Unityはじめるよ
～フェイスアニメーション編～

統合開発環境を内蔵したゲームエンジン
http://japan.unity3d.com/

※いろんな職業の方が見る資料なので説明を簡単にしてある部分があります。正確には本来の意味と違いますが上記理由のためです。ご了承ください。
この資料内の一部の画像、一部の文章はUnity公式サイトから引用しています。

フェイスアニメーション

「キャラの表情を表現する方法」
を紹介するぜ。




■フェイスアニメーションの方法は？

大きく分けて２通りの方法がある

・顔のポリゴンをアニメーションさせる

・顔のテクスチャを切り替える

※上記を組み合わせたハイブリッドな方法もある

メリットとデメリットを見てみよう

■顔のポリゴンをアニメーションさせる

顔にボーンを仕込み、ポリゴンの頂点アニメーションで表情を作る技法

メリット

- メモリ使用量が少ない
- アニメーションの繋がりが自然

デメリット

- 表現力に限界がある（ポリゴンが破たん）
- 制作に技術力が必要

３Ｄモデリングソフトでアニメーションを作り、Ｕｎｉｔｙ側ではＡｎｉｍａｔｏｒを使って制御する。

表情の切り替えにＵｎｉｔｙのモーションブレンドが使えるので、表情の変化が自然に仕上がる。

ちなみに、
Ｕｎｉｔｙちゃんはこの方法で作られていた

■顔のテクスチャを切り替える

顔にテクスチャを貼り、
そのテクスチャを切り替えることで表情を再現する技法

メリット

- 豊かな表情が可能
- 作るのが容易

デメリット

- アニメーションが滑らかではない
- メモリ使用量が多い

今回は

「顔のテクスチャを切り替える方法」

を紹介するぜ。

■顔のテクスチャを切り替える方法

大きく分けて２通りの方法がある

・表情毎に画像を用意する

・１枚の画像に複数の表情を配置する
ＵＶアニメーション

メリットとデメリットを見てみよう

■表情毎に画像を用意する

メリット

・表情の追加が容易

デメリット

・処理が重い

■１枚の画像に複数の表情を配置するＵＶアニメーション

メリット

・処理が軽い

デメリット

・表情の追加が大変

今回は

「１枚の画像に複数の表情を配置する
ＵＶアニメーション」

を紹介するぜ。

■考えなくてはいけないこと

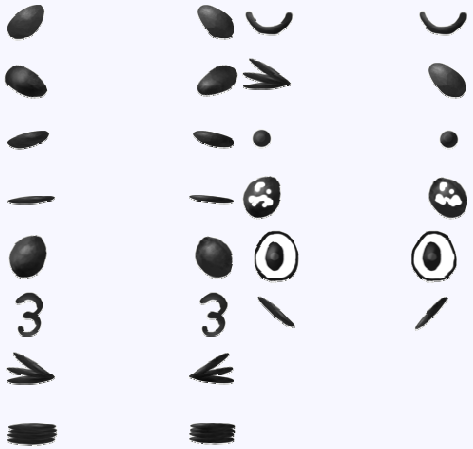
・表情毎に画像を用意すると、
画像が膨大になってしまう。

・そもそもどうやって表情テクスチャを張り付けるんだい？

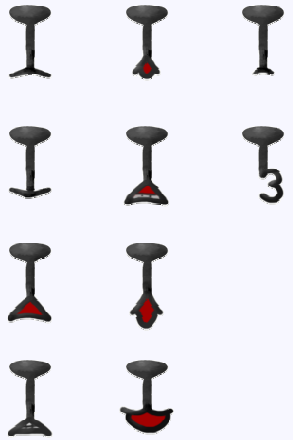
■表情毎に画像を用意すると、画像が膨大になってしまう問題。

表情を表現するのに変化が大きいパーツは、目(眉毛含む)と口である。
なので目と口を別の画像で用意し組み合わせることで、少ない画像数で多彩な表情を再現できる。

つまりこう



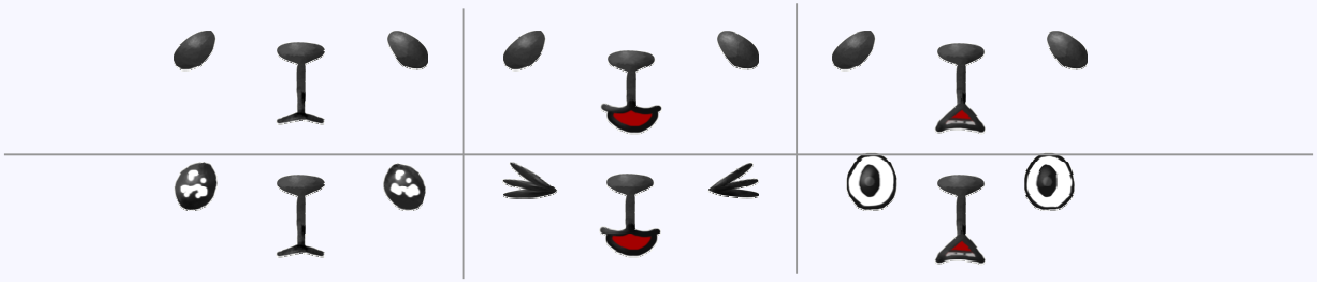
目用テクスチャ



口用テクスチャ

別々に用意しとく

組み合わせることで…

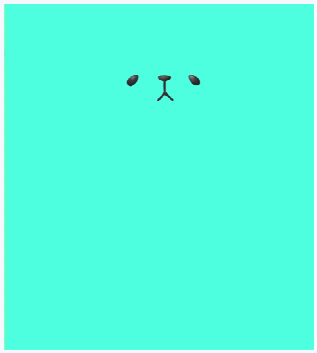


■そもそもどうやって表情テクスチャを張り付けるんだい？問題。

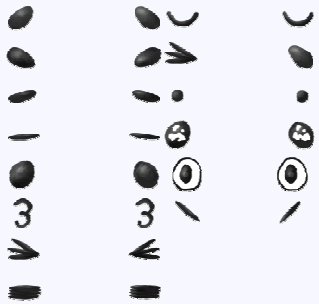
目や口の周りの肌を含めたテクスチャにするよりも、
肌テクスチャの上に表情用のテクスチャを張る
やり方が良いかな。

Ｕｎｉｔｙ４以前なら　デカールシェーダで出来そう。
Ｕｎｉｔｙ５以降なら　スタンダードシェーダの
SecondaryMapsで出来そう。

でもどうやって顔の位置にテクスチャ貼るって指定するんだろ？



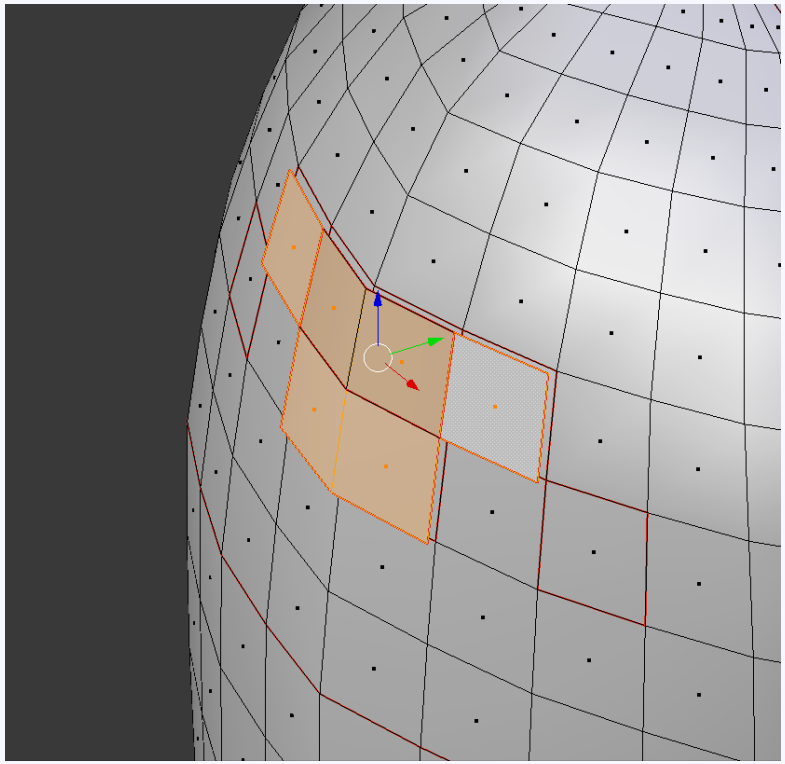
ベーステクスチャと同じ位置関係で
表情テクスチャを用意すればうまくいく。

でもこれじゃ  は使えないじゃん！

調べてみた。

けど
わからんかった。

なので知ってるやり方で。（スマートさに欠ける）



表情の部分に別ポリゴンを
重ね合わせる方法。
デカールシェーダも不要！
モデル作成時もＵｎｉｔｙ
側も難しい設定はなし！

問題があるとすれば、
ポリゴンが近接してるので
Ｚバッファの解像度的に
遠距離にいるときに
ちらつく可能性がある。

良い方法があったら教えてください
<(_　_)>

■ＵｎｉｔｙでＵＶアニメーション方法

目用、口用のマテリアルの
テクスチャのオフセット（位置）を
スクリプトから移動してやればよい。

スクリプトからのオフセット変更方法は

```
Vector2  offset;
ここでoffsetの値を指定
マテリアル.SetTextureOffset(”_MainTex”,  offset);

ちなみにサイズの変更は
Vector2  scale;
ここでscaleの値を変更
マテリアル.SetTextureScale(”_MainTex”,  scale);
```

ワンポイント！

マテリアルってなんだっけ？

３Ｄモデルの質感を決める機能のこと。

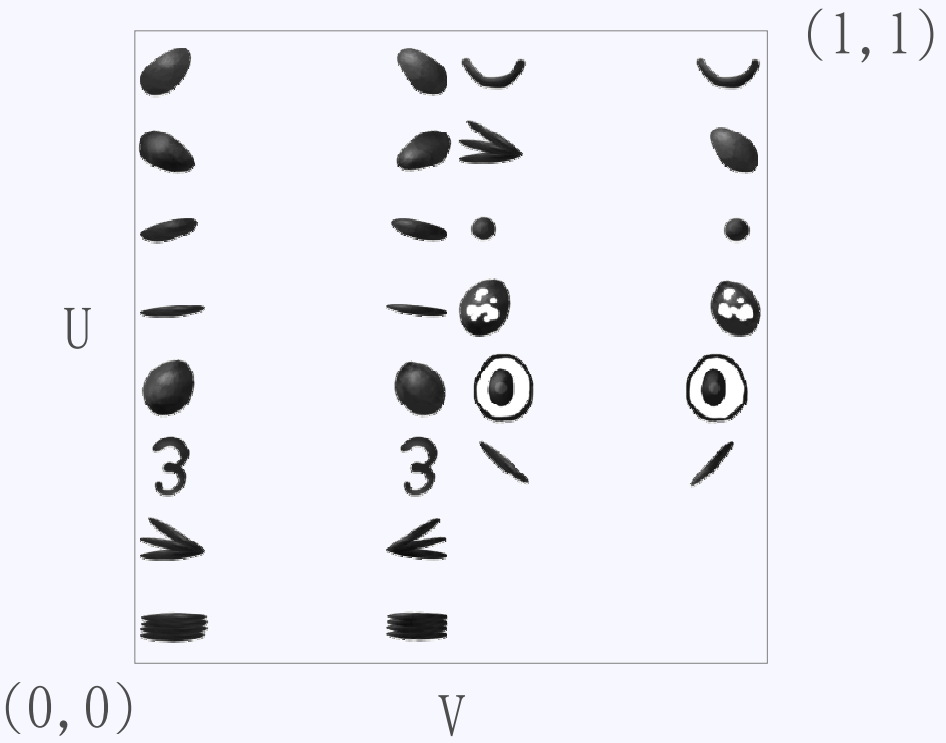
この中にテクスチャのどの部分を使うって情報も入っているのだ。

Ｕｎｉｔｙでのテクスチャの扱い方

・座標はＵＶ(0.0f～1.0f)

　　テクスチャのサイズに関わらず(0～1)で扱う

・左下が(0,0)右上が(1,1)



赤枠の怒ってる目のオフセットUV座標を求めてみよう

※プログラムでの扱いを簡単にするために、個々の目パーツのサイズが同じになるように画像を作成してある。

パーツに番号を割り振る

計算式

```
int   パーツX座標   =   パーツ番号   ÷   縦の個数   ×   パーツ幅
int   パーツY座標   =   パーツ番号   %   縦の個数   ×   パーツ高さ
float   オフセットU   =   パーツX座標   /   テクスチャ幅
float   オフセットV   =   1.0f   -   (パーツY座標   /   テクスチャ高さ)
```

赤枠の怒ってる目のオフセットUV座標を求めてみよう

テクスチャのサイズを512x512とした場合

計算式に当てはめてみる

```
int   パーツX座標   =   1   ÷   8   ×   256   =   0;
int   パーツY座標   =   1   %   8   ×   64   =   64;
float   オフセットU   =   0.0f   /   512.0f   =   0.0f;
float   オフセットV   =   1.0f   -   (64.0f   /   512.0f)   =   0.875f;
```

答え.　オフセットUVは(0.0f,　0.875f)

計算式

```
int   パーツX座標   =   パーツ番号   ÷   縦の個数   ×   パーツ幅
int   パーツY座標   =   パーツ番号   %   縦の個数   ×   パーツ高さ
float   オフセットU   =   パーツX座標   /   テクスチャ幅
float   オフセットV   =   1.0f   -   (パーツY座標   /   テクスチャ高さ)
```

■クラス化して使いやすくしたもの

```
using UnityEngine;
using System.Collections;

namespace Monolizm
{
    /// <summary>
    /// フェイスアニメーション管理クラス.
    /// </summary>
    public class FaceAnimationController : MonoBehaviour
    {

        #region public enumerate ---------------------------------------------------------------
        /// <summary>
        /// 目の種類.
        /// </summary>
        public enum EyeTypes:int
        {
            Default,        /// デフォルト.
            Angry,          /// 怒ってる.
            Blink,          /// 瞬き.
            Close,          /// 閉じている.
            Open,           /// 見開いている.
            Dizzy,          /// フラフラ.
            Happy,          /// 嬉しい.
            Hmm,            /// ふーん.
            Feel,           /// 感じる.
            Wink,           /// ウィンク.
            Dot,            /// 目が点.
            Wet,            /// 潤んでる.
            Surprised,      /// 驚き.
            Boast,          /// 自慢.
            NUM,            /// 目の種類数.
        }


        /// <summary>
        /// 口の種類.
        /// </summary>
        public enum MouthTypes:int
        {
            Default,        /// デフォルト.
            Smile,          /// 笑顔.
            A,              /// あ.
            I,              /// い.
            U,              /// う.
            E,              /// え.
            O,              /// お.
            Happy,          /// 嬉しい.
            Nmm,            /// ん.
            Chu,            /// チュ.
            NUM,            /// 口の種類数.
        }
        #endregion


        #region public variable ----------------------------------------------------------------
        public    Material          EyeMaterial;                  /// 目のマテリアル
        public    Material          MouthMaterial;                /// 口のマテリアル
        public    Vector2             EyeAtlasSize;               /// 目テクスチャ内の一つのマップサイズ.
        public    Vector2             MouthAtlasSize;             /// 口テクスチャ内の一つのマップサイズ.
        #endregion


        #region private variable ---------------------------------------------------------------
        /// <summary>
        /// アニメーション情報
        /// </summary>
        private struct AnimationInfo
        {
            public    Vector2         TextureSize;                /// テクスチャのサイズ.
            public    Rect            Atlas;                      /// アトラス情報.
            public    int             Type;                       /// 種類.
            public    Material        Mat;                        /// マテリアル.
            public    int             VNum;                       /// 縦方向のアトラス数
            public    int             HNum;                       /// 横方向のアトラス数
        }
        private    AnimationInfo      m_eyeInfo;                  /// 目のアニメーション情報
        private    AnimationInfo      m_mouthInfo;                /// 目のアニメーション情報
        #endregion
```

```
    #region Unity MonoBehaviour OverRide----------------------------------------------------------
    void Awake ()
    {
        // 目のテクスチャ情報
        m_eyeInfo.Mat           = EyeMaterial;
        Texture texture         = m_eyeInfo.Mat.mainTexture;
        m_eyeInfo.TextureSize.x = texture.width;
        m_eyeInfo.TextureSize.y = texture.height;
        m_eyeInfo.Atlas.width   = EyeAtlasSize.x;
        m_eyeInfo.Atlas.height  = EyeAtlasSize.y;
        m_eyeInfo.VNum          = (int)(m_eyeInfo.TextureSize.y / m_eyeInfo.Atlas.height);
        m_eyeInfo.HNum          = (int)(m_eyeInfo.TextureSize.x / m_eyeInfo.Atlas.width);

        // 口のテクスチャ情報
        m_mouthInfo.Mat             = MouthMaterial;
        texture                 = m_mouthInfo.Mat.mainTexture;
        m_mouthInfo.TextureSize.x   = texture.width;
        m_mouthInfo.TextureSize.y   = texture.height;
        m_mouthInfo.Atlas.width     = MouthAtlasSize.x;
        m_mouthInfo.Atlas.height    = MouthAtlasSize.y;
        m_mouthInfo.VNum            = (int)(m_mouthInfo.TextureSize.y / m_mouthInfo.Atlas.height);
        m_mouthInfo.HNum            = (int)(m_mouthInfo.TextureSize.x / m_mouthInfo.Atlas.width);

    }
    #endregion


    #region public methods ------------------------------------------------------------------------
    /// <summary>
    /// 目の種類を変更.
    /// </summary>
    /// <param name="type">Type.</param>
    public void ChangeEyeType(EyeTypes type)
    {
        m_eyeInfo.Type = (int)type;

        // 座標を求める
        m_eyeInfo.Atlas.x = ((int)type / m_eyeInfo.VNum);
        m_eyeInfo.Atlas.y = ((int)type - (m_eyeInfo.Atlas.x * m_eyeInfo.VNum));
        m_eyeInfo.Atlas.x *= m_eyeInfo.Atlas.width;
        m_eyeInfo.Atlas.y *= m_eyeInfo.Atlas.height;

        // UV座標計算
        Vector2 offset;
        offset    = new Vector2(m_eyeInfo.Atlas.x / m_eyeInfo.TextureSize.x,      1.0f-(m_eyeInfo.Atlas.y / m_eyeInfo.TextureSize.y));

        // 適用
        m_eyeInfo.Mat.SetTextureOffset("_MainTex", offset);
    }


    /// <summary>
    /// 口の種類を変更.
    /// </summary>
    /// <param name="type">Type.</param>
    public void ChangeMouthType(MouthTypes type)
    {
        m_mouthInfo.Type = (int)type;

        // 座標を求める
        m_mouthInfo.Atlas.x = ((int)type / m_mouthInfo.VNum);
        m_mouthInfo.Atlas.y = ((int)type - (m_mouthInfo.Atlas.x * m_mouthInfo.VNum));
        m_mouthInfo.Atlas.x *= m_mouthInfo.Atlas.width;
        m_mouthInfo.Atlas.y *= m_mouthInfo.Atlas.height;

        // UV座標計算
        Vector2 offset;
        offset    = new Vector2(m_mouthInfo.Atlas.x / m_mouthInfo.TextureSize.x,      1.0f-(m_mouthInfo.Atlas.y / m_mouthInfo.TextureSize.y));

        // 適用
        m_mouthInfo.Mat.SetTextureOffset("_MainTex", offset);
    }

    #endregion


    #region private methods ------------------------------------------------------------------------
    #endregion
  }
}
```

実演

ご清聴ありがとうございました