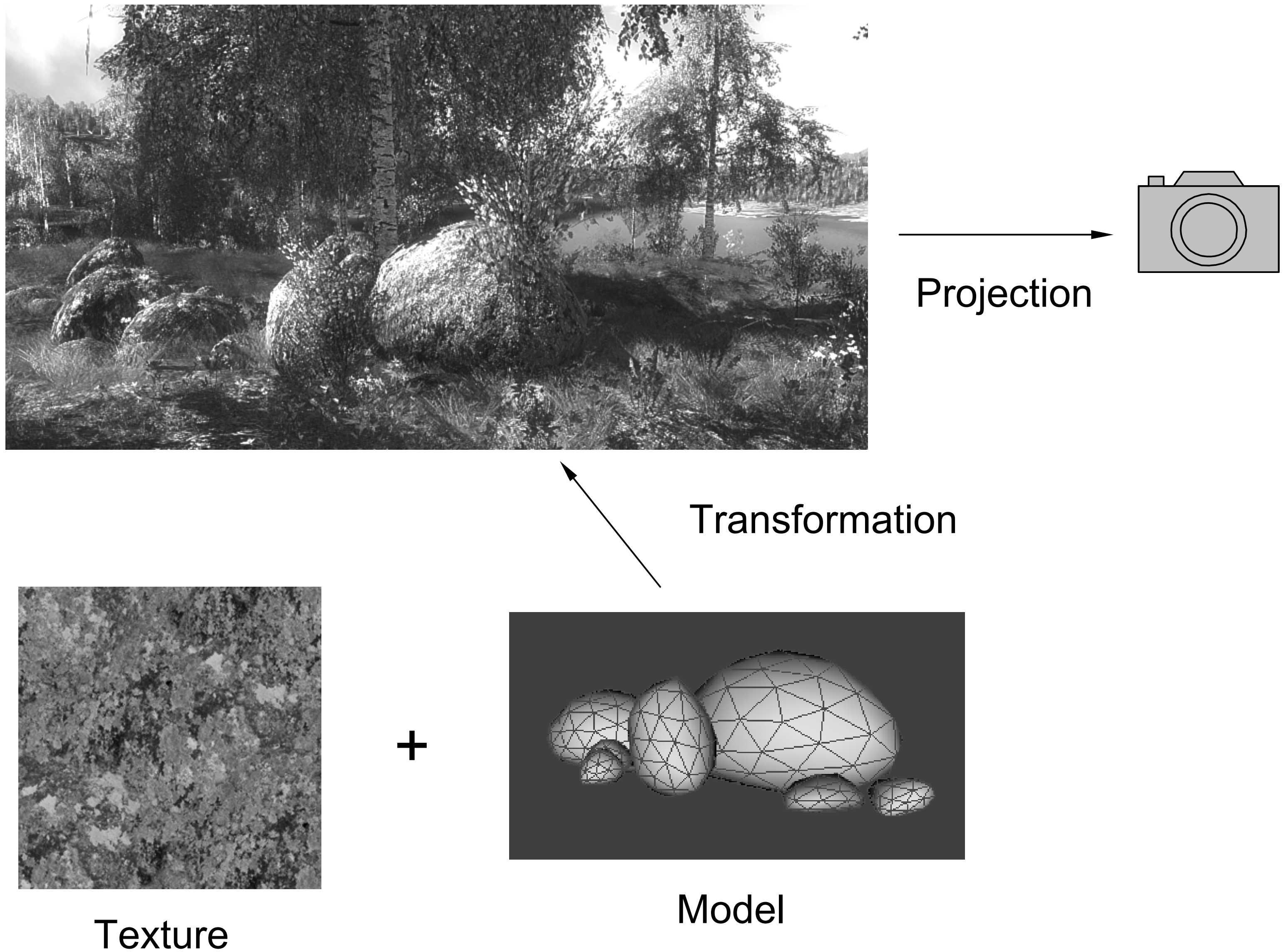
OpenGL

Gustav Taxén
Associate Professor
School of Computer Science and Communication

gustavt@csc.kth.se

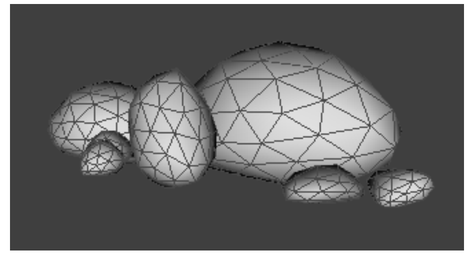
Terrain Demo,
Avalanche Studios

3D graphics systems



Overview of OpenGL

Geometry

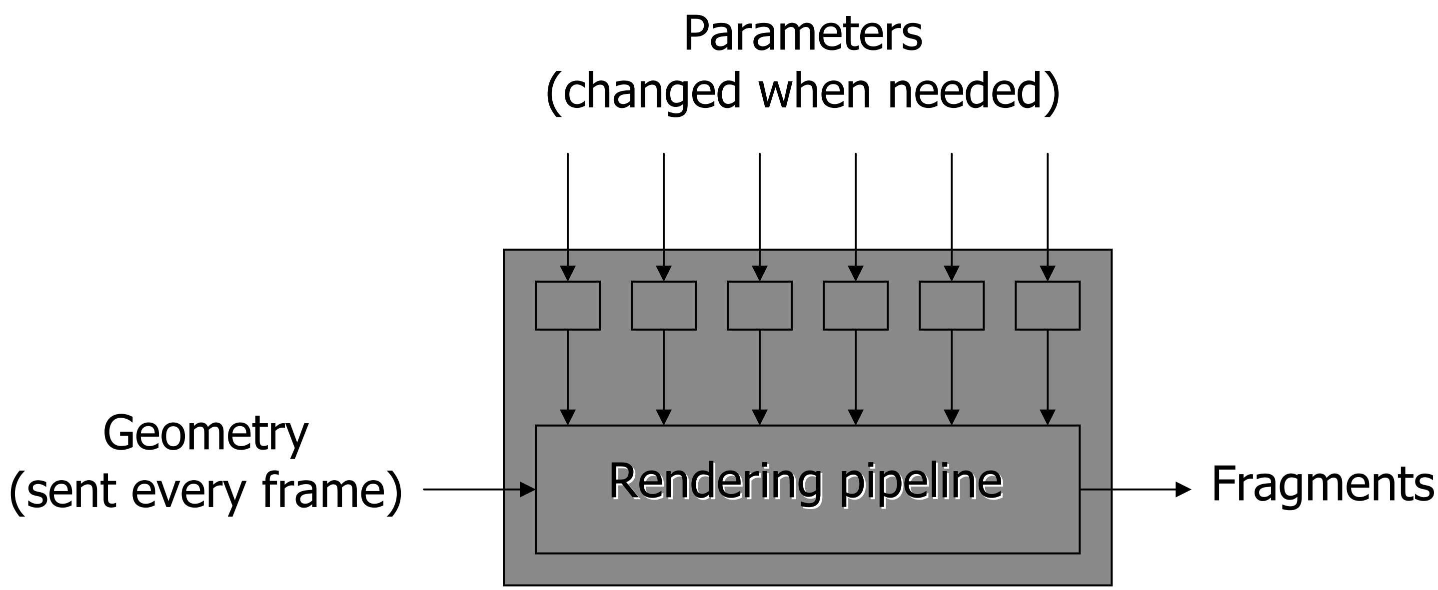


Parameters

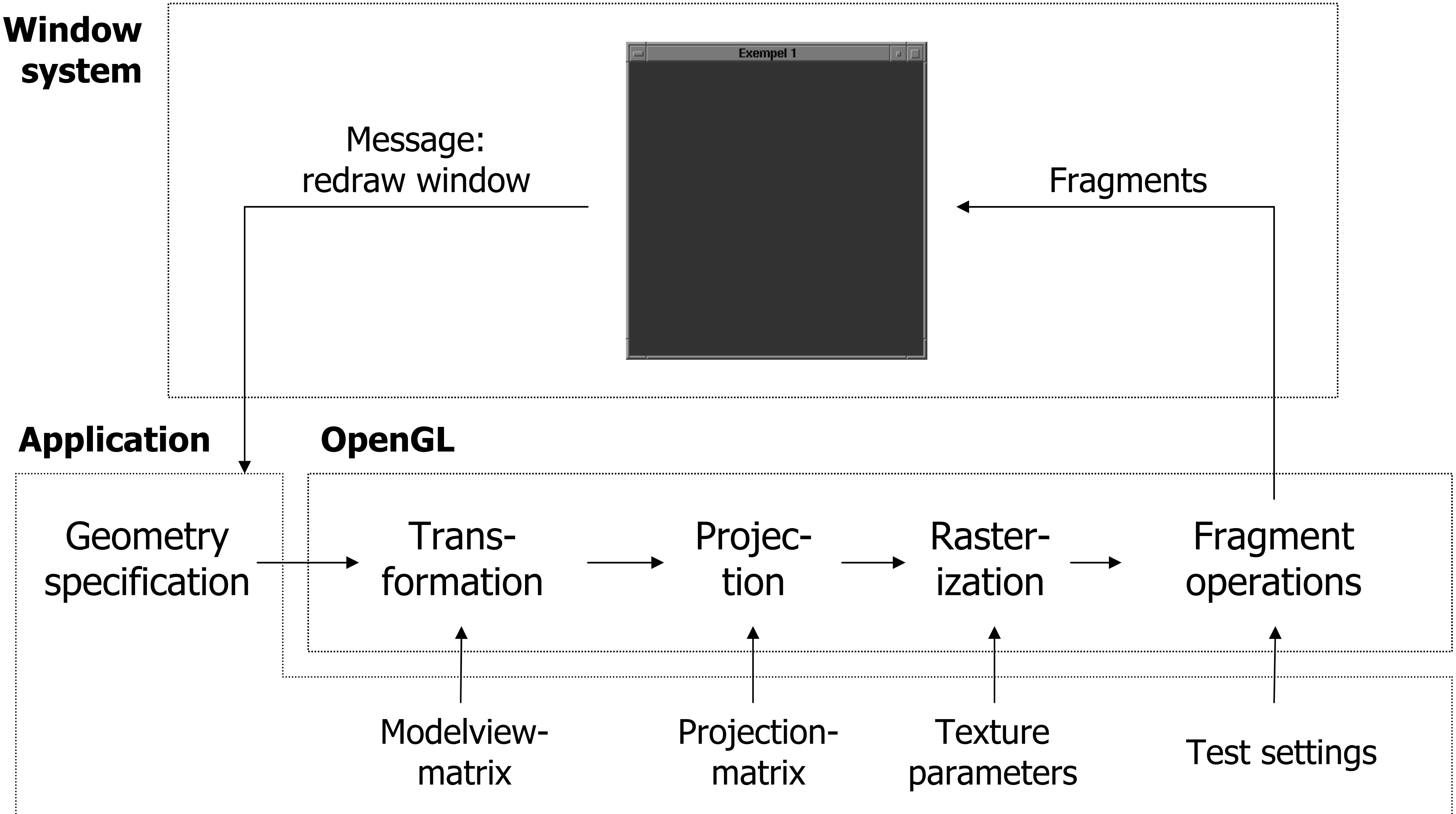
OpenGL

Fragments

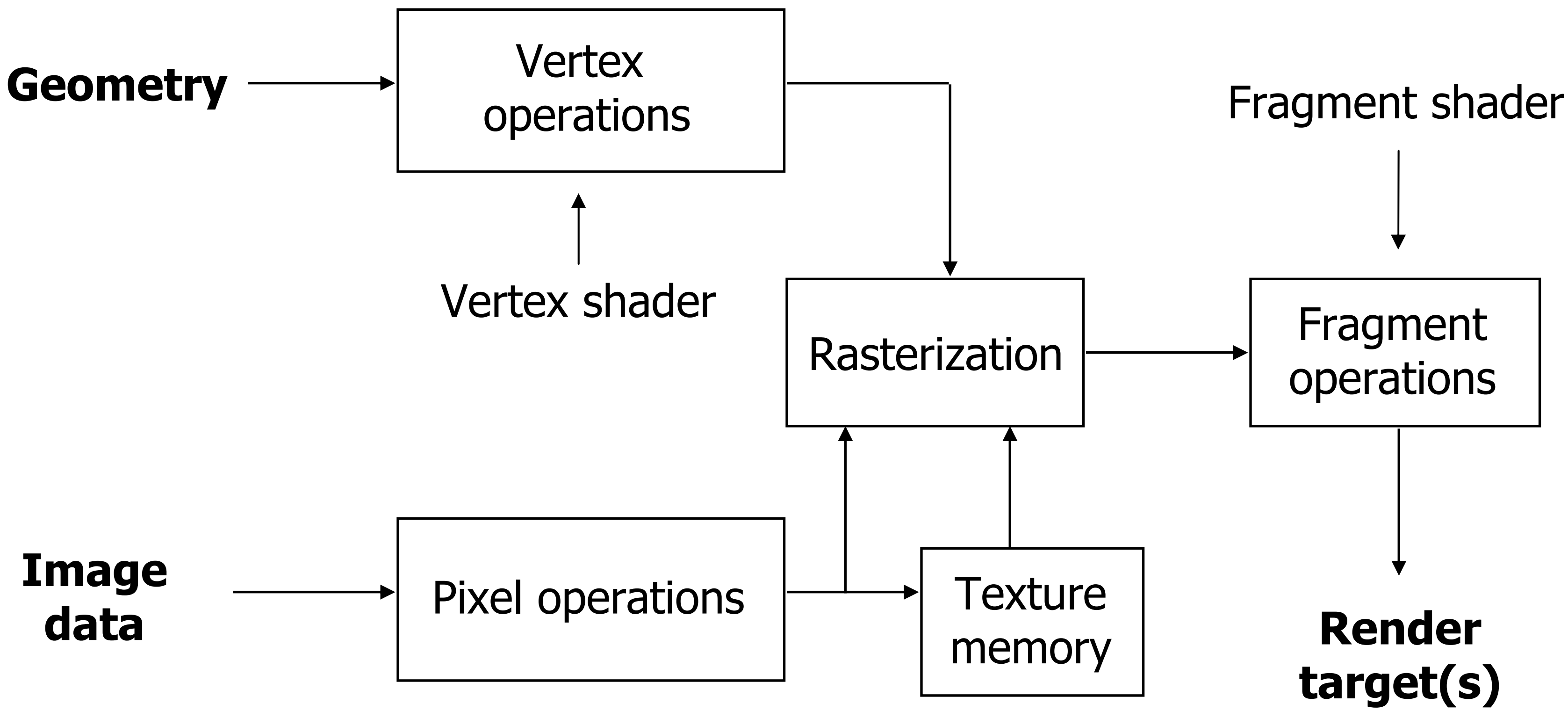
Overview of OpenGL



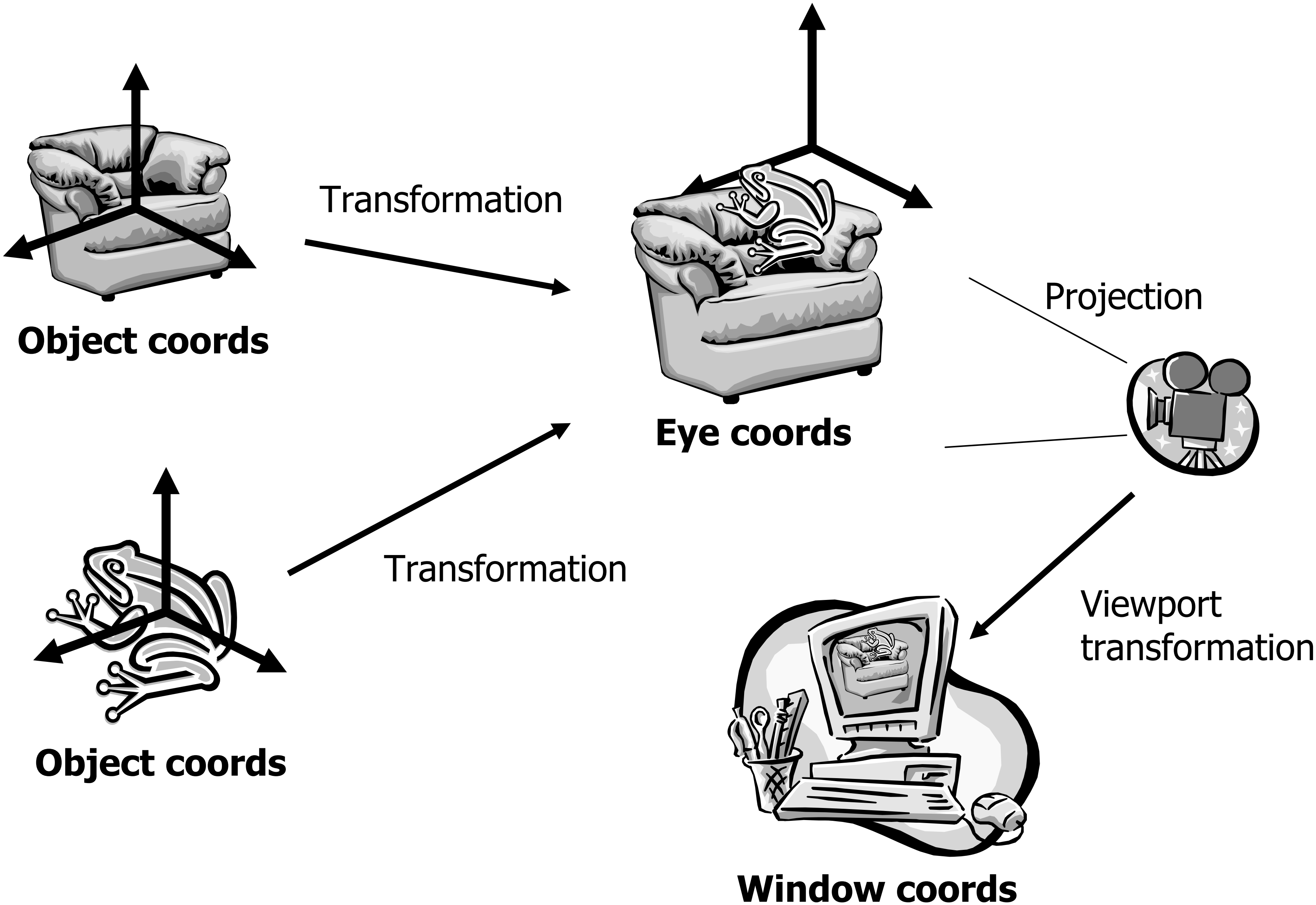
Overview of OpenGL



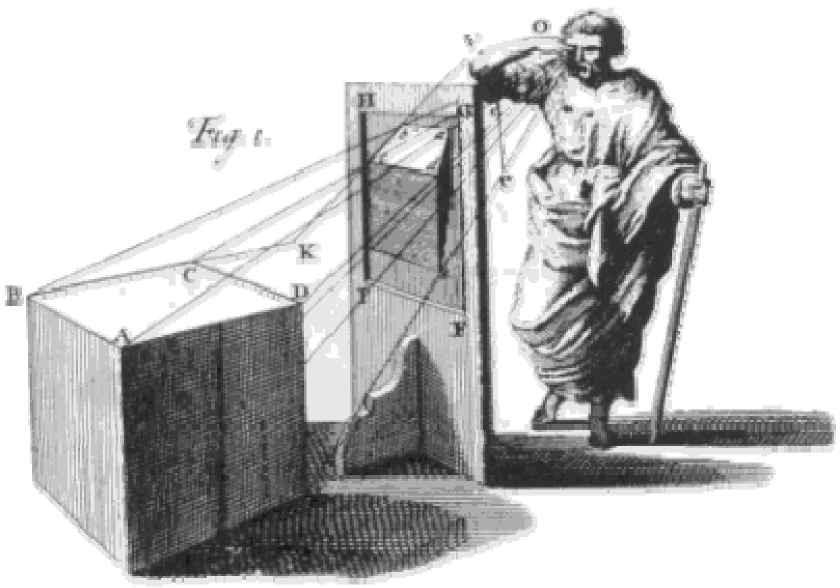
Overview of OpenGL



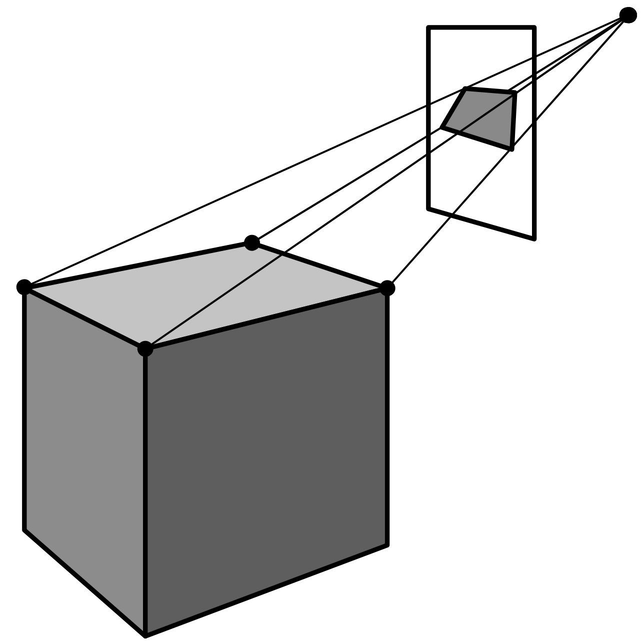
Transformations



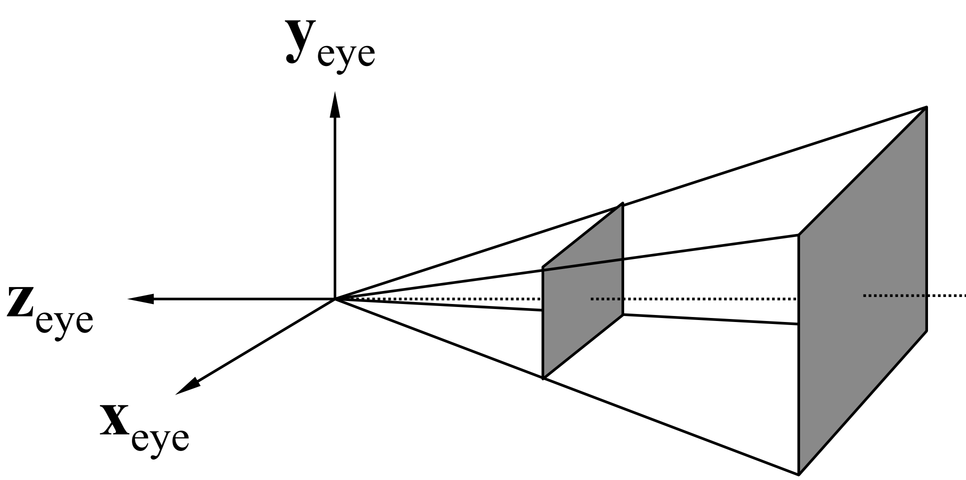
One point perspective

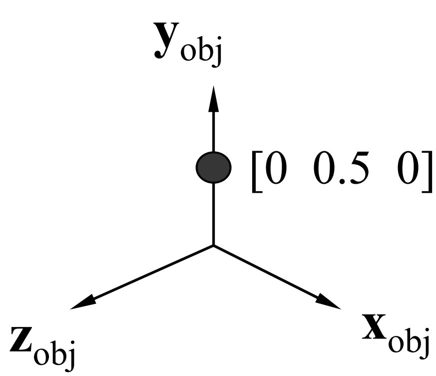


Brook Taylor, **New Principles of Linear Perspective**, 1719.



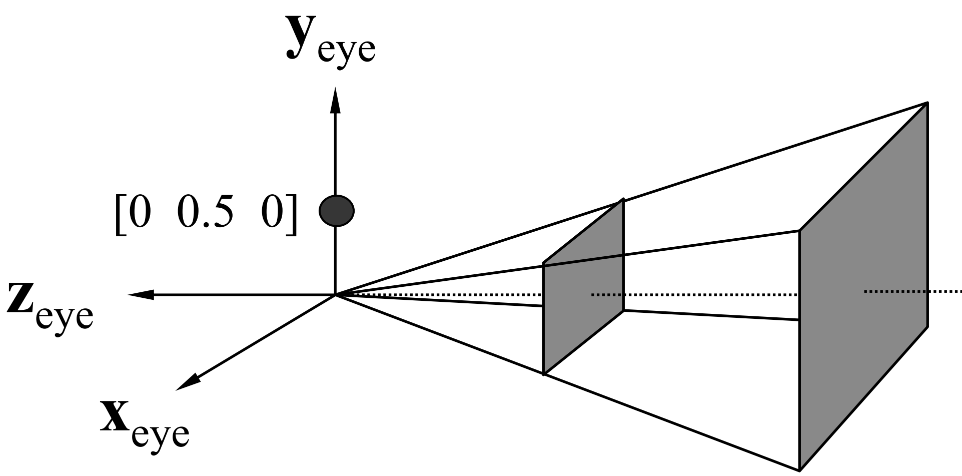
The OpenGL camera

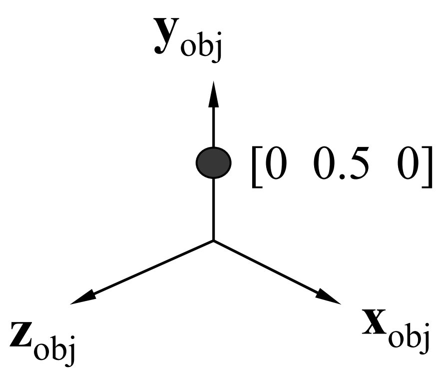




$$\mathbf{p}_{eye} = \mathbf{M}\ \mathbf{p}_{obj}$$

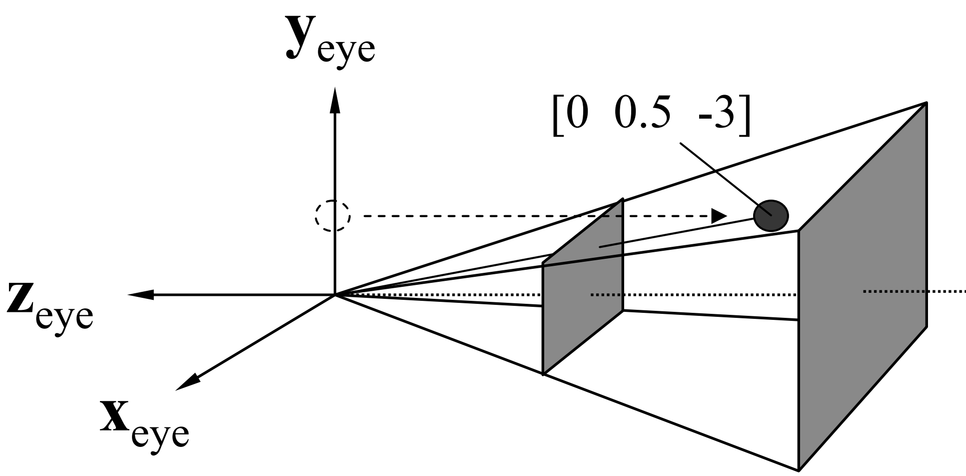
If $\mathbf{M}$ is the identity matrix...

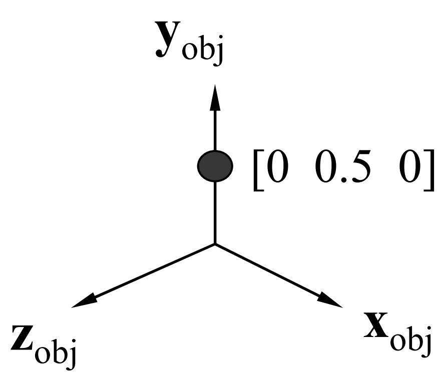




$$\mathbf{p}_{eye} = \mathbf{I}\,\mathbf{p}_{obj} = \mathbf{p}_{obj}$$

If $\mathbf{M}$ is a translation matrix...

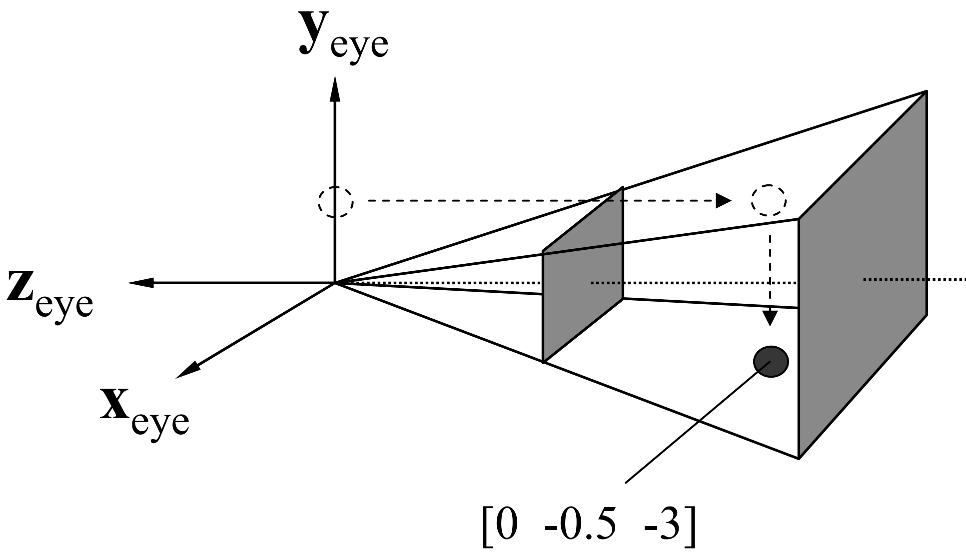




$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

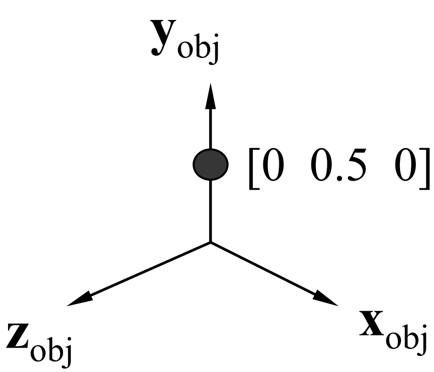
$$\mathbf{p}_{eye} = \mathbf{T}\,\mathbf{p}_{obj}$$

Concatenated transformations

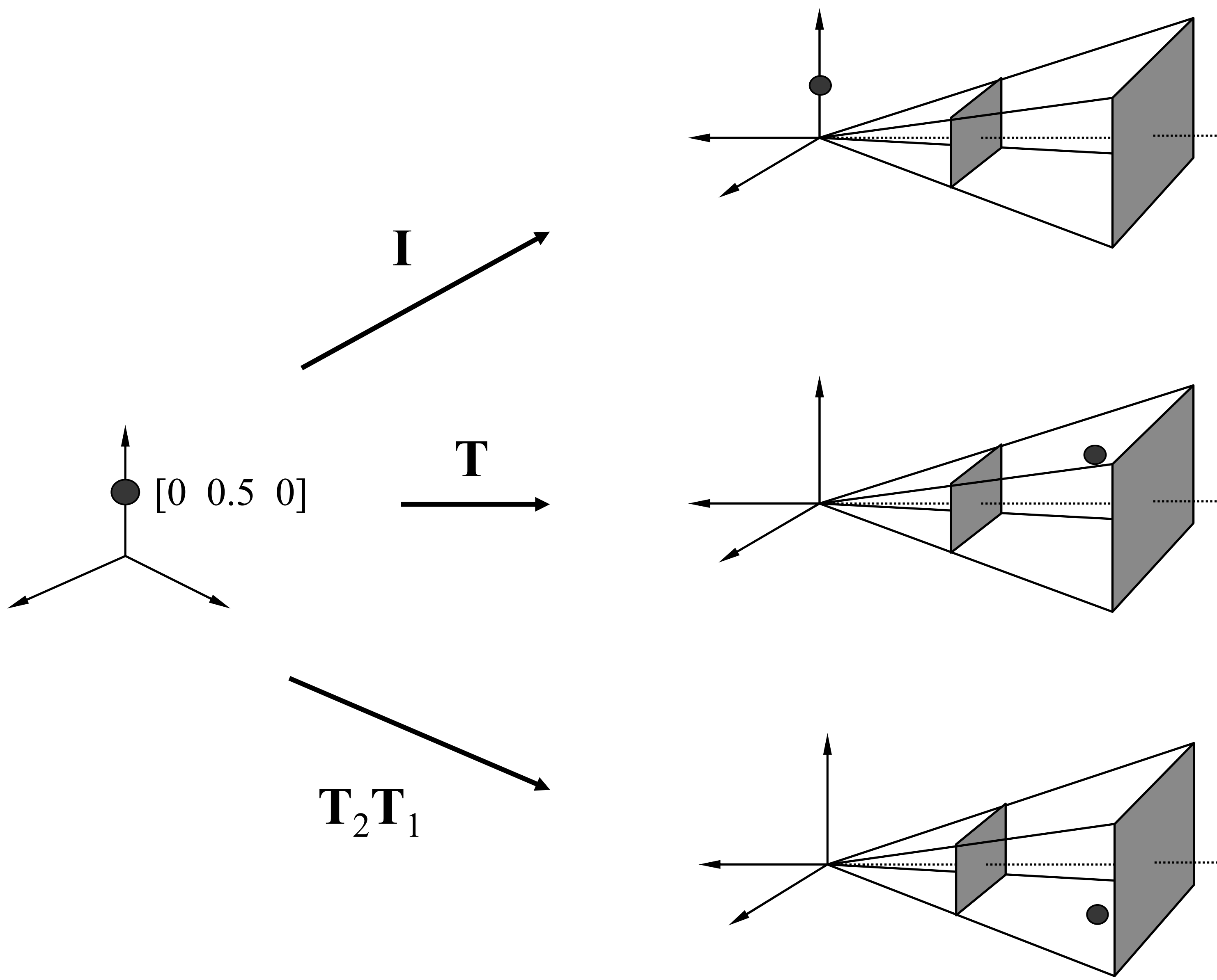


$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{p}_{eye} = \mathbf{T}_2\,(\mathbf{T}_1\,\mathbf{p}_{obj}) = (\mathbf{T}_2\,\mathbf{T}_1)\,\mathbf{p}_{obj}$$



[0 0.5 0]
I
T
T_2T_1

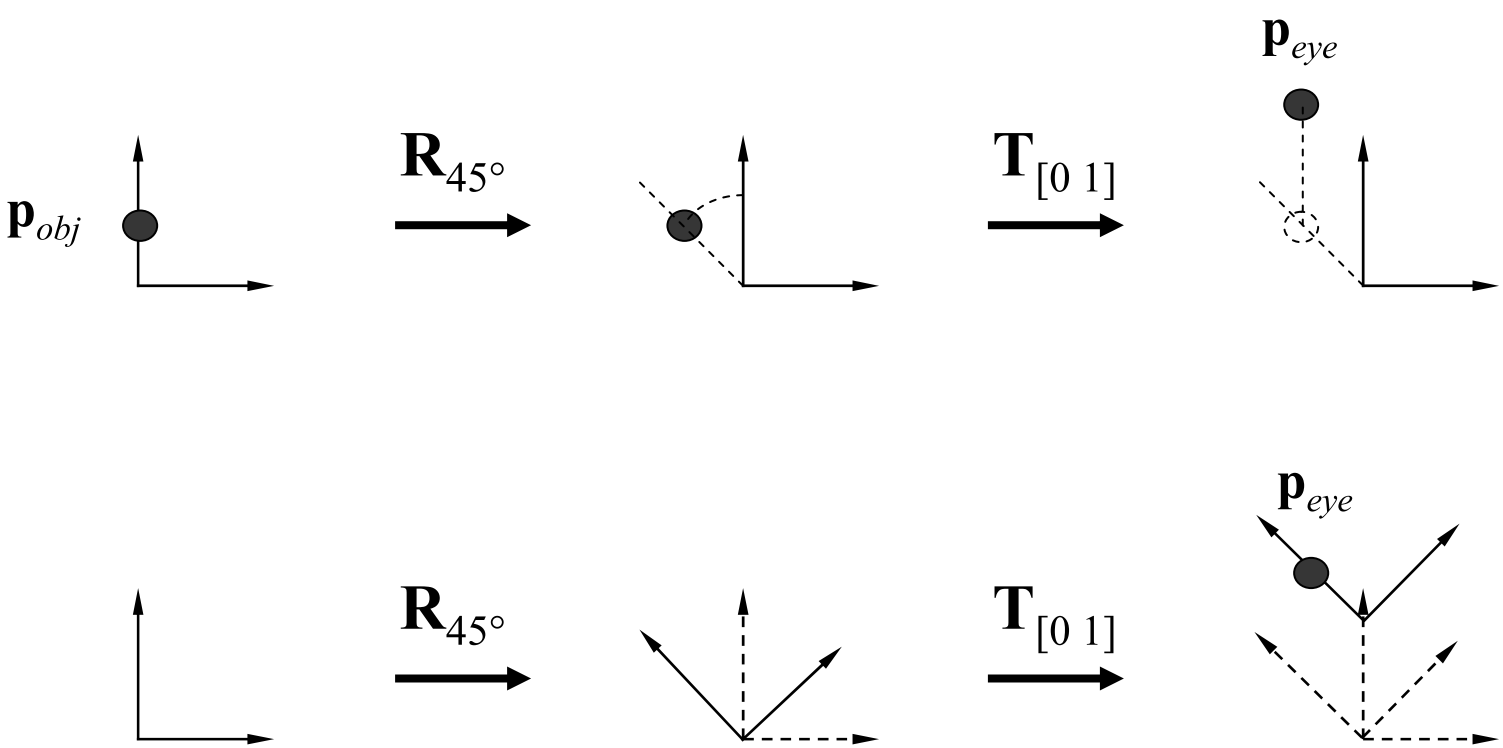
$$\mathbf{p}_{eye} = \mathbf{M}\ \mathbf{p}_{obj}$$

$\mathbf{M}$ defines where $\mathbf{p}_{obj}$ ends up in the ”world”

It is visible if it is ”in front of” the camera
(inside the clip volume)

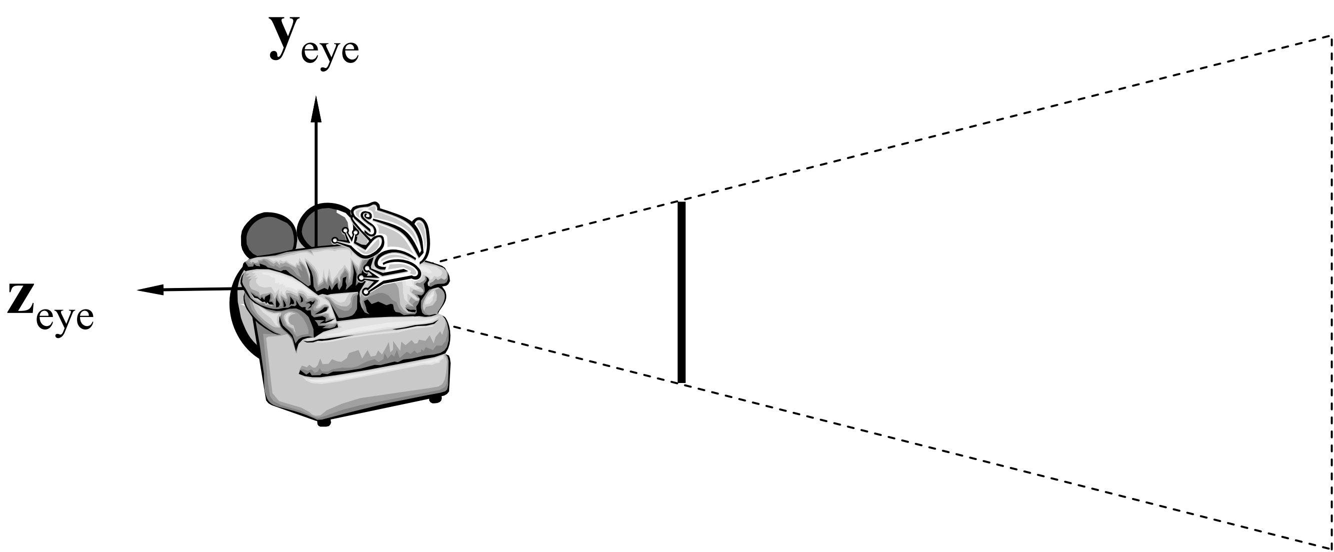
$\mathbf{M}$ is called the ”modelview matrix”

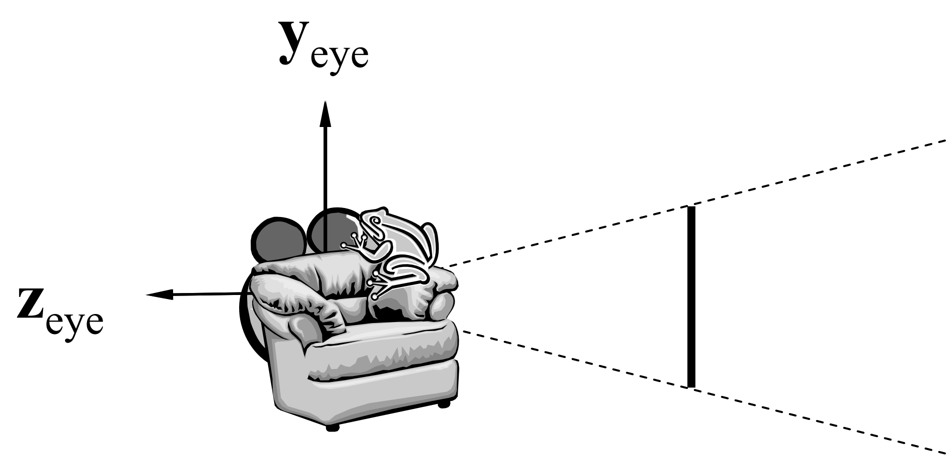
$$\mathbf{p}_{eye} = \mathbf{T}_{[0\ 1]}\,(\mathbf{R}_{45^\circ}\,\mathbf{p}_{obj})$$



A transformation is equivalent with a coordinate system change!

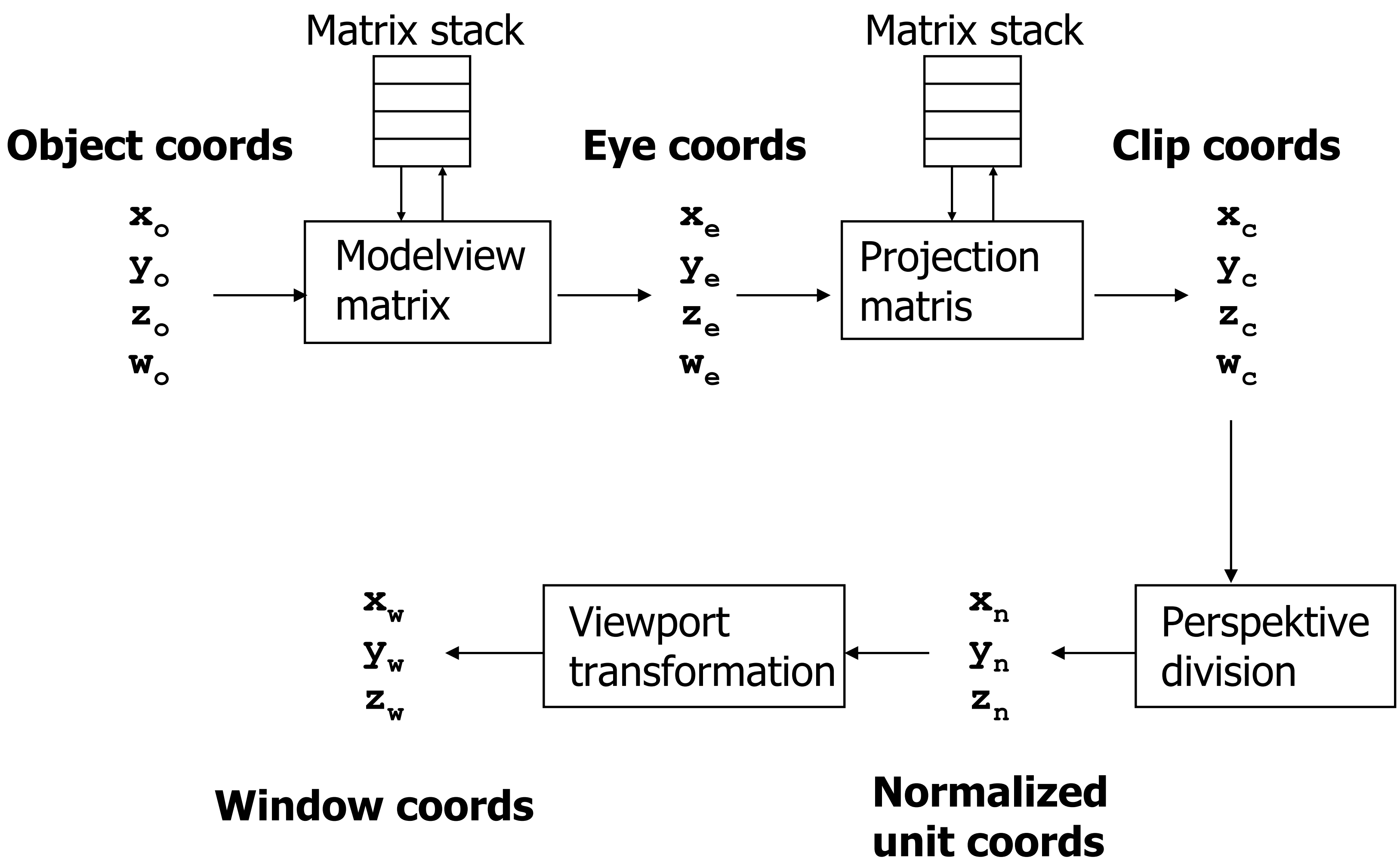
Camera placement



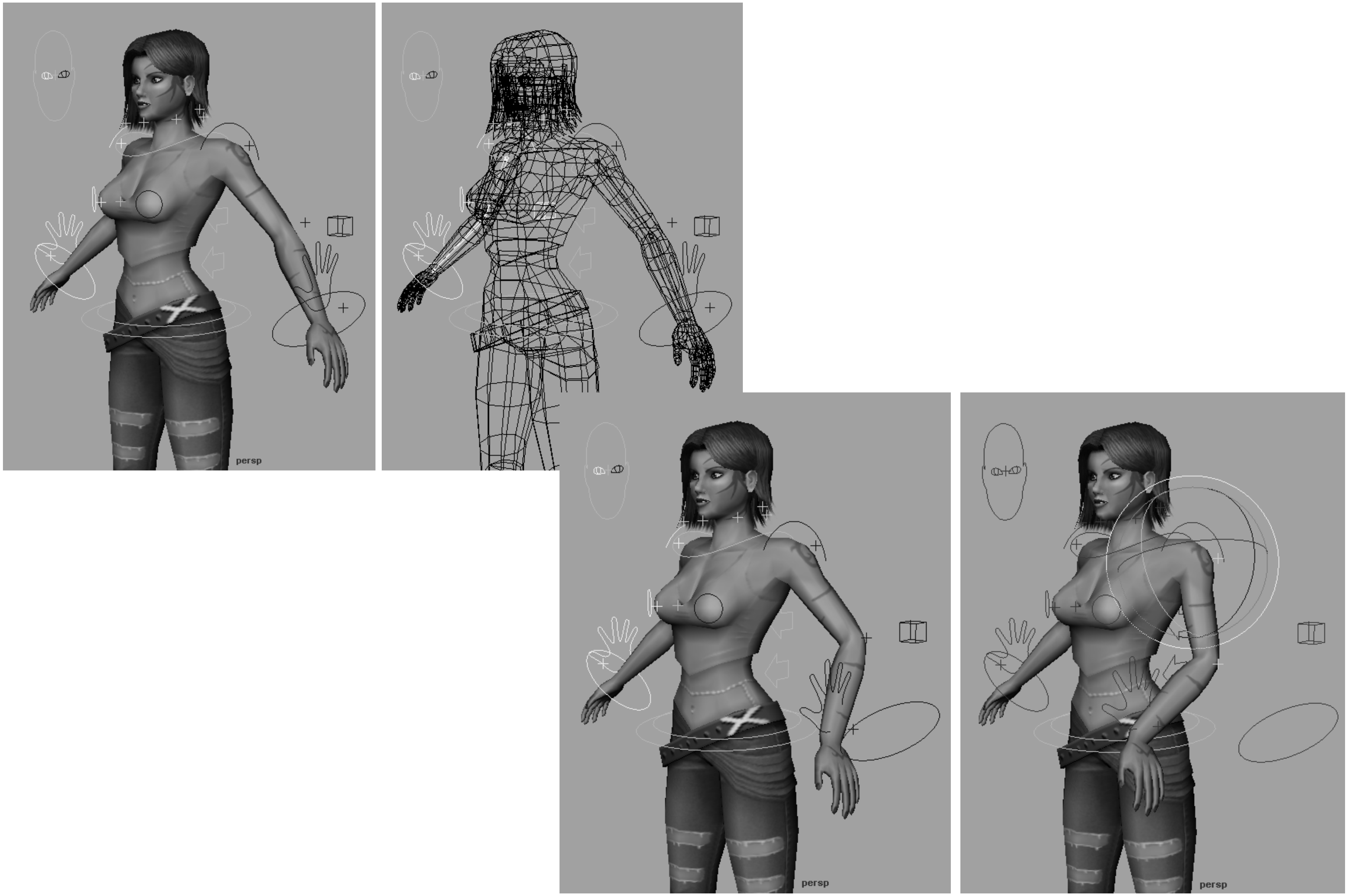


Moving the camera is like moving the world with an inverse transform!

Transformations in OpenGL

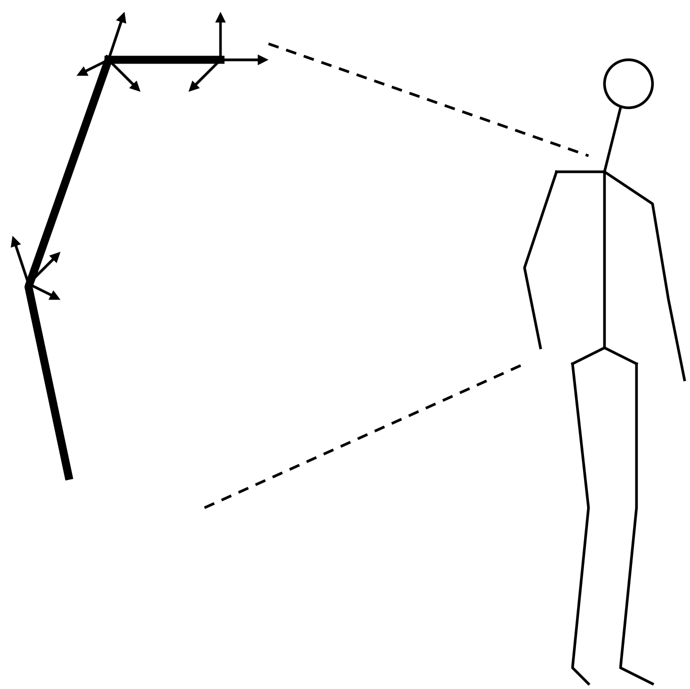


Skinning and hierarchical models



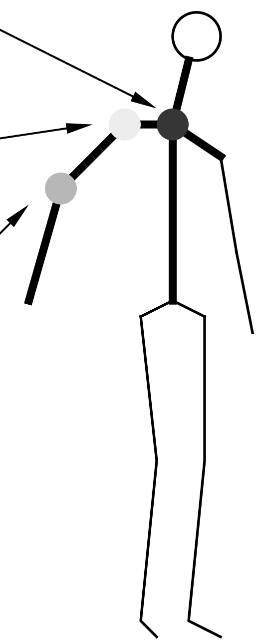
Hierarchical coordinate systems

Each child part needs to be offset/rotated/scaled
in relation to its parent

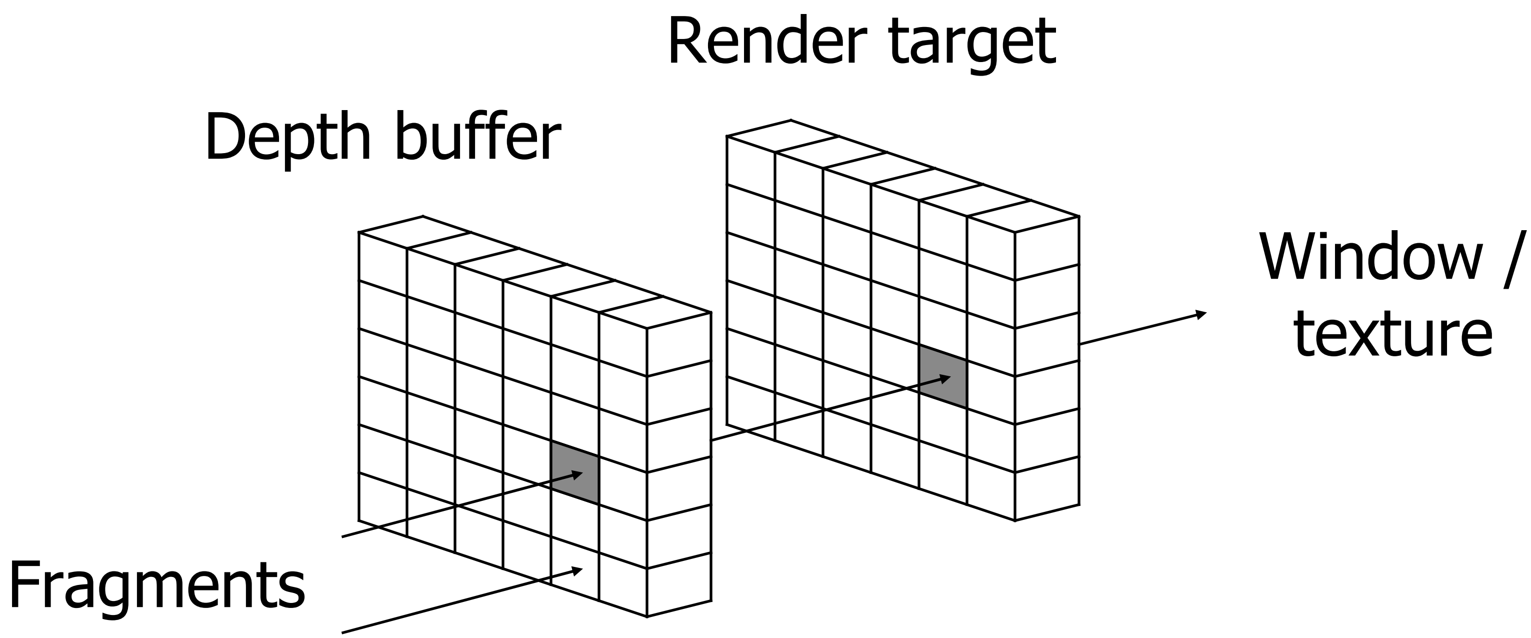


Hierarchical coordinate systems

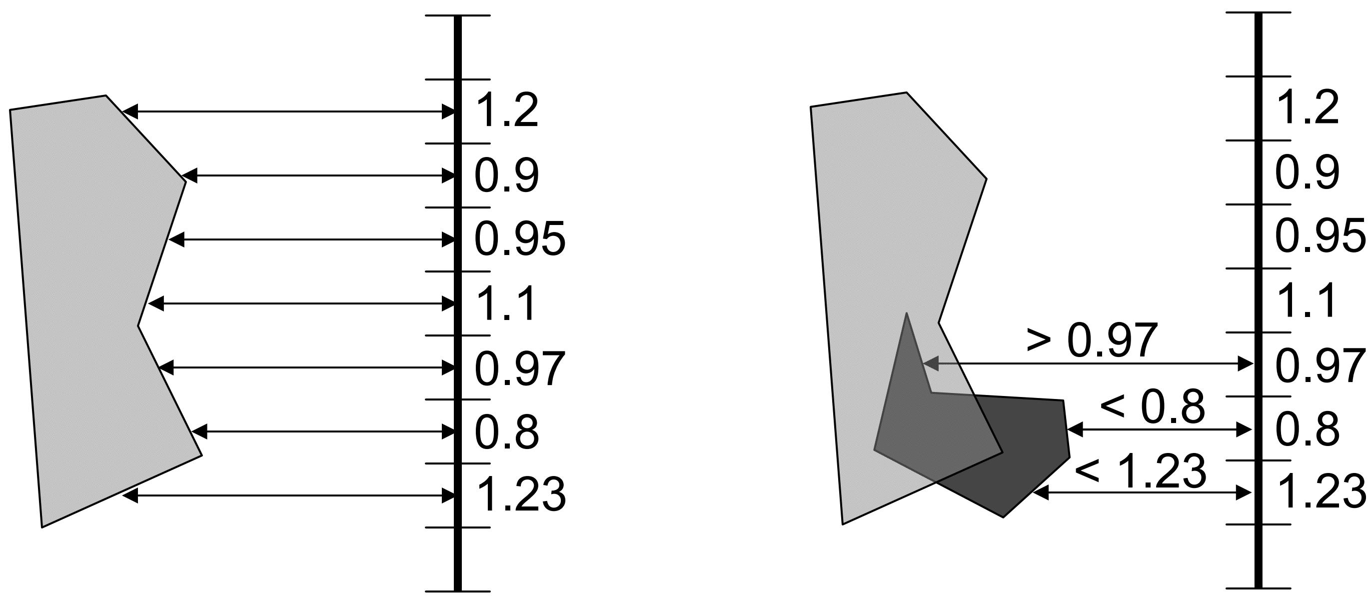
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

glPushMatrix();
glTranslatef(torso position);
draw torso;

  glPushMatrix();
  glTranslatef(pivot for upper arm);
  glRotatef(...);
  draw upper arm;

    glPushMatrix();
    glTranslatef(pivot for lower arm);
    glRotatef(...);
    draw lower arm;
    glPopMatrix();

  glPopMatrix();

glPopMatrix();
```



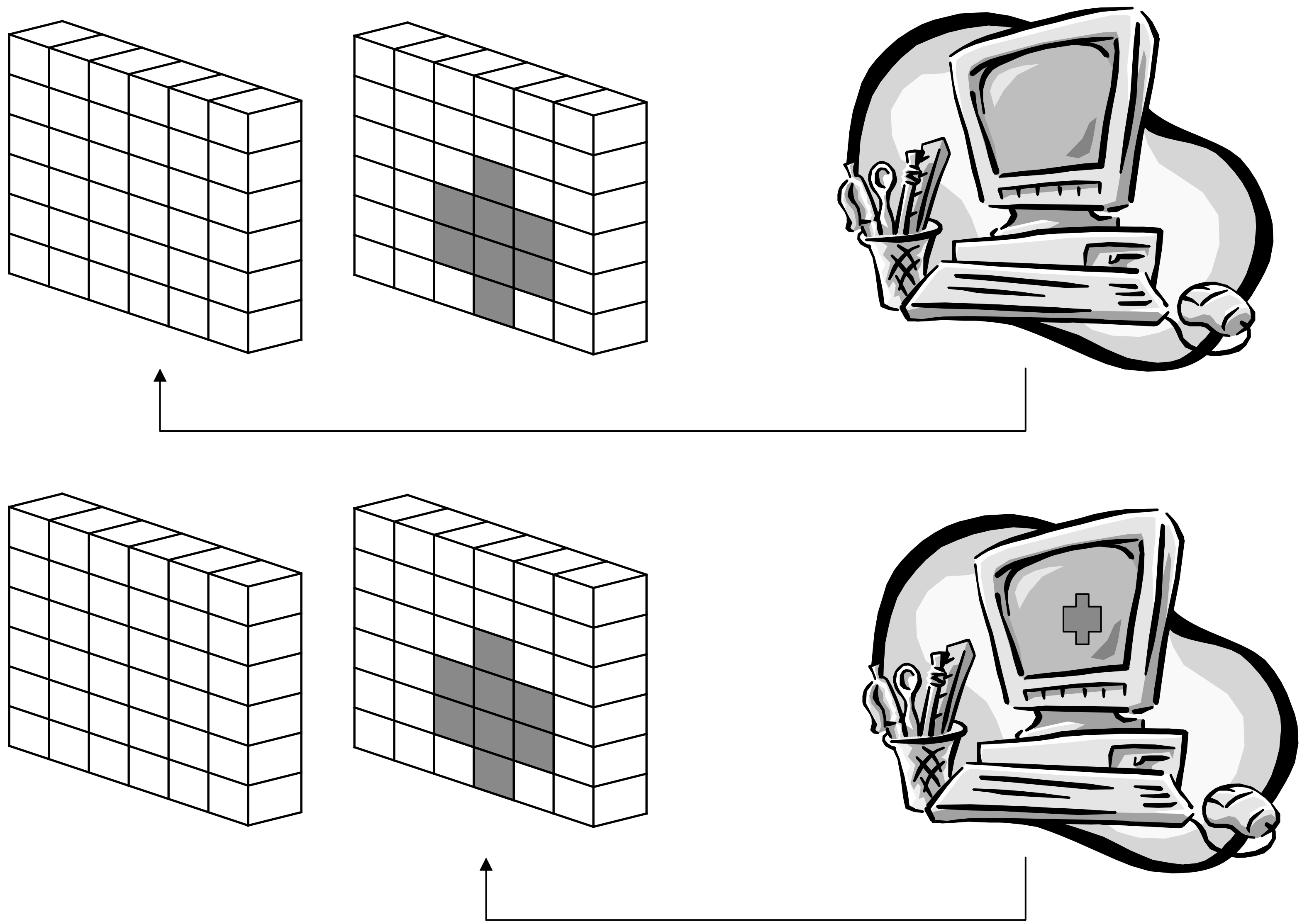
Depth test



Depth test

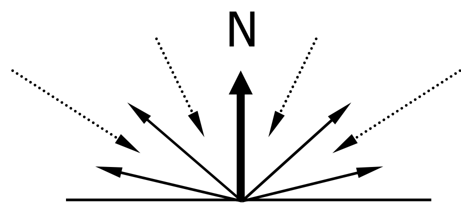


Double buffering



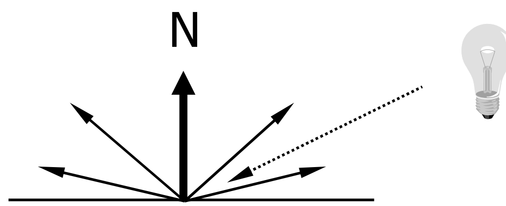
The Phong reflection model

Ambient



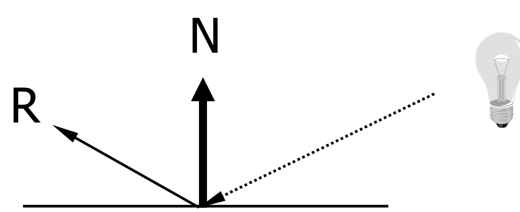
+

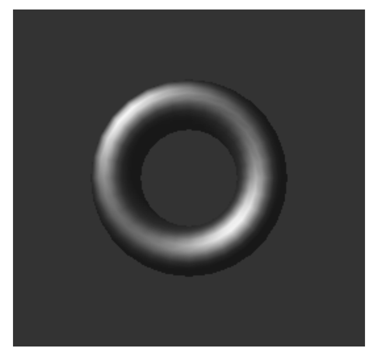
Diffuse



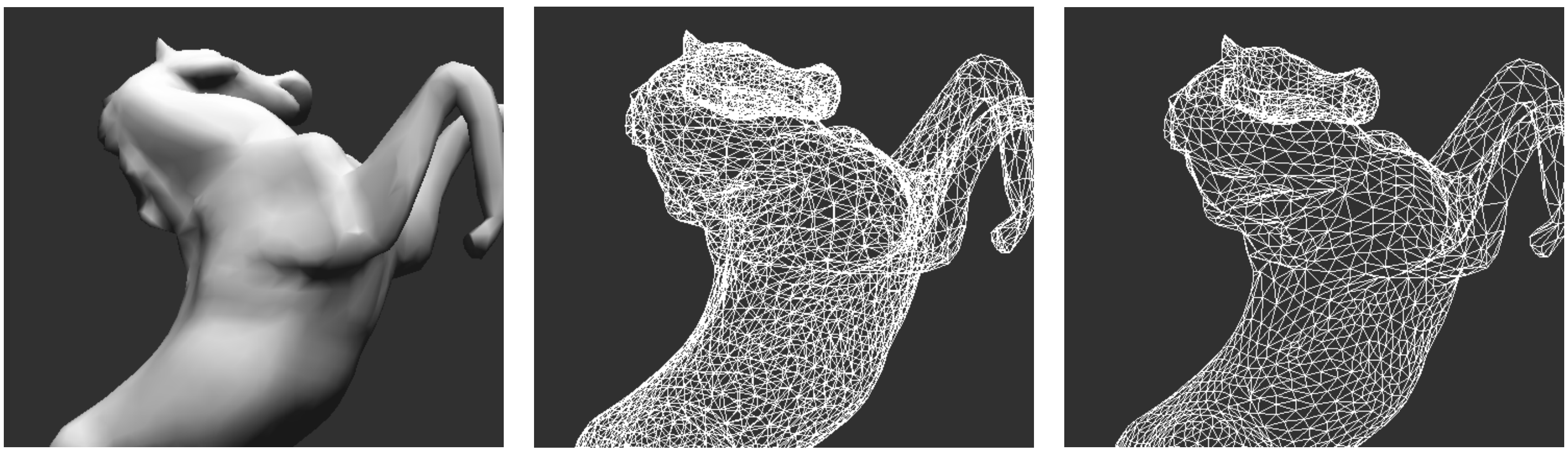
+

Specular

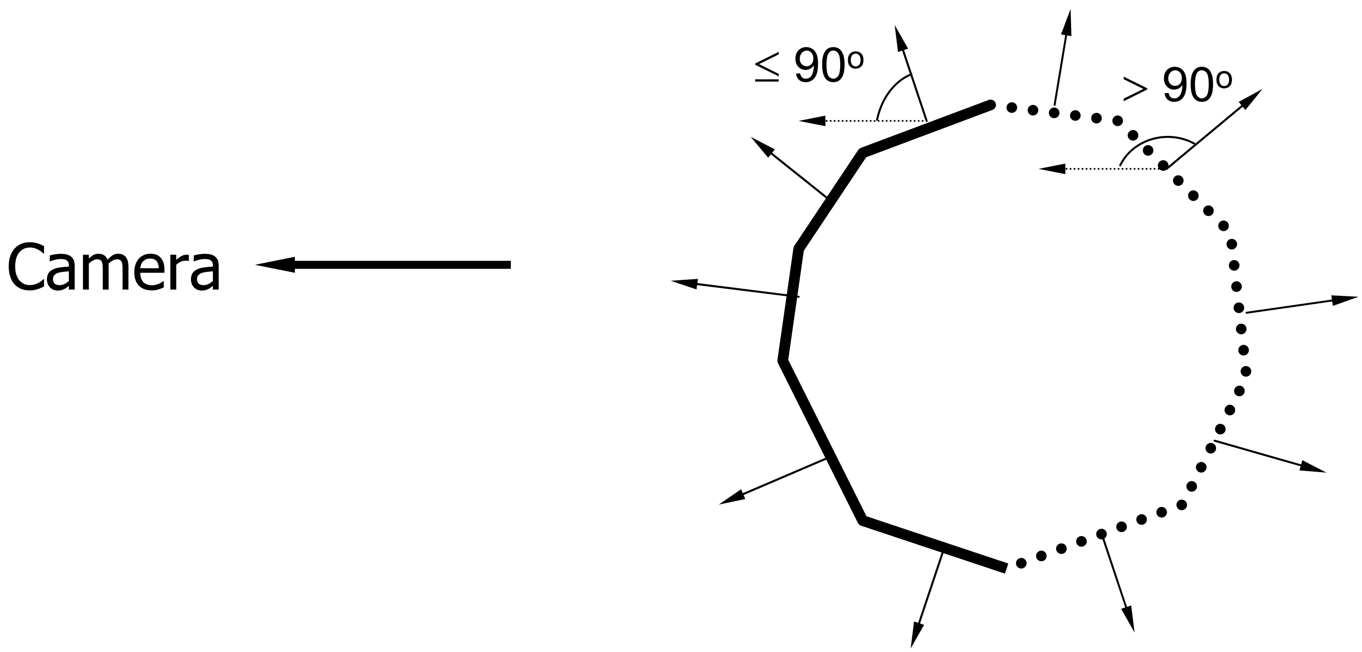




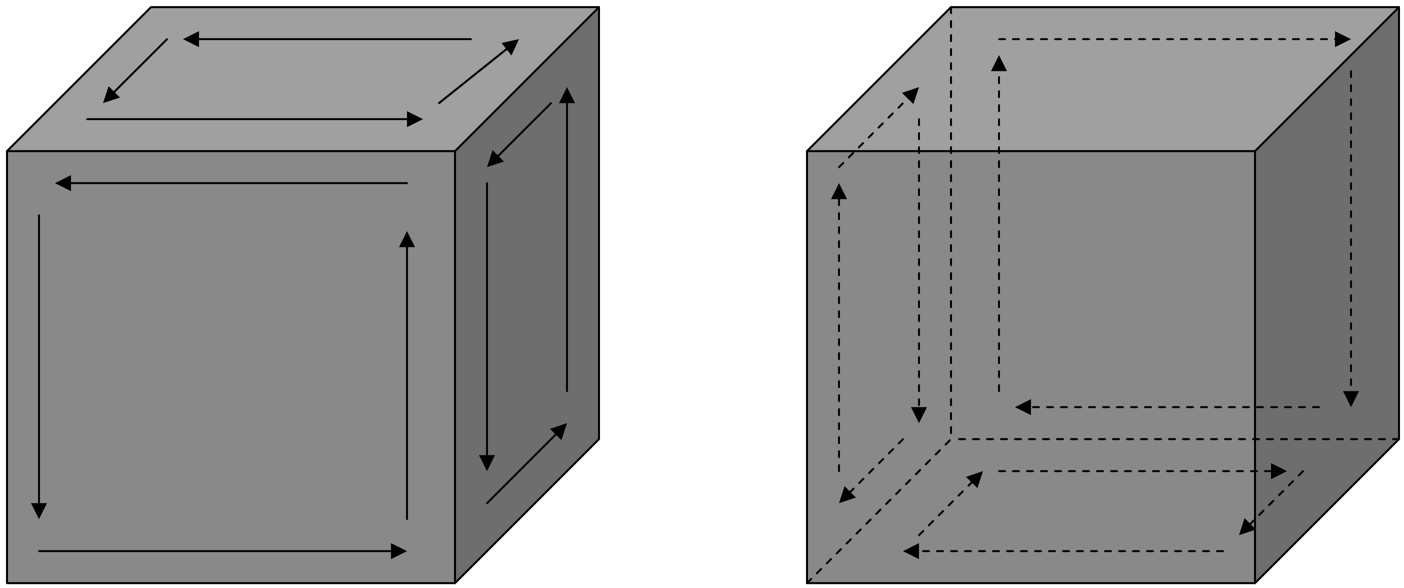
Back face culling



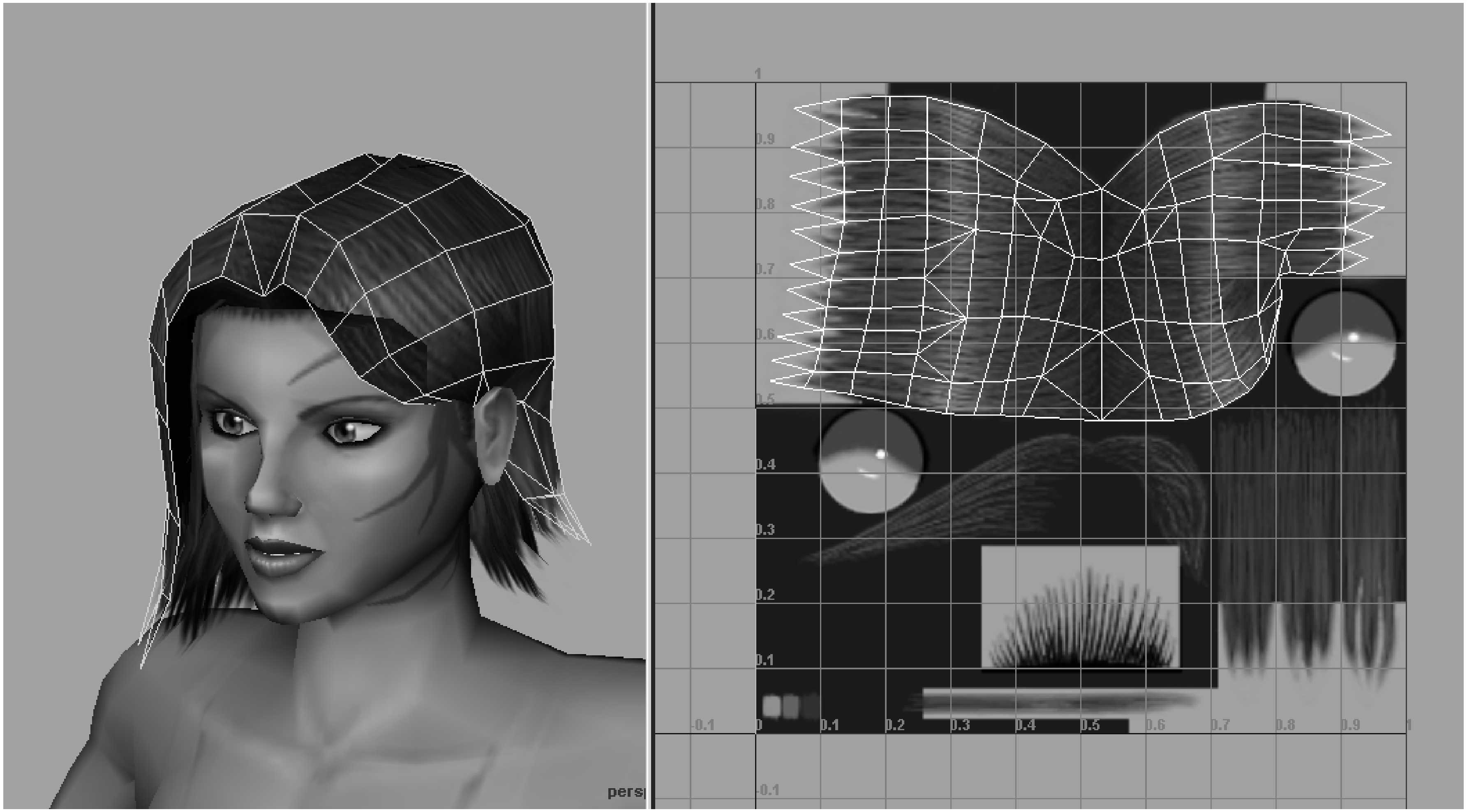
Back face culling



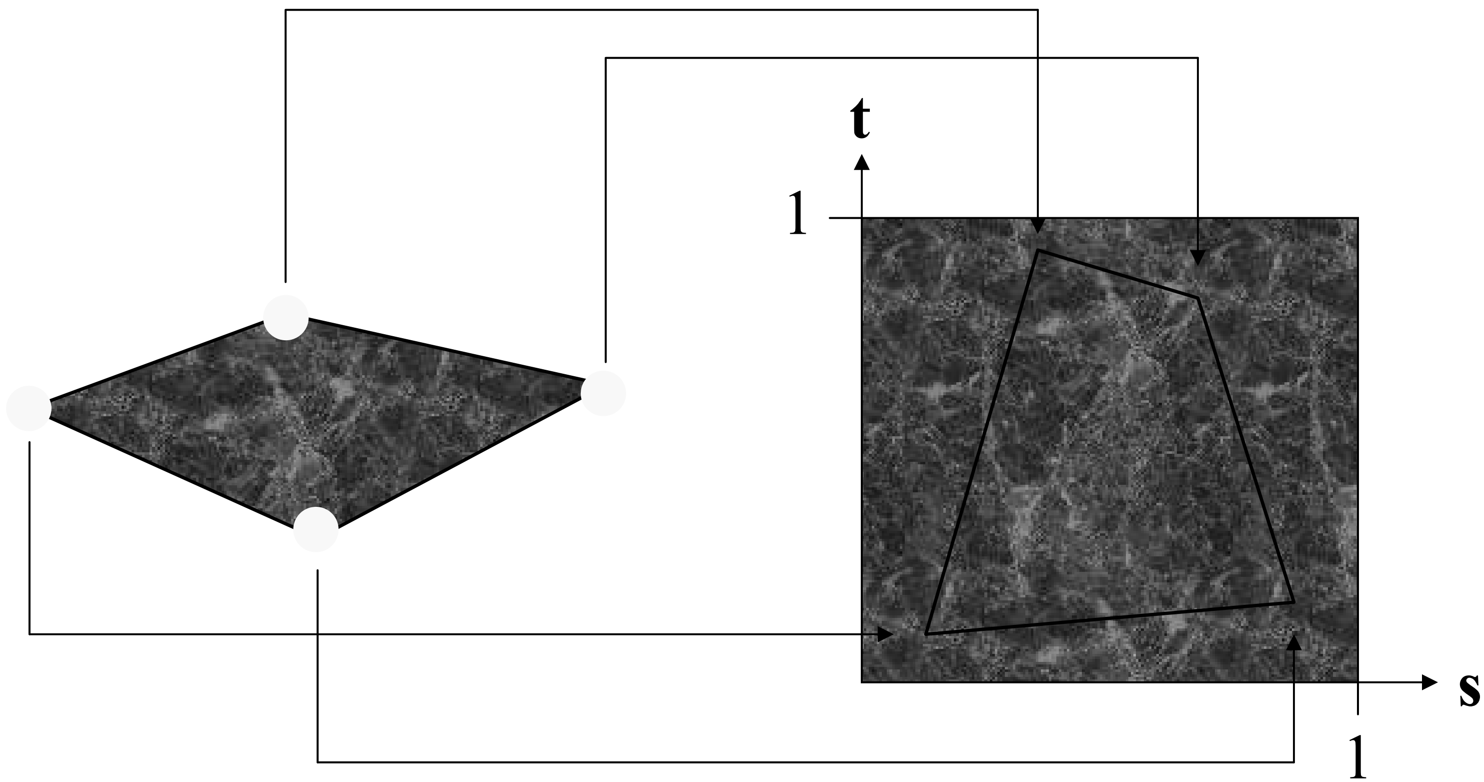
Front facing vs. back facing



Textures

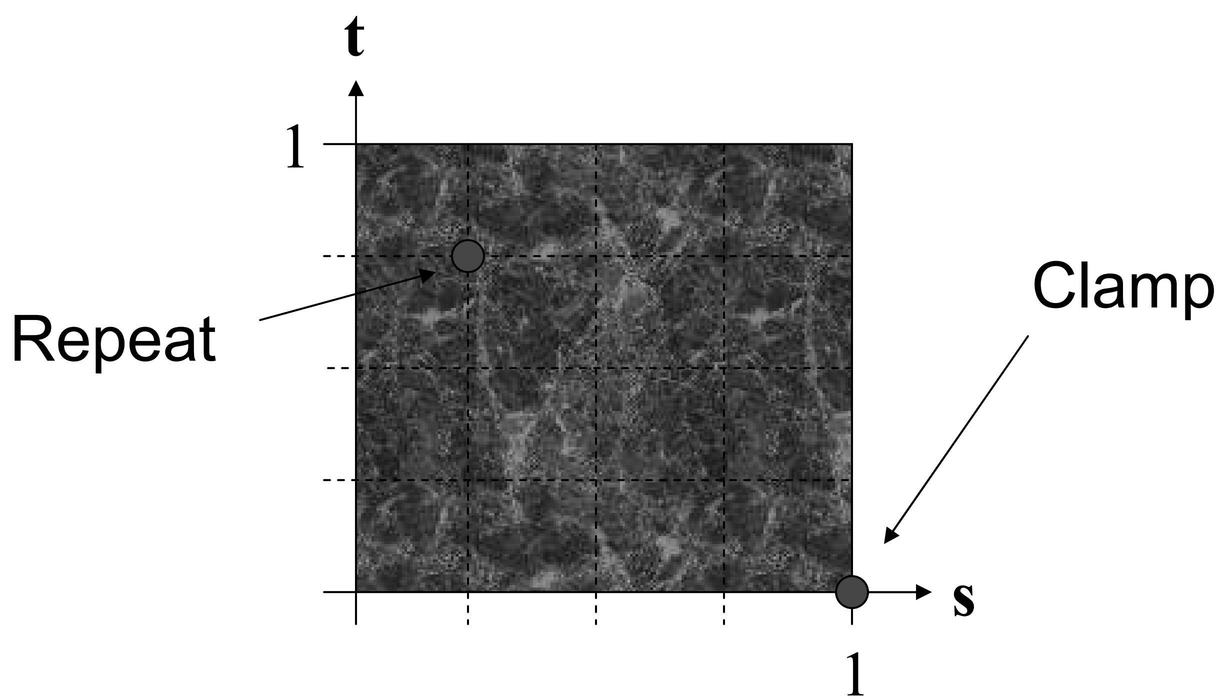


Texture coordinates

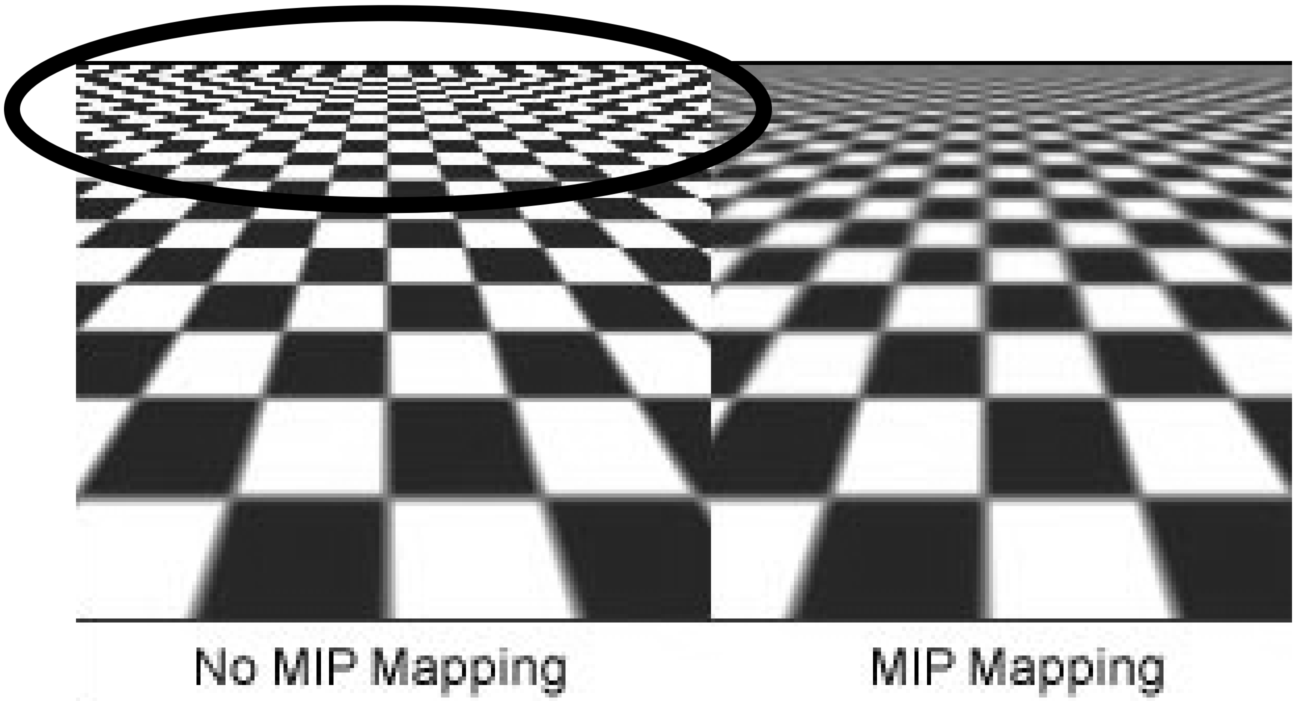


Clamp / repeat

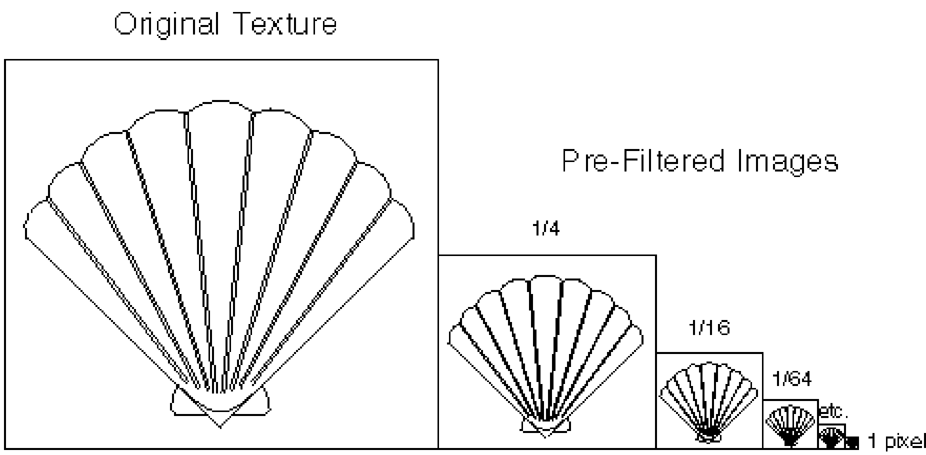
$(s, t) = (1.25, -2.75)$

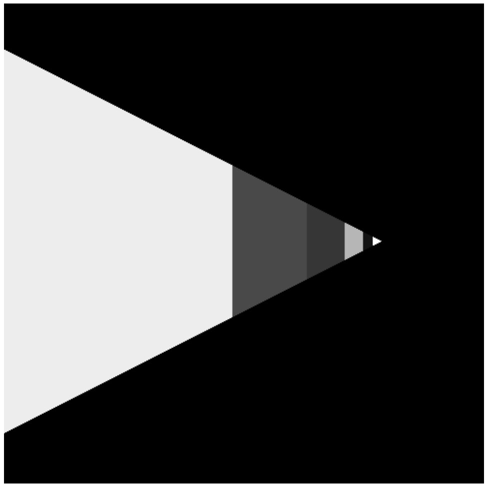


MIP-mapping

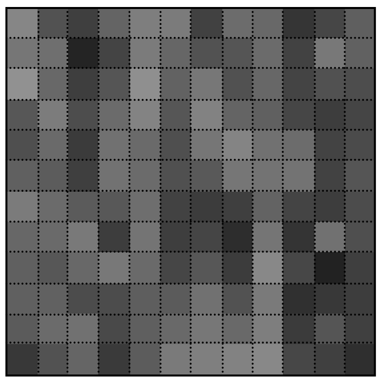


MIP-mapping

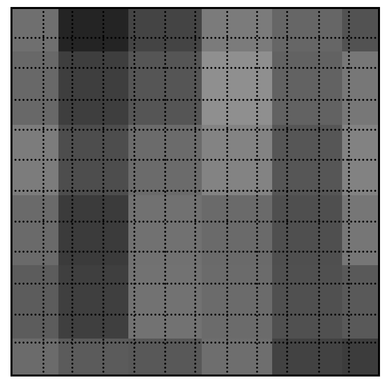




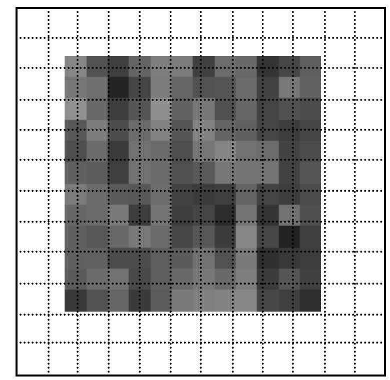
Magnification and minification



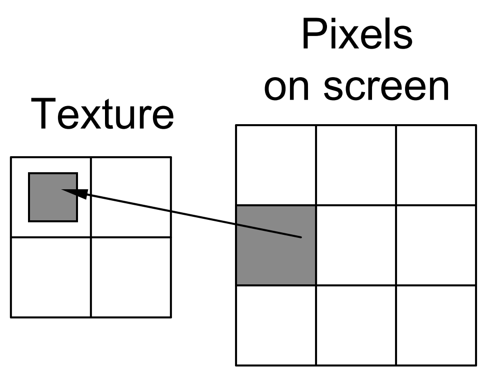
Texel size = pixel size

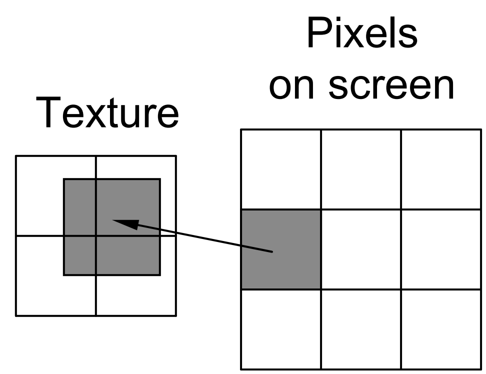


Magnification

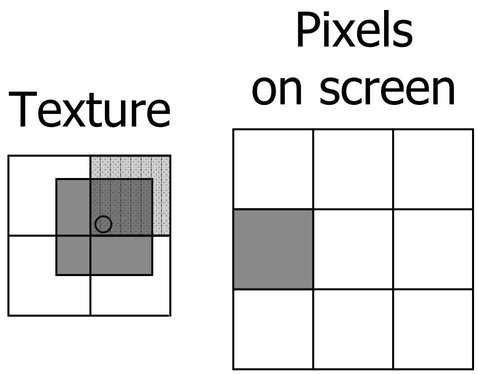


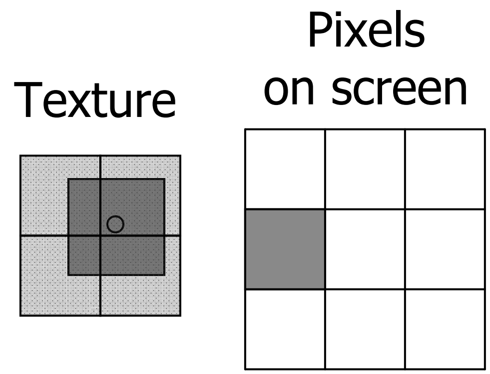
Minification



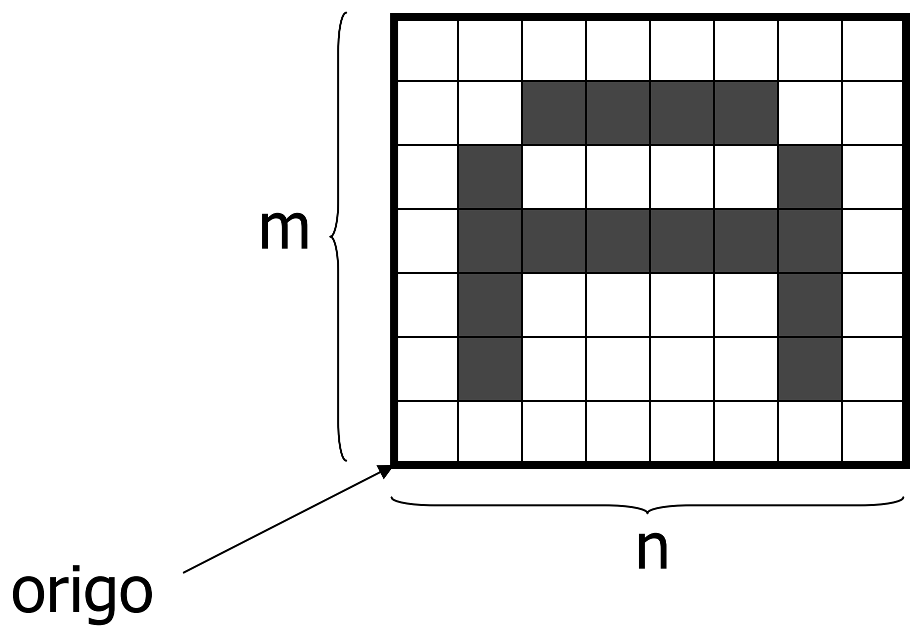


Filtering options





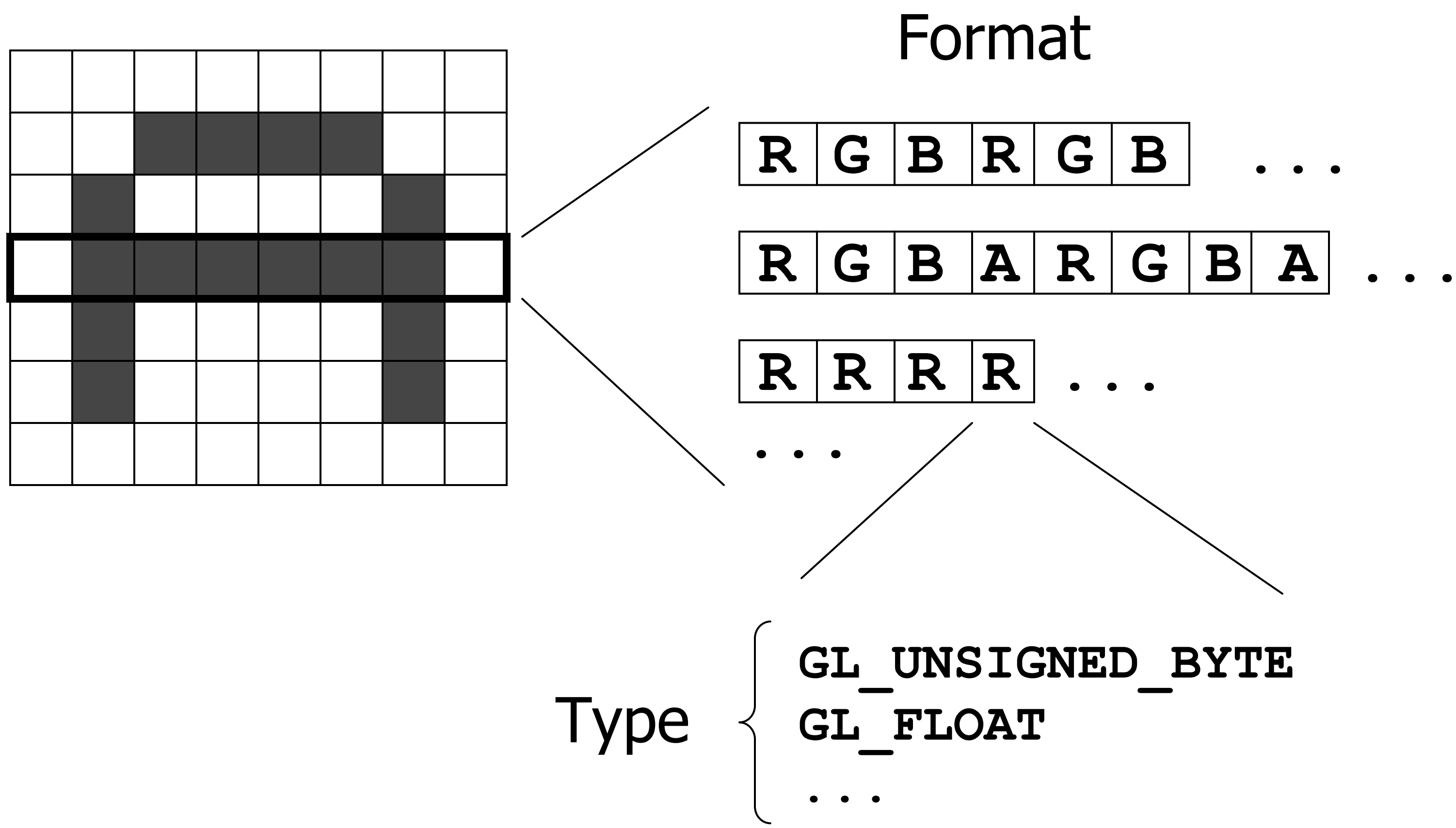
Texels



$(\texttt{\textbf{width}}, \texttt{\textbf{height}}) = (2^n, 2^m)$

where m, n are integers ≥ 0

Formats and data types



LIVE CODE SESSION