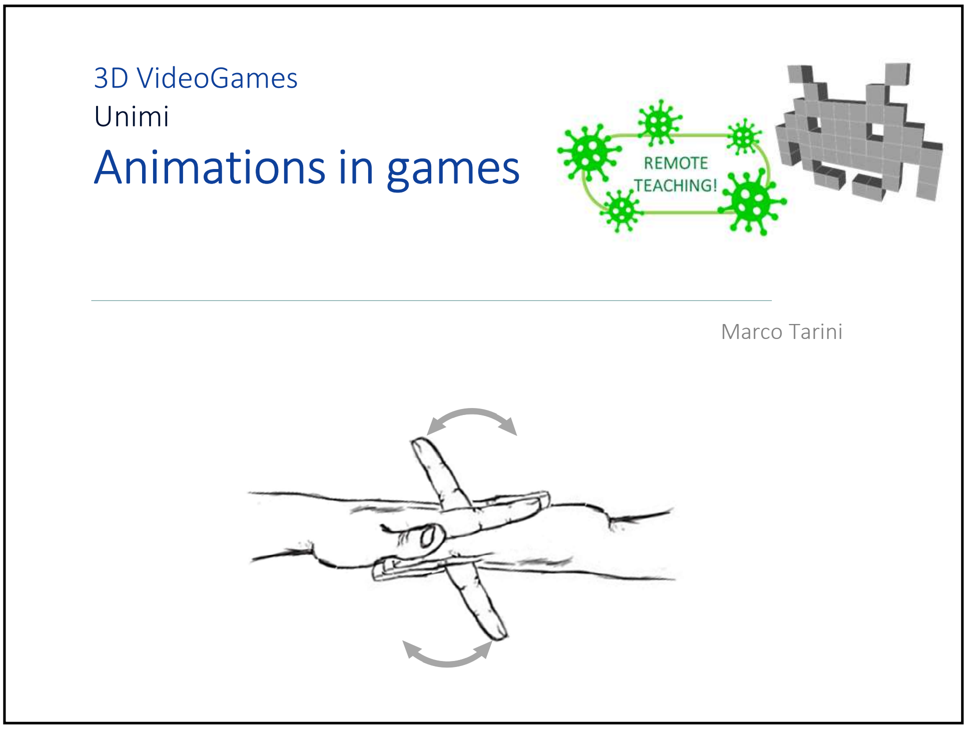
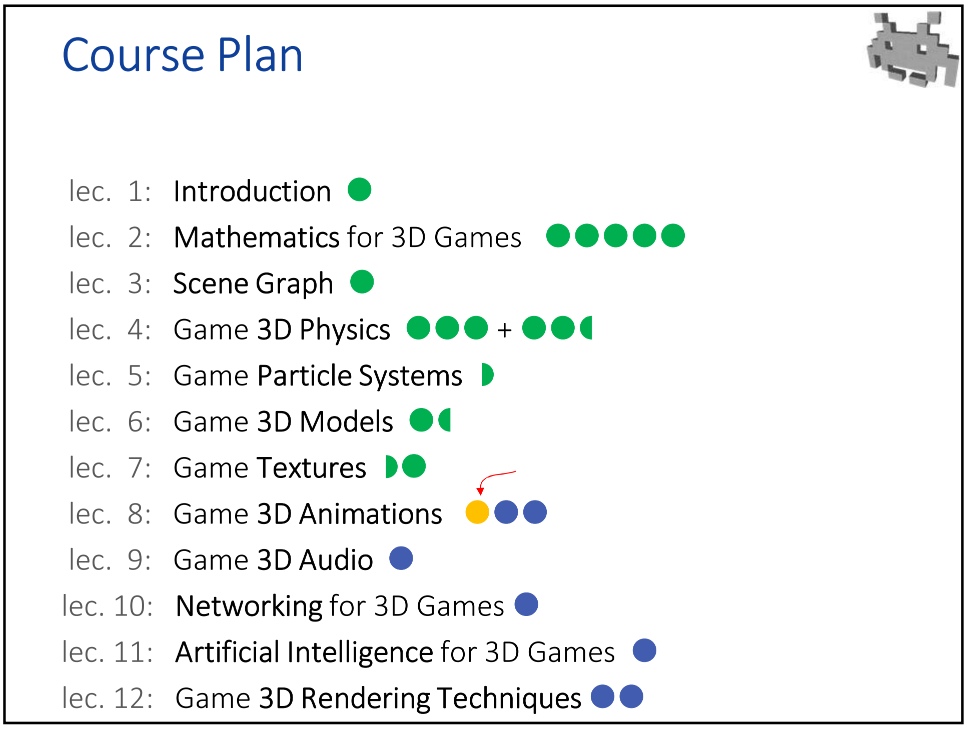


3D VideoGames
Unimi

Animations in games

Marco Tarini

1



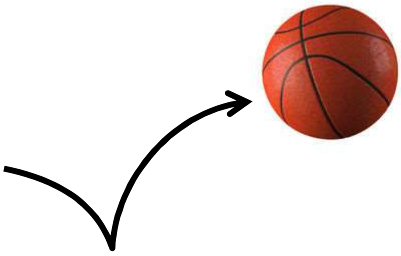
Course Plan

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

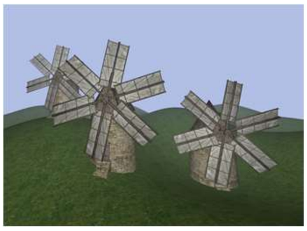


(6 DoF per object)

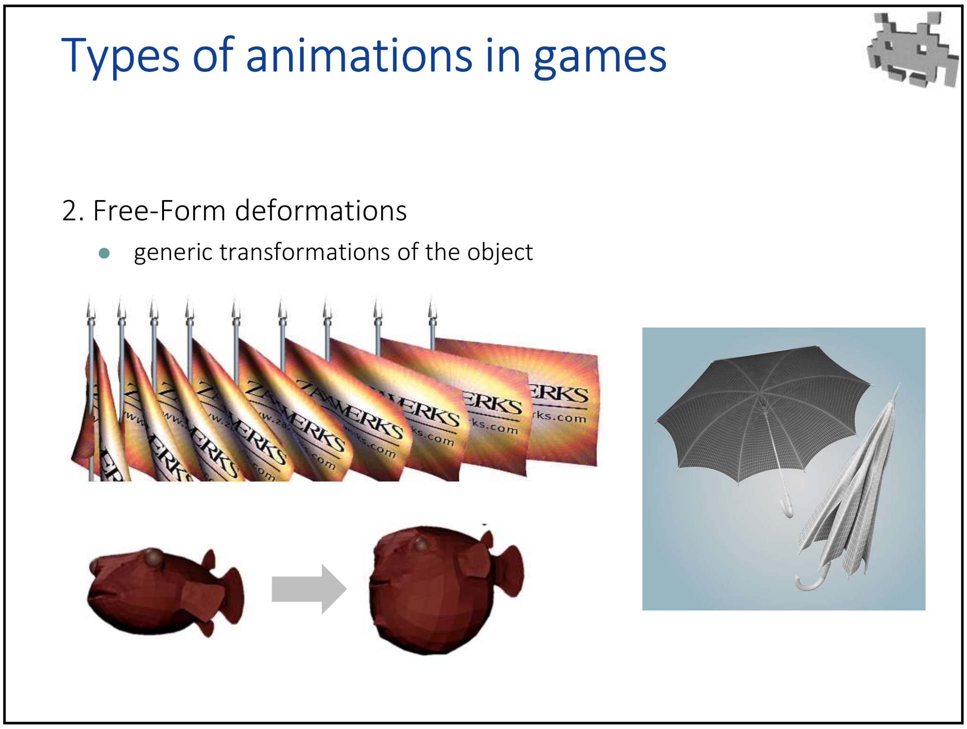
3

Types of animations in games

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



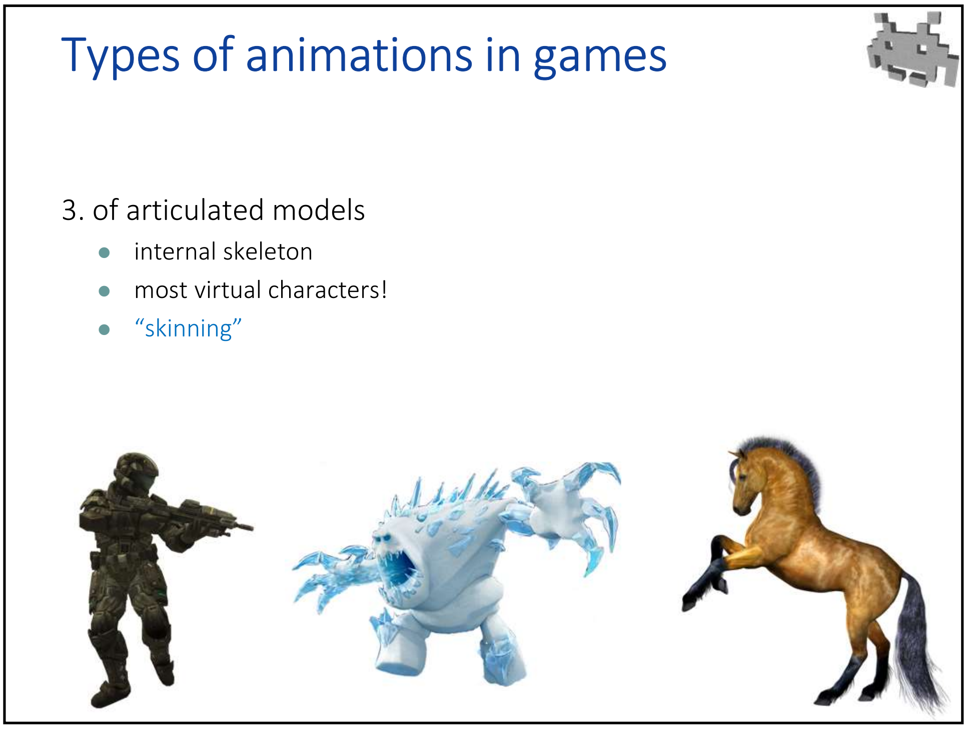
4



Types of animations in games

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

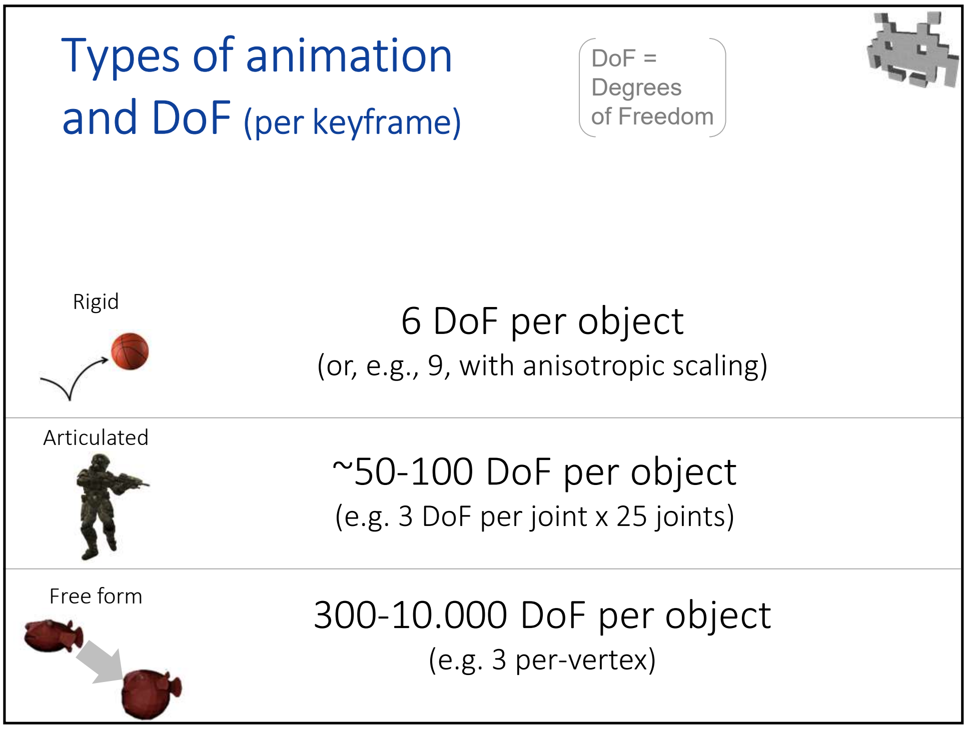
5



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)

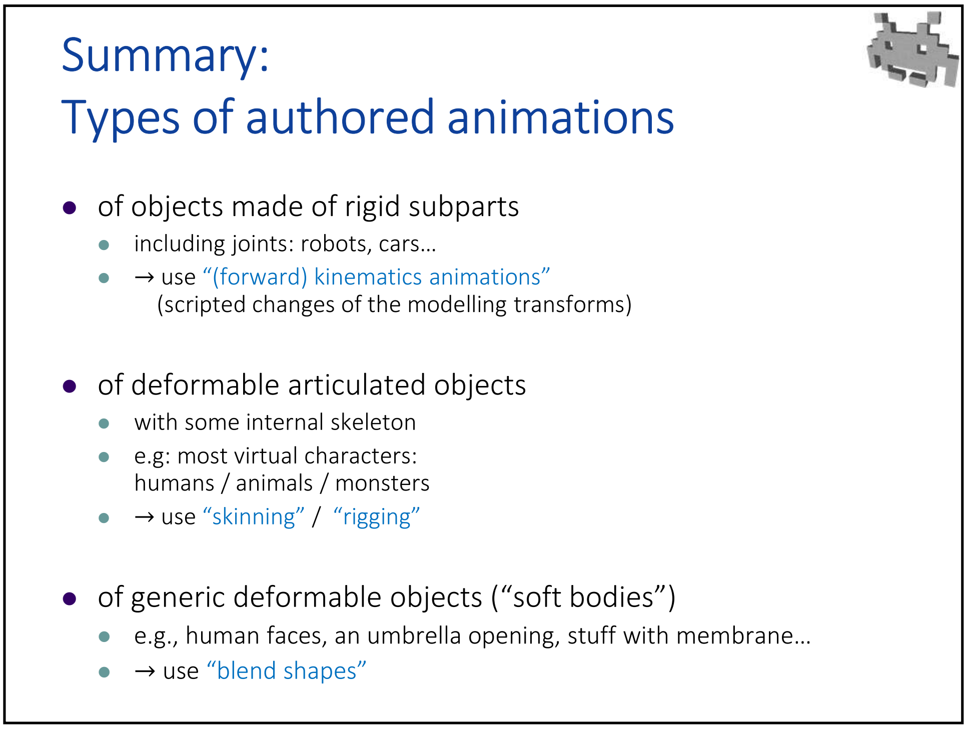
Articulated

~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)

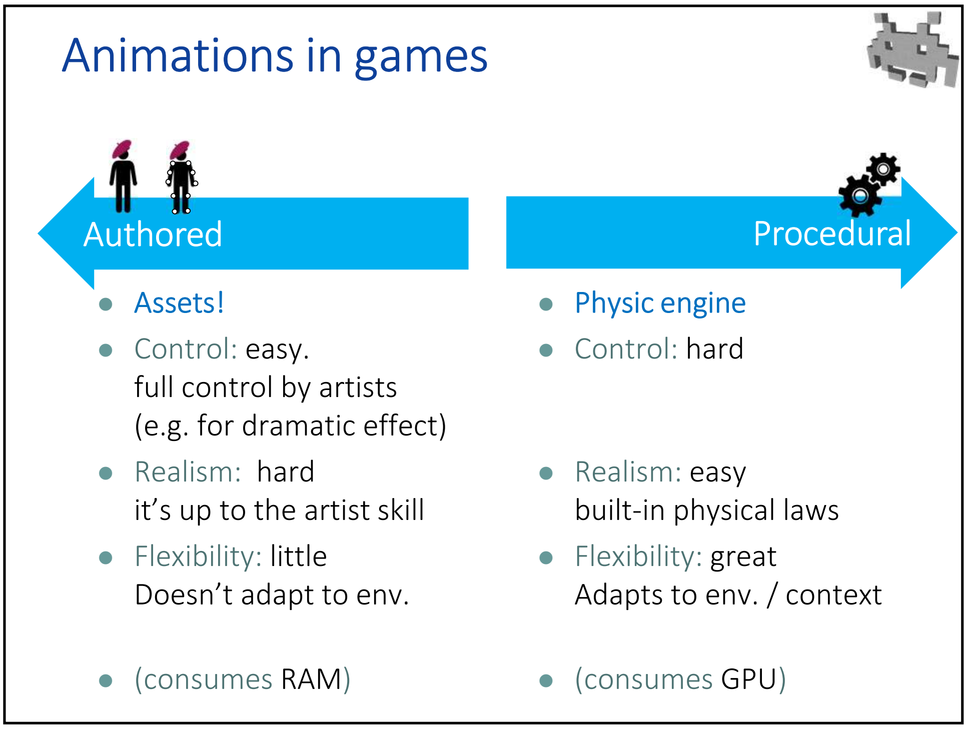
7



Summary: Types of authored animations

- of objects made of rigid subparts
  - including joints: robots, cars…
  - → use "(forward) kinematics animations"
    (scripted changes of the modelling transforms)
- of deformable articulated objects
  - with some internal skeleton
  - e.g: most virtual characters:
    humans / animals / monsters
  - → use "skinning" / "rigging"
- of generic deformable objects ("soft bodies")
  - e.g., human faces, an umbrella opening, stuff with membrane…
  - → use "blend shapes"

8



Animations in games

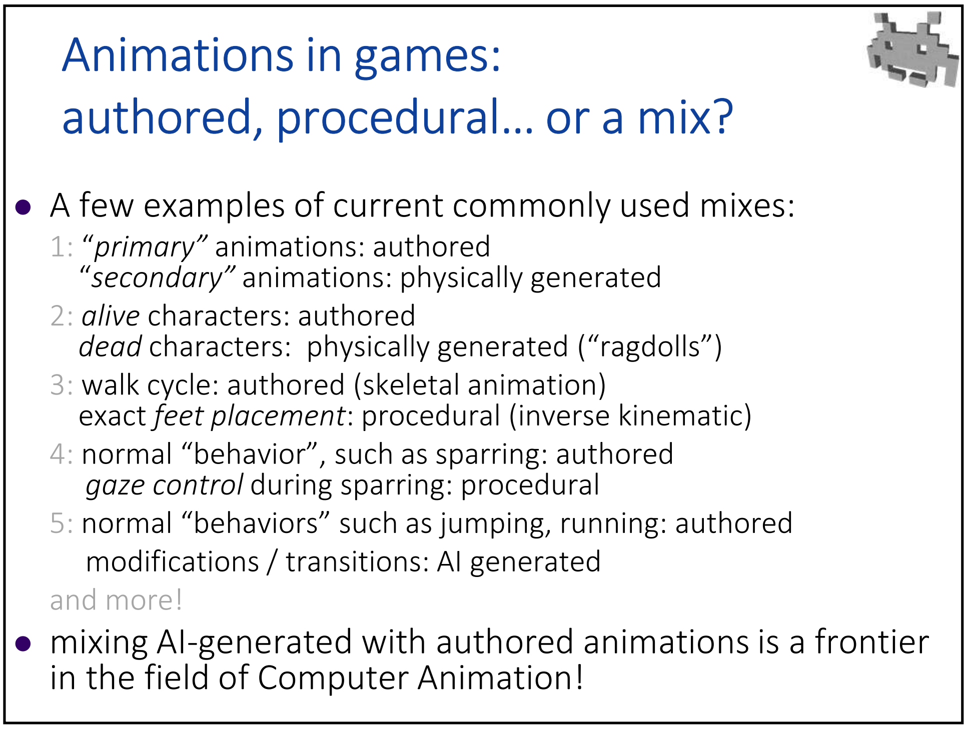
Authored

- Assets!
- Control: easy.
  full control by artists
  (e.g. for dramatic effect)
- Realism: hard
  it's up to the artist skill
- Flexibility: little
  Doesn't adapt to env.
- (consumes RAM)

Procedural

- Physic engine
- Control: hard
- Realism: easy
  built-in physical laws
- Flexibility: great
  Adapts to env. / context
- (consumes GPU)

9



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")

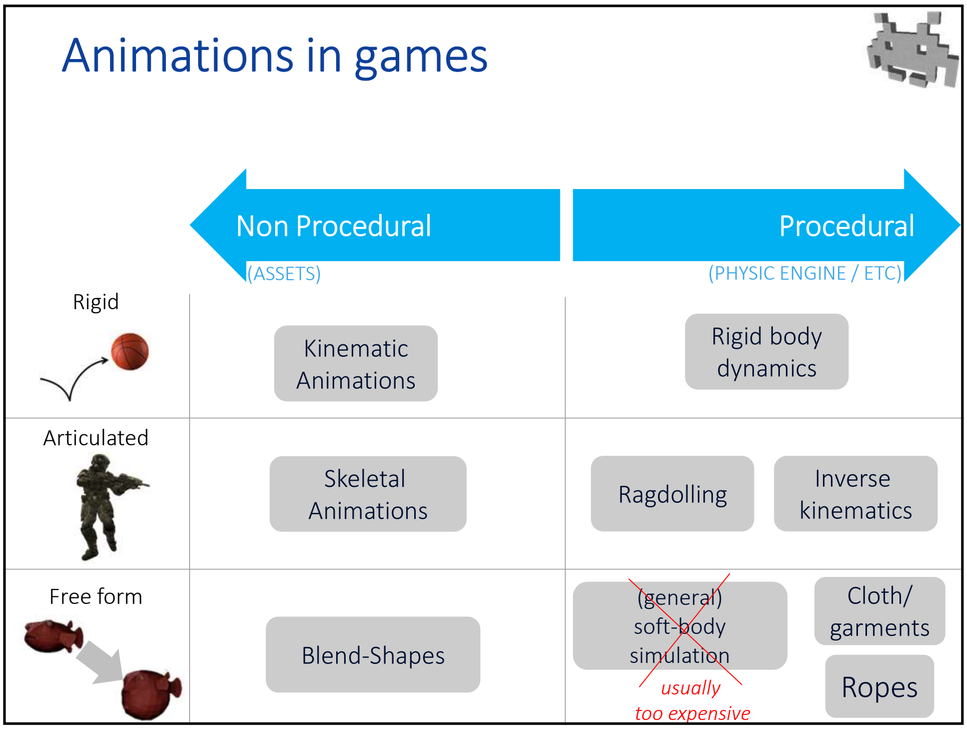
  3: walk cycle: authored (skeletal animation)
  exact *feet placement*: procedural (inverse kinematic)

  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!

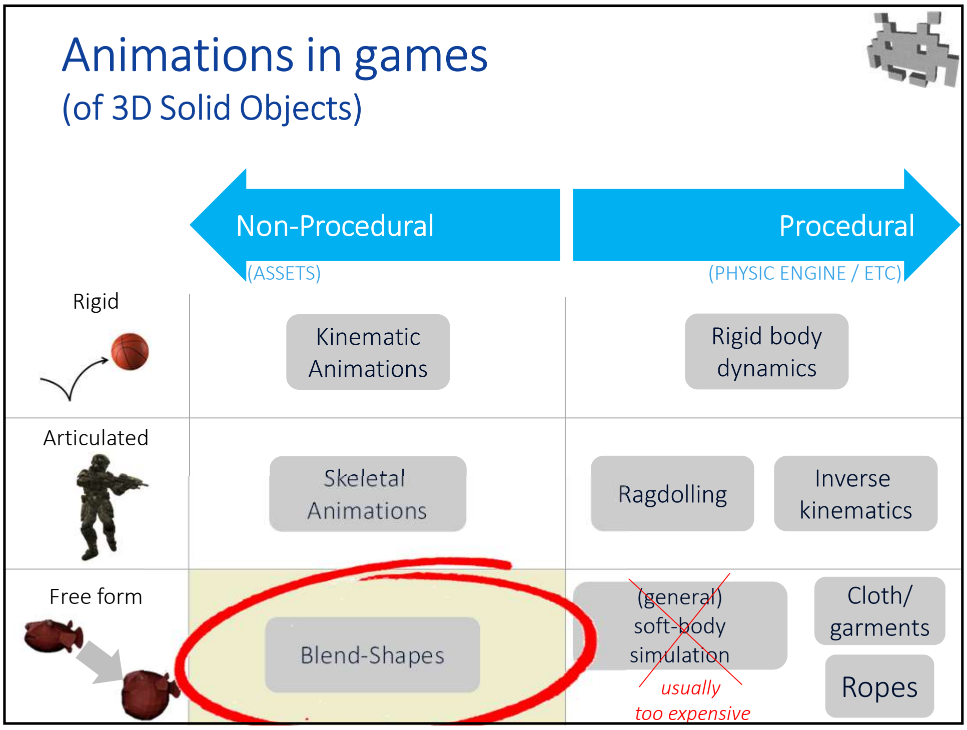
10



Animations in games

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

12



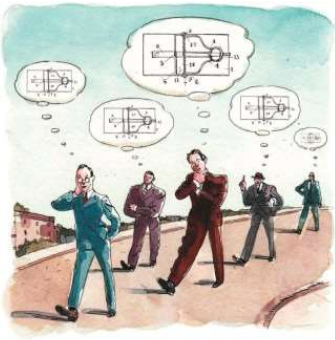
Animations in games
(of 3D Solid Objects)

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

13

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)

14

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?

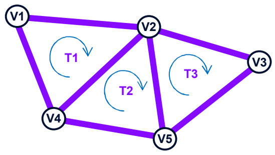
15

Reminder: representation of a mesh

- Indexed mode :
  - Geometry:
    - a 3D position for each vertex
  - Attributes:
    - more data, also stored in each vertex
    - (to be interpolated inside faces)
  - Connectivity:
    - Array of triangles (faces)
    - Each triangle = a triplet of indexes to vertex

16

Mesh (data structure)



connectivity (indexed)

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

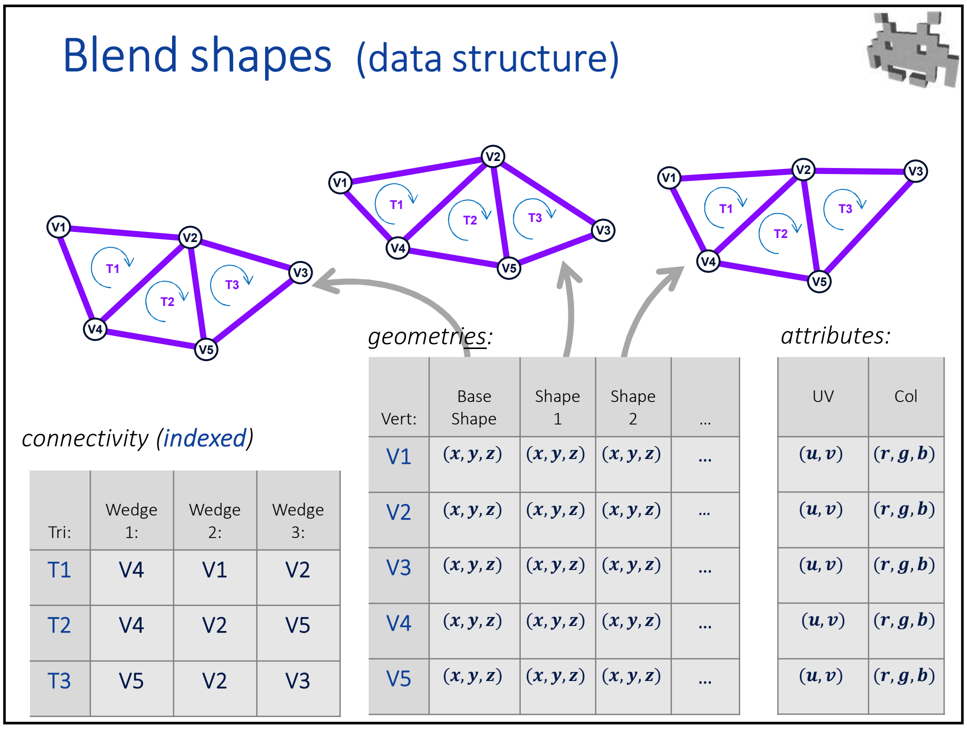
geometry:

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)

17



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

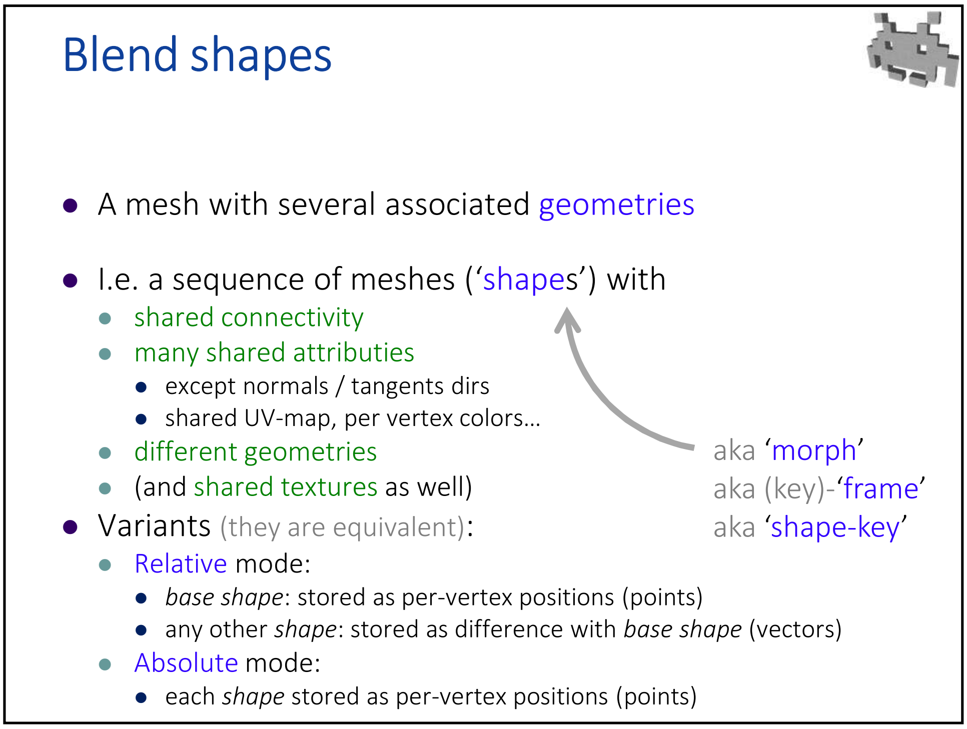
geometries:

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)

18

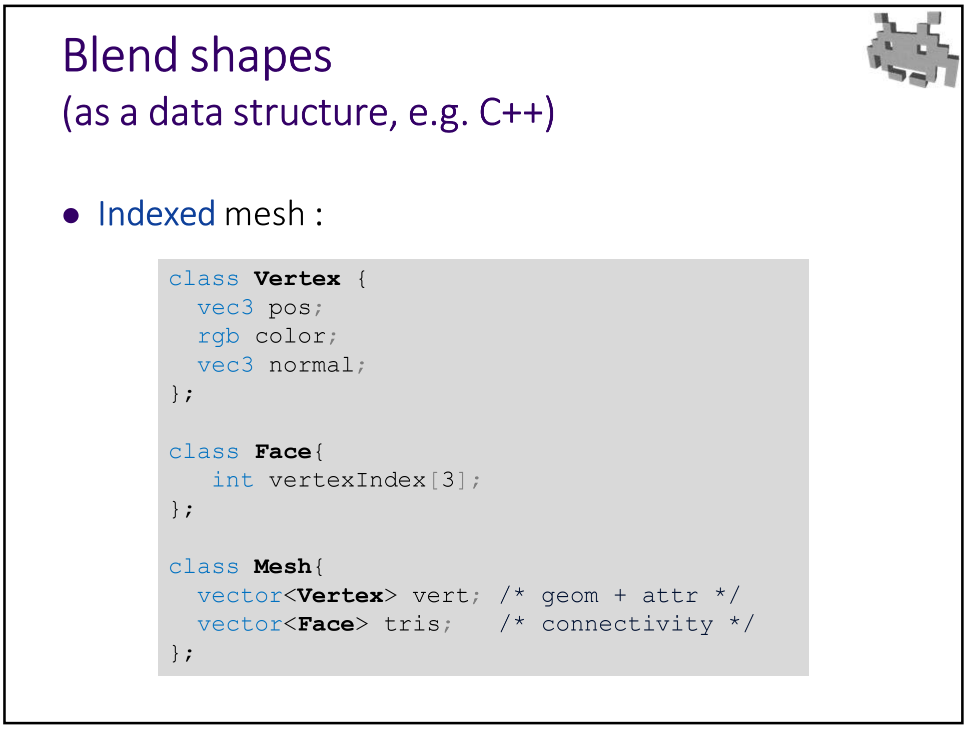


Blend shapes

- 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)
- 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’

19



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

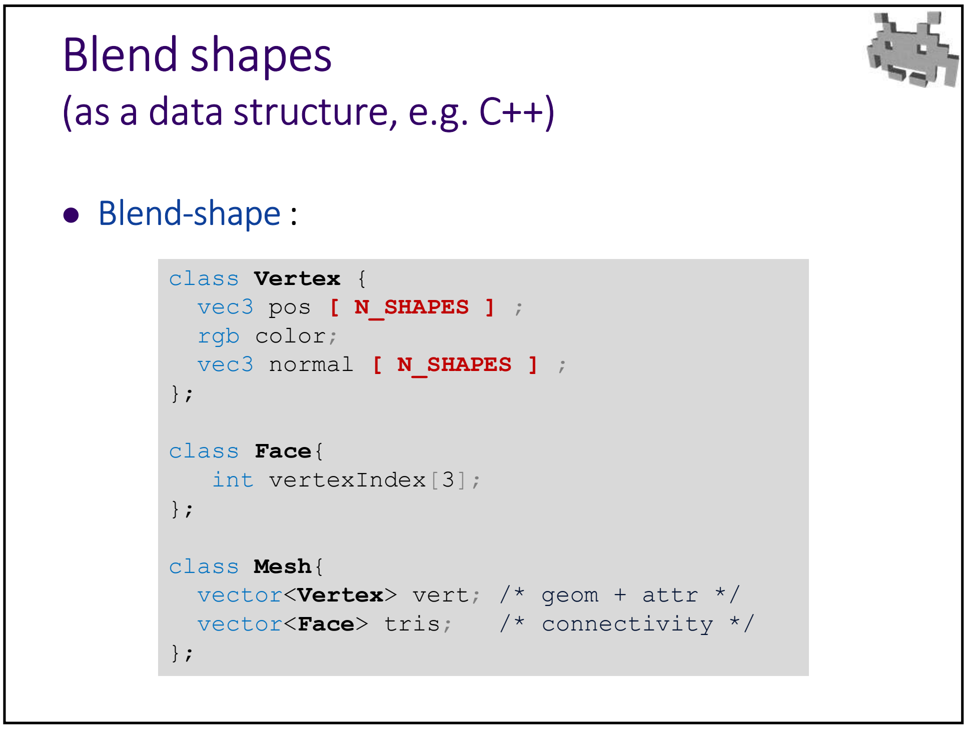
- Indexed mesh :

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

class Face{
   int vertexIndex[3];
};

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

20



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

- Blend-shape :

```cpp
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 */
};
```

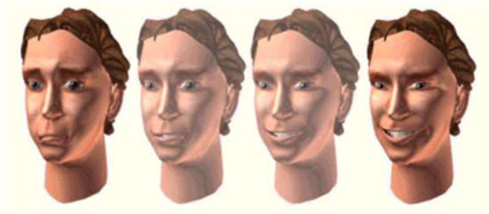
21

Blend-shapes: most common interchange formats

- Simple:
  - .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)
- Complex:
  - .DAE (Collada)
  - .FBX (Autodesk)

22

Uses of Blend-Shapes: facial expressions

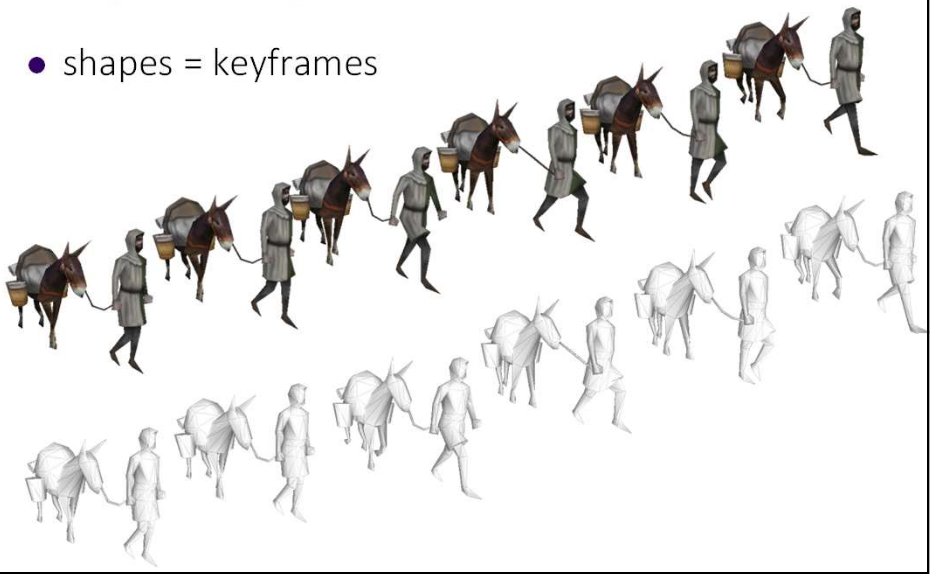


here: shapes = facial expressions
(typical use; that's why they are also called "face morphs"

23

Uses of Blend shapes: temporal sequences

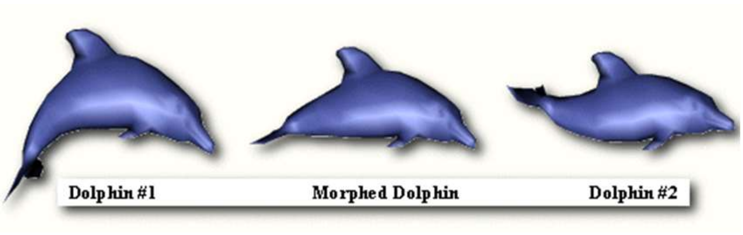
- shapes = keyframes



24

Uses of Blend shapes: temporal sequences

- Temporal sequences
  - shapes = keyframes



25

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? $w_B = \dfrac{t - t_C}{t_B - t_C}$ $w_C = (1 - w_B) = \dfrac{t - t_B}{t_C - t_B}$

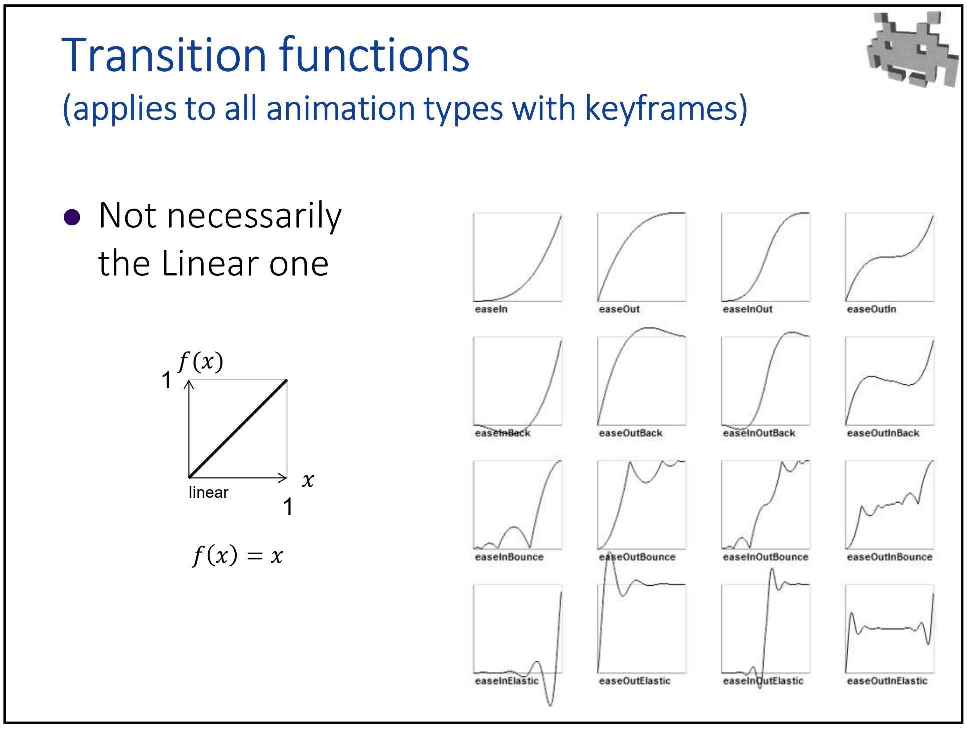
26

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? $w_B = f\left(\dfrac{t - t_C}{t_B - t_C}\right)$ $w_C = (1 - w_B)$

27

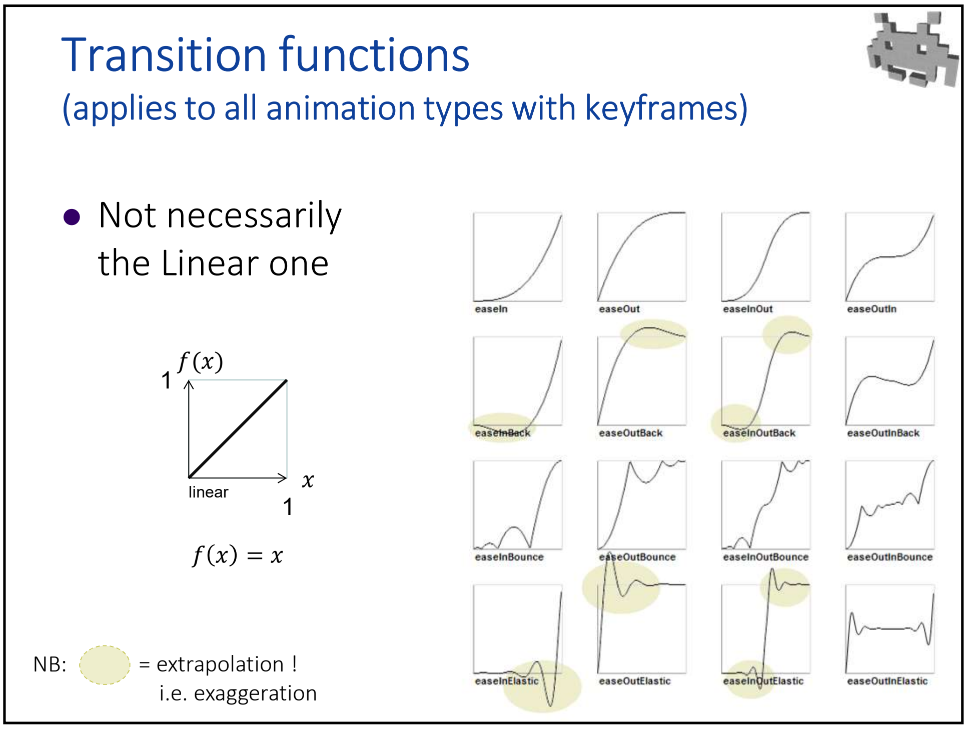


Transition functions
(applies to all animation types with keyframes)

- Not necessarily the Linear one

$f(x) = x$

28



Transition functions
(applies to all animation types with keyframes)

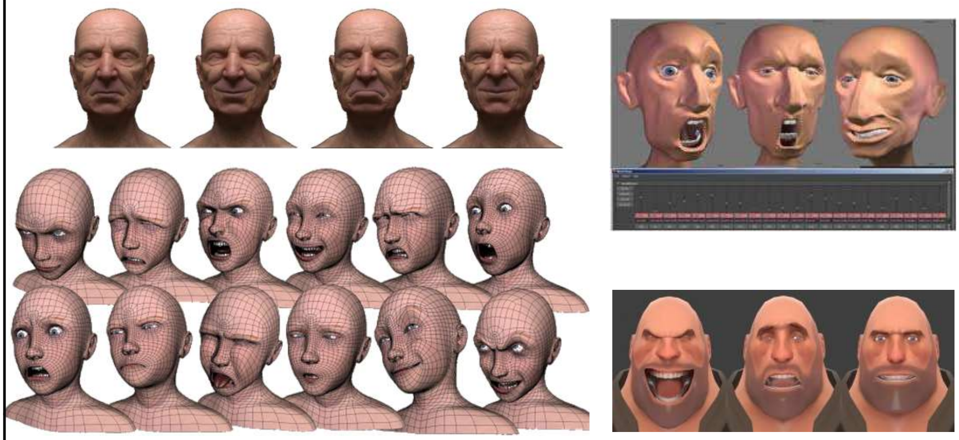
- Not necessarily the Linear one

$f(x) = x$

NB: = extrapolation !
i.e. exaggeration

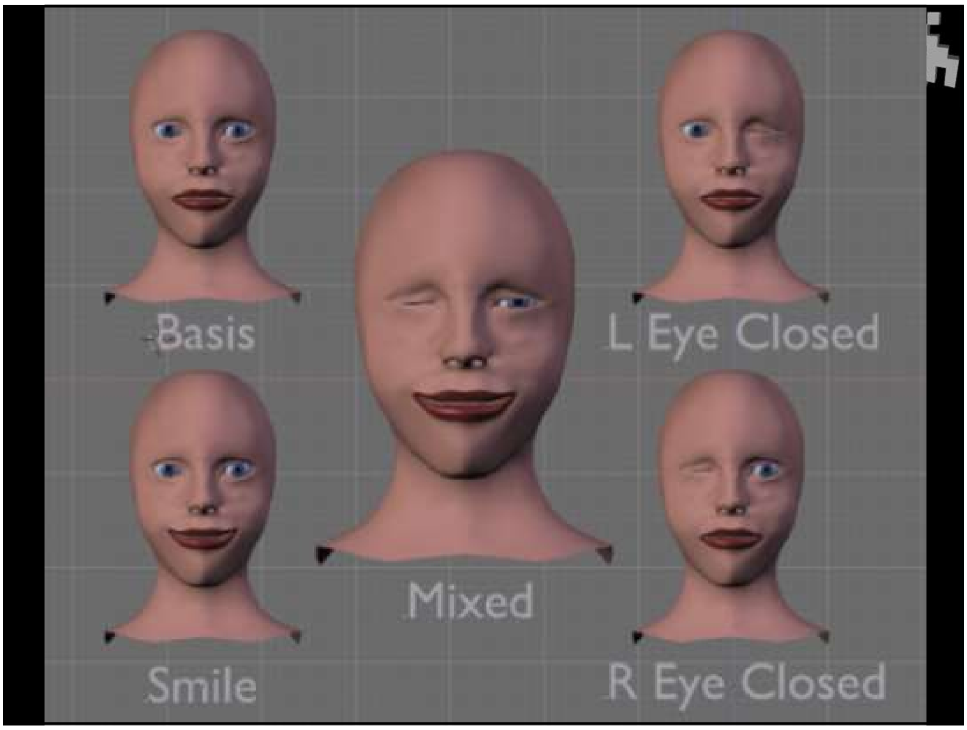
29

Uses of Blend shapes: facial animations

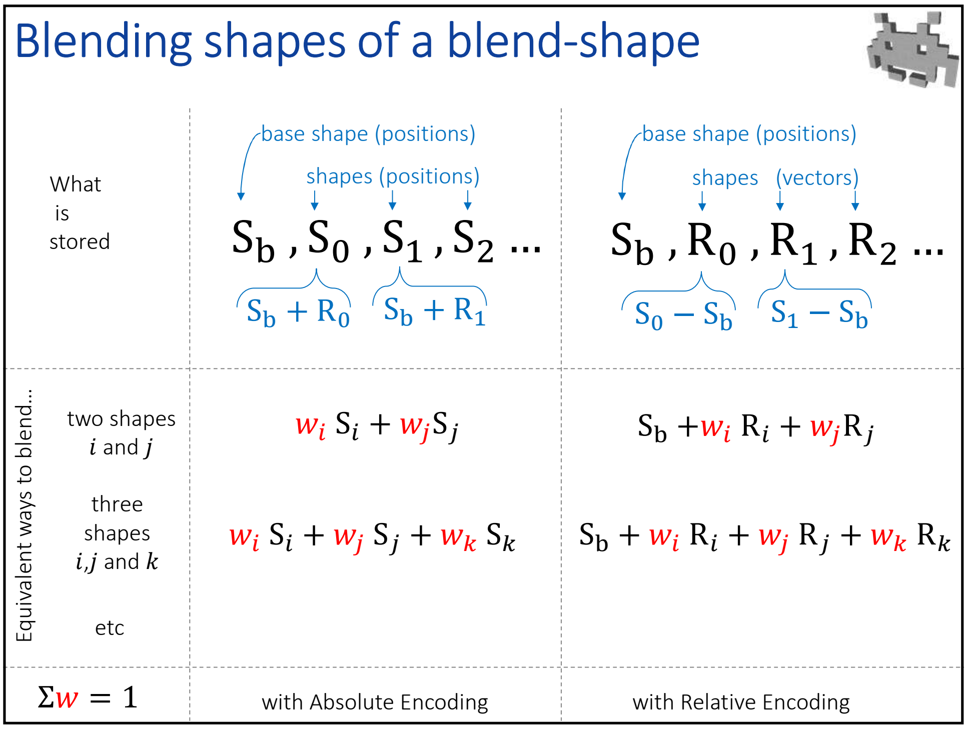


Here, used together with skeletal animations (see next lecture)
(for mandible, neck, eyeballs)

32



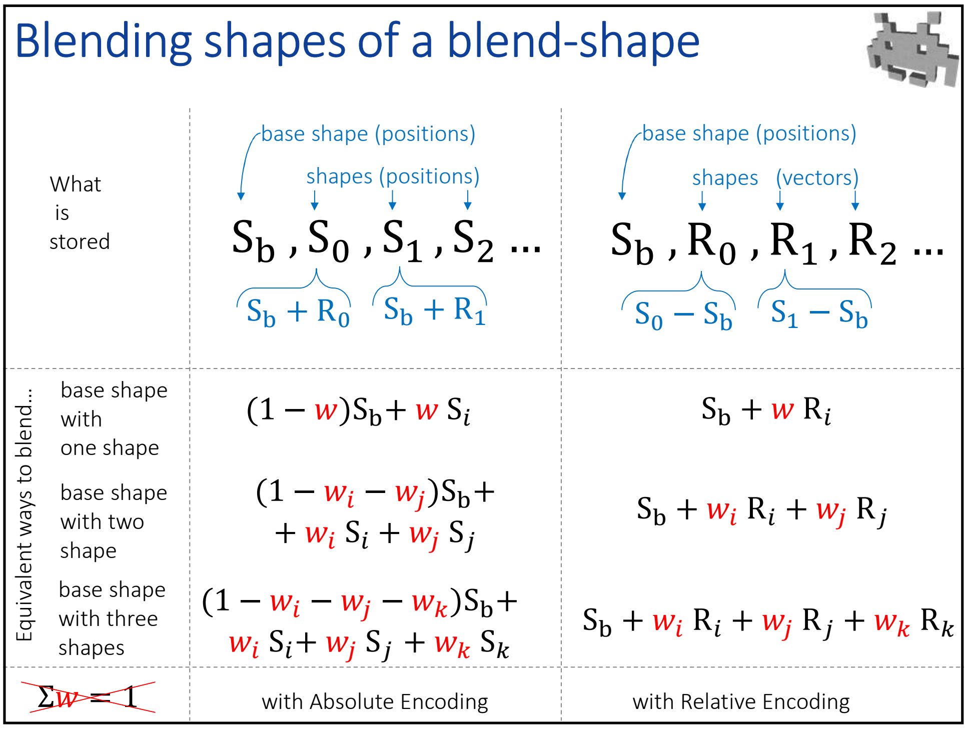
33



Blending shapes of a blend-shape

	What is stored	base shape (positions) shapes (positions) $S_b\, ,\underbrace{S_0}_{S_b + R_0} ,\underbrace{S_1}_{S_b + R_1} , S_2 \dots$	base shape (positions) shapes (vectors) $S_b\, ,\underbrace{R_0}_{S_0 - S_b} ,\underbrace{R_1}_{S_1 - S_b} , R_2 \dots$
Equivalent ways to blend...	two shapes i and j	$w_i\, S_i + w_j S_j$	$S_b + w_i\, R_i + w_j R_j$
	three shapes i,j and k	$w_i\, S_i + w_j\, S_j + w_k\, S_k$	$S_b + w_i\, R_i + w_j\, R_j + w_k\, R_k$
	etc		
	$\Sigma w = 1$	with Absolute Encoding	with Relative Encoding

34



Blending shapes of a blend-shape

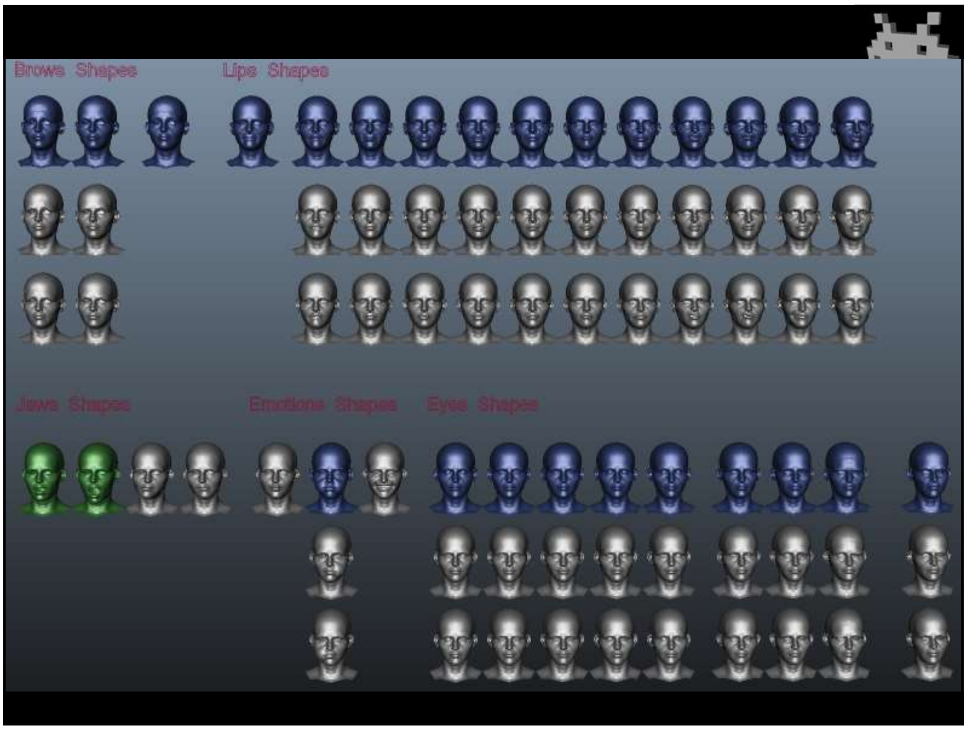
	What is stored	base shape (positions) shapes (positions) $S_b\, ,\underbrace{S_0}_{S_b + R_0} ,\underbrace{S_1}_{S_b + R_1} , S_2 \dots$	base shape (positions) shapes (vectors) $S_b\, ,\underbrace{R_0}_{S_0 - S_b} ,\underbrace{R_1}_{S_1 - S_b} , R_2 \dots$
Equivalent ways to blend...	base shape with one shape	$(1 - w)S_b + w\, S_i$	$S_b + w\, R_i$
	base shape with two shape	$(1 - w_i - w_j)S_b +$ $+ w_i\, S_i + w_j\, S_j$	$S_b + w_i\, R_i + w_j\, R_j$
	base shape with three shapes	$(1 - w_i - w_j - w_k)S_b +$ $w_i\, S_i + w_j\, S_j + w_k\, S_k$	$S_b + w_i\, R_i + w_j\, R_j + w_k\, R_k$
	$\cancel{\Sigma w = 1}$	with Absolute Encoding	with Relative Encoding

35

Blending shapes of a blend-shape: notes

- The two ways to store a bland-shape are equivalent
  - The can achive the same set of morphed shapes
  - Note: when Σw_i=1 the formula for absolute is simpler
  - Note: when Σw_i>1 it becomes an extrapolation (beware)
- The absolute way is more natural when shapes are designed to be used as *alternatives* (and Σw_i=1)
  - Examples: keyframes of an animation
- The relative way is more natural when shapes are designed to be *superimposed* with various degrees of strength. E.g.:
  - shape$_0$ = close left eye
  - shape$_1$ = smile
  - shape$_0$ + shape$_1$ = wink
  - shape$_0$ = fat
  - shape$_1$ = long chin
  - 0.4 shape$_0$ + 0.9 shape$_1$ = a bit fat & quite long chin

36



40

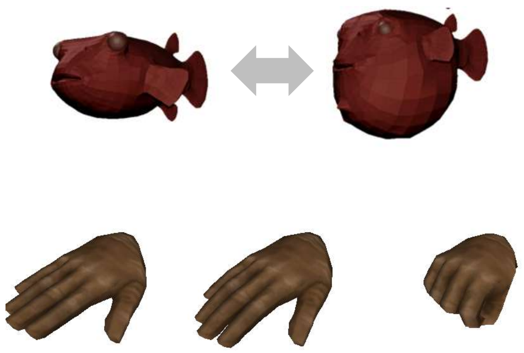
Using facial animations as Blend shapes

- 3D Modeller authors:
  produces the blend-shapes (aka: the “facial rig”)
- Animator (of expressions) picks:
  weights
  - eg.: with sliders
  - assisted / substituted by automatisms
    - e.g., lip sync
    - e.g., dynamically determined expressions
- Keyshape Blending: by rendering engine

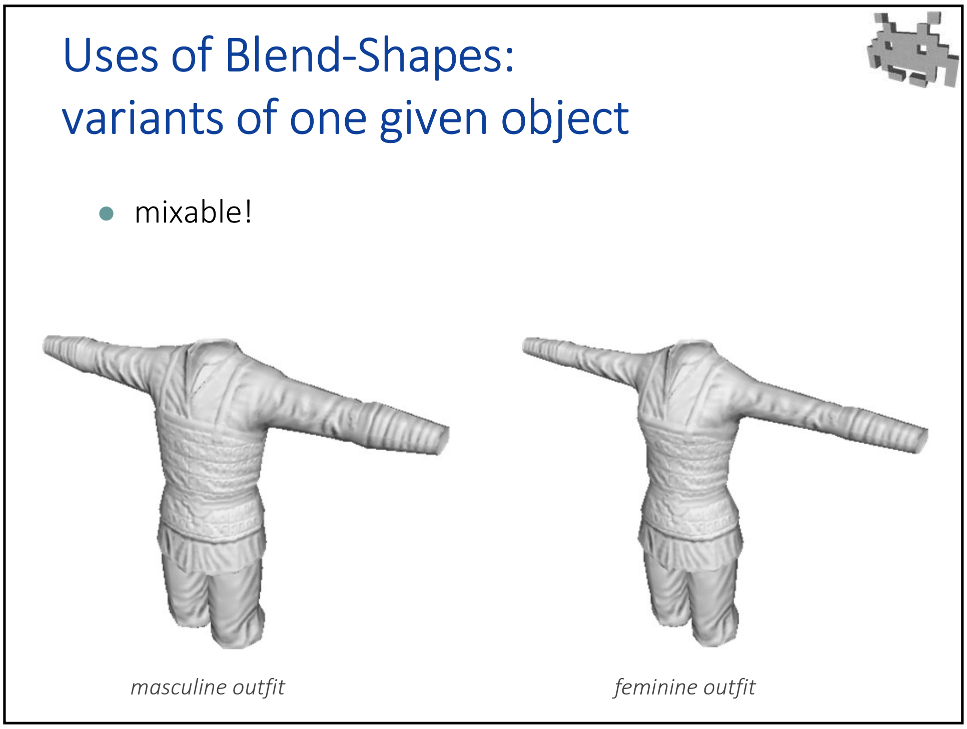
41

Uses of Blend-Shapes: generic deformations

- Baked poses



42



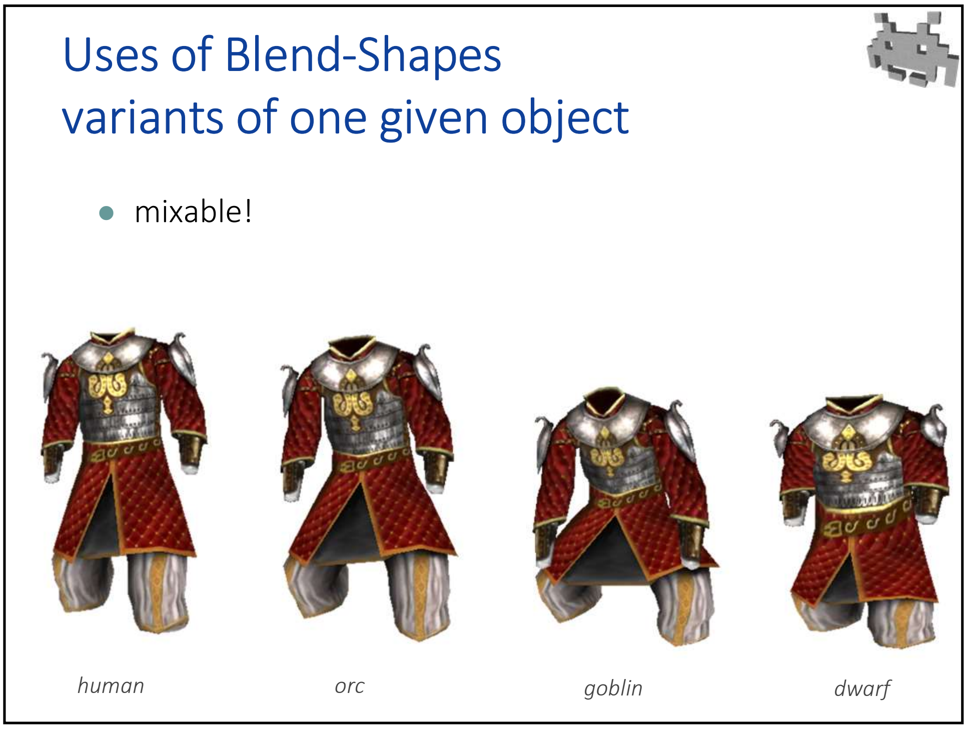
Uses of Blend-Shapes: variants of one given object

- mixable!

masculine outfit

feminine outfit

43



Uses of Blend-Shapes variants of one given object

- mixable!

human

orc

goblin

dwarf

44

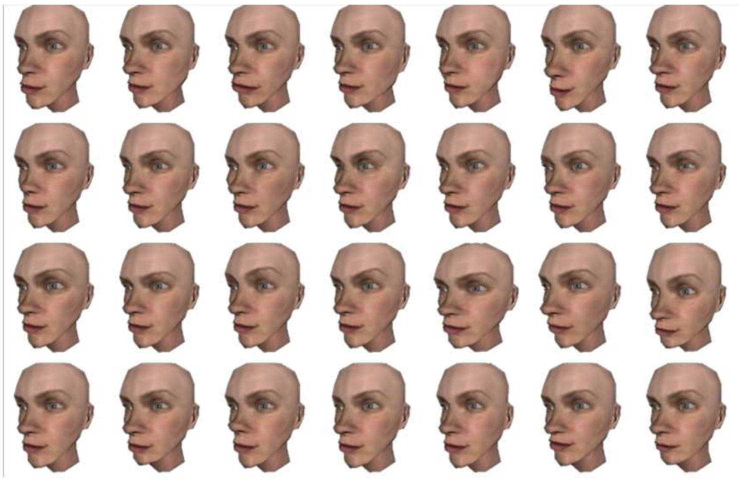
Uses of Blend-Shapes

- 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

45

Uses of Blend shapes

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



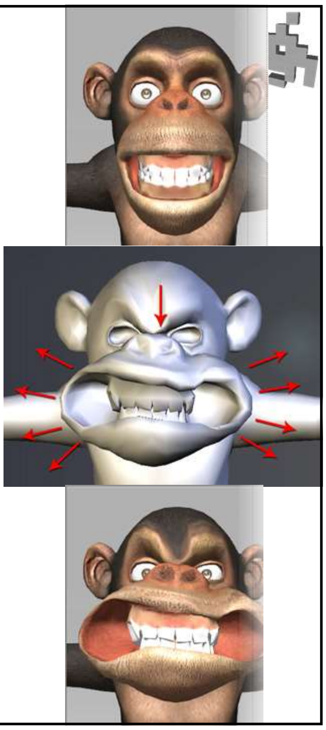
46

All morph-shape share… (so, a blend-shape *cannot* change)…

- The mesh connectivity
  - Eg. no change mesh res, remeshing
- Therfore, 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?

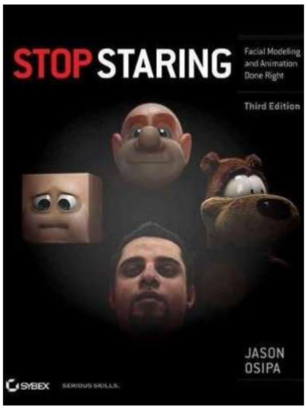
Blend shapes: authoring

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.



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



49

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”

50

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,

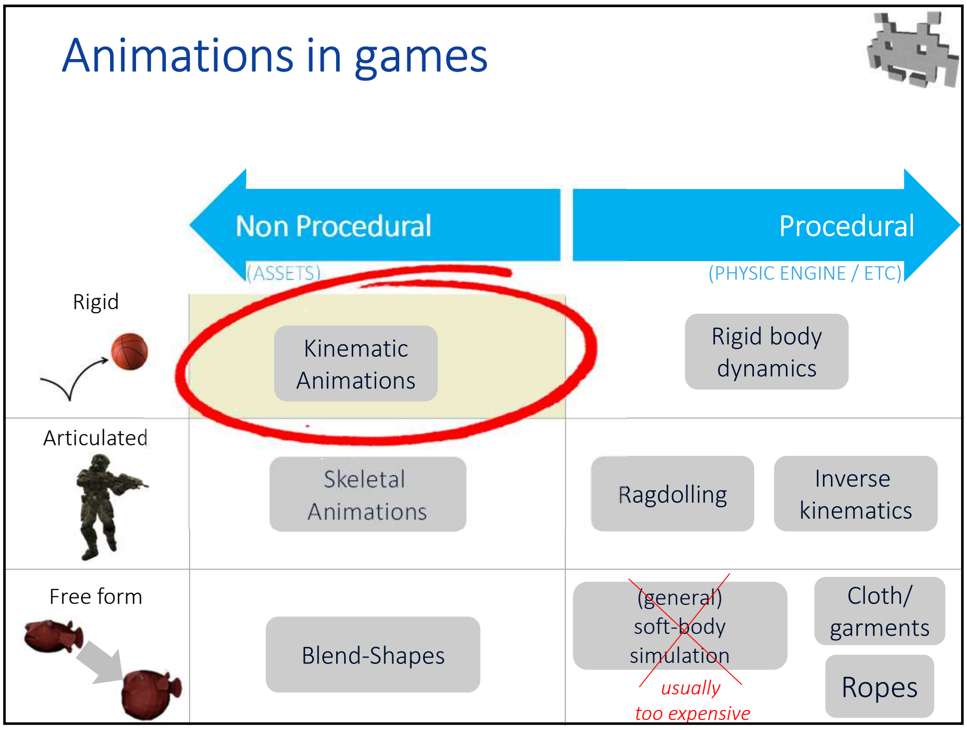
because
(1) shared of connectivity, textures, attributes
(2) keyframes / inbetweens!

51

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)

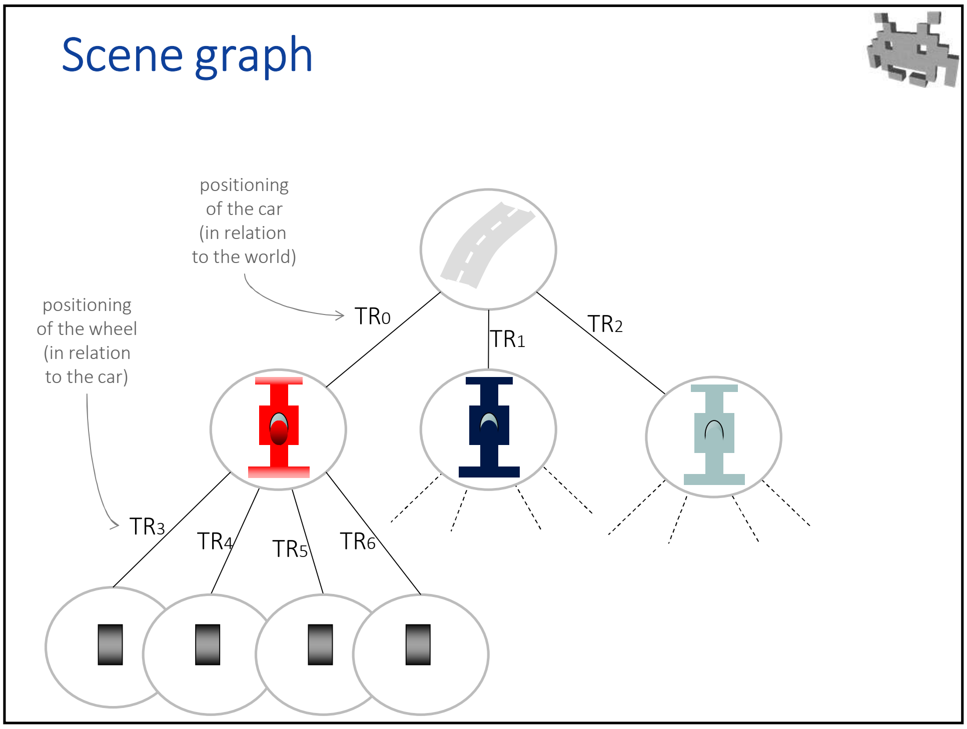
52



Animations in games

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

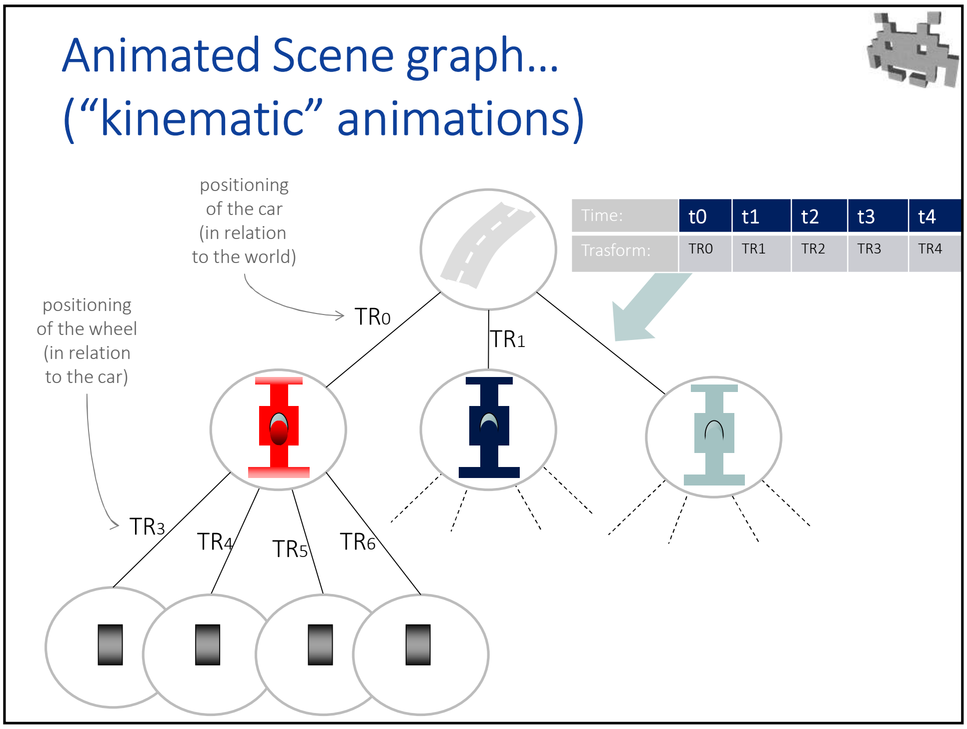
53



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

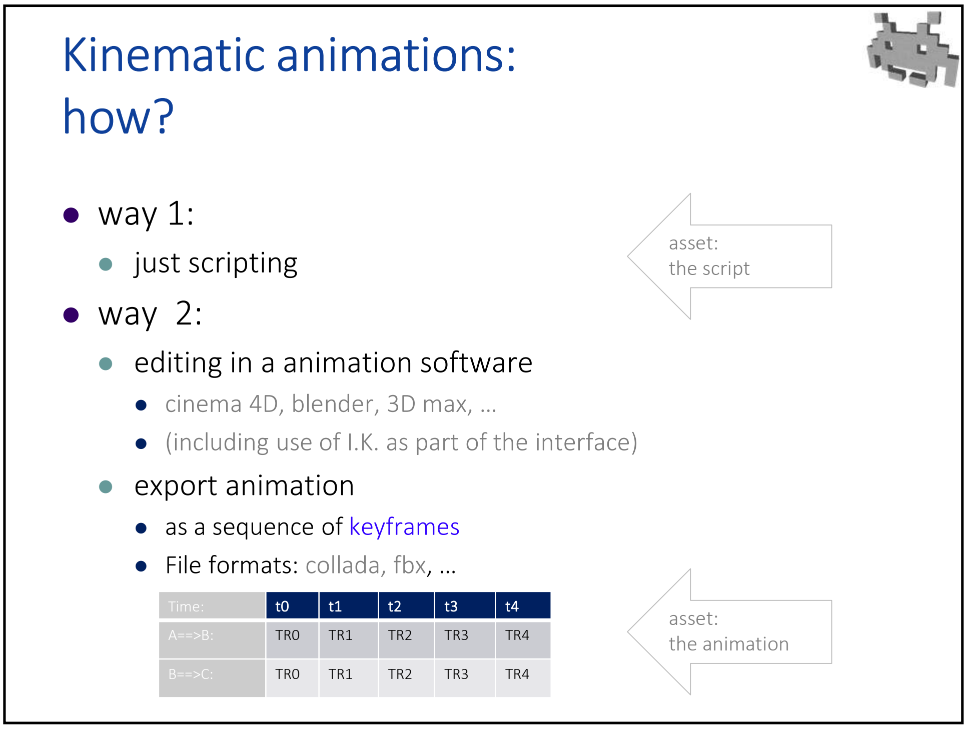
54



Animated Scene graph… (“kinematic” animations)

Time:	t0	t1	t2	t3	t4
Trasform:	TR0	TR1	TR2	TR3	TR4

55



Kinematic animations: how?

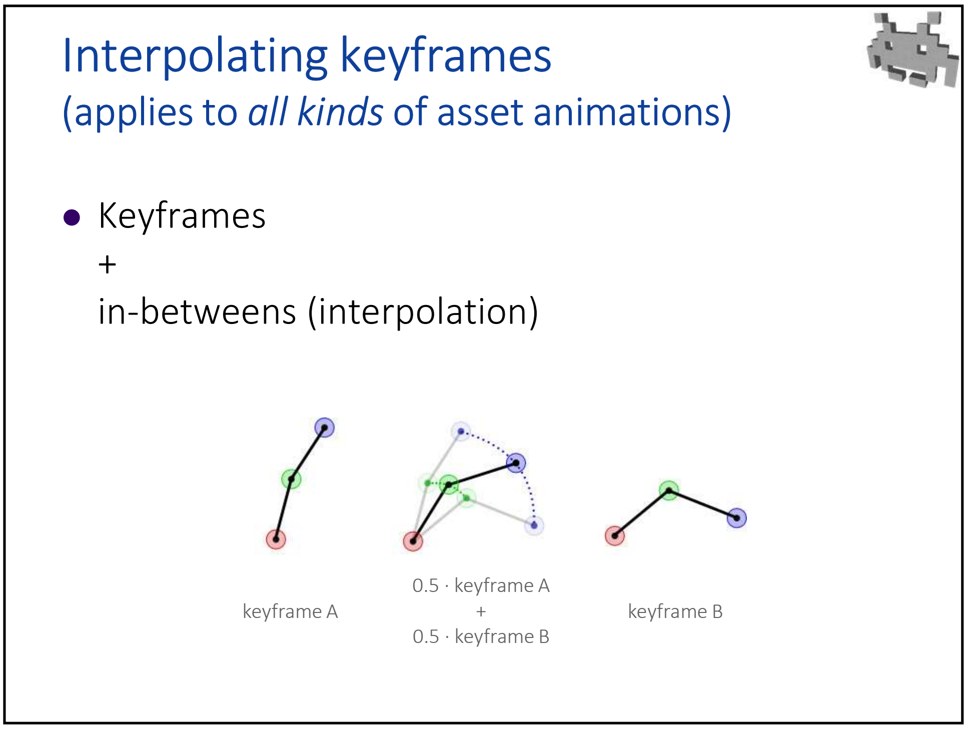
- way 1:
  - just scripting
- way 2:
  - editing in a animation software
    - cinema 4D, blender, 3D max, …
    - (including use of I.K. as part of the interface)
  - export animation
    - as a sequence of keyframes
    - File formats: collada, fbx, …

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

asset:
the script

asset:
the animation

56

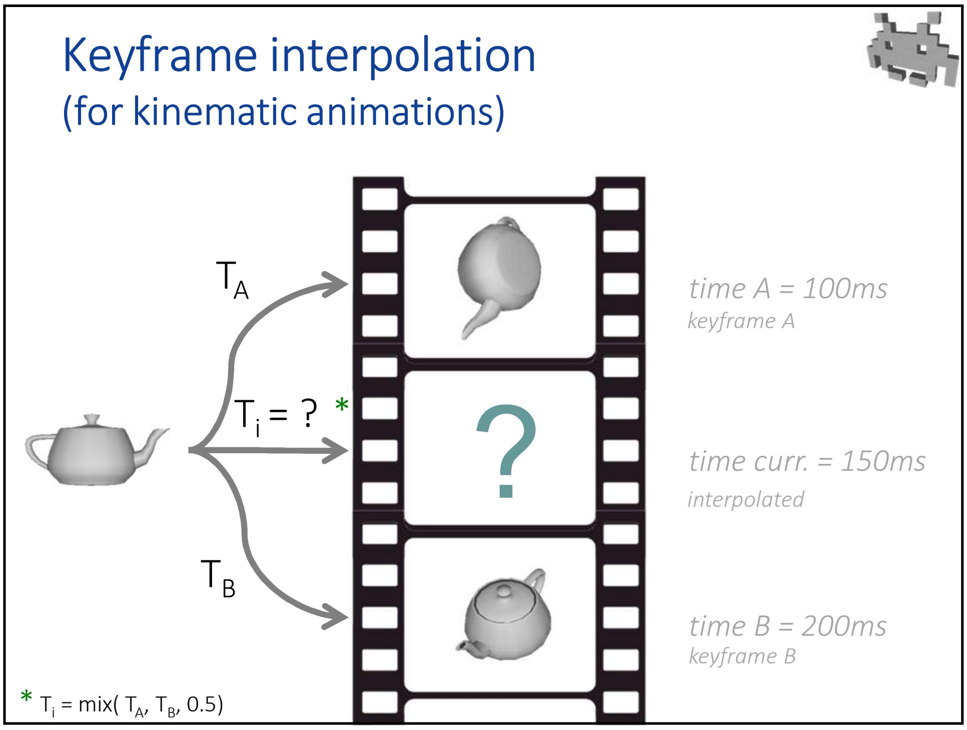


Interpolating keyframes

(applies to *all kinds* of asset animations)

- Keyframes
  +
  in-betweens (interpolation)

57

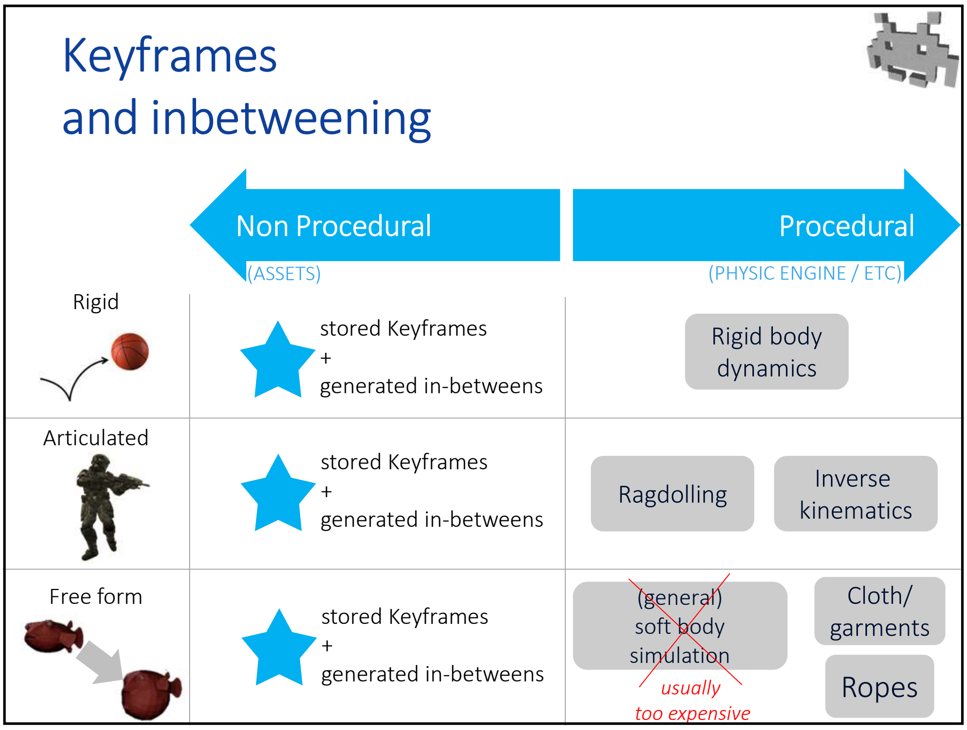


Keyframe interpolation

(for kinematic animations)

* T_i = mix(T_A, T_B, 0.5)

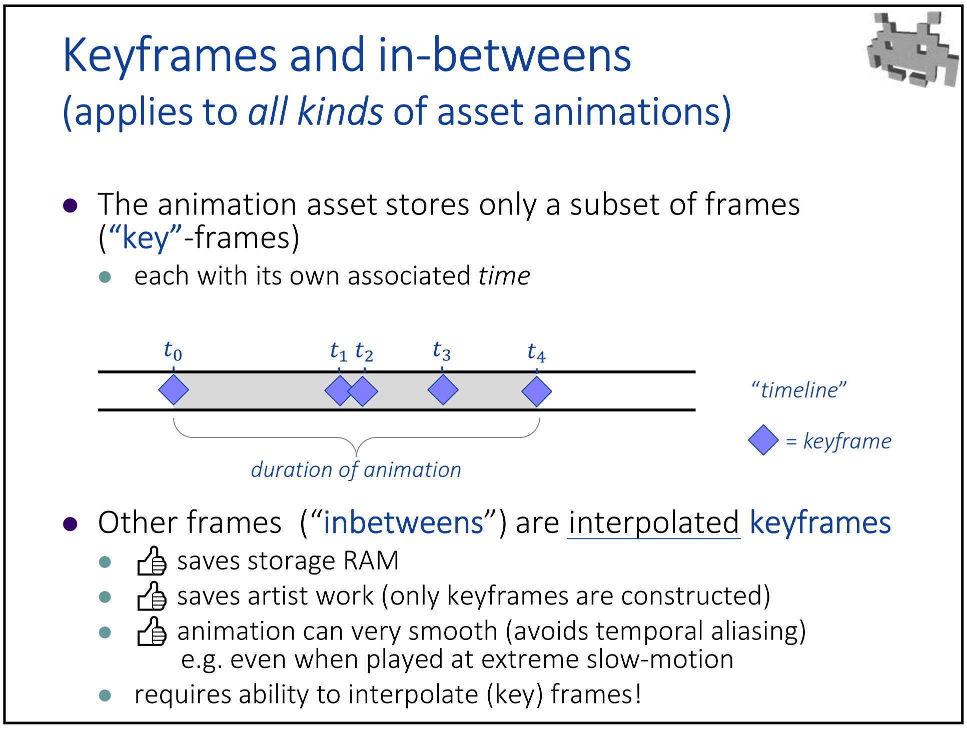
58



Keyframes and inbetweening

	Non Procedural (ASSETS)	Procedural (PHYSIC ENGINE / ETC)
Rigid	★ stored Keyframes + generated in-betweens	Rigid body dynamics
Articulated	★ stored Keyframes + generated in-betweens	Ragdolling Inverse kinematics
Free form	★ stored Keyframes + generated in-betweens	~~(general) soft body simulation~~ *usually too expensive* Cloth/ garments Ropes

59

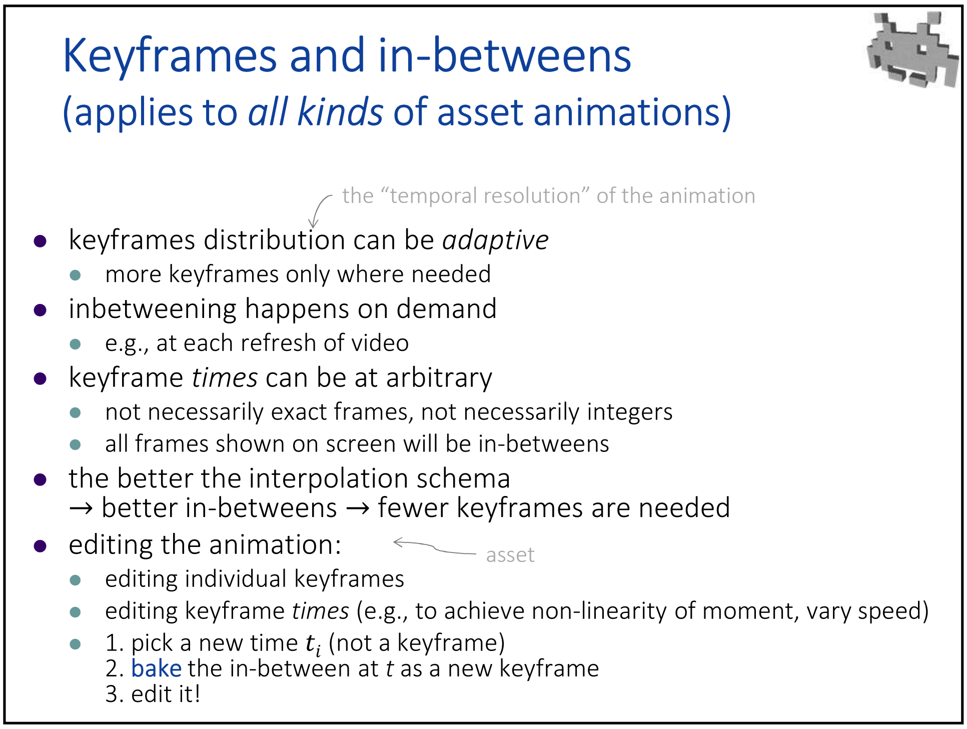


Keyframes and in-betweens (applies to *all kinds* of asset animations)

- 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!

60

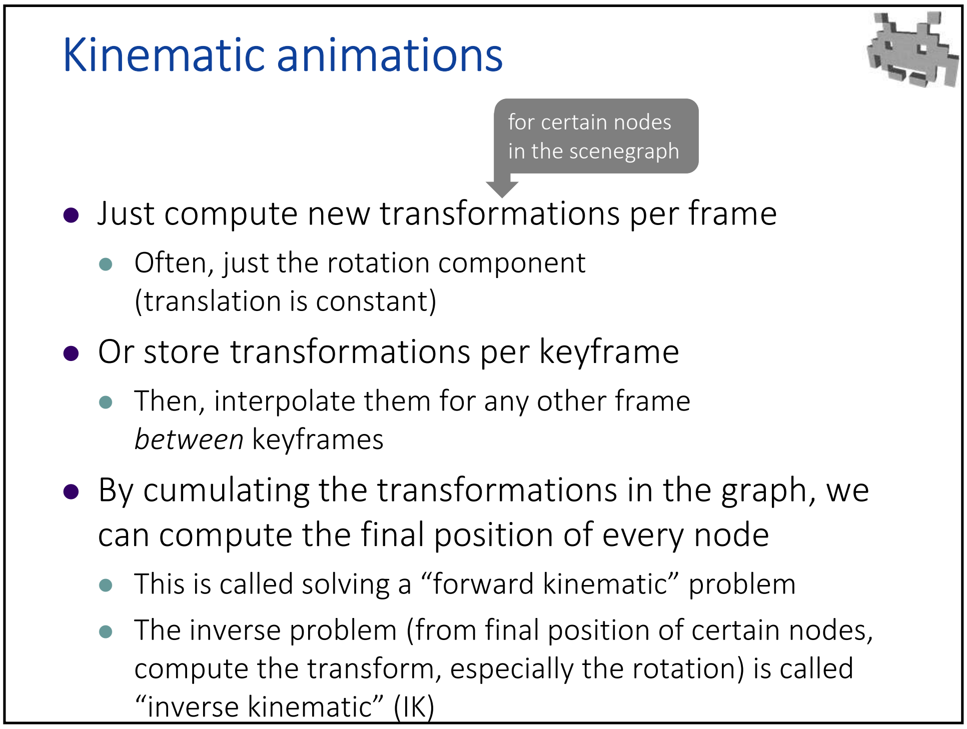


Keyframes and in-betweens
(applies to *all kinds* of asset animations)

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 of video
- keyframe *times* can be at arbitrary
  - not necessarily exact frames, not necessarily integers
  - all frames shown on screen will be in-betweens
- the better the interpolation schema
  → better in-betweens → fewer keyframes are needed
- editing the animation: asset
  - 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!

61



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)

62