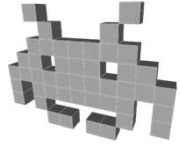


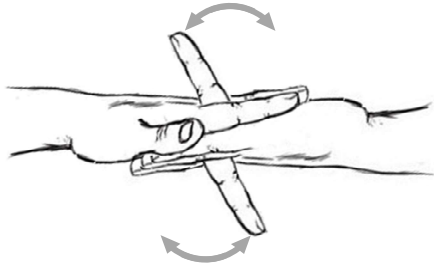
3D VideoGames  
Unimi

# Animations in games





---

Marco Tarini



1

## Course Plan



- lec. 1: **Introduction** ●
- lec. 2: **Mathematics** for 3D Games ●●●●●●
- lec. 3: **Scene Graph** ●
- lec. 4: **Game 3D Physics** ●●●● + ●●
- lec. 5: **Game Particle Systems** ▶
- lec. 6: **Game 3D Models** ●●
- lec. 7: **Game Textures** ▶●
- lec. 9: **Game Materials** ●
- lec. 8: **Game 3D Animations** 📍●●
- lec. 10: **Networking** for 3D Games ●
- lec. 11: **3D Audio** for 3D Games ●
- lec. 12: **Rendering Techniques** for 3D Games ●
- lec. 13: **Artificial Intelligence** for 3D Games ●

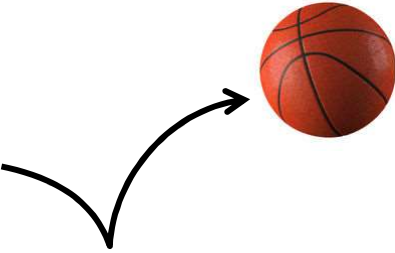
computer animation

2


## Computer animation in games

1. of rigid objects

- animate scene transformations



(6 DoF per object)


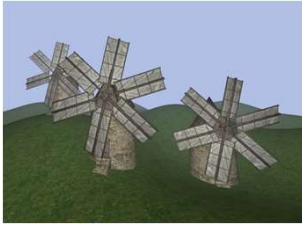




5

## Computer animation in games

1. of rigid objects

- or objects made of rigid sub-parts



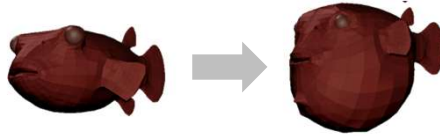
6

## Computer animation in games



### 2. Free-Form deformations

- generic transformations of the object



7

## Computer animation in games




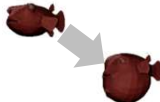


### 3. of articulated models




- internal skeleton
- most virtual characters!
- "skinning"



8

Types of animation and DoF (per keyframe)		DoF = Degrees of Freedom
<p>Rigid</p> 	6 DoF per object (or, e.g., 9, with anisotropic scaling)	
<p>Articulated</p> 	~50-100 DoF per object (e.g. 3 DoF per joint x 25 joints)	
<p>Free form</p> 	300-10.000 DoF per object (e.g. 3 per-vertex)	

9

Animations in games		
 <p><b>Authored</b></p> <ul style="list-style-type: none"> <li>● <b>Assets!</b></li> <li>● Control: easy. full control by artists (e.g. for dramatic effect)</li> <li>● Realism: hard it's up to the artist skill</li> <li>● Flexibility: little Doesn't adapt to env.</li> <li>● (consumes RAM)</li> </ul>	 <p><b>Procedural</b></p> <ul style="list-style-type: none"> <li>● <b>Physic engine</b></li> <li>● Control: hard</li> <li>● Realism: easy built-in physical laws</li> <li>● Flexibility: great Adapts to env. / context</li> <li>● (consumes GPU)</li> </ul>	

10

## A digression on terminology



Depending on the context, the *opposite* of “**procedural**”, is...

- **baked** :  
‘pre-cooked’, ‘computed-then-stored’, ‘frozen’ into an asset,  
(when something is produced procedurally);  
*as opposed to* : the game engine produces it on-the-fly/on-demand
- **asset** :  
stored as an asset (irrespective of origin),  
i.e., read by the game engine from the disk (or streamed from web);  
*as opposed to* : the game engine produced it on the fly/on-demand
- **scripted** :  
computed by a (*simple!*) script  
the procedure to create something can well be a script!  
*as opposed to* : produced by a full-fledged, complex simulation
- **authored / manually designed / manually edited** :  
made by a digital artist (*as opposed to*: by a program)
- **(fully) simulated** :  
output of a (*complex!*) simulation (*as opposed to*: by a simple one)  
e.g. “this anim. is just procedural, is not a real physical simulation”

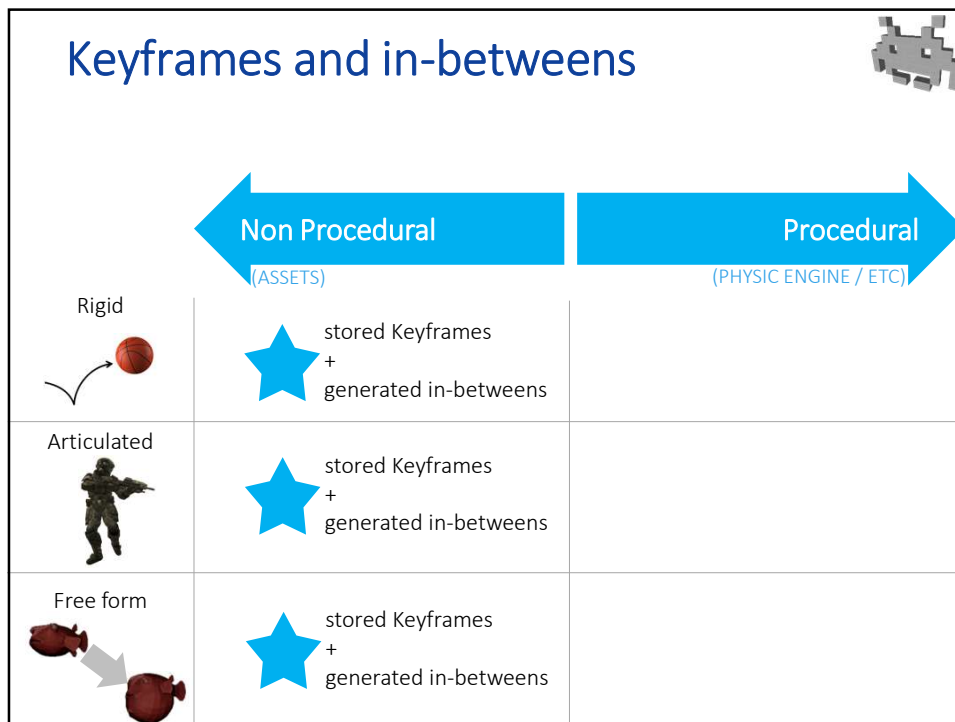
11

## Animations in games: authored, procedural... or a mix?

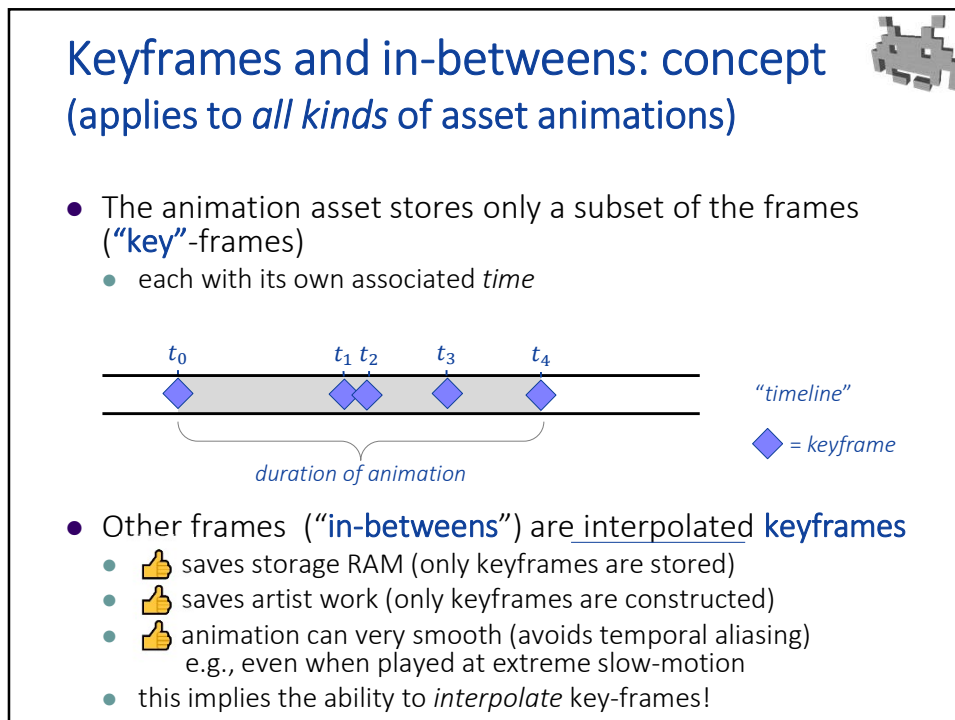


- A few examples of current commonly used mixes:
  - 1: “*primary*” animations: authored  
“*secondary*” animations: physically generated
  - 2: *alive* characters: authored  
*dead* characters: physically generated (“ragdolls”)
  - 3: walk cycle: authored (skeletal animation)  
exact *feet placement*: procedural (inverse kinematic)
  - 4: normal “behavior”, such as sparring: authored  
*gaze control* during sparring: procedural
  - 5: normal “behaviors” such as jumping, running: authored  
modifications / transitions: AI generated  
and more!
- mixing AI-generated with authored animations is a frontier in the field of Computer Animation

12



14



15

## Keyframes and in-betweens: concept (they apply to *all kinds* of asset animations)




the "temporal resolution" of the animation

- keyframes distribution can be *adaptive*
  - more keyframes only where needed
- in-betweening happens automatically on demand, in real time
  - e.g., at each refresh of video
- *times* associated to keyframe are chosen arbitrarily
  - not necessarily as an integer number, of video frames
  - all frames shown on screen will be in-betweens
- the better the interpolation schema  
→ the better in-betweens → the fewer keyframes are needed
- editing the animation:
  - editing individual keyframes
  - editing keyframe *times* (e.g., to achieve non-linearity of moment, vary speed)
  - 1. pick a new time  $t_i$  (not a keyframe)
  - 2. **bake** the in-between at  $t$  as a new keyframe
  - 3. edit it!

asset

16

## Animations in games (of 3D Solid Objects)

	← Non-Procedural (ASSETS)	→ Procedural (PHYSIC ENGINE / ETC)
Rigid 	Kinematic Animations	Rigid body dynamics
Articulated 	Skeletal Animations	Ragdolling    Inverse kinematics
Free form 	Blend-Shapes	<del>(general) soft-body simulation</del> <i>usually too expensive</i> Cloth/garments    Ropes

18

## Asset for free-form animations: Blend-shapes

- Also known as:
  - Morph-targets
  - Face-morphs
  - Shape-keys
  - Per-vertex animations
  - Vertex-animations
  - ...



BARRY BLITT (THE NEW YORKER)

19

## Blend shapes: concept



Walk cycle  
(Monkey Island  
LucasArt 1991)

- Animation in 2D (old school) games:  
a sequence of sprites
- Animation in 3D games:  
just a sequence of meshes?

20



### Mesh (data structure)

**connectivity (indexed)**

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

**geometry:**

Vert:	Pos
V1	(x, y, z)
V2	(x, y, z)
V3	(x, y, z)
V4	(x, y, z)
V5	(x, y, z)

**attributes:**

UV	Col
(u, v)	(r, g, b)
(u, v)	(r, g, b)
(u, v)	(r, g, b)
(u, v)	(r, g, b)
(u, v)	(r, g, b)

22

### Blend shapes (data structure)

**connectivity (indexed)**

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

**geometries:**

Vert:	Base Shape	Shape 1	Shape 2	...
V1	(x, y, z)	(x, y, z)	(x, y, z)	...
V2	(x, y, z)	(x, y, z)	(x, y, z)	...
V3	(x, y, z)	(x, y, z)	(x, y, z)	...
V4	(x, y, z)	(x, y, z)	(x, y, z)	...
V5	(x, y, z)	(x, y, z)	(x, y, z)	...

**attributes:**

UV	Col
(u, v)	(r, g, b)
(u, v)	(r, g, b)
(u, v)	(r, g, b)
(u, v)	(r, g, b)
(u, v)	(r, g, b)

23

## Blend shapes



- A mesh with several associated **geometries**
  - I.e. a sequence of meshes ('**shapes**') with
    - **shared connectivity**
    - **many shared attributes**
      - except normals / tangents dirs
      - shared UV-map, per vertex colors...
    - **different geometries**
    - (and **shared textures** as well)
  - **Encoding** (they are equivalent):
    - **Relative** mode:
      - *base shape*: stored as per-vertex positions (points)
      - any other *shape*: stored as difference with *base shape* (vectors)
    - **Absolute** mode:
      - each *shape* stored as per-vertex positions (points)
- aka '**morph**'  
 aka '**morph-target**'  
 aka (key)-'**frame**'  
 aka '**shape-key**'

24

## Blend shapes (as a data structure, e.g. C++)



- **Indexed** mesh :

```
class Vertex {
    vec3 pos;
    rgb color;
    vec3 normal;
};

class Face{
    int vertexIndex[3];
};

class Mesh{
    vector<Vertex> vert; /* geom + attr */
    vector<Face> tris; /* connectivity */
};
```

25

## Blend shapes

(as a data structure, e.g. C++)

- Blend-shape :

```
class Vertex {
    vec3 pos [ N_SHAPES ] ;
    rgb color;
    vec3 normal [ N_SHAPES ] ;
};

class Face{
    int vertexIndex[3];
};

class Mesh{
    vector<Vertex> vert; /* geom + attr */
    vector<Face> tris; /* connectivity */
};
```

26

All shapes of a blend-shape will always share...  
(so, a blend-shape *cannot* change)...

- The mesh connectivity
  - Eg. no change mesh res, remeshing
- Therefore, the surface topology
  - E.g. no breaking apart, fusing parts
- The mesh attributes
  - Such as color, UV-map...
  - Exceptions: positions, normals
- The textures
  - Use a texture animation instead?

27

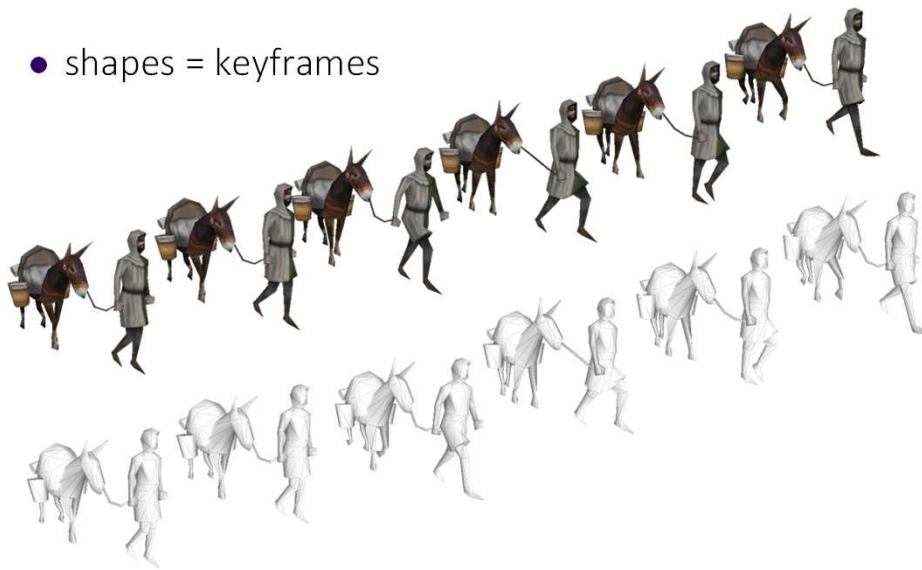
## Blend-shapes: a few common interchange formats

- Formats supporting blend-shapes include:
  - **.GLTF** (Khronos)  
“morphTarget”, relative encoding
  - **.DAE** (Collada)
  - **.FBX** (Autodesk)
- Older / simpler alternatives:
  - **.MD5** (“quake”, valve)
  - or, just store a sequence of meshes (es **.OBJ**)
    - making sure connectivity is coherent!  
(vertex, face ordering must be the same – can be tricky)

28

## Uses of Blend-Shapes: temporal sequences

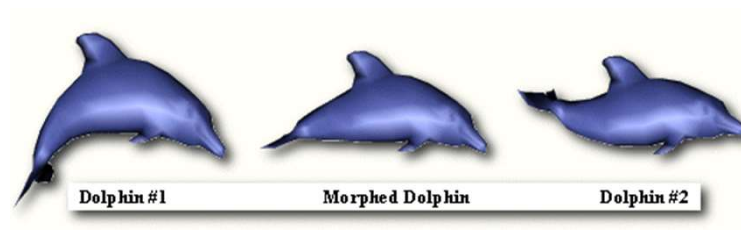
- shapes = keyframes



29





## Uses of Blend-Shapes: temporal sequences

- Temporal sequences
  - shapes = keyframes







30

## Blending keyframes of a temporal sequence

- shapes = keyframes of the animation
  - $shape_A$   with time  $t_A$
  - $shape_B$   with time  $t_B$
  - $shape_C$   with time  $t_C$
  - $shape_D$   with time  $t_D$
- given current time  $t$  with  $t_B \leq t \leq t_C$
- then...
  - which shapes to blend?  $shape_B$  ,  $shape_C$
  - weights?  $w_B = \frac{t - t_C}{t_B - t_C}$      $w_C = (1 - w_B) = \frac{t - t_B}{t_C - t_B}$

31

## Blending keyframes of a temporal sequence with transition functions

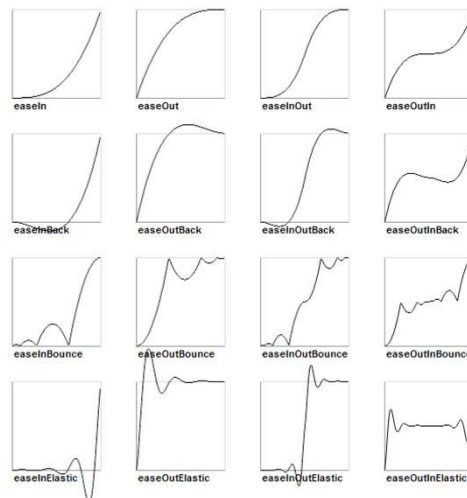
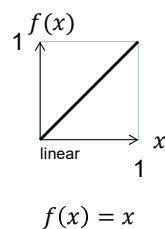
- shapes = keyframes of the animation
  - $shape_A$   with time  $t_A$
  - $shape_B$   with time  $t_B$
  - $shape_C$   with time  $t_C$
  - $shape_D$   with time  $t_D$
- given current time  $t$  with  $t_B \leq t \leq t_C$
- then... *transition function*
  - which shapes to blend?  $shape_B$ ,  $shape_C$
  - weights?  $w_B = f\left(\frac{t - t_C}{t_B - t_C}\right)$   $w_C = (1 - w_B)$

32

## Transition functions

(applies to all animation types with keyframes)

- Not necessarily the Linear one

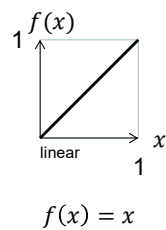



33

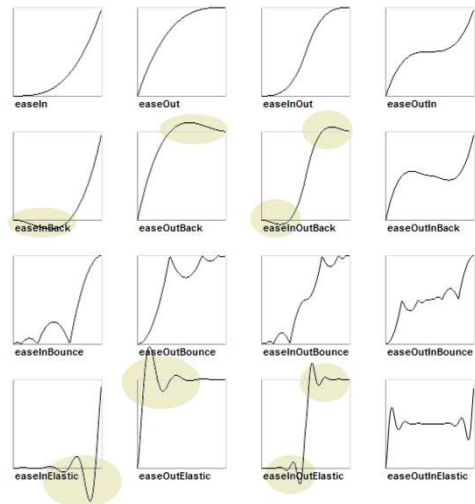
## Transition functions

(applies to all animation types with keyframes)

- Not necessarily the Linear one



NB:  = extrapolation !  
i.e. exaggeration



34

## Uses of Blend-Shapes: facial expressions / animations



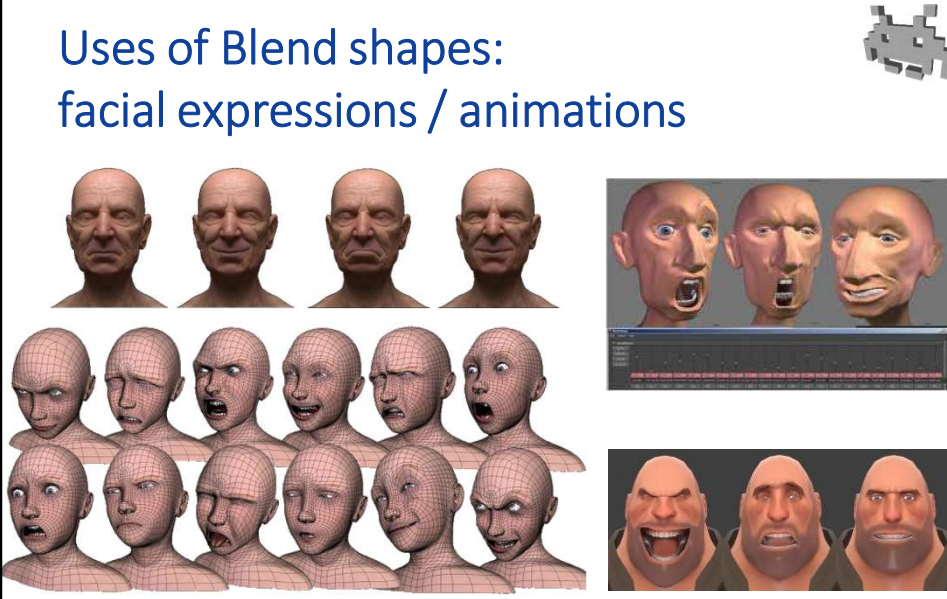
shape A

shape B

here: shapes = facial expressions  
(typical use; that's why they are also called "face morphs")

35

## Uses of Blend shapes: facial expressions / animations



Here, used together with skeletal animations (see next lecture)  
(for mandible, neck, eyeballs)

38

## About the two ways to encode a blend-shape

- The **absolute mode** is natural when shapes are designed to be used as *alternatives*.  
Example:
  - keyframes of an animation sequence.  
 $\text{shape}_1 = \text{keyframe of time 5}$   
 $\text{shape}_2 = \text{keyframe of time 9}$   
 at time 6, we use ....  $0.75 \text{ shape}_1 + 0.25 \text{ shape}_2$
- The **relative mode** is natural when shapes are designed to be *superimposed* with various degrees of strength.  
Example:
  - $\text{shape}_1 = \text{left-eye closed}$
  - $\text{shape}_2 = \text{smile}$
  - $\text{base} + \text{shape}_1 + \text{shape}_2 = \text{wink}$
 Another example
  - $\text{shape}_1 = \text{fat}$
  - $\text{shape}_2 = \text{long chin}$
  - $\text{base} + 0.4 \text{ shape}_1 + 0.9 \text{ shape}_2 = \text{a bit fat \& long-ish chin}$
- But the two ways are equivalently expressive.
  - They can achieve the same set of shapes
  - In **absolute mode**,  $\sum w_i = 1$
  - In **relative mode**, there is no such condition, but it's equivalent to use the absolute mode, with  $w_{\text{base}} = 1 - \sum w_i$
  - when  $\sum w_i > 1$ , this means that we are implicitly using an **extrapolation**

39



### Blending shapes of a blend-shape

What is stored		
Equivalent ways to blend...	<p>two shapes <math>i</math> and <math>j</math></p> $w_i S_i + w_j S_j$ <p>three shapes <math>i, j</math> and <math>k</math></p> $w_i S_i + w_j S_j + w_k S_k$ <p>etc</p> <p style="text-align: center;">with <math>\sum w = 1</math></p>	<p>two shapes <math>i</math> and <math>j</math></p> $S_b + w_i R_i + w_j R_j$ <p>three shapes <math>i, j</math> and <math>k</math></p> $S_b + w_i R_i + w_j R_j + w_k R_k$
	using Absolute Encoding	using Relative Encoding

41

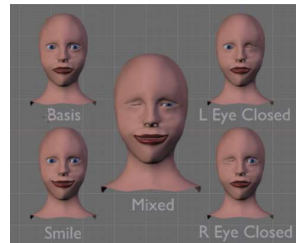
### Blending shapes of a blend-shape

What is stored		
Equivalent ways to blend...	<p>base shape with one shape <math>i</math></p> $(1 - w)S_b + w S_i$ <p>base shape with two shapes <math>(i, j)</math></p> $(1 - w_i - w_j)S_b + w_i S_i + w_j S_j$ <p>base shape with three shapes</p> $(1 - w_i - w_j - w_k)S_b + w_i S_i + w_j S_j + w_k S_k$	<p>base shape with one shape <math>i</math></p> $S_b + w R_i$ <p>base shape with two shapes <math>(i, j)</math></p> $S_b + w_i R_i + w_j R_j$ <p>base shape with three shapes</p> $S_b + w_i R_i + w_j R_j + w_k R_k$
<del><math>\sum w = 1</math></del>	using Absolute Encoding	using Relative Encoding

42

## Example

- Shape 0 = base shape
- Shape 1 = smile
- Shape 2 = left eye closed
- Shape 3 = right eye closed



«wink» = 75% smile, right eye fully closed

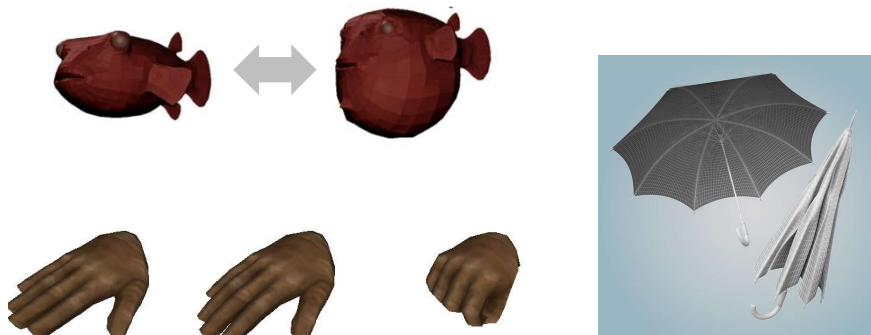
How to blend the shapes to get a wink (which weight to use), if the blend-shape is encoded in...

- A: Relative mode?
- B: Absolute mode?

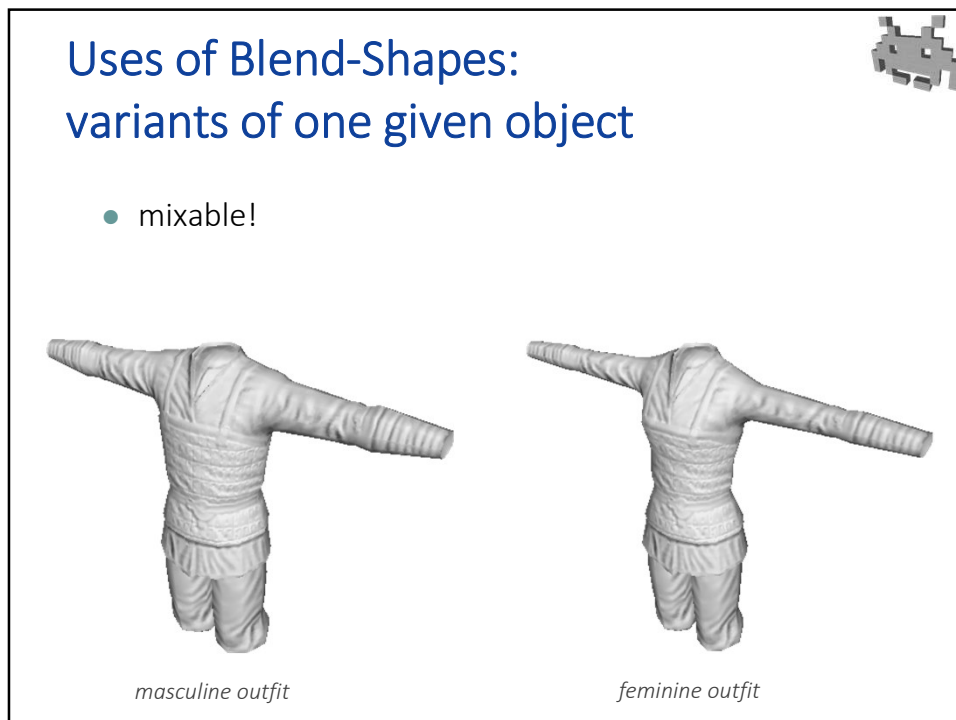
43

## Uses of Blend-Shapes: generic deformations

- Baked poses



48



49

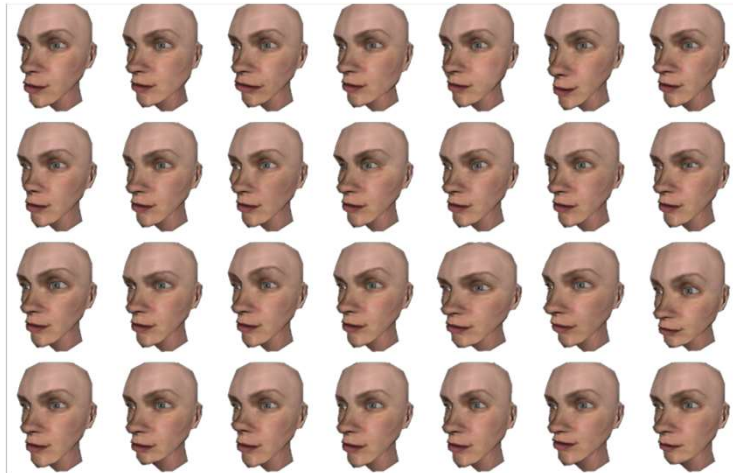


50

## Uses of Blend shapes

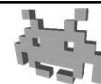


- A **blend shape** modelling a **face space** (“face-morphs”)



52

## Uses of Blend shapes



- Provide a range of variations for a given class of model (e.g. a characters)
- Many games (especially RPGs) will use this as a part of a mechanism to allow players to customize their character appearance (custom avatars)
  - At the end of the process, the result of the blending can be **baked** into a standard mesh!
- See <http://www.makehumancommunity.org/> for a good open-source suite (to create humanoid meshes) based on this concept



53

## Blend shapes: authoring

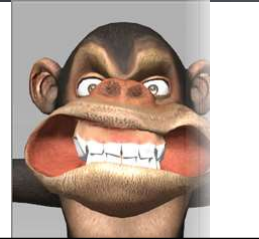
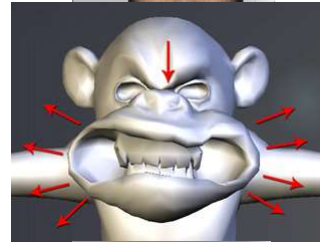
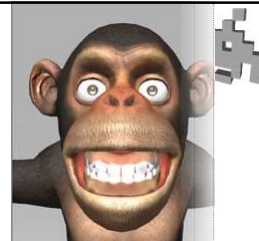
### 1. Editing base shape

- including:  
uv-mapping, texturing, etc.

### 2. Re-edit it

for each shape-key!  
...while preserving:  
connectivity,  
textures, etc:

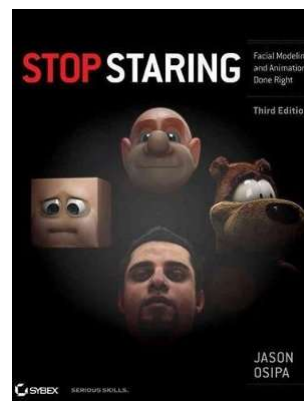
- with low poly editing
- or with subdivision surfaces...
- or with parametric surfaces...
- or with sculpting.



54


## Blend shapes: authoring

- Handbook for blend-shape based face animation:
  - "Stop Staring" (3d edition)  
Jason Osipa
  - Covers: style, expression...
  - Non technical (high level)
  - Not about specific tools  
e.g. Blender, Maya




55

## Blend shapes: pros and cons



- During authoring:
  - 👍 flexible, expressive, huge number of DOF... (too many?)
  - 👎 work intensive to construct
  - 👎 expensive to store
- During use (by animators)
  - 👍 easy to use (just define global weights)
  - 👎 RAM cost
  - 👎 very little degree of freedoms (too few?)


but, not as bad as old sprites,



because  
(1) shared of connectivity, textures, attributes  
(2) keyframes / inbetweens!

58

## Blend shapes: open challenges



- Capturing:
  - from a stream of meshes
  - e.g. : from a RGBD camera (like Microsoft Kinect) to a blend-shape: open problem!
- Compression
  - e.g.: reduce number of keyframes (can you think of an algorithm?)
- Streaming
  - server sends animation to client while it runs
- LOD-ding
  - like for meshes (but more difficult: same connectivity must be good for all shapes)

59