CS 6204 Character Animation

Mesh Based Animation (Facial Animation)

Yong Cao
Virginia Tech

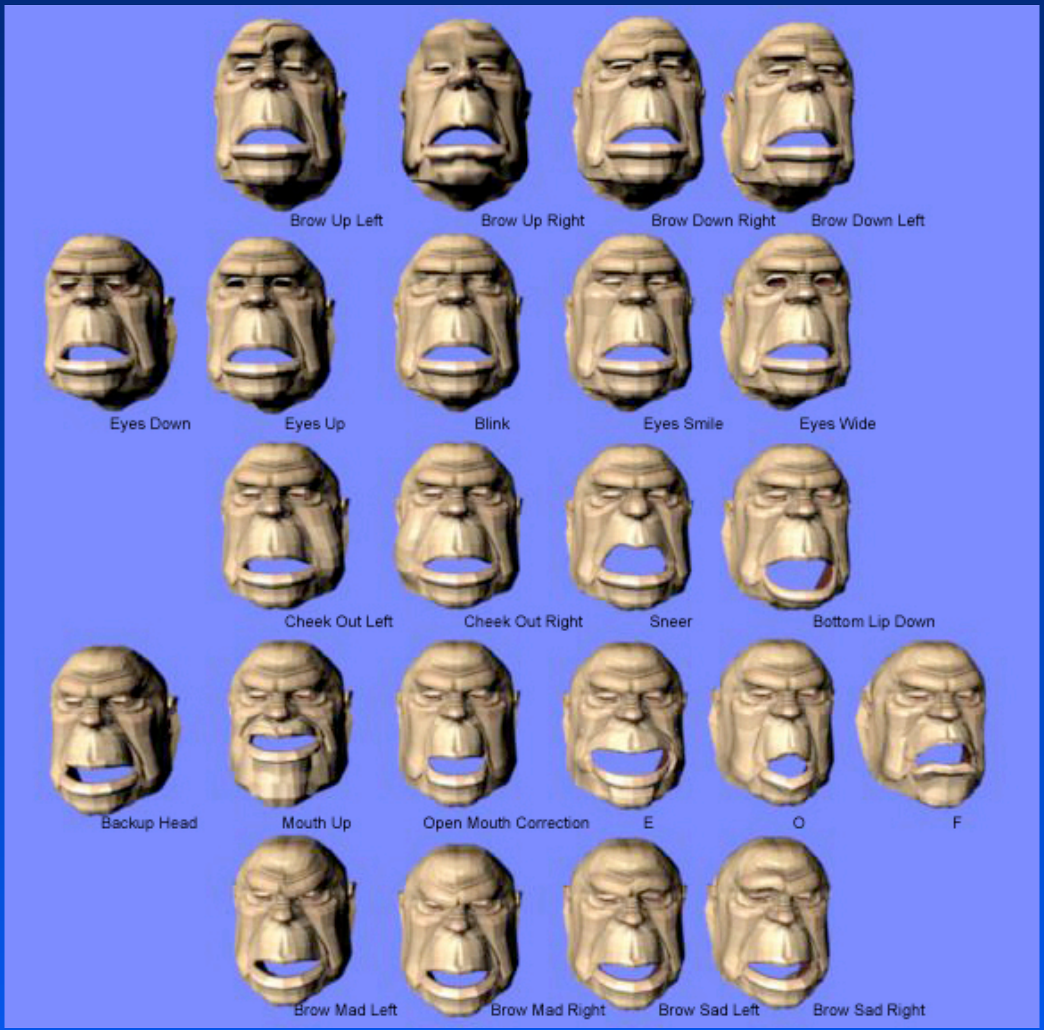
Why use mesh directly

- *For smooth deformation, like human face or rubber ball, it's NOT intuitive to use "Bone" to group vertices.*
- *So use shape interpolation and control points*

Shape Interpolation Methods

Blend Shapes

Morph Targets



Shape Interpolation Methods

- ***Several different key expressions are sculpted ahead of time***
- ***The key expressions can then be blended on the fly to generate a final expression***
- ***One can interpolate the entire face (happy to sad) or more localized zones (left eyelid, brow, nostril flare…)***

Shape Interpolation

Limitation:

- High set up time
- High interpolation time (run time)
- High storage space

Interpolation Targets

1. ***Base Mesh (Neutral expression)***
2. ***Individual targets (shapes) --- Vertex offset from base mesh***
   1. The *topology* of the target meshes must be the same as the base model (i.e., same number of verts & triangles, and same connectivity).
   2. Each target is controlled by a DOF Φ_i that will range from 0 to 1.

Shape Interpolation Algorithm

To compute a blended vertex position:

$$\mathbf{v}' = \mathbf{v}_{base} + \sum \phi_i \cdot \left(\Delta \mathbf{v}_i\right) \qquad \text{where } \Delta \mathbf{v}_i = \mathbf{v}_i - \mathbf{v}_{base}$$

Weighted Blending & Averaging

Weighted sum:

$$x' = \sum_{i=0} w_i x_i$$

Weighted average:

$$\sum_{i=0} w_i = 1$$

$$0 \le w_i \le 1$$

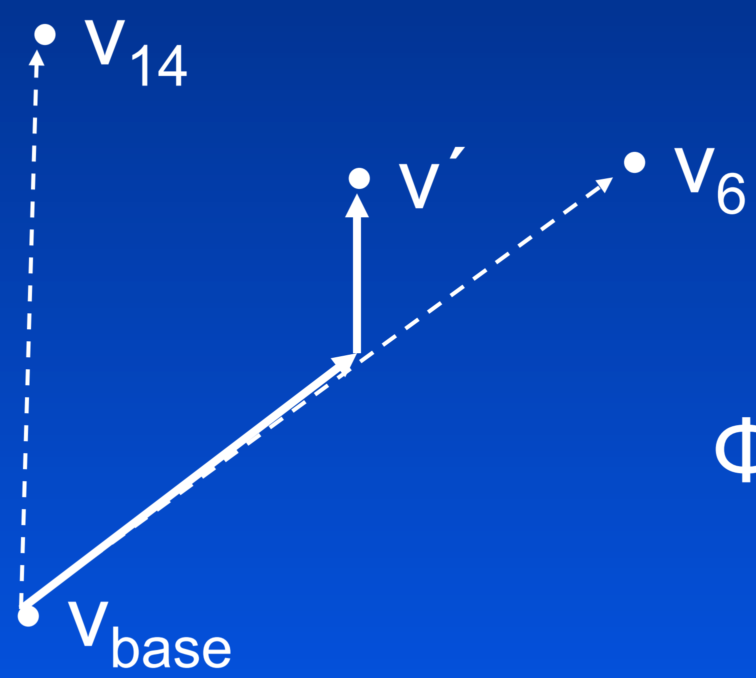
Convex average:

$$x' = x_0 + \sum_{i=1} w_i \left(x_i - x_0\right)$$

Additive blend:

$$= \left(1 - \sum_{i=1} w_i\right) x_0 + \sum_{i=1} w_i x_i$$

Additive Blend of Position



Φ_6=0.5

Φ_{14}=0.25

Normal Interpolation

To compute the blended normal:

$$\mathbf{n}^* = \mathbf{n}_{base} + \sum \phi_i \cdot (\mathbf{n}_i - \mathbf{n}_{base})$$

$$\mathbf{n}' = \frac{\mathbf{n}^*}{|\mathbf{n}^*|}$$

Note: if the normal is going to undergo further processing (i.e., skinning), we might be able to postpone the normalization step until later

Target Storage

Morph targets can take up a lot of memory.

The base model is typically stored in whatever fashion a 3D model would be stored internally (verts, normals, triangles, texture maps, texture coordinates…)

The targets, however, don't need all of that information, as much of it will remain constant (triangles, texture maps…)

Also, most target expressions will only modify a small percentage of the verts

Therefore, the targets really only need to store the positions and normals of the vertices that have moved away from the base position (and the indices of those verts)

Target Storage

Also, we don't need to store the full position and normal, only the difference from the base position and base normal

i.e., other than storing v_3, we store v_3-v_{base}

There are two main advantages of doing this:

- Fewer vector subtractions at runtime (saves time)
- As the deltas will typically be small, we should be able to get better compression (saves space)

Target Storage

- *In a pre-processing step, the targets are created by comparing a modified model to the base model and writing out the 'difference'*
- *The information can be contained in something like this:*

```
class MorphTarget {
    int NumVerts;
    int Index [ ];
    Vector3 DeltaPosition [ ];
    Vector3 DeltaNormal [ ];
}
```

Colors and Other Properties

In addition to interpolating the positions and normals, one can interpolate other per -vertex data:

- Colors
- Alpha
- Texture coordinates
- Auxiliary shader properties