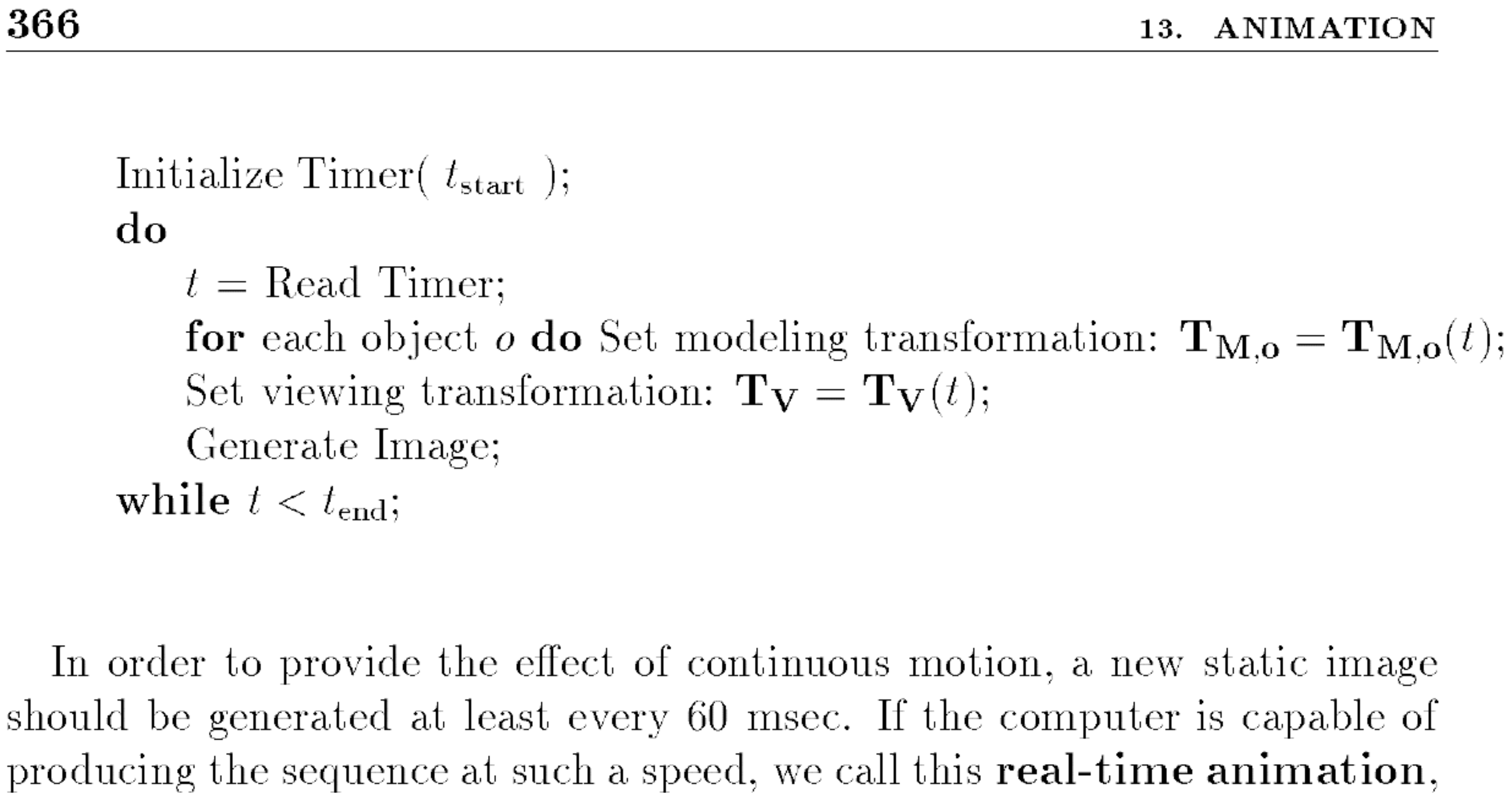
Computer Graphics 3, texturing

Andrej Ferko

ferko@fmph.uniba.sk

Short version 2020, www.sccg.sk/PG3, v281020

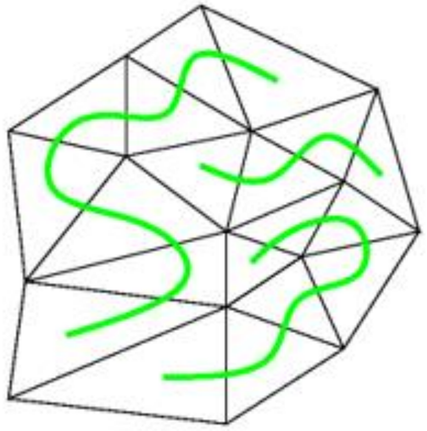
Real-time animation [Szirmay-Kalos]

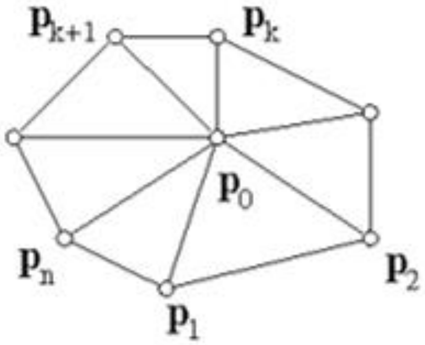


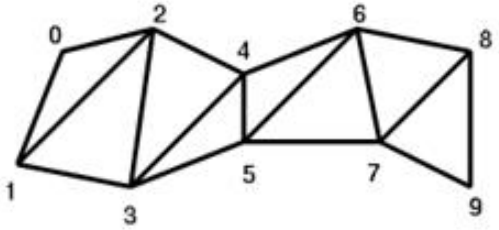
```
Initialize Timer( $t_{\text{start}}$ );
do
    $t$ = Read Timer;
    for each object $o$ do Set modeling transformation: $\mathbf{T}_{\mathbf{M,o}} = \mathbf{T}_{\mathbf{M,o}}(t)$;
    Set viewing transformation: $\mathbf{T}_{\mathbf{V}} = \mathbf{T}_{\mathbf{V}}(t)$;
    Generate Image;
while $t < t_{\text{end}}$;
```

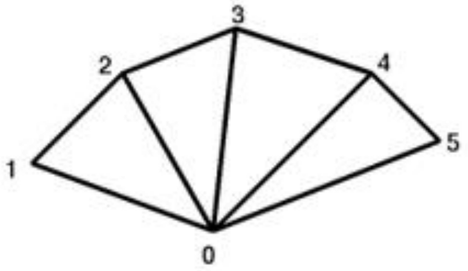
In order to provide the effect of continuous motion, a new static image should be generated at least every 60 msec. If the computer is capable of producing the sequence at such a speed, we call this **real-time animation**,

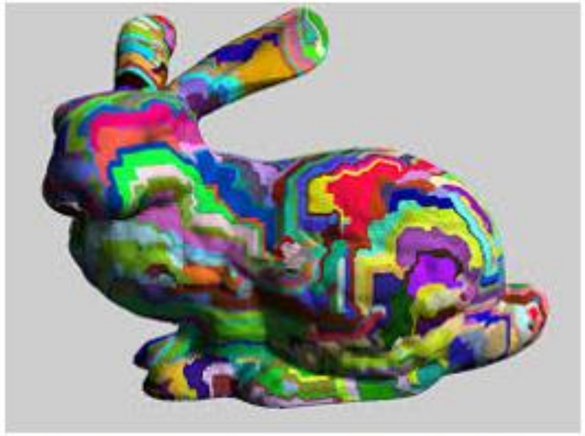
Stripes, fans, Hamiltonian mesh











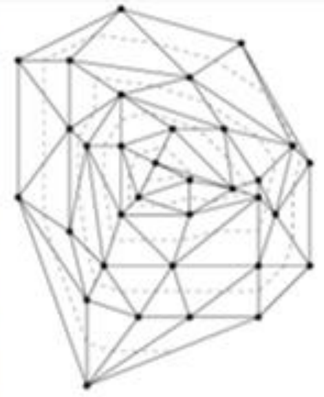


Figure 11: Hamiltonian triangulation via onion method.

http://wscg.zcu.cz/wscg2004/Papers_2004_Full/G07.pdf https://www.cs.cmu.edu/~kmcrane/Projects/StripePatterns/ http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.3639&rep=rep1&type=pdf

Texturing [Szirmay-Kalos, Ruzicky]

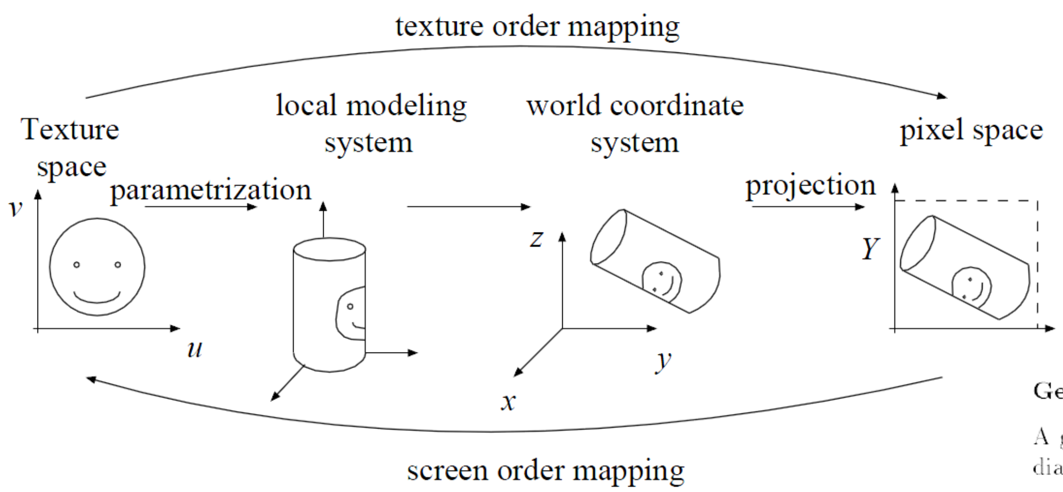


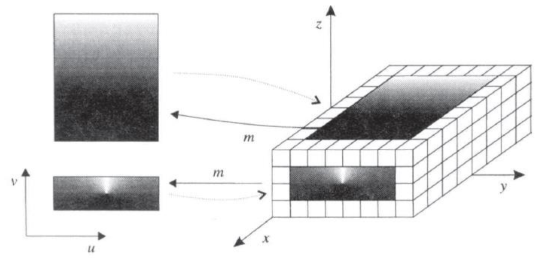
Figure 12.1: Survey of texture mapping

Z matematického hľadiska definujeme **všeobecnú textúru** ako zobrazenie rovinnej oblasti do modulovaného priestoru, ktorým môže byť priestor farieb alebo úrovní šedej

$t : D_t \to M$, kde $D_t \subseteq R^2$ a $M \subseteq R\ (R^3)$.

Ak máme zadaný tvar objektu, tak pomocou inverzného mapovania budeme zobrazovať pre každý bod povrchu bod z oblasti textúry

$m : D_m \to D_t$, kde D_m je oblasť na povrchu objektu.



Obr. 15.1 Mapovanie textúry (texture mapping)

General surfaces

A general technique developed by Bier and Sloan [BS86] uses an intermediate surface to establish a mapping between the surface and the texture space. When mapping from the texture space to the surface, first the texture point is mapped onto the intermediate surface by its parameterization, then some "natural" projection is used to map the point onto the target surface. The texturing transformation is thus defined by a two-phase mapping.

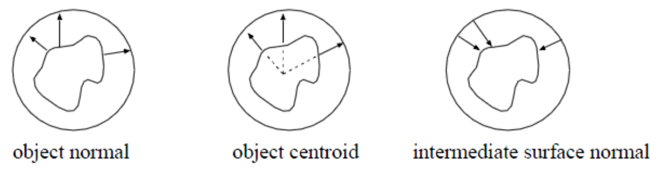
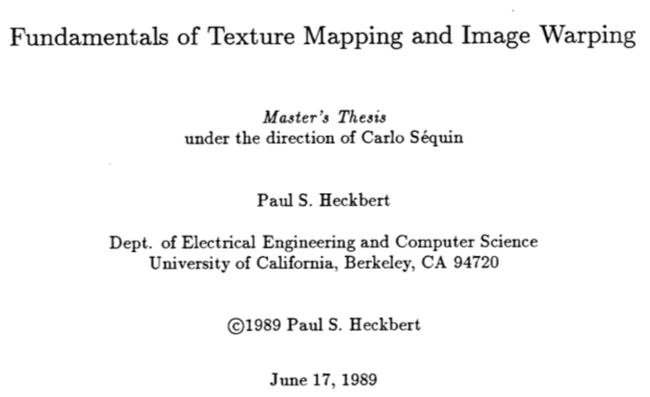


Figure 12.3: Natural projections

The intermediate surface must be easy to parameterize and therefore usually belongs to one of the following categories:

1. Planar polygon
2. Sphere
3. Cylinder
4. Faces of a cube.

Texturing [Heckbert]



Fundamentals of Texture Mapping and Image Warping

Master's Thesis
under the direction of Carlo Séquin

Paul S. Heckbert

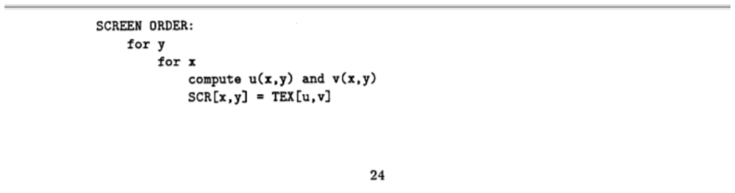
Dept. of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

©1989 Paul S. Heckbert

June 17, 1989

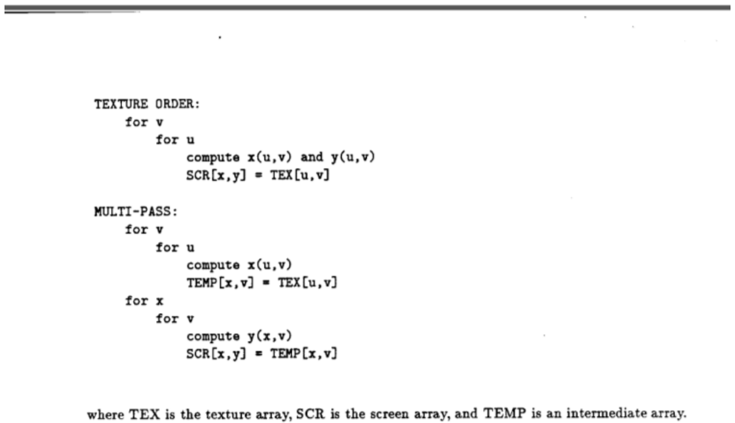
$$\begin{tikzcd}
\textbf{2-D texture space} \arrow[d, "\textit{parameterization}"] \arrow[dd, bend right=60, "\textit{compound mapping}"'] \\
\textbf{3-D object space} \arrow[d, "\textit{projection}"] \\
\textbf{2-D screen space}
\end{tikzcd}$$

Figure 2.1: *The compound mapping is the composition of the surface parameterization and the viewing projection.*



```
SCREEN ORDER:
    for y
        for x
            compute u(x,y) and v(x,y)
            SCR[x,y] = TEX[u,v]
```

24



```
TEXTURE ORDER:
    for v
        for u
            compute x(u,v) and y(u,v)
            SCR[x,y] = TEX[u,v]

MULTI-PASS:
    for v
        for u
            compute x(u,v)
            TEMP[x,v] = TEX[u,v]
    for x
        for v
            compute y(x,v)
            SCR[x,y] = TEMP[x,v]
```

where TEX is the texture array, SCR is the screen array, and TEMP is an intermediate array.

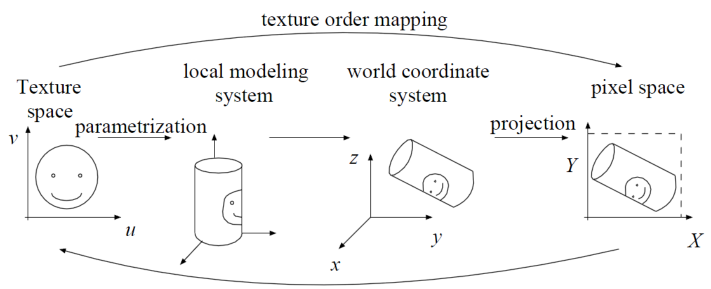


Figure 12.1: Survey of texture mapping

Texturing [Heckbert]

$$\begin{tikzcd}
\text{2-D texture space} \arrow[d, "parameterization"] \arrow[dd, bend right=60, swap, "\text{compound mapping}"] \\
\text{3-D object space} \arrow[d, "projection"] \\
\text{2-D screen space}
\end{tikzcd}$$

Figure 2.1: *The compound mapping is the composition of the surface parameterization and the viewing projection.*

⇓ MAP1 ∘ MAP2 ⇒	affine	bilinear	projective
affine	affine	bilinear	projective
bilinear	bilinear	biquadratic	rational bilinear
projective	projective	rational biquadratic	projective

Thus, the composition of two bilinear mappings is a biquadratic mapping.

PROPERTY	AFFINE	BILINEAR	PROJECTIVE
preserves parallel lines	yes	no	no
preserves lines	yes	no[†]	yes
preserves equispaced points	yes	no[†]	no
maps square to	parallelogram	quadrilateral	quadrilateral
degrees of freedom	6	8	8
closed under composition	yes	no, biquadratic	yes
closed under inversion	yes	no, solve quadratic	yes
single-valued inverse	yes	no	yes
forms a group	yes	no	yes
incremental forward mapping	2 adds	2 adds	2 divs, 3 adds[‡]
incremental inverse mapping	2 adds	1 square root, more	2 divs, 3 adds

[†]except for horizontal and vertical lines in source space

[‡]see §2.3.5

For the designer, affine mappings are the simplest of the three classes. If more generality is needed, then projective mappings are preferable to bilinear mappings because of the predictability of line-preserving mappings. For the implementer, the group properties of affine and projective mappings make their inverse mappings as easy to compute as their forward mappings. Bilinear mappings are computationally preferable to projective mappings only when the forward mapping is used much more heavily than the inverse mapping.

```
SCREEN ORDER:
    for y
        for x
            compute u(x,y) and v(x,y)
            SCR[x,y] = TEX[u,v]
```

24

```
TEXTURE ORDER:
    for v
        for u
            compute x(u,v) and y(u,v)
            SCR[x,y] = TEX[u,v]

MULTI-PASS:
    for v
        for u
            compute x(u,v)
            TEMP[x,v] = TEX[u,v]
    for x
        for v
            compute y(x,v)
            SCR[x,y] = TEMP[x,v]
```

Bilinear Interpolation [Ru]

[Akenine-Moller, RTR]

Zobrazenie t sa často definuje tabuľkou s celočíselnými hodnotami. Môžu to byť obrázky zosnímané skenerom alebo vytvorené grafickým editorom, ktoré ukladajú informáciu v diskrétnej podobe. Inverzné zobrazenie mapuje do oblasti D_t vo všeobecnosti reálnymi hodnotami, preto musíme vedieť interpolovať chýbajúce hodnoty. Najčastejšie sa využíva **bilineárna interpolácia**.

Chceme získať hodnotu $t(x, y)$, preto označme najbližšie hodnoty nasledovne:

$\lfloor x \rfloor$ - zaokrúhlenie smerom dole na celočíselnú hodnotu a

$\lceil x \rceil$ - zaokrúhlenie smerom hore,

$t_{11} = t(\lfloor x \rfloor, \lfloor y \rfloor)$, $\quad t_{12} = t(\lfloor x \rfloor, \lceil y \rceil)$,

$t_{21} = t(\lceil x \rceil, \lfloor y \rfloor)$, $\quad t_{22} = t(\lceil x \rceil, \lceil y \rceil)$.

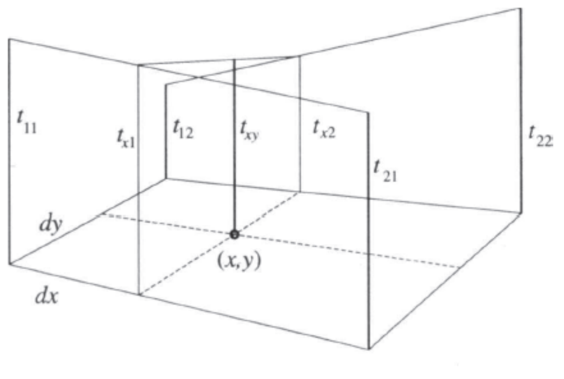
Z obrázku 15.2 vidíme ako vypočítať hodnotu $t(x, y)$ pomocou interpolácie:

$t_{x1} = t_{11}(1 - dx) + t_{21}(dx)$, $\qquad t_{x2} = t_{12}(1 - dx) + t_{22}(dx)$,

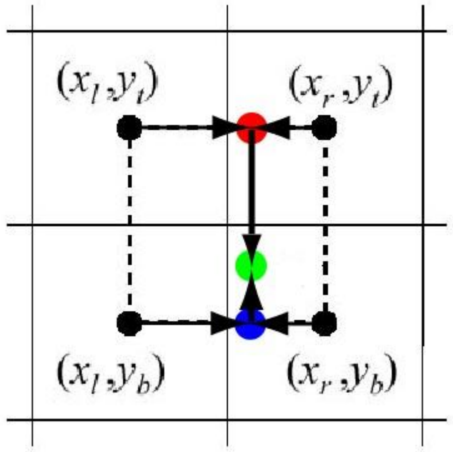
$t(x, y) = t_{x1}(1 - dy) + t_{x2}(dy)$.

Po úprave

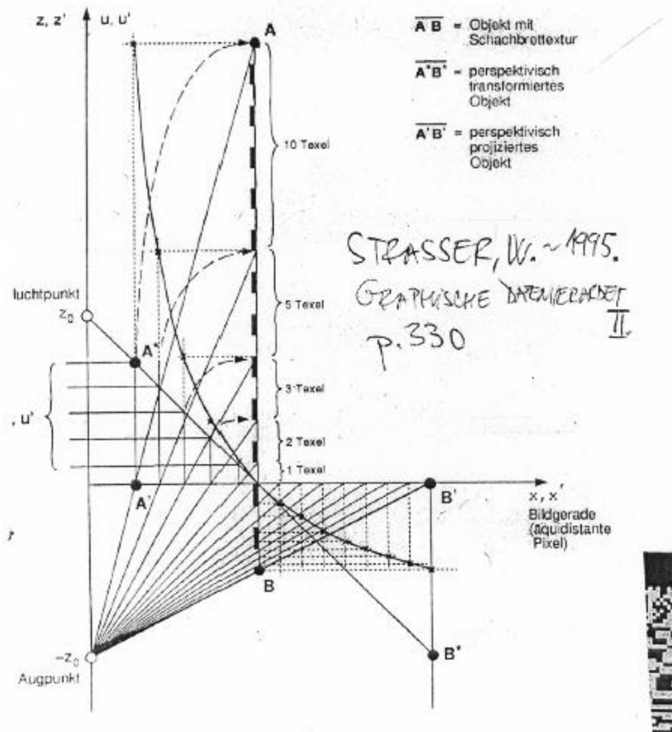
$$t(x,y) = t_{11} + (t_{12} - t_{11})dy + [t_{21} - t_{11} + (t_{11} - t_{12} - t_{21} + t_{22})dy]dx$$



Obr. 15.2 Výpočet interpolovanej hodnoty t(x, y)



Aliasing [Strasser, 1995]



z, z'
u, u'
A
$\overline{AB}$ = Objekt mit Schachbrettextur
$\overline{A''B''}$ = perspektivisch transformiertes Objekt
$\overline{A'B'}$ = perspektivisch projiziertes Objekt
10 Texel
5 Texel
3 Texel
2 Texel
1 Texel
luchtpunkt
z_0
A''
, u'
A'
B'
x, x'
Bildgerade (äquidistante Pixel)
B
B''
$-z_0$
Augpunkt

STRASSER, W. ~1995.
GRAPHISCHE DATENVERARBEIT
II.
p. 330

Abb. 7.4: Zusammenhänge bei der perspektivischen Projektion einer Textur

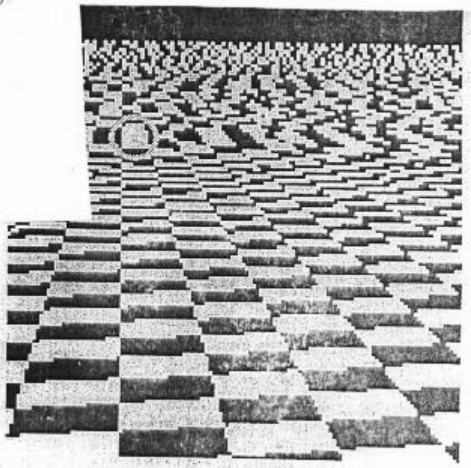
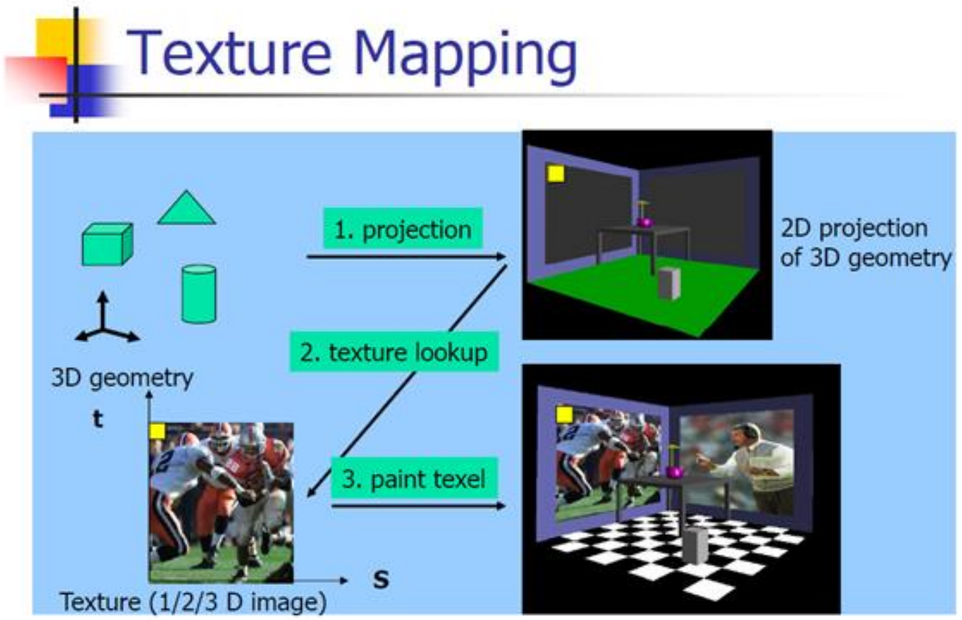


Abb. 7.10: Aliasing: Auftreten von Scheinstrukturen durch diskretes Abtasten

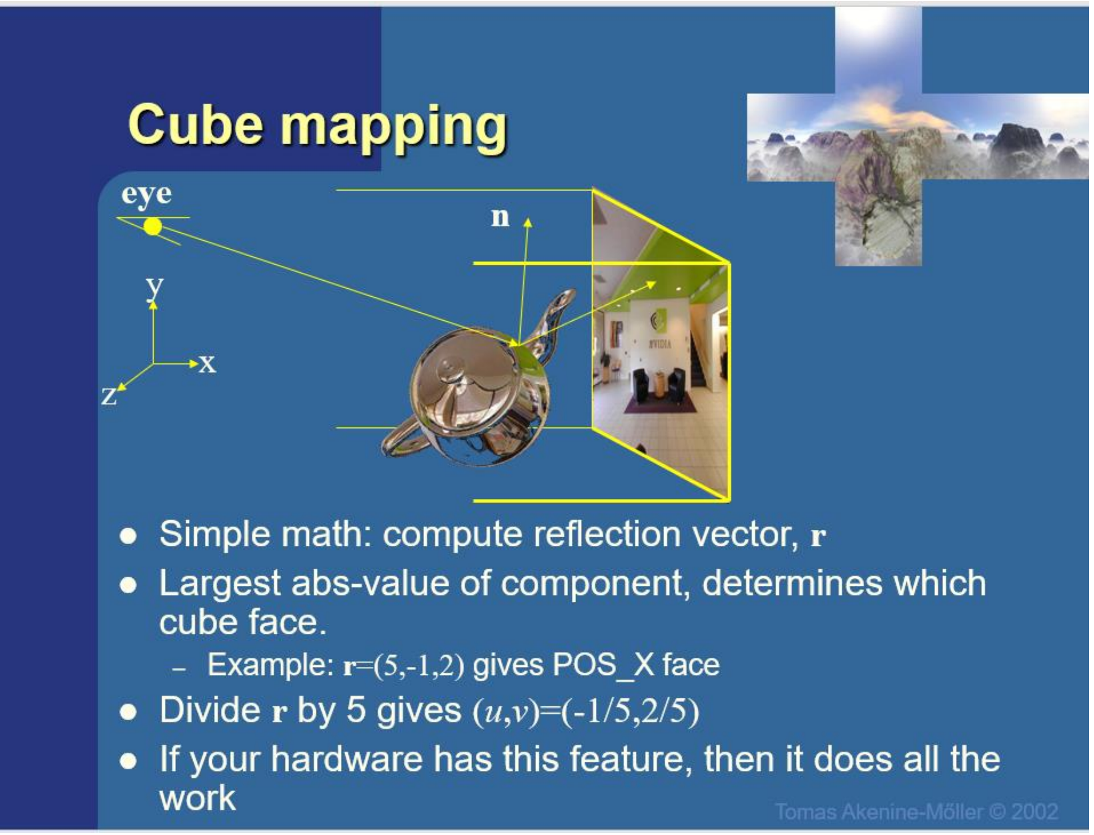
Visual Abstract + anim [wi]



Wang, Huamin. 2016. *"Texture Mapping"* @ https://en.wikipedia.org/wiki/Texture_mapping

Animation *Mapping a two-dimensional texture onto a 3D model*

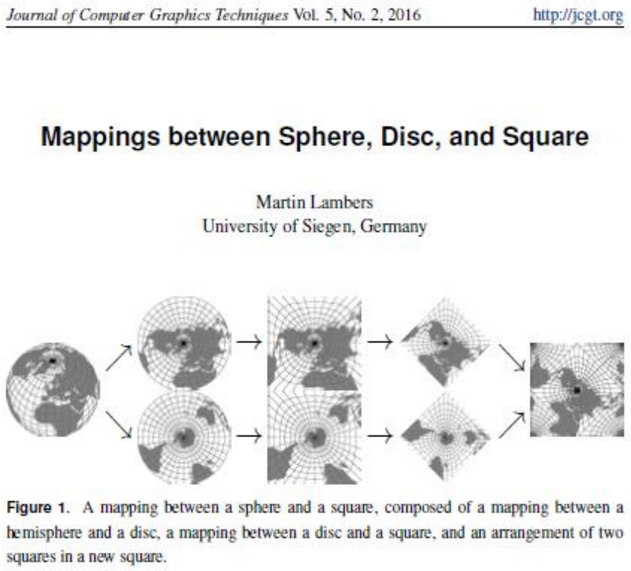
Cube Mapping [Akenine-Moller, RTR]



Cube mapping

- Simple math: compute reflection vector, **r**
- Largest abs-value of component, determines which cube face.
  - Example: **r**=(5,-1,2) gives POS_X face
- Divide **r** by 5 gives $(u,v)=(-1/5,2/5)$
- If your hardware has this feature, then it does all the work

Tomas Akenine-Möller © 2002



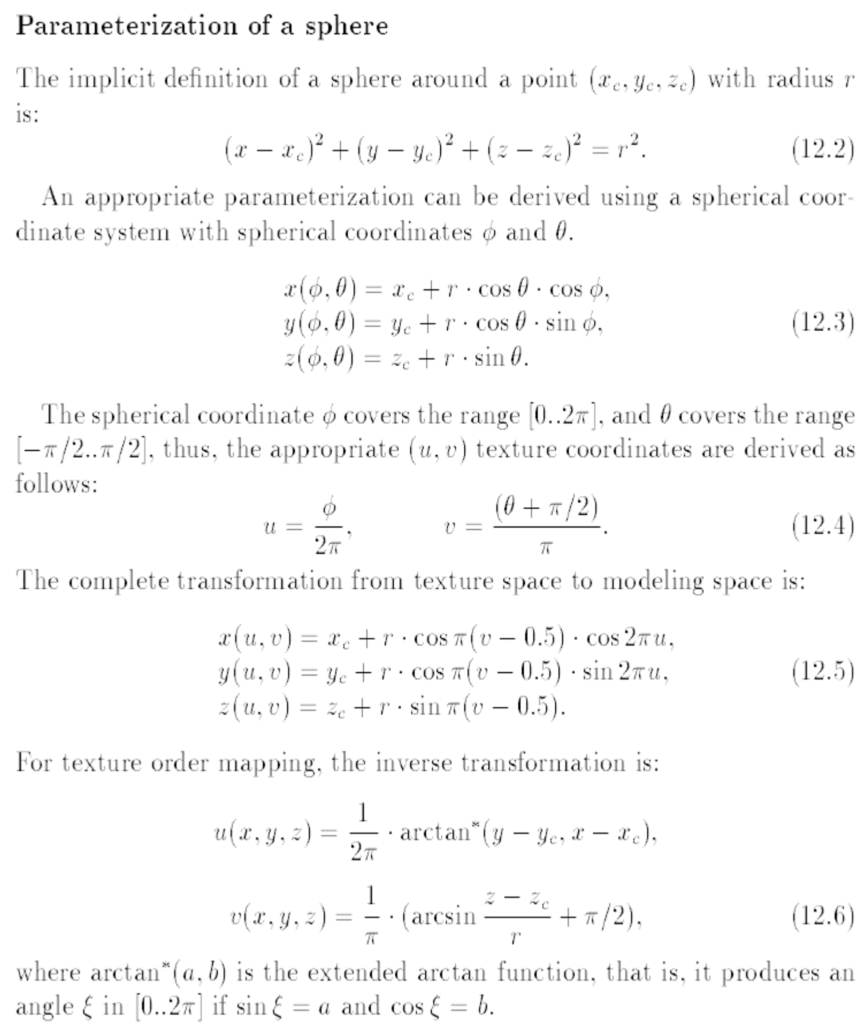
Journal of Computer Graphics Techniques Vol. 5, No. 2, 2016 http://jcgt.org

Mappings between Sphere, Disc, and Square

Martin Lambers
University of Siegen, Germany

Figure 1. A mapping between a sphere and a square, composed of a mapping between a hemisphere and a disc, a mapping between a disc and a square, and an arrangement of two squares in a new square.

Sphere Param **[Szirmay-Kalos]**



Parameterization of a sphere

The implicit definition of a sphere around a point (x_c, y_c, z_c) with radius r is:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2. \tag{12.2}$$

An appropriate parameterization can be derived using a spherical coordinate system with spherical coordinates ϕ and θ.

$$\begin{aligned} x(\phi, \theta) &= x_c + r \cdot \cos\theta \cdot \cos\phi, \\ y(\phi, \theta) &= y_c + r \cdot \cos\theta \cdot \sin\phi, \\ z(\phi, \theta) &= z_c + r \cdot \sin\theta. \end{aligned} \tag{12.3}$$

The spherical coordinate ϕ covers the range $[0..2\pi]$, and θ covers the range $[-\pi/2..\pi/2]$, thus, the appropriate (u, v) texture coordinates are derived as follows:

$$u = \frac{\phi}{2\pi}, \qquad v = \frac{(\theta + \pi/2)}{\pi}. \tag{12.4}$$

The complete transformation from texture space to modeling space is:

$$\begin{aligned} x(u, v) &= x_c + r \cdot \cos\pi(v - 0.5) \cdot \cos 2\pi u, \\ y(u, v) &= y_c + r \cdot \cos\pi(v - 0.5) \cdot \sin 2\pi u, \\ z(u, v) &= z_c + r \cdot \sin\pi(v - 0.5). \end{aligned} \tag{12.5}$$

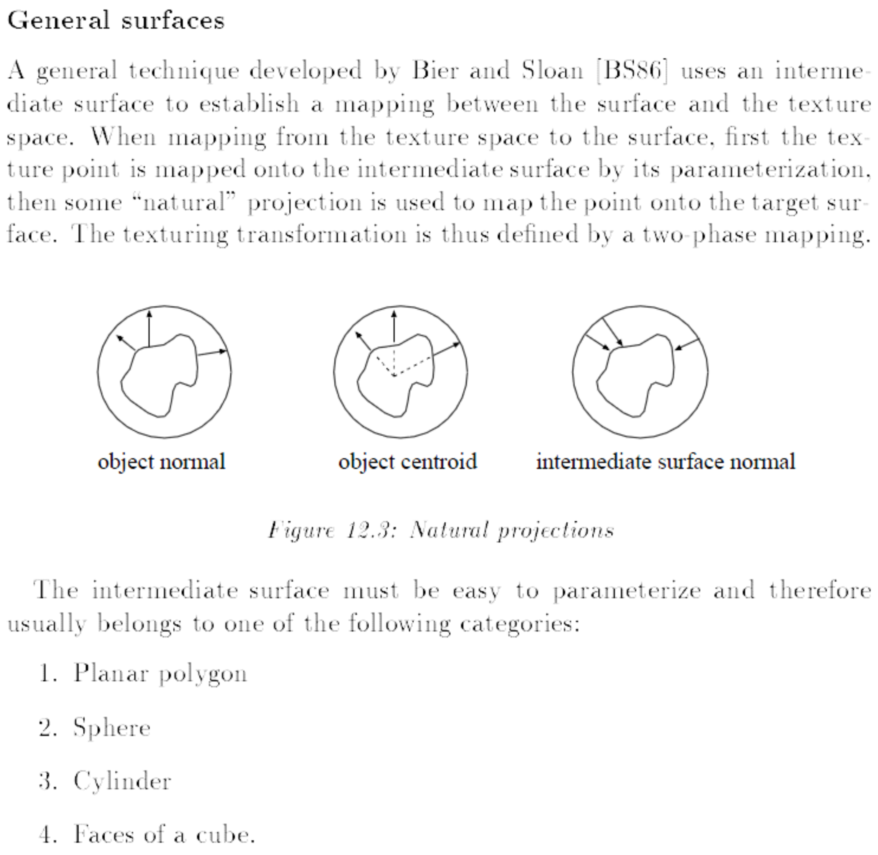
For texture order mapping, the inverse transformation is:

$$\begin{aligned} u(x, y, z) &= \frac{1}{2\pi} \cdot \arctan^*(y - y_c, x - x_c), \\ v(x, y, z) &= \frac{1}{\pi} \cdot (\arcsin\frac{z - z_c}{r} + \pi/2), \end{aligned} \tag{12.6}$$

where $\arctan^*(a, b)$ is the extended arctan function, that is, it produces an angle ξ in $[0..2\pi]$ if $\sin\xi = a$ and $\cos\xi = b$.

Grimm & Niebruegge. 2007.
Continuous Cube Mapping.
Journal of Graphics GPU and Game Tools 12(4):25-34

Intermediate Surface [SK]



General surfaces

A general technique developed by Bier and Sloan [BS86] uses an intermediate surface to establish a mapping between the surface and the texture space. When mapping from the texture space to the surface, first the texture point is mapped onto the intermediate surface by its parameterization, then some "natural" projection is used to map the point onto the target surface. The texturing transformation is thus defined by a two-phase mapping.

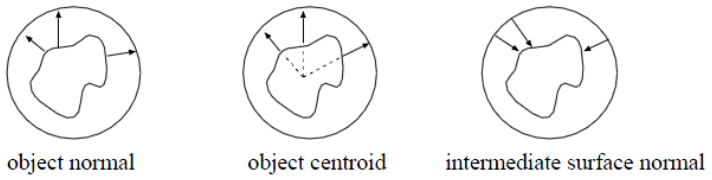


Figure 12.3: Natural projections

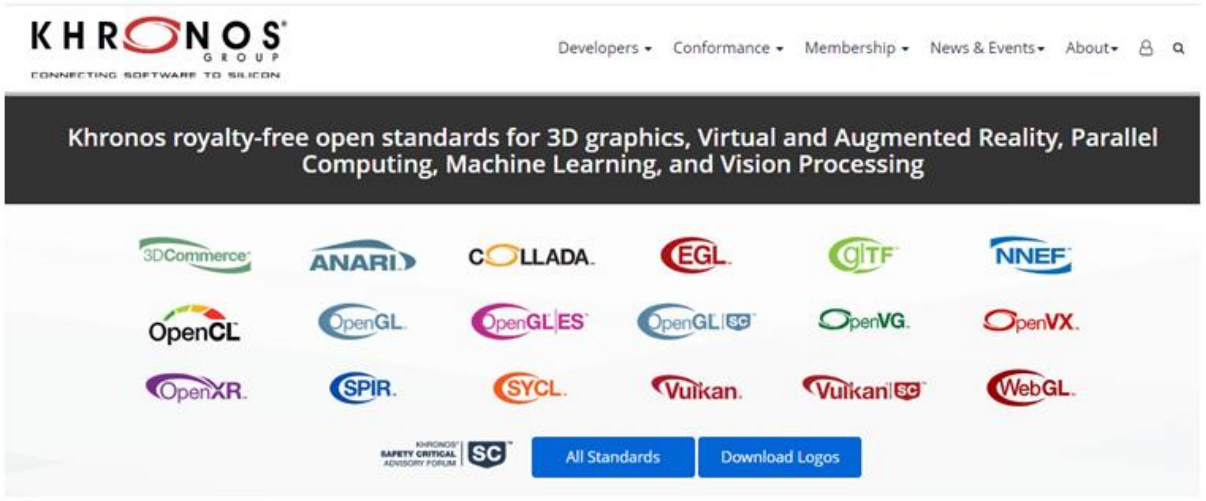
The intermediate surface must be easy to parameterize and therefore usually belongs to one of the following categories:

1. Planar polygon
2. Sphere
3. Cylinder
4. Faces of a cube.

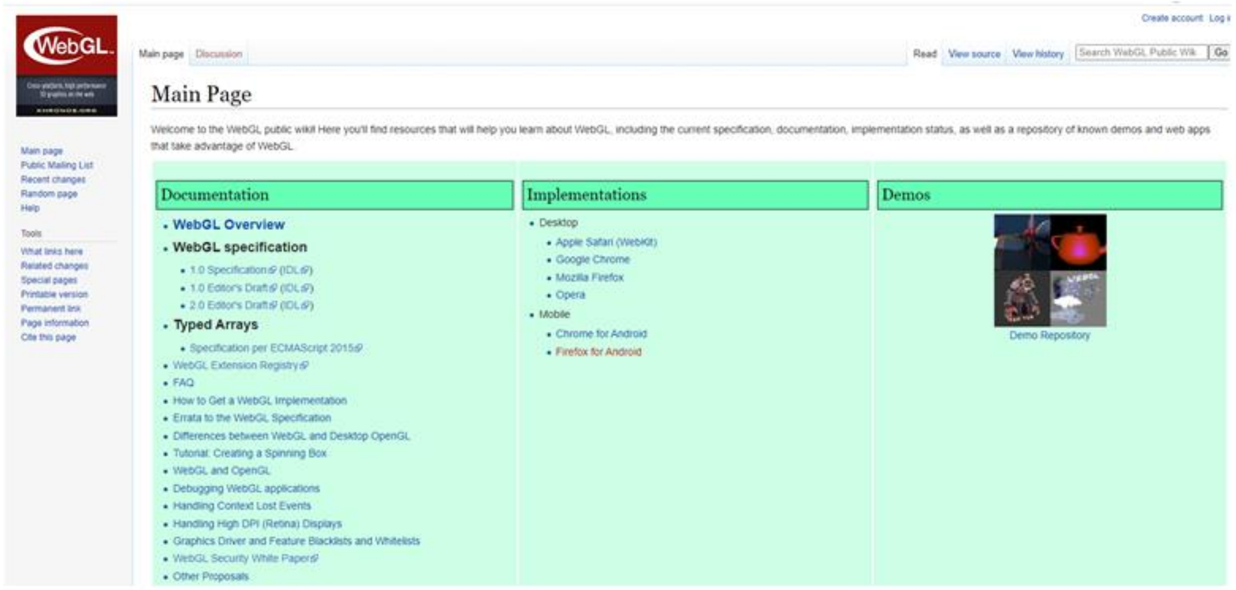
Grimm & Niebruegge. 2007.
Continuous Cube Mapping.
Journal of Graphics GPU and Game Tools 12(4):25-34

VULKAN, WebGL, Collada…

VULKAN 2018 https://www.khronos.org/registry/vulkan/specs/1.1/refguide/Vulkan-1.1-web.pdf



Khronos royalty-free open standards for 3D graphics, Virtual and Augmented Reality, Parallel Computing, Machine Learning, and Vision Processing



Main Page

Welcome to the WebGL public wiki! Here you'll find resources that will help you learn about WebGL, including the current specification, documentation, implementation status, as well as a repository of known demos and web apps that take advantage of WebGL.

Documentation

- **WebGL Overview**
- **WebGL specification**
  - 1.0 Specification (IDL)
  - 1.0 Editor's Draft (IDL)
  - 2.0 Editor's Draft (IDL)
- **Typed Arrays**
  - Specification per ECMAScript 2015
- WebGL Extension Registry
- FAQ
- How to Get a WebGL Implementation
- Errata to the WebGL Specification
- Differences between WebGL and Desktop OpenGL
- Tutorial: Creating a Spinning Box
- WebGL and OpenGL
- Debugging WebGL applications
- Handling Context Lost Events
- Handling High DPI (Retina) Displays
- Graphics Driver and Feature Blacklists and Whitelists
- WebGL Security White Paper
- Other Proposals

Implementations

- Desktop
  - Apple Safari (WebKit)
  - Google Chrome
  - Mozilla Firefox
  - Opera
- Mobile
  - Chrome for Android
  - Firefox for Android

Demos

Demo Repository

24 Content Descriptors in MPEG-7

- **Color 7 – space, quantization, dominant, scalable, layout…**
- **Texture descriptors 3 – homogenous, browsing, edge histogram**
- **Shape descriptors 3 – region shape, contour shape, shape 3D**
- **Motion 4 – camera, trajectory, parametric motion, action**
- **Others 2 – localization, face**
- **Audio 5 – signature, instrument, melody, indexing, spoken**
- **In total, 10 radiometric, 9 geometric, 5 others**

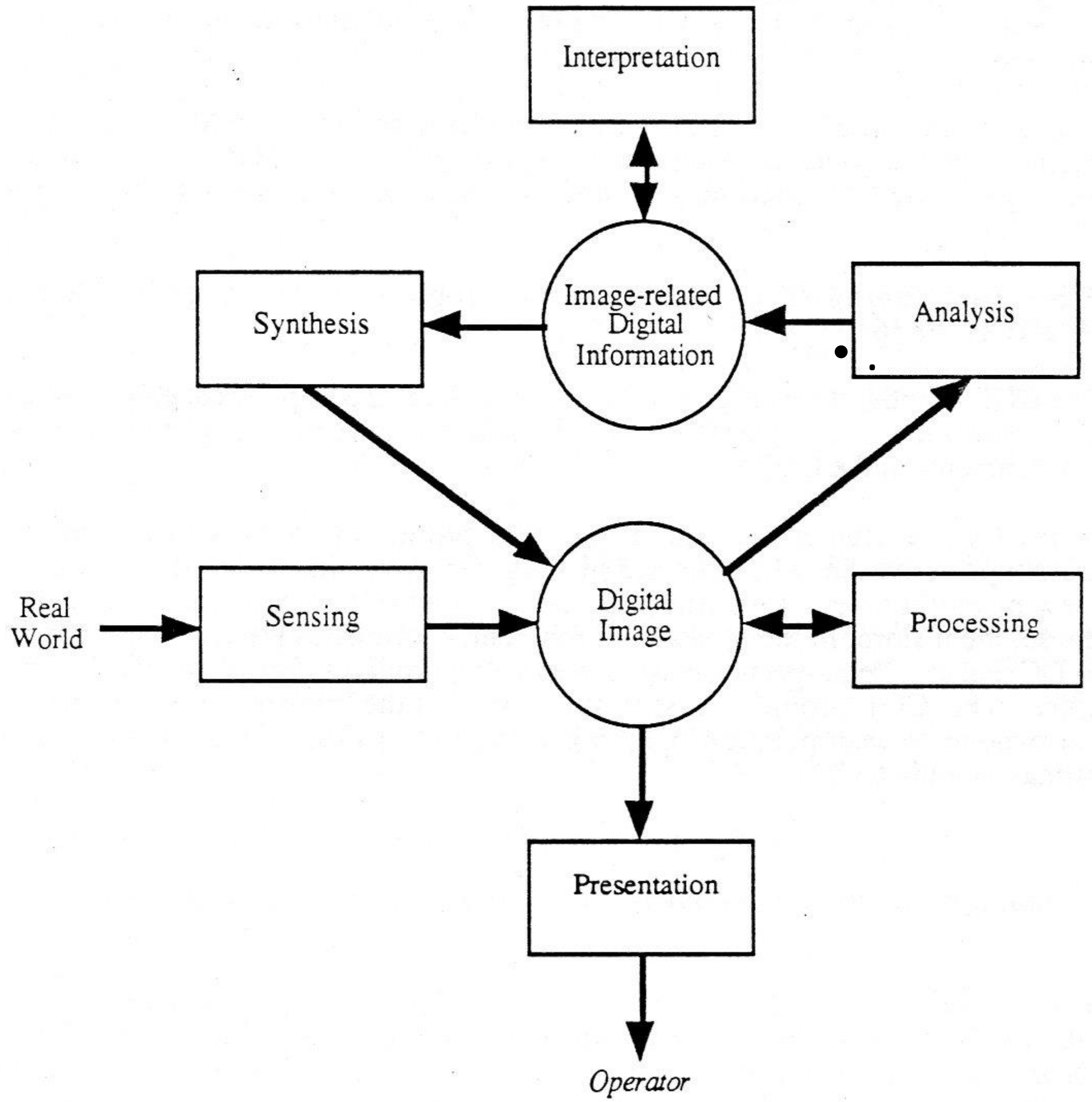


Figure B.1 – Computer imaging model

Real-time ? >> IBR, IBL

- Image based rendering, no model, plenoptic function
- Image based illumination, no simulation, just data, Debevec

Computer Graphics 3, texturing

Andrej Ferko

ferko@fmph.uniba.sk

Short version 2020, www.sccg.sk/PG3, v281020