) 簡易モデル.3ds　約 300KB 3D Studio
(4) Kio2.lwo 約 300KB Light Wave
(5) 簡易モデル.lwo 約 180KB Light Wave
(6) 簡易モデル.obj.obj 約 2MB Wavefront
(7) 六角標準骨入り初音ミク (Miku_Bone.6KT)

結局（7）を使うことにする。

おいたち

- うぶごえ

キオ式 MIKU 初音ミク 簡易モデルの六角大王
　データ　http://kiomodel3.sblo.jp/

宇宙仮面さんが WaveFront Obj 形式に変換

Expression Blend にインポート

階層構造を変更

色つけ

タイムラインを設定　単純な踊り

おいたち

- ハイハイ

前と同じデータ

素材を取り込むように変更

データが重すぎてノートパソコンの貧弱な環境では動きが悪いことが判明

おいたち

- よちよち

キオ式 MIKU 初音ミク 簡易モデル

2008年3月16日簡易モデル.3ds

Shade にインポート

Shade でパーツに分割してジョイントパーツを作成

Shade から WaveFront Obj 形式でエクスポート

Expression Blend にインポート

階層構造を変更

タイムラインを設定　踊り2パターン

おいたち

- バタバタ

前と同じデータ

ジョイント用のプログラムを作成

子パーツの角度の変化を検出して親パーツと子パーツのあいた隙間をスキニングの技術で補間

おいたち

発表会

- 2008/03/15　わんくま同盟 東京勉強会 #18
- 2008/03/29　コミュニティ勉強会に参加しよう～第4回･福井編
- 2008/04/14（月）から 2008/04/17(木) MVPグローバルサミット
- 2008/04/30　第27回codeseek勉強会
- そして　**今日！**

- 3dグラフィックデータの調整
  - Shade に 3dグラフィックデータ(簡易モデル.3ds)をインポートします。
  - 部品の構造を確認して名前付けをします。
  - 別々に動かしたい物が同じ部品になっているものはポリゴンに変換して別々のパーツに分けます。
  - 色をマテリアルに登録して再利用します。
  - Shade から WaveFront Obj 形式でエクスポートします。

よそおい

- 3dグラフィックデータのXAML化
  - Expression Blend で プロジェクトを作ります。
  - プロジェクト-既存アイテムの追加で WaveFront Obj 形式を読み込みます。
  - Expression Blend にボーン（骨）はなく、階層的に内包していくことで連動する仕組みになっているので階層を作ります。
  - Trackball.cs を組み込みます。

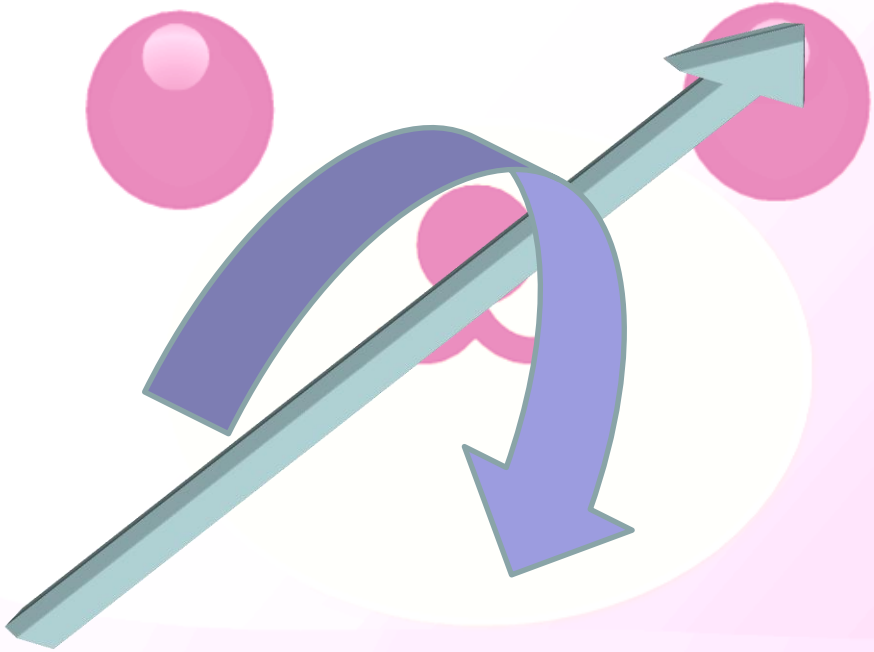
XAML化するまでのデモ

てならい

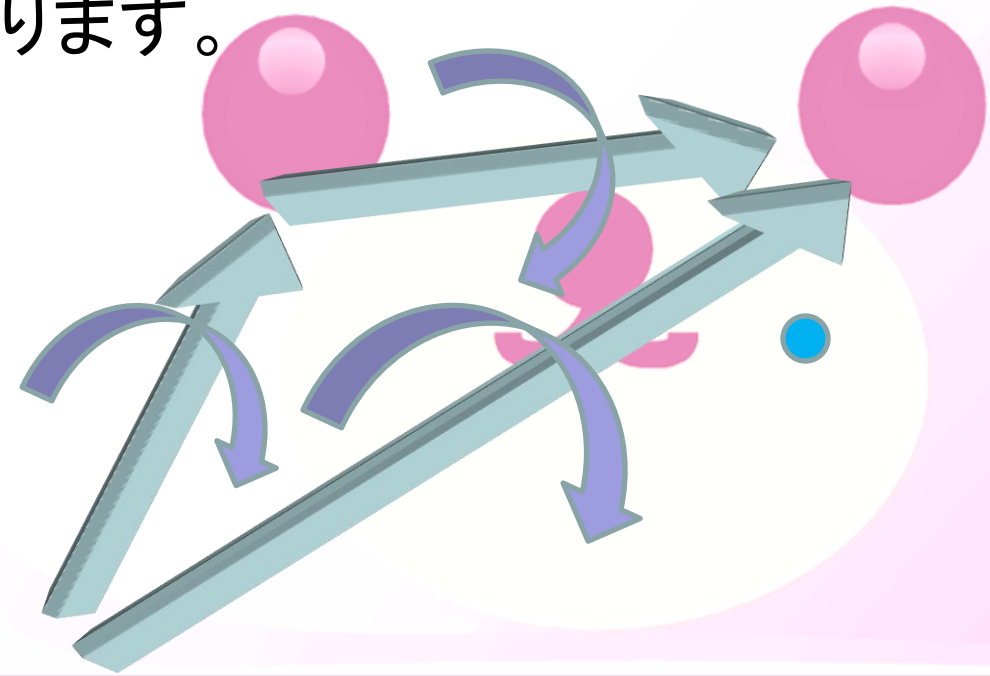
- WPF初音ミクを自然に踊ってくれるように育てるためにいろいろな技術があります。
  - WPFにおける3D
  - クオータニオン（四元数）
  - スキニング
  - 関節クラスの作成
  - モーフィング
  - モーフィングクラスの作成
  - 部分変形
  - インバースキネマティクス

- WPFにおける3D
  - WPFの3Dの基本は ModelVisual3D です。
  - ModelVisual3D の内部に、GeometryModel3D を配置して、三角形の集合体で3Dの面を構成します。
  - ModelVisual3D の内部に、Material を配置して放射・拡散・反射の3種類の色や画像を指定します。
  - ModelVisual3D は Transform プロパティでまとめて移動・拡大縮小・回転できます。

- クオータニオン（四元数）
  - クオータニオンはベクトルと回転角
  - ベクトルを軸にどのくらい回転するかを表します。



- クオータニオンの合成
  - 階層の親が回転し、子も回転するときに、ある点が実際どこにあるかは「クオータニオンの積」で求まります。

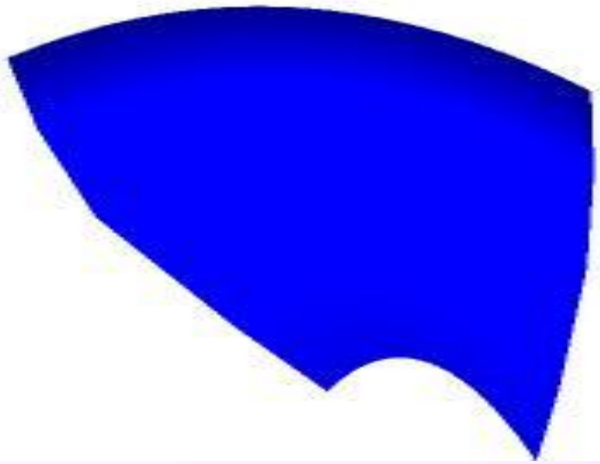


- クオータニオンから3Dの回転クラスへ

```
Vector3D leftAxis = new Vector3D(lx.Value, ly.Value, lz.Value);
Quaternion leftQuaternion = new Quaternion(leftAxis, lrot.Value);
QuaternionRotation3D qrot3d = new QuaternionRotation3D(leftQuaternion);
RotateTransform3D leftRot3D = new RotateTransform3D(qrot3d);

// クオータニオンの積
Quaternion resultQuaternion = leftQuaternion * rightQuaternion;
QuaternionRotation3D resultQR3D = new
            QuaternionRotation3D(resultQuaternion);
RotateTransform3D resultRot3D = new RotateTransform3D(resultQR3D);
```

- スキニング
  - 皮膚を滑らかに見せるための手法で頂点座標を曲線で補間します。



てならい

- 中核になるクラスはUIElement3D
  - 今までは新しい ModelVisual3D を作成して古いもの通りかえることしかできませんでした。
  - ModelVisual3D は Geometry3D や Material や Transform など必要なものは皆用意しないといけません。
  - UIElement3D は新しく作成した Geometry3D だけ入れ替えればいいことになります。
  - CPUの負荷がかかりますので多用することはできません。

てならい

- 関節クラスは、親階層の ModelVisual3D と、子階層の ModelVisual3D を指定して、その角度のずれからくる隙間を埋めます。
- Joint3D を動的生成して親階層の ModelVisual3D の子どもに加えます。
- 親と子の開口部を調査します、開口部の再接近しているものを対象とします。
- 子の位置や角度が変わったらスキニングで Joint3D の Geometry3D を再計算します。

- 関節クラスの作成
  - 周辺ユーティリティの整備

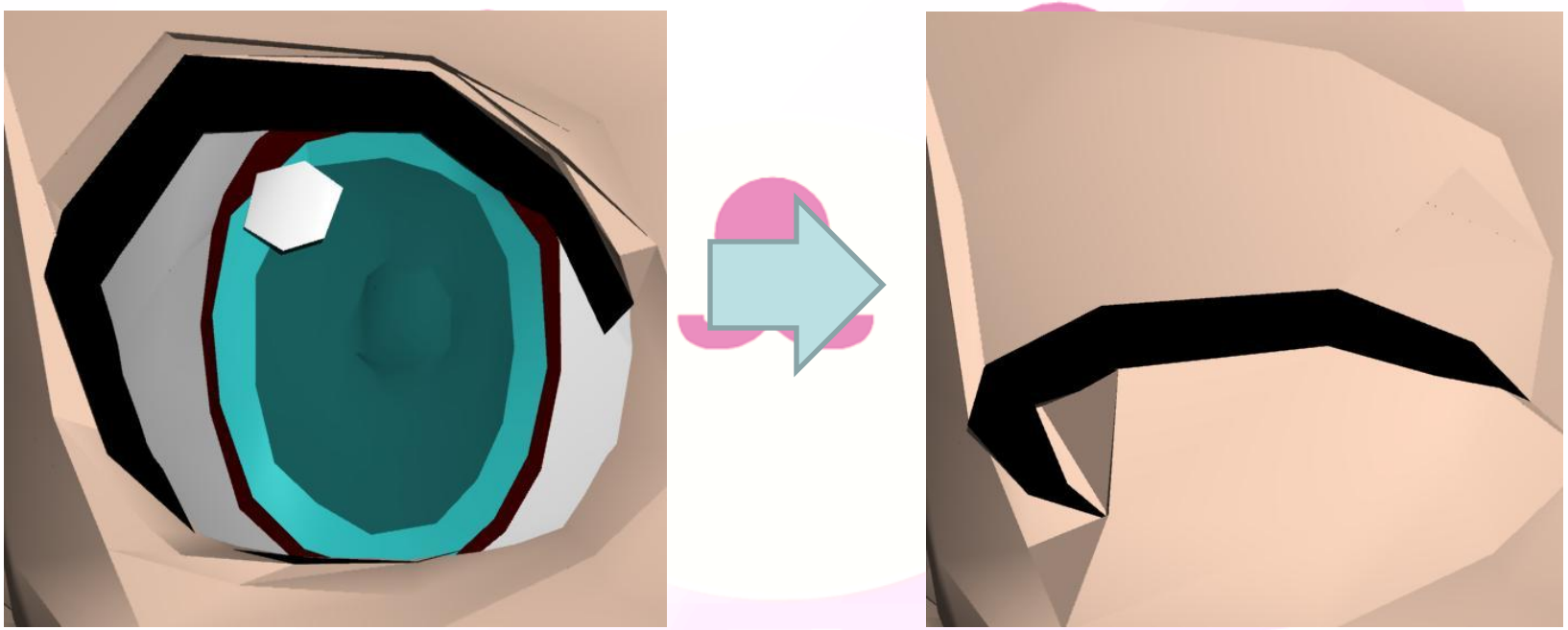
BlendUtility	Blend 固有の決まりごとを吸収します。
Int32CollectionExtension	Int32Collection を拡張します。
Point3DCollectionExtension	Point3DCollection を拡張します。
ElementTreeList	Visual Tree や Logical Tree を解析します。
EnumerateMember	Refrection を使用して Load している Assembliy からターゲットとする型のインスタンスを列挙します。
ModelVisual3DTypeConverter	ModelVisual3D のインスタンスをコンボボックスで選択するための TypeConverter です。
LineIndices, LineIndicesList	WPF の 3D で扱う三角形の線情報を扱うクラスです。

• 関節クラスの作成

Joint	Control から派生して一つの関節を作成します、Joint3D クラスを動的に生成します。
JointProperty	Joint の Property を定義した Partial クラスです。
Joint3D	Control から派生して一つの関節を作成します、Joint3D クラスを動的に生成します。 関節の作成はスキニングの技術を利用しています。
Joint3DProperty	Primitive3D から派生して新たな関節の Geometry を生成します。
Primitive3D	UIElement3D から派生して Material や Geometry を管理します。

てならい

- モーフィング
  - モーフィングは、ある物体から別の物体へと自然に変形する手法です。



- 単純に点座標をratioで比例配分します。

```
Point3DCollection fromPoint3D = fromMeshGeometry3d.Positions;
Point3DCollection toPoint3D = toMeshGeometry3d.Positions;
Point3DCollection targetPoint3D = targetMeshGeometry3d.Positions;
double ratio = this.Ratio;
int count = targetPoint3D.Count;
for (int index = 0; index < count; index++)
{
    targetPoint3D[index] = new Point3D(
        toPoint3D[index].X * ratio + fromPoint3D[index].X * (1.0 - ratio),
        toPoint3D[index].Y * ratio + fromPoint3D[index].Y * (1.0 - ratio),
        toPoint3D[index].Z * ratio + fromPoint3D[index].Z * (1.0 - ratio));
}
targetMeshGeometry3d.Positions = targetPoint3D;
```

- モーフィングクラスの作成
  - モーフィングクラスは、変形前の ModelVisual3D と、変形後の ModelVisual3D と適用先の ModelVisual3D を指定して、Ratio でその割合を指定します。
  - モーフィングは単純なものは頂点の数や順番が合わないとうまくいきません。
  - モーフィングはCPUの負荷がかかりますので多用することはできません。

- 部分変形
  - WPF の ModelVisual3D はまとめて移動・拡大縮小・回転できますが、一パーツであるスカートのすそが広がったり、髪の毛が広がったりはできません。
  - スキニングやモーフィングの技術を応用して開口部の移動・拡大縮小・回転を行い、他の頂点は自動的にそれに合わせて補間することができます。
  - CPUの負荷がかかりますので多用することはできません。

てならい

- インバースキネマティクスは逆運動学ともいわれます、通常人間がコップを手に取るには、肩の関節を動かせ、ひじの関節を動かせ、さらに手首や指先の関節を動かします。
- インバースキネマティクスは指先がコップを触るためには手首・ひじ・肩の順に関節がどうなっていればいいかを計算します。
- 通常、沢山の解があるので最少動作を基本とします、解がない場合もあります。

各種技術のデモ

最新版のデモ

まとめ

- 宇宙仮面さんには「私は、何の意味があるんだ、馬鹿だなと思いながらやっているのに、あなたはそこまでやりますか」と言われ、当然ですと答える。
- でも、これを一通りやったら、XAMLの3Dグラフィックのライブラリ整備の面もあります。
- これから他の物を作る場合も、皆さんが作りたい場合も基本ライブラリとして使えるでしょう。