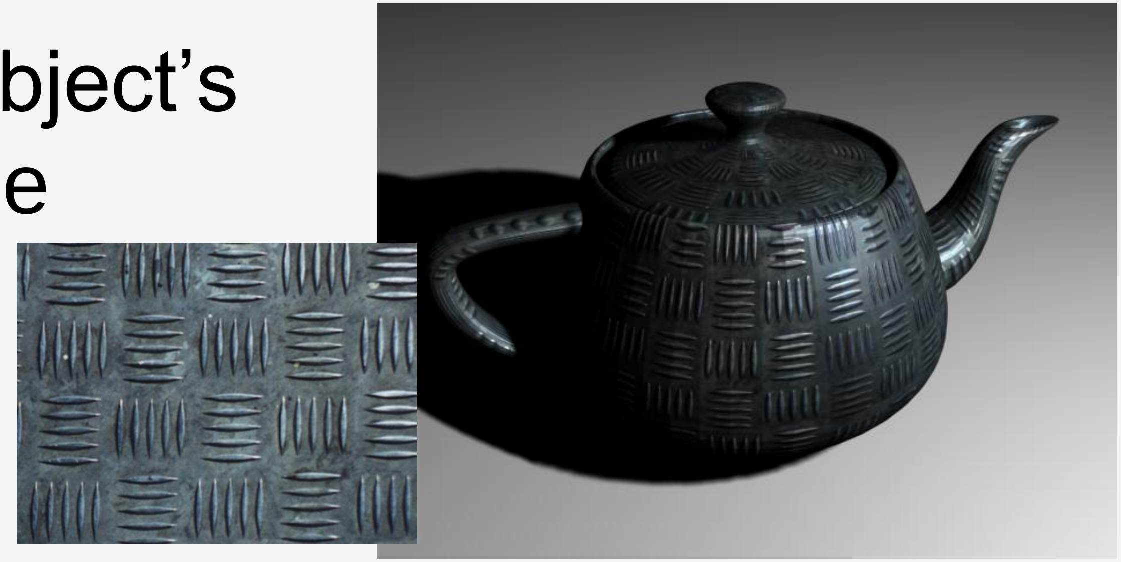
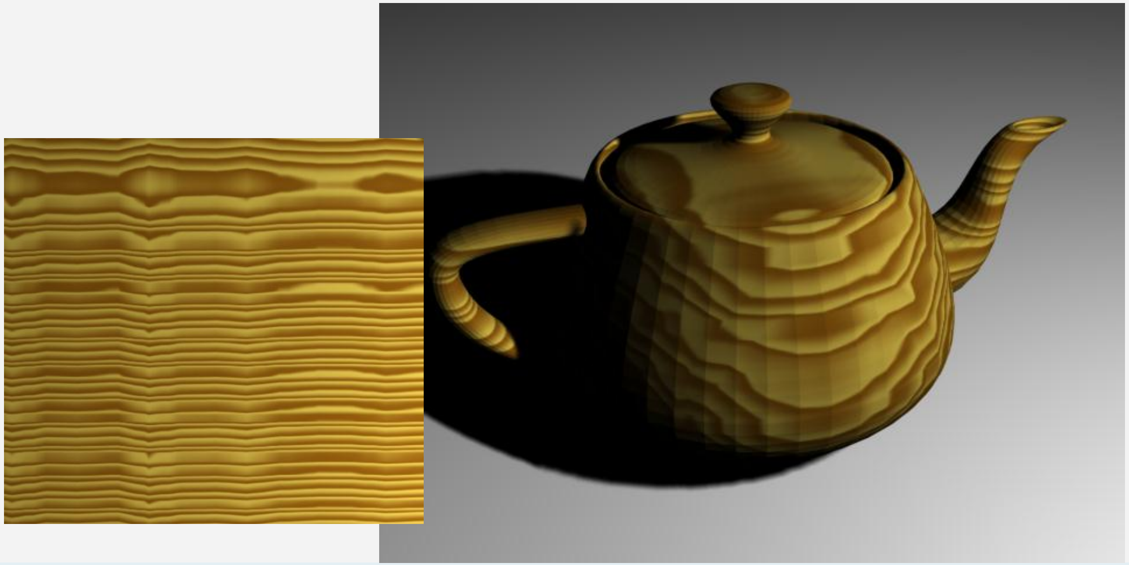
Textures and materials

Texture

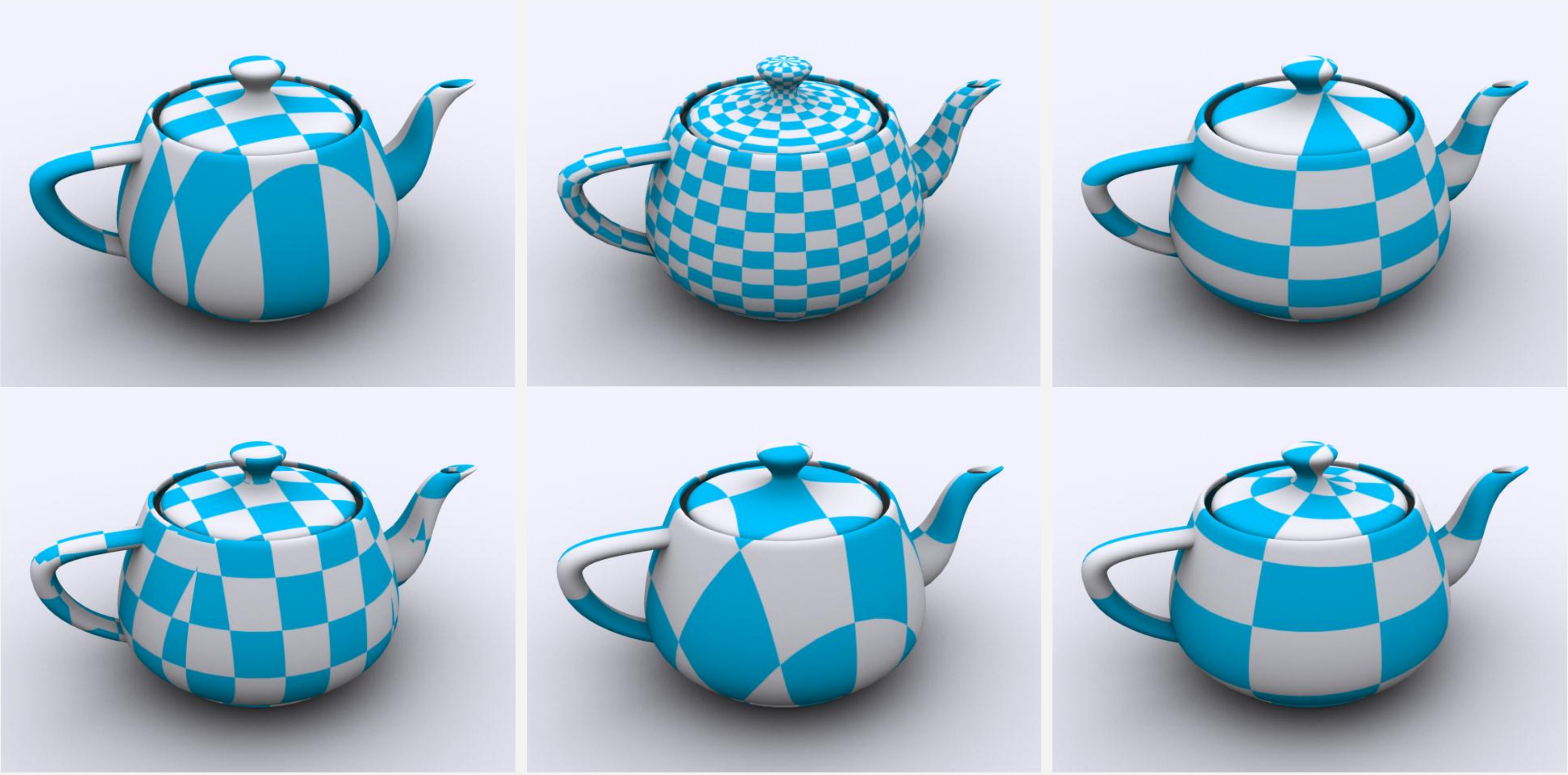
- used to define object’s color appearance

- 2D bitmap
- 3D bitmap
- texel
- procedural texture





Texture mapping



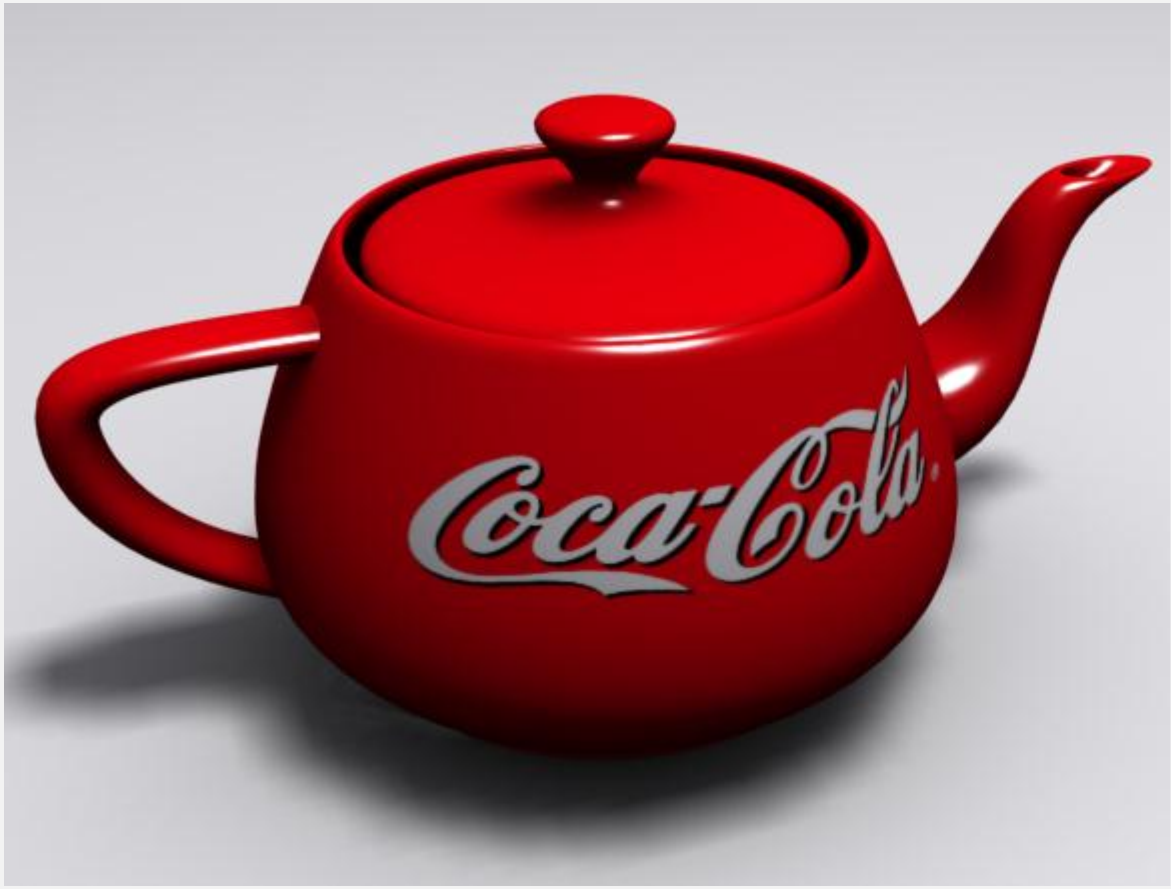
Texture usage

- object diffuse color
  - patterns, decals
- modulate surface properties
  - bumps, displacements
- modulate lighting properties
  - e.g. shininess
- simulate physical phenomena
  - reflection, refraction, global illumination

MULTITEXTURING

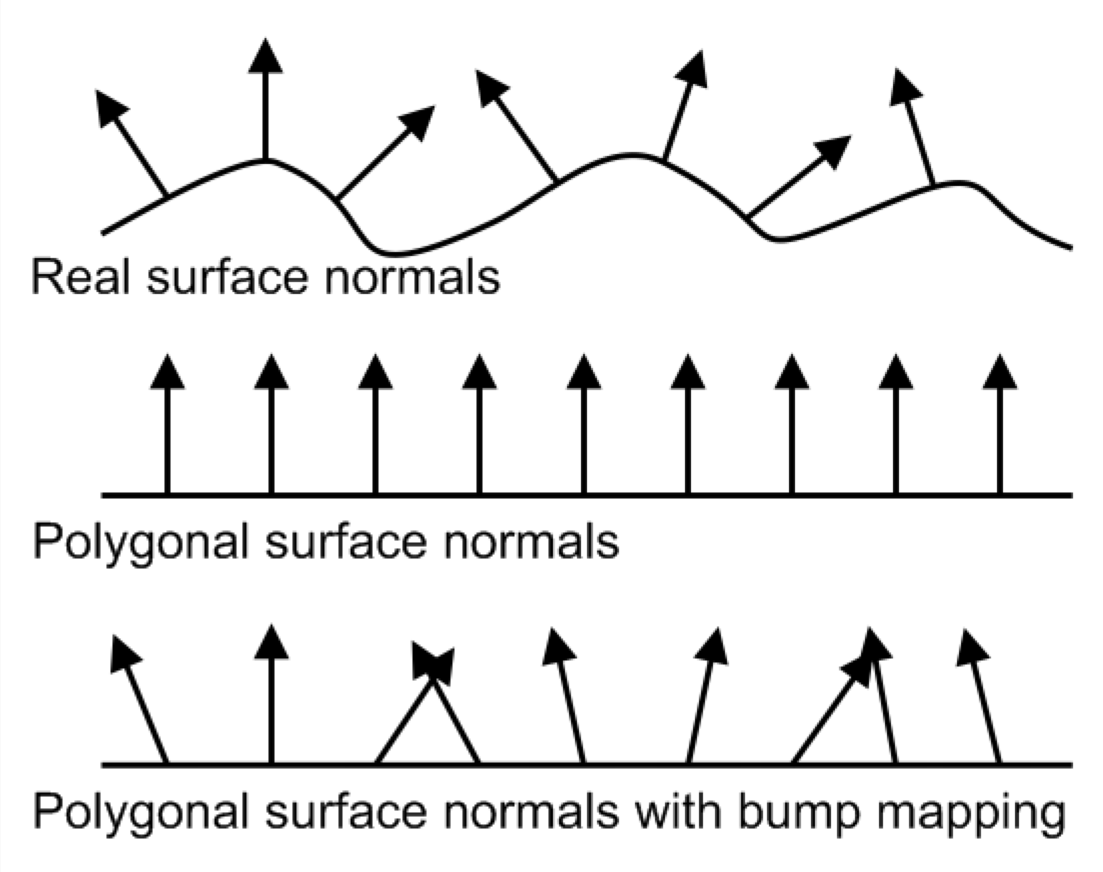
Diffuse color

- simulate paint, decals, patterns

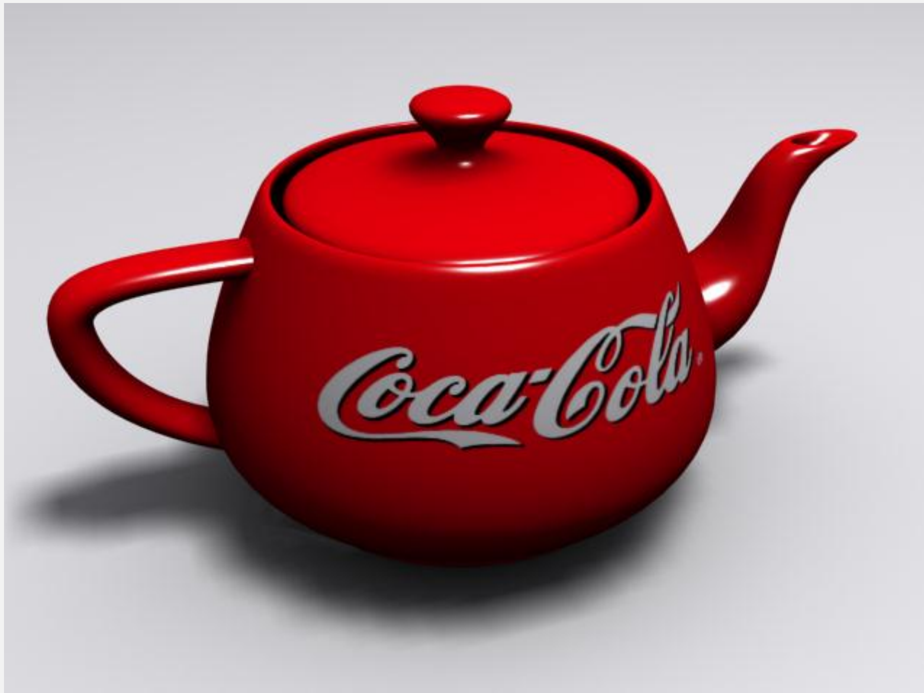


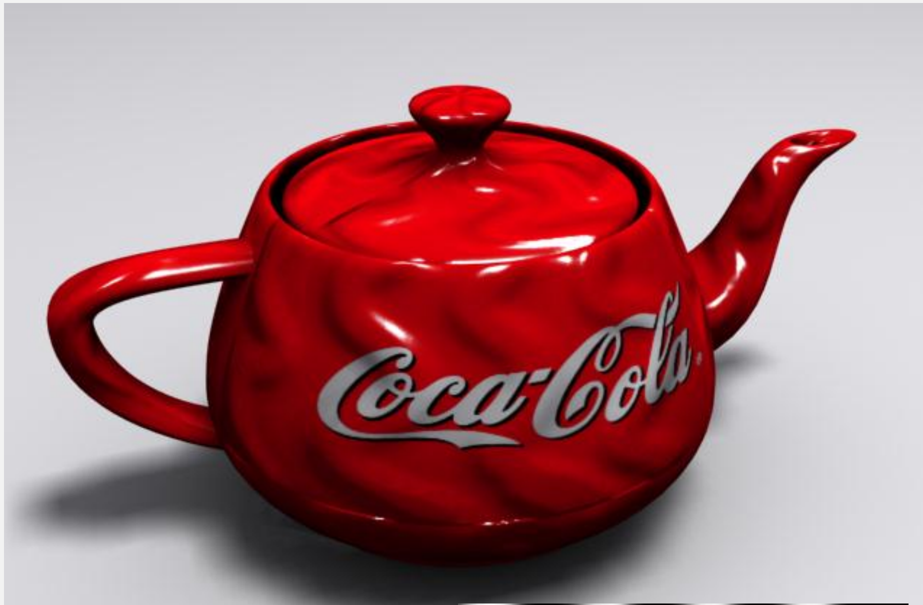
Bump mapping

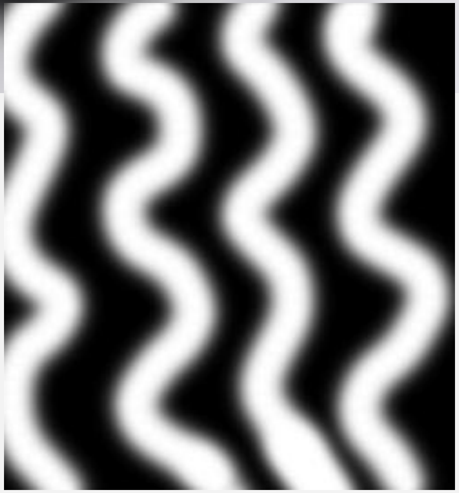
- modulate surface normal of a low-polygon model to simulate detailed surface geometry


Real surface normals
Polygonal surface normals
Polygonal surface normals with bump mapping

Bump mapping example





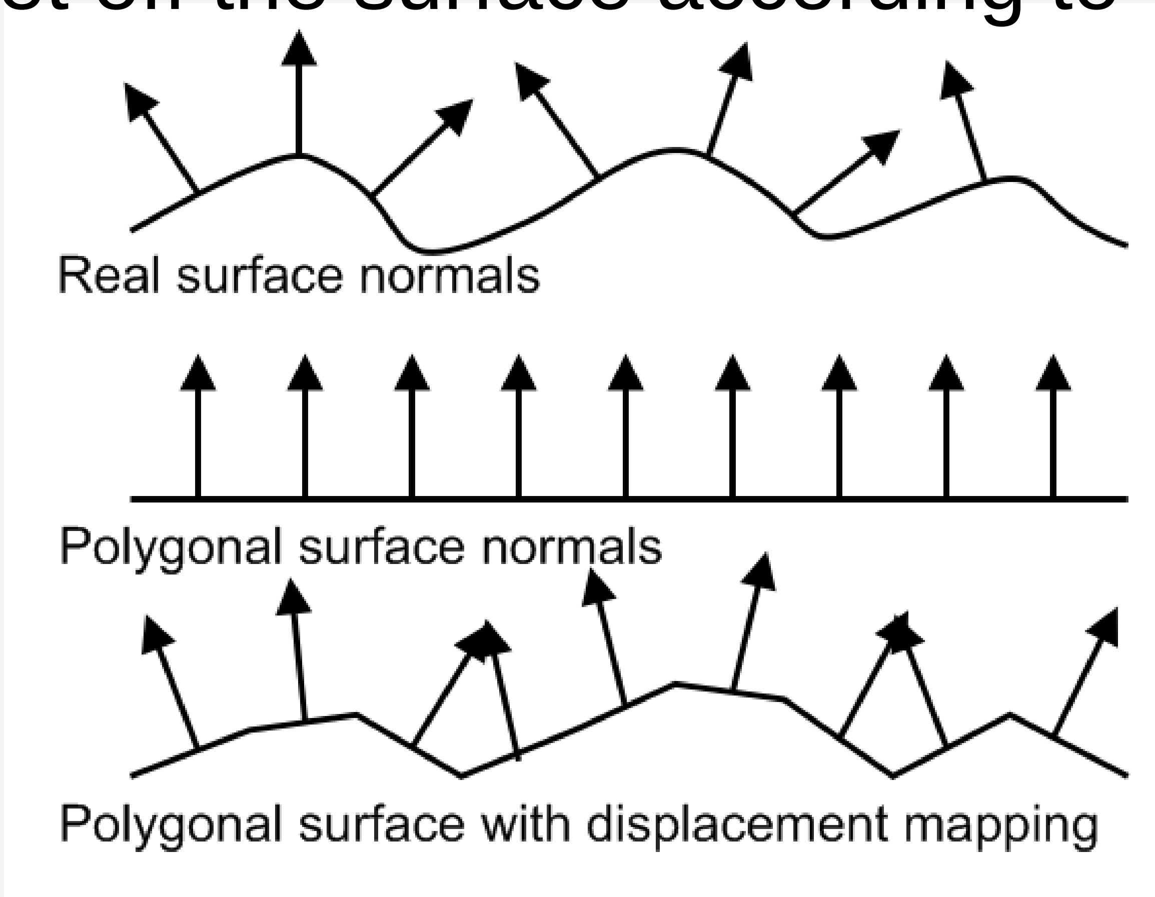


bump texture

- color intensity encodes difference between real surface and polygon

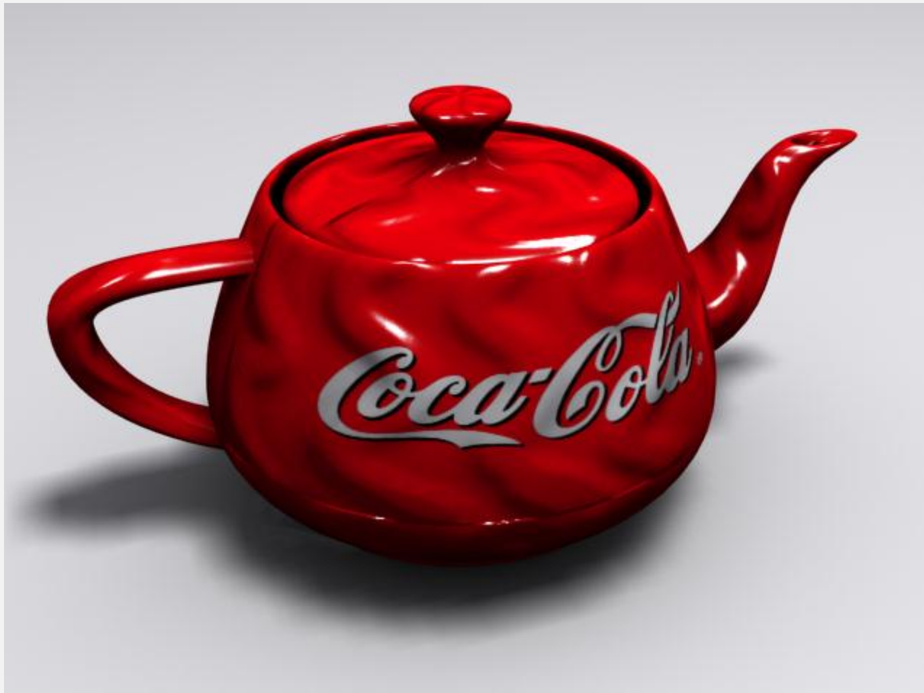
Displacement mapping

- Face is tessellated into smaller faces
- Vertexes are set off the surface according to color intensity stored in the texture

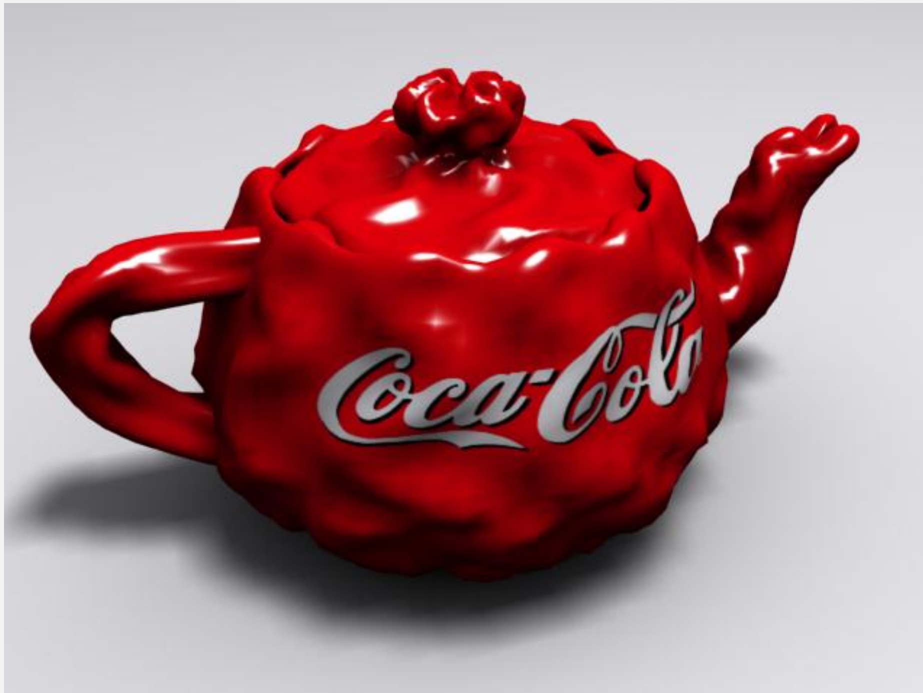

Real surface normals
Polygonal surface normals
Polygonal surface with displacement mapping

Displacement mapping

- similar to bump mapping
- but changes geometry



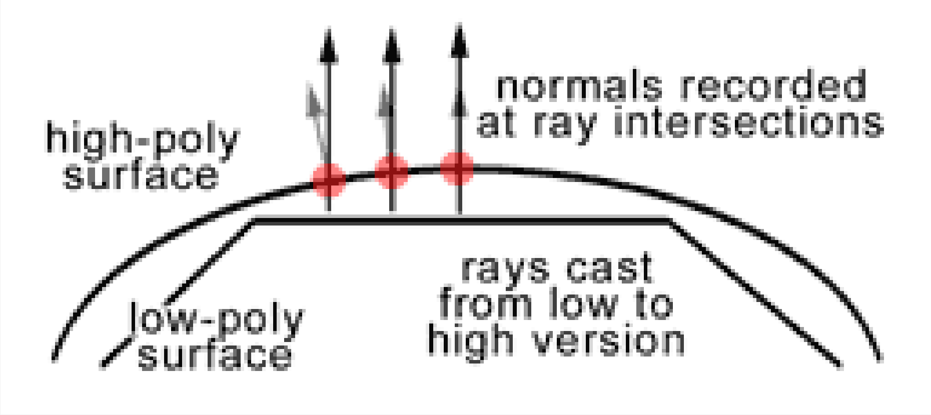
bump mapping



displacement mapping

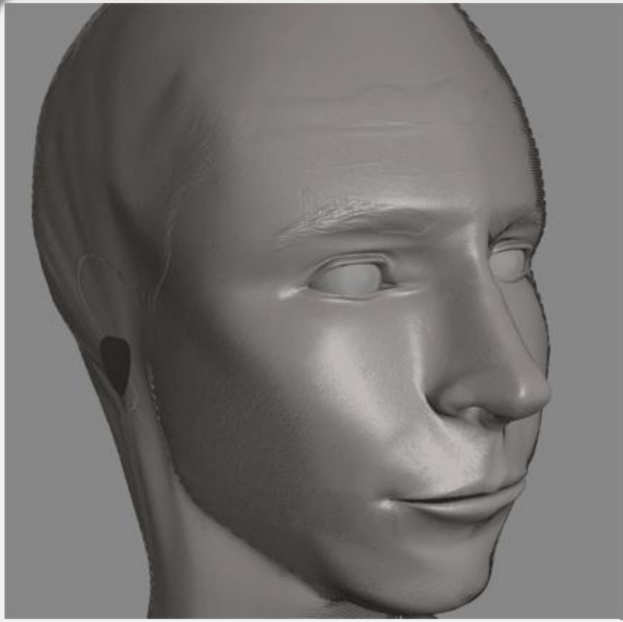
Normal mapping

- Texture’s RGB values are used to store x,y,z coordinates of the local normal vector
- Normal vectors computed on hi-poly model, then mapped by means of a texture to a low-polygon model.

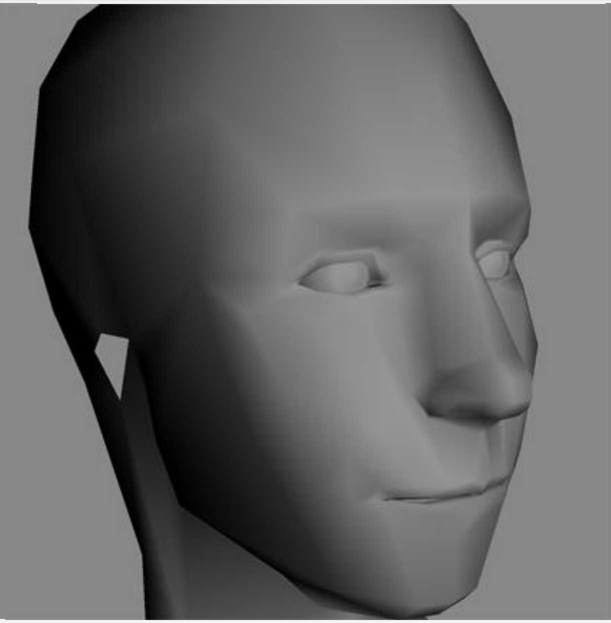

normals recorded at ray intersections
high-poly surface
low-poly surface
rays cast from low to high version

http://www.bencloward.com/tutorials_normal_maps1.shtml

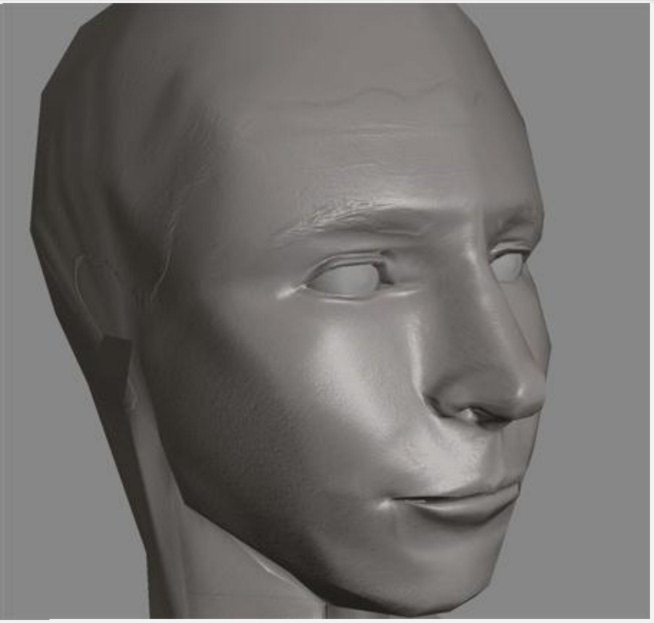
Normal mapping



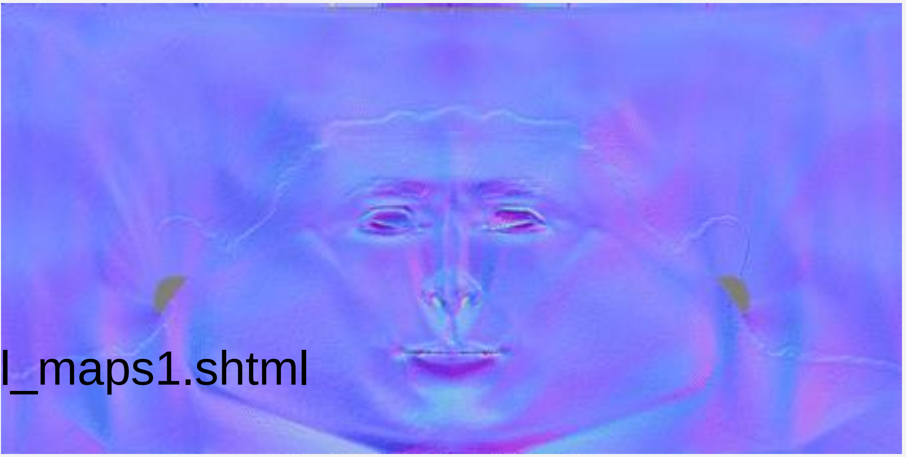
30,000 polys



632 polys



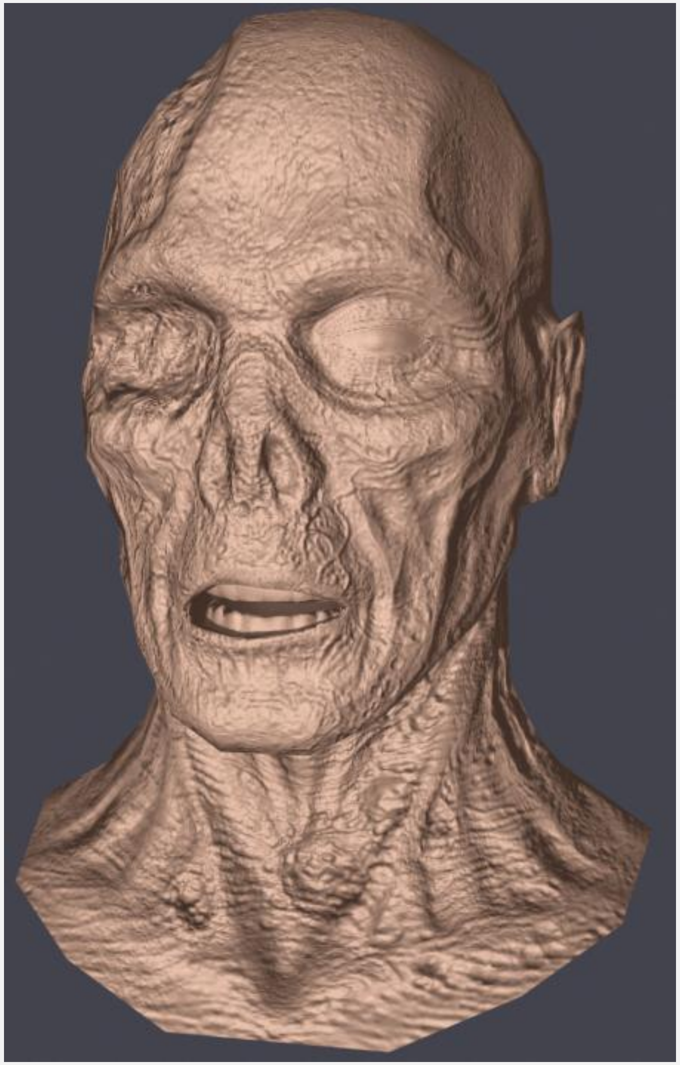
632 polys + normal map

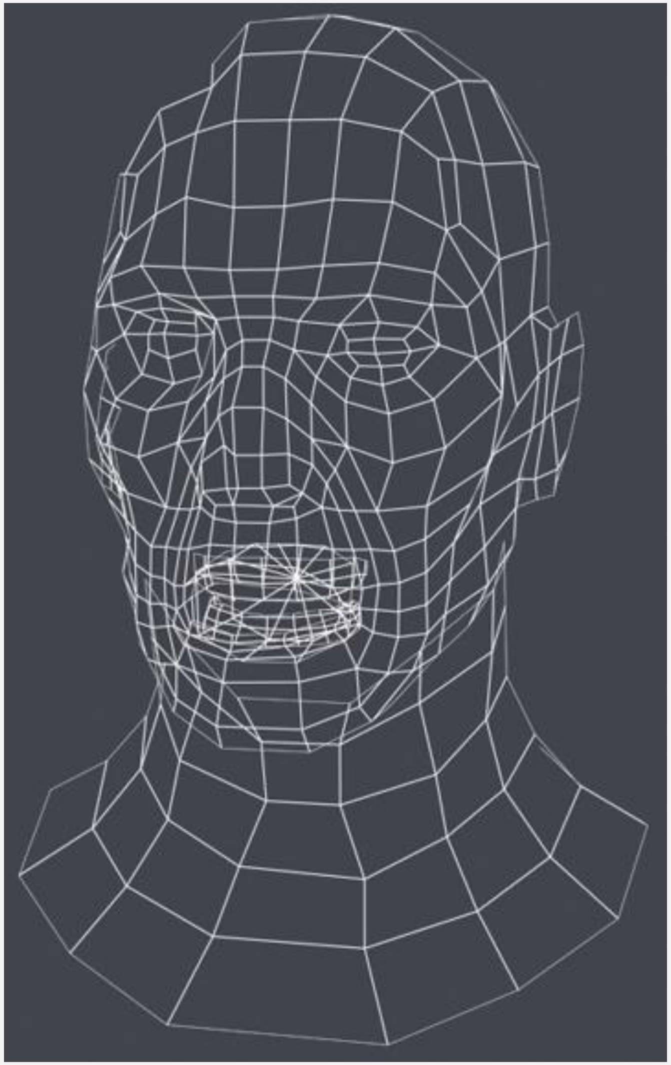


http://www.bencloward.com/tutorials_normal_maps1.shtml

Another example

- 531 polys



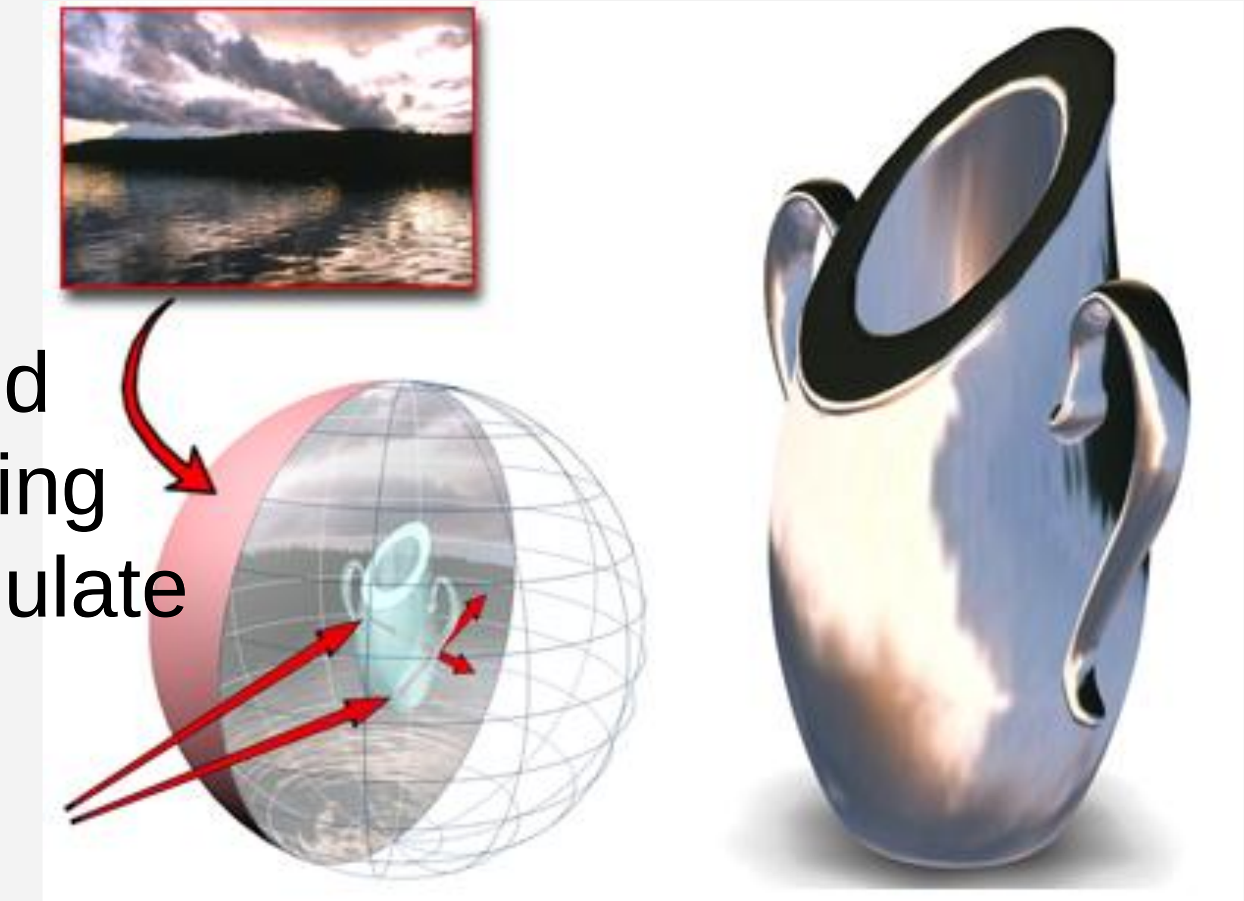


www.tomas-studio.com

Cheating on physics

- Environment mapping

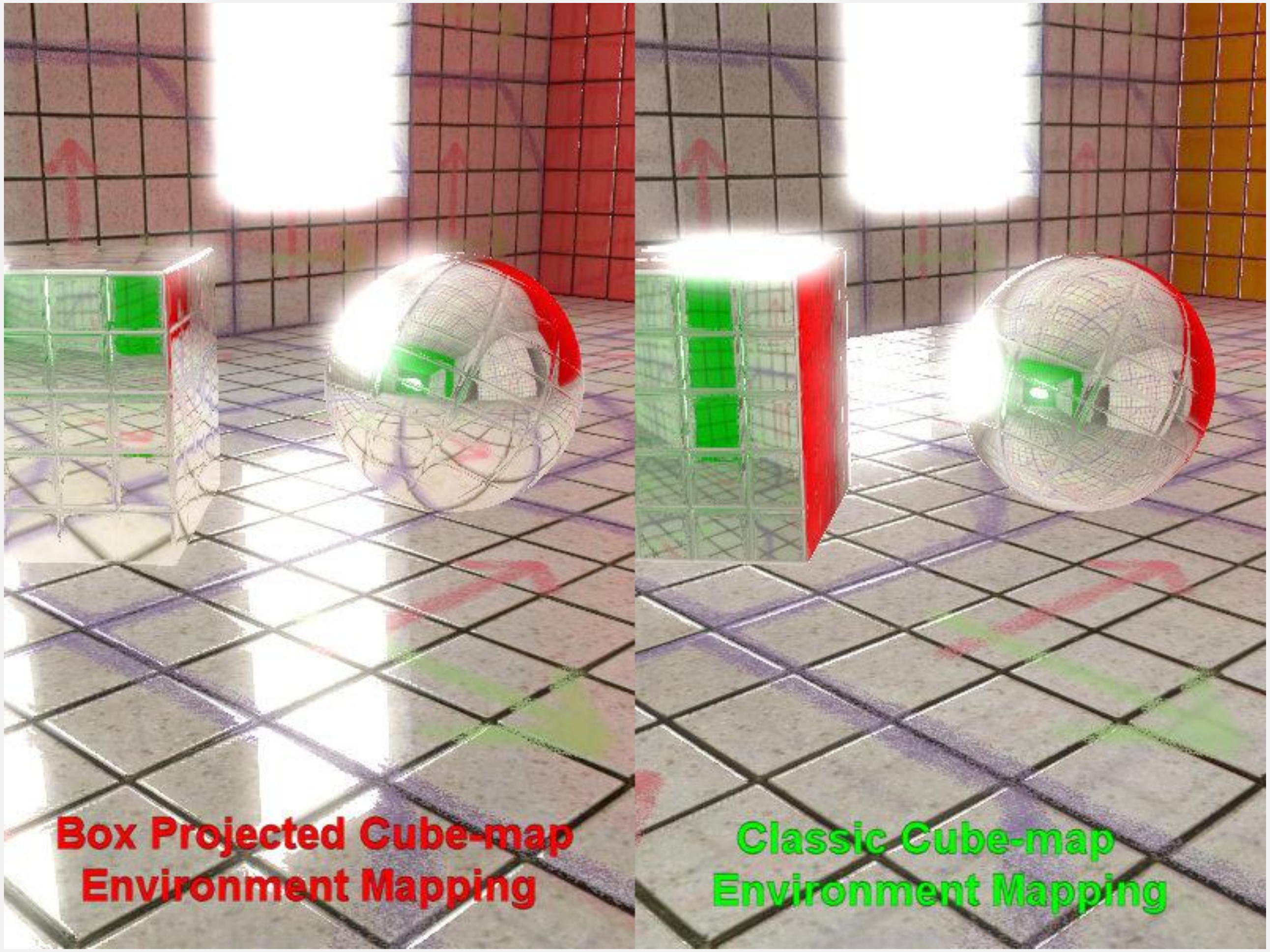
- texture applied to a surrounding sphere to simulate world reflections



www.autodesk.com

Environment mapping demo

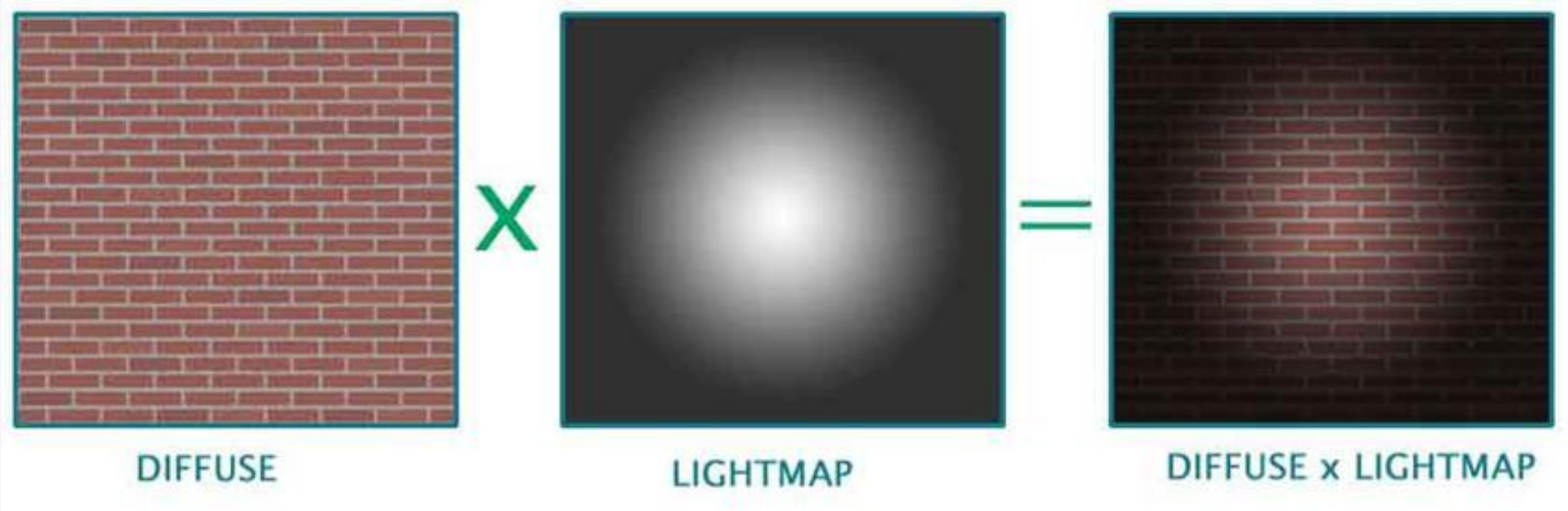
http://www.gamedev.net/topic/568829-box-projected-cubemap-environment-mapping/



Light maps

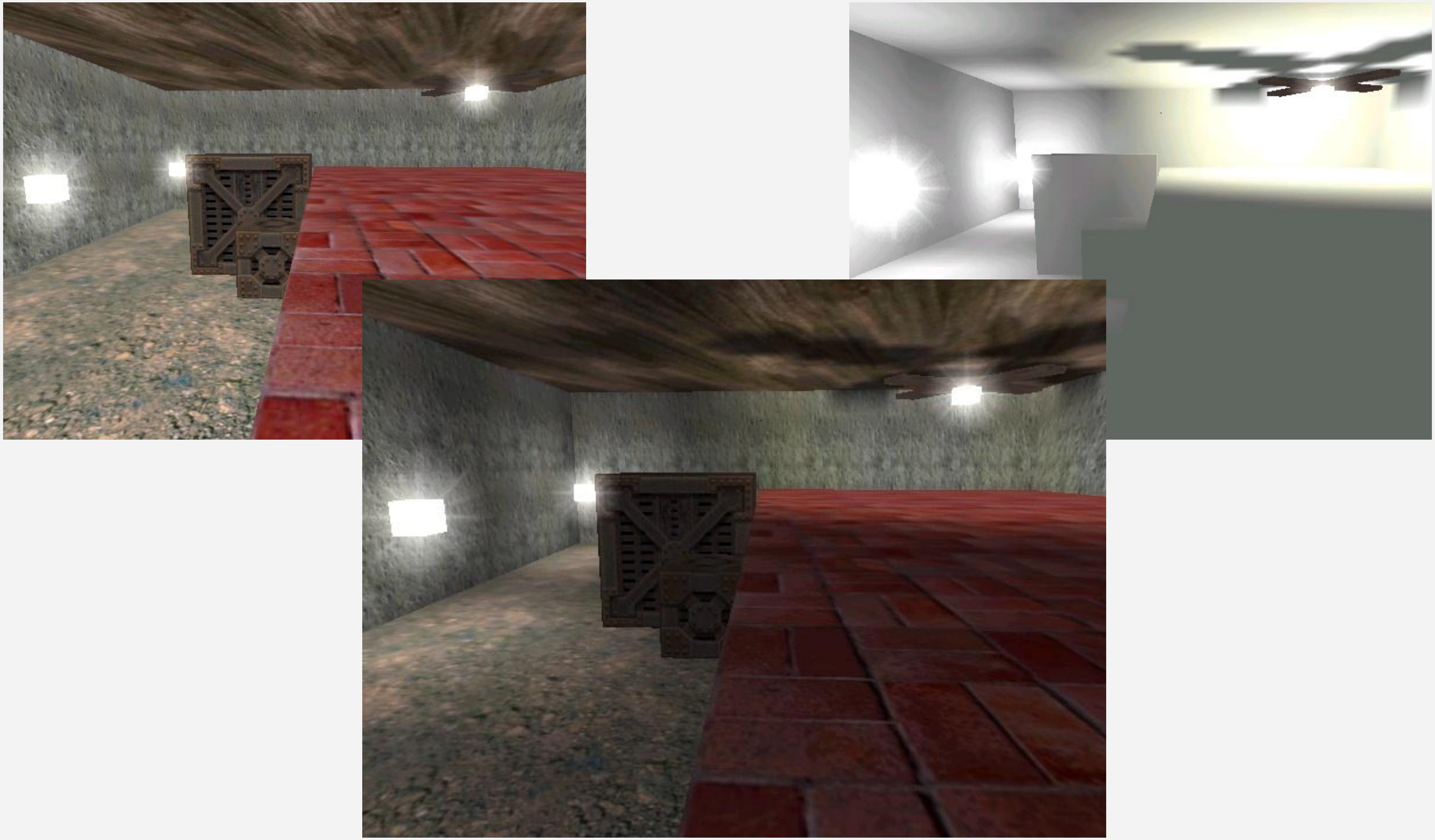
Light maps

- Pre-computed high-quality lighting
- Stored into special texture (light map)
- Light map combined with the texture
- Texture baking (permanent)



Keshav Channa: Light Mapping - Theory and Implementation

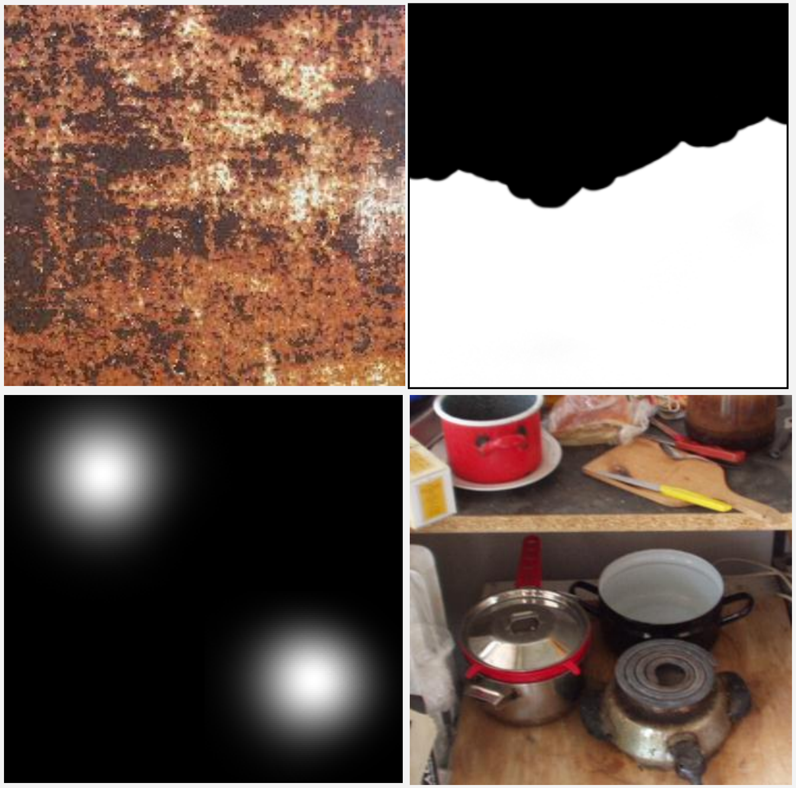
Light map example

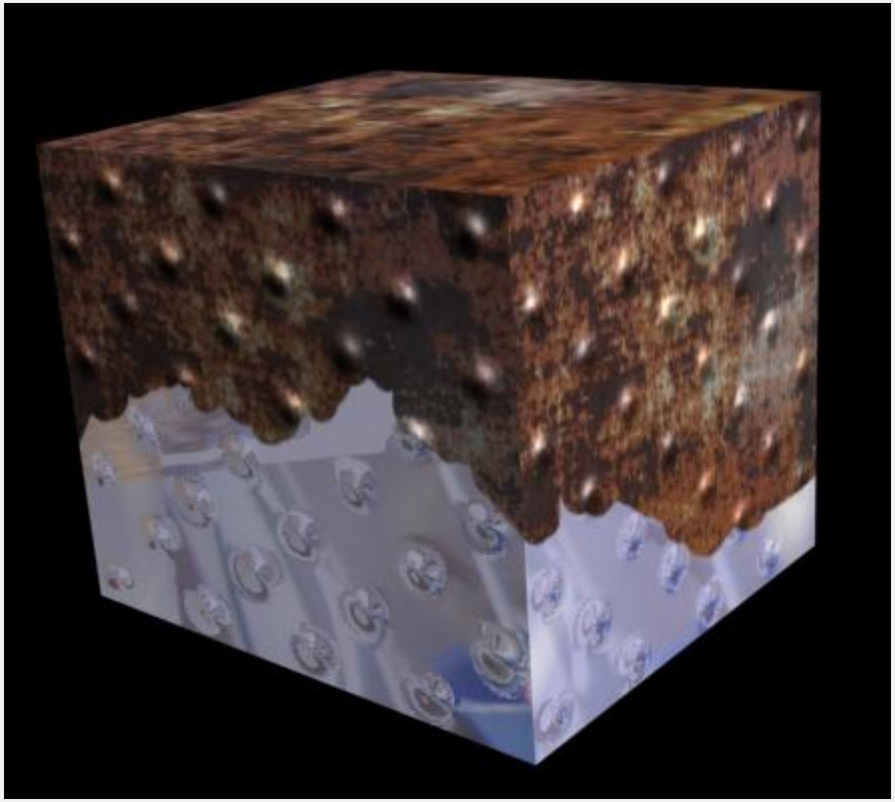


http://www.cs.bath.ac.uk/~pjw/NOTES/pics/lightmap.html

Multitexturing

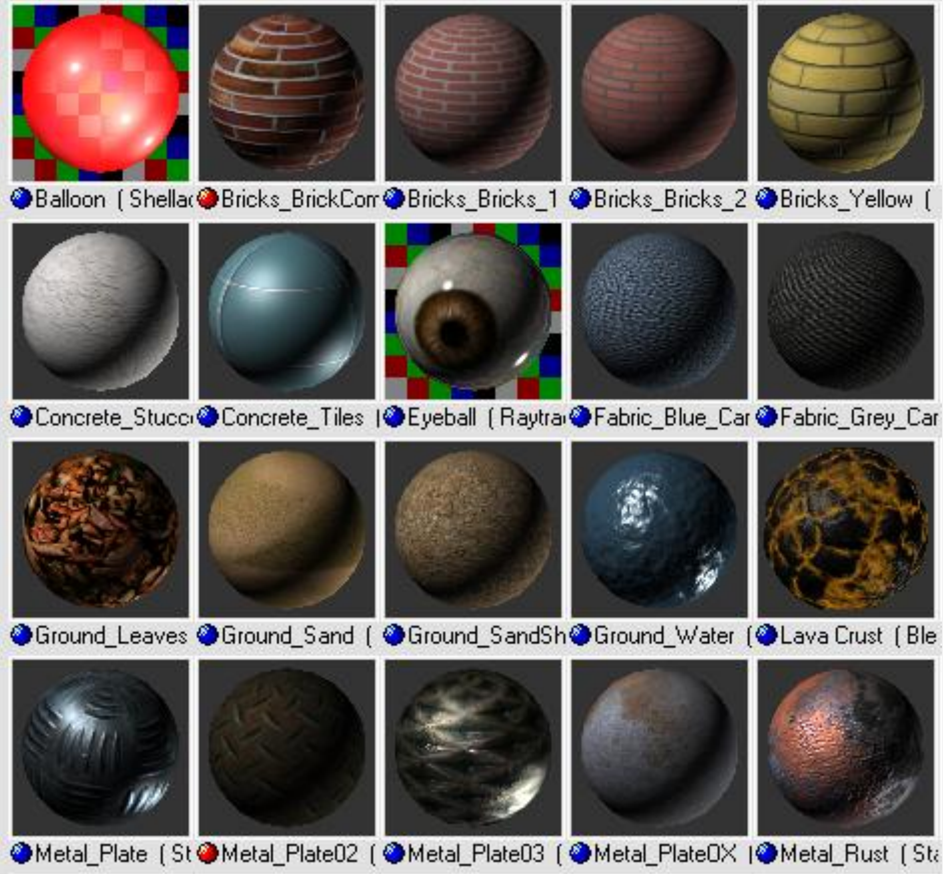
- Combine multiple textures

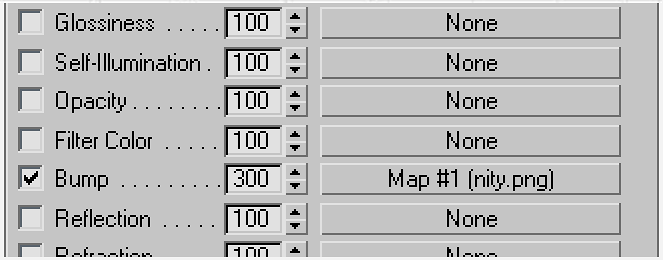




Material

- Textures
- Shaders
- Lighting parameters
  - depend on light model
  - e.g. Phong:
    - ambient
    - diffuse
    - specular, shininess
  - translucency, reflectivity, index of refraction ...



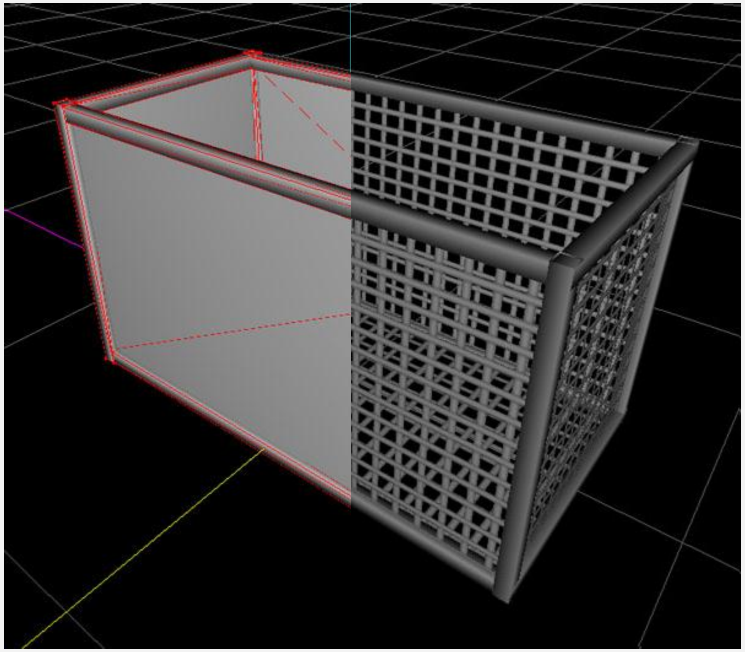

☐ Glossiness ..... 100 None
☐ Self-Illumination . 100 None
☐ Opacity .... 100 None
☐ Filter Color ..... 100 None
☑ Bump .... 300 Map #1 (nity.png)
☐ Reflection ..... 100 None
☐ Refraction 100 None

Texture representation

- width × height array of pixels
- pixel formats:
  - indexed color (8bit), grayscale (8bit)
  - 16bit, 24bit (RGB),32bit (RGBA)
- Color planes (24bit = 3x8bit, 32bit = 4x8bit)
- Alpha – opacity value
  - 0 = transparent (also 0.0 or 0%)
  - 255 = fully opaque (also 1.0 or 100%)

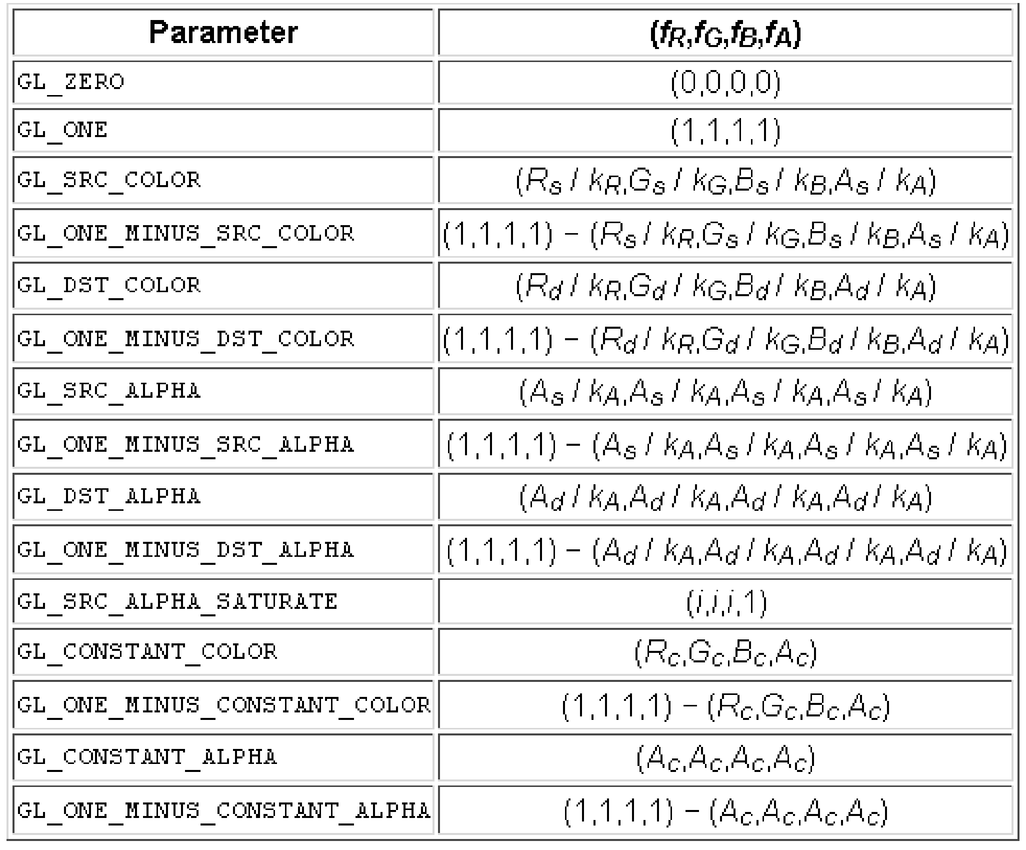
Blending, alpha channel

- Rendering translucent objects does not overwrite old pixels by new pixels
- But combines new pixels with old pixels

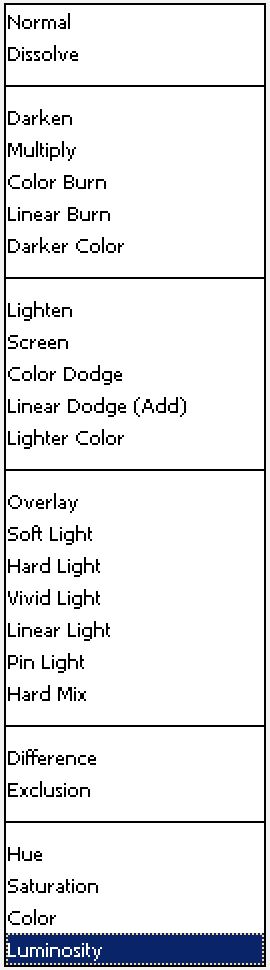


Different blending modes

- Res = Src * f_{src} + Dest * f_{src} (RGBA separate)
- e.g. Src*$Alpha_{src}$ + Dest*(1-$Alpha_{src}$)
- e.g. OpenGL or Photoshop



Parameter	(f_R,f_G,f_B,f_A)
GL_ZERO	$(0,0,0,0)$
GL_ONE	$(1,1,1,1)$
GL_SRC_COLOR	$(R_s / k_R, G_s / k_G, B_s / k_B, A_s / k_A)$
GL_ONE_MINUS_SRC_COLOR	$(1,1,1,1) - (R_s / k_R, G_s / k_G, B_s / k_B, A_s / k_A)$
GL_DST_COLOR	$(R_d / k_R, G_d / k_G, B_d / k_B, A_d / k_A)$
GL_ONE_MINUS_DST_COLOR	$(1,1,1,1) - (R_d / k_R, G_d / k_G, B_d / k_B, A_d / k_A)$
GL_SRC_ALPHA	$(A_s / k_A, A_s / k_A, A_s / k_A, A_s / k_A)$
GL_ONE_MINUS_SRC_ALPHA	$(1,1,1,1) - (A_s / k_A, A_s / k_A, A_s / k_A, A_s / k_A)$
GL_DST_ALPHA	$(A_d / k_A, A_d / k_A, A_d / k_A, A_d / k_A)$
GL_ONE_MINUS_DST_ALPHA	$(1,1,1,1) - (A_d / k_A, A_d / k_A, A_d / k_A, A_d / k_A)$
GL_SRC_ALPHA_SATURATE	$(i,i,i,1)$
GL_CONSTANT_COLOR	(R_c,G_c,B_c,A_c)
GL_ONE_MINUS_CONSTANT_COLOR	$(1,1,1,1) - (R_c,G_c,B_c,A_c)$
GL_CONSTANT_ALPHA	(A_c,A_c,A_c,A_c)
GL_ONE_MINUS_CONSTANT_ALPHA	$(1,1,1,1) - (A_c,A_c,A_c,A_c)$

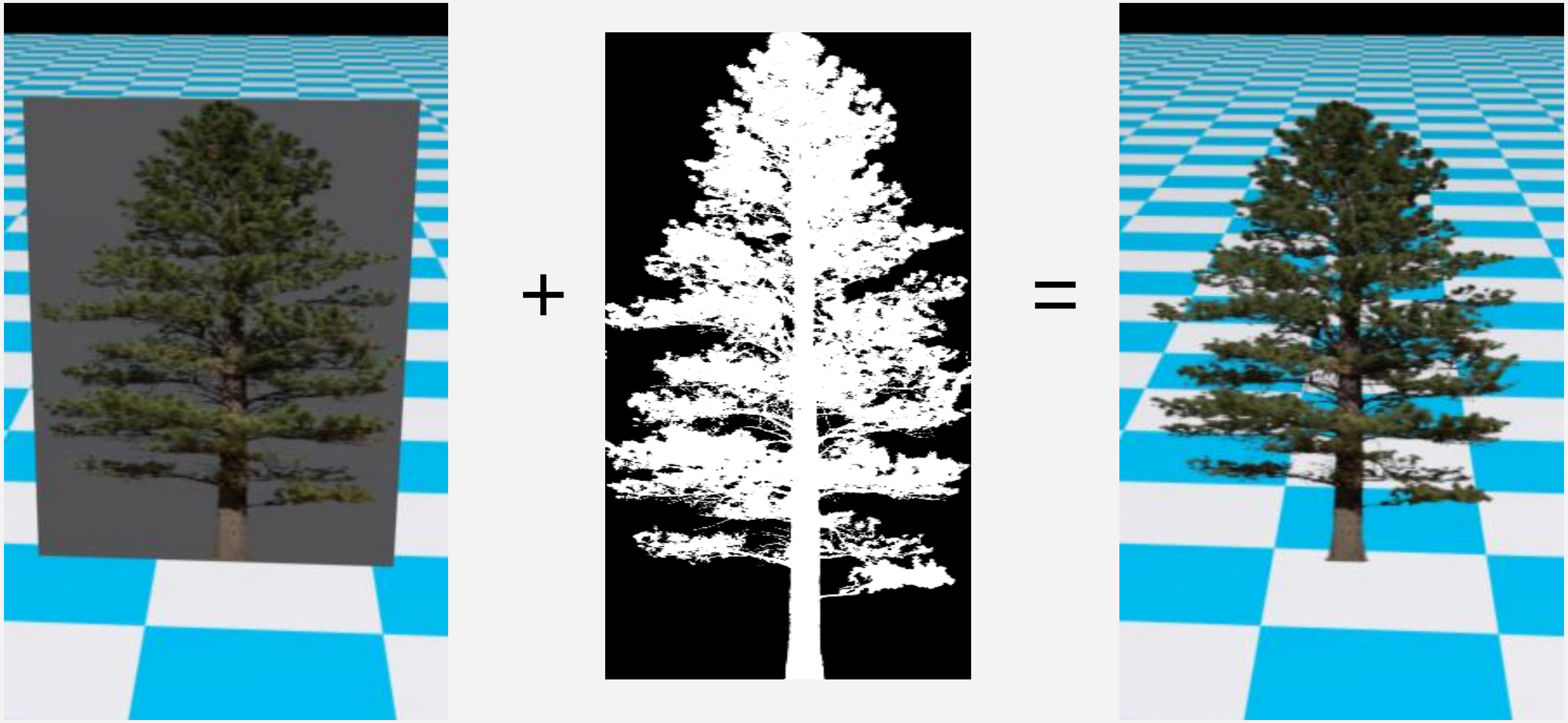


Color separation

- Color needs to be treated R,G,B piecewise
  - example:
    Color1 = 0xFF0000 = 16.711.680 ●
    Color2 = 0x0000FF = 255 ●
  - Color = 0.5*Color1+0.5*Color2 = ?
  - 0.5 * 16711680 + 0.5 * 255 = ●
  - Correct computation: ●
    Color[R] = (Color1[R] + Color2[R]) / 2
    Color[G] = (Color1[G] + Color2[G]) / 2
    Color[B] = (Color1[R] + Color2[B]) / 2

Transparent materials

- Glass, plastic
- Fake fine geometry details of vegetation, grids, wires, fences, …



Animation

Motion

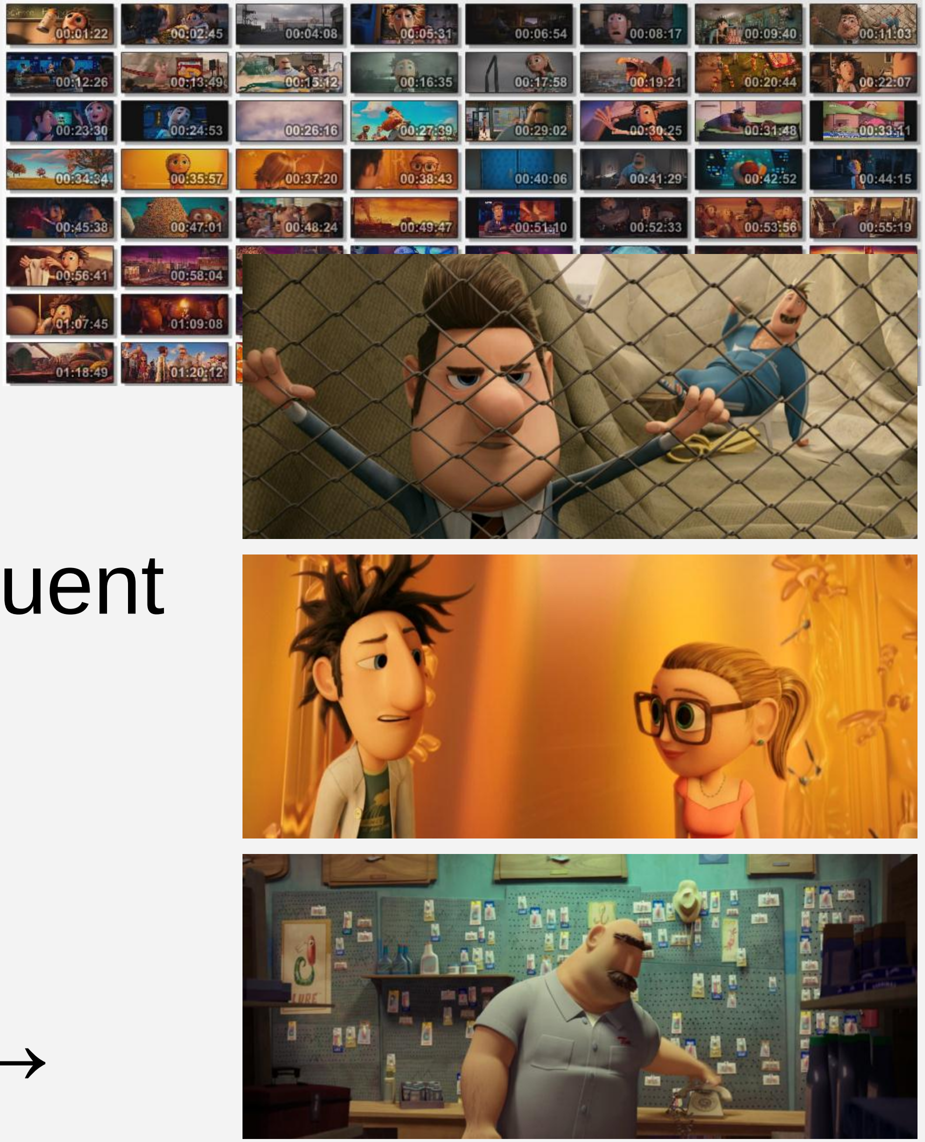
Eadweard Muybridge – The Horse in Motion (1878)

Time in computer graphics

- 3dimensional graphics = geometry
- 4th dimension = time
- Object attributes change over time
- Result = movie

Movie

- Sequence of frames
- Frame rate
- ~ 25fps and more is fluent
- ~10^5 frames / movie
- e.g. 129 311 frames →



Frame rates

- Frame rate for movies/TV
  - 24 (Cinema, Blu Ray)
  - 23.976, 29.97 (NTSC)
  - 25 (PAL)
- Frame rate for real time CG
  - 30+

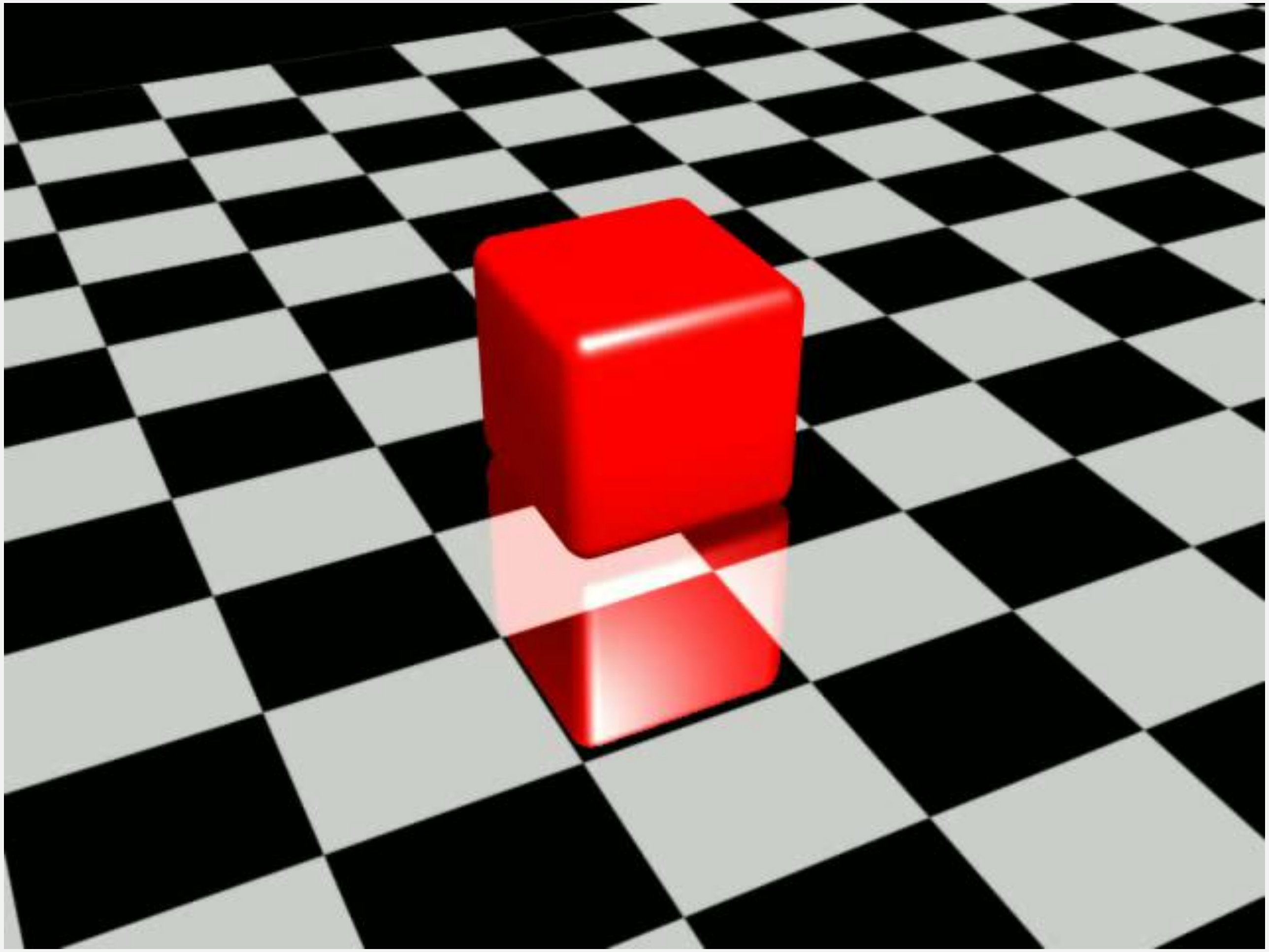
Computer animation

- Real-time
  - – Speed is priority
  - – Quality is second

- Offline
  - – Quality is priority
  - – Speed is second

What can be animated?

- Position
- Rotation
- Scale
- Geometry
- Texture
- Color
- Transp.



- … any numeric parameter

How to create animation

- Change values of parameters over time
- Manually
  - Values are set for each individual frame
- Procedurally
  - Values are computed by algorithm
- Keyframing
  - Important frames are manual, rest is parametric
- Motion capture
  - Real world motion is scanned to computer

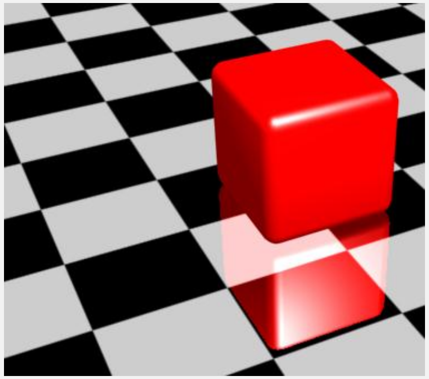
Manual animation

- Stop-motion animation
- e.g. Coraline, Wallace & Gromit, etc.

Using key frames

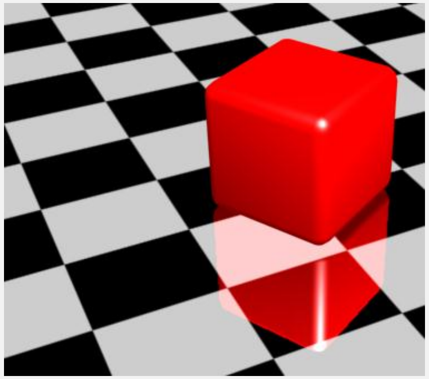
Animation key frames

- Manual setting of parameters not for all frames but only for some particular

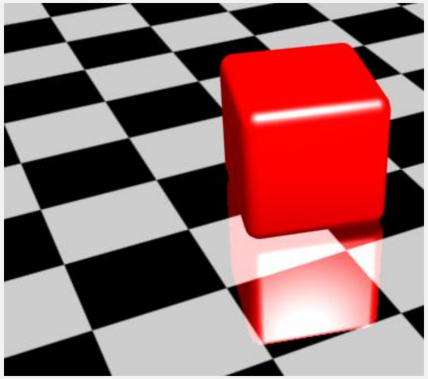


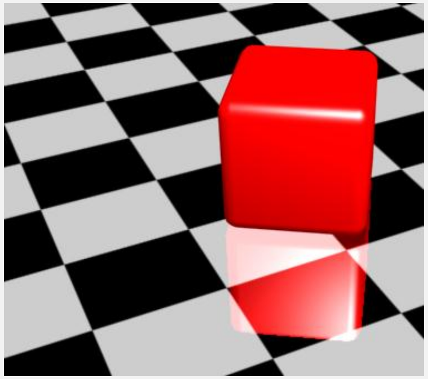
Rotation = 0°

. . . .



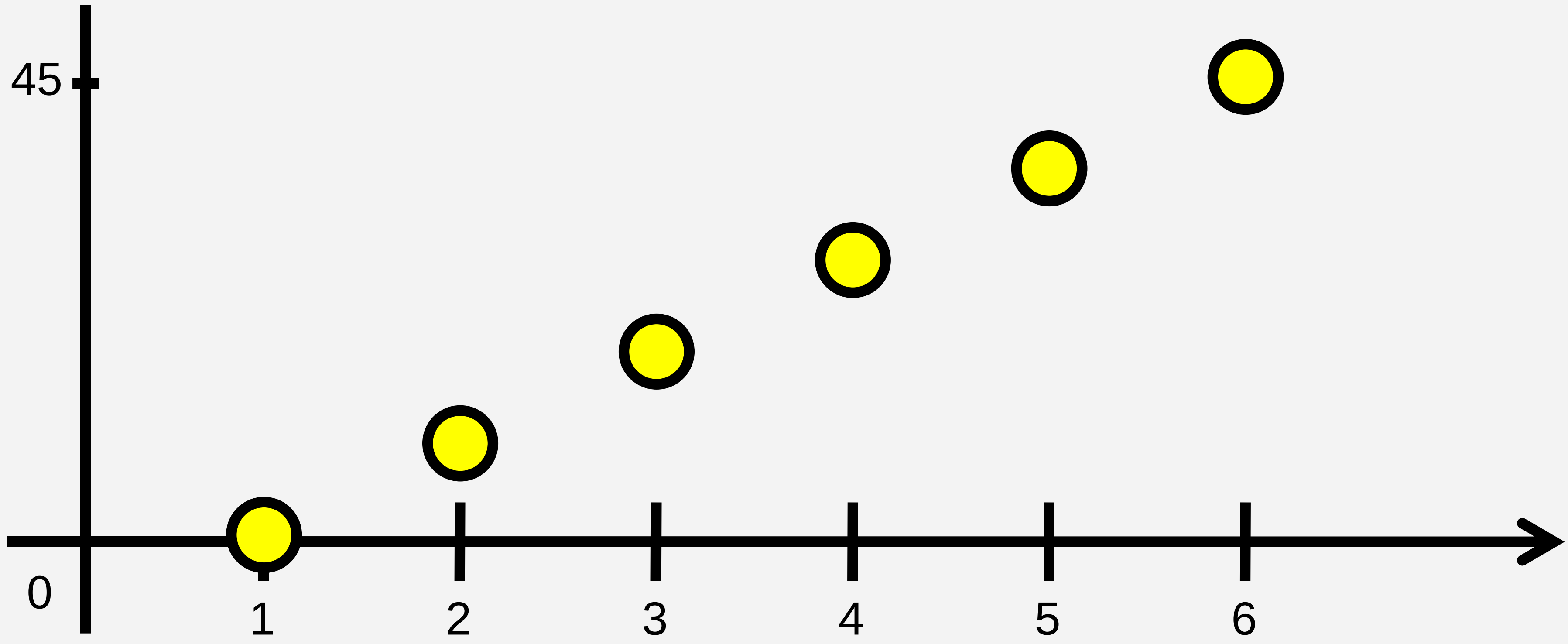
Rotation = 45°





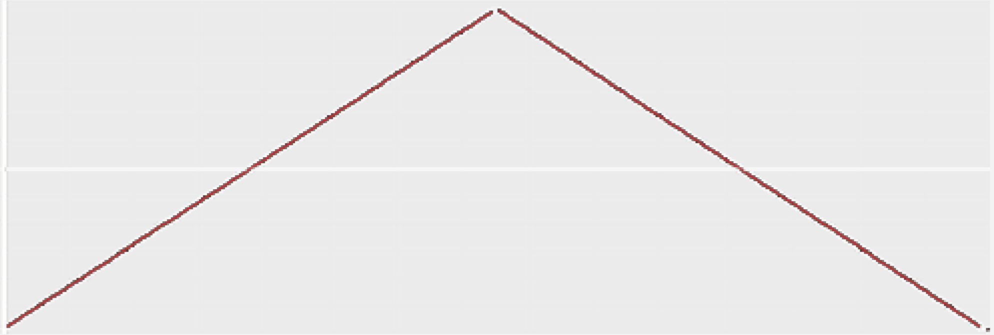
Inbetweening (“tweening”)

- Computing missing values based on existing surrounding values

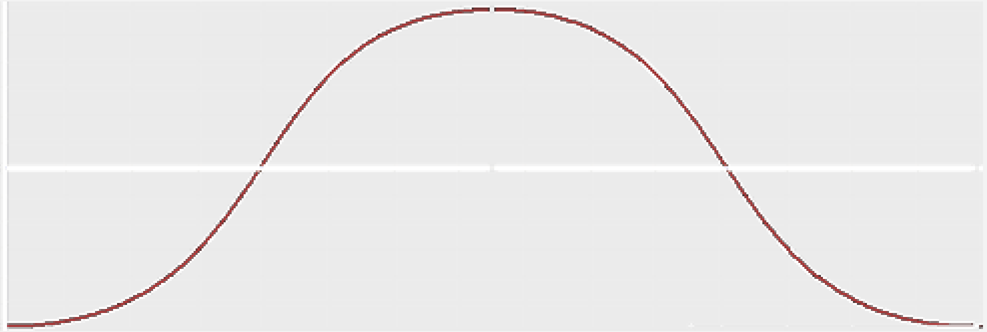


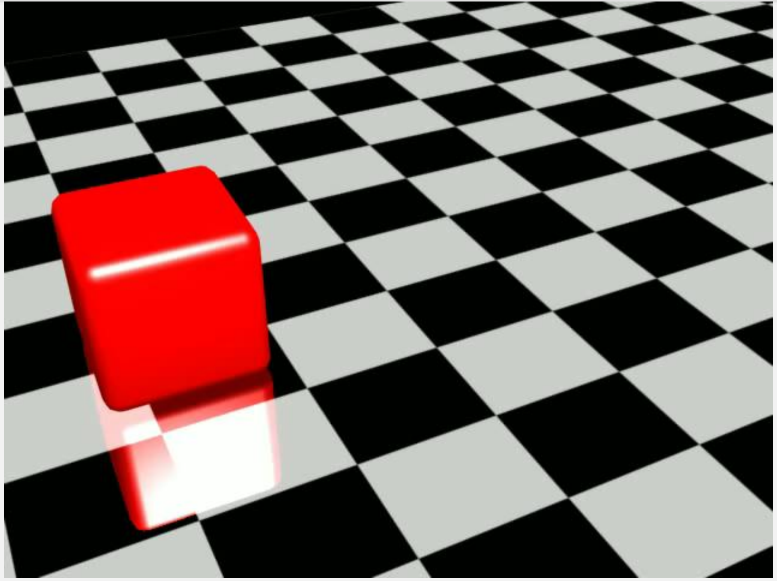
Tweening

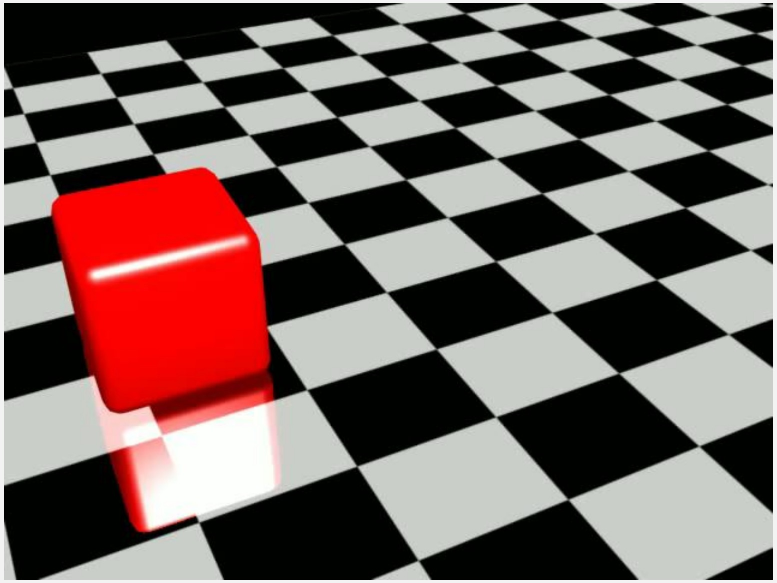
- Linear (constant)



- Ease-in, ease-out

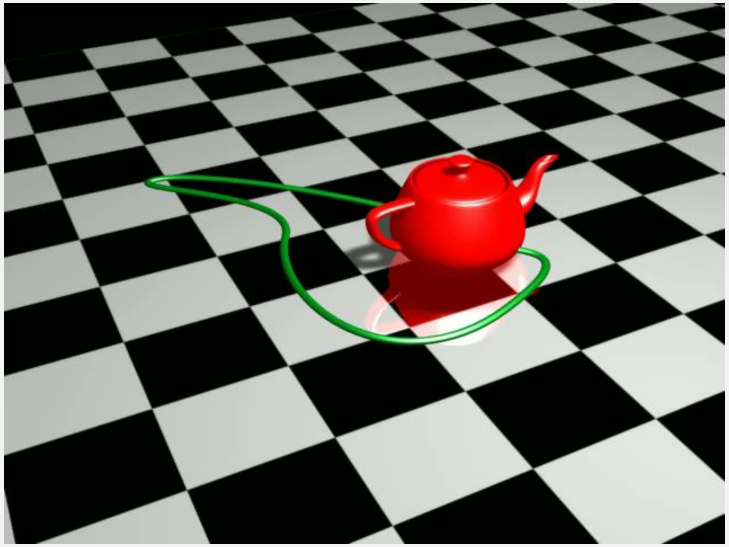


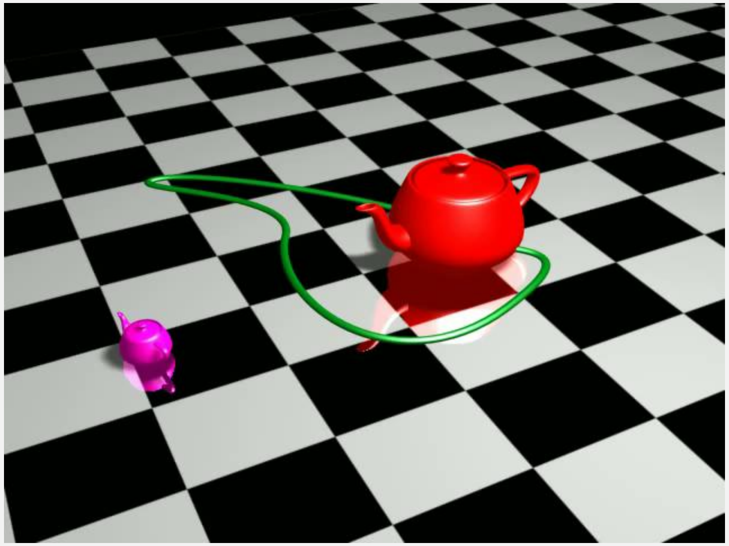


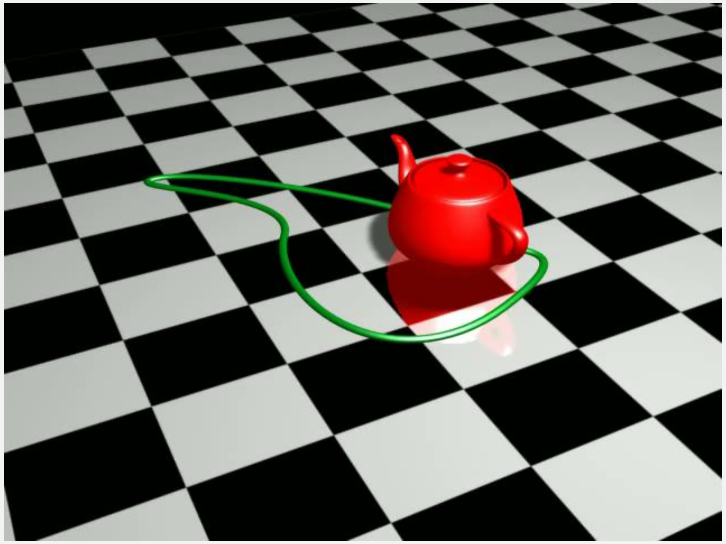


Simple controllers

- Position
  - – Follow path
- Rotation
  - – Follow path, Look at

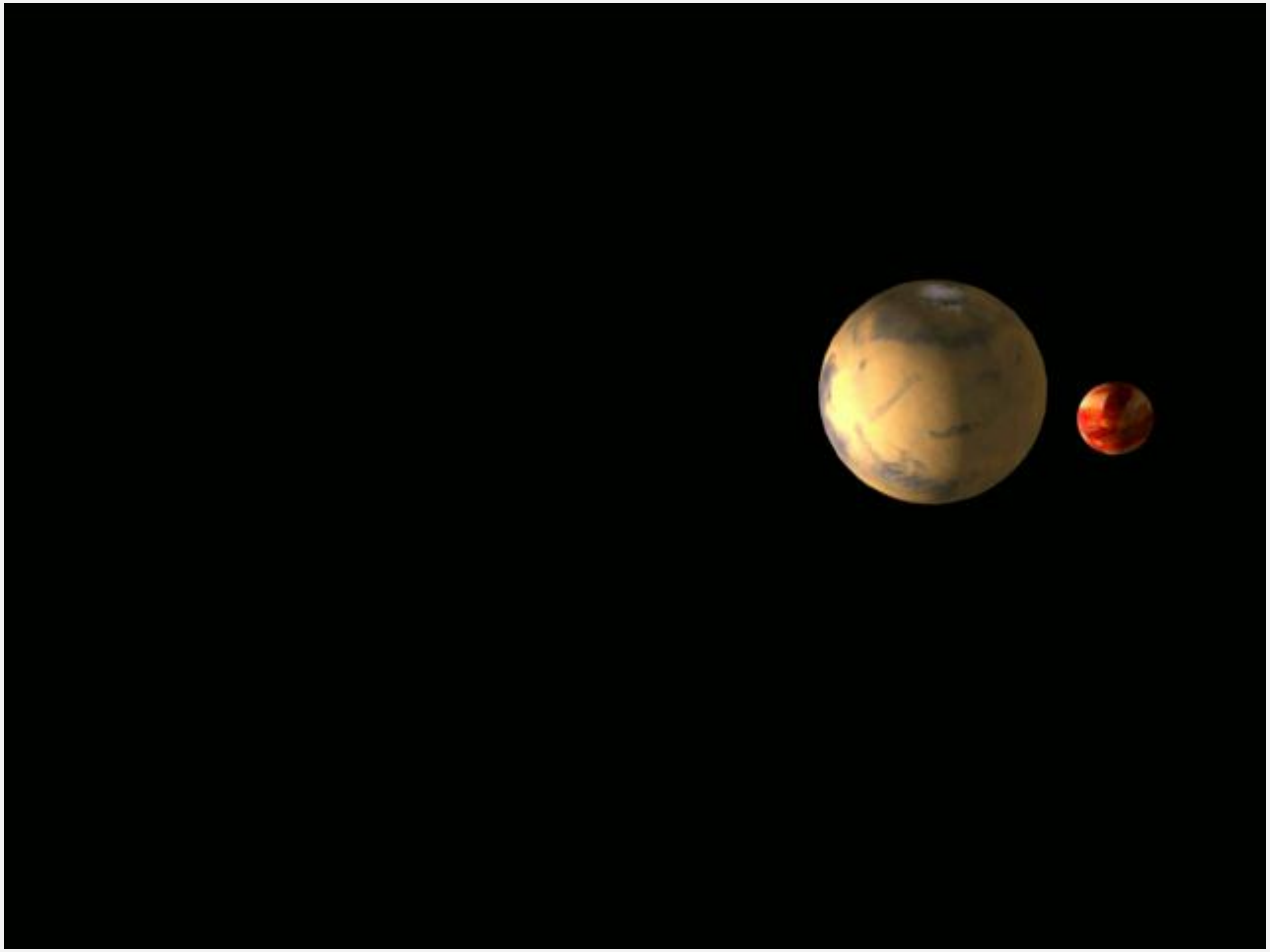






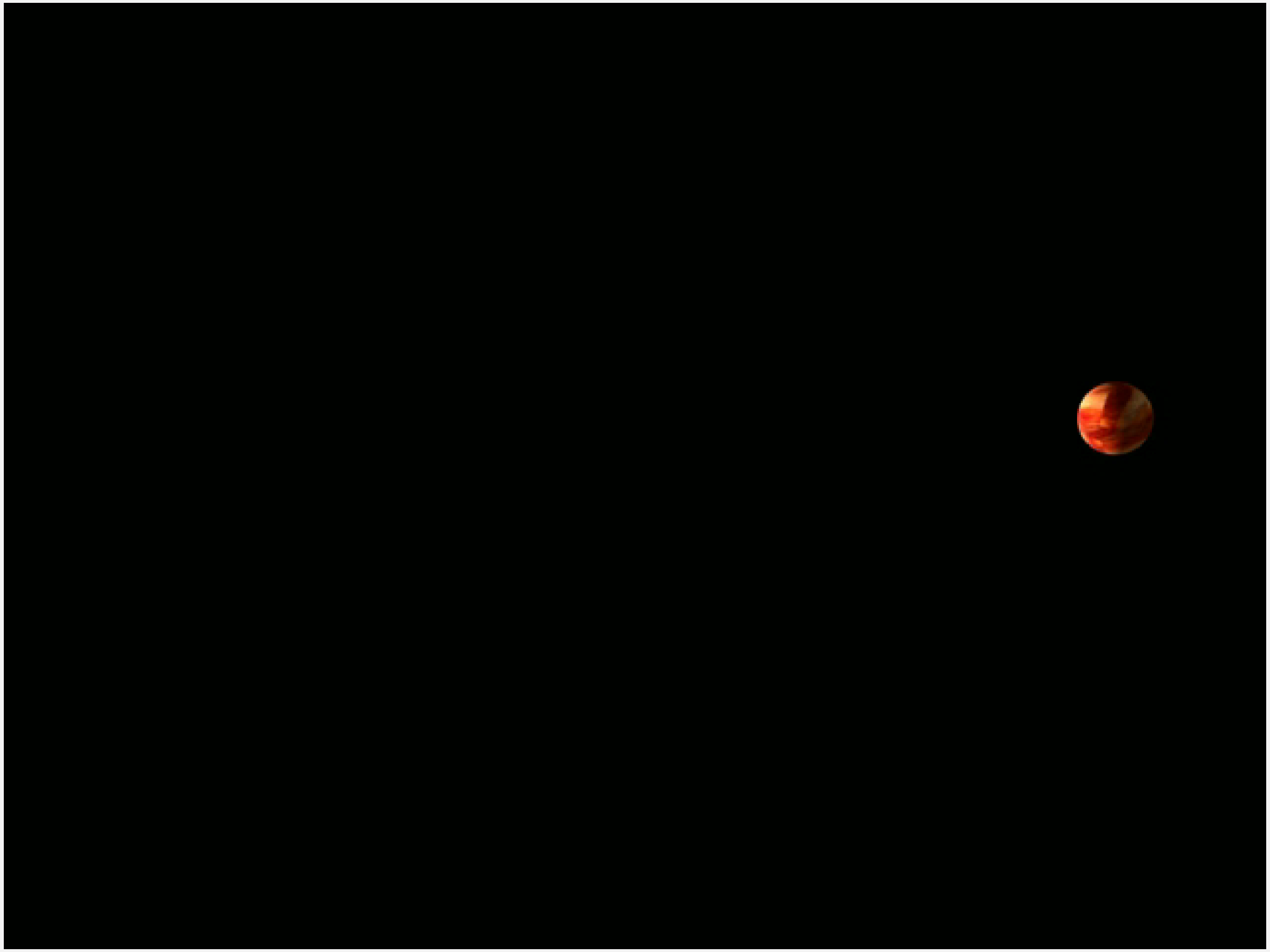
Animating complex objects

- Remember local coordinates



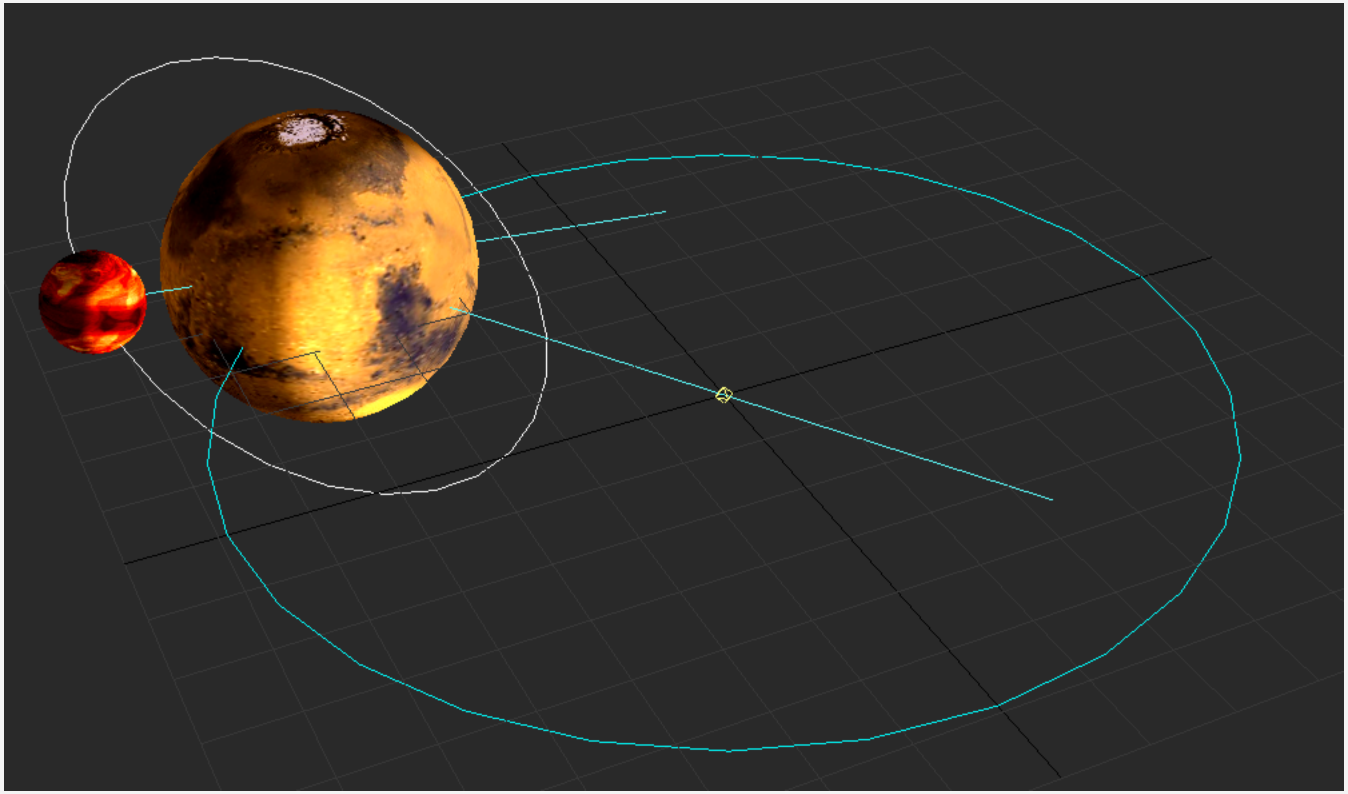
Animating complex objects

- Remember local coordinates



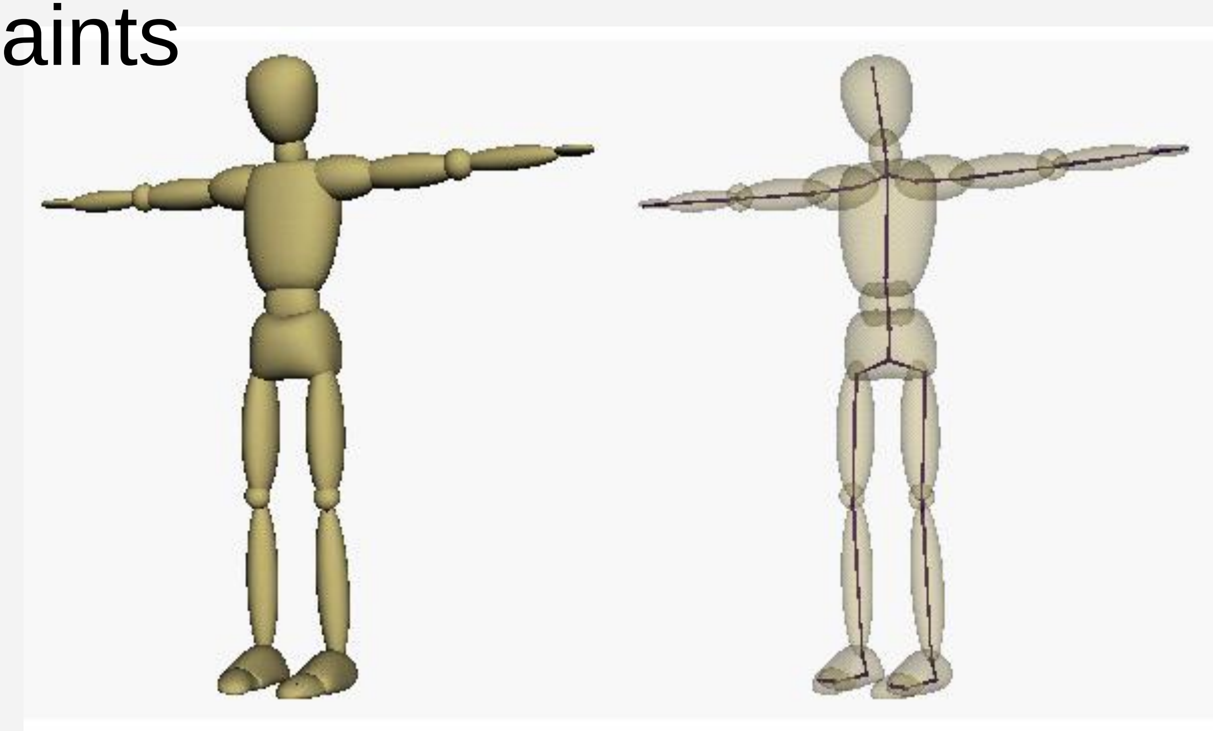
Animating complex objects

- Skeletons, chains, systems
  - Simulate physical constraints



Animating complex models

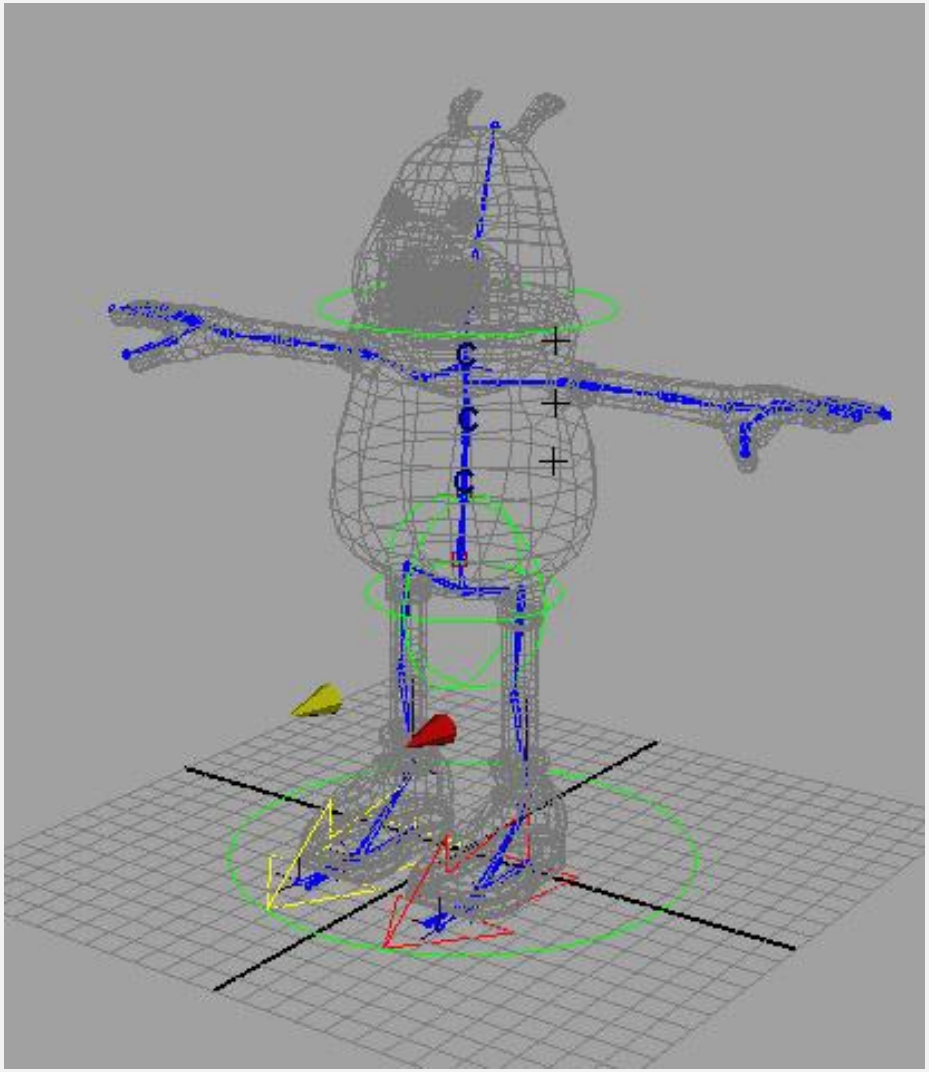
- Model/system decomposed into hierarchy
- Nodes, links, chains, joints, skeleton
- Motion constraints



http://caad.arch.ethz.ch/info/maya/manual/

Skeleton

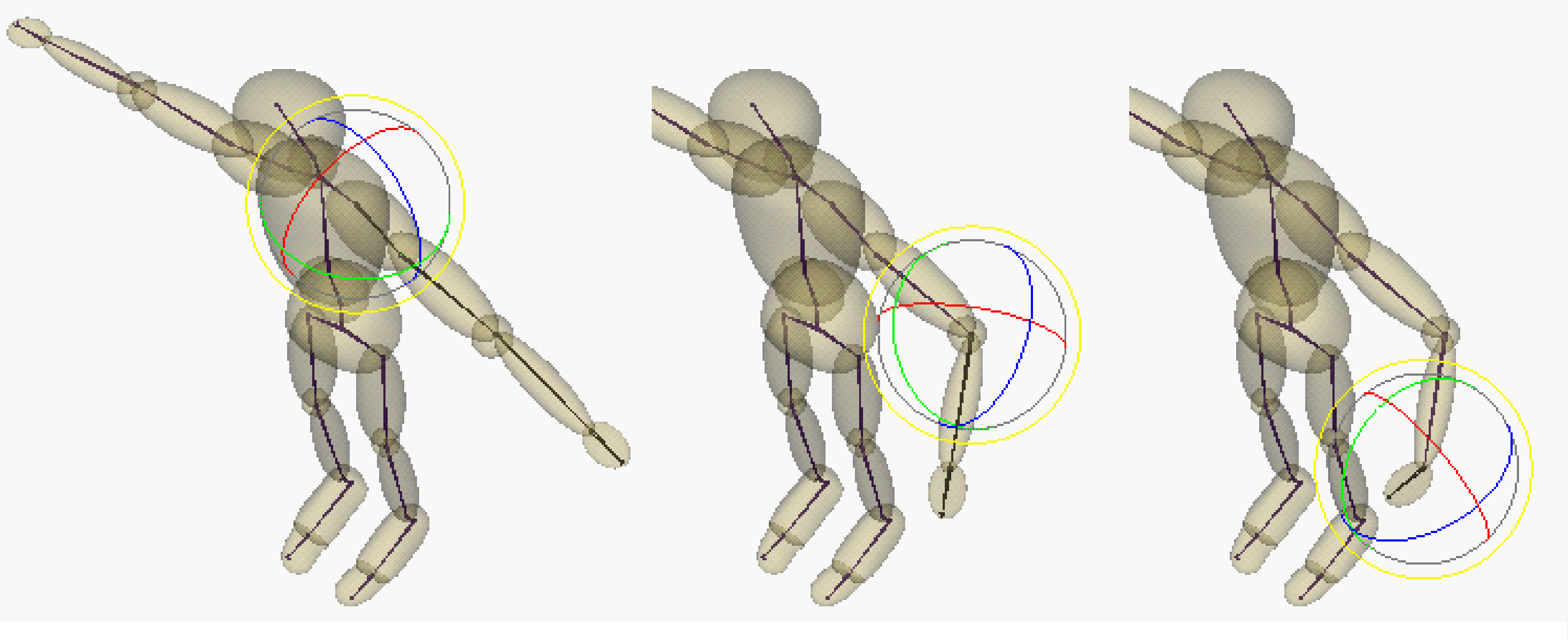
- Hierarchy
  - Bones
    - Rigid element
  - Joints
    - Rotation
    - Sliding
  - Springs
    - Change length
- Controllers
- It’s reusable!



http://en.9jcg.com/comm_pages/blog_content-art-16.htm

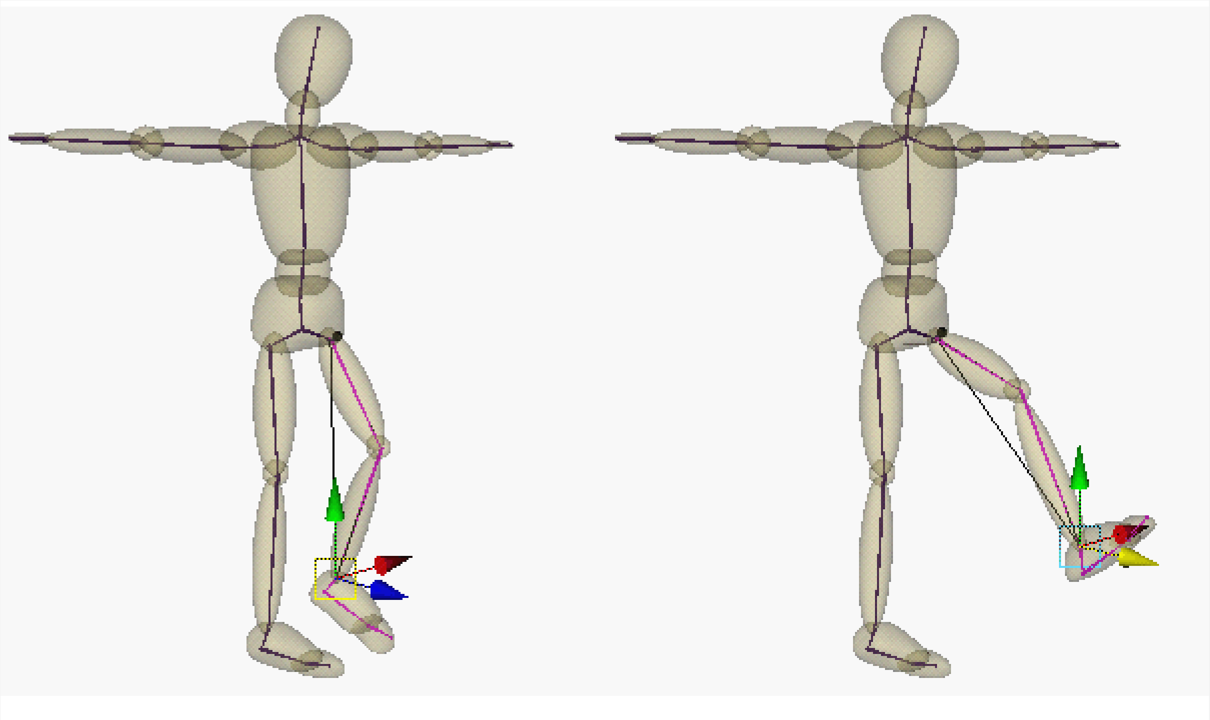
Forward kinematics

- Motion is initiated on top of the hierarchy and propagates downwards in the hierarchy



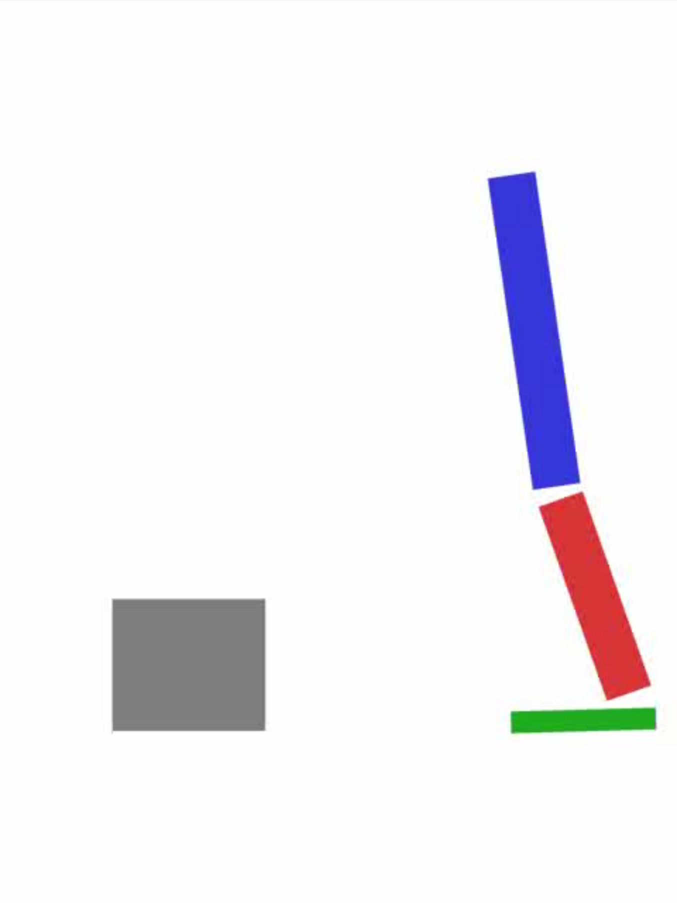
Inverse kinematics

- Motion is initiated on the bottom of the hierarchy and propagates upwards
- Motion constraints need to be set

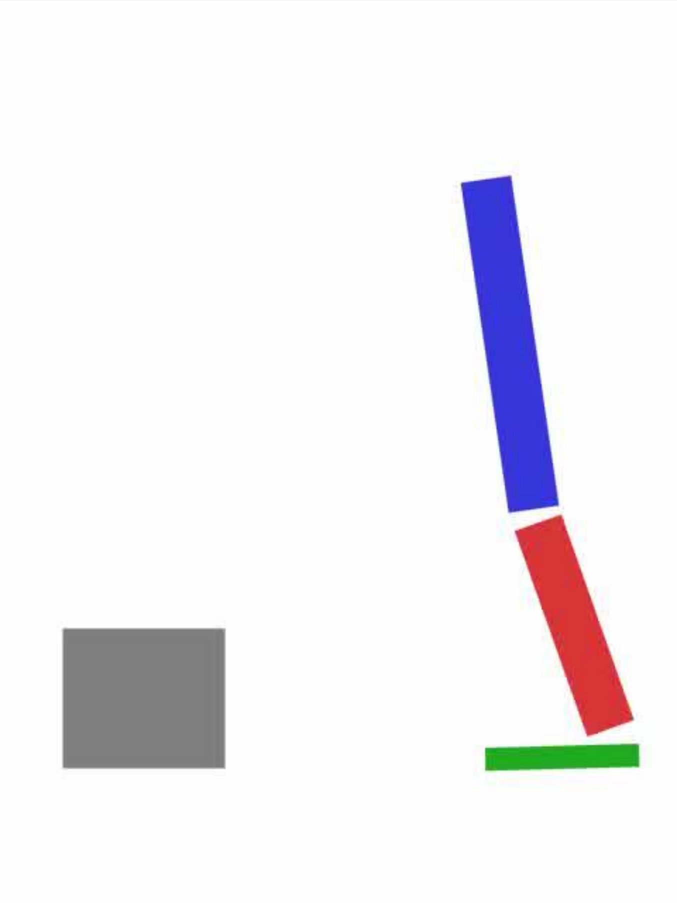


Two types of kinematics

Forward

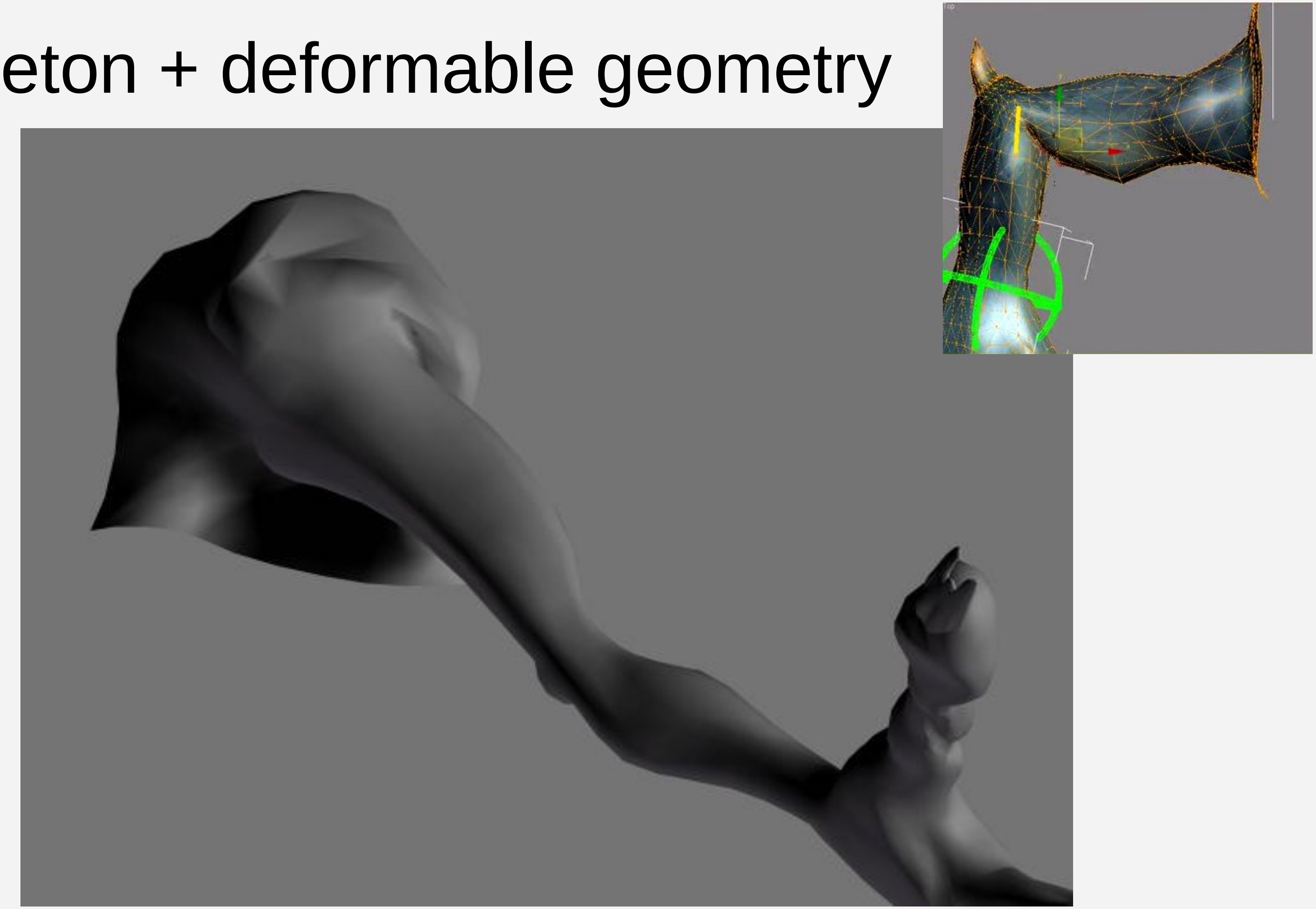


Inverse



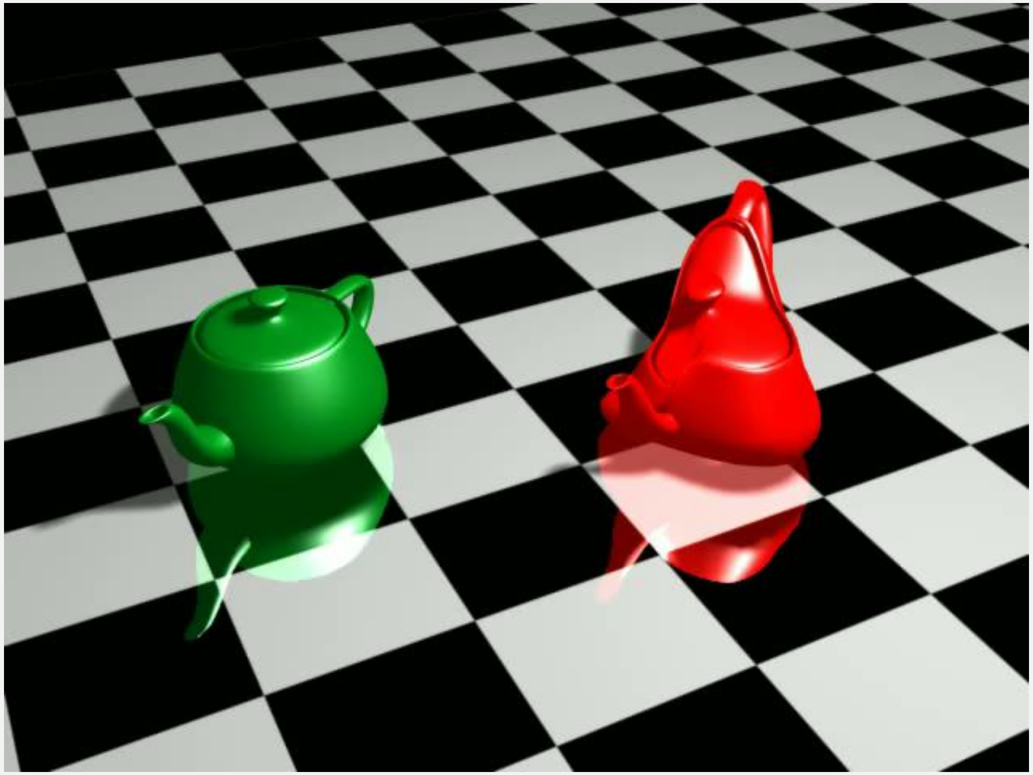
Skinning

- Skeleton + deformable geometry



Morphing

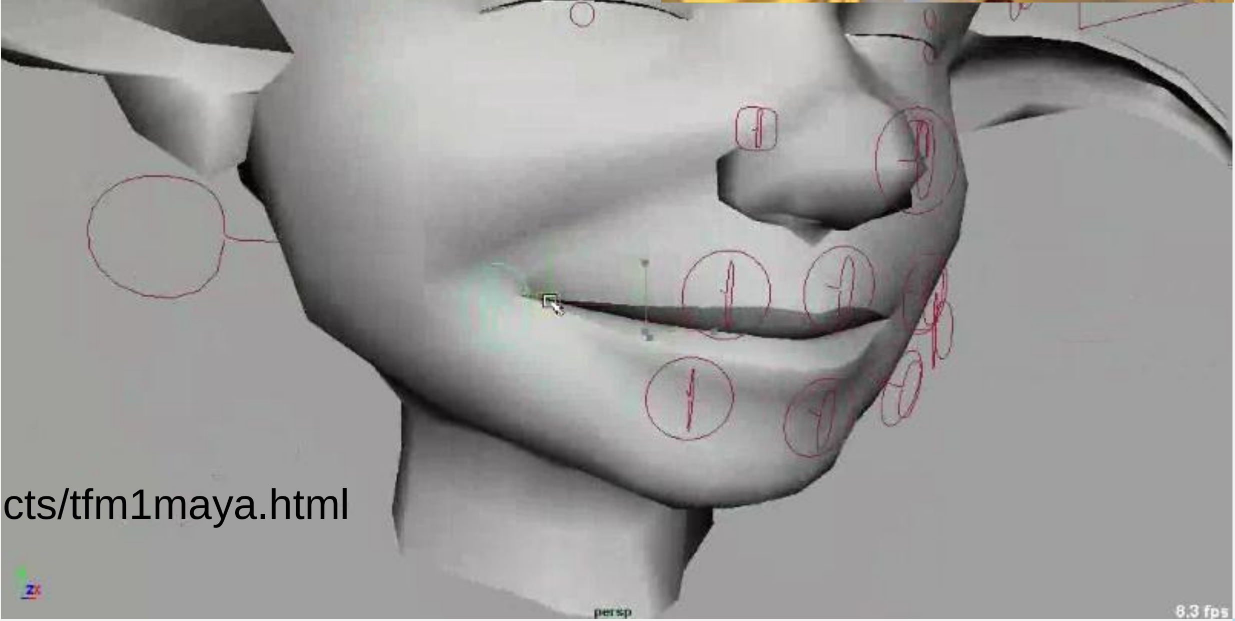
- Tweening deformations of the same model



- In simple cases works for different models too

Facial animation

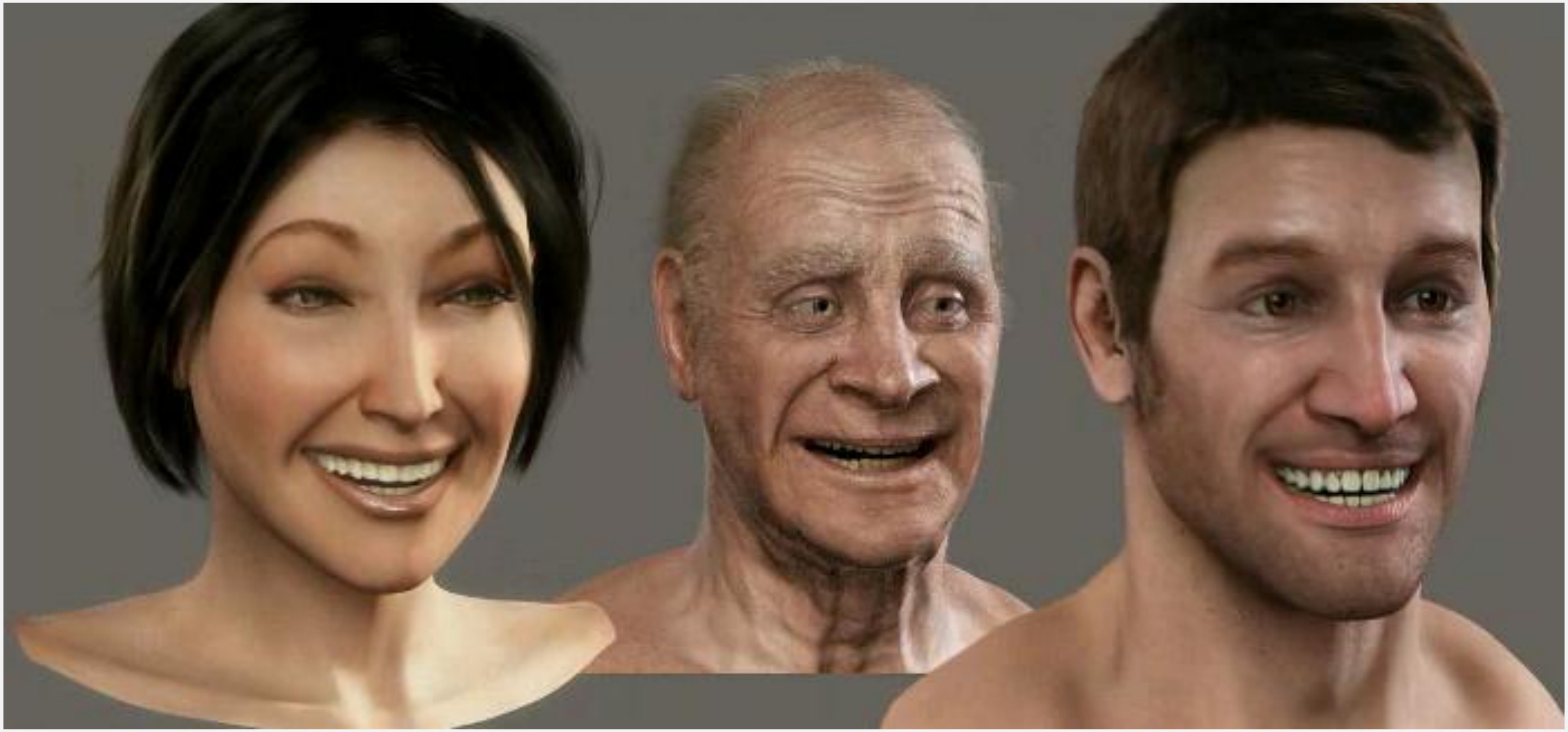
- Facial expressions
- Lips to speech synchronization
- Controllers
- Skinning
- Morphing



http://www.anzovin.com/products/tfm1maya.html

Reusable animation

- One skeleton – different models



http://www.studiopendulum.com/alterego/

Animation blending

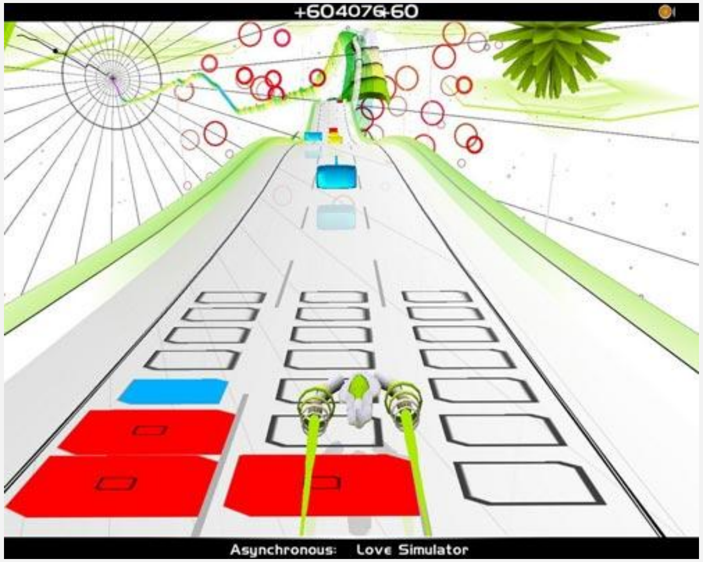
- Separate activities performed simultaneously
  - e.g. walking and shooting

- Smooth transitions between activities
  - e.g. standing up and walking

Procedural and physically-based animations

Procedural animation

- Programmed rules for changing parameters of the animated objects
- E.g. according to music, physics, psychology



CROWD SIMULATION (MASSIVE)
0:05:18

Physically based animation

- Rigid bodies
  - No geometry deformation
  - Collision response

- Soft bodies
  - Allow for deformation
  - Energy damping

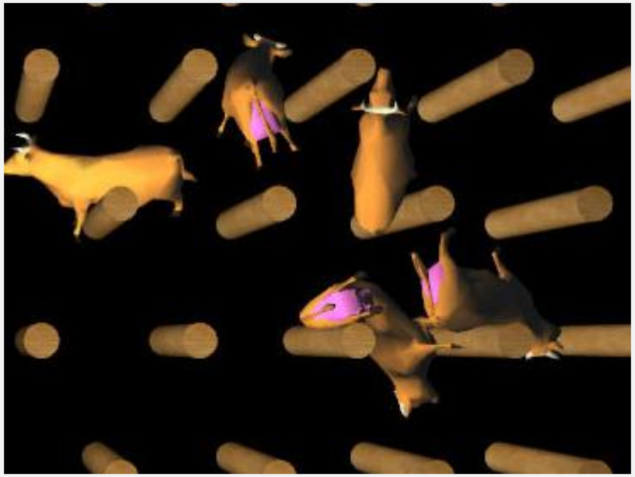
Animation construction

- Set body properties
  - Mass, elasticity, friction, …
- Set physical rules
  - Gravity, collisions, wind, …
- Set initial state
  - Position, velocity, direction, …
- Set constraints

- Run simulation / animation

Examples

- www.realmatter.com
- www.realflow.com
- www.massivesoftware.com
- www.audio-surf.com
- www.lagoatechnologies.com



Motion capture

Real world action captured

- Markers on actor’s body
- Optical / magnetic sensors
- 3D reconstruction of markers’ position
- Motion mapping to virtual character