Computer Graphics

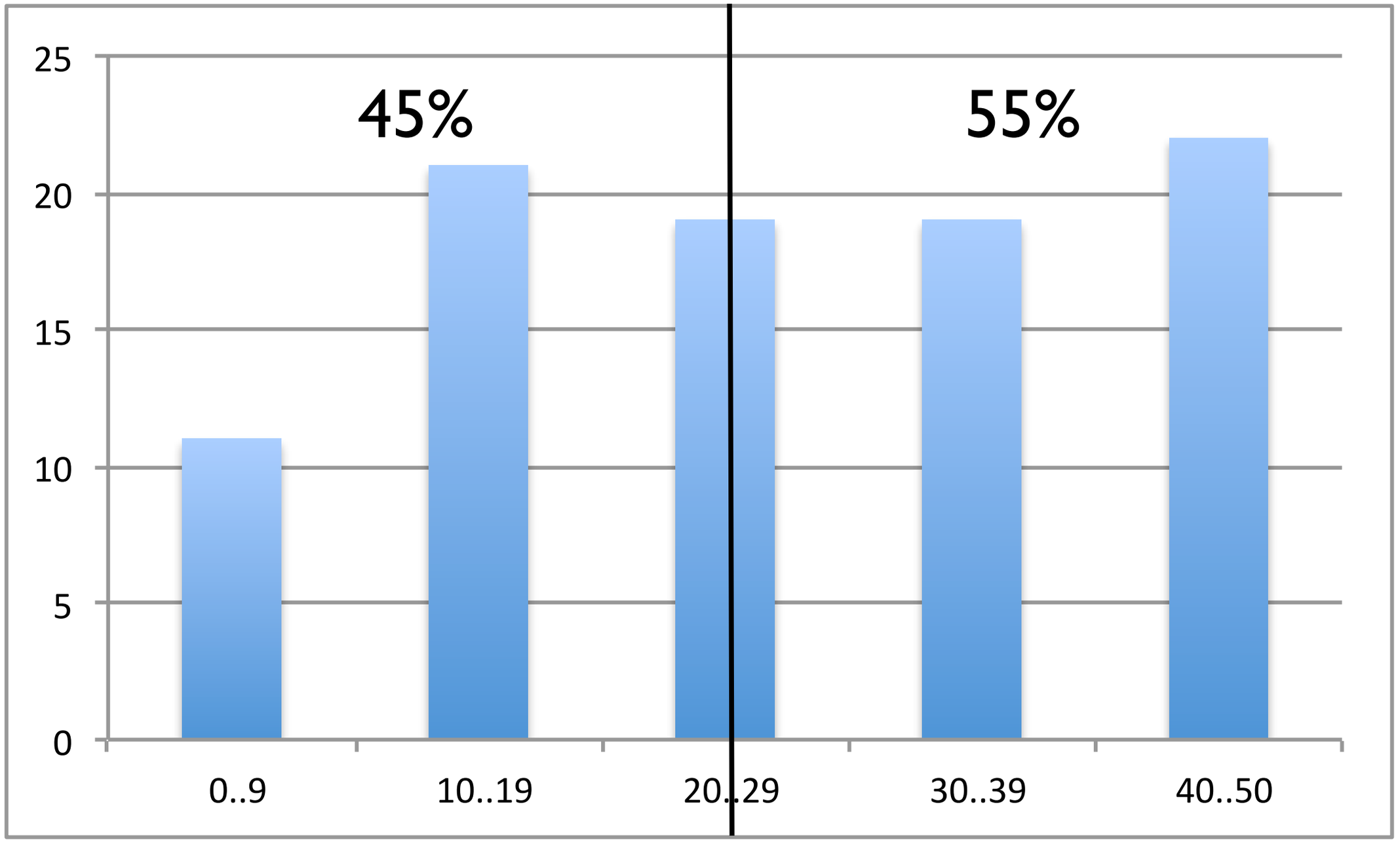
-Basics of Animation-

Oliver Bimber

Course Schedule

Type	Date	Time	Room	Topic	Comment
C1	01.03.2016	13:45-15:15	HS 18	Introduction and Course Overview	Conference
C2	15.03.2016	13:45-15:15	HS 18	Transformations and Projections	Easter Break
C3	05.04.2016	13:45-15:15	HS 18	Raster Algorithms and Depth Handling	
C4	12.04.2016	13:45-15:15	HS 18	Local Shading and Illumination	
C5	19.04.2016	13:45-15:15	HS 18	Texture Mapping Basics	
C6	26.4.2016	13:45-15:15	HS 18	Advanced Texture Mapping & Graphics Pipelines	
C7	03.05.2016	13:45-15:15	HS 18	Intermediate Exam	
C8	09.05.2016	17:15-18:45	HS 18	Global Illumination I: Raytracing	
C9	10.05.2016	13:45-15:15	HS 18	Global Illumination II: Radiosity	Conference / Holiday
C10	31.05.2016	13:45-15:15	HS 18	Volume Rendering	
C11	07.06.2016	13:45-15:15	HS 18	Scientific Data Visualization	
C12	14.06.2016	13:45-15:15	HS 18	Curves and Surfaces	
C13	21.06.2016	13:45-15:15	HS 18	Basics of Animation	
C14	28.06.2016	13:45-15:15	HS 18	Final Exam	
C15	04.10.2016	13:45-15:15	TBA	Retry Exam	

A Friendly Reminder



Visual Computing Club

- Each year, we will invite a very small number of you to join our Visual Computing Club to promote excellent students with interest in computer graphics / computer vision / visualization

- Requirements: straight -sehr gut- in exam and exercise

- Benefits:

  - free access to our labs and equipment

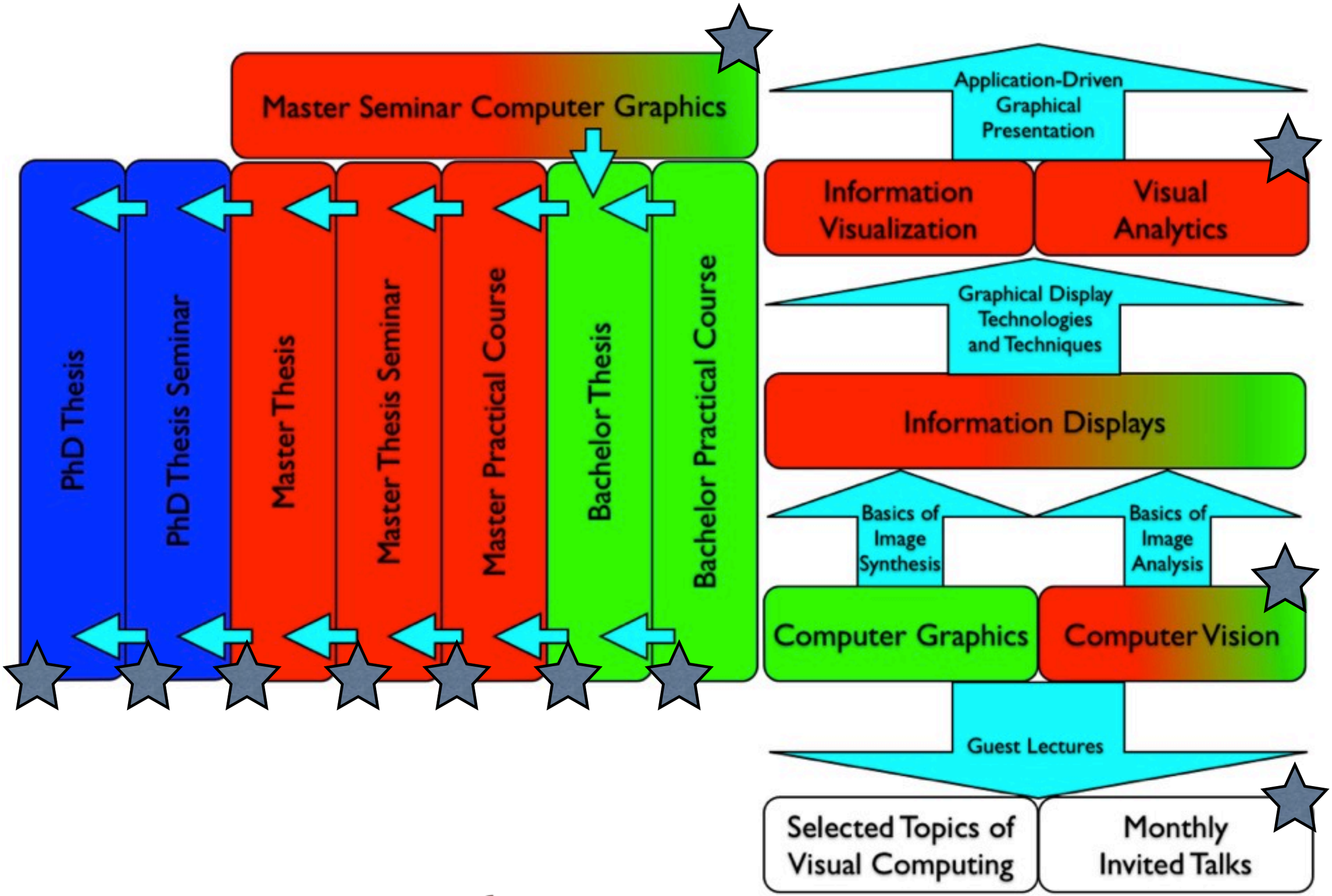
  - direct collaboration with our staff on own projects of interest

  - participation at international conferences (funded by institute)

  - organization of exchange possibilites with our collaboration partners (e.g., Stanford U., MIT, Harvard U., Microsoft, etc.)

  - participation in organized social events of the institute (e.g. summer hiking / winter skiing, weekly evening run, barbecues, christmas party, etc.)

What We Teach Next Semester

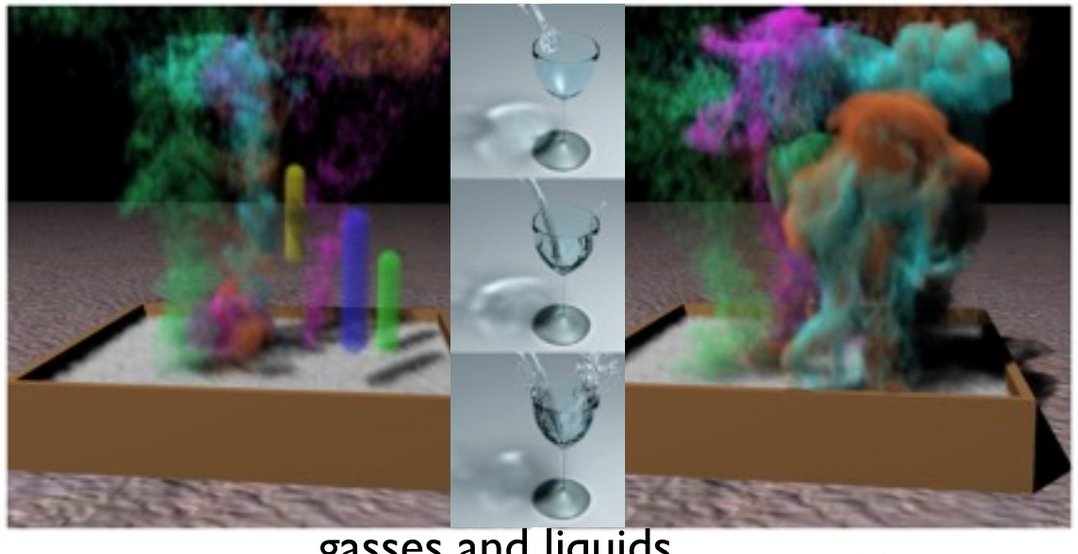


Computer Animation

- The field of computer animation is large
- As for other topics, it can cover an entire lecture on its own
- There are different themes, like:
  - physically-based animation (forward/ inverse kinematic, spring-mass systems, particle systems, rigid body simulation, etc.)
  - motion capture from real entities, like humans (face, body, movements, etc.)
  - animation of fluids, like liquids and gasses (fluid dynamics, etc.)
  - modeling and animating human figures (reaching, grasping, walking, dressing, etc.)
  - facial animation (muscle models, skin, lip synchronization, etc.)
- We will look at one of the most fundamental topics: interpolation-based animation

physics simulation

motion capturing



gasses and liquids

Computer Animation

- The field of computer animation is large
- As for other topics, it can cover an entire lecture on its own
- There are different themes, like:
  - physically-based animation (forward/ inverse kinematic, spring-mass systems, particle systems, rigid body simulation, etc.)
  - motion capture from real entities, like humans (face, body, movements, etc.)
  - animation of fluids, like liquids and gasses (fluid dynamics, etc.)
  - modeling and animating human figures (reaching, grasping, walking, dressing, etc.)
  - facial animation (muscle models, skin, lip synchronization, etc.)
- We will look at one of the most fundamental topics: interpolation-based animation

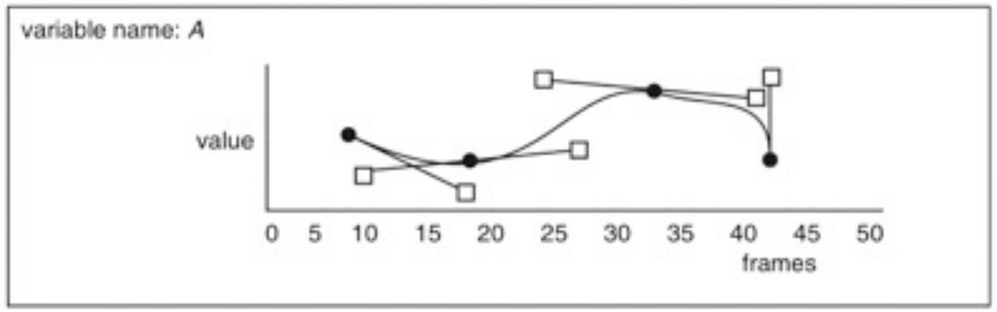
human figure and cloths

faces

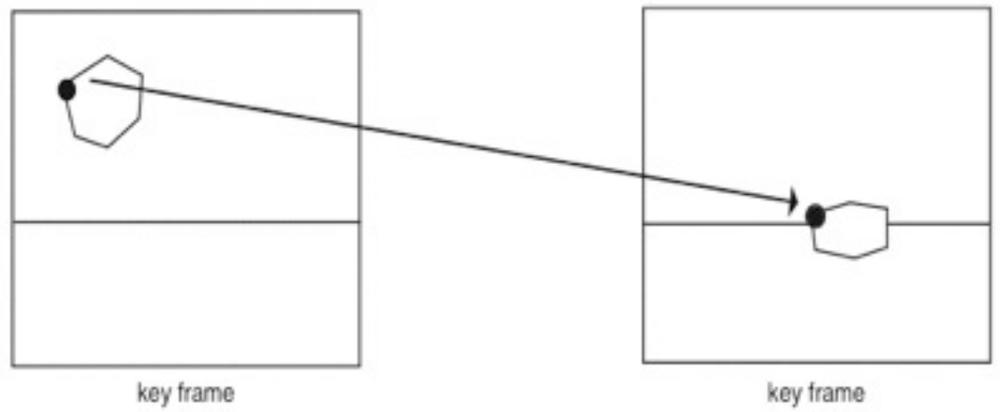
keyframes

Interpolation-Based Animation

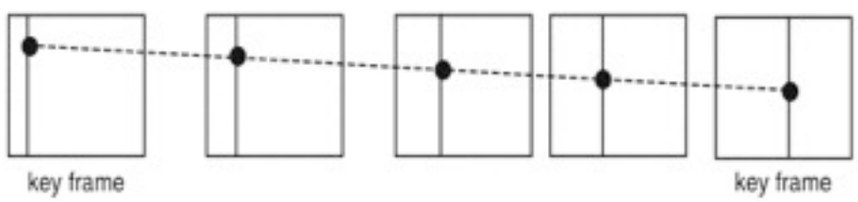
- The basic idea of interpolation-based animation is to pre-define particular states (called keyframes), and try to interpolate intermediate states that lead to a fluid animation
- Thus, the animator does not have to model each individual intermediate state, but only keyframes
- The system uses interpolation to compute intermediate frames automatically
- This is also called keyframe animation
- The parameters that are defined in each keyframe (which are interpolated for each intermediate frame) can be arbitrary, like vertices, normals or other tangential information, color, etc.
- The type of parameter does in general not influence the interpolation method itself (as we have seen in several other cases of CG)



parameters/variables, such as vertices and tangents at segments are defined at key positions - they are interpolated in between



Simple key frames in which each curve of a frame has the same number of points as its counterpart in the other frame



Keys and three intermediate frames with linear interpolation of a single point (with reference showing the progression of the interpolation in *x* and *y*)

Keyframe Interpolation

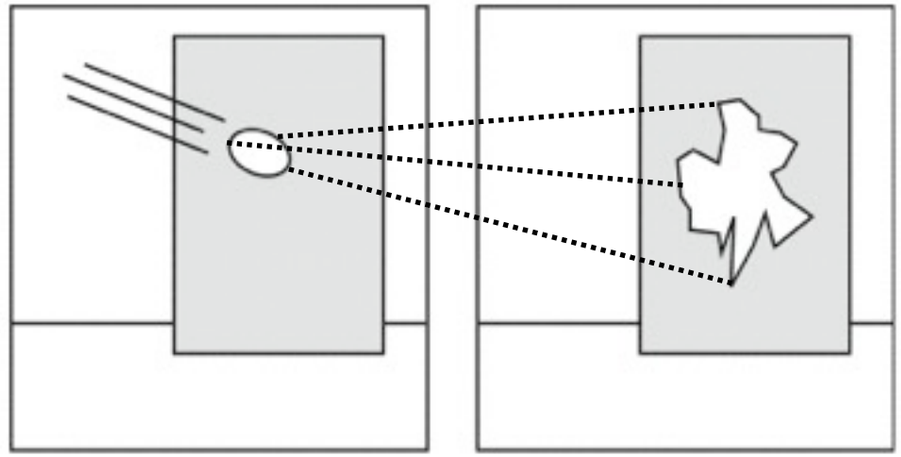
- One of the simples possibilities is to use linear interpolation, as we did many times in other situations

- Yet, this might not result in the desired animations, and hence, other possibilities exist

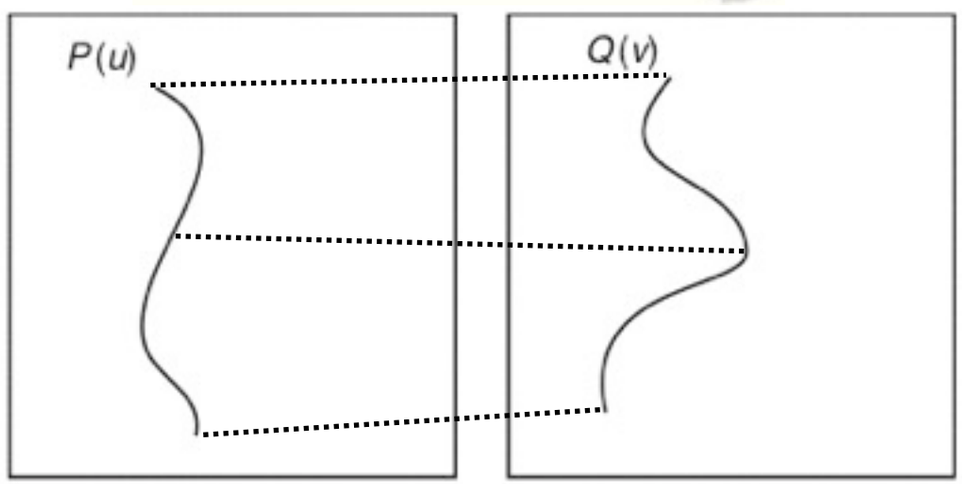
- If the surface that has to be animated is a spline surface (Bézier, B-spline, or NURBS) then the keyframes can be defined by modifying the control points only. They are interpolated, while the actual surface points are computed through the corresponding functions (i.e., Bézier, B-spline, or NURBS interpolation), as discussed in curves and surfaces

- One problem is, that each keyframe requires to have the same number of parameters, since we need a bijective (one-to-one) correspondence

- If this is not considered during modeling, then algorithms can be used that re-mesh the keyframes to ensure this



Frame f1 Frame f2



Rigid Body Animation of Hierarchies

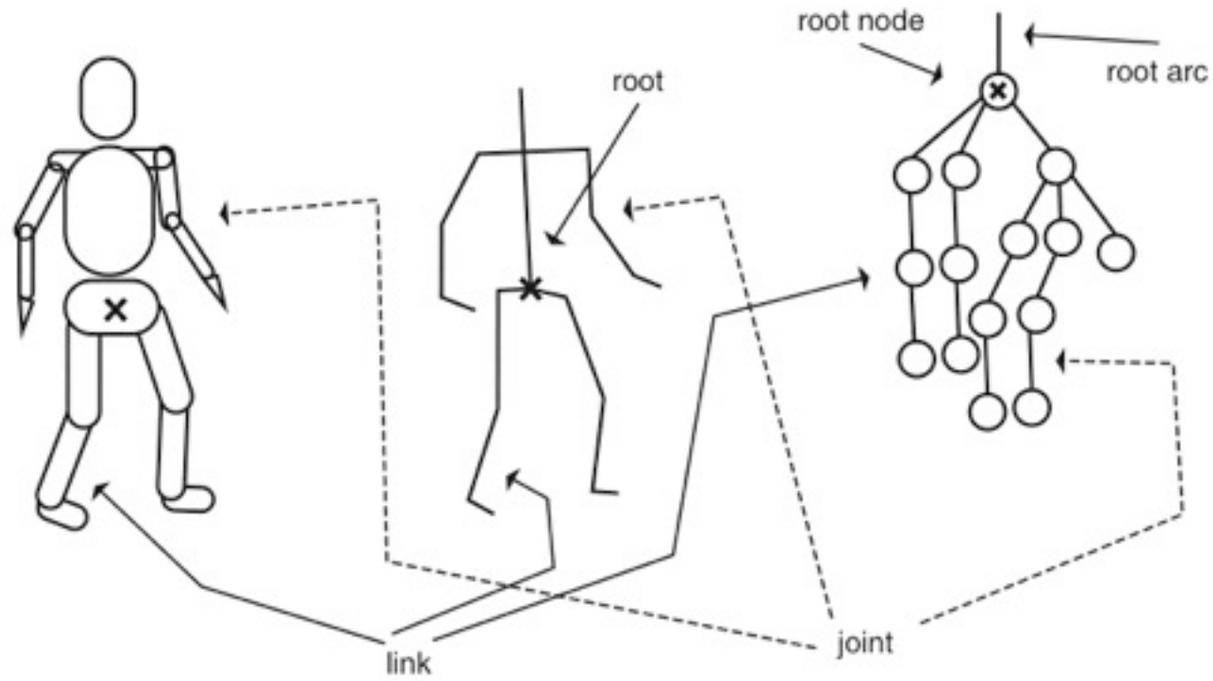
- We talked already about scene graphs

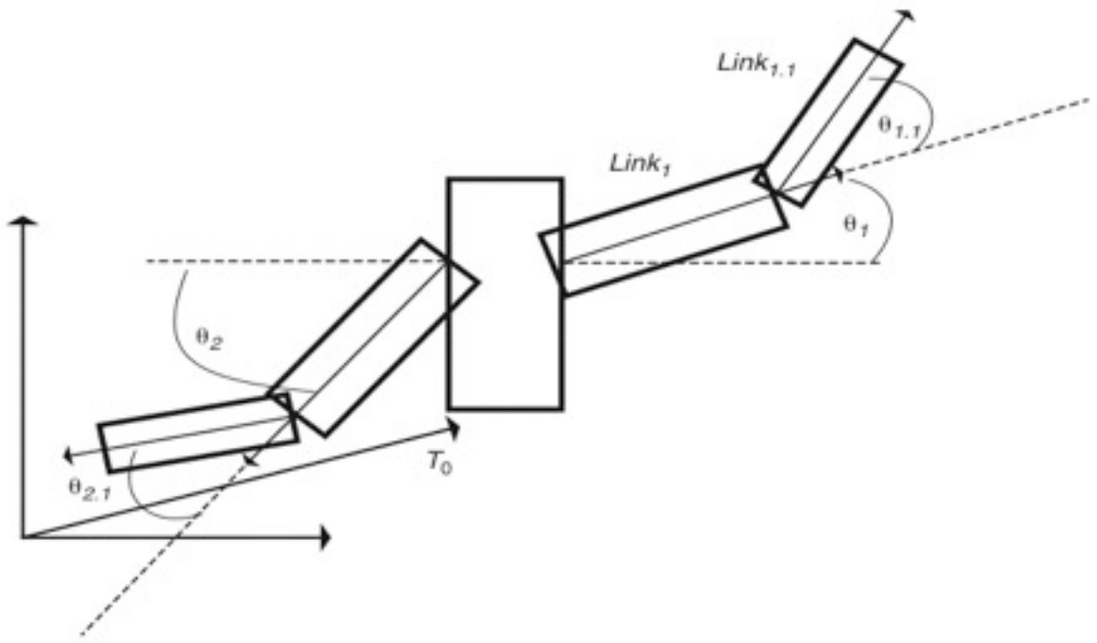
- We can animate rigid bodies that are composed hierarchically by defining keyframes (i.e., the parameters are the transformations at the nodes) and interpolating them to compute intermediate transformations

- Care has to be taken, since the components of arbitrary transformation matrices cannot be interpolated:

  - for example: translations and scales can be interpolated - components of a rotation matrix cannot (they are not independent!)

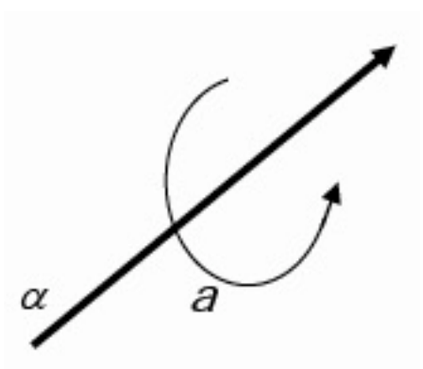
  - for rotations, we can use other forms of representations, such as Euler angles for each main axis, or quaternions (spherical linear interpolation required)





Quaternions in a Nutshell

- Quaternions can be used to represent rotations in space in a convenient manner (Hamilton, 1844)
- They are defined by a vector (imaginary part) a=(b,c,d) and an angle (real part) α: q=a+α
- Thereby, q is a unit quaternion if $|q|=\alpha^2+b^2+c^2+d^2=1$
- Note, that α,b,c,d are not independent - thus we cannot simply interpolate them - they have to be interpolated spherically (spherically linear interpolation) - ensuring they remain normal
- A unit quaternion can be converted into a rotation matrix R
- If q_1 and q_2 are unit quaternions then $q_1q_2=R_1R_2$
- The conjugate of q=a+α is $\overline{q}$=a-α
- Given a vector v, then $Rv=qv\overline{q}$ and $q\overline{q}=\overline{q}q=|q|^2=1$



$$R=\begin{pmatrix} a^2+b^2-c^2-d^2 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & a^2-b^2+c^2-d^2 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & a^2-b^2-c^2+d^2 \end{pmatrix}$$

Forward and Inverse Kinematics

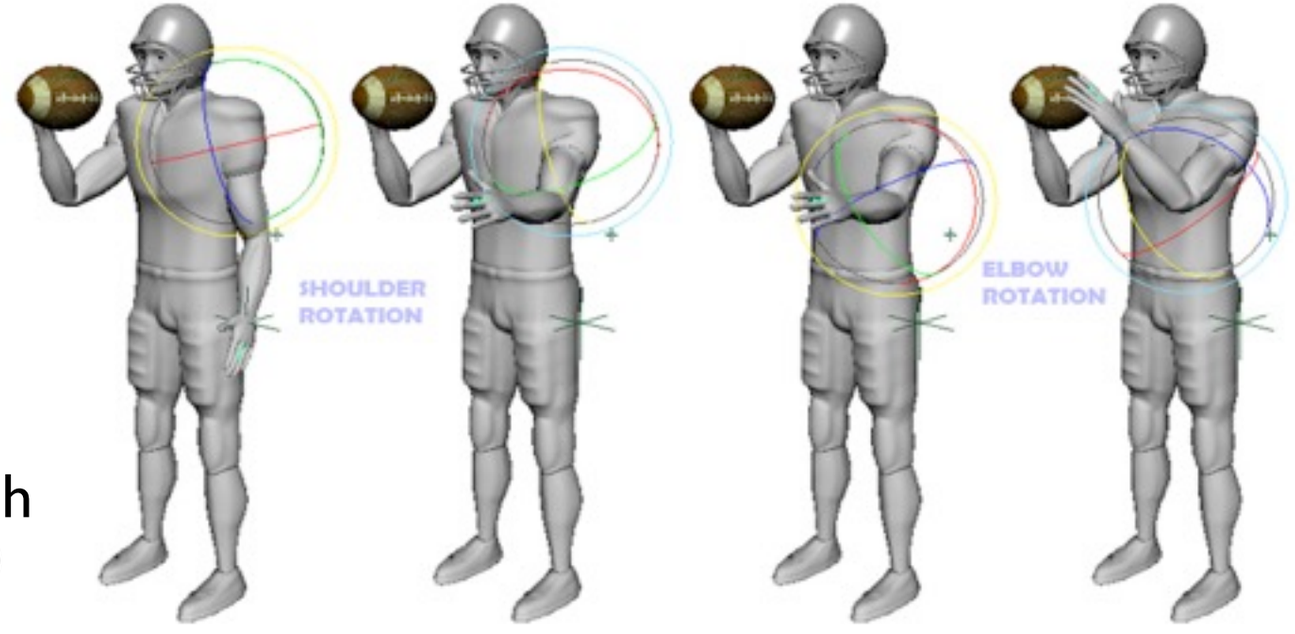
- Traversing a scene graph in depth-first order from root to leaves is equivalent to forward kinematics (e.g., we consider different joints, etc. by going down the tree in top down order)

- Finally, we have computed the positions of the leave objects (which are sometimes called end effectors)

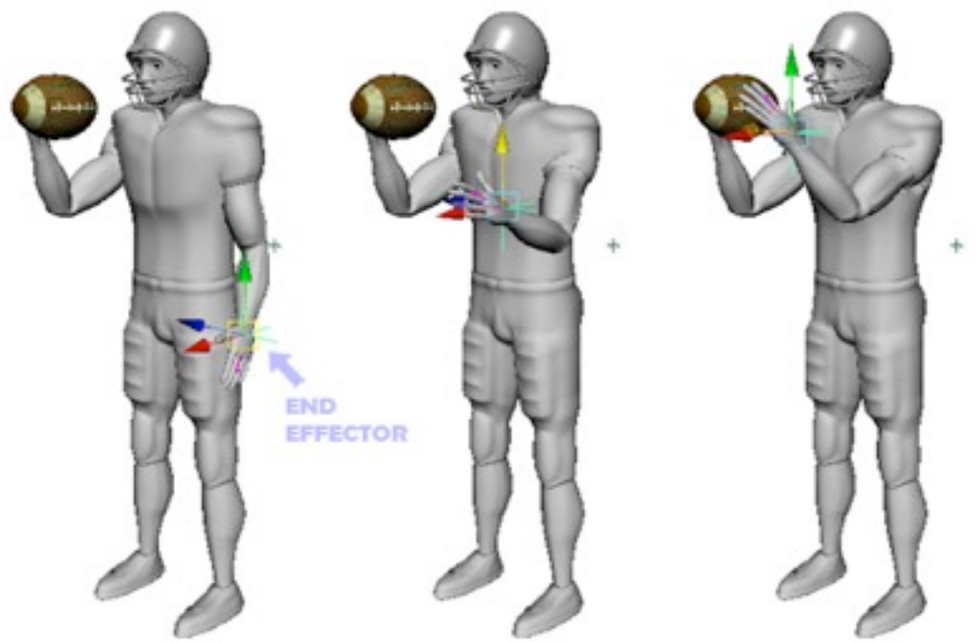
- For animation, it is sometimes much easier to define the end positions of the end effectors, and then compute the kinematically possible transformations by traversing the tree in bottom up order - this is called inverse kinematics

- For inverse kinematics, we have to solve (sometimes quite large) equation systems for computing the solutions numerically (we skip the details here)

- But how can we animate non-rigid, deformable objects?



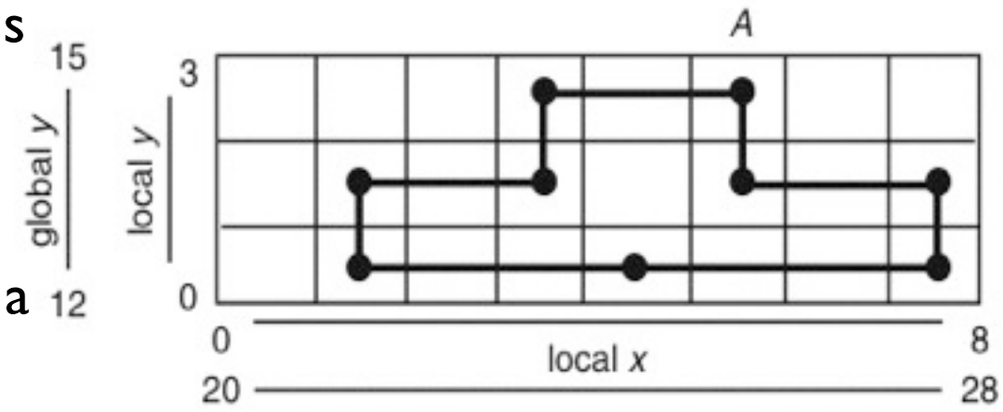
forward kinematics

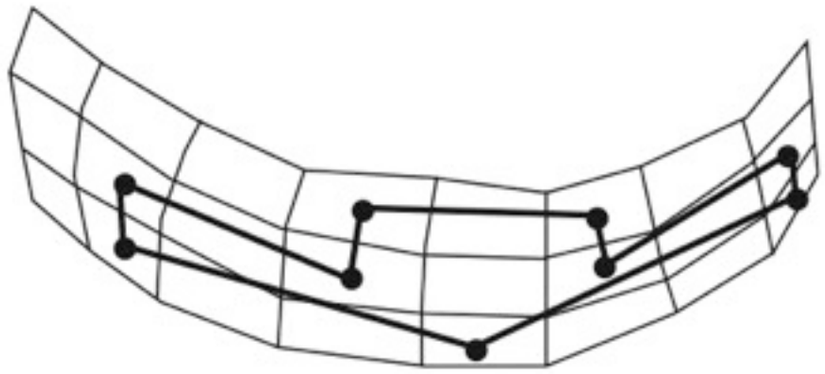


inverse kinematics

Free-Form Deformation (FFD)

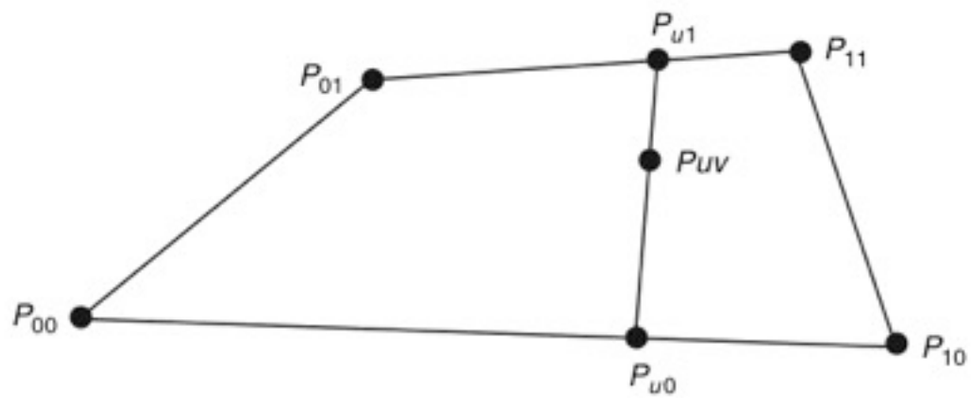
- Deforming objects by modifying object vertices directly is not necessarily the most efficient way
- Another possibility exists, that establishes a local coordinate system which encases the area of an object
- In a simple case, the local coordinate system is segmented into a grid structure
- Instead of deforming the object vertices, the grid vertices are deformed first
- After the grid vertices have been deformed, the new positions of the enclosed object vertices are computed within each cell
- This is more efficient, if the object consists of many vertices, since the local coordinate grid can have a much coarser resolution
- This technique is called free-form deformation, and is very common
- Compute local coordinates u,v of object vertices within undeformed cells, then deform cells, and compute new object vertices using u,v





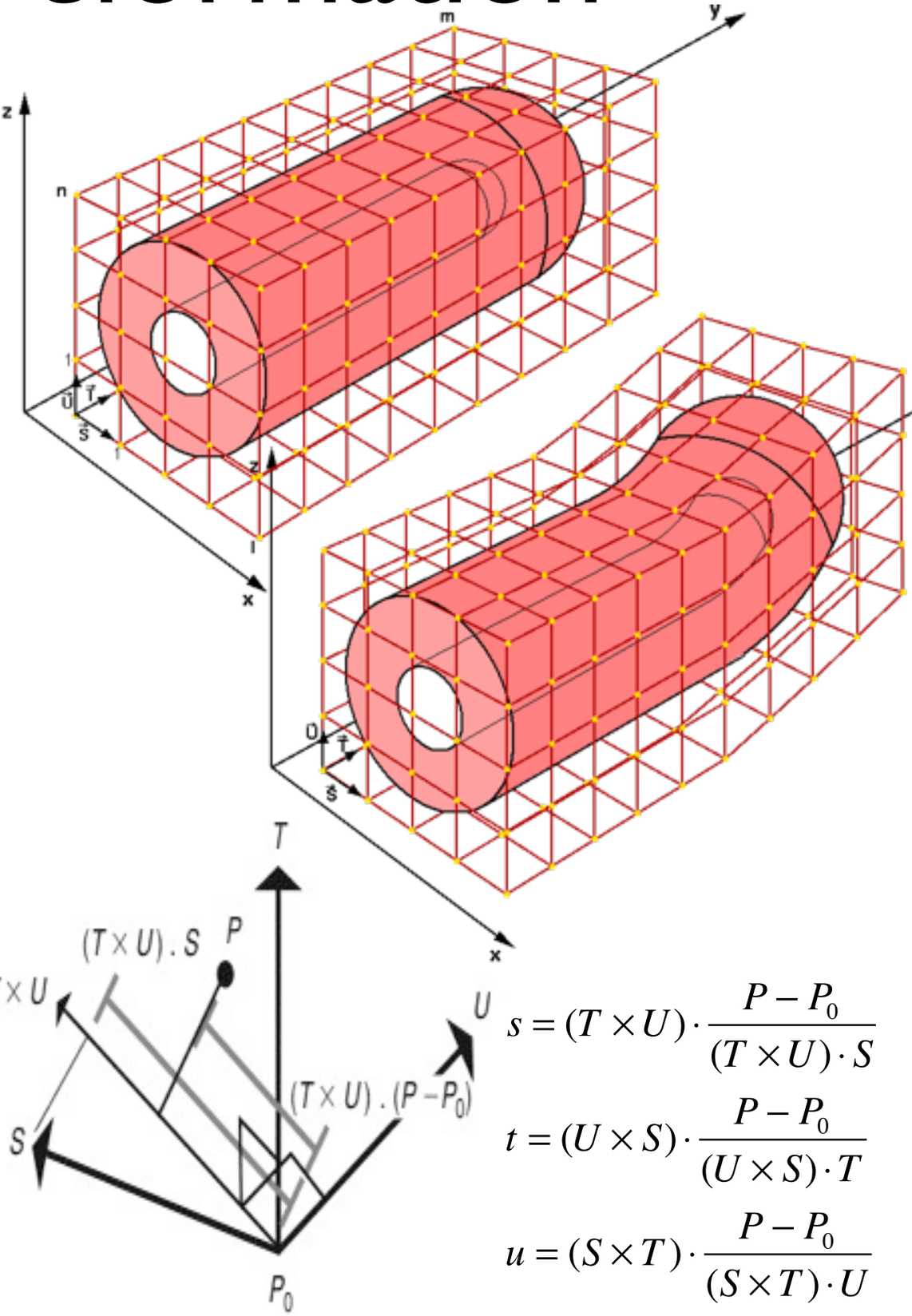
bilinear interpolation of object vertices within local grid cell

$$P_{u0} = (1-u)P_{00} + uP_{10}$$
$$P_{u1} = (1-u)P_{01} + uP_{11}$$
$$P_{uv} = (1-v)P_{u0} + vP_{u1}$$
$$= (1-u)(1-v)P_{00} + (1-v)uP_{01} + u(1-v)P_{10} + uvP_{11}$$



3D Free-Form Deformation

- Free-form deformation is the same in 3D as in 2D
- Yet, the grid cells become 3D too and the interpolation of the grid vertices usually becomes trilinear
- The local coordinates become also 3D (called trilinear interpolants): s,t,u
- The local grid axes S,T,U do not necessarily have to be orthogonal
- The denominators normalize the local coordinates
  - for s, for instance, the denominator is the projected distance of S onto TxU (and similar for the other values)



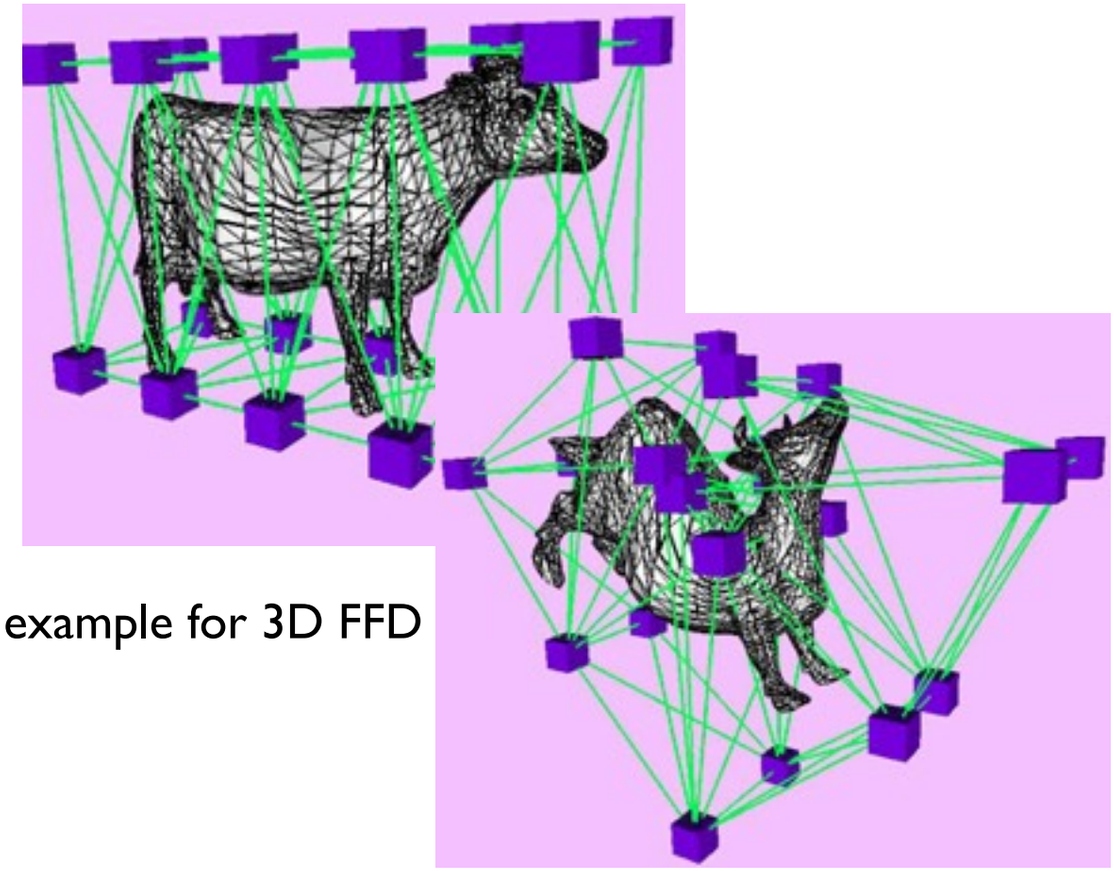
$$s = (T \times U) \cdot \frac{P - P_0}{(T \times U) \cdot S}$$

$$t = (U \times S) \cdot \frac{P - P_0}{(U \times S) \cdot T}$$

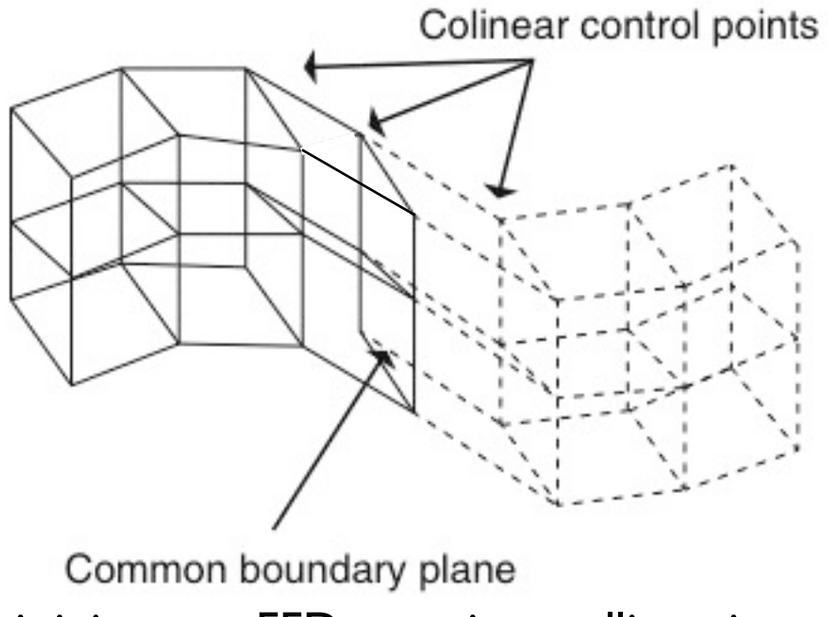
$$u = (S \times T) \cdot \frac{P - P_0}{(S \times T) \cdot U}$$

Combining Free-Form Deformations

- We can segment the objects into multiple FFDs (i.e., local coordinate systems) that can be merged
- This allows animating different parts of an object differently
- But we have to ensure continuity at the boundaries!
- Like Bézier curves and surfaces, Bézier solids can be joined with continuity constraints across the boundaries (we did not talk about this at curves and surfaces)
- To ensure positional continuity, adjacent control lattices must share the same control points along the boundary plane
- We can ensure this, by enforcing co-linearity among adjacent control points across the common boundary



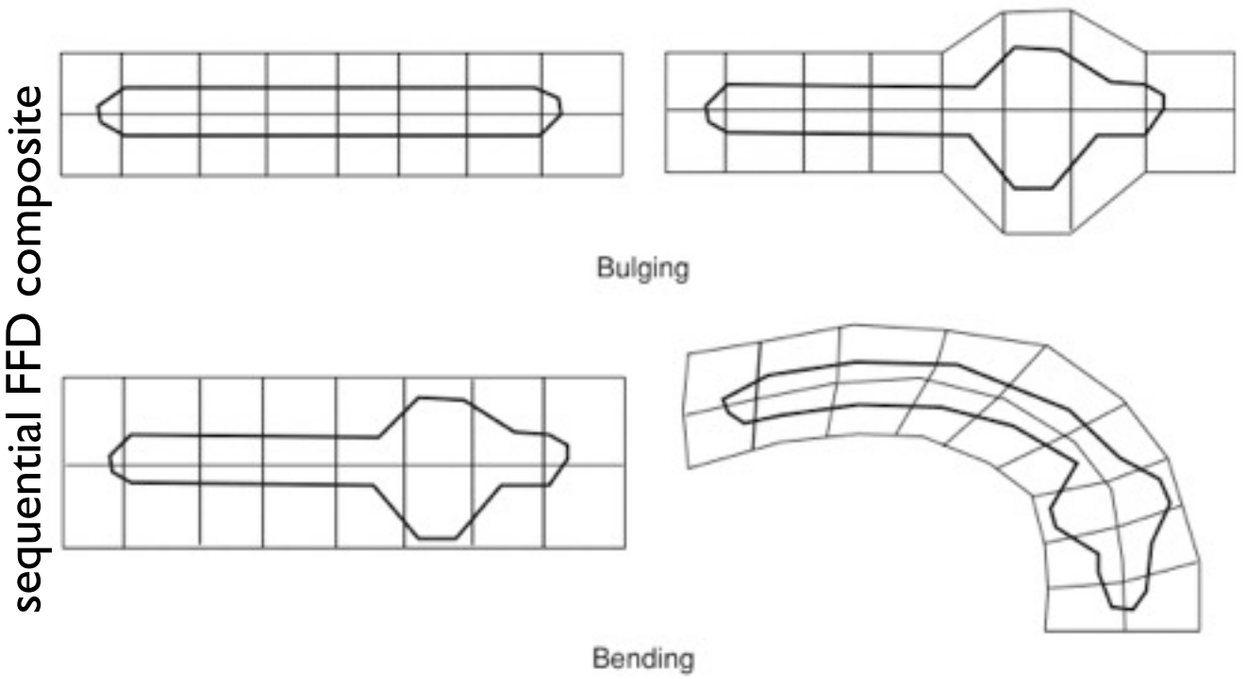
example for 3D FFD



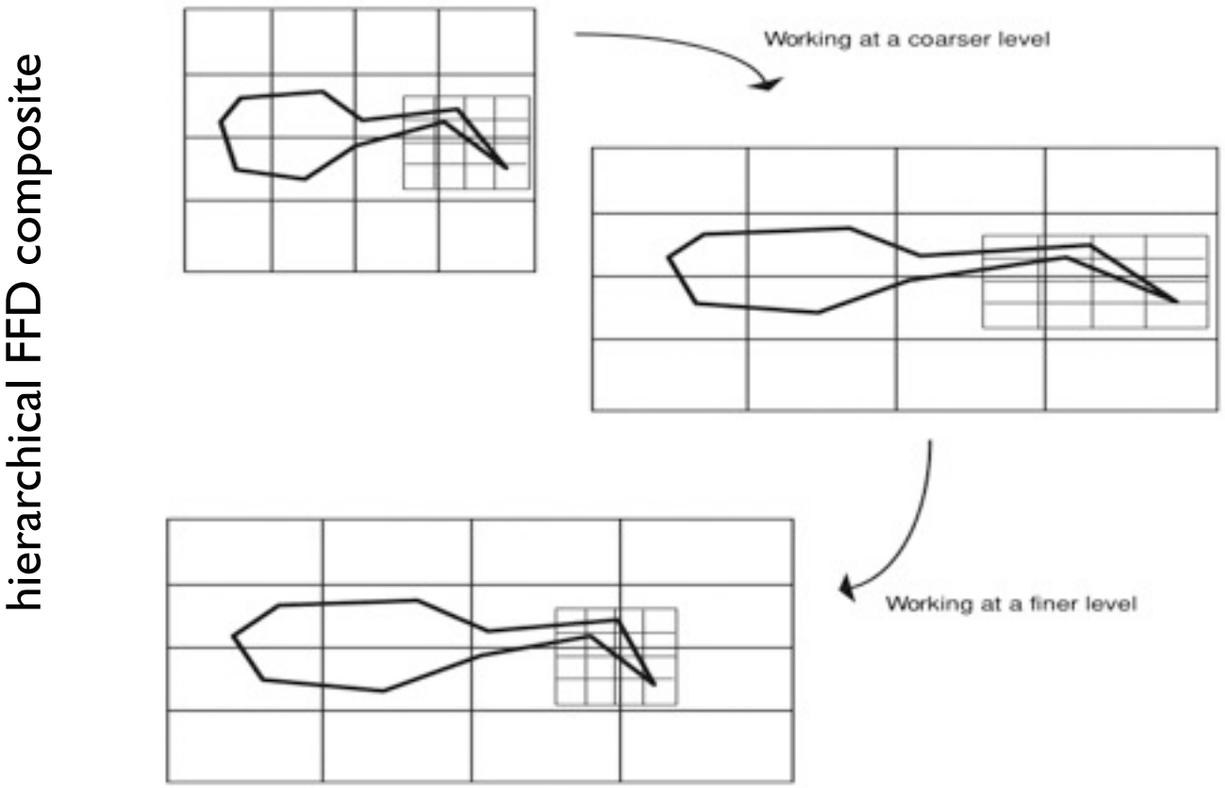
joining two FFDs requires collinearity among adjacent control points

Composite Free-Form Deformations

- Free-form deformations can be composed sequentially or hierarchically
- Sequentially: progressing through a sequence of FFDs (allows combining different modeling steps)

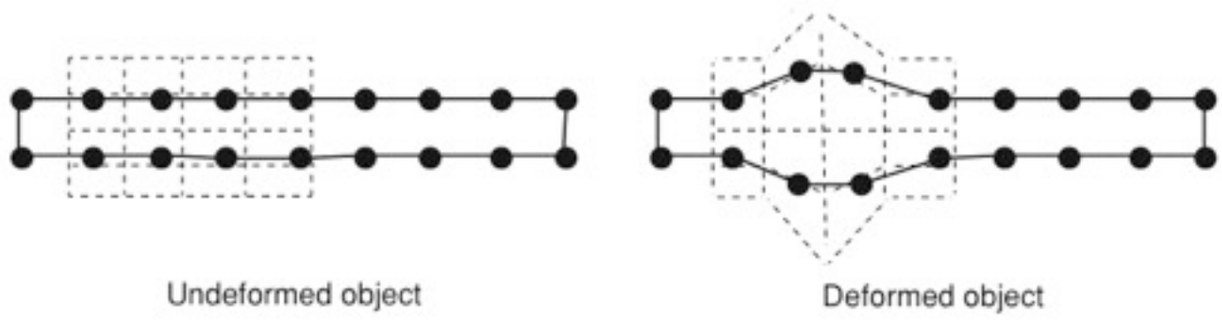


- Hierarchically: finer scale FFDs are embedded in coarser scale FFDs (allows to work at different level of details) - object points and control points of fine FFDs can be animated through coarser FFDs

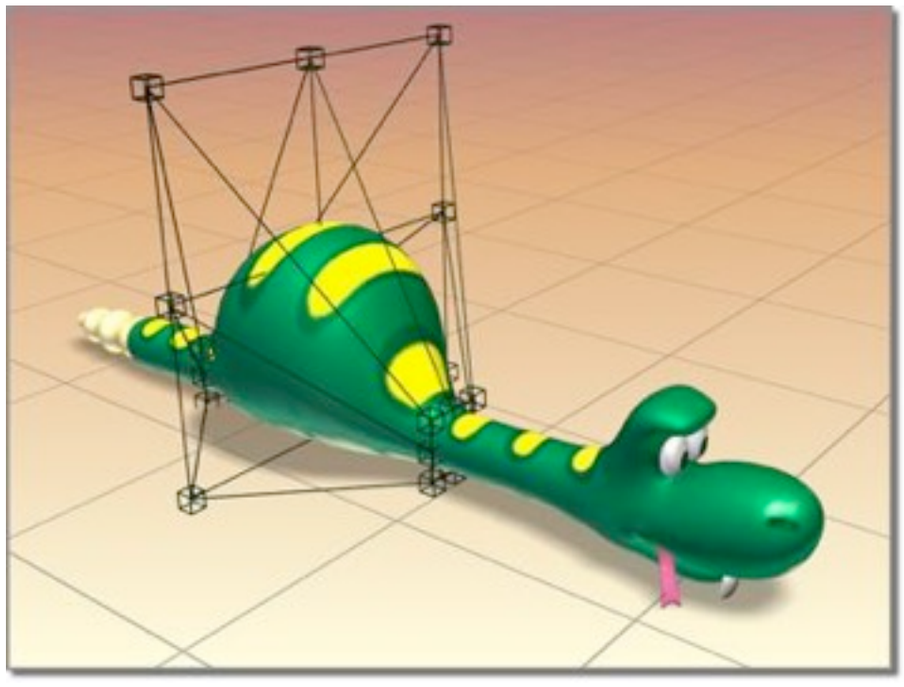


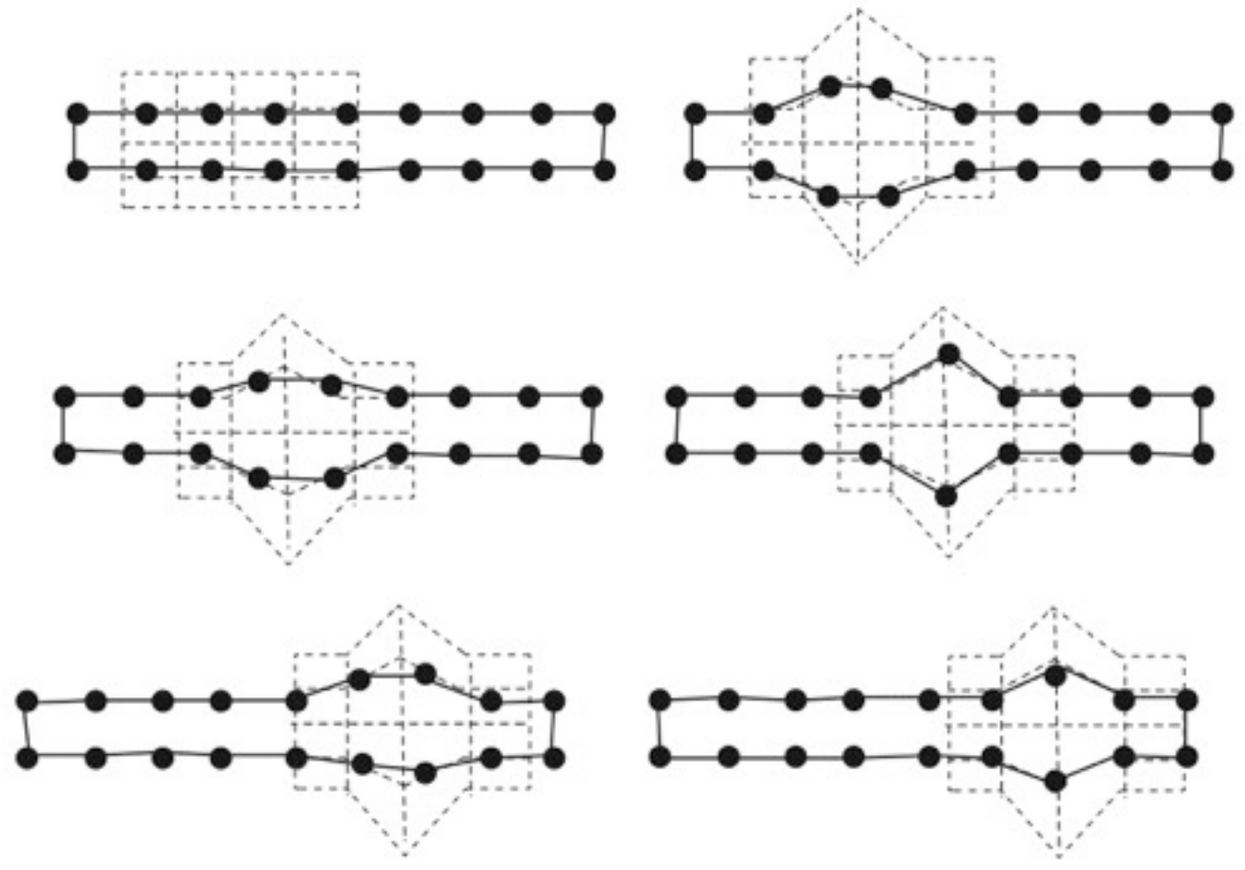
Free-Form Deformation Tools

- Individual FFDs can be modeled, and then be reused as tools for deforming objects
- Example: template can be swept though object geometry to model various animation effects
- Instead of manually defining the FFDs control points, physically-based simulations and other functions can be used to compute them



FFDs can be modeled and animated through object geometry as a (reusable) tool for creating a special animation effects





Computer Graphics

- In this course, you only got an overview - each topics can be a lecture on its own

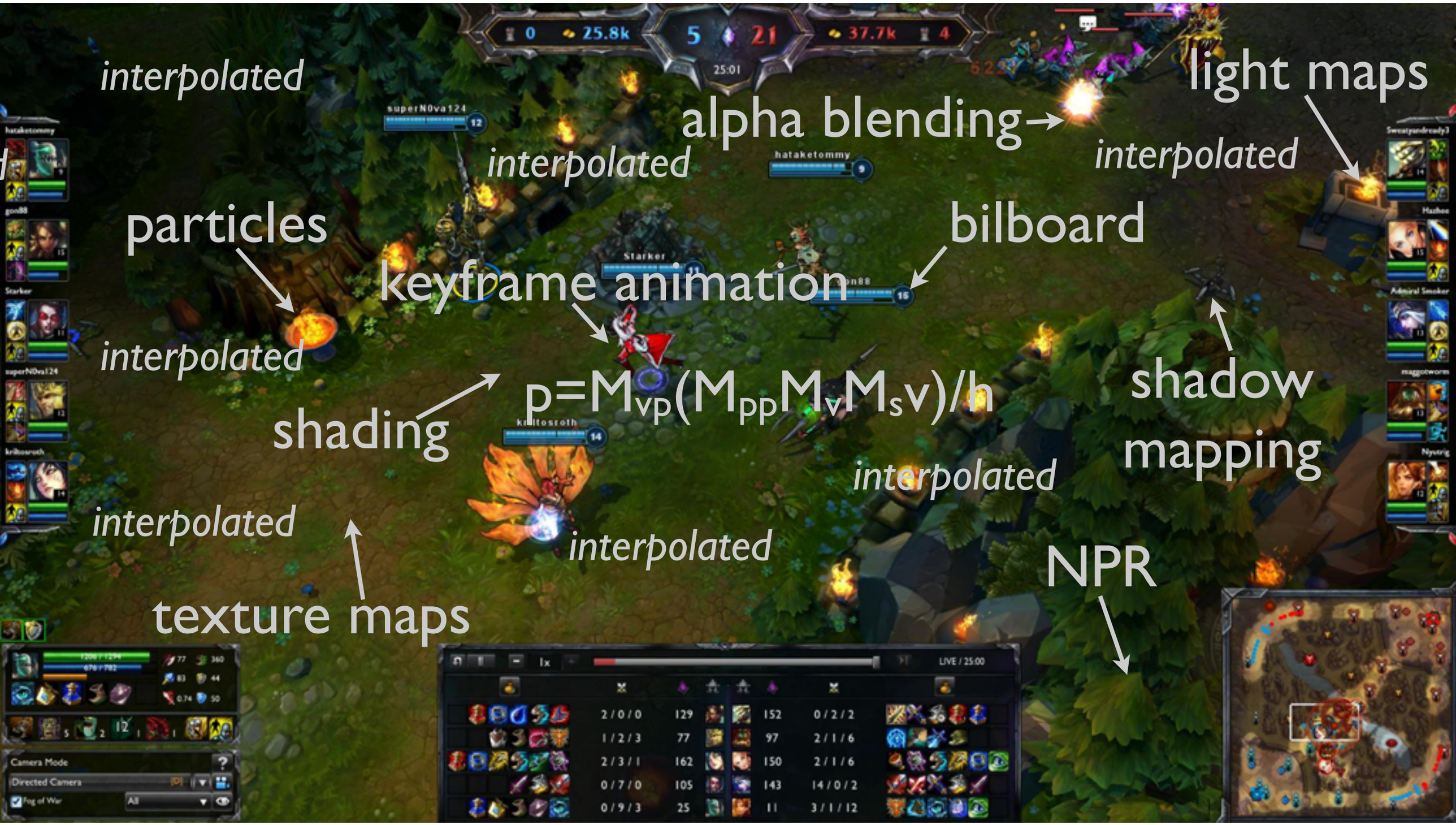
- But you learned the most fundamental techniques in graphics

"All problems in computer graphics can be solved with a matrix inversion."
-- Jim Blinn

... you need a little bit of interpolation, too...

Computer Graphics

Computer Graphics



Computer Graphics

- In this course, you only got an overview - each topics can be a lecture on its own
- But you learned the most fundamental techniques in graphics
- Next time you watch a movie with visual effects or play a game, remember that in principle all you see comes down to these fundamental methods at some point (off course, with more computational power than we had available in our labs)
- However, for good graphics (visual effects in movies or games) your need much more than a fundamental understanding of these techniques - you also need an understanding of graphical design (most animators have an artistic background - not a scientific one)
- So, computer graphics is not magic - its just a successful (also commercially) combination of art and science

"All problems in computer graphics can be solved with a matrix inversion."
-- Jim Blinn

... you need a little bit of interpolation, too...

Course Schedule

Type	Date	Time	Room	Topic	Comment
C1	03.03.2015	13:45-15:15	HS 18	Introduction and Course Overview	
C2	12.03.2015	12:00-13:30	HS 19	Transformations and Projections	Thursday instead of Tuesday (Conference)
C3	17.03.2015	13:45-15:15	HS 18	Raster Algorithms and Depth Handling	
C4	24.03.2015	13:45-15:15	HS 18	Local Shading and Illumination	
C5	14.04.2015	13:45-15:15	HS 18	Texture Mapping Basics	Easter Break: 30.03.2015-10.4.2015
C6	20.4.2015	17:15-18:45	HS 18	Advanced Texture Mapping & Graphics Pipelines	Monday instead of Tuesday (Conference)
C7	05.05.2015	13:45-15:15	HS 18	Intermediate Exam	
C8	12.05.2015	13:45-15:15	HS 18	Global Illumination I: Raytracing	
C9	19.05.2015	13:45-15:15	HS 18	Global Illumination II: Radiosity	
C10	20.05.2015	8:30-10:00	HS 19	Volume Rendering	Wednesday additional (Whitsun)
C11	27.05.2015	8:30-10:00	HS 19	Scientific Data Visualization	Wednesday instead of Tuesday (Whitsun)
C12	02.06.2015	13:45-15:15	HS 18	Curves and Surfaces	
C13	03.06.2015	8:30-10:00	HS 19	Basics of Animation	Shifted Appointment
C14	30.06.2015	13:45-15:15	HS 18	Final Exam	
C15	06.10.2015	8:00-9:30	TBA	Retry Exam	

Thank You!