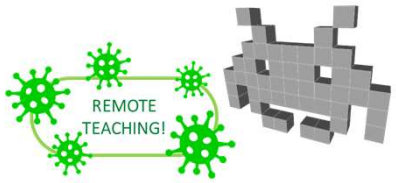
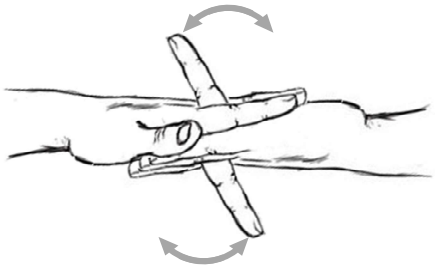
3D VideoGames
Unimi

Animations in games

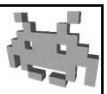


Marco Tarini



1

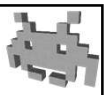
Course Plan



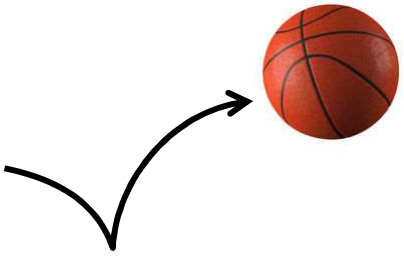
lec. 1: **Introduction**
lec. 2: **Mathematics** for 3D Games
lec. 3: **Scene Graph**
lec. 4: Game **3D Physics** +
lec. 5: Game **Particle Systems**
lec. 6: Game **3D Models**
lec. 7: Game **Textures**
lec. 8: Game **3D Animations**
lec. 9: Game **3D Audio**
lec. 10: **Networking** for 3D Games
lec.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

2

Types of animations in games



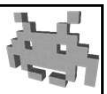
1. of rigid objects
   - animate scene transformations



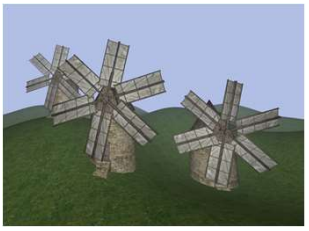
(6 DoF per object)

3

Types of animations in games

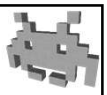


1. of rigid objects
   - or objects made of rigid sub-parts

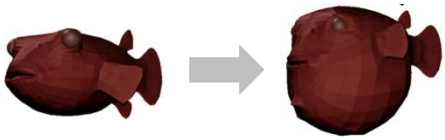


4

Types of animations in games

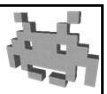


2. Free-Form deformations
   - generic transformations of the object



5

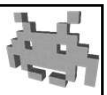
Types of animations in games



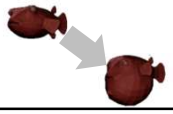
3. of articulated models
   - internal skeleton
   - most virtual characters!
   - "skinning"

6

Types of animation and DoF (per keyframe)

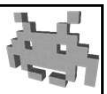


DoF = Degrees of Freedom

Rigid 	6 DoF per object (or, e.g., 9, with anisotropic scaling)
Articulated 	~50-100 DoF per object (e.g. 3 DoF per joint x 25 joints)
Free form 	300-10.000 DoF per object (e.g. 3 per-vertex)

7

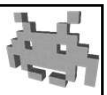
Animations in games



Authored	Procedural
• Assets!	• Physic engine
• Control: easy. full control by artists (e.g. for dramatic fx)	• Control: hard
• Realism: hard it's up to the artist skill	• Realism: easy built-in physical laws
• Flexibility: little Doesn't adapt to env.	• Flexibility: great Adapts to env. / context
• (consumes RAM)	• (consumes GPU)

8

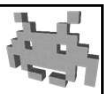
A digression on terminology



- depending on the context,
  the opposite of "procedural", can be…
  - baked :
    'pre-cooked', 'frozen' into an asset,
    (when something was produced procedurally)
  - asset :
    stored as an asset (irrespective of origin),
    that is, read from the disk (or streamed from web)
  - scripted :
    stored as a (*simple!*) script
    but, the procedure to create something can well be a script!
    e.g. "this *script procedurally generates* a level"
  - authored / manually designed / edited :
    made by a digital artist (as opposed to, by a program)
  - (fully) simulated :
    the output of a (*complex!*) simulation

9

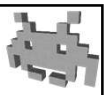
Summary: Types of authored animations



- of objects made of rigid subparts
  - including joints: robots, cars…
  - → "forward kinematics animations"
    (dynamic changes of modelling transform)
- of deformable articulated objects
  - with some internal skeleton
  - e.g: most virtual characters:
    humans / animals / monsters / anything in between
  - → "skinning" / "rigging"
- of generic deformable objects
  - e.g. faces, an umbrella, stuff with membrane…
  - "per-vertex animations" / "blend shapes" / "morph targets"

10

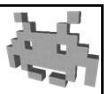
Animations in games: authored or procedural?

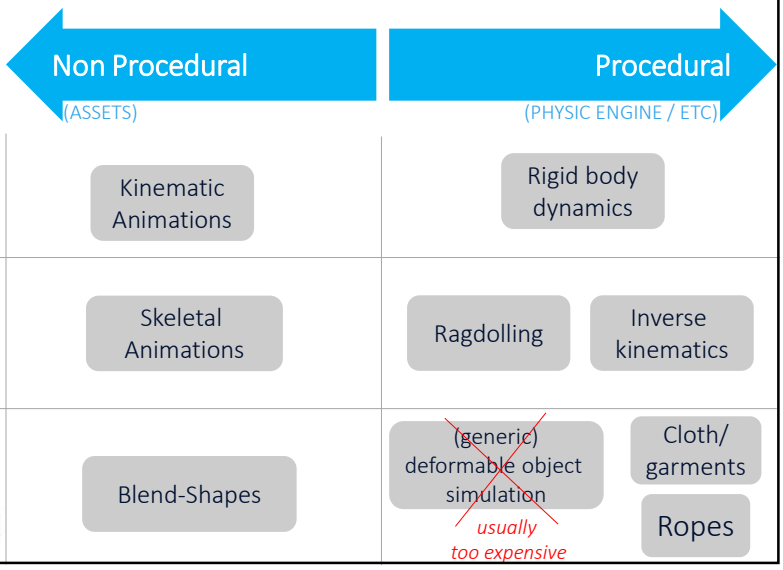


- Or… a mix
  - Example 1: *primary* animations: authored
    secondary animations: physically generated
  - Example 2: *alive* characters: scripted
    dead characters: physically generated (ragdolls)
  - Example 3: walk cycle: authored (skeletal animation)
    feet placement: procedural (inverse kinematic)
  - Example 4: normal "behavior", e.g. sparring: authored
    gaze control: scripted
  - Example 5: normal "behaviors" e.g. jumping, running
    modifications / transitions: AI generated
  - *and more*
- mixing AI with authored animations is a frontier in CG and Interactive techniques!

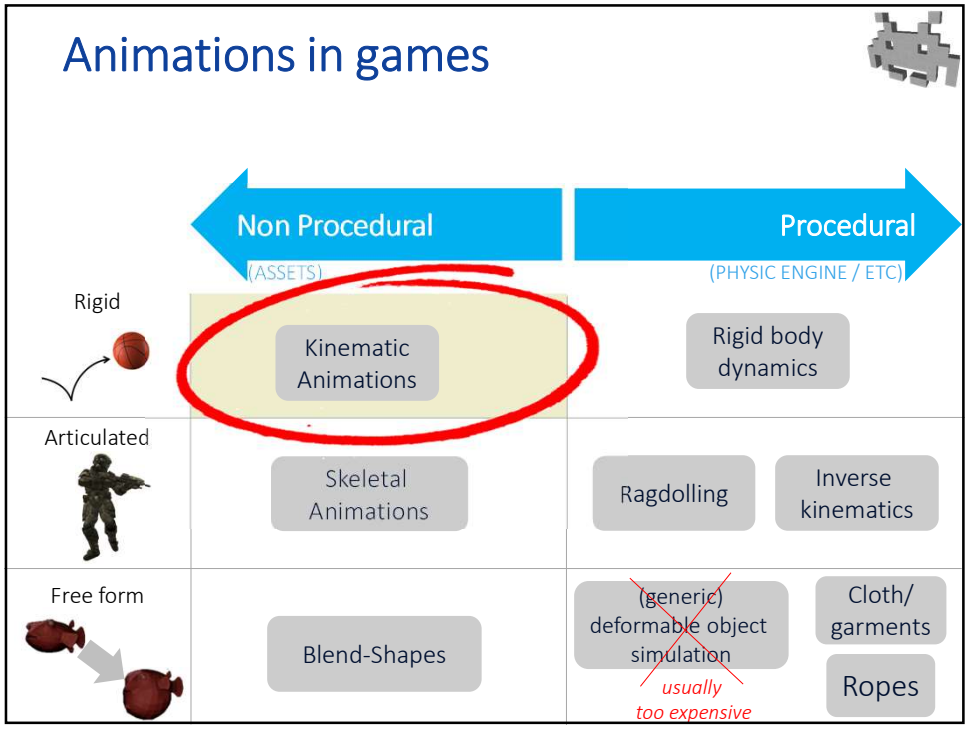
11

Animations in games

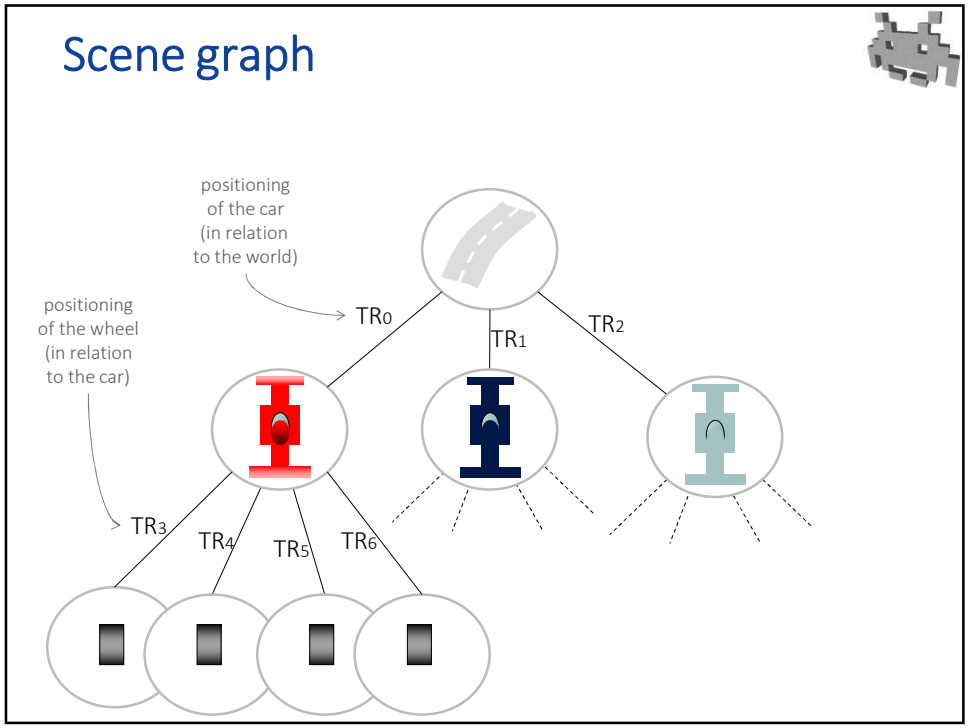




12

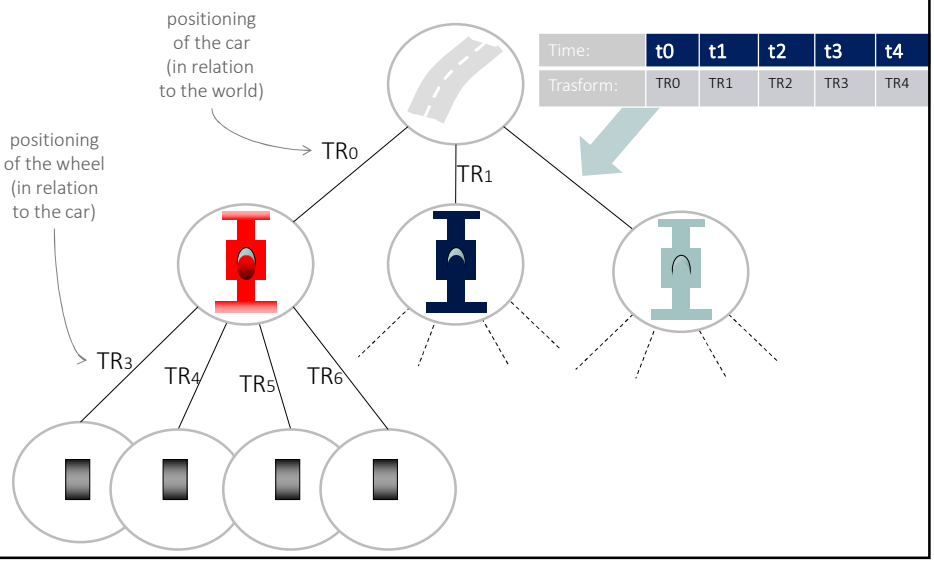

Animations in games
Non Procedural
(ASSETS)
Procedural
(PHYSIC ENGINE / ETC)
Rigid
Articulated
Free form
Kinematic Animations
Rigid body dynamics
Skeletal Animations
Ragdolling
Inverse kinematics
Blend-Shapes
(generic) deformable object simulation
usually too expensive
Cloth/ garments
Ropes

13


Scene graph
positioning of the car (in relation to the world)
positioning of the wheel (in relation to the car)
TR0
TR1
TR2
TR3
TR4
TR5
TR6

14

Animated Scene graph… (“kinematic” animations)



15

Kinematic animations: how?

- way 1:
  - just scripting
- way 2:
  - editing in a animation software
    - cinema 4D, blender, 3D max, …
    - (including use of I.K. as part of the interface)
  - export animation
    - as a sequence of keyframes
    - File formats: collada, fbx, …

asset: the script

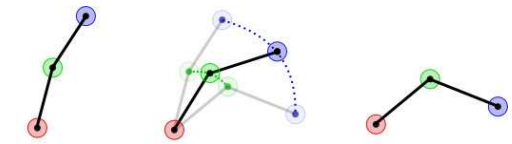
Time:	t0	t1	t2	t3	t4
A==>B:	TR0	TR1	TR2	TR3	TR4
B==>C:	TR0	TR1	TR2	TR3	TR4

asset: the animation

16

Interpolating keyframes
(applies to *all kinds* of asset animations)

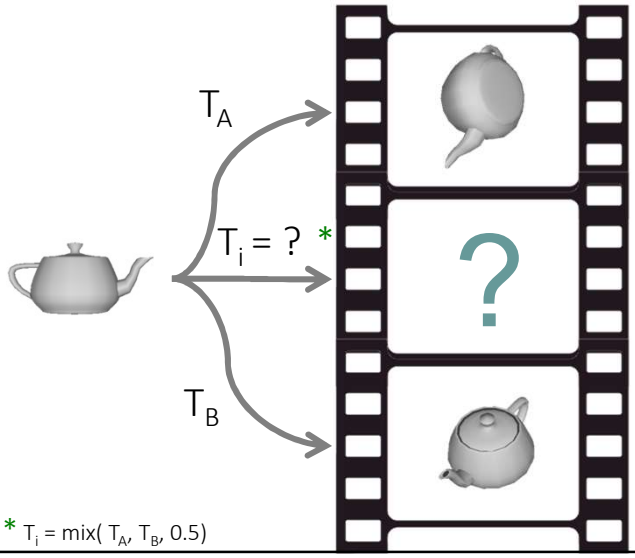
- Keyframes
  +
  in-betweens (interpolation)



keyframe A | 0.5 · keyframe A + 0.5 · keyframe B | keyframe B

17

Keyframe interpolation
(for kinematic animations)



time A = 100
keyframe A

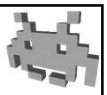
time curr. = 150
interpolated

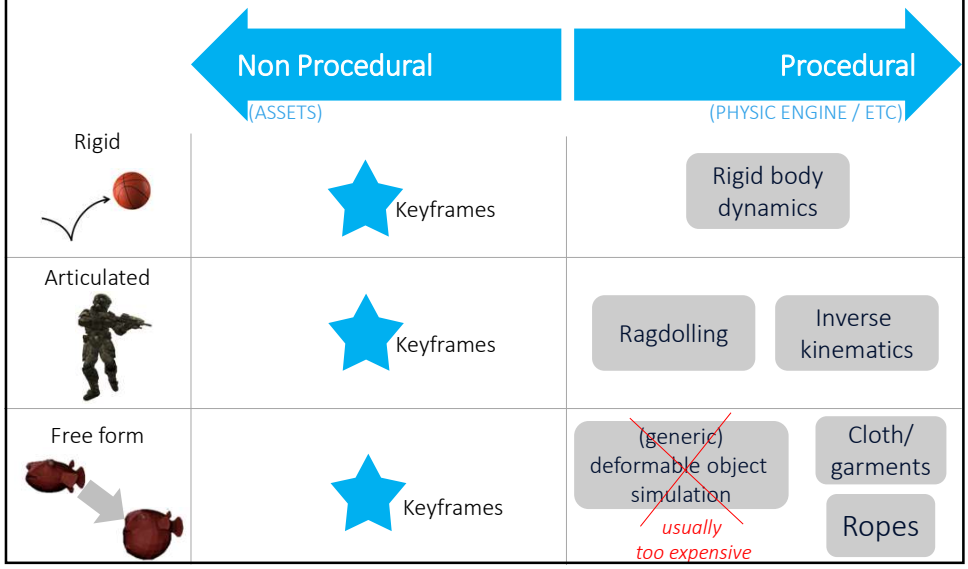
time B = 200
keyframe B

* T_i = mix(T_A, T_B, 0.5)

18

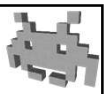
Keyframes and inbetweening



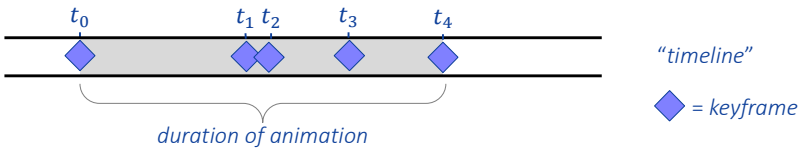


19

Keyframes and in-betweens
(applies to *all kinds* of asset animations)



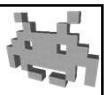
- The animation asset stores only a subset of frames ("key"-frames)
  - each with its own associated *time*



- Other frames ("inbetweens") are interpolated keyframes
  - saves storage RAM
  - saves artist work (only keyframes are constructed)
  - animation can very smooth (avoids temporal aliasing)
    e.g. even when played at extreme slow-motion
  - requires ability to interpolate (key) frames!

20

Keyframes and in-betweens
(applies to *all kinds* of asset animations)



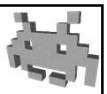
the "temporal resolution" of the animation

- keyframes distribution can be *adaptive*
  - more keyframes only where needed
- inbetweening happens on demand
  - e.g. at each refresh rate of videoframe
- keyframe *times* can be at arbitrary
  - not necessarily exact frames, not necessarily integers
  - all frames shown on screen will be inbetweens
- the better the interpolation schema
  → better in-betweens → fewer keyframes are needed
- editing the animation:
  - editing individual keyframes
  - editing keyframe *times* (e.g. achieves non-linearity of speed)
  - pick a new time t_i (not a keyframe)
    bake the inbetween at *t* as a new keyframe
    edit it!

asset

21

Kinematic animations

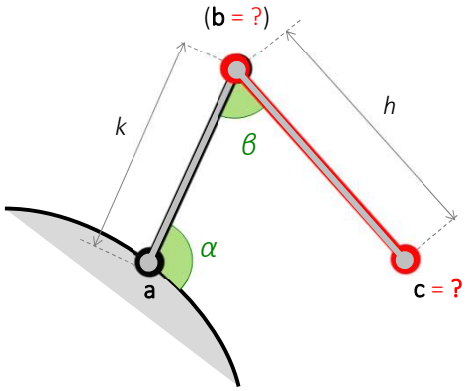


for certain nodes in the scenegraph

- Just compute new transformations per frame
  - Often, just the rotation component (translation is constant)
- Or store transformations per keyframe
  - Then, interpolate them for any other frame *between* keyframes
- By cumulating the transformations in the graph, we can compute the final position of every node
  - This is called solving a "forward kinematic" problem
  - The inverse problem (from final position of certain nodes, compute the transform, especially the rotation) is called "inverse kinematic" (IK)

22

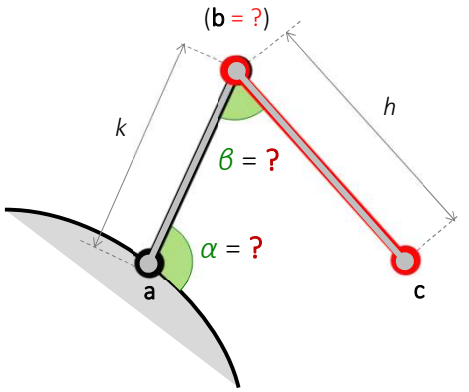
"Forward Kinematic"



find **c** (and **b**), given **a**, α, β, k, h

23

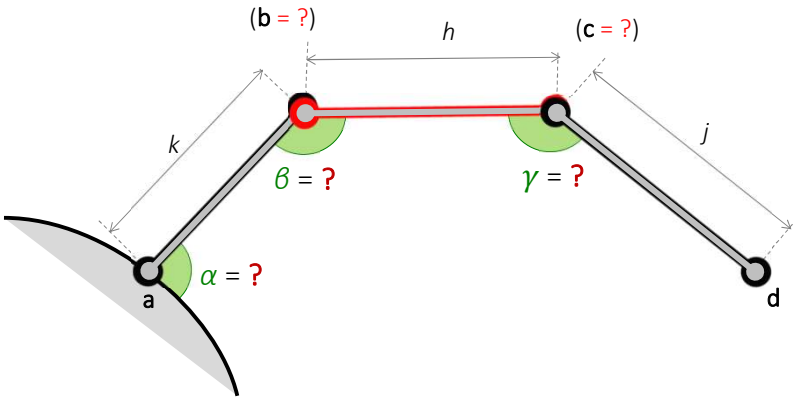
Inverse Kinematic (IK)



find α, β (and **b**), given **a**, **c**, k, h

26

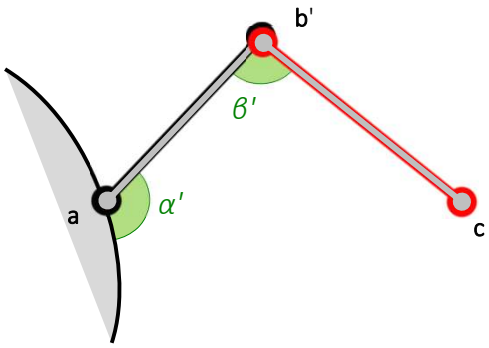
Inverse Kinematic (IK):
for any number of joints



solve for α , β , γ (and **b**, **c**), given **a**, **d**, k, h, j

29

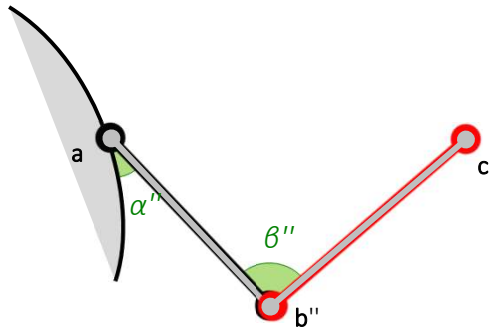
Inverse Kinematic (IK):
an ambiguity



solve for α, β (and **b**), given **a**, **c**, k, h

30

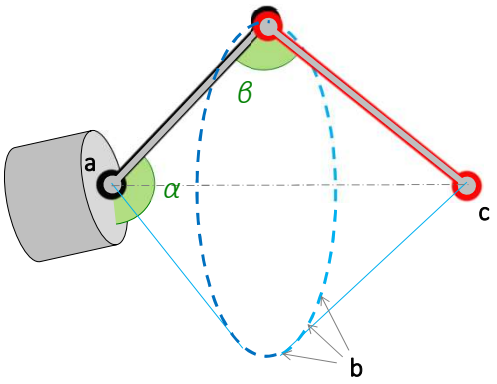
Inverse Kinematic (IK): an ambiguity



solve for α, β (and **b**), given **a**, **c**, *k*, *h*

31

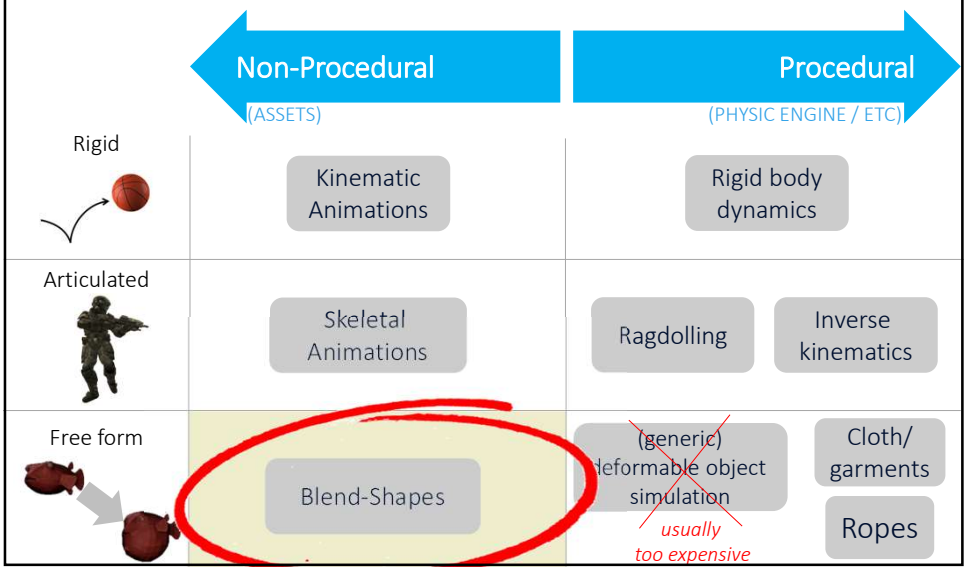
Inverse Kinematic (IK) in 3D: more ambiguities



solve for α, β (and **b**), given **a**, **c**, *k*, *h*

32

Animations in games
(of 3D Solid Objects)



33

Asset for free-form animations: Blend-shapes

- A.K.A:
  - Blend-shapes
  - Per-vertex animations
  - Vertex-animations
  - Face-morphs
  - Shape-keys
  - Morph-targets
  - …

BARRY BLITT (THE NEW YORKER)

34

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?

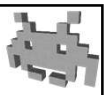
35

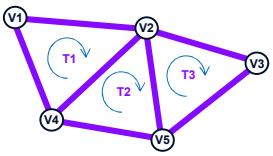
Reminder: representation of a mesh

- Indexed mode :
  - Geometry: array of vertex positions
  - Attributes:
    - stored at vertices
  - Connectivity:
    - Array of triangles (or polygons)
    - Each triangle = a triplet of indexes to vertice

36

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:

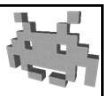
Vert:	Pos
V1	(x,y,z)
V2	(x,y,z)
V3	(x,y,z)
V4	(x,y,z)
V5	(x,y,z)

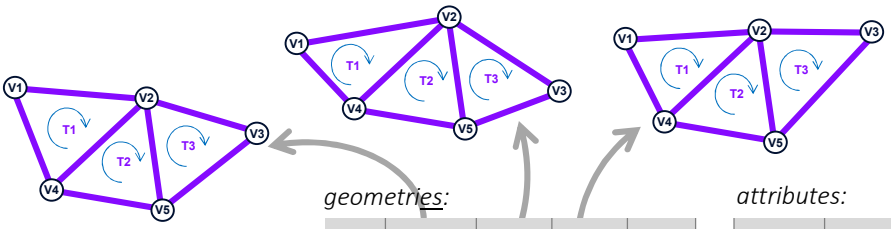
attributes:

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)

37

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:

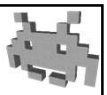
Vert:	Base Shape	Shape 1	Shape 2	...
V1	(x,y,z)	(x,y,z)	(x,y,z)	...
V2	(x,y,z)	(x,y,z)	(x,y,z)	...
V3	(x,y,z)	(x,y,z)	(x,y,z)	...
V4	(x,y,z)	(x,y,z)	(x,y,z)	...
V5	(x,y,z)	(x,y,z)	(x,y,z)	...

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)

38

Blend shapes

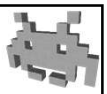


- A mesh with several associated geometries
- I.e. a sequence of meshes ('shapes') with
  - shared connectivity
  - many shared attributies
    - except normals / tangents dirs
    - shared UV-map, per vertex colors…
  - different geometries
  - (and shared textures as well)
- Variants (they are equivalent):
  - 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 (key)-'frame'
aka 'shape-key'

39

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



- Indexed mesh :

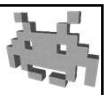
```
class Vertex {
  vec3 pos;
  rgb color;
  vec3 normal;
};

class Face{
   int vertexIndex[3];
};

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

40

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



- Blend-shape :

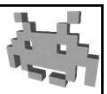
```
class Vertex {
  vec3 pos [ N_SHAPES ] ;
  rgb color;
  vec3 normal [ N_SHAPES ] ;
};

class Face{
   int vertexIndex[3];
};

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

41

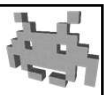
Blend-shapes:
most common file formats



- Simple:
  - .MD5 (“quake”, valve)
  - or, just store a sequence of meshes (es .OBJ)
    - making sure connectivity is coherent!
      (vertex ordering = the same)
- Complex:
  - .DAE (Collada)
  - .FBX (Autodesk)

42

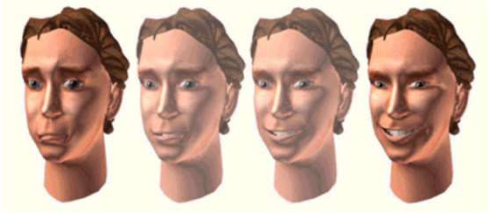
Blending shapes of a blend-shape



- Blending between two $shape_A$ and $shape_B$:
  - with relative BlendShapes:
    $shape_{base} + w_A \cdot delta\_shape_A + w_B \cdot delta\_shape_B \cdot w_B$
    - with w_A , w_B usually between 0 and 1
    - otherwise, it's an "exaggeration"
  - with absolute BlendShapes : (equivalently) (verify!)
    $(1- w_A - w_B) \cdot shape_{base} + w_A \cdot delta\_shape_A + w_B \cdot delta\_shape_B$
    - Note: it's an extrapolation if $0 \leq w_A , w_B , w_A + w_B \leq 1$
    - Otherwise, it-s an extrapolation
- It's trivial to extend to a blend of multiple shapes

43

Uses of Blend-Shapes: facial expressions



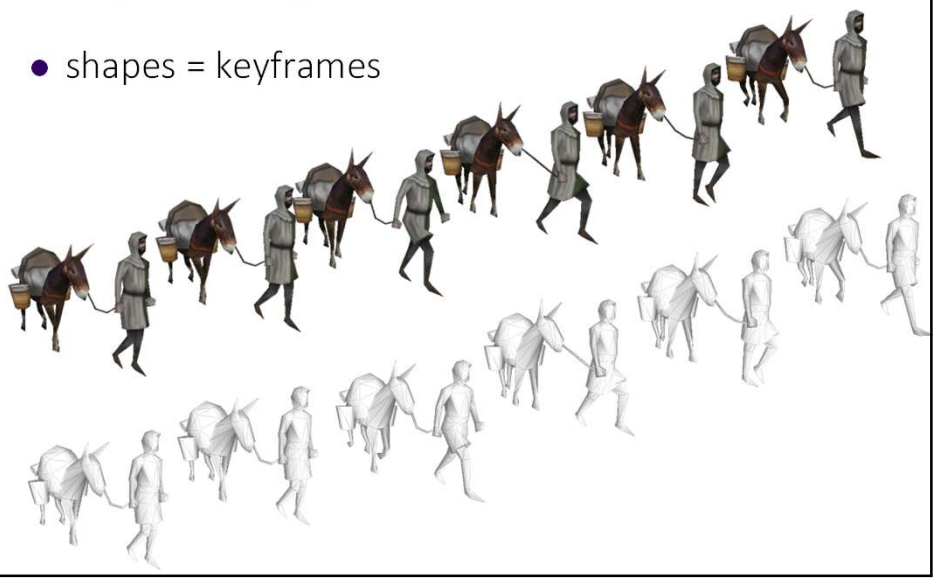
shape A shape B

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

45

Uses of Blend shapes: temporal sequences

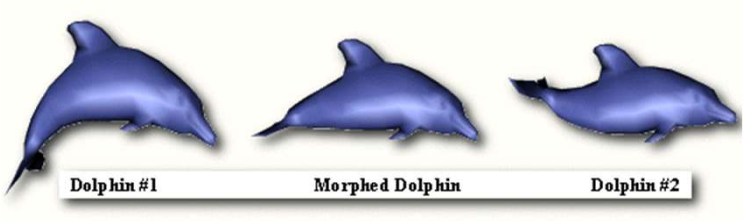
- shapes = keyframes



46

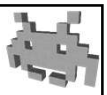
Uses of Blend shapes: temporal sequences

- Temporal sequences
  - shapes = keyframes



47

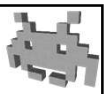
Blending keyframes of a temporal sequence



- 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 $t_B < t < t_C$
- *then…*
  - which shapes to blend? shape_B , shape_C
  - weights? $w_B = \frac{t - t_C}{t_B - t_C}$ $w_C = (1 - w_B) = \frac{t - t_B}{t_C - t_B}$

48

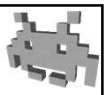
Blending keyframes of a temporal sequence with transition functions



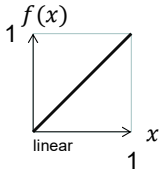
- shapes = keyframes of the animation
  - shape_A  with time t_A
  - shape_B  with time t_B
  - shape_C  with time t_C
  - shape_D  with time t_D
- given current time t with $t_B < t < t_C$
- *then…* *transition function*
  - which shapes to blend? shape_B , shape_C
  - weights? $w_B = f\left(\frac{t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

49

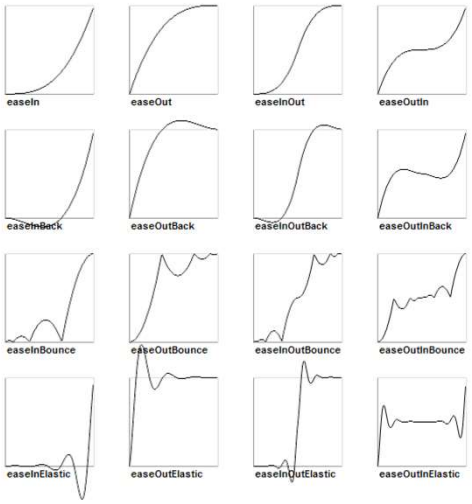
Transition functions
(applies to all animation types with keyframes)



- Not necessarily the Linear one

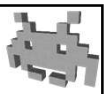


$f(x) = x$

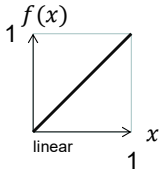


50

Transition functions
(applies to all animation types with keyframes)

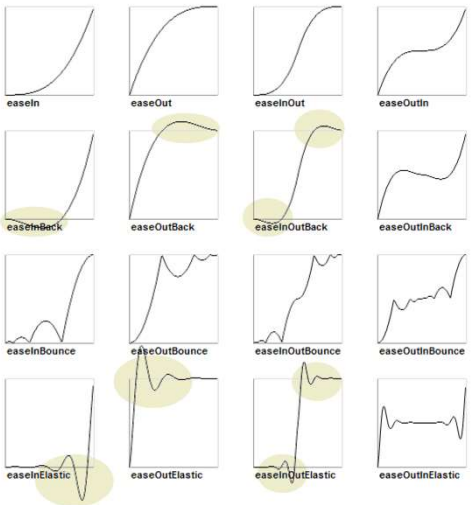


- Not necessarily the Linear one



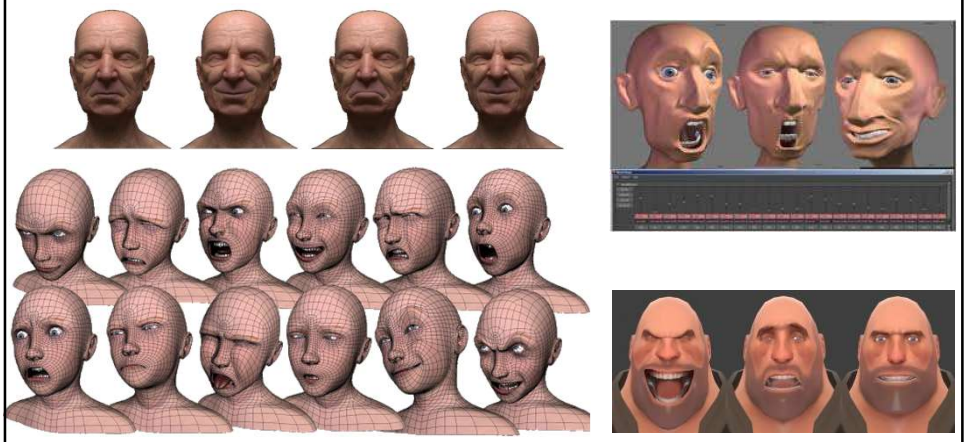
$f(x) = x$

NB: = extrapolation !
i.e. exaggeration



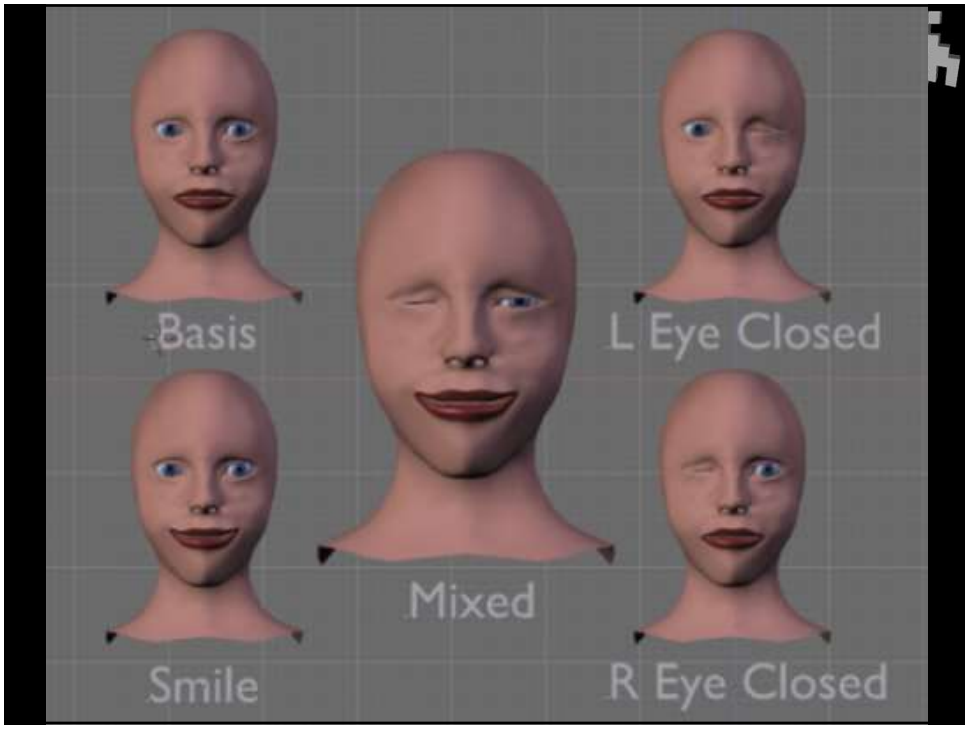
51

Uses of Blend shapes: facial animations



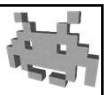
Here, used together with skeletal animations (see next lecture)
(for mandible, neck, eyeballs)

54



55

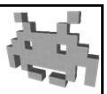
Using facial animations as Blend shapes



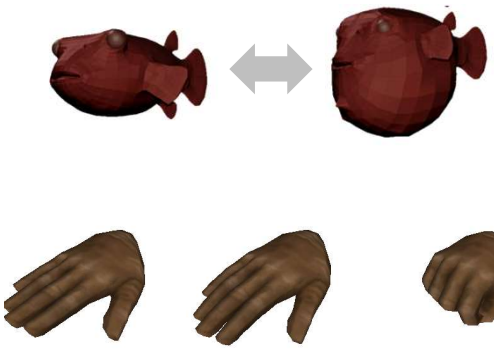
- 3D Modeller authors:
  the set of blend-shapes
- Animator (of expressions) picks:
  weights
  - eg.: with sliders
  - assisted / substituted by automatisms
    - lip sync
    - dynamically determined expressions
- Keyshape Blending: by rendering engine

58

Uses of Blend-Shapes: generic deformations

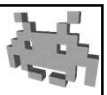


- Baked poses



59

Uses of Blend-Shapes: variants of one given object



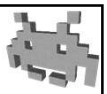
- mixable!

masculine outfit

feminine outfit

60

Uses of Blend-Shapes variants of one given object



- mixable!

human

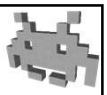
orc

goblin

dwarf

61

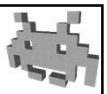
Uses of Blend-Shapes



- Defines shapes of a class of objects
  - get a shape in the class = just choose the weights
    - 3D modelling at a high-level of abstraction
  - the weights "span" one shape space
    - one given shape = one point in the space
    - weights = coords
  - the space is the more useful the more:
    - *all and only* the reasonable shapes are represented in the space
- Typical Example: face morphologies
  - "face-space"
  - note: face morphology ≠ facial expression

62

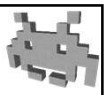
Uses of Blend shapes



- A blend shape modelling a face space ("face-morphs")

63

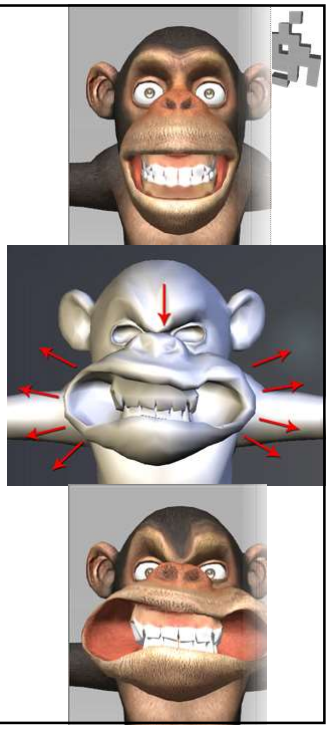
What a blend shape *cannot* do



- Change connectivity
  - eg. change mesh res, remeshing
- Change topology
  - breaking apart, fusing parts
- Change most attributes
  - eg color, UV-map…
- Change textures
  - Use a texture animation instead?

64

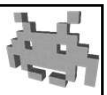
Blend shapes: authoring



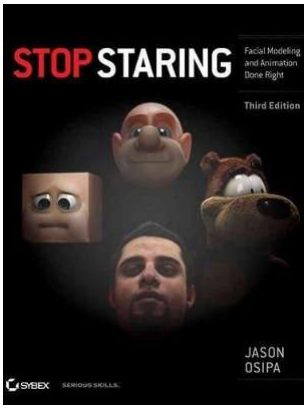
1. Editing base shape
   - including:
     uv-mapping, texturing, etc.
2. Re-edit it
   for each shape-key!
   …while preserving:
   connectivity,
   textures, etc:
   - with low poly editing
   - or with subdivision surfaces…
   - or with parametric surfaces…
   - or with scupting.

65

Blend shapes: authoring

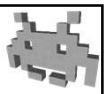


- Handbook for blend-shape based face animation:
  - "Stop Staring" (3d edition) Jason Osipa
  - Covers: style, expression…
  - Non technical (high level)
  - Not about specific tools e.g. Blender, Maya



66

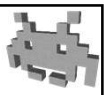
Blend shapes: hot to obtain them



- Capture:
  - 3D acquisition of base shape B0
    - (including: simplification, remeshing, uv-mapping, etc)
  - capture subsequent shapes B1, B2…
    - e.g. real-time (kinect), o 3D scanning for each shape
  - compute a morph B0 => B1
    - "non rigid mesh alignment"

67

Blend shapes: pros and cons



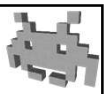
- During authoring:
  - 👍 flexible, expressive, huge number of DOF… (too many?)
  - 👎 work intensive to construct
  - 👎 expensive to store
- During use (by animator)
  - 👍 easy to use (just define global weights)
  - 👎 RAM cost
  - 👎 very little degree of freedoms (too few?)

but, not as bad as old sprites,

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

68

Blend shapes: open challenges



- Capturing:
  - from a stream of meshes
  - e.g. : from a RGBD camera (like Microsoft Kinect) to a blend-shape: difficult!
- Compression
  - e.g.: reduce number of keyframes
- Streaming
  - server sends animation to client while it runs
- LOD-ding
  - like for meshes (but more difficult)

69