3D Graphics & VR (Virtual Reality)

- ★ Synthetic content representation (Prof. Francisco Morán)
  - ✸ Introduction
    - Spatial transforms
    - Historical perspective of viewing devices
  - ✸ Modelling
    - Description of each 3D object: shape & appearance [& animation ⇒ 4D]
    - Id. of [interactive] 3D scene with different objects, lights, cameras…
    - Very time-consuming task for humans
  - ✸ Rendering
    - 2D image/video generation from 3D/4D scene model, usually seeking photo-realism
    - Approximation of physical laws through lighting models
    - Very time-consuming task for computers
- ★ 3D scanning and applications (Prof. Luis Salgado)

Contents (1/2)

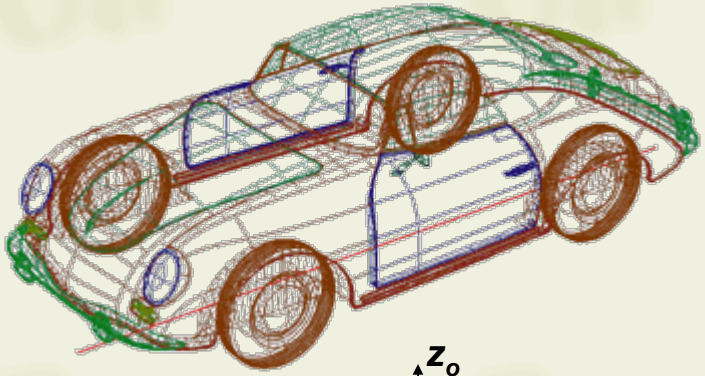
- ★ Introduction
  - ✸ Spatial transforms
    - ● Types of coordinates: MC/WC/DC, affine/homogeneous, etc.
    - ● Rendering pipeline
    - ● Projection zoo
  - ✸ Historical perspective of viewing devices
- ★ Modelling
  - ✸ Static objects: description of both shape and appearance
    - ● Shape: (“truly 3D”) solids/volumes vs. (“2,5D”) surfaces
    - ● Appearance: colours, normals, textures
    - ● Hybrid techniques: IB[M]R (Image-Based [Modelling and] Rendering)
  - ✸ Dynamic objects: description of their animation
    - ● Interpolation based on key frames
    - ● BBA (Bone-Based Animation), IK (Inverse Kinematics)

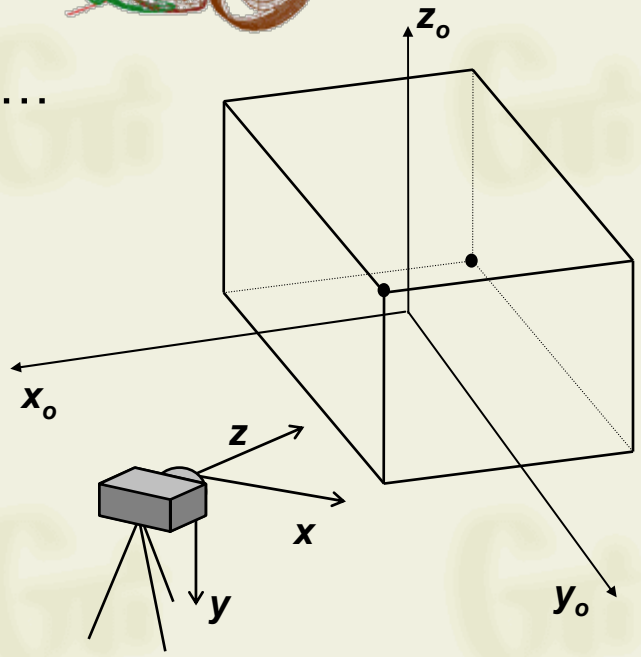
Contents (2/2)

- ★ Modelling (cont'd)
  - ✸ [Interactive] scenes
    - ● Spatiotemporal object placement: scene graph
    - ● Lights, camera[, interaction]!
  - ✸ Standards
    - ● VRML97 & X3D
    - ● MPEG-4
    - ● COLLADA
- ★ Rendering
  - ✸ 2D techniques
  - ✸ Hidden surface removal
  - ✸ Illumination and shading (colour interpolation) models
  - ✸ "Luxury": transparencies, shadows, reflections… vs. NPR
  - ✸ Image processing techniques: anti-aliasing filtering, etc.

Types of coordinates (1/2)

- ★ 3D (high precision)
  - ✸ Hierarchical object/scene modelling
    - ⇒ Object/**scene graph** concept
  - ✸ **Many local** reference frames…
    - ⇒ MCs (Modelling Coordinates)
  - ✸ … vs. just **one global** reference frame…
    - ⇒ WCs (World Coordinates)
  - ✸ … optionally **tied to the camera**
    - ⇒ CCs (Camera Coordinates)
- ★ 2D (low precision)
  - ✸ Image reference frame
    - ⇒ DCs (Device Coordinates)





Types of coordinates (2/2)

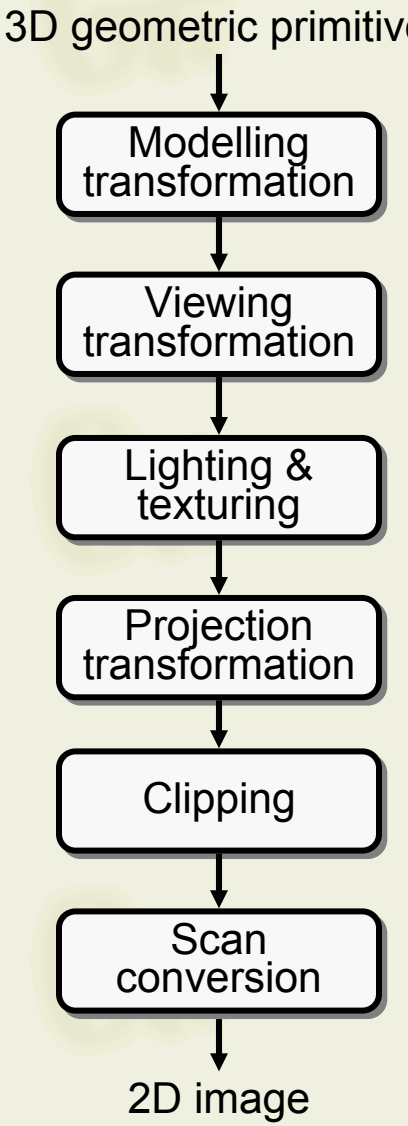
- Explicit vs. implicit "vs." parametric
  - Ex: circumference "$x^2 + y^2 = 1$" vs. "$x = \cos\theta$, $y = \sin\theta$, $\theta \in [0, 2\pi[$"

- Affine vs. homogeneous
  - Homogeneous coordinate w for 4th dimension (projective space)
  - Translation may also be expressed through a matrix product

$$\begin{pmatrix} x' = x+t_x \\ y' = y+t_y \\ z' = z+t_z \\ w' = 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 1 & 0 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w=1 \end{pmatrix}$$

  - Transform concatenation = matrix product

Rendering pipeline



Transform into WCs

Transform into CCs
Done with modelling transformation

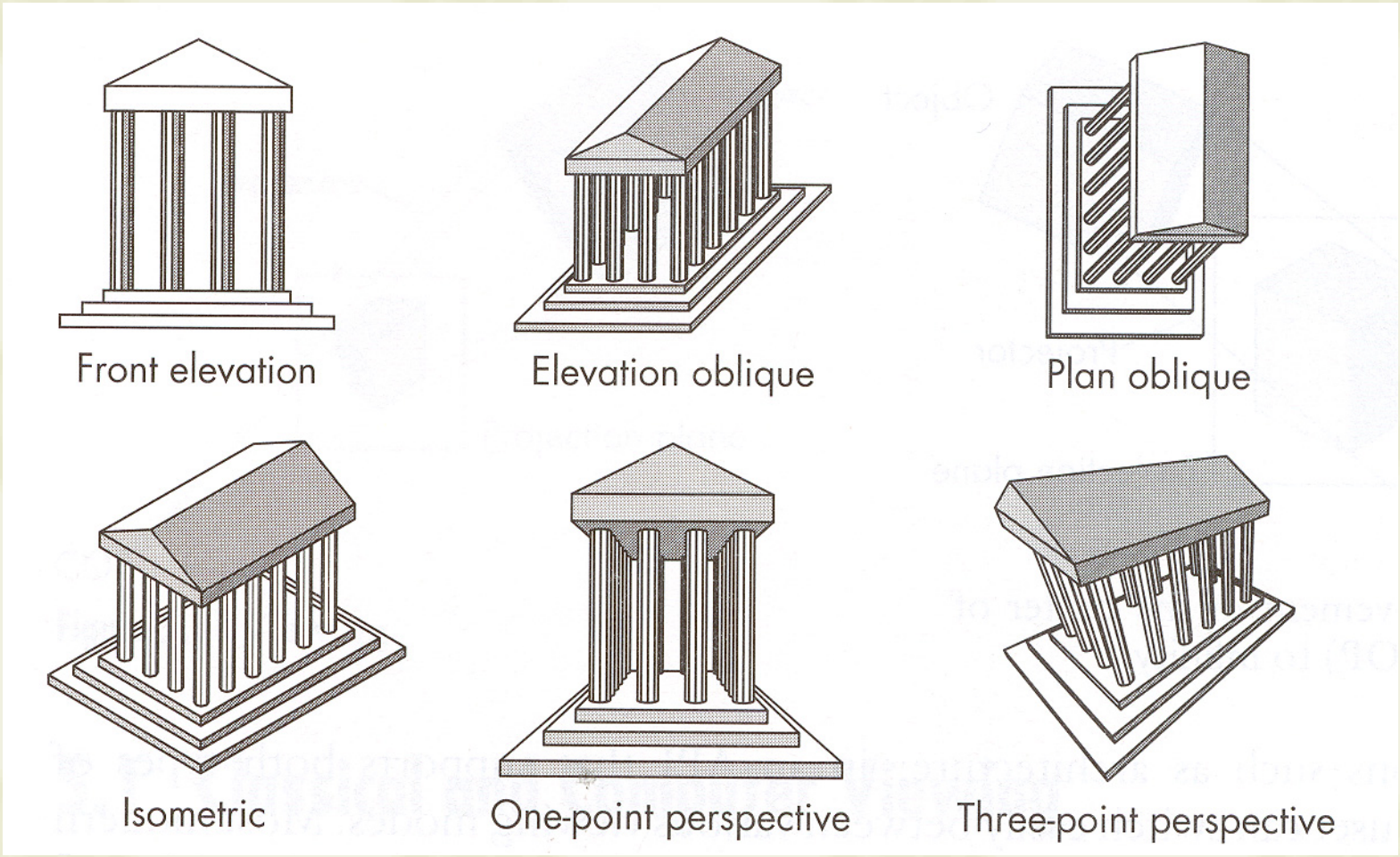
Illuminate according to lighting and reflectance
Apply texture maps

Transform into DCs

Clip primitives outside camera's view

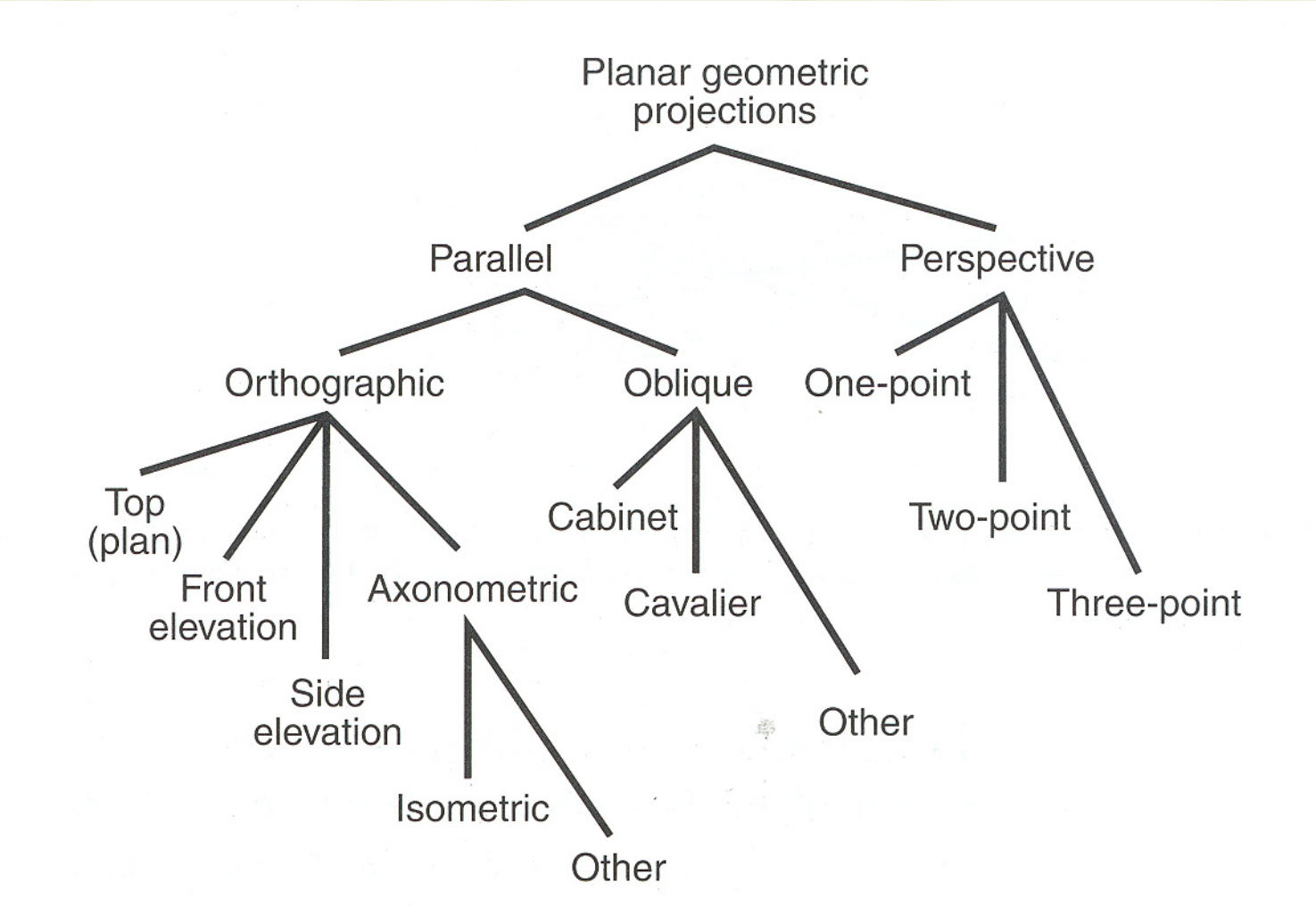
Draw pixels (includes texturing, HSR, etc.)

Projection zoo (1/2)



© J.D. Foley et al.: *Computer Graphics: Principles and Practice (2nd ed. in C)*, Addison-Wesley, 1997 [but still the "[3D] computer graphics bible"!]

Projection zoo (2/2)



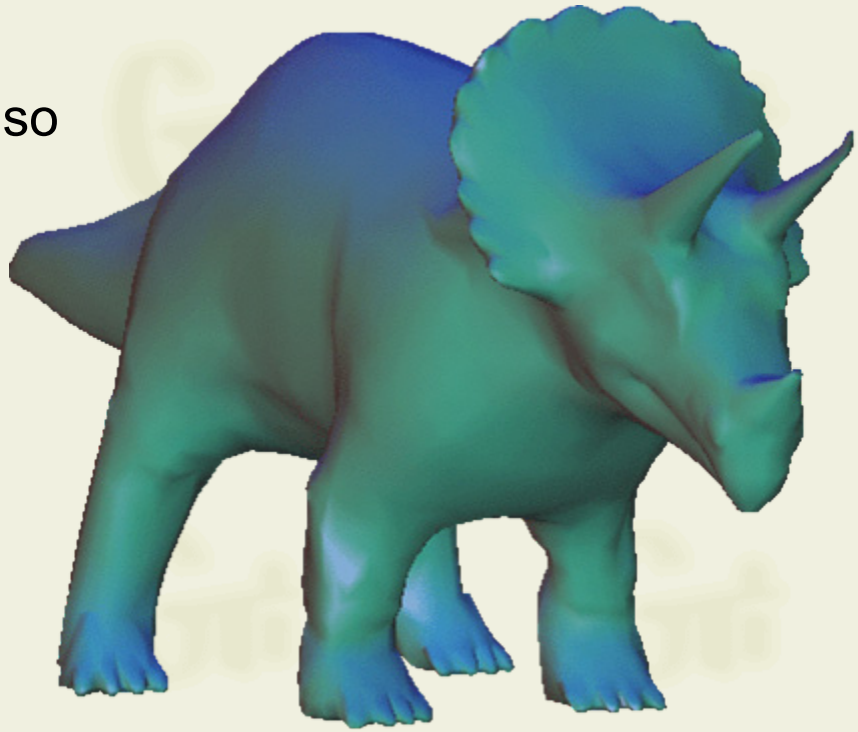
© J.D. Foley et al.: *Computer Graphics: Principles and Practice (2nd ed. in C)*, Addison-Wesley, 1997 [but still the "[3D] computer graphics bible"!]

Historical perspective of monitors & co.

- ★ Monochrome phosphor CRT (Cathode Ray Tube) monitors
  - ✸ 1! shade of grey, green or amber (black in printers)

    ⇒ impossible to convey “3D feeling” ⇒ wireframe rendering only
- ★ Colour monitors (CRT, plasma, LCD/TFT, LED, etc.)
  - ✸ 2^{24} (>17M) colours

    ⇒ hidden parts can be removed so the following make sense:
    - ● Faceted/flat shading
    - ● Gouraud’s shading
    - ● Phong’s shading
    - ● Texture mapping, shadows, etc.
- ★ “3D” monitors
  - ✸ [Auto-]stereoscopic
  - ✸ Holographic, etc.



Static 3D object modelling: summary

★ Volumes/solids: “true 3D”

★ Surfaces: “2,5D”

- ✸ Shape
  - ● Meshes of polygons/patches
  - ● Subdivision surfaces
- ✸ Appearance: colours/textures

★ IB[M]R (Image-Based [Modelling and] Rendering)

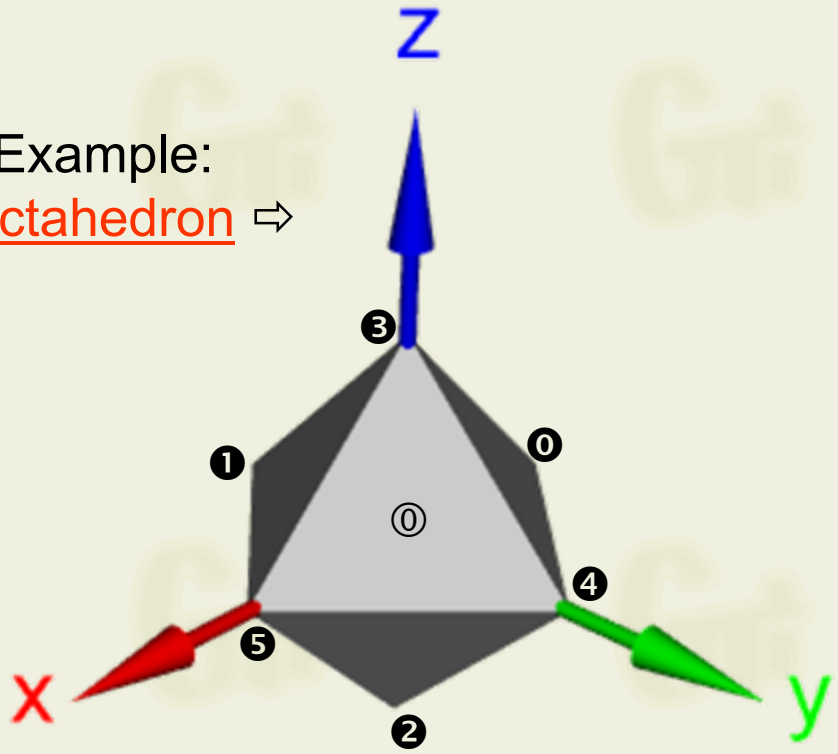
- ✸ Simultaneous (and not explicit) description of both shape and texture
- ✸ Example: mythical bullet (pre “Dodge this!”) scene from *The Matrix*

Meshes of polygons (1/3)

- ★ Usually, of triangles (or triangulated polygons)
- ★ Two kinds of information ⇒ two lists/tables
  1. Geometry: vertex table
  2. Topology/connectivity: polygon table
- ★ IFS: Indexed Face Set

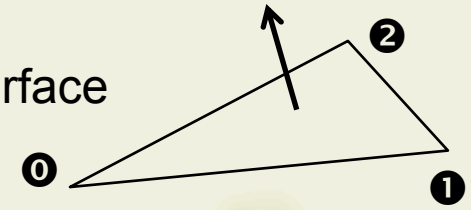
1. Vertices	2. Polygons
nº (*x*, *y*, *z*)	nº (*i*, *j*, *k*…,•)
❶ (-1,0,0)	⓪ (❸,❺,❹,•)
❶ (0,-1,0)	① (❸,❹,❶,•)
❷ (0,0,-1)	② (❸,❶,❶,•)
❸ (0,0,+1)	③ (❸,❶,❺,•)
❹ (0,+1,0)	④ (❷,❹,❺,•)
❺ (+1,0,0)	⑤ (❷,❶,❹,•)
	⑥ (❷,❶,❶,•)
	⑦ (❷,❺,❶,•)

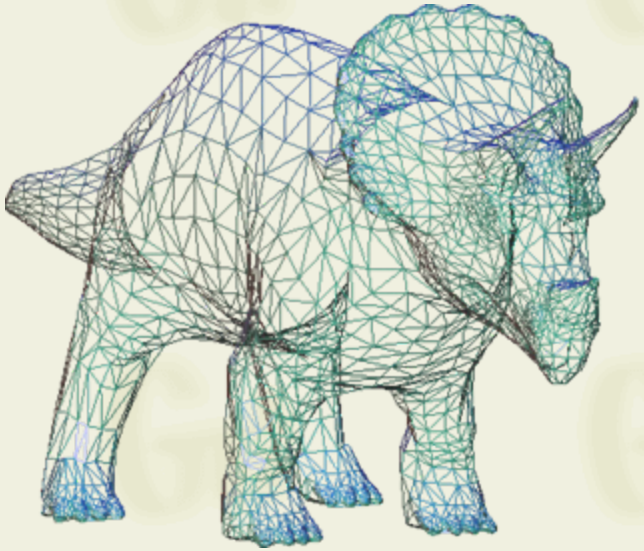
Example:
⇦ octahedron ⇨



Meshes of polygons (2/3)

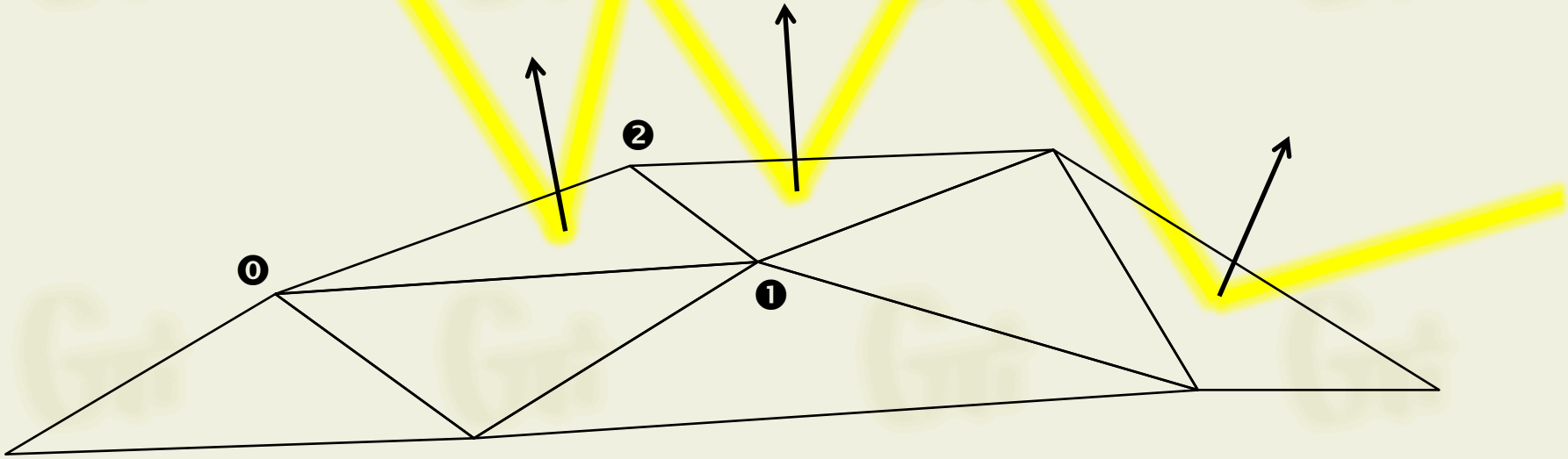
- ★ Orientation of face[t]s/polygons
  - ✸ Vertices are ordered to define outward [pointing] normal[vector]s…
    - Usual ordering: CCW (Counter-ClockWise)
    - NB: outward normals ⇔ closed and orientable surface
  - ✸ … used at rendering time
    - Improved efficiency thanks to [back-facing polygon] culling





Meshes of polygons (3/3)

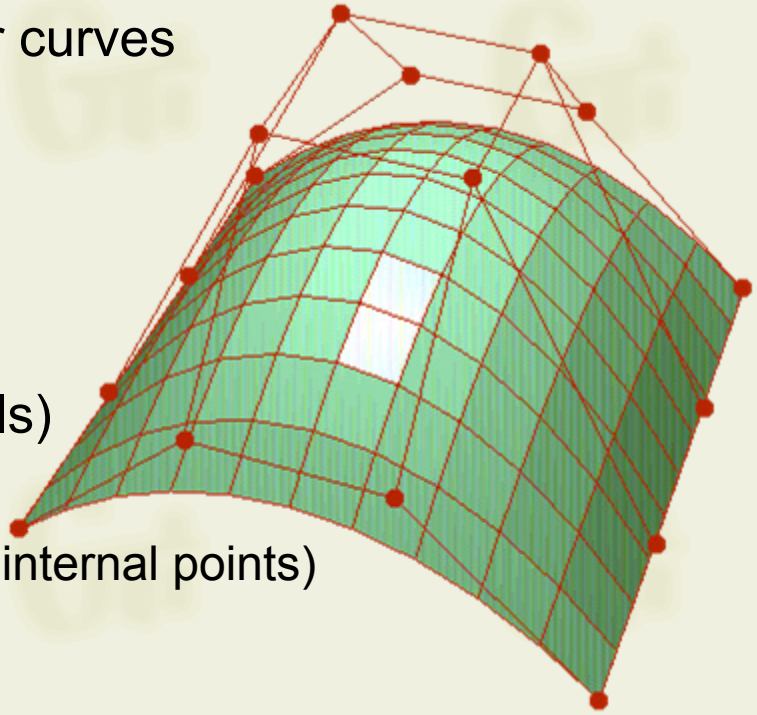
- Orientation of face[t]s/polygons
  - […] outward normals…
    - […]
  - … used at rendering time
    - […]
    - Incident light is reflected accordingly



© S. Arnaldo

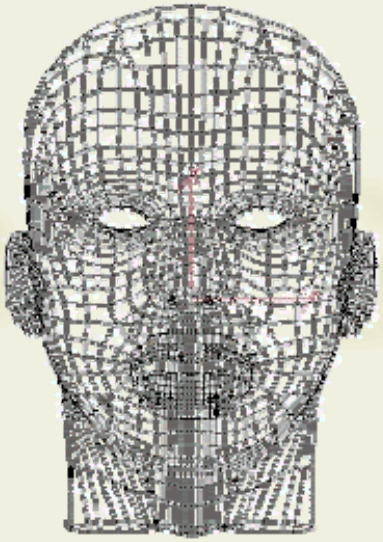
Meshes of patches (1/2)

- Atomic element: curved patch (vs. flat polygon)
  - Piecewise higher order (vs. linear) surface approximation
  - Several kinds
    - Polynomial, e.g., Bézier['s] patches
    - Rational, e.g., NURBSs (Non-Uniform Rational B-Splines)
  - Tensor product of two perpendicular curves
    - Two parametric variables: (u, v)
    - Mesh of control points (vs. vertices)
    - [Two knot/weight vectors]
- Example: bicubic Bézier patch
  - Degree: 3x3 (Bernstein's polynomials)
  - Order: 4x4 (control mesh)
    - Corners always interpolated (unlike internal points)
  - No need for knot vectors



Meshes of patches (2/2)

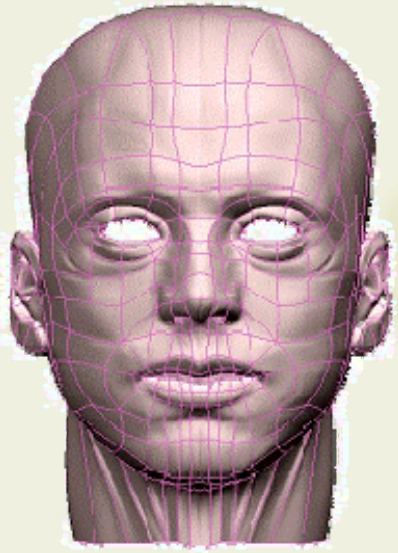
- ★ Advantages (vs. polygons)
  - ✸ Compactness for editing (⇒ animation) and coding



⇦ 3547 facets and
1215 vertices

vs.

86 patches and
212 control points ⇨

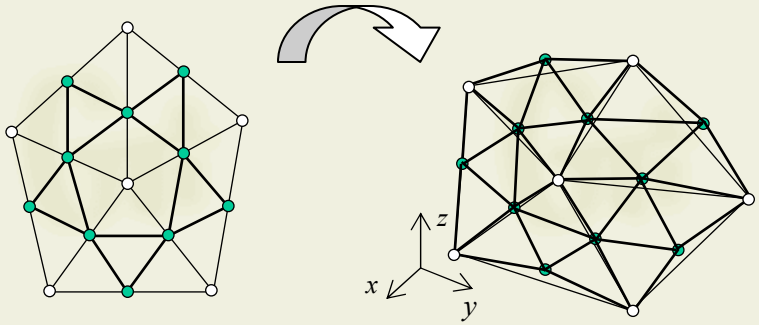


- ★ Disadvantages
  - ✸ Fixed topology (surface homeomorphic to the plane)
    - ⇒ *Need for trimming & stitching* mechanisms

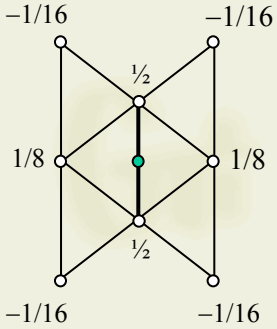
SSs (Subdivision Surfaces)

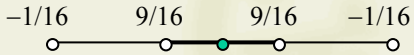
★ SS = limit of control mesh recursive refinement process

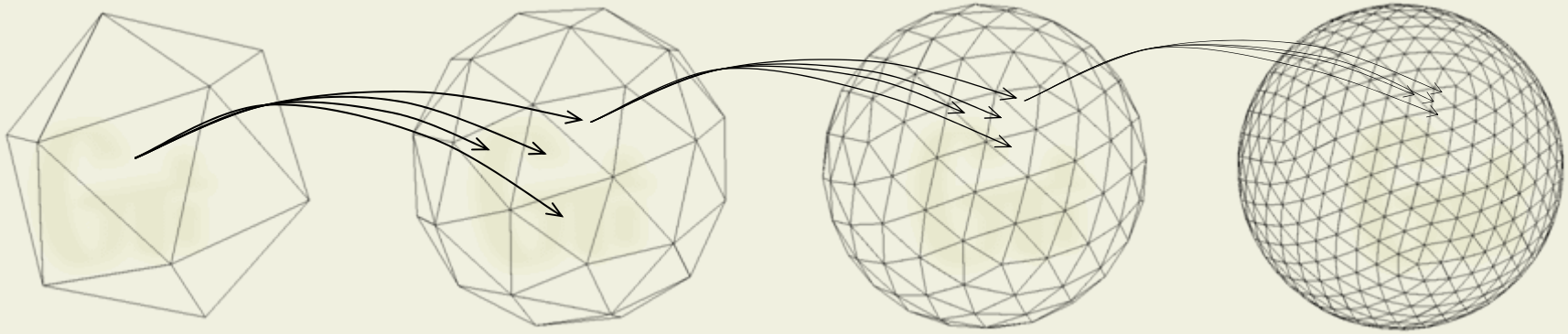
Simultaneous refinement of both
⇩ (abstract graph) topology and
(3D mapping) geometry ⇩



Geometric smoothing achieved thanks to mask/stencil (particular to each subdivision scheme)

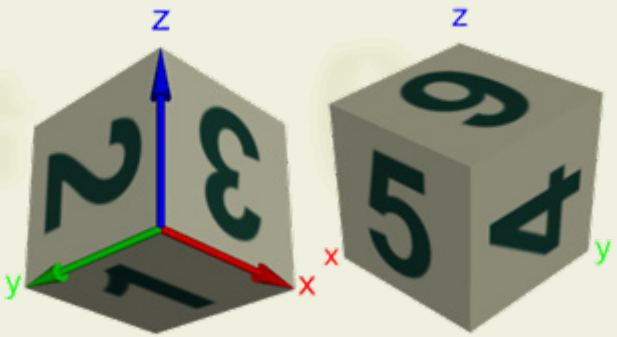




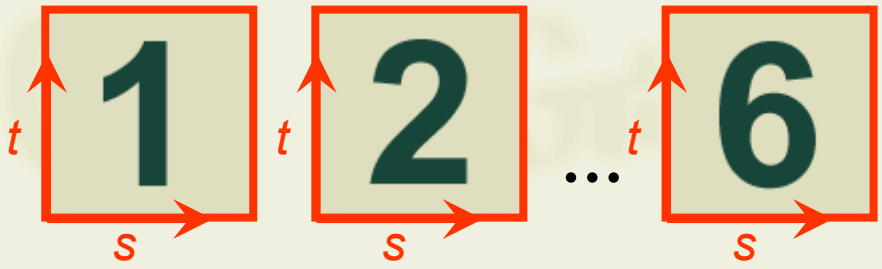


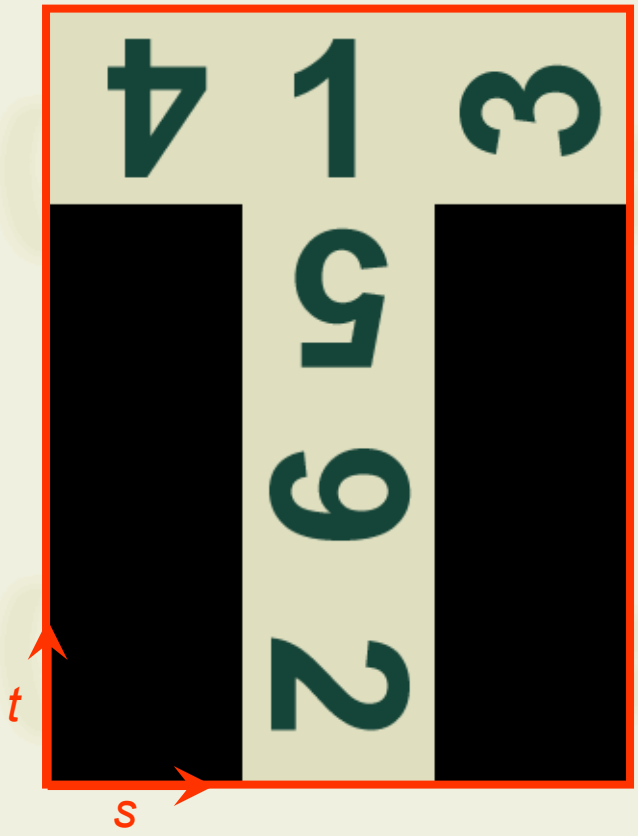
Texturing

- Wrapping of surface in one (or more) texture(s)
  - Image (perhaps virtual, formed by several images: texture atlas)
  - Assignment of texture coordinates (*s*, *t*) to vertices (or corners)
- Example: die modelled with…



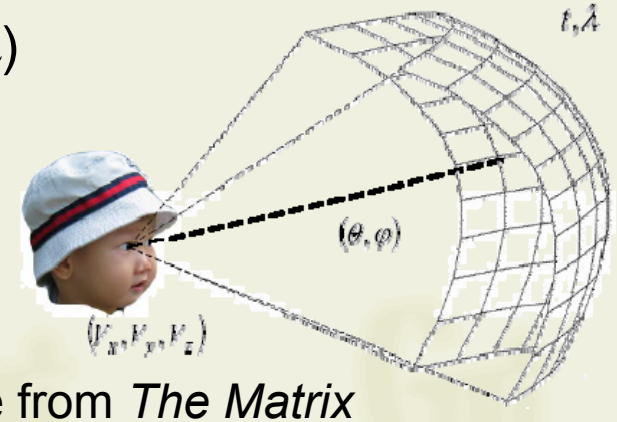
… 6 IFSs ⇩ vs. 1 IFS ⇨





IB[M]R (Image-Based [Modelling &] Rendering)

- ★ Techniques without complete explicit 3D scene model
  - ✸ Plenoptic function: P_7 (V_x, V_y, V_z, θ, φ, t, λ)
    - ⇒ 7 variables ⇒ TOUGH!
  - ✸ Restrictions upon P_7
    - P_5: fixed scene and lighting
    - P_4: Light Field Rendering, “Lumigraph”
    - P_3: 3D concentric mosaics, “FloMo” scene from *The Matrix*
    - P_2: 2D mosaics and panoramas
  - ✸ Some techniques (e.g., DIBR) include partial geometry information



- ★ Comparison with traditional method (explicit model)
  - ☺ Very simple modelling, based on “real” photographs
  - ☺ CPU rendering cost independent from scene complexity
  - ☹ RAM rendering cost can turn huge
  - ☹ GPUs (still?) not tuned for IBR

Animation (1/2)

- Interpolation based on key frames
  - Modelling: human specifies couples (key frame, key value)
  - Rendering: machine generates intermediate “frames” by
    - linear interpolation
    - higher order interpolation
  - [Coding (program) removes redundant information]
- Very common mechanism in games
- Similar to dead reckoning (used for extrapolation)
  - Sea/air navigation: different positioning mechanisms
  - Car/foot navigation: GPS
- Example: key

Animation (2/2)

- ★ BBA (Bone-Based Animation) model
  - ✸ A.k.a. skin & [muscles &] bones
  - ✸ Concept: articulated hierarchical model
    - ● Skeleton = hierarchical rigid bone structure
    - ● Muscles tied to bones through weights
    - ● Skin tied to muscles
  - ✸ Process
    - ● Modelling: specification of skeleton, muscles (weights) and skin
    - ● Animation: movement of bones, muscle deformation
    - ● Rendering: skin defined with polygon mesh, SSs… after transforms
- ★ IK (Inverse Kinematics)
  - ✸ Dynamic (physical) restriction set upon skeleton
  - ✸ Eases animation

© Télécom-SudParis

Interactive scenes

- Hierarchical scene graph
  - Different coordinates: MCs (Modelling Coords.) vs. WCs (World Cs.)
  - Hierarchical structure: (acyclic) node tree
  - Modelling transforms affect whole subtrees
  - Enables efficient mechanisms for rendering management
    - Space partitioning, cells & portals

- Interaction
  - Event-based model
  - Routes between nodes
  - Examples: key++ and keyboard

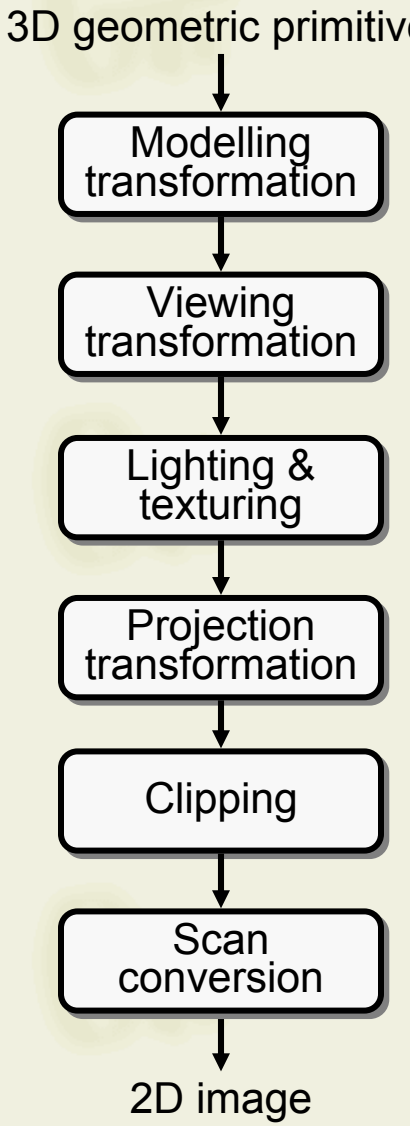
Standards (1/2)

- De jure
  - Published by ISO (International Organization for Standardization)
    - Jointly developed by companies and universities or research centres
    - Proposals discussed and approved by consensus
    - Balloting process involving member countries
  - VRML family developed by ISO/IEC JTC1/SC24
    - *.{wrl,vrml}: VRML97 (= VRML 2.0), created with VRML (Virtual Reality Modeling Language) Consortium
    - *.x3d: X3D, created with Web 3D Consortium
  - *.mp4: MPEG-4 developed by ISO/IEC JTC1/SC29/WG11 = MPEG (Moving Picture Experts Group)
    - Part 16: AFX (Animation Framework eXtension)
    - Part 2: Visual
    - Part 11: Scene description and application engine

Standards (2/2)

- De facto
  - Adopted by industry
  - May be closed (proprietary) or open
  - Autodesk
    - *.dxf: Autodesk AutoCAD
    - *.{3ds,max}: 3D Studio [MAX] ⇒ Autodesk 3ds Max
    - *.obj: (Alias ⇒ Alias|Wavefront ⇒ Autodesk) Maya
    - *.xsi: Softimage|3D ⇒ Avid Softimage|XSI ⇒ Autodesk Softimage
  - Khronos: consortium of 100+ companies
    - *.dae: COLLADA
    - NB: Khronos also maintains other open standards such as OpenGL

Rendering pipeline (reminder)



Transform into WCs

Transform into CCs
Done with modelling transformation

Illuminate according to lighting and reflectance
Apply texture maps

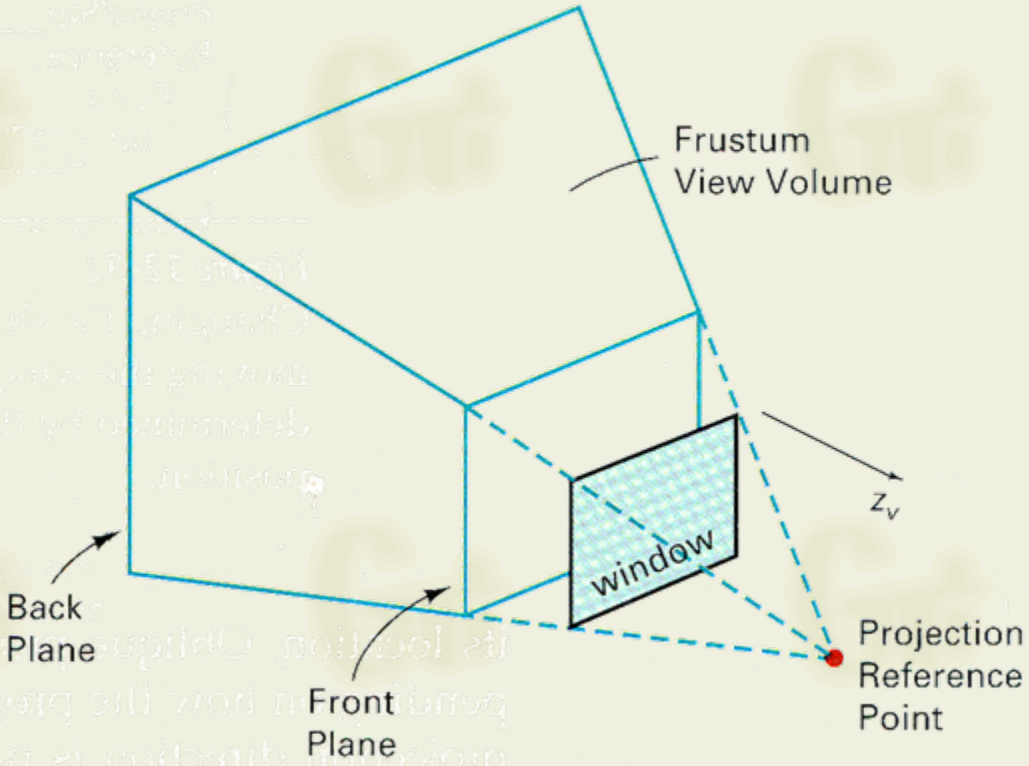
Transform into DCs

Clip primitives outside camera's view

Draw pixels (includes texturing, HSR, etc.)

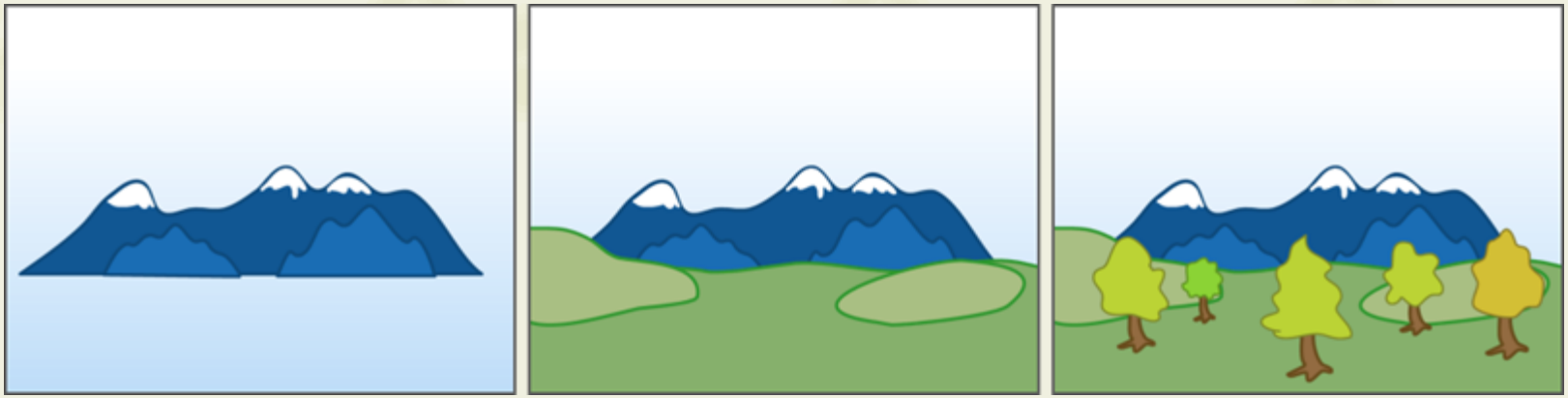
2D techniques

- ★ Bresenham's algorithms
  - ✸ Very efficient: incremental and designed for fixed point arithmetic
  - ✸ Versions for lines and circles/ellipses
- ★ Polygons
  - ✸ Triangulation
  - ✸ Filling
- ★ Clipping
  - ✸ Window borders…
    - ● Left, right, top, bottom
  - ✸ … but also
    - ● Front and back! ⇨

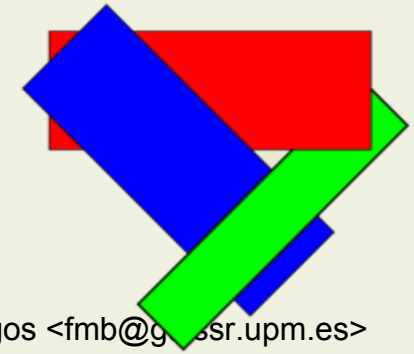


Hidden surface removal (1/2)

- ★ Problem: hide (if possible, don't even draw) what's "behind"
- ★ Solutions
  - ✸ Image precision algorithms, e.g., ray tracing
  - ✸ Object precision algorithms
    - ● Ordering of graphics primitives prior to drawing, e.g., painter's algorithm



    - ● Problem: not always possible!

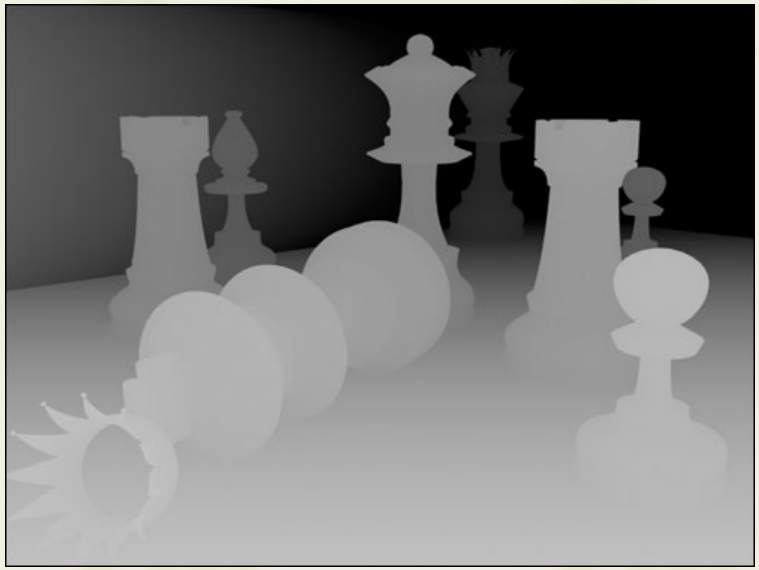


Hidden surface removal (2/2)

- ★ Solutions
  - ✸ Object precision algorithms
    - ● […] and the winner is…
    - ● Depth/*z* buffer*:* extra memory to store depth (in CCs)

      ⇒ Implemented in graphic cards with HW acceleration

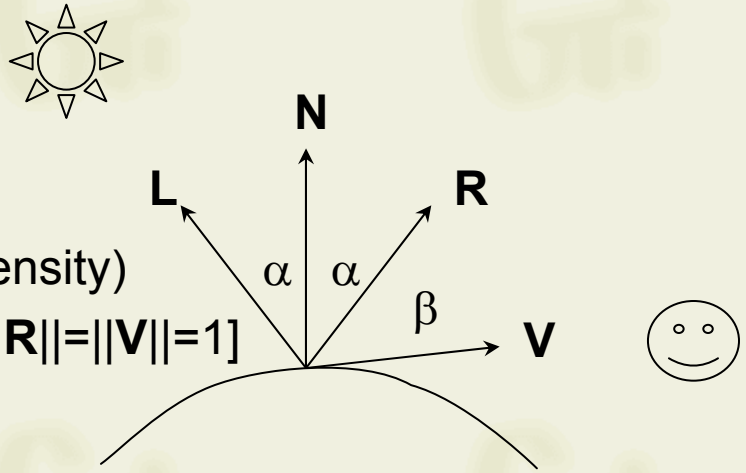
Frame buffer



Depth/*z* buffer

Illumination/lighting

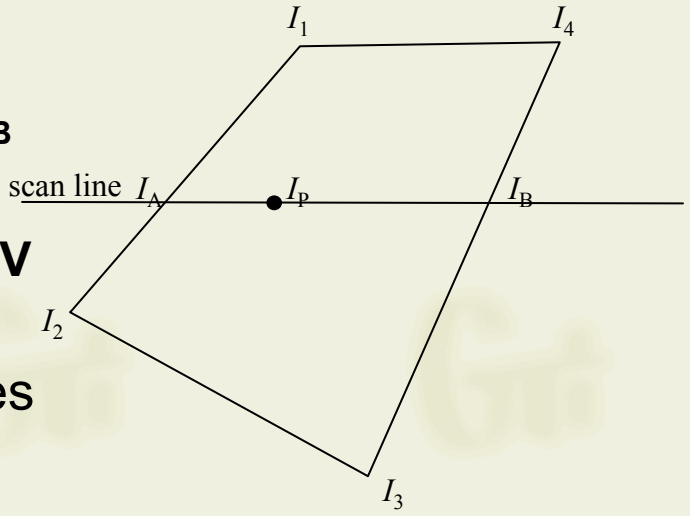
- ★ Light sources (intensity $\mathbf{I}_{RGB}$): directional vs. positional
- ★ Importance of normals
- ★ Diffuse reflection
  - ✸ Matte material (e.g., chalk, paper) ⇒ coefficient k_d (intrinsic colour)
  - ✸ Lambert's model: $\mathbf{I}_{RGB,d} = k_d \mathbf{I}_{RGB} \cos \alpha$ [$= k_d \mathbf{I}_{RGB} \mathbf{L}\cdot\mathbf{N}$ if $||\mathbf{L}||=||\mathbf{N}||=1$]
- ★ Specular reflection
  - ✸ Shiny material (e.g., metal, plastic)
  - ✸ Phong's model
    - Coefficients k_s and n ("colour" and intensity)
    - $\mathbf{I}_{RGB,s} = k_s \mathbf{I}_{RGB} \cos^n \beta$ [$= k_s \mathbf{I} (\mathbf{R}\cdot\mathbf{V})^n$ if $||\mathbf{R}||=||\mathbf{V}||=1$]



- ★ Total reflection
  - ✸ Kludge: ambient light (sum of other contributions)
  - ✸ $\mathbf{I}_{RGB,tot} = \mathbf{I}_{RGB,d} + \mathbf{I}_{RGB,s} + \mathbf{I}_{RGB,a}$

Shading (colour interpolation)

- ★ NB: shading ≠ shadow[ing]
  - ✸ Gouraud's technique
    - ● Bilinear (2D) interpolation of colours: $\mathbf{I_{RGB}}$
  - ✸ Phong's technique
    - ● Bilinear interpolation of vectors: **L**, **N**, **R**, **V** (and then, of course, colour: $\mathbf{I_{RGB}}$)
  - ✸ Similar techniques for texture coordinates
    - ● Problem: warping

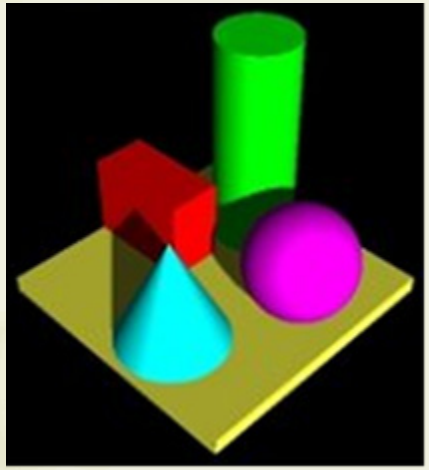


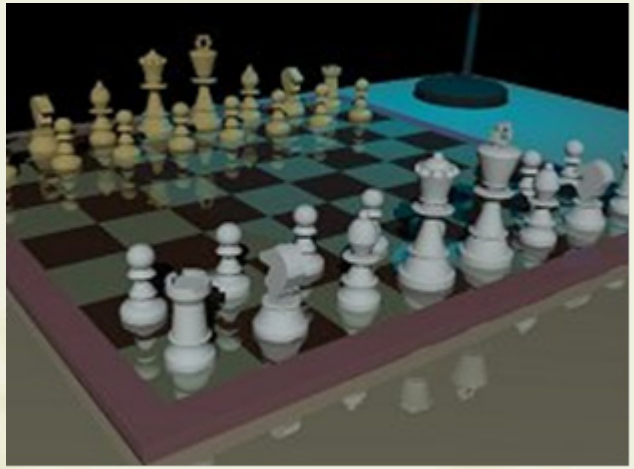
Flat Affine Correct

© G. Wolberg: *Digital Image Warping*, IEEE Computer Society Press, 1990

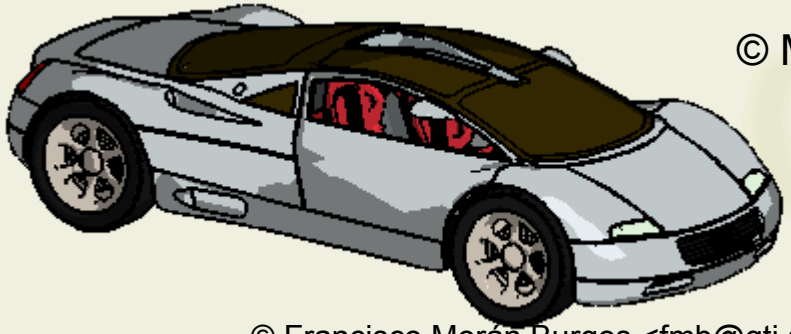
“Luxury”

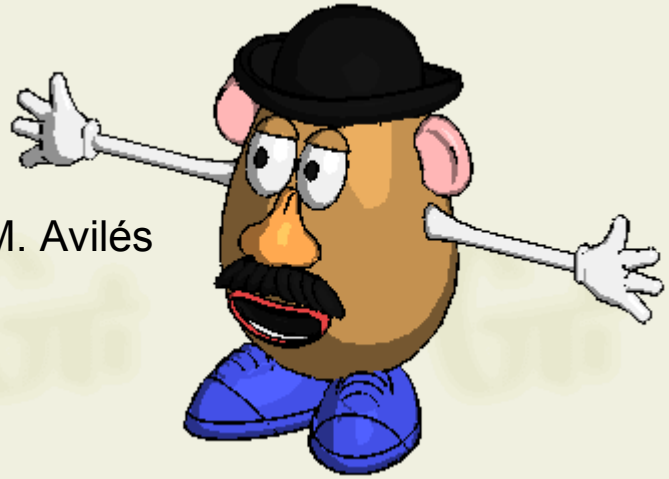
- ★ Luxury++
  - ✸ Transparencies
  - ✸ Shadows
  - ✸ Ref{le,ra}ctions
  - ✸ Depth of field
  - ✸ Motion blurring
  - ✸ etc., etc., etc.





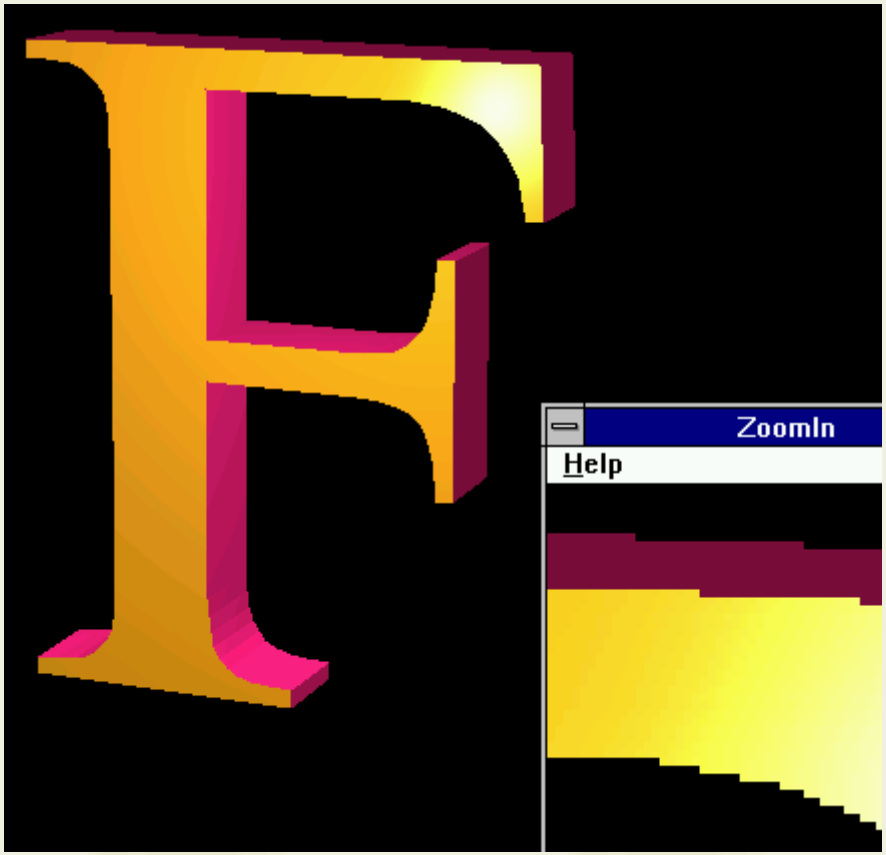
- ★ But also luxury--
  - ✸ NPR: Non-Photorealistic Rendering, e.g., cartoon shading

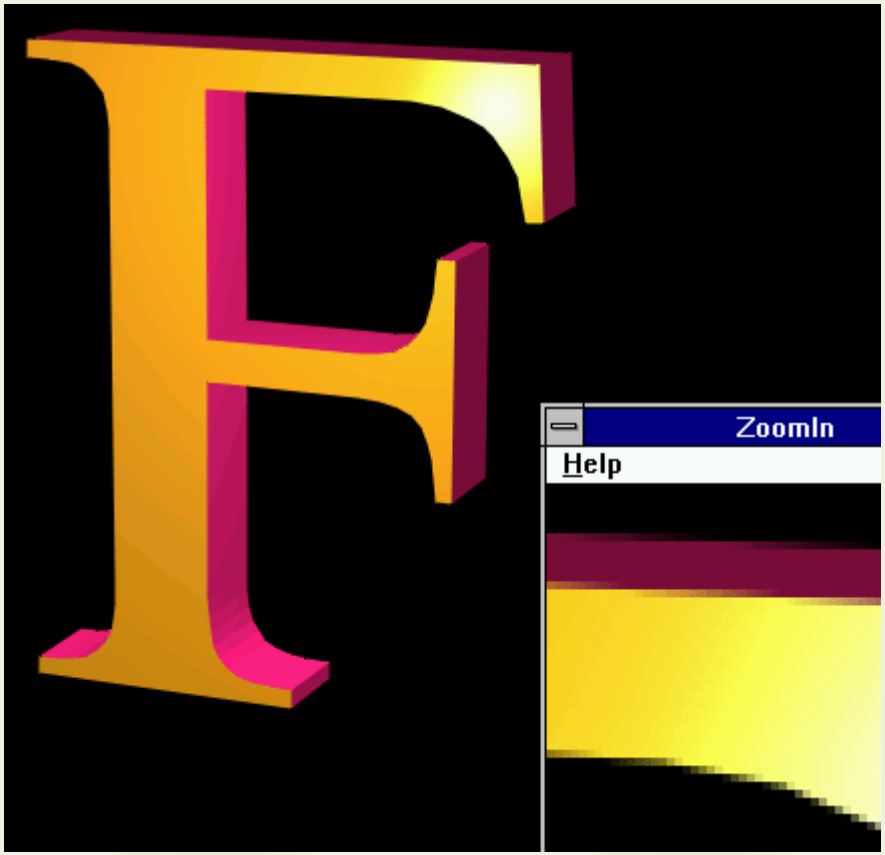




© M. Avilés

Image processing: anti-aliasing filtering, etc.





That's all, folks!

- ★ Synthetic content representation
  - ✸ Introduction
    - ● Spatial transforms
    - ● Historical perspective of viewing devices
  - ✸ Modelling
    - ● Description of each 3D object: shape & appearance [& animation ⇒ 4D]
    - ● Id. of [interactive] 3D scene with different objects, lights, cameras…
    - ● Very time-consuming task for humans
  - ✸ Rendering
    - ● 2D image/video generation from 3D/4D scene model, usually seeking photo-realism
    - ● Approximation of physical laws through lighting models
    - ● Very time-consuming task for computers
- ★ Francisco Morán Burgos <fmb@gti.ssr.upm.es>