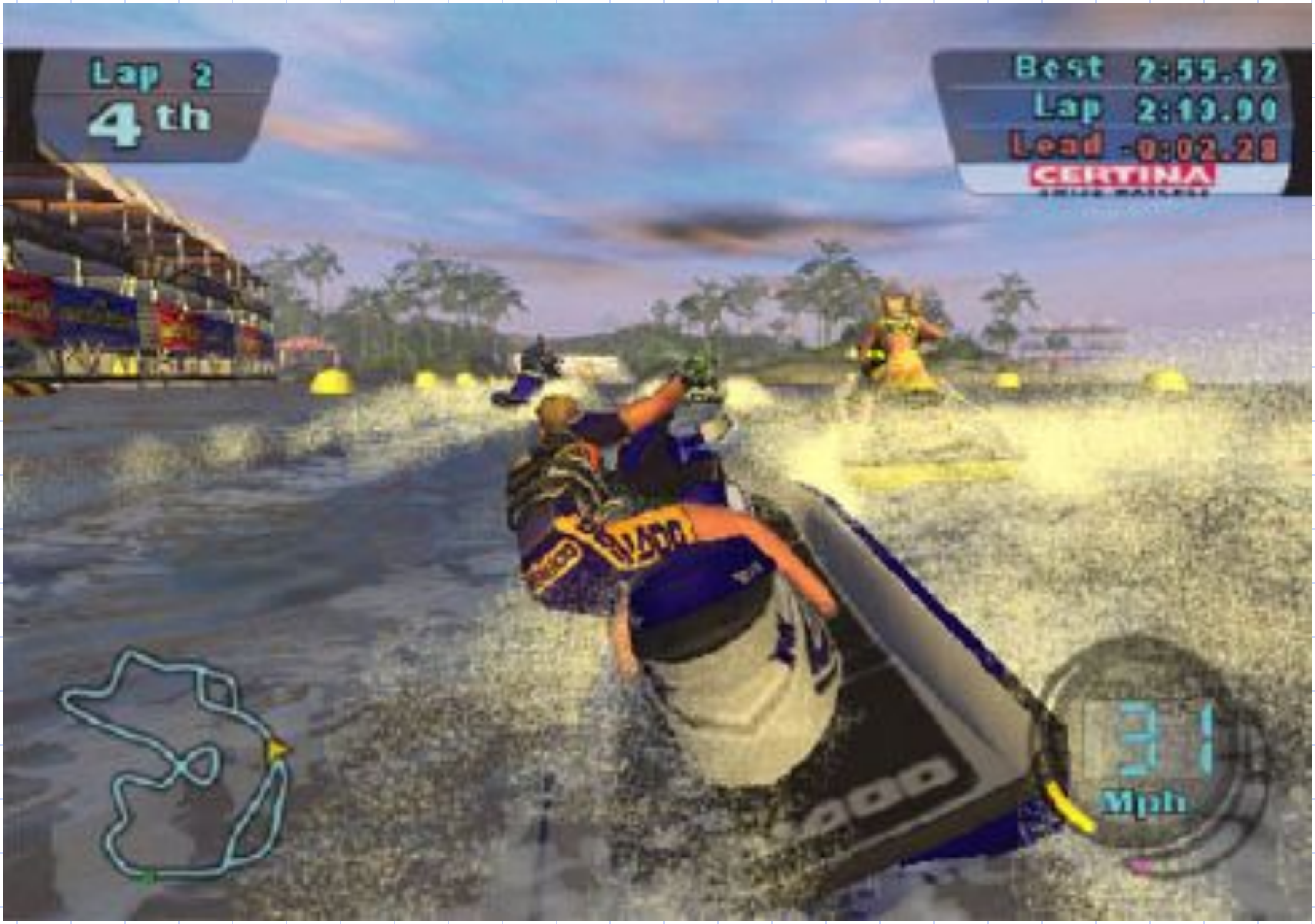
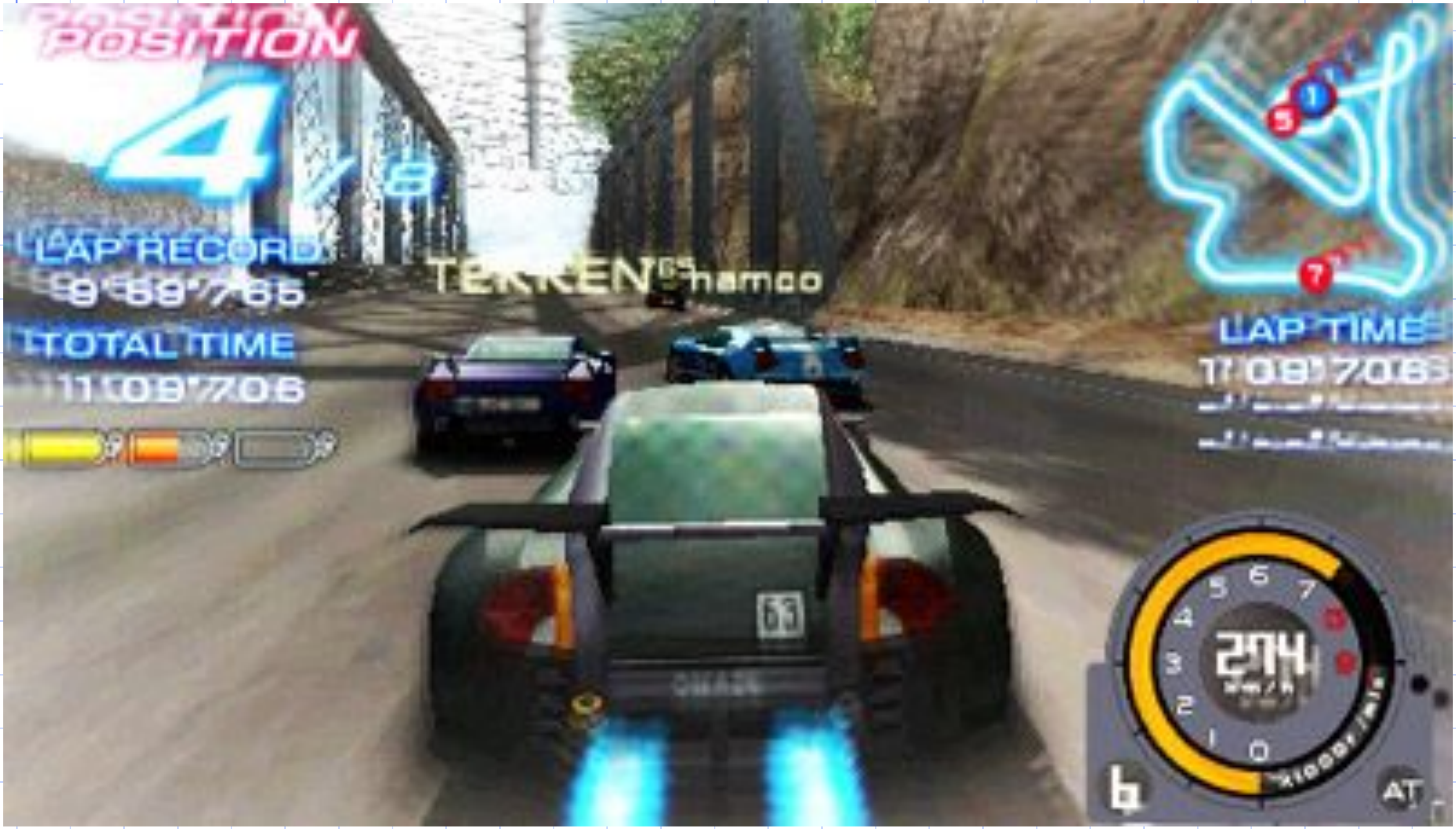
Image-based Rendering

Ref: RTRv2

Introduction



Ridge Racer (Namco)



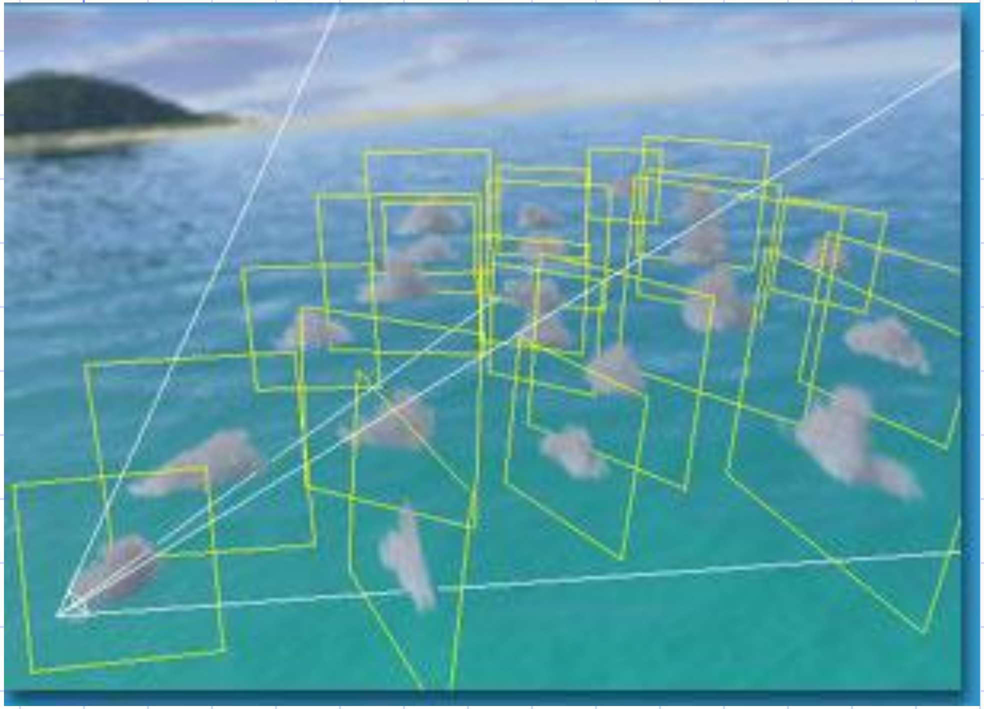
Techniques

- Sprite:
  - A small image, often used in animated games but also sometimes used as a synonym for icon.

- World-aligned billboards
  - Cylinderical and spherical
- Impostors

See-Thru Textures (aka, cutouts)

- Test α of transparent part
- Enable alphatest



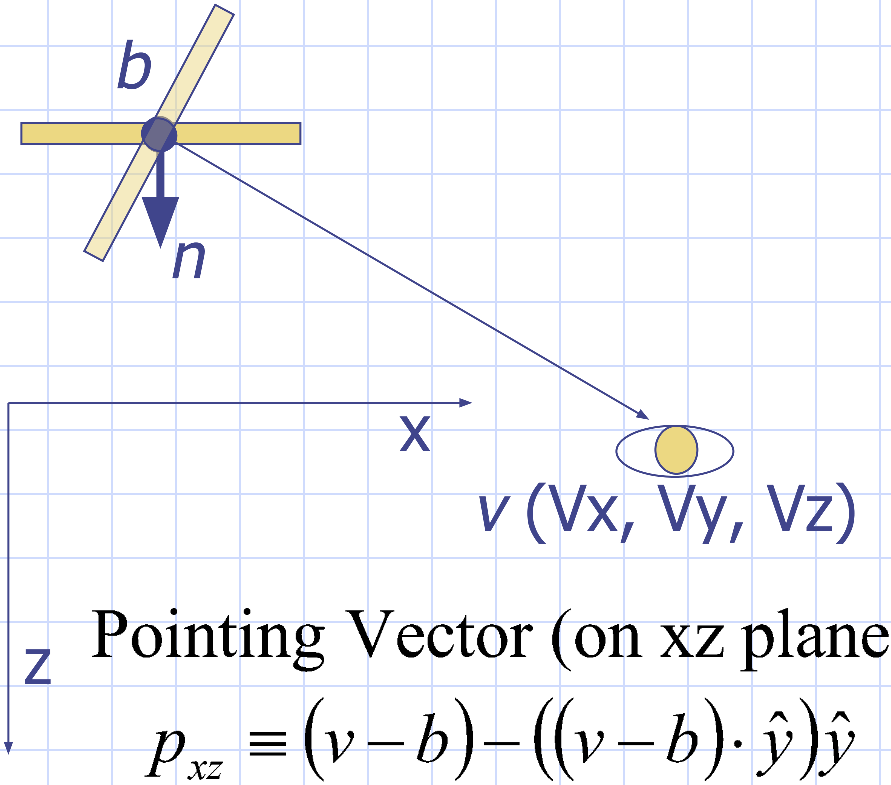
Definition (Billboard)

- A textured polygon always rotated to face the viewer
- Two kinds:
  - Cylindrical: for cylindrically symmetric objects (e.g., trees)
  - Spherical: for spherically symmetric objects (e.g., smoke)

Assumptions

- For cylindrical billboards:
  - In OpenGL coordinate system
  - the scene is on XZ plane
  - View Up is Y-axis
  - Viewer need not stay on XZ plane

Cylindrical Billboard



Pointing Vector (on xz plane) :

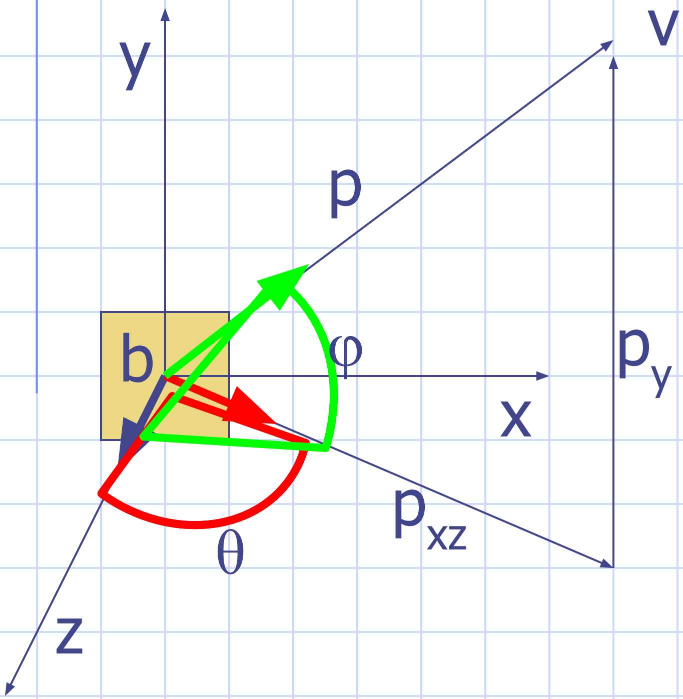
$$p_{xz} \equiv (v-b)-((v-b)\cdot\hat{y})\hat{y}$$

Rotation axis : $\hat{n}\times\hat{p}_{xz}$

$\boxed{\hat{p}_{xz} : \text{unit vector along } p_{xz}}$

Rotation angle : acos $(\hat{n}\cdot\hat{p}_{xz})$

Spherical billboards



Rotate about localY

$\theta = \text{acos}\ (n,p_{xz})$

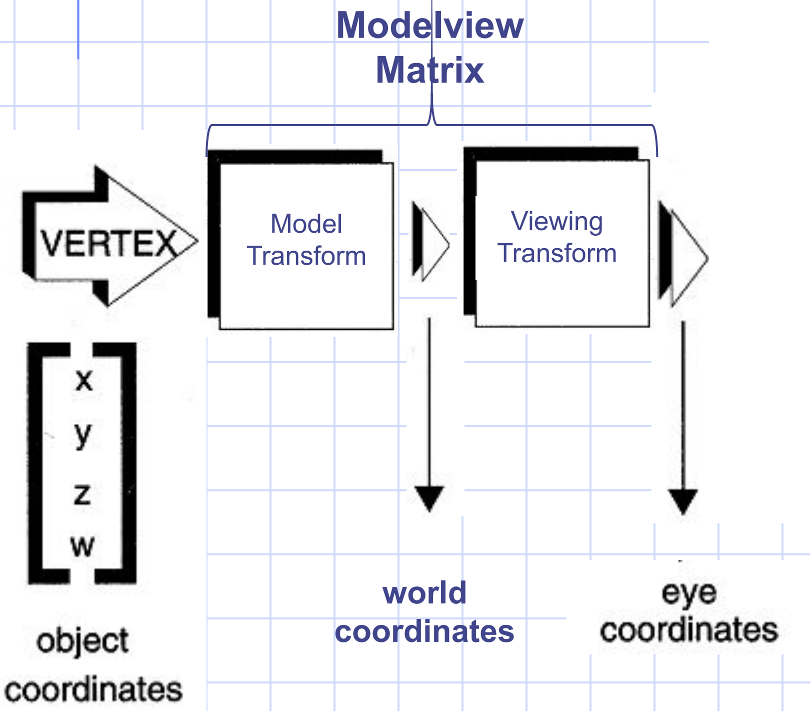
Rotate about localX

$\varphi = \text{acos}\ (p_{xz},p)$

Watch out for signs of angles

Where is the camera?

- For any reason (e.g., cylindrical billboard), you want to know where camera is, just remember that the eye coordinate of the camera is always at the origin [facing –Z]



$$Mp = 0$$

$$\left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right]\left[\begin{array}{c} p \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1 \end{array}\right]$$

$$Rp + t = 0$$

$$p = -R^{-1}t = -R^{T}t$$

$$[R \text{ consists of rotation and transl only}]$$

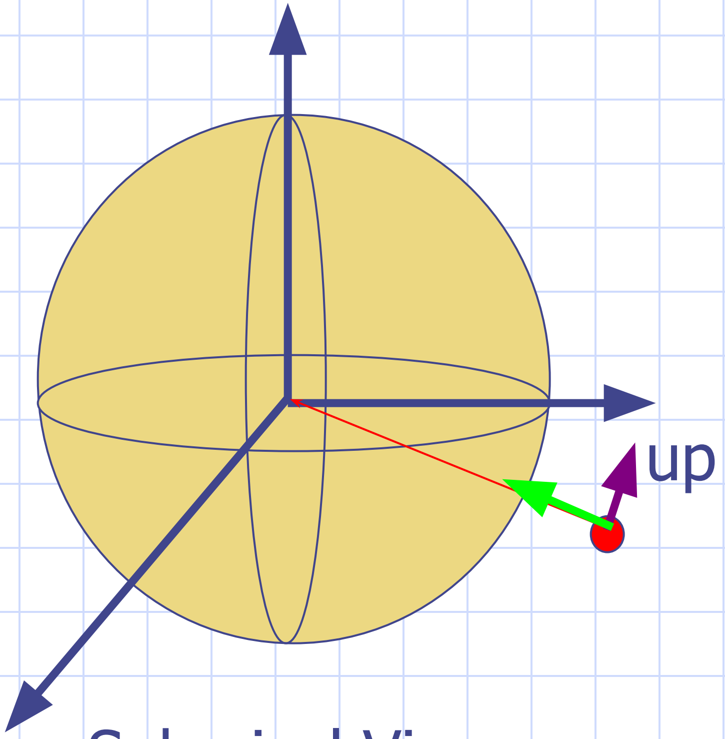
Finding Up Direction

$$\left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right]\left[\begin{array}{c} up \\ \hline 0 \end{array}\right]=\left[\begin{array}{c} 0 \\ 1 \\ 0 \\ \hline 0 \end{array}\right]$$

$$R(up)=y$$

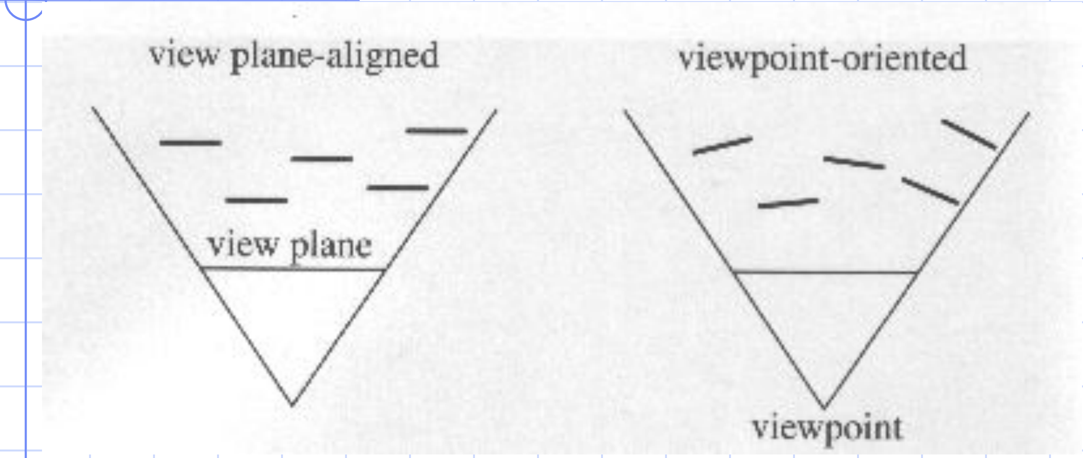
$$up=R^{-1}y=R^{T}y$$

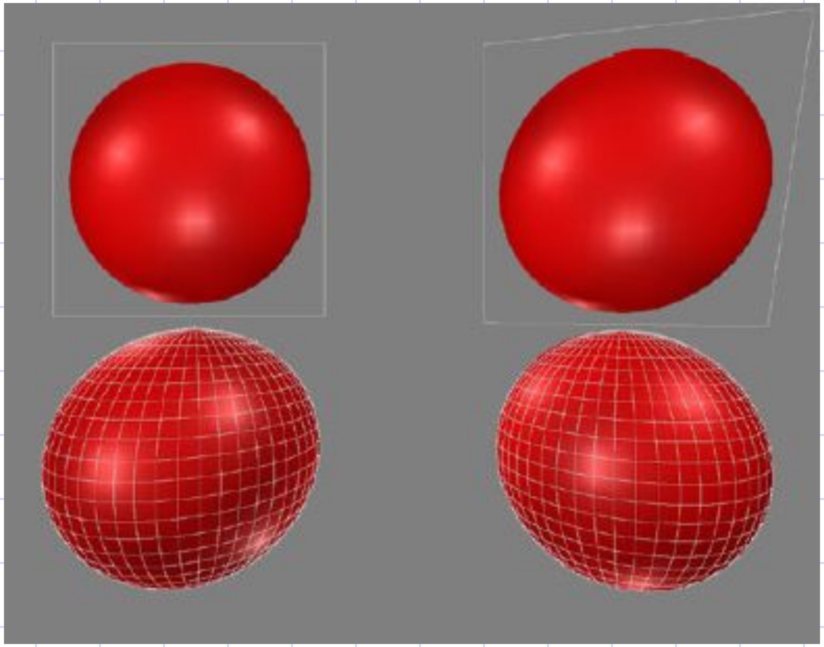
[*R* consists of rotation and transl only]



Spherical Viewer

Screen/World Aligned Billboards





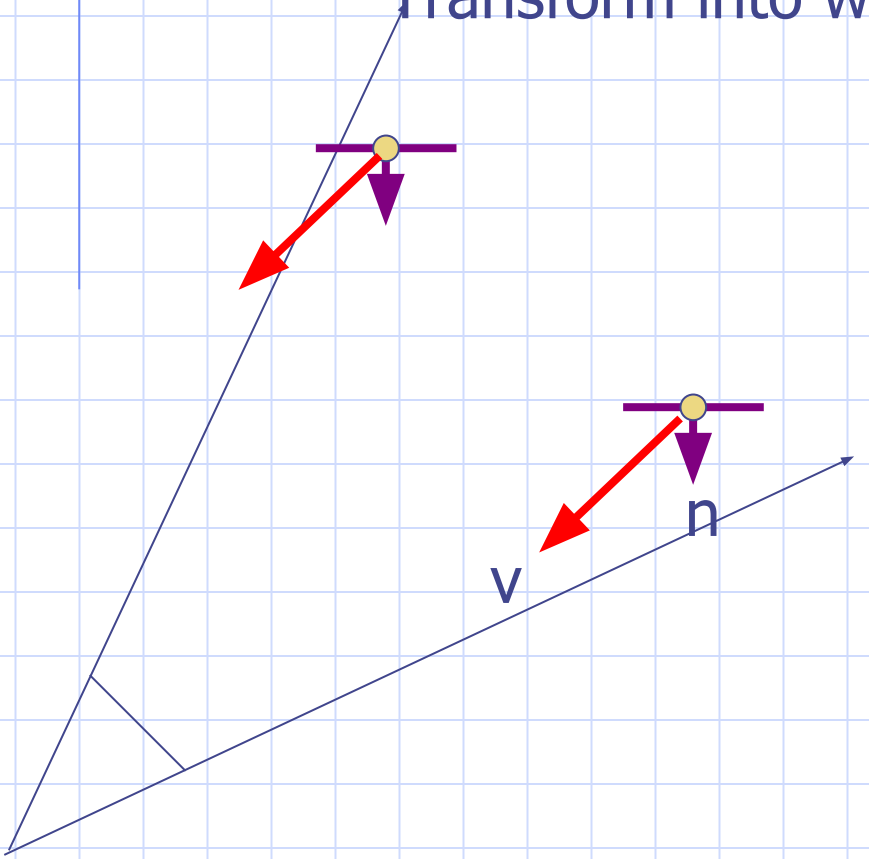
Billboards

Real spheres

Screen-aligned Billboard

v: [0,0,1] in eye coord.
Transform into world coord.

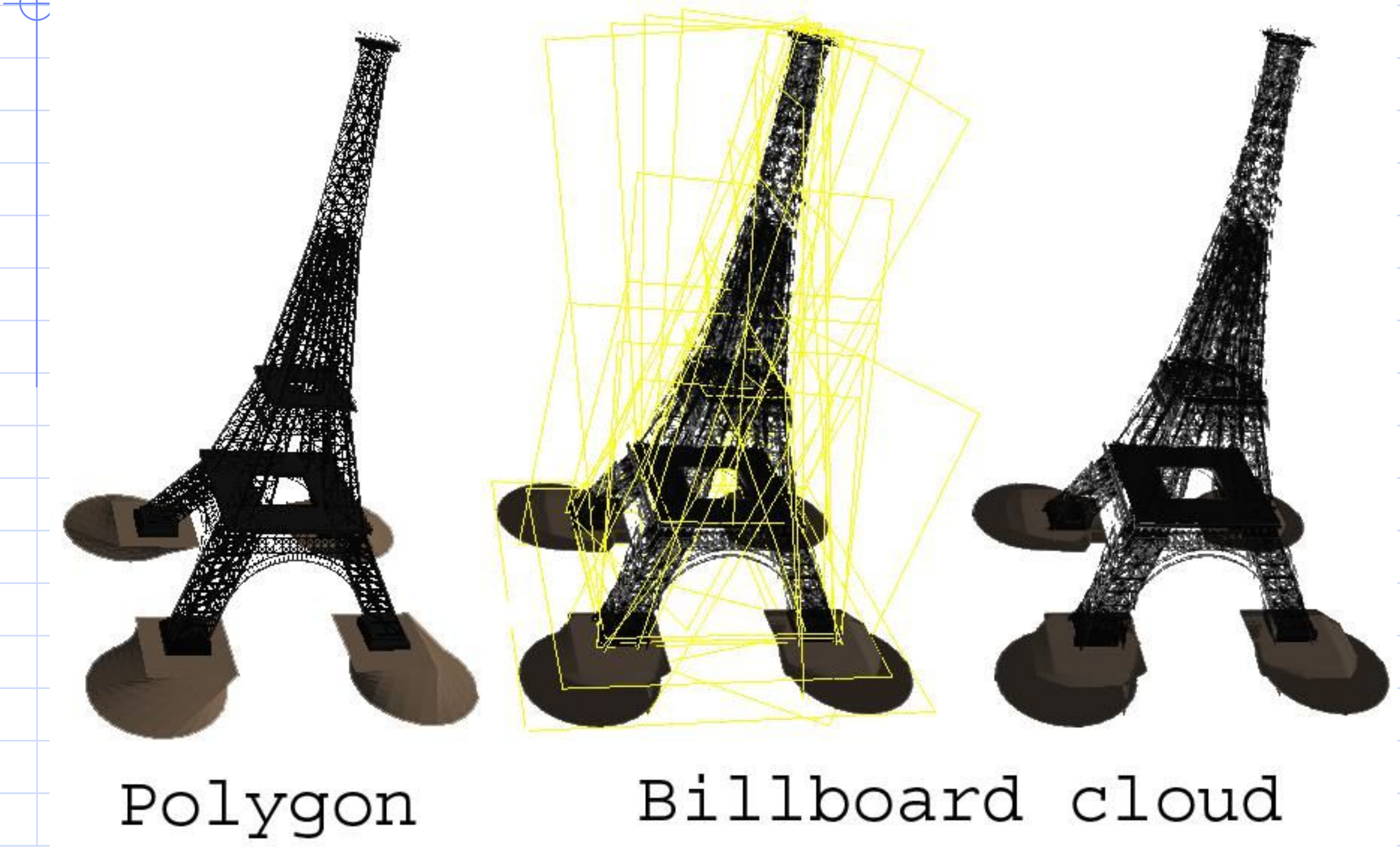
During each frame:
Rotate every n into v



$$\left[\begin{array}{c:c} R & t \\ \hdashline 0 & 1 \end{array}\right]\left[\begin{array}{c} v \\ \hline 0 \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ \hline 0 \end{array}\right]$$

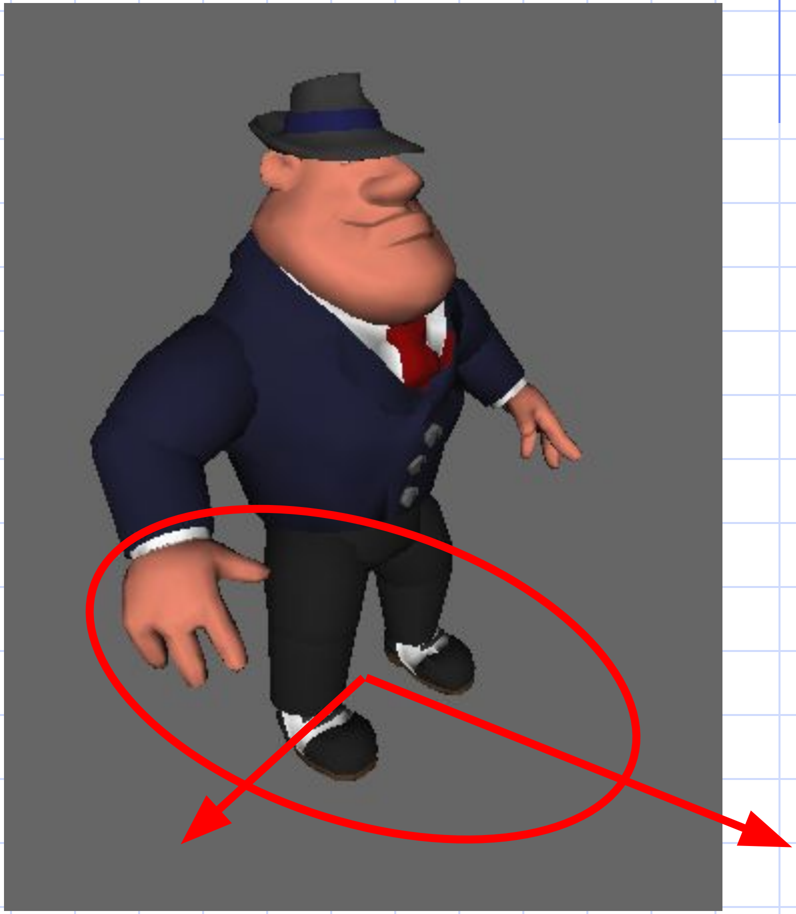
$$\begin{bmatrix} R \end{bmatrix}\begin{bmatrix} v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad v = R^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Billboard Cloud for Extreme Model Simplification (url)



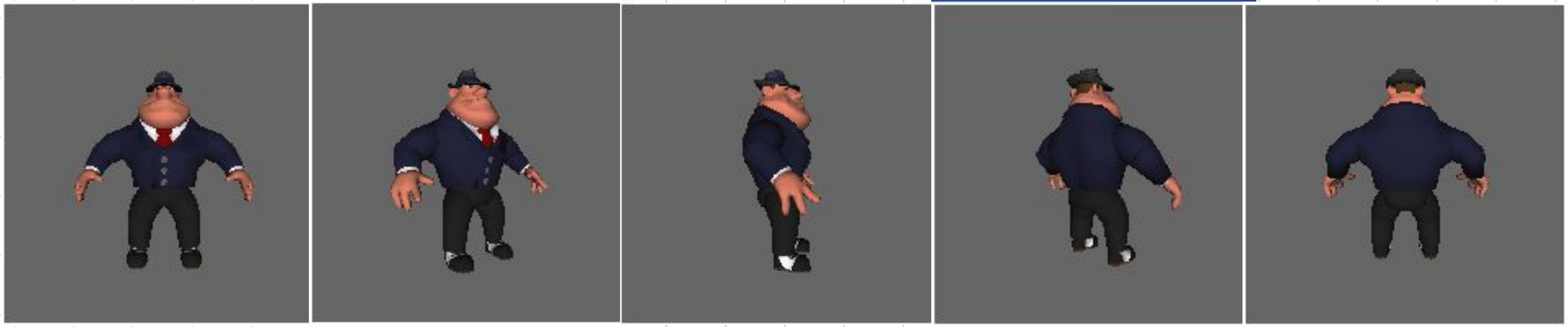
Application: (Quasi-)Impostor

- Impostor: render 3D object into texture, in various viewing angles
- Instead of showing the facet model, show a polygon with correct texture



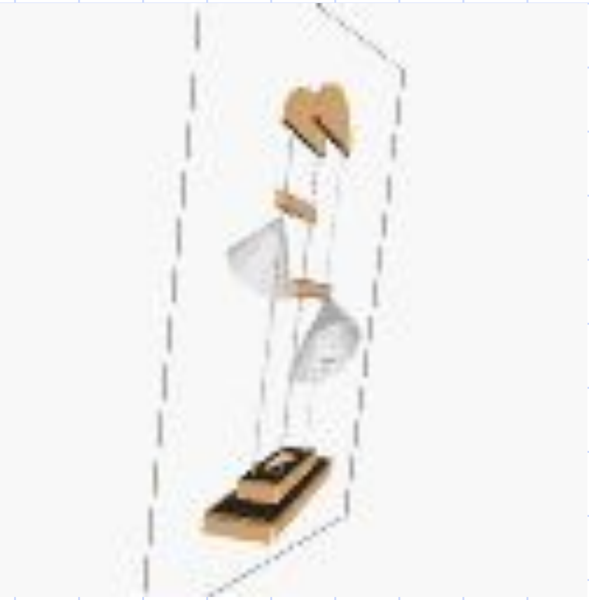
Impostor-2

- Usually textures are stored in a single texture object, to avoid repeated binding
- Use texture transform to switch to the correct one
- glGetIntegerv (GL_MAX_TEXTURE_SIZE, &size)



Impostors

- A billboard created *on the fly*
  - rendering a complex object from the current viewpoint into an image texture, then post on a billboard)



Impostors (Schaufler)

- The top of figure 1 shows an object for which an imposter is generated together with its bounding box, which is used to bound the extent of the object's picture. The imposter is chosen to lie in the plane **P** through the object center **C** normal to the direction from **C** to the point of view **V** (normal to **n** as shown at the bottom of figure 1). The extent of the imposter is taken as <u>the smallest rectangle in **P**</u> which contains the projection of the object's bounding box onto **P**

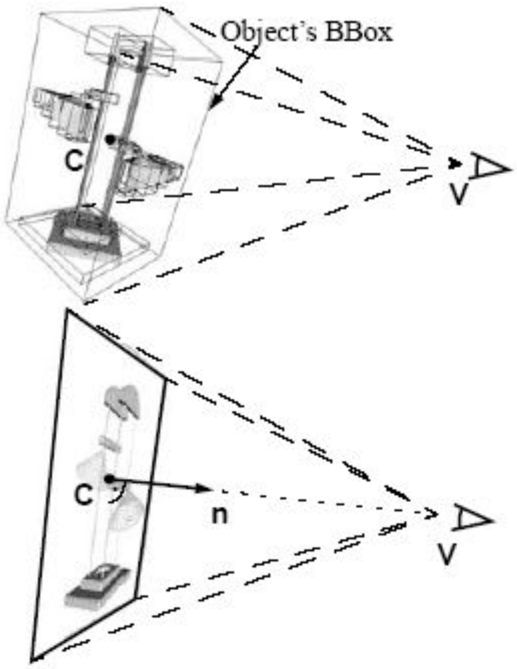
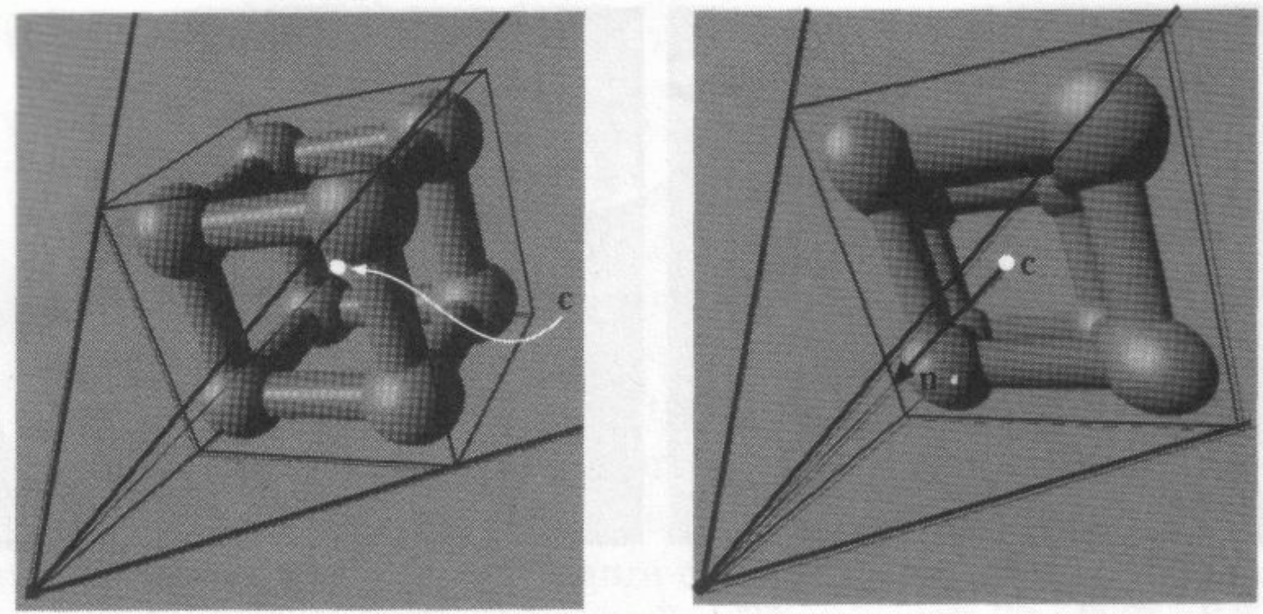


Figure 1. Original Object and Imposter

Impostors (cont)

- Idea:
  - Project the bounding box of the object to determine size of impostor polygon
- Issues
  - Alignment (texture placement)
  - Texture resolution
  - Invalid impostor (require regeneration)

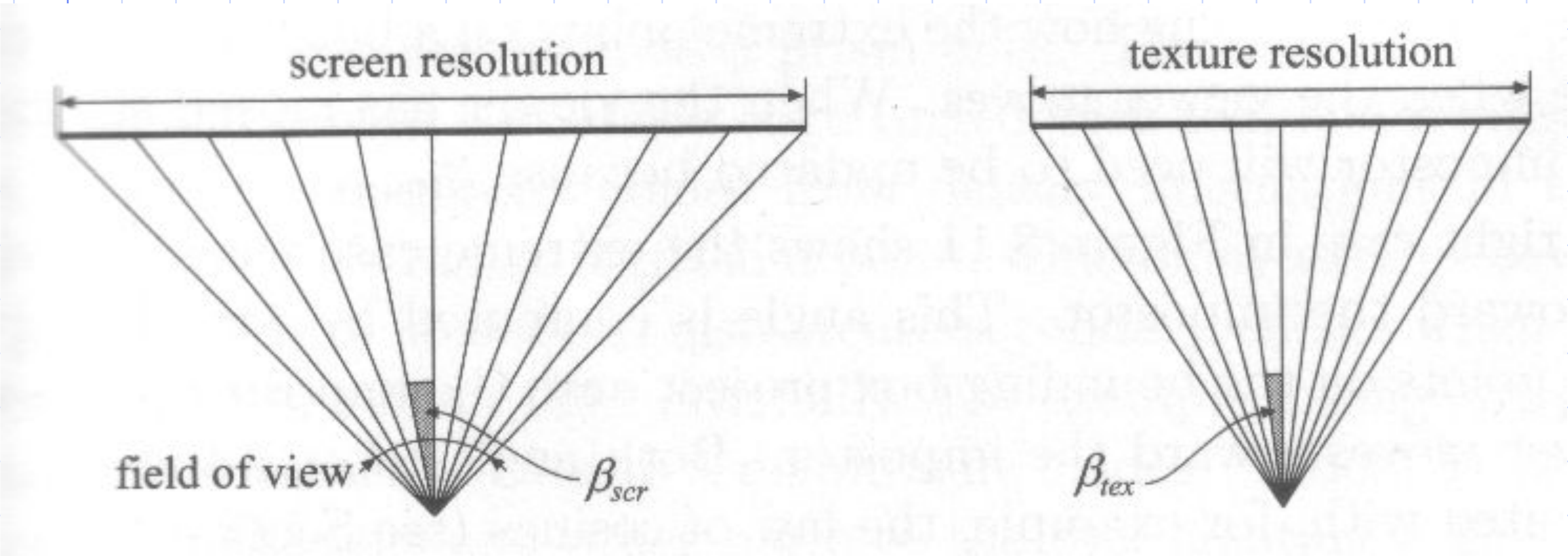


Impostors (cont)

- Rendering speed depends on the number of pixels
- Usage:
  - A few instances of the object
  - A few frames of the object (strategy for updating imposters)
  - Rendering distant objects
    - Movement of project image diminishes with an increased distance from viewer
    - Image lowpass filtered to create depth-of-field effect

Impostor: texture resolution

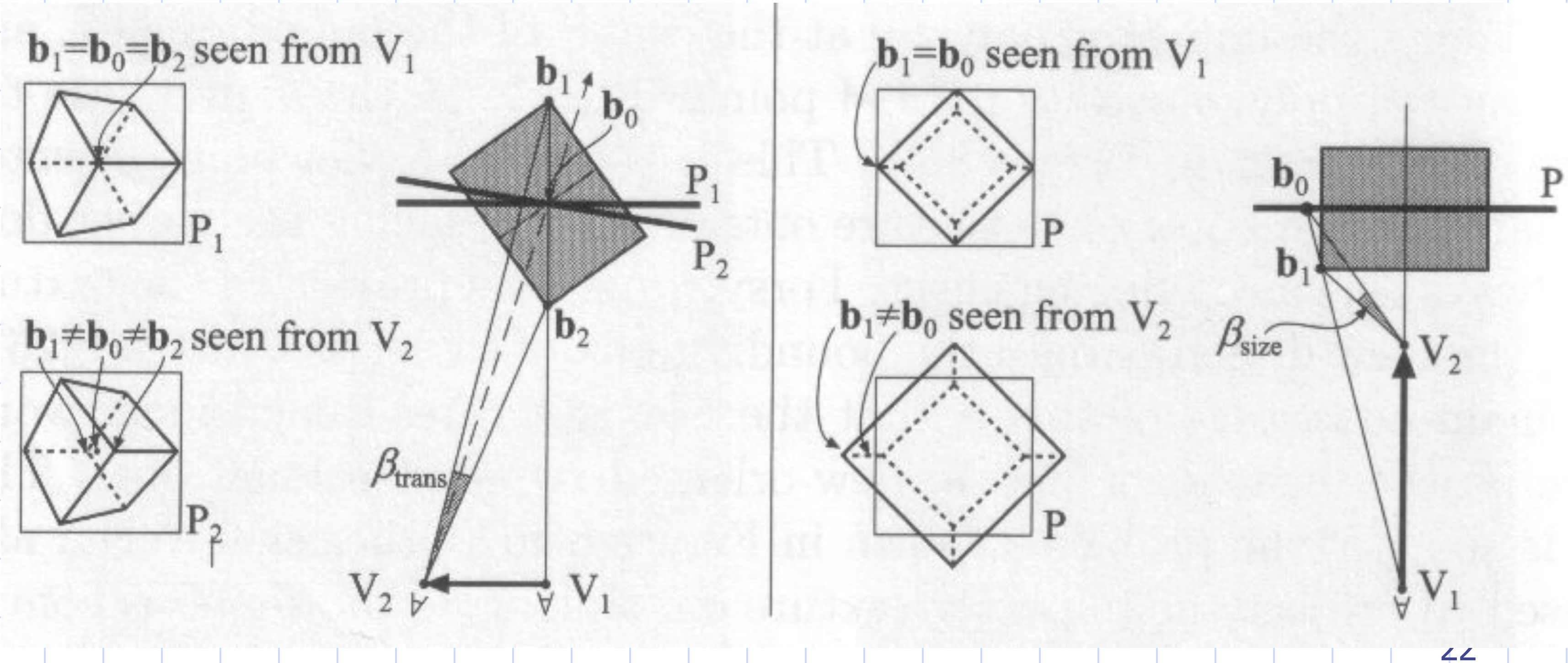
- $$texres = screenres \frac{objsize}{2 \cdot \mathrm{distance} \cdot \tan(fov/2)}$$



Impostors Regeneration

- Schaufler

  $\beta_{tex}, \beta_{trans}, \text{ or } \beta_{size} \geq \beta_{scr}$



Problem of Billboard

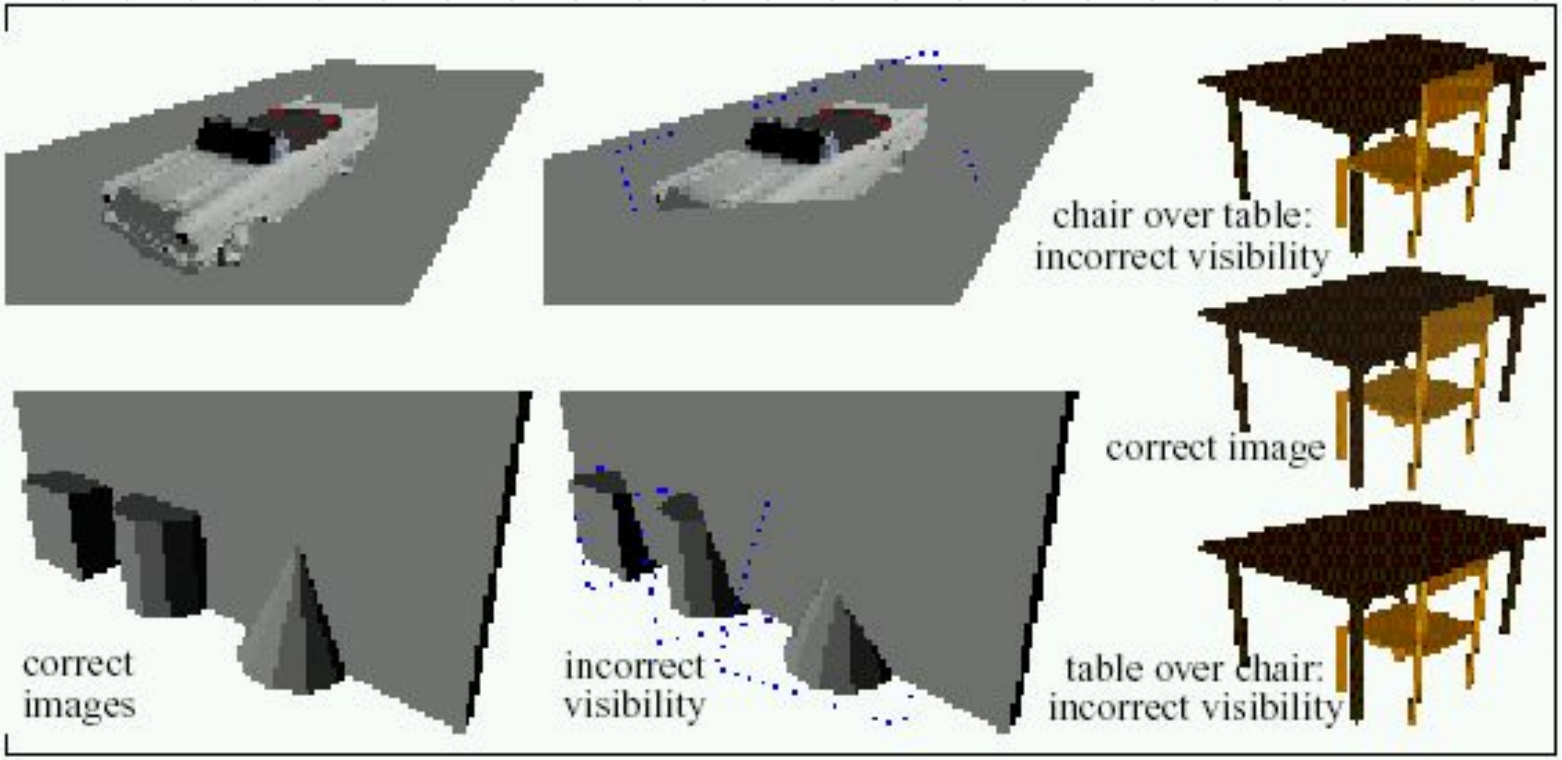
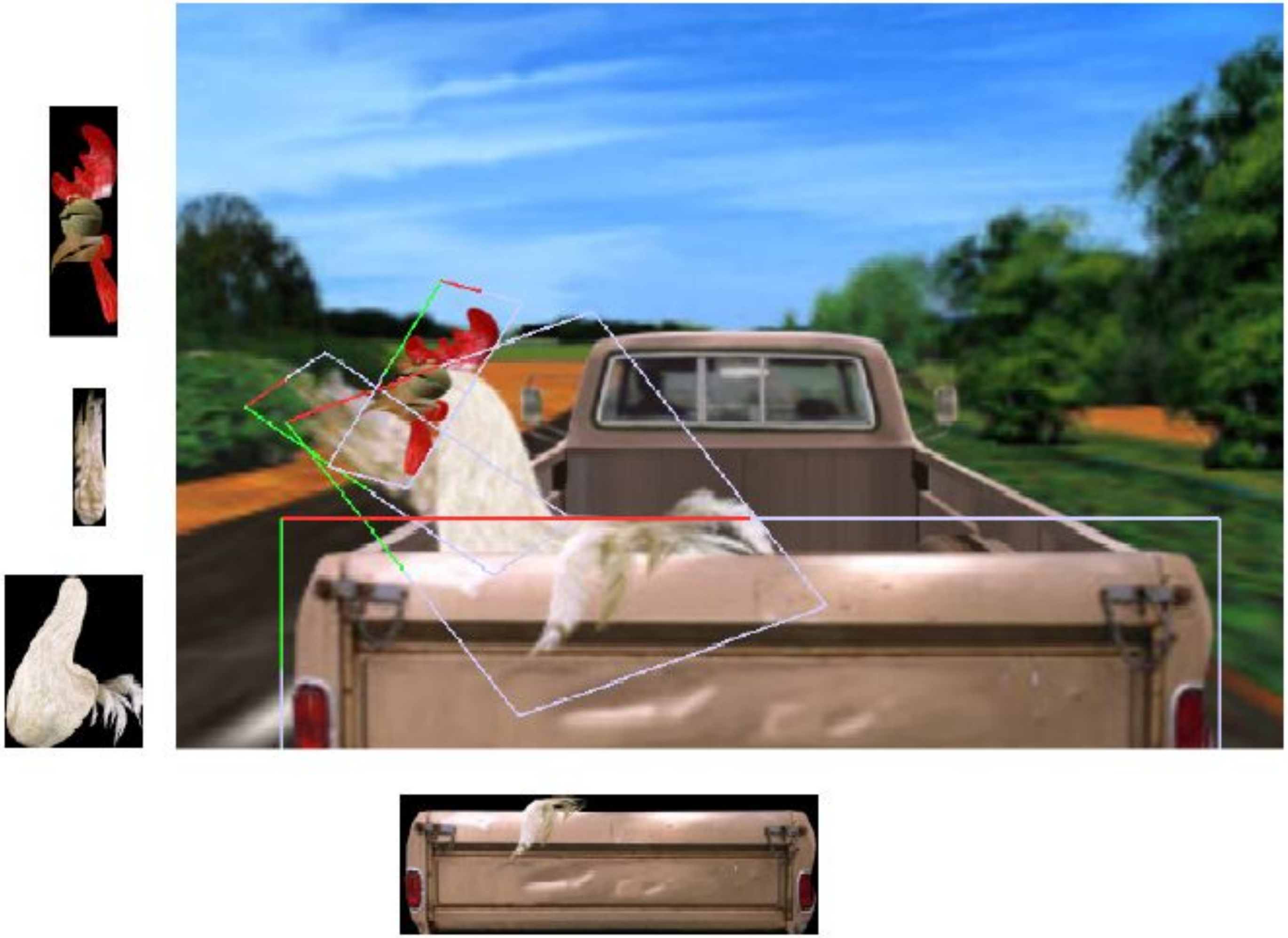


Fig. 1: Left and Middle: Incorrect visibility due to interpenetrating polygons when individual objects are replaced by partially-transparent textures.
Right: Incorrect visibility due to layering of object images.

Coherent Layers
(Lengyel and Snyder, 1997)

- Hand animators use layers to reduce the number of cells to draw
  - One layer for background, one for middle ground, one for character,…
  - Background layers need to be changed less frequently than foreground, slow moving less frequently than fast moving,…
  - Layers are composited as a final step
- Coherent layers was designed to work with hardware that supports fast compositing and layer warping
- Approach:
  - Break scene into layers by hand
  - At run-time, warp some layers, re-render others
  - Composite the layers into the frame buffer (back to front)

Chicken Run (80 layers)



Depth Sprites (Nailboards)

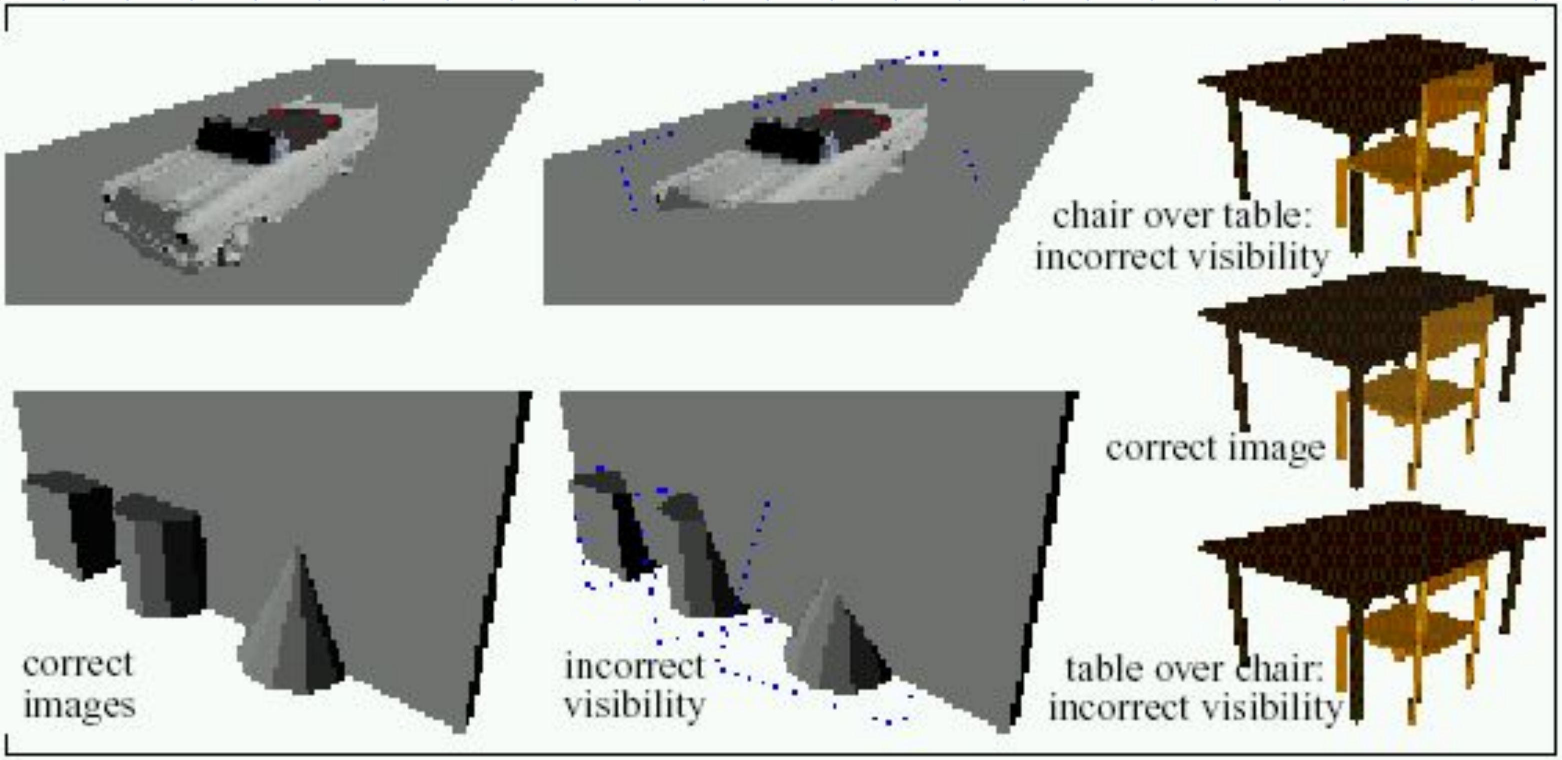
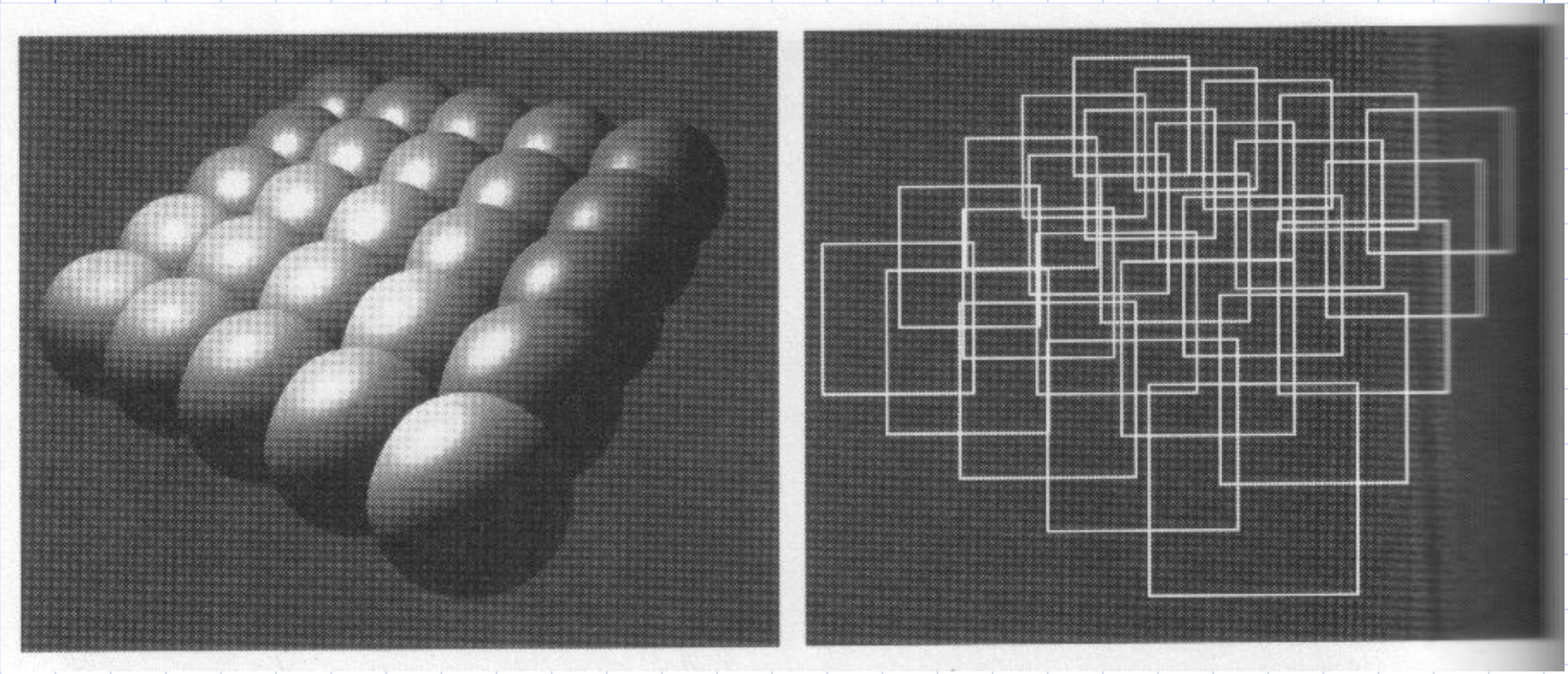


Fig. 1: Left and Middle: Incorrect visibility due to interpenetrating polygons when individual objects are replaced by partially-transparent textures.
Right: Incorrect visibility due to layering of object images.

Nailboards (cont)



Hierarchical Image Caching

- Use impostors in a hierarchy for better performance
- Partition the scene into a hierarchy of boxes and create an impostor for each box
- Rerendering: done only when one or more of its children's impostors need to be updated due to movements.

Full-screen Billboarding

- Not really “billboard” per se; just something that covers the whole screen
- Blend in front
  - (camera) flash effects
  - Night vision goggle
  - high fill rate vs. changing colors of every polygon
  - Stencil buffer technique to create video effects (*iris* in Adobe Premiere)
- Background: sky in flight simulators
  - Move the background as the car moves