Computer Graphics (Introduction)

Thilo Kielmann
Fall 2004
Vrije Universiteit, Amsterdam
kielmann@cs.vu.nl

http://www.cs.vu.nl/˜graphics/

The Course in a Nutshell

- Credits: 6 (ECTS)
- Wednesdays, 11:00 – 12:45, S1.11
- Book: E.Angel *Interactive Computer Graphics* 3rd Ed., Addison Wesley, 2003
  get it from STORM or the VU Boekhandel
- Grading: written exam (1/3) plus programming assignments (2/3)
  both parts must be graded "sufficient"

Programming Assignments

- Organized by: *Tom van der Schaaf* and *Mathijs den Burger*
- Programming in C (or C++) with OpenGL
- Programming on Windows PCs:
  (rooms S3.29, S3.45, S3.53, P4.23, P4.29, P4.37, P4.47)
- Details via the WWW page: **http://www.cs.vu.nl/~graphics/**

The assignments:

- Exercise 1: “the basics”
- Exercise 2: “solar system”
- Final project:
  - ⋆ Dino or
  - ⋆ Pony

How to get Credits for the Assignments

1. Register (now and/or next week)
2. Submit first exercise until October 15
3. Submit second exercise until November 26
4. Submit **one** of the two projects until January 7, 2004

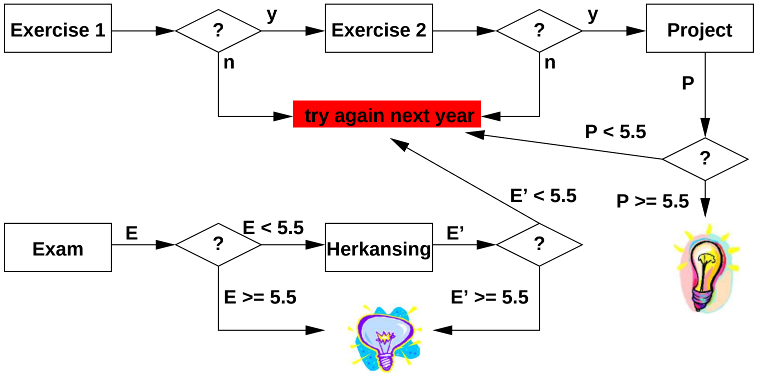
These deadlines are strict!

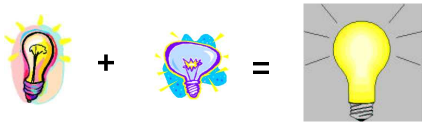
Submission: email your programs to **graphics@cs.vu.nl**

Exam (theory part)

- Registration via the TIS system (mandatory)
- Date for exams
  - ⋆ first: 22 December 2004, 9:30–12:30
  - ⋆ "second chance": TBD
- Written exam, 3 hours **(closed book)**

Overall Grading of the Course





G = 2/3 * P + 1/3 * E

And now for “Computer Graphics”. . .

- Where do we find computer graphics?
  - ⋆ display of information
  - ⋆ design
  - ⋆ simulation
  - ⋆ user interfaces

A Scene from Toy Story 2

How they do it. . .

http://www.pixar.com

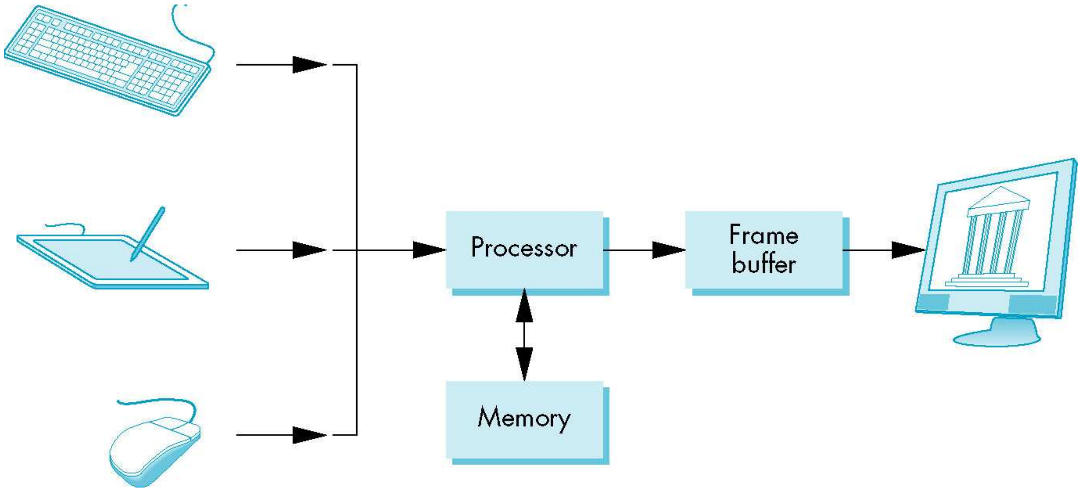
How we do it. . . (Course Outline)

1. Introduction (today)
2. Graphics Programming (basic OpenGL)
3. Excursion to the CAVE
4. Input and Interaction
5. Geometric Objects and Transformations ($2\times$)
6. Viewing (3D and perspectives)
7. Shading (light and matter)
8. Object Hierarchies (scene graphs) ($2\times$)
9. Discrete Techniques (texture etc.)
10. Implementation of a Renderer
11. Curves and Surfaces

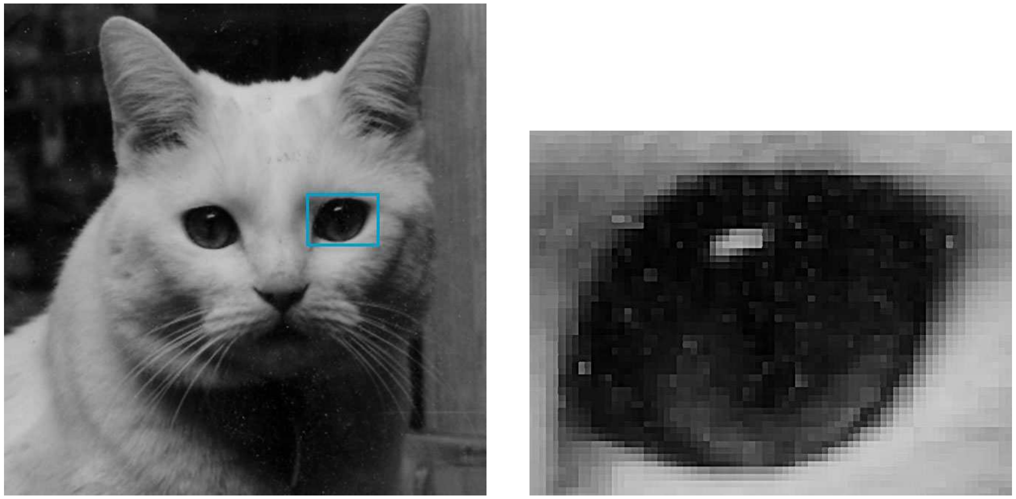
Outline for today

- Graphics systems architectures
- High-end graphics systems
- Making images: objects and viewers
- The human visual system
- Image formation: the pinhole camera
- The synthetic camera model
- Application programmer's interface

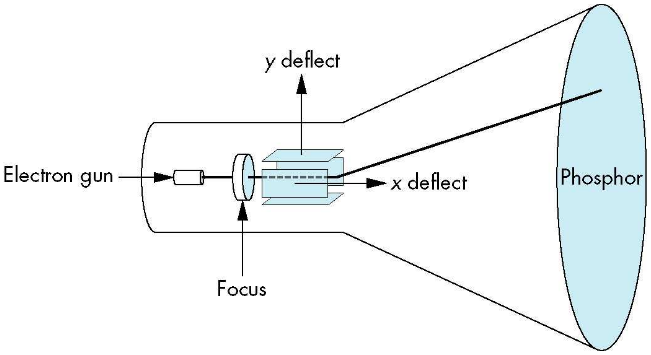
Graphics Systems Architectures



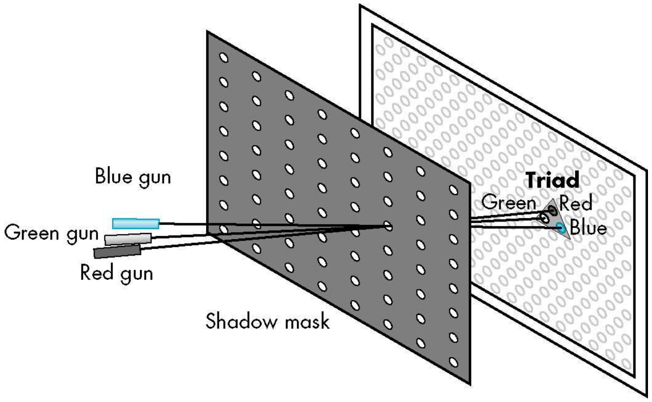
Raster Graphics



The Cathod Ray Tube (CRT)

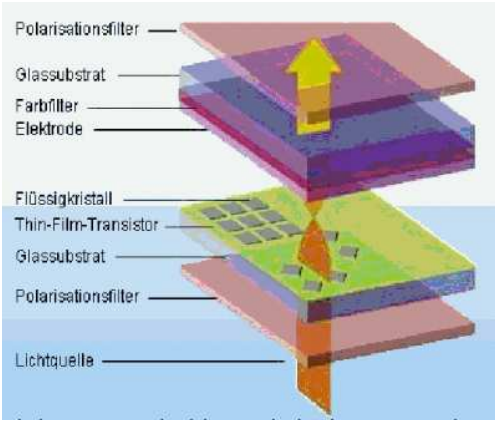


Shadow-mask CRT

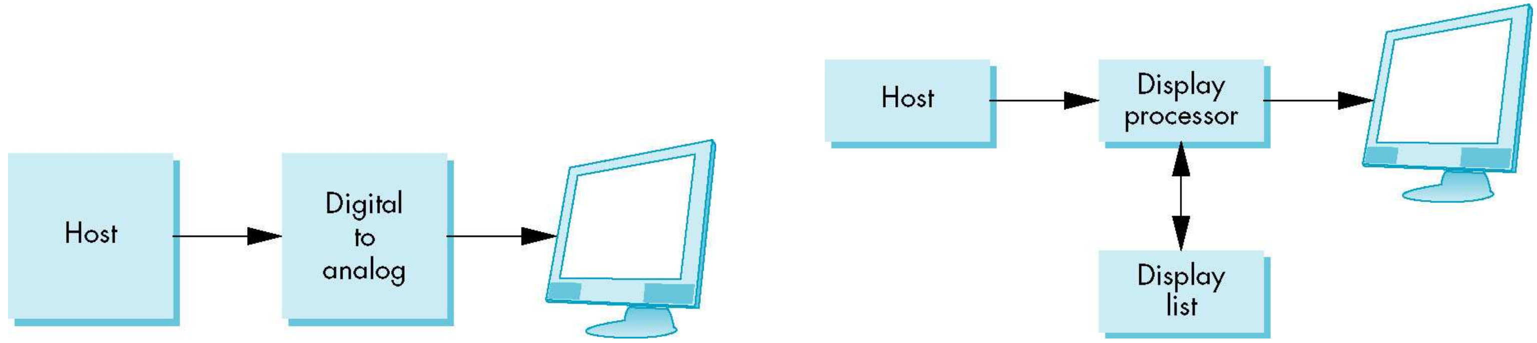


The Liquid Crystal Display (LCD)

- uses matrix of horizontal and vertical electrical wires
- voltage at the intersection point lights up pixel
- passive matrix:
  only electrical wires
- active matrix (TFT):
  transistors at intersection points



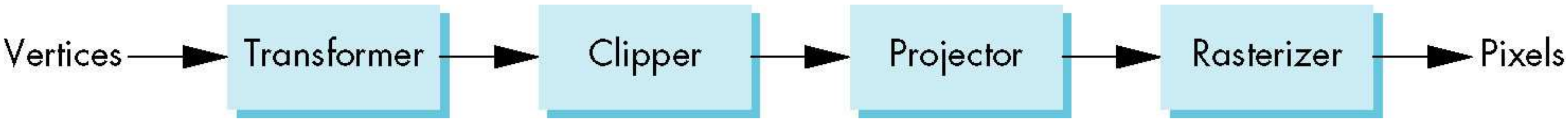
Graphics Architectures



early

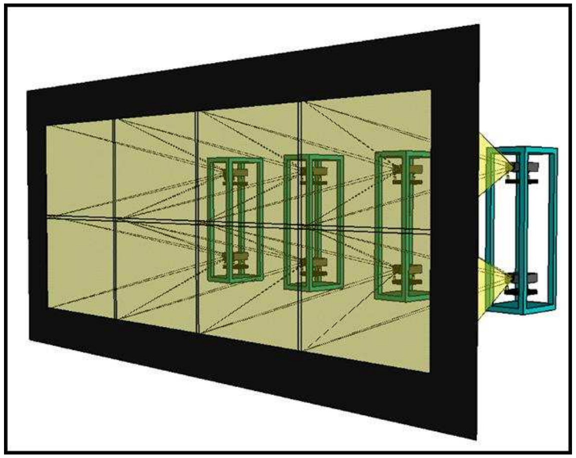
recent

Pipeline Architecture

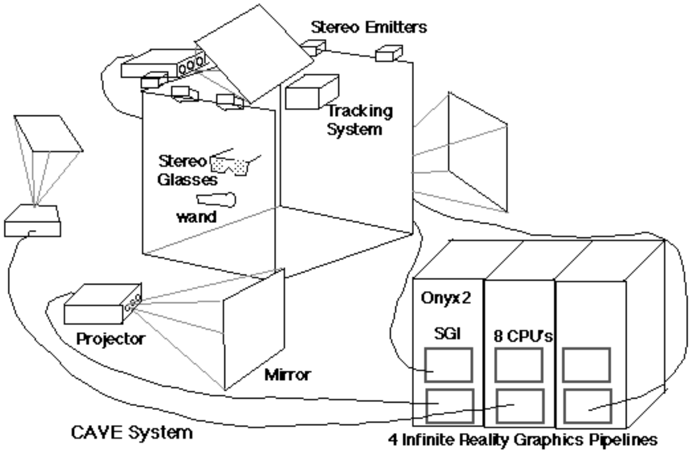


The most important architecture we are dealing with.

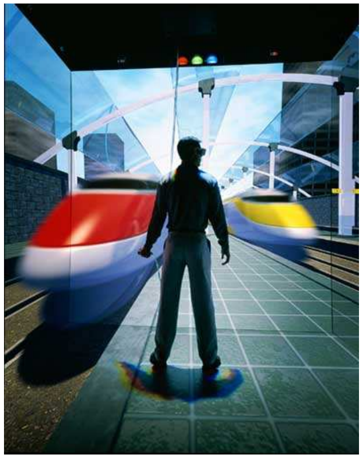
High-End Graphics: Tiled Video Wall (ICWall)



The CAVE Automatic Virtual Environment



Inside a CAVE

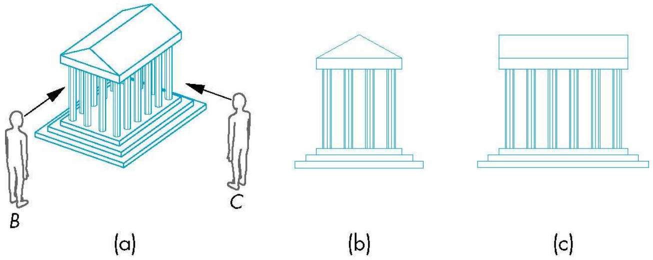


Excursion to the CAVE

- Demonstration of High-end Graphics
- Excursion to the CAVE, A'dam Watergraafsmeer (get there on your own)
- Date: september 22 (instead of the lecture)
- Signing up: now!

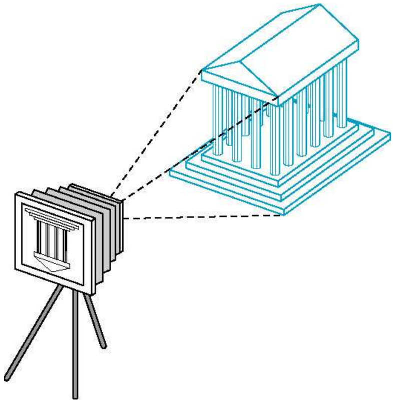
Making Images: Objects and Viewers

An image seen by three different viewers:

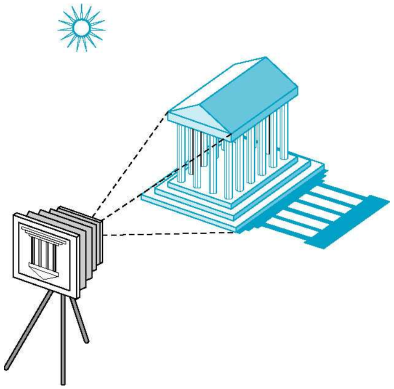


Goal in computer graphics (here): View synthetic objects like physical objects!

A Camera System



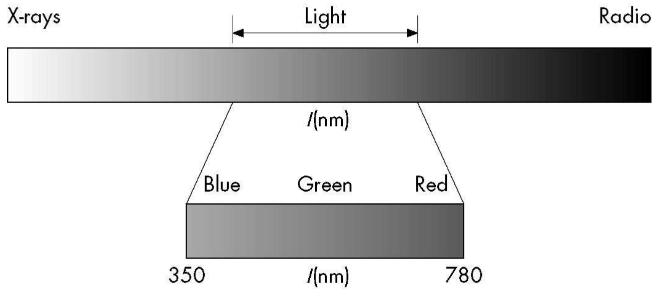
the camera is the viewer



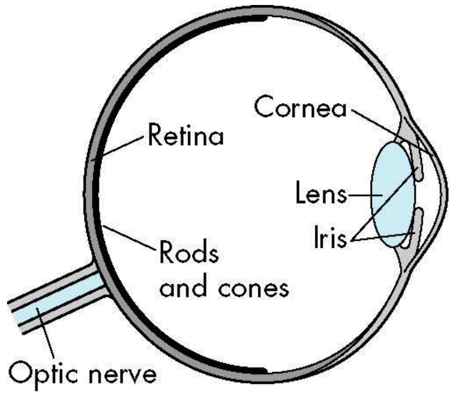
with a light source

BTW: What is Light?

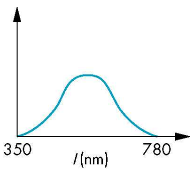
The Electromagnetic Spectrum:

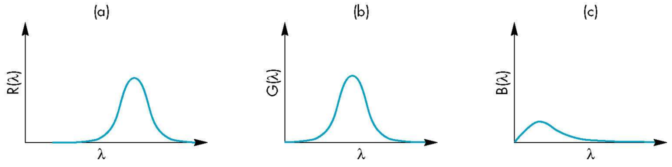


The Human Visual System

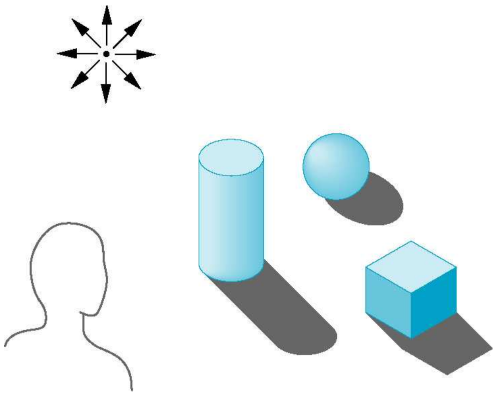


Human (Cone) Observer Curves

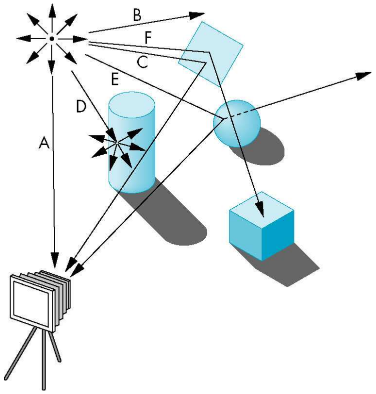




Scene with a Single Point Source

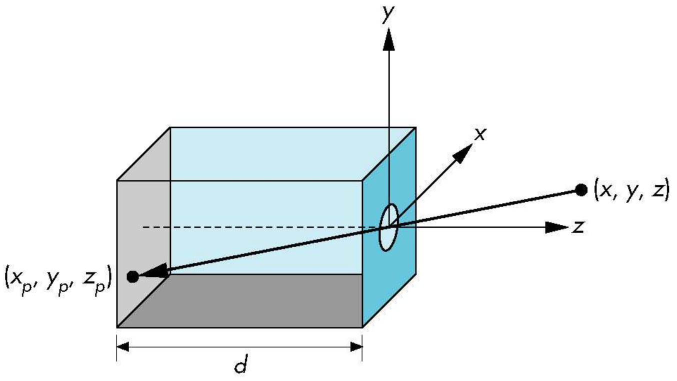


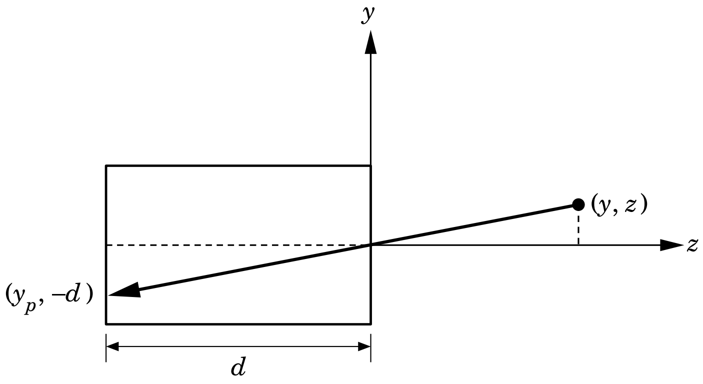
Ray Tracing



- Ray tracing can produce very realistic images (including shadows and reflections of objects on each other)
- However, ray tracing is **very** compute intensive (takes too long for interactive graphics)
- We will use simpler (faster) models, along with OpenGL

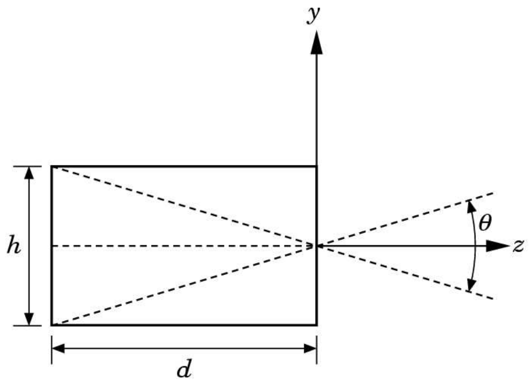
Image Formation: The Pinhole Camera





sideview

Pinhole Camera: Angle of View



The Synthetic Camera Model

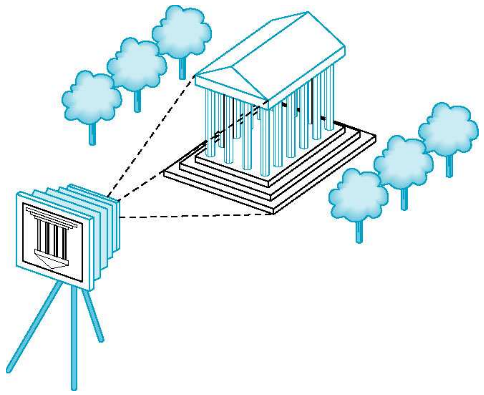
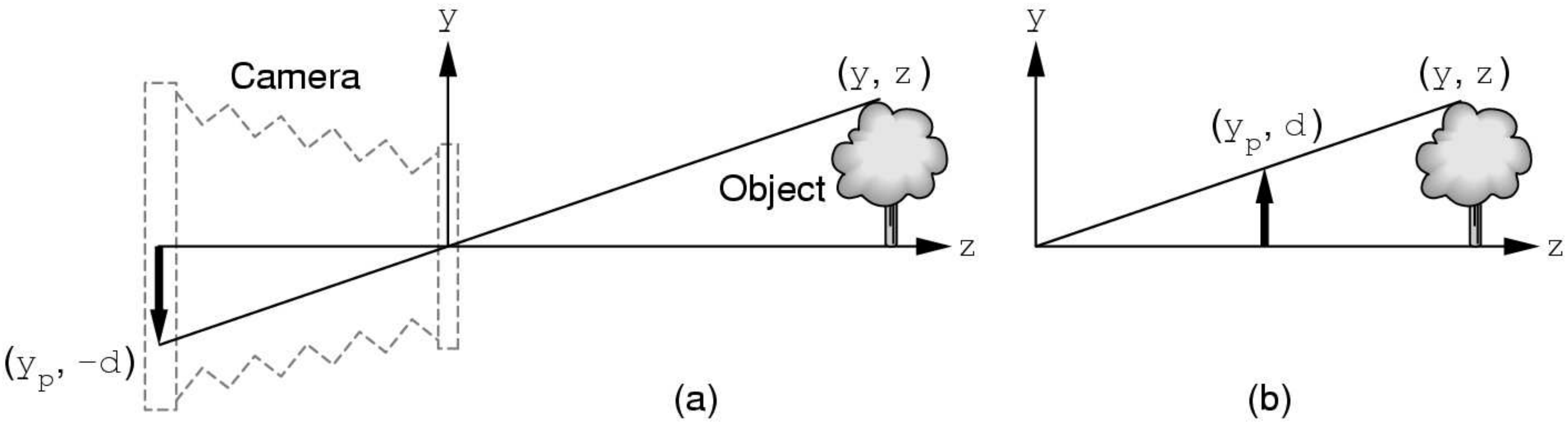
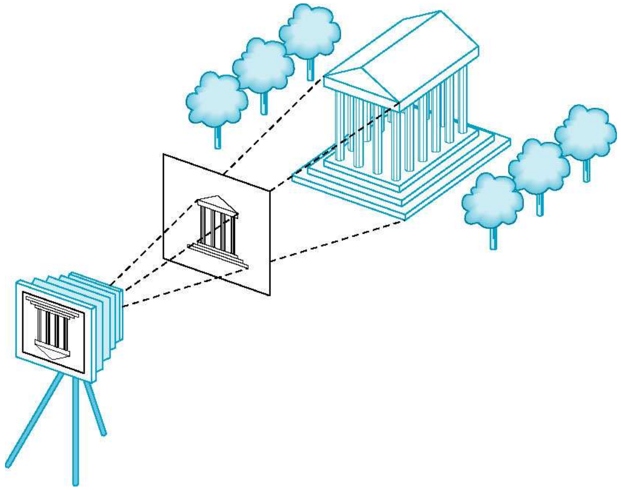


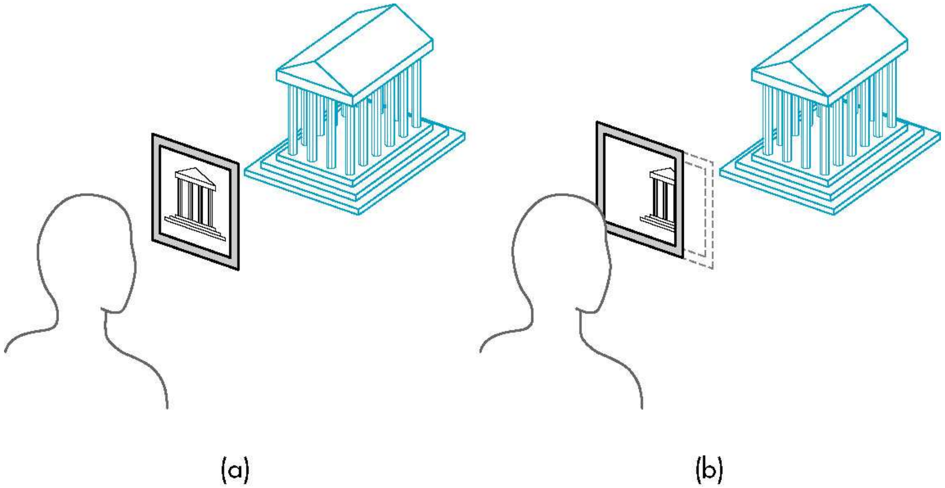
Image Formation with the Synthetic Camera



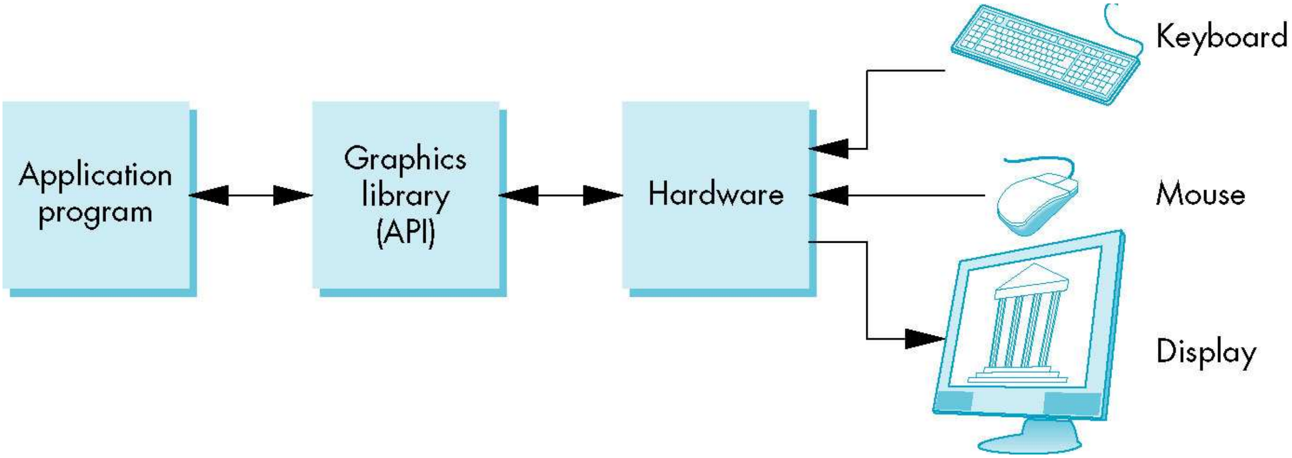
Imaging with the Synthetic Camera



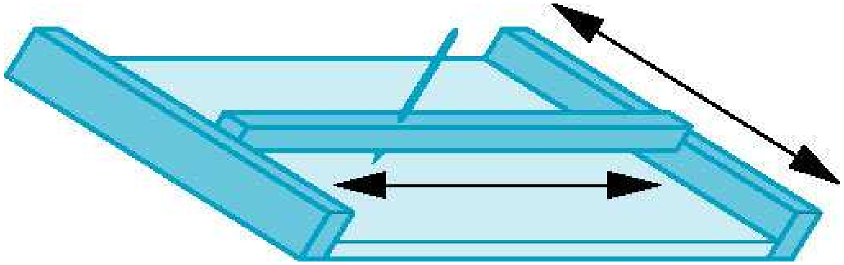
Clipping



Programming Interface (API)

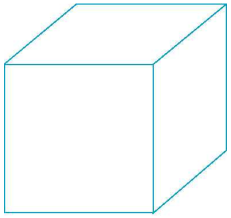


2D API: The Pen-Plotter Model



Functions:

```
moveto(x,y);
lineto(x,y);
```



(b)

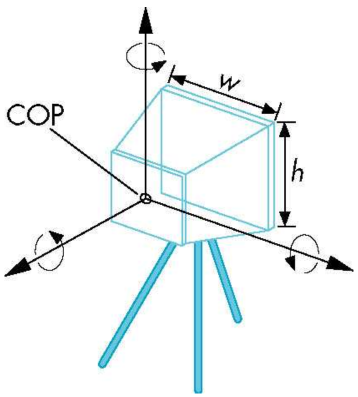
```
moveto(0,1);
lineto(0.5, 1.866);
lineto(1.5, 1.866);
lineto(1.5, 0.866);
lineto(1,0);
moveto(1,1);
lineto(1.5, 1.866);
...
```

Too low-level abstraction . . .

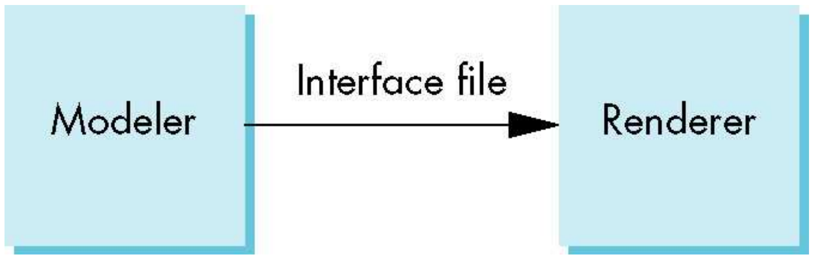
Three-Dimensional APIs

- Objects
- Viewer
- Light sources
- Material properties

API: Camera Specification



API: Modeling and Renderer



This course is (mostly) about rendering.

Summary

What to remember:

- Objects and viewers
- Synthetic camera model
- Raster images

Next lecture:

- Basic OpenGL programming

http://www.cs.vu.nl/˜graphics/